Some Relationships between the FOIES and Σ_1^1 Arity Hierarchies*

Guozhu Dong[†] Limsoon Wong [‡]
Univ. of Melbourne Inst. of Systems Science

11 July 1996
Revised 22 November 1996

1 Introduction

A first-order incremental evaluation system (foies) uses first-order queries to maintain a database view which is defined by a non-first-order query; some auxiliary relations (views) may also need to be maintained similarly. In foies, monadic auxiliary relations were known to be strictly weaker than binary ones; however, it is open whether k-ary auxiliary relations are strictly weaker than $(k+1)$-ary ones for $k \geqslant 2$ if we limit the arity of the input relations to be at most k. (In the full version of the PODS'95 paper [DS95b] which will appear in JCSS, it is shown that the answer to this question is positive if we do not limit the arity of the input relations.) It appears hard to generalize the techniques used for current separation results to solve the above problem. On the other hand, the arity hierarchy of Σ_1^1 was shown to be strict.

The purposes of this note are two folds: Firstly, we aim to explore the relationship between the foies arity hierarchy and the Σ_1^1 arity hierarchy. Given a boolean query Q over k-ary input relations, we show that (i) Q is in $(k+1)$-ary Σ_1^1 whenever Q has a foies, and (ii) Q is in k-ary Σ_1^1 whenever Q has a foies with boolean auxiliary relations. It is still open whether Q is in k-ary Σ_1^1 whenever Q has a m-ary foies ($m > 1$). We note that there is a query Q (namely the $EVEN_k$ query which tests whether a k-ary relation R has even cardinality) which has a foies without auxiliary relations [DS95b] but which is not in $max(1, k-1)$-ary Σ_1^1 [Ajt83].

The above results seem to indicate that Σ_1^1 is more sensitive to input arity than foies. Furthermore, Σ_1^1 and foies are somewhat orthogonal in power: There are queries which are easy in Σ_1^1 (such as 3-colorability) but difficult and perhaps impossible in foies (3-colorability has no foies with monadic auxiliary relations [DS95b], and $PTIME$ would be equal to $NPTIME$ if it has a foies), while at the same time there are queries which are easy in foies (such as $EVEN_k$ and EqC_k, defined below) but difficult in Σ_1^1. Here by difficult we mean requiring auxiliary relations with large arity.

Current knowledge regarding the relationship between the foies and Σ_1^1 arity hierarchies is not symmetric. Indeed, the above results state that the existence of foies will guarantee membership

*Appeared in *Bulletin of European Association for Theoretical Computer Science*, 61:72–79, February 1997.

†Department of Computer Science, University of Melbourne, Parkville, Vic 3052, Australia. Email: dong@cs.mu.oz.au. Partially supported by grants from Australian Research Council.

‡Institute of Systems Science, Heng Mui Keng Terrace, Singapore 119597. E-mail: limsoon@iss.nus.sg. Part of this work was done when Limsoon Wong was visiting University of Melbourne.

in Σ_1^1. However, there is currently no result for the other direction.

Secondly, we report results on the position of some queries in the foies and Σ_1^1 hierarchies. The query $EVEN_k$ is a separator for k-ary Σ_1^1 and $(k-1)$-ary Σ_1^1 for $k > 1$; in contrast, $EVEN_k$ has foies without auxiliary relations and is thus useless for separating the foies arity hierarchy. Somewhat surprisingly, we show that the EqC_k query[1], which tests whether two k-ary relations R_1 and R_2 have equal cardinalities, has a foies using only binary auxiliary relations (by using a new order on k-ary tuples, which is very "stable" when constants disappear), though it is open if it has one using monadic auxiliary relations. So this query is again useless for separating the foies arity hierarchy. In comparison (and as another example to show the differences between the foies and Σ_1^1 arity hierarchies), we show that EqC_k is in $(k+1)$-ary Σ_1^1 but not in $(k-1)$-ary Σ_1^1; it is open if EqC_k is in k-ary Σ_1^1, although this is true when one of the two k-ary relations is "sparse."

