Journal of Universal Computer Science, vol. 16, no. 20 (2010), 2934-2955
submitted: 7/1/10, accepted: 13/9/10, appeared: 1/11/10 © *J.UCS*

Redundant Relations in Relational Databases: A Model Theoretic Perspective

Flavio Antonio Ferrarotti
(Yahoo! Research Latin America and
Universidad de Santiago de Chile, Chile
flaviof@yahoo-inc.com)

Alejandra Lorena Paoletti
(lorena.paoletti@gmail.com)

José María Turull Torres
(School of Engineering and Advanced Technology
College of Sciences, Massey University, New Zealand
J.M.Turull@massey.ac.nz)

Abstract: We initiate in this work the study of a sort of redundancy problem revealed by what we call redundant relations. Roughly, we define a *redundant relation* in a database instance (dbi) as a k-ary relation R such that there is a first-order query which evaluated in the reduced dbi, (i.e., the dbi without the redundant relation R) gives us R. So, given that first-order types are isomorphism types on finite structures, we can eliminate that relation R as long as the equivalence classes of the relation of equality of the first-order types for all k-tuples in the dbi are not altered. It turns out that in a fixed dbi, the problem of deciding whether a given relation in the dbi is redundant is *decidable*, though intractable, as well as the problem of deciding whether there is any relation symbol in the schema which is a redundant relation in the given dbi. We then study redundant relations with a restricted notion of equivalence so that the problem becomes *tractable*.

Key Words: first-order types, isomorphism types, redundancy, relational databases.
Category: H.2, H.2.1, H.2.3

1 Introduction

From a conceptual point of view it is desirable for a model of computation of queries to be *representation independent* [Abiteboul et al. 1994]. This means, roughly, that queries to databases (in the present work we will refer to *database instances* simply as *databases*) which represent the "same" reality should evaluate to the "same" result. In mathematical terms, the previous concept was captured by asking queries to isomorphic databases to evaluate to the same result [Chandra and Harel 1980]. The principle of preservation of isomorphisms has an important consequence if we consider a single database, namely the preservation of automorphisms. That is, considering a fixed database, two elements with the same "structural" properties should be considered as undistinguishable. By structural properties we roughly mean the way in which the two elements are related to all other elements in the database, by means of the different relations according to the schema. The same is also true for tuples of elements, i.e., two tuples with the same "structural" properties should be considered as undistinguishable. To formalize this concept we can make use of the model theoretic notion of type. The notion of type of a tuple is a topic which has been deeply studied in the context of finite model theory [Dawar 1993, Otto 1997], but which has not received the same attention in the context of database theory. Roughly, if $\mathcal{L}$ is a logic, the $\mathcal{L}$ *type of a tuple* of length k in a given database is the set of $\mathcal{L}$ formulas with up to k free variables which are satisfied by that tuple in the database.

As databases are finite structures, it follows that two arbitrary tuples have the same first-order type if and only if they are commutable by some automorphism. So, two arbitrary tuples have the same "structural" properties and should be considered undistinguishable, if and only if, they have the same first-order type.

Designing a relational database schema is usually a complex task which has important practical consequences. Redundant storage of information can lead to a variety of practical problems on the updating, insertion and deletion of data. This anomaly is usually known as the redundancy problem and has been studied extensively in the field of databases. Traditionally, the redundancy problem is studied by considering a particular class of properties, the functional dependencies, that are supposed to be satisfied by all instances of a given database. By taking a quite different approach, we will make use of the model theoretic concept of type to study the redundancy problem.

Specifically, we initiate in this work the study of a sort of redundancy problem revealed by what we call redundant relations. Roughly, we define a *redundant relation* as a relation R such that there is a first-order query which evaluated in the *reduced database*, (i.e., the database without the redundant relation R), gives us R. So, given that first-order types are isomorphism types on finite structures, we can eliminate that relation R as long as the equivalence classes of the relation of equality of the first-order types for all k-tuples in the database are not altered. In practical terms, this means that we do not lose information if we eliminate such redundant relation from a database. It turns out that in a fixed database of some relational schema, the problem of deciding whether a given relation in the database is redundant is *decidable*, though intractable, as well as the problem of deciding whether there is any relation symbol in the schema which is a redundant relation in the given database. We then study redundant relations with a restricted notion of equivalence so that the problem becomes *tractable*.

We also give the construction of a formula in polynomial time which, provided that R is a redundant relation in the database, will evaluate to R in the reduced database.

Though in this work we do not consider classes of databases, it certainly makes sense to think on relations which are redundant not only in a particular database but in a whole class of databases of a given schema. Note that the problem of deciding whether a given relation (schema) is redundant in a given class of databases is clearly not decidable in the general case.

We organized the article as follows. In Section 2 we give a brief description of the concepts and results of finite model theory and databases, as well as the notations that we use in this work. In Section 3 we formally introduce the concept of redundant relation in databases and discuss in detail its main implications and consequences. Finally, in Section 4 we establish our main decidability result regarding redundant relations, and we study redundant relations with a restricted notion of equivalence so that the problem becomes tractable.

The outcome of this research can be of a great relevance to applications like census databases, where we have a huge and stable database instance of a very large schema, and where by eliminating redundant relations we can save an important amount of space and time in the evaluation of queries. We aim to follow this research towards defining a kind of normal form for database instances and further for restricted classes of databases where the problem of checking for redundant relations can be tractable.

Note that, this paper is an extended version of [Ferrarotti, et al. 2009] which was presented at the ETheCoM 2009 workshop held on November 2009 in Gramado, Brazil. Furthermore, several results presented here first appeared in [Paoletti 2005].

2 Preliminaries

We define a *relational database schema*, or simply *schema*, as a set of relation symbols with associated arities, unless otherwise explicitly stated. 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 σ, is a structure $\mathcal{I} = \langle D^{\mathcal{I}}, R_1^{\mathcal{I}}, \ldots, R_s^{\mathcal{I}} \rangle$ where $D^{\mathcal{I}}$ is a finite set which contains exactly all elements of the database, and for $1 \leq i \leq s$, $R_i^{\mathcal{I}}$ is a relation of arity r_i, i.e., $R_i^{\mathcal{I}} \subseteq (D^{\mathcal{I}})^{r_i}$. We often use $\mathrm{dom}(\mathcal{I})$ instead of $D^{\mathcal{I}}$. We use $\simeq$ to denote isomorphism. A *k-tuple* over a database $\mathcal{I}$, with $k \geq 1$, is a tuple of length k formed with elements from $\mathrm{dom}(\mathcal{I})$. We denote a k-tuple of $\mathcal{I}$ as $\bar{a}_k$, and also as $\bar{a}$. We use $\mathcal{DB}_\sigma$ to denote the class of all databases of schema σ.

Computable Queries: In this paper, we will consider *total* queries only. Let σ 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 σ* ([Chandra and Harel 1980]), is a total recursive function $q^r : \mathcal{DB}_\sigma \rightarrow \mathcal{DB}_{\langle R \rangle}$ which preserves isomorphisms such that for every database $\mathcal{I}$ of schema σ, $\mathrm{dom}(q(\mathcal{I})) \subseteq \mathrm{dom}(\mathcal{I})$. A Boolean query is a 0-ary query. We denote the class of computable queries of schema σ as $\mathcal{CQ}_\sigma$, and $\mathcal{CQ} = \bigcup_\sigma \mathcal{CQ}_\sigma$.

Finite Model Theory and Databases: We use the notion of a *logic* in a general sense. A formal definition would only complicate the presentation and is unnecessary for our work. As usual in finite model theory, we regard a logic as a language, that is, as a set of formulas (see [Ebbinghaus and Flum 1999, Abiteboul et al. 1994]). Unless otherwise explicitly stated, we only consider signatures, or vocabularies, which are purely *relational*. We 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 *equivalence* between structures or databases, denoted as $\equiv_\mathcal{L}$, is related to only finite structures. If $\mathcal{L}$ is a logic and σ is a signature, we denote as $\mathcal{L}_\sigma$ the class of formulas from $\mathcal{L}$ with signature σ. If $\mathcal{I}$ is a structure of signature σ, or σ*-structure*, we define the $\mathcal{L}$ *theory of* $\mathcal{I}$ as $Th_\mathcal{L}(\mathcal{I}) = \{\varphi \in \mathcal{L}_\sigma : \mathcal{I} \models_\mathcal{L} \varphi\}$. A *database schema* is regarded as a *relational signature*, and a *database instance* of some schema σ as a finite and relational σ*-structure*. By $\varphi(x_1, \ldots, x_r)$ we denote a formula of some logic whose free variables are *exactly* $\{x_1, \ldots, x_r\}$. We denote the set of free variables of a formula φ as $\mathrm{free}(\varphi)$. If $\varphi(x_1, \ldots, x_k) \in \mathcal{L}_\sigma$, $\mathcal{I} \in \mathcal{DB}_\sigma$, $\bar{a}_k = (a_1, \ldots, a_k)$ is a k-tuple over $\mathcal{I}$, let $\mathcal{I} \models \varphi(x_1, \ldots, x_k)[a_1, \ldots, a_k]$ denote that φ is TRUE, when interpreted by $\mathcal{I}$, 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: $\varphi^\mathcal{I} = \{(a_1, \ldots, a_k) : a_1, \ldots, a_k \in \mathrm{dom}(\mathcal{I}) \wedge \mathcal{I} \models \varphi(x_1, \ldots, x_k)[a_1, \ldots, a_k]\}$. That is, $\varphi^\mathcal{I}$ is the relation defined by φ in the structure $\mathcal{I}$, and its arity is given by the number of free variables in φ. Sometimes, we use the same notation when the set of free variables of the formula is *strictly* included in $\{x_1, \ldots, x_k\}$. We denote as FO^k with some integer $k \geq 1$ the fragment of first-order logic (FO) where only formulas whose variables 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, a sentence of the form $\exists^{\geq m} x(\varphi)$ means that there are at least m *different* elements in the database which satisfy φ.

