GLOBAL DECISION PROBLEMS FOR RELATIONAL DATABASES

M. Y. Vardi

Department of Computer Science
Hebrew University of Jerusalem
Jerusalem 91904, ISRAEL

ABSTRACT

Database dependencies are first-order sentences describing the semantics of databases. Decision problems concerning dependencies divide into local problem, such as whether a set of dependencies logically implies another dependency, and global problems, such as whether a set of dependencies is redundant. In this paper we investigate global problems, that of recognizing properties of sets of dependencies. The main result is a negative result in the spirit of Adjan-Markov-Rabin result for global properties of finitely presented semigroups and groups. We show that the decision problem for any property which is well-behaved in a certain sense (specifically, if it is nice, non-trivial and hereditary) is recursively unsolvable.

1. Introduction

In the relational model one views the database as a collection of relations, each relation being a set of tuples over some domain of values [Codd1]. One of the notable features of this model is its being almost devoid of semantics. A tuple in a relation represents a relationship between certain values, but from the mere syntactic definition of the relation one knows nothing about the nature of this relationship, not even if it is a relationship between two entities or between an entity and its attributes.

Two approaches have been taken to remedy this deficiency. The first approach is to extend the relational model to capture more semantics [Codd3]. The second approach, which is the subject matter of this paper, is to devise means to specify the missing semantics. These semantic specfications are often called semantic or integrity constraints, since they specify which databases are meaningful for the application under consideration and which are meaningless. Thus, the database schema is conceived as syntactic specifications accompanied by semantic specifications.

Several approaches have been taken with regard to integrity constraints. Of particular interest are the constraints called data dependencies, or dependencies for short. Dependencies are sentences in first-order logic stating, intuitively, that if some tuples, fulfilling certain conditions, exist in the database then either some other tuples must also exist in the database or some values in the given tuples must be equal. The study of dependencies began with the functional dependencies of [Codd2]. After the introduction of multivalued dependencies by [Fag1, Zan] the field became chaotic for a few years, in which researchers introduced many new classes of dependencies. Only recently has a unified framework been suggested by [BV1,Fag2]. The class of tuple and equality generating dependencies seems to contain all cases of interest.

Most of the papers in dependency theory deal exclusively with various aspects of the implication problem, i.e., the problem of deciding for a given set of dependencies D and a dependency d whether D semantically implies d. The reason for the prominence of this problem is that an algorithm for deciding implication of dependencies enables us to decide whether two given sets of dependencies are equivalent or whether a given set of dependencies is redundant. A solution for the last two problems seems a significant step towards automated database schema design ([Bern,BMSU,BR]), which some researchers ([BBG]) see as the ultimate goal for research in dependency theory. Unfortunately, the implication problem was shown to be recursively unsolvable in general, and in less general cases where it is solvable it is computationally intractable [BV1,BV2,CLM].

We view decision problems dealing with individual dependencies as local problems and decision problems dealing with properties of sets of dependencies as a whole as global problems. This distinction is analogous to that due to [Bo] regarding decision problems for finitely presented groups. The problem of whether two words in a finitely presented group denote the same element is local, while the problem of whether the group is free or simple is global. Analogously, the implication problem for dependencies is local, while the redundancy problem is global. Other examples of global problems are: does a set of dependencies have a non-trivial model, or is the implication problem for a set of dependencies solvable.

The interest in global decision problems for dependencies is not merely theoretical. As with all formal specifications of intuitive notions, the task of verifying the consistency and the correctness of the semantic specifications is

extremely important and difficult. The first interface in the ANSI/X3/SPARC architecture is the interface between the enterprize administrator and the conceptual database schema processor whose task is to verify the consistency of the database schema [ANS]. The difficulty in verifying the correctness of the specification was discussed by [Beer] for multivalued dependencies, and few meagre aids were suggested ([Luk,SM]), but by and large the problem was all but neglected. It is our opinion that the ability to recognize global properties is essential in the task of verifying the consistency and correctness of the semantic specifications. If, for example, the designer is informed that his set of dependencies has no non-trivial model then he can be quite sure that something is "fishy" with this set.

