The Model Theoretic Notion of Type in Relational Databases

A thesis presented in partial fulfillment of the requirements for the degree of

Master
of
Information Sciences

at Massey University,
Wellington,
New Zealand.

Alejandra Lorena Paoletti

2005

Dr. José María Turull-Torres
Thesis' Supervisor

Abstract. It is well known that finite model theory and database theory are two disciplines intimately connected. A topic which has been deeply studied in the context of finite model theory as well as in classic model theory, but which has not received the same attention in the context of databases, is the concept of *type* of a tuple. We believe that it is very important to have a clear understanding of the implications of the concept of type in the field of databases. Therefore, we study here the notion of type in FO (first-order logic) and in FO^k (the restriction of FO to formulas with up to k variables), as well as in their corresponding infinitary extensions $\mathcal{L}_{\infty\omega}$ and $\mathcal{L}^k_{\infty\omega}$, respectively. The FO^k types ($\mathcal{L}^k_{\infty\omega}$ types) are quite relevant in database theory since they characterize the discerning power of the class of reflective relational machines of S. Abiteboul, C. Papadimitriou and V. Vianu with variable complexity k. We examine the three main characterizations of FO equivalence. These characterizations are based in Ehrenfeucht-Fraïssé games, back and forth systems of partial isomorphisms and isolating formulas for types. We study in detail all of them because they give a clear idea of the relevance of the notion of type in the context of databases, and make the fact that the types of the tuples in a database reflect all the information contained in it very clear. We found that the concept of type turned out to be useful to examine from a new perspective a well known problem of the field of databases, consisting on the redundant storage of information. Specifically, we initiate a study towards a method of *normalization* of relational databases based on the detection of *redundant relations*, i.e., relations which can be eliminated from the database without loosing information, by using *isolating formulas* for the types of the tuples in the database.

Acknowledgements

This project would have been impossible to finish without the knowledge and encouragement of many people

First, I would like to express a special acknowledgement to my supervisor, Associate Professor, Dr. José María Turull-Torres (Massey University) for sharing his expertise and knowledge and for his valuable guidance.

I would like to thank Massey University, College of Business, for offering me a Graduate Assistant position. Particularly, I would like to thank the Head of Department of Information Systems, Associate Professor, Chris Freyberg, for his support.

I also wish to thank to the reviewers of this thesis, Dr. Alejandro Petrovich (University of Buenos Aires) and Professor, Dr. Klaus-Dieter Schewe (Massey University).

From the beginning of this project I have been in the most encouraged working environment provided by my partner, Flavio, to whom I am deeply grateful for his constant support.

Finally, I would like to thank my family, specially to my parents, Lilia and Lucio, and my sisters, Andrea and Vanesa, for their support from the other side of the world.

Table of Contents

1 Introduction 9
2 Preliminaries 12
  2.1 Finite Model Theory and Databases 12
  2.2 Computable Queries 17
3 Type of a Tuple 19
  3.1 The Notion of *FO* Type in Databases 21
4 Partial Isomorphisms 23
5 Ehrenfeucht-Fraïssé Games 25
  5.1 Inductive construction of the m-isomorphic type 27
6 Pebble Games 34
  6.1 Inductive construction of the k-m-isomorphic type 37
  6.2 Isolating Formula 39
7 Databases with Redundant Relations 42
  7.1 Redundant Relations in Databases 43
  7.2 Redundant Relations in Classes of Databases 51
8 Conclusion and Future Work 58

1 Introduction

Finite model theory and database theory are two disciplines intimately connected. While finite model theory provides a solid theoretical foundation to databases, databases provides one of the main concrete scenarios for finite model theory within computer science. Most of the overlaps between finite model theory and database theory occur in the theory of query languages. In consequence, this has been the area which has received more attention. Here, we concentrate in a topic which has been deeply studied in the context of finite model theory [8, 18] as well as in classic model theory [6], but which has not received the same attention in the context of databases, namely the concept of *type* of a tuple. Roughly, if $\mathcal{L}$ is a logic, the $\mathcal{L}$ type of a tuple of length k in a given database (relational structure) is the set of $\mathcal{L}$ formulas with up to k free variables which are satisfied by that tuple in the database.

From a conceptual point of view it is desirable for a model of computation of queries to be *representation independent* [24]. This means, roughly, that queries to databases which represent the "same" reality should evaluate to the "same" result. In mathematical terms, Chandra and Harel [5] captured the previous concept by asking queries to isomorphic databases to evaluate to the same result. 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 indistinguishable. 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 (signature). The same is also true for tuples of elements, i.e., two tuples with the same "structural" properties should be considered as indistinguishable. 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 indistinguishable, 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. A "bad" design could lead to unwanted results as redundant storage of information within the database. This anomaly is usually known as the *redundancy problem* and has been studied extensively in the field of databases, indeed almost all database textbooks include this topic, see for instance [1, 20]. Redundant storage of information can lead to a variety of practical problems on the updating, insertion and deletion of data. For instance, if a copy of some redundant information stored in a database is updated while other copy is not, we would end up with an inconsistency; or it may not be possible to store certain information unless some other, unrelated, information is stored as well; or it may not be possible to delete certain information without losing other. Traditionally, the redundancy problem is studied by considering a particular class of properties, called *functional dependencies*, that are supposed to be satisfied by all instances of a given database. Afterwards, the database is usually decomposed (i.e., some relations are replaced by two or more relations of smaller arities) in order to satisfy some desired *normal forms*, which are defined by using functional dependencies (see [1, 20]).

By taking a quite different approach, the redundancy problem will provide within this work a concrete scenario to apply the concept of type. 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 in practical terms, we do not lose information if we eliminate such redundant relation from a database.

Given that the first-order types of all k-tuples in a given database $\mathcal{A}$ describe all the FO properties which are satisfied by the tuples of arity up to k in $\mathcal{A}$, we can eliminate a relation $R^{\mathcal{A}}$ (i.e., the interpretation in $\mathcal{A}$ of the relation symbol R) from $\mathcal{A}$ as long as the first-order types of all k-tuples in $\mathcal{A}$ are not altered, because we can always write an FO query which evaluated on the new reduced database will give us the relation $R^{\mathcal{A}}$. So, by using this fact we will study redundant relations in databases as well as in classes of databases in connection with the concept of type. Up to our knowledge, this subject has not been previously investigated in the field of databases.

Two databases are called FO equivalent if they satisfy the same FO sentences. Moreover, if two databases are FO equivalent, then they are isomorphic, i.e., indistinguishable. Here we consider the three main characterizations of FO equivalence, and in all of them the concept of type plays a central role. These characterizations are based in Ehrenfeucht-Fraïssé games, back and forth systems of partial isomorphisms and isolating formulas for types. We examine in detail all of them because they give a clear idea of the relevance of the notion of type in the context of databases, and make the fact that the types of the tuples in a database reflect all the information contained in it very clear.

We organize this work as follows. In Section 2 we give a brief background of the main technical concepts regarding finite model theory and its relation to database theory, and computable queries as well as the notation we will use throughout this work. The concepts of type of a tuple introduced in Section 3 and of partial isomorphism introduced in Section 4, are central theoretical concepts used in the present work. The concept of partial isomorphism is later used to provide an algebraic characterization of the winning strategies in the Ehrenfeucht-Fraïssé games and in the pebble games, which are introduced in Sections 5 and 6, respectively. These games are useful for determining what can be expressed in various logics. In particular, we will see a Ehrenfeucht-Fraïssé game for first-order logic and infinitary logic, and a pebble game for their respective fragments with bounded number of variables. The concept of FO types of tuples in databases and their isolating formulas are extensively used in Section 7 where we initiate a study towards a method of *normalization* of relational databases with redundant relations. We start with the study of redundant relations over single databases in Subsection 7.1, then we extend this concept to classes of databases in Subsection 7.2. A relation symbol is redundant in a class if its interpretation in all databases in the class is a redundant relation. 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, 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. Nevertheless, regarding classes of databases, those problems are clearly not decidable in the general case.

2 Preliminaries

2.1 Finite Model Theory and Databases

A (finite) *relational vocabulary* is a finite set of *relation symbols* $R_1, R_2, \ldots$ Usually *function symbols* and *constant symbols* are included in a vocabulary but we do not allow them here. Every relation symbol is equipped with its *arity*, a natural number ≥ 1. We denote a vocabulary by $\tau, \sigma, \ldots$ For instance: $\sigma = \langle R_1, \ldots, R_s \rangle$ is a relational vocabulary with s relation symbols and their corresponding arities $r_1, \ldots, r_s$ respectively, for some $s \geq 1$.

A *structure* $\mathcal{A}$ of a relational vocabulary σ (concisely a σ-structure) consists of a non-empty set $dom(\mathcal{A})$ called the domain of $\mathcal{A}$, and a relation $R^{\mathcal{A}}$ over $dom(\mathcal{A})$ for every relation symbol R in σ of the corresponding arity. A relation $R^{\mathcal{A}}$ of arity n is a subset of $(dom(\mathcal{A}))^n$ (the set of n-tuples of elements of $dom(\mathcal{A})$). A structure $\mathcal{A}$ is finite if its domain is a finite set. The structures considered here will be always finite structures of relational vocabularies. We denote by $\mathcal{B}_\sigma$ the class of all finite structures of vocabulary σ.

Most of the examples of relational structures we use in this work are graphs, since they are the simplest relational structures and it is not difficult to visualize structural properties on them. For a better understanding in Graph Theory we refer the reader to [4] nevertheless, chapters 8 and 9 in [21] are sufficient to cover the graph concepts used in this work. A *graph* or *undirected graph* is a finite structure $\mathcal{G}$ of vocabulary $\sigma = \langle E \rangle$, where E is of arity 2 and satisfies the following conditions:

1. for all $a \in dom(\mathcal{G})$: not $E^{\mathcal{G}}(a, a)$
2. for all $a, b \in dom(\mathcal{G})$: if $E^{\mathcal{G}}(a, b)$ then $E^{\mathcal{G}}(b, a)$.

if only the first condition is required, then the graph is a *directed graph.* The elements of $dom(\mathcal{G})$ are usually named as *nodes* or *vertices* and the elements of $E^{\mathcal{G}}$ are called *edges.* Let n be a non-negative integer and $\mathcal{G}$ an undirected graph, then

$$E(a_0, a_1), E(a_1, a_2), \ldots, E(a_{n-1}, a_n)$$

is a *path of length n* from a_0 to a_n in $\mathcal{G}$. A graph is called *regular* if every vertex of this graph has the same degree. A regular graph is called *n-regular* if every vertex in this graph has degree n. An undirected graph is a *tree* if and only if there is a unique simple path between any two of its vertices. A *rooted tree* is a tree in which one vertex has been designated as the *root.* A *binary tree* is a rooted tree where every internal vertex has no more than two children. A binary tree is called a *complete binary tree* if every internal vertex has exactly two children and all the leaves in the tree have the same depth. A *colored graph* is a structure formed by a graph together with unary relations, i.e., $\mathcal{G} = \langle E, C_1, \ldots, C_r \rangle$ which may be thought of as representing colors on the vertices. Relations $C_1, \ldots, C_r$ encode a range of 2^r possible colors on the vertices. We always assume that each vertex satisfies exactly one color relation, that is, there are r colors instead of 2^r.

A k-tuple over a σ-structure $\mathcal{A}$ with $k \geq 1$, is a tuple of length k formed with elements from $dom(\mathcal{A})$, that is $(a_1, \ldots, a_k) \in (dom(\mathcal{A}))^k$. We will denote a k-tuple of $\mathcal{A}$ as $\bar{a}_k$, and also as $\bar{a}$.

Two σ-structures $\mathcal{A}$ and $\mathcal{B}$ are isomorphic if there exists a bijection from $dom(\mathcal{A})$ to $dom(\mathcal{B})$ that preserves all the relations in σ. An *automorphism* over a structure $\mathcal{A}$ is a bijection from $dom(\mathcal{A})$ to $dom(\mathcal{A})$ that preserves all the relations in σ.

As our theoretical framework will be Finite Model Theory we will regard a *logic* as a language, that is, as a set of formulas. We refer the reader to [16], [10] and, [17] for an in-depth study of Finite Model Theory. We will use the notion of a logic in a general sense, without a formal definition and we will apply this notion to a few concrete logics like First Order Logic (from now on denoted by FO), second order logic and infinitary logic.

As to the semantics of the logics, we use the classical Tarski semantics, except that since our theoretical framework is Finite Model Theory, only finite structures, or interpretations are considered. So that our structures will always be *finite relational structures*, that is, finite structures of a relational vocabulary.

If $\mathcal{L}$ is a logic, the notion of *satisfaction* of formulas, denoted as $\models_{\mathcal{L}}$, and *equivalence* between structures, denoted as $\equiv_{\mathcal{L}}$, will be related only to finite structures. We will use $\mathcal{L}_\sigma$ to denote the class of formulas from a logic $\mathcal{L}$ with vocabulary σ.

Definition 1. *Let $\mathcal{L}$ be a logic, σ a vocabulary and $\mathcal{A}$ a σ-structure. The $\mathcal{L}$ theory of $\mathcal{A}$, denoted $Th_{\mathcal{L}}(\mathcal{A})$, is the set of sentences from $\mathcal{L}_\sigma$ which are TRUE when interpreted by $\mathcal{A}$. In symbols*

$$Th_{\mathcal{L}}(\mathcal{A}) = \{\varphi \in \mathcal{L}_\sigma | \mathcal{A} \models_{\mathcal{L}} \varphi\}.$$

Embedding Database Theory in Finite Model Theory turns out to be quite natural. A *database schema* may be regarded as a relational vocabulary, and a *database instance* or simply a *database* of some schema

σ, as a finite relational structure of vocabulary σ. Thus, logics turn out to be languages which can be used as query languages, where the formulas of this logics express queries.

Since the definition of the Relational Model for databases, by E. F. Codd in 1970 [7], *FO* has been considered as one of the main models for query languages. Codd actually used a slight variation of *FO* called *Relational Calculus* where the domains of the databases are considered *infinite*. A sub-class of *FO* is defined, the *safe formulas*, as those formulas which express *safe queries*, that is, queries that always evaluate to *finite* relations which do not depend on the underlying infinite domain. As we consider databases as *finite* structures, we are not concerned about safety of queries here.

We will see now the use of logics as query languages. In particular, the formulas of first-order logic over a relational vocabulary $\sigma = \langle R_1, \ldots, R_s \rangle$ are formulas constructed from atomic formulas $R_i(x_1, \ldots, x_{r_i})$ and equality $x = y$, using a set of individual variables $x_1, x_2, \ldots$, standard connectives $\vee, \wedge, \neg, \rightarrow$ and, the quantifiers $\exists$, $\forall$.

A *free* variable in a formula is a variable that is not in the scope of a quantifier. Free variables of $\forall x \varphi$ or $\exists x \varphi$ are the free variables of φ except x. By $\varphi(x_1, \ldots, x_k)$ we denote an *FO* formula whose free variables are exactly $\{x_1, \ldots, x_k\}$. We denote by $free(\varphi)$ the set of free variables of the formula φ.

A *sentence* is an *FO* formula in which every variable is used in the scope of an existential or universal quantifier, that is, a formula without free variables.

If σ is a vocabulary, $\mathcal{A}$ is a σ-structure in $\mathcal{B}_\sigma$, $\varphi(x_1, \ldots, x_k)$ is an *FO* formula and $\bar{a} = (a_1, \ldots, a_k)$ is a k-tuple over $dom(\mathcal{A})$, we denote by

$$\mathcal{A} \models \varphi(x_1, \ldots, x_k)[a_1, \ldots, a_k]$$

or simply $\mathcal{A} \models \varphi[a_1, \ldots, a_k]$, the fact that φ is *TRUE*, when interpreted by the structure $\mathcal{A}$ under a *valuation* v, where $v(x_i) = a_i$ for $1 \leq i \leq k$. A valuation is a function which assigns to every variable in the logic an element of the structure $\mathcal{A}$. We will use the same notation even when the formula is a sentence. Then, we can consider the set of all such valuations, as follows:

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

That is, $\varphi^{\mathcal{A}}$ is the relation defined by φ in the structure $\mathcal{A}$ and its arity is given by the number of free variables in φ.

Suppose having a database of a specific airline where we can find information about flights between some cities of the world. We can think this database as a structure $\mathcal{A}$ of a given vocabulary σ with only one binary relation symbol E, i.e., $\sigma = \langle E \rangle$, such that given two cities, say a and b, by $E(a,b)$ we denote the existence of a flight from city a to b without stopovers. Suppose someone wants to know whether this airline brings a flight from a given city to another with at most one stopover. Using first-order logic we can construct a formula that expresses it as follows:

$$\varphi(x,y) := E(x,y) \vee \exists z \big(E(x,z) \wedge E(z,y) \big)$$

where $\varphi(x,y)$ is a formula in first-order logic with two free variables. Then $\varphi(x,y)$ is TRUE when it is interpreted in $\mathcal{A}$, for every pair of cities a and b where we can fly from a to b with at most one stopover. Then, $\varphi(x,y)$ is a formula in first-order logic that expresses a query over $\mathcal{A}$.

The following is an important remark concerning the expressive power of first-order logic in the finite.

