AI Magazine Volume 20 Number 1 (1999) (© AAAI)

Automated Deduction

Looking Ahead

Donald W. Loveland

■ In this article, the body of a report on automated deduction is presented that notes some significant achievements and takes a studied look at the future of the field.

Automated deduction is concerned with the mechanization of the deductive process in the fullest meaning of the concept. Mechanization of the deductive process includes not only proving new mathematical results by computer but also formally verifying the correctness of (certain properties of) computer chip designs and programs and even deducing the programs themselves from formal specifications of the task. A less obvious application is the use of automated inference tools within programming languages and within programs that produce scheduling algorithms and optimize other programs. Some of the early work in this field is already part of the fabric of the AI world. The machinery developed in this field is useful in inferring missing rules or facts in a problem specification (abduction) and generalizing from examples to full specifications (inductive inference, learning from examples), although I do not deal with these forms of reasoning in this article.

Some of the goals of the automated deduction field, such as fully mechanizing the proof capability of a mathematician, are still distant, but others, such as verifying computer chip designs, are already doable in part. This combination of profound, if distant, goals and important reachable goals makes automated deduction an exciting field. The purpose of this article is to focus on the opportunities that seem both important and promising and to note especially those that have promise for near-term payoff.

This article is the slightly modified body of a report to the National Science Foundation (NSF) Division of Computer and Computation Research, July 1997, entitled "Automated Deduction: Some Achievements and Future Directions." The major change is the deletion of a short section on human resources. (Also missing here is an extensive executive summary.) The report is the outcome of the Workshop on the Future Directions of Automated Deduction held in Chicago on 20–21 April 1996. To obtain broader input, especially from countries outside North America, a call for commentaries was issued to the automated deduction community. The commentaries, together with the full report, are available on the web at www.cs.duke.edu/AutoDedFD. It is important to note that the material presented in both the report and this article represents an American viewpoint; input to the report by non-Americans was limited to the commentaries. Furthermore, the material and viewpoints presented here are an extension of the workshop contributions, and I am solely responsible for the substance and manner of presentation of this article.

I begin with a summary of the major accomplishments of the automated deduction field. This summary is followed by consideration of the future directions of automated deduction, first with a brief consideration of near-term opportunities and then with a discussion of the longer-term issues and opportunities. The latter presentation is organized by application area, where for convenience the core area of mathematics is considered an application area.

Major Accomplishments

Automated deduction has been an active area of research since the 1950s. (For a historical and sociological view of the field, see MacKenzie

Settling the open problem, known as the Robbins algebra problem, is a milestone because it is by far the most difficult open problem to date whose proof was discovered by a computer (not just using a computer).

[1995]. For an overview of the first 25 years of the field, see Loveland [1984]; see Siekmann and Wrightson [1983] for a comprehensive collection of the "classic" papers of 1957–1970.) Significant work has been done both in the early years and lately. Early important work includes the Davis-Putnam procedure and the resolution proof calculus. For recent notable work, I focus on some significant theorem-proving systems: the geometry theorem provers of Chou (Chou, Gao, and Zhang 1994; Chou 1988); the Boyer and Moore (1988) interactive theorem prover NQTHM and its successor ACL2 (Kaufmann and Moore 1996); the rewrite rule laboratory (RRL) of Kapur and Zhang (1995); the resolution prover OTTER (McCune and Otter 1997; McCune 1994) and the equational logic prover EQP by McCune (1996); the interactive higher-order logic provers NUPRL (Constable et al. 1986), PVS (Owre, Rushby, and Shankar (1992), and HOL (Gordon and Melham 1993); and ANALYTICA (Clarke and Zhao 1993), which embeds a symbolic computation program, MATHEMATICA.[1] In the following subsections, I briefly discuss some of the successes realized with use of these and other procedures and systems. I mention here just one recent and significant success: the discovery by the theorem prover EQP of a proof of an important mathematical conjecture. Settling the open problem, known as the Robbins algebra problem, is a milestone because it is by far the most difficult open problem to date whose proof was discovered by a computer (not just using a computer). This problem had been attempted by many mathematicians, including the famous logician Tarski, who subsequently directed a number of young mathematicians to it as a challenge problem. (The accomplishment by EQP, and McCune, was featured by the *New York Times* in December 1996 [Kolata 1996].) In the remainder of this section, I review quickly some major achievements—both early and more recent—in automated deduction; such accomplishments lay the foundation for understanding the opportunities that I see ahead and the future directions I emphasize.

Early Achievements

I make no attempt to be historically complete but, rather, note events of particular pertinence to our present status. I do note the first automated theorem prover, the LOGIC THEORIST developed by Newell, Shaw, and Simon (1963) in the mid-1950s. The LOGIC THEORIST was significant for several innovations, including the first use of linked lists and its use of human problem-solving techniques to solve reasoning problems in logic. This approach to proving logic theorems drew a reaction from several logicians, such as Paul Gilmore, Hao Wang, and Martin Davis. They felt (correctly) that better results in proving theorems could be realized immediately by using tools of mathematical logic rather than being restrained to the imitation of humans.

One of the proof procedures that followed the LOGIC THEORIST was the Davis and Putnam (1960) procedure (Davis, Logemann, and Loveland 1962). This procedure is a propositional decision procedure when provided a fixed propositional formula but is also a first-order logic proof procedure whenever larger propositional formulas are formed in a standard way from the given first-order formula. Although the later resolution proof method formalized a generally better way of relating propositional formulas and first-order formulas, the propositional decision procedure is receiving as much attention today as when first presented. (There are actually two related procedures with the same general label; compare the two previously referenced papers or see, for example, Dechter and Pearl [1988].) The Davis-Putnam procedure has been used recently in the mathematics realm to solve some open problems in quasigroups (Slanet, Fujita, and Stickel 1995) and in the AI realm as a method to solve constraint-satisfaction problems (Dechter and Rish 1994; Dechter and Pearl 1988).

Another important program developed in the 1950s was the GEOMETRY THEOREM-PROVING MACHINE, a project headed by Gelernter at IBM (Gelernter 1963; Gelernter, Hanson, and Loveland 1963). This program used a geometry diagram as semantic information to prune the search tree sufficiently to prove many results that high school students prove on final exams. This use of problem-specific information, called the *semantics* of the problem, is one of the biggest challenges in automated deduction. The ability to prove interesting theorems in geometry and give readable proofs has been surpassed only recently by Chou, Gao, and Zhang (1994) in automated theorem provers using the area method that they perfected. I say more about the Chou, Gou, Zhang automated theorem prover later.

The resolution proof method introduced by J. A. Robinson (1965) was a significant advance over preceding proof procedures for its introduction of the unification algorithm that provided a provably complete method for working at the first-order level; the Herbrand terms did not have to be enumerated. The simplicity of the inference rules and the later discovery of how to retrieve "answers" from the substitu-

tions made in the proof process (Green 1969) led to its inclusion in many programs, from the ML programming language system (polymorphic-type inference) to natural language processors, question-answering systems, explanation-based reasoners, and deductive database systems. The inference engine of the logic programming language Prolog is a resolution system. The resolution calculus has been at the heart of several successful theorem provers developed at Argonne National Laboratory, of which the latest is the theorem prover OTTER developed by McCune (1994). OTTER is currently being used by mathematicians in research published in standard mathematics journals; has been used to solve several open mathematics problems; and has been used in ways only indirectly related to mathematical theorem proving, such as circuit design and verification.

An important contribution of the automated deduction community was introduced under the name *demodulation* by Wos and G. Robinson at Argonne National Laboratory (Wos et al. 1967). It is an equational inference rule later developed into the subdiscipline of *term rewriting,* pushed forward substantially by the paper of Knuth and Bendix (1970). Term rewriting is used in most of the automated deduction systems that I discuss in this article and is used in computer algebra systems and functional language implementations.

One success story for the application of automated deduction concepts and tools is the logic programming area. The most famous example is Prolog, which contains a resolution-based inference engine at its core (Robinson 1992; Bratko 1990). (Historically, the deduction mechanism of Prolog is a merger of resolution and a related procedure called MODEL ELIMINATION [Kowalski 1984; Loveland 1969].) Prolog is one of the two major AI languages, predominately in Europe (whereas Lisp is predominant in the United States). It was instrumental in the implementation of many of the early backchaining expert systems and is used extensively as a prototyping language for many process designs, among other uses. Two applications conferences are currently held each year, one in Europe and one in Japan. More recently, logic programming technology has been combined with constraint solving to create the important concept of constraint logic programming and the realization of immensely useful constraint logic programming systems. Other languages based on deduction include ML (Milner, Tofte, and Harper 1990) and LAMBDA-PROLOG (Nadathur and Miller 1988). ML is becoming an influential language for type inference, among other things, and LAMBDA-PROLOG is used to prototype theorem provers.

Early Automated Theorem Provers and Proof Techniques

Logic Theorist—1956—reflected human problem-solving techniques.

Geometry Theorem-Proving Machine—1959—proved many theorems given on high school geometry exams using a diagram for heuristic aid.

Davis-Putnam proof procedure—1960—was one of the most-used proof procedures for determining tautologies or refutable formulas in propositional logic.

Resolution proof procedure—1965—combined a perfected pattern-matching algorithm with a variant of modus ponens to give the most-used proof procedure to date.

Model-elimination proof procedure—1968—when augmented with resolution concepts, led to the inference engine used in the programming language Prolog.

Demodulation, paramodulation—1967, 1968—are inference rules for equality that replaced most equality axioms to permit direct reasoning with equality.

Knuth-Bendix completion procedure—1970—showed how equality rewriting (demodulation) could be structured to provide a decision procedure for some equality systems and a powerful proof procedure for other equality systems.

More Recent Achievements

The recent achievements I name are programs because the recent advances that are conceptual are usually embedded in systems. I discuss some of these achievements in more detail later. I consider the more recent achievements in the context of a problem area.

Program synthesis is the perfect real-world, economically important application for automated deduction; however, program synthesis, like most tasks requiring an automated intellect, is much further from reality than expect-

ed by optimists 40 years ago (cf. the AI world). Logic programming is, in theory, program synthesis but, in practice, has too heavy a procedural component and too limited an area of effective application at present to be the final solution. For program synthesis, the present secret to realizing applications is to greatly constrain the task that the automated deduction machinery has to do. This key idea of constraining search applies in many areas where automated deduction is applicable. There have been some notable successes, of which I note two of the best here. One is the KIDS system by D. Smith (1990) at Kestrel Institute, which provides algorithmic schema such as divide-and-conquer and other search algorithm schema within which a theorem prover operates. Success has been most dramatic in the scheduling algorithm area, where some derived scheduling algorithms have greatly outperformed the currently used algorithms. Another program synthesis success is the AMPHION project (Stickel et al. 1994), sponsored by the National Aeronautics and Space Administration (NASA) and developed by Lowry and colleagues. The program provides astronomical calculations for satellite guidance. A theorem prover is used to glue together appropriate routines from a subroutine library by matching specifications of each routine with specifications of the current demands.