Unfortunately, the results of this paper are mainly negative. We show that the formalism of dependencies is so expressive that the decision problem for any global property which is well-behaved in a certain sense is recursively unsolvable. The situation is again very similar to that of global group-theoretic decision problems, where any property satisfying a certain condition called the Markov condition fails to be recursive. This result was obtained by [Adj,Ra] following an analogous result for semigroups [Mar]. Thus, our result answers in a single stroke most questions of interest in the area. In view of the negative results for the implication problem, this renders almost hopeless the prospect of fully automated database schema design. We will return to this point at the end of the paper.

2. Basic Definition

2.1. Dependencies

We use a first-order language L(n) with no function symbols, with equality and with one n-ary predicate symbol. Indexed x's are used for existentially quantified variables symbols, and indexed y's for universally quantified variable symbols. A dependency is a sentence

$$\forall y_1 \ldots \forall y_k \exists x_1 \ldots \exists x_m (A_1 \wedge \ldots \wedge A_p \rightarrow B),$$

where:

a) $k,p \geq 1$, $m \geq 0$.

b) The A's are atomic formulas that are not equalities using exactly the variables symbols $y_1,\ldots,y_k$.

c) Either B is an atomic formula that is not an equality using all the variables symbols $x_1,\ldots,x_m$ and possibly some y's, or B is an equality $y_i = y_j$, for some $1 \leq i, j \leq k$, and $m = 0$.

Remark. Actually, the class of dependencies defined in [BV1,Fag2] is more general than the class defined here.

We leave the parameter n unspecified. All our results hold for some fixed n. We use D to denote finite sets of dependencies and d,d' to denote single dependencies.

2.2. Relations

An n-ary relation is a non-empty subset of N^n. (N is the set of natural numbers). A relation R satisfies a dependency d if the structure <N,R> satisfies d. R satisfies a set D if it satisfies every dependency in D. SAT(D) is the set of relations satisfying D, and FSAT(D) is the set of finite relations satisfying D. DEP(R) is the set of dependencies satisfied by R, i.e., DEP(R) = {d: R satisfies d}.

Let R be a relation, and let $1 \leq i_1 < \ldots < i_k \leq n$, $1 \leq k \leq n$. The projection of R on the coordinates $i_1,\ldots,i_k$ is:

$$P_{<i_1,\ldots,i_k>}(R) =$$

$$= \{<a_{i_1},\ldots,a_{i_k}>: <a_1,\ldots,a_n> \in R\}.$$

A relation R is trivial if $R = \{<a,a,\ldots,a>\}$ for some a in N. Observe that if R is trivial then R is in FSAT(D) and in SAT(D) for every set D. That is, every set D is satisfied by the trivial relations.

A set D implies a dependency d, denoted $D \models d$, if $SAT(D) \subseteq SAT(d)$, and it finitely implies d, denoted $D \models_f d$, if $FSAT(D) \subseteq FSAT(d)$. The set of dependencies implied by D is $IMPL(D) = \{d: D \models d\}$, and the set of dependencies finitely implied by D is $FIMPL(D) = \{d: D \models_f d\}$.

2.3. Properties of Sets of Dependencies

A property P is a set of finite sets of dependencies. We usually say that D is P instead of saying that D is in P. P is decidable if it is recursive. P is non-trivial if there is a set D which is not P. We give now several examples of properties. Almost every property has two "versions"; the second "version" is obtained by restricting attention to finite relations.

D is trivial if all relations in SAT(D) are trivial. It is finitely trivial if all relations in FSAT(D) are trivial.

D is redundant if there is some d in D such that $D - \{d\} \models d$. It is finitely redundant if there is some d in D such that $D - \{d\} \models_f d$.

D is decidable if IMPL(D) is recursive. It is finitely decidable if FIMPL(D) is recursive. It is finitely implying if IMPL(D) = FIMPL(D). If D is finitely implying then D is decidable and finitely decidable [BV1].

D is Armstrong [Fag2] if there is a relation R such that DEP(R) = IMPL(D). It is finitely Armstrong if there is a finite relation R such that DEP(R) = FIMPL(D). R is called a (finite) Armstrong relation for D.

D is complete if for all d not in IMPL(D), D ∪ {d} is trivial. It is finitely complete if for all d not in FIMPL(D), D ∪ {d} is finitely trivial. If D is complete then every nontrivial relation in SAT(D) is an Armstrong relation for D. Similarly, if D is finitely complete then every nontrival relation in FSAT(D) is a finite Armstrong relation for D. Also, if D is complete then it is decidable, and if D is finitely complete then it is finitely decidable.

