AUTOMATIC DEDUCTION FOR COMMONSENSE REASONING:
AN OVERVIEW

Technical Note 239

April 1981

By: Robert C. Moore
      Artificial Intelligence Center
      Computer Science and Technology Division

SRI Projects 1350 and 1894

ABSTRACT

How to enable computers to draw conclusions automatically from
bodies of facts has long been recognized as a central problem in
artificial-intelligence (AI) research. Any attempt to address this
problem requires choosing an application (or type of application), a
representation for bodies of facts, and methods for deriving
conclusions. This article provides an overview of the issues involved
in drawing conclusions by means of deductive inference from bodies of
commonsense knowledge represented by logical formulas. We first briefly
review the history of this enterprise: its origins, its fall into
disfavor, and its recent revival. We show why applications involving
certain types of incomplete information resist solution by other
techniques, and how supplying domain-specific control information seems
to offer a solution to the difficulties that previously led to
disillusionment with automatic deduction. Finally, we discuss the
relationship of automatic deduction to the new field of "logic
programming," and we survey some of the issues that arise in extending
automatic-deduction techniques to nonstandard logics.

I   HISTORICAL BACKGROUND

Automatic deduction, or mechanical theorem proving, has been a major concern of AI since its earliest days. By the time of the first formal conference on AI, held at Dartmouth College in the summer of 1956, Newell and Simon [1956] had already written a deduction system for propositional logic, the Logic Theorist, and Minsky [McCorduck, 1979, p. 106] was developing the ideas that were later embodied in Gelernter's theorem prover for elementary geometry [Gelernter, 1963]. Shortly after this, Wang [1960] produced the first implementation of a reasonably efficient, complete algorithm for proving theorems in propositional logic.

Following these early efforts, the next important step in the development of automatic-deduction techniques was Robinson's [1965] description of a relatively simple procedure that he showed to be a logically complete method for proving theorems in first-order predicate calculus. Robinson's procedure and those derived from it are usually referred to as <u>resolution</u> procedures, because the basic rule of inference they use is the resolution principle:

From (A OR B) and (NOT(A) OR C), infer (B OR C)

Robinson's work had a major influence on two more or less distinct lines of research. One of these was mathematical theorem proving, which aims at providing practical tools for discovering new results in mathematics, but Robinson's work also had a major impact on research into commonsense reasoning and problem solving.

In this area Robinson's ideas brought about a rather dramatic shift in attitudes regarding automatic deduction. The early efforts at automatic theorem proving were generally thought of as exercises in expert problem solving, with the domain of application being

1

propositional logic, in the Logic Theorist, or geometry, in Gelernter's program. The resolution method, however, seemed powerful enough to make it possible to build a completely general problem solver by describing problems in first-order logic and deducing solutions by a general proof procedure.

This approach to commonsense reasoning and problem solving obviously assumes the use of formal logic as a knowledge representation formalism. The idea of using a formal logic as the representation language for a commonsense reasoning system, with the reasoning done by deductive inference, was apparently first put forward in 1959 by McCarthy in his "Advice Taker" proposal (included in [McCarthy, 1968]), with Black making the first serious attempt to implement McCarthy's idea in 1964 [Black, 1968]. Robinson's work provided encouragement for this approach, and a few years later Green [1969] carried out extensive experiments with a question-answering and problem-solving system based on resolution.

The results of Green's experiments and several similar projects were disappointing, however. The difficulty was that, in the general case, the search space generated by the resolution method grows exponentially with the number of formulas used to describe a problem, so that problems of even moderate complexity cannot be solved in reasonable time. Several domain-independent heuristics were proposed to try to deal with this issue, but they proved too weak to produce satisfactory results.

It appears that these failures resulted principally from two constraints the researchers had imposed upon themselves: they attempted to use only uniform, domain-independent, proof procedures; and they tried to force all reasoning and problem-solving behavior into the framework of logical deduction. In other words, like a number of earlier ideas such as "self-organizing systems" and "heuristic search," automatic theorem proving turned out not to be the magic formula that would solve all of AI's problems at once. In the reaction that followed, however, not only was there a turning away from attempts to

2

use deduction to create general problem solvers, but there was also widespread condemnation of _any_ use of logic or deduction in commonsense reasoning or problem solving. Arguments made by Minsky [1980, Appendix] and Hewitt [1973, 1975] seem to have been particularly influential in this regard.