Types: Given a database $\mathcal{I}$ and a k-tuple $\bar{a}_k$ in $\mathrm{dom}(\mathcal{I})^k$, we would like to consider *all* properties of $\bar{a}_k$ in the database $\mathcal{I}$ 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 $\mathcal{I}$ be a database of some schema σ and let $\bar{a}_k = (a_1, \ldots, a_k)$ be a k-tuple over $\mathcal{I}$. The $\mathcal{L}$ *type* of $\bar{a}_k$ in $\mathcal{I}$, denoted as $tp^\mathcal{L}_\mathcal{I}(\bar{a}_k)$, 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 $\mathcal{I}$ 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 $tp^{\mathcal{L}}_{\mathcal{I}}(\bar{a}_k) = \{\varphi \in \mathcal{L}_\sigma : \text{free}(\varphi) \subseteq \{x_1, \ldots, x_k\} \wedge \mathcal{I} \models \varphi[a_1, \ldots, a_k]\}$. We say that a relation $R \subseteq \text{dom}(\mathcal{I})^r$ of arity $r \geq 1$ has *complete $\mathcal{L}$-types* in $\mathcal{I}$ iff, for every pair of r-tuples $\bar{a}$ and $\bar{b}$ in $\text{dom}(\mathcal{I})^r$, if $\bar{a} \in R$ and $tp^{\mathcal{L}}_{\mathcal{I}}(\bar{a}) = tp^{\mathcal{L}}_{\mathcal{I}}(\bar{b})$, then $\bar{b} \in R$. Note that 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 appropriate free variables) as long as the resulting set remains consistent. Let α be the $\mathcal{L}$ type of some k-tuple over some database in $\mathcal{B}_\sigma$. We say that a database $\mathcal{I}$ *realizes* the type α if there is a k-tuple $\bar{a}_k$ over $\mathcal{I}$ whose $\mathcal{L}$ type is α. That is, if $tp^{\mathcal{L}}_{\mathcal{I}}(\bar{a}_k) = \alpha$. The following is a well known result.

Proposition 1. *For every schema σ and for every pair of (finite) databases $\mathcal{I}$, $\mathcal{J}$ of schema σ the following holds: $\mathcal{I} \equiv_{FO} \mathcal{J}$ iff $\mathcal{I} \simeq \mathcal{J}$.*

Although types are infinite sets of formulas, due to results in [Dawar 1993] and [Otto 1996], a *single* FO^k (C^k) formula is equivalent to the FO^k (C^k) type of a tuple over a given database. The equivalence holds for all databases of the same schema.

Proposition 2. *([Dawar 1993, Otto 1996]): For every schema σ, for every database $\mathcal{I}$ of schema σ, for every $k \geq 1$, for every $1 \leq l \leq k$, and for every l-tuple $\bar{a}_l$ over $\mathcal{I}$, there is an FO^k formula $\chi \in tp^{FO^k}_{\mathcal{I}}(\bar{a}_l)$ and a C^k formula $\phi \in tp^{C^k}_{\mathcal{I}}(\bar{a}_l)$, such that for any database $\mathcal{J}$ of schema σ and for every l-tuple $\bar{b}_l$ over $\mathcal{J}$, $\mathcal{J} \models \chi[\bar{b}_l]$ iff $tp^{FO^k}_{\mathcal{I}}(\bar{a}_l) = tp^{FO^k}_{\mathcal{J}}(\bar{b}_l)$ and $\mathcal{J} \models \phi[\bar{b}_l]$ iff $tp^{C^k}_{\mathcal{I}}(\bar{a}_l) = tp^{C^k}_{\mathcal{J}}(\bar{b}_l)$.*

Moreover, such formulas χ and ϕ can be built inductively for a given database. If an FO^k formula χ (C^k formula ϕ, respectively) satisfies the condition of Proposition 2, we call χ an *isolating formula* for $tp^{FO^k}_{\mathcal{I}}(\bar{a}_l)$ (ϕ an *isolating formula* for $tp^{C^k}_{\mathcal{I}}(\bar{a}_l)$, respectively).

Remark. Isolating formulas for the FO types of k-tuples can be built in a similar way to that used to build the isolating formulas for FO^k types and C^k types. Considering the formulas $\varphi^m_{\bar{u}}(\bar{x})$, defined in [Ebbinghaus and Flum 1999] in Theorem 2.2.8, as we are dealing with finite structures there is always an m big enough such that for all σ-structures $\mathcal{B}$ and k-tuples $\bar{v}$ over $\text{dom}(\mathcal{B})^k$ we have that $\mathcal{B} \models \varphi^m_{\mathcal{A},\bar{u}}[\bar{v}]$ iff $tp^{FO}_{\mathcal{A}}(\bar{u}) = tp^{FO}_{\mathcal{B}}(\bar{v})$, and that is the *isolating formula* for the FO type of $\bar{u}$ in $\mathcal{A}$. It is well known (see [Ebbinghaus and Flum 1999]) that $n + 1$ is a value of m big enough to build the isolating formula for an arbitrary k-tuple in a given database of size n. The size of these formulas is exponential in n. However, for FO types there are other isolating formulas, built from the so called *diagram* of the database, which are of size polynomial in n (see Proposition 6 below).

Let $\bar{a}_k = (a_1, \ldots, a_k)$ be a k-tuple over $\mathcal{I}$. We say that the type $tp^{\mathcal{L}}_{\mathcal{I}}(\bar{a}_k)$ is an *automorphism type* in the database $\mathcal{I}$ if for every k-tuple $\bar{b}_k = (b_1, \ldots, b_k)$ over

$\mathcal{I}$, if $tp^{\mathcal{L}}_{\mathcal{I}}(\bar{a}_k) = tp^{\mathcal{L}}_{\mathcal{I}}(\bar{b}_k)$, then there exists an automorphism f in the database $\mathcal{I}$ 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 $\mathcal{I}$, 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 $\mathcal{I}$.

3 Databases with Redundant Relations

It is well known that, depending on its design, a database may contain *redundant information*, i.e., it may contain the same information stored in more than one place within the database. In this section, we use the model theoretic concept of type to detect the presence of a particular kind of redundancy which we call redundant relations.

Definition 3. Let $\mathcal{A}$ be a dbi of some schema σ, let $\bar{a} \in \mathrm{dom}(\mathcal{A})^r$, let $\bar{x} = (x_1, \ldots, x_r)$, and let $m \geq 0$. The m-isomorphism type (or m-Hintikka formula) $\varphi^m_{\mathcal{A},\bar{a}}(\bar{x})$ of $\bar{a}$ in $\mathcal{A}$ is defined as follows:

$$\varphi^m_{\mathcal{A},\bar{a}}(\bar{x}) \equiv \bigwedge\{\varphi(\bar{x}) : \varphi \text{ is atomic or negated atomic, } \mathcal{A} \models \varphi[\bar{a}]\}$$

and for $m > 0$,

$$\varphi^m_{\mathcal{A},\bar{a}}(\bar{x}) \equiv \bigwedge_{a \in \mathrm{dom}(\mathcal{A})} \exists x_{r+1} \varphi^{m-1}_{\mathcal{A},\bar{a}a}(\bar{x}, x_{r+1}) \wedge \forall x_{r+1} \bigvee_{a \in \mathrm{dom}(\mathcal{A})} \varphi^{m-1}_{\mathcal{A},\bar{a}a}(\bar{x}, x_{r+1}).$$

$\varphi^m_{\mathcal{A},\bar{a}}(\bar{x})$ describes the isomorphism type of the substructure generated by $\bar{a}$ in $\mathcal{A}$, and for $m > 0$ the formula $\varphi^m_{\mathcal{A},\bar{a}}(\bar{x})$ describes to which isomorphism types the tuple $\bar{a}$ can be extended in m steps adding one element in each step.

The fundamental observation which leads to our definition of redundant relation is that, as the FO types of all k-tuples in a database $\mathcal{A}$ describe all FO properties which are satisfied by the tuples of arity k in $\mathcal{A}$, every FO query of arity k is equivalent in $\mathcal{A}$ to the disjunction of some of the FO isolating formulas for the FO types for k-tuples in $\mathcal{A}$. This is a consequence of the following well known result.

Proposition 4 (see Theorem 2.2.11 in [Ebbinghaus and Flum 1999]). *Let $\varphi(x_1, \ldots, x_r)$ be an FO formula of quantifier rank $\leq m$. Then,*

$$\varphi \equiv \bigvee\{\varphi^m_{\mathcal{A},\bar{a}} : \mathcal{A} \textit{ is a dbi, } \bar{a} \in \mathrm{dom}(\mathcal{A})^r \textit{ and } \mathcal{A} \models \varphi(x_1, \ldots, x_r)[\bar{a}]\},$$

where, $\varphi^m_{\mathcal{A},\bar{a}}$ is the m-isomorphism type of $\bar{a}$ in $\mathcal{A}$ and the disjunction is taken over a finite set (see Lemma 2.2.6 in [Ebbinghaus and Flum 1999]).

Thus, we could eliminate a relation $R^{\mathcal{A}}$ of arity k from $\mathcal{A}$ as long as the relationship among the FO types of the different k-tuples in $\mathcal{A}$ is not altered.

Definition 5. Let σ be a relational schema, let $\mathcal{A}$ be a database of schema σ, and let R_i be a given relation symbol in σ of some arity $k \geq 1$. We denote as:

– $\sigma - R_i$ the schema obtained by eliminating from σ the relation symbol R_i, i.e., if $\sigma = \langle R_1, \ldots, R_i, \ldots, R_n \rangle$, then $\sigma - R_i = \langle R_1, \ldots, R_{i-1}, R_{i+1}, \ldots, R_n \rangle$;

– FO_σ and $FO_{\sigma - R_i}$ the set of formulas of FO over the schemas σ and $\sigma - R_i$, respectively; and

– $\mathcal{A}|_{\sigma - R_i}$ the *reduced database* of schema $\sigma - R_i$ obtained by eliminating the relation $R_i^{\mathcal{A}}$ from $\mathcal{A}$.