From the point of view of the database designer, some of these properties are desirable (e.g., decidability, Armstrongness), some are non-desirable (e.g., redundancy), and some indicate that his specifications are probably incorrect (e.g., triviality).

A key observation is that triviality implies most of the other properties. A trivial set D is decidable, finitely decidable, Armstrong, etc. Similarly, a finitely trivial set is finitely decidable, finitely Armstrong, etc. We say that a property P is nice if every trivial set D is P, and is finitely nice if every finitely trivial set D is P.

3. Decidability of Properties

Let D be a set of n-ary dependencies, and let D' be a set of k-ary dependencies, $1 \le k \le n$. We say that D' is a projection of D if there is a sequence of coordinates $1 \le i_1 < \ldots < i_k \le n$ such that if R is in SAT(D) then $P_{<i_1,\ldots,i_k>}(R)$ is in SAT(D'), and if R' is in SAT(D') then $R' = P_{<i_1,\ldots,i_k>}(R)$ for some R in SAT(D).

Similarly, D' is a finite projection of D if there is a sequence of coordinates $1 \le i_1 \le \ldots \le i_k \le n$ such that if R is in FSAT(D) then $P_{<i_1,\ldots,i_k>}(R)$ is in FSAT(D'), and if R' is in FSAT(D') then $R' = P_{<i_1,\ldots,i_k>}(R)$ for some R in FSAT(D).

A property P is hereditary if whenever a set D is P and D' is a projection of D then D' is also P. P is finitely hereditary if whenever a set D is P and D' is a finite projection of D then D' is also P.

In [BV1] we have shown that triviality and finite triviality are undecidable. In fact, the following stronger result was proven there.

Theorem 1. Let L(3) have the predicate symbol Q, and let d_1, d_2 be the dependencies:

d_1: $\forall y_1 \forall y_2 \forall y_3 \exists x_1 \exists x_2$

$(Q(y_1,y_2,y_3) \rightarrow Q(y_3,x_1,x_2))$

d_2: $\forall y_1 \forall y_2 \forall y_3 \exists x_1 \exists x_2$

$(Q(y_1,y_2,y_3) \rightarrow Q(y_2,x_1,x_2))$.

The sets $\{D: D \subseteq L(3)$ and $D \cup \{d_1,d_2\}$ is trivial$\}$ and $\{D: D \subseteq L(3)$ and $D \cup \{d_1,d_2\}$ is finitely trivial$\}$ are not recursive. □

In [BV1] this theorem is used to prove that the implication problem and the finite implication problem are unsolvable. Here we use it to prove our general result about decidability of properties.

Theorem 2.

a) Let P be a finitely nice non-trivial finitely hereditary property then P is undecidable.

b) Let P be a nice non-trivial hereditary property then P is undecidable.

Outline of the proof. We use a reduction from the non-recursive sets of Theorem 1. Let $D \subseteq L(3)$ be a set for which we want to decide whether $D \cup \{d_1,d_2\}$ is (finitely) trivial. We construct a set $D' \subseteq L(n)$, for some $n > 3$, such that if $D \cup \{d_1,d_2\}$ is (finitely) trivial then so is D', and if $D \cup \{d_1,d_2\}$ is not (finitely) trivial then there is a set D'' which is a (finite) projection of D' and is not P, and, consequently, D' is not P. It follows that $D \cup \{d_1,d_2\}$ is (finitely) trivial iff D' is P, so P is undecidable. □

Corollary. The following properties are undecidable: decidability, finite decidability, finite implication, Armstrongness, finite Armstrongness, completeness and finite completeness.

Proof. All of these properties are non-trivial and either finitely nice and finitely hereditary or nice and hereditary. □

Unfortunately, redundancy is neither (finitely) nice nor (finitely) hereditary so its decidability issue is not answered by Theorem 2. Nevertheless, by an argument similar to that in the proof of Theorem 2, we can show:

Theorem 3. Redundancy and finite redundancy are undecidable. □

4. Conclusions

Twenty years ago it was believed that a powerful and efficient proof procedure that can create demonstrations of logical propositions is a major part in getting a machine to behave intelligently. Years passed, the desired proof procedure continued to elude researchers, and this approach was rendered naive and simplistic. Artificial intelligence research is now detailed and nitty-gritty rather than vague and general.

