A Semantic Characterization of Relational Machines and the Effects of Homogeneity

JOSE MARIA TURULL TORRES[1] [2]

Universidad Nacional de San Luis
Fac. Cs. Fís-Matemáticas - Depto. de Informática

Universidad Tecnológica Nacional
Fac. Regional Buenos Aires - Depto. de Sistemas

Peron 2315, piso 4, depto. P
1040 Capital Federal, ARGENTINA

email: turull@iamba.edu.ar turull@unsl.edu.ar

Abstract

For each $k > 0$, we define the class $\mathcal{QCQ}^k$ as the total and typed queries in $\mathcal{CQ}$ which preserve realization of FO^k types. With these classes we give a semantic characterization of the class of total and typed queries computable by k-ary Relational Machines. Then we prove that this hierarchy is strict, and it follows that it is orthogonal with the usual classification of computable queries in complexity classes. On the other hand we define four different notions related to homogeneity: *k-homogeneity*, *strong k-homogeneity*, *$(f(n), k)$-homogeneity* and *pairwise k-homogeneity*, which are similar to some properties defined in [Set,95]. We prove that the computability power of k-ary Relational Machines is much bigger when working with classes of structures which are homogeneous, for three of these notions. As to the fourth one, we prove that the computability power of Reflective Machines with sub-linear variable complexity also increases when working on databases which satisfy that property. The strongest notion, pairwise k-homogeneity, allows k-ary Relational Machines to achieve completeness, while k-homogeneity and strong k-homogeneity allows this model to skip out of the whole hierarchy $\mathcal{QCQ}^\omega$. Next, we define the class $\mathcal{QCQ}$ as the subclass of queries in *total-typed*$(\mathcal{CQ})$ which preserve realization of automorphism types. And we prove that this class strictly contains the whole hierarchy $\mathcal{QCQ}^\omega$. Finally, we show that, roughly, k-ary Relational Machines working with classes of structures which are k-homogeneous lie somewhere between $\mathcal{QCQ}^\omega$ and $\mathcal{QCQ}$ as to computability power.

[1]The work presented here was done during a visit to the Department of Mathematics of the University of Helsinki.

[2]This work has been partially supported by Grant 1011049 of the Academy of Finland and by a Grant of Universidad CAECE, Argentina.

Section 1: Introduction.

The GM^{loose} machine ([AV,91]), or *Relational Machine*, is a suitable model for computing typed queries in the class $\mathcal{CQ}$ of *computable queries* ([CH,80]), on classes of finite and relational structures, or databases. It was noted there that the class of queries computable by this machine is exactly the same class known as WHILE$^+$. This is the RQL programming language of [CH,82] with the addition of the set of natural numbers as a secondary domain with arithmetic restricted to it. This is a syntactic characterization of the computation model GM^{loose} for all possible arities. As to the authors knowledge it is not known at present neither a semantic characterization of it, nor a syntactic characterization for the sub-class of the machines of each particular arity. We will denote the class of k-ary GM^{loose} machines as $k\text{-}GM^{loose}$.

In the first part of the present article we give a semantic characterization for the sub-classes $total(k\text{-}GM^{loose})$, for every k, as well as for the whole class $total(GM^{loose})$. By $total$ we mean the restriction of the corresponding class to total queries. We do this by defining, for every $k \geq 1$, the class $\mathcal{QCQ}^k \subset total\text{-}typed(\mathcal{CQ})$, as the total and typed queries which preserve FO^k types realization (see [Tur,97] for a study of *untyped* queries). And we show that $\mathcal{QCQ}^k = total(k\text{-}GM^{loose})$. We show that these classes form a strict hierarchy inside $total\text{-}typed(\mathcal{CQ})$. We denote the whole hierarchy as $\mathcal{QCQ}^\omega$, and this class gives us the semantic characterization for $total(GM^{loose})$.

The properties of preservation of the realization of types in FO (First Order Logic) with different bounds in the number of variables for queries are quite natural, since in the definition of *computable query* ([CH,80]) the very property of preservation of isomorphisms is crucial. And, as it is well known, on finite structures isomorphism is equivalent to equivalence in realization of FO types. So, when we define different restrictions in the types, as to the number of variables which may be used for their definition, what we are actually doing is defining different levels as to the amount of information on the structure which every query really needs.

An interesting characteristic of the hierarchy of the classes $\mathcal{QCQ}^k$, for $k \geq 1$, is the fact that it is orthogonal with the usual classification of the computable queries in complexity classes. That is, in every level in the hierarchy there are queries of *any* complexity class in the usual classification of logarithmic time and space, polynomial time and space, exponential time and space, etc.

As there are many classes of structures for which no bound in the amount of variables is enough to define the automorphism types for tuples for every structure in the class, we also define a sub-class of total and typed queries which we denote as $\mathcal{QCQ}$. Queries in this class are defined in such a way that they preserve realization of types built with the number of variables which are actually needed for each structure to define the automorphism types for their tuples. And we prove that this class strictly includes the hierarchy $\mathcal{QCQ}^\omega$.

So that we have the following picture inside $total\text{-}typed(\mathcal{CQ})$, where $\subset$ denotes proper inclusion:

$$\ldots \mathcal{QCQ}^k \subset \mathcal{QCQ}^{k+1} \subset \ldots \subset \mathcal{QCQ}^\omega \subset \mathcal{QCQ} \subset \textit{total-typed}(\mathcal{CQ})$$

In the second part of this article we define four notions related to homogeneity, and we study them as properties of classes of structures, or databases, which allow incomplete relational machines to increase their computability power when working on this kind of classes. These notions are similar to some properties defined in [Set,95], where they are studied in relation to the expressibility of fixed point logics on classes of structures which satisfy them. In Remark 3.2 we analyze these notions of Seth and we relate them to ours. We also include three results from Seth's work which are meaningful in the context of our research.

The main question to this regard is, for a given structure and for a given integer k, how many variables do we need to define the types for k-tuples of the structure up to automorphism?

In [Daw,93] and [Tur,95] bounded rigidity in classes of finite and relational structures, or databses, was studied as a property which allows incomplete relational machines to achieve completeness. In the second work, unbounded rigidity was also studied with the same purpose, considering incomplete reflective machines ([APV,94]). Finally, in [Tur,96] bounded and unbounded *partial* rigidity was studied also as a means to increase the computability power of incomplete relational and reflective machines.

That is, there is a main point leading to a research program, which is finding properties of databases which are "good properties", in the sense that a query language, or any kind of suitable formalism for this matter, which is incomplete, in the sense that it cannot compute or express every computable query for all the databases of the corresponding schema (or for all the classes of structures of the corresponding signature), can benefit from these properties and increase its computation power when working on databases which satisfy them.

Up to now, order, rigidity, either bounded or unbounded, and partial rigidity, also either bounded or unbounded, are properties of databases which are known to be quite relevant as to computability power. The results presented here show that different notions related to homogeneity are also quite relevant in the same way.