Because programs will be written by humans for a long time to come, program verification is clearly a very obvious and attractive goal. Considerable funds and effort were invested in program verification in the 1970s without commercial success. This situation has led to a considerable scaling back of expectations and investment in program verification and a shift of attention to hardware verification. In hardware verification, the state of the art is approaching real-world systems, although the time required to accomplish the verification is not close to the needs of industry. An advantage of attention to hardware verification is that the verification of one process (a microchip) can safeguard a $100 million investment. (Such high-investment projects exist in software also, but such systems usually are so complex that formal verification for them is rare.) The ACL2 prover (Kaufmann and Moore 1996), the successor of NQTHM, recently was used interactively to obtain a formal proof of the correctness of the floating-point divide code for AMD's newest PENTIUM-like microprocessor. By *interactively*, I mean the interaction of machine and human where (in this case) the human provides some search guidance and supplies key ideas that the program cannot find on its own, such as difficult induction hypotheses and key lemmas. The verification was done at a detailed level and is an impressive analysis. ANALYTICA, developed at Carnegie Mellon University (CMU) by Clarke and Zhao (1993), has just recently produced fully automatically a proof of the correctness of a division circuit that implements the floating-point standard set forth by the Institute of Electrical and Electronics Engineers. The prover is built around the symbolic computation system MATHEMATICA. This exciting accomplishment does not void the work mentioned previously because the analysis is less detailed than the interactive verifications. The HYTECH system developed at Cornell University and the University of California at Berkeley is a reasoning system for analysis of hybrid systems, primarily a symbolic model checker. Earlier versions of the system were built on top of MATHEMATICA (Hensinger and Ho (1995).

Other specification-verification systems exist that handle commercial problems. HOL (Gordon and Melham 1993), an interactive prover developed at Cambridge University and now being used and upgraded in industry, has produced several large correctness proofs of real security systems that were delivered to customers. In another application, AT&T is using HOL to analyze and secure its large switching net. Lucent Technologies (Bell Laboratories) is using a combination of HOL and NUPRL to verify the SCI cache coherency protocol. The RRL system of Kapur and others has verified commercial-size adder and multiplier circuits (Kapur and Subramaniam 1996).

At present, the automated deduction technology most widely used in industry is in the realm of hardware verification. Such methods typically involve a representation for propositional formulas called binary decision diagrams (BDDs) (Moore 1994). BDD manipulation procedures can perform the same task, determining a tautology or its negation, as the Davis-Putnam procedure and propositional resolution procedures. However, they have been found particularly useful for circuit representation.

BDD packages (for creation and combining of BDDs) are used in *logic checkers* that check the equivalence of two logic circuit designs. BDD packages are often also used in model checking, where specifications, presented as propositional temporal logic formulas, are checked in the intended model, the circuit, presented as a state-transition diagram (Burche et al. 1992; Clarke, Emerson, and Sistla 1986).

Checking only one model, instead of all models, as theorem provers do, allows special-purpose algorithms to verify properties of moderately sized circuits fully automatically. IBM, Intel, and Motorola are among the companies that use model checking to verify specifications of microcontrollers. (*Microcontrollers* are integrated circuits, usually smaller than microprocessors, that make possible the "intelligent" appliances we have today.) Besides logic checkers and model checkers, other BDD-based techniques are used for formal verification in industry. Of particular note is the symbolic trajectory evaluation method developed by Bryant and others (Beatty and Bryant 1994).

As suggested previously, for software verification, progress is much slower; the objects being handled are more complex than hardware structures. However, recently there have been real-world applications of software verification (King and Arthan 1996). Multiple examples exist in the areas of networks, transaction systems, and safety-critical systems. The general area of concurrent systems is appropriate for formal verification with today's technology because the complexity overwhelms human capabilities but is relatively low compared with microprocessors. Sufficient success is seen in these areas that formal methods are receiving renewed attention by funding agencies. This attention to formal methods is much broader than automated deduction technology, but some of the other techniques, such as symbolic testing, do use automated deduction concepts and methodologies. The ability to tackle these real-world problems is primarily the result of the improved capabilities of interactive systems such as NQTHM, ACL2, PVS, NUPRL, HOL, COQ (Coquand and Huet [INRIA] 1988), and ISABELLE (Paulson [Cambridge] 1994). (These systems are also used for purposes other than verification.) An anecdote underscores the economic and service value of the validation of systems and specifications. A subsidiary of a German aircraft company was hired to produce software to control the switching of a complex broadcast network for the main German television company. The first submitted program failed to function properly, as did the second attempt. Under threat of a lawsuit by the television company, among other actions taken, the producing subsidiary turned to the German research center DFKI for help. The software tool VSE (verification support environment) developed in Germany, in part by DFKI, was used to respecify and verify an essential component of the system. In the process, it was discovered that the original specifications for the network software given to the aircraft subsidiary were in fact inconsistent! Plans for the court case included calling the theorem prover as a witness, but by then, the parties had settled out of court. The first opportunity for automated deduction software to get expert witness status was lost.

As for success in the mathematics realm, I already mentioned the settling of an important mathematical conjecture by a computer using general proof discovery techniques under consideration here. The field of mathematics is influenced in other ways by automated deduction systems and methods. Some mathematicians are using automated deduction programs as mathematical assistants to achieve new mathematics. Ken Kunen, a University of Wisconsin mathematician, has used OTTER to prove several results concerning quasigroups that have been published in the *Journal of Algebra,* a traditional mathematics journal (Kunen 1996a, 1996b). (A *quasigroup* is a binary system [G; ·] whose multiplication table is a Latin square.) Another paper on finding proofs of quasigroup problems interactively, by Fujita, Slaney, and Bennett (the last author a leading mathematician in this area), won a (shared) Publisher's Prize (best paper award) at the International Joint Conference on Artificial Intelligence in 1993, the leading international AI conference, as a recognition of the significant work done (Fujita, Slaney, and Bennett 1993). In this paper, a forward-reasoning model-generation system built at ICOT (Japan) was used. OTTER has been used to settle other open problems, such as finding minimal axiom sets (McCune 1993). Work is not limited to first-order logic and propositional logic provers. NUPRL (Constable et al.) helped confirm two conjectures, Higman's lemma (Constable and Murthy 1991) and Gerard's paradox (Murthy 1991; Constable and Murthy 1991), that were under active investigation by humans at the time. (For Higman's lemma, the humans won, but a subsequent machine-aided proof showed that the original design for the proof method, which the humans had abandoned, was indeed a feasible method of proof.) The remarkable geometry provers developed by Chou, Gao, and Zhang have been used to obtain new mathematical results in non-Euclidean geometry.

Mathematics and logic education are natural applications for automated deduction. Some success stories already exist, and there are new opportunities in the form of inference systems potentially useful in education. Of note was an early appli-

Three Areas Where Automated Deduction Has Strongly Impacted the Real World

Logic programming—The inference engine at the core of Prolog is a product of automated deduction research.

Program synthesis—The KIDS system has produced scheduling algorithms and programs superior to those previously in use.

System verification—Interactive proof systems are being used to prove that concurrent systems (for example, networks), safety-critical systems, and security systems meet their specifications.

cation of automated deduction in the completely computerized course in elementary logic at Stanford University designed by Patrick Suppes (1960s). He followed this course by one in axiomatic set theory (1970s) (Suppes and McDonald 1984). Both computerized course software embedded interactive theorem provers, with the student providing the major steps through a proof (usually with hints) and the theorem prover handling the details. The most sophisticated new systems are the geometry provers of Chou, Gao, and Zhang already mentioned. These systems can prove almost any theorem in rectilinear planar geometry (plus circles), Euclidean or non-Euclidean. Proofs for the standard theorems are usually short and elegant, with very readable proofs as output. Admittedly, the ability of the Chou, Gao, Zhang theorem prover to provide short readable proofs is not the same as having the provers embedded into a course structure, but the opportunity is there. These examples establish that there indeed have been many accomplishments in the field of automated deduction. With this review of examples as background, I now explore the most promising future directions for research in this field.

Future Directions

The preceding section on accomplishments should make clear that much has occurred in the field of automated deduction and that the activity level is high. However, much also remains to be done, and in this section, I address the question of what can be done in the near term and in the longer view. There should be no question of the worth of continuing this research. For example, both very large system integrated chips and networks are increasing in complexity, and the need to verify that these systems function as intended is growing proportionally, which is made vivid by the PENTIUM incident and reinforced by the examples in the preceding section. (The PENTIUM microprocessor had a faulty floating-point divide implementation that ultimately cost the manufacturer millions of dollars with the recall of hundreds of thousands of chips.) Thus, there is considerable economic incentive pushing the continued research in this area. Besides economic forces, social and intellectual forces also drive research directions and investment. Our social role is in education, mathematics primarily, but certainly parts of computer science and other subjects as well. There is also a clear intellectual role centered on mathematics: the solution of open problems and, eventually, the development of new mathematics. At the end of this section, I argue that the intellectual role actually is much larger than mathematics; in fact, it extends to all systematic inquiry.

Near-Term Opportunities

In any discussion of future opportunities and accomplishments, the inevitable question is when such accomplishments will appear. Estimating arrival times for innovations is, in general, a risky business to be avoided. My concession to this interest is limited to comments on possible near-term directions and achievements (roughly five to seven years ahead). In this time frame, I look to existing successes for the seed of new successes, including further use of OTTER (and maybe E-SETHEO, a competing first-order system new to the scene, whose reason for note I consider later) for general first-order reasoning applications. OTTER might have the widest distribution and broadest range of applications of automated deduction systems yet developed. However, many other systems now have user communities of their own, some sizable: NQTHM, ACL2, and RRL, for example, and the higher-order logic systems HOL, NUPRL, ISABELLE, PVS, COQ, and TPS. Some have already had sufficient impact on real-world problems that industry is adopting and aiding the development of the systems. For example, ICL of England is developing a commercial version of HOL. That so many provers have significant user communities is an indication that

the level of accomplishment in automated deduction has jumped noticeably in the past five years and that applications are expected to continue and show increasing sophistication.[2]

There is some interest in making automated deduction systems available to a wider community in the near future and even some consideration of commercial automated reasoning systems. To succeed, the system must have a targeted clientele to whom the system can be tailored. Tailoring involves selection of a theorem prover (most often a fully automated version will be appropriate) and design of a good user interface. The user interface includes choice of input and output languages and manner of providing search information with the problem input. User interface issues are discussed near the beginning of the section Beyond the Subareas. The best current example of a proof system ready for a wider audience is the geometry program(s) of Chou et al. It has a well-specified problem domain, the deduction power, the user friendliness, and multiple hardware versions. However, to be effective in the education realm, its primary application area, a software educational support package might be needed. Some type of application support package might be a requirement of all application systems using theorem provers; certainly, this general need for support packages seems true of today's automated deduction systems.

The verification community is now exhibiting real-world success. I noted earlier that microcontrollers are being verified in commercial applications, as are various concurrent systems and transaction systems. Safety-critical systems are being formally verified. The growing practicality of this technology is recognized by NASA in the convening of a workshop on formal methods where practicing engineers can meet with leading researchers (more on this later). There is no denying that automated deduction technology has increased in visibility and usefulness in the past few years. This pace will continue, and applications will come faster as the examples of success cause more application engineers to investigate the current opportunities. However, this increased success rate for applications does not mean that the field now breaks open its cornucopia. The major problem of handling induction, for example, is as tough as ever and will yield only to slow, determined work on many fronts. It has taken a long time by industrial standards (not relative to the development of the other mechanism for reasoning, by evolution) to get to the state we now enjoy, and progress from here might be as slow as fundamental work has traditionally been. However, now each accomplishment can have a noticeable incremental impact on applications because it can be added on top of systems already productive. For these reasons, I treat the specifics of future verification research in the subsection on longer-term opportunities but acknowledge here that any accomplishment that "shows up early" likely will be immediately welcomed in the applications world.

