ON SAFETY, DOMAIN INDEPENDENCE, AND CAPTURABILITY OF DATABASE QUERIES*

(Preliminary Report)

Michael Kifer
Department of Computer Science
Stony Brook University
Stony Brook, NY 11794, USA

ABSTRACT

We study the relationship between domain independent and safe queries, showing that for some query classes these notions are the same, while for the others they are different. We also consider the question of when these properties can be decided and the corresponding query classes be evaluated (captured).

1. Introduction

Domain independent and safe queries have attracted considerable attention in the database community. Unfortunately, there has been some confusion about these terms in the literature. Domain independent queries were introduced in [1,2], and Nicolas and Demolombe [3] proved that this class coincides with the class of *definite* queries previously studied in [4,5]. The term ''safety'' has a shorter but more confusing history. It was introduced in [6] to denote a subclass of first order queries which can be translated into relational algebra. Unfortunately, the very same term was later used for different purpose: to denote the class of queries with finite set of answers [7,8,9]. This latter notion of safety is *semantic* in nature, as opposed to the *syntactic* definition in [6]. The class of semantically safe queries properly contains the class of syntactically safe ones. In the past few years, the word ''safety'' has been used to denote these two query classes interchangeably, which added to the confusion.

The relationship between syntactic safety and domain independence was investigated in [3]. However, the connection between domain independence and semantic safety remained unclear. Some researchers mistakenly believed that the two classes coincide. In this paper we investigate this issue, and show that for some classes of queries the two notions indeed coincide, while for the others they do not. We also discuss the decidability of detecting safety and domain independence. Some un/decidability results are known [10,1,5,11,12], and we just summarize them, while others (e.g., decidability of universal safety for Horn queries without function symbols, and of the relative safety for fixpoint queries) are new.

Recently Aylamazyan et. al. [10] proposed yet another notion of safety which is closely related to semantic safety. While a semantically safe query has finite number of answers regardless of the data stored in the relations, Aylamayan et. al. considered the question of whether a query has a finite number of answers with respect to a *given* database instance. They showed that the two notions are quite different, and called their new notion *relative safety*, while semantically safe queries were referred to as *universally safe*. Relative safety seems more important practically, since queries are always evaluated w.r.t. a *given* database instance, and the user usually wants to know whether this particular evaluation yields a finite set of answers. Our study includes relatively safe queries as well. From now on we will use the term ''safety'' in

* Research sponsored in part by the NSF grant DCR-8603676.

the semantic sense.

Likewise, we distinguish between *universally* and *relatively* domain independent queries. The former class corresponds to the notion studied in [2,1,4,5], while the latter includes queries which do not depend on the domain, given that the database instance is fixed. There was some terminological confusion regarding domain independence either. For instance, in [12] the two different notions of domain independence are referred to by the same name.

Besides safety and domain independence, there is one more way of looking at the issue of "reasonableness" of a query. From a practical standpoint, we want to be able to *evaluate* queries. Hence, the issue here is whether there exists an *algorithm* which for each query in a given class enumerates all answers and terminates. Classes of queries for which such algorithms can be found are called *capturable*[1] [13] (also cf. [15]). Capturability is closely related to safety, since every query in a capturable class is safe. However, there are classes of safe queries which are not capturable [13]. Studying capturability of various classes of safe queries is another purpose of this paper. For instance, we show that without function symbols the classes of safe and domain independent queries are capturable, even though these classes are not recursively enumerable. Thus, capturability provides, in a sense, an orthogonal way of looking at these issues.

We concentrate our attention on the following query classes: first order queries without function symbols (FO-f), and with function symbols (FO+f); Horn queries without functions (H-f), and with functions (H+f); stratified queries without without function symbols (S-f), and with function symbols (S+f); fixpoint queries with (FP+f) and without (FP-f) function symbols. Chandra and Harel [16] and Kolaitis [17] (also see [18] for a survey) investigated the relationship among the aforementioned query classes.[2] In particular, they have shown that these classes can be organized in a hierarchy of Figure 1, where each directed arc denotes strict inclusion.

2. Preliminaries

We assume that the reader is familiar with the basic concepts of Deductive Databases and Logic Programming [19,9,20], and with the notion of stratification [20,16,21]. Database **D** consists of a set of

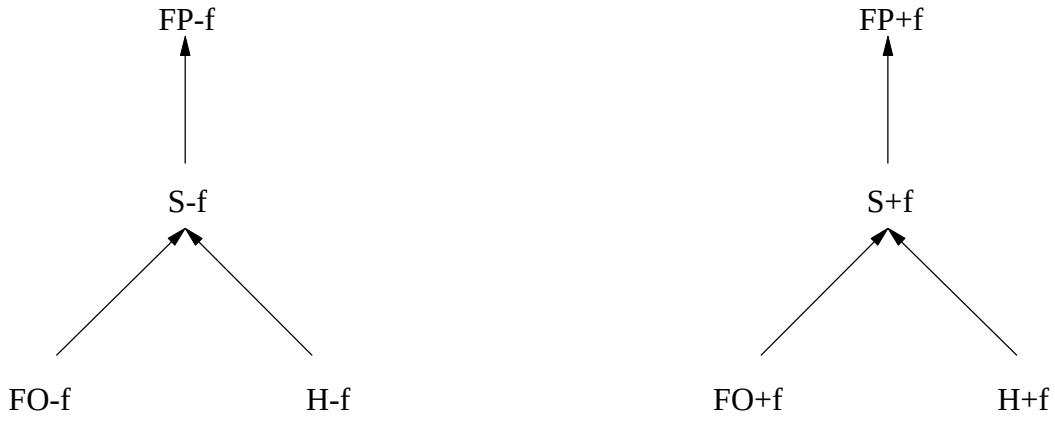

Figure 1 - Query Hierarchy

[1] Essentially the same notion was independently proposed in[14], where it was called *effective computability*.

