# Relational Databases and Homogeneity in Logics with Counting

José María Turull Torres[*]

**Abstract**

We define a new hierarchy in the class of computable queries to relational databases, in terms of the preservation of equality of theories in fragments of first order logic with bounded number of variables with the addition of counting quantifiers ($C^k$). We prove that the hierarchy is strict, and it turns out that it is orthogonal to the TIME-SPACE hierarchy defined with respect to the Turing machine complexity. We introduce a model of computation of queries to characterize the different layers of our hierarchy which is based on the reflective relational machine of S. Abiteboul, C. Papadimitriou, and V. Vianu. In our model the databases are represented by their $C^k$ theories. Then we define and study several properties of databases related to homogeneity in $C^k$ getting various results on the change in the computation power of the introduced machine, when working on classes of databases with such properties. We study the relation between our hierarchy and a similar one which we defined in a previous work, in terms of the preservation of equality of theories in fragments of first order logic with bounded number of variables, but *without* counting quantifiers ($FO^k$). Finally, we give a characterization of the layers of the two hierarchies in terms of the infinitary logics $C^k_{\infty\omega}$ and $\mathcal{L}^k_{\infty\omega}$, respectively.

**Keywords:** query languages, database machines, query computability, completeness of models, counting

## 1 Introduction

Given a relational database schema, it is natural to think about the whole class of queries which might be computed over databases of that schema. That is, if we do not restrict ourselves to a given implementation of certain query language on some computer, in the same way as the notion of computable function over the natural numbers was raised in computability theory. In [CH80], A. Chandra and D. Harel devised a formalization for that notion. They defined a *computable query*

[*]Massey University, Department of Information Systems, Information Science Research Centre, PO Box 756, Wellington, New Zealand E-mail: `j.m.turull@massey.ac.nz`

as a function over the class of databases of some given schema which is not only recursive but preserves isomorphisms as well. Isomorphism preservation is what formalizes the intuitive property of representation independence.

In [Tur01a, Tur04] a strict hierarchy was defined, in the class of computable queries ($\mathcal{CQ}$) of A. Chandra and D. Harel, in terms of the preservation of equivalence in bounded variable fragments of first order logic ($FO$), which we denote by $FO^k$. The logic $FO^k$ is the fragment of $FO$ which consists of the formulas with up to $k$ different variables. We denote the whole hierarchy as $\mathcal{QCQ}^\omega$. For every natural $k$, the layer denoted as $\mathcal{QCQ}^k$ was proved to be a semantic characterization of the computation power of the reflective relational machine of [APV98] with variable complexity $k$ ($RRM^k$). The $RRM^k$ is a model of computation of queries to relational databases which has been proved to be incomplete, i.e., it does not compute all computable queries. Then, we defined and studied in [Tur01a, Tur04] several properties of relational databases, related to the notion of homogeneity in model theory [CK92], as properties which increase the computation power of the $RRM^k$ when working on databases having those properties.

That research was enrolled in a very fruitful research program in the field of finite model theory considered as a theoretical framework to study relational databases. In that program different properties of the databases have been studied, which allow incomplete computation models to change their expressive power when computing queries on databases with those properties. Order [AV95, EF99], different variants of rigidity [DLW95, Tur96, Tur98], and different notions related to homogeneity [Tur01a, Tur04] are properties which turned out to be quite relevant to the expressive power of certain models of computation.

In the present paper, and following the same research program, we define a new hierarchy in the class of computable queries, which we denote as $\mathcal{QCQ}^{C^\omega}$. We define this hierarchy in terms of the preservation of equivalence in bounded variable logics with *counting quantifiers* ($C^k$). For every natural $k$, we denote as $\mathcal{QCQ}^{C^k}$ the layer of the hierarchy $\mathcal{QCQ}^{C^\omega}$ which consists of those queries that preserve equivalence in $C^k$. The logic $C^k$ is obtained by adding quantifiers "there exists at least $m$ different elements in the database such that..." to the logic $FO^k$ for every natural $m$. The logic $C^k$ has been deeply studied during the last decade [CFI92, Gro96, Hel96, Ott97].

Defining the classes $\mathcal{QCQ}^{C^k}$ appears to be rather natural, since in the definition of *computable query* of [CH80] the property of preservation of isomorphisms is essential, and, as it is well known, in finite databases isomorphism coincides with equivalence in first order logic. Moreover, it is also well known that for every natural $k$, the logic $C^k$ is strictly weaker than $FO$. So, when we define subclasses of computable queries in terms of the preservation of $C^k$ equivalence, for different values of $k$, in a certain way we are classifying queries according to the *different levels in the amount of information which we really need about the input database* to evaluate a given query on that database.

The hierarchy $\mathcal{QCQ}^{C^\omega}$ turns out to have quite similar structure and behavior as the hierarchy $\mathcal{QCQ}^\omega$ [Tur01a, Tur04]. The results of Sections 3 and 4 are analogous to the results in [Tur01a, Tur04] regarding $\mathcal{QCQ}^{\omega}$, and their proofs follow a similar strategy. However, there is a very important difference in the expressiveness of the layers of both hierarchies, which is not surprising given the well known difference in expressive power between the logics $FO^k$ and $C^k$. For every $k \geq 2$ the subclass $\mathcal{QCQ}^{C^k}$ is "big", whereas each subclass $\mathcal{QCQ}^k$ is "small", in a sense which we will make precise in Section 5 using results on asymptotic probabilities. Roughly, we can say that for every computable query $q$ there is a query $q'$ in the layer $\mathcal{QCQ}^{C^2}$ which is equivalent to $q$ over *almost all* databases. And this is not true for any layer in $\mathcal{QCQ}^{\omega}$, and not even for the whole hierarchy.

We prove that the hierarchy $\mathcal{QCQ}^{C^{\omega}}$ is strict, and that it is strictly included in the class $\mathcal{CQ}$ of computable queries. Furthermore, it turns out to be *orthogonal* to the TIME-SPACE hierarchy defined with respect to Turing machine complexity, as it was also the case with the hierarchy $\mathcal{QCQ}^{\omega}$. Hence, we can define finer classifications in the class $\mathcal{CQ}$ by intersecting $\mathcal{QCQ}^{C^{\omega}}$ with the Turing machine complexity hierarchy (see [Tur01b] for a preliminary discussion of this approach). As an illustrating example, we include a classification of some problems in finite group theory at the end of Section 5. This may be derived from results in [KL99] and [KL00] together with our characterization of the layers of $\mathcal{QCQ}^{\omega}$ in terms of fragments of the infinitary logic $\mathcal{L}^{\omega}_{\infty\omega}$.

Having defined the different classes $\mathcal{QCQ}^{C^k}$ in a semantic way, we look next for a syntactic characterization of these classes in terms of a computation model. For that sake we define a machine which we call *reflective counting machine* with bounded variable complexity $k$ ($RCM^k$), as a variant of the reflective relational machine of [APV98]. In our model, dynamic queries are formulas of $C^k$, instead of $FO^k$. In [Ott96] a similar model has been defined to characterize the expressibility of fixed point logics with counting terms, but it was based on the relational machine of [AV95], instead. Then we prove that for every natural $k$, the class $\mathcal{QCQ}^{C^k}$ characterizes exactly the expressive power of the machine $RCM^k$.

The model $RCM^k$ turns out to be incomplete, i.e., there are computable queries which cannot be computed by any such machine. Then, we define several properties related to homogeneity (which are quite analogous to the properties studied in ([Tur01a], [Tur04]) regarding the model $RRM^k$), and we study the way in which the computation power of the model $RCM^k$ changes when working on such classes of databases. Such properties are $C^k$*-homogeneity*, *strong* $C^k$*-homogeneity* and *pairwise* $C^k$*-homogeneity*. A database is $C^k$*-homogeneous* if the properties of every $k$-tuple in the database, up to automorphism, can be expressed by $C^k$ formulas (i.e., whenever two tuples satisfy the same properties expressed by $FO$ formulas with $k$ variables and with counting quantifiers, then there is an automorphism of the database mapping each tuple onto each other). We prove that for every $k \geq 1$ there are queries whose restriction to $C^k$-homogeneous databases can be computed by $RCM^k$ machines, whilst the same queries *cannot* be computed by any $RCM^k$ on the *whole class* of databases of the given schema. A database is *strongly* $C^k$*-homogeneous* if it is $C^r$-homogeneous for every $r \geq k$. Here we show that, roughly speaking, for every $r > k \geq 1$, the class of queries whose restriction to such classes of databases can be computed by $RCM^r$ machines, strictly includes the class of queries whose restriction to classes of databases which are $C^k$-homogeneous but which are not strongly $C^k$-homogeneous can be computed by $RCM^r$ machines. Concerning the third notion, we say that a class of databases is *pairwise $C^k$-homogeneous* if for every pair of databases in the class, and for every pair of $k$-tuples taken respectively from the domains of the two databases, if both $k$-tuples satisfy the same properties expressible by $C^k$ formulas, then the two databases are isomorphic and there is an isomorphism mapping one tuple onto the other. We show that for every $k$, machines in $RCM^k$ working on such classes achieve completeness provided the classes are recursive.

Considering that equivalence in $C^k$ is decidable in polynomial time [Ott97], a very important line of research, which is quite relevant to complexity theory, is the identification of classes of graphs where $C^k$ equivalence coincides with isomorphism. These classes are the classes which we define as *pairwise $C^k$-homogeneous*, and include the class of trees [IL90] and the class of planar graphs [Gro98b]. In the study of $C^k$-homogeneity we intend to generalize this approach by defining a formal framework, and by considering not only those "optimal" classes, but also other properties which, still not being so powerful as to equate $C^k$ equivalence with isomorphism, do increase the computation power of the model $RCM^k$ to an important extent.

In Section 5, we investigate the relationship between our classes $\mathcal{QCQ}^{C^k}$ and recursive fragments of the infinitary logic $C^\omega_{\infty\omega}$. We prove that for every natural $k$, the restriction of $\mathcal{QCQ}^{C^k}$ to Boolean queries characterizes the expressive power of $C^k_{\infty\omega}$ restricted to sentences with recursive classes of models. As a corollary, we get a characterization of the expressive power of the model $RRM^k$ restricted to Boolean queries, for every natural $k$, in terms of the infinitary logic $\mathcal{L}^k_{\infty\omega}$. The characterization for the whole class of *relational machines* ($RM$) (and, hence, also of $RRM^{O(1)}$, given the equivalence of the two classes which was proved in [AV95]) in terms of the infinitary logic $\mathcal{L}^\omega_{\infty\omega}$ was proved in [AVV95], but the expressive power of each subclass of machines $RRM^k$ in terms of the corresponding fragment of the infinitary logic was unknown up to the author's knowledge.

Some of the results presented here have been included in [Tur01b].

An extended abstract of this article has been published as [Tur02].

## 2 Preliminaries

Unless otherwise stated, in the present article we will follow the usual notation in finite model theory, as in [EF99]. We define a *relational database schema*, or simply *schema*, as a set of relation symbols with associated arities. We do not allow constraints in the schema, and we do not allow constant symbols either. If $\sigma = \langle R_1, \ldots, R_s \rangle$ is a schema with arities $r_1, \ldots, r_s$, respectively, a *database instance* or simply *database* over the schema $\sigma$, is a structure $I = \langle D^I, R^I_1, \ldots, R^I_s \rangle$ where $D^I$ is a finite set which contains exactly all elements of the database, and for $1 \leq i \leq s$, $R^I_i$ is a relation of arity $r_i$, i.e., $R^I_i \subseteq (D^I)^{r_i}$. We will often use $dom(I)$ instead of

$D^I$. We define the *size* of the database $I$ as the cardinality of $D^I$, i.e., $|D^I|$. We will use $\simeq$ to denote the isomorphism relation. A *k-tuple* over a database $I$, for $k \geq 1$, is a tuple of length $k$ formed from elements of $dom(I)$. We will denote a $k$-tuple of $I$ by $\bar{a}_k$, or simply by $\bar{a}$. We use $\mathcal{B}_\sigma$ to denote the class of all *finite* databases of schema $\sigma$.