I must remark that in the United States the danger exists that because of funding problems, some of the systems named earlier will not continue to be developed. This discontinuation of development of deduction systems might be true even of verification systems that have the most immediate and focused economic reasons for development and have shown some strong recent success. At present, it seems likely that Europe will keep its funding level high. This situation might hurt the United States commercially in a decade or so, but for the intellectual outcome, it is nice that funding continues somewhere.

In the area of program synthesis, I consider two outstanding systems that will have near-term impact, both mentioned in the preceding section. The impact is quite different in the two cases. For AMPHION, the interest is in an exportable system design, whereas for the KIDS program, the continued development of the existing system, as well as the applications that will follow, is of most interest. Both systems are examples of highly constrained search, the secret to harnessing automated deduction in program synthesis with current technology. I somewhat arbitrarily focus on AMPHION here and postpone discussion of KIDS until the section on the longer-term view.

AMPHION derives programs from graphic specifications (input through a menu-driven regimen), which is translated into a theorem whose proof defines a program that makes extensive use of existing software from an in-system library. The theorem prover, an OTTER-like prover written at SRI International, works with an application domain theory that contains axioms encoding (in this application) basic properties of solar system astronomy and axioms giving specifications of the available subroutines. The proof is constrained to be constructive so that a program can be developed from the proof. The methodology is derived from the work of Manna and Waldinger (Loveland 1984). The system is (in effect) an extremely sophisticated expert system but uses inference capability well beyond that associated with the "expert system" label.

I recognize that the existence of a library of

There is some interest in making automated deduction systems available to a wider community in the near future and even some consideration of commercial automated reasoning systems. To succeed, the system must have a targeted clientele to whom the system can be tailored.

subroutines that interact to address a problem class is not common. With a few successes like AMPHION, task groups might be motivated to construct such subroutine libraries. This opportunity is, of course, in the longer term. A major concern now is how to identify other points of application for AMPHION-like projects. Specific industries do learn through meetings, trade papers, and the like that particular problems can be addressed with the new technology. These are major ways in which expert systems technology spread. For the past four years, NASA has held an annual workshop to promote such technology transfer for applications of mechanized formal methods (specification and verification) to safety-critical systems. The existence of such workshops underscores the real-world value of some automated deduction technologies. Perhaps a workshop in applied program synthesis could feature the AMPHION technology. Other methods of technology transfer are briefly discussed in the section on human resource use. The point is that AMPHION incorporates a powerful technology that is likely to have other applications at the present level of our capabilities in automated deduction. Exploring other possible applications will also cause the capabilities to be expanded, thereby providing a useful focus and helping to teach the subfield how to scale up to the more complex problems.

Another important development direction is the continued automation of verification procedures, almost all of which have strong dependence on highly skilled user input.

Addressing other possibilities, there might be near-term products that come from the wedding of computer algebra and automated deduction techniques. Likewise, meaningful application of automated deduction techniques to model-checking methods of verification might already be under way. In the next section, I discuss both subjects (and others) but suppress any temptation to guess timing beyond the conservative calls I have already made.

Longer-Term Opportunities

I now consider the longer-range opportunities and research directions of various areas of automated deduction. I begin with the verification area because of its application importance, the emergence of real application successes, and the number of projects in the research community for which verification is a major application.

Verification One of the most promising directions for development of verification systems is the integration of tools and techniques that have developed in diverse domains. We are beginning to see theorem provers that integrate inductive proof techniques, general rewrite procedures, model checking, propositional provers (such as the BDD method considered earlier), linear arithmetic, other decision procedures, tactic facilities, execution capabilities, enhanced static checking, lemma generation, and computer algebra systems. These are not yet all in one system, and it is not clear how to optimize joint use, but the benefits of success are clear. Some systems that are partially successful in combining techniques are PVS, NQTHM, ACL2, HOL, EVES (Craigen et al. 1992, 1991), and RRL. PVS is particularly aggressive about integrating many of these tools, with quite impressive results. The linking of systems is also occurring. The verification of the SCI cache coherency protocol mentioned earlier uses a linking of HOL and NUPRL, which allows many theorems from HOL to be imported into NUPRL. The linking is a nontrivial task that took several years of theoretical groundwork (Howe 1996) because of the different formulations of higher-order logic each system uses. The gain from all this theoretical effort is the ability to use the deductive aspects of each where each is strong.

Another important development direction is the continued automation of verification procedures, almost all of which have strong dependence on highly skilled user input. This automation task is very difficult in general, of course, and research has been steadily pushing on the boundaries of this problem. Progress is being made at the edges of the problem, such as type checking and array-bound checking. (SIMPLIFY is a system that uses simple decision procedures to find bugs in MODULA-3 programs [Detlefs 1996]. Although bug finding is not verification, it is another illustration that some program analysis tasks can be automated now by automated deduction–related techniques.)

Big gains in the automation of verification systems are likely from the combination of computer algebra and automated deduction systems, although problems exist. The possible gains are illustrated by ANALYTICA. ANALYTICA runs in fully automated mode. Although it does not handle the complexity of proof arguments of a good interactive system, it surpasses almost all fully automated verifiers not using computer algebra. However, there are problems of soundness (correctness) in all the major commercial computer algebra systems—MATHEMATICA, MAPLE, MACSYMA—which is disturbing. Automated deduction research can help here, as I discuss later.

Increased automation of induction proofs would also realize big gains for verification systems. Progress is being made, but this problem is a long-range one; I discuss it briefly under the paragraph on mathematics. Much of the progress to date has been made by researchers

addressing the verification problem specifically. There are advantages to addressing the automatic generation of induction hypotheses (one of the most creative tasks in deductive mathematics) within the verification setting because verification tasks have certain forms that allows specialization of the induction hypothesis methods. Also, verification proofs are shallower (but messier) than general mathematical proofs. These characteristics are reasons why research specifically in verification must be pursued and not resisted with the expectation that general automated deduction research will provide the solution. Rather, the overall message of 40 years of research in automated deduction is that when an important special task can be identified, research should focus on this special task.

Automated deduction technology might be able to enhance the capability of the model-checking method. Indeed, has already been accomplished in one manner in the PVS system by viewing model checking as a decision procedure for a finite (fixed-point) subcalculus of the PVS higher-order logic. The application of induction techniques could allow the checking of parameterized systems by generalized model checking. An important application would be the use of a theorem prover to verify the abstraction mapping of a large system to a smaller finite-state machine. Then model checking could be used to check the abstract machine. As an extension of this idea, systems might be decomposed for processing by model checkers with the composition of the components verified by a theorem prover. It is likely that the theorem-proving part would be done interactively for some time into the future, whereas the checking of the abstraction could be done fully automatically by model checking.

Program Synthesis To many people, program synthesis is the most exciting application area for automated deduction in economic terms. The idea of presenting human-understandable task specifications to a computer system and receiving in return a program that is known to meet the specifications submitted is hard to fully appreciate in terms of both the intellectual accomplishment represented and the economic impact such capability would have. Once achieved—and full realization of this dream is surely well in the future—all investment in automated deduction will be repaid many times over. I note that this task is not isolated from other automated deduction tasks because work such as verification research moves forward the technology that program synthesis will (and does) exploit.

The output of a synthesis machine is no better than the quality of the specifications that are entered. The earlier anecdote about the German network software makes clear the problem of assuring the correctness of specifications. Because developing specifications is formalizing the informal, there can be no formal proof of correctness issue here. One aspect to address is the design issue; human error will be minimized if the specification language is tailored to the task domain, particularly at the level of concept representation ("granularity"). The other major issue to address is tests of representation quality. Consistency of the specifications is one concern, which clearly uses automated deduction technology. Another concern is semantic checks, meaning items from type checking to operational semantics for some constructs of the specification language. This research is under way, as examples cited earlier illustrate.

I address the future prospects of program synthesis in general by first commenting further on the KIDS project of D. Smith at Kestrel Institute. The KIDS program, together with the associated research effort, is a sophisticated example of the constrained program synthesis concept. Inference is used in a major way but under the control of design tactics (plans) that are custom built to the algorithm and problem classes within which the problem of interest falls. Problems are formally specified by giving information that includes an input domain and a notion of what constitutes a problem solution. This problem theory is fitted to an algorithm class. (Actually an algorithm theory extends the problem theory in the sense of adding appropriate logical structure, including axioms about the algorithm class, which corresponds to constructing an interpretation between theories.) For example, there is the problem-reduction algorithm theory that includes divide-and-conquer and dynamic programming algorithm theories in hierarchical fashion. The divide-and-conquer algorithm theory, when coupled with a problem specification for sorting integers, provides the structure to yield a quick-sort algorithm. For an actual program, the coupled algorithm theory and problem specification theory is all fit to a program theory that introduces the control and programming language chosen. The design tactics that control the inferencing are carefully tailored to the theories as they are developed. For example, the divide-and-conquer tactic will effectively produce a specific divide-and-conquer algorithm that is a model of the general divide-and-conquer theory. The design tactics have a constructive aspect that first allows a user to select standard compo-

Two Ways Automated Deduction Has Impacted Mathematics

1. Solving open mathematics problems (selected set)
- Robbins Algebra problem (1996)
- Quasigroups (1993–1996)
- The finding of minimal axiom sets (1993)
- Ternary Boolean algebras (1978)

2. Aiding the study of Euclidean and non-Euclidean geometry
- The Chou-Gao-Zhang theorem provers
  - Capable of proving hundreds of the theorems of classical geometry
  - Now with user interface for educational or research use by others

nents for parts of an algorithm and then has the inference system deductively solve for the remaining components by using the axioms of the algorithm theory.

I will not comment extensively on the other research in program synthesis and the opportunities there because many of the critical technologies needed are those of other domains of automated deduction and are discussed under other areas. The lack of extensive comment is not to slight the need for other high-quality work within the domain of program synthesis. I already mentioned the work of Manna and Waldinger as influencing the design of AMPHION. Good work on program synthesis has been done within the logic programming community (Deville 1990). I mention a problem area not yet named that is germane here, proof by analogy. This problem is usually associated with mathematical reasoning, but some researchers in software engineering, aware of the usefulness of automated deduction, see a role for proof by analogy. The idea is to use interactive automated deduction tools in program development in such a way that the specifications and program are constructed, or at least finalized, together. Thus, one develops prototype programs and uses these to converge to the final product and its specifications. The proof by analogy is appropriate in the updating of correctness proofs that show that the implementation meets the specifications. Although efforts have been applied to the problem of proof by analogy, this problem area appears to be one of the harder ones in this field, and substantial progress seems some distance away at present.

Mathematics The automated deduction field has been concerned with finding mathematical proofs for most of its 40 years of existence. (Concept formation, the heart of mathematics to mathematicians, is far more difficult to automate than proof discovery. Only small beginnings have been made on the concept formation challenge.) Although the Robbins algebra proof discovery stands out at present, I noted earlier that several provers have been used to solve open mathematical problems. Their logical power ranges from propositional logic to higher-order logic. Some researchers in the automated deduction field, at least in the United States, think that automated deduction research support for mathematics now should go primarily to solving open mathematics problems. The merit is that a focus then is set for funding at a time when little funding seems likely in automated mathematics in the United States. Some other automated deduction researchers strongly oppose this view. They point out that open problems have been solved in certain types of mathematics only: discrete mathematics, logic and algebraic systems primarily. In particular, the area of continuous mathematics has had no new theorem proved by fully automated means. Funding only systems that have solved some open problem might halt progress in addressing continuous mathematics, for example.