Then, and following the same research program, we study different notions which are related to the notion of homogeneity of classical Model Theory (see [CK,92] among others). We call them *k-homogeneity*, *strong k-homogeneity* and *pairwise k-homogeneity*. Roughly, a structure is k-homogeneous if k variables suffice to define the automorphism types for their k-tuples. We prove that, for every k, $total(k\text{-}GM^{loose})$ working on such classes of structures augment their computability power to such an extent that, in a sense which will be clarified later (Corollary 4.1), they go through the whole hierarchy $\mathcal{QCQ}^\omega$ and fall somewhere between $\mathcal{QCQ}^\omega$ and $\mathcal{QCQ}$

(recall that we said above that we proved that the class of these machines is characterized by $\mathcal{QCQ}^k$). A structure is strongly k-homogeneous if it is r-homogeneous for every $r \geq k$. Here we show that, roughly, for every $r > k \geq 1$, the class of queries computable by $total(r\text{-}GM^{loose})$ working on such classes of structures strictly includes the class of queries computable by the same machines on classes of structures which are k-homogeneous but which are not strongly k-homogeneous. And as to the third notion, we say that a class of structures is pairwise k-homogeneous if for every pair of structures in the class, and for every pair of k-tuples $\bar{a}$ and $\bar{b}$ taken respectively from the two structures, if both k-tuples have the same FO^k type, then the two structures are isomorphic and there is an isomorphism mapping one tuple onto the other. We show that, for every k, $k\text{-}GM^{loose}$ working on such classes achieve completeness, in a sense which will be clarified later (Lemma 3.2), provided the classes are recursive. Finally, we consider reflective machines with bounded variable complexity. For this case we define a notion of homogeneity for which the number of variables which we may use for the types for k-tuples is given by a function on the size of the structure. We denote this notion as $(f(n), k)$-homogeneity. And we prove that, roughly, for every sub-linear function f defined over the naturals, the class of queries which can be computed by $GM^{f(n)}$ machines (which are known to be incomplete, [APV,94]) on classes of structures which are $(f(n), k)$-homogeneous strictly includes the class of queries which can be computed by $GM^{f(n)}$ machines on classes of arbitrary structures.

So, we conclude to this regard that the different notions related to homogeneity in classes of structures are all relevant regarding computability power. The first three notions allow an important increase in the computability power of incomplete relational machines, and the fourth notion allows an important increase in the computability power of incomplete reflective machines.

Section 2: A Characterization of the Total Relational Machines.

In the present article we will deal only with finite structures whose signature is finite and relational. Let σ be a signature, and let I be a σ-structure. We will denote the domain of I as $dom(I)$. For any $k > 0$, we will denote a k-tuple of I as $\bar{a}_k$. This is not to be confused with the k-th component of $\bar{a}_k$, which we will usually denote as a_k. If σ is a finite relational signature, we will denote the set of all finite σ-structures as $\mathcal{E}_{\sigma,fin}$.

Unless otherwise stated, in the present article we will follow the usual notation in Finite Model Theory, as in [EF,95].

Recall that, given a σ-structure I, an FO^k *type* for an l-tuple $\bar{a}_l$ of I, with $l \leq k$, is the set of σ-formulae in FO^k with l free variables, such that every formula in the set is TRUE when interpreted by I for any valuation which assigns the i-th component of $\bar{a}_l$ to the i-th free variable in the formula, for every $1 \leq i \leq l$. We will say that a structure I *realizes* a type ϕ for k-tuples, if there exists a k-tuple in $dom(I)$ whose type is ϕ.

Definition 2.1:
Let I_1 and I_2 be two σ-structures for some signature σ, and let $k \geq l \geq 1$. We will say that I_1 and I_2 *agree on* FO^k *types for* l*-tuples*, if they both realize the same set of FO^k types for l-tuples. We will denote this (equivalence) relation as $\equiv_{\tau_{k,l}}$. In case that it is $l = k$, we will denote the relation as $\equiv_{\tau_k}$.

Note that, for every $k \geq l \geq 1$, and for every pair I_1, I_2 of structures of the same signature, the following holds:

$$I_1 \equiv_{FO^k} I_2 \qquad \Longleftrightarrow \qquad I_1 \equiv_{\tau_k} I_2 \qquad \Longleftrightarrow \qquad I_1 \equiv_{\tau_{k,l}} I_2$$

In [Daw,93] it is shown that, for every $k \geq 1$, every FO^k type for k-tuples can be *isolated* through an FO^k formula with k free variables. That is, a single formula defines the same FO^k type for k-tuples on *every* structure of the same signature. We will call such a formula the *isolating formula* for the corresponding FO^k type for k-tuples. Dawar builds these formulae through an inductive definition, and at every stage of the induction a formula which characterizes a sub-type with a bounded quantifier rank is obtained. Then he shows that for a structure of size n, n^k steps in the induction suffice to build the isolating formula for the corresponding type. An important observation which follows from the way in which these formulae are built, is that the FO^k type for a k-tuple is given by the *atomic* type of the k-tuple and by each of the FO^k types of its sub $(k-1)$-tuples. An atomic type is the conjunction of the atomic formulae which the tuple satisfy, and of the negation of the atomic formulae which the tuple does not satisfy.

Let I_1 and I_2 be two σ-structures for some signature σ, and let $k \geq 1$. We will say that I_1 and I_2 are FO^k *equivalent* if they satisfy the same set of FO^k sentences. We will denote this (equivalence) relation as $\equiv_{FO^k}$.

From the aforementioned results of [Daw,93] the two following observations are straightforward:

Fact 2.1: For every $k \geq l \geq 1$, and for every pair I_1, I_2 of structures of the same signature, the following holds:

$$I_1 \equiv_{\tau_{k,l}} I_2 \qquad \Longrightarrow \qquad I_1 \equiv_{\tau_l} I_2$$

Fact 2.2: For every $k \geq 1$, for every $k \geq l \geq 1$ and for every pair I_1, I_2 of structures of the same signature, if $I_1 \equiv_{\tau_k} I_2$, then $I_1 \equiv_{\tau_{k,l}} I_2$.

Let's denote by $\equiv_k$ the (equivalence) relation induced in the set of k-tuples of a given structure I, by the equality in the FO^k types of the k-tuples. That is, for every $\bar{a}_k$ and $\bar{b}_k$ in $(dom(I))^k$ $\bar{a}_k \equiv_k \bar{b}_k$ iff $\bar{a}_k$ and $\bar{b}_k$ have the same FO^k type for k-tuples.

As it is pointed out in [AV,91], a given GM^{loose} M doesn't necessarily refine the set of k-tuples of its input structure up to $\equiv_k$, because it has a limited discerning power which is due to the finite set of FO^k formulae of its finite control. We will denote as $\equiv_{k,M}$ the equivalence relation on k-tuples which the GM^{loose} machine M can actually distinguish. Clearly, for every input structure, every equivalence class in $\equiv_{k,M}$ is a union of equivalence classes in $\equiv_k$.

The following two Facts follow from [AV,91].

Fact 2.3: For every structure I of the corresponding signature, the result of an r-ary query which is computed by a k-GM^{loose} M on I is a union of equivalence classes in the relation $\equiv_{k,M}$ on I, reduced to r-tuples.

Fact 2.4: For every $1 \leq r \leq k$, and for every structure I of the corresponding signature, equivalence classes in $\equiv_r$ on I are unions of equivalence classes in $\equiv_k$ on I, reduced to r-tuples.