## 2.1 Computable Queries and Relational Machines

In this paper, we will consider *total* queries only. Let $\sigma$ be a schema, let $r \geq 1$, and let $R$ be a relation symbol of arity $r$. A *computable query of arity r and schema* $\sigma$ [CH80], is a total recursive function $q^r : \mathcal{B}_\sigma \to \mathcal{B}_{\langle R \rangle}$ which preserves isomorphisms and such that for every database $I$ of schema $\sigma$, $dom(q(I)) \subseteq dom(I)$. By preservation of isomorphisms we mean that for every $I, J \in \mathcal{B}_\sigma$ and for every isomorphism $f : dom(I) \to dom(J)$, $q(J) = f(q(I))$. A Boolean query is a 0-ary query. We denote the class of computable queries of schema $\sigma$ as $\mathcal{CQ}_\sigma$, and $\mathcal{CQ} = \bigcup_\sigma \mathcal{CQ}_\sigma$. Relational machines ($RM$) have been introduced in [AV95]. A $RM$ is a one-tape Turing machine ($TM$) with the addition of a *relational store* ($rs$) formed by a possibly infinite set of relations whose arity is *bounded* by some integer. The *only* way to access the relations in the $rs$ is through $FO$ (first order logic) formulas in the finite control of the machine. The input database as well as the output relation are in $rs$. In a transition of the machine one of these $FO$ formulas can be evaluated in $rs$. The resulting relation is then assigned to some relation symbol of the appropiate arity in the $rs$. The *arity* of a given relational machine is the maximum number of variables (free or bound) of any formula in its finite control.

Reflective relational machines ($RRM$) have been introduced in [APV98] as an extension of $RMs$. In an $RRM$, $FO$ queries are generated *during the computation* of the machine, and they are called *dynamic queries*. Each of these queries is written on a *query tape* and it is evaluated by the machine in one step. A further important difference to $RM$ is that in $RRM$ relations in the $rs$ can be of *arbitrary arity*. New relations can be created in rs during a computation, and they can be used in building the dynamic queries. An integer index can be used in the formulas to name a relation, and if that relation does not exist in rs it is created with the appropriate arity. The *variable complexity* of an $RRM$ is the maximum number of variables which may be used in the dynamic queries generated by the machine throughout any computation. We will denote as $RRM^k$, with $k \geq 1$, the sub-class of $RRM$ with variable complexity $k$. Furthermore, we define $RRM^{O(1)} = \bigcup_{k \geq 1} RRM^k$.

In [AVV97] it was shown that $RRM^{O(1)} = RM$, i.e., that the class of queries which can be computed by reflective relational machines of bounded variable complexity is exactly the same as the class of queries which can be computed by relational machines. However, it is *not* known whether for every $RRM$ of variable complexity $O(1)$ there exists an equivalent $RM$ whose arity is the *same* as the variable complexity of the given $RRM$. Moreover, this is strongly believed to be *not* true (see [AVV95], particularly Remark 3.3, and [AV95]).

## 2.2 Finite Model Theory and Databases

We refer the reader to [EF99] and [Imm99] for an in-depth study of finite model theory, and to [AHV94] for the relation between databases and finite model theory. We will use the notion of a *logic* in a general sense. A formal definition would only complicate the presentation and is unnecessary for our work. The interested reader can see [Ebb85] for a formal study of a general framework of abstract logics. As usual in finite model theory, we will regard a logic as a language, that is, as a set of formulas (see [EF99, AHV94]). We will only consider signatures, or vocabularies, which are purely *relational*. We will always assume that the signature includes a symbol for equality. We consider *finite* structures only. Consequently, if $\mathcal{L}$ is a logic, the notion of *satisfaction*, denoted as $\models_{\mathcal{L}}$, will be related to only finite structures. If $\mathcal{L}$ is a logic and $\sigma$ is a signature, we will denote by $\mathcal{L}_\sigma$ the class of formulas from $\mathcal{L}$ with signature $\sigma$. If $I$ is a structure of signature $\sigma$, or $\sigma$-*structure*, we define the $\mathcal{L}$ *theory of* $I$ as follows:

$$Th_{\mathcal{L}}(I) = \{\varphi \in \mathcal{L}_\sigma : I \models_{\mathcal{L}} \varphi\}$$

A *database schema* will be regarded as a *relational signature*, and a *database instance* of some schema $\sigma$ as a finite and relational $\sigma$-*structure*. If $\varphi$ is a sentence in $\mathcal{L}_\sigma$, we define

$$MOD(\varphi) = \{I \in \mathcal{B}_\sigma : I \models \varphi\}$$

By $\varphi(x_1, \ldots, x_r)$ we denote a formula of some logic whose free variables are *exactly* $\{x_1, \ldots, x_r\}$. Let $free(\varphi)$ be the set of free variables of the formula $\varphi$. If $\varphi(x_1, \ldots, x_k) \in \mathcal{L}_\sigma$, $I \in \mathcal{B}_\sigma$, $\bar{a}_k = (a_1, \ldots, a_k)$ is a $k$-tuple over $I$, let $I \models \varphi(x_1, \ldots, x_k)[a_1, \ldots, a_k]$ denote that $\varphi$ is TRUE, when interpreted by $I$, under a valuation $v$ where for $1 \leq i \leq k$ $v(x_i) = a_i$. Then we consider the set of all such valuations as follows:

$$\varphi^I = \{(a_1, \ldots, a_k) : a_1, \ldots, a_k \in dom(I) \wedge I \models \varphi(x_1, \ldots, x_k)[a_1, \ldots, a_k]\}$$

That is, $\varphi^I$ is the relation defined by $\varphi$ in the structure $I$, and its arity is given by the number of free variables in $\varphi$. Sometimes, we will use the same notation when the set of free variables of the formula is *strictly* included in $\{x_1, \ldots, x_k\}$. Formally, we say that a formula $\varphi(x_1, \ldots, x_k)$ of signature $\sigma$, *expresses* a query $q$ of schema $\sigma$, if for every database $I$ of schema $\sigma$, $q(I) = \varphi^I$. Similarly, a sentence $\varphi$ expresses a Boolean query $q$ if for every database $I$ of schema $\sigma$, is $q(I) = 1$ iff $I \models \varphi$. We will also deal with *extensions* of structures. If $R$ is a relation of arity $k$ in the domain of a structure $I$, we denote as $\langle I, R\rangle$ the $\tau$-structure resulting by adding the relation $R$ to $I$, where $\tau$ is obtained from $\sigma$ by adding a relation symbol of arity $k$. Similarly, if $\bar{a}_k$ is a $k$-tuple over $I$, we denote by $\langle I, \bar{a}_k\rangle$ the $\tau$-structure resulting by adding the $k$-tuple $\bar{a}_k$ to $I$, where $\tau$ is obtained from $\sigma$ by adding $k$ constant symbols $c_1, \ldots, c_k$, and where for $1 \leq i \leq k$, the constant symbol $c_i$ of $\tau$ is interpreted in $I$ by the $i$-th component of $\bar{a}_k$. This is the only case where we allow constant symbols in a signature. We denote by $FO^k$, where $k \geq 1$ is an integer, the fragment of $FO$ where only formulas whose variables, either free or bound, are in $\{x_1, \ldots, x_k\}$ are allowed. In this setting, $FO^k$ itself is a logic. This logic is obviously *less expressive* than $FO$. We denote as $C^k$ the logic which is obtained by adding to $FO^k$ *counting quantifiers*, i.e., all existential quantifiers of the form $\exists^{\geq m} x$ with $m \geq 1$. Informally, $\exists^{\geq m} x(\varphi)$ means that there are at least $m$ *different* elements of the database which satisfy $\varphi$.

## 2.3 Types

Given a database $I$ and a $k$-tuple $\bar{a}_k$ over $I$, we would like to consider *all* properties of $\bar{a}_k$ in the database $I$ including the properties of every component of the tuple and the properties of all different sub-tuples of $\bar{a}_k$. Therefore, we use the notion of *type.* Let $\mathcal{L}$ be a logic. Let $I$ be a database of some schema $\sigma$ and let $\bar{a}_k = (a_1, \ldots, a_k)$ be a $k$-tuple over $I$. The $\mathcal{L}$ *type* of $\bar{a}_k$ in $I$, denoted $tp_I^{\mathcal{L}}(\bar{a}_k)$, is the set of formulas in $\mathcal{L}_\sigma$ with free variables *among* $\{x_1, \ldots, x_k\}$ such that every formula in the set is TRUE when interpreted by $I$ for any valuation which assigns the $i$-th component of $\bar{a}_k$ to the variable $x_i$, for every $1 \leq i \leq k$. In symbols

$$tp_I^{\mathcal{L}}(\bar{a}_k) = \{\varphi \in \mathcal{L}_\sigma : free(\varphi) \subseteq \{x_1, \ldots, x_k\} \wedge I \models \varphi[a_1, \ldots, a_k]\}$$

It is noteworthy that, according to this definition, the $\mathcal{L}$ *theory* of the database $I$, i.e., $Th_{\mathcal{L}}(I)$, is included in the $\mathcal{L}$-type of *every* tuple over $I$. That is, the class of all the properties of a given tuple, which are expressible in $\mathcal{L}$, includes not only the properties of all its sub-tuples, but also the properties of the database itself.

Note that the type of two different $k$-tuples of the same database may be different, even if the types of their components are the same. Think of a complete binary tree of depth $h$. If we consider types for single elements (i.e., $k = 1$), then we have only $h + 1$ different types, because all the nodes of the same depth satisfy the same properties. They can be exchanged by an automorphism of the tree. Now let us consider types for pairs ($k = 2$). We can build two pairs, such that the type of the two first components is the same, and the type of the two second components is also the same, but the types of the two pairs are different. Just take for one pair a node in some depth $> 0$ and one of its sons, and for the other pair we take a node of the same depth as the first component of the first pair, and another node in the next level of the tree, but which is *not* the son of the first component.

We may also regard an $\mathcal{L}$-type as a *set of queries*, and even as a query. We can think of a type *without having a particular database in mind.* That is, we add properties (formulas with the appropiate free variables) as long as the resulting set remains consistent. Let us denote as $Tp^{\mathcal{L}}(\sigma, k)$ for some $k \geq 1$ the class of *all* $\mathcal{L}$-types for $k$-tuples over databases of schema $\sigma$. In symbols

$$Tp^{\mathcal{L}}(\sigma, k) = \{tp_I^{\mathcal{L}}(\bar{a}_k) : I \in \mathcal{B}_\sigma \wedge \bar{a}_k \in (dom(I))^k\}$$

Hence, $Tp^{\mathcal{L}}(\sigma, k)$ is a class of properties, or a set of sets of formulas. Let $\alpha \in Tp^{\mathcal{L}}(\sigma, k)$ (i.e., $\alpha$ is the $\mathcal{L}$-type of some $k$-tuple over some database in $\mathcal{B}_\sigma$). We say that a database $I$ *realizes* the type $\alpha$ if there is a $k$-tuple $\bar{a}_k$ over $I$ whose $\mathcal{L}$-type is $\alpha$. That is, if $tp_I^{\mathcal{L}}(\bar{a}_k) = \alpha$. We denote by $Tp^{\mathcal{L}}(I,k)$ the class of all $\mathcal{L}$-types for $k$-tuples which are realized in $I$. That is, it is the class of properties of all the $k$-tuples over the database $I$ which can be expressed in $\mathcal{L}$. In symbols

$$Tp^{\mathcal{L}}(I,k) = \{tp_I^{\mathcal{L}}(\bar{a}_k) : \bar{a}_k \in (dom(I))^k\}$$

The following well known fact (see [Ott97]) relates types of tuples with $C^k$-theories of databases (and also $FO$, see Fact 2).