We contend that the hope of having a fully automated database schema processor is similarly naive and simplistic. Modelling the real world is an immensely complicated task for which perhaps no elegant algorithm exists. It is also known that even when one deals with the simple functional and multivalued dependencies there are pathological cases that defy elegant solutions. We do appreciate the merits of theoretical research, but we think that research has got to the point where heuristics and methodologies should enter the game (see for example [BFMMUY,Sc]).

Acknowlegement

I would like to thank David Harel for his helpful comments.

REFERENCES

[Adj] Adjan, S.I.: The algorithmic unsolvability of checking certain properties of groups (in Russian). Dokl. Akad. Nauk SSSR 103(1955), 533-535.

[ANS] ANSI/X3/SPARC Interim Report 75-02-08, FDT-SIGMOD 7(1975).

[BBG] Beeri, C., Bernstein, P.A., Goodman, N.: A sophisticate's introduction to database normalization theory. Proc. Conf. on VLDB, Berlin, 1978, 113-124.

[Beer] Beeri, C.: On the role of data dependencies in the construction of relational database schemas. Hebrew University of Jerusalem, 1979.

[Bern] Bernstein, P.A.: Synthesizing third normal form relations from functional dependencies. ACM TODS 1(1976), 277-298.

[BFMMUY] Beeri, C., Fagin, R., Maier, D., Mendelzon, A.O., Ullman, J.D., Yannakakis, M.: Properties of acyclic database schemes. Proc. 13th ACM STOC, 1981.

[BMSU] Beeri, C., Mendelzon, A.O., Sagiv, Y., Ullman, J. D.: Equivalence of relational database schemes. Proc. 11th ACM STOC, 1979, 319-329.

[Bo] Boone, W.W.: Decision problems about logical systems as a whole and recursively enumerable degrees of unsolvability. In Contribution to mathematical Logic (H.A. Schmidt, K. Schutte and H.J. Thiele, eds.), North-Holland, 1968, 13-33.

[BR] Beeri, C., Rissanen, J.: Faithful representation of relational database schemas. IBM Research Report, San Jose, 1979.

[BV1] Beeri, C., Vardi, M.Y.: The implication problem for data dependencies. Proc. XP1 Workshop on Relational Database Theory, Stony Brook, 1980. Also, Hebrew University of Jerusalem, 1980.

[BV2] Beeri, C., Vardi, M.Y.: On the complexity of testing implications of data dependencies. Hebrew University of Jerusalem, 1980.

[CLM] Chandra, A.K., Lewis, H.R., Makowsky, J.A.: Embedded implicational dependencies and their inference problem. Proc. XP1 Workshop on Relational Database Theory, Stonybrook, 1980. Revised, 13th ACM STOC.

[Codd1] Codd. E.F.: A relational model for large shared data bases. CACM 13(1970), 377-387.

[Codd2] Codd, E.F.: Further normalization of the database relational model.In Data Base Systems (R. Rustin, ed.), Prentice-Hall, 1972.

[Codd3] Codd, E.F.: Extending the database relational model to capture more meaning. ACM TODS 4(1980).

[Fag1] Fagin, R.: Multivalued dependencies and a new normal form for relational databases. ACM TODS 2(1977), 262-278.

[Fag2] Fagin, R.: Horn clauses and database dependencies. Proc. 12th ACM STOC, 1980, 123-134.

[Luk] Luk, W.S.: "Possible" membership of a multivalued dependency in a relational database. Info. Proc. Lett. 9(1979), 80-83.

[Mar] Markov, A.A.: Impossibility of algorithms for distinguishing certain properties of associative systems (in Russian). Dokl. Akad. Nauk SSSR 77(1951), 953-956.

[Ra] Rabin, M.O.: Recursive unsolvability of group theoretic problems. Ann. Math. 67(1958), 172-194.

[Sc] Sciore, E.: Real world mvd's. Proc. ACM-SIGMOD International Conf. on Management of Data, 1981.

[SM] Silva, A.M. Melkanoff, M.A.: A methodology for inferring the dependencies of a relation. Proc. Workshop on Formal Bases for Data Bases, Toulouse, 1979.

[Zan] Zaniolo, C.: Analysis and design of relational schemata for database systems. UCLA, 1976.