Definition 2.2:

- For every $k \geq 1$, we define the class $\mathcal{QCQ}^k$ as the class of l-ary queries $f \in$ *total-typed*$(\mathcal{CQ})$, of some signature σ and for some $l \leq k$, such that for every pair of structures I_1, I_2 in $\mathcal{E}_{\sigma,fin}$, if $I_1 \equiv_{\tau_k} I_2$ then $f(I_1) \equiv_{\tau_{k,l}} f(I_2)$.
- $\mathcal{QCQ}^\omega = \bigcup_{k\in\omega} \mathcal{QCQ}^k$.

That is, queries in $\mathcal{QCQ}^k$ *preserve realization of FO^k types.*

Remark 2.1: Let $f \in$ *total-typed*$(\mathcal{CQ})$, of signature σ. Note that, for every $k \geq 1$ and for every $i \in \omega$, if f preserves realization of FO^k types for k-tuples on every pair of structures in $\mathcal{E}_{\sigma,fin}$ then f also preserves realization of FO^{k+i} types for $(k+i)$-tuples on every pair of structures in $\mathcal{E}_{\sigma,fin}$. This follows from the very definition of preservation of realization of types by a function, and it can be also derived as a Corollary to Lemma 2.1.

Next, we will give a semantic characterization of the class of total typed queries computable by a total k-GM^{loose} machine.

Theorem 2.1: *For every $k \geq 1$, the class of total queries which are computable by a k-GM^{loose} machine is exactly the class $\mathcal{QCQ}^k$.*

Proof:

a) ($\subseteq$): Suppose that a total r-ary query f of signature σ is computable by a k-GM^{loose} M, for some $k \geq r$. And let I_1 and I_2 be two σ-structures such that $I_1 \equiv_{\tau_k} I_2$. The only way to assign a relation to an r-ary relation symbol in the relational store of a k-GM^{loose} is through a FO^k formula with r free variables, say φ. And, by Fact 2.3, the result of the evaluation of φ on the structure in the relational store of M is a union of equivalence classes in $\equiv_{k,M}$, which in turn is a union of equivalence classes in $\equiv_k$. So, it turns out that φ is equivalent to the disjunction of some isolating formulae of FO^k types for r-tuples. But, as we mentioned before in this article, the isolating formulae define the same FO^k types in every structure of the corresponding signature. So that the r-tuples from $dom(I_2)$ which form the relation induced by φ in I_2 must be those with the same FO^k types as the r-tuples from $dom(I_1)$ which form the relation induced by φ in I_1. Then, it follows that $f(I_1) \equiv_{\tau_{k,r}} f(I_2)$ and, hence, $f \in \mathcal{QCQ}^k$. Note that the only way for the machine not to evaluate the same formula φ for both structures is the existence of a FO^k sentence which evaluates to a different truth value on I_1 and I_2. But this is not possible by Fact 2.2 because $I_1 \equiv_{\tau_k} I_2$.

b) ($\supseteq$): Let $f \in \mathcal{QCQ}^k$ be an r-ary query of signature σ. We will build a k-GM^{loose} $\mathcal{M}_f$, which will compute f. We will use dynamic queries in FO^k for $\mathcal{M}_f$, as well as a countably infinite number of k-ary relation symbols in its relational store. In [AVV,92b] it was shown that none of these variations alters the expressive power of GM^{loose}. With the input structure I in its relational store, $\mathcal{M}_f$ will build in its TM tape an encoding of a structure I' which will agree with I on the FO^k types for k-tuples.

For this purpose, $\mathcal{M}_f$ will work as follows:

- For every $n \geq 1$, $\mathcal{M}_f$ builds every possible σ-structure I' of size n in its TM tape, with domain $\{1, \ldots, n\}$.
- For every k-tuple $s_{n,i}$ of $\{1, \ldots, n\}$, $\mathcal{M}_f$ builds in its TM tape the isolating formula $\phi_{s_{n,i}}$ for the FO^k type of $s_{n,i}$, as in [Daw,93]. The inductive construction is followed up to the step n^k.

  In this way we get the isolating formulae for the FO^k types for k-tuples which are realized in I'.
- $\mathcal{M}_f$ evaluates as dynamic queries the formulae $\phi_{s_{n,i}}$, for every i, which are in FO^k, and assigns the results to the working k-ary relation symbols S_i in the relational store, respectively.

- If every relation S_i is non empty, and if the union of all of them is the set of k-tuples of I, then it means that $I_1 \equiv_{\tau_k} I_2$. Then I' is the structure we were looking for.

  Otherwise, we try another structure I'. Note that the set of k-tuples of I can be generated by the dynamic query $\psi(x_1, \ldots, x_k) \equiv x_1 = x_1 \wedge \ldots \wedge x_k = x_k$.

- Now $\mathcal{M}_f$ computes $f(I')$ in its TM tape, which is possible because $f \in \mathcal{CQ}$ and f is defined on I' because it is total by definition. Then the machine expands the r-ary relation $f(I')$ to a k-ary relation $f^k(I')$, with cartesian product.

- $\mathcal{M}_f$ builds $f^k(I)$ in the relational store as the union of the relations S_i which correspond to the FO^k types $\phi_{s_{n,i}}$ which are realized in $f^k(I')$. Finally it reduces the k-ary relation $f^k(I)$ to an r-ary relation $f(I)$, by projecting out the components $(r+1)$-th trough k-th in every k-tuple.

$\blacksquare$

The next result follows immediately from Theorem 2.1.

Corollary 2.1: $\mathcal{QCQ}^\omega = total(GM^{loose})$. $\blacksquare$

Now we will show that the classes $\mathcal{QCQ}^k$ form a strict hierarchy inside *total-typed*($\mathcal{CQ}$). In [APV,94] it is shown that reflective machines with sub-linear variable complexity form a strict hierarchy, according to the function which bounds the number of variables in dynamic queries depending on the size of the input structure. Their result is related, to some extent, to our next result, by Theorem 2.1.

Lemma 2.1: *For every* $k > 0$, $\mathcal{QCQ}^{k+1} \supset \mathcal{QCQ}^k$.

Proof:

a) ($\supseteq$): We can use Theorem 2.1. Then we know that $total(k\text{-}GM^{loose}) = \mathcal{QCQ}^k$ and $total((k+1)\text{-}GM^{loose}) = \mathcal{QCQ}^{k+1}$.

And clearly $total((k+1)\text{-}GM^{loose}) \supseteq total(k\text{-}GM^{loose})$. For every $total(k\text{-}GM^{loose})$ we can build a $total((k+1)\text{-}GM^{loose})$ which computes the same query, but which uses FO formulae in its finite control with $(k+1)$ variables. We can simply add a dummy variable to every FO^k formula in the original machine, like $\ldots \wedge \exists x_{k+1}(x_{k+1} = x_{k+1})$.

b) ($\neq$): Let's consider the signature of the graphs. We will define a query f such that $f \in \mathcal{QCQ}^{k+1}$, but $f \notin \mathcal{QCQ}^k$.