A way of assessing quality without the mandate to solve open problems is by performance comparison among provers having the same goals. One means for doing this performance comparison among provers began in 1996 as a competition among first-order logic provers. The first TPTP competition, named for the THOUSANDS OF PROBLEMS FOR THEOREM PROVERS database containing at least 3000 problems with full axiom sets, was held at the 1996 Conference on Automated Deduction and enjoyed admirable planning, publicity (within the field), and contest controls (Sutcliffe and Suttner 1997a, 1997b). Such competitions take great resources to assemble and run; much credit goes to Geoff Sutcliffe (James Cook University, Australia) and Christian Suttner (TUM, Munich, Germany). It is important that this competition thrive and be recognized for the value it adds to the development of the field. It needs to broaden to include different types of prover, such as higher-order logic provers and induction provers, although efforts might already be under way to extend

the competition.

The value of such a competition is shown by the surprise win of E-SETHEO, a connection tableaux and model-elimination–based theorem prover from Munich (Moser et al. 1997), in one major category of the 1996 competition. At the finalizing of this article (over a year after the original report appeared), two further TPTP competitions had been held, and the results continue to surprise. The 1997 competition had as the top three finishers in the primary category the systems GANDALF, SPASS, and OTTER, in that order (Suttner and Sutcliffe 1998). GANDALF is a theorem prover that time-slices many strategies of the type incorporated in OTTER. SPASS is based on a superposition calculus (similar to paramodulation) and incorporates a splitting rule for case analysis. (See Suttner and Sutcliffe [1998] for brief overviews of the systems.) In 1998, the same category had as top finishers the systems GANDALF, BLIKSEM, SPASS, and AI-SETHEO in that order, with the top three very close together. BLIKSEM appeared in the competition for the first time yet took second place. BLIKSEM is a highly tuned coding of procedures somewhat similar to those of OTTER. Major factors in the strong performances of GANDALF and BLIKSEM are engineering and implementation features rather than distinctive proof procedures. The value of competitions, like solving hard problems, is that significant engineering and implementation techniques can show their worth. To summarize, competitions can improve the breed in a manner similar to solving open problems and is applicable in domains where open problems are presently beyond reach.

Of the top performers named earlier, all but OTTER are European in origin, with two provers (SETHEO and SPASS) from Germany. This systems capability, and a strong conceptual capability as well—measured by accepted papers at the Conference on Automated Deduction—is not a surprise to members of the research field. The European community, with Germany at the fore, has become the dominant force in the automated deduction field. A partial explanation is that the German deduction community won a six-year award of roughly US$15 million in 1992 for basic research directed toward applications in programming (the Schwerpunkt Deduktion Project). This award allowed expansion of an already strong research base. The recent emergence of applications for automated deduction systems and formal methods suggests their timing for increased support of automated deduction might be excellent.

The problem of proving theorems in the continuous mathematics domain was mentioned earlier, and I comment briefly here on this area. Many important theorems concerning continuous functions, for example, can be proven as logical consequences of other theorems. Definitions such as continuity for functions have reasonable logical encodings. However, in this domain, most proofs, even for theorems of modest difficulty, demand substantial reasoning capabilities. Particularly needed are enhanced automated reasoning capabilities in set theory or in the closely related type theories of higher-order logics. Good first steps have been made in first-order logic (Bledsoe and Feng 1993) and higher-order logic (Andrews et al. 1996; Paulson et al. 1996; Paulson 1993). (The Paulson papers on higher-order logic systems consider the interactive system ISABELLE, but Andrews et al. (1996) discuss the system TPS directed toward automatic proofs.) As good as this work is, the accomplishments do not come close to providing the reasoning capabilities needed. In particular, the combinatorics of possible instantiations of variables to values explode when dealing with sets or higher-order formulas (which can be values for higher-order variables). In another direction, reasoning that is routine to humans regarding the real numbers is beyond our reach at present, with a few striking exceptions. One notable example was work done by Bledsoe and Hines (1980) in the development of a variable elimination method for inequality formulas and a manner of chaining together inequalities, which they incorporated in the system STR_+VE (Hines 1994). The resultant theorem prover works in the first-order logic setting, using resolution, but replaces resolution proofs handling inequalities with a system as capable in most respects as a human in simplifying systems of inequalities. Much more work of this type is needed and should be supported, but such results are difficult to develop. It seems clear that interactive and semiautomatic automated deduction systems, most likely using higher-order logic formulations, will provide useful mathematical assistance, but new approaches or even major breakthroughs are needed before fully automated deduction systems can tackle difficult or open problems in continuous mathematics.

One single step that will strengthen current theorem provers considerably, whether interactive or fully automated, is the incorporation of computer algebra systems.

One single step that will strengthen current theorem provers considerably, whether interactive or fully automated, is the incorporation of computer algebra systems. This incorporation of computer algebra systems does not benefit provers uniformly, of course, but only those for which algebraic manipulation is part of the proof task. Still, within the pertinent domains, the effect will be dramatic. Clarke

and Zhao demonstrated the power of an enhanced algebraic simplifier coupled with automated deduction proving techniques by proving fully automatically a number of summation equations of Ramanujan (Clarke and Zhao 1996). The theorem provers were ANALYTICA and a successor prover. The Ramanujan results are beyond the capabilities of other present provers. (Other interesting work, involving series summation using a proof planning method and "rippling," is considered later.)

The combination of computer algebra and theorem-proving technologies also can be used to overcome present shortcomings of the computer algebra systems. These algebraic manipulators are unsound by such faults as dividing by zero without acknowledging the fact or not checking termination conditions. (It is not as simple as checking for a zero, of course; some systems execute symbolic computations, and then the condition must be retained explicitly.)

The automation of mathematical induction remains a central problem, as noted several times. In particular, we need to better handle the choosing of induction hypotheses and induction variable. Specialized cases continue to be identified and custom handled, and induction systems continue to improve in performance, but we lack general techniques. We cannot expect global, powerful techniques that match in flavor the universal enumeration proof procedures that exist for first-order logic. The best we can hope for is techniques that handle certain problem classes. One such promising insight, called rippling, originated in work by Bundy and colleagues (Bundy et al. 1993) at Edinburgh University. It is a syntactic strategy that takes advantage of similarities between the induction step consequent and the induction hypothesis and executes a monotonic series of local manipulation steps to change one to the other. It has shown surprising robustness, even finding use in proof planning (Walsh, Nunes, and Bundy 1992). However, too few paradigms such as rippling are known at present. Support of research on induction is important if broader principles are to be discovered and developed. Like the exploitation of computer algebra and automated deduction as discussed previously, the improvement of induction automation will have strong effects on verification work as well as general mathematics.

Part of the environment of proof discovery is *counterexample discovery,* the finding of an example that refutes a conjecture. This counterexample discovery is receiving attention with model-generation schemes and procedures. Counterexample discovery is not new to automated deduction technology, having been undertaken at Argonne National Laboratory with the predecessor of OTTER (for example, Winker and Wos [1978]). Recently, this task has received more attention, from use of the Davis-Putnam procedure for satisfiability checks to model-generation use in the quasigroup work (Slanet, Fujita, and Stickel 1995; Fujita, Slaney, and Bennett 1993). Efforts are needed to better incorporate counterexample production with proof search, where the counterexample is used to truncate a false probe in the proof search.

An apparent opportunity that we have never been able to exploit to the degree imagined in the early days of automated deduction research is the use of semantics to guide the proof search. It was used elegantly in the geometry theorem-proving machine, where the geometry diagram was used to eliminate many irrelevant search candidates (Gelernter, Hanson, and Loveland 1963). It quickly became clear that there were few problem domains that had a problem-representation language at the same granularity as the proof language. The use of problem semantics on a broader scale now seems to occur primarily in the use of decision procedures for various subdomains. Decision procedures are being incorporated in provers, especially in verification provers. We should continue to address the production of decision procedures on whatever problem domains we deem of some interest. They will not be attention getters, such as Presburger arithmetic or Tarski's real closed-fields procedure, but simply utilitarian for a job on hand. They then should be shared with other developers in the sense of a common web site, where such procedures and the domain they encompass can be announced. Like heuristics for induction-variable choice, on such mundane items, much progress is gradually made.

The Robbins algebra accomplishment reinforces the value of continued research in equational reasoning. The system EQP uses *associative-commutative unification,* a fancy way of pattern matching where order and grouping of term occurrences do not matter (Stickel 1975). That order and grouping of terms do not matter is a property of equality reasoning in particular. In the presence of variables, it is a difficult process to control. Another feature used by EQP is a sophisticated version of paramodulation, the equational inference rule mentioned earlier. Like associative-commutative unification, the paramodulation rule is explosive in the possibilities it generates. Extensive research, much of it arising first in the term-rewriting

community, has led to the sharp trimming of possibilities that need be considered. Without the prior research that produced associative-commutative unification and controlled the combinatorial explosion of equational inference in general, the Robbins algebra problem would not have been solved. Although the term-rewriting community is at the center of the equational reasoning efforts, and support there is important, work outside this community also takes place, as the original paramodulation work makes clear.

Because so much emphasis has been placed on directed research in this article, I belabor the point that basic, often purely theoretical, work must be supported for the field to move forward. In addition, we must not be too eager to leave the "old" behind, feeling that old areas have been explored, and only new ideas from new areas will contribute to progress. Strong evidence of these points comes in the previous paragraph. The paramodulation inference rule was introduced in the 1960s, yet it is the fairly recent work on basic paramodulation (Bachmair et al. 1992) that was incorporated in EQP. Also, the associative-commutative unification work was done in the early 1970s but not used to this point as much as it now surely will be.

I mentioned that the term-rewriting community is central to the work in equational logics. This point warrants elaboration. Rewriting-based automated deduction systems often work with equational logics using oriented equations, or rewrite rules. Although the use of rewrite rules derived from equations is actually older than electronic computers, the rewrite theory and consequent practice were initiated by a paper by Knuth and Bendix (1970). This paper contains a powerful result: Given a set of equations (an equational logic) and appropriate added information, there exists a unique (under certain conditions) set of rules that define a decision procedure for the theory presented by the equations. The "completion" of the initial rule set to the extended decision procedure rule set often is troublesome in practice, but even when completion is impractical, the rules that are generated are powerful additions to the original set. Much significant work, such as the basic paramodulation results mentioned earlier, has followed over the years from this beginning. Many automated deduction systems have been developed around these ideas directly; the RRL system mentioned earlier is one example. Such automated deduction systems rely extensively on rewriting as a deduction mechanism and also support sophisticated unification methods (for example, associative-commutative unification) for completion (McCune's EQP is one such system). Approaches for reasoning by induction and semantics of data structures using decision procedures have also been integrated, leading to powerful automated deduction systems that can be also used for applications, including analysis of equational specifications and implementations. As an example, the latest version of RRL has been used by Kapur and M. Subramaniam to verify properties of the SRT division circuit, as specified by Clarke et al. This work was done completely automatically and without a computer algebra system, as was used by ANALYTICA, Clarke and Zhao's system (private communication from D. Kapur, March 1997). Many of the term-rewriting systems are being developed in Europe and are being used for tasks such as circuit verification as well as for mathematics. See Ganzinger (1996) for sample system descriptions as well as recent papers on rewriting techniques. This line of research is important and definitely needs sustained support.