II   WHY THE DEDUCTION PROBLEM WILL NOT GO AWAY

Despite the disappointments of the late 60s and early 70s, there has been a recent revival of interest in deduction-based approaches to commonsense reasoning. This is apparent in the work of McDermott [1978], Doyle [1979, 1980], and Moore [1980a, 1980b], current work on nonmonotonic reasoning [Bobrow, 1980], and recent textbooks by Nilsson [1980] and Kowalski [1979]. To a large extent this renewed interest seems to stem from the recognition of an important class of problems that resist solution by any other method.

To understand the crux of the issue, it should be noted that, even if one decides to use a representation formalism based on logic, it may not be necessary to use general deductive methods to manipulate expressions in the formalism. The key question is whether the description of a problem situation is complete in terms of the objects, properties, and relations relevant to the problem. If we have a complete description of a situation, we can answer any question by evaluation; deduction is unnecessary. To illustrate, suppose we have a knowledge base of personnel information for a company and we want to know whether there is any secretary who earns more than some programmer. We could express this question in first-order logic as:

    SOME (X, Y) ((TITLE(X) = SECRETARY) AND
                 (TITLE(Y) = PROGRAMMER) AND
                 (SALARY(X) > SALARY(Y)))

If we have recorded in our knowledge base the job title and salary of every employee, we can simply find the salary of each secretary and compare it with the salary of every programmer. No deduction is involved in this process. On the other hand, we may not have specific salary information for each employee. Instead we may have general information about classes of employees such as

4

```
All programmers are professionals.
ALL (X) ((TITLE(X) = PROGRAMMER) ->
        (CATEGORY(X) = PROFESSIONAL))

All secretaries are clericals.
ALL (X) ((TITLE(X) = SECRETARY) ->
        (CATEGORY(X) = CLERICAL))

All clericals earn less than all professionals.
ALL (X,Y) (((CATEGORY(X) = CLERICAL) AND
           (CATEGORY(Y) = PROFESSIONAL)) ->
          (SALARY(X) < SALARY(Y)))
```

From this infomation we can deduce that no secretary earns more than any programmer, although we have no information about the exact salary of any employee.

A representation formalism based on logic gives us the ability to express many kinds of generalizations, even when we don't have a complete description of the problem situation. Using deduction to manipulate expressions in the representation formalism allows us to ask logically complex queries of a knowledge base containing such generalizations, even when we cannot "evaluate" a query directly. On the other hand, AI inference systems that are not based on automatic-deduction techniques either do not permit logically complex queries to be asked, or they answer such queries by methods that depend on the possesion of complete information. Any knowledge representation formalism that is capable of handling the kinds of incomplete information people can understand must at least be able to

```
Say that something has a certain property without saying which
thing has that property:
SOME (X) P(X)

Say that everything in a certain class has a certain property
without saying what everything in that class is:
ALL (X) (P(X) -> Q(X))

Say that at least one of two statements is true without saying
which statement is true:
P OR Q

Explicitly say that a statement is false, as distinguished from
simply not saying that it is true:
NOT(P)
```

5

Any representation formalism that has these capabilities will be, at the very least, an extension of classical first-order logic, and any inference system that can deal adequately with these kinds of generalizations will have to have at least the capabilities of an automatic-deduction system.

III   THE NEED FOR SPECIFIC CONTROL INFORMATION

As we remarked above, the fundamental difficulty with attempting to base a general, domain-independent problem solver on automatic-deduction techniques is that there are too many possible inferences that can be drawn at any one time. Finding the inferences that are relevant to a particular problem can be an impossible task, unless domain-specific guidance is supplied to control the deductive process.

One type of guidance that is often critical to efficent system performance is information about whether to use facts in a forward-chaining or backward-chaining manner. The deductive process can be thought of as a bidirectional search process, partly working forward from known facts to new ones, partly working backward from goals to subgoals, and meeting somewhere in between. Thus, if we have a fact of the form (P -> Q), we can use it either to generate Q as a fact, given P as a fact, or to generate P as a goal, given Q as a goal. Early theorem-proving systems used every fact both ways, leading to highly redundant searches. More sophisticated methods were gradually devised that eliminated these redundancies. Eliminating redundancies, however, creates choices as to which way facts are to be used. In the systems that attempted to use only domain-independent control heuristics, a uniform strategy had to be imposed. Often the strategy was to use all facts only in a backward-chaining manner, on the grounds that this would at least guarantee that all the inferences drawn would be relevant to the problem at hand.