Proposition 1. *([10], Proposition 2.1.1) Every finite structure can be characterized in first-order logic up to isomorphism, i.e., for every finite structure $\mathcal{A}$ of a given signature σ there is a sentence $\varphi_{\mathcal{A}}$ of first-order logic such that for all structures $\mathcal{B}$ in $\mathcal{B}_\sigma$ we have*

$$\mathcal{B} \models \varphi_{\mathcal{A}} \qquad \text{iff} \qquad \mathcal{A} \cong \mathcal{B}.$$

Proof. Suppose $dom(\mathcal{A}) = \{a_1, \dots, a_n\}$ and let $\bar{a} = (a_1, \dots, a_n)$. We define Ψ as the set of atomic formulas ψ with free variables among $(x_1, \dots, x_n)$ as follows:

$$\begin{aligned} \Psi_n = \{\psi | \psi \text{ has the form } R(x_{i_1}, \dots, x_{i_k}) \text{ where } 1 \le i_1, \dots, i_k \le n, \\ \text{and } R \in \sigma \text{ with arity } k \ge 1\} \end{aligned}$$

and

$$\begin{aligned} \varphi_{\mathcal{A}} \equiv \exists x_1 \dots \exists x_n \big(\bigwedge \{\psi | \psi \in \Psi_n, \mathcal{A} \models \psi[\bar{a}]\} \wedge \\ \bigwedge \{\neg\psi | \psi \in \Psi_n, \mathcal{A} \models \neg\psi[\bar{a}]\} \wedge \\ \forall x_{n+1} (x_{n+1} = x_1 \vee \dots \vee x_{n+1} = x_n) \big). \end{aligned}$$

□

Using first-order logic we can express an important class of database queries, however there are simple queries that cannot be expressed in FO , (like transitive closure, parity, etc.). The *Infinitary Logic* $\mathcal{L}_{\infty\omega}$ is more expressive than FO. Formulas in this logic are defined as in FO with the exception that disjunctions and conjunctions can be taken over arbitrary sets of formulas. If Ψ is a set of formulas of arbitrary cardinality in $\mathcal{L}_{\infty\omega}$ the infinitary conjunction (disjunction) over all formulas in Ψ, $\bigwedge\Psi$ ($\bigvee\Psi$ respectively) is also a formula in $\mathcal{L}_{\infty\omega}$. Connectivity in graphs is one of the queries that cannot be expressed in FO, however it is easily expressable using the following $\mathcal{L}_{\infty\omega}$ formula:

$$\forall x\forall y(\neg x = y \rightarrow \bigvee\{\varphi_n(x,y) | n \geq 1\})$$

where φ_n is an FO formula saying that there is a path from x to y of length n

$$\varphi_n(x,y) = \exists z_0 \ldots \exists z_n(z_0 = x \wedge z_n = y \wedge E(z_0,z_1) \wedge \cdots \wedge E(z_{n-1},z_n)).$$

By Proposition 1, every finite structure I can be characterized up to isomorphism by an FO sentence φ_I which, in general, needs $|dom(I)|+1$ variables. Hence, an arbitrary class K of finite structures closed under isomorphisms can be axiomatized in $\mathcal{L}_{\infty\omega}$ by the sentence

$$\bigvee\{\varphi_I | I \in K\}$$

which, in general, contains infinitely many variables. Since every class of finite structures is axiomatizable in it, $\mathcal{L}_{\infty\omega}$ is too powerful in the finite to characterize classes of finite structures, since even *non-recursive* classes of finite structures can be characterized in this logic.

We denote as FO^k, for any $k \geq 1$, the fragment of FO of the formulas whose free and bound variables are among $\{x_1, \ldots, x_k\}$. Clearly FO^k is less expressive than FO. Note that FO is the union of all FO^k logics, for every $k \geq 1$.

The infinitary logic $\mathcal{L}^k_{\infty\omega}$, for any $k \geq 1$ is the fragment of $\mathcal{L}_{\infty\omega}$ of the formulas whose free and bound variables are among $\{x_1, \ldots, x_k\}$. The infinitary logic $\mathcal{L}^\omega_{\infty\omega}$ is the union of all $\mathcal{L}^k_{\infty\omega}$. That is, formulas in $\mathcal{L}^\omega_{\infty\omega}$ have only *finitely many* different variables.

Note that, $\mathcal{L}_{\infty\omega}$ strictly includes $\mathcal{L}^\omega_{\infty\omega}$, that is, $\mathcal{L}_{\infty\omega} \supsetneq \mathcal{L}^\omega_{\infty\omega}$. Let $\mathcal{R}$ be the class of all regular graphs of vocabulary $\sigma = \langle E\rangle$, and let Φ be the $\mathcal{L}_{\infty\omega}$ sentence $\bigvee\{\varphi_n | n \geq 1\}$ where, φ_n is an FO sentence expressing that a given graph is an n-regular graph as follows

$$\varphi_n \equiv \forall x_1 \exists x_2 \ldots x_{n+1}\Big(E(x_1,x_2) \wedge \cdots \wedge E(x_1,x_{n+1}) \wedge$$
$$\forall x_{n+2}\big(E(x_1,x_{n+2}) \rightarrow$$

$$(x_{n+2} = x_2 \vee \cdots \vee x_{n+2} = x_{n+1}))\Big)$$

Then, the models of Φ are the regular graphs. Note that the sentence Φ belongs to $\mathcal{L}_{\infty\omega}$ but not to $\mathcal{L}^{\omega}_{\infty\omega}$, and it is well known that the class of regular graphs is not expressible in $\mathcal{L}^{\omega}_{\infty\omega}$.

The *quantifier rank* $qr(\varphi)$ of a formula φ in a given logic is the maximum number of nested quantifiers occurring in it, according to the following rules:

$qr(\varphi) := 0$, if φ is atomic; $\quad qr(\neg\varphi) := qr(\varphi)$;
$qr(\exists x\varphi) := qr(\varphi) + 1$; $\quad qr(\forall x\varphi) := qr(\varphi) + 1$;
$qr(\varphi \wedge \psi) := max\{qr(\varphi), qr(\psi)\}$; $\quad qr(\varphi \vee \psi) := max\{qr(\varphi), qr(\psi)\}$;
$qr(\varphi \rightarrow \psi) := max\{qr(\varphi), qr(\psi)\}$;

Given two structures $\mathcal{A}$ and $\mathcal{B}$ and $m \in \mathbb{N}$ we write $\mathcal{A} \equiv_m \mathcal{B}$ and say that $\mathcal{A}$ and $\mathcal{B}$ are *m-equivalent* if $\mathcal{A}$ and $\mathcal{B}$ satisfy the same FO sentences of quantifier rank $\leq m$.

Similarly, we say that $\mathcal{A}$ and $\mathcal{B}$ are *k-m-equivalent* if $\mathcal{A}$ satisfies the same FO^k sentences of quantifier rank $\leq m$ as $\mathcal{B}$, denoted $\mathcal{A} \equiv^k_m \mathcal{B}$.

We use $\mathcal{A} \equiv^{\mathcal{L}_{\infty\omega}} \mathcal{B}$ to say that $\mathcal{A}$ and $\mathcal{B}$ are $\mathcal{L}_{\infty\omega}$ *equivalent* i.e., they satisfy the same $\mathcal{L}_{\infty\omega}$ sentences. Similarly, by $\mathcal{A} \equiv^{\mathcal{L}^k_{\infty\omega}} \mathcal{B}$ we denote that $\mathcal{A}$ and $\mathcal{B}$ are $\mathcal{L}^k_{\infty\omega}$ *equivalent* i.e., they satisfy the same $\mathcal{L}^k_{\infty\omega}$ sentences.

2.2 Computable Queries

Following [5] we define the formal notion of computable query.

Definition 2. *For a given vocabulary σ and an integer $r > 0$ we define a* computable query of arity r and vocabulary σ, *as a partial function $q : \mathcal{B}_\sigma \rightarrow \mathcal{B}_{\langle R \rangle}$ where R is a relation symbol of arity r, and q has the following properties:*

i For every database $\mathcal{A}$ of vocabulary σ where q is defined, $dom(q(\mathcal{A})) \subseteq dom(\mathcal{A})$.

ii q is a partial recursive function in some linear encoding of the σ-structures.

iii q preserves isomorphisms, i.e., for every pair of databases of vocabulary σ, $\mathcal{A}$ and $\mathcal{B}$, and for every isomorphism $h : dom(\mathcal{A}) \rightarrow dom(\mathcal{B})$, either $q(\mathcal{B}) = h(q(\mathcal{A}))$, or q is undefined on both $\mathcal{A}$ and $\mathcal{B}$.

Definition 3. *For a given vocabulary σ we define a* computable Boolean query of vocabulary σ, *as a partial function $q : \mathcal{B}_\sigma \to \{0,1\}$, with the following properties:*

i q is a partial recursive function in some linear encoding of the σ-structures.

ii q preserves isomorphisms, i.e., for every pair of databases of vocabulary σ, $\mathcal{A}$ and $\mathcal{B}$ which are isomorphic, either $q(\mathcal{B}) = q(\mathcal{A})$, or q is undefined on both $\mathcal{A}$ and $\mathcal{B}$.

Boolean queries may also be regarded as 0-ary queries. We denote the class of computable queries of vocabulary σ as $\mathcal{CQ}_\sigma$ and, $\mathcal{CQ}$ denotes the whole class of computable queries of all vocabularies.

There are many formalisms used to compute or express computable queries, like formal programming languages, abstract machines and, logics. Here, we will regard a *language* as a logic because we will use logics rather than other formalisms. We say that a language is *complete* if it expresses the whole class $\mathcal{CQ}$ of computable queries.

3 Type of a Tuple

Given a σ-structure $\mathcal{A}$ and a k-tuple $\bar{a} = (a_1, \ldots, a_k)$ over $(dom(\mathcal{A}))^k$, we would like to consider now all properties of the tuple, regarding the relations in $\mathcal{A}$. One of those properties would be, for instance, whether $\bar{a}$ belongs to a k-ary relation R in $\mathcal{A}$. But, we are interested also in the properties of the sub-tuples of $\bar{a}$. For instance, we want to know whether a sub-tuple (of length $l < k$) belongs to a tuple in a relation R in $\mathcal{A}$ of arity l. Even, we are interested in the properties of each component of $\bar{a}$, for instance, given two components of $\bar{a}$, a_1 and a_2, we would like to consider whether a_1 and a_2 have some common properties because of their position in different tuples of a relation R in $\mathcal{A}$.

There are many properties we can express regarding the components of $\bar{a}$, its sub-tuples, $\bar{a}$ itself, and the relations in $\mathcal{A}$. We will use a given logic $\mathcal{L}$ to express them and analyze what sort of properties will be expressible using that particular logic.

There is a formal concept in Model Theory that is used for the purpose of studying those properties, the concept of *type*. In [8] and [18] the concept of type is studied in depth.

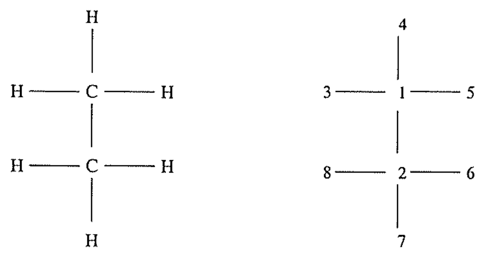
Definition 4. *([8], Definition 2.17) Given a logic $\mathcal{L}$, a structure $\mathcal{A}$ of vocabulary σ, a k-tuple $\bar{a} = (a_1, \ldots, a_k)$ over $(dom(\mathcal{A}))^k$ for some $k > 0$, we define the $\mathcal{L}$ type of $\bar{a}$ in $\mathcal{A}$ as follows:*

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

That 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 $\mathcal{A}$ for every valuation which assigns the i-th element of $\bar{a}$ to the variable x_i for every $1 \leq i \leq k$.

Let's consider an example.

Example 1. In chemistry, graphs are used to represent molecules, where atoms are represented by vertices and bonds between them by edges. For instance, the graph in Figure 1(a) represents the molecule of ethane. Clearly, we can see this graph as a structure of vocabulary $\sigma = \langle E, H, C \rangle$ where E is a binary relation symbol which may be interpreted as bonds between atoms, and H and C are unary relations symbols which may be thought of as representing atoms of hydrogen and carbon respectively. Let $\mathcal{G}$ be a σ-structure where $dom(\mathcal{G}) = \{1, 2, 3, 4, 5, 6, 7, 8\}$, $H^{\mathcal{G}} = \{3, 4, 5, 6, 7, 8\}$, $C^{\mathcal{G}} = \{1, 2\}$ and $E^{\mathcal{G}}$ is as shown in Figure 1(b).



(a) Molecule of Ethane (b) Graph G representing the molecule of ethane

Figure 1

Now, let's focus on the FO properties of the atoms of $\mathcal{G}$, i.e., tuples where $k = 1$. It is straightforward to note that all hydrogen atoms have the same FO type, as each of them is connected to exactly one carbon atom. Let p be a function from $dom(\mathcal{G})$ to $dom(\mathcal{G})$ where,

$$\begin{array}{llll} p(1) = 2 & p(2) = 1 & p(3) = 6 & p(4) = 7 \\ p(5) = 8 & p(6) = 3 & p(7) = 4 & p(8) = 5 \end{array}$$

then, p is an automorphism of $\mathcal{G}$ as it preserves all relations in $\mathcal{G}$, i.e., for every $x, y \in dom(\mathcal{G})$,

- $E(x, y)$ iff $E(p(x), p(y))$,
- $C(x)$ iff $C(p(x))$, and
- $H(x)$ iff $H(p(x))$.

In fact, each pair of hydrogen atoms can be exchanged by some automorphism of $\mathcal{G}$. Moreover, carbon atoms have the same FO type.
Note that, if we consider the types of pairs of elements on $\mathcal{G}$, it might be the case that they have a different FO type even though their corresponding elements have the same FO type. Let's for instance choose two pairs of atoms where the first element of both pairs is the same carbon atom, say 1, the second element of the first pair is an hydrogen atom which is adjacent to the first element of its pair, say 3, and, the second element of the second pair is an hydrogen atom which is not adjacent to the first element of its pair, say 8. These tuples in $\mathcal{G}$ are $(1, 3)$ and $(1, 8)$, and clearly, these tuples do not have the same FO type, therefore they cannot be exchanged by any automorphism of $\mathcal{G}$.

Note that, the type of a k-tuple is an *infinite* set of formulas. This set is *maximally consistent* since there is a structure and a valuation that make all the formulas in this set TRUE. We can then think of the type of a tuple without having a structure in mind, by just adding formulas with the appropriate free variables to a set as long as it remains consistent. That is, we may define the type of a given vocabulary and for a given length of tuple, and then ask ourselves whether in a given database there is a tuple of that length that has that type.

Definition 5. *Let $\mathcal{L}$ be a logic, let σ be a relational vocabulary and $k \geq 0$, we denote by*

$$Tp^{\mathcal{L}}(\sigma, k) = \{tp^{\mathcal{L}}_{\mathcal{A}}(\bar{u}) : \mathcal{A} \in \mathcal{B}_\sigma \wedge \bar{u} \in (dom(\mathcal{A}))^k\}$$

the class of all $\mathcal{L}$ types for k-tuples over structures of vocabulary σ.
Let α be the $\mathcal{L}$ type of some k-tuple over some structure in $\mathcal{B}_\sigma$, i.e., $\alpha \in Tp^{\mathcal{L}}(\sigma, k)$. If $\mathcal{A}$ is a structure in $\mathcal{B}_\sigma$, we say that $\mathcal{A}$ realizes *the type α if there is a k-tuple $\bar{u}$ over $(dom(\mathcal{A}))^k$ whose $\mathcal{L}$ type is α, that is, $tp^{\mathcal{L}}_{\mathcal{A}}(\bar{u}) = \alpha$.*

Definition 6. *Let $\mathcal{L}$ be a logic, let σ be a relational vocabulary, let $\mathcal{A}$ be a structure in $\mathcal{B}_\sigma$, let $\bar{u}$ be a k-tuple in $(dom(\mathcal{A}))^k$ and $k \geq 0$, we denote by*

$$Tp^{\mathcal{L}}(\mathcal{A}, k) = \{tp^{\mathcal{L}}_{\mathcal{A}}(\bar{u}) : \bar{u}_k \in (dom(\mathcal{A}))^k\}$$

the class of all $\mathcal{L}$ types for k-tuples which are realized in $\mathcal{A}$.

3.1 The Notion of *FO* Type in Databases

In the perspective of databases, the *FO* type of a k-tuple $\bar{u}$ in $\mathcal{A}$ is the set of all *queries* of arity $\leq k$ which are expressible in *FO* and such that the corresponding sub-tuple of $\bar{u}$ is in the answer of the respective query when evaluated on the database $\mathcal{A}$.

Example 2. Consider the complete binary tree $\mathcal{A}$ as shown in Figure 2 of vocabulary $\sigma = \langle E \rangle$ where E is the edge relation symbol.

Let's consider the following *FO* formulas:

$$\varphi_1 \equiv \exists z, x(\neg\exists y(E(y,z)) \wedge E(z,x))$$
$$\varphi_2(x) \equiv \exists z(\neg\exists y(E(y,z)) \wedge E(z,x))$$
$$\varphi_3(x,y) \equiv \exists z(\neg\exists w(E(w,z)) \wedge E(z,x) \wedge E(z,y))$$

those are some of the formulas that are TRUE when evaluated on $\mathcal{A}$ on the pair (b, c), and hence belong to the *FO* type of the tuple (b, c) in $\mathcal{A}$, i.e.,

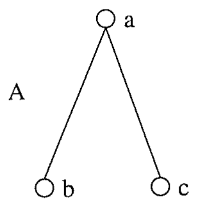


Figure 2

$$tp^{FO}_{\mathcal{A}}(b,c) = \{\varphi_1, \varphi_2(x), \varphi_3(x,y), \dots\}$$

Thus, the FO type of every tuple in a given database includes not only the properties of all its sub-tuples, but also the properties of the database itself, in the sense that it includes the FO theory of the database (see Definition 1).

Regarding tuples, by Proposition 1 first-order logic is sufficiently expressive as to include all the properties which might make a k-tuple $\bar{u}$ distinguishable from other k-tuples in a given database $\mathcal{A}$ of schema σ, and even in the whole class $\mathcal{B}_\sigma$ of all databases of schema σ. That is, for all $k \geq 0$, structures $\mathcal{A}, \mathcal{B}$ in $\mathcal{B}_\sigma$, and k-tuples $\bar{u} \in (dom(\mathcal{A}))^k$, $\bar{v} \in (dom(\mathcal{B}))^k$ $tp^{FO}_{\mathcal{A}}(\bar{u}) = tp^{FO}_{\mathcal{B}}(\bar{v})$ if and only if there is an isomorphism $f : dom(\mathcal{A}) \rightarrow dom(\mathcal{B})$ such that for $1 \leq i \leq k$ it is $v_i = f(u_i)$.

We will see the relationship between queries and the FO type of tuples in databases with more detail in Subsections 7.1 and 7.2 where we relate queries with the Isolating Formulas of the FO types of k-tuples, in databases and in classes of databases, respectively.

4 Partial Isomorphisms

In this section we explain in detail the technical concept of partial isomorphism which plays an important role in this work.

Definition 7. *Let $\mathcal{A}$ and $\mathcal{B}$ be two structures of a relational vocabulary σ and let p be a function with $do(p) \subseteq dom(\mathcal{A})$ and $rg(p) \subseteq dom(\mathcal{B})$, where $do(p)$ and $rg(p)$ denote the domain and range of p, respectively. Then, p is a* partial isomorphism *from $\mathcal{A}$ to $\mathcal{B}$ if*

- *p is injective, and*
- *for each n-ary relation symbol $R \in \sigma$, with $n \geq 1$ and for all n-tuples $(a_1, \dots, a_n) \in (do(p))^n$,*

$$R^{\mathcal{A}}(a_1, \dots, a_n) \quad \textit{iff} \quad R^{\mathcal{B}}(p(a_1), \dots, p(a_n))$$

We will identify a mapping p by its graph, i.e., $p = \{(a, p(a)) | a \in do(p)\}$. Given two partial isomorphisms p and q, if $p \subseteq q$, we say that q is an *extension* of p. Note that the empty map ($p = \emptyset$) is also a partial isomorphism.