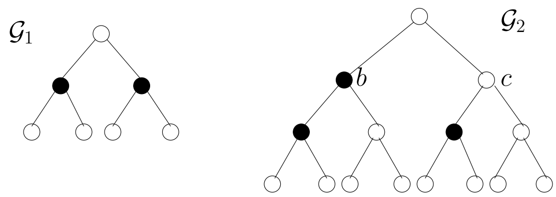
We say that $R_i^{\mathcal{A}}$ is a *redundant relation* in the database $\mathcal{A}$ if for all k-tuples $\bar{u}$ and $\bar{v}$ in $\mathrm{dom}(\mathcal{A})^k$,

$$tp_{\mathcal{A}}^{FO_\sigma}(\bar{u}) = tp_{\mathcal{A}}^{FO_\sigma}(\bar{v}) \text{ iff } tp_{\mathcal{A}|_{\sigma - R_i}}^{FO_{\sigma - R_i}}(\bar{u}) = tp_{\mathcal{A}|_{\sigma - R_i}}^{FO_{\sigma - R_i}}(\bar{v}),$$

i.e., if the equivalence classes induced by the relation of equality of FO_σ types of the k-tuples in $\mathrm{dom}(\mathcal{A})^k$ coincide with the equivalence classes induced by the relation of equality of $FO_{\sigma - R_i}$ types of k-tuples in $\mathrm{dom}(\mathcal{A}|_{\sigma - R_i})^k$.

Let us see a few examples of database instances with a redundant relation.

Example 1. Below, we show two complete binary trees $\mathcal{G}_1$ and $\mathcal{G}_2$. They can be seen as databases of schema $\tau = \langle E, C \rangle$ with E a binary relation symbol interpreted as the edge relation and C a unary relation symbol interpreted as the set of black nodes.



Clearly, if we consider the FO types for tuples of arity 1 in a complete binary tree of depth n then we have $n + 1$ different types, because all nodes of the same depth have the same FO type. That is, a node in a complete binary tree cannot be distinguished by any FO formula from another node at the same depth in the tree, therefore, nodes of the same depth can be exchanged by an automorphism of the tree. This fact points out that in our complete binary tree $\mathcal{G}_1$, the relation $C^{\mathcal{G}_1}$ is a redundant relation, i.e., for every elements $u, v \in \mathrm{dom}(\mathcal{G}_1)$, $tp_{\mathcal{G}_1}^{FO_\tau}(u) = tp_{\mathcal{G}_1}^{FO_\tau}(v)$ iff $tp_{\mathcal{G}_1|_{\tau - C}}^{FO_{\tau - C}}(u) = tp_{\mathcal{G}_1|_{\tau - C}}^{FO_{\tau - C}}(v)$. On the other hand, this is not the case for the tree $\mathcal{G}_2$ as the relation $C^{\mathcal{G}_2}$ allows us to distinguish, for levels two and three, some nodes from the others in the same level. So it is not longer the case that all nodes in the same level have the same FO type. Take for instance the nodes b and c in $\mathcal{G}_2$. Let $\varphi_b(x) \equiv \exists y(E(y,x) \wedge \neg\exists z(E(z,y))) \wedge C(x)$ and let $\varphi_c(x) \equiv \exists y(E(y,x) \wedge \neg\exists z(E(z,y))) \wedge \neg C(x)$. Then, $\mathcal{G}_2 \models \varphi_b(x)[b]$ but $\mathcal{G}_2 \not\models \varphi_b(x)[c]$ and $\mathcal{G}_2 \models \varphi_c(x)[c]$ but $\mathcal{G}_2 \not\models \varphi_c(x)[b]$. Clearly, $tp^{FO_\tau}_{\mathcal{G}_2}(b) \neq tp^{FO_\tau}_{\mathcal{G}_2}(c)$ while $tp^{FO_{\tau-C}}_{\mathcal{G}_2|_{\tau-C}}(b) = tp^{FO_{\tau-C}}_{\mathcal{G}_2|_{\tau-C}}(c)$.

Example 2. Let us consider the classical database example of suppliers, parts and projects (see [Ullman 1988] for instance). Assume a database schema $\sigma = \langle S, P, J, SPJ, SP\rangle$ and a dbi $\mathcal{A}$ of schema σ. Let $S^{\mathcal{A}}$, $P^{\mathcal{A}}$ and $J^{\mathcal{A}}$ be the suppliers, parts and projects relations, respectively. A tuple (s_id, p_id, j_id, c) is in the relation $SPJ^{\mathcal{A}}$ iff supplier s_id supplies c parts p_id to project j_id. The relation $SP^{\mathcal{A}}$ is the projection of the columns s_id and p_id of $SPJ^{\mathcal{A}}$, i.e., $SP^{\mathcal{A}} = \pi_{s\_id,p\_id}(SPJ^{\mathcal{A}})$. Since $\varphi(s\_id, p\_id) \equiv \exists j\_id\, c\,(SPJ(s\_id, p\_id, j\_id, c))$ is an $FO_{\sigma-SP}$ formula such that $\varphi^{\mathcal{A}|\sigma-SP} = SP^{\mathcal{A}}$, it follows that $SP^{\mathcal{A}}$ is a redundant relation in $\mathcal{A}$.

Example 3. Again, let us consider the classical database example of suppliers, parts and projects with the same schema σ of Example 2. All relation symbols are interpreted in the same way as in Example 2, except for the relation symbol SP that is interpreted as the relation resulting from evaluating the relational calculus query: "Supplier Sid supplies part Pid to every project to which Sid supplies some part". Let $\mathcal{A}$ be a dbi of schema σ and $\varphi(Sid, Pid) \equiv \forall p'\, j\, c\,(SPJ(Sid, p', j, c) \rightarrow \exists c'(SPJ(Sid, Pid, j, c'))$. Then $\varphi^{\mathcal{A}|\sigma-SP} = SP^{\mathcal{A}}$, and hence $SP^{\mathcal{A}}$ is a redundant relation in $\mathcal{A}$.

Next, we show that there is, for every redundant relation $R^{\mathcal{A}}$ in a database $\mathcal{A}$ of schema σ, an FO formula ϕ_R of vocabulary $\sigma - R$ such that if ϕ_R is evaluated in the reduced database $\mathcal{A}|_{\sigma-R}$, it defines the relation $R^{\mathcal{A}}$.

Proposition 6. *Let R be a relation symbol of arity r in a schema σ, let $R^{\mathcal{A}}$ be a redundant relation in a database $\mathcal{A}$ of schema σ, let $\bar{a}$ be an r-tuple in $R^{\mathcal{A}}$, and let $\bar{b}$ be an r-tuple in* $\mathrm{dom}(\mathcal{A})$*. Then, there is a formula $\psi_{\bar{a}}(z_1, \ldots, z_r)$ of $FO_{\sigma-R}$ such that $\mathcal{A}|_{\sigma-R} \models \psi_{\bar{a}}(z_1, \ldots, z_r)[\bar{b}]$ iff $tp^{FO_{\sigma-R}}_{\mathcal{A}|_{\sigma-R}}(\bar{a}) = tp^{FO_{\sigma-R}}_{\mathcal{A}|_{\sigma-R}}(\bar{b})$. And, hence, if $\mathcal{A}|_{\sigma-R} \models \psi_{\bar{a}}(z_1, \ldots, z_r)[\bar{b}]$ then $\bar{b} \in R^{\mathcal{A}}$.*

Proof. Following [Ebbinghaus and Flum 1999] we build $\psi_{\bar{a}}(z_1, \ldots, z_r)$ by using the diagram of $\mathcal{A}|_{\sigma-R}$. Assume $|\mathrm{dom}(\mathcal{A}|_{\sigma-R})| = n$. Let $v : \{x_1, \ldots, x_n\} \rightarrow \mathrm{dom}(\mathcal{A}|_{\sigma-R})$ be an injective valuation such that $v(x_{i_1}) = a_1, \ldots, v(x_{i_r}) = a_r$, where $1 \leq i_1, \ldots, i_r \leq n$. Let

$$\Theta = \{\alpha : \alpha \text{ has the form } P(x_{i_1}, \ldots, x_{i_k}) \text{ where } 1 \leq i_1, \ldots, i_k \leq n, \text{ and } P \in \sigma - R \text{ with arity } k \geq 1\}$$

and let

$$\psi_{\bar{a}}(z_1,\dots,z_r) \equiv \exists x_1 \dots x_n \big(\bigwedge\{\alpha : \alpha \in \Theta, (\mathcal{A}|_{\sigma-R}, v) \models \alpha\} \wedge \\ \bigwedge\{\neg\alpha : \alpha \in \Theta, (\mathcal{A}|_{\sigma-R}, v) \models \neg\alpha\} \wedge \bigwedge_{1 \leq i < j \leq n}(x_i \neq x_j) \wedge \\ \forall x_{n+1}(x_{n+1} = x_1 \vee \dots \vee x_{n+1} = x_n) \wedge z_1 = x_{i_1} \wedge \dots \wedge z_r = x_{i_r} \big)$$

The following facts complete the proof. Clearly, a given tuple $\bar{b} = (b_1, \dots, b_r)$ satisfies $\psi_{\bar{a}}(z_1, \dots, z_r)$ iff there exists an automorphism f in $\mathcal{A}|_{\sigma-R}$ which maps $\bar{a}$ onto $\bar{b}$, i.e., for $1 \leq i \leq r$, $f(a_i) = b_i$. That is, the formula $\psi_{\bar{a}}$ is an isolating formula for the FO type of $\bar{a}$ in $\mathcal{A}|_{\sigma-R}$ (see remark following Proposition 2). Furthermore, as $R^{\mathcal{A}}$ is redundant, every tuple $\bar{b}$ whose $FO_{\sigma-R}$ type coincides with the $FO_{\sigma-R}$ type of $\bar{a}$, is also in $R^{\mathcal{A}}$. Note that, since we are dealing with finite databases, FO types are automorphism types. □

Observe that in Example 1, the relation $C^{\mathcal{G}_1}$ is a redundant relation in $\mathcal{G}_1$ as it has complete FO types for the 1-tuples for nodes in the second level on the tree, while the relation $C^{\mathcal{G}_2}$ is not a redundant relation in $\mathcal{G}_2$ as it does not have complete FO types for the 1-tuples either for nodes in the second or in the third level of $\mathcal{G}_2$.