Some variants of EqC_k were considered earlier: The query testing whether two chains have the same lengths has a monadic foies but not a 0-ary foies [DS95b]; the query testing whether two cycles have the same lengths has a monadic foies and a binary deterministic foies but not a deterministic monadic foies [DS96]; the query EqC_1 has no deterministic foies [DS96].

Organizationally, we will define and illustrate foies and Σ_1^1 in the next section. In Section 3 we prove the results regarding the existence of foies implies the membership in Σ_1^1. In Sections 4 and 5 we discuss the position of the EqC_k query in the two hierarchies respectively.

2 Definitions and Basic Facts

In this section we define and illustrate the concepts of Σ_1^1 and foies.

A k-ary Σ_1^1 formula, also called k-ary NP formula in the literature, is a second order formula of the form $\exists A_1 \ldots A_m \phi$, where each A_i is a relation variable of arity at most k, and ϕ is a first order sentence. Relation symbols $R \notin \{A_1, \ldots, A_m\}$ which occur in ϕ are referred to as the *input relations*. The formula defines a boolean query over the input relations[2].

Example 2.1 Suppose $EVEN_k$ is the query which tests whether a k-ary relation R has even cardinality, where $k \geqslant 1$. Then $EVEN_k$ is in $max(2,k)$-ary Σ_1^1 [Ajt83] [3] but not in monadic Σ_1^1 [Fag75].

To illustrate the concept of Σ_1^1, we now review the argument for proving that $EVEN_k$ is in $max(2,k)$-ary Σ_1^1. Let T be a binary relation containing a total order on the active domain $adom(R)$, which is defined as the set of constants occurring in R. We can define, in first order using the relation symbol T, the lexical order[4] $<_k$ over $adom(R)^k$ such that $(x_1, \ldots, x_k) <_k (y_1, \ldots, y_k)$ iff

$$\text{there exists } 1 \leqslant j \leqslant n \text{ satisfying } x_i = y_i \text{ for each } i < j \text{ and } T(x_j, y_j).$$

Observe that $<_k$ is the same as T if $k = 1$. Let $\mathtt{succ}(s,t)$ denote the formula $s \in R \wedge t \in R \wedge s <_k t \wedge \nexists r (r \in R \wedge s <_k r \wedge r <_k t)$. Now, we partition R into two disjoint sets O of "odd" elements and E of "even" elements: (i) The set O contains the smallest tuple t in R according to $<_k$. (ii) For each tuple s in O, the tuple t satisfying $\mathtt{succ}(s,t)$ is in E; and for each tuple s in E, the tuple

[1] We wish to thank Jerzy Tyszkiewicz for suggesting that we consider this query.

[2] A n-ary query Q is simply a generic [CH82, Hul86] mapping from input databases to n-ary relations. A boolean query is a 0-ary query.

[3] This fact can now be obtained as a corollary of Example 2.3 and Theorem 3.1.

[4] In the sequel we shall use T, $<_k$ and $\mathtt{succ}$ defined in this way several times.

t satisfying $\texttt{succ}(s,t)$ is in O. Clearly, R has even cardinality iff its largest element is in E. The maximum arity of the relation variables used, namely T, O and E, is $max(2,k)$. ∎

We now turn to the concept of "foies" for a query Q. Intuitively such a system consists of a finite set of first-order queries, each having as input the old database, the old answer, the old auxiliary database, and the update; for each permissible update[5], the system will use its first-order queries to compute the new answer to the query and the new auxiliary relations. For each relation symbol R, we will use R^o to refer to the instance of R in the database *before* an update, and R^n (or simply R) the instance of R *after* the update.

