The Database Approach to Knowledge Representation

Jeffrey D. Ullman

Department of Computer Science
Stanford University
Stanford CA 94305
ullman@cs.stanford.edu
http://db.stanford.edu/~ullman

Abstract

The database theory community, centered around the PODS (Principles of Database Systems) conference has had a long-term interest in logic as a way to represent "data," "information," and "knowledge" (take your pick on the term — it boils down to facts or atoms and rules, usually Horn clauses). The approach of this community has been "slow and steady," preferring to build up carefully from simple special cases to more general ideas, always paying attention to how efficiently we can process queries and perform other operations on the facts and rules. A powerful theory has developed, and it is beginning to have some impact on applications, especially information-integration engines.

Datalog

The term *Datalog* has been coined to refer to Prolog-like rules without function symbols, treated as a logic program. Unlike Prolog, however, the conventional least-fixed-point semantics of the rules is used whenever possible.

Example 1: The rules for ancestors can be written as the following Datalog program.

```
anc(X,Y) :- par(X,Y)
anc(X,Y) :- anc(X,Z), anc(Z,Y)
```

That is, Y is an ancestor of X if Y is a parent of X or if there is some Z that is an ancestor of X and a descendant of Y. Because of the least-fixed-point semantics, there is no question of this program entering a loop, as the corresponding Prolog program would. □

Generally, we divide predicates into two classes. EDB (*extensional database*) predicates are stored as relations, while IDB (*intensional database*) predicates are defined by the *heads* (left sides) of rules only. Either EDB or IDB predicates can appear in subgoals of the *bodies* (right sides) of rules.

Sometimes, Datalog is extended to allow negated subgoals. That extension causes the least-fixed-point semantics to become problematic when the rules are recursive, and several approaches such as stratified negation and well-founded semantics have been developed to define suitable meanings for such Datalog programs. A survey of this subject, analogous to "nonmonotonic reasoning," can be found in Ullman [1994], and we shall not address this set of issues further here.

Example 2: A Datalog rule for ancestors that were not parents could be expressed as

```
oldAnc(X,Y) :- anc(X,Y), NOT par(X,Y)
```

□

Conjunctive Queries

A single Datalog rule in which an IDB predicate is defined in terms of one or more IDB and EDB predicates other than itself is called a *conjunctive query* (CQ). For instance, the rule of Example 2 is a CQ.

Containment and Equivalence

We say one CQ or Datalog program is *contained* in another if whatever the values of the EDB predicates (the "database") is, the set of facts provable from the first is a subset of those provable from the second. CQ's or Datalog programs are *equivalent* if the sets of provable facts are always the same for any database, i.e., the containment goes both ways.

Example 3: Consider

```
Q1: p(X) :- arc(X,Y), arc(Y,X)
Q2: p(X) :- arc(X,X)
```

We say that $Q_2 \subseteq Q_1$. Intuitively, Q_1 defines the set of nodes of a directed graph that are on any cycle of two nodes or loop of one node, while Q_2 defines only the set of nodes that have loops. It is also easy to check that Q_1 is not contained in Q_2; that is, there are graphs with cycles of two nodes but no loops. Thus, $Q_1 \neq Q_2$. □

Chandra and Merlin [1977] first studied conjunctive queries and showed that there is a simple test for containment, and thus for equivalence. The question of whether one CQ is contained in another is NP-complete,

but all the complexity is caused by "repeated predicates," that is, predicates appearing three or more times in the body. In the very common case that no predicate appears more than twice in any one query, containment can be tested in linear time (Saraiya [1991]). Moreover, CQ's tend to be short, so in practice containment testing is not likely to be too inefficient.

Also, in exponential time at most we can test whether a CQ is contained in a Datalog program (Ramakrishnan et al. [1989]). The opposite containment, of a Datalog program in a CQ is harder, but still decidable (Chaudhuri and Vardi [1992]).

When we extend rules by allowing negated subgoals and/or by allowing arithmetic comparisons in the subgoals, things rapidly become undecidable. However, if we restrict ourselves to CQ's, not Datalog programs, then the problem is exponential at worst. Levy and Sagiv [1993] give a containment test for CQ's with negation, while the most efficient known algorithm for CQ's with arithmetic comparisons (but no negation) is found in Zhang and Ozsoyoglu [1993].

Application to Information Integration

A system for integrating heterogeneous information sources can be described logically by *views* that tell us what queries the various sources can answer. These views might be CQ's or Datalog programs, for example. The "database" of EDB predicates over which these views are defined is not a concrete database but rather a collection of "global" predicates whose actual values are determined by the sources, via the views.

Given a query Q, typically a CQ, one can ask whether it is possible to answer Q by using the various views in some combination. For example, *Information Manifold* at Bell Labs (Levy, Rajaraman, and Ordille [1996]) searches for all combinations of views that answer a query, while *Tsimmis* at Stanford (Garcia et al. [1995]) tries to find one such solution.