The difficulty with this approach is that the question of whether it is more efficient to use a fact for forward or backward chaining depends very much on the specific content of that fact. For instance, according to the Talmud, the primary criterion for determining whether someone is Jewish is:

7

```
ALL (X) (JEWISH(MOTHER(X)) -> JEWISH(X))
```

That is, a person is Jewish if his or her mother is Jewish.[*] Suppose we were to try to use this rule for backward chaining, as most "uniform" proof procedures would. It would apply to any goal of the form JEWISH(X), producing the subgoal JEWISH(MOTHER(X)). This expression, however, is also of the form JEWISH(X), so the process would be repeated, resulting in an infinite descending chain of subgoals:

```
GOAL: JEWISH(MORRIS)
GOAL: JEWISH(MOTHER(MORRIS))
GOAL: JEWISH(MOTHER(MOTHER(MORRIS)))
GOAL: JEWISH(MOTHER(MOTHER(MOTHER(MORRIS)))), etc.
```

If, on the other hand, we use the rule for forward chaining, the number of applications is limited by the complexity of the fact that originally triggers the inference:

```
FACT: JEWISH(MOTHER(MOTHER(MORRIS)))
FACT: JEWISH(MOTHER(MORRIS))
FACT: JEWISH(MORRIS)
```

It turns out, then, that the efficent use of a particular fact often depends on exactly what that fact is and also on the context of other facts in which it is embedded. Many examples illustrating this point are given by Kowalski [1979] and Moore [1980a], involving not only the forward/backward-chaining distinction, but other control decisions as well.

Since specific control information needs to be associated with particular facts, the question arises as to how to provide it. The simplest way is to embed it in the facts themselves. For instance, the forward/backward-chaining distinction can be encoded by having two versions of implication; e.g., (P -> Q) to indicate forward chaining and (Q <- P) to indicate backward chaining. This approach originated in the distinction made in the programming language PLANNER between antecedent and consequent theorems. A more sophisticated approach is to make such

----

[*] I am indebted to Richard Waldinger for suggesting this example.

8

decisions as whether to use a fact in the forward or backward direction themselves questions for the deduction system to reason about, by using so-called "metalevel" knowledge. The first detailed proposal along these lines seems to have been made by Hayes [1973], while experimental systems have been built by McDermott [1978] and de Kleer et al. [1979], among others. Weyhrauch [1980] has perhaps done the most to explore the kind of system architecture in which this sort of reasoning would be possible.

IV   THEORY FORMULATION AND LOGIC PROGRAMMING

Another factor that can greatly affect the efficiency of deductive reasoning is the exact way in which a body of knowledge is formalized. That is, logically equivalent formalizations can have radically different behavior when used with standard deduction techniques. For example, we could define ABOVE as the transitive closure of ON in at least three ways:[*]

```
ALL (X,Y) (ABOVE(X,Y) <->
        (ON(X,Y) OR SOME (Z) (ABOVE(X,Z) AND ON(Z,Y))))

ALL (X,Y) (ABOVE(X,Y) <->
        (ON(X,Y) OR SOME (Z) (ON(X,Z) AND ABOVE(Z,Y))))

ALL (X,Y) (ABOVE(X,Y) <->
        (ON(X,Y) OR SOME (Z) (ABOVE(X,Z) AND ABOVE(Z,Y))))
```

Each of these formalizations will produce different behavior in a standard deduction system, no matter how we make local control decisions of the kind discussed in the previous section.   Kowalski [1974] noted that choosing among alternatives such as these involves very much the same sort of decisions as are made writing programs in a conventional programming language.   In fact, he observed that there are ways to formalize many functions and relations so that applying standard deduction . methods will have the effect of executing them as computer programs. These observations have led to the development of the field of "logic programming" [Kowalski, 1979] and the creation of new computer languages such as PROLOG [Warren and Pereira, 1977].

----
[*] These formalizations are not quite equivalent, as they allow for different possible interpretations of ABOVE if infinitely many objects are involved.   They are equivalent, however, if only finitely many objects are being considered.

V   AUTOMATIC DEDUCTION IN NONSTANDARD LOGICS

So far, we have discussed automatic deduction for classical first-order logic only.   Many commonsense concepts, however, are most naturally treated in either higher-order or nonclassical logics.   This presents a problem, because classical first-order logic is the most general logic for which automatic-deduction techniques are at all well developed.   It turns out, though, that there are a number of techniques for reformulating representations in nonstandard logics in terms of logically equivalent representations in classical first-order logic.

Higher-order logic differs from first-order logic in that it allows quantification over properties and relations as well as individuals. That is, if we have a first-order logic that allows us to make statements about all physical objects, the corresponding second-order logic would allow us to make statements about all properties of and relations among physical objects. A third-order logic would allow us to make statements about properties of and relations among these properties and relations, and so forth.