Definition 2.2 Let Q be a query from database schema $\mathcal{S}_i$ to relation schema R_0. A *k-ary first-order incremental evaluation system (foies)* for Q is a tuple $(\alpha_0^{\pm}, \ldots, \alpha_m^{\pm}, \mathcal{S}_{aux}, \alpha)$ such that

1. m, k are natural numbers, and $\mathcal{S}_{aux} = (R_0, \ldots, R_m)$ is a database schema, called the *auxiliary database schema*, where the arities of $R_1, \ldots, R_m$ are at most k;
2. α is a mapping, called the *aux-mapping*, from $\mathcal{S}_i$ to $\mathcal{S}_{aux}$ such that $\alpha(I)(R_0) = Q(I)$ and $adom(\alpha(I)) \subseteq adom(I)$ for each instance I in $dom(Q)$;
3. $\alpha_0^{\pm}, \ldots, \alpha_m^{\pm}$, called the *aux-maintaining* queries, are first-order queries having $(m+4)$ inputs satisfying the following: for each database I^o in $dom(Q)$, each permissible update $\delta = (\vartriangle, \triangledown)$, and each auxiliary database $(r_0^o, \ldots, r_m^o)$ in $\alpha(I^o)$, if $I^n = \delta I^o$ is the new database, then the following is an auxiliary database in $\alpha(I^n)$:

$$(\alpha_0^{\pm}(I^o, r_0^o, r_1^o, ..., r_m^o, \vartriangle, \triangledown), \ldots, \alpha_m^{\pm}(I^o, r_0^o, r_1^o, ..., r_m^o, \vartriangle, \triangledown))$$

The foies is *space-free* if $m = 0$, and it is an *insertion-only* foies if it can only handle insertions.

The old answer and the old auxiliary relations are initialized by evaluating Q and the aux-mappings on the initial database (such as the empty database). Once initialized, the foies derives new answers and auxiliary relations by computing its first-order queries.

A foies can be alternatively specified by giving the auxiliary database schema, the first-order queries, and $\alpha(\varnothing)$. The auxiliary mapping α can then be determined (theoretically at least) as follows: For each I in $dom(Q)$, we enumerate all possible sequences of permissible updates $\delta_1, \ldots, \delta_n$ such that $\delta_1 \circ \ldots \circ \delta_n(\varnothing) = I$, then execute the foies on all these sequences; $\alpha(I)$ should contain precisely all the instances of the auxiliary database that can be obtained in this way.

Example 2.3 For the query Even_k over the schema $\mathcal{S}_k$ which consists of a k-ary relation symbol R (see Example 2.1), $dom(\mathit{Even}_k)$ is the set of all possible databases. Each permissible update either inserts one tuple and deletes nothing, or it inserts nothing and deletes one tuple. Even_k has a space-free foies $(\mathit{Even}_k^{\pm}, \mathcal{S}_{aux}, \mathit{Even}_k)$ constructed below, where $\mathcal{S}_{aux}$ consists of a relation name E with arity 0 (for containing the answer to the query).

If an update inserts a tuple t ($\vartriangle = \{R(t)\}$ and $\triangledown = \varnothing$), the new answer $\mathit{Even}_k(I^n)$ is *true* iff (i) t was not in the old relation and the old answer was *false*, or (ii) t was already in the old relation and the old answer was *true* (note that $E^o = \mathit{Even}_k(I^o)$):

$$\mathit{Even}_k^{\pm}(I^o, E^o, \{t\}, \varnothing) = \left\{ \mathit{true} \;\middle|\; \neg R^o(t) \wedge E^o{=}\mathit{false} \bigvee R^o(t) \wedge E^o{=}\mathit{true} \right\}.$$

[5]Permissible updates are those updates whose sizes are bounded by a constant dependent only on the query Q and which transforms the old database in $dom(Q)$ to a new database in $dom(Q)$.

Similarly, if an update deletes t ($\Delta = \varnothing$ and $\nabla = \{R(t)\}$), $\mathit{Even}_k(I^n)$ is *true* iff (i) t was in the old relation and the old answer was *false*, or (ii) t was not in the old relation and the old answer was *true*:

$$\mathit{Even}_k^{\pm}(I^o, E^o, \varnothing, \{t\}) = \left\{ \mathit{true} \;\middle|\; R^o(t) \wedge E^o{=}\mathit{false} \bigvee \neg R^o(t) \wedge E^o{=}\mathit{true} \right\}. \qquad \blacksquare$$

Starting from [DS95b], the maximum arity of the auxiliary relations has been used as a measure of how hard it is to maintain a query using foies. Observe that, with maximal arity k, the auxiliary relations can hold at most $O(n^k)$ tuples, where n is the number of constants in the input database.

