Ordering Finite Variable Types with Generalized Quantifiers

Anuj Dawar*
Dept. of Comp. Science
Univ. of Wales, Swansea
Swansea SA2 8PP, U.K.
a.dawar@swansea.ac.uk

Lauri Hella
Dept. of Mathematics
P.O. Box 4
Univ. of Helsinki, Finland
hella@cc.helsinki.fi

Anil Seth
The Inst. of Math. Sciences
C.I.T. Campus, Taramani
Madras 600113, India
seth@imsc.ernet.in

Abstract

Let $\mathbf{Q}$ *be a finite set of generalized quantifiers. By* $L^k(\mathbf{Q})$ *we denote the* k*-variable fragment of* $\mathrm{FO}(\mathbf{Q})$*, first order logic extended with* $\mathbf{Q}$*. We show that for each* k*, there is a* $\mathrm{PFP}(\mathbf{Q})$*-definable linear pre-order whose equivalence classes in any finite structure* $\mathfrak{A}$ *are the* $L^k(\mathbf{Q})$*-types in* $\mathfrak{A}$*. For some special classes of generalized quantifiers* $\mathbf{Q}$*, we show that such an ordering of* $L^k(\mathbf{Q})$*-types is already definable in* $\mathrm{IFP}(\mathbf{Q})$*. As applications of the above results, we prove some generalizations of the Abiteboul-Vianu theorem. For instance, we show that for any finite set* $\mathbf{Q}$ *of modular counting quantifiers,* $\mathrm{P} = \mathrm{PSPACE}$ *if, and only if,* $\mathrm{IFP}(\mathbf{Q}) = \mathrm{PFP}(\mathbf{Q})$ *over finite structures.*

On the other hand, we show that an ordering of $L^k(\mathbf{Q})$*-types is not always definable in* $\mathrm{IFP}(\mathbf{Q})$*. Indeed, we construct a single, polynomial time computable quantifier* P *such that the equivalence relation* $\equiv^{k,P}$*, and hence ordering on* $L^k(P)$*-types, is not definable in* $\mathrm{IFP}(P)$*.*

1 Introduction

Since Chandra and Harel [8] first posed the question of the existence of a language that could express exactly the polynomial time generic queries on a relational database, much work in finite model theory has been devoted to this problem. Immerman [18] and Vardi [26] proved that the extension of first order logic with an operator for forming the least fixed points of positive formulas (LFP) captures exactly the polynomial time computable queries on an *ordered* relational structure. By a result of Gurevich and Shelah [15], LFP is equivalent in expressive power to an inflationary fixed point logic (IFP). Attempts to extend such results in order to obtain a logic that can express exactly the polynomial time properties over arbitrary finite relational structures (or to show that no such logic is possible) have produced two distinct research directions. One is a closer analysis of the expressive power of IFP and related logics on finite structures, and the other is the study of extensions of IFP, for instance by counting mechanisms and generalized quantifiers.

The former direction has resulted in a clear understanding of IFP in terms of finite variable types. That is, the queries definable in IFP are exactly those that are computable in polynomial time on a reduced structure which is obtained from a structure $\mathfrak{A}$ by taking its quotient under a k variable equivalence relation $\equiv^k$, for some k. Similarly, the partial fixed point logic PFP defines exactly those queries that are computable in polynomial space over the reduced structures. Such an analysis led Abiteboul and Vianu [4] to the theorem to the effect that IFP and PFP have equal expressive power if, and only if, the complexity classes P and PSPACE co-incide. This was later extended to a large number of other complexity classes [1, 10]. For instance, it was shown in [10] that a restricted existential second order logic $\Sigma_1^{1,\omega}$ is equivalent to IFP if, and only if, P = NP. Crucial to a proof of such results is the fact that the k variable equivalence relation is itself an IFP-definable query, and moreover there is an IFP-formula that orders the equivalence classes induced by this relation (see [12]).

In the second direction, a variety of extensions of IFP have been considered in the literature. Cai *et al.* [6] showed that there is a polynomial time query that is not definable in an extension of IFP with a counting mechanism. The proof of this essentially shows that there is a polynoimial time query that is not closed under $\equiv^{k,\mathbf{C}}$, the equivalence relation for k variable logic with counting quantifiers, for any k. This result was generalized by Hella [16] who showed that there is a polynomial time graph property that is not closed under the relation $\equiv^{k,\mathbf{Q}}$ for any k, where $\mathbf{Q}$ is the collection of all unary Lindström quantifiers, or more generally, for any collection of quantifiers $\mathbf{Q}$ of bounded arity, there is a vocabulary and a polyno-

*Supported by EPSRC grant GR/L69596.

mial time query in that vocabulary that is not closed under $\equiv^{k,\mathbf{Q}}$ for any k. A related result was proved in [11], where it was shown that for any finite set of polynomial time computable quantifiers $\mathbf{Q}$, there is a polynomial time computable graph property that is not expressible in IFP($\mathbf{Q}$).

Results such as these, which depend on the equivalence relations $\equiv^{k,\mathbf{Q}}$ for various k and $\mathbf{Q}$ raise the question to what extent can the analysis of fixed point logics in terms of finite variable logics be extended to the situation where generalized quantifiers are present. In particular, is it the case that for any quantifier Q and any k, there is a formula of IFP(Q) which defines the relation $\equiv^{k,Q}$? Moreover, is there a formula of IFP(Q) that can order the corresponding equivalence classes? We show in this paper that there are quantifiers for which this is not true. Indeed, we exhibit a polynomial time computable quantifier for which this fails. Nonetheless, the relation $\equiv^{k,Q}$ and an ordering of its equivalence classes are always definable in PFP(Q), and even in SO$^\omega$(Q), where SO$^\omega$ is the restricted second order logic defined in [10]. This result also establishes therefore that there are quantifiers Q such that the logics IFP(Q) and $\Sigma_1^{1,\omega}(Q)$ are distinct. This can be seen as a separation of the logics IFP and $\Sigma_1^{1,\omega}$ in a relativized world, analogous to the separation of P from NP by an oracle. What is remarkable is that the separation is achieved by a single *polynomial time* quantifier, so that the logics that have been separated, IFP(Q) and $\Sigma_1^{1,\omega}(Q)$, are actually contained within the respective complexity classes P and NP.

On the other hand, by analysing our proof for ordering $L^k(\mathbf{Q})$-types in PFP($\mathbf{Q}$) we do obtain a number of positive results about the definability of the relations $\equiv^{k,Q}$ in IFP(Q). For instance, we establish this for some specific quantifiers, such as modular counting quantifiers and bounded monotone quantifiers. Using this methodology we can also derive in a uniform way already known results about ordering L^k-types and $L^k(\mathbf{C})$-types, where $\mathbf{C}$ is the set of all unary counting quantifiers, in IFP and the extension of IFP with the Rescher quantifier, respectively.

In some cases where $L^k(\mathbf{Q})$-types can be ordered in IFP($\mathbf{Q}$), we can relate the separation of IFP($\mathbf{Q}$) and PFP($\mathbf{Q}$) to the separation of P and PSPACE. For instance, we show that if $\mathbf{Q}$ is a 'nice' finite set of modular counting quantifiers then IFP($\mathbf{Q}$) $\neq$ PFP($\mathbf{Q}$) implies P $\neq$ PSPACE. Moreover, if all the modular counting quantifiers in $\mathbf{Q}$ are polynomial time computable, then we get that IFP($\mathbf{Q}$) = PFP($\mathbf{Q}$) *if, and only if,* P = PSPACE. Of course, this is the case for the ordinary counting modulo quantifiers "there are j mod m elements", but we use the term 'modular counting quantifier' in a wider sense which allows non-unary and even non-recursive quantifiers. The same Abiteboul-Vianu type results are also shown to hold for any finite set $\mathbf{Q}$ of bounded monotone quantifiers (see Section 4.4 for a definition). However, the proofs of these results exploit not just the IFP($\mathbf{Q}$) definability of an ordering of $L^k(\mathbf{Q})$-types but they also rely on some other properties of $\mathbf{Q}$. For instance, one step in proving these results, using the methods of [12], is to construct a quotient structure by collapsing $L^k(\mathbf{Q})$-types to a single point. This step can be carried out successfully only if a polynomial amount of information with respect to the number of $L^k(\mathbf{Q})$-types is sufficient to encode each quantifier $Q \in \mathbf{Q}$ in the quotient structure. For an arbitrary quantifier, it seems that the amount of information needed may sometimes be exponential.

The rest of this paper is organised as follows. Section 2 presents the necessary background and notation, including the definitions of fixed point logics, finite variable logics and generalized quantifiers. Section 3 shows that ordering on $L^k(\mathbf{Q})$-types is always definable in PFP($\mathbf{Q}$) and SO$^\omega$($\mathbf{Q}$). Section 4 gives some examples of quantifier sets $\mathbf{Q}$ such that $L^k(\mathbf{Q})$-types can be ordered in IFP($\mathbf{Q}$). Section 5 constructs the polynomial time quantifier P for which the relation $\equiv^{k,P}$ is not definable in IFP(P). In Section 6 some generalizations of the Abiteboul-Vianu theorem for specific quantifier sets are established, and Section 7 contains some further applications of the definability results proved in the paper.

2 Background

We assume the standard definitions of a first order vocabulary and a structure interpreting it. All vocabularies we consider are finite and relational, and unless otherwise stated, all structures are finite. An (m-ary) *query* q (also sometimes called a global relation) is a map from structures (over some fixed vocabulary σ) to (m-ary) relations on the structures, that is closed under isomorphism. A 0-ary query is also called a *Boolean query.* Thus a first order formula with up to m free variables defines in a natural way an m-ary query and a sentence defines a Boolean query.

FO denotes first order logic. In general a logic L (such as FO or IFP($\mathbf{Q}$)) denotes both a set of formulas as well as the class of queries definable in that logic.

2.1 Fixed Point Logics

By closing first order logic under an operator for forming the least fixed points of positive formulas, we obtain the logic LFP. This turns out to be equivalent in expressive power (over finite structures) to the logic IFP which allows instead the formation of inflationary fixed points of arbitrary operators (see [15]). Moreover IFP is more robust in that it can be naturally extended with arbitrary generalized quantifiers. For this reason, in the present paper we always use IFP rather than LFP. Formally, we define the logic as follows.

IFP is the logic obtained by closing first order logic under the operation of forming inflationary fixed points, defined as follows: If φ is a formula of IFP, then so is $\mathbf{ifp}_{S,\vec{x}}\varphi(\vec{t})$, where

- S is a relation symbol,
- $\vec{x}$ a tuple of first order variables whose length is equal to the arity of S; and
- $\vec{t}$ is a tuple of terms whose length is equal to the arity of S.

The operator **ifp** binds the relation symbol S and all occurrences of the variables in $\vec{x}$ that appear in φ, but all occurrences of these variables in $\vec{t}$ remain free.

PFP is the similar logic where in place of the operator **ifp** we have an operator **pfp**, with a similar syntax.

The semantics of IFP is given by the rule: $\mathfrak{A} \models \mathbf{ifp}_{S,\vec{x}}\varphi(\vec{t})$ if the tuple $\vec{t}^{\mathfrak{A}}$ is in the relation that is the limit of the sequence

$$\varphi^0 = \emptyset$$

$$\varphi^{i+1} = \varphi^i \cup \varphi(\varphi^i)$$

where $\varphi(\varphi^i)$ is the relation defined in $\mathfrak{A}$ by φ when S is interpreted as φ^i.

For the semantics of PFP, we take instead the sequence

$$\varphi^0 = \emptyset$$

$$\varphi^{i+1} = \varphi(\varphi^i)$$

and let $\mathbf{pfp}_{S,\vec{x}}\varphi(\vec{t})$ be true if, and only if, this sequence converges to a limit and the tuple denoted by $\vec{t}$ is in the relation that is the limit of this sequence.