[2] Chandra and Harel [16] used the names F for FO-f and YE^+ for H-f. We use different notation because of the simpler mnemonics.

predicate names (relation symbols) $R_1, \ldots, R_n$. For convenience, we also let the database contain a special predicate *Domain* whose extension is always the entire domain over which values of database relations and queries may range. To distinguish R_1, ..., R_n from the special predicate *Domain*, we will call the former *proper* predicates of **D**. We assume the existence of an interpreted equality predicate. However, our proofs go through also for the languages without "=", if one simply ignores the parts related to the equality. In addition, the language has a set of function symbols $\mathbf{F}_0$, ..., $\mathbf{F}_i$, ..., where each $\mathbf{F}_i$ denotes a countably infinite set of i-ary function symbols. The 0-ary function symbols (elements of $\mathbf{F}_0$) are called *constants*. Given a subset $\mathbf{F}'$ of the set of function symbols, $\mathbf{F}' \subseteq \bigcup_i \mathbf{F}_i$, a *domain* U of **D** consists of the set of all ground terms which can be constructed using the constants and other function symbols of $\mathbf{F}'$. Database *instance* **d** of **D** over U consists of a finite set of ground atomic formulae of the form $R(t_1, \ldots, t_k)$, where $R \in \mathbf{D}$ is a relation name and t_js are members of U, plus a set of atomic formulae *Domain*(s) for each s in U.

Thus, by the definition, domain is part of the database. Nevertheless, it will be sometimes convenient to write $(\mathbf{d}, U)$ to emphasize the fact that **d** is a proper part of a database instance and U is a domain. Notice that the extension of *Domain* may be infinite, while extensions of the proper database predicates are finite. This restriction will be temporarily lifted in Proposition 7, where certain designated relations will be allowed to have infinite extensions.

An m-ary *query* **q** to a database **D** is a mapping from the set of instances of **D** to the set of (possibly infinite) subsets of U^m. We will be interested in some specific classes of queries as outlined in Section 1.

The class FO+f consists of queries of the form $\mathbf{q} = \{ \bar{X} \mid \Phi(\bar{Y}) \}$, where $\bar{Y}$ is the set of all free variables of the first-order formula Φ, and $\bar{X}$ is a list of variables of length m containing all variables in $\bar{Y}$. Suppose $\bar{X} = \{ X_1, \ldots, X_m \}$ and $\bar{Y} = \{ X_{i_1}, \ldots, X_{i_n} \}$. Given a database instance **d**, the semantics of such a query is the set of all tuples $t = \{ t_1, \ldots, t_m \} \in U^m$, such that $(\mathbf{d}, U) \models \Phi(t_{i_1}, \ldots, t_{i_n})$. Here we use $(\mathbf{d}, U)$ instead of just **d** to emphasize that in any interpretation I of **d**, each element of U is interpreted by some value in the domain of I, and vice versa. The class FO-f is just like FO+f, except that all $\mathbf{F}_i$, $i \geq 1$, are empty, and thus U contains constants only.

Classes H+f and H-f consist of queries defined by means of Horn rules [20,9] with and without function symbols (other than constants), respectively. More precisely, a query **q** in $H \pm f$ is a pair $(\mathbf{H}, Q)$, where **H** is a set of Horn rules defining an m-ary query predicate Q. Given a database instance $(\mathbf{d}, U)$, the meaning of the query is the set of all tuples $t \in U^m$ such that $(\mathbf{d}, U) \cup \mathbf{H} \models Q(t)$. Equivalently, meaning of Q is the set of all tuples t such that $Q(t)$ is in the minimal Herbrand model of $(\mathbf{d}, U) \cup \mathbf{H}$ (note that here U plays the role of Herbrand Universe). We will use the fact that Q can be computed by applying the rules of **H** to **d** bottom-up (in forward chaining)[22]. In[16] it is shown that H±f coincides with the class of queries constructed by applying the least fixed point operator to positive existential first-order queries.

A *generalized Horn*[3] rule is an implication $Q(\ldots) :- S_1(\ldots), \ldots, S_k(\ldots)$, where Q is a positive literal, and the S_is may be positive or negative. Informally, a set of generalized Horn rules **H** is *stratified* if **H** does not contain recursion through negation. A formal definition can be found in [21,20]. The class S±f consists of the queries $\mathbf{q} = (\mathbf{H}, Q)$, where **H** is a set of generalized Horn stratified rules defining the query predicate Q. The difference between the classes S-f and S+f is that, in the former, formulas do not contain function symbols other than constants.

The semantics of a query $\mathbf{q} = (\mathbf{H}, Q) \in$ S±f, where **H** is a set of generalized Horn rules defining an m-ary query predicate Q, is the set of all tuples $t \in U^m$ such that $Q(t)$ is in the *perfect* model of $\mathbf{d} \cup \mathbf{H}$ [23]. Computationally, **q** can be evaluated by applying the rules of **H** in a bottom-up manner so that the rules in the lower strata are applied before the rules in the higher strata. In this computation, relations corresponding to negative occurrences of any m-ary predicate, $\neg P$, are defined to be $U^m - p$, where p is the relation for P, computed at a lower stratum.

---

[3] Lloyd [20] calls these rules *program* or *database clauses*.

Fixpoint queries constitute the largest class of queries considered in this paper (see Fig. 1). A query of this class is a statement of the form $\mathbf{q} = \{ \bar{X} | \Psi(\bar{Y}) \}$, where Ψ is a fixpoint formula with free variables $\bar{Y}$, and $\bar{X}$ is a list of variables containing all variables in $\bar{Y}$. A fixpoint formula (see [16] for details) is either a first order formula or a formula obtained from simpler fixpoint formulas by means of the usual logical connectives and the least fixed point operator (abbr. LFP). If $\bar{X} = \{ X_1, \ldots, X_m \}$, $\bar{Y} = \{ X_{i_1}, \ldots, X_{i_n} \}$, and $(\mathbf{d}, U)$ is a database instance, the semantics of such a query is the set of all tuples $t = \{ t_1, \ldots, t_m \} \in U^m$, such that $(\mathbf{d}, U) \models \Psi(t_{i_1}, \ldots, t_{i_n})$. Again, $\models$ is restricted here to the interpretations whose domain is an epimorphic image of U.