Definition 2.4 For each natural number k, let FOIES_k be the class of queries having k-ary foies; let FOIES_ϵ be the class of queries having space-free foies; and let $\text{FOIES} = \text{FOIES}_\epsilon \cup \bigcup_k \text{FOIES}_k$.

Obviously, $\text{FOIES}_\epsilon \subseteq \text{FOIES}_0$ and $\forall k \geqslant 0, \text{FOIES}_k \subseteq \text{FOIES}_{k+1}$. In [DS95b] it was shown that $\text{FOIES}_\epsilon \subset \text{FOIES}_0 \subset \text{FOIES}_1 \subset \text{FOIES}_2$, but it is open whether $\text{FOIES}_k \subset \text{FOIES}_{k+1}$ for all $k > 1$.

To illustrate the arity measure and to give more interesting examples of queries maintainable by foies, we now discuss some previous results on the transitive closure of graphs of various kinds. In [DT92, DST95] some insertion-only binary foies were given for generalized transitive closure of labelled graphs. For the transitive closure of acyclic directed graphs, [DS93, DS95a] gave a space-free foies. For undirected graphs, there is a ternary foies [PI94] which maintains an undirected spanning forest for the undirected graph, from which the reachability relation can be extracted. (In [PI94] a foies is called a DynFO, meaning dynamic first order.) There is also a binary foies [DS95b] which maintains a directed spanning forest of the undirected graphs, plus some approximation of a total order on the nodes in the graph; it was shown [DS95b] that there is no monadic foies for this query. It is still open if there is a foies for the transitive closure of arbitrary directed graphs.

3 From FOIES to Σ_1^1

Theorem 3.1 Suppose Q is a boolean query over k-ary input relations. Then the following hold:

1. Q is in $(k{+}1)$-ary Σ_1^1 whenever it has a foies.
2. Q is in k-ary Σ_1^1 whenever it has a 0-ary foies.
3. For each $k \geqslant 1$, there exists a query Q (e.g. EVEN_k) which has a space-free foies, but which is not in $\mathit{max}(1, k{-}1)$-ary Σ_1^1.

Proof: To prove (1), suppose we are given a foies for Q. Let $R_1, \ldots, R_q$ be an enumeration of all the input relation names. By padding with a fixed constant if necessary, we can assume that each R_i either has arity 0 or arity k.

Suppose we are given a particular database instance, i.e., each relation name is now associated with a k-ary relation. Let T be a total order on the active domain $\mathcal{A}$ of the database, and let $<_k$ be the lexical order on $\mathcal{A}^k$ (see Example 2.1). Let `succ` be as defined in Example 2.1, except each expression of the form $x \in R$ is now replaced by $\bigvee_1^q (x \in R_i)$. More specifically, let $\mathtt{succ}(s,t)$ denote the formula

$$(\bigvee_1^q s \in R_i) \wedge (\bigvee_1^q t \in R_i) \wedge s <_k t \wedge \nexists r((\bigvee_1^q r \in R_i) \wedge s <_k r \wedge r <_k t).$$

Let $t_1, \ldots, t_n$ be the sequence such that (a) $\texttt{succ}(t_i, t_{i+1})$ for each $1 \leqslant i < n$, and (b) there is no s satisfying either $\texttt{succ}(s, t_1)$ or $\texttt{succ}(t_n, s)$. For each i, let $R_{j^i_1}, \ldots, R_{j^i_{p_i}}$ enumerate the input relation names R such that $t_i \in R$ and $j^i_1 < j^i_2 < \ldots < j^i_{p_i}$. The $(k{+}1)$-ary Σ^1_1 sentence will simulate the foies for the following insertion sequence

$$R_{j^1_1}(t_1), \ldots, R_{j^1_{p_1}}(t_1), \ldots, R_{j^i_1}(t_i), \ldots, R_{j^i_{p_i}}(t_i), \ldots, R_{j^n_1}(t_n), \ldots, R_{j^n_{p_n}}(t_n)$$

where, for example, $R_{j^i_1}(t_i)$ says that t_i is inserted to $R_{j^i_1}$. So $R_i(t)$ is an element in the sequence iff $t \in R_i$ in the given database instance; and the position of $R_i(t)$ in the sequence can be defined in terms of $\texttt{succ}$ and the ordering of the relation names in the enumeration $R_1, \ldots, R_q$.