Define f as *"nodes in* $(k+1)$*-subcliques"*. Clearly, $f \in \mathcal{QCQ}^{k+1}$, because in FO^{k+1} we can say that a node belongs to a clique of size $(k+1)$. So it is not possible that two structures, say I_1 and I_2, agree on FO^{k+1} types for $(k+1)$-tuples while $f(I_1)$ and $f(I_2)$ don't.

Now, let's consider two graphs which are both the disjoint union of two cliques. The two cliques in I_1 have size k, while the two cliques in I_2 are of sizes k and $(k+1)$, respectively. It is easy to see with a k-pebble game, that every FO^k type which is realized in one clique is also realized in the other. So, I_1 and I_2 agree on FO^k types for k-tuples. But $f(I_1)$ is empty, while $f(I_2)$ has all the elements in the larger clique. So, f does not preserve realization of FO^k types and, hence, is not in $\mathcal{QCQ}^k$. ∎

Remark 2.2: The hierarchy defined by the classes $\mathcal{QCQ}^k$ is not only *strict*, but it is orthogonal with respect to complexity classes. Note that *any* recursive predicate evaluated on the number of equivalence classes in $\equiv_k$ is in $\mathcal{QCQ}^k$. So that there is no complexity class which include $\mathcal{QCQ}^k$. And this is the case for every $k > 1$.

Intuitively, what these hierarchy classify is how careful needs the structure to be analyzed for the query to be evaluated. The lower is the parameter k, the *less information* on the structure is needed for the query to be evaluated on it.

Remark 2.3: In [AV,91] it is stated that $total(GM^{loose}) \subset \mathcal{L}^{\omega}_{\omega_1,\omega}$, where $\mathcal{L}^{\omega}_{\omega_1,\omega}$ is the infinitary logic which allows conjunctions and disjunctions over countably infinite sets of formulae, while restricting the number of variables in every formula to be finite. It is also noted there that, as this logic has the so called 0–1 Law, then it follows that $total(GM^{loose})$ has also the 0–1 Law. This means that the asymptotic probability of every property which can be expressed in the formalism always exist, and is 0 or 1 (see [KV,90]). Then, from Theorem 2.1 and Corollary 2.1, it follows that the classes $\mathcal{QCQ}^{\omega}$ and $\mathcal{QCQ}^k$, for every $k \geq 1$, also have the 0–1 Law. It is well known that there are computable queries which don't have this Law, like the parity of the size of the domain of the structure, by case. So, the next result follows immediately from this property.

Corollary 2.2: $\mathcal{QCQ}^{\omega} \subset total\text{-}typed(\mathcal{CQ})$. ∎

Section 3: Homogeneous Structures.

From Fact 2.4, it follows that as we allow for more variables to be used in writing the isolating formulae for types of tuples, we get a more precise view of the properties which identify a given set of tuples among the whole set in the domain of the structure. The limit is FO, in which logic we can express the automorphism types for tuples of every length. But FO has no limit in the number of variables which can be used in a formula. So, we ask the natural question arising from this observation: how many variables are needed for a given structure and for a given integer k, to express the properties of k-tuples in that structure up to automorphism?[3]

In this Section we define different notions regarding what we define as *homogeneity* for its close relation with the classical concept in Model Theory (see [CK,92] among others).

For every $k \geq 1$, and for any two k-tuples $\bar{a}_k$ and $\bar{b}_k$ of a given structure I, we will denote as $\equiv_{\simeq}$ the (equivalence) relation defined by the existence of an automorphism mapping one tuple onto the other. That is, $\bar{a}_k \equiv_{\simeq} \bar{b}_k$ iff there exists an automorphism f on I such that, for every $1 \leq i \leq k$ is $f(a_i) = b_i$. Note that, according to what we said above about FO types, we can also define this relation using types. That is, $\bar{a}_k \equiv_{\simeq} \bar{b}_k$ iff, for every formula $\varphi \in FO$ with k free variables $\bar{a}_k \in \varphi^I$ iff $\bar{b}_k \in \varphi^I$.

Definition 3.1:

- Let $k \geq 1$. A structure I is *k-homogeneous* if for every pair of k-tuples $\bar{a}_k$ and $\bar{b}_k$ in $(dom(I))^k$, the following holds:

$$\bar{a}_k \equiv_k \bar{b}_k \qquad \Longrightarrow \qquad \bar{a}_k \equiv_{\simeq} \bar{b}_k$$

- Let $k \geq 1$, and let σ be a signature. A class $\mathcal{C}$ of σ-structures is *k-homogeneous* if every structure $I \in \mathcal{C}$ is k-homogeneous.

The next Fact follows immediately from the definition of the relation $\equiv_{\simeq}$ and from Definition 3.1.

Fact 3.1: Let $k \geq 1$. If a structure I is k-homogeneous, then for every $1 \leq l \leq k$ and for every pair of l-tuples $\bar{a}_l$ and $\bar{b}_l$ in $(dom(I))^l$, the following holds:

$$\bar{a}_l \equiv_k \bar{b}_l \qquad \Longrightarrow \qquad \bar{a}_l \equiv_{\simeq} \bar{b}_l$$

Note that if a structure I is k-homogeneous, for some k, it is not necessarily r-homogeneous for every $r \geq k$. So, we next define a stronger notion regarding homogeneity.

Definition 3.2:

[3] This was suggested to me by Lauri Hella.

- Let $k \geq 1$. A structure I is *strongly k-homogeneous* if for every $r \geq k$ it is r-homogeneous.
- Let $k \geq 1$, and let σ be a signature. A class $\mathcal{C}$ of σ-structures is *strongly k-homogeneous* if every structure $I \in \mathcal{C}$ is strongly k-homogeneous.

It seems that it is not easy to find examples of classes of structures which are k-homogeneous, for some k, but which are not strongly k-homogeneous. However, we believe that such classes do exist. That is, we believe that strong k-homogeneity is actually a *stronger* notion than k-homogeneity.

The next observation follows from the well known fact that every finite structure of size n can be characterized up to isomorphism in FO^n.

Fact 3.2: Every finite structure is strongly k-homogeneous for some $k \geq 1$.

However, it is clear that this is not the case with *classes* of structures. And it is quite easy to build classes which are not even k-homogeneous for any k. Let's consider a class of graphs where each one is the disjoint union of two trees as follows. Every tree is built over a linear sub-graph, by adding > 1 sons to the maximal node of the linear sub-graph. And in every graph in the class the length of the linear sub-graphs in both trees is the same, and the out degree is different in both of them. The lengths and the out degrees can grow arbitrarily in the class. Now let's take one graph with, say, out degrees k and $(k+1)$, respectively, in the trees which form it. This graph certainly cannot be k-homogeneous, because we can easily choose two k-tuples, one from each tree, which have the same FO^k type and for which there is no automorphism mapping one to each other. This is because in FO^k we cannot distinguish between out degrees k and $(k+1)$. Then, as the out degrees are not bounded in the class, it is clear that the class cannot be k-homogeneous for no k.

In [AV,91] it was noted that the discerning power of a k- GM^{loose} is restricted to FO^k types for k-tuples. So, the following result seems quite natural, and is somehow implicit in Abiteboul and Vianu's work. Anyway, it is worthwhile to point out that k-homogeneity is a very important property for a class of structures, or database, as it is a means to increase the computability power of an incomplete machine as GM^{loose}. If f is a query of signature σ, and $\mathcal{C}$ is a class of σ-structures, we will denote as $f|_{\mathcal{C}}$ the restriction of f to the class $\mathcal{C}$.