A query $\mathbf{q}$ to $\mathbf{D}$ is *universally safe* if for any database instance $(\mathbf{d}, U)$ of $\mathbf{D}$, the set of answers to $\mathbf{q}$ is finite. Given a database instance $\mathbf{d}$, the query $\mathbf{q}$ is *safe relatively to* $\mathbf{d}$, if $\mathbf{q}$ has a finite number of answers when it is evaluated on $\mathbf{d}$, regardless of the domain U. By the definition, universal safety entails relative safety w.r.t. all database instances. Query $\mathbf{q}$ is *universally domain independent* if for every pair of database instances $(\mathbf{d}, U)$ and $(\mathbf{d}, \hat{U})$ (i.e., same relations, different domains), such that U and $\hat{U}$ are both finite, $\mathbf{q}$ has the same set of answers. A query $\mathbf{q}$ is domain independent *relatively* to a *given* database instance $\mathbf{d}$ iff for every pair of finite domains, U and $\hat{U}$, the sets of answers to $\mathbf{q}$ w.r.t. $(\mathbf{d}, U)$ and $(\mathbf{d}, \hat{U})$ coincide.

Our notion of universal domain independence is what is called ''domain independence'' in [2,1]. On the other hand, the concept of domain independent databases in [12] is different from domain independence of [2,1]: it corresponds to our notion of relative domain independence.[4]

In the definition of safety, it is essential that the domains can be infinite (or else every query will be safe). The concept of domain independence on the other hand, refers only to finite domains. However, when function symbols are allowed, domains are always infinite, hence the condition in the definition of domain independence trivially holds true for every query. Fortunately, this obstacle can be removed, which enables an extension of the notion of domain independence to infinite domains.

THEOREM 1. Query $\mathbf{q} \in$ FP-f is domain independent relatively to $\mathbf{d}$ iff for every pair of finite of infinite domains U and $\hat{U}$, $\mathbf{q}$ has the same set of answers w.r.t. $(\mathbf{d}, U)$ and $(\mathbf{d}, \hat{U})$.

COROLLARY 1. $\mathbf{q} \in$ FP-f is universally domain independent iff for every pair of instances, $(\mathbf{d}, U)$ and $(\mathbf{d}, \hat{U})$, where U and $\hat{U}$ may be infinite, $\mathbf{q}$ has the same set of answers.

PROOF. The ''if'' direction is trivial. For the ''only if'' part, suppose that for some $\mathbf{d}$ the query has different sets of answers w.r.t.$(\mathbf{d}, U)$ and $(\mathbf{d}, \hat{U})$. Since $\mathbf{q}$ is universally domain independent, it is domain independent relatively to $\mathbf{d}$, and the claim follows from Theorem 1. □

To avoid possible confusion we note that Vardi [1] introduced the notion of *strongly* domain independent queries which is different from universal domain independence, since the former class is recursively enumerable, while the latter is not. Strong domain independence does not require the domains to be finite (as in Theorem 1), but, unlike Theorem 1, it does not require the relations in $\mathbf{d}$ to be finite either, and therefore Theorem 1 does not contradict Vardi's result.

We now relax the definition of domain independence by dropping the requirement that the domains must be finite. By Theorem 1 and Corollary 1, when function symbols are excluded, this new notion agrees with the notion of domain independence as defined in [2,1].[5]

Before proving the theorem, we need some additional notions inspired by[10]. Let $(\mathbf{d}, U)$ be a database instance and $\mathbf{q}$ be a query. Let $Dom(\mathbf{d}, \mathbf{q})$ denote the *active* domain of $\mathbf{q}$ w.r.t. $\mathbf{d}$, i.e. the domain consisting precisely of the ground terms built out of constants and function symbols mentioned in $\mathbf{d}$ or in $\mathbf{q}$. We say that U

---

[4] A deductive database is domain independent in the sense of [12] iff each its predicate, considered as a query, is domain independent relatively to the extensional part of the database.

[5] In [3] finiteness of domains is not required, and the authors freely alternate between finite and infinite domains. However, no proof is given there that the two notions coincide.

is *sufficiently large* with respect to $\mathbf{q}$ and $\mathbf{d}$ if:

(i) $U \supseteq Dom(\mathbf{d}, \mathbf{q})$, and

(ii) The number of elements in $U - Dom(\mathbf{d}, \mathbf{q})$ is at least 1+2*(number of occurrences of "=" and "≠" in $\mathbf{q}$).

Let $t_1, t_2 \in U^m$ be a pair of tuples. We say that t_1 is *similar* to t_2 w.r.t. $(\mathbf{d}, U)$ and $\mathbf{q}$ if the following conditions are met:

(i) for each j, $t_1[j] \in Dom(\mathbf{d}, \mathbf{q})$ iff $t_2[j] \in Dom(\mathbf{d}, \mathbf{q})$. In addition, whenever $t_1[j]$, $t_2[j] \in Dom(\mathbf{d}, \mathbf{q})$ then $t_1[j] = t_2[j]$. Here $t[j]$ denotes the j-th component of tuple t;

(ii) $t_1[j] = t_1[k]$ iff $t_2[j] = t_2[k]$, for all $j, k \in [1, m]$.

LEMMA 1. Let $\mathbf{q} \in$ FP-f be a query to $(\mathbf{d}, U)$, and suppose t_1 is an answer tuple to $\mathbf{q}$. Then every tuple t_2 similar to t_1 (w.r.t. $(\mathbf{d}, U)$ and $\mathbf{q}$) is also an answer to $\mathbf{q}$.