Lemma 1. *([10], Remark 2.2.2 (c)) Let σ be a relational vocabulary, let $\mathcal{A}$ and $\mathcal{B}$ be two structures in $\mathcal{B}_\sigma$, let $\bar{a}$ be a k-tuple in $(dom(\mathcal{A}))^k$, let $\bar{b}$ be a k-tuple in $(dom(\mathcal{B}))^k$, then the following conditions are equivalent:*

- *$p(a_i) = b_i$, for all $1 \leq i \leq k$, defines a mapping, which is a partial isomorphism from $\mathcal{A}$ to $\mathcal{B}$,*
- *for all* quantifier free formula *$\varphi(x_1, \dots, x_k) : \mathcal{A} \models \varphi[\bar{a}]$ iff $\mathcal{B} \models \varphi[\bar{b}]$ and,*
- *for all* atomic formula *$\varphi(x_1, \dots, x_k) : \mathcal{A} \models \varphi[\bar{a}]$ iff $\mathcal{B} \models \varphi[\bar{b}]$.*

Note that a partial isomorphism does not preserve the validity of formulas with quantifiers. Let's see an example. Suppose we have two structures $\mathcal{A} = (\{0, 1, 2\}, <)$ and $\mathcal{B} = (\{0, 1, 2, 3\}, <)$ of vocabulary $\sigma = \langle < \rangle$, where $<$ denotes the natural ordering. Then, $p_0 = \{(0, 0), (2, 1)\}$ is a partial isomorphism from $\mathcal{A}$ to $\mathcal{B}$ such that:

$$\mathcal{A} \models \exists x_3 (x_1 < x_3 \wedge x_3 < x_2)[0, 2]$$

but,

$$\mathcal{B} \not\models \exists x_3 (x_1 < x_3 \wedge x_3 < x_2)[p_0(0), p_0(2)]$$

however, the validity of

$$\mathcal{B} \models \exists x_3(x_1 < x_3 \wedge x_3 < x_2)[p(0), p(2)],$$

for any partial isomorphism p from $\mathcal{A}$ to $\mathcal{B}$ with $do(p) = \{0, 2\}$, can be determined by the existence or not of some partial isomorphism q such that $p \subseteq q$ and $1 \in do(q)$. Take for instance the partial isomorphism p_0, it is clear that such partial isomorphism q which extends p_0 and has 1 in its domain does not exist. On the other hand, consider the partial isomorphism $r = \{(0, 0), (2, 3)\}$. In this case, we do have a partial isomorphism q such that $r \subseteq q$ and $1 \in do(q)$, take for instance $q = \{(0, 0), (2, 3), (1, 1)\}$ and then,

$$\mathcal{B} \models \exists x_3(x_1 < x_3 \wedge x_3 < x_2)[r(0), r(2)].$$

this suggests that the truth of formulas with quantifiers could be preserved by partial isomorphisms if they admit certain extensions.

We will see now that the m-equivalence of structures is related with this, in the sense that a partial isomorphism can be extended m times and structures $\mathcal{A}$ and $\mathcal{B}$ still satisfy the same formulas of quantifier rank $\leq m$. Later we will define back and forth systems as *sequences of sets of partial isomorphisms* $I_m, \ldots, I_0$ from a given structure $\mathcal{A}$ to $\mathcal{B}$ where each I_j, for some $0 \leq j < m$, is a non-empty set. Roughly, a partial isomorphism p in I_{j+1} is extended with an element $a \in dom(\mathcal{A})$ ($b \in dom(\mathcal{B})$ respectively) if there is a partial isomorphism q in I_j such that $p \subseteq q$ and $a \in do(q)$ ($b \in rg(q)$ respectively).

5 Ehrenfeucht-Fraïssé Games

The concept of Ehrenfeucht-Fraïssé games is an interesting way to study the characterization of equivalence between structures. The Ehrenfeucht-Fraïssé method can be considered from three different perspectives, gametheoretic [12], algebraic [13] and logical. The method is used to prove that certain properties on finite structures cannot be expressed in some logics. So, the Ehrenfeucht-Fraïssé games constitute an important, and quite interesting method used to show the expressibility of different logics related to their capability of distinguishing finite structures (recall that in this work we don't consider infinite structures). We shall see that the notion of type plays a relevant role in these characterizations.

The Ehrenfeucht-Fraïssé games, are played by two players called the *spoiler* and the *duplicator*. They play over two structures $\mathcal{A}$ and $\mathcal{B}$ of the same vocabulary σ, a k-tuple over each structure, say $\bar{u} \in (dom(\mathcal{A}))^k$ and $\bar{v} \in (dom(\mathcal{B}))^k$, and a given $m \in \mathbb{N}$. Each player has to make m *moves* during the game. They take turns. The spoiler will try to find a difference between $\mathcal{A}$ and $\mathcal{B}$, and the duplicator will make moves trying to hide any possible difference between the two structures.

The spoiler starts the game. In a given i-th move, for any $1 \leq i \leq m$, the spoiler selects a structure, either $\mathcal{A}$ or $\mathcal{B}$, and an i-th element in it. Then the duplicator makes his i-th move where, if the spoiler had chosen the structure $\mathcal{A}$ and element a_i, the duplicator has to choose the structure $\mathcal{B}$ and some element b_i in $dom(\mathcal{B})$. If the spoiler had chosen the structure $\mathcal{B}$ and an element in it in his i-th move, the duplicator has to choose the structure $\mathcal{A}$ and an element in $dom(\mathcal{A})$. Given the elements chosen during the game in both structures, the pairs $\{(u_1, v_1), \ldots, (u_k, v_k), (a_1, b_1), \ldots, (a_m, b_m)\}$ determine the result of the Ehrenfeucht-Fraïssé game denoted $G_m(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$ for m moves on k-tuples $\bar{u}$ and $\bar{v}$ over structures $\mathcal{A}$ and $\mathcal{B}$ respectively. Note that, elements of the k-tuples $\bar{u}$ and $\bar{v}$ are elements of the domain of structures $\mathcal{A}$ and $\mathcal{B}$ respectively and can then be chosen by the players during the game. Moreover, the same given element of a structure may be chosen in more than one move.

Informally, to understand whether the spoiler or the duplicator wins the game we will use the concept of partial isomorphism. If after m moves the spoiler never could find a difference between structures $\mathcal{A}$ and $\mathcal{B}$ and in every i-th move the duplicator could find an element (on $\mathcal{A}$ or $\mathcal{B}$ according to the move) in order to show that both structures remain "similar", i.e. partially isomorphic, we say that the duplicator wins the game. That is, the duplicator *wins* the game $G_m(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$ if after m moves there exists a partial isomorphism p from $\mathcal{A}$ to $\mathcal{B}$ such that $p = \{(u_1, v_1), \ldots, (u_k, v_k), (a_1, b_1), \ldots, (a_m, b_m)\}$. Otherwise, the spoiler wins the game.

We say that a player (the duplicator or the spoiler) has a *winning strategy* in $G_m(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$ if it is possible for him to win each play of the game whatever choice is made by the opponent.

Given two structures $\mathcal{A}$ and $\mathcal{B}$ we say that the tuples $\bar{u}$ and $\bar{v}$ are *not distinguishable* by any FO formula of quantifier rank $\leq m$ if for every formula $\varphi(x_1, \ldots, x_k)$ in FO of quantifier rank $\leq m$ $\mathcal{A} \models \varphi[\bar{u}]$ if and only if $\mathcal{B} \models \varphi[\bar{v}]$. If $\mathcal{A}$ and $\mathcal{B}$ satisfy the same FO sentences of quantifier rank $\leq m$ we say that $\mathcal{A}$ and $\mathcal{B}$ are m-equivalent denoted $\mathcal{A} \equiv_m \mathcal{B}$.

Example 3. Take for instance the following two σ-structures (undirected and colored graphs) $\mathcal{A}$ and $\mathcal{B}$, where $\sigma = \langle E, R, B \rangle$ with a binary relation symbol E and two unary relation symbols R and B that denote two different colors of the nodes of the graphs.

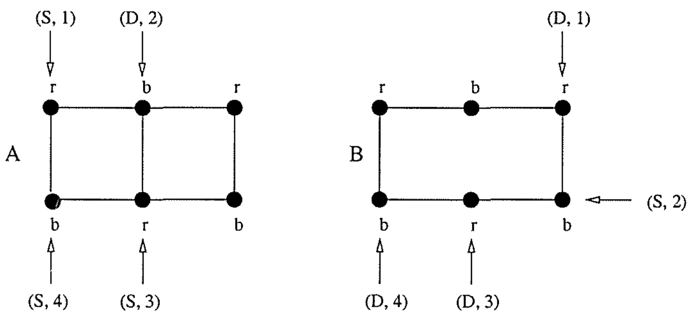


Figure 3

The spoiler starts the game $G_m(\mathcal{A}, \mathcal{B})$ choosing a red node of $\mathcal{A}$ (denoted by (S, 1) in Figure 3). The duplicator responds choosing a red node in $\mathcal{B}$ (denoted by (D, 1)). In the second move the spoiler chooses an adjacent blue node to the previous red one in $\mathcal{B}$, and the duplicator responds in $\mathcal{A}$. In the third move, the spoiler chooses a red node adjacent to the previous blue node in $\mathcal{A}$, and the duplicator responds in $\mathcal{B}$. In next move, the spoiler chooses a blue node adjacent to the previous node chosen in $\mathcal{A}$ and the duplicator answers with a blue node adjacent to the previous red node chosen in $\mathcal{B}$.

We see that the spoiler has a winning strategy for the game $G_4(\mathcal{A}, \mathcal{B})$ choosing four adjacent nodes which form a square in the $\mathcal{A}$. The duplicator has no option because there is not a square in $\mathcal{B}$ and therefore the spoiler wins. Note that the winning strategy for the spoiler is trying to satisfy the following sentence:

$$\varphi \equiv \exists x_1, x_2, x_3, x_4(E(x_1,x_2) \wedge E(x_2,x_3) \wedge E(x_3,x_4) \wedge E(x_4,x_1))$$

that expresses that there exists a square, i.e., a cycle of length 4.

Note that in the preceding example the tuples $\bar{u}$ and $\bar{v}$ were empty. We use $G_m(\mathcal{A},\mathcal{B})$ to denote that the k-tuples $\bar{u}$ in $\mathcal{A}$ and $\bar{v}$ in $\mathcal{B}$ are empty in the Ehrenfeucht-Fraïssé game, i.e., $k = 0$. It means that the game is played on the structures $\mathcal{A}$ and $\mathcal{B}$ without considering any particular tuple together with them. We will see later that if the duplicator wins the game $G_m(\mathcal{A},\mathcal{B})$, the structures $\mathcal{A}$ and $\mathcal{B}$ are m-equivalent, that is, $\mathcal{A}$ and $\mathcal{B}$ satisfy the same FO sentences of quantifier rank $\leq m$. That is, every sentence φ of quantifier rank $\leq m$ is satisfied by $\mathcal{A}$ if and only if it is satisfied by $\mathcal{B}$, i.e., $\mathcal{A} \models \varphi$ iff $\mathcal{B} \models \varphi$. If $k > 0$, the game is denoted by $G_m(\mathcal{A},\bar{u},\mathcal{B},\bar{v})$. It is played on the structures $\mathcal{A}$ and $\mathcal{B}$ with two k-tuples $\bar{u}$ and $\bar{v}$ respectively. If the duplicator wins this game we say that the tuple $\bar{u}$ satisfies in $\mathcal{A}$ the same formulas of quantifier rank $\leq m$ with free variables among $(x_1,\ldots,x_k)$ as the tuple $\bar{v}$ in $\mathcal{B}$. Those statements will be shown later in the points (i) and (iv) of the Corollary 1 and Theorem 1 respectively.

From the standpoint of databases and queries, Corollary 1 can be expressed as follows. Every boolean query q expressible in FO restricted to formulas of quantifier rank $\leq m$ has the same answer (0 or 1) when it is evaluated on $\mathcal{A}$ and $\mathcal{B}$. Considering Theorem 1 we say that for every query q of arity k expressible in FO restricted to formulas of quantifier rank $\leq m$ the tuple $\bar{u}$ will be in the answer of q when it is evaluated on $\mathcal{A}$ if and only if the tuple $\bar{v}$ is in the answer of the same query q when it is evaluated on $\mathcal{B}$.

5.1 Inductive construction of the m-isomorphic type

Let $\mathcal{A}$ be a σ-structure and $\bar{u} = (u_1,\ldots,u_k) \in (dom(\mathcal{A}))^k$ for some $k > 0$, and let $m \geq 0$. Following [10] we introduce the formula $\varphi^m_{\bar{u}}(x_1,\ldots,x_k)$ that in a way describes the Ehrenfeucht-Fraïssé game $G_m(\mathcal{A},\bar{u},\mathcal{B},\bar{v})$ for an arbitrary σ-structure $\mathcal{B}$ and a k-tuple $\bar{v}$ in $(dom(\mathcal{B}))^k$. If the structure $\mathcal{A}$ is not clear from the context we use the notation $\varphi^m_{\mathcal{A},\bar{u}}$ for $\varphi^m_{\bar{u}}$ and when $k = 0$ we write $\varphi^m_{\mathcal{A}}$ for the sentence $\varphi^m_{\mathcal{A},\emptyset}$. We define $\varphi^m_{\bar{u}}$ in such a way that for any structure $\mathcal{B}$ and $\bar{v} = (v_1,\ldots,v_k) \in (dom(\mathcal{B}))^k$ the duplicator has a winning strategy for the game $G_m(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$ in m moves if and only if $\mathcal{B} \models \varphi^m_{\bar{u}}(x_1, \ldots, x_k)[\bar{v}]$.

Let $\bar{x} = (x_1, \ldots, x_k)$

$$\varphi^0_{\bar{u}}(\bar{x}) := \bigwedge\{\varphi(x_1, \ldots, x_k) | \varphi \text{ atomic or negated atomic, } \mathcal{A} \models \varphi[\bar{u}]\}$$

for $m > 0$

$$\varphi^m_{\bar{u}}(\bar{x}) = \bigwedge_{a \in dom(\mathcal{A})} \exists x_{k+1} \varphi^{m-1}_{\bar{u}a}(\bar{x}, x_{k+1}) \wedge \forall_{x_{k+1}} \bigvee_{a \in dom(\mathcal{A})} \varphi^{m-1}_{\bar{u}a}(\bar{x}, x_{k+1}).$$

Where $\varphi^0_{\bar{u}}$ describes the isomorphism type of the substructure induced by $\bar{u} \in (dom(\mathcal{A}))^k$ in $\mathcal{A}$, and also by $\bar{v} \in (dom(\mathcal{B}))^k$ in $\mathcal{B}$ for any structure $\mathcal{B}$ such that $\mathcal{B} \models \varphi^0_{\bar{u}}[\bar{v}]$, if and only if there is a partial isomorphism $\{(u_1, v_1), \ldots, (u_k, v_k)\}$ from $\mathcal{A}$ to $\mathcal{B}$. That is, if and only if $(\mathcal{B}, \bar{v})$ satisfies the conjunction of all atomic and negated atomic formulas in $\varphi^0_{\bar{u}}$ with free variables among $(x_1, \ldots, x_k)$, where obviously $\mathcal{A} \models \varphi^0_{\bar{u}}[\bar{u}]$.

For $m > 0$ the formula describes to which isomorphism type the tuple $\bar{u}$ (and therefore $\bar{v}$) will be extended in m moves adding at most one element in each move.

We will use the concept of type of a tuple to describe the formula $\varphi^m_{\bar{u}}$. Each new element $a \in dom(\mathcal{A})$ extends the tuple $\bar{u}$ of length k to a tuple of length $k+1$ denoted $\bar{u}a$, i.e. $\bar{u}a = (u_1, \ldots, u_k, a)$.

We give now an explanation of the intuition behind $\varphi^m_{\bar{u}}$.

Given a structure $\mathcal{B}$ and a k-tuple $\bar{v}$ in $\mathcal{B}$, what does it mean that $\mathcal{B} \models \varphi^m_{\bar{u}}[\bar{v}]$? Let's consider the first part of $\varphi^m_{\bar{u}}$, the big conjunction. For each element $a \in dom(\mathcal{A})$ there is an element $b \in dom(\mathcal{B})$ such that the tuple $\bar{v}b$ in the structure $\mathcal{B}$ satisfies the same FO formulas of quantifier rank $\leq m-1$ than the tuple $\bar{u}a$ in structure $\mathcal{A}$, i.e. $\mathcal{B} \models \varphi^{m-1}_{\bar{u}a}[\bar{v}b]$. That is, the FO type of quantifier rank $m-1$ of the tuple $\bar{u}a$ of length $k+1$ in $\mathcal{A}$ is the same as the FO type of quantifier rank $\leq m-1$ of the tuple $\bar{v}b$ of length $k+1$ in $\mathcal{B}$.

The second part of the formula, the universal quantifier followed by the big disjunction, describes that, for every $b \in dom(\mathcal{B})$ there is an $a \in dom(\mathcal{A})$ such that the FO type of quantifier rank $m-1$ of $\bar{u}a$ in $\mathcal{A}$ is the same as the FO type of quantifier rank $m-1$ of $\bar{v}b$ in $\mathcal{B}$. Then, $\bar{u}a$ in $\mathcal{A}$ and $\bar{v}b$ in $\mathcal{B}$ satisfy the same FO formulas of quantifier rank $\leq m-1$ (clearly $qr(\varphi^m_{\bar{a}}) = m$).

This formula can be described in terms of types of tuples in FO logic but also in terms of Ehrenfeucht-Fraïssé Games. The duplicator wins the game $G_0(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$ if there is a partial isomorphism $\{(u_1, v_1), \ldots, (u_k, v_k)\}$

from $\mathcal{A}$ to $\mathcal{B}$, that is, if for all atomic formulas $\varphi(\bar{x})$, $\mathcal{A} \models \varphi[\bar{u}]$ if and only if $\mathcal{B} \models \varphi[\bar{v}]$. For $m > 0$, the duplicator wins the game $G_m(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$ if for all $a \in dom(\mathcal{A})$ that the spoiler might choose there is a $b \in dom(\mathcal{B})$ such that the duplicator wins the game $G_{m-1}(\mathcal{A}, \bar{u}a, \mathcal{B}, \bar{v}b)$. Also, for all $b \in dom(\mathcal{B})$ that the spoiler might choose, there is an $a \in dom(\mathcal{A})$ such that the duplicator wins the game $G_{m-1}(\mathcal{A}, \bar{u}a, \mathcal{B}, \bar{v}b)$. This means that there is a sequence of sets of partial isomorphisms $I_m, \ldots, I_0$ such that in each move of the game, say j where $1 \leq j < m$, every partial isomorphism p in the set I_{j+1} can be extended to a partial isomorphism q in the set I_j, that is, $p \subseteq q$, with every element $a \in dom(\mathcal{A})$ and $b \in dom(\mathcal{B})$ and in each direction, that is, for every element a in $dom(\mathcal{A})$ there is an element b in $dom(\mathcal{B})$ and for every element b in $dom(\mathcal{B})$ there is an element a in $dom(\mathcal{A})$ such that $a \in do(q)$ and $b \in rg(q)$. This is the concept of back and forth properties, introduced in the following definition.