These logics are also closed under a stronger iteration scheme defined by simultaneous iteration of a sequence of formulas as follows. Let $\Phi = (\varphi_1, \ldots, \varphi_m)$ be a tuple of formulas in the vocabulary $\sigma \cup \{S_1, \ldots, S_m\}$, where each S_i is an n_i-ary relation symbol, and let $\vec{x}_i$ be an n_i-tuple of variables for $1 \leq i \leq m$. Then $\mathbf{ifp}^j{}_{S_1\vec{x}_1,\ldots,S_m\vec{x}_m}(\varphi_1, \ldots, \varphi_m)(\vec{t})$ is a formula for each j, $1 \leq j \leq m$, and each n_j-tuple $\vec{t}$ of terms. The simultaneous fixed point operator $\mathbf{ifp}^j$ binds the occurrencies of the relation symbol S_i and the variables in $\vec{x}_i$ only in the formula φ_i; all other occurrencies of them remain free.

The semantics of the operator $\mathbf{ifp}^j$ is based on the sequence $\Phi^i = (\varphi_1^i, \ldots, \varphi_m^i)$, $i < \omega$, of m-tuples of relations, where for each l, $1 \leq l \leq m$,

$$\varphi_l^0 = \emptyset$$

and

$$\varphi_l^{i+1} = \varphi_l^i \cup \varphi_l(\varphi_1^i, \ldots, \varphi_m^i).$$

The limit of this sequence is denoted by

$$\Phi^\infty = (\varphi_1^\infty, \ldots, \varphi_m^\infty).$$

We let $\mathfrak{A} \models \mathbf{ifp}^j{}_{S_1\vec{x}_1,\ldots,S_m\vec{x}_m}(\varphi_1, \ldots, \varphi_m)(\vec{t})$ if, and only if, $\vec{t}^{\mathfrak{A}} \in \varphi_j^\infty(\mathfrak{A})$.

The syntax and semantics of simultaneous PFP-iteration of a tuple of formulas is defined in an analogous fashion. For more details, we refer the reader to [13, Chapter 7].

The interest of these fixed point logics in finite model theory is largely due to the results that show that their expressive power corresponds to natural computational complexity classes. This correspondence requires that we restrict our attention to structures which contain a distinguished relation that is always a total order of the universe. In particular, Immerman and Vardi independently showed:

Theorem 1 ([18, 26]) *On ordered structures* LFP = P.

Similarly, Abiteboul and Vianu [2] showed that the expressive power of PFP is equivalent to that of a *while* language introduced by Chandra and Harel [7], which Vardi [26] proved captures PSPACE on ordered structures.

Theorem 2 ([2]) *On ordered structures* PFP = PSPACE.

In the absence of order, these logics are too weak, i.e., everything that can be expressed in IFP is polynomial time computable, but the converse does not hold, and similarly for PFP and PSPACE.

2.2 Finite Variable Logics

Finite variable logics turn out to be a very useful tool for analysing the expressive power of fixed point logics. We review them here briefly.

Let L^k denote the fragment of first order logic in which we are only allowed the variables $x_1, \ldots, x_k$. We write $\mathfrak{A} \equiv^k \mathfrak{B}$ to denote that the two structures agree on all sentences of L^k. Similarly, if $\vec{a} \in |\mathfrak{A}|^l$ and $\vec{b} \in |\mathfrak{B}|^l$ $(l \leq k)$, we write $(\mathfrak{A}, \vec{a}) \equiv^k (\mathfrak{B}, \vec{b})$ to denote that for any formula $\varphi \in L^k$, $\mathfrak{A} \models \varphi[\vec{a}]$ if, and only if, $\mathfrak{B} \models \varphi[\vec{b}]$. Here $|\mathfrak{A}|$ denotes the universe of the structure $\mathfrak{A}$.

What makes these relations useful in studying fixed point logics is the following fact:

Theorem 3 ([21]) *For any formula φ of* IFP *or* PFP*, there is a k such that if $(\mathfrak{A}, \vec{a}) \equiv^k (\mathfrak{B}, \vec{b})$ then $\mathfrak{A} \models \varphi[\vec{a}]$ if, and only if, $\mathfrak{B} \models \varphi[\vec{b}]$.*

Moreover, the equivalence relations $\equiv^k$ have an elegant characterisation in terms of a two player pebble game (essentially due to Barwise [5]). The game board consists of two structures $\mathfrak{A}$ and $\mathfrak{B}$ and a supply of k pairs of pebbles $(p_i, q_i), 1 \leq i \leq k$. The pebbles $p_1, \ldots, p_l$ are initially placed on the elements of an l-tuple $\vec{a}$ of elements in $\mathfrak{A}$, and the pebbles $q_1, \ldots, q_l$ on a tuple $\vec{b}$ in $\mathfrak{B}$. There are two players, Spoiler and Duplicator. At each move of the game, Spoiler picks up a pebble (either an unused pebble or one that is already on the board) and places it on an element of the corresponding structure. For instance he[1] might take pebble q_i and place it on an element of $\mathfrak{B}$. Duplicator must respond by placing the other pebble of the pair in the other structure. In the above example, she must place p_i on an element of $\mathfrak{A}$. If at the end of the move the partial map $f : \mathfrak{A} \to \mathfrak{B}$ given by $p_i \mapsto q_i$ is not a partial isomorphism, then the game is over and Spoiler wins, otherwise it continues for another move.

The result that links this game with the equivalence relation $\equiv^k$ is the following:

> If Duplicator has a strategy for playing the k-pebble game for infinitely many moves (for d moves) without losing starting with the position $(\mathfrak{A}, \vec{a}), (\mathfrak{B}, \vec{b})$, then $(\mathfrak{A}, \vec{a}) \equiv^k (\mathfrak{B}, \vec{b})$ ($(\mathfrak{A}, \vec{a}) \equiv^k_d (\mathfrak{B}, \vec{b})$, respectively).

Here we write $(\mathfrak{A}, \vec{a}) \equiv^k_d (\mathfrak{B}, \vec{b})$ to indicate that $(\mathfrak{A}, \vec{a})$ and $(\mathfrak{B}, \vec{b})$ cannot be distinguished by any formula of $L^{k,d}$, the fragment of L^k with quantifier rank d or less.

2.3 Defining Equivalence and Order

The inductive definition of the equivalence relation $\equiv^k$ provided by the pebble game indicates that this relation is itself definable in IFP. Indeed, such a definition of $\equiv^k$ as a query was obtained in [20, 12]:

[1] By convention, Spoiler is male and Duplicator female.

Theorem 4 *In every vocabulary, there is an* IFP*-formula γ such that on any structure $\mathfrak{A}$, $\mathfrak{A} \models \gamma[\vec{a}, \vec{b}]$ if, and only if, $(\mathfrak{A}, \vec{a}) \equiv^k (\mathfrak{A}, \vec{b})$.*

Moreover, we can order the $\equiv^k$ equivalence classes. That is, there is an IFP-formula that on any structure defines a linear pre-order which is a total ordering of the equivalence classes under $\equiv^k$.

Theorem 5 ([4, 12]) *In every vocabulary, there is a formula δ of* IFP*, with $2k$ free variables, such that on any finite structure $\mathfrak{A}$, δ defines an irreflexive and transitive relation $<^k$ on k-tuples such that for every two k-tuples $\vec{a}$ and $\vec{b}$, either $\vec{a} <^k \vec{b}$ or $\vec{a} <^k \vec{b}$ or $(\mathfrak{A}, \vec{a}) \equiv^k (\mathfrak{A}, \vec{b})$.*

A proof of this result is sketched in Section 4.1.

Theorem 5 allows us to construct an IFP-definable interpretation (re-defining equality) from arbitrary structures to ordered structures, preserving the meaning of formulas of L^k, which is used to establish the celebrated theorem of Abiteboul and Vianu.

Theorem 6 ([3, 4]) IFP = PFP *if, and only if,* P = PSPACE.

Analogous results can be reproduced for many other complexity classes above P. In fact one can define a relational measure of complexity, such that two complexity classes above P are equivalent if, and only if, their relational counterparts are [1]. In this scheme, IFP is exactly relational polynomial time and PFP is relational polynomial space. Below we exhibit one characterization of the relational polynomial hierarchy.

2.4 Restricted Second Order Logic

The following definition from [10], gives a restricted form of second order quantifier, which provides a characterisation of the relational complexity classes corresponding to the polynomial hierarchy.

Definition 7
- *For an l-ary relation symbol R, and $k \geq l$, we define the second order quantifier $\exists^k R$ to have the following semantics: $\mathfrak{A} \models \exists^k R \varphi$ if there is an $X \subseteq |\mathfrak{A}|^l$ such that X is closed under the equivalence relation $\equiv^k$ in $\mathfrak{A}$, and $(\mathfrak{A}, X) \models \varphi$. As usual, $\forall^k R$ abbreviates $\neg \exists^k R \neg$.*
- *$\Sigma^{1,\omega}_1$ denotes the class of formulas of the form $\exists^{k_1} R_1 \ldots \exists^{k_m} R_m \varphi$, where φ is first order.*
- *$\Pi^{1,\omega}_1$ denotes the class of formulas of the form $\forall^{k_1} R_1 \ldots \forall^{k_m} R_m \varphi$, where φ is first order.*

- $\Sigma^{1,\omega}_{n+1}$ *denotes the class of formulas of the form* $\exists^{k_1} R_1 \ldots \exists^{k_m} R_m \varphi$, *where* φ *is* $\Pi^{1,\omega}_n$.
- $\Pi^{1,\omega}_{n+1}$ *denotes the class of formulas of the form* $\forall^{k_1} R_1 \ldots \forall^{k_m} R_m \varphi$, *where* φ *is* $\Sigma^{1,\omega}_n$.
- $\mathrm{SO}^\omega = \bigcup_{n\in\omega} \Sigma^{1,\omega}_n$.

The expressive power of this second order logic is intermediate between IFP and PFP. In fact, the following is shown in [10].

$$\mathrm{IFP} \subseteq \Sigma^{1,\omega}_1 \cap \Pi^{1,\omega}_1 \subseteq \mathrm{SO}^\omega \subseteq \mathrm{PFP}.$$

The following is also established in [10] by the methods of Theorem 6.

Theorem 8 ([10]) IFP $= \Sigma^{1,\omega}_1$ *if, and only if,* P = NP.

Theorem 9 ([10]) $\Sigma^{1,\omega}_1 =$ PFP *if, and only if,* NP = PSPACE.

Analogous results hold for all levels of the polynomial hierarchy. Details can be found in [10].

2.5 Generalized Quantifiers

Let K be any class of structures over the vocabulary $\sigma = \{R_1, \ldots, R_m\}$, where R_i has arity n_i. We associate with K the generalized (or Lindström) quantifier Q_K.

For a logic L, define the extension $L(Q_K)$ by closing the set of formulas of L under the following formula formation rule: if $\varphi_1, \ldots, \varphi_m$ are formulas of $L(Q_K)$ and $\vec{x}_1, \ldots, \vec{x}_m$ are tuples of variables (with the length of $\vec{x}_i$ being n_i), then $Q_K \vec{x}_1, \ldots, \vec{x}_m(\varphi_1, \ldots, \varphi_m)$ is a formula of $L(Q_K)$. The type of this quantifier is the tuple $(n_1, \ldots, n_m)$ and its arity is $\max(n_i)$.

The semantics of the quantifier is given by: $\mathfrak{A} \models Q_K \vec{x}_1, \ldots, \vec{x}_m(\varphi_1(\vec{x}_1, \vec{y}), \ldots, \varphi_m(\vec{x}_m, \vec{y}))[\vec{a}]$ if, and only if, $(|\mathfrak{A}|, \varphi_1^{\mathfrak{A}}[\vec{a}], \ldots, \varphi_m^{\mathfrak{A}}[\vec{a}]) \in K$, where $\varphi_i^{\mathfrak{A}}[\vec{a}] = \{\vec{b} \in |\mathfrak{A}|^{n_i} \mid \mathfrak{A} \models \varphi_i[\vec{b}, \vec{a}]\}$.