The following proposition shows that, given a redundant relation $R^{\mathcal{A}}$ in a database $\mathcal{A}$ of schema σ, there is an FO formula ϕ_R of vocabulary $\sigma - R$ such that if ϕ_R is evaluated in the reduced database $\mathcal{A}|_{\sigma-R}$, it defines the relation $R^{\mathcal{A}}$, and that such formula can be build in polynomial time.

Proposition 7. *Let $\mathcal{A}$ be a database of schema σ, and let $R^{\mathcal{A}} = \{\bar{a}^1_k, \dots, \bar{a}^n_k\}$ be a redundant relation of arity k and cardinality n in $\mathcal{A}$. Then, the following FO formula $\phi_R(x_1, \dots, x_k) \equiv \psi_1(x_1, \dots, x_k) \vee \dots \vee \psi_n(x_1, \dots, x_k)$ where, for $1 \leq i \leq n$, ψ_i is the formula described in Proposition 6 for the k-tuple $\bar{a}^i_k$, defines the relation $R^{\mathcal{A}}$ when evaluated in the reduced database $\mathcal{A}|_{\sigma-R}$, i.e., $\phi_R^{\mathcal{A}|_{\sigma-R}} = R^{\mathcal{A}}$. Furthermore, there is an algorithm which builds the formula ϕ_R in polynomial time.*

Proof. (sketch). It follows from Proposition 6 and the fact that a relation $R^{\mathcal{A}}$ of arity r is redundant in $\mathcal{A}$ if and only if for every FO type for the r-tuples α realized by the database $\mathcal{A}$, either all r-tuples whose type is α belong to $R^{\mathcal{A}}$ or none of them does. Furthermore, it takes polynomial time to build each subformula ψ_i of ϕ_R, since the task of building the diagram of a database is known to take polynomial time. And that is what we did in Proposition 6. □

Remark. If we omit in the previous proposition the condition of $R^{\mathcal{A}}$ being a redundant relation, then the relation $\phi_R^{\mathcal{A}|_{\sigma-R}}$ would include not only the tuples in $R^{\mathcal{A}}$, but also all the tuples which are commutable by an automorphism with some tuple in $R^{\mathcal{A}}$.

Note that given an FO formula φ_q which expresses an arbitrary query q over a database $\mathcal{A}$ of schema σ, it can be translated in a straightforward way to a formula φ'_q of schema $\sigma - R$ which expresses the same query q over the reduced database $\mathcal{A}|_{\sigma-R}$. By Proposition 7, a redundant relation $R^{\mathcal{A}}$ of arity k in $\mathcal{A}$ can be expressed by an FO formula $\phi_R(x_1, \dots, x_k)$ in $\mathcal{A}|_{\sigma-R}$. Therefore, every arbitrary query q which is expressed by an FO formula φ_q in which the relation symbol R occurs, could be expressed in the reduced database $\mathcal{A}|_{\sigma-R}$ using the formula $\phi_R(x_1, \dots, x_k)$. That is, every atomic formula formed with the relation symbol R in φ_q can be replaced in φ'_q by the formula $\phi_R(x_1, \dots, x_k)$ in the database $\mathcal{A}|_{\sigma-R}$. We only need to take care of the appropriate re-naming of variables in ϕ_R. In general, we can say that given a logic $\mathcal{L}$ and a formula φ_q in that logic that expresses an arbitrary query q over a database $\mathcal{A}$ of schema σ, it can be translated to a formula φ'_q in the same logic of schema $\sigma - R$ which expresses the same query q over the reduced database $\mathcal{A}|_{\sigma-R}$ provided that the formula ϕ_R can be expressed in the logic $\mathcal{L}$.

Up to now, in this article we have used somehow informally three different ways to characterize redundant relations. Next we show that these three characterizations are indeed equivalent. These are direct consequences of well known facts in finite model theory, but we include a direct proof for clarity.

Fact 8 *Let σ be a relational vocabulary, let R be a relation symbol of arity $r \geq 1$ in σ, and let $\mathcal{A}$ be a dbi of schema σ. Then the following are equivalent:*

i. *$R^{\mathcal{A}}$ has complete $FO_{\sigma-R}$ types for r-tuples.*

ii. *$R^{\mathcal{A}}$ is a redundant relation in the dbi $\mathcal{A}$.*

iii. *There is an $FO_{\sigma-R}$ formula φ such that $\varphi^{\mathcal{A}|_{\sigma-R}} = R^{\mathcal{A}}$.*

Proof.

- $(i) \Rightarrow (iii)$: $R^{\mathcal{A}}$ has complete $FO_{\sigma-R}$ types for r-tuples iff for every two r-tuples $\bar{a}, \bar{b} \in \mathrm{dom}(\mathcal{A})^r$ such that $tp^{FO_{\sigma-R}}_{\mathcal{A}|_{\sigma-R}}(\bar{a}) = tp^{FO_{\sigma-R}}_{\mathcal{A}|_{\sigma-R}}(\bar{b})$ and $\bar{a} \in R^{\mathcal{A}}$, it holds that $\bar{b}$ is also in $R^{\mathcal{A}}$. Then, by using the formulas ψ of Propositions 7 and 6, $\varphi \equiv \psi_{\bar{a}^1_r} \vee \ldots \vee \psi_{\bar{a}^m_r}$, where $R^{\mathcal{A}} = \{\bar{a}^1_r, \ldots, \bar{a}^m_r\}$.

- $(iii) \Rightarrow (ii)$: Suppose, running towards a contradiction, that $R^{\mathcal{A}}$ is not redundant. Then, there are two r-tuples $\bar{a}, \bar{b} \in \mathrm{dom}(\mathcal{A})^r$ such that

$$tp^{FO_{\sigma-R}}_{\mathcal{A}|_{\sigma-R}}(\bar{a}) = tp^{FO_{\sigma-R}}_{\mathcal{A}|_{\sigma-R}}(\bar{b}) \qquad (4)$$

  but $tp^{FO_\sigma}_{\mathcal{A}}(\bar{a}) \neq tp^{FO_\sigma}_{\mathcal{A}}(\bar{b})$, so that there is an FO_σ formula ψ with $r' \leq r$ free variables such that

$$\mathcal{A} \models \psi(x_1, \ldots, x_{r'})[\bar{a}] \text{ while } \mathcal{A} \not\models \psi(x_1, \ldots, x_{r'})[\bar{b}] \qquad (5)$$

By (4) $\psi \notin FO_{\sigma-R}$, i.e., ψ contains $R(x_1,...,x_r)$ as an atomic sub-formula. By (4) and (iii) $\bar{a} \in \varphi^{\mathcal{A}|_{\sigma-R}}$ iff $\bar{b} \in \varphi^{\mathcal{A}|_{\sigma-R}}$, so that either $\bar{a}, \bar{b} \in R^{\mathcal{A}}$, or $\bar{a}, \bar{b} \notin R^{\mathcal{A}}$, an hence $\mathcal{A} \models R(x_1,...,x_r)[\bar{a}]$ iff $\mathcal{A} \models R(x_1,...,x_r)[\bar{b}]$, which is a contradiction with (5).

– **$(ii) \Rightarrow (i)$**: Suppose, running towards a contradiction, that $R^{\mathcal{A}}$ has no complete $FO_{\sigma-R}$ types for r-tuples. Then, there are two r-tuples $\bar{a} \in R^{\mathcal{A}}$ and $\bar{b} \notin R^{\mathcal{A}}$ such that $tp^{FO_{\sigma-R}}_{\mathcal{A}|_{\sigma-R}}(\bar{a}) = tp^{FO_{\sigma-R}}_{\mathcal{A}|_{\sigma-R}}(\bar{b})$. But then $tp^{FO_\sigma}_{\mathcal{A}}(\bar{a}) \neq tp^{FO_\sigma}_{\mathcal{A}}(\bar{b})$, since $\mathcal{A} \models R(x_1,...,x_r)[\bar{a}]$ but $\mathcal{A} \not\models R(x_1,...,x_r)[\bar{b}]$, which is a contradiction with (ii).

□

That is, to prove that a relation $R^{\mathcal{A}}$ is redundant in a given dbi $\mathcal{A}$ we can use any of the two properties (i) or (iii) of Fact 8 above.

3.1 Kernel Databases

Though in our examples we include databases with only one redundant relation, databases may contain several redundant relations. We define a kernel database as a dbi which has no redundant relations.