Theorem 3.1: *For every $k \geq 1$, there is a set of queries F of some signature σ*

such that if $\mathcal{C}$ and $\mathcal{C}'$ are any two classes of σ-structures, such that $\mathcal{C}$ is k-homogeneous and $\mathcal{C}'$ is not k-homogeneous, then, for every $f \in F$, $f|_{\mathcal{C}}$ is computable by a k-GM^{loose} machine, but it is not the case that for every $f \in F$, $f|_{\mathcal{C}'}$ is computable by a k-GM^{loose} machine.

Proof: For every given k, we will exhibit a class of queries for which we will prove that there is at least one which is not computable by any k-GM^{loose} on classes which are not k-homogeneous. For every structure I, let's define the k-ary query q_i, for $i \geq 1$, as the i-th equivalence class in the relation $\equiv_{\simeq}$ for k-tuples in I, considering any pre-defined and computable order. These queries are certainly computable on k-homogeneous classes of structures. To see this, note that we can use the same construction as in Theorem 2.1 to build the isolating formulae for FO^k types for k-tuples, which for this kind of classes are also automorphism types for k-tuples.

Now, for arbitrary classes of structures, let's fix a $k \geq 1$, and let $\mathcal{C}$ be a class of structures which is not k-homogeneous. Let $I \in \mathcal{C}$, such that I is not k-homogeneous either. Then, there are two k-tuples $\bar{a}_k$ and $\bar{b}_k$ in $(dom(I))^k$ such that $\bar{a}_k \equiv_k \bar{b}_k$ but $\bar{a}_k \not\equiv_{\simeq} \bar{b}_k$. Let q_i be the query such that $\bar{a}_k$ is in $q_i(I)$ but $\bar{b}_k$ is not in $q_i(I)$. We will prove that there is no k-ary query f computable by a k-GM^{loose} such that f on input I includes in its result the tuple $\bar{a}_k$, but it doesn't include in its result the tuple $\bar{b}_k$. Consequently, q_i is not computable by a k-GM^{loose}.

Let T be the output k-ary relation symbol in the relational store of a k-GM^{loose} which could compute q_i. We will see that at every stage of the computation of anyone of these machines with the input structure I in their relational stores, either both k tuples $\bar{a}_k$ and $\bar{b}_k$ are in T or none of them are in T. As we mentioned before in this article, the only way to assign a relation to a k-ary relation symbol in the relational store of a k-GM^{loose}, say M, is through a FO^k formula with k free variables, say φ. By Fact 2.3, φ defines in I a relation which is the union of some equivalence classes in $\equiv_{k,M}$, which in turn is the union of some equivalence classes in $\equiv_k$. And we know that $\bar{a}_k \equiv_k \bar{b}_k$ and, hence, $\bar{a}_k \equiv_{k,M} \bar{b}_k$ for every k-GM^{loose} M. So, at any stage of the computation of the machine, either both k tuples are in T or none of them are in T. Note that it can be the case that some auxiliary relation symbols are used in φ. But the arity of all of them must be $\leq k$, and by the same argument their contents at any stage of the computation must include either both k tuples or none of them.

So, every k-GM^{loose} with a k-ary output relation symbol will compute a query which on input I will either include both tuples $\bar{a}_k$ and $\bar{b}_k$, or it will include none of them. ■

Remark 3.1:

Note that the subclass of queries which can be computed by k-GM^{loose} machines on k-homogeneous classes of structures, but which *cannot* be computed by such machines on arbitrary classes of structures, is quite big. Let's consider by case that a k-GM^{loose}

can count the number of equivalence classes in $\equiv_{\simeq}$ for k-tuples on a k-homogeneous structure, as an intermediate result in its TM tape. This can be done by following the same process as we did in the Proof of Theorem 2.1, and then by counting the isolating formulae for FO^k types on whose realization both structures (I and I') agree. And it should be clear by now that, for any signature σ, this number cannot be computed by any k-GM^{loose} on the whole class $\mathcal{E}_{\sigma,fin}$. Then we can use this parameter, which we will call *type index for k-tuples* following [AV,91], as the argument for *any* partial recursive predicate in the k-GM^{loose}.

So that the whole class of partial recursive predicates, evaluated on the type index for k-tuples of the input structure is included in the subclass of queries which can be computed with k-GM^{loose} machines, due to k-homogeneity.

However, k-homogeneity is not enough to achieve completeness with k-GM^{loose} machines, as queries like parity of the size of the input structure, parity on the sizes of the equivalence classes in $\equiv_{\simeq}$, and isomorphism, are not computable with these machines, even on k-homogeneous classes of structures. Let's consider as an example a class $\mathcal{C}$ of graphs which are the disjoint union of some set of cliques of size k. This class is clearly k-homogeneous. Now, let's consider the equivalence class in $\equiv_k$ of the k-tuples of nodes which belong to the same clique. It is clearly also a class in $\equiv_{\simeq}$, and for different graphs it can have certainly a different size. But a k-GM^{loose} cannot count the tuples in this equivalence class.

Next, we will show that the property of *strong* k-homogeneity will allow the GM^{loose} machines to get still more power as to computability of queries. But this time we will use r-ary machines, for every $r > k$, considering that according to Definition 3.2 the strong k-homogeneity of a class implies its r-homogeneity, for every $r \geq k$. And this is not the case for k-homogeneous classes which are not strongly k-homogeneous. Let's denote by $\{F_r\}_{r\geq k}$ a countably infinite class of sets of queries F_r, for every $r \geq k$.

Lemma 3.1: *For every $k \geq 1$, there is a class of sets of queries $\mathcal{F} = \{F_r\}_{r\geq k}$ of some signature σ, such that, if $\mathcal{C}$ and $\mathcal{C}'$ are any two classes of σ-structures, such that $\mathcal{C}$ is strongly k-homogeneous and $\mathcal{C}'$ is not strongly k-homogeneous, then for every $F_r \in \mathcal{F}$ and for every $f \in F_r$, $f|_{\mathcal{C}}$ is computable by an r-GM^{loose} machine, but it is not the case that for every $F_r \in \mathcal{F}$ and for every $f \in F_r$, $f|_{\mathcal{C}'}$ is computable by an r-GM^{loose} machine.*

Proof: We can use here the same strategy as we used in the Proof of Theorem 3.1. As to the queries q_i we ask for the equivalence classes in the relation $\equiv_{\simeq}$ for r-tuples in the given input structure. If the class of structures is strongly k-homogeneous, it is r-homogeneous for every $r \geq k$. So that the equivalence classes in $\equiv_r$ which will be complete in the relation resulting from the evaluation of every FO^r formula with

r free variables in the machine, will actually be equivalence classes in $\equiv_{\simeq}$ for r-tuples in that structure. So, the queries q_i will be computable by r-GM^{loose} machines for this kind of classes of structures. And this is the case for every $r \geq k$. On the other hand, if the class of structures is k-homogeneous, but not strongly k-homogeneous, it is not r-homogeneous for every $r \geq k$. Hence, the equivalence classes in $\equiv_r$ will not be equivalence classes in $\equiv_{\simeq}$ for r-tuples, for some r, say r'. Then, the corresponding class of queries q_i is not computable by r'-GM^{loose} machines on these classes of structures. ■

Now we will define a stronger concept regarding homogeneity, but this time involving classes of structures, instead of a single one. We will show that with this stronger notion we can achieve completeness with k-GM^{loose} machines, in a sense which will be clarified later (Lemma 3.2).