Example 10 1. The *unary counting quantifiers* are those associated with the classes $C_i = \{(A, S) \mid S \subseteq A \text{ and card(S)} \geq \text{i}\}$, for each $i \in \omega$. Each of these has type (1) and arity 1.

2. The *Rescher quantifier* R is given by the class $C = \{(A, S_1, S_2) \mid S_1, S_2 \subseteq A \text{ and card(S}_1) < \text{card(S}_2)\}$. Thus, R is of type $(1, 1)$ and arity 1.

3. The (ordinary) *modular counting quantifiers* M_n are those associated with the classes $\{(A, S) \mid S \subseteq A \text{ and card(S)} \equiv 0 \text{ (mod n)}\}$. Each of these has type (1) and arity 1.

Let $\mathbf{Q}$ be a collection of generalized quantifiers. For any logic L, $L(\mathbf{Q})$ is the logic obtained by closing L under the quantifiers in $\mathbf{Q}$. Define the equivalence relation $\equiv^{k,\mathbf{Q}}$ by: $(\mathfrak{A}, \vec{a}) \equiv^{k,\mathbf{Q}} (\mathfrak{B}, \vec{b})$ if for all formulas $\varphi \in L^k(\mathbf{Q})$, $\mathfrak{A} \models \varphi[\vec{a}]$ if, and only if, $\mathfrak{B} \models \varphi[\vec{b}]$.

A straightforward adaptation of the proof that PFP $\subseteq L^\omega_{\infty\omega}$ (say in [21]) gives the following:

Lemma 11 ([16]) *Let* $\mathbf{Q}$ *be a finite set of generalized quantifiers. Then* IFP($\mathbf{Q}$) $\subseteq$ PFP($\mathbf{Q}$) $\subseteq L^\omega_{\infty\omega}(\mathbf{Q})$.

Furthermore, from the results of [11], it follows that $\equiv^{k,\mathbf{Q}}$ coincides with the type-equivalence relation induced by the infinitary logic $L^k_{\infty\omega}(\mathbf{Q})$, as stated in the lemma below.

Lemma 12 *Let* $\mathbf{Q}$ *be a finite set of generalized quantifiers. Then* $(\mathfrak{A}, \vec{a}) \equiv^{k,\mathbf{Q}} (\mathfrak{B}, \vec{b})$ *if, and only if,* $(\mathfrak{A}, \vec{a})$ *and* $(\mathfrak{B}, \vec{b})$ *satisfy exactly the same* $L^k_{\infty\omega}(\mathbf{Q})$*-formulas.*

As a direct consequence to Lemmas 11 and 12 we get the following generalization of Theorem 3:

> For any formula φ of IFP($\mathbf{Q}$) (or PFP($\mathbf{Q}$)), there is a k such that the query defined by φ is closed under $\equiv^{k,\mathbf{Q}}$.

Cai *et al.* [6] in seeking to answer the question of what needs to be added to IFP in order to obtain a logic capturing exactly the polynomial time queries showed essentially the following.

Theorem 13 ([6]) *There are polynomial time decidable classes of graphs that are not closed under* $\equiv^{k,\mathbf{C}}$*, where* $\mathbf{C}$ *is the collection of all counting quantifiers.*

Further work along these lines established:

Theorem 14 ([16]) *For every* n*, there is a vocabulary* σ *and a polynomial time query on* σ*-structures that is not closed under* $\equiv^{k,\mathbf{Q}_n}$*, where* $\mathbf{Q}_n$ *is the collection of all* n*-ary quantifiers.*

Theorem 15 ([11])
There are polynomial time queries on graphs that are not closed under $\equiv^{k,\mathbf{Q}}$*, where* $\mathbf{Q}$ *is any finite collection of polynomial time computable quantifiers.*

3 Ordering Types with Quantifiers

In this section we investigate in which logics it is possible to define an ordering of equivalence classes, as in Theorem 5, with respect to the equivalence relations $\equiv^{k,\mathbf{Q}}$. Since the formula defining the order would only contain finitely many quantifiers, we are mostly concerned with the case when $\mathbf{Q}$ is a finite collection. To avoid separate treatment for $\exists$, we will assume it to be included in $\mathbf{Q}$.

3.1 Ordering $L^k(\mathbf{Q})$-types in $\mathrm{PFP}(\mathbf{Q})$

We start by presenting an algorithm to order $L^k(\mathbf{Q})$-types and show how to express it in $\mathrm{PFP}(\mathbf{Q})$. The main step is to construct a $\mathrm{PFP}(\mathbf{Q})$-formula which given an ordering of $L^{k,d}(\mathbf{Q})$-types defines an ordering of $L^{k,d+1}(\mathbf{Q})$-types. The algorithm itself is described by an IFP-iteration of this $\mathrm{PFP}(\mathbf{Q})$-formula.

So the top-level structure of our algorithm is similar to that in [12]. However, the step refining ordering from $L^{k,d}(\mathbf{Q})$-types to $L^{k,d+1}(\mathbf{Q})$-types is very different from that in [12] and is an original contribution of this section.

For the base case of the algorithm, we define an ordering on the atomic types (or $L^{k,0}(\mathbf{Q})$-types) in the same way as in [12]. The number of atomic types is fixed by the vocabulary alone, so let $\alpha_1(\vec{x}), \ldots, \alpha_q(\vec{x})$ be an enumeration of quantifier free formulas characterizing each atomic type. The formula $\varphi_0(\vec{x}, \vec{y}) = \bigvee_{1 \leq i < j \leq q} (\alpha_i(\vec{x}) \wedge \alpha_j(\vec{y}))$ defines the desired ordering on atomic types.

Note that by the phrase "ordering $L^{k,d}(\mathbf{Q})$-types" we mean defining a query which on any structure $\mathfrak{A}$ defines a strict linear pre-order on $|\mathfrak{A}|^k$, whose equivalence classes coincide with the equivalence classes of the relation $\equiv_d^{k,\mathbf{Q}}$. In the following, we will show how to construct a $\mathrm{PFP}(\mathbf{Q})$-formula which refines the ordering from $L^{k,d}(\mathbf{Q})$-types to $L^{k,d+1}(\mathbf{Q})$-types.

For this purpose, we examine the equivalence classes of the order on $L^{k,d}(\mathbf{Q})$-types and determine which of these classes need to be split to obtain $L^{k,d+1}(\mathbf{Q})$-types. The following elementary lemma shows that it suffices to look only at a specific kind of $L^{k,d+1}(\mathbf{Q})$-formulas.

Definition 16 *Let k be a positive integer. We define* basic formulas *of quantifier rank $d+1$ as $L^{k,d+1}(\mathbf{Q})$-formulas of the form*

$$Q\vec{y}_1 \ldots \vec{y}_j (\psi_1(\vec{y}_1, \vec{z}_1), \ldots, \psi_j(\vec{y}_j, \vec{z}_j))$$

which are not in $L^{k,d}(\mathbf{Q})$. For a basic formula as above we call the tuple $(\vec{z}_1, \ldots, \vec{z}_j)$ of tuples of variables its free variable pattern.

Lemma 17 *If $(\mathfrak{A}, \vec{a}) \equiv_d^{k,\mathbf{Q}} (\mathfrak{B}, \vec{b})$, but $(\mathfrak{A}, \vec{a}) \not\equiv_{d+1}^{k,\mathbf{Q}} (\mathfrak{B}, \vec{b})$, then there is a basic formula $\varphi(\vec{x})$ of quantifier rank $d+1$ such that either $\mathfrak{A} \models \varphi[\vec{a}]$ or $\mathfrak{A} \models \varphi[\vec{b}]$, but not both.*

Proof: Easy. ■

Let us now consider different ways of constructing basic formulas of quantifier rank $d+1$ from the quantifiers in $\mathbf{Q}$ and a given list of formulas of quantifier rank $\leq d$. Without loss of generality, we can assume that the ψ's in Definition 16 above are all of arity k.

Let Q be of type $(n_1, \ldots, n_j)$. We assume the k-variable formulas under consideration are constructed using the variables $\{x_1, \ldots, x_k\}$ only. A k-variable basic formula starting with Q is completely fixed by a sequence of k-variable formulae $\psi_1, \ldots, \psi_j$, which are to be quantified and a free variable pattern constructed from $\{x_1, \ldots, x_k\}$. In order to capture the class of basic formulae succinctly, we define below the set S_Q as the set of all free variable patterns for Q.

More precisely, for Q as above, we define S_Q to be the set of all j-tuples of tuples of variables as follows:

$$S_Q = \{(s_1, \ldots, s_j) |\ s_i \text{ is a } (k - n_i)\text{-tuple of distinct variables from } \{x_1, \ldots, x_k\}\}.$$

Note that for a fixed collection of formulas $\psi_1, \ldots, \psi_j$ (which are to be viewed as defining subsets of $|\mathfrak{A}|^k$ in a structure $\mathfrak{A}$), the basic formulas beginning with Q and binding the formulas $\psi_1, \ldots, \psi_j$ are in one-to-one correspondence with the elements of S_Q. Indeed given a tuple $s = (s_1, \ldots, s_j) \in S_Q$, we have the unique formula $Q\overline{s}_1, \ldots, \overline{s}_j(\psi_1(\overline{s}_1, s_1), \ldots, \psi_j(\overline{s}_j, s_j))$, where each $\overline{s}_i$ is the complement of s_i in $\{x_1, \ldots, x_k\}$. We call this formula the (Q, s) *basic formula based on* $\psi_1, \ldots, \psi_m$, for $m \geq j$ and arbitrary $\psi_{j+1} \ldots \psi_m$.

Conversely, each basic formula of quantifier rank $d+1$ beginning with Q and binding the formulas $\psi_1, \ldots, \psi_j$ corresponds to a unique tuple $s \in S_Q$.

Let $m = \max\{\mathit{width}(Q) \mid Q \in \mathbf{Q}\}$, where $\mathit{width}(Q)$ is the length of the type of Q (i.e., the number of formulas that Q binds). Let $\psi_1, \ldots, \psi_m$ be a sequence of $L^{k,d}(\mathbf{Q})$-formulas. (Recall that we view formulas as subsets of $|\mathfrak{A}|^k$ in any structure $\mathfrak{A}$). For each quantifier $Q \in \mathbf{Q}$, we consider the basic formulas of quantifier rank $d+1$ based on $\psi_1, \ldots, \psi_m$.

We assume a fixed linear order, $<$, on the finitely many pairs (Q, s), where $Q \in \mathbf{Q}$ and $s \in S_Q$.

The following easy observation relates $L^{k,d}(\mathbf{Q})$-definable sets and $L^{k,d}(\mathbf{Q})$-types.

Lemma 18 *Let k, d be positive integers and $\mathfrak{A}$ be a structure. A subset of $|\mathfrak{A}|^k$ is $L^{k,d}(\mathbf{Q})$-definable if, and only if, it is a union of $L^{k,d}(\mathbf{Q})$-types.*

Proof: Easy. ■

By the lemma above, all $L^{k,d}(\mathbf{Q})$-definable subsets can be extracted from a linear pre-order of $L^{k,d}(\mathbf{Q})$-types, as they are unions of equivalence classes of this ordering. We will use the following lemma to obtain a $\mathrm{PFP}(\mathbf{Q})$ system, whose iteration in any structure $\mathfrak{A}$ gives rise to all possible m-tuples of $L^{k,d}(\mathbf{Q})$-definable subsets of $|\mathfrak{A}|^k$.

