THE DECISION PROBLEM FOR DATABASE DEPENDENCIES

M.Y. VARDI *

Department of Applied Mathematics, The Weizmann Institute of Science, Rehovot, Israel

Received 13 August 1980; revised version received 31 March 1981

Database, dependency, decision problem, recursive, recursively enumerable

1. Introduction

In the last decade a great deal of interest has been aroused in sentences of first order logic which are adequate as database constraints — the so called data dependencies. Since the introduction of functional dependencies by Codd [5], several additional classes of dependencies have been investigated, e.g., multi-valued dependencies [8,17], join dependencies [1,13], etc. These classes are all syntactically specified. Recently, there has been an effort to characterize these sentences model-theoretically [4,9]. Chandra et al. [4] define a dependency as a sentence that holds in the trivial structure and is domain independent, i.e., the truth or falsity of such a sentence in some structure is independent of the domain and depends only on the relations of the structure. The decision problem for dependencies is to decide whether a given sentence is a dependency. We show that this problem is recursively unsolvable. In fact, the set of dependencies is not recursively enumerable. This is closely related to the recursive unsolvability of the decision problem for definite formulas of Di Paola [7].

R₁, ..., — the arity of Rᵢ is i, however, each sentence s contains at most one predicate symbol ¹. Thus, s(Rᵢ) denotes the sentence s with the predicate symbols Rᵢ.

A structure for L, M = (D, R₁, R₂, R₃, ...), is finite if D is finite. Unless explicitly stated otherwise, the structures dealt with are assumed to be finite. When concerned with a sentence s(Rᵢ) we denote a structure by (D, Rᵢ). A structure (D, Rᵢ) is empty if Rᵢ = φ, is trivial if |D| = |Rᵢ| = 1, and is simple if for some a ∈ D, Rᵢ = {⟨a, ..., a⟩}. Note that a trivial structure is simple.

A sentence s(Rᵢ) is domain independent if for every two structures M₁ = (D₁, Rᵢ) and M₂ = (D₂, Rᵢ) (i.e., different domains but the same relation), we have M₁ ⊨ s iff M₂ ⊨ s, i.e., s holds in M₁ iff it holds in M₂. A domain independent sentence is called definite in [7], permissible in [6], range restricted in [12], and safe in [11]). A sentence s is a dependency if it is domain independent and holds in the trivial structure.

2. Dependencies

The language L that we use is the language of first order logic with equality with predicate symbols R₁,

* Current address: Department of Computer Science, The Hebrew University of Jerusalem, Jerusalem, Israel.

¹ The assumption that the database consists of a single relation is called the 'universal relation assumption'. Our results clearly hold for sentences with several predicate symbols.

0020-0190/81/0000-0000/$02.50 © 1981 North-Holland

3. Some lemmas

Let Rᵢ be an i-ary relation, i > 1. We define the projection of Rᵢ, denoted p(Rᵢ), as the i-1-ary relation:

p(Rᵢ) = {⟨a₁, ..., aᵢ₋₁⟩ | for some a₀, ⟨a₀, ..., aᵢ⟩ ∈ Rᵢ}.
Let \( s(R_i) \) be a sentence. By replacing each occurrence of an atomic formula \( R_i(u_1, \ldots, u_l) \) in \( s \) by \( \exists x R_{i+1}(u_1, \ldots, u_l, x) \), where \( x \) is a new variable, we get a new sentence \( s'(R_{i+1}) \).

**Lemma 1.** (a) If \( \langle D, R_{i+1} \rangle \models s' \) then \( \langle D, p(R_{i+1}) \rangle \models s \).
(b) If \( \langle D, R_i \rangle \models s \) then \( \langle D, R_i \times D \rangle \models s' \).

**Proof.** For a formula \( f(R_i) \), let \( f'(R_{i+1}) \) be the result of applying the above transformation to \( f \). For a formula \( f(R_i) \) with free variables \( x_1, \ldots, x_k \) and a structure \( M = \langle D, R_i \rangle \), we define \( T(f, M) \) as the set of members of \( D^k \) for which \( f \) holds in \( M \), i.e.,

\[ T(f, M) = \{ \langle a_1, \ldots, a_k \rangle \mid \langle M, a_1, \ldots, a_k \rangle \models f \} \]

If \( k = 0 \) then \( T(f, M) = \text{true} \) if \( M \models f \), and \( T(f, m) = \text{false} \) otherwise. It can easily be shown, by a routine induction on the structure of \( f \), that:
(a) \( T(f', \langle D, R_{i+1} \rangle) = T(f, \langle D, p(R_{i+1}) \rangle) \).
(b) \( T(f, \langle D, R_i \rangle) = T(f', \langle D, R_i \times D \rangle) \).

The claim follows as a specific case.

Let \( s(R_i) \) be a sentence, and let \( x \) be a variable not occurring in \( s \). We replace each atomic formula \( R_i(u_1, \ldots, u_l) \) by the conjunction \( u_l = x \land \cdots \land u_1 = x \), and prefix the resulting formula with \( \exists x \) to get a new sentence \( s^+ \).