**Fact 1.** For every $k > 0$, for every schema $\sigma$, and for every pair of databases $I$, $J$ of schema $\sigma$, the following holds:

$$I \equiv_{C^k} J \Longleftrightarrow Tp^{C^k}(I,k) = Tp^{C^k}(J,k)$$

□

The following fact means that when two databases realize the same $C^k$-types for tuples, it doesn't matter which length of tuples we consider. It is well known (see [Ott97]).

**Fact 2.**

- For every $k > 0$, for every schema $\sigma$, for every pair of databases $I$, $J$ of schema $\sigma$, and for every $1 \leq l \leq k$, the following holds:

$$Tp^{C^k}(I,k) = Tp^{C^k}(J,k) \Longleftrightarrow Tp^{C^k}(I,l) = Tp^{C^k}(J,l)$$

- For every schema $\sigma$, for every pair of databases $I$, $J$ of schema $\sigma$, and for every $l \geq 1$, the following holds:

$$I \equiv_{FO} J \Longleftrightarrow Tp^{FO}(I,l) = Tp^{FO}(J,l)$$

□

Note that the $\mathcal{L}$-type of the 0-tuple is the $\mathcal{L}$-theory of the database.

The following is a well known result which, among other sources, can be found as Proposition 2.1.1 in [EF99].

**Proposition 3.** *For every schema $\sigma$ and for every pair of (finite) databases $I$, $J$ of schema $\sigma$ the following holds:*

$$I \equiv_{FO} J \Longleftrightarrow I \simeq J$$

□

Although types are infinite sets of formulas, a *single* $C^k$ formula is equivalent to the $C^k$-type of a tuple over a given database. The equivalence holds for all databases of the same schema. This result has been proved by M. Otto.

**Proposition 4.** *([Ott96]): For every schema $\sigma$, for every database $I$ of schema $\sigma$, for every $k \geq 1$, for every $1 \leq l \leq k$, and for every $l$-tuple $\bar{a}_l$ over $I$, there is a $C^k$ formula $\chi \in tp_I^{C^k}(\bar{a}_l)$ such that for any database $J$ of schema $\sigma$ and for every $l$-tuple $\bar{b}_l$ over $J$*

$$J \models \chi[\bar{b}_l] \Longleftrightarrow tp_I^{C^k}(\bar{a}_l) = tp_J^{C^k}(\bar{b}_l)$$

□

Moreover, such a formula $\chi$ can be built inductively for a given database. If a $C^k$ formula $\chi$ satisfies the condition of Proposition 4, we call $\chi$ an *isolating formula* for $tp_I^{C^k}(\bar{a}_l)$.

Concerning logical characterizations of databases, the next two propositions are quite important and well known, and we will make use of them in the present work (see [Ott97] and [EF99]). Recall that we are considering only *finite* databases in the present article.

**Proposition 5.** *Let $\mathcal{L} \in \{FO^k, C^k\}$, for some $k \geq 1$. Let $I$ be a database of some schema $\sigma$. Then there exists a sentence $\alpha_I$ in $\mathcal{L}$ which characterizes $I$ up to $\equiv_{\mathcal{L}}$, i.e., for every database $J$ in $\mathcal{B}_\sigma$, the following holds:*

$$J \models \alpha_I \Longleftrightarrow I \equiv_{\mathcal{L}} J$$

□

**Proposition 6.** *Let $k \geq 1$. Then the following holds:*

*(i) $\equiv_{FO^k}$ coincides with $\equiv_{\mathcal{L}^k_{\infty\omega}}$, i.e., for every schema $\sigma$, and for every two databases $I$, $J$ in $\mathcal{B}_\sigma$:*

$$I \equiv_{FO^k} J \Longleftrightarrow I \equiv_{\mathcal{L}^k_{\infty\omega}} J$$

*(ii) $\equiv_{C^k}$ coincides with $\equiv_{\mathcal{C}^k_{\infty\omega}}$, i.e., for every schema $\sigma$, and for every two databases $I$, $J$ in $\mathcal{B}_\sigma$:*

$$I \equiv_{C^k} J \Longleftrightarrow I \equiv_{\mathcal{C}^k_{\infty\omega}} J$$

□

Let $\bar{a}_k = (a_1, \ldots, a_k)$ be a $k$-tuple over $I$. We say that the type $tp_I^{\mathcal{L}}(\bar{a}_k)$ is an *automorphism type* in the database $I$ if for every $k$-tuple $\bar{b}_k = (b_1, \ldots, b_k)$ over $I$, if $tp_I^{\mathcal{L}}(\bar{a}_k) = tp_I^{\mathcal{L}}(\bar{b}_k)$, then there exists an automorphism $f$ of the database $I$ which maps $\bar{a}_k$ onto $\bar{b}_k$, i.e., for $1 \leq i \leq k$, $f(a_i) = b_i$. Regarding the tuple $\bar{a}_k$ in the database $I$, the logic $\mathcal{L}$ is therefore sufficiently expressive with respect to the properties which might make $\bar{a}_k$ distinguishable from other $k$-tuples in the database $I$. We say that the type $tp_I^{\mathcal{L}}(\bar{a}_k)$ is an *isomorphism type* if for every database $J \in \mathcal{B}_\sigma$, and for every $k$-tuple $\bar{b}_k = (b_1, \ldots, b_k)$ over $J$, if $tp_I^{\mathcal{L}}(\bar{a}_k) = tp_J^{\mathcal{L}}(\bar{b}_k)$, then there exists an isomorphism $f : dom(I) \rightarrow dom(J)$ which maps $\bar{a}_k$ in $I$ onto $\bar{b}_k$ in $J$, i.e., for $1 \leq i \leq k$, $f(a_i) = b_i$.

### 2.4 Asymptotic Probabilities

See [EF99], among other sources. Let $\varphi$ be a sentence in $\mathcal{L}_\sigma$. We define

$$MOD_n(\varphi) = \{I \in \mathcal{B}_\sigma : dom(I) = \{1, \ldots, n\} \wedge I \models \varphi\}$$

Let's denote as $\mathcal{B}_{\sigma,n}$ the sub-class of databases of schema $\sigma$ with domain $\{1, \ldots, n\}$. We define the following limit, which we call *asymptotic probability of* $\varphi$:

$$\mu_\varphi = \lim_{n \to \infty} (|MOD_n(\varphi)| / |\mathcal{B}_{\sigma,n}|)$$

We say that a logic $\mathcal{L}$ has a 0–1 *Law* if for every sentence $\varphi \in \mathcal{L}$ $\mu_\varphi$ exists, and is either 0 or 1. The same notion can also be defined on classes of databases, or Boolean queries. This means that the asymptotic probability of every property which can be expressed in the formalism (or of the given class) always exists, and is either 0 or 1. Among other logics, $FO$ has this Law.

## 3 Definition of the Hierarchy

One of the main reasons for the weakness of $FO^k$ regarding expressibility of queries, is its inability to count beyond the bound given by $k$. For instance, note that we need $k+2$ different variables to express that a node in a graph has out degree exactly $k$. Hence, it seems quite natural to add to $FO^k$ the capability to count beyond that bound, while still restricting the number of different variables which may be used in a formula. In this way we get the logic $C^k$ (see Section 2), which turns out to be much more expressive than $FO^k$ (see [EF99] and [Imm99]). In logics with counting, 2 variables are enough to express *any* out degree of a node in a graph. Then, we define in this section a hierarchy which is similar to the one defined in [Tur01a, Tur04], but for which we consider the logics $C^k$ instead of $FO^k$. In this way we get a new hierarchy whose layers are much bigger than the corresponding layers in the hierarchy of [Tur01a, Tur04]. In Section 5 we compare the two hierarchies.

**Definition 7.** *Let $\sigma$ be a database schema and let $k \geq 1$ and $k \geq r \geq 0$. Then we define*

$$\mathcal{QCQ}_\sigma^{C^k} = \{f^r \in \mathcal{CQ}_\sigma \mid \forall I, J \in \mathcal{B}_\sigma :$$

$$Tp^{C^k}(I,k) = Tp^{C^k}(J,k) \Longrightarrow Tp^{C^k}(\langle I, f(I)\rangle, k) = Tp^{C^k}(\langle J, f(J)\rangle, k)\}$$

*where $\langle I, f(I)\rangle$ and $\langle J, f(J)\rangle$ are databases of schema $\sigma \cup \{R\}$, with $R$ being a relation symbol of arity $r$.*

*We also require that the answer to the query $f$ must be the union of complete $C^k$-types, i.e., for all databases $I$ in $\mathcal{B}_\sigma$, and for all tuples $\bar{a}, \bar{b}$ in $dom(I)^r$, if $\bar{a} \in f(I)$ and $tp_I^{C^k}(\bar{a}) = tp_I^{C^k}(\bar{b})$, then also $\bar{b} \in f(I)$.*

*We define further $\mathcal{QCQ}^{C^k} = \bigcup_\sigma \mathcal{QCQ}_\sigma^{C^k}$ and $\mathcal{QCQ}^{C^\omega} = \bigcup_{k \geq 1} \mathcal{QCQ}^{C^k}$.*

That is, a query is in the sub-class $\mathcal{QCQ}^{C^k}$ if it *preserves realization* of $C^k$-types for $k$-tuples. By preservation of $C^k$-types realization, we mean the property that for every pair of databases of the corresponding schema, say $\sigma$, and for every $C^k$-type for $k$-tuples (i.e., for every type in $Tp^{C^k}(\sigma, k)$), if both databases have the *same number* of $k$-tuples of that type then the relations defined by the query in each database also agree in the same sense, considering the schema of the databases with the addition of a relation symbol with the arity of the query. Note the difference with the classes $\mathcal{QCQ}^k$ in ([Tur01a], [Tur04]), where the cardinalities of the corresponding sets of tuples may be different. By Fact 1 equality in the set of $C^k$-types for $k$-tuples realized in two given databases is equivalent to $\equiv_{C^k}$, i.e., equality of $C^k$ theories. Moreover, by Fact 2 the size of the tuples which we consider for the types is irrelevant. Thus, queries in $\mathcal{QCQ}^{C^k}$ may also be regarded as those which *preserve equality of $C^k$ theories.* Note that the different classes $\mathcal{QCQ}^{C^k}$ form a hierarchy, i.e., for every $k \geq 1$, $\mathcal{QCQ}^{C^k} \subseteq \mathcal{QCQ}^{C^{k+1}}$. This follows from the notion of $C^k$-type and from the definition of the classes $\mathcal{QCQ}^{C^k}$. It can be also obtained as a straightforward corollary of Theorem 13.

