

CONDITIONAL ANSWERS IN QUESTION-ANSWERING SYSTEMS

Philip Klahr
The Rand Corporation
Santa Monica, California 90406

In many cases a deductive question-answering system cannot find complete proofs to answer questions requiring deductive support. In some cases information needed to complete proofs is missing from the knowledge base. In other cases processing limits may have been reached before proofs could be completed. Rather than disregarding such partial proofs as most systems do, the DADM system displays them to users and identifies subproblems that remain unresolved. Answers emanating from partial proofs include remaining subproblems as "conditions" which must be true for the answers to be valid.

1. INTRODUCTION

Most deductive question-answering systems output only complete proofs and answers in response to given queries. Partial proofs are usually ignored. Many partial proofs, however, are relevant to deducing an answer to a query but have not been completed because of missing information or because processing limits have been exceeded. We argue that such partial proofs and the resulting partial answers can be of significance to users for the following reasons:

1. Identify what deductions the system has discovered to show a user how the system interpreted his original query.
2. Allow a user to participate and aid in the construction of proofs by letting him examine proofs under development and allowing him to advise and select potentially fruitful deductive paths.
3. Identify incomplete knowledge about particular predicates either in the data base of facts or in the set of knowledge-based rules.
4. Identify what information is needed to complete existing partial proofs and to give complete answers.

In the DADM system [3, 4, 7], partial proofs are not ignored or disregarded because they

represent proofs in progress. If DADM is unable to find complete proofs within specified time limits, partial proofs are displayed along with their associated "conditional answers" (answers conditional on the truth of the remaining subproblems).

2. OVERVIEW OF DADM

DADM (Deductively Augmented Data Management) is a natural-deduction system (Bledsoe [1] reviews such systems) designed to interface with existing and emerging relational data base management systems. To facilitate this design criterion, there is a distinction and separation between rules and facts. Rules (axioms, theorems, rule-based knowledge) are in the form of predicate-calculus implications. Facts are single literals containing only constants (predicates whose arguments contain no variables), to be consistent with the form of facts typical in relational data bases.

Another design criterion is that the system should be efficient in dealing with large numbers of rules and facts. To this end, DADM uses PATHFINDER [5], a planning system designed to locate relevant rules before rules are actually applied in the course of constructing proofs. PATHFINDER uses the process of middle-term chaining to locate deductive implication chains by combining forward chaining from assumptions and backward chaining from goals. This process may be envisioned as one of generating expanding wavefronts in the two directions. The purpose of middle-term

chaining is not to construct proofs but to locate potentially relevant deductive implication chains through the rules.

Middle-term chaining does not operate directly on the rules. It uses a predicate connection graph [7] which contains information about the deductive connections (unifications) among the rules and implication connections within rules. (The connection graph is similar to other theorem-proving connection graphs, e.g. [8], although in DADM it is used as a planning tool within a natural-deduction system.) This graph is compiled when rules are first entered into the system. Thus, during proof planning and proof construction, the system has knowledge about all the deductive interactions among the rules and need not compute them dynamically.

Middle-term chains form the basis of the planning process designed to selectively focus on potentially relevant rules. The planning process forms skeleton proofs whose variable substitutions remain to be shown consistent throughout the proof. This latter task is the function of the verifier. Verification is delayed until the system has planned out potential proofs. The verification process examines the variable substitutions involved in the deductions (unifications) of a proof to test that no variable takes on conflicting values.

A middle-term chain represents a deductive implication chain through a sequence of rules. Using these rules, DADM creates a partial proof and determines if subproblems exist. Remaining subproblems are set up as goals to be resolved. (STRIPS [2] is another system that makes use of partial proofs. Unresolved subproblems are used to select relevant operators that allow the proof to be continued.) DADM has three methods available to resolve remaining subproblems: by deduction through the rules, by data-base search over the file of facts, or by computation if the subgoal predicate had been defined by a computational procedure. When subproblems cannot be resolved because of missing information or time constraints, partial proofs and conditional answers are displayed to the user.

3. CONDITIONAL ANSWERS

Consider the very simple example of trying to show that $A \Rightarrow D$. DADM begins by forming assumptions and goals from the initial query. In this example there is a single assumption

(A) and a single goal (D). Suppose the rules $A \& B \Rightarrow C$ and $C \Rightarrow D$ exist in the DADM's knowledge base. Using these two rules, DADM finds a middle-term implication chain from A to D through the middle term C (i.e., A to C to D). B is set up as a subproblem. If deductive processing stops here, DADM would display a partial proof and give the answer "yes, if B." (More detailed examples are given in [6].)

When are such conditional answers given? It must be emphasized that DADM tries to find complete proofs and answers. It will find and display complete proofs before giving the user any information about partial proofs. Also, DADM does not find just one proof. It will continue its deductive processing at the user's request or until DADM exhausts its processing limits.

