On Negative Information in Deductive Databases

Marek A. Suchenek
The Wichita State University

Rajshekhar Sunderraman
The Wichita State University

An overview of deductive databases with an emphasis on problems related to negative information is presented. The subject is embedded into the wider context of logic programming, exposing certain peculiarities pertinent to treatment of negation in this field.

Introduction

In this paper, we discuss problems related to representing and handling negative information in deductive databases. The subject, albeit over ten years old, has not lost anything from its appeal and momentum. To the contrary, it appears to gain continuously even more attention from the database community. It may be embedded in the context of another, nowadays fashionable topic known under the name of non-monotonic logic.

The deductive model of a database emerged from its predecessor, the relational model, by incorporating certain elements and methods of automated theorem proving. In the relational model of a database facts are organized in a finite collection of relations with the standard operations on these relations such as: insertion, deletion, selection, projection, product, join, union, intersection, and difference. Every relation is represented by a finite set of tuples. Database users access the information stored in the database by means of queries expressed in a suitable language. The database management system plays the role of an interface between the user and the physical database, and is responsible for accepting the user’s queries and operations as well as returning answers to the user. Moreover, it ensures that the integrity constraints pertinent to the database are not violated. Although this last task involves certain elements of reasoning, because the integrity constraints are usually expressed in the form of rules, the relational database, or rather its management system, does not go much beyond passive verification of the legality of the performed operations against the coded integrity constraints. Another feature of relational databases that requires some form of reasoning is the concept of view, which may be interpreted as rule based definitions of relations not explicitly represented in a physical database. However, views are usually limited to certain non-recursive rules and are intended to hide information rather than to equip a database with deductive abilities.

The seemingly natural observation, that an active use of rules which are present in a database may result in the derivation of new facts, not explicitly represented in it, gave rise to deductive databases. Conceptually, a deductive database may be seen as a relational database furnished with additional, “intelligent” layers capable of complex reasoning from the content of the relational component. In fact, this point of view seems to prevail. One should note, however, that since contemporary rela-
tional databases are not purely passive in executing their integrity constraints, the distinction between relational and deductive databases is rather fuzzy, and one certainly can find a variety of cases which may be classified “relational” as well as “deductive”. On the other hand, their different forms of representation require, for the purpose of making a deductive database out of a relational one, a suitable translation process. It turns out that this process is largely responsible for the need for non-standard treatment of negation in processing queries against deductive databases.

If one translates the content of a purely relational database into its deductive counterpart, the problem with the interpretation of negative information manifests itself. Certainly, while translating tuples onto predicates, or more precisely, onto ground literals, only the positive information of the relational part is explicitly processed, despite the fact, that the relational part, by the absence of some tuples, may encode negative information as well. The first systematic treatment of this phenomenon by means of closed world assumption (cwa) may be found in (Reiter, 1978b), and since then, a mature age of deductive databases seems to have begun.

The closed world assumption may be recognized as a benchmark which makes a difference between relational and deductive databases, since in the relational model the closed world assumption is somewhat hard-wired, and is therefore superfluous, while deductive databases do need cwa to allow negative conclusions from purely positive information. Under cwa, all simple positive facts that cannot be derived from the database are assumed to be false, or in other words, their negations are asserted. This straightforward and seemingly unquestionable rule creates serious problems, however, when mechanically applied to deductive databases which are not the result of translating the content of a relational database. That is why a proper treatment of negation is so crucial in this context.

In the present understanding of the term, deductive databases consist of the following syntactic elements:

1. Extensional part, which corresponds to the relations, and usually has the form of a set of so called ground positive literals,
2. Intensional part, which corresponds to view definitions, integrity constraints, and other dependencies, and usually has the form of a set of so called clauses,
3. Resolution theorem prover, which derives logical consequences of the extensional and intensional content of the database, and
4. Meta-rules for negative inferences, e.g., the closed world assumption.

In this paper, after a general introduction to deductive databases, we focus on the last element of this list, discussing how they affect the entire database and its information content, and what meaning may be consistently associated with such rules. Because the area of deductive databases is not the only one concerned with proper treatment of negation, we also briefly discuss how the problem is approached in logic programming. We conclude with a succinct history of developments in this area, mentioning current experimental research.

Preliminaries

Deductive databases have the form of a collection of clauses expressible in a language of first-order logic. In this section, we provide a concise introduction to first-order clause logic, and its minimal model semantics.

Syntax of First-Order Logic. Logic is a branch of mathematics which investigates infallible rules of reasoning. Propositional logic is, perhaps, the simplest form of logic which deals with propositional statements, such as “It is sunny today” or “It is not raining today”, and reasoning within this context. Propositional symbols together with the falsehood symbol \( \square \) are used to represent statements, and may incorporate logical connectives \( \land \) (and), \( \lor \) (or), \( \rightarrow \) (implies), and \( \neg \) (not) to combine simple statements into more complex ones. For example, if \( P \) denotes “It is sunny today”, and \( Q \) denotes “It is not raining today”, then \( P \rightarrow Q \) denotes “If it is sunny today, then it is not raining today”. Other customarily used connectives may be understood as abbreviations. In particular, the equivalence \( P \equiv Q \) stands for the two way implication \( (P \rightarrow Q) \land (Q \rightarrow P) \).

Despite its elegant simplicity, propositional logic is neither particularly suitable to proper treatment of statements of the form: “All men are mortal”, “Socrates is a man”, nor to reason within this context. For example, it is not possible to conclude that “Socrates is mortal” from these statements using only propositional axioms and modus ponens. This kind of limitation of propositional provability does not appear in first-order logic, which is a more powerful form of logic capable of formally expressing such statements and reasoning with them. In addition to logical connectives, first-order logic allows quantifiers \( \forall \) (for all) and \( \exists \) (there exists) ranging over individual variables appearing in statements. Moreover, it contains more axioms and an extra rule of inference. For example, given \( (\forall x) (\text{Man}(x) \rightarrow \text{Mortal}(x)) \), which formally expresses the statement “All men are mortal”, and \( \text{Man}(\text{Socrates}) \), which articulates the statement “Socrates is a man”, one can conclude \( \text{Mortal}(\text{Socrates}) \), which denotes “Socrates is mortal”.

Since part of first-order logic provides a framework for deductive databases, we will take a closer look at
Related Content

Flexible Querying Techniques Based on CBR
[www.igi-global.com/chapter/flexible-querying-techniques-based-cbr/20353?camid=4v1a](www.igi-global.com/chapter/flexible-querying-techniques-based-cbr/20353?camid=4v1a)

Assigning Ontological Meaning to Workflow Nets
Pnina Soffer, Maya Kaner and Yair Wand (2010). *Journal of Database Management* (pp. 1-35).
[www.igi-global.com/article/assigning-ontological-meaning-workflow-nets/43728?camid=4v1a](www.igi-global.com/article/assigning-ontological-meaning-workflow-nets/43728?camid=4v1a)

Adoption, Improvement, and Disruption: Predicting the Impact of Open Source Applications in Enterprise Software Markets
[www.igi-global.com/chapter/adoption-improvement-disruption/8052?camid=4v1a](www.igi-global.com/chapter/adoption-improvement-disruption/8052?camid=4v1a)

Evolutionary Database: State of the Art and Issues
[www.igi-global.com/chapter/evolutionary-database-state-art-issues/20693?camid=4v1a](www.igi-global.com/chapter/evolutionary-database-state-art-issues/20693?camid=4v1a)