Hence, queries in $\mathcal{CQ}$ may be considered as ranging from those for whose computation we need to consider every property of the input database up to isomorphism (i.e., every $FO$ property), to those for whose computation it is enough to consider the $C^k$ properties of the input database, for some fixed $k$. Different sub-classes $\mathcal{QCQ}^{C^k}$ in the hierarchy $\mathcal{QCQ}^{C^\omega}$ correspond to different degrees of "precision" with which we *need* to consider the input database to evaluate the queries in the sub-class on that database.

Next, we give an important result from [CFI92] which we will use in most of our proofs. Then, we show that the hierarchy defined by the sub-classes $\mathcal{QCQ}_\sigma^{C^k}$, for $k \geq 1$, is *strict.*

**Proposition 8.** *([CFI92]) For every $k \geq 1$, there are two non isomorphic graphs $G_k$, $H_k$, such that $G_k \equiv_{C^k} H_k$.* $\square$

**Proposition 9.** *For every $k \geq 1$, there is some $h > k$ such that $\mathcal{QCQ}_\sigma^{C^h} \supset \mathcal{QCQ}_\sigma^{C^k}$.*

*Proof.* The inclusion is trivial and can also be easily obtained as a corollary to Theorem 13. For the strict inclusion we will use the graphs $G_k$, $H_k$ of Proposition 8. Note that by Proposition 3, for every pair of the graphs $G_k$, $H_k$ there exists an integer $h > k$ such that the $C^h$-types are $FO$-types for both graphs. Let us write $h$ as $h(k)$. Then, for every $k \geq 1$, by Proposition 8 there are nodes in one of the graphs, say $G_k$, whose $C^{h(k)}$-types are not realized in the other graph $H_k$. Then we define for every $k \geq 1$, the query $f_k$ in the schema of the graphs, say $\sigma$, as the nodes of the input graph whose $C^{h(k)}$-types are not realized in $H_k$. We will show first that $f_k \in \mathcal{QCQ}_\sigma^{C^{h(k)}}$. Let $I$, $J$ be an arbitrary pair of graphs with $I \equiv_{C^{h(k)}} J$. If they are $C^{h(k)}$ equivalent to $H_k$, then the result of $f_k$ will be the empty set for both graphs. If they are not $C^{h(k)}$ equivalent to $H_k$, then clearly the nodes in the result of $f_k$ will have the same $C^{h(k)}$-types in both graphs. So, $f_k$ preserves realization of $C^{h(k)}$-types and hence $f_k \in \mathcal{QCQ}_\sigma^{C^{h(k)}}$. Now, we will show that $f_k \notin \mathcal{QCQ}_\sigma^{C^k}$. To see that, note that $f_k(H_k) = \emptyset$ by definition of $f_k$, but by Proposition 8 and by our assumption $f_k(G_k) \neq \emptyset$ and $H_k \equiv_{C^k} G_k$. Thus, $f_k$ does not preserve realization of $C^k$-types and hence $f_k \notin \mathcal{QCQ}_\sigma^{C^k}$. $\square$

**Proposition 10.** *$\mathcal{QCQ}^{C^\omega} \subset \mathcal{CQ}$.*

*Proof.* The inclusion is trivial, since clearly every query in $\mathcal{QCQ}^{C^\omega}$ is computable. For the strict inclusion we will use again the graphs $G_k$, $H_k$ of Proposition 8. We define a query $f$ on the schema of the pairs of disjoint graphs, say $\sigma$, as the nodes in the first graph, whose $FO$-type is not realized in the second graph. Clearly, $f$ is computable and total. Now, towards a contradiction, let us assume that $f \in \mathcal{QCQ}^{C^\omega}$. Then, for some $h \geq 1$, $f \in \mathcal{QCQ}_\sigma^{C^h}$. For some order of the pairs of corresponding graphs as in Proposition 8, say $(G_h, H_h)$, the result of $f$ is non empty, by the definition of $f$. If we consider now the pair $(G_h, G_h)$, the result of $f$ is empty. Since $(G_h, H_h) \equiv_{C^h} (G_h, G_h)$, it turns out that $f \notin \mathcal{QCQ}_\sigma^{C^h}$, which is a contradiction. Thus, $f \notin \mathcal{QCQ}^{C^\omega}$. $\square$

## 3.1 A Reflective Machine for Logics with Counting.

We will now define a model of computation to characterize the sub-classes $\mathcal{QCQ}_\sigma^{C^k}$.

In [Ott96], M. Otto defined a new model of computation of queries inspired by the *RM* of [AV95], to characterize the expressive power of fixed point logic with *counting terms* (see [Imm99]). Here, we define a machine which is similar to the model of Otto, but which is inspired by the *RRM* of [APV98], instead. In this paper, we will not compare the expressive power of the model of Otto and ours. However, it is straightforward to prove that the machine of Otto can be simulated in our model.

**Definition 11.** *For every $k \geq 1$, we define the* reflective counting machine of variable complexity *$k$ which we denote by $RCM^k$, as a reflective relational machine $RRM^k$ where dynamic queries are $C^k$ formulas, instead of $FO^k$ formulas. In all other aspects, our model works in exactly the same way as $RRM^k$. We define $RCM^{O(1)} = \bigcup_{k\geq 1} RCM^k$.*

We need first a technical result. Then, we can prove the characterization of the expressive power of the model $RCM^k$.

Let $\varphi$ be a formula of some logic $\mathcal{L}$, where the relation symbols $R_1, \ldots, R_k$, with arities $r_1, \ldots, r_k$, are used. Let $\varphi_1(x_{11}, \ldots, x_{1r_1})$, …, $\varphi_k(x_{k1}, \ldots, x_{kr_k})$ be also formulas in $\mathcal{L}$. We say that *$\hat{\varphi}$ is obtained from $\varphi$ by composition with $\varphi_1, \ldots, \varphi_k$*, if for every $1 \leq i \leq k$, every occurrence of an atomic formula of the form $R_i(z_1, \ldots, z_{r_i})$ in $\varphi$, is replaced by the formula $\varphi_i$ in such a way that, for all $1 \leq j \leq r_i$, each occurrence of the free variable $x_{ij}$ in $\varphi_i$ is replaced by the variable $z_j$.

**Lemma 12.** *Let $k \geq 1$, let $\sigma$ be a schema, let $I$ be a database of schema $\sigma$ and let $\mathcal{M}$ be an $RCM^k$ which computes a query of schema $\sigma$. Then, there is a formula $\varphi_{\mathcal{M},I}$ in $C^k$ which defines on $I$ the relation resulting from the computation of $\mathcal{M}$ on input $I$. Moreover, $\varphi_{\mathcal{M},I}$ depends only on $\mathcal{M}$ and $I$.*

*Proof.* The only transitions of $\mathcal{M}$ which may affect the content of the output relation in the relational store of $\mathcal{M}$, are those transitions where a $C^k$ formula is evaluated, and the relation defined by the evaluation is assigned to some relation symbol in the relational store. Informally, in every computation step where a $C^k$ formula, say $\psi$, is evaluated, we can use composition as defined above, replacing each relation symbol $R$ which is used in $\psi$, and which is not in $\sigma$, by the last $C^k$ formula whose resulting relation from its evaluation on the relational store was assigned to $R$. Then, by induction on the length of the computation of $\mathcal{M}$ on input $I$, and by applying composition, it is straightforward to prove that we get finally a $C^k$ formula $\varphi_{\mathcal{M},I}$ which, evaluated on $I$, defines the same output relation as the computation of $\mathcal{M}$ on input $I$. And clearly this formula depends only on $\mathcal{M}$ and $I$. Finally, note that $C^k$ is closed under composition, as defined above. □

**Theorem 13.** *For every $k \geq 1$, the class of total queries which are computable by $RCM^k$ machines is exactly the class $\mathcal{QCQ}^{C^k}$.*

*Proof.* **a)** ($\subseteq$): Suppose that an $r$-ary query $f$ of schema $\sigma$ is computable by a $RCM^k$ $\mathcal{M}$, for some $k \geq r$. And let $I$ and $J$ be two databases of schema $\sigma$ such that $Tp^{C^k}(I,k) = Tp^{C^k}(J,k)$. According to the definition of the $RCM^k$ machine, the only way to assign a relation to an $r$-ary relation symbol in the relational store is through a $C^k$ formula with $r$ free variables, say $\varphi$. And, by Fact 18 the result of the evaluation of $\varphi$ on the database in the relational store of $\mathcal{M}$ is the projection of the union of some equivalence classes in the relation of equality of $C^k$-types for $k$-tuples over $I$. That is, $\varphi$ is equivalent to the disjunction of some isolating formulas of $C^k$-types for $r$-tuples. But, by definition, the isolating formulas express the same $C^k$-types in every database of the corresponding schema. Thus, the $r$-tuples from $dom(J)$ which form the relation induced by $\varphi$ in $J$ must be those with the same $C^k$-types as the $r$-tuples from $dom(I)$ which form the relation induced by $\varphi$ in $I$. So $Tp^{C^k}(\langle I, f(I)\rangle, r) = Tp^{C^k}(\langle J, f(J)\rangle, r)$. By Fact 2, also $Tp^{C^k}(\langle I, f(I)\rangle, k) = Tp^{C^k}(\langle J, f(J)\rangle, k)$, so $f \in \mathcal{QCQ}^{C^k}$. Note that the only way for the machine not to evaluate the same formula $\varphi$ for both databases is the existence of a $C^k$ sentence which evaluates to different truth values on $I$ and $J$. But this is not possible by Fact 1 because $I \equiv_{C^k} J$.

**b)** ($\supseteq$): Let $f \in \mathcal{QCQ}^{C^k}$ be an $r$-ary query of schema $\sigma$ for some $k \geq r$. We build an $RCM^k$ machine $\mathcal{M}_f$, which will compute $f$. We use a countably infinite number of $k$-ary relation symbols in its relational store. With the input database $I$ in its relational store, $\mathcal{M}_f$ will build an encoding of a database $I'$ in its $TM$ tape such that $Tp^{C^k}(I,k) = Tp^{C^k}(I',k)$. For this purpose, $\mathcal{M}_f$ will work as follows:

(i) $\mathcal{M}_f$ finds out the size of $I$, say $n$. Note that this can be done through an iteration by varying $m$ in the query $\exists^{\geq m} x(x = x) \wedge \neg\exists^{\geq m+1} x(x = x)$ which is in $C^1$.

(ii) Then $\mathcal{M}_f$ builds an encoding of every possible database $I'$ of schema $\sigma$ and of size $n$ in its $TM$ tape with domain $\{1, \ldots, n\}$.

(iii) For every $I'$ and for every $k$-tuple $s_i$ over $I'$, $\mathcal{M}_f$ builds on its $TM$ tape the isolating formula $\chi_{s_i}$ (as in Proposition 4) for the $C^k$-type of $s_i$ in the database $I'$, and in this way we get isolating formulas for the types in $Tp^{C^k}(I', k)$.

(iv) $\mathcal{M}_f$ evaluates as dynamic queries the formulas $\chi_{s_i}$, for every $i$, which are in $C^k$ and assigns the results to the working $k$-ary relation symbols $S_i$ in the relational store, respectively (note that these queries are evaluated on the input database $I$).

(v) If every relation $S_i$ is non-empty, and if the union of all of them is the set of $k$-tuples over $I$, then it means that $Tp^{C^k}(I, k) = Tp^{C^k}(I', k)$ and $I'$ is the database we were looking for; otherwise, we try another database $I'$. Note that $\mathcal{M}_f$ can check whether the relation $S_i$ is empty with the dynamic query $\exists x_1 \dots x_k (S_i(x_1, \dots, x_k))$.