For each m-ary auxiliary relation A $(m > 0)$ used by the given k-ary foies, we use $q \times m$ relations $A^j_1, ..., A^j_m$, each having arity $k+1$. For each tuple t, we use $A^j_1(t, y_1), ..., A^j_m(t, y_m)$ to say that $(y_1, ..., y_m)$ is a tuple in A at the database state immediately after t is inserted into R_j according to the above insertion sequence. For A we also have m relations $A_1, \ldots, A_m$ each having arity k; these A_i's will be used to hold the value of A of the aux-mapping on the empty database (and thus they should all be empty by the definition of aux-mappings in foies).

For each 0-ary relation A in the foies, we use q k-ary relation A^j_1 such that $A^j_1(t)$ is true iff A is true at the database state immediately after t is inserted into R_j. For this A we also have a relation A_1 having arity 0; A_1 will be used to hold the value of A of the aux-mapping on the empty database, which is either $\varnothing$ or $\{[]\}$.

The $(k{+}1)$-ary Σ^1_1 formula should say: for each position in the insertion sequence, the corresponding content of each auxiliary relation is indeed derived according to the foies. So it needs to say that the foies starts correctly and proceeds correctly. Such formulas can be constructed in a straightforward way. So (1) is proven.

Statement (2) follows because the relation variables used are all k-ary when the foies only uses boolean auxiliary relations. Statement (3) follows by Example 2.3, [Ajt83] and [Fag75]. ■

4 EqC_k in the FOIES Hierarchy

Theorem 4.1 For all k, EqC_k has a binary foies but it has no 0-ary foies.

The key idea in the proof is a new order over $adom^k$, which is more "stable" than the lexical order when constants disappear because of tuple deletions.

Proof: The fact that EqC_k does not have 0-ary foies was proven in [DS95b].

We now show that EqC_k has a binary foies. Let R_1, R_2 be the two k-ary relations to be compared. Let $\mathcal{A}$ denote $adom(R_1 \cup R_2)$. As in Example 2.1, we will have a total order T on $\mathcal{A}$, and the lexical order $<_k$ on $\mathcal{A}^k$ based on T.

The foies will use a new order on $\mathcal{A}^k$, which we call $<_M$, obtained by making some changes to $<_k$ so that $<_M$ is more attentive to larger constants. This makes $<_M$ more "stable" when a constant disappears. Intuitively, in $<_M$ the larger a constant is the later it appears. In fact, the first[6] $(|\mathcal{A}| - 1)^k$ tuples of $<_M$ do not contain the largest constant in T!

[6]We use $|X|$ to denote the number of elements in the set X.

The purpose of having $<_M$ is to remember how many tuples are extra in R_1 (respectively, in R_2) than in R_2 (respectively, in R_1). It will be convenient (for the discussion) to refer to each tuple in $\mathcal{A}^k$ as a positive integer; note however this convenience is never used in the formulae. We will have $2 \times k$ monadic relations $A_1, \ldots, A_k$ and $B_1, \ldots, B_k$ so that, if $t = (t_1, \ldots, t_k)$ satisfies $t_i \in A_i$ for each $1 \leqslant i \leqslant k$ then $|R_1| = |R_2| + t$; and if $t = (t_1, \ldots, t_k)$ satisfies $t_i \in B_i$ for each $1 \leqslant i \leqslant k$ then $|R_2| = |R_1| + t$. At any time there will be at most one such t.

Given s and t are different tuples from $\mathcal{A}^k$, we say $s <_M t$ if (assuming s^0 and t^0 are the largest elements of s and t respectively):

- $(s^0, t^0) \in T$, or
- $s^0 = t^0$, and the j^{th} value of s is less than the j^{th} value of t according to T, where j is the rightmost position where s and t differ.