If processing limits have been reached, DADM will display partial proofs under construction (as in the simple example above). The user may then request DADM to increase its processing time and direct DADM to continue working on particular partial proofs. This interactive process between the user and DADM allows a user to aid in the development and expansion of proofs by advising on directions for continued processing.

Partial proofs and conditional answers are also given in those cases where remaining subproblems cannot be resolved because of missing information. Information could be missing from the fact file (the set of facts in a real-world data base is typically incomplete) or from the rule set (DADM may not be able to prove or disprove remaining subproblems). Conditional answers serve to identify information that, if true, would complete partial proofs and provide complete answers. Such knowledge may also serve to suggest the insertion of new rules and facts.

An important concern in displaying partial proofs is deciding which proofs to display to a user, particularly when DADM has generated a large number of partial proofs. It is always the case that a partial proof is verified before it is displayed. The deductions in a proof must be consistent within one another in terms of the variable substitutions required. Those partial proofs that do not successfully verify are ignored and never shown. Any partial proof that does verify may be of potential importance to a user. The system should then order the display of partial proofs on the basis of relevance and importance.

Three main methods are used for ordering partial proofs. These methods are actually used during proof construction in the planning process rather than after proofs are formed. Thus the methods apply to constructing proofs in general, partial or otherwise. These methods involve the use of advice, plausibility, and the number of remaining subproblems.

DADM allows a user to give advice on the use of rules that he feels may be particularly appropriate for deducing an answer to a query. Furthermore, a user may advise the system to focus on particular predicates that may be appropriate middle-terms for chaining. (The user may also give negative advice, in which case specified rules and/or predicates will be avoided in proof generation.) In addition to problem-specific advice, a permanent advice file also exists for storing general domain advice which is accessed for each query [7].

Advice is transformed into rule and predicate alert lists which are used during middle-term chaining to order and prune the predicates and rules used. The system will try chaining through advised predicates and through advised rules whenever it can. Advice thus serves as a focus of attention mechanism for the chaining and planning processes. Proofs using advised rules and predicates will be generated and displayed first.

The use of plausibility measures serves a similar focusing function. Rules have plausibility measures associated with them. During middle-term chaining, rules are ordered according to their plausibilities. Resulting proofs will be generated and displayed on the basis of the plausibility of the rules used in the proof. Thus most plausible proofs will be displayed first. (Note that advice is given highest priority in the generation of proofs.)

The third method for ordering proofs concerns the number of remaining subgoals in the proof. This is particularly important for ordering the display of partial proofs. Since unresolved subgoals result in conditionals in the partial answer, the fewer the number of such conditionals, the more valuable a partial proof and partial answer will be to a user. Partial answers are conditional on the missing information being true. The greater the number of conditionals, the less likely that all of them would be true. Thus partial proofs are ordered according to the fewest number of remaining subgoals.

DADM has been designed to be transparent and user-oriented. It tries to give the user as much information as it can or as much as is desired. DADM's ability to recognize and report partial proofs and conditional answers has enhanced the system's usability, particularly over incomplete knowledge bases (where partial proofs and answers become more significant) and over large knowledge bases (where constraints on processing time may preclude the discovery of complete proofs).

Acknowledgments

Charles Kellogg and Larry Travis have been instrumental in the design of the DADM system throughout its development. They have contributed substantially to the ideas presented here.

REFERENCES

- [1] Bledsoe, W. W. "Non-resolution theorem proving." Artificial Intelligence, 9 (1977) 1-35.
- [2] Fikes, R. E. and Nilsson, N. J. "STRIPS: a new approach to the application of theorem proving to problem solving." Artificial Intelligence, 2 (1971) 189-208.
- [3] Kellogg, C., Klahr, P., and Travis, L. "Deductive methods for large data bases." Proc. IJCAI-77, MIT, Cambridge, Mass., 1977, 203-209.
- [4] Kellogg, C., Klahr, P., and Travis, L. "Deductive planning and pathfinding for relational data bases." In Logic and Data Bases, H. Gallaire and J. Minker (Eds.), New York: Plenum, 1978, 179-200.
- [5] Klahr, P. "Planning techniques for rule selection in deductive question-answering." In Pattern-Directed Inference Systems, D. A. Waterman and F. Hayes-Roth (Eds.), New York: Academic Press, 1978, 223-239.
- [6] Klahr, P. "Partial proofs and partial answers." Technical paper P-6239, The Rand Corporation, Santa Monica, Calif., 1978.
- [7] Klahr, P., Travis, L., and Kellogg, C. "A deductive system for natural-language question answering." To appear in Natural Language Based Computer Systems, L. Bolc (Ed.).
- [8] Sickel, S. "A search technique for clause interconnectivity graphs." IEEE Transactions on Computers, C-25 (1976) 823-835.