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

(vii) $\mathcal{M}_f$ builds $f^k(I)$ in the relational store as the union of the relations $S_i$ which correspond to the $C^k$-types $\chi_{s_i}$ of the $k$-tuples $s_i$ which form $f^k(I')$, and finally it reduces the $k$-ary relation $f^k(I)$ to an $r$-ary relation $f(I)$.

□

**Corollary 14.** $RCM^{O(1)} = \mathcal{QCQ}^{C^\omega}$. □

**Corollary 15.** *The computation model $RCM^{O(1)}$ is incomplete. That is, there are computable queries which cannot be computed by any $RCM^{O(1)}$ machine.* □

**Corollary 16.** *Let $f \in \mathcal{CQ}$, of some schema $\sigma$. Then, for every $k \geq 1$ and for every natural number $i$, if $f$ preserves realization of $C^k$-types for $k$-tuples for every pair of databases in $\mathcal{B}_\sigma$, then $f$ also preserves realization of $C^{k+i}$-types for $(k+i)$-tuples for every pair of databases in $\mathcal{B}_\sigma$.* □

**Remark 17.** The hierarchy defined by the classes $\mathcal{QCQ}^{C^k}$ is not only *strict*, but it is *orthogonal* to the hierarchy of complexity classes defined in terms of TIME and SPACE of Turing machines (like $LOGSPACE \subseteq PTIME \subseteq NP \subseteq PSPACE \subseteq EXPTIME$). This is also the case with the classes $\mathcal{QCQ}^k$ of ([Tur01a], [Tur04]). Note that *any* recursive predicate, evaluated on the number of equivalence classes in the (equivalence) relation defined by equality of $C^k$-types in the set of $k$-tuples of a database, is in $\mathcal{QCQ}^{C^k}$. Therefore, there is no complexity class defined by any bounds in TIME or SPACE of Turing machines which may include $\mathcal{QCQ}^{C^k}$. And this is the case for every $k \geq 1$. In Section 5 we make some considerations to this regard.

## 4 Homogeneity in Logics with Counting

In this section, we will study some properties of databases which are relevant to the computation power of the machines $RCM^{O(1)}$, when working on databases with such properties. That is, we will prove that there are important queries which cannot be computed by an $RCM^{O(1)}$ on the whole class of databases of a given schema, but whose restriction to certain classes of databases are computable by such machines.

First, we will give some definitions and well known facts which we need (see [Ott97]). Let $k$, $l$ be positive integers such that $k \geq l \geq 1$. Let us denote by $\equiv_k$ the (equivalence) relation induced on the set of $l$-tuples over a given database $I$ by equality of $C^k$-types of $l$-tuples. That is, for every pair of $l$-tuples $\bar{a}_l$ and $\bar{b}_l$ over $I$, $\bar{a}_l \equiv_k \bar{b}_l$ iff $tp_I^{C^k}(\bar{a}_l) = tp_I^{C^k}(\bar{b}_l)$.

**Fact 18.** For every schema $\sigma$, for every database $I$ of schema $\sigma$, for every $k \geq 1$, for every $1 \leq r \leq k$, and for every $C^k$ formula $\varphi$ of signature $\sigma$, with $r$ free variables, the relation which $\varphi$ defines on $I$ is the projection of the union of some equivalence classes in the relation $\equiv_k$ for $k$-tuples over $I$. $\square$

When the query which $\varphi$ expresses is of arity $k$, i.e., when $r = k$, the result simply means that the query cannot distinguish between two $k$-tuples whose $C^k$-types are the same. On the other hand, if $r < k$, the intuitive idea is the same, but of course the $k$-tuples in the equivalence classes of $\equiv_k$ must be reduced to $r$-tuples in some uniform way.

So that the equivalence classes in the relation $\equiv_k$ for $r$-tuples, are unions of equivalence classes in the relation $\equiv_k$ for $k$-tuples, reduced to $r$-tuples in some uniform way. And this is what the first part of the following fact shows.

**Fact 19.**

- For every schema $\sigma$, for every database $I$ of schema $\sigma$, for every $k \geq 1$, for every $1 \leq r \leq k$, every equivalence class in the relation $\equiv_k$ for $r$-tuples over $I$, is the projection of the union of some equivalence classes in the relation $\equiv_k$ for $k$-tuples over $I$.
- For every schema $\sigma$, for every database $I$ of schema $\sigma$, for every $k \geq 1$, for every $1 \leq r \leq k$, every equivalence class in the relation $\equiv_r$ for $r$-tuples over $I$, is the projection of the union of some equivalence classes in the relation $\equiv_k$ for $k$-tuples over $I$. $\square$

From Fact 19 it follows that allowing more variables in isolating formulas for $C^k$-types of tuples results in a more precise view of the properties which identify a given sub-set of tuples among the whole set of tuples which may be built over the database. By Proposition 3, we know that the limit is $FO$ which includes the logic $C^k$ for every natural $k$. Types in $FO$ are isomorphism types (and, hence, also automorphism types) for tuples of every length in every database. So, we want to consider the number of variables which are needed for a given database and for a given integer $k$ to express in $FO$ with counting quantifiers the properties of the $k$-tuples in that database up to automorphism. For this purpose we define different variants of the notion of homogeneity from model theory (see [EF99] and [CK92]) in the context of logics with counting. For every $k \geq 1$ and for any two $k$-tuples $\bar{a}_k$ and $\bar{b}_k$ over a given database $I$, we will denote as $\equiv_{\simeq}$ the (equivalence) relation defined by the existence of an automorphism in the database $I$ mapping one $k$-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 $1 \leq i \leq k$, $f(a_i) = b_i$.

Let $k \geq 1$. A database $I$ is *$C^k$-homogeneous* if for every pair of $k$-tuples $\bar{a}_k$ and $\bar{b}_k$ over $I$, if $\bar{a}_k \equiv_k \bar{b}_k$, then $\bar{a}_k \equiv_{\simeq} \bar{b}_k$. Let $\sigma$ be a schema. A class $\mathcal{C}$ of databases of schema $\sigma$ is *$C^k$-homogeneous* if every database $I \in \mathcal{C}$ is $C^k$-homogeneous.

The next fact is immediate (see [Ott97]).

**Fact 20.** Let $k \geq 1$. If a database $I$ is $C^k$-homogeneous, then for every $1 \leq l \leq k$ and for every pair of $l$-tuples $\bar{a}_l$ and $\bar{b}_l$ over $I$, if $\bar{a}_l \equiv_k \bar{b}_l$, then $\bar{a}_l \equiv_{\simeq} \bar{b}_l$. □

We define next a presumably stronger notion regarding homogeneity: *strong $C^k$-homogeneity*. To the author's knowledge it is not known whether there exist examples of classes of databases which are $C^k$-homogeneous for some $k$, but which are not strongly $C^k$-homogeneous. This was also the case with the analogous notions in ([Tur01a], [Tur04]). However, the consideration of strong $C^k$-homogeneity makes sense not only because of the intuitive appeal of the notion, but also because this is the property which we use in Proposition 27 to prove that the class $\mathcal{QCQ}^C$ (see Definition 26) is a lower bound with respect to the increment in computation power of the machine $RCM^k$ when working on strongly $C^k$-homogeneous databases. Up to know, we could not prove that this result holds for $C^k$-homogeneous databases as well. Let $k \geq 1$. A database $I$ is *strongly $C^k$-homogeneous* if it is $C^r$-homogeneous for every $r \geq k$. Let $\sigma$ be a schema. A class $\mathcal{C}$ of databases of schema $\sigma$ is *strongly $C^k$-homogeneous* if every database $I \in \mathcal{C}$ is strongly $C^k$-homogeneous.

Note that, by Proposition 3, every database is strongly $C^k$-homogeneous for some $k \geq 1$. However, it is clear that this is not the case with *classes* of databases.

In [Ott96] it was noted that the discerning power of the machine defined there, which is similar to our machine $RCM^k$ (see discussion before Definition 11), is restricted to $C^k$-types for $k$-tuples. So, the following result seems quite natural, and is somehow implicit in Otto's work. If $f$ is a query of schema $\sigma$, and $\mathcal{C}$ is a class of databases of schema $\sigma$, we will denote as $f|_{\mathcal{C}}$ the restriction of $f$ to the class $\mathcal{C}$.

**Theorem 21.** *For every schema $\sigma$ and for every $k \geq 1$, there is a query $f$ such that if $\mathcal{C}$ and $\mathcal{C}'$ are any two classes of databases of schema $\sigma$ such that $\mathcal{C}$ is $C^k$-homogeneous and $\mathcal{C}'$ is not $C^k$-homogeneous, then $f|_{\mathcal{C}}$ is computable by an $RCM^k$ machine but $f|_{\mathcal{C}'}$ is not computable by any $RCM^k$ machine.*

*Proof.* For every $k \geq 1$, and for every database $I$, we define the Boolean query $f$ as follows: $f(I) = TRUE$ *iff the number of equivalence classes in the relation $\equiv_{\simeq}$ for $k$-tuples over $I$ is even.* The query $f$ is clearly computable by an $RCM^k$ machine on classes of databases which are $C^k$-homogeneous. To see this, note that we can use the same construction as in the proof of Theorem 13 to build the isolating formulas for $C^k$-types for $k$-tuples, which for $C^k$-homogeneous classes of databases, by definition, are also automorphism types for $k$-tuples. Note that different relation symbols $S_i$ of item *iv* in that proof can have the same contents, since there can be several $k$-tuples of the same $C^k$-types in $I$. For each pair $S_i, S_j$ of those relations, $\mathcal{M}_f$ can check whether they are different simply with the query $\forall x_1 \dots x_k (S_i(x_1, \dots, x_k) \leftrightarrow S_j(x_1, \dots, x_k))$. Then, $\mathcal{M}_f$ can count on its TM tape the number of different relations $S_i$. Note that this is the number of $C^k$-types for $k$-tuples realized in $I$, and as $I \in \mathcal{C}$ and $\mathcal{C}$ is $C^k$-homogeneous, this is also the number of equivalence classes in $\equiv_{\simeq}$.

On the other hand, for some $J \in \mathcal{C}'$, $J$ is not $C^k$-homogeneous. Then in that database $J$, the number of $C^k$-types realized is different than the number of equivalence classes in $\equiv_{\simeq}$ since, by definition, there are at least two $k$-tuples $\bar{a}, \bar{b}$ in $J$ such that $\bar{a} \equiv_k \bar{b}$ but $\bar{a} \not\equiv_{\simeq} \bar{b}$. And, as we saw in Theorem 13, no $RCM^k$ machine can distinguish between the $k$-tuples $\bar{a}, \bar{b}$ in $J$, so that no such machine can count the number of equivalence classes in $\equiv_{\simeq}$ on that database. □

**Remark 22.** Note that the sub-class of queries whose restriction to $C^k$-homogeneous databases can be computed by $RCM^k$ machines, but which *cannot* be computed by such machines on arbitrary classes of databases, is quite big. As we saw in the proof of the previous theorem, an $RCM^k$ machine can count the number of equivalence classes in $\equiv_{\simeq}$ for $k$-tuples on a $C^k$-homogeneous database as an intermediate result on its $TM$ tape. Then we can use this parameter, which we will call *type index for k-tuples* following [AV95], as the argument for *any* recursive predicate in an $RCM^k$ machine. Thus, the *whole* class of recursive predicates, evaluated on the type index for $k$-tuples of the input database, is included in the sub-class of queries whose restriction to $C^k$-homogeneous databases can be computed with $RCM^k$ machines, but which cannot be computed by such machines on arbitrary classes of databases.