In some cases, the transition from first-order to higher-order logic presents fewer difficulties than it might at first appear. In fact, the standard deductive procedures for first-order logic also work for higher-order logic, except that general predicate abstraction is not performed. That is, these procedures will not construct predicates out of arbitrary complex formulas. If "John is a man" is represented as MAN(JOHN), the predicate MAN can be retrieved when we ask the second-order question, "What properties does John have?" All the deduction system has to do is match X(JOHN) against MAN(JOHN) and return MAN as the value of the variable X. But, from the assertion that John is either a butcher or a baker, represented as

11

BUTCHER(JOHN) OR BAKER(JOHN),

the system could not infer, without using predicate abstraction, that John has the disjunctive property of being a butcher-or-baker. The system would have to recognize that this complex expression could be reformulated as a one-place predicate applied to JOHN,

(LAMBDA (Y) (BUTCHER(Y) OR BAKER(Y)))(JOHN),

which is of the right form to match X(JOHN).

If this sort of predicate abstraction is not required, standard first-order deduction techniques are sufficient. There has been some work extending the standard techniques to handle the more general case (e.g., [Huet, 1975]), but this makes the deduction problem much harder because of the combinatorics of all the different ways predicate abstraction may be performed.

Another problem commonly encountered is how to do automatic deduction in logics that allow intensional operators. These are operators, such as "believe" or "know," that produce sentences whose truth-value depends fully on the meanings, not just the truth-values, of their arguments. Classical logic is purely extensional, because the truth-value of a complex formula depends only on the extensions (denotations, referents) of its subexpressions. The extension of a formula is considered to be its truth-value, so the operator OR is extensional because the truth of (P OR Q) depends only on the truth of P and the truth of Q; no other properties of P and Q matter. "Believe," on the other hand, is intensional because the truth of "A believes that P" depends generally on the meaning of P, not just its truth-value.

Many of the rules of classical logic, such as substitution of equals for equals, do not apply within the scope of an intensional operator. To use a classic example, since "the morning star" and "the evening star" refer to the same object, it must be the case that "the morning star is Venus" is true if and only if "the evening star is Venus" is true. However, it might be that "John believes the morning

12

star is Venus" is true, but that "John believes the evening star is Venus" is false because, although the two embedded sentences possess the same truth-value, they differ in meaning.

Fortunately, many of the difficulties presented by intensional operators can be overcome by reformulating the statements in which they occur. There are a number of methods for doing this, but one that is particularly elegant is to reformulate intensional operators in terms of their "possible-world semantics" [Kripke, 1971] [Hintikka, 1971]. The idea is that, rather than talking about what statements a person believes, we talk instead about what states of affairs, or possible worlds, are compatible with what he believes. Essentially, "A believes that P" is paraphrased as "P is true in every world that is compatible with what A believes." This can be expressed in ordinary first-order logic by making all predicates and functions depend explicitly on the particular possible world they are evaluated in. The failure of equality substitution in the preceding example is then accounted for by noting that what John believes depends on what is true in all possible worlds that are compatible with what he believes, but an assertion that the morning star and the evening star are the same is a statement about the actual world only. Application of this idea to reasoning about intensional operators in AI systems has been explored in depth by Moore [1980b].

Finally, a type of nonstandard logic that has received much recent attention is nonmonotonic logic. Minsky [1980, Appendix] has noted that the treatment of commonsense reasoning as purely deductive ignores one of its crucial aspects--the ability to retract a conclusion in the face of further evidence. A frequently cited example is that, if we know something is a bird, we normally assume it can fly. If we find out that it is an ostrich, however, we will withdraw that conclusion. This sort of reasoning is called "nonmonotonic" because the set of inferable conclusions does not increase monotonically with the set of premises as in conventional deductive logics. While many procedures have been implemented that support this type of reasoning, their theoretical

13

foundations are questionable. Most of the recent work on nonmonotonic logic [Bobrow, 1980] has thus been directed at developing a coherent logical basis for this kind of reasoning.

REFERENCES

Black, F. [1968] "A Deductive Question-Answering System," in Semantic Information Processing, M. Minsky, ed., pp. 354-402 (The MIT Press, Cambridge, Massachusetts, 1968).

Bobrow, D. G., ed. [1980] "Special Issue on Non-Monotonic Logic," Artificial Intelligence, Vol. 13, Nos. 1, 2 (April 1980).

de Kleer, J. et al. [1979] "Explicit Control of Reasoning," in Artificial Intelligence: An MIT Perspective, Vol. 1, P. H. Winston and R. H. Brown, eds., pp. 93-116 (The MIT Press, Cambridge, Massachusetts, 1979).

Doyle, J. [1979] "A Truth Maintenance System," Artificial Intelligence, Vol. 12, No. 3, pp. 231-272 (November 1979).