Mathematics Education I have addressed the economic and intellectual aspects of our field; now, I address the social aspect of automated deduction technology. This aspect centers on education and the tools we can offer to enhance mathematics and logic education. There is much need, at least in the United States, for improved performance in mathematics skills and logical thinking at the high school level, and support should continue into the college years. Earlier, I mentioned past automated deduction contributions to education; also I noted several times the Chou-Gao-Zhang geometry programs. Although the programs are impressively polished as theorem provers, with proof output in human-readable form, a tutorial environment needs to enclose the programs to provide the setting in which students can use the tools effectively. For logic education, at the base level, there are now Tarski's WORLD (Barwise and Etchemendy 1992) and the follow-up HYPERPROOFS (Barwise and Etchemendy 1994). At an advanced level, a portion of the interactive theorem prover TPS (Andrews et al. 1996) has been used at CMU and distributed to interested parties and has been highly rated by educators. This system, called ETPS, consists primarily of the interactive facilities for natural deduction reasoning from TPS. For reasoning at a somewhat higher level than pure axiomatic systems, where computations can be included as atomic steps, the interactive mathematical proof system (IMPS) (Thayer, Farmer, and Guttman 1993) offers possibilities for educational use. These tools,

The Robbins algebra accomplishment reinforces the value of continued research in equational reasoning.

Approachable Opportunities (Examples)

Program synthesis
- The building of programs using existing modules as subroutines

Verification
- Partial verification of prototype commercial chips
- The integration of theorem-proving and model-checking methodologies

Mathematics
- Theorem provers with natural language output
- Theorem provers incorporating computer algebra capabilities
- The guiding of proof discovery with counterexample use

Education
- Tutors for algebra and combinatorial mathematics

Formal methods
- Formal theory use in the social sciences

and similar tools with extended capabilities, could be very effective in addressing the subjects that are so difficult for many students. Again, a major challenge in this domain is to develop complete educational packages capable of use in isolation. Once achieved, the packages can be sent to thousands of school systems for use by individual students or for integration into regular courses. (This integration has occurred now at the college level for ETPS [Goldson, Reeves, and Bornet 1993].) The expertise and support measures needed to actually make this work in school systems will need to come from outside the automated deduction research community but will need our involvement.

Beyond the Subareas Some topics of importance do not fall into categories by application subarea. I consider such topics here.

Interfaces: An important issue is that of human-machine and machine-machine interfaces. One pure case of machine-machine interface is the combined HOL-NUPRL effort. In this case, the interface was a significant undertaking, with serious theoretical work needed to accomplish the interface (see comments on the HOL-NUPRL Project in Verification). Incorporation of one system into another (for example, the use of MATHEMATICA in ANALYTICA) and even integration of procedures present interface problems. These interface problems are nontrivial in most cases and real challenges in some cases, as noted previously. Support for this aspect of automated deduction research is important. It clearly is money efficiently spent because it provides for combinations of existing powers, in effect getting the best of two or more worlds.

At least as important as the machine-machine interface issue is the human-machine issue. For the most part, the systems discussed in this article are difficult to use in that considerable training is needed to use them and interpret their output. An exception is the Chou-Gao-Zhang geometry programs. Some programs, such as OTTER, that are used outside the automated deduction research community have made an effort to be user friendly, with some success at a primitive level. What is needed before automated deduction systems really will be used by the outside community are significant interface upgrades, such as input and output in the user's language at the level humans communicate to colleagues. Ideally, such an interface would incorporate natural language input-output, discussed later, but certainly it includes input parameters determined only by the human description of the problem, not process-determined parameters. Interactive automated deduction systems have dramatic human-machine interface concerns, but all systems must address this issue. Many automated deduction researchers give human-machine interface research high priority in the list of research challenges for the automated deduction field.

Attention has been given to making the output of automated provers more readable. The normal forms and inference rules used by most provers result in proofs that are difficult for humans to understand. One early attempt to address this problem was undertaken by Andrews, who defined a procedure, later improved by Miller, for converting the logical structures of his proof system to a natural deduction proof style (Miller 1984; Andrews 1981). More recently, this conversion of logical structure to natural deduction proof style has been pushed one step further with investigations into translating proofs to natural language. Researchers in Germany have taken the lead here; I mention two major research efforts under way in Germany. One system, ILF, operates a mail server that accepts proofs directly from SETHEO, SPASS, and certain other first-order logic theorem provers. ILF prepares a LATEX file with a natural language presentation of the proof (Dahn and Wolf 1996). (More generally, the ILF system seeks to provide the nonexpert with a shell through which the user can use one of several theorem provers and related systems such as model checkers (Dahn et al. 1997). The second system, PROVERB, first takes a resolution proof as input and translates it into a natural deduction proof. After several levels of restructuring and abstraction, the latter

stages based on state-of-the-art natural language–generation techniques, a proof with English text is produced (Huang and Fiedler 1996). Some of the output is strikingly close to that appearing in a textbook. Because mathematics often can be presented with limited natural language support, the presentation of proofs in human-readable form is an excellent limited domain for natural language–generation research. (Significant progress has been made in natural language understanding and generation in recent years, but the complexity of the general task means that restricted domains are chosen for development and display of natural language capabilities.) The First International Workshop on Proof Transformation and Presentation was held in spring 1998 in Germany to bring together researchers in the different disciplines that impinge on this problem. The automated deduction field can contribute to this task significantly because it is central to one of two (perhaps complementary) approaches to this problem. One is better natural language capabilities, and the other is better logics and associated automated inference systems. More readable logics directly reduce the difficulty of translation.

Proof planning: Having completed the discussion of computer interface issues, I return to issues of the deductive structure of automated deduction systems. A research area of importance yet to be addressed is proof planning. Although this area is difficult, it is indeed very important and warrants more attention than it has received. A few groups in the United States and Europe are investigating planning. Techniques under exploration include working with user-supplied diagrams (Barker-Plummer, Bailin, and Ehrlichman 1996; Barker-Plummer and Bailin 1992) and using proof by analogy (Melis and Whittle 1996; Owen 1990). Other techniques, such as learning, are being studied in the AI community and should be applicable at some point. (For one attempt to apply learning to automated theorem proving, see Denzinger and Schulz [1996].) Any success in proof planning will have a strong effect on the capabilities of provers, but the difficulty of the problem has discouraged many from undertaking serious work in this area. When good work is located, it should be supported strongly. (Examples are work of Barker-Plummer and Bailin in the United States on the use of diagrams in theorem proving noted earlier and Bundy's Edinburgh group in Europe addressing proof planning in various forms, some noted earlier; for another example, see Bundy et al. [1991].) Relevant AI work is considered later.

Higher-order systems: I discussed many facets of higher-order logic theorem provers within the various areas, particularly the verification area. However, the discussion usually has been applicable to all interactive provers, of which the higher-order logic provers are a subclass. The higher-order logic systems have special concerns that warrant separate attention. For example, the representation problem is greatly enhanced because of the much more flexible substitution capability. If unification (the heart of the resolution scheme at the first-order–logic level) is used, then to representation problems are added computation problems because the full unification procedure is generally unbounded. The central focus is to make these provers more automated while we deal with complexities such as those just mentioned. The representation power of higher-order logic often allows much shorter proofs relative to first-order logic, with the potential for much smaller search spaces, but whether this potential for smaller search spaces will be realized in fully automated systems is not clear at present. There is certainly evidence of the power of higher-order logic in interactive mode with an appropriately trained person. (I remind the reader that the expressive powers are the same; first-order logic can express anything expressible in higher-order logic by use of set theory.) That a number of higher-order logic systems are under development indicates the potential many see for such systems. Verification, the most immediate commercial application, need not be done by using higher-

Very Difficult Tasks in Automated Deduction (Examples)

Program synthesis
- Free-form program synthesis

Verification
- Fully automated software verification

Mathematics
- Proof discovery in continuous mathematics

Foundations
- A complete archive of formally proof-checked mathematics

Automated Deduction in the United States and Europe: An Observation

Funding of research in the automated deduction field in the United States has declined relative to the funding of automated deduction research in Europe over the past decade. The effect of the shift in relative funding levels is reflected in the number of contributors of papers to the premier conference of the field, the Conference on Automated Deduction (CADE). The table given here reports the percentage of authors from the United States, Europe, and other locales for the CADE conferences of the past decade through 1996. Funding of automated deduction research in the United States traditionally has come from two sources: the United States Department of Defense (DoD) and the National Science Foundation (NSF). The DoD's interest centers on automated verification of hardware and software systems. This interest peaked in the 1970s and has declined steadily since then. Figures for the support levels are not available. The NSF support has been almost constant over the past decade at roughly $1 million in the Numeric, Symbolic, and Geometric Computation Program, the core program for support of automated deduction research. This support is often augmented by cost sharing from the AI, software engineering, and logic programs of NSF, which provides as much as 25 percent in additional funds.

Support for automated deduction research in Europe has increased substantially in the past decade, primarily in Germany. Support has been present in Europe since the 1950s, with strong work done in France, Germany, and the United Kingdom in particular. (The University of Edinburgh was one of the first centers of automated deduction research.) The recent increase in support comes primarily from Germany, with the centerpiece being the DFG Schwerpunktprogramm Deduktion. This program spans the years 1992 to 1998 and provides approximately $2 to $3 million a year (U.S. dollar equivalent) for basic automated deduction research, with a special interest in applications to programming. There is increased support in other countries but nothing as large or identifiable as the increase in Germany.

Origins of Research Papers for the Conference on Automated Deduction, 1988–1996.

Year	U.S. (%)	Europe (%)	Other	Site
1988	61	35	4	Argonne, Illinois
1990	51	45	4	Kaiserslautern, Germany
1992	44	49	7	Saratoga Springs, New York
1994	23	73	4	Nancy, France
1996	19	79	2	New Brunswick, New Jersey

Source of Table: Automated Deduction as an International Discipline—An Open Letter to the President of CADE. In *Association for Automated Reasoning Newsletter* 35, December 1996. The letter was authored by Pierre Lescanne (Nancy, France) and Christoph Walther (Darmstadt, Germany). Some text regarding sites has been altered.

order logic, as NQTHM and ACL2 demonstrate. However, the success of PVS and other systems shows that higher-order logic can be well used in this domain. The potential of this approach justifies continued interest and support here.

Deduction engineering: In a discussion of the development of automated deduction systems, it is important to understand a characteristic of development that can be called *deduction engineering*. It denotes the often nonglamorous, time-consuming process of adjusting a functioning deductive system for improved performance. Deduction engineering can involve anything from a new decision procedure to a new heuristic for finding an induction variable to a new data structure that speeds retrieval of terms used in rewriting or subsumption. Deduction engineering includes the important process of strategy formulation, itself having a range of outcomes, from publishable restrictions of considerable sophistication to the simple delaying of the use of clauses that have many free variables. That this difficult evolution-oriented engineering process is key to much of the progress in the automated deduction field has policy implications that need addressing. A major implication is that development is more resource consuming, in time and energy, than is expected, and sufficient support is needed to allow systems to progress. Another implication is that if most progress is found now by augmenting systems rather than implementing new basic procedures, then the systems grow in complexity. As such, they take more resources to sustain, so that fewer systems are sustainable. Perhaps that the more complex systems take more resources should be anticipated by giving funding priority to more established

centers, and isolated researchers would do better to cooperate with existing centers, much as physicists link to locations housing accelerators. The question of funding priorities and cooperation with existing centers is a complex question, regarding the validity of both the reasoning and the resource allocation.