PROOF. The proof is by induction on the structure of $\mathbf{q}$. □

LEMMA 2. If $t \in U^m$ is an answer to $\mathbf{q} \in$ FP-f w.r.t. $(\mathbf{d}, U)$ for *some* sufficiently large domain $\hat{U}$, then t is an answer to $\mathbf{q}$ w.r.t. $(\mathbf{d}, \bar{U})$, for *every* sufficiently large domain $\bar{U}$ such that $t \in \bar{U}^m$.

PROOF. Uses Lemma 1 and the induction on the structure of $\mathbf{q}$. □

Proof of Theorem 1: The "if" direction is trivial. For the "only if" part let U be an infinite countable domain. Since $\mathbf{q}$ and $\mathbf{d}$ do not have function symbols, any infinite domain, U in particular, is sufficiently large. Since $\mathbf{q}$ is domain independent relatively to $\mathbf{d}$, its set of answers does not depend on the choice of a finite domain, $\hat{U}$. Particularly, if $\hat{U}$ is chosen to be sufficiently large, then, by Lemma 2, the sets of answers to $\mathbf{q}$ w.r.t. $(\mathbf{d}, \hat{U})$ and $(\mathbf{d}, U)$ have to coincide. □

COROLLARY 2. Suppose $(\mathbf{d}, U)$ is an instance with an infinite domain U. Then, if a query $\mathbf{q} \in$ FP-f has a finite number of answers to $(\mathbf{d}, U)$, it is safe relatively to $\mathbf{d}$. □

Thus, in order to verify relative safety, it suffices to consider only one infinite domain.

Let $\mathbf{Q}$ be a class of queries. This class is *capturable* [13] if there exists an algorithm that for every query $\mathbf{q} \in \mathbf{Q}$ and every database instance enumerates all answers to $\mathbf{q}$ and terminates. Obviously, every query in a capturable class is universally safe, but not vice versa. In order to be able to talk about capturability of relatively safe or domain independent queries, we extend the above definition as follows. Let $\mathbf{S}$ be a set $\{\mathbf{q}_i, (\mathbf{d}_i, U_i)\}_{i \in I}$, where the $\mathbf{q}_i$s are queries and the $(\mathbf{d}_i, U_i)$s are database instances. We do not assume that $i \neq j$ implies $\mathbf{q}_i \neq \mathbf{q}_j$ or $\mathbf{d}_i \neq \mathbf{d}_j$, etc. We say that $\mathbf{S}$ is *capturable* if there is an algorithm which for every pair $(\mathbf{q}_i, (\mathbf{d}_i, U_i)) \in \mathbf{S}$ enumerates all answers to $\mathbf{q}_i$ w.r.t. $(\mathbf{d}_i, U_i)$ and terminates.

These two notions of capturability are related as follows: Let $\mathbf{q}$ be a class of queries and $\mathbf{S}$ be a set of *all* pairs $(\mathbf{q}, \mathbf{d})$, where $\mathbf{q} \in \mathbf{Q}$ and $\mathbf{d}$ is some database instance. Then it easily follows from the definitions that $\mathbf{Q}$ is capturable iff $\mathbf{S}$ is. Thus, talking about capturability of universally safe or domain independent queries, we mean the former notion. On the other hand, speaking of the capturability of the relatively domain independent or safe queries we mean the later concept, namely, the capturability of the class of all pairs $(\mathbf{q}, \mathbf{d})$ s.t. $\mathbf{q}$ is safe relatively to $\mathbf{d}$ and $\mathbf{q}$ belongs to an appropriate query class (e.g., FO±f, H±f, etc.).

3. First-Order Queries

It is shown in [5,1] that the class of all universally domain independent queries is not recursive, and even not recursively enumerable. A slight modification of the proof in [5] shows that the class of all universally safe queries (in FO-f, hence also in FO+f) is not recursively enumerable.

To see this, recall that the class of all finitely valid sentences (i.e., formulas without free variables, which are true in all interpretations with finite domains, called *finite* interpretations) is not recursively enumerable [24]. In our case, interpretations are database instances, which are similar to finite interpretations

in that each predicate is interpreted by a finite relation, but different in that the domain of a database instance may be infinite. Let us call such interpretations *database interpretations*. Obviously, each finite interpretation is also a database interpretation, but not vice versa. By Lemma 2, a sentence S is true (which is equivalent to saying that the empty tuple <> is the answer to S) in a database interpretation with infinite domain if and only if it is true in all finite, sufficiently large interpretations. Therefore, the class of sentences valid in the database interpretations coincides with the class of finitely valid sentences, and thus is not recursively enumerable. Since S is valid iff $\neg S$ is unsatisfiable, the class of all sentences unsatisfiable in database interpretations is not recursively enumerable either.

We can now prove the undecidability of safety in FO-f. First note that the relative safety is decidable:

PROPOSITION 1. ([10]) Relative safety in FO-f is decidable, and the class of relatively safe queries in FO-f is capturable. □

As a consequence, the class of queries which are not universally safe is recursively enumerable. Indeed, for any query we can enumerate all the database instances and check whether the result of the query is infinite. If the query is not universally safe, then eventually we will find such an instance.

PROPOSITION 2. The class of all universally safe queries in FO-f is not recursively enumerable.