Definition 8. *([10], Definition 2.3.1) Let $m \geq 0$. Two structures $\mathcal{A}$ and $\mathcal{B}$ are m-isomorphic, denoted $\mathcal{A} \cong_m \mathcal{B}$ if there is a non-empty sequence of sets of partial isomorphisms from $\mathcal{A}$ to $\mathcal{B}$ denoted $(I_j)_{j \leq m}$ with forth and back properties as follows:*

* *Forth property: For every $0 \leq j < m$, $p \in I_{j+1}$ and $a \in dom(\mathcal{A})$ there is a $q \in I_j$ such that $q \supseteq p$ and $a \in do(q)$,*
* *Back property: For every $0 \leq j < m$, $p \in I_{j+1}$ and $b \in dom(\mathcal{B})$ there is a $q \in I_j$ such that $q \supseteq p$ and $b \in rg(q)$.*

the fact that $\mathcal{A}$ and $\mathcal{B}$ are m-isomorphic via $(I_j)_{j \leq m}$, is denoted as $(I_j)_{j \leq m} : \mathcal{A} \cong_m \mathcal{B}$.

Theorem 1. *([10]) Given two structures $\mathcal{A}$ and $\mathcal{B}$ of the same vocabulary, two k-tuples $\bar{u} \in (dom(\mathcal{A}))^k$ and $\bar{v} \in (dom(\mathcal{B}))^k$ and $m \geq 0$, the following are equivalent:*

(i) *the duplicator wins $G_m(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$.*
(ii) *$\mathcal{B} \models \varphi^m_{\bar{u}}[\bar{v}]$.*
(iii) *there is a sequence of sets of partial isomorphisms $(I_j)_{j \leq m}$ from $\mathcal{A}$ to $\mathcal{B}$, with $\bar{u} \mapsto \bar{v} \in I_m$ such that $(I_j)_{j \leq m} : \mathcal{A} \cong_m \mathcal{B}$.*
(iv) *$\bar{u}$ in $\mathcal{A}$ and $\bar{v}$ in $\mathcal{B}$ satisfy the same FO formulas of quantifier rank $\leq m$, that is, if $\varphi(x_1, \ldots, x_k)$ is of quantifier rank $\leq m$, then*

$$\mathcal{A} \models \varphi[\bar{u}] \quad \textit{iff} \quad \mathcal{B} \models \varphi[\bar{v}].$$

Proof.
For $(iii) \Rightarrow (i)$. Suppose that $\mathcal{A}$ and $\mathcal{B}$ are m-isomorphic via $(I_j)_{j \leq m}$, the

set of sets of partial isomorphisms from $\mathcal{A}$ to $\mathcal{B}$, i.e., $(I_j)_{j\leq m} : \mathcal{A} \cong_m \mathcal{B}$, and every I_j is non-empty and $\bar{u} \mapsto \bar{v} \in I_m$. The winning strategy in the game $G_m(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$ for the duplicator is as follows: In his i-th move he should choose the element $a_i \in dom(\mathcal{A})$ ($b_i \in dom(\mathcal{B})$) such that for the partial isomorphism $p_i = \{(u_1, v_1), \ldots, (u_k, v_k), (a_1, b_1), \ldots, (a_i, b_i)\}$ it is true that $p_i \subseteq q$ for some $q \in I_{m-i}$, this is always possible because of the forth and back properties of $(I_j)_{j\leq m}$. Then, when $i = m$, the duplicator wins the game $G_m(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$.

(iv) implies (ii) since $\bar{u}$ and $\bar{v}$ satisfy the same formulas of quantifier rank $\leq m$, and $qr(\varphi^m_{\bar{u}}) = m$, and $\mathcal{A} \models \varphi^m_{\bar{u}}[\bar{u}]$ then $\mathcal{B} \models \varphi^m_{\bar{u}}[\bar{v}]$.

$(i) \Leftrightarrow (ii)$. This proof is by induction on m.
For $m = 0$, the duplicator wins the game $G_0(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$ iff there is a partial isomorphism $\bar{u} \mapsto \bar{v}$ from $\mathcal{A}$ to $\mathcal{B}$ iff $\mathcal{A}$ and $\mathcal{B}$ satisfy the same atomic and negated atomic formulas in $\varphi^0_{\bar{u}}$ where $\mathcal{B} \models \varphi^0_{\bar{u}}[\bar{v}]$ and obviously $\mathcal{A} \models \varphi^0_{\bar{u}}[\bar{u}]$.
For $m > 0$ the duplicator wins the game $G_m(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$ iff
for all $a \in dom(\mathcal{A})$ there is $b \in dom(\mathcal{B})$ such that the duplicator wins $G_{m-1}(\mathcal{A}, \bar{u}a, \mathcal{B}, \bar{v}b)$, and for all $b \in dom(\mathcal{B})$ there is $a \in dom(\mathcal{A})$ such that the duplicator wins $G_{m-1}(\mathcal{A}, \bar{u}a, \mathcal{B}, \bar{v}b)$ iff
for all $a \in dom(\mathcal{A})$ there is $b \in dom(\mathcal{B})$ with $\mathcal{B} \models \varphi^{m-1}_{\bar{u}a}[\bar{v}b]$, and for all $b \in dom(\mathcal{B})$ there is $a \in dom(\mathcal{A})$ with $\mathcal{B} \models \varphi^{m-1}_{\bar{u}a}[\bar{v}b]$ (ind. hyp.) iff
$\mathcal{B} \models \bigwedge_{a\in dom(\mathcal{A})} \exists x_{k+1} \varphi^{m-1}_{\bar{u}a}(x_1, \ldots, x_{k+1}) \wedge \forall x_{k+1} \bigvee_{a\in dom(\mathcal{A})} \varphi^{m-1}_{\bar{u}a}(x_1, \ldots, x_{k+1})$ $[\bar{v}]$ iff $\mathcal{B} \models \varphi^m_{\bar{u}}[\bar{v}]$.

$(i) \Rightarrow (iv)$. If $m = 0$ the proof is handled as above.
If $m > 0$. Suppose that the duplicator wins the game $G_m(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$. The set of formulas $\varphi(x_1, \ldots, x_k)$ of quantifier rank $\leq m$ such that $\mathcal{A} \models \varphi[\bar{u}]$ iff $\mathcal{B} \models \varphi[\bar{v}]$ contains the atomic and negated atomic formulas and this set is closed under $\neg$ and $\vee$. Suppose $\varphi(\bar{x}) \equiv \exists y\psi$ is in that set and $qr(\varphi) \leq m$. Since $y \notin free(\varphi)$ we can assume that y is different from the variables in $\bar{x}$, hence $\varphi(\bar{x}) \equiv \exists y\psi(\bar{x}, y)$. Assume that $\mathcal{A} \models \varphi[\bar{u}]$, then there is $a \in dom(\mathcal{A})$ such that $\mathcal{A} \models \psi[\bar{u}a]$. Since by (i), the duplicator wins the game $G_m(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$ then there is $b \in dom(\mathcal{B})$ such that the duplicator wins $G_{m-1}(\mathcal{A}, \bar{u}a, \mathcal{B}, \bar{v}b)$. As the $qr(\psi) \leq m - 1$ by the ind. hyp. $\mathcal{B} \models \psi[\bar{v}b]$, hence $\mathcal{B} \models \varphi[\bar{v}]$.

$(i) \Rightarrow (iii)$.
If the duplicator wins the game $G_0(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$ there is a partial isomorphism $\bar{u} \mapsto \bar{v}$ from $\mathcal{A}$ to $\mathcal{B}$ iff $\mathcal{A}$ and $\mathcal{B}$ satisfy the same atomic and negated atomic formulas in $\varphi^0_{\bar{u}}$ on $\bar{u}$, $\bar{v}$, respectively. This gives us a non-empty

set I_0 with the partial isomorphism $\bar{u} \mapsto \bar{v}$ from $\mathcal{A}$ to $\mathcal{B}$.
When $m > 0$, if the duplicator wins the game $G_m(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$ then in every move $j \leq m$, for every $a \in dom(\mathcal{A})$ (or $b \in dom(\mathcal{B})$) that the spoiler chooses, the duplicator can chose a correspondent element $b \in dom(\mathcal{B})$ ($a \in dom(\mathcal{A})$) and a partial isomorphism is defined by the sequence of *pairs* of elements chosen up to that move by the two players, which extends the partial isomorphism $\bar{u} \mapsto \bar{v}$. This, together with the induction hypothesis gives us the back and forth properties for $(I_j)_{j<m}$.

□

Note that, we can consider another item in the previous Theorem as follows:

(*v*) The FO type of quantifier rank m of $\bar{u}$ in $\mathcal{A}$ (which may be denoted as $tp_{\mathcal{A}}^{FO_m}(\bar{u})$) is the same as the FO type of quantifier rank m of $\bar{v}$ in $\mathcal{B}$, i.e. $tp_{\mathcal{A}}^{FO_m}(\bar{u}) = tp_{\mathcal{B}}^{FO_m}(\bar{v})$.

Proof. (sketch)
If the duplicator wins the game $G_m(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$ then $tp_{\mathcal{A}}^{FO_m}(\bar{u}) = tp_{\mathcal{B}}^{FO_m}(\bar{v})$.
For $m = 0$ the proof is handled as above.
For $m > 0$, for all $a \in dom(\mathcal{A})$ there is $b \in dom(\mathcal{B})$ (and, for all $b \in dom(\mathcal{B})$ there is $a \in dom(\mathcal{A})$ respectively) such that $\mathcal{B} \models \varphi_{\bar{u}a}^{m-1}[\bar{v}b]$ that is, the $tp_{\mathcal{A}}^{FO_{m-1}}(\bar{u}a)$ of quantifier rank $m-1$ is the same as the $tp_{\mathcal{B}}^{FO_{m-1}}(\bar{v}b)$ of quantifier rank $m-1$ and, the duplicator wins $G_{m-1}(\mathcal{A}, \bar{u}a, \mathcal{B}, \bar{v}b)$. Finally, $\mathcal{B} \models \varphi_{\bar{u}}^{m}[\bar{v}]$ where the duplicator wins $G_m(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$ and also, $tp_{\mathcal{A}}^{FO_m}(\bar{u})$ of quantifier rank m is the same as the $tp_{\mathcal{B}}^{FO_m}(\bar{v})$ of quantifier rank m.

□

If we consider $k = 0$ we have the following corollary.

Corollary 1. *([10]) Given two structures $\mathcal{A}$ and $\mathcal{B}$ of the same vocabulary and for $m \geq 0$, the following are equivalent:*

(*i*) *the duplicator wins $G_m(\mathcal{A}, \mathcal{B})$.*
(*ii*) $\mathcal{B} \models \varphi_{\mathcal{A}}^{m}$.
(*iii*) $\mathcal{A} \cong_m \mathcal{B}$.
(*iv*) $\mathcal{A} \equiv_m \mathcal{B}$.

The equivalence between $\mathcal{A} \equiv_m \mathcal{B}$ and $\mathcal{A} \cong_m \mathcal{B}$ is known as the Fraïseé theorem. The proof of the preceding theorem and specially the equivalence of (*i*) and (*iii*) and the Fraïseé theorem show that these are different formulations of the same fact regarding the close relationship between sequences of partial isomorphisms $(I_j)_{j \leq m}$ and winning strategies for the duplicator over the game $G_m(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$.

We can define Ehrenfeucht-Fraïssé games for the logic $\mathcal{L}_{\infty\omega}$ as well. For two structures $\mathcal{A}$ and $\mathcal{B}$ and two tuples $\bar{u} \in (dom(\mathcal{A}))^k$ and $\bar{v} \in (dom(\mathcal{B}))^k$ the game $G_\infty(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$ is played as $G_m(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$ with the only difference that now, each player can make infinitely many moves. That is, at the end of the game $G_\infty(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$ we will have a map, say $p = \{(u_1, v_1), \ldots, (u_k, v_k), (a_1, b_1), \ldots\}$ for the arbitrary elements chosen by the players during the game, where for all $i \geq 0$, every a_i, b_i chosen by the players are not necessarily elements of the tuples $\bar{u}$ or $\bar{v}$. The duplicator wins the game, if for every move i the map $\{(u_1, v_1), \ldots, (u_k, v_k), (a_1, b_1), \ldots, (a_i, b_i)\}$ is a partial isomorphism, in other case the spoiler wins the game. Next, we give the definition of the back and forth properties for the characterization of equivalence of formulas in $\mathcal{L}_{\infty\omega}$.

Definition 9. *([10], Definition 3.2.6) Two structures $\mathcal{A}$ and $\mathcal{B}$ of the same vocabulary are said to be* partially isomorphic, *denoted $\mathcal{A} \cong_{part} \mathcal{B}$ if there is a nonempty set I of partial isomorphisms from $\mathcal{A}$ to $\mathcal{B}$ with the back and forth properties as follows:*

* *Forth property: For every $p \in I$ and $a \in dom(\mathcal{A})$ there is a $q \in I$ such that $q \supseteq p$ and $a \in do(q)$,*
* *Back property: For every $p \in I$ and $b \in dom(\mathcal{B})$ there is a $q \in I$ such that $q \supseteq p$ and $b \in rg(q)$.*

then, we write $I : \mathcal{A} \cong_{part} \mathcal{B}$.

We obtain in $\mathcal{L}_{\infty\omega}$ similar results to those in FO.

Theorem 2. *([10]) Given two structures $\mathcal{A}$ and $\mathcal{B}$ of the same vocabulary and two k-tuples, for some $k > 0$, $\bar{u} \in (dom(\mathcal{A}))^k$ and $\bar{v} \in (dom(\mathcal{B}))^k$ the following are equivalent:*

(i) *the duplicator wins $G_\infty(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$.*
(ii) *There is a set I of partial isomorphisms from $\mathcal{A}$ to $\mathcal{B}$ with $\bar{u} \mapsto \bar{v} \in I$ such that $I : \mathcal{A} \cong_{part} \mathcal{B}$.*
(iii) *$\bar{u}$ and $\bar{v}$ satisfy the same $\mathcal{L}_{\infty\omega}$ formulas in $\mathcal{A}$ and $\mathcal{B}$, respectively. That is, if $\varphi(x_1, \ldots, x_k)$ is a formula of $\mathcal{L}_{\infty\omega}$, then*

$$\mathcal{A} \models \varphi[\bar{u}] \quad \textit{iff} \quad \mathcal{B} \models \varphi[\bar{v}].$$

When $k = 0$, we get the following results:

Corollary 2. *([10]) Given two structures $\mathcal{A}$ and $\mathcal{B}$ of the same vocabulary, the following are equivalent:*

(i) *the duplicator wins $G_\infty(\mathcal{A}, \mathcal{B})$.*
(ii) $\mathcal{A} \cong_{part} \mathcal{B}$.
(iii) $\mathcal{A} \equiv^{\mathcal{L}_{\infty\omega}} \mathcal{B}$.

Note that, if $\mathcal{A}$ and $\mathcal{B}$ are finite, they satisfy the same FO formulas if and only if they satisfy the same $\mathcal{L}_{\infty\omega}$ formulas [10]. That is,

$$\mathcal{A} \equiv_{FO} \mathcal{B} \quad \text{if and only if} \quad \mathcal{A} \equiv^{\mathcal{L}_{\infty\omega}} \mathcal{B}$$

We know by Proposition 1 that every finite structure $\mathcal{A}$ can be characterized in first-order logic up to isomorphism by a given formula $\varphi_{\mathcal{A}}$ such that $\mathcal{A} \equiv_{FO} \mathcal{B} \Leftrightarrow \mathcal{A} \cong \mathcal{B}$. The logic $\mathcal{L}_{\infty\omega}$ can express *any* class of finite structures $\mathcal{C}$ by the sentence $\bigvee\{\varphi_{\mathcal{A}} | \mathcal{A} \in \mathcal{C}\}$. Clearly, if $\mathcal{A} \cong \mathcal{B}$ it is also the case that $\mathcal{A} \equiv^{\mathcal{L}_{\infty\omega}} \mathcal{B}$.

6 Pebble Games

In this section we present the pebble games where the spoiler and the duplicator have a fixed set of pairs of pebbles and each move consists of placing a pebble which is off the board on an element of a structure or removing a pebble from an element and placing it on another element (or even on the same element).

Consider the following example,

Example 4. Let $\sigma = \langle < \rangle$ be a vocabulary where $<$ denotes natural ordering, two σ-structures $\mathcal{A} = (\{0,1,2\},<)$ and $\mathcal{B} = (\{3,4\},<)$ and the following sentence

$$\varphi = \exists x \exists y (x < y \land \exists x (y < x))$$

Clearly $\mathcal{A} \models \varphi$ but, $\mathcal{B} \not\models \varphi$ and the spoiler has a winning strategy in $G_3(\mathcal{A},\mathcal{B})$. The moves made during the game are sketched out in the following table

	$\mathcal{A}$	$\mathcal{B}$
first move	0	3
second move	1	4
third move	2	?

The elements in column $\mathcal{A}$ are the elements selected by the spoiler and the duplicator has no other element in $\mathcal{B}$ in his third move in order to win the game. But, how could be possible that φ has two variables and even with just two variables the spoiler could win the game? For the first two existential quantifiers ($\exists x \exists y$), the spoiler chooses elements 0 and 1 on $\mathcal{A}$ in his first an second move respectively. The duplicator plays those moves choosing elements 3 and 4 on $\mathcal{B}$. Finally, for the third existential quantifier ($\exists x$), the spoiler chooses in his third move the element 2 in $\mathcal{A}$ such that

$$\mathcal{A} \models \varphi \quad \text{where} \quad 0 < 1 < 2$$

the previous value for variable x (0) makes no longer sense, it has been replaced by element 2. After that, the duplicator has no other choice in $\mathcal{B}$ and finally the spoiler wins.

This version of the games is also played in m moves but this time we use k pair of pebbles. The spoiler and the duplicator will place the pebbles on elements of the structures $\mathcal{A}$ or $\mathcal{B}$ along the game. In a given *pebble game* $G^k_m(\mathcal{A},\bar{u},\mathcal{B},\bar{v})$ where $\bar{u} \in (dom(\mathcal{A}))^l$ and $\bar{v} \in (dom(\mathcal{B}))^l$, for some $1 \leq l \leq k$ we have k pairs of pebbles denoted $\alpha_1, \ldots, \alpha_k$ for $\mathcal{A}$ and $\beta_1, \ldots, \beta_k$ for $\mathcal{B}$. In a given move j of the game, where $1 \leq j \leq m$ the pebble α_i is placed on an element $a_i \in dom(\mathcal{A})$ and the pebble β_i is placed on some element $b_i \in dom(\mathcal{B})$ with $1 \leq i \leq k$. Initially, the pebbles $\alpha_1, \ldots, \alpha_l$ $(\beta_1, \ldots, \beta_l)$ are placed on the elements $u_1, \ldots, u_l$ $(v_1, \ldots, v_l$ respectively) and the pebbles $\alpha_{l+1}, \ldots, \alpha_k$ $(\beta_{l+1}, \ldots, \beta_k)$ are off the board.