Definition 9. Let $\sigma = \langle R_1, \ldots, R_s \rangle$ be a relational vocabulary, let $\mathcal{A}$ be a dbi of schema σ and let $\rho = \langle R_{i_1}, \ldots, R_{i_t} \rangle$ be a sub-vocabulary of σ. The relations $R^{\mathcal{A}}_{i_1}, \ldots, R^{\mathcal{A}}_{i_t}$ corresponding to the relations symbols in ρ are *simultaneously redundant* in $\mathcal{A}$ if for each $R_j \in \rho$, there is an $FO_{\sigma-\{R_{i_1},...,R_{i_t}\}}$ formula φ_j such that

$$\varphi_j^{\mathcal{A}|_{\sigma-\{R_{i_1},...,R_{i_t}\}}} = R^{\mathcal{A}}_j.$$

The dbi $\mathcal{A}|_{\sigma-\{R_{i_1},...,R_{i_t}\}}$ is a *kernel* if:

i. $R^{\mathcal{A}}_{i_1}, \ldots, R^{\mathcal{A}}_{i_t}$ are simultaneously redundant in $\mathcal{A}$, and

ii. for no $R \in \sigma - \{R_{i_1}, \ldots, R_{i_t}\}$ is $R^{\mathcal{A}|_{\sigma-\{R_{i_1},...,R_{i_t}\}}}$ redundant in $\mathcal{A}|_{\sigma-\{R_{i_1},...,R_{i_t}\}}$

Note that a dbi $\mathcal{A}$ can have more than one kernel. Think of a dbi $\mathcal{A}$ formed by the relations $R^{\mathcal{A}}_1$ and $R^{\mathcal{A}}_2$, where R_1 and R_2 are relation symbols of arities r_1 and r_2, respectively. Then there could exist two formulas $\varphi_1(x_1, \ldots, x_{r_1})$ and $\varphi_2(x_1, \ldots, x_{r_2})$, of vocabularies $\langle R_2 \rangle$ and $\langle R_1 \rangle$, respectively, such that $\varphi_1^{\mathcal{A}|_{\langle R_2 \rangle}} = R^{\mathcal{A}}_1$ and $\varphi_2^{\mathcal{A}|_{\langle R_1 \rangle}} = R^{\mathcal{A}}_2$.

Recall that a dbi $\mathcal{A}$ of some schema σ is *rigid* if its only automorphism is the identity function. That is, the only bijection in dom($\mathcal{A}$) which preserves all relation symbols in σ is the identity bijection. By Proposition 1, a dbi is rigid iff there are no two elements $a, b \in$ dom($\mathcal{A}$) with the same FO type for elements. Then the following Fact is straightforward.

Fact 10 *Let σ be a relational vocabulary, let $R \in \sigma$ and let $\mathcal{A}$ be a dbi of schema σ. If $\mathcal{A}|_{\sigma-R}$ is rigid then the relation $R^{\mathcal{A}}$ is redundant in $\mathcal{A}$.*

Proof. If $\mathcal{A}|_{\sigma-R}$ is rigid then, for every $r \geq 1$,

$$tp^{FO_{\sigma-R}}_{\mathcal{A}|_{\sigma-R}}(a_1, \ldots, a_r) \neq tp^{FO_{\sigma-R}}_{\mathcal{A}|_{\sigma-R}}(b_1, ..., b_r) \text{ whenever some } a_i \neq b_i (1 \leq i \leq r).$$

Thus, there are no two different r-tuples in $\mathcal{A}|_{\sigma-R}$ with the same FO type for r-tuples. On the other hand,

$$\text{if } tp^{FO_{\sigma-R}}_{\mathcal{A}|_{\sigma-R}}(a_1, ..., a_r) = tp^{FO_{\sigma-R}}_{\mathcal{A}|_{\sigma-R}}(b_1, ..., b_r),$$

$$\text{then it means that for all } 1 \leq i \leq r \text{ is } tp^{FO_{\sigma-R}}_{\mathcal{A}|_{\sigma-R}}(a_i) = tp^{FO_{\sigma-R}}_{\mathcal{A}|_{\sigma-R}}(b_i),$$

which since $\mathcal{A}|_{\sigma-R}$ is rigid implies that for all $1 \leq i \leq r$ is $a_i = b_i$. Then, never minding which relation R we add to $\mathcal{A}|_{\sigma-R}$, the FO types for different r-tuples will still be different. Hence, $R^{\mathcal{A}}$ is redundant in $\mathcal{A}$. □

Note that a dbi $\mathcal{A}$ which is *not rigid* can be "converted" to a rigid one by adding a relation which "breaks" the equivalence classes defined in the set of k-tuples of its domain, by equality of FO types for k-tuples. This can be done for instance by adding a binary relation $R^{\mathcal{A}}$ which is a total order in the domain of the dbi, since a total order defines a rigid sub-structure in the domain of the dbi. Let the domain of the dbi be $\{a_1, ..., a_n\}$, then for every $1 \leq i \leq n$ there is an FO formula $\varphi_i(x)$ which says "x is the i-th element in the total order given by $R^{\mathcal{A}}$ in dom($\mathcal{A}$)". Clearly, every such formula will be true in $\mathcal{A}$ only when x is replaced by a_i.

On the other hand, a dbi $\mathcal{A}$ which is *rigid* cannot be "converted" to a non-rigid one by adding a relation, since the FO types for different k-tuples being different in $\mathcal{A}$, means that not minding which relation we add to $\mathcal{A}$, by the definition of type, the relation of equality of FO types for k-tuples will not change. Hence, for an arbitrary relation $R^{\mathcal{A}}$, if $\mathcal{A}$ is rigid then the dbi of schema $\sigma \cup \{R\}$ form by the dbi $\mathcal{A}$ plus $R^{\mathcal{A}}$ is also rigid.

Then, the following Proposition is immediate.

Proposition 11. *Let σ be a relational vocabulary, let $\mathcal{A}$ be a dbi of schema σ. If there are relation symbols $R_1, \ldots, R_s$ in σ, such that $\mathcal{A}|_{\sigma-\{R_1,\ldots,R_s\}}$ is* rigid, *then all the relations $R_1^{\mathcal{A}}, \ldots, R_s^{\mathcal{A}}$ are simultaneously redundant in $\mathcal{A}$, and $\mathcal{A}|_{\sigma-\{R_1,\ldots,R_s\}}$ is a kernel.*

Corollary 12. *Let σ be a relational vocabulary, let $\mathcal{A}$ be a dbi of schema σ. Let $\{R_1, \ldots, R_s\}$ and $\{S_1, \ldots, S_t\}$ be two disjoint subsets of relation symbols in σ. If both $\mathcal{A}|_{\sigma-\{R_1,\ldots,R_s\}}$ and $\mathcal{A}|_{\sigma-\{S_1,\ldots,S_t\}}$ are rigid, and there are no relation symbols R and S in σ such that $\mathcal{A}|_{\sigma-\{R_1,\ldots,R_s,R\}}$ and $\mathcal{A}|_{\sigma-\{S_1,\ldots,S_t,S\}}$ are both rigid, then $\mathcal{A}|_{\sigma-\{R_1,\ldots,R_s\}}$ and $\mathcal{A}|_{\sigma-\{S_1,\ldots,S_t\}}$ are both kernels in $\mathcal{A}$.*

4 Computing Redundant Relations

First we will establish that in a fixed database of some schema σ, the problem of deciding whether a given relation in the database is redundant is decidable, as well as the problem of deciding whether there is any relation symbol in σ which is a redundant relation in the given database.

Proposition 13. *The following problems are decidable:*

i. Given a schema σ, a relation symbol $R \in \sigma$ of arity $k \geq 1$, and a database $\mathcal{A}$ of schema σ, to decide whether $R^{\mathcal{A}}$ is a redundant relation in $\mathcal{A}$.

ii. Given a schema σ and a database $\mathcal{A}$ of schema σ, to decide whether there is any relation symbol R in σ such that $R^{\mathcal{A}}$ is a redundant relation in $\mathcal{A}$.

Proof. (sketch). We use the formulas $\psi_{\bar{a}}$ of Proposition 6. We denote by $\psi_{\mathcal{A},\bar{a}}$ the formula built following that fact for the database $\mathcal{A}$. The following algorithm decides (i).

redundant := True;
For every $\bar{u} \in \mathrm{dom}(\mathcal{A})^k$ {
  Build $\psi_{\mathcal{A},\bar{u}}(\bar{x})$; Build $\psi_{\mathcal{A}|_{\sigma-R},\bar{u}}(\bar{x})$;
  For every $\bar{v} \in \mathrm{dom}(\mathcal{A})^k$ {
    # If $\bar{u}$ and $\bar{v}$ have different FO type in $\mathcal{A}$ and $\mathcal{A}|_{\sigma-R}$
    If $\neg\big(\mathcal{A} \models \psi_{\mathcal{A},\bar{u}}(\bar{x})[\bar{v}] \leftrightarrow \mathcal{A}|_{\sigma-R} \models \psi_{\mathcal{A}|_{\sigma-R},\bar{u}}(\bar{x})[\bar{v}]\big)$ then {
      redundant := False; Return redundant } } };
Return redundant;

As relational database schemas have a finite number of relation symbols. We can decide (ii) by simply checking, using the previous algorithm, whether for some relation symbol R in σ, $R^{\mathcal{A}}$ is a redundant relation. $\square$

Unfortunately, the algorithm we gave in the proof of Proposition 13 to decide whether a given relation is redundant in a given database, has exponential time complexity. Note that while the formulas $\psi_{\mathcal{A},\bar{a}}$ of the previous proposition and Proposition 6 can be built in polynomial time, their evaluation on a given database takes time $\mathcal{O}(n^n)$, since we must consider all valuations on the n variables of the formulas to that end. It is very unlikely that there is a polynomial time algorithm for this problem since it is equivalent to deciding isomorphism. In this section, we attack this problem by restricting:

a. the *notion of redundant relations* to relations which are definable in logics which are less expressive than first-order logic.

b. the *class of databases* to classes where deciding whether a relation is redundant in a database which belongs to the class is in P.

We need first the following definition.

Definition 14. Let $\mathcal{L}$ be a logic, let σ be a relational schema, let $\mathcal{A}$ be a database of schema σ, let $r \geq 1$, and let R be an r-ary relation symbol in σ. We say that $R^{\mathcal{A}}$ is an $\mathcal{L}$-*redundant relation* in the database $\mathcal{A}$ if there is an $\mathcal{L}$-formula $\phi_R(x_1, \ldots, x_r)$ such that, for every computable query q, it holds that $q(\mathcal{A}) = q(\langle \mathcal{A}|_{\sigma-R}, \phi_R^{\mathcal{A}|_{\sigma-R}} \rangle)$, where $\langle \mathcal{A}|_{\sigma-R}, \phi_R^{\mathcal{A}|_{\sigma-R}} \rangle$, of schema σ, denotes the reduced database $\mathcal{A}|_{\sigma-R}$ augmented with the relation defined by the formula ϕ_R in $\mathcal{A}|_{\sigma-R}$.