Given a strict linear pre-order X on $|\mathfrak{A}|^k$, we denote the equivalence relation on $|\mathfrak{A}|^k$ associated with it as

$\sim_X$. In other words, for all $\vec{x}, \vec{y} \in |\mathfrak{A}|^k$, $\vec{x} \sim_X \vec{y} \leftrightarrow \neg X\vec{x}\vec{y} \wedge \neg X\vec{y}\vec{x}$. Intuitively we think that $X\vec{x}\vec{y}$ stands for '$\vec{x}$ is less than $\vec{y}$'.

Lemma 19 *Let k, m be positive integers and X a global strict linear pre-order on k-tuples of elements of finite structures. Then there is a system $\Phi = (\varphi_1, \ldots, \varphi_m)$ of first order formulas with the following properties. Each φ_i has free relational variables $X, Y_1, \ldots, Y_m$, such that X is parameter variable (of arity $2k$) and $Y_1, \ldots, Y_m$ are recursion variables of arity k each. On any structure $\mathfrak{A}$, PFP-iteration of this system goes through all m tuples of subsets of equivalence classes induced by X on $|\mathfrak{A}|^k$. That is, given $(U_1, \ldots, U_m)$, where each U_j, $1 \leq j \leq m$ is $\sim_X$ closed subset of $|\mathfrak{A}|^k$, then there is an i s.t. in the PFP-iteration i of the of system Φ, $\Phi^i(\mathfrak{A}) = (\varphi_1^i, \ldots, \varphi_m^i)$, where $\varphi_j^i(\vec{x}) \leftrightarrow \vec{x} \in U_j$, for $1 \leq j \leq m$. Further the system Φ converges to a fixpoint, that is for each $\mathfrak{A}$, there is an m_0 with $\Phi^{m_0}(\mathfrak{A}) = \Phi^{m_0+1}(\mathfrak{A})$.*

Proof: The proof is straightforward. The details are left to the full version. A similar observation in the case of linearly ordered structures was previously made by Kerkko Luosto in [17]. ■

Let Φ be the system as in the lemma above. We augment this system by adding one more component for a $2k$-ary relation. The new system Φ' will be $(\Phi, \varphi(X, Y_1, \ldots, Y_m, Z))$, where φ is to be defined below. (Components in Φ do not contain the new relation variable Z). If X is an order on $L^{k,d}(\mathbf{Q})$-types, then the new component will store approximations to the order on $L^{k,d+1}(\mathbf{Q})$-types during the iteration of system Φ'.

Assume that X denotes the ordering of $L^{k,d}(\mathbf{Q})$-types. Consider the PFP-iteration of the system Φ' on a structure $\mathfrak{A}$. As the components of Φ assume as values $L^{k,d}(\mathbf{Q})$-definable subsets of $|\mathfrak{A}|^k$, we would like to consider the basic formulas of quantifier rank $d+1$ based on the tuple of these subsets (or formulas). At any stage of the iteration of the system Φ', Z will contain a strict linear pre-order, which refines the pre-order X. If an equivalence class of Z splits at any stage of the iteration of the system Φ', then we order the resulting classes in a fixed way using the order $<$ on pairs (Q, s) that we fixed earlier. The following formulas achieve this.

For each $Q \in \mathbf{Q}$ of type $(n_1, \ldots, n_j)$ and $s = (s_1, \ldots, s_j) \in S_Q$, let $\theta_{Q,s}(x_1, \ldots, x_k, Y_1, \ldots, Y_m)$ be the formula

$$Q\overline{s}_1 \ldots \overline{s}_j(Y_1(\overline{s}_1, s_1), \ldots, Y_j(\overline{s}_j, s_j)),$$

where, as before, each $\overline{s}_i$ is the complement of s_i in $\{x_1, \ldots, x_k\}$. Thus, $\theta_{Q,s}$ is the (Q, s) basic formula based on the subsets $Y_1, \ldots, Y_m$ of $|\mathfrak{A}|^k$.

In the following, let $\vec{x}, \vec{y}$ stand for $(x_1, \ldots, x_k)$ and $(y_1, \ldots, y_k)$, respectively. Relational variables which do not play any active role in the formulas below will not be displayed.

We define $\beta_{Q,s}(\vec{x}, \vec{y}, Z)$ to be the formula

$$\begin{array}{cc} \bigwedge_{(Q',s')<(Q,s)} & (\theta_{Q',s'}(\vec{x}) \leftrightarrow \theta_{Q',s'}(\vec{y})) \\ \wedge & (\theta_{Q,s}(\vec{x}) \wedge \neg\theta_{Q,s}(\vec{y})) \\ \wedge & (\neg Z\vec{x}\vec{y} \wedge \neg Z\vec{y}\vec{x}) \wedge (\neg X\vec{x}\vec{y} \wedge \neg X\vec{y}\vec{x}). \end{array}$$

Let $\varphi(\vec{x}, \vec{y}, X, Y_1, \ldots, Y_m, Z)$ be the formula

$$X\vec{x}\vec{y} \vee Z\vec{x}\vec{y} \vee \bigvee_{Q,s} \beta_{Q,s}(\vec{x}, \vec{y}, Z).$$

For any fixed relations $X, Y_1, \ldots, Y_m, Z$, the formula φ can be thought of as formalizing the following sequential procedure. All basic formulas based on $Y_1, \ldots, Y_m$ are examined according to the order $<$ on the pairs (Q, s). If (Q^*, s^*) is the least pair distinguishing between two tuples in the *same* equivalence class of the pre-order X, then these tuples are ordered by the formula β_{Q^*,s^*}.

In more detail, this ordering is introduced as follows. A class to be split is partitioned into two subsets depending on whether $\theta_{Q^*,s^*}(\vec{x})$ holds or not for an element $\vec{x}$ of the class, and all elements of one subset of this class are made "lesser" than every element in the other subset. All new pairs of tuples introduced in the pre-order relation are of this kind only. It is easy to see that this operation gives us again a strict linear pre-order.

It may be noted that during the above sequential procedure one equivalence class of Z may split several times, but each such split will be completely within the new classes created in the previous stages. More precisely, a pair $\vec{x}, \vec{y}$ is ordered at some stage only when it has not been ordered at any previous stage. This is ensured by the first conjunct of the formula $\beta_{Q,s}$.

After all the basic formulas based on $Y_1, \ldots, Y_m$ have been examined, φ defines a refined pre-order. In the next iteration step, Z is updated to this value and the components $Y_1, \ldots Y_m$ change their value, and the process repeats.

So, we have proved:

Lemma 20 *Consider the iteration of the system Φ' for a fixed strict linear pre-order X. At the end of each iteration step $i > 0$ of the system Φ', Z defines a strict linear pre-order which refines X.*

Let $\chi_0(X, \vec{x}, \vec{y})$ be a PFP($\mathbf{Q}$)-formula which defines the partial fixed point of the *last* component of the system Φ'. From the above discussion, it is easy to see that if X is a linear order of $L^{k,d}(\mathbf{Q})$-types, then $\chi_0(X, \vec{x}, \vec{y})$ defines a linear order of $L^{k,d+1}(\mathbf{Q})$-types. This is because during the iteration of the system Φ', the tuple $(Y_1, \ldots, Y_m)$ of variables goes through all possible combinations of $L^{k,d}(\mathbf{Q})$-definable relations, and all basic formulae of based on $Y_1, \ldots, Y_m$ are considered.

The desired formula, ψ, defining a linear order of $L^k(\mathbf{Q})$-types is now obtained as follows:

Define $\chi(\vec{x}, \vec{y}) \equiv \varphi_0(\vec{x}, \vec{y}) \vee \chi_0(X, \vec{x}, \vec{y})$.

We have $\psi(\vec{x}, \vec{y}) \equiv \mathbf{ifp}_{X,\vec{x}\vec{y}}\chi(\vec{x}, \vec{y})$.

It is straightforward to see that stage $d+1$ of the iteration of ψ defines an ordering of $L^{k,d+1}(\mathbf{Q})$-types in a structure. $\Psi^d(\mathfrak{A}) = \Psi^{d+1}(\mathfrak{A})$, where $\Psi^d(\mathfrak{A})$ is the value of ψ at iteration d, implies by a quantifier elimination argument (similar to that in [11]) that formulas of quantifier rank $\leq d$ can isolate all $L^k(\mathbf{Q})$-types within $\mathfrak{A}$, hence $\Psi^d(\mathfrak{A})$ defines an ordering on $L^k(\mathbf{Q})$-types within $\mathfrak{A}$. Therefore, we have proved the following.

Theorem 21 *Let k be a positive integer and $\mathbf{Q}$ be a finite set of quantifiers. Then for every vocabulary σ, there is a formula $\gamma(\vec{x}, \vec{y})$ of* PFP($\mathbf{Q}$) *such that on any finite structure $\mathfrak{A}$, γ defines an ordering of the $L^k(\mathbf{Q})$-types of $\mathfrak{A}$.*

3.2 The Complexity of $L^k(\mathbf{Q})$-equivalence

Let $\Sigma^1_1(\mathbf{Q})$ and $\Pi^1_1(\mathbf{Q})$ denote existential second order logic and universal second order logic, extended with the generalized quantifiers in $\mathbf{Q}$, respectively. The following theorem is due to Eric Rosen [23].

Theorem 22 (Rosen) *Let $\mathbf{Q}$ be a finite set of generalized quantifiers, k a positive integer and σ a vocabulary. Then the query $\not\equiv^{k,\mathbf{Q}}$ is definable in $\Sigma^1_1(\mathbf{Q})$. It follows that $\equiv^{k,\mathbf{Q}}$ is definable in $\Pi^1_1(\mathbf{Q})$.*

We do not give a proof of Theorem 22 here. However, since we are concerned with computational complexity in this section, we state, with a proof sketch, the following consequence of this theorem. For the interested reader, we point out that it is not difficult to convert the proof sketch of Theorem 23 below into a proof of Theorem 22, using techniques employed in the next section.

Theorem 23 *Let $\mathbf{Q}$ be a finite set of generalized quantifiers and k a positive integer. Consider structures over a fixed vocabulary σ only. Then the set $\{(\mathfrak{A}, \vec{a}, \vec{b}) \mid (\mathfrak{A}, \vec{a}) \not\equiv^{k,\mathbf{Q}} (\mathfrak{A}, \vec{b})\}$ is in* NP$^{\mathbf{Q}}$*, and consequently the set $\{(\mathfrak{A}, \vec{a}, \vec{b}) \mid (\mathfrak{A}, \vec{a}) \equiv^{k,\mathbf{Q}} (\mathfrak{A}, \vec{b})\}$ is in co*-NP$^{\mathbf{Q}}$.

Proof: *(sketch)* The idea is based on the proof of Theorem 21 in Section 3.1. On input $(\mathfrak{A}, \vec{a}, \vec{b})$ the NP-machine guesses a list $S_0, S_1, \ldots S_l$, where $1 \leq l \leq |\mathfrak{A}|^k$. Each S_i in this list contains a partition of $|\mathfrak{A}|^k$such that the partition in S_{i+1} refines the partition in S_i. Each S_{i+1} also contains a list of witnessing pairs (Q, s) and sets $Y_1, \ldots, Y_m$ (which are unions of some equivalence classes of the partition in S_i) which justifies each new refinement from S_i to S_{i+1}. S_0 corresponds to atomic types and can be verified by the machine in polynomial time. The machine can check that the partition in S_{i+1} is $\equiv^{k,\mathbf{Q}}_i$-closed by checking each tuple in an equivalence class of the partition with all tuples in the other equivalence classes using the witnessing formula provided. The machine accepts if the sequence of $S_0, \ldots, S_l$ is consistent and $\vec{a}, \vec{b}$ are in different classes of the partition in S_l. ■

Using the same ideas as in Section 3.1 we can prove the following complexity upper bound for ordering $L^k(\mathbf{Q})$-types.