In each move, say for instance j, for $1 \leq j \leq m$, the spoiler selects a structure ($\mathcal{A}$ or $\mathcal{B}$) and a pebble of this structure (α_i or β_i respectively) being it off the board or already placed on an element. If the spoiler selects the structure $\mathcal{A}$ he will put the pebble α_i on an element $a_i \in dom(\mathcal{A})$. After that, the duplicator selects the structure $\mathcal{B}$ and pebble β_i and places it on some element $b_i \in dom(\mathcal{B})$. But, if the spoiler selects the structure $\mathcal{B}$ and β_i, he places it on an element b_i of $dom(\mathcal{B})$, then the duplicator selects the structure $\mathcal{A}$ and α_i and places it on some element a_i in $dom(\mathcal{A})$. Note that there could be several pebbles on the same element at a given stage in the game. The elements of the k-tuples $\bar{u}$ and $\bar{v}$ are elements of the domain of structures $\mathcal{A}$ and $\mathcal{B}$ respectively and can then be chosen by the players during the game.

Let's denote by $\bar{u}_{\frac{a}{i}}$ the substitution of the u_i component in $\bar{u}$ by the element $a \in dom(\mathcal{A})$, that is, $u_1, \ldots, u_{i-1}, a, u_{i+1}, \ldots, u_l$. If $\bar{u} \mapsto \bar{v}$ is a partial isomorphism as in Definition 7, in this context we will also call it a *k-partial isomorphism.*

Let's see again Example 4. Before the first move all the pebbles (say α_1, α_2, and β_1, β_2) were off the board. In the first move, the spoiler chose structure $\mathcal{A}$, pebble α_1 and placed it over element 0 in $dom(\mathcal{A})$ and, α_2 was still off the board. The duplicator chose structure $\mathcal{B}$, pebble β_1 and placed it on element 3 in $dom(\mathcal{B})$, meanwhile β_2 was off the board. In second move, the spoiler chose again the structure $\mathcal{A}$, but this time, the pebble α_2 and placed it on 1 in $dom(\mathcal{A})$, then the duplicator chose the structure $\mathcal{B}$ and β_2 and placed it on 4 in $dom(\mathcal{B})$. In the third move, the spoiler chose again structure $\mathcal{A}$ and α_1 and placed it, this time, on 2 in $dom(\mathcal{A})$. In this move there was no a choice for the duplicator in $\mathcal{B}$, then the spoiler won the game.

The duplicator *wins* the game if after the move m he could find a k-partial isomorphism which maps $\bar{u}$ to $\bar{v}$ and the set of elements on $\mathcal{A}$, marked by $\alpha_1, \ldots, \alpha_k$, to the set of elements on $\mathcal{B}$, marked by $\beta_1, \ldots, \beta_k$. Otherwise, the spoiler *wins* the game. We say that the duplicator has a *winning strategy* in a game $G^k_m(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$ if he wins the game independently of the choice of the spoiler at each move.

If we make a slight modification of the Example 4 as follows

Example 5. Let $\sigma = \langle < \rangle$ be a vocabulary, two σ-structures $\mathcal{A} = (\{0,1,2\}, <)$ and $\mathcal{B} = (\{3,4,5\}, <)$ and the following sentence $\varphi = \exists x \exists y (x < y \wedge \exists x y < x)$. The duplicator has a winning strategy for the game $G_3^2(\mathcal{A}, \mathcal{B})$ where $\mathcal{A} \models \varphi$ and $\mathcal{B} \models \varphi$ because there is a k-partial isomorphism from $\mathcal{A}$ to $\mathcal{B}$.

We define the corresponding back and forth properties for Pebble games.

Definition 10. *([10], Definition 3.3.8) Structures $\mathcal{A}$ and $\mathcal{B}$ are* k-m-isomorphic *denoted by $\mathcal{A} \cong_m^k \mathcal{B}$ if there is a sequence $(I_j)_{j \leq m}$ of nonempty sets of k-partial isomorphisms with the following properties:*

* *k-forth property:*
  (*i*) *for $j < m$, $\bar{u} \mapsto \bar{v} \in I_{j+1}$ with length of $\bar{u} = k$, $1 \leq i \leq k$ and, $a \in dom(\mathcal{A})$ there is $b \in dom(\mathcal{B})$ such that $\bar{u}\frac{a}{i} \mapsto \bar{v}\frac{b}{i} \in I_j$.*
  (*ii*) *for $j < m$, $\bar{u} \mapsto \bar{v} \in I_{j+1}$ with length of $\bar{u} < k$, and $a \in dom(\mathcal{A})$ there is $b \in dom(\mathcal{B})$ such that $\bar{u}a \mapsto \bar{v}b \in I_j$.*

* *k-back property:*
  (*i*) *for $j < m$, $\bar{u} \mapsto \bar{v} \in I_{j+1}$ with length of $\bar{u} = k$, $1 \leq i \leq k$ and, $b \in dom(\mathcal{B})$ there is $a \in dom(\mathcal{A})$ such that $\bar{u}\frac{a}{i} \mapsto \bar{v}\frac{b}{i} \in I_j$.*
  (*ii*) *for $j < m$, $\bar{u} \mapsto \bar{v} \in I_{j+1}$ with length of $\bar{u} < k$, and $b \in dom(\mathcal{B})$ there is $a \in dom(\mathcal{A})$ such that $\bar{u}a \mapsto \bar{v}b \in I_j$.*

Then we write $(I_j)_{j \leq m} : \mathcal{A} \cong_m^k \mathcal{B}$.

The pebble game $G_\infty^k(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$ is defined as $G_m^k(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$ with the difference that $G_\infty^k(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$ allows infinitely many moves.

Theorem 3. *([10], Theorem 3.3.5) Given two σ-structures $\mathcal{A}$ and $\mathcal{B}$, two l-tuples $\bar{u} \in dom(\mathcal{A})^l$ and $\bar{v} \in dom(\mathcal{B})^l$, with $1 \leq l \leq k$*

- *$\bar{u}$ satisfies in $\mathcal{A}$ the same FO^k formulas of quantifier rank $\leq m$ as $\bar{v}$ in $\mathcal{B}$ iff the duplicator wins the game $G_m^k(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$.*
- *$\bar{u}$ satisfies in $\mathcal{A}$ the same $\mathcal{L}_{\infty\omega}^k$ formulas as $\bar{v}$ in $\mathcal{B}$ iff the duplicator wins the game $G_\infty^k(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$.*

similarly

- *$\mathcal{A} \equiv_m^k \mathcal{B}$ iff the duplicator wins $G_m^k(\mathcal{A}, \mathcal{B})$.*
- *$\mathcal{A} \equiv^{\mathcal{L}_{\infty\omega}^k} \mathcal{B}$ iff the duplicator wins $G_\infty^k(\mathcal{A}, \mathcal{B})$.*

The *k-partially isomorphic* structures denoted $\mathcal{A} \cong_{part}^k \mathcal{B}$ and the set $I : \mathcal{A} \cong_{part}^k \mathcal{B}$ of *k-partial isomorphism* for $\mathcal{L}_{\infty\omega}^k$ are similarly defined.

Definition 11. *Two structures $\mathcal{A}$ and $\mathcal{B}$ are* k-partial isomorphic *denoted $\mathcal{A} \cong^k_{part} \mathcal{B}$ if there is a nonempty set I of partial isomorphisms from $\mathcal{A}$ to $\mathcal{B}$ with the k-back and k-forth properties as follows:*

* *k-forth property:*
  (*i*) *$\bar{u} \mapsto \bar{v} \in I$ with length of $\bar{u} = k$, $1 \leq i \leq k$ and, $a \in dom(\mathcal{A})$ there is $b \in dom(\mathcal{B})$ such that $\bar{u}_{\frac{a}{i}} \mapsto \bar{v}_{\frac{b}{i}} \in I$.*
  (*ii*) *$\bar{u} \mapsto \bar{v} \in I$ with length of $\bar{u} < k$, and $a \in dom(\mathcal{A})$ there is $b \in dom(\mathcal{B})$ such that $\bar{u}a \mapsto \bar{v}b \in I$.*

* *k-back property:*
  (*i*) *$\bar{u} \mapsto \bar{v} \in I$ with length of $\bar{u} = k$, $1 \leq i \leq k$ and, $b \in dom(\mathcal{B})$ there is $a \in dom(\mathcal{A})$ such that $\bar{u}_{\frac{a}{i}} \mapsto \bar{v}_{\frac{b}{i}} \in I$.*
  (*ii*) *$\bar{u} \mapsto \bar{v} \in I$ with length of $\bar{u} < k$, and $b \in dom(\mathcal{B})$ there is $a \in dom(\mathcal{A})$ such that $\bar{u}a \mapsto \bar{v}b \in I$.*

6.1 Inductive construction of the k-m-isomorphic type

Given a vocabulary σ, a structure $\mathcal{A}$ of vocabulary σ, an integer $k > 0$ and an l-tuple $\bar{u} = (u_1, \ldots, u_l)$ over $(dom(\mathcal{A}))^l$ with $1 \leq l \leq k$, and following [10], we define the *k-m-isomorphic type* of $\mathcal{A}$ by an FO^k formula $\psi^m_{\bar{u}}$ by induction on m.

$$\psi^0_{\bar{u}}(x_1, ..., x_l) \equiv \bigwedge\{\psi(x_1, ..., x_l) | \psi \text{ atomic or negated atomic formulas such that } \mathcal{A} \models \psi[u_1, ..., u_l]\}$$

if the length of $\bar{u} < k$ then

$$\psi^{m+1}_{\bar{u}}(x_1, ..., x_l) \equiv \psi^m_{\bar{u}}(x_1, ..., x_l) \wedge \Big(\bigwedge_{a \in dom(\mathcal{A})} \exists x_{l+1} \psi^m_{\bar{u}a}(x_1, ..., x_{l+1})\Big) \wedge \forall_{x_{l+1}} \bigvee_{a \in dom(\mathcal{A})} \psi^m_{\bar{u}a}(x_1, ..., x_{l+1})$$

if the length of $\bar{u} = k$ then

$$\psi^{m+1}_{\bar{u}}(x_1, \ldots, x_k) \equiv \psi^0_{\bar{u}}(x_1, \ldots, x_k) \wedge \bigwedge_{1 \leq i \leq k} \Big(\bigwedge_{a \in \mathcal{A}} \exists x_i \psi^m_{\bar{u}\frac{a}{i}}(x_1, \ldots, x_k) \wedge \forall x_i \bigvee_{a \in \mathcal{A}} \psi^m_{\bar{u}\frac{a}{i}}(x_1, \ldots, x_k)\Big).$$

The first part of the formula, $\psi^0_{\bar{u}}(x_1, \ldots, x_l)$, is the conjunction of all atomic and negated atomic formulas with free variables among $\{x_1, \ldots, x_l\}$ such that $\mathcal{A} \models \psi^0_{\bar{u}}[\bar{u}]$. It describes the isomorphic type of the substructure generated by $\bar{u} \in (dom(\mathcal{A}))^l$ in $\mathcal{A}$ and also by $\bar{v} \in (dom(\mathcal{B}))^l$ in $\mathcal{B}$, for any structure $\mathcal{B}$ such that $\mathcal{B} \models \psi^0_{\bar{u}}[\bar{v}]$ if and only if there is a k-partial isomorphism $\bar{u} \mapsto \bar{v}$ from $\mathcal{A}$ to $\mathcal{B}$. That is, $\mathcal{B}$ satisfies the conjunction of all atomic and negated atomic formulas in $\psi^0_{\bar{u}}$ with free variables among $\{x_1, \ldots, x_l\}$, where obviously $\mathcal{A} \models \psi^0_{\bar{u}}[\bar{u}]$.

When $m > 0$, the formula describes to which isomorphic type the tuple $\bar{u}$ (and $\bar{v}$) will be extended in m moves, adding (or replacing) at most one element in each move. We will explain intuitively the meaning of $\mathcal{B} \models \psi^{m+1}_{\bar{u}}[\bar{v}]$. Each new element $a \in dom(\mathcal{A})$ ($b \in dom(\mathcal{B})$) extends the tuple $\bar{u}$ ($\bar{v}$) of length l, where $l < k$, to a tuple $\bar{u}a$ ($\bar{v}b$) of length $l+1$ denoted $\bar{u}a = (u_1, \ldots, u_l, a)$ ($\bar{v}b = (v_1, \ldots, v_l, b)$ respectively).

If the length of $\bar{u}$ is $< k$, the formula guarantees that the tuple $\bar{u}$ in $\mathcal{A}$ satisfies the same FO^k formulas of quantifier rank $\leq m$ than the tuple $\bar{v}$ in $\mathcal{B}$. That is, up to quantifier rank $\leq m$, the FO^k type of tuple $\bar{v}$ in $\mathcal{B}$ is the same as the FO^k type of tuple $\bar{u}$ in $\mathcal{A}$.
In the big conjunction followed by the existential quantifier, for each element $a \in dom(\mathcal{A})$ there is an element $b \in dom(\mathcal{B})$ such that the tuple $\bar{v}b$ in structure $\mathcal{B}$ satisfies the same FO^k formulas of quantifier rank $\leq m$, with free variables $(x_1, \ldots, x_{l+1})$, than the tuple $\bar{u}a$ in structure $\mathcal{A}$, i.e. $\mathcal{B} \models \psi^m_{\bar{u}a}[\bar{v}b]$. That is, the FO^k type of quantifier rank $\leq m$ of the tuple $\bar{u}a$ of length $l+1$ in $\mathcal{A}$ is the same as the FO^k type of quantifier rank $\leq m$ of the tuple $\bar{v}b$ of length $l+1$ in $\mathcal{B}$.
The following part of the formula, the universal quantifier followed by the big disjunction describes that, for every element $b \in dom(\mathcal{B})$ there is an element $a \in dom(\mathcal{A})$ such that the tuple $\bar{u}a$ in $\mathcal{A}$ satisfies the same FO^k formulas of quantifier rank $\leq m$ with free variables $(x_1, \ldots, x_{l+1})$ than the tuple $\bar{v}b$ in structure $\mathcal{B}$. Then, the FO^k type of quantifier rank $\leq m$ of tuple $\bar{v}b$ in $\mathcal{B}$ is the same as the FO^k type of quantifier rank $\leq m$ of tuple $\bar{u}a$ in structure $\mathcal{A}$.

Games $G^k_m(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$ and $G^k_\infty(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$ are played with k pairs of pebbles. That is, the formulas satisfied by the tuples $\bar{u}$ in $\mathcal{A}$ and $\bar{v}$ in $\mathcal{B}$ are formulas over the logics FO^k and $\mathcal{L}^k_{\infty\omega}$. For that reason, the length of the tuples used in the formula $\psi^{m+1}_{\bar{u}}(x_1, \ldots, x_l)$ has to be considered. When the length of $\bar{u}$ is $= k$, a sequence of k pair of elements from $\mathcal{A}$ and $\mathcal{B}$ have been chosen. We consider the substitution $\bar{u}_{\frac{a}{i}}$ ($\bar{v}_{\frac{b}{i}}$) of the u_i component of $\bar{u}$ (v_i component of $\bar{v}$), for any $1 \leq i \leq k$, by the element $a \in dom(\mathcal{A})$ ($b \in dom(\mathcal{B})$) in order to guarantee the following move of the game $G^k_m(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$ or $G^k_\infty(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$.

In terms of the pebble games, we say that the duplicator wins the game $G^k_m(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$ as follows. When $m = 0$, if there is a partial isomorphism

$\bar{u} \mapsto \bar{v}$ from $\mathcal{A}$ to $\mathcal{B}$ such that $\mathcal{A}$ and $\mathcal{B}$ satisfy the same FO^k atomic formulas. When the length of $\bar{u}$ ($\bar{v}$) is $< k$, the duplicator has a winning strategy for all the previous m moves. For every pebble that the spoiler places on an element in structure $\mathcal{A}$, the duplicator will place his pebble in an element of $\mathcal{B}$ such that the duplicator wins the game $G^k_m(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$. For every pebble that the spoiler places on an element of $\mathcal{B}$ the duplicator will place his corresponding pebble on an element of $\mathcal{A}$ such that the duplicator wins the game $G^k_m(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$.
If the length of $\bar{u}$ is equal to k, it means that the k pairs of pebbles have been chosen and placed on k pairs of elements of $\mathcal{A}$ and $\mathcal{B}$. For the next move, a pair of pebbles α_i, β_i is removed and placed on elements $a \in dom(\mathcal{A})$ and $b \in dom(\mathcal{B})$ replacing, respectively, the component u_i in $\bar{u}$ by a and v_i in $\bar{v}$ by b.

The formula $\psi^{m+1}_{\bar{u}}$ characterizes in $\mathcal{A}$ the l-tuples $\bar{u}$ up to equivalence on FO formulas with k variables and quantifier rank $m+1$.

Theorem 4. *([10]) Let $\mathcal{A}$ and $\mathcal{B}$ be two structures of the same relational vocabulary and let $\bar{u} \in (dom(\mathcal{A}))^l$ and $\bar{v} \in (dom(\mathcal{B}))^l$ for some $0 < l \leq k$ and $m \geq 0$. The following are equivalent:*

(i) the duplicator wins $G^k_m(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$.
(ii) There is a sequence of partial isomorphisms from $\mathcal{A}$ to $\mathcal{B}$ $(I_j)_{j \leq m}$ with $\bar{u} \mapsto \bar{v} \in I_m$ such that $(I_j)_{j \leq m} : \mathcal{A} \cong^k_m \mathcal{B}$.
(iii) $\mathcal{B} \models \psi^m_{\bar{u}}[\bar{v}]$.
(iv) $\bar{u}$ satisfies in $\mathcal{A}$ the same FO^k formulas of quantifier rank $\leq m$ as $\bar{v}$ in $\mathcal{B}$.
(v) up to quantifier rank m, the FO^k type of $\bar{u}$ in $\mathcal{A}$ is the same than the FO^k type of $\bar{v}$ in $\mathcal{B}$.