PROOF. Let S be an arbitrary sentence and P be an n-ary ($n \geq 1$) predicate symbol not occurring in S. Let $F(x_1, \ldots, x_n) \equiv S \wedge (P(x_1, \ldots, x_n) \vee \neg P(x_1, \ldots, x_n))$. Clearly, F is universally safe iff S is unsatisfiable in database interpretations. Therefore, if the class of universally safe queries, **A**, were recursive, so would be the class of unsatisfiable sentences - a contradiction. Since we have just shown that the complement to **A** is recursively enumerable, we conclude that **A** itself is not. □

Next, we consider the relationship between domain independence and safety in FO-f. It is easy to see that universally (resp. relatively) domain independent queries are also universally (resp. relatively) safe. Indeed, if Φ is domain independent and $(\mathbf{d}, U)$ is a database instance, we can choose U to be finite without changing the result of the query. But any query evaluated w.r.t. an instance with finite domain may yield only a finite number of answers. On the other hand, there are safe, but not domain independent queries. A simple example consists a query $\{ X | (\forall Y) P(X, Y) \}$. This query is safe because relation P may contain only a finite number of tuples. It is not relatively (and hence universally) domain-dependent, however. Indeed, let $\mathbf{d} = \{ P(a, a) \}$. Then **q** has an answer $\{ a \}$ if $Domain = \{ a \}$, but the answer would be empty if $Domain = \{ a, b \}$. Note also that the above query is not ''safe'' in the syntactic sense of [6]. Therefore, syntactic safety is different from the semantic one. We thus have the following result:

PROPOSITION 3. In FO-f, universal and relative domain independence implies universal and relative safety, respectively, but not vice versa. □

As far as the capturability is concerned, Proposition 1 guaranties that the class **S** of all pairs $(\Phi, \mathbf{d})$, where Φ is safe relatively to **d**, is capturable. Therefore, classes of domain independent and universally safe queries, being subclasses of **S**, are capturable.

PROPOSITION 4. Classes of universally and relatively domain independent and safe queries in FO-f are capturable. □

Turning to the class FO+f, it follows from the corresponding results about FO-f that the classes of universally domain independent and safe queries are not recursively enumerable. It is still an open issue whether the results of Propositions 3 and 4 carry over to FO+f, but we believe they do. We summarize the results of this section in the following theorem:

THEOREM 2.
In FO-f:

(i) Domain independence implies safety (universal or relative, respectively), but not vice versa.

(ii) Universal domain independence and safety are not recursively enumerable, but relative domain independence and safety are recursive.

(iii) Classes of universally and relatively domain independent and safe queries are capturable.
In FO+f:

(iv) Universal domain independence and safety are not recursively enumerable. □

4. Horn Queries: Classes H+f and H-f

The relationship between domain independence and safety is radically different in the case of Horn queries. For the class H-f, universal (resp. relative) domain independence still implies universal (resp. relative) safety, by the same argument as in the case of FO-f (see Proposition 3). This is no longer true if function symbols are added. Indeed, suppose the database instance consists of a relation p for the predicate P, and let the query predicate Q be defined by a pair of rules $Q(X) \leftarrow P(X)$, $Q(f(X)) \leftarrow Q(X)$. Then the answer to the query is $\{t, f(t), f(f(t)), \ldots\}_{t \in p}$, which does not depend on the domain, but is infinite, hence the query is unsafe.

However, unlike the first order case, safety implies domain independence in H±f.

PROPOSITION 5. Let $\mathbf{q} = (\mathbf{H}, Q) \in$ H±f be a (relatively) universally safe query to a database $\mathbf{D}$. Then $\mathbf{q}$ is (relatively) universally domain independent.

PROOF. (Sketch) Without loss of generality assume that in each rule of $\mathbf{H}$ all variables in the head also appear in the body. To achieve that, we can always modify the rules so that if a rule $r \in \mathbf{H}$ has variables X, Y, Z, etc., appearing in the head of r but not in its body, we add literals $Domain(X)$, $Domain(Y)$, $Domain(Z)$, etc., to the body of r.

The universal part of the claim easily follows from the relative one. So, suppose that $\mathbf{q}$ is safe relatively to some $\mathbf{d}$, but is not domain independent. We show that then it cannot be safe - a contradiction. The proof is by induction on the number of (bottom-up) rule applications needed to establish that $\mathbf{q}$ is not domain independent. In the inductive step we show that if a change in the domain leads to a corresponding change in a set of possible bindings for a variable, say X, appearing in a literal, then the set of bindings for X should have the same cardinality as that of the domain. □

COROLLARY 3. In H-f domain independence coincides with universal safety.

PROOF. At the beginning of this section we observed that in H-f domain independence implies safety. It then follows from Proposition 5 that these notions are equivalent. □

LEMMA 3. In H±f and S±f, the decision problem for relative safety and domain independence reduces to the decision problem for universal safety and domain independence, respectively. Likewise, the capturability problem for relative safety (domain independence) reduces to the corresponding problem for universal safety (domain independence).

PROOF. The trick is very simple. Let $\mathbf{d}$ be a database instance and $\mathbf{q} = (\mathbf{H}, Q)$ be a query. Introduce a new database predicate symbol, $W(X)$, and let the new database instance, $\mathbf{d}'$, consist of some nonempty relation w for W. Turn all the database predicates into derived ones by replacing each database fact $P(\bar{\mathbf{p}})$ by the rule $P(\bar{\mathbf{p}}) \text{:-} W(X)$, where X is some new variable. Let us denote the set of rules thus obtained by $\mathbf{H}'$. Clearly, the set of answers to $\mathbf{q}$ w.r.t. $\mathbf{d}$ is the same as the set of answers to $\mathbf{q}' = (\mathbf{H} \cup \mathbf{H}', Q)$ w.r.t. $\mathbf{d}'$, which completes the construction.

The rest of the proof is easy. For instance, to show that the above construction is, in fact, a reduction for universal safety, it remains to show that $\mathbf{q}$ is safe relatively to $\mathbf{d}$ iff $\mathbf{q}'$ is universally safe. If $\mathbf{q}'$ is universally safe, it is also safe w.r.t. $\mathbf{d}'$. But since the sets of answers to $\mathbf{q}$ w.r.t. $\mathbf{d}$ coincides with the answer set to $\mathbf{q}'$ w.r.t. $\mathbf{d}'$, $\mathbf{q}$ is safe relatively to $\mathbf{d}$. Conversely, if $\mathbf{q}$ is safe relatively to $\mathbf{d}$, then, by the construction, $\mathbf{q}'$ is safe relatively to any nonempty database instance $\mathbf{d}'$. On the other hand, if $\mathbf{d}'$ is empty then so will be the set of answers to $\mathbf{q}'$. Again, $\mathbf{q}'$ will be safe relatively to $\mathbf{d}'$. Hence, $\mathbf{q}'$ is universally safe.