Definition 3.3: Let σ be a signature, and let $\mathcal{C}$ be a class of σ-structures. Let $k \geq 1$. We say that $\mathcal{C}$ is *pairwise k-homogeneous*, if for every pair of structures I_1, I_2 in $\mathcal{C}$, and for every pair of k-tuples $\bar{a}_k \in (dom(I_1))^k$ and $\bar{b}_k \in (dom(I_2))^k$, if $\bar{a}_k \equiv_k \bar{b}_k$ then there exists an isomorphism $f : dom(I_1) \longrightarrow dom(I_2)$, such that $f(a_i) = b_i$, for every $1 \leq i \leq k$.

That is, in pairwise k-homogeneous classes FO^k types for k-tuples are *isomorphic types* for k-tuples.

As an example of such classes, let's define the classes $\mathcal{T}_r$, for every $r \geq 1$, as classes of trees of arbitrary depth, of out degree bounded by r, and where all the nodes at the same depth have the same out degree. So, we can write a formula with one free variable which defines a type for 1-tuples, expressing the distance from the node to the root, the distance from the node to a leaf, and the out degree of every ancestor, as well as the out degree of every descendant, in any of the paths from the node to a leaf (it doesn't matter which path we choose, because in all of them all the nodes at the same depth will have the same degree). This formula can be written with $(r+1)$ variables (though it should be noted that for trees which are not in the class $\mathcal{T}_r$ we need $(r+2)$ variables). And it is clear that we can write the FO^{r+1} types for $(r+1)$-tuples by using a formula of this kind for every component of the tuple, and then by defining, with 2 variables, the directed or undirected path between every pair of components. Finally, the conjunction of all those formulae isolates the desired FO^{r+1} type for $(r+1)$-tuples. Then it is clear that if two $(r+1)$-tuples of two different trees in the class have the same FO^{r+1} type, the trees must be isomorphic, and one tuple can certainly be mapped onto the other. So that the classes $\mathcal{T}_r$, for every $r \geq 1$, are $(r+1)$-isomorphic.

Remark 3.2:

These three notions related to homogeneity are quite similar to three properties defined in [Set,95], where they are studied in their relation to expressive power of fixed point logics on classes which satisfy them. What we defined as pairwise k-homogeneity (for k-tuples) corresponds to Seth's notion of *bounded k-quasi rigidity*. Our notion of k-homogeneity corresponds to Seth's *weak bounded k-quasi rigidity*. And strong k-homogeneity is quite similar to Seth's *weak bounded quasi rigidity*. The difference in our three notions with the corresponding notions of Seth is that he considers that the number of variables n used to define the types in FO^n may be bigger than the length of the tuples, k. Further, Seth asks for this number n to exist for the class of structures, without caring in his study about the relation of n to k. Finally, regarding our notion of strong k-homogeneity, for classes of structures to have the property of *weak bounded quasi rigidity*, he asks for every k the existence of a number, say n_k such that for every structure in the class n_k variables suffice to define the automorphism types for k-tuples in FO. But he doesn't ask for the different n_k numbers to be related in any way.

On the other hand, there are three results in Seth's work which are worthwhile to be pointed out in the context of our research. Let PFP denote FO with the addition of the fixed point operator such that the iteration may not stop (see [EF,95]). And let $f_{k,m}$, for $m \geq k$, be a mapping from the set of isomorphism classes of k-tuples in a class $\mathcal{C}$ of structures (which he denotes as $ISO_k(\mathcal{C})$), to the class of the quotients of the structures in $\mathcal{C}$ under the equivalence relation $\equiv_m$, such that, for every $I \in \mathcal{C}$, $f_{k,m}(I, a_1, \ldots, a_k) = (I/\equiv_m, [a_1, \ldots, a_k, a_k, \ldots, a_k])$. Where $[a_1, \ldots, a_k, a_k, \ldots, a_k]$ is built from the sequence $[a_1, \ldots, a_k]$ by repeating the element a_k to get a sequence of length m (for the construction of the quotient structures $I/\equiv_m$, see [Daw,93]):

- **Theorem [Set,95]:** Let $\mathcal{C}$ be a class of structures. If PFP on $\mathcal{C}$ equals PSPACE on $\mathcal{C}$, then $\mathcal{C}$ is of weak bounded quasi rigidity.
- **Theorem [Set,95]:** Let $\mathcal{C}$ be a class of structures. Then, the following are equivalent:
    a) There is an $m \in \omega$ such that the map $f_{k,m}$ is one to one.
    b) $\mathcal{C}$ is of bounded k-quasi rigidity.
- **Theorem [Set,95]:** Let $\mathcal{C}$ be a class of structures in PSPACE which is of bounded m-quasi rigidity. Then, all PSPACE m-ary queries on $\mathcal{C}$ are expressible in PFP.

Our next result, which shows that completeness can be achieved with k-GM^{loose} when working on pairwise k-homogeneous classes of structures, is an extension of the previous Seth's result, as to computability power.

Lemma 3.2: *For every $k \geq 1$, and for every query f of any signature σ in typed($\mathcal{CQ}$), if $\mathcal{C}$ is a class of σ-structures which is recursive and which is also pairwise k-homogeneous, then there is a k-GM^{loose} machine which computes $f|_{\mathcal{C}}$.*

Proof: One direction is trivial. As to the other direction, we use the same strategy to build a *k-GM^{loose}* $\mathcal{M}_f$ which computes a given query $f \in$ *typed*($\mathcal{CQ}$), as we did in the Proof of Theorem 2.1. In this case we must check that the structure I' which we build in the TM tape is in the class of structures on which $\mathcal{M}_f$ is defined to work. This is the reason why we ask for the class to be recursive. So we will know that if I' and I (the input structure) realize the same FO^k types for k-tuples and if the union of the equivalence classes in I is the whole set of k-tuples from $dom(I)$, then the structures are isomorphic, and we can compute f on I' in the TM part of $\mathcal{M}_f$. Finally, to build $f(I)$, we just take the corresponding equivalence classes which realize in I the FO^k types which are realized in $f(I')$. ∎

Finally, let's consider *Reflective Machines* from [APV,94]. We will denote reflective machines with dynamic queries restricted to use a number of variables given by a recursive function f on the size of the domain of the input structure as $GM^{f(n)}$. As we observed before in this article, there are many classes of structures for which there is no $k \in \omega$ such that the class is k-homogeneous. However it is quite easy to build classes of structures where the number of variables which are needed for automorphism types for tuples is bounded by a *sub-linear* function. For our purpose, it is enough to define a sub-linear function as a function over the naturals such that for every $n \in \omega$ it is $f(n) < n$. This is because as it is well known in FO^n we can characterize a structure of size n up to isomorphism, and we cannot count the number of elements with less than n variables. As an example of such classes of structures, think of graphs which are formed by a certain number of isolated nodes and the disjoint union of two trees of depth 1, where the out degree of the trees differs by 1. We can make the number of isolated nodes arbitrary big in relation with the degrees. And it is clear that if we can use a number of variables big enough to separate the trees we can define the automorphism types, and then we can adjust the number of isolated nodes so that the number of variables is bounded by any given recursive and sub-linear function. For this last part of the Section, we will denote as $\equiv_{f(n),k}$ the equivalence of $FO^{f(n)}$ types for k-tuples. This is because in the case of reflective machines we will consider types in FO, where the bound on the number of variables is given by a function, but for tuples of a fixed size.

