Logical Reasoning Systems

David McAllester
AT&T Labs-Research

Logical Reasoning Systems derive sound conclusions from formal declarative knowledge. Such systems are usually defined by abstract *rules of inference*. For example, the rule of modus ponens states that given P, and "P implies Q" (usually written $P \rightarrow Q$), we can infer Q. Logical systems have a rich history starting with Greek syllogisms and continuing through the work many prominent mathematicians such as Déscartes, Leibniz, Boole, Frege, Hilbert, Gödel, and Cohen. A good discussion of the history of logical reasoning systems can be found in [3].

Logical reasoning provides a well understood general method of *symbolic computation*. Symbolic computation manipulates expressions involving variables. For example, a computer algebra system can simplify the expression $x(x+1)$ to x^2+x. The equation $x(x+1) = x^2+x$ is true under any interpretation of x as a real number. Unlike numerical computation, symbolic computation derives truths that hold under a wide variety of interpretations and can be used when only partial information is given, e.g., that x is some (unknown) real number. Logical inference systems can be used to perform symbolic computation with variables that range over conceptual domains such as sets, sequences, graphs, and computer data structures. Symbolic computation underlies esentially all modern efforts to formally verify computer hardware and software.

There are at least two ways in which symbolic inference is relevant to cognitive science. First, symbolic inference rules have traditionally been used as models of human mathematical reasoning. Second, symbolic computation also provides a model of certain common sense inferences. For example, suppose one is given a bag of marbles and continues removing marbles from the bag as long as it remains nonempty. People easily reach the conclusion that, barring unusual or magical circumstances, the bag will eventually become empty. They reach this conclusion without being told any particular number of marbles — they reach a conclusion about an arbitrary set s. When building computer systems for drawing such conclusions approaches based on "visualization" do not currently work as well as approaches based on symbolic computation [6].

Here I will divide symbolic computation research into five general types. First, there are the so-called symbolic algebra systems such as Maple or Mathematica [7]. These are designed to manipulate expressions satisfying certain algebraic properties such as those satisfied by the real numbers.

Although they have important applications in the physical sciences, these systems are not widely used for hardware or software verification and seem too specialized to provide plausible models of common sense reasoning. I will not discuss them further here. Second, are the symbolic model checking systems [2]. These systems perform symbolic inference where the variables range over finite sets such as the finite set of possible states of a certain piece of computer hardware. Although very effective for the finitary problems where they apply, these systems are too restricted for software verfication and also seem too restricted to provide plausible models of common sense reasoning. I will not discuss them further. The remaining three approaches to symbolic computation all claim to be general purpose, or domain independent. I will call these the first-order approach, the higher-order approach, and the induction approach. I will discuss each in turn.

The first order approach is based on making inferences from axioms written in first order logic. This includes a wide variety of resolution and term-rewriting systems [4]. For problems that can be naturally axiomatized in first order logic the first order systems seem superior to other approaches. Unfortunately, most mathematical theorems and verification problems have no natural first order formulation. A detailed discussion of first order logic and inference methods is beyond the scope of this article. However, it is possible to give a somewhat superficial description of its limitations. First order logic allows one to state properties of concepts but it does not generally allow us to *define* concepts. For example, suppose we want to describe the concept of a finite set. We can write a formal expression stating that "a set is finite if it is either empty or can be derived by adding a single element to some other finite set". Unfortunately this statement does not uniquely determine the concept — it is also true that "a set is countable if it is either empty or can be derived by adding a single element to some other countable set". Statements true of a concept often fail to uniquely specify them. Finiteness is not definable in first order logic. The bag of marbles inference mentioned above implicitly relies on finiteness. Almost all program verfication problems involve concepts not definable in first order logic. Methods of simulating more expressive logics in first order logic are generally inferior in practice to systems specifically designed to go beyond first order logic.

Higher order systems allow quantification over predicates (concepts) [5]. Finiteness is definable in higher order logics — by quantifying over predicates we can say that "finite" is the *least* predicate satisfying the condition given in the preceding paragraph. Unfortunately, the use of higher order logic

makes automation difficult. So computer systems based on higher order logic typically verify human-written derivations rather than attempt to find derivations automatically.

In practice there is no clear distinction between systems that verify human-written derivations and systems that find derivations automatically — there is a continuum in the degree of automation provided. Higher order computer systems do usually provide some degree of automation but it is typically less than the degree of automation provided by induction systems.

Induction systems represent a middle ground between the expressively weak first order resolution and term rewriting systems and the expressively strong systems based on higher order logic [1]. Induction systems are first order in the sense that they do not typically allow quantification over predicates. But, unlike first order systems, all objects are assumed to be finite. A variable in a symbolic expression ranges over infinitely many different possible values each one of which is a finite object. This is different from symbolic model checking where each variable ranges over only a finite number of possible values. Also induction systems allow well-founded recursive definitions of the kind found in functional programming languages. The underling logic of an induction system is best viewed as a programming language, such as cons-car-cdr Lisp. But unlike traditional implementations of a programming language, an induction system can derive symbolic equations, such as the following, which are true under any allowed interpretation of the variables apprearing in the expressions.

$$\mathtt{append}(\mathtt{x}, \mathtt{append}(\mathtt{y}, \mathtt{z})) = \mathtt{append}(\mathtt{append}(\mathtt{x}, \mathtt{y}), \mathtt{z})$$

Induction systems seem most appropriate for program verification and for modeling common sense reasoning about an arbitrary (or unknown) finite set.

References

[1] Robert S. Boyer and J Struther Moore, 1979, *A Computational Logic*. ACM Monograph Series. Academic Press.

[2] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and J. Hwang, 1990, Symbolic Model Checking:10^{20} States and Beyond, in *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society Press.

[3] M. Davis, 1983, The Prehistory and Early History of Automated Deduction, in *Automation of Reasoning 1*, Jörg Seikmann and Graham Wrightson, eds. Springer-Verlag.

[4] Melvin Fitting, 1996, *First-Order Logic and Automated Theorem Proving: Second Edition*. Springer-Verlag.

[5] M. Gordon, 1987, Hol: A proof generating system for higher-order logic, In *VLSI Specification, Verification, and Synthesis*, G. Birtwistle and P. A. Subrahmanyam, eds. Kluwer.

[6] David McAllester, 1991, Some Observations on Cognitive Judgments, In *Proceedings of the ninth national conference on artificial intelligence*, Morgan Kaufmann Publishers, pp. 910-915

[7] Stephen Wolfram, 1996, *Mathematica: Third Edition*, Wolfram Media Inc and Cambridge University Press.