Theorem 5. *([10]) Let $\mathcal{A}$ and $\mathcal{B}$ be two structures of the same relational vocabulary and let $\bar{u} \in (dom(\mathcal{A}))^l$ and $\bar{v} \in (dom(\mathcal{B}))^l$ for some $0 < l \leq k$. The following are equivalent:*

(i) the duplicator wins $G^k_\infty(\mathcal{A}, \bar{u}, \mathcal{B}, \bar{v})$.
(ii) There is set I of partial isomorphism from $\mathcal{A}$ to $\mathcal{B}$ with $\bar{u} \mapsto \bar{v} \in I$ such that $I : \mathcal{A} \cong^k_{part} \mathcal{B}$.
(iii) $\bar{u}$ satisfies in $\mathcal{A}$ the same formulas of $\mathcal{L}^k_{\infty\omega}$ as $\bar{v}$ in $\mathcal{B}$.

6.2 Isolating Formula

Let's introduce a fragment of FO^k logic, denoted $FO^{k,m}$ for some integers k and m that contains the FO^k formulas of quantifier rank up to

m. $FO^{k,m}$ is itself a logic but clearly, less expressive than FO^k. Given a σ-structure $\mathcal{A}$, $k \geq l > 0$ and, two l-tuples $\bar{u}$ and $\bar{u}' \in (dom(\mathcal{A}))^l$ tuples $\bar{u}$ and $\bar{u}'$ may have the same $FO^{k,m}$ type but different FO^k type because of the difference as to expressibility of the two logics. That is, some properties of these tuples which are expressible in FO^k might not be expressible in $FO^{k,m}$. Note that, considering that $\mathcal{A}$ is a finite structure, we could choose a sufficient large m such that those tuples can be distinguishable via $FO^{k,m}$ in $\mathcal{A}$. That is the idea of the following definition.

Definition 12. *([8], Definition 2.19) Let σ be a vocabulary, let $\mathcal{A}$ be a σ-structure, let $\bar{u}$ be a tuple of length l in $(dom(\mathcal{A}))^l$, for a given $1 \leq l \leq k$. The* Scott rank *of the tuple $\bar{u}$ in $\mathcal{A}$ with respect to k, denoted $sr^k_{\mathcal{A}}(\bar{u})$ is the least m such that for every l-tuple $\bar{u}'$ over $\mathcal{A}$, whenever*

$$\langle \mathcal{A}, \bar{u} \rangle \equiv^k_m \langle \mathcal{A}, \bar{u}' \rangle \quad \textit{then} \quad \langle \mathcal{A}, \bar{u} \rangle \equiv^k \langle \mathcal{A}, \bar{u}' \rangle$$

The Scott rank *of a structure $\mathcal{A}$ with respect to k denoted $sr^k(\mathcal{A})$, is equal to $sup(\{sr^k_{\mathcal{A}}(\bar{u}) | \bar{u} \in (dom(\mathcal{A}))^k\})$.*

$sr^k_{\mathcal{A}}(\bar{u})$ provides a value for m sufficiently big such that, tuples $\bar{u}$ and $\bar{u}'$ can be distinguished using $FO^{k,m}$ if they can be distinguished in FO^k over $\mathcal{A}$.

Let's define the *k-Scott formula* of tuple $\bar{u}$ in $\mathcal{A}$

$$\sigma_{\mathcal{A},\bar{u}}(x_1, \ldots, x_l) := \psi_{\bar{u}}^{sr^k(\mathcal{A})}(x_1, \ldots, x_l) \wedge \bigwedge_{\bar{v} \in dom(\mathcal{A})^k} \forall x_1, \ldots, \forall x_k (\psi_{\bar{v}}^{sr^k(\mathcal{A})}(x_1, \ldots, x_k) \rightarrow \psi_{\bar{v}}^{sr^k(\mathcal{A})+1}(x_1, \ldots, x_k)).$$

This is an FO^k-formula of quantifier rank $sr^k(\mathcal{A}) + 1 + k$ that characterizes the structure $\mathcal{A}$ up to equivalence of FO^k formulas. In particular, the Scott sentence $\sigma_{\mathcal{A}}$ captures the whole $\mathcal{L}^k_{\infty\omega}$ theory of $\mathcal{A}$, that is

Theorem 6. *([8], Theorem 2,12) For every finite structure $\mathcal{A}$ and any k, there is a sentence $\sigma_{\mathcal{A}} \in FO^k$ such that for any finite structure $\mathcal{B}$, $\mathcal{B} \models \sigma_{\mathcal{A}}$ if and only if $\mathcal{A} \equiv^k \mathcal{B}$.*

Intuitively, the formula $\sigma_{\mathcal{A},\bar{u}}$ expresses that, for each tuple $\bar{u} = (u_1, \ldots, u_l)$ of elements of $dom(\mathcal{A})$, with $1 \leq l \leq k$ and for each tuple $\bar{v} = (v_1, \ldots, v_l)$ of elements of $dom(\mathcal{B})$ for some structure $\mathcal{B}$, if the $FO^{k,m}$ type of tuple $\bar{v}$ in $\mathcal{B}$ is the same as the $FO^{k,m}$ type of $\bar{u}$ in $\mathcal{A}$ then, the

formula guarantees that the $FO^{k,m+1}$ type of $\bar{v}$ in $\mathcal{B}$ is the same as the $FO^{k,m+1}$ type of tuple $\bar{u}$ in $\mathcal{A}$ for a given m which denotes the Scott rank of $\mathcal{A}$. Note that, if $\mathcal{B} \models \sigma_{\mathcal{A},\bar{u}}[\bar{v}]$ then the $sr^k(\mathcal{A}) \geq sr^k(\mathcal{B})$. That is, $sr^k(\mathcal{A})$ is an m sufficiently big such that it permits us to express the FO^k types of all l-tuples in $\mathcal{B}$. For every l-tuple $\bar{v}$ in $\mathcal{B}$ the FO^k type of $\bar{v}$ is the same as the $FO^{k,m}$ type of $\bar{v}$ in $\mathcal{B}$.
For the given sequence of elements $v_1, \ldots, v_l$ of $\mathcal{B}$, and therefore for $u_1, \ldots, u_l$ in $\mathcal{A}$, m is big enough such that we can express enough properties of those tuples in their corresponding structures in order to distinguish them from tuples in the same structure with different FO^k types. For two finite structures $\mathcal{A}$ and $\mathcal{B}$, m is considered to be $\leq (|\mathcal{A}|+1)^k \cdot (|\mathcal{B}|+1)^k$ since there are less than $(|\mathcal{A}|+1)^k \cdot (|\mathcal{B}|+1)^k$ k-partial isomorphisms from $\mathcal{A}$ to $\mathcal{B}$.

Corollary 3. *([8]) For every structure $\mathcal{A}$, $l \leq k$ and, sequence $u_1, \ldots, u_l$ of elements in $(dom(\mathcal{A}))^k$, there is a formula $\phi \in tp_{\mathcal{A}}^{FO^k}$ such that for any structure $\mathcal{B}$ and elements $v_1, \ldots, v_l \in (dom(\mathcal{B}))^k$,*

$$\mathcal{B} \models \phi[v_1, \ldots, v_l] \quad \textit{iff} \quad tp_{\mathcal{A}}^{FO^k}(u_1, \ldots, u_l) = tp_{\mathcal{B}}^{FO^k}(v_1, \ldots, v_l).$$

If ϕ satisfies the conditions of the corollary we say that ϕ *isolates* $tp_{\mathcal{A}}^{FO^k}(u_1, \ldots, u_l)$. We also say that ϕ is an *isolating formula* for the FO^k type of $\bar{u}$ in $\mathcal{A}$, denoted $tp_{\mathcal{A}}^{FO^k}(u_1, \ldots, u_l)$. In this sense the k-Scott formula is an isolating formula for the FO^k type of the tuple $\bar{u}$ over the structure $\mathcal{A}$. It is a single (finite) formula expressible in FO^k with quantifier rank $sr^k(\mathcal{A}) + 1 + k$.

7 Databases with Redundant Relations

It is well known that, depending on the design, a database may contain *redundant information*, i.e., it may contain the same information stored in more than one place within the database. Redundant storage of information can lead to a variety of problems on the updating, insertion and deletion of data. This is a very important problem and has been studied extensively in the field of databases, indeed almost all database textbooks include this topic, see for instance [1, 20], among other sources. Traditionally, the redundancy of information is studied by incorporating a particular class of *integrity constraints* to databases (that is, properties that are supposed to be satisfied by all instances of a given database) known as *functional dependencies*, and the well known Normalization techniques. We will take here a different approach based on the notion of FO types of tuples and their isolating formulas.

Consider a nonempty schema $\sigma = \langle R_1, \ldots, R_s \rangle$. We will say that given a database $\mathcal{A}$ of schema σ, a relation $R_i^{\mathcal{A}}$ does not contribute with any additional information to $\mathcal{A}$ than the information already contributed by the other relations in $\mathcal{A}$ if the relation $R_i^{\mathcal{A}}$ can be obtained (i.e., defined) by an FO formula over the reduced database of schema $\sigma - R_i$. We will call this kind of relations, *redundant relations*. Let's consider an example.

Example 6. Let $\sigma = \langle Clients, Investments, MatInv \rangle$, with respective arities 4, 6 and 3, be the schema of a Bank's database $\mathcal{B}$ where $MatInv^{\mathcal{B}}$ contains the following information for the investments that are close to the maturity date.

ClientID	InvestmentID	MaturityDate
1012	01 – 4503	14/01/2001
1012	03 – 2941	23/01/2001
2315	01 – 1541	15/01/2001

Moreover, the relation $Clients^{\mathcal{B}}$ records the following details of the clients: ClientID, FirstName, LastName, and Address; and the relation $Investment^{\mathcal{B}}$ records the following details of all investments: InvestmentID, ClientID, InterestRate, Term, StartDate, MaturityDate.

Clearly, every tuple in $MatInv^{\mathcal{B}}$ is a sub-tuple of some tuple in $Investment^{\mathcal{B}}$ and $MatInv^{\mathcal{B}}$ can be obtained by evaluating an FO formula over the reduced database $\mathcal{B}'$ of vocabulary $\tau = \langle Clients,\ Investments \rangle$.

As the FO types of all k-tuples in a database $\mathcal{A}$ of some schema σ describe all FO properties which are satisfied by the tuples of arity up to k in $\mathcal{A}$, every FO query of arity up to k will be equivalent in $\mathcal{A}$ to the disjunction of some of the FO isolating formulas in the FO types for k-tuples in $\mathcal{A}$. A consequence of this is that, we can eliminate a relation $R^{\mathcal{A}}$ of arity k from $\mathcal{A}$ as long as the FO types for k-tuples realized in $\mathcal{A}$ are not altered, because we can always recover the eliminated relation $R^{\mathcal{A}}$, i.e., we can always write an FO formula which evaluated on $\mathcal{A}$ will return the relation $R^{\mathcal{A}}$. For instance, if we eliminate the relation $MatInv^{\mathcal{B}}$ in the previous example the FO types for 3-tuples realized in $\mathcal{B}$ will not be altered, that is, the FO types of all 3-tuples $\bar{a} \in (dom(\mathcal{B}))^3$ will be the same with or without the relation $MatInv^{\mathcal{B}}$ in the database and hence this relation can be eliminated without any loss of information. Redundant relations are relations that do not provide any further information in the sense that two arbitrary tuples of a given database $\mathcal{A}$ will be distinguishable or not, up to isomorphisms, independently of the presence of the redundant relations in the database. Next, we will analyze redundant relations in databases as well as in classes of databases.

7.1 Redundant Relations in Databases

We start this section with two definitions which formalize some of the notions discussed above.

Definition 13. *Let σ be a schema, let $\mathcal{A}$ be a database of schema σ, and let R be a given relation symbol in σ. We denote as*

- *$\sigma - R$ the schema obtained by eliminating from σ the relation symbol R, i.e., if $\sigma = \langle R, R_1, \ldots, R_s\rangle$, then $\sigma - R = \langle R_1, \ldots, R_s\rangle$,*
- *FO^{σ} and $FO^{\sigma-R}$ the set of formulas of FO over the schemas σ and $\sigma - R$ respectively and,*
- *$\mathcal{A}|_{\sigma-R}$ the* reduced database *of schema $\sigma - R$ obtained by eliminating the relation $R^{\mathcal{A}}$ from $\mathcal{A}$.*

Definition 14. *Given a schema $\sigma = \langle R, R_1, \ldots, R_s\rangle$ with $s > 0$ where R is of arity k, and a database $\mathcal{A}$ in $\mathcal{B}_{\sigma}$, R is a* redundant relation *in the database $\mathcal{A}$ if for all k-tuples $\bar{u}$ and $\bar{v}$ in $(dom(\mathcal{A}))^k$,*

$$tp_{\mathcal{A}}^{FO^{\sigma}}(\bar{u}) = tp_{\mathcal{A}}^{FO^{\sigma}}(\bar{v}) \quad \textit{iff} \quad tp_{\mathcal{A}}^{FO^{\sigma-R}}(\bar{u}) = tp_{\mathcal{A}}^{FO^{\sigma-R}}(\bar{v}).\ \textit{i.e.,}$$

the equivalence classes induced by the FO^{σ} types of the k-tuples in $(dom(\mathcal{A}))^k$ coincide with the equivalence classes induced by the $FO^{\sigma-R}$ types of k-tuples in $(dom(\mathcal{A}))^k$.

Consider the following example.

Example 7. Let BT be the class of complete binary trees of schema $\tau = \langle E, C\rangle$ where E is the edge relation symbol and C is a unary relation symbol. In Figure 7 we show two binary trees, $\mathcal{G}_1$ and $\mathcal{G}_2$ which belong to BT. In them, the unary relation symbol C is interpreted as the set of painted nodes.

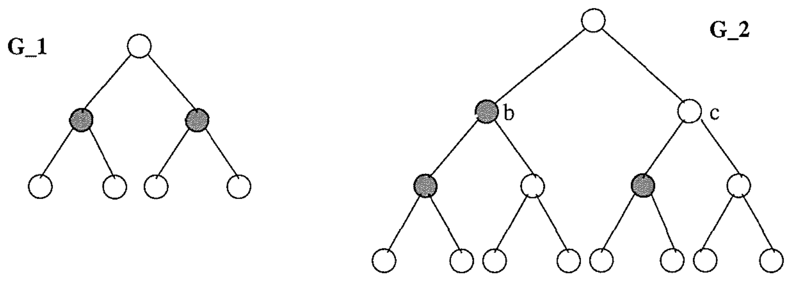


Figure 7

Note that, 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. It is straightforward to note that the relation $C^{\mathcal{G}_1}$ is a redundant relation in $\mathcal{G}_1$, i.e., for every elements $u, v \in dom(\mathcal{G}_1)$,

$$tp_{\mathcal{G}_1}^{FO^\tau}(u) = tp_{\mathcal{G}_1}^{FO^\tau}(v) \quad \text{iff} \quad tp_{\mathcal{G}_1}^{FO^{\tau-C}}(u) = tp_{\mathcal{G}_1}^{FO^{\tau-C}}(v).$$

But this is certainly 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 $\mathcal{G}_2$. Clearly, $tp_{\mathcal{G}_2}^{FO^\tau}(b) \neq tp_{\mathcal{G}_2}^{FO^\tau}(c)$ while $tp_{\mathcal{G}_2}^{FO^{\tau-C}}(b) = tp_{\mathcal{G}_2}^{FO^{\tau-C}}(c)$.

We will see now how to build, for any redundant relation $R^{\mathcal{A}}$ in a database $\mathcal{A}$ of schema σ, an FO formula φ_R such that $\varphi_R^{\mathcal{A}|_{\sigma-R}} = R^{\mathcal{A}}$. Informally, φ_R is an FO query which evaluated in the reduced database $\mathcal{A}|_{\sigma-R}$ gives the relation R.

The proof of the next Theorem readily follows from the definition of isolating formulas for the FO types of k-tuples, and of redundant relations. Note that, 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 for l-tuples in Subsection 6.2. Considering the formulas $\varphi^m_{\bar{u}}(\bar{x})$, defined in Subsection 5.1 and Theorem 2.2.8 in [10], as we are dealing with finite structures there will always be an m big enough such that for all σ-structures $\mathcal{B}$ and k-tuples $\bar{v}$ over $(dom(\mathcal{B}))^k$ we have that

$$\mathcal{B} \models \varphi^m_{\mathcal{A},\bar{u}}[\bar{v}] \quad \text{iff} \quad tp^{FO}_{\mathcal{A}}(\bar{u}) = tp^{FO}_{\mathcal{B}}(\bar{v})$$

and that is the *isolating formula* for the FO type of $\bar{u}$ in $\mathcal{A}$. It is well known (see [10]) that n^k is a value of m big enough to build the isolating formula for an arbitrary k-tuple in a given database of size n.

Theorem 7. *Let $\mathcal{A}$ be a database of schema $\sigma = \langle R, R_1, \ldots, R_s\rangle$ with $s > 0$, let $R^{\mathcal{A}} = \{\bar{a}^1_k, \ldots, \bar{a}^n_k\}$ be a redundant relation of arity k and cardinality n in $\mathcal{A}$, and let $\varphi_R(x_1, \ldots, x_k)$ be the FO formula*

$$\psi_1(x_1, \ldots, x_k) \vee \psi_2(x_1, \ldots, x_k) \vee \ldots \vee \psi_n(x_1, \ldots, x_k)$$

where, for $1 \le i \le n$, ψ_i is the isolating formula for the $FO^{\sigma-R}$ type of the k-tuple $\bar{a}^i_k$. It follows that $\varphi_R^{\mathcal{A}|_{\sigma-R}} = R^{\mathcal{A}}$. And, obviously also $\varphi_R^{\mathcal{A}} = R^{\mathcal{A}}$.

Note that, every redundant relation R of arity k can be obtained by an FO query on the corresponding reduced database as described in the previous theorem because a relation is redundant if and only if it has *complete FO types for the k-tuples.* That is, a relation R of arity k is redundant if and only if for every FO type for the k-tuples α realized by the database, either all k-tuples whose type is α belong to R or none of them does.

Observe that in Example 7, 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 neither for nodes in the second nor in the third level of $\mathcal{G}_2$.

So far, we have been considering databases with only one redundant relation however databases can have several redundant relations. Let's see the following example.

Example 8. Consider the complete binary tree $\mathcal{A}$ as in Figure 8 of schema $\sigma = \langle E, R, B\rangle$ where E is the edge relation symbol and, R and B are unary relation symbols that may be thought as colorings of the nodes of $\mathcal{A}$. Where $R^{\mathcal{A}} = \{b\}$ and $B^{\mathcal{A}} = \{c\}$ (i.e., node a is not colored).