However, we still do not know whether the machine $RCM^k$ can achieve *completeness* when computing queries on databases which are $C^k$-homogeneous. This problem has important consequences in query computability and complexity, and is also related to the expressibility of fixed point logics (see [Ott96]). In particular, it is related to the problem of knowing whether whenever two databases are $C^k$-homogeneous, and are also $C^k$ equivalent then they are isomorphic.

Next, we will show that the property of *strong* $C^k$-homogeneity will allow $RCM^{O(1)}$ machines to extend their power with respect to computability of queries.

**Proposition 23.** *For every schema $\sigma$ and for every $k \geq 1$, there is a class of queries $F = \{f_r\}_{r \geq k}$ such that, if $\mathcal{C}$ and $\mathcal{C}'$ are any two classes of databases of schema $\sigma$ such that $\mathcal{C}$ is strongly $C^k$-homogeneous and $\mathcal{C}'$ is not strongly $C^k$-homogeneous, then for every $f_r \in F$, $f_r|_{\mathcal{C}}$ is computable by an $RCM^r$ machine, but it is* not *the case that for every $f_r \in F$, $f_r|_{\mathcal{C}'}$ is computable by an $RCM^r$ machine.*

*Proof.* We can use here the same strategy as we used in the proof of Theorem 21. For every $r \geq k$, we define $f_r$ as follows: $f_r(I) = TRUE$ *iff the number of* *equivalence classes in the relation $\equiv_{\simeq}$ for $r$-tuples over $I$ is even.* Clearly, for $I \in \mathcal{C}$, and for every $r \geq k$, the equivalence classes in $\equiv_r$ coincide with the equivalence classes in $\equiv_{\simeq}$ for $r$-tuples over $I$. So, for each $r \geq k$, $f_r$ can be computed by an $RCM^r$ machine, as we did in the proof of Theorem 21.

On the other hand, for some $J \in \mathcal{C}'$, which is not strongly $C^k$-homogeneous, there wil be some $r \geq k$, such that the equivalence classes in $\equiv_r$ will not coincide with the equivalence classes in $\equiv_{\simeq}$ for $r$-tuples over $J$. Then no $RCM^r$ machine will be able to distinguish between $r$-tuples which have the same $C^r$-type, and hence no $RCM^r$ machine will be able to count the equivalence classes in $\equiv_{\simeq}$ for $r$-tuples over $J$. □

With the next notion of homogeneity we intend to formalize the property of the classes of databases where $C^k$ equivalence coincides with isomorphism. Among other classes, this is the case with the class of trees ([IL90]), the class of planar graphs ([Gro98b]) and the class of databases of bounded tree-width ([GM98]). With this stronger notion we can achieve *completeness* with $RCM^k$ machines.

**Definition 24.** *Let $\sigma$ be a schema and let $\mathcal{C}$ be a class of databases of schema $\sigma$. Let $k \geq 1$. We say that $\mathcal{C}$ is* pairwise $C^k$-homogeneous, *if for every pair of databases $I$, $J$ in $\mathcal{C}$, and for every pair of $k$-tuples $\bar{a}_k \in (dom(I))^k$ and $\bar{b}_k \in (dom(J))^k$, if $\bar{a}_k \equiv_k \bar{b}_k$, then there exists an isomorphism $f : dom(I) \longrightarrow dom(J)$ such that $f(a_i) = b_i$ for every $1 \leq i \leq k$.*

**Proposition 25.** *For every schema $\sigma$, for every $k \geq 1$ and for every query $f$ of schema $\sigma$ in $\mathcal{CQ}$, if $\mathcal{C}$ is a recursive and pairwise $C^k$-homogeneous class of databases of schema $\sigma$, then there is an $RCM^k$ machine which computes $f|_{\mathcal{C}}$.*

*Proof.* We use the same strategy to build an $RCM^k$ machine $\mathcal{M}_f$ which computes a given query $f \in \mathcal{CQ}$, as we did in the proof of Theorem 13. In this case we must also check that the database $I'$ which we build in the TM tape is in the class of databases 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 database) realize the same $C^k$-types for $k$-tuples, then the two databases 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 of the $k$-tuples which realize in $I$ the $C^k$-types for $k$-tuples which are realized in $f(I')$. □

## 4.1 A Lower Bound for Strong Homogeneity

The queries which preserve realization of $FO$-types for tuples (i.e., the computable queries), but which do not preserve realization of $C^k$-types for $k$-tuples for any $k$, do not belong to *any* $\mathcal{QCQ}^{C^k}$ class. 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 databases may be different. And the number of different bounds for the number of variables may be infinite. Thus, we define a new class of queries, following the same strategy as in [Tur01a, Tur04] regarding $FO^k$. The intuitive idea behind this new class is that queries which belong to it preserve, for any two databases of the corresponding schema, the realization of types in $C^k$ where $k$ is the number of variables which is *sufficient* for *both* databases to define tuples up to automorphism.

**Definition 26.** *For any schema $\sigma$, let us denote as $arity(\sigma)$ the maximum arity of a relation symbol in the schema. We define the class $\mathcal{QCQ}^C$ as the class of queries $f \in \mathcal{CQ}$ of some schema $\sigma$ and of any arity, for which there exists an integer $n \geq max\{arity(f), arity(\sigma)\}$ such that for every pair of databases $I$, $J$ in $\mathcal{B}_\sigma$, the following holds:*

$$Tp^{C^h}(I,h) = Tp^{C^h}(J,h) \Longrightarrow Tp^{C^h}(\langle I, f(I)\rangle, h) = Tp^{C^h}(\langle J, f(J)\rangle, h)$$

*where $\langle I, f(I)\rangle$ and $\langle J, f(J)\rangle$ are databases of schema $\sigma \cup \{R\}$ with $R$ being a relation symbol with the arity of the query $f$, and $h = max\{n, min\{k : I \text{ and } J \text{ are strongly } C^k\text{-homogeneous}\}\}$.*

*We also require that the answer to the query $f$ must be the union of complete $C^h$-types, i.e., for all databases $I$ in $\mathcal{B}_\sigma$, and for all tuples $\bar{a}, \bar{b}$ in $dom(I)^r$, if $\bar{a} \in f(I)$ and $tp_I^{C^h}(\bar{a}) = tp_I^{C^h}(\bar{b})$, then also $\bar{b} \in f(I)$.*

Clearly, $\mathcal{QCQ}^C \supseteq \mathcal{QCQ}^{C^\omega}$. But, unlike the analogous class $\mathcal{QCQ}$ in [Tur01a, Tur04], we do not know neither whether the inclusion is strict, nor whether $\mathcal{CQ}$ strictly includes $\mathcal{QCQ}^C$. Both questions seem to be non trivial since they are related to the problem which we mentioned after Remark 22, by Proposition 27 below.

Note that the queries based on the $C^k$-type index for $k$-tuples which we mentioned in Remark 22 are in $\mathcal{QCQ}^C$. Therefore, the class of queries which can be computed by $RCM^{O(1)}$ machines on classes of databases which are $C^k$-homogeneous is actually quite big. It is big enough to allow for a $RCM^k$ machine to go through the whole hierarchy $\mathcal{QCQ}^{C^\omega}$, in a sense which will be clarified next, and, further, to get into $\mathcal{QCQ}^C$. This is actually quite natural, because classes of databases which are strongly $C^k$-homogeneous will not benefit from the possibility of using $> k$ variables in dynamic queries, since $C^k$-types on them cannot be further refined, as long as the number of variables which may be used remains constant for the whole class of databases. So, the next result is rather intuitive.

**Proposition 27.** *Let $f$ be a query of schema $\sigma$ in $\mathcal{QCQ}^C$, with parameter $n$ according to Definition 26. Let $\mathcal{C}$ be a class of databases of schema $\sigma$ which is strongly $C^k$-homogeneous for some $k \geq 1$. Then, the restriction of $f$ to $\mathcal{C}$ is computable by an $RCM^h$ machine where $h = max\{n, k\}$.*

*Proof.* Let $f$ and $\mathcal{C}$ be as in the statement of the proposition, and let $f'$ be the restriction of $f$ to $\mathcal{C}$. Let $I, J \in \mathcal{C}$, such that $I$ is is strongly $C^{k_1}$-homogeneous and $J$ is strongly $C^{k_2}$-homogeneous. Without loss of generality, let $k_1 \leq k_2 \leq k$. We must prove that $f'$ can be computed on $I$ and $J$ by an $RCM^n$ machine, if $k \leq n$, or by an $RCM^k$ machine, otherwise. If $k \leq n$, by Definition 26, if $Tp^{C^n}(I,n) = Tp^{C^n}(J,n)$, then $Tp^{C^n}(\langle I, f'(I)\rangle, n) = Tp^{C^n}(\langle J, f'(J)\rangle, n)\}$. So, by Definition 7 and Theorem 13, $f'$ can be computed on $I$ and $J$ by a $RCM^n$ machine. As for every pair of databases in $\mathcal{C}$, the minimum $k'$ such that both databases are strongly $C^{k'}$-homogeneous is bounded by $k$, then $f'$ preserves realization of $C^n$-types for $n$-tuples in $\mathcal{C}$. So, $f'$ is computable by a $RCM^n$ machine.

If, on the other hand, $k > n$, we have two different cases. Either $k_2 \leq n < k$ or $n \leq k_2 \leq k$. In the first case $f'$ preserves realization of $C^n$-types for $n$-tuples in $I$ and $J$, and by Corollary 16 $f'$ also preserves realization of $C^k$-types for $k$-tuples in those databases. In the second case $f'$ preserves realization of $C^{k_2}$-types for $k_2$-tuples in $I$ and $J$, and by the same corollary $f'$ also preserves realization of $C^k$-types for $k$-tuples in those databases. Then, by the same argument as before, in both cases $f'$ preserves realization of $C^k$-types for $k$-tuples for any pair of databases in $\mathcal{C}$. So, $f'$ is computable by a $RCM^k$ machine. □

## 5 Further Considerations

### 5.1 Infinitary Logics with Counting

The *infinitary logic* $\mathcal{L}^k_{\infty\omega}$, for every $k \geq 1$, has the usual first order rules for the formation of formulas. In addition, it is closed under infinitary conjunctions and disjunctions. Formulas in $\mathcal{L}^k_{\infty\omega}$ contain at most $k$ different variables. We write $\mathcal{L}^\omega_{\infty\omega} = \bigcup_k \mathcal{L}^k_{\infty\omega}$. Similarly, we define *infinitary logics with counting*, denoted as $C^k_{\infty\omega}$ and $C^\omega_{\infty\omega}$, respectively. These logics are defined in the same way as $\mathcal{L}^k_{\infty\omega}$ and $\mathcal{L}^\omega_{\infty\omega}$ with the addition of all *counting quantifiers*. That is, $C^k_{\infty\omega}$ has the same formation rules as $\mathcal{L}^k_{\infty\omega}$ plus the following one: if $\psi$ is a formula in $C^k_{\infty\omega}$, $x$ is a variable and $m \geq 1$, then $\exists^{\geq m} x(\psi)$ is also a formula in $C^k_{\infty\omega}$, provided the number of different variables in $\exists^{\geq m} x(\psi)$ is $\leq k$. Then $C^\omega_{\infty\omega} = \bigcup_{k \geq 1} C^k_{\infty\omega}$.