Doyle, J. [1980] "A Model for Deliberation, Action, and Introspection," MIT Artificial Intelligence Laboratory, AI-TR-581, Massachusetts Institute of Technology, Cambridge, Massachusetts (May 1980).

Gelernter, H. [1963] "Realization of a Geometry Theorem Proving Machine," in Computers and Thought, E. A. Feigenbaum and J. Feldman, eds., pp. 134-152 (McGraw Hill Book Company, New York, New York, 1963).

Green, C. [1969] "Theorem-Proving by Resolution as a Basis for Question-Answering Systems," in Machine Intelligence 4, B. Meltzer and D. Michie, eds., pp. 183-205 (American Elsevier Publishing Company, Inc., New York, New York, 1969).

Hayes, P. J. [1973] "Computation and Deduction," Proc. 2nd Symposium on Mathematical Foundations of Computer Science, Czechoslovak Academy of Sciences, pp. 105-116.

Hewitt, C. et al. [1973] "A Universal Modular ACTOR Formalism for Artificial Intelligence," Advance Papers of the Third International Conference on Artificial Intelligence, Stanford, California, pp. 235-245 (20-23 August 1973).

Hewitt, C. [1975] "How to Use What You Know," Advance Papers of the Fourth International Joint Conference on Artificial Intelligence, Tbilisi, Georgia, USSR, pp. 189-198 (3-8 September 1975).

Hintikka, J. [1971] "Semantics for Propositional Attitudes," in Reference and Modality, L. Linsky, ed., pp. 145-167 (Oxford University Press, London, England, 1971).

15

Huet, G. [1975] "A Unification Algorithm for Typed Lambda Calculus," _Theoretical Computer Science_, Vol. 1, No. 1, pp. 27-57 (January 1975).

Kowalski, R. [1974] "Predicate Logic as a Programming Language," in _Information Processing 74_, pp. 569-574 (North-Holland Publishing Company, Amsterdam, The Netherlands, 1974).

Kowalski, R. [1979] _Logic for Problem Solving_ (Elsevier North Holland, Inc., New York, New York, 1979).

Kripke, S. A. [1971] "Semantical Considerations on Modal Logic," in _Reference and Modality_, L. Linsky, ed., pp. 63-72 (Oxford University Press, London, England, 1971).

McCarthy, J. [1968] "Programs with Common Sense," in _Semantic Information Processing_, M. Minsky, ed., pp. 403-417 (The MIT Press, Cambridge, Massachusetts, 1968).

McCorduck, P. [1979] _Machines Who Think_ (W. H. Freeman and Company, San Francisco, Califonia, 1979).

McDermott, D. [1978] "Planning and Acting," _Cognitive Science_, Vol. 2, No. 2, pp. 71-109 (April-June 1978).

Minsky, M. [1980] "A Framework for Representing Knowledge," in _Mind Design: Philosophy, Psychology, and Artificial Intelligence_, J. Haugeland, ed. (Bradford Books, Publishers, Inc., Montgomery, Vermont, 1980).

Moore, R. C. [1980a] _Reasoning from Incomplete Knowledge in a Procedural Deduction System_ (Garland Publishing, Inc., New York, New York, 1980).

Moore, R. C. [1980b] "Reasoning About Knowledge and Action," Artificial Intelligence Center Technical Note 191, SRI International, Menlo Park, California (October 1980).

Newell, A. and Simon, H. A. [1956] "The Logic Theory Machine," _IRE Transactions on Information Theory_, Vol. IT-2, No. 3, pp. 61-79 (September 1956).

Nilsson, N. J. [1980] _Principles of Artificial Intelligence_ (Tioga Publishing Company, Palo Alto, California, 1980).

Robinson, J. A. [1965] "A Machine-Oriented Logic Based on the Resolution Principle," _Journal of the Association for Computing Machinery_, Vol. 12, No. 1, pp. 23-41 (January 1965).

Wang, H. [1960] "Toward Mechanical Mathematics," _IBM Journal of Research and Development_, Vol. 4, No. 1., pp. 2-22 (January 1960).

Warren, D. H. D. and Pereira, L. M. [1977] "PROLOG--The Language and its Implementation Compared with LISP," in Proceedings of the Symposium on Artificial Intelligence and Programming Languages (ACM); SIGPLAN Notices, Vol. 12, No. 8; and SIGART Newsletter, No. 64, pp 109-115 (August 1977).

Weyhrauch, R. W. [1980] "Prolegomena to a Theory of Mechanized Formal Reasoning," Artificial Intelligence, Vol. 13, Nos. 1, 2, pp. 133-170 (April 1980).