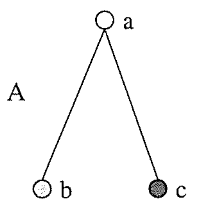


Figure 8

Let's focus in the depth one of the tree. Node b can be distinguished from node c even being both in the same depth of the tree. That is, there are two FO formulas, $\psi_b(x)$ and $\psi_c(x)$, that when evaluated on $\mathcal{A}$ result in nodes b and c respectively, as follows:

$$\psi_b(x) \equiv R(x) \quad \text{and} \quad \psi_c(x) \equiv B(x)$$

We note that $R^{\mathcal{A}}$ as well as $B^{\mathcal{A}}$ are redundant relations in $\mathcal{A}$. If we eliminate the relation symbol R from σ, node b can still be distinguished from node c in $\mathcal{A}|_{\sigma-R}$ by the FO formulas:

$$\psi_c(x) \equiv B(x) \quad \text{and,} \quad \varphi_b(x) \equiv \exists y\big(E(y,x) \wedge \forall z(\neg E(z,y))\big) \wedge \neg B(x)$$

Note also that, if we eliminate the relation symbol B from σ, node b can still be distinguished from node c in $\mathcal{A}|_{\sigma-B}$ by the FO formulas:

$$\varphi_c(x) \equiv \exists y\big(E(y,x) \wedge \forall z(\neg E(z,y))\big) \wedge \neg R(x) \quad \text{and,} \quad \psi_b(x) \equiv R(x)$$

Nevertheless, this is not the case if we eliminate both relations (R and B) *simultaneously* from σ. In that case, b would not be distinguished from node c in $\mathcal{A}_{\sigma-\{R,B\}}$.

None of the first two reductions *alter* the FO types of the 1-tuples in $\mathcal{A}$. That is, the equivalence classes induced by the FO^{σ} types of the 1-tuples in $(dom(\mathcal{A}))^1$ coincide with the equivalence classes induced by the $FO^{\sigma-R}$ types of the 1-tuples in $(dom(\mathcal{A}))^1$ and also coincide with the equivalence classes induced by the $FO^{\sigma-B}$ types of the 1-tuples in $(dom(\mathcal{A}))^1$.

The following theorem follows from Definition 14 and the constructibility of the formulas $\varphi_{\bar{u}}^{m}$ of Subsection 5.1 considering the remark before Theorem 7.

Theorem 8. *The following problems are decidable:*

(*i*) *Given a schema σ, a relation symbol $R \in \sigma$ and a database $\mathcal{A}$ of schema σ, to decide whether R 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 R in σ which is redundant in $\mathcal{A}$.*

7.1.1 Evaluating *FO* Formulas in Reduced Databases

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 Theorem 7, the k-tuples in a redundant relation $R^{\mathcal{A}}$ of arity k in $\mathcal{A}$ can be expressed by an FO formula $\varphi_R(x_1, \ldots, x_k)$ 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 $\varphi_R(x_1, \ldots, x_k)$. That is, every atomic formula formed with the relation symbol R in φ_q can be replaced in φ'_q by the formula $\varphi_R(x_1, \ldots, x_k)$ in the database $\mathcal{A}|_{\sigma-R}$. We only need to take care of the appropriate re-naming of variables in φ_R. Let's see this in an example.

Example 9. Recall the tree $\mathcal{G}_1$ of Figure 7 as in Example 7 where $C^{\mathcal{G}_1}$ is a redundant relation in $\mathcal{G}_1$, and let's consider the reduced database $\mathcal{G}_1|_{\tau-C}$ where we can define the relation $C^{\mathcal{G}_1}$ by the following FO formula:

$$\varphi_C(x) \equiv \exists y\big(E(y,x) \wedge \forall z(\neg E(z,y))\big)$$

Consider also the following query q over $\mathcal{G}_1$:

"Which are the children of the colored nodes in $\mathcal{G}_1$?".

the query q can be expressed by the following FO formula in $\mathcal{G}_1$:

$$\varphi_{(q,\mathcal{G}_1)}(x) \equiv \exists z(C(z) \wedge E(z,x))$$

finally, replacing the atomic formula $C(z)$ in $\varphi_{(q,\mathcal{G}_1)}(x)$ by the formula $\varphi_C(x)$ we obtain in $\mathcal{G}_1|_{\tau-C}$ the following formula (note that we have replaced variable x by z and variable z by w in $\varphi_C(x)$):

$$\varphi_{(q,\mathcal{G}_1|_{\tau-C})}(x) \equiv \exists z\Big(\exists y\big(E(y,z) \wedge \forall w(\neg E(w,y))\big) \wedge E(z,x)\Big)$$

Note that, 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 $\mathcal{L}$.

7.1.2 Redundant Relations in Databases with Basic Schemas

We will introduce the concept of *basic schemas* with the purpose of approximating the notion of databases with redundant relations to the real-life databases. But our main goal is to show that we can include the basic schemas in the relational databases and still our notion of redundant relations do not change.

We will use basic schemas to include on our formal setting the well known notion of relation schema in Database Theory. Roughly, in relational databases we assume the existence of four infinite and disjoint sets of symbols: the set of *attributes*, the set of *variables*, the *domain* of the database and the set of *relation names.* A *relation schema* is a relation name R with an associated finite tuple of attributes and a *database schema* σ or simply a *database*, is a finite set of relation schemas. We include here the concept of basic schemas to denote the attributes of every relation schema in a database schema. Recall for instance, Example 6 where σ is a database schema with three relation schemas, $Clients, Investments$ and $MatInv$. Particularly, $MatInv$ has three attributes associated, ClientID, InvestmentID and MaturityDate that can be seen precisely as the basic schemas of the relation schema $MatInv$ in the database schema σ.

We will use basic schemas of relations in databases as defined in [23] where they were named *basic types.* Basically, the basic schemas consist of unary relations that partition the domain of the databases. Then, each attribute of every relation in the databases is associated to exactly one of those unary relations. Let's see the definition:

Definition 15. *([23], Definition 7.3, (1)) Let* $\sigma = \langle S_1, \dots, S_t, R_1, \dots, R_s\rangle$ *be a relational vocabulary, or* schema *of a database, such that, for* $1 \leq i \leq t$ *the arity of* S_i *is 1 and, for* $1 \leq i \leq s$ *the arity of* R_i *is* $r_i \geq 1$. *We call the unary relations* S_i basic schemas.
Let $\Lambda = \{\lambda_1, \dots, \lambda_s\}$ *be the set of sentences in FO that represent the relation schemas for all the relation symbols* $R_i \in \sigma$. *For* $1 \leq i \leq s$, *we call* schema of R_i *to the sentence* λ_i *of the form:*

$$\lambda_i \equiv \forall x_1 \dots \forall x_{r_i}(R_i(x_1, \dots, x_{r_i}) \rightarrow (S_{j_1}(x_1) \wedge \dots \wedge S_{j_{r_i}}(x_{r_i})))$$

Where for $1 \leq h \leq r_i$ is $1 \leq j_h \leq t$ and sub-indexes j_h can be repeated. That is, for every relation R_i different than a basic schema, λ_i expresses the basic schema of each one of the r_i components in all tuples of the relation which interprets R_i in any given database and different components could have the same basic schema.

Note that, not all databases of schema σ with basic schemas satisfy Λ. So, in this subsection, for a given schema σ with basic schemas, we will *only* consider the databases of schema σ which satisfy Λ. Let's see an example of the use of basic schemas.

Example 10. Recall Example 6 where $\sigma = \langle Clients, Investments, MatInv \rangle$. For simplicity we rename here the relation symbols in σ as C, I and M respectively. Including the basic schemas in σ, the schema changes as follows:

$$\sigma = \langle ClientID, FirstName, LastName, Address, InvestmentID, \\ InterestRate, Term, StartDate, MaturityDate, C, I, M \rangle$$

The first nine relation symbols are the basic schemas. We rename them by CID, FN, LN, A, IID, IR, T, SD, MD respectively. The set of schemas of the relation symbols, different than the basic schemas, is:

$$\Lambda = \{\lambda_C, \lambda_I, \lambda_M\}$$

where:

$\lambda_C \equiv \forall x_1 \ldots x_4 \big(C(x_1, \ldots, x_4) \rightarrow (CID(x_1) \wedge FN(x_2) \wedge LN(x_3) \wedge A(x_4))\big)$
is the schema of $Clients$,

$\lambda_I \equiv \forall x_1, \ldots, x_6 \big(I(x_1, \ldots, x_6) \rightarrow (IID(x_1) \wedge CID(x_2) \wedge IR(x_3) \wedge T(x_4) \wedge SD(x_5) \wedge MD(x_6))\big)$
is the schema of $Investments$ and,

$\lambda_M \equiv \forall x_1 \ldots x_3 \big(M(x_1, \ldots, x_3) \rightarrow (CID(x_1) \wedge IID(x_2) \wedge MD(x_3))\big)$
is the schema of $MatInv$.

When we consider a given schema σ with basic schemas, there is an FO sentence λ_i for each relation symbol R_i in σ, for $1 \leq i \leq s$, different than the basic schemas. And this is the case even for the relation symbols which correspond to redundant relations $R_j^{\mathcal{A}}$, with $1 \leq j \leq s$, in some given database $\mathcal{A}$ of schema σ. Then, each redundant relation $R_j^{\mathcal{A}}$ in $\mathcal{A}$ satisfies its corresponding λ_j. Moreover, the database instance $\mathcal{A}$ of schema σ satisfies Λ, i.e., $\mathcal{A} \models \Lambda$.

On the other hand, by definition of types, the basic schemas in σ *do* alter the FO types of the tuples in the databases of schema σ, in the sense that basic schemas will be included in the formulas which form the FO type of tuples in databases of schema σ. Observe that the basic schemas are relation symbols in σ.

The notion of redundant relations is based on the notion of the FO types of the tuples in databases. If $R_j^{\mathcal{A}}$ of arity k in a database $\mathcal{A}$ of some schema σ is a redundant relation then the consideration of basic schemas do not necessarily modify our notion of redundant relations. The basic schemas of the relation $R_j^{\mathcal{A}}$ appear also as basic schemas for the remaining relations R_i in $\mathcal{A}|_{\sigma - R_j}$. Observe for instance the schema of the relation $MatInv$ in Example 6, where every basic schema in $MatInv$ is in $Investments$ as well. Then, by Theorem 7, there will be an FO formula $\varphi_{R_j}(x_1, \dots, x_k)$, which is the disjunction of the isolating formulas for the FO types of the k-tuples in $R_j^{\mathcal{A}}$, such that $\varphi_{R_j}^{\mathcal{A}|\sigma - R_j} = R_j^{\mathcal{A}}$. This formula will be formed by atomic formulas where the remaining relation symbols in $\mathcal{A}|_{\sigma - R_j}$ appear, as well as the corresponding basic schemas of these relations. Note that, when we eliminate a redundant relation $R_j^{\mathcal{A}}$ of arity k from a database $\mathcal{A}$ of schema σ with basic schemas, we should also eliminate the corresponding schema λ_j for the relation symbol R_j from Λ.

Definition 16. *Let $\sigma = \langle S_1, \dots, S_t, R_1, \dots, R_s \rangle$ be a schema with basic schemas, let Λ be the set of FO sentences that represent the relations schemas for all the relation symbols $R_i \in \sigma$, for $1 \leq i \leq s$ and let $R_j^{\mathcal{A}}$ be a redundant relation in a given database $\mathcal{A}$ of schema σ. We denote by $\Lambda|_{\sigma - R_j}$ the set of FO sentences that represent the relation schemas for all the relation symbols $R_i \in \sigma - R_j$ by eliminating the sentence λ_j of the corresponding relation symbol $R_j \in \sigma$.*

Then, the reduced database $\mathcal{A}|_{\sigma - R_j}$ of schema $\sigma - R_j$ clearly satisfies $\Lambda|_{\sigma - R_j}$, i.e., $\mathcal{A}|_{\sigma - R_j} \models \Lambda|_{\sigma - R_j}$.

Let's consider arbitrary queries over databases with redundant relations of some schema σ with basic schemas. Every formula φ_q of some logic which is an extension of FO, that expresses a given query q over a database $\mathcal{A}$ of schema σ will be formed by atomic formulas formed with relation symbols R_i, for some $1 \leq i \leq s$, in σ and atomic formulas of the basic schemas in R_i. Note that, if we consider *all* databases of schema σ, i.e., even those which do not satisfy Λ, the formula φ_q could include the schema λ_i of the corresponding relations R_i. As long as we can build the FO formula $\varphi_{R_j}^{\mathcal{A}|\sigma - R_j} = R_j^{\mathcal{A}}$ for every redundant relation R_j of some arity k in $\mathcal{A}$, there is a straightforward way to transform every arbitrary query

q from $\mathcal{A}$ to $\mathcal{A}|_{\sigma-R_j}$ following the procedure described in Subsection 7.1.1, provided that the logic in which φ_q is expressed, can express φ_{R_j} too.

7.2 Redundant Relations in Classes of Databases

In this section we will continue studying the notion of redundant relations but now over classes of databases. That is, we will now consider relations which are redundant in *all* databases in a given class.

Definition 17. *Given a schema $\sigma = \langle R, R_1, \ldots, R_s \rangle$ with $s > 0$, a relation symbol R of arity k, and a class $\mathcal{C}$ of databases in $\mathcal{B}_\sigma$, R is a* redundant relation symbol *in σ for the class $\mathcal{C}$ if for every $\mathcal{A}_i$ in $\mathcal{C}$ and for all k-tuples $\bar{u}$ and $\bar{v}$ over $(dom(\mathcal{A}_i))^k$,*

$$tp^{FO^\sigma}_{\mathcal{A}_i}(\bar{u}) = tp^{FO^\sigma}_{\mathcal{A}_i}(\bar{v}) \quad \textit{iff} \quad tp^{FO^{\sigma-R}}_{\mathcal{A}_i}(\bar{u}) = tp^{FO^{\sigma-R}}_{\mathcal{A}_i}(\bar{v}).$$

i.e., the equivalence classes induced by the FO^σ types of the k-tuples in $(dom(\mathcal{A}_i))^k$ coincide with the equivalence classes induced by the $FO^{\sigma-R}$ types of the k-tuples in $(dom(\mathcal{A}_i))^k$.

Note that the relation symbol $MatInv$ in Example 6 is a redundant relation symbol in σ for the whole class of databases of schema σ, $\mathcal{B}_\sigma$.

Let $\mathcal{C}$ be a class of databases of schema $\sigma = \langle R, R_1, \ldots, R_s \rangle$, with $s > 0$ and let R be a relation symbol of arity k which is a redundant relation symbol in σ for the class $\mathcal{C}$. By Theorem 7, for each $\mathcal{A}_i \in \mathcal{C}$ there is a formula $\varphi_{\mathcal{A}_i,R}$ which interpreted on $\mathcal{A}_i|_{\sigma-R}$ defines the relation $R^{\mathcal{A}_i}$, in symbols

$$\varphi_{\mathcal{A}_i,R}^{\mathcal{A}_i|_{\sigma-R}} = R^{\mathcal{A}_i}$$

Next, we will examine the relationship between every database $\mathcal{A}_i$ in the class $\mathcal{C}$ and its corresponding redundant relation $R^{\mathcal{A}_i}$.

Definition 18. *Let $\mathcal{C}$ be a class of databases of schema $\sigma = \langle R, R_1, \ldots, R_s \rangle$, with $s > 0$, and R a relation symbol of arity k which is a redundant relation symbol in σ for the class $\mathcal{C}$. We define the partial function $f_R : \mathcal{B}_\sigma \to \mathcal{B}_{\langle R \rangle}$ from the set of databases of schema σ to the set of databases of schema $\langle R \rangle$, as follows:*

$$f_R = \{(\mathcal{A}_i, R^{\mathcal{A}_i}) : \mathcal{A}_i \in \mathcal{C}\}$$

Now, given that f_R clearly preserves isomorphisms and that $dom(f_R(\mathcal{A}_i)) \subseteq dom(\mathcal{A}_i)$, for every $i \geq 1$ then, the function f_R is a query. But note that,

f_R is not necessarily a *computable* query, i.e., f_R might be a non-recursive function. Informally, it depends on whether the construction of the formulas $\varphi_{\mathcal{A}_i,R}$ for each $\mathcal{A}_i$ in the class $\mathcal{C}$ as described in Theorem 7 is an "arbitrary" construction or not.

If f_R is a recursive function then, there is a Turing Machine (see [9] for a study of recursive functions and computability) that for every database $\mathcal{A}_i$ in $\mathcal{B}_\sigma$ builds the relation $R^{\mathcal{A}_i}$ in its output tape. Then, that Turing Machine can be modified to build the isolating formulas for the FO types for the k-tuples in $R^{\mathcal{A}_i}$, by building the formulas $\varphi^m_{\bar{u}}$, as explained in Subsection 5.1, for each tuple $\bar{u}$ in $R^{\mathcal{A}_i}$, with $m = |dom(\mathcal{A}_i)|^k$ (see [10]).

Regarding Example 6 (see above) the function f_{MatInv} is clearly recursive, furthermore it is expressible in FO.

If f_R is a non-recursive function, the disjunction of isolating formula for the FO types of k-tuples in $\mathcal{A}_i$ is "arbitrary" for different databases $\mathcal{A}_i$ in $\mathcal{C}$.

In any case, note that the following formula in $\mathcal{L}_{\infty\omega}$ expresses f_R

$$\Phi_{\mathcal{C},R} \equiv \bigvee_{\mathcal{A}_i \in \mathcal{C}} (\Delta_{\mathcal{A}_i} \wedge \varphi_{\mathcal{A}_i,R})$$

where $\Delta_{\mathcal{A}_i}$ is the formula defined in Proposition 1 that characterizes the database $\mathcal{A}_i$ up to isomorphism.

The formula $\Phi_{\mathcal{C},R}$ expresses that, for every database $\mathcal{A}_i$ the formula $\varphi_{\mathcal{A}_i,R}$ must be evaluated on that database, to define the relation $R^{\mathcal{A}_i}$. Recall that $\mathcal{L}_{\infty\omega}$ can express *non-recursive* queries.

Remark 1. Note that the function f_R can be seen as a *view* in the class $\mathcal{C}$. Views are virtual relations in a class of databases whose tuples are defined by a query (typically expressed in SQL). This query can relate tuples from many other relations in the database. Once we delete $R^{\mathcal{A}_i}$ from every database $\mathcal{A}_i$ in $\mathcal{C}$, if f_R is *recursive* then it becomes a view. A view has the appearance of a relation with a set of tuples, however, a view does not exist physically as a stored set of tuples, and any change of the tuples in the relations are clearly automatically reflected in the views derived from them. Regarding the tuples in a view, if we eliminate a given view W from a class $\mathcal{C}$ we get the same situation as in redundant relations in the sense that, for each $\mathcal{A}_i \in \mathcal{C}$ there is an FO formula $\varphi_{\mathcal{A}_i,W}$ as described in Theorem 7, that when evaluated over the database $\mathcal{A}_i$ provides all the tuples of the view W evaluated in $\mathcal{A}_i$.

7.2.1 Evaluating Queries in Classes of Reduced Databases

We will see now some initial ideas towards the translation of a formula φ_q in a given logic that expresses an arbitrary query q over a class of databases of schema σ, to a formula φ'_q in the same logic that expresses the query q over the class (of schema $\sigma - R$) of the reduced databases.