Showing that $\mathbf{q}$ is relatively domain independent iff $\mathbf{q}'$ is universally domain independent is similar. The capturability part of the lemma is now straightforward: If the class of universally safe or domain independent queries is capturable, then, by the above two reductions, the relative classes would be capturable either. □

PROPOSITION 6. The classes of relatively and universally domain independent and safe queries in H+f are not recursively enumerable.

PROOF. Shmueli [11] has shown that universal safety for H+f is undecidable (and even not recursively enumerable) by reducing the complementary problem to the Post Correspondence Problem to the universal safety problem. Answers to the queries used in this reduction do not depend on the database instance (as long as it is not empty). Hence, Shmueli's reduction can be used to show that the complement to the Post Corresponding Problem reduces to the problem of relative safety as well.

To show that relative domain independence in H+f is not recursively enumerable, we reduce the satisfiability problem for Horn clauses to the question of relative domain independence. It is shown in [25,11] that the class of all unsatisfiable sets of clauses of the form $\{\mathbf{d}, \mathbf{H}, \neg Q\}$, where $\mathbf{H}$ is a set of Horn rules (with nonempty head and body), Q is a literal, and $\mathbf{d}$ is the database instance, is not recursive. This class is recursively enumerable though (e.g., SLD-resolution is a semi-decision procedure), hence its complement, the set of satisfiable programs, is not.

Given $\{\mathbf{d}, \mathbf{H}, \neg Q\}$, let $\mathbf{q} = (\{\mathbf{H}, Ans(X) :- Q\}, Ans(X))$ be a query, where Ans is a new predicate and X does not appear in Q. Obviously, Ans has an answer iff $\{\mathbf{d}, \mathbf{H}, \neg Q\}$ is unsatisfiable, in which case the answer is the entire domain, and $\mathbf{q}$ is domain-dependent. Thus, $\mathbf{q}$ is domain independent relatively to $\mathbf{d}$ iff $\{\mathbf{d}, \mathbf{H}, \neg Q\}$ is satisfiable. This takes care of the case when the arity of queries is ≥ 1.

For the 0-ary queries we can show, following [5], that a 1-ary query in H+f, say $F(X)$, is domain independent iff $\forall X(F(X) \rightarrow G(X))$ is, where G is a new 1-ary predicate. The 0-ary case now follows from the impossibility to recursively enumerate the 1-ary relatively domain independent queries. The case of universal domain independence now follows from Lemma 3. □

The situation is quite different for H-f. The decidability of universal safety was claimed in [26] as part of a more general result, but the proof had a mistake. We provide a proof below. Note that in [8] a result (due to C.H. Papadimitriou) is mentioned which states that the universal safety problem for a set of Horn clauses, even without recursion, is undecidable in the presence of infinite *C-relations*, i.e. relations representing the comparison predicates $>$, $\leq$, $\neq$, etc. This does not contradict Proposition 7: universal safety assumes all possible interpretations for the relations, while C-relations have predefined interpretations. Thus, a non universally safe query in H-f may well be safe w.r.t. C-relations.

PROPOSITION 7. In H-f, domain independence and universal safety are decidable, even when some of the base relations are allowed to have infinite extensions.

PROOF. (Sketch) Since, by Proposition 5, universal safety and domain independence in H-f are the same, we will only consider safety. In the proof, we allow *infinite* database relations [26] other than *Domain()*. In fact, we could even allow relations in which certain attributes have only a finite number of different values, while others may have an infinite number of values (but we will not do that, for notational simplicity).

We create a database such that each relation contains only one tuple. Tuples are composed of only two values: d_0 and d_∞. The first value, d_0, represents a finite number of distinct values, while d_∞ denotes an infinite set of values containing those of d_0. For a finite k-ary database predicate $R \in \mathbf{D}$, the corresponding database relation r contains a single tuple $< d_0, d_0, \ldots, d_0 >$ of length k. For an infinite database predicate, such as *Domain*, the corresponding relation consists of a tuple $< d_\infty, d_\infty, \ldots, d_\infty >$ of the appropriate arity. In fact, these tuples are used as meta-tuples: if S (resp. S') denotes a set of values associated with d_0 (resp. d_∞), then the meta-tuple $< d_0, d_0, \ldots, d_0 >$ should be viewed as a concise representation of the Cartesian product $S \times S \times \ldots \times S$, while $< d_\infty, d_\infty, \ldots, d_\infty >$ denotes $S' \times S' \times \ldots \times S'$. Thus, $< d_0, d_0, \ldots, d_0 > \subseteq < d_\infty, d_\infty, \ldots, d_\infty >$. Particularly,

this implies that if, say, a rule $P(X,Y) \text{:-} Q(X,Z), S(Z,Y)$ is applied in a forward chaining to a pair of tuples $< d_0, d_0 >$, $< d_\infty, d_\infty >$ then $Q(\ldots, d_0)$ "matches" $S(d_\infty, \ldots)$, and P gets the tuple $< d_0, d_\infty >$.

Next, we apply the rules bottom-up to the database constructed above. Obviously, this process terminates, since the database is finite and all variables in the heads of the rules also appear in their bodies (see the proof of Proposition 5). To determine whether the query is safe, one has to examine the tuples computed for the query predicate. If one of them contains d_∞ then the query is unsafe. Otherwise, it is safe. The proof is an easy induction on the number of rule applications needed to compute the query. □

PROPOSITION 8. Relative safety in H-f is decidable.