Theorem 24 *Let $\mathbf{Q}$ be a finite set of generalized quantifiers and k a positive integer. Consider structures over a fixed vocabulary σ only. Then there is a* P$^{\mathrm{NP}^{\mathbf{Q}}}$ *algorithm, that is a polynomial time machine with* NP$^{\mathbf{Q}}$ *oracle, which on any finite structure $\mathfrak{A}$ produces an ordering of the $L^k(\mathbf{Q})$-types of $\mathfrak{A}$.*

Proof: We present below a P$^{\mathrm{NP}^{\mathbf{Q}}}$ algorithm to refine a given order of $L^{k,d}(\mathbf{Q})$-types to an order on $L^{k,d+1}(\mathbf{Q})$-types. The algorithm to order $L^k(\mathbf{Q})$-types will then follow by polynomially many iterations of the above P$^{\mathrm{NP}^{\mathbf{Q}}}$ step.

Let X be a strict linear pre-order whose equivalence classes correspond to $L^{k,d}(\mathbf{Q})$-types, and let $m = \max\{width(Q) \mid Q \in \mathbf{Q}\}$. Using X, one can define an order of the set of m-tuples of equivalence classes of X, such that two such m-tuples can be compared in polynomial time, and given two m tuples, their mid-point can be computed in polynomial time. (For example one can take this order as defined in the proof of Lemma 19).

Define the following set of triples

$$T = \{(Q, s, \overline{I}) \mid Q \in \mathbf{Q}, s \in S_Q, \overline{I} = (I_1, \ldots, I_m), \text{ each } I_j \text{ is a } \sim_X \text{-closed subset of } |\mathfrak{A}|^k\}.$$

Note that an element of T completely specifies a basic $L^{k,d+1}(\mathbf{Q})$-formula. By assuming some fixed order on the pairs (Q, s) for $Q \in \mathbf{Q}$, $s \in S_Q$, we extend the order above on m-tuples of equivalence classes to an order of elements of T.

Let $\mathcal{S}$ be an $\mathrm{NP}^{\mathbf{Q}}$ algorithm which for each $\vec{a} \sim_X \vec{b}$ and $t_1, t_2 \in T$, determines if $\vec{a}$ and $\vec{b}$ can be distinguished by a $t \in T$ which is between t_1 and t_2. Using binary search based on the algorithm $\mathcal{S}$ above, a $\mathrm{P}^{\mathrm{NP}^{\mathbf{Q}}}$ machine determines if $\vec{a}$ and $\vec{b}$ are in distinct $L^{k,d+1}(\mathbf{Q})$-types, and if so, produces the *least* $(Q, s, \vec{I}) \in T$ that witnesses this. We now order $\vec{a}$ and $\vec{b}$ according to the convention that $\vec{a} < \vec{b}$ if, and only if, $\vec{a}$ satisfies the basic formula corresponding to the least witness and $\vec{b}$ does not satisfy it. This procedure can be repeated for all pairs of tuples in each equivalence class of X, which requires at most $O(|\mathfrak{A}|^k)$ calls to the binary search routine. To see that the resulting relation $<$ will be a strict linear pre-order, we can use reasoning similar to that in Section 3.1. We need to use the fact that the $\vec{a}$ and $\vec{b}$ are ordered using the *least* witness.

It is easy to verify that the resulting strict linear pre-order is independent of the representation of the input structure and depends only on the structure itself. Here we use the fact that the ordering on T used above is independent of the representation. ∎

3.3 Ordering $L^k(\mathbf{Q})$-types in $\mathrm{SO}^\omega(\mathbf{Q})$

Define the logics $\Sigma_n^{1,\omega}(\mathbf{Q})$ and $\Pi_n^{1,\omega}(\mathbf{Q})$ to be the extensions of $\Sigma_n^{1,\omega}$ and $\Pi_n^{1,\omega}$ (see Definition 7) by the quantifiers in $\mathbf{Q}$, where the second order quantifier $\exists^k$ is now replaced by $\exists^{k,\mathbf{Q}}$ which quantifies over relations that are closed under the relation $\equiv^{k,\mathbf{Q}}$.

Making use of Theorem 21 we can show that actually $\Sigma_n^{1,\omega}(\mathbf{Q}), \Pi_n^{1,\omega}(\mathbf{Q}) \subseteq \mathrm{PFP}(\mathbf{Q})$, thus extending the corresponding inclusions proved in [10] to the case with extra quantifiers. Using the same idea as in Theorem 23 we can also prove the following.

Theorem 25 *For each $n \geq 1$, $\Sigma_n^{1,\omega}(\mathbf{Q}) \subseteq \Sigma_n^p(\mathbf{Q})$ and $\Pi_n^{1,\omega}(\mathbf{Q}) \subseteq \Pi_n^p(\mathbf{Q})$, where Σ_n^p and Π_n^p are the nth levels of the polynomial hierarchy.*

This raises the question in which fragment of $\mathrm{SO}^\omega(\mathbf{Q})$ is it possible to define $\equiv^{k,\mathbf{Q}}$ and order $L^k(\mathbf{Q})$-types.

It is trivial to define $\equiv^{k,\mathbf{Q}}$ in $\mathrm{SO}^\omega(\mathbf{Q})$ exploiting the semantics of the second order quantifiers in $\mathrm{SO}^\omega(\mathbf{Q})$. Indeed, the formulas ψ_1 and ψ_2 below define the relations $\equiv^{k,\mathbf{Q}}$ and $\not\equiv^{k,\mathbf{Q}}$, respectively.

$$\psi_1(\vec{x}, \vec{y}) \equiv \forall^{k,\mathbf{Q}} Y(Y\vec{x} \leftrightarrow Y\vec{y}),$$
$$\psi_2(\vec{x}, \vec{y}) \equiv \exists^{k,\mathbf{Q}} Y(Y\vec{x} \wedge \neg Y\vec{y}).$$

We will now show that an ordering of $L^k(\mathbf{Q})$-types can be defined in $\Sigma_2^{1,\omega}(\mathbf{Q}) \cap \Pi_2^{1,\omega}(\mathbf{Q})$. This is done by formalizing the proof of Theorem 24 in $\Sigma_2^{1,\omega}(\mathbf{Q})$ and $\Pi_2^{1,\omega}(\mathbf{Q})$. We present a brief sketch below. Formulas such as $\theta_{Q,s}$, φ_0 etc. used but not defined below refer to those defined in Section 3.1.

Let X be a linear pre-order on k-tuples and let E_X denote the set of equivalence classes of X. Let $\vec{Y} \equiv Y_1, \ldots, Y_m$ and $\vec{Z} \equiv Z_1, \ldots, Z_m$ where $Y_1, \ldots, Y_m, Z_1, \ldots, Z_m$ are k-ary relation symbols.

Let $\alpha(X, \vec{Y}, \vec{Z})$ be an $\mathrm{FO}(X, \vec{Y}, \vec{Z})$-formula which defines the lexicographic ordering of m-tuples of sets of equivalence classes of X.

Let R be a $3k$ ary predicate which is a subset of $(E_X)^3$. We would like to define R such that if $\vec{a}$ is in i^{th} equivalence class of X then $R_{\vec{a}}(\vec{z}_1, \vec{z}_2) = R(\vec{a}, \vec{z}_1, \vec{z}_2)$ is a pre-order on $L^{k,i-1}(\mathbf{Q})$-types.

Let U be a linear pre-order on k-tuples which is coarser than X. Define $\mathit{Least}(U, Y_1, \ldots, Y_m, \vec{x}, \vec{y})$ which is true if, and only if, $Y_1, \ldots, Y_m$ is the least tuple of unions of equivalence classes of U according to the ordering α such that $\bigvee_{Q,s} \neg[\theta_{Q,s}(\vec{x}, Y_1, \ldots Y_m) \leftrightarrow \theta_{Q,s}(\vec{y}, Y_1, \ldots Y_m)]$

The predicate Least can be defined in $\Sigma_2^{1,\omega}(\mathbf{Q}) \cap \Pi_2^{1,\omega}(\mathbf{Q})$ by formalizing the execution of the polynomial time algorithm with $\mathrm{NP}^{\mathbf{Q}}$ oracle in the proof of Theorem 24. Note that this algorithm runs in time polynomial in $|E_U|$ and does not need to know about elements inside the equivalence classes of U.

Let $\mathit{succ}(\vec{a})$ be successor of $\vec{a}$ in pre-order X. Let

$$\begin{aligned}
&\beta'_{Q,s}(\vec{x}, \vec{y}, \vec{Y}) \equiv \\
&[\bigwedge_{(Q',s')<(Q,s)} (\theta_{Q',s'}(\vec{x}, \vec{Y}) \leftrightarrow \theta_{Q',s'}(\vec{y}, \vec{Y}))] \\
&\wedge(\theta_{Q,s}(\vec{x}, \vec{Y}) \wedge \neg\theta_{Q,s}(\vec{y}, \vec{Y}))
\end{aligned}$$

Let

$$\begin{aligned}
&\gamma(X) \equiv \\
&\forall \vec{z}\vec{x}\vec{y}[R(\mathit{succ}(\vec{z}), \vec{x}, \vec{y}) \leftrightarrow \\
&\qquad (\exists^{k,\mathbf{Q}}\vec{Y}\exists^{2k,\mathbf{Q}}U[\forall \vec{z}_1\vec{z}_2(R(\vec{z}, \vec{z}_1, \vec{z}_2) \leftrightarrow U\vec{z}_1\vec{z}_2) \\
&\qquad\qquad \wedge \mathit{Least}(U, \vec{Y}, \vec{x}, \vec{y}) \\
&\qquad\qquad \wedge \bigvee_{(Q,s)} \beta'_{Q,s}(\vec{Y}, \vec{x}, \vec{y})])]
\end{aligned}$$

Using $\Sigma_2^{1,\omega}(\mathbf{Q}) \cap \Pi_2^{1,\omega}(\mathbf{Q})$ definition of $\mathit{Least}(\vec{Y}, \vec{x}, \vec{y})$, R can also be seen to be in $\Sigma_2^{1,\omega}(\mathbf{Q}) \cap \Pi_2^{1,\omega}(\mathbf{Q})$.

$$\begin{aligned}
&\chi(\vec{x}, \vec{y}) \equiv \\
&\exists^{2k,\mathbf{Q}}X[X \text{ is a linear pre-order on } L^k(\mathbf{Q}) \text{ types} \\
&\qquad \wedge \forall \vec{z}(st(\vec{z}) \to (R(\vec{z}, \vec{x}, \vec{y}) \leftrightarrow \varphi_0(\vec{x}, \vec{y}))) \\
&\qquad \wedge \gamma(X) \\
&\qquad \wedge \forall \vec{z}(gt(\vec{z}) \to R(\vec{z}, \vec{x}, \vec{y}))]
\end{aligned}$$

Here $st(x)$ stands for 'x is in the smallest equivalence class of X', $gt(x)$ stands for 'x is in the greatest equivalence class of X'. X is a linear pre-order on $L^k(\mathbf{Q})$-types can easily be expressed in $\Sigma_2^{1,\omega}(\mathbf{Q}) \cap \Pi_2^{1,\omega}(\mathbf{Q})$. Using the definition of R we see that χ is in

$\Sigma_2^{1,\omega}(\mathbf{Q})$.

A $\Pi_2^{1,\omega}(\mathbf{Q})$ ordering can be obtained by using $\forall^{2k,\mathbf{Q}}X$ instead of $\exists^{2k,\mathbf{Q}}X$ in χ above. This does not change the meaning of the formula because X is being used only to simulate a computation. Therefore we have the following.