As a consequence of this definition and Proposition 7, we get the following.

Fact 15 *Let R be a relation symbol in σ of arity r. The relation $R^{\mathcal{A}}$ is FO-redundant in a database $\mathcal{A}$ iff it is redundant in the sense of Definition 5.*

Remark. In [Ferrarotti and Turull 2008] a restricted second-order logic called SO^{ω} (which was first introduced in [Dawar 1998]) was studied, where second-order quantifiers range over relations that are closed under equality of FO^k types of k-tuples, and it was proved that SO^{ω} captures the relational polynomial-time hierarchy. In [Grosso and Turull 2009], the logic *SOF* was defined, where valuations assign to r-ary second-order variables, relations which are closed under equality of *FO* types for r-tuples. Note that from the perspective of the present article, valuations in SO^{ω} can assign only FO^k-redundant relations to second-order variables, for some k, and valuations in *SOF* can assign only *FO*-redundant relations to relational variables. However, the use of redundant relations as intermediate results turned out to be *relevant* as to expressive power. Even some NP-complete problems can be expressed in the existential fragments of SO^{ω} and *SOF* (recall that the existential fragment of *SO* captures NP and the existential fragment of SO^{ω} captures relational NP), even when as we show in this article, they do not alter the information contents of a database, in a given precise way.

4.1 FO^k and C^k Redundant Relations

Fact 16 *Let $k \geq 1$,*

i. a relation $R^{\mathcal{A}}$ of arity $1 \leq r \leq k$ is FO^k-redundant in a database $\mathcal{A}$ iff for all r-tuples $\bar{u}$ and $\bar{v}$ in $\mathrm{dom}(\mathcal{A})^r$, $tp_{\mathcal{A}}^{FO^k_{\sigma}}(\bar{u}) = tp_{\mathcal{A}}^{FO^k_{\sigma}}(\bar{v})$ iff $tp_{\mathcal{A}|_{\sigma-R}}^{FO^k_{\sigma-R}}(\bar{u}) = tp_{\mathcal{A}|_{\sigma-R}}^{FO^k_{\sigma-R}}(\bar{v})$,

ii. a relation $R^{\mathcal{A}}$ of arity $1 \leq r \leq k$ is C^k-redundant in a database $\mathcal{A}$ iff for all r-tuples $\bar{u}$ and $\bar{v}$ in $\mathrm{dom}(\mathcal{A})^r$, $tp_{\mathcal{A}}^{C^k_{\sigma}}(\bar{u}) = tp_{\mathcal{A}}^{C^k_{\sigma}}(\bar{v})$ iff $tp_{\mathcal{A}|_{\sigma-R}}^{C^k_{\sigma-R}}(\bar{u}) = tp_{\mathcal{A}|_{\sigma-R}}^{C^k_{\sigma-R}}(\bar{v})$.

Proof. We prove item (i). Item (ii) is completely analogous. Running towards a contradiction, let us suppose that $R^{\mathcal{A}}$ is an FO^k-redundant relation while there exists tuples $\bar{u}, \bar{v} \in \mathrm{dom}(\mathcal{A})^r$ such that

$$tp_{\mathcal{A}}^{FO^k_\sigma}(\bar{u}) \neq tp_{\mathcal{A}}^{FO^k_\sigma}(\bar{v}) \text{ and } tp_{\mathcal{A}|_{\sigma-R}}^{FO^k_{\sigma-R}}(\bar{u}) = tp_{\mathcal{A}|_{\sigma-R}}^{FO^k_{\sigma-R}}(\bar{v}).$$

If that is the case, then either $\bar{u} \in R^{\mathcal{A}}$ and $\bar{v} \notin R^{\mathcal{A}}$, or vice-versa. Let us assume w.l.o.g. that $\bar{u} \in R^{\mathcal{A}}$ and $\bar{v} \notin R^{\mathcal{A}}$. Let $\phi_R \in FO^k$ be the formula required by Definition 14 to determine that $R^{\mathcal{A}}$ is FO^k-redundant. Since $\phi_R^{\mathcal{A}|_{\sigma-R}}$ has complete FO^k-types, it contains all tuples in $R^{\mathcal{A}}$ plus any r-tuple in $\mathrm{dom}(\mathcal{A})$ whose FO^k-type in $\mathcal{A}|_{\sigma-R}$ coincides with the FO^k-type of some tuple in $R^{\mathcal{A}}$. In particular, given that $\bar{u} \in R$, then both $\bar{u}$ and $\bar{v}$ are in $\phi_R^{\mathcal{A}|_{\sigma-R}}$. Thus, $\phi_R^{\mathcal{A}|_{\sigma-R}}$ strictly includes $R^{\mathcal{A}}$. Now, let q be the computable query expressed by the formula $R(x_1, \dots, x_r)$. We have that $q(\mathcal{A}) = R^{\mathcal{A}}$ is strictly included in $q(\langle \mathcal{A}|_{\sigma-R}, \phi_R^{\mathcal{A}|_{\sigma-R}} \rangle) = \phi_R^{\mathcal{A}|_{\sigma-R}}$, which contradicts our hypothesis.

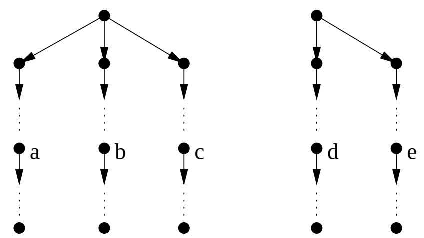
In the other direction, suppose that for all r-tuples $\bar{u}, \bar{v} \in \mathrm{dom}(\mathcal{A})^r$,

$$tp_{\mathcal{A}}^{C^k_\sigma}(\bar{u}) = tp_{\mathcal{A}}^{C^k_\sigma}(\bar{v}) \text{ iff } tp_{\mathcal{A}|_{\sigma-R}}^{C^k_{\sigma-R}}(\bar{u}) = tp_{\mathcal{A}|_{\sigma-R}}^{C^k_{\sigma-R}}(\bar{v}).$$

Then $R^{\mathcal{A}}$ has complete FO^k-types in $\mathcal{A}|_{\sigma-R}$, i.e., if there is an r-tuple $\bar{a} \in R^{\mathcal{A}}$ whose FO^k-type is α, then every r-tuple $\bar{b} \in \mathrm{dom}(\mathcal{A})^r$ which has the same FO^k-type α than $\bar{a}$ is also in $R^{\mathcal{A}}$. Let $\alpha_1, \dots, \alpha_m$ the different FO^k-types realized by the tuples of $R^{\mathcal{A}}$. Then, there is an FO^k-formula $\phi_R \equiv \chi_1 \vee \dots \vee \chi_m$, where for $1 \leq i \leq m$, χ_i is the isolating formula of Proposition 2 for the type α_i, such that $R^{\mathcal{A}} = \phi_R^{\mathcal{A}|_{\sigma-R}}$. Since ϕ_R is also a first-order logic formula, by Fact 8 it follows that $R^{\mathcal{A}}$ is a redundant relation in $\mathcal{A}$ and also a FO^k-redundant relation. □

Note that, for a fixed k, a relation $R^{\mathcal{A}}$ can be FO-redundant in $\mathcal{A}$ and not be FO^k-redundant at the same time. Let us see an example.

Example 4. Let $\mathcal{G}$ be the database of schema $\tau = \langle E \rangle$, where $E^{\mathcal{G}}$ is the edge relation corresponding to the following directed graph formed by two disconnected trees.



Let us assume that the distance from the root of the left hand side tree to the nodes a b and c, as well as the distance from the root of the right hand side tree to the nodes d and e, is in all cases m. We also assume that the distance from a, b, c, d and e to their corresponding leaves, is also the same in all cases. Clearly, to add the relation $R_1^{\mathcal{G}} = \{a, b, c\}$ to the database $\mathcal{G}$, would be redundant since it can be defined on $\mathcal{G}$ by the following *FO*-formula:

$$\phi_{R_1}(x') \equiv \exists x \forall y(\neg E(y,x) \wedge \exists y\, z\, w(E(x,y) \wedge E(x,z) \wedge E(x,w) \wedge y \neq z \neq w) \wedge$$

$$\text{“There is a path from } x \text{ to } x' \text{ of length } m\text{”})$$

Note that, we need only three variables to express “there is a path from x to x' of length m” as we can re-use variables. For instance, if $m = 3$, that formula could be $\varphi(x') \equiv \exists y(E(y,x') \wedge \exists x'(E(x',y) \wedge \exists y(E(y,x') \wedge \forall x'(\neg E(x',y)))))$. But, we need four variables to distinguish a, b and c from d and e, since they also are at distance m from the root. In fact, since a, b, c, d and e, have all the same FO^3-type in $\mathcal{G}$, there is no FO^3-formula that can distinguish a, b and c from d and e. Therefore, $R_1^{\mathcal{G}}$ is not FO^3-redundant in $\mathcal{G}$. If we add to $\mathcal{G}$ the relation $R_2^{\mathcal{G}} = \{d, e\}$, again we would have the same situation for exactly the same reasons. That is, the relation $R_2^{\mathcal{G}}$ would be *FO*-redundant, but it would not be FO^3-redundant. Now, if we add the relation $R_3^{\mathcal{G}} = \{a, b, c, d, e\}$ to $\mathcal{G}$, it would be not only *FO* redundant, but also FO^3-redundant.