It is easy to verify that $<_M$ is a total order on $\mathcal{A}^k$.

For example, suppose $T = \{(1,2),(2,3)\}$. Then

- $(1,2,2,2) <_M (2,1,3,2)$, because the largest element in $(2,1,3,2)$, 3, is larger.
- $(3,3,1,2) <_M (2,3,2,2)$, because they have the same largest elements, and $(2,3,2,2)$ is larger at the rightmost position where they differ (i.e., 1 is less than 2).

If a new constant is added to the active domain because of an insertion we simply make it the new largest element of T.

Suppose after the deletion of a tuple a constant c disappears from the active domain. We simply exchange it with the old largest constant c' (of T) first, and then do the minor adjustment to A_i's and B_i's if necessary. (This exchange is a permutation on the active domain, and we should simply change the A_i's and B_j's by applying this permutation.)

Apart from the difficulty that a constant may disappear from the active domain, the other computations by the foies is straightforward: For example, if $|R_1| = |R_2| + t$ before we delete a tuple from R_1, then after the deletion, we should have (i) $|R_1| = |R_2|$ when t was the first element according to $<_M$, and (ii) $|R_1| = |R_2| + s$ when s was the predecessor of t in $<_M$.

Now, EqC_k is true on R_1 and R_2 iff A_1 is empty and B_1 is empty. So EqC_k is in FOIES$_2$! ■

5 EqC_k in the Σ^1_1 Hierarchy

Theorem 5.1 EqC_k is in $(k+1)$-ary Σ^1_1 but not in $max(1, k-1)$-ary Σ^1_1.

Proof: The fact that "EqC_k is in $(k+1)$-ary Σ^1_1" follows by Theorems 4.1 and 3.1.

For the statement that EqC_k is not in $(k-1)$-ary Σ^1_1, the key idea is to show that, if EqC_k is in $(k-1)$-ary Σ^1_1 then $EVEN_k$ is in $(k-1)$-ary Σ^1_1. (The crux of the reduction is from [Ajtai, private communication].) Assume that EqC_k is in $(k-1)$-ary Σ^1_1, and is defined by the $(k-1)$-ary Σ^1_1 formula $\exists A_1 \ldots A_m \phi(R_1, R_2)$.

We define $EVEN_k$ in $(k-1)$-ary Σ_1^1 as follows. Intuitively, the formula says that $EVEN_k(R)$ is true iff R has the same cardinality with an initial segment of the lexical order, and the last element of the segment is even:

$$\exists A_1 \dots A_m T_2' E_2' T_2 E_2 \exists z_1 \dots z_k (\phi(R, \psi(x_1, ..., x_k, z_1, \dots, z_k)) \wedge \tau \wedge E_2(z_k))$$

By $\phi(R, \psi(x_1, ..., x_k, z_1, \dots, z_k))$ we mean the formula obtained by replacing each occurrence of $R_2(t_1, ..., t_k)$ with $\psi(t_1, ..., t_k, z_1, \dots, z_k)$; $\psi(x_1, ..., x_k, \vec{w})$ is $(x_1, \dots, x_k) <_k \vec{w} \vee (x_1, \dots, x_k) = \vec{w}$; and τ says:

- T_2' is a binary relation containing a total order on $adom(R)$,
- E_2' is a binary relation containing the set of even elements of $adom(R)$ in the order of T_2',
- T_2 is T_2' if the largest element of T_2' is in E_2', and otherwise $T_2 = T_2' \cup \{(b, a)\}$, where b is the largest element in T_2' and a is a fixed constant (which will not occur in the input relations),
- and E_2 is E_2' if the largest element of T_2' is in E_2', and otherwise $E_2 = \{a\} \cup E_2'$.

Observe that we added a fixed constant a to $adom(R)$ when $adom(R)$ contains an odd number of elements. The lexical order $<_k$ is in terms of T_2. Thus $E_2(z_k)$ says that $(z_1, \dots, z_k)$ is even in $<_k$.