I illustrate the nature of deduction engineering by anecdote to impart some feeling for the nature of development. One anecdote highlights the need to augment proof procedures by deduction strategies that restrict proof search size. I consider what might be the first instance where a restriction strategy was superimposed on a complete proof calculus, the resolution calculus as it happens. The resolution strategy with the subsumption test to purge redundant information was a powerful inference system; it appeared to Larry Wos and his colleagues at Argonne to far surpass the alternatives of the day (1964). However, the new resolution prover failed to prove a simple group theory problem regarding exponent two groups. Wos noted that many formulas (clauses) seemed to be general expressions pertaining to group theory and conjectured that more focus on the theorem under consideration was needed. He devised a strategy that separated clauses into two classes, one class to contain clauses that had used special theorem hypotheses or the conclusion's denial in their derivation. This class, the supported set, was given strong preference by requiring its use in each binary inference made. The effect was to obtain the theorem in three central processing unit seconds, whereas without the strategy, memory was exhausted and no proof found. The set of support strategy is one of the most important strategies currently used by OTTER (Wos 1996; Wos, Robinson, and Carson 1965).

Another story regarding deduction engineering addresses the difficulty often encountered in implementing what seems promising on paper. The example concerns the adoption of an improved version of the modification method (Brand 1975) (a 1970s algorithm in base form) by SETHEO to "build in" equality, defining E-SETHEO. When first installed, the modification method performed far worse than direct use of the equality axioms. Only by eventually revising the very basic linear architecture of the underlying model-elimination procedure (Loveland 1969) was the new feature made to yield performance superior to the naive approach. Given E-SETHEO's win in the competition reported earlier, the effort was justified. Sometimes progress is painful and accomplished only when sufficient resources are present.

The same pragmatic, resource-consuming energy goes into improving interactive provers, indeed into any deduction system that is performing at the state of the art today. Successful systems are composites and will become more so. Resources must exist to let the systems grow in capability because they usually grow more slowly than we would like.

Incorporating AI: The interaction between automated deduction and AI deserves comment because automated deduction studies began together with AI 40 years ago. Many of the topics covered here, such as program synthesis and proof discovery, are major AI topics. Outside these areas, AI has undergone a paradigm shift in methodology in recent years; stochastic and decision-theoretic methods are predominant in areas such as vision, natural language understanding, expert systems, learning, and even certain types of problem solving. Of course, not all AI research disavows formal methods and use of automated deduction tools. The study of formalizing contexts so that expert systems can know the bounds of their expertise, for example, is a current domain for formal methods.

AI planning is another area using formal methods. One of the major approaches to planning in AI is the explicit use of logic. Planning involves modeling and manipulating a dynamic world, and AI researchers have developed special logics to address action and change. The situational calculus introduced by McCarthy (McCarthy and Hayes 1969; McCarthy 1968) has recently been substantially enhanced in expressibility, and real-world applications are being pursued. For example, the Cognitive Robotics Group at the University of Toronto has developed the GOLOG programming language for tasks such as controlling robots and software agents (Levesque et al. 1996). The need for automated deduction can be substantial in this robot domain as the problem of incomplete knowledge is addressed, for example, in the setting of multiagent interaction. The other major approach to planning in the AI community is procedural, mostly using the STRIPS methodology, akin to databases with explicit updates. This approach blossomed in the 1970s when the theorem-proving approach seemed not to scale up. Recently, there has been a return to the use of (propositional) logic with the discovery of fast stochastic local search methods for determining satisfiability and methods to encode planning as a satisfaction problem (Kautz and Selman 1996). Although the local search methods usually outperform the systematic automated deduction methods such as the Davis-Putnam

If one were to attempt a single image, phrase, or slogan for the focal point of future research in automated deduction, the best choice could be the word chosen by Larry Wos: strategies.

procedure on the satisfiability problems that produce plans, the systematic procedures are useful in a complementary role. For example, to show that simpler plans do not exist generally requires more constrained presentations to be shown unsatisfiable, something the local search satisfiability programs cannot do (Kautz and Selman 1996). As an aside, I note that a recent paper shows how to extend recent satisfiability methods with the *lifting* technique central to automated deduction to reduce the size (complexity) of the problem, at least asymptotically (Kautz, McAllester, and Selman 1996). That automated deduction methodology and tools are reappearing in the planning realm is significant and, perhaps, contains a lesson. The use of automated deduction that seemed so natural in AI in the early years of AI but never scaled up might indeed return to play a role in a number of AI domains. Again, it is simply that the problems, and thus the technology to solve the problems, are much more difficult than we understood.

Another area associated with AI is learning. In machine learning, the investigation into inductive logic programming is having considerable success, primarily in Europe. The learning model has been applied to drug design, finite-element analysis, natural language understanding, and other areas (Bratko and Muggleton 1995). The technique calls for generalizations from positive examples so as not to intersect negative examples. The determination of logical consequence is essential here, and more powerful inference techniques, such as better methods for controlling search, could enable larger problems to be undertaken.

In the other direction, there is interest in using AI techniques to help automated deduction. Experiments in the use of neural nets to learn parameter control in theorem provers have experienced modest success. A track of the FLAIRS 1997 Conference (Daytona Beach, Florida, 11–14 May) addressed AI methods for controlling search in automated deduction systems. Real progress here can have a powerful effect on theorem prover capability.

A Wider Role The scope of automated deduction is not limited to the traditional areas of computer science and mathematics. In response to my call for contributions on future directions, Michael Musuch, Jaap Kamp, and others at the University of Amsterdam highlighted one of the most important long-term roles of automated deduction: the use of automated deduction in building and testing theories. Musuch and colleagues are investigating the use of automated deduction techniques in building formal theories in the social sciences (Péli and Musuch 1997; Péli et al. 1994). These formal theories allow the testing of hypotheses for consistency and logical consequence. The ability to determine the logical consequences of a formally specified theory not only saves expensive field studies but yields a more precise representation of the current hypotheses of the field. Alternative theories are more easily assessed as well. This opportunity to use formal methods to advance and test various theories highlights the need for more axiomatizations of subdisciplines and problem areas.

Looking to the more distant future, we see that the use of formal methods to study major disciplines can be the most significant intellectual application of automated deduction technology because it can encompass most of human intellectual endeavors. Our field has undertaken to build general reasoning engines that make the idea of formal theories of knowledge useful. Within the realm of mathematics, I have put forth proposals to verify, classify, and archive the main results of the various branches of the subject. The QED Project, now in the formative stage, has the audacious goal of building a computer system to effectively represent all important mathematical knowledge and techniques (QED 1994). Important questions arise immediately, such as the question of presentation of the mathematical theorems and theories. Should we strive to have theorems verified within a fixed framework? Is it acceptable to have multiple systems certified for verification of theorems? Phrasing this at a more fundamental level, do we choose a constructivist mathematics as our framework (perhaps a bias of computer science–oriented mathematicians) or a classical foundation? The QED Project is clearly a long-range project but one where success adds a new dimension to the world of mathematics. More correctly, it would extend the ambitions of the Bourbaki (1968) group (to formalize mathematics) to a level of rigor only dreamed about when the Bourbaki effort began.

Another project, the Mizar Project, originally undertaken by Polish mathematicians to provide software support for writing mathematics papers, has developed into an attempt to formalize large portions of mathematics. Thus, many regard this project as an interesting start toward the QED concept. The Mizar Project is well under way, even maintaining its own electronic journal, the *Journal of Formalized Mathematics*. Although some technical material appears in hard copy, the project is

truly oriented to effectively use the capabilities of the electronic medium, including extensive cross-referencing practical only in an electronic environment. For more information, the reader should consult the web at cs.ualberta.ca:80/piotr/Mizar/. What now exists is a substantial number of formal proofs, proof checked by machine based on a fixed axiom set. The effort now has some foreign (to Poland) sites active in the effort. Much has been accomplished with little funding, and some people in the automated deduction community feel strongly that this project should receive more financial support from the international community. (We understand that the *Journal of Formalized Mathematics* does receive funding from the Office of Naval Research.) These attempts to formalize the core of mathematics will certainly be paralleled by projects in engineering, medicine, law (under way at present), and the social sciences. For those who dismiss this projection with the comment that all such models (except mathematics and law) should be stochastic, I remind them that no probabilistic inference theory yet exists that is practical for large formal systems. It might be true that traditional formal theories can be viewed only as approximations for the real world in most settings, but, as Musuch's note is telling us, they are still of great importance when complex domains are under study.

Conclusion

If one were to attempt a single image, phrase, or slogan for the focal point of future research in automated deduction, the best choice could be the word chosen by Larry Wos: *strategies*. I used this word to describe a range of technical innovations within deduction engineering that are centered on the need to direct proof searches within highly restricted search spaces. One can also extend this notion to include better methods for induction hypotheses selection and the construction of effective proof-planning paradigms, for example. I leave to the reader the review of the preceding sections to see the number of important activities that can be captured by a fluid notion of the concept strategies.

In this article, I listed some of the achievements of the automated deduction field and considered some future directions for the field. We have seen that practical applications are now appearing and that more are likely in the near future. The most grandiose applications, such as full program synthesis, the mechanized mathematician, or the general reasoning machine, are not yet within view, and I should not venture to state when they will arrive. However, such automated deduction applications clearly can and will be developed, and they will be enormously beneficial.

Acknowledgments

The workshop on which this article and the underlying report are based was attended by 23 U.S. researchers, listed here; Wolfgang Bibel from the Technische Hochschule Darmstadt, Germany; Kamal Abdali, program director, Numeric, Symbolic, and Geometric Computation Program, Division of Computer and Computation Research; and Richard Kieburtz, division director, Division of Computer and Computation Research, NSF. Wolfgang Bibel was asked to represent the European branch of automated deduction in his role as initiator and director of the (then-existing) DFG Schwerpunkt Deduktion Project (see sidebar). The U.S. researchers attending were Peter Andrews (CMU), Robert Boyer (University of Texas at Austin), Shang-Ching Chou (Wichita State University), Edmund Clarke (CMU), Robert Constable (Cornell University), Nachum Dershowitz (University of Illinois), Xiao-Shan Gao (Wichita State University), Elsa Gunter (Lucent Technologies), Lawrence Henschen (Northwestern University), Deepak Kapur (State University of New York at Albany), Matt Kaufmann (Motorola Corporation, Austin), Kenneth Kunen (University of Wisconsin at Madison), Donald Loveland (Duke University), Michael Lowry (NASA Ames Research Center), Ewing Lusk (Argonne National Laboratory), William McCune (Argonne National Laboratory), Ross Overbeek (Argonne National Laboratory), Natarajan Shankar (SRI International), Mark Stickel (SRI International), Richard Waldinger (SRI International), Larry Wos (Argonne National Laboratory), Hantao Zhang (University of Iowa), and Xudong Zhao (CMU).

Besides the contributions of the participants during the workshop itself, there were many added contributions of time and effort to the original report that are acknowledged in the report preface, and I acknowledge them collectively at this point. Without this added assistance, there likely would have been no report. The workshop was funded by NSF grant CCR-9625544.