Example 4: The following example is contrived but will illustrate the ideas. Suppose we have a global parent relation $par(X, Y)$, meaning that Y is a parent of X. Suppose also that one source is capable only of providing a grandparent view. That is, it gives us the view:

```
gp(X,Y) :- par(X,Z), par(Z,Y)
```

A second source gives us a great-grandparent view:

```
ggp(X,Y) :- par(X,A), par(A,B), par(B,Y)
```

Our query Q is "find the first cousins once removed of individual a. That is, we must go up two generations from a, then down three generations. Formally, in terms of the global "database":

```
fcor(X) :- par(a,A), par(A,B),
    par(C,B), par(D,C), par(X,D)
```

We can answer Q in terms of the given views by:

```
fcor(X) :- gp(a,B), ggp(X,B)
```

□

Solving Queries in Terms of Views

The fundamental work on how to find an expression for a given query in terms of views is Levy, Mendelzon, Sagiv, and Srivastava [1994]. This paper handles the case where both the views and the query are CQ's.

Rajaraman, Sagiv, and Ullman [1994] extends the latter to the case where the views have "binding patterns"; that is, the source can only answer a query in which one or more arguments are bound. An example of this situation is a bibliographic source that can find a book given an author or an author given a book, but cannot answer the query "tell be about all books and their authors." In Example 4, there is a solution only if the source of view *gp* can handle a query with the first argument bound, and the source of *ggp* can handle a query with the second argument bound.

There has also been some progress allowing the views to be described by Datalog programs rather than CQ's. Specifically, each Datalog program can be expanded into a (possibly infinite) set of CQ's, and we may suppose that a source will answer any one of these CQ's. That model covers the case where the source is an SQL database, for instance.

In Papakonstantinou et al. [1995], the test for containment of a CQ in a Datalog program is exploited to find an expansion of the Datalog program that contains a given query. Levy, Rajaraman, and Ullman [1996] show how to decide equivalence of a query to some expression built from (a finite subset of) the infinite set of views that are the expansions of a Datalog program. This result extends Levy, Mendelzon et al. [1995] to the situation where sources support infinite sets of views that are described by a Datalog program.

Acknowledgements

This work was supported by NSF grant IRI–92–23405, ARO grant DAAH04–95–1–0192, and USAF contract F33615–93–1–1339.

References

Chandra, A. K. and P. M. Merlin [1977]. "Optimal implementation of conjunctive queries in relational databases," *Proc. Ninth Annual ACM Symposium on the Theory of Computing*, pp. 77–90.

Chaudhuri, S. and M. Y. Vardi [1992]. "On the equivalence of datalog programs," *Proc. Eleventh ACM Symposium on Principles of Database Systems*, pp. 55–66.

Garcia-Molina, H., Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, and J. Widom [1995]. "The TSIMMIS approach to mediation: data models and languages," Second Workshop on Next-Generation Information Technologies and Systems, Naharia, Israel, June, 1995.

Levy, A., A. Mendelzon, Y. Sagiv, and D. Srivastava [1995]. "Answering queries using views," *Proc. Fourteenth ACM Symposium on Principles of Database Systems*, pp. 113–124.

Levy, A. Y., A. Rajaraman, and J. J. Ordille [1996]. "Querying heterogeneous information sources using source descriptions," ATT Technical Memorandum, submitted for publication.

Levy, A. Y., A. Rajaraman, and J. D. Ullman [1996]. "Answering queries using limited external processors," to appear in *PODS* 1996.

Levy, A. Y. and Y. Sagiv [1993]. "Queries independent of update," *Proc. International Conference on Very Large Data Bases*, pp. 171–181.

Papakonstantinou, Y., A. Gupta, H. Garcia-Molina, and J. D. Ullman [1995]. "A query translation scheme for rapid implementation of wrappers," Fourth *DOOD*, Singapore, Dec., 1995.

Rajaraman, A., Y. Sagiv, and J. D. Ullman [1995]. "Query optimization using templates with binding patterns," *Proc. Fourteenth ACM Symposium on Principles of Database Systems*, pp. 105–112.

Ramakrishnan, R., Y. Sagiv, J. D. Ullman, and M. Y. Vardi [1989]. "Proof tree transformation theorems and their applications," *Proc. Eighth ACM Symposium on Principles of Database Systems*, pp. 172–181.

Saraiya, Y. [1991]. "Subtree elimination algorithms in deductive databases," Doctoral Thesis, Dept. of CS, Stanford Univ., Jan., 1991.

Ullman, J. D. [1994]. "Assigning an appropriate meaning to database logic with negation," in *Computers as Our Better Partners* (H. Yamada, Y. Kambayashi, and S. Ohta, eds.), pp. 216–225, World Scientific Press.

Zhang, X. and M. Z. Ozsoyoglu [1993]. "On efficient reasoning with implication constraints," *Proc. Third DOOD Conference*, pp. 236–252.