Theorem 26 *The relation $\equiv^{k,\mathbf{Q}}$ is definable in $\Sigma_1^{1,\omega}(\mathbf{Q})$, the relation $\not\equiv^{k,\mathbf{Q}}$ is definable in $\Pi_1^{1,\omega}(\mathbf{Q})$, and an ordering of $L^k(\mathbf{Q})$-types is definable in $\Sigma_2^{1,\omega}(\mathbf{Q}) \cap \Pi_2^{1,\omega}(\mathbf{Q})$.*

4 Ordering Types in IFP($\mathbf{Q}$)

An analysis of the proof of Theorem 21, in Section 3.1, shows that the step which involves partial fixpoint iteration in an essential way, is to run through all possible m-tuples of $L^{k,d}(\mathbf{Q})$-definable subsets (or formulas) in refining the order from $L^{k,d}(\mathbf{Q})$-types to $L^{k,d+1}(\mathbf{Q})$-types. There are exponentially many such tuples in the number of $L^{k,d}(\mathbf{Q})$-types (or in the length of the order constructed at the end of stage d).

In some situations however, it is possible, due to special properties of the generalized quantifiers under consideration, to restrict attention to only polynomially many $L^{k,d}(\mathbf{Q})$-definable sets (again, polynomial in the number of $L^{k,d}(\mathbf{Q})$-types) to determine all the $L^{k,d+1}(\mathbf{Q})$-types. We will consider some such sets of generalized quantifiers here, to illustrate the techniques involved.

We will use the following notational convention. If s is a tuple of variables from $\{x_1, \ldots, x_k\}$ then we let $s(a_1, \ldots, a_k)$ denote the tuple $s[a_1/x_1, \ldots, a_k/x_k]$ obtained by substitution. For instance, if $s = (x_3, x_1, x_4)$ then $s(a_1, \ldots, a_k)$ is simply a_3, a_1, a_4.

4.1 Ordering L^k-types

In this section, we consider the usual first order logic. So the only generalized quantifier is $\exists$ associated with the class $C = \{(A, U) \mid U \subseteq A, U \neq \emptyset\}$ of unary structures. We will continue to denote $\exists$ by Q, so that we do not get distracted by a change in notation.

Let ψ be an $L^{k,d}(Q)$-formula. For $s = (s_1)$, define $\chi_\psi(\vec{x}) \equiv Q\overline{s}_1\psi(\overline{s}_1, s_1)$, where $\overline{s}_1$ is the complement of s_1 in $\{x_1, \ldots, x_k\}$.

For any structure $\mathfrak{A}$ and $\vec{a}, \vec{b} \in |\mathfrak{A}|^k$, if $\mathfrak{A} \models \chi_\psi[\vec{a}]$, but $\mathfrak{A} \not\models \chi_\psi[\vec{b}]$, then there is an $\equiv_d^{k,\mathbf{Q}}$-*equivalence class* τ, and a formula θ defining τ, such that $\mathfrak{A} \models \chi_\theta(\vec{a})$ but $\mathfrak{A} \not\models \chi_\theta(\vec{b})$. ($\tau$ can simply be taken to be any equivalence class on which ψ is true and which contains a tuple ending in $s_1(\vec{a})$.)

This allows us to consider only $L^{k,d}(Q)$-types instead of arbitrary $L^{k,d}(Q)$-formulas (which are unions of $L^{k,d}(Q)$-types) when considering the splitting of $L^{k,d}(Q)$-types into $L^{k,d+1}(Q)$-types. Hence the PFP-iteration of Lemma 19 and the system Φ' used in the proof of Theorem 21 can be replaced by a system of IFP-iterations as follows.

Let X be a strict linear pre-order. Define $Succ(\vec{y}, Y, X) \equiv \forall\vec{z}(Y\vec{z} \to X\vec{z}\vec{y}) \wedge \neg\exists\vec{z}(X\vec{z}\vec{y} \wedge \neg Y\vec{z})$. We now define an IFP-iteration system $\Psi = (\varphi_1, \varphi_2)$ by letting

$$\varphi_1(Y, Z, \vec{y}) \equiv Succ(\vec{y}, Y)$$

and

$$\varphi_2(Y, Z, \vec{z}_1, \vec{z}_2) \equiv X\vec{z}_1\vec{z}_2 \vee Z\vec{z}_1\vec{z}_2 \vee \bigvee_{(Q,s)} \beta_{(Q,s)}(\vec{z}_1, \vec{z}_2, Z),$$

where $\beta_{Q,s}$ is as defined in Section 3.1, with all occurrences of $Y_1\vec{y}$ replaced by $Succ(\vec{y}, Y)$ in it.

During the IFP-iteration of the system Ψ, the first component goes through each equivalence class of X in turn. At each stage of the iteration, the least class not examined so far is added to Y and the second component determines the new splits that are witnessed by a formula quantifying over this equivalence class, and updates Z to the resulting refined order.

As all the other steps in the proof of Theorem 21 use only IFP-iterations, by replacing Φ' with Ψ we obtain an IFP-formula defining an order on L^k-types. (As, IFP = LFP over finite structures, we also get an LFP-formula defining this ordering).

4.2 Ordering Types with Counting

For each integer i, let Q_i be the counting quantifier associated with the class $C_i = \{(A, U) \mid U \subseteq A, |U| \geq i\}$, and let $\mathbf{C}$ be the *infinite* collection of these quantifiers, $\mathbf{C} = \{Q_i \mid i < \omega\}$.

In this section we will consider the problem of ordering the $L^k(\mathbf{C})$-types. The type-equivalence relation induced by this logic is the same as that induced by the infinitary counting logic $C_{\infty\omega}^k$ studied in [14]. We will derive the result of [14] that $C_{\infty\omega}^k$-types can be ordered in IFP(R), where R is the Rescher quantifier (see Example 10).

Our proof relies on the following easy observation.

Lemma 27 *Let k, d be positive integers and s_1 be a $(k-1)$-tuple of variables from $\{x_1, \ldots, x_k\}$. For any structure $\mathfrak{A}$ and for any $\vec{a}, \vec{b} \in |\mathfrak{A}|^k$, the following are equivalent:*

1. *There is an i and an $L^{k,d}(\mathbf{C})$-formula ψ such that $\vec{a}, \vec{b}$ can be distinguished using a $(Q_i, (s_1))$ basic formula based on ψ.*

2. *There is a $L^{k,d}(\mathbf{C})$-type φ (within $\mathfrak{A}$) such that*

$$Rx,y(\varphi(x,s_1(\vec{a})),\varphi(y,s_1(\vec{b})))\vee$$
$$Rx,y(\varphi(x,s_1(\vec{b})),\varphi(y,s_1(\vec{a})))$$

is true in $\mathfrak{A}$.

Proof: (**1.** $\Rightarrow$ **2.**) Let $\psi = \bigvee_j \psi_j$, where each ψ_j is a single $L^{k,d}(\mathbf{C})$-type. Let $S_j = \{x \in |\mathfrak{A}| \mid \mathfrak{A} \models \psi_j(x, s_1(\vec{a}))\}$ and $T_j = \{x \in |\mathfrak{A}| \mid \mathfrak{A} \models \psi_j(x, s_1(\vec{b}))\}$. All S_j's (and similarly T_j's) are mutually disjoint. We are given that $|\bigcup_j S_j| \neq |\bigcup_j T_j|$, so there must exist an l such that $|S_l| \neq |T_l|$. Let φ be ψ_l.

(**2.** $\Rightarrow$ **1.**) Let φ be such that (2) holds. Let $i_1 = |\{c \in |\mathfrak{A}| \mid \mathfrak{A} \models \varphi(c, s_1(\vec{a}))\}|$ and $i_2 = |\{c \in |\mathfrak{A}| \mid \mathfrak{A} \models \varphi(c, s_1(\vec{b}))\}|$. We can take $i = \max\{i_1, i_2\}$ and $\psi \equiv \varphi$. (Note that φ can easily be defined as an $L^{k,d}(\mathbf{C})$-formula as shown in the proof of Lemma 18, in full version). ∎

Note that the formula in clause (**2.**) above, that expresses inequivalence, can be used to order $\vec{a}$ and $\vec{b}$ depending on which disjunct of this formula is true (both disjuncts cannot be true for the same $\vec{a}, \vec{b}$). Say, we make $\vec{a}$ less than $\vec{b}$ if, and only if, $Rx,y(\varphi(x,s(\vec{a})),\varphi(y,s(\vec{b})))$ holds. It is not difficult to see that this operation will create a strict linear pre-order within each $L^{k,d}(\mathbf{C})$-equivalence class and therefore will keep the overall relation also a strict linear pre-order.

The important point of Lemma 27 is that it reduces the search to $L^{k,d}(\mathbf{C})$-types from arbitrary $L^{k,d}(\mathbf{C})$-formulas. Thus, as in Section 4.1, we can replace the PFP-iteration in the proof of Theorem 21 by an IFP-iteration which runs through each $L^{k,d}(\mathbf{C})$-type and splits types and orders them according to the above formula. Therefore we get an IFP(R)-definable ordering on $L^k(\mathbf{C})$-types.

4.3 Ordering Types with Modular Counting Quantifiers

Definition 28 *A quantifier Q of type $(n_1,\ldots,n_r)$ is counting modulo j based if the defining class of Q is of the form*

$$C_P = \{(A, R_1,\ldots,R_r) \mid R_1 \subseteq A^{n_1},\ldots,R_r \subseteq A^{n_r}, P(|A|, |R_1| \bmod j,\ldots,|R_r| \bmod j)\}$$

for some predicate $P \subseteq \omega \times \{0,1,\ldots,j-1\}^r$.

Note that the usual modular counting quantifiers are included in the above definition of counting modulo based quantifiers. Also, since the predicate P is arbitrary in the above definition, there are non-recursive counting modulo based quantifiers.

We will use the same methodology as in Section 4.1, to show that the $L^k(\mathbf{Q})$-types, for a finite set of counting modulo based quantifiers can be ordered in IFP($\mathbf{Q}$).

Lemma 29 *Let Q be a counting modulo j based quantifier of width r. Let $\psi_1,\ldots,\psi_r$ be $L^{k,d}(Q)$-formulas and $s \in S_Q$. For any structure $\mathfrak{A}$ and for any $\vec{a}, \vec{b} \in |\mathfrak{A}|^k$, if $\vec{a}, \vec{b}$ can be distinguished using a (Q,s) basic formula based on $\psi_1,\ldots,\psi_r$ then $\vec{a}, \vec{b}$ can also be distinguished using a (Q,s) basic formula based on $\varphi_1,\ldots,\varphi_r$, where each φ_i, $1 \le i \le r$ is a union of at most j^3 many $L^{k,d}(Q)$-types within $\mathfrak{A}$.*

Proof: This is proved using a simple combinatorial argument, details are given in the full paper. ∎

Let j be the largest number such that there is counting modulo j based quantifier in $\mathbf{Q}$. By virtue of the above lemma, we have to consider only those $L^{k,d}(\mathbf{Q})$-formulas which are unions of at most j^3 many $L^{k,d}(\mathbf{Q})$-types. This gives a polynomial bound on the number of $L^{k,d}(\mathbf{Q})$-formulas to be considered in order to determine $L^{k,d+1}(\mathbf{Q})$-types, and can be done in IFP($\mathbf{Q}$). The details of this will be given in the full paper.

Lemma 30 *Let k be a positive integer and $\mathbf{Q}$ a finite set of counting modulo based quantifiers. Then there is an IFP($\mathbf{Q}$)-formula which uniformly orders $L^k(\mathbf{Q})$-types in all finite structures.*

4.4 Bounded Monotone Quantifiers

Definition 31 *A quantifier Q of type $(n_1,\ldots,n_j)$ is monotone if its defining class C satisfies the following: if $(A, U_1,\ldots,U_j) \in C$ and $V_1,\ldots,V_j$ are s.t. $U_1 \subseteq V_1 \subseteq A^{n_1},\ldots,U_j \subseteq V_j \subseteq A^{n_j}$ then $(A, V_1,\ldots,V_j) \in C$. For each n, the basis $B_n(Q)$ of Q is defined to be the set of minimal elements (with respect to component-wise inclusion ordering) of the set $\{(A, U_1,\ldots,U_j) \in C \mid |A| = n\}$.*

The following definition is motivated by [19] and generalizes the concept of simple bounded montone quantifier in [19].