Definition 3.4:

- Let $f : \omega \longrightarrow \omega$ be a sub-linear and recursive function, and let $k \geq 1$. A structure I is $(f(n), k)$-*homogeneous* if $k \leq f(|dom(I)|)$ and if, for every pair of

k-tuples $\bar{a}_k$ and $\bar{b}_k$ in $(dom(I))^k$, the following holds:

$$\bar{a}_k \equiv_{f(n),k} \bar{b}_k \qquad \Longrightarrow \qquad \bar{a}_k \equiv_{\simeq} \bar{b}_k$$

- A class $\mathcal{C}$ of σ-structures is *$(f(n),k)$-homogeneous* if every structure $I \in \mathcal{C}$ is $(f(n),k)$-homogeneous.

Lemma 3.3: *Let $k \geq 1$. Then, for every sub-linear and recursive function f, there is a set of queries Q of some signature σ, such that if $\mathcal{C}$ and $\mathcal{C}'$ are any two classes of σ-structures, such that $\mathcal{C}$ is $(f(n),k)$-homogeneous and $\mathcal{C}'$ is not $(f(n),k)$-homogeneous, then for every $q \in Q$, $q|_{\mathcal{C}}$ is computable by a $GM^{f(n)}$ machine, but it is not the case that for every $q \in Q$, $q|_{\mathcal{C}'}$ is computable by a $GM^{f(n)}$ machine.*

Proof Sketch: The Proof is quite similar to the Proof of Theorem 3.1. For every structure I, let's define the k-ary query q_i, for $i \geq 1$, as the i-th equivalence class in the relation $\equiv_{\simeq}$ for k-tuples in I. The difference here is that we can be allowed to use more variables than we actually need for certain structures. But, certainly there are arbitary classes of structures with structures for which the number of variables given by $f(n)$, with n being their size, will not be enough to isolate classes in $\equiv_{\simeq}$. ■

Section 4: Preservation of Automorphism types.

With the classes $\mathcal{QCQ}^k$ we characterized the total and typed queries which preserve realization of FO^k types for k-tuples. But, as we noted in the observation following Fact 3.2, it is not the case that every countably infinite class of structures is k-homogeneous for some k. So, the queries which preserve realization of FO types for tuples, but which don't preserve realization of FO^k types for k-tuples, for any k, will not belong to any $\mathcal{QCQ}^k$ class, for no k. This is because if we fix some $k \geq 1$, the number of variables which are needed for the automorphism types for k-tuples in different structures can be different. And the number of different bounds for the number of variables is actually infinite.

So that for this purpose we need to define a new class of queries in a different way than the one we used before. The intuitive idea behind this new class is that queries which belong to it preserve, for any two structures of the corresponding signature, the realization of types in FO bounded to a number of variables which is enough for *both* structures to define tuples up to automorphism. Note that for different pairs of structures these queries will preserve types defined with possibly different number of variables.

Definition 4.1: For any signature σ, let's denote as $arity(\sigma)$ the maximum arity of a relation symbol in the signature. We define the class $\mathcal{QCQ}$ as the class of queries

$f \in total\text{-}typed(\mathcal{CQ})$, of some signature σ and of any arity, for which there exists an integer $n \geq max\{arity(f), arity(\sigma)\}$, such that for every pair of structures I_1, I_2 in $\mathcal{E}_{\sigma,fin}$, the following holds:

$$I_1 \equiv_{\tau_h} I_2 \qquad \Longrightarrow \qquad f(I_1) \equiv_{\tau_h} f(I_2)$$

where

$$h = max\{n, min\{k : I_1 \text{ and } I_2 \text{ are strongly } k\text{-homogeneous}\}\}$$

Note that by Fact 3.2, for every pair of structures there is a k such that both structures are strongly k-homogeneous.

Now we will show that this class strictly includes the whole hierarchy which we defined in Section 2, that is, it includes the class $\mathcal{QCQ}^\omega$.

Lemma 4.1: $total\text{-}typed(\mathcal{CQ}) \supset \mathcal{QCQ} \supset \mathcal{QCQ}^\omega$.

Proof:

a) (second $\neq$): We will exhibit a query f in the signature of the graphs, such that $f \in \mathcal{QCQ}$, but $f \notin \mathcal{QCQ}^\omega$. Let f be *the nodes belonging to the smaller clique* in the sub-class $\mathcal{G}$ of graphs which are the disjoint union of two cliques whose sizes differ at most by 1. And we define f to be empty if the two cliques are of the same size or if the graph is not in that sub-class. For any $k \geq 1$, we can consider two graphs, I_1 and I_2, such that in I_1 the two cliques are of size k and $(k+1)$, respectively, while in I_2 both cliques are of size k. Clearly, both graphs realize the same FO^k types for k-tuples. This is because with k variables cliques of size k cannot be separated from cliques of a greater size (this can be easily proved with a k-pebble game). But the query doesn't preserve realization of FO^k types for k-tuples, because $f(I_2)$ is empty, while $f(I_1)$ has all the nodes in the smaller clique. So, $f \notin \mathcal{QCQ}^k$. As we can choose two such graphs for every $k \geq 1$, it follows that $f \notin \mathcal{QCQ}^k$ for any $k \geq 1$, and, hence, $f \notin \mathcal{QCQ}^\omega$.

On the other hand, if there are two cliques of a different size in a graph I_1, say k_1 and k_2, with $k_1 = k_2 + 1$, the graph will be strongly k_1-homogeneous, because k_1 variables suffice to define the automorphism types for l-tuples, for $1 \leq l \leq k_1$. Note that for $(k_1 + i)$-tuples, for every $i \in \omega$, $(k_1 + i)$ variables suffice to define the automorphism types for l-tuples, for $1 \leq l \leq (k_1 + i)$. This is because for the automorphism type of each node it is enough with k_1 variables, and then we only need, for every pair of elements in the tuple, to say whether they are connected or not.

And I_1 will not realize the same FO^{k_1} types as a graph with two disjoint cliques of the same size, say I_2. If the size of the two cliques in I_2 is $k_3 \leq k_1$, it is clear, since I_2 is strongly k_3-homogeneous. Otherwise, if the size of the two cliques in I_2 is $k_3 > k_1$, as I_2 is strongly k_3-homogeneous, the two graphs will not realize the same FO^{k_3} types for k_3-tuples.