Remark. For every k, FO^{k+1}-types are refinements of FO^k-types, i.e., the set of r-tuples $(1 \leq r \leq k)$ of a given FO^{k+1}-type realized by a database $\mathcal{A}$, is a subset of (or equal to) the set of r-tuples of an FO^k-type realized by $\mathcal{A}$. Thus, we can have a database $\mathcal{A}$ of some schema σ and a relation symbols $R \in \sigma$ of arity r $(1 \leq r \leq k)$ such that the corresponding relation $R^{\mathcal{A}}$ has complete FO^{k+1}-types in the reduced database $\mathcal{A}|_{\sigma - R}$ and does not have complete FO^k-types in the same reduced database. This implies that the relation of equivalence of FO^k-types of $\mathcal{A}$ is altered if we eliminate $R^{\mathcal{A}}$. That is, it is *not* true that, for all r-tuples $\bar{u}$ and $\bar{v}$ in $\mathrm{dom}(\mathcal{A})^r$, $tp_{\mathcal{A}}^{FO^k_\sigma}(\bar{u}) = tp_{\mathcal{A}}^{FO^k_\sigma}(\bar{v})$ iff $tp_{\mathcal{A}|_{\sigma-R}}^{FO^k_{\sigma-R}}(\bar{u}) = tp_{\mathcal{A}|_{\sigma-R}}^{FO^k_{\sigma-R}}(\bar{v})$. Therefore, by Fact 16, $R^{\mathcal{A}}$ is not FO^k-redundant in $\mathcal{A}$. However, it is FO^{k+1}-redundant as there is a FO^{k+1}-formula ϕ_R such that $\phi_R^{\mathcal{A}|_{\sigma-R}} = R^{\mathcal{A}}$. ϕ_R is simply the disjunction of the FO^{k+1} isolating formulas (see Proposition 2) for the different FO^{k+1}-types realized in $\mathcal{A}$ by the tuples in $R^{\mathcal{A}}$.

The observation in the previous remark also holds if we replace the logic FO^k by C^k. Therefore, for a fixed k, it also holds that a relation $R^{\mathcal{A}}$ can be *FO*-redundant in $\mathcal{A}$ and not be C^k-redundant.

By a result of Grohe, equivalence in FO^k is complete for polynomial time.

Proposition 17. *[Grohe 1996] For every $k \geq 1$, the following problems are complete for polynomial time:*

i. Given two databases $\mathcal{A}$ and $\mathcal{B}$ of schema σ, is it the case that $\mathcal{A} \equiv_{FO^k} \mathcal{B}$?

ii. Given a database $\mathcal{A}$ of schema σ and two r-tuples $\bar{u}, \bar{v} \in \mathrm{dom}(\mathcal{A})^r$ with $1 \leq r \leq k$, is it the case that $tp_{\mathcal{A}}^{FO^k_\sigma}(\bar{u}) = tp_{\mathcal{A}}^{FO^k_\sigma}(\bar{v})$?

The same is also true for C^k.

Proposition 18. *[Grohe 1996] For every $k \geq 1$, the following problems are complete for polynomial time:*

i. Given two databases $\mathcal{A}$ and $\mathcal{B}$ of schema σ, is it the case that $\mathcal{A} \equiv_{C^k} \mathcal{B}$?

ii. Given a database $\mathcal{A}$ of schema σ and two r-tuples $\bar{u}, \bar{v} \in \mathrm{dom}(\mathcal{A})^r$ with $1 \leq r \leq k$, is it the case that $tp_{\mathcal{A}}^{C^k_\sigma}(\bar{u}) = tp_{\mathcal{A}}^{C^k_\sigma}(\bar{v})$?

Then we can check in P, FO^k equivalence as well as C^k equivalence between every two extensions of a database with any given pair of tuples. So, we have the following important propositions.

Proposition 19. *Let $k \geq 1$. Given a schema σ, a relation symbol $R \in \sigma$ of arity r ($1 \leq r \leq k$) and a database $\mathcal{A}$ of schema σ, to decide whether $R^{\mathcal{A}}$ is a FO^k-redundant relation in $\mathcal{A}$, is in P.*

Proof. By Fact 16 (i), we only need to check whether, for every r-tuple $\bar{u}, \bar{v} \in \mathrm{dom}(\mathcal{A})$, it holds that

$$tp_{\mathcal{A}}^{FO^k_\sigma}(\bar{u}) = tp_{\mathcal{A}}^{FO^k_\sigma}(\bar{v}) \text{ iff } tp_{\mathcal{A}|_{\sigma-R}}^{FO^k_{\sigma-R}}(\bar{u}) = tp_{\mathcal{A}|_{\sigma-R}}^{FO^k_{\sigma-R}}(\bar{v})$$

Since there are n^r r-tuples in $\mathcal{A}$ and, by Proposition 17, the test for equality of FO^k types is in P, it follows that these checks can be computed in polynomial time. $\square$

The same proposition holds for the case of deciding C^k redundancy. We omit the proof as it is similar to the proof for FO^k redundancy.

Proposition 20. *Let $k \geq 1$. Given a schema σ, a relation symbol $R \in \sigma$ and a database $\mathcal{A}$ of schema σ, to decide whether $R^{\mathcal{A}}$ is a C^k-redundant relation in $\mathcal{A}$, is in P.*

Remark. As to the existence for FO^k-redundant relations of an equivalent result to Proposition 7, unfortunately it seems very unlikely. By using the isolating formulas for FO^k-types for k tuples (denoted as χ in Proposition 2), we can indeed build a formula $\phi_R \in FO^k$ which defines R in $\mathcal{A}|_{\sigma-R}$, but that formula is of size $\mathcal{O}(n^{(n^k)})$. And the same is true also for C^k-redundant relations (see [Otto 1996]).

4.1.1 A Database Design Perspective of FO^k and C^k Redundancy

From a database design point of view, a redundant relation $R^{\mathcal{A}}$ in a database $\mathcal{A}$ of schema σ, might indicate the existence of a computable query q which would represent the design intention behind the inclusion of the relation symbol R in σ. That is, it might indicate that there is a computable query q such that for every database $\mathcal{A}_i$ of schema σ, $q(\mathcal{A}_i) = R^{\mathcal{A}_i}$. In particular, for C^k-redundant relations, and also for FO^k redundant relations, it might indicate that such query q belongs to a well studied class of computable queries which is strictly included on $\mathcal{CQ}$. More precisely, in those cases q might belong to one of the classes that characterize the expressive power of some variations of the *reflective relational machine* (RRM) developed in [Abiteboul et al. 1998].

Roughly, an RRM is a deterministic Turing machine with an additional *relational store* (rs) and a *query tape*. The input database, the output relation, and a set of auxiliary relations form the rs. The machine can access relations in the rs *only* through formulas of First Order Logic (FO), which in turn are generated dynamically in the query tape. This feature is what enforces preservation of isomorphisms in the queries computed by the machine.

In [Turull 2004] a strict hierarchy was defined in $\mathcal{CQ}$, in terms of the preservation of equivalence in FO^k. 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 RRM of [Abiteboul et al. 1998] if we restrict to k the number of different variables which can be used in any FO query generated during a computation (i.e., if we restrict what is known as the *variable complexity* of the model). The class of RRM machines with variable complexity k is usually denoted as RRM^k.

A variation of RRM called *reflective counting machine* (RCM) was defined in [Turull 2006] together with a characterization of its expressive power through a hierarchy denoted as $\mathcal{QCQ}^{C^{\omega}}$. This hierarchy was defined in terms of the preservation of equivalence in 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 RCM with variable complexity k (RCM^k) is defined as a variant of the RRM^k in which the dynamic queries are formulas in the logic C^k, instead of FO^k. For every natural k, the layer denoted as $\mathcal{QCQ}^{C^k}$ characterizes exactly the expressive power of the RCM^k.

The following fact is a direct consequence of Definition 14 and the fact that the $\mathcal{QCQ}^k$ and $\mathcal{QCQ}^{C^k}$ classes preserve equality of FO^k types and C^k types, respectively, in the set of k-tuples of a database.

Fact 21 *Let σ be a relational schema, let $\mathcal{A}$ be a database of schema σ and let R be a relation symbol in σ of arity r. For all query $q \in \mathcal{QCQ}^k$ ($\mathcal{QCQ}^{C^k}$), if $q(\mathcal{A}) = R^{\mathcal{A}}$, then $R^{\mathcal{A}}$ is FO^k-redundant (C^k-redundant) in the sense of Definition 14,* *i.e., there is an FO^k-formula (C^k-formula) $\phi_R(x_1, \ldots, x_r)$ such that, for every computable query q', it holds that $q'(\mathcal{A}) = q'(\langle \mathcal{A}|_{\sigma-R}, \phi_R^{\mathcal{A}|_{\sigma-R}} \rangle)$.*

Example 5. Let $\sigma = \langle E, F \rangle$ and let $\mathcal{A} = \langle D^{\mathcal{A}}, E^{\mathcal{A}}, F^{\mathcal{A}} \rangle$ be a relational structure of schema σ such that $F^{\mathcal{A}}$ is the binary relation which contains the transitive closure of the directed graph with domain $D^{\mathcal{A}}$ and edge relation $E^{\mathcal{A}}$. Since the transitive closure query belongs to $\mathcal{QCQ}^3$, it follows that there is an FO^3-formula that express it in $\mathcal{A}$. Recall that for every fixed n, the query "there is a path from node x to node y of length n" can be expressed by a formula $\psi_n(x, y)$ in FO^3 by re-using variables. Suppose $|D^{\mathcal{A}}| = n$, the following FO^3 formula $\phi_F(x, y) \equiv \psi_1(x, y) \vee \ldots \vee \psi_{n-1}(x, y)$, when evaluated in $\mathcal{A}$, returns the transitive closure relation $F^{\mathcal{A}}$. Then, $F^{\mathcal{A}}$ is FO^3-redundant in $\mathcal{A}$, since there is an FO^3 formula which satisfies Definition 14.

