Recursively Indefinite Databases
(Extended Abstract)

R. van der Meyden

Dept. Computer Science, Rutgers University,
New Brunswick N.J. 08903, U.S.A.

Abstract: We define recursively indefinite databases, a new type of logical database in which indefinite information arises from partial knowledge of the fixpoint of a datalog program. Although, in general, query answering is undecidable, there exists a broad class of queries for which it is decidable, a result we establish by making connections with the theory of hypergraph edge replacement graph grammars. We analyze the complexity of query answering for this class of queries under various constraints and demonstrate a class of databases which generalizes disjunctive databases, but without increasing data complexity.

Section 1 Introduction

Logical databases permit the representation of several types of indefinite information (the term 'incomplete' is often applied to this sort of information). If in addition to ground atomic facts we allow existentially quantified conjunctions of atoms we obtain a class of logical databases which can represent the null values familiar from relational databases [21]. Another extension is obtained by allowing various sorts of disjunctive formulae in the database [8, 9]. In this paper we introduce recursively indefinite logical databases, which generalize both of these sorts of databases.

Example 1: Let G be a directed graph some of whose nodes are colored red or green. A “red-green path” is any path through G each of whose nodes is colored either red or green. The following datalog program defines the relation \(rgpath(x, y)\): “there exists a red-green path from \(x\) to \(y\)”

\[
\begin{align*}
rg(x) : & \neg\text{red}(x). \\
rg(x) : & \neg\text{green}(x). \\
rgpath(x, y) : & \neg\text{rg}(x), \text{rg}(y), \text{edge}(x, y). \\
rgpath(x, y) : & \neg\text{rgpath}(x, x), \text{rgpath}(x, y).
\end{align*}
\]

Suppose we know that there is a red-green path between nodes \(a\) and \(b\) and also that \(a\) is colored red and \(b\) is colored green. Even if this is all we know about the graph, we are already able to conclude that the graph contains an edge from a red node to a green node. That is, on the basis of the information
red(a), green(b), rgpath(a, b), the query $\exists x, y (\text{red}(x) \land \text{green}(y) \land \text{edge}(x, y))$ should be answered "true". Notice that information about the intensional database predicate rgpath gives us additional information about the extensional predicates red, green, edge.

Example 2: Two nodes a and c in a graph are "equidistant" to a node b if there exists a path p from a to b and a path q from b to c such that the lengths of p and q are equal. The nodes a and c are equidistant to b just when the atom $eq(a, b, c)$ is derived by the datalog program

$$eq(x, y, z) : -\text{edge}(z, y), \text{edge}(y, z)$$

$$eq(x, y, z) : -\text{edge}(z, z'), eq(z', y, z'), \text{edge}(z', z).$$

If we know $eq(a, b, c)$ and $eq(c, d, f)$ then the query $\exists z(eq(b, c, z) \lor eq(z, c, d))$ should be answered "true". For, suppose that we have paths of length n from a to b and from b to c, as well as paths of length m from c to d and from d to f, respectively. If $n < m$ then $eq(b, c, z)$ for some z along the path from c to d. The converse relation gives the other disjunct.

The sort of indefinite information we are contemplating here is given in two components: a datalog program and sets of "known" atoms in each of the intensional and extensional predicates. The extensional predicates are interpreted as usual, under an "open world assumption". That is, we do not assume that the extensional relations are completely known; rather, the database tells us only some of the facts true in the actual state of the world. The intensional predicates are assumed to be calculated from the extensional predicates using the datalog program. Applying the program to the set of known base atoms, we obtain a set of intensional atoms which must be true. However, since there may be other base atoms true, this set is not assumed to be the complete set of such facts. In particular, it need not include the known intensional atoms. Thus the known intensional atoms provide us with additional information about the extensional predicates.

The queries we consider will be first order positive existential formulae. If a query contains occurrences of intensional predicates, it will be called intensional. Base queries will contain just base predicates. The class of base queries is very natural if one takes the point of view that the role of the intensional predicates is to provide a representational mechanism for indefinite information about the base predicates. We will show that the problem of answering arbitrary intensional queries of arbitrary recursively indefinite databases is undecidable. However, if we restrict to base queries, this problem becomes decidable, a result that will be shown to follow from a general result of Courcelle [4] concerning context-free graph grammars. Our main contribution is to analyze the exact complexity of various decision problems. We also show that Courcelle's result may be used to overcome the general undecidability result for suitably constrained intensional queries.

One way recursively indefinite information arises is from updates to recursively defined views. Each intensional predicate of the database can be interpreted as a view of the database defined by the rules of the program. If updates are assumed to add information to the database only, i.e. only updates completing the information in the database are permitted, not revisions or deletions, then the incomplete