DEDUCTIVE QUESTION-ANSWERING ON

RELATIONAL DATA BASES

Raymond Reiter
The University of British Columbia
Vancouver, British Columbia

ABSTRACT

The principal concern of this paper is the design of a retrieval system which combines current techniques for query evaluation on relational data bases with a deductive component in such a way that the interface between the two is both clean and natural. The result is an approach to deductive retrieval which appears to be feasible for data bases with very large extensions (i.e. specific facts) and comparatively small intensions (i.e. general facts). More specifically, a suitably designed theorem prover "sweeps through" the intensional data base, extracting all information relevant to a given query. This theorem prover never looks at the extensional data base. The end result of this sweep is a set of queries, each of which is extensionally evaluated. The union of answers returned from each of these queries is the set of answers to the original query.

One consequence of this decomposition into an intensional and extensional processor is that the latter may be realized by a conventional data base management system. Another is that the intensional data base can be compiled using a theorem prover as a once-only compiler.

This paper is essentially an impressionistic survey of some results which are rigorously treated elsewhere. As such, no proofs are given for the theorems stated, and the basic system design is illustrated by means of an extended example.
The principal concern of this paper is the design of a retrieval system which combines current techniques for query evaluation on relational data bases, e.g. Codd [1972] with a deductive component in such a way that the interface between the two is both clean and natural. The result is an approach to deductive retrieval which appears to be feasible for data bases with very large extensions (i.e. specific facts) and comparatively small intensions (i.e. general facts). More specifically, a suitably designed theorem prover "sweeps through" the intensional data base, extracting all information relevant to a given query. In particular, this theorem prover never looks at the extensional data base. The end result of this sweep is a set of queries, each of which is extensionally evaluated. The union of answers returned from each of these queries is a set of answers to the original query.

There are two important consequences of this decomposition of the question-answering task into a theorem prover computing on the intensional data base, and an extensional processor computing on the extensional data base:

1. The extensional processor can be realized by a conventional data base management system.

2. Because the theorem prover never accesses the extensional data base, the intensional data base can be compiled using the theorem prover as a once-only compiler. This means that at query evaluation time there is no need for a theorem prover, nor are there the usual problems involving search which can plague a theorem proving system.

This paper is essentially a survey of some of the results in Reiter [1977]. As such, it is necessarily impressionistic, so that no proofs are given for the theorems stated, and the basic approach to query evaluation which decouples the theorem prover from the extensional processor is described by means of an extended example. A rigorous presentation is contained in Reiter [1977] to which the interested reader is referred for the painful details.

DATA BASES

The results of this paper apply only to first order data bases with the following properties:

(i) The data base consists of finitely many twffs (typed well-formed formulae). For example, in an inventory domain, such a twff might be

\[(x/\text{Manufacturer})(y/\text{Part})(z/\text{Part}) \text{manufactures } x,y \land \text{Subpart } z,y \supset \text{Supplies } x,z\]