We define the following recursive fragments of the infinitary logics $C^\omega_{\infty\omega}$ and $\mathcal{L}^\omega_{\infty\omega}$ as in [AVV95]. For every $k \geq 1$, let $C^k_{\infty\omega}|_{rec}$ denote the fragment of the infinitary logic $C^k_{\infty\omega}$ that contains exactly all sentences with a recursive class of models. We also define $\mathcal{L}^k_{\infty\omega}|_{rec}$ in the same way. Now we give a characterization of the expressive power of the computation model $RCM^k$ (and, hence, also of each subclass $\mathcal{QCQ}^{C^k}$) in terms of the fragment $C^k_{\infty\omega}|_{rec}$.

**Theorem 28.** *For every $k \geq 1$, the expressive power of the $RCM^k$ machine, restricted to Boolean queries, is exactly $C^k_{\infty\omega}|_{rec}$.*

*Proof.* **$\subseteq$):** Let $\mathcal{M}$ be an $RCM^k$ of schema $\sigma$ for some natural $k$. By Propositions 5 and 6, every database $I$ is characterized up to $\equiv_{C^k_{\infty\omega}}$ by a $C^k$ formula. Let $\alpha_I$ be such a formula. On the other hand, by Lemma 12, the computation of $\mathcal{M}$ on a database $I$ is equivalent to a $C^k$ formula $\varphi_{\mathcal{M},I}$. Then we build the $C^k_{\infty\omega}$ formula $\Psi_{\mathcal{M}} \equiv \bigvee_{I \in \mathcal{B}_\sigma} (\alpha_I \wedge \varphi_{\mathcal{M},I})$. Clearly, the models of the formula $\Psi_{\mathcal{M}}$ are the databases accepted by $\mathcal{M}$. To see that the formula is in the fragment $C^k_{\infty\omega}|_{rec}$, note that we can build a Turing machine $M$ which simulates $\mathcal{M}$ and which accepts a database iff it is accepted by $\mathcal{M}$.

**$\supseteq$):** Let $\Psi$ be a $C^k_{\infty\omega}$ formula of schema $\sigma$ such that its class of models is recursive, i.e., it is decidable by some Turing machine $M$. We build an $RCM^k$

machine $\mathcal{M}$ which works as follows. On input $I$ it builds on the $TM$ tape an encoding of every possible database $I'$ of schema $\sigma$ building the corresponding isolating formulas for the $C^k$-types realized in $I'$, as in the proof of Theorem 13, until $I \equiv_{C^k} I'$. Since by Proposition 6 in finite databases $C^k$ equivalence coincides with $C^k_{\infty\omega}$ equivalence, we have $I \models \Psi$ iff $I' \models \Psi$. Hence, $\mathcal{M}$ simulates the $TM$ $M$ in its tape on input $I'$ and $\mathcal{M}$ accepts $I$ iff $M$ accepts $I'$. $\square$

We can use the same strategy to characterize the expressive power of the model $RRM^k$ (and, hence also of each subclass $\mathcal{QCQ}^k$ of ([Tur01a], [Tur04]) in terms of the corresponding fragment $\mathcal{L}^k_{\infty\omega}|_{rec}$. For the proof, we use Corollary 2.3 of [Ott97] and a result from ([APV98], Proof of Theorem 5.6) which is analogous to our Lemma 12 for $RRM^k$ machines.

**Theorem 29.** *For every $k \geq 1$, the expressive power of the $RRM^k$ machine, restricted to Boolean queries, is exactly $\mathcal{L}^k_{\infty\omega}|_{rec}$.* $\square$

Now we will get a similar characterization for the classes of machines which might not halt on all input databases, i.e., for those machines which compute *partial* queries. For the rest of the present sub-section we will consider partial computable queries, that is, queries which are not necessarily defined on all the databases of a given schema. It is noteworthy that this is the way in which computable queries were originally defined (see [CH80] and [AHV94]), though usually only total queries are considered in the literature.

Following the definition in [AVV95] for the logic $\mathcal{L}^\omega_{\infty\omega}$, for every $k \geq 1$, let $C^k_{\infty\omega}|_{r.e.}$ denote the fragment of the infinitary logic $C^k_{\infty\omega}$ that contains exactly all sentences whose classes of models are recursively enumerable. We also define $\mathcal{L}^k_{\infty\omega}|_{r.e.}$ in the same way. Now we give a characterization of the expressive power of the computation models $RCM^k$ when we consider also machines which compute partial queries, in terms of the fragment $C^k_{\infty\omega}|_{r.e.}$.

**Theorem 30.** *For every $k \geq 1$, the expressive power of the $RCM^k$ machine which computes partial queries, restricted to Boolean queries, is exactly $C^k_{\infty\omega}|_{r.e.}$.*

*Proof.* **$\subseteq$):** We follow a similar strategy to the one used in the proof of Theorem 28. For the construction of the $C^k_{\infty\omega}$ formula $\Psi_\mathcal{M}$, we only consider the databases $I$ for which the machine $\mathcal{M}$ halts, i.e., the databases on which the query computed by $\mathcal{M}$ is defined. The models of $\Psi_\mathcal{M}$ are clearly the databases for which the query computed by $\mathcal{M}$ evaluates to true. To prove that the class of models of $\Psi_\mathcal{M}$ is r.e., we construct a Turing machine $M$ which builds on its work tape a sequence (which of course is countably infinite) of all the databases of the schema of the query of every possible size, ordered by their size. At the same time $M$ simulates the operation of the $RCM^k$ $\mathcal{M}$ on all the different databases built on the work tape, executing one step at a time on each of them, in such a way that after $M$ executes the first step of the computation of $\mathcal{M}$ on a given database in the sequence, say $I_i$, it then executes one more step of the computation of $\mathcal{M}$ on each one of the previous databases in the sequence, i.e., on $I_1, I_2, \ldots, I_{i-1}$. Whenever the simulation of $\mathcal{M}$ on a given database halts, if it halts in an accepting state, then $M$ outputs that database in the sequence which it builds on the output tape of the machine. Then, $M$ stops the simulation of $\mathcal{M}$ on that particular database. In this way we get a recursive enumeration of the models of the sentence $\Psi_{\mathcal{M}}$ in the output tape of $M$.

**$\supseteq$):** In this case we also follow a similar strategy as in the corresponding case in the proof of Theorem 28. Once $\mathcal{M}$ built on its $TM$ tape an encoding of a database $I'$ such that $I \equiv_{C^k} I'$, the machine $\mathcal{M}$ uses the $TM$ $M$ of the first part of the proof of the present theorem, to enumerate the models of $\Psi_{\mathcal{M}}$. Then, $\mathcal{M}$ accepts the input database if the database $I'$ is generated in the enumeration of $M$. □

Using an analogous strategy we get a characterization of the expressive power of the computation models $RRM^k$ when we consider also machines which compute partial queries, in terms of the fragment $\mathcal{L}^k_{\infty\omega}|_{r.e.}$.

**Theorem 31.** *For every $k \geq 1$, the expressive power of the $RRM^k$ machines which compute partial queries, restricted to Boolean queries, is exactly $\mathcal{L}^k_{\infty\omega}|_{r.e.}$.* □

## 5.2 The Relevance of Counting in the Hierarchy

Note, that the hierarchy defined by the sub-classes $\mathcal{QCQ}^{C^k}$ by using $k$ as a parameter, has some similar properties to the hierarchy defined by $\mathcal{QCQ}^k$ classes of ([Tur01a], [Tur04]). Both are strict and properly contained in $\mathcal{CQ}$, both are orthogonal to the Turing machine complexity hierarchy, and are characterized syntactically by machines which have a very similar behaviour. However, the expressive power of every logic $C^k$ is much bigger than the expressive power of the corresponding logic $FO^k$ ([Ott97]). It turns out that the sub-classes $\mathcal{QCQ}^k$ are "*very small*", while the sub-classes $\mathcal{QCQ}^{C^k}$ are "*very big*", in a certain precise sense which we define below. This fact can be clearly noted by using the notion of asymptotic probability (see Section 2). Recall from ([Tur01a], [Tur04]) that each sub-class $\mathcal{QCQ}^k$, as well as the whole hierarchy $\mathcal{QCQ}^\omega$, have a 0–1 Law. This implies a strong limitation with respect to expressive power. It is well known that a query as simple as the *parity* query (i.e., $q(I) = true$ iff $|dom(I)|$ is even) has not a 0–1 Law, and this means that this query does not even belong to the whole hierarchy $\mathcal{QCQ}^\omega$. On the other hand, the parity query belongs to the first layer of the hierarchy $\mathcal{QCQ}^{C^\omega}$. Recall the second part of the proof of Theorem 13. There, we defined a machine $RCM^k$ which in first place computed the size of the input database. This was done using a dynamic query in $C^1$, so that the parity query can be computed by an $RCM^1$ machine. Hence, it belongs to the sub-class $\mathcal{QCQ}^{C^1}$. Then the following result follows immediately.

**Proposition 32.** *For any $k \geq 1$, $\mathcal{QCQ}^{C^k}$ does not have a 0–1 Law. Hence, the whole hierarchy $\mathcal{QCQ}^{C^\omega}$ does not have a 0–1 Law either.* □

Propositions 33 and 35 below (see [Gro98a]) will help us to understand the difference in the expressiveness of the corresponding sub-classes, as well as of the hierarchies. Proposition 33 is obtained as a corollary to the combination of a result by L. Babai, P. Erdős, and S. Selkow [BES80], and a result by N. Immerman and

E. Lander [IL90]. Proposition 35 can be also found in [Gro98a], and follows from results of P. Kolaitis and M. Vardi [KV92].

**Proposition 33.** *([BES80] and [IL90]) There is a class $\mathcal{C}$ of graphs with $\mu_{\mathcal{C}} = 1$ such that for all graphs $I, J \in \mathcal{C}$ we have $I \simeq J \Longleftrightarrow I \equiv_{C^2} J$. Moreover, for all $I \in \mathcal{C}$ and $a, b \in dom(I)$, there is an automorphism mapping $a$ to $b$ iff $tp_I^{C^2}(a) = tp_I^{C^2}(b)$.* □

Note that the class $\mathcal{C}$ of Proposition 33 is $C^2$-homogeneous, moreover, it is also strongly $C^2$-homogeneous. So, the following corollary is immediate.

**Corollary 34.**

*(i)* Almost all graphs are strongly $C^2$-homogeneous.

*(ii)* Almost all graphs which are $C^2$-homogeneous are also strongly $C^2$-homogeneous. □

Further note that this result is quite relevant not only in our context, but also in complexity theory, since the isomorphism problem is well known to be in $NP$, whereas M. Grohe has proved that for every $k \geq 1$, $C^k$ equivalence is $PTIME$ complete under quantifier free reductions ([Gro96]). Examples of classes of well known graphs, though not having asymptotic probability 1, where $C^k$ equivalence coincides with isomorphism are the class of planar graphs ([Gro98b]) and the class of trees ([IL90]).

On the other hand, the class of linear graphs is an example of a class where $FO^2$ equivalence coincides with isomorphism (see [EF99]).