The mechanism is the same as the one used in Subsection 7.1 but now we will apply it to a class of databases rather than to a single database. We consider the function f_R as described in Definition 18 for this purpose and we will consider f_R as a *recursive function.*

Let k, l be natural numbers. Given a class $\mathcal{C}$ of databases of schema σ and R a redundant relation symbol in σ for $\mathcal{C}$, let's denote by $\varphi_{\mathcal{C},R}(\bar{x}_k)$ the formula, in some logic, that expresses f_R and by $\varphi_{\mathcal{C},q}(\bar{x}_l)$, the formula that expresses an arbitrary l-ary query q over the class $\mathcal{C}$. We will next consider a few cases as to illustrate the general strategy.

1. If $\varphi_{\mathcal{C},R}(\bar{x}_k) \in FO$ and $\varphi_{\mathcal{C},q}(\bar{x}_l) \in FO$.
   It is handled as in Subsection 7.2. The difference is that, now every time that an atomic formula with the relation symbol R appears in the formula $\varphi_{\mathcal{C},q}(\bar{x}_l)$ we will replace it by the formula $\varphi_{\mathcal{C},R}(\bar{x}_k)$ with the appropriate renaming of variables.
2. If $\varphi_{\mathcal{C},R}(\bar{x}_k) \in FO$ and $\varphi_{\mathcal{C},q}(\bar{x}_l) \in SO$.
   Basically, second order logic (shortly, SO) is an extension of first-order logic which allows us in addition to FO, to quantify variables over relations instead of ranging over elements of the domain of the database. For more details on SO see [10, 16].
   This case is handled in a similar manner as in the previous one. In some places in the formula $\varphi_{\mathcal{C},q}(\bar{x}_l)$ there will be SO atoms of the form $R(t_1, \ldots, t_k)$ and then we can replace them by the formula $\varphi_{\mathcal{C},R}(\bar{x}_k)$ with the appropriate renaming of variables. This is possible because SO includes FO.

Let's consider the following example:

Example 11. Let $\tau = \langle E, R, B, Y \rangle$ be the vocabulary of graphs in a class $\mathcal{C} \in \mathcal{B}_\tau$ whose nodes can be colored in three different colors R, B, and Y. Let $\mathcal{G}$ be a tree in the class $\mathcal{C}$ as in Figure 11 where the colored nodes are marked by the letters r, b and y respectively.

Clearly, the relation B is a redundant relation in $\mathcal{G}$ but this is not the case for relations Y and R.
Let's consider the following query over the class $\mathcal{C}$:

"The number of nodes whose parent has color B is the same as the number of nodes whose parent has color Y."

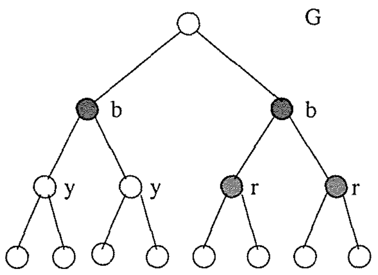


Figure 11

This is a computable query that cannot be expressed in first-order logic. It is well known that it is not possible to count beyond a constant in first-order logic. Instead, it can be expressed in SO by the following formula:

$$\begin{aligned}\exists XY\Big(&\forall x\big(X(x) \leftrightarrow \exists y(E(y,x) \wedge B(y))\big)\wedge\\ &\forall x\big(Y(x) \leftrightarrow \exists y(E(y,x) \wedge Y(y))\big)\wedge\\ &\exists F(\text{"}F \text{ is a bijection from } X \text{ to } Y\text{"})\Big)\end{aligned}$$

where "F is a bijection from X to Y" is expressed by,

$$\begin{aligned}\forall x,y,z\big((F(x,y) \rightarrow X(x) \wedge Y(y)) &\qquad \text{"}F \subseteq X \times Y\text{"}\\ \wedge\\ (F(x,y) \wedge F(x,z) \rightarrow y = z) &\qquad \text{"}F \text{ is a function"}\\ \wedge\\ (X(x) \rightarrow \exists y(F(x,y))) &\qquad \text{"}F \text{ is total"}\\ \wedge\\ (F(x,z) \wedge F(y,z) \rightarrow x = y) &\qquad \text{"}F \text{ is injective"}\\ \wedge\\ (Y(y) \rightarrow \exists x(F(x,y)))\big) &\qquad \text{"}R \text{ is surjective"}\end{aligned}$$

Note that $F : X \rightarrow Y$ is a bijection iff X and Y contain the same number of elements.
Clearly, the sub-formula $B(y)$ in the previous SO formula can be replaced in $\mathcal{G}$ by the FO formula:

$$\varphi_{\mathcal{G},B}(y) \equiv \exists x\big(E(x,y) \wedge \forall z(\neg E(z,x))\big)$$

Now, suppose that all the trees in the class $\mathcal{C}$ are colored in such a way that the nodes with color B are exactly the children of the root node. Then B becomes a redundant relation symbol in τ for $\mathcal{C}$, and furthermore it is $\varphi_{C,B} \equiv \varphi_{G,B}$.

3. If $\varphi_{C,R}(\bar{x}_k) \in FO$ and $\varphi_{C,q}(\bar{x}_l) \in HO^i$ where $i > 2$.
   The translation is also possible in this case, since higher order logic of order i includes FO.

The question that now follows is: What happens if $\varphi_{C,R}(\bar{x}_k)$ is not in FO? As long as the logic in which $\varphi_{C,q}$ is written extends the logic in which $\varphi_{C,R}$ is written the translation should be possible too, but it is out of the objective of this work.

Depending on the complexity of the query q (see [16, 3, 19] for details in Complexity), that is, the amount of computational resources such as time and space, that the query needs to be computed, will be the logic in which q needs to be expressed. Moreover, we have been working with f_R as a computable query that can be expressed by an FO formula. But, it is well known that first-order logic is not complete, that is, it does not express the whole class of computable queries. Instead in the general case, f_R could be expressed in a complete logic such as VO or $\mathcal{L}_*$.

The *variable order logic* (VO) introduced in [15], is a very powerful logic, indeed it is complete, that is, it can express all computable queries. VO is a two sorted Higher Order Logic in which relation variables are untyped, i.e., variables have no associated order. Instead, an order is assigned to a given variable when the variable is quantified in a formula, and this is done by means of a set of variables of a second sort, namely *order variables*. The domain of the second sort is the subset of the natural numbers greater than 1.

The infinitary logic $\mathcal{L}_*$ is a fragment of the well known logic $\mathcal{L}_{\omega 1 \omega}$. It was demonstrated in [22] that $\mathcal{L}_*$ captures exactly the whole class CQ.

We conclude that in general, *if we know* the formula $\varphi_{C,R}(\bar{x}_k)$ (i.e., if we have a finite representation of it in a language of decidable syntax), we can construct an algorithm that translates the formula $\varphi_{C,q}(\bar{x}_l)$ which expresses an arbitrary query q in a given logic in a class C of schema σ to a formula that expresses the same query q in the class of the reduced databases of schema $\sigma - R$. This can be done provided that the logic in which f_R is expressed is at most as expressive as the logic in which $\varphi_{C,q}(\bar{x}_l)$ is expressed.

We have been working with graph structures in order to explain the redundant relations and the translation of arbitrary queries in databases given that it is easier to visualize structural properties on them. Nevertheless those concepts can be easily extended to every finite database of a relational schema.

7.2.2 Classes of Databases with Basic Schemas

We will extend the notion of *basic schemas* described in Subsection 7.1.2 to a class of databases of some schema σ with basic schemas. We will see that it is not necessary to modify the function f_R as described in Definition 18 which represents the relationship between every database $\mathcal{A}_i$ in a given class $\mathcal{C}$ of some schema σ and its corresponding redundant relation $R^{\mathcal{A}_i}$.

Recall Definition 15 of basic schemas. Considering a class $\mathcal{C} \subseteq \mathcal{B}_\sigma$ of databases of schema σ with basic schemas, and every database $\mathcal{A}_i \in \mathcal{C}$, we can find two different situations:

(i) $\mathcal{A}_i \in \mathcal{B}_\sigma$, or
(ii) $\mathcal{A}_i \in \mathcal{B}_\sigma$ and $\mathcal{A}_i \models \Lambda$.

That is, in (i), every database $\mathcal{A}_i$ in the class $\mathcal{C}$ is a database of schema σ or, in (ii), in addition to this condition, every database $\mathcal{A}_i$ also satisfies the set of sentences in Λ that specify the basic schemas of every relation symbol R_j, for $1 \leq j \leq s$, in σ.

Take for instance, the schema σ in Example 10 and consider a class $\mathcal{C}$ of databases of schema σ. If we consider (ii), every database $\mathcal{A}_i \in \mathcal{C}$ will be obviously in $\mathcal{B}_\sigma$ and also $\mathcal{A}_i \models \Lambda$. That is, for every tuple in the relation $MatInv^{\mathcal{A}_i}$ in each database $\mathcal{A}_i$ in the class $\mathcal{C}$, the first component will be ClientID, the second component will be InvestmentID and the third component will be MaturityDate.

If we consider (i), every database $\mathcal{A}_i \in \mathcal{C}$ will be obviously in $\mathcal{B}_\sigma$ and for every relation symbol R_j, for some $1 \leq j \leq s$, in σ there will be a relation in the database $\mathcal{A}_i$ that interprets R_j. But $\mathcal{A}_i$ does not necessarily satisfy the set of sentences in Λ that specify the basic schemas of every relation symbol in σ, and hence the first component in a tuple in $MatInv^{\mathcal{A}_i}$ might be an arbitrary element in $dom(\mathcal{A}_i)$, like a StartDate or a Term, instead of a ClientID.

Regarding redundant relation symbols for a class $\mathcal{C}$ of databases of some schema σ with basic schemas we will consider the situation (ii) where every database in the class satisfies Λ. Then, Definition 17 does not change, we just add the basic schemas and we have the following:

Definition 19. *Given a schema $\sigma = \langle S_1, \ldots, S_t, R, R_1, \ldots, R_s \rangle$ with basic schemas $S_1, \ldots, S_t$ where $s, t > 0$, a relation symbol R of arity k, and a class $\mathcal{C}$ of databases in $\mathcal{B}_\sigma$, R is a* redundant relation symbol *in σ for* *the class $\mathcal{C}$ if for every $\mathcal{A}_i$ in $\mathcal{C}$, $\mathcal{A}_i \models \Lambda$ and for all k-tuples $\bar{u}$ and $\bar{v}$ over $(dom(\mathcal{A}_i))^k$,*

$$tp_{\mathcal{A}_i}^{FO^{\sigma}}(\bar{u}) = tp_{\mathcal{A}_i}^{FO^{\sigma}}(\bar{v}) \quad \textit{iff} \quad tp_{\mathcal{A}_i}^{FO^{\sigma-R}}(\bar{u}) = tp_{\mathcal{A}_i}^{FO^{\sigma-R}}(\bar{v}).$$

i.e., the equivalence classes induced by the FO^{σ} types of the k-tuples in $(dom(\mathcal{A}_i))^k$ coincide with the equivalence classes induced by the $FO^{\sigma-R}$ types of the k-tuples in $(dom(\mathcal{A}_i))^k$.

Moreover, the relationship between every database $\mathcal{A}_i$ in the class $\mathcal{C}$ and its corresponding redundant relation $R^{\mathcal{A}_i}$ is a query, but, as we saw in Subsection 7.2, it is not necessarily a computable query. That is, the construction of the formulas $\varphi_{\mathcal{A}_i,R}$ for each reduced database $\mathcal{A}_i|_{\sigma-R}$ of $\mathcal{A}_i$ in the class $\mathcal{C}$ as described in Theorem 7 might be an arbitrary construction. Furthermore, in this case, $\varphi_{\mathcal{A}_i,R}$ includes the atomic formulas of the basic schemas as well.

Finally, if we consider f_R as a recursive function considering schemas with basic schemas, it is clearly possible to translate any formula φ_q that expresses an arbitrary query q over a class $\mathcal{C}$ of databases of schema σ with basic schemas to a formula φ'_q in the same logic that expresses the query q over the class (of schema $\sigma - R$ with basic schemas) of the reduced databases, such that every $\mathcal{A}_i|_{\sigma-R} \models \Lambda|_{\sigma-R}$. This is possible following the mechanism used in Subsection 7.2 and under the same conditions explained there.

Even if we want to consider the whole class $\mathcal{B}_{\sigma}$ (i.e., not only those databases which satisfy $\Lambda|_{\sigma-R}$), such translation can be done, provided that the logic in which φ_q is written is at least as expressible as first-order logic, since the formula φ_q includes the basic schemas as well, and these are expressed in first-order logic.

8 Conclusion and Future Work

By Theorem 8, the problem of deciding whether a given relation $R^{\mathcal{A}}$ of arity k is a redundant relation in a fixed database $\mathcal{A}$ of some schema σ, is decidable. Unfortunately, it is very unlikely that there is a polynomial time algorithm for this problem since it is equivalent to deciding isomorphism. Possible alternatives to overcome in part this problem could consist on restricting,

a. the *class of queries* to a sub-class of $\mathcal{CQ}$ in such a way that determining whether a relation is redundant regarding only such sub-class of queries is in $PTIME$.
b. the *class of databases* to classes where deciding whether a relation is redundant in a database which belongs to the class is in $PTIME$.

Note that we study here the notion of type in FO and in FO^k, as well as in their corresponding infinitary extensions $\mathcal{L}_{\infty\omega}$ and $\mathcal{L}^k_{\infty\omega}$, respectively. The FO^k types ($\mathcal{L}^k_{\infty\omega}$ types) are quite relevant in database theory since they characterize the discerning power of the class of reflective relational machines of [2] with variable complexity k. Among other possible continuations of this work, we think in defining the notion of redundant relation with respect to a given language $\mathcal{L}$. That is, a relation R in a database $\mathcal{A}$ of schema σ would be $\mathcal{L}$-redundant if for every query q expressable in $\mathcal{L}$ it is $q(\mathcal{A}) = q(\langle \mathcal{A}|_{\sigma-R}, \varphi_R^{\mathcal{A}|_{\sigma-R}} \rangle)$, for some formula φ_R. In particular, if we take $\mathcal{L} = RRM^k$ (the reflective relational machines of variable complexity k [2]) such a notion is defined by considering FO^k types instead of FO types in Definition 14. It is interesting to note that by a result of Grohe [14], equivalence in FO^k is complete for polynomial time. Then we can check in $PTIME$, FO^k equivalence between every two extensions of a database with any given pair of tuples.

Independently of the complexity of deciding whether a given relation is redundant in a given database, the complexity of building the formula φ_R defined in Theorem 7 for a given relation $R^{\mathcal{A}}$, is in general exponential in time. Nevertheless, if we know that a relation $R^{\mathcal{A}}$ is redundant in a given database $\mathcal{A}$, then each of the corresponding isolating formulas $\psi_i(x_1, \ldots, x_k)$ for the types of the tuples in $R^{\mathcal{A}}$, can be built in polynomial time, since $R^{\mathcal{A}}$ has complete FO types for k-tuples. This can be done by using a construction similar to the construction of the diagram of a database. As this construction takes polynomial time, the construction of φ_R will also take polynomial time.

Regarding classes of databases, the problem of deciding whether a given relation R is a redundant relation in a given class of databases of some relational schema is clearly not decidable in the general case.

In this direction we think as a possible future work the development of a method of normalization for databases with redundant relations, in the spirit of the well known normal forms in databases (see [1, 20]). In this sense, we aim to develop an algorithm, based in the concept of isolating formulas for the FO type of tuples in databases, that allows us to determine all possible redundant relations in a given database. For classes of databases, we will study sub-classes of databases of a given schema where given a relation symbol in the schema we can determine whether this relation symbol is a redundant relation symbol for the whole given sub-class of databases.

References

1. Abiteboul, S., Hull, R. and Vianu, V.: *Foundations of Databases.* Addison-Wesley, 1994.
2. Abiteboul, S., Papadimitriou, C. and Vianu, V.: *Reflective Relational Machines.* Information and Computation 143, pp.110-136, 1998.
3. Balcazar, J. L., Diaz, J., Gabarro, J.: *Structural Complexity I.* Springer, 1995.
4. Berge, C: *Graphs.* Elsevier North-Holland, 1985.
5. Chandra, A. K., Harel, D.: *Computable Queries for Relational Data Bases.* Journal of Computer and System Sciences 21(2), pp.156-178, 1980.
6. Chang, C, Kreisler, H.: *Model Theory.* Elsevier North-Holland, 3rd. ed., 1992.
7. Codd, E.: *A Relational Model Of Data for Large Shared Data Banks.* Communications of ACM 13, 6, 1970.
8. Dawar, A.: *Feasible Computation Through Model Theory.* Ph.D. thesis, University of Pennsylvania, Philadelphia, 1993.
9. Davis, M. D., Sigal, R., Weyuker, E. J.: *Computability, Complexity, and Languages.* Academic Press, 1994.
10. Ebbinghaus, H. D., Flum, J.: *Finite Model Theory.* Springer-Verlag Berlin Heidelberg, New York, 2nd. ed., 1999.
11. Ebbinghaus, H. D., Flum, J., Thomas, W.: *Mathematical Logic.* Springer, 1984.
12. Ehnrenfeucht, A.: *An Application of games to the Completeness Problem for Formalized Theories.* Fund. Math. 49 (1961), 129-141.
13. Fraïssé, R.: *Sur les Classifications des Systems de Relations.* Publ. Sci. Univ. Alger I (1954).
14. 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.
15. Hella, L., Turull-Torres, J.M.: *Expressibility of Higher Order Logics.* Electronic Notes in Theoretical Computer Science, Vol. 84, 2003.
16. Immerman, N.: *Descriptive Complexity.* Springer, 1999.
17. Libkin, L.: *Elements of Finite Model Theory.* Springer, 2004.
18. Otto, M.: *Bounded Variable Logics and Counting.* Springer (1997).
19. Papadimitriou, C. H.: *Computational Complexity* Addison-Wesley, 1994.
20. Ramakrishnan, R., Gehrke, J.: *Database Management Systems.* McGraw-Hill, 3rd. ed., 2003.
21. Rosen, K. H.: *Discrete Mathematics and its Applications.* McGraw-Hill, 5th. ed., 2003.
22. Turull-Torres, J.M.: *On the Expressibility and the Computability of Untyped Queries.* Annals of Pure and Applied Logic 108, pp. 345-371, 2001.
23. Turull Torres J.M.: *L-rigid Databases and the Expressibility of Incomplete Relational Languages.* Doctoral Thesis, Universidad Nacional de San Luis, 1996.
24. Ullman, J. D.: *Principles of Database and Knowledge Base Systems.* Volume I and II. Computer Science Press, 1988.