Finally, note that for every pair of graphs which are not in the sub-class $\mathcal{G}$, f will trivially preserve realization of FO^h types, for *any* $h \geq 1$, because its result is empty in both cases by definition. And if we consider a graph I_1 in $\mathcal{G}$ and a graph I_2 not in $\mathcal{G}$, they will not realize the same automorphism types, because either I_2 has at least one node which does not belong to a clique, or it is the disjoint union of two cliques whose sizes differ by more than 1.

So, it follows that $f \in \mathcal{QCQ}$.

b) (second $\supseteq$): Let $f \in \mathcal{QCQ}^k$, for some k, with signature σ. So, f preserves $\equiv_{\tau_k}$ on every pair of structures in $\mathcal{E}_{\sigma,fin}$. We choose $n = max\{k, arity(f), arity(\sigma)\}$. Let I_1 and I_2 be two arbitrary σ-structures, and let

$$h = max\{n, min\{l : I_1 \textit{ and } I_2 \textit{ are strongly } l\textit{-homogeneous}\}\}.$$

Now we must show that f preserves $\equiv_{\tau_h}$ on I_1 and I_2 to prove that $f \in \mathcal{QCQ}$.

As $h \geq k$, by Remark 2.1 f preserves $\equiv_{\tau_h}$ on every pair of structures in $\mathcal{E}_{\sigma,fin}$, and in particular on I_1 and I_2. So that it belongs to $\mathcal{QCQ}$.

c) (first $\supset$): Let f be the parity query, on the size of the domain of a structure, in the signature of the identity relation. That is, we define f in the class of sets, with identity. It is clear that every set is strongly 1-homogeneous. So, for whatever n we choose f will not preserve realization of FO^n types in the whole class of sets, since in FO^n we cannot count elements beyond n. Then, it follows that $f \notin \mathcal{QCQ}$.

On the other hand, it follows from Definition 4.1 that *total-typed*$(\mathcal{CQ}) \supseteq \mathcal{QCQ}$.

■

We should point out a very important difference between queries in $\mathcal{QCQ}$ and queries in $\mathcal{QCQ}^\omega$. A query in the first class has to be aware of every property which a tuple may have in the structure, up to automorphism. And this fact can imply that it will not preserve realization of coarser types, that is, types defined in FO with fewer variables. In part **b** of the Proof of Lemma 2.1 we showed an example of a query which preserves realization of FO^{k+1} types, but which does not preserve realization of FO^k types.

On the other hand, a query $f \in \mathcal{QCQ}^k$, for some k, must preserve realization of FO^k types for k-tuples in *every pair* of structures in the corresponding signature. And, by Remark 2.1, if this is the case, the query f will also preserve the realization of types defined with any bigger number of variables, say $(k+i)$ for some $i \in \omega$, also for every pair of structures in the signature. Now, note that f is sufficiently precise as to consider FO^{k+i} types, but, at the same time, it must also be sufficiently simple so it doesn't separate equivalence classes in $\equiv_k$ in equivalence classes in $\equiv_{k+i}$ which form them (by Fact 2.4), arbitrarily, because f must preserve realization of FO^{k+i} types.

And queries in $\mathcal{QCQ}$ are not restricted in this way.

Remark 4.1: Note that the queries based on the type index for k-tuples which we mentioned in Remark 3.1 as being computable by a k-GM^{loose} on k-homogeneous classes of structures, and which we argued that cannot be computed by any k-GM^{loose} on arbitrary classes of structures, are in $(\mathcal{QCQ} - \mathcal{QCQ}^{\omega})$, if we define them on the whole class of structures of a given signature.

So that the increment in the computability power of the GM^{loose} machine when working on classes of structures which are homogeneous, is actually quite big. It is big enough to allow for a k-GM^{loose} to go through the whole hierarchy $\mathcal{QCQ}^{\omega}$, in a sense which will be clarified next, and, further, to get out of it into $\mathcal{QCQ}$. This is actually quite natural, because classes of structures which are k-homogeneous will not benefit from the possibility of using $> k$ variables in dynamic queries, since FO^k types on them cannot be further refined. So, the next results follow immediately.

Corollary 4.1: *There are queries f of some signature σ in $(\mathcal{QCQ} - \mathcal{QCQ}^{\omega})$ whose restriction to any class of σ-structures which is k-homogeneous, for some $k \geq 1$, is computable by a k-GM^{loose} machine.* $\blacksquare$

Corollary 4.2: *Let f be a query of signature σ whose restriction to any class of σ-structures which is k-homogeneous, for some $k \geq 1$, is computable by a k-GM^{loose} machine. Then, $f \in \mathcal{QCQ}$.* $\blacksquare$

Acknowledgements

The work presented here originated from a question suggested by Lauri Hella, during a visit to the Department of Mathematics of the University of Helsinki. I am deeply grateful to him and to Jouko Väänänen for letting me visit this Department, as well as for their partial funding. Also warm thanks to Lauri Hella for the stimulating discussions we had, and mainly for his questions. His deep knowledge was also of a great help to me.

References

[APV,94] Abiteboul, S., Papadimitriou, C. and Vianu, V., “The Power of the Reflective Relational Machine”, in Proceedings of the 9th IEEE Symposium on Logic in Computer Science, 1994.

[AV,91] Abiteboul, S. and Vianu, V., “Generic Computation and its Complexity”, In Proceedings of the 24th ACM Symposium on Theory of Computing, 1991.

[AVV,92b] Abiteboul, S., Vardi, M. and Vianu, V., “Computing with Infinitary Logic”, Research Report RJ 8925, 1992.

[CH,80] Chandra, A., Harel, D., “Computable Queries for Relational Data Bases”, Journal of Computer and System Sciences 21, 1980.

[CH,82] Chandra, A., Harel, D., “Structure and Complexity of Relational Queries”, Journal of Computer and System Sciences 25, 1982.

[CK,92] Chang, C. and Keisler, H., “Model Theory”, 3rd ed., Elsevier North Holland, 1992.

[Daw,93] Dawar, A., “Feasible Computation Through Model Theory”, PhD Thesis, University of Pennsylvania, Philadelphia, 1993.

[EF,95] Ebbinghaus, H., Flum, J., “Finite Model Theory”, Springer Verlag, 1995.

[KV,90] Kolaitis, P., Vardi, M., “0–1 Laws for Infinitary Logic”, In Proceedings of the 5th IEEE Symposium on Logic in Computer Science, 1990.

[Set,95] Seth, Anil, “When do Fixed Point Logics Capture Complexity Classes?”, preprint, published in Proceedings of the 10th IEEE Symposium on Logic in Computer Science, 1995.

[Tur,95] Turull Torres, J.M., “Query Completeness, Distinguishability and Relational Machines”, to appear in the Proceddings of the X Latin American Symposium on Mathematical Logic, Bogotá, 1995.

[Tur,96] Turull Torres, J.M., “Partial Distinguishability and Query Computability on Finite Structures”, communicated in the XI Brazilean Symposium on Logic, XI-EBL, Salvador, Brazil, 1996. Abstract published in the Bulletin of the IGPL, London, 3-1996.

[Tur,97] Turull Torres, J.M., “Untyped Queries, Untyped Reflective Machines and Conditional Quantifiers”, Research Report 146 of the Department of Mathematics of the University of Helsinki, 1997.