**Lemma 2.** (a) Let \( M = \langle D, R_i \rangle \) be a simple structure. \( M \models s \) implies \( \langle D \rangle \models s^+ \).
(b) If \( \langle D \rangle \models s^+ \) and \( a \in D \) then \( \langle D, \{ a, \ldots, a \} \rangle \models s \).

**Proof.** Analogous to the proof of Lemma 1.

Let \( s(R_i) \) be a sentence, and let \( x \) be a variable not occurring in \( s \). We replace each atomic formula \( R_i(u_1, \ldots, u_l) \) by \( x \neq x \) and prefix the resulting formula with \( \exists x \) to get a new sentence \( s^a \).

**Lemma 3.** \( \langle D, \phi \rangle \models s \) if and only if \( \langle D \rangle \models s^a \).

**Proof.** Analogous to the proof of Lemma 1.

**Lemma 4.** The set of sentences having a non-empty model is not recursive.

\( \times \) denotes the Cartesian product.

**Proof.** By Lemma 3, \( s \) has an empty model iff \( s^a \) is satisfiable. Since \( s^a \) is an equality sentence we can effectively test whether it is satisfiable \([13]\). Thus, the set of sentences having an empty model is recursive. If the set of sentences having a non-empty model were recursive, then the set of satisfiable sentences would also be recursive. But the last set is not recursive \([15]\) — a contradiction. The claim follows.

**4. The main result**

**Theorem 1.** The set of domain independent sentences is not recursive.

**Proof.** We show that the decision problem for sentences having a non-empty model is reducible to the decision problem for domain independent sentences. Let \( s(R_1) \) be a sentence. We construct another sentence \( t(R_{i+1}) : s \land \forall x_1 \ldots \exists x_i \forall x_{i+1} \exists x_{i+1} \forall x_1 \cdots x_{i+1} \).

\( s \) has a non-empty model iff \( t \) is not domain independent: suppose first that \( s \) has no non-empty model, then \( t \) has no model, hence, it is domain independent. Suppose now that \( s \) has a non-empty model \( M = \langle D, R_i \rangle \), then by Lemma 1, \( s' \) has a non-empty model \( M_i = \langle D, R_{i+1} \rangle \), where \( R_{i+1} = R_i \times D \). \( M_i \) is also a model of \( t \). Let \( a \) be any element not in \( D \), and let \( M_2 = \langle D \cup \{ a \}, R_{i+1} \rangle \). Since \( M_2 \models t \), \( t \) is not domain independent.

A sentence \( s(R_i) \) is **weakly domain independent** if for every two non-simple structures \( M_1 = \langle D_1, R_i \rangle \) and \( M_2 = \langle D_2, R_i \rangle \), we have \( M_1 \models s \iff M_2 \models s \).

**Lemma 5.** The set of weakly domain independent sentences is not recursive.

**Proof.** We show that the decision problem for domain independent sentences is reducible to the decision problem for weakly domain independent sentences. Let \( s(R_1) \) be a sentence. If there are two simple structures \( M_1 = \langle D_1, R_i \rangle \) and \( M_2 = \langle D_2, R_i \rangle \), such that \( M_1 \models s \) but \( M_2 \not\models s \), then \( s \) is not domain independent. If, on the other hand, we have that either for all simple structures \( M, M \models s \), or for all simple structures \( M, M \not\models s \), then \( s \) is domain independent iff \( s \) is weakly domain independent. Suffice it to show that we can effectively decide whether there are two simple struct-
Proof. Since the set of satisfiable sentences $s(R_2)$ is not recursive [3], we can prove the claim by repeating the chain of arguments leading to Theorem 2.

Theorem 4. The set of strongly dependencies is recursively enumerable.

Proof. Suffice it to show that the set of strongly domain independent sentences $s(R_i)$ is recursively enumerable, for all $i \geq 1$. We show that the decision problem for strongly domain independent sentences $s(R_i)$ is reducible to the validity problem for first order logic with equality which is recursively enumerable [10]. Let $f$ be a formula, and let $P$ be a unary predicate symbol. If we replace every subformula $\forall x(g)$ or $\exists x(g)$ in $f$, by the formula $\forall x(P(x) \rightarrow g)$ or $\exists x(P(x) \land g)$ respectively, then the resulting formula $f(P)$ is said to be obtained by relativizing $f$ to $P$ [14]. Let $s(R_i)$ be a sentence, let $P$ and $Q$ be two unary predicate symbols, and let $s(R_i, P)$ and $s(R_i, Q)$ be the result of relativizing $s$ to $P$ and $Q$ respectively. We construct three sentences:

$$t_1: \forall x_1 \ldots \forall x_i (R(x_1, \ldots, x_i) \rightarrow P(x_1) \land \ldots \land P(x_i)),$$

$$t_2: \forall x_1 \ldots \forall x_i (R(x_1, \ldots, x_i) \rightarrow Q(x_1) \land \ldots \land Q(x_i)),$$

$$t: t_1 \land t_2 \rightarrow (s(R_i, P) \leftrightarrow s(R_i, Q)).$$

The reader can verify that $s$ is strongly domain independent iff $t$ is valid.

Remark. The solvability of the decision problem for the monadic predicate calculus [2,11] entails the solvability of the decision problem for (strong) dependencies $s(R_i)$.

Acknowledgement

I wish to thank Ron Fagin for his helpful comments.

References