Notes

1. These are some leading proof systems. Unfortunately, I cannot list all the important systems here.

2. See the web site at www.mcs.anl.gov/home/mccune/ar/ for information on OTTER and pointers to web pages for many other systems named in this article.

References

Andrews, P. B. 1981. Transforming Matings into Natural Deduction Proofs. In *Proceedings of the Fifth Conference on Automated Deduction,* eds. W. Bibel and R. Kowalski, 281–292. Lecture Notes in Computer Science 87. Berlin: Springer-Verlag.

Andrews, P. B.; Bishop, M.; Issar, S.; Nesmith, D.; Pfenning, F.; and Xi, H. 1996. TPS: A Theorem-Proving System for Classical Type Theory. *Journal of Automated Reasoning* 16(3): 321–353.

Astrachan, O. L. 1994. METEOR: Exploring Model Elimination Theorem Proving. *Journal of Automated Reasoning* 13(3): 283–296.

Bachmair, L.; Ganzinger, H.; Lynch, C.; and Snyder, W. 1992. Basic Paramodulation and Superposition. In *Proceedings of the Eleventh Conference on Automated Deduction,* ed. D. Kapur, 462–476. Lecture Notes in Artificial Intelligence 607. Berlin: Springer-Verlag.

Barker-Plummer, D., and Bailin, S. C. 1992. Graphical Theorem Proving: An Approach to Reasoning with the Help of Diagrams. In *Proceedings of the European Conference on Artificial Intelligence,* 55–59. New York: Wiley.

Barker-Plummer, D.; Bailin, S. C.; and Ehrlichman, S. M. T. 1996. Diagrams and Mathematics. Paper presented at the Fourth International Symposium on Artificial Intelligence and Mathematics, eds. B. Selman and H. Kautz, 3–5 January, Fort Lauderdale, Florida.

Barwise, J., and Etchemendy, J. 1994. Hyperproof. Stanford, Calif.: Center for the Study of Language and Information.

Barwise, J., and Etchemendy, J. 1992. The Language of First-Order Logic. 3d ed. Stanford, Calif.: Center for the Study of Language and Information.

Beatty, D. L., and Bryant, R. E. 1994. Formally Verifying a Microprocessor Using a Simulation Methodology. In Proceedings of the Thirty-First Design Automation Conference, 596–602. New York: Association of Computing Machinery.

Bledsoe, W. W., and Feng, G. 1993. SET-VAR. *Journal of Automated Reasoning* 11(3): 293–314.

Bledsoe, W. W., and Hines, L. M. 1980. Variable Elimination and Chaining in a Resolution-Based Prover for Inequalities. In *Proceedings of the Fifth Conference on Automated Deduction,* eds. W. Bibel and R. Kowalski, 70–87. New York: Springer Verlag.

Bourbaki, N. 1968. *Elements of Mathematics*. Reading, Mass.: Addison Wesley.

Boyer, R. S., and Moore, J. S. 1988. *A Computational Logic Handbook*. San Diego, Calif.: Academic.

Brand, D. 1975. Proving Theorems with the Modification Method. *SIAM Journal on Computing* 4:412–430.

Bratko, I. 1990. *Prolog Programming for Artificial Intelligence*. Reading, Mass.: Addison-Wesley.

Bratko, I., and Muggleton, S. 1995. Applications of Inductive Logic Programming. *Communications of the ACM* 38(11): 65–70.

Bundy, A.; Stevens, A.; van Harmelen, F.; Ireland, A.; and Smaill, A. 1993. RIPPLING: A Heuristic for Guiding Induction Proofs. *Artificial Intelligence* 62(2): 185–253.

Bundy, A.; van Harmelen, F.; Hesketh, J.; and Smaill, A. 1991. Experiments with Proof Plans for Induction. *Journal of Automated Reasoning* 7(3): 303–324.

Burch, J. R.; Clarke, E. M.; McMillan, K. L.; Dill, D. L.; and Hwang, L. J. 1992. Symbolic Model Checking: 10^{20} States and Beyond. *Information and Computation* 98(2): 142–170.

Chou, S. C. 1988. *Mechanical Geometry Theorem Proving*. Dordrecht, The Netherlands: D. Reidel.

Chou, S. C.; Gao, X. S.; and Zhang, J. Z. 1994. *Machine Proofs in Geometry*. Singapore: World Scientific.

Clarke, E. M., and Zhao, X. 1996. Combining Symbolic Computation and Theorem Proving: Some Problems of Ramanujan. In P*roceedings of the Thirteenth Conference on Automated Deduction,* eds. M. A. McRobbie and J. K. Slaney, 758–763. Lecture Notes in Artificial Intelligence 1104. Berlin: Springer-Verlag.

Clarke, E. M., and Zhao, X. 1993. ANALYTICA: A Theorem Prover for MATHEMATICA. *The Mathematica Journal* 3(1): 56–71.

Clarke, E. M.; Emerson, E. A.; and Sistla, A. P. 1986. Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications. *ACM Transactions on Programming Languages and Systems* 8(2): 244–263.

Constable, R., and Murthy, C. 1991. Finding Computational Content from Classical Proofs. In *Logical Frameworks,* eds. G. Huet and G. Plotkin, 341–362. Oxford, U.K.: Cambridge University Press.

Constable, R. L.; Allen, S. F.; Bromley, H. M.; Cleaveland, W. R.; Cremer, J. F.; Harper, R. W.; Howe, D. J.; Knoblock, T. B.; Mendler, N. P.; Panangaden, P.; Sasaki, J. T.; and Smith, S. F. 1986. *Implementing Mathematics with the NUPRL Proof Development System*. New York: Prentice Hall.

Coquand, T., and Huet, G. 1988. The Calculus of Constructions. *Information and Computation* 76(2): 95–120.

Craigen, D.; Kromodimoeljo, S.; Meisels, I.; Pase, B.; and Saaltink, M. 1992. EVES System Description. In *Proceedings of the Eleventh Conference on Automated Deduction,* ed. D. Kapur, 771–775. Lecture Notes in Artificial Intelligence 607. Berlin: Springer-Verlag.

Craigen, D.; Kromodimoeljo, S.; Meisels, I.; Pase, B.; and Saaltink, M. 1991. EVES: An Overview. In Proceedings of VDM'91, Noordwijkerhout, The Netherlands, October.

Dahn, B. I., and Wolf, A. 1996. Natural Language Presentation and Combination of Automatically Generated Proofs. In *Proceedings of the First Conference on Frontiers of Combining Systems,* 175–192. Norwell, Mass.: Kluwer Academic.

Dahn, B. I.; Gehne, J.; Honigmann, T.; and Wolf, A. 1997. Integration of Automated and Interactive Theorem Proving in ILF. In *Proceedings of the Fourteenth Conference on Automated Deduction,* ed. W. W. McCune, 57–60. Lecture Notes in Artificial Intelligence 1249. Berlin: Springer-Verlag.

Davis, M., and Putnam, H. 1960. A Computing Procedure for Quantification Theory. *Journal of the ACM* 7:201–215.

Davis, M.; Logemann, G.; and Loveland, D. 1962. A Machine Program for Theorem Proving. *Communications of the ACM* 5(7): 394–397.

Dechter, R., and Pearl, J. 1988. Network-Based Heuristics for Constraint-Satisfaction Problems. *Artificial Intelligence* 34(1): 1–38.

Dechter, R., and Rish, I. 1994. Directional Resolution: The Davis-Putnam Procedure Revisited. In Proceedings of the Third International Conference on Knowledge Representation and Reasoning (KR'94), 134–145. San Francisco, Calif.: Morgan Kaufmann.

Denzinger, J., and Schulz, S. 1996. Learning Domain Knowledge to Improve Theorem Proving. In *Proceedings of the Thirteenth Conference on Automated Deduction,* eds. M. A. McRobbie and J. K. Slaney, 62–76. Lecture Notes in Artificial Intelligence 1104. Berlin: Springer-Verlag.

Detlefs, D. 1996. An Overview of the Extended Static Checking System. In Proceedings of the First Workshop on Formal Methods in Software Practice (FMSP '96), 1–9. New York: Association for Computing Machinery.

Deville, Y. 1990. *Logic Programming: Systematic Program Development*. Reading, Mass.: Addison-Wesley.

Fujita, M.; Slaney, J.; and Bennett, F. 1993. Automatic Generation of Some Results in Finite Algebra. In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, 52–57. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Ganzinger, H., ed. 1996. *Proceedings of the Seventh International Conference on Rewriting Techniques and Applications, RTA-96*. Lecture Notes in Computer Science 1103. Berlin:

Springer-Verlag.

Gelernter, H. 1963. Realization of a Geometry Theorem-Proving Machine. In *Computers and Thought,* eds. E. Feigenbaum and J. Feldman, 134–152. New York: McGraw-Hill.

Gelernter, H.; Hanson, J. R.; and Loveland, D. W. 1963. Empirical Explorations of the Geometry-Theorem Proving Machine. In *Computing and Thought,* eds. E. Feigenbaum and J. Feldman, 153–163. New York: McGraw-Hill.

Goldson, D.; Reeves, S.; and Bornet, R. 1993. A Review of Several Programs for the Teaching of Logic. *The Computer Journal* 36(4): 373–386.

Gordon, M. J., and Melham, T. F., eds. 1993. *Introduction to HOL: A Theorem-Proving Environment for Higher-Order Logic*. Oxford, U.K.: Cambridge University Press.

Green, C. 1969. Theorem Proving by Resolution as a Basis for Question-Answering Systems. In *Machine Intelligence 4,* eds. B. Meltzer and D. Michie, 183–205. Edinburgh, U.K.: Edinburgh University Press.

Henzinger, T. A., and Ho, P.-H. 1995. HYTECH: The Cornell Hybrid Technology Tool. In *Hybrid Systems II,* 265–294. Lecture Notes in Computer Science 999. Berlin: Springer-Verlag.

Hines, L. 1994. STR _ +VE and Integers. In *Proceedings of the Twelfth Conference on Automated Deduction,* ed. A. Bundy, 416–430. New York: Springer Verlag.

Ho, P. H. 1995. The Algorithmic Analysis of Hybrid Systems. Ph.D. thesis, Computer Science Department, Cornell University.

Howe, D. J. 1996. Importing Mathematics from HOL into NUPRL. In *Theorem Proving in Higher-Order Logics,* eds. J. von Wright, J. Grundy, and J. Harrison, 267–282. Lecture Notes in Computer Science 1125. Berlin: Springer-Verlag.

Howe, D. J. 1987. The Computational Behavior of Girard's Paradox. In Proceedings of the IEEE Symposium on Logic in Computer Science, 205–214. Washington, D.C.: IEEE Computer Society.

Huang, X., and Fiedler, A. 1996. Presenting Machine-Found Proofs. In *Proceedings of the Thirteenth Conference on Automated Deduction,* eds. M. A. McRobbie and J. K. Slaney, 221–225. Lecture Notes in Artificial Intelligence 1104. Berlin: Springer-Verlag.

Kapur, D., and Subramaniam, M. 1996. Mechanically Verifying a Family of Multiplier Circuits. In Proceedings of the Conference on Computer-Aided Verification, eds. R. Alur and T. Henzinger, 135–146. Lecture Notes in Computer Science 1102. Berlin: Springer-Verlag.