Definition 32 *A monotone quantifier Q of type $(n_1,\ldots,n_j)$ is t-bounded if there is an integer t such that for all n, $(A, U_1,\ldots,U_j) \in B_n(Q)$ and each i, $1 \le i \le j$, either $|U_i| \le t$ or $|U_i| \ge |A|^{n_i} - t$. Q is bounded if it is t-bounded for some t.*

Note that there are non-recursive bounded monotone quantifiers. The following lemma can be proved is the same manner as Lemma 29, exploiting the definition of bounded monotone quantifiers.

Lemma 33 *Let Q be a t-bounded monotone quantifier of width r. Let $\psi_1, \ldots, \psi_r$ be $L^{k,d}(Q)$-formulas and $s \in S_Q$. For any structure $\mathfrak{A}$ and for any $\vec{a}, \vec{b} \in |\mathfrak{A}|^k$ if $\vec{a}$ and $\vec{b}$ can be distinguished using a (Q, s) basic formula based on $\psi_1, \ldots, \psi_r$, then $\vec{a}$ and $\vec{b}$ can also be distinguished using a (Q, s) basic formula based on $\varphi_1, \ldots, \varphi_r$, where each φ_i, $1 \le i \le r$, is either the union of at most t $L^{k,d}(Q)$-types or the union of all but t $L^{k,d}(Q)$-types within $\mathfrak{A}$.*

Using Lemma 33 and combining it with the reasoning in Section 4.3, we immediately get:

Lemma 34 *Let k be a positive integer and $\mathbf{Q}$ a finite set of bounded monotone quantifiers. Then there is an* IFP($\mathbf{Q}$)*-formula which uniformly orders $L^k(\mathbf{Q})$-types over all finite structures.*

4.5 A General Construction

Sometimes it suffices to define an order on classes which refine the $L^k(\mathbf{Q})$-types, rather than on $L^k(\mathbf{Q})$-types themselves. The theorem below shows that there are many quantifiers Q' for which an ordering on classes finer than $L^k(Q')$-types can be defined in IFP(Q').

Theorem 35 *For every quantifier Q, there is a quantifier Q' such that:*

1. PFP(Q) = PFP(Q') *and Q is definable in* FO(Q')*.*
2. *For each k, there is an* IFP(Q')*-definable ordering of the $L^k(Q)$-types.*
3. *For each k, there is an* IFP(Q')*-definable linear pre-order of k-tuples, whose equivalence classes refine the $L^k(Q')$-types.*

Proof: In the full paper. ■

5 A Quantifier with no IFP Definable Order

In this section we exhibit a quantifier P such that $\equiv^{k,P}$ is not definable in IFP(P), and there is *per force* no ordering of the equivalence classes definable in IFP(P).

From Theorem 21 it follows immediately that IFP(P) $\neq$ PFP(P) for this quantifier P. Moreover, we will see that the quantifier P is polynomial time computable, which means that every query in IFP(P) is also polynomial time computable. Thus, we get the following picture:

$$\begin{array}{ccccc} \text{IFP} & \subset & \text{IFP}(P) & \subset & \text{P} \\ & & \neq & & \\ \text{PFP} & \subset & \text{PFP}(P) & \subset & \text{PSPACE} \end{array}$$

By Theorem 6, IFP and PFP are equivalent if, and only if, P and PSPACE are. Yet, we can show that these logics can be extended by a single quantifier to obtain two logics that are provably separable, while remaining weak enough to be contained in the respective complexity classes. This may seem paradoxical, but it should be noted that we cannot prove a statement analogous to Theorem 6 for the logics IFP(P) and PFP(P) because the proof of that theorem depends on, for one thing, the IFP definability of the query ordering the $\equiv^k$ equivalence classes — the very query whose analogue is provably not in IFP(P). Since the equivalence relation $\equiv^{k,P}$ is definable also in $\Pi_1^{1,\omega}(P)$, we obtain a picture similar to the one above, with $\Sigma_1^{1,\omega}$ replacing both occurrences of PFP and NP in place of PSPACE.

However, in some cases it is possible to relate the separation of IFP($\mathbf{Q}$) and PFP($\mathbf{Q}$) to separation of complexity classes. We will prove some results to this effect in the next section.

The quantifier is given by: $P = Q_K$, where K is the class of all (finite) structures $(A, S, <, B)$ such that

- S is a unary relation;
- $<$ is a binary relation that is a linear order on a subset of A;
- B is a unary relation interpreted by a subset of the field of $<$; and
- the integer encoded by the binary string $(<, B)$ is the cardinality of S.

Theorem 36 *The query $\equiv^{k,P}$ is not definable in* IFP(P)*.*

Proof: *(sketch)* Let σ be the vocabulary $\{E\}$, where E is binary. We show that for any formula $\varphi(x, y) \in$ IFP(P) there is a pair $\mathfrak{A}, \mathfrak{B}$ of σ-structures, and elements $a, a' \in |\mathfrak{A}|$, $b, b' \in |\mathfrak{B}|$ such that

- $\mathfrak{A} \models \varphi[a, a']$ if, and only if, $\mathfrak{B} \models \varphi[b, b']$;
- $(\mathfrak{A}, a) \equiv^{k,P} (\mathfrak{A}, a')$; and
- $(\mathfrak{B}, b) \not\equiv^{k,P} (\mathfrak{B}, b')$.

Clearly this means that φ does not define the query $\equiv^{k,P}$ for singletons, whence the claim follows.

Thus, let $\varphi(x,y) \in \mathrm{IFP}(P)$, and let l be such that the binary query defined by φ is closed under $\equiv^{l,P}$. We fix a large enough m, and let $P' = Q_{K'}$ be the quantifier that is defined like P, except that it only counts up to $m+l$, i.e., K' is the subclass of K given by the condition $|S| \leq m+l$. Furthermore, let $\varphi'(x,y) \in \mathrm{IFP}(P')$ be the formula that is obtained from φ by replacing the quantifier P with the quantifier P'. Observe that P' is first-order definable, whence φ' is actually an IFP-formula. Thus, for some r, the query defined by φ' is closed under $\equiv^r$. Indeed, it is not difficult to show that this holds for $r = 2(m+l)$.

Consider now the class J of all σ-structures that consist of a disjoint union of two parts, one of which is a linear order of length m, and the other an equivalence relation having exactly two equivalence classes and cardinality at most 2^{m+1}. Let H be the set of all triples $(\mathfrak{A}, a, a')$ such that $\mathfrak{A} \in J$, both equivalence classes of $\mathfrak{A}$ contain at least r elements, and a and a' are non-equivalent elements in the equivalence relation part of $\mathfrak{A}$. An easy pebble game argument shows that $(\mathfrak{A}, a, a') \equiv^r (\mathfrak{B}, b, b')$ holds whenever $(\mathfrak{A}, a, a'), (\mathfrak{B}, b, b') \in H$. Hence, the formula φ' cannot separate the triples $(\mathfrak{A}, a, a')$ in H from each other.

The crucial observation is that in the process of evaluating whether $\mathfrak{A} \models \varphi'[a,a']$ holds for a given triple $(\mathfrak{A}, a, a') \in H$, the number of times a subformula starting with the quantifier P' is encountered, is polynomial with respect to m. In particular, only polynomially many different binary strings are tested with the quantifier P'. Furthermore, the set of binary strings that are tested does not depend on the choice of $(\mathfrak{A}, a, a')$. Hence, there exists $s \leq 2^m$ such that no binary string that encodes an integer belonging to either of the intervals $[s-r-1, s+r+1]$ and $[2s-r-1, 2s+r+1]$ is tested during the evaluation of φ' on any triple $(\mathfrak{A}, a, a') \in H$.

Let $(\mathfrak{A}, a, a'), (\mathfrak{B}, b, b') \in H$ be such that both equivalence classes in $\mathfrak{A}$ are of size s, while in $\mathfrak{B}$ the equivalence classes are of size s and $s+1$. It is easy to show that in both of these models the size of a subset defined by any subformula of φ' is either at most $m+l$ or in one of the intervals $[s-r-1, s+r+1]$ and $[2s-r-1, 2s+r+1]$. This means that in the process of evaluating the original formula φ on $(\mathfrak{A}, a, a')$ and $(\mathfrak{B}, b, b')$, all subformulas starting with P are always false, unless the number encoded by the binary string that is being tested is at most $m+l$. Thus, the truth value of φ on these models is not affected if P is replaced by P'. From this we conclude that $\mathfrak{A} \models \varphi[a,a']$ if, and only if, $\mathfrak{B} \models \varphi[b,b']$. Since there is an automorphism of $\mathfrak{A}$ mapping a to a', it is clear that $(\mathfrak{A}, a) \equiv^{k,P} (\mathfrak{A}, a')$. Finally note that there is a formula $\psi(x)$ of $L^k(P)$ saying that the equivalence class of x contains exactly s elements (this is because any subset of a given linear order is definable in L^2, see for instance [13]), whence $(\mathfrak{B}, b) \not\equiv^{k,P} (\mathfrak{B}, b')$. ■

6 Generalizing the Abiteboul-Vianu Theorem

In this section, we will show that in many cases if $L^k(\mathbf{Q})$-types can be ordered in $\mathrm{IFP}(\mathbf{Q})$, then we can relate the separation of $\mathrm{IFP}(\mathbf{Q})$ and $\mathrm{PFP}(\mathbf{Q})$ to the separation of P and PSPACE. We do this by constructing an appropriate quotient structure as in [12].

We will illustrate our method in detail for modular counting quantifiers (see Section 4.3), though the techniques are quite general. The basic idea is the same as in [12], that is we look upon the evaluation of a formula in a structure $\mathfrak{A}$ as the evaluation of another formula over a structure fomed by taking its quotient with respect to $L^k(\mathbf{Q})$-types. However, in general the quotient structure we get in the presence of generalized quantifiers is not a first order structure but contains some second order constants, as in [25]. We will exploit the fact that for evaluating counting modulo based quantifiers we need to consider only polynomially many combinations of types. This will allow us to keep the quotient structure first order though our translation is more complex than in [12].

We need our set of quantifiers to satisfy the following technical but not very restrictive 'soundness' condition.

Definition 37 *Let $\mathbf{Q}$ be a finite set of counting modulo based quantifiers. Let $l = \max\{arity(Q) \mid Q \in \mathbf{Q}\}$. We call $\mathbf{Q}$* **sound** *if for each j, such that there is a counting modulo j based quantifier in $\mathbf{Q}$, and each $i \leq l$, the quantifier M^i_j, associated with class $\{(A, U) \mid U \subseteq A^i,\ |U| \equiv 0 \pmod{j}\}$, is also in $\mathbf{Q}$.*

Our reduction will work for sound sets of quantifiers only. In order to make things less cumbersome we will only consider counting modulo j based *unary* quantifiers, for a fixed j. Generalization to the most general case is straightforward.

We now describe the construction of the quotient structure. Let $\sigma = \{R_1, \ldots, R_m\}$ be the vocabulary of structure $\mathfrak{A}$. The vocabulary σ'_k (we will drop the subscript k if it is clear from the context) of the quotient structure $\mathfrak{A}/{\equiv^{k,\mathbf{Q}}}$ consists of the symbols $<_k$, $='$, $R'_1, \ldots, R'_m$, $X_1, \ldots, X_k$, $P_{s_1}, \ldots, P_{s_l}$, for

$l = k^k$, $S_{s,j}$, where s is a sequence of $n-1$ variables from $\{x_1, \ldots, x_k\}$, $S^T_{Q,s}$, where $Q \in \mathbf{Q}$, $s \in S_Q$, $T \subseteq \{1, \ldots, t\}$, $t = width(Q)$.