To prove that EqC_1 is not in monadic Σ_1^1, we use the same idea: If EqC_1 is in monadic Σ_1^1, then $EVEN_1$ would be in monadic Σ_1^1 too. ∎

Even though it is open if EqC_k is in k-ary Σ_1^1, the following can be shown, where a k-ary relation R is called *sparse* if $|R| \leqslant |adom(R)|^{k-1}$.

Proposition 5.2 In k-ary Σ_1^1 we can test the following:

1. Do a k-ary relation R_1 and a sparse k-ary relation R_2 have the same cardinality?
2. Do a k-ary relation and a $(k-1)$-ary relation have the same cardinality?

Proof: It suffices to prove (1), since a (k–1)-ary relation can be viewed as a sparse k-ary relation (by padding). To prove (1), we construct a k-ary Σ_1^1 sentence to do the following. Let $T_{21}, \dots, T_{2k}$ be k of k-ary relations which encode a mapping from $adom(R_2)^{k-1}$ to R_2: $(y_1, \dots, y_{k-1}) \in adom(R_2)^{k-1}$ is mapped to $(x_1, \dots, x_k) \in R_2$ iff $(y_1, \dots, y_{k-1}, x_1) \in T_{21}$, ..., $(y_1, \dots, y_{k-1}, x_k) \in T_{2k}$. Clearly, R_2 is sparse if the above mapping is one-to-one and onto.

We can use another k of k-ary relations to define a mapping from $adom(R_2)^{k-1}$ to R_1. The answer to the query is true iff there is a one-to-one onto mapping from $adom(R_2)^{k-1}$ to R_2 and there is a one-to-one onto mapping from $adom(R_2)^{k-1}$ to R_1 and the two one-to-one mappings are defined over exactly the same tuples. ∎

Acknowledgement: We wish to thank Miklos Ajtai, Leonid Libkin, Jianwen Su and Jerzy Tyszkiewicz for helpful discussions.

References

[Ajt83] M. Ajtai. Σ_1^1-formulae of finite structures. *Annals of Pure and Applied Logic*, 24:1–48, 1983.

[CH82] A. K. Chandra and D. Harel. Structure and complexity of relational queries. *Journal of Computer and System Sciences*, 25(1):99–128, 1982.

[DS93] G. Dong and J. Su. Incremental and decremental evaluation of transitive closure queries by first-order queries (extended abstract). In *Proc. 16th Australian Computer Science Conference*, 1993. Full version appears in [DS95a].

[DS95a] G. Dong and J. Su. Incremental and decremental evaluation of transitive closure by first-order queries. *Information and Computation*, 120(1):101–106, July 1995. Preliminary version in [DS93].

[DS95b] G. Dong and J. Su. Space bounded FOIES. In *Proc. of the ACM Symposium on Principles of Database Systems*, pages 139–150, May 1995. Full paper has been invited to J. of Computer and Systems Science.

[DS96] G. Dong and J. Su. Deterministic FOIES are strictly weaker. Technical Report 96/3, Computer Science Department, University of Melbourne, 1996. First presented in *Workshop on Nondeterminism in Databases, Prague*, Jan 1995. Tentatively accepted by AMAI.

[DST95] G. Dong, J. Su, and R. Topor. Nonrecursive incremental evaluation of datalog queries. *Annals of Mathematics and Artificial Intelligence*, 14(2-4):187–223, 1995.

[DT92] G. Dong and R. Topor. Incremental evaluation of datalog queries. In *Proc. Int'l Conference on Database Theory*, pages 282–296, Berlin, Germany, October 1992. Full version appears as part of [DST95].

[Fag75] R. Fagin. Monadic generalized spectra. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 21:89–96, 1975.

[Hul86] R. Hull. Relative information capacity of simple relational schemata. *SIAM Journal on Computing*, 15(3):856–886, 1986.

[PI94] Sushant Patnaik and Neil Immerman. Dyn-FO: A parallel dynamic complexity class. In *Proc. ACM Symp. on Principles of Database Systems*, pages 210–221, 1994.