PROOF. To decide whether $\mathbf{q} = (\mathbf{H}, Q)$ is safe relatively to $(\mathbf{d}, U)$ pick some sufficiently large, but finite domain $\bar{U}$, such that $Dom(\mathbf{q}, \mathbf{d}) \subseteq \bar{U} \subseteq U$. If U is finite then, obviously, $\mathbf{q}$ is safe. So, assume U is infinite. Evaluate the query with respect to $\bar{U}$. Since $\bar{U}$ is finite, the evaluation terminates. If the answer contains constants from the set $\bar{U} - Dom(\mathbf{q}, \mathbf{d})$ then $\mathbf{q}$ is unsafe. Otherwise, it is safe w.r.t. $(\mathbf{d}, U)$.

The key in the proof is Lemma 1: If there is an answer tuple, t, containing values from $\bar{U} - Dom(\mathbf{q}, \mathbf{d})$, then there is an infinite number of tuples over U which are similar to t. By Lemma 1, all of them are answers to $\mathbf{q}$ w.r.t. $(\mathbf{d}, U)$. Vice versa, if the query is unsafe w.r.t. $(\mathbf{d}, U)$ then there should be an answer tuple t', which contains constants not in $Dom(\mathbf{q}, \mathbf{d})$. Since $\bar{U}$ is sufficiently large, there should be a similar tuple, t, over $\bar{U}$. By Lemma 1, t is also an answer tuple. Obviously, t contains constants from $\bar{U} - Dom(\mathbf{q}, \mathbf{d})$, or else t' would have to contain only the values drawn from $Dom(\mathbf{q}, \mathbf{d})$, contrary to the choice of t'. □

As a consequence, we see that the class of relatively safe queries in H-f is capturable. Indeed, the algorithm in Proposition 8 actually evaluates the query, and, having found it safe, returns all the answers. By the same argument as in Proposition 4, classes of domain independent and universally safe queries are capturable too.

On the other hand, in [13] it is shown that the class of all relatively safe queries in H+f is non-capturable. Uncapturability of universally safe queries now follows from Lemma 3. By Proposition 7, the class of universally (relatively) domain independent queries in H+f properly contains the class of universally (relatively) safe ones, hence domain independent queries are uncapturable either. In summary, we obtain the following result:

THEOREM 3.

(i) Classes of universally and relatively domain independent and safe queries in H-f are capturable, while in H+f they are not.

(ii) In H-f universal and relative domain independence and safety are recursive, while in H+f they are not recursively enumerable.

(iii) In H+f universal (resp. relative) safety implies universal (resp. relative) domain independence (but not vice versa), and in H-f they coincide. □

5. Stratified Queries: Classes S+f and S-f

In this section we study safety and domain independence of stratified databases. Observe that S+f (resp. S-f) properly includes both FO+f and H+f (resp. FO-f and H-f) [18]. Therefore, the undecidability results for these query classes carry over to S±f.

THEOREM 4.

(i) In S+f, classes of universally and relatively domain independent and safe queries are not recursively enumerable; in S-f, relative safety and domain independence is recursive, while the other two (universal) classes remain not recursively enumerable.

(ii) In S+f, the above four classes are not capturable, while in S-f they are.

(iii) In S-f, universal (relative) domain independence implies universal (relative) safety, while in S+f neither property implies the other one.

PROOF. (i) Universal safety and domain independence are not recursively enumerable in FO-f, by Proposition 2 and the results in [5,1]. Relative domain independence and safety are not recursively enumerable in H+f, by Proposition 6. Decidability of relative safety and domain independence in S-f follows by the same argument as in Proposition 8.
(ii) These query classes are not capturable in H+f, by (i) of Theorem 3. In S-f, relative safety and domain independence are capturable. The argument is the same as in the case of H-f (see the paragraph after the proof of Proposition 8). The proof that universal safety and domain independence are capturable is the same as in Proposition 4.
(iii) In S-f domain independence implies safety by the same argument as in Proposition 3. As to S+f, we saw that in FO-f there are safe but domain dependent queries, while in H+f - the other way around. □

It should be noted that decidability of relative domain independence in S-f was previously established in [12].

6. Fixpoint Queries: Classes FP+f and FP-f

Since FP±f $\supsetneq$ S±f (see Figure 1), most of the results of the previous section carry over.

THEOREM 5.

(i) In FP+f, classes of universally and relatively domain independent and safe queries are not recursively enumerable; in FP-f, relative safety and domain independence is recursive, while the other two (universal) classes remain not recursively enumerable.

(ii) In FP+f, the above four classes are not capturable, while in FP-f they are.

(iii) In FP-f, universal (relative) domain independence implies universal (relative) safety, while in FP+f neither property implies the other one.

PROOF. (i) Non recursive enumerability follows from (i) of Theorem 4. Decidability of the relative safety and domain independence in FP-f is proved by induction on the query structure.
(ii) Uncapturability part follows from (ii) of Theorem 4. The capturability part is proved by induction as in Propositions 4 and 8.
(iii) Same as (iii) of Theorem 4. □

7. Summary

We considered the relationship between domain independence and query safety. For each query class we also showed whether it is recursive or not. The results are summarized in Figures 2 and 3. In these figures, ‘r’ means that the query class is recursive, ‘r.e.’ - that it is recursively enumerable, and ‘c’ - that it is capturable. The classes of universally and relatively domain independent and safe queries are denoted by ‘u.d.i.’, ‘r.d.i.’, ‘u.s.’, and ‘r.s.’, respectively. The question mark means that the corresponding entry is a conjecture only.