**Proposition 35.** *([KV92]) Let $k \geq 1$. If $\mathcal{C}$ is a class of graphs such that for all graphs $I, J \in \mathcal{C}$ we have $I \simeq J \Longleftrightarrow I \equiv_{FO^k} J$, then $\mu_{\mathcal{C}} = 0$.* □

Following [HKL96], though using a slightly different perspective, we define the notion of equality of queries *almost everywhere*. Let $\sigma$ be a schema, and let $q$,$q'$ be two computable queries of schema $\sigma$. Let $\mu_{(q=q')}$ be as follows:

$$\mu_{(q=q')} = \lim_{n\to\infty} \frac{|\{I \in \mathcal{B}_\sigma : dom(I) = \{1, \ldots, n\} \wedge q(I) = q'(I)\}|}{|\{I \in \mathcal{B}_\sigma : dom(I) = \{1, \ldots, n\}\}|}$$

By Proposition 33 for *every* computable query $q$ there is a query $q'$ in $\mathcal{QCQ}^{C^2}$ (and, hence in each layer $\mathcal{QCQ}^{C^k}$, for $k \geq 2$) such that $\mu_{(q=q')} = 1$, i.e., such that $q'$ coincides with $q$ over *almost all* databases. On the other hand, by Proposition 35, this cannot be true for any layer in $\mathcal{QCQ}^\omega$, and not even for the whole hierarchy.

On the other hand, regarding expressive power, the following result shows that the number of variables allowed in formulas is more relevant than allowing the formulas to be infinite and allowing the use of counting quantifiers. Moreover, as it is well known (see [Gro98a]), if we consider only *ordered* databases, then $\mathcal{L}^\omega_{\infty\omega} = C^\omega_{\infty\omega}$, and hence, $\mathcal{QCQ}^\omega = \mathcal{QCQ}^{C^\omega}$.

**Proposition 36.** *([Ott97]) For every $k \geq 1$, there is a Boolean query which is expressible in $FO^{k+1}$, but which is not expressible in $C^k_{\infty\omega}$.* $\square$

As every query which is expressible in $FO$ is clearly recursive, by using Theorem 13 and Theorem 28 we get the following corollary:

**Corollary 37.** *For every $k \geq 1$, $\mathcal{QCQ}^{k+1} \not\subseteq \mathcal{QCQ}^{C^k}$.* $\square$

On the other hand, if we fix the number of variables, the strict inclusion is straightforward.

**Proposition 38.** *For every $k \geq 1$, $\mathcal{QCQ}^k \subset \mathcal{QCQ}^{C^k}$.*

*Proof.* The inclusion follows from Theorems 28 and 29. As to the strict inclusion, note that for every $k \geq 1$, an $RCM^k$ machine can compute the $C^k$-type index for $k$-tuples of its input (see Remark 22), while clearly no $RRM^k$ machine can do it for arbitrary databases. $\square$

Finally, we give some examples of known classifications of queries in the infinitary logics $C^\omega_{\infty\omega}$ and $\mathcal{L}^\omega_{\infty\omega}$ and, hence, by Theorems 28 and 29, in the hierarchies $\mathcal{QCQ}^{C^\omega}$ and $\mathcal{QCQ}^\omega$, respectively.

**Example 39.**

**1):** *The size of a database is even* $\in \mathcal{QCQ}^{C^1}$ and $\notin \mathcal{QCQ}^\omega$ (see observation before Proposition 32).

**2):** *A graph is regular* $\in \mathcal{QCQ}^{C^2}$ and $\notin \mathcal{QCQ}^\omega$ ([Ott97]).

**3):** *A graph is Eulerian* $\in \mathcal{QCQ}^{C^2}$ and $\notin \mathcal{QCQ}^\omega$ ([Ott97]).

**4):** *A graph is a disjoint union of an even number of cliques* $\in \mathcal{QCQ}^{C^2}$ and $\notin \mathcal{QCQ}^\omega$ ([KV95]).

**5):** *A graph is connected* $\in \mathcal{QCQ}^3$ and $\notin \mathcal{QCQ}^{C^2}$ ([Gro98a]).

**6):** *A graph has an even number of connected components* $\in \mathcal{QCQ}^{C^\omega}$ and $\notin \mathcal{QCQ}^\omega$ ([KV95]).

**7):** *A projective plane is Desargian* $\notin \mathcal{QCQ}^{C^3}$ ([Gro98a], see there also the definition of projective planes).

As we pointed out in Remark 17 (see also [Tur01a], [Tur04]), the hierarchies $\mathcal{QCQ}^\omega$ and $\mathcal{QCQ}^{C^\omega}$ are *orthogonal* to the hierarchy of complexity classes defined in terms of TIME and SPACE in Turing machine complexity. Then, we can use these hierarchies to refine the TIME and SPACE complexity classes by intersecting these classes with the different hierarchies $\mathcal{QCQ}^\omega$ and $\mathcal{QCQ}^{C^\omega}$. In this way, we obtain complexity classes which are much finer. This may result in a deeper and more subtle understanding on the nature of queries. Next, we give some examples to illustrate this point.

Combining results from [KL99] and [KL00] with known results in descriptive complexity and with Theorem 29, we get that the following Boolean queries, defined in the class of finite groups, belong to the sub-class $(\mathcal{QCQ}^4 \cap NLOGSPACE)$: *1): Given a group G, it is completely reducible and centreless; 2): Given a group G and a pair of elements a, b, the subgroup generated by a is a subgroup of the subgroup generated by b; 3): Given a group G and a pair of elements a, b, the two elements generate the same subgroup.*

As to the hierarchy $\mathcal{QCQ}^{C^\omega}$, combining known results in descriptive complexity (see [Ott97]) with results from computational complexity and with Theorem 28, we have that the following two Boolean queries belong to the sub-class $(\mathcal{QCQ}^{C^2} \cap PTIME)$: *1): A graph is Eulerian; 2) A graph is regular.*

## Acknowledgements

I am deeply grateful to Lauri Hella for the interesting and stimulating discussions we had on this subject. The comments of an anonymous referee were very helpful in improving both the contents and the presentation of this article.

## References

[AHV94] Abiteboul, S., Hull, R. and Vianu, V., *Foundations of Databases*, Addison-Wesley, 1994.

[APV98] Abiteboul, S., Papadimitriou, C. and Vianu, V., "Reflective Relational Machines", *Information and Computation* 143, pp. 110-136, 1998.

[AV95] Abiteboul, S., Vianu, V., "Computing with first-order logic", *Journal of Computer and System Sciences* 50(2), pp. 309-335, 1995.

[AVV95] Abiteboul, S., Vardi, M. and Vianu, V., "Computing with Infinitary Logic", *Theoretical Computer Science* 149(1), pp. 101-128, 1995.

[AVV97] Abiteboul, S., Vardi, M. and Vianu, V., "Fixpoint Logics, Relational Machines, and Computational Complexity", *Journal of ACM* 44(1), pp. 30-56, 1997.

[BES80] Babai, L., Erdös, P. and Selkow, S., "Random Graph Isomorphism". *SIAM Journal on Computing* 9, pp. 628-635, 1980.

[CFI92] Cai, J. Y., Fürer, M. and Immerman, N., "An Optimal Lower Bound on the Number of Variables for Graph Identification". *Combinatorica* 12(4), pp. 389-410, 1992.

[CH80] Chandra, A. K., Harel, D., "Computable Queries for Relational Data Bases", *Journal of Computer and System Sciences* 21(2), pp. 156-178, 1980.

[CK92] Chang, C. and Keisler, H., *Model Theory*, 3rd ed., Elsevier North Holland, 1992.

[DLW95] Dawar, A., Lindell, S., Weinstein, S., "Infinitary Logic and Inductive Definability over Finite Structures", *Information and Computation* 119(2), pp. 160-175, 1995.

[Ebb85] Ebbinghaus, H., "Extended Logics: The General Framework", in *Model Theoretic Logics*, ed. Barwise and Fefferman, Springer-Verlag, pp. 25-76, 1985.

[EF99] Ebbinghaus, H., Flum, J., *Finite Model Theory*, Springer, 2nd. ed., 1999.

[GM98] Grohe, M. and Mariño, J., "Definability and Descriptive Complexity on Databases of Bounded Tree-Width", in Proceedings of International Conference on Database Theory, ICDT 1999, Springer, LNCS 1540, pp. 70-82, 1998.

[Gro96] Grohe, M., "Equivalence in Finite Variable Logics is Complete for Polynomial Time", in Proceedings of 37th IEEE Symposium on Foundations of Computer Science, pp. 264-273, 1996.

[Gro98a] Grohe, M., "Finite Variable Logics in Descriptive Complexity Theory", Preliminary version, 1998.

[Gro98b] Grohe, M., "Fixed Point Logics on Planar Graphs", in Proceedings of 13th IEEE Symposium on Logic in Computer Science, LICS 1998, pp. 6-15, 1998.

[Hel96] Hella, L., "Logical Hierarchies in PTIME", *Information and Computation* 129(1), pp. 1-19, 1996.

[HKL96] Hella, L., Kolaitis, P. and Luosto, K., "Almost Everywhere Equivalence of Logics in Finite Model Theory", *The Bulletin of Symbolic Logic* 2(4), pp. 422-443, 1996.

[IL90] Immerman, N. and Lander, E., "Describing Graphs: A First Order Approach to Graph Canonization", in *Complexity Theory Retrospective*, ed. A. Selman, Springer, pp. 59-81, 1990.

[Imm99] Immerman, N., *Descriptive Complexity*, Springer, 1999.

[KL99] Koponen, A. and Luosto, K., "Definability of Group Theoretic notions", *Research Report of the Department of Mathematics of the University of Helsinki* 227, 1999.

[KL00] Koponen, A. and Luosto, K., personal comunication, 2000.

[KV92] Kolaitis, P. and Vardi, M., "Infinitary Logic and 0–1 Laws", *Information and Computation* 98, pp. 258-294, 1992.

[KV95] Kolaitis, P. and Väänänen, J., "Generalized Quantifiers and Pebble Games on Finite Structures", *Annals of Pure and Applied Logic* 74, pp. 23-75, 1995.

[Ott96] Otto, M., "The Expressive Power of Fixed Point Logic with Counting", *Journal of Symbolic Logic* 61 (1), pp. 147-176, 1996.

[Ott97] Otto, M., *Bounded Variable Logics and Counting*, Springer, 1997.

[Tur96] 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.

[Tur98] Turull Torres, J.M., "Query Completeness, Distinguishability and Relational Machines", in *Models, Algebras and Proofs*: Selected Papers from the X Latin American Symposium on Mathematical Logic, Bogotá 1995, ed. Marcel-Dekker, pp. 135-163, 1998.

[Tur01a] Turull Torres, J. M., "A Study of Homogeneity in Relational Databases", *Annals of Mathematics and Artificial Intelligence* 33(2), pp. 379-414, 2001.

[Tur01b] Turull Torres, J. M., "Semantic Classifications of Queries to Relational Databases", in *Semantics in Databases*, L. Bertossi, G. Katona, K.-D. Schewe, B. Thalheim (Eds.), Springer LNCS.

[Tur02] Turull Torres, J. M., "Relational Databases and Homogeneity in Logics with Counting", in Foundations of Information and Knowledge Systems: Second International Symposium, FoIKS 2002, Proceedings, Germany, Springer LNCS 2284, 2002.

[Tur04] Turull Torres, J. M., "Erratum for: A Study of Homogeneity in Relational Databases [Annals of Mathematics and Artificial Intelligence 33(2) (2001) 379-414]", *Annals of Mathematics and Artificial Intelligence* 42(4), pp. 443-444, 2004.

*Received August, 2004*