As to C^k redundancy, let us consider the query $q =$ "pairs of nodes with the same out-degree" on the same schema σ. This query is in $\mathcal{QCQ}^{C^2}$ and hence, for every dbi, there is a C^2-formula that expresses it. For instance, in a dbi $\mathcal{A} = \langle D^{\mathcal{A}}, E^{\mathcal{A}}, F^{\mathcal{A}} \rangle$ where $F^{\mathcal{A}} = q(\mathcal{A})$, if $\mathcal{A}$ has n vertices, the C^2 formula $\varphi_F(x, y) \equiv \bigvee_{i \leq n} \big(\exists^{\geq i} y(E(x, y)) \wedge \neg \exists^{\geq i+1} y(E(x, y)) \wedge \exists^{\geq i} x(E(y, x)) \wedge \neg \exists^{\geq i+1} x(E(y, x))\big)$ expresses q. And hence $F^{\mathcal{A}}$ is C^2-redundant in $\mathcal{A}$.

We believe this observation is of interest because on one hand, by Proposition 20, C^k-redundancy is decidable in polynomial time, and on the other hand, for $k \geq 2$, the classes $\mathcal{QCQ}^{C^k}$ capture a relevant portion of the class $\mathcal{CQ}$ of computable queries. Following [Hella et al. 1996] though using a slightly different perspective, we define the notion of equality of queries *almost everywhere*, as follows:

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

where q, q' are computable queries of schema σ. If $\mathcal{C}$ is a class of finite structures,

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

Let us consider the following result.

Proposition 22. *([Babai et al. 1980, Immerman and Lander 1990]) There is a class $\mathcal{C}$ of graphs with $\mu_{\mathcal{C}} = 1$ such that for all graphs $\mathcal{I}, \mathcal{J} \in \mathcal{C}$ we have $\mathcal{I} \simeq \mathcal{J}$ iff $\mathcal{I} \equiv_{C^2} \mathcal{J}$. Moreover, for all $\mathcal{I} \in \mathcal{C}$ and $a, b \in \mathrm{dom}(\mathcal{I})$, there is an automorphism mapping a to b iff $tp_{\mathcal{I}}^{C^2}(a) = tp_{\mathcal{I}}^{C^2}(b)$.*

Then it follows that, 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. Furthermore, there is a large amount of relevant queries, which are not expressible in relational calculus (or FO), that belong to the lower levels of the $\mathcal{QCQ}^{\omega}$ and $\mathcal{QCQ}^{C^{\omega}}$ hierarchies.

(i) Assume we have a database with a ternary relation R such that a tuple (a, b, c) is in R iff the supplier a supplies part b to project c. Then, there is an RCM^3 machine which computes the query "suppliers who supply the biggest amount of different parts supplied by any supplier in the database". Thus, this query is in the class $\mathcal{QCQ}^{C^3}$.
(ii) The property of the graph being *regular of even degree*, or equivalently of having an *Eulerian cycle*, is decidable by an RCM^2 machine and then it is in the class $\mathcal{QCQ}^{C^2}$ [Kolaitis and Väänänen 1995].
(iii) There is an RRM^3 machine which decides whether a graph is connected [Grohe 1998]. This shows that *connectivity* is in $\mathcal{QCQ}^3$.
(iv) The problem usually known as *parity* consisting in determining whether the cardinality of the domain of a database is even, belongs to $\mathcal{QCQ}^{C^1}$, i.e., there is an RCM^1 machine which decides parity.
(v) There is an RRM^3 machine which computes *transitive closure* over graphs. So, this problem is in $\mathcal{QCQ}^3$.
(vi) By a result from [Kolaitis and Väänänen 1995], there is an RCM^2 machine that decides whether a binary relation R is an equivalence relation with an even number of equivalence classes. That means that this problem is in $\mathcal{QCQ}^{C^2}$.

4.2 Subclasses of Databases

Now, we consider the second kind of restriction that we mentioned at the beginning of this section.

Proposition 23. *Let $k \geq 1$ and let $\mathcal{C}$ be a class of databases in which C^k (FO^k) equivalence coincides with isomorphism. Then, the problem of deciding whether a given relation is redundant in a database which belongs to $\mathcal{C}$, is in P, as well as the problem of deciding whether a given database in $\mathcal{C}$ has any redundant relation.*

Proof. (sketch). By Proposition 20 the problem of deciding whether a given relation is C^k-redundant is decidable in P. As we are considering only classes of databases in which C^k equivalence coincides with isomorphism, then in those classes deciding whether a given relation is C^k-redundant in a database coincides with deciding whether it is FO-redundant. $\square$

Some examples of classes where C^k equivalence coincides with isomorphism are: (i) the class of *planar* graphs, where there is a $k \geq 1$ such that C^k equivalence coincides with isomorphism; (ii) for all $k \geq 1$, the class of graphs of k-*bounded tree-width* [Grohe and Mariño 1998], where C^{k+3} equivalence coincides with isomorphism; (iii) the class of *trees*, where C^2 equivalence coincides with isomorphism.

Regarding FO^k, in the class of *linear* graphs [Ebbinghaus and Flum 1999], FO^2 equivalence coincides with isomorphism, and in the class of graphs with *color class size* ≤ 3 [Grohe 1998], FO^3 equivalence coincides with isomorphism. In these two classes, the problem of deciding whether a given relation is redundant, as well as the problem of deciding whether a given database has any redundant relation, is in P.

Note that, even if C^k equivalence and FO^k equivalence do not coincide with isomorphism, we have the following result.

Fact 24 *Let C be a class of databases in which isomorphism is decidable in P, then the problem of deciding whether a given relation is redundant in a database which belongs to $\mathcal{C}$, is in P, as well as the problem of deciding whether a given database in $\mathcal{C}$ has any redundant relation.*

Proof. Let $\mathcal{C} \subseteq \mathcal{DB}_\sigma$, where for all dbi $\mathcal{A}$, $\mathcal{B}$ in $\mathcal{C}$, checking $\mathcal{A} \simeq \mathcal{B}$ is in P. Then, by Proposition 1, $\equiv_{FO}$ is also in P, so that $\langle \mathcal{A}, \bar{a} \rangle \equiv_{FO} \langle \mathcal{A}, \bar{b} \rangle$ can be checked in P, where $\langle \mathcal{A}, \bar{a} \rangle$ is a dbi of schema $\tau = \sigma \cup \{c_1, \ldots, c_r\}$ with constant symbols $c_1, \ldots, c_r$, and $\bar{a} \in \mathrm{dom}(\mathcal{A})^r$, for some $r \geq 1$. Then the following algorithm is in P and decides whether $R^{\mathcal{A}}$ is redundant in $\mathcal{A}$:

```
redundant := True;
For every ā ∈ R^A {
    For every b̄ ∈ dom(A)^r {
        If ¬(⟨A, ā⟩ ≡_FO ⟨A, b̄⟩ iff ⟨A|_{σ−R}, ā⟩ ≡_FO ⟨A|_{σ−R}, b̄⟩) then
            redundant := False; Return redundant   }   };
Return redundant;
```

Note that $\langle \mathcal{A}, \bar{a} \rangle \equiv_{FO} \langle \mathcal{A}, \bar{b} \rangle$ iff $tp_{\mathcal{A}}^{FO_\sigma}(\bar{a}) = tp_{\mathcal{A}}^{FO_\sigma}(\bar{b})$. □

The classes of linear graphs, trees, planar graphs and graphs with bounded tree-width are examples of such classes where isomorphism is decidable in P.

References

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

[Abiteboul et al. 1998] Abiteboul, S., Papadimitriou, C. and Vianu, V.: *Reflective Relational Machines.* Information and Computation 143, pp.110–136, 1998.

[Babai et al. 1980] Babai, L., Erdös, P. and Selkow, S.: *Random Graph Isomorphism.* SIAM Journal on Computing 9, pp. 628–635, 1980.

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

[Dawar 1998] Dawar, A.: *A Restricted Second Order Logic for Finite Structures.* Inf. Comput. 143, pp. 154–174, 1998.

[Dawar 1993] Dawar, A.: *Feasible Computation Through Model Theory.* Ph.D. thesis, University of Pennsylvania, Philadelphia, 1993.

[Ebbinghaus and Flum 1999] Ebbinghaus, H. D. and Flum, J.: *Finite Model Theory.* Springer-Verlag Berlin Heidelberg, New York, 2nd. ed., 1999.

[Ferrarotti, et al. 2009] Ferrarotti, F. A., Paoletti, A. L. and Turull Torres, J. M. *First-Order Types and Redundant Relations in Relational Databases.* Proceedings of ER 2009 Workshops: First International Workshop in Evolving Theo- ries of Conceptual Modelling, ETheCoM 2009, LNCS 5833, pp. 65–74, 2009.

[Ferrarotti and Turull 2008] Ferrarotti, F. A. and Turull Torres J. M.: *The Relational Polynomial-Time Hierarchy and Second-Order Logic.* Proceedings of Semantics in Data and Knowledge Bases: Third international Workshop, SDKB 2008, Springer, LNCS 4925, pp. 48–76, 2008.

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

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

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

[Grosso and Turull 2009] Grosso A. and Turull Torres J. M.: *SOF: A Second-Order Logic in which Second-Order Quantifiers Range over Relations Closed Under Equality of First-Order Types.* Manuscript, 2008.

[Hella et al. 1996] 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.

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

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

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

[Otto 1997] Otto, M.: *Bounded Variable Logics and Counting.* Springer (1997).

[Paoletti 2005] Paoletti, A. L.: *The Model Theoretic Notion of Type in Relational Databases.* Master Thesis, Massey University, New Zealand, 2005.

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

[Turull 2006] Turull Torres, J. M.: *Relational Databases and Homogeneity in Logics with Counting.* Acta Cybernetica, 17(3), pp. 485–511, 2006.

[Ullman 1988] Ullman, J. D.: *Principles of Database and Knowledge Base Systems.* Volume I and II. Computer Science Press, 1988.