	FO-f		FO+f		H-f		H+f, S+f, FP+f		S-f, FP-f	
u. d. i.	¬r.e.	c.	¬r.e.	c. (?)	r.	c.	¬r.e.	¬c.	¬r.e.	c.
u. s.	¬r.e.	c.	¬r.e.	c. (?)	r.	c.	¬r.e.	¬c.	¬r.e.	c.
r. d. i.	r.	c.	r. (?)	c. (?)	r.	c.	¬r.e.	¬c.	r.	c.
r. s.	r.	c.	r. (?)	c. (?)	r.	c.	¬r.e.	¬c.	r.	c.

Figure 2

FO-f	FO+f	H-f	H+f	S-f, FP-f	S+f, FP+f
u.d.i. $\subsetneq$ u.s.	u.d.i. $\subsetneq$ u.s. (?)	u.d.i. = u.s.	u.s. $\subsetneq$ u.d.i.	u.d.i. $\subsetneq$ u.s.	u.d.i. $\not\subseteq$ u.s. u.s. $\not\subseteq$ u.d.i.
r.d.i. $\subsetneq$ r.s.	r.d.i. $\subsetneq$ r.s. (?)	r.d.i. = r.s.	r.s. $\subsetneq$ r.d.i.	r.d.i. $\subsetneq$ r.s.	r.d.i. $\not\subseteq$ r.s. r.s. $\not\subseteq$ r.d.i.

Figure 3

Acknowledgements: This paper is mainly due to the following two circumstances:

(1) a discussion with Paris Kanellakis, who raised some of the questions addressed in Theorem 4, and

(2) mistakes and misconceptions about the notions in question abundant in the database literature.

Thanks also goes to Rodney Topor for the very useful comments on an earlier draft of this paper.

References

1. M.Y. Vardi, "The Decision Problem for Database Dependencies," *Information Processing Letters*, pp. 251-254 (Oct. 1981).
2. R. Fagin, "Horn Clauses and Database Dependencies," *JACM* **29**(4), pp. 952-985 (Oct. 1982).
3. J.-M. Nicolas and R. Demolombe, "On the Stability of Relational Queries," ONERA-CERT Report, Toulouse, France (1983).
4. J.L. Kuhns, "Answering Questions by Computer: A Logical Study," TR# R-511-PR, Rand Corp., Santa Monica (Dec. 1967).
5. R.A. Di Paola, "The Recursive Unsolvability of the Decision Problem for the Class of Definite Formulas," *JACM*, pp. 324-327 (April 1969).
6. J.D. Ullman, *Principles of Database Systems*, CSPRESS, Rockville, MD (1982).
7. G.M. Kuper and M.Y. Vardi, "A New Approach to Database Logic," *PODS* (1984).
8. C. Zaniolo, "Safety and Compilation of Non-Recursive Horn Clauses," *Proc. of the 1st Expert Database Conf.*, pp. 167-178, Charleston, SC (1986).
9. F. Bancilhon and R. Ramakrishnan, "An Amateur's Introduction to Recursive Query Processing Strategies," *ACM SIGMOD* (1986).
10. A.K. Aylamazyan, M.M. Gilula, A.P. Stolboushkin, and G.F. Schwartz, "Reduction of the Relational Model with Infinite Domain to the Case of Finite Domains," *Proc. of USSR Acad. of Science (Doklady)* **286**(2), pp. 308-311 (1986).
11. O. Shmueli, "Decidability and Expressiveness Aspects of Logic Queries," *PODS*, pp. 237-249 (1987).

12. R.W. Topor and E.A. Sonenberg, “On Domain Independent Databases,” pp. 217-240, in *Foundations of Deductive Databases and Logic Programming,* ed. J. Minker, Morgan-Kaufmann, Los Altos, CA (1988).
13. M. Kifer and E.L. Lozinskii, “SYGRAF: Implementing Logic Programs in a Database Style,” *IEEE Transactions on Software Engineering,* pp. 922-935 (July, 1988).
14. R. Krishnamurthy, R. Ramakrishnan, and O. Shmueli, “A Framework for Testing Safety and Effective Computability,” *SIGMOD* (1988).
15. J.D. Ullman, “Implementation of Logical Query Languages for Databases,” *TODS*, pp. 289-321 (Sept. 1985).
16. A.K. Chandra and D. Harel, “Horn Clauses and Generalizations,” *J. of Logic Programming,* pp. 1-15 (1985).
17. P.G. Kolaitis, *The Expressive Power of Stratified Logic Programs*. unpublished manuscript (Nov. 1987).
18. A. Chandra, “Theory of Database Queries,” *PODS*, pp. 1-9 (1988).
19. H. Gallaire, J. Minker, and J-M. Nicolas, “Logic and Databases: A Deductive Approach,” *Computing Surveys* **16**(2), pp. 153-185 (June 1984).
20. J.W. Lloyd, *Foundations of Logic Programming (Second Edition),* Springer Verlag (1987).
21. K.R. Apt, H. Blair, and A. Walker, “Towards a Theory of Declarative Knowledge,” pp. 89-148, in *Foundations of Deductive Databases and Logic Programming,* ed. J. Minker, Morgan-Kaufmann (1988).
22. M.H. van Emden and R.A. Kowalski, “The Semantics of Predicate Logic as a Programming Language,” *JACM* **23**(4), pp. 733-742 (OCT 1976).
23. T.C. Przymusinski, “On the Declarative Semantics of Deductive Databases and Logic Programs,” pp. 193-216, in *Foundations of Deductive Databases and Logic Programming,* ed. J. Minker, Morgan-Kaufmann, Los Altos, CA (1988).
24. B.A. Trahtenbrot, “Impossibility of an algorithm for the decision problem in finite classes,” *Amer. Math. Soc. Translations, Series 2* **23**, pp. 1-5 (1963).
25. Z. Manna, *Mathematical Theory of Computation*, McGraw-Hill (1974).
26. R. Ramakrishnan, F. Bancilhon, and A. Silberschatz, “Safety of Recursive Horn Clauses with Infinite Relations,” *PODS*, pp. 328-339 (1987).