Kapur, D., and Zhang, H. 1996. An Overview of Rewrite Rule Laboratory. *Computers and Mathematics with Applications* 29(2): 91–114.

Kaufmann, M., and Moore, J. S. 1996. ACL2: An Industrial Strength Version of NQTHM. In Proceedings of the Eleventh Annual Conference on Computer Assurance, 23–34. Washington, D.C.: IEEE Computer Society.

Kautz, H., and Selman, B. 1996. Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search. In Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96), 1194–1201. Menlo Park, Calif.: American Association for Artificial Intelligence.

Kautz, H.; McAllester, D.; and Selman, B. 1996. Encoding Plans in Propositional Logic. In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning,* 374–384. San Francisco, Calif.: Morgan Kaufmann.

King, D., and Arthan, R. 1996. Development of Practical Verification Tools. *ICL Systems Journal* 11(1): 106–122.

Knuth, D. E., and Bendix, P. B. 1970. Simple Word Problems in Universal Algebra. In *Combinatorial Problems in Abstract Algebras,* ed. J. Leech, 263–270. New York: Pergamon.

Kolata, G. 1996. With Major Math Proof Brute Computers Show Flash of Reasoning Power. *New York Times,* Sunday, 10 December, C1, C8–C10.

Kowalski, R. 1984. Robert Kowalski on Logic Programming: An Interview. *Future Generations Computer Systems,* 1(1): 79–83.

Kunen, K. 1996a. Moufang Quasigroups. *Journal of Algebra* 183(1): 231–234.

Kunen, K. 1996b. Quasigroups, Loops, and Associative Laws. *Journal of Algebra* 185(1): 194–204.

Levesque, H. J.; Reiter, R.; Lespérance, Y.; Lin, F.; and Scherl, R. B. 1997. GOLOG: A Logic Programming Language for Dynamic Domains. *Journal of Logic Programming* 31(1–3): 59–83.

Loveland, D. W. 1984. Automated Theorem Proving: A Quarter-Century Review. In Automated Theorem Proving: after 25 Years, eds. D. Bledsoe and D. Loveland, 1–45. Contemporary Mathematics 29. Providence, R.I.: American Mathematical Society.

Loveland, D. W. 1969. A Simplified Format for the Model Elimination Procedure. *Journal of the ACM* 16(3): 349–363.

McCarthy, J. 1968. Programs with Common Sense. In *Semantic Information Procession,* ed. M. Minsky, 403–418. Cambridge, Mass.: The MIT Press.

McCarthy, J., and Hayes, P. 1969. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In *Machine Intelligence 4,* eds. B. Meltzer and D. Michie, 463–502. Edinburgh, U.K.: Edinburgh University Press.

McCune, W. 1996. 33 Basic Test Problems: A Practical Evaluation of Some Paramodulation Strategies. Technical Report ANL/MCS-P618-1096, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois.

McCune, W. 1994. OTTER 3.0 Reference Manual and Guide. Technical Report ANL-94/6, Argonne National Laboratory, Argonne, Illinois.

McCune, W. 1993. Single Axioms for Groups and Abelian Groups with Various Operations. *Journal of Automated Reasoning* 10(1): 1–13.

McCune, M., and Wos, L. 1997. OTTER. The CADE-13 Competition Incarnations. *Journal of Automated Reasoning* 18(2): 211–220.

MacKenzie, D. 1996. The Automation of Proof, A Historical and Sociological Exploration. *IEEE Annals of the History of Computing* 17(3): 7–29.

Manna, Z., and Waldinger, R. 1980. A Deductive Approach to Program Synthesis. *ACM Transactions on Programming Languages* 2(1): 90–121.

Melis, E., and Whittle, J. 1996. Internal Analogy in Theorem Proving. In *Proceedings of the Thirteenth Conference on Automated Deduction,* eds. M. A. McRobbie and J. K. Slaney, 92–105. Lecture Notes in Artificial Intelligence 1104. Berlin: Springer-Verlag.

Miller, D. A. 1984. Expansion Tree Proofs and Their Conversion to Natural Deduction Proofs. In *Proceedings of the Seventh Conference on Automated Deduction,* ed. R. Shostak, 347–370. Lecture Notes in Computer Science 170. Berlin: Springer-Verlag.

Milner, R.; Tofte, M.; and Harper, R. 1990. *The Definition of Standard ML*. Cambridge, Mass.: MIT Press.

Moore, J. S. 1994. Introduction to the OBDD Algorithm for the ATP Community. *Journal of Automated Reasoning* 12(1): 33–45.

Moser, M.; Ibens, O.; Letz, R.; Steinbach, J.; Goller, C.; Schumann, J.; and Mayr, K. 1997. SETHEO and E-SETHEO: The CADE-13 Systems. *Journal of Automated Reasoning* 18(2): 237–246.

Murthy, C. 1991. An Evaluation Semantics for Classical Proofs. In Proceedings of the IEEE Symposium on Logic in Computer Science, 96–109. Washington, D.C.: IEEE Computer Society.

Nadathur, G., and Miller, D. 1988. An Overview of LAMBDA-PROLOG. In *Proceedings of the Fifth International Conference and Symposium on Logic Programming,* 810–827. Cambridge, Mass.: MIT Press.

Newell, A.; Shaw, J. C.; and Simon, H. A. 1963. Empirical Explorations of the Logic Theory Machine: A Case Study in Heuris-

tics. In *Computers and Thought,* eds. E. Feigenbaum and J. Feldman, 134–152. New York: McGraw-Hill.

Owen, S. 1990. *Analogy for Automated Reasoning.* San Diego, Calif.: Academic.

Owre, S.; Rushby, J. M.; and Shankar, N. 1992. PVS: A Prototype Verification System. In *Proceedings of the Eleventh Conference on Automated Deduction,* ed. D. Kapur, 748–752. Lecture Notes in Artificial Intelligence 607. Berlin: Springer-Verlag.

Paulson, L. C. 1994. *ISABELLE: A Generic Theorem Prover.* Lectures Notes in Computer Science 828. Berlin: Springer-Verlag.

Paulson, L. C. 1993. Set Theory for Verification: I. From Foundations to Functions. *Journal of Automated Reasoning* 11(3): 353–389.

Paulson, L. C., and Grabczewski, K. 1996. Mechanizing Set Theory. *Journal of Automated Reasoning* 17(3): 291–323.

Péli, G., and Masuch, M. 1997. The Logic of Propagation Strategies: Axiomatizing a Fragment of Organizational Ecology in First-Order Logic. *Organization Science* 8(3): 310–331.

Péli, G.; Bruggeman, J.; Masuch, M.; and Ó Nualláin, B. 1994. A Logical Approach to Formalizing Organizational Ecology. *American Sociological Review* 59(4): 571–593.

Plaisted, D. A., and Zhu, Y. 1997. Ordered Semantic Hyper Linking. In Proceedings of the Fourteenth National Conference on Artificial Intelligence, 472–477. Menlo Park, Calif.: American Association for Artificial Intelligence.

QED. 1994. The QED Manifesto. In *Proceedings of the Twelfth International Conference on Automated Deduction,* ed. A. Bundy, 238–251. Berlin: Springer-Verlag. A comprehensive view of QED can be obtained on the web site at www.mcs.anl.gov/qed.

Robinson, J. A. 1992. Logic and Logic Programming. *Communications of the ACM* 35(3): 40–65.

Robinson, J. A. 1965. A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM* 12(1): 23–41.

Slanet, J.; Fujita, M.; and Stickel, M. 1995. Automated Reasoning and Exhaustive Search: Quasigroup Existence Problems. *Computers and Mathematics with Applications* 29(2): 115–132.

Siekmann, J. H., and Wrightson, G., eds. 1983. *The Automation of Reasoning, Volumes 1 and 2.* Berlin: Springer-Verlag.

Smith, D. R. 1990. KIDS: A Semi-Automated Program Development System. *IEEE Transactions on Software Engineering* (Special Issue on Formal Methods) 16(9): 1024–1043.

Stickel, M. E. 1975. A Complete Unification Algorithm for Associative-Commutative Functions. In Proceedings of the Fourth International Joint Conference on Artificial Intelligence, 423–434. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Stickel, M.; Waldinger, R.; Lowry, M.; Pressburger, T.; and Underwood, I. 1994. Deductive Composition of Astronomical Software from Subroutine Libraries. In *Proceedings of the Twelfth Conference on Automated Deduction,* ed. A. Bundy, 341–355. Lecture Notes in Artificial Intelligence 814. Berlin: Springer-Verlag.

Suppes, P., and McDonald, J. 1984. Student Use of an Interactive Theorem Prover. In Automated Theorem Proving: After 25 Years, eds. W. Bledsoe and D. Loveland, 315–360. Contemporary Mathematics 29. Providence, R.I.: American Mathematical Society.

Sutcliffe, G., and Suttner, C. B. 1997. The Results of the CADE-13 ATP System Competition. *Journal of Automated Reasoning* 18(2): 271–286.

Suttner, C. B., and Sutcliffe, G. 1998. The CADE-14 ATP System Competition. *Journal of Automated Reasoning* 21(1): 99–134.

Suttner, C. B., and Sutcliffe, G. 1997. The Design of the CADE-13 ATP System Competition. *Journal of Automated Reasoning* 18(2): 139–162.

Thayer, F. J.; Farmer, W. M.; and Guttman, J. D. 1993. IMPS: An Interactive Mathematical Proof System. *Journal of Automated Reasoning* 11(2): 213–248.

Walsh, T.; Nunes, A.; and Bundy, A. 1992. The Use of Proof Plans to Sum Series. In *Proceedings of the Eleventh Conference on Automated Deduction,* ed. D. Kapur, 325–339. Lecture Notes in Artificial Intelligence 607. Berlin: Springer-Verlag.

Weidenbach, C. 1997. SPASS: Version 0.49. *Journal of Automated Reasoning* (Special Issue on the CADE-13 ATP System Competition) 18(2): 247–252.

Winker, S., and Wos, L. 1978. Automated Generation of Models and Counter Examples and Its Application to Open Questions in Ternary Boolean Algebra. In Proceedings of the Eighth International Symposium on Multiple-Valued Logic, 251–256. Washington, D.C.: IEEE Computer Society.

Wos, L. 1996. OTTER and the Moufang Identity Problem. *Journal of Automated Reasoning* 17(2): 215–257.

Wos, L.; Robinson, G.; and Carson, D. 1965. Efficiency and Completeness of the Set of Support Strategy in Theorem Proving. *Journal of the ACM* 12(4): 536–541.

Wos, L.; Robinson, G.; Carson, D.; and Shalla, L. 1967. The Concept of Demodulation in Theorem Proving. *Journal of the ACM* 14(4): 698–709.

Donald W. Loveland is a professor and present acting chair of the Computer Science Department at Duke University, which he joined as first chair in 1973. He received his Ph.D. in mathematics from New York University (NYU) and was on the faculties of NYU and Carnegie Mellon University prior to joining Duke. His long involvement with automated deduction includes defining the model-elimination proof procedure and the notion of linear resolution. He is author of a book on the logical basis of automated theorem proving and editor and coeditor of two other books in the field. He has also written in the areas of algorithms, complexity, expert systems, and logic programming. He is a fellow of the American Association for Artificial Intelligence. His e-mail address is dwl@cs.duke.edu.