The symbol $<_k$ is binary, $='$, $R'_1, \ldots, R'_m$ are unary relation symbols, $P_{s_1}, \ldots, P_{s_l}$, for $l = k^k$, and $X_1, \ldots X_k$ are binary relation symbols.

The relation symbols $S_{s,j}$ are of arity $1 + j$.

For each $Q \in \mathbf{Q}$ of width t, $T \subseteq \{1, \ldots, t\}$, and $s \in S_Q$, $S^T_{Q,s}$ is a relational symbol of arity $1 + j^2.t$.

Given a structure $\mathfrak{A}$ the quotient structure $\mathfrak{A}/\equiv^{k,\mathbf{Q}}$ has domain consisting of the $L^k(\mathbf{Q})$-type equivalence classes of $|\mathfrak{A}|^k$. For $a_1, \ldots, a_k \in |\mathfrak{A}|^k$, we denote its equivalence class with respect to the relation $\equiv^{k,\mathbf{Q}}$ by $[a_1, \ldots, a_k]$. The relation symbols in σ'_k are assigned meaning as follows.

The symbols $<_k$ is interpreted by the IFP($\mathbf{Q}$) definable total order of $L^k(\mathbf{Q})$-types as in Lemma 30. The meaning of $='$, R'_i, X_i and P_s is the same as in [25] or in [12]. We omit these routine definitions here. Instead we only discuss the case of the relations $S_{s,j}$ and $S^T_{Q,s}$.

Let s be a sequence of $n-1$ variables from $\{x_1, \ldots, x_k\}$. $S_{s,j}([\vec{a}], [\vec{b}_1], \ldots, [\vec{b}_j])$ if, and only if, $|\{c \in A \mid (c, s(\vec{a})) \in \cup_{i=1}^{j} [\vec{b}_i]\}| \equiv 0 \pmod{j}$.

Let Q be a counting modulo j based quantifier of type $(n_1, \ldots, n_t)$. Let $s = (s_1, \ldots, s_t) \in S_Q$ and $T \subseteq \{1, \ldots, t\}$.

$$S^T_{Q,s}([\vec{a}], [\vec{b}_1], \ldots, [\vec{b}_{j^2}], \ldots, [\vec{b}_{(t-1).j^2+1}], \ldots, [\vec{b}_{t.j^2}])$$
$$\text{if, and only if}$$
$$\mathfrak{A} \models Q\overline{s}_1, \ldots, \overline{s}_t(\alpha^T_1(\overline{s}_1, s_1(\vec{a})), \ldots, \alpha^T_t(\overline{s}_t, s_t(\vec{a}))),$$

where, for $1 \leq l \leq t$,

$$\alpha^T_l = \begin{cases} \theta_l & \text{if } l \in T \\ \exists x_1(x_1 \neq x_1) & \text{otherwise} \end{cases}$$

and θ_{l+1}, $0 \leq l \leq t-1$, is a formula over σ defining the set $\{\vec{c} \in |\mathfrak{A}|^k \mid \vec{c} \in [\vec{b}_{l.j^2+1}] \vee \ldots \vee \vec{c} \in [\vec{b}_{(l+1).j^2}]\}$

Note that we need to show that $S_{s,j}$, $S^T_{Q,s}$ are well defined. This is easy and we leave the details to the full version.

It may be mentioned that we could have chosen $S^T_{Q,s}$ to be of arity $1 + |T|.j^2$ but for the sake of uniformity we chose it to be of arity $1 + t.j^2$.

Lemma 38 *For every* FO*-formula φ' in vocabulary σ'_k with m free variables there is an* FO($\mathbf{Q}$)*-formula φ in vocabulary σ with $k.m$ free variables such that for any structure $\mathfrak{A}$, $\mathfrak{A} \models \varphi(\vec{a}_1, \ldots, \vec{a}_m)$ if, and only if, $\mathfrak{A}/\equiv^{k,\mathbf{Q}} \models \varphi'([\vec{a}_1], \ldots, [\vec{a}_m])$. Similarly, for every* IFP*-formula φ' in vocabulary σ'_k there is an* IFP($\mathbf{Q}$)*-formula φ in vocabulary σ with the above property.*

Proof: Follows by the same technique as in [12], using the fact that the quotient structure is uniformly definable in IFP($\mathbf{Q}$). ■

Given an FO($\mathbf{Q}$)-formula φ constructed using variables $x_1, \ldots, x_k$, we define a formula $\varphi^*(x)$, in one free variable x, over σ'_k to reflect the evaluation of φ in the quotient structure. The definition is by induction on the structure of φ. Here we just point out the difficulty in the case involving generalized quantifiers, the other cases in the definition of $\varphi^*(x)$ being the same as in [25, 12]. In the following, $\vec{x}$ stands for $x_1, \ldots, x_k$.

Let $\varphi(\vec{x}) \equiv Q\overline{s}_1, \ldots, \overline{s}_t(\varphi_1(\overline{s}_1, s_1), \ldots, \varphi_t(\overline{s}_t, s_t))$.

A formula φ^*_i when considered as a union of $L^k(\mathbf{Q})$-types in a structure, may be the union of unboundedly many types over the class of all finite structures. However, we have kept information in the quotient structure for evaluating a quantifier only on the union of at most j^2 types. This difficulty is resolved by realizing that there are $\psi_1, \ldots, \psi_t$, with parameter x, over σ'_k which define the union of not more than j^2 types, for any fixed x, and can be used in place of φ^*_i's to evaluate Q. The IFP construction of ψ_i's and other details will be given in the full version. It is in this construction that we need to use the assumption that $\mathbf{Q}$ is a sound set, in the sense of Definition 37. Finally we have:

Lemma 39 *For every $L^k(\mathbf{Q})$-formula φ of vocabulary σ, there is an* IFP*-formula φ^*, with one free variable, of vocabulary σ'_k such that for any structure $\mathfrak{A}$, $\mathfrak{A} \models \varphi(a_1, \ldots, a_k)$ if, and only if, $\mathfrak{A}/\equiv^{k,\mathbf{Q}} \models \varphi^*([a_1, \ldots, a_k])$. For every* PFP($\mathbf{Q}$)*-formula φ with k free variables, of vocabulary σ, there is a k_1 and a* PFP*-formula φ^*, of vocabulary σ'_{k_1} such that for any structure $\mathfrak{A}$, $\mathfrak{A} \models \varphi(\vec{a})$ if, and only if, $\mathfrak{A}/\equiv^{k_1,\mathbf{Q}} \models \varphi^*([\vec{a}])$.*

Proof: In the full version. ■

Theorem 40 *Let $\mathbf{Q}$ be a finite and sound set of counting modulo based quantifiers. Then* PFP($\mathbf{Q}$) $\neq$ IFP($\mathbf{Q}$) $\Rightarrow$ PSPACE $\neq$ P.

Proof: This follows from standard arguments, as in [12, Theorem 11], using Lemma 38 and Lemma 39. ■

Note that the set $\mathbf{Q}$ may contain quantifiers of arbitrary complexity (including non-recursive quantifiers), still separating PFP($\mathbf{Q}$) from IFP($\mathbf{Q}$) implies the separation of PSPACE from P.

We do not know the other direction of the above implication. However, if quantifiers in $\mathbf{Q}$ are polynomial time computable then it is immediate. Indeed since in this case $\mathrm{IFP}(\mathbf{Q})$ and $\mathrm{PFP}(\mathbf{Q})$ capture P and PSPACE respectively on ordered structures, $\mathrm{IFP}(\mathbf{Q}) = \mathrm{PFP}(\mathbf{Q})$ implies P = PSPACE. So we have the following corollary to Theorem 40.

Theorem 41 *Let $\mathbf{Q}$ be a finite and sound set of polynomial time computable counting modulo based quantifiers. Then* $\mathrm{PFP}(\mathbf{Q}) \neq \mathrm{IFP}(\mathbf{Q})$ *if, and only if,* PSPACE $\neq$ P. *In particular, if $\mathbf{Q} = (M_{j_1}, \ldots M_{j_m})$, that is if $\mathbf{Q}$ consists of finitely many usual* mod *quantifiers (see Example 10), then* $\mathrm{PFP}(\mathbf{Q}) \neq \mathrm{IFP}(\mathbf{Q})$ *if, and only if,* PSPACE $\neq$ P.

Using the techniques used to prove Theorems 40, 41, we can also prove the following.

Theorem 42 *Let $\mathbf{Q}$ be a finite set of bounded monotone quantifiers. Then* $\mathrm{PFP}(\mathbf{Q}) \neq \mathrm{IFP}(\mathbf{Q}) \Rightarrow$ PSPACE $\neq$ P.

Theorem 43 *Let $\mathbf{Q}$ be a finite set of polynomial time computable bounded monotone quantifiers. Then* $\mathrm{PFP}(\mathbf{Q}) \neq \mathrm{IFP}(\mathbf{Q})$ *if, and only if,* PSPACE $\neq$ P.

7 Further Applications

In this section we show two more applications of our results from Sections 3.1 and 4. The first concerns proving McColm's second conjecture in the presence of generalized quantifiers.

Note that if a class C of structures realizes unboundedly many $L^k(\mathbf{Q})$-types for some k, then in cases where we can define an ordering of $L^k(\mathbf{Q})$-types in $\mathrm{IFP}(\mathbf{Q})$, we can write using this ordering an $\mathrm{IFP}(\mathbf{Q})$-formula which admits unbounded induction over C. So combining a result of [11] (Corollary 4.5), with results of Section 4 we have:

Theorem 44 *Let $\mathbf{Q}$ be a finite set of counting modulo based quantifiers or bounded monotone quantifiers. Then the following are equivalent for any class C, of finite structures.*

1. $\mathrm{IFP}(\mathbf{Q})$ *admits an unbounded induction over C.*
2. $L^{\omega}_{\infty\omega}(\mathbf{Q}) \neq \mathrm{FO}(\mathbf{Q})$ *over C.*

The construction in Section 4.5 gives another class of generalized quantifiers for which the above result holds although it is not clear if an ordering on $L^k(Q')$-types can be defined in $\mathrm{IFP}(Q')$. Using the same argument as in Theorem 44 above, we have the following:

Theorem 45 *Let Q' be a quantifier as in Theorem 35. Then the following are equivalent for any class C, of finite structures.*

1. $\mathrm{IFP}(Q')$ *admits an unbounded induction over C.*
2. $L^{\omega}_{\infty\omega}(Q') \neq \mathrm{FO}(Q')$ *over C.*

Since order makes simulation of computation possible, we can analyse the ability of $\mathrm{PFP}(\mathbf{Q})$ to define all PSPACE queries over classes of structures. We present only one such result here.

Definition 46 *Let $\mathbf{Q}$ be a finite set of generalized quantifiers. A class C of rigid structures is of bounded $\mathbf{Q}$ rigidity if there is a k such that for each $\mathfrak{A} \in C$ and $a_1, a_2 \in |\mathfrak{A}|$, $(\mathfrak{A}, a_1)$ and $(\mathfrak{A}, a_2)$ have different $L^k(\mathbf{Q})$-types.*

Lemma 47 *Let $\mathbf{Q}$ be a finite set of generalized quantifiers.* PSPACE $\subseteq \mathrm{PFP}(\mathbf{Q})$ *over a class C of rigid structures if, and only if, C is of bounded $\mathbf{Q}$ rigidity.*

It is possible to generalize some other results of [24] in the presence of finitely many generalized quantifiers.

8 Conclusion

The definability in IFP of an ordering $<^k$ of k-variable equivalence classes is crucial to results showing that the relationship between a variety of fixed point and related logics mirrors the relationship of complexity classes above P. This has served as the basis of a thorough understanding of the expressive power of these logics which capture the corresponding complexity classes on ordered structures.

We showed that in the presence of a finite set of quantifiers $\mathbf{Q}$, an ordering $<^{k,\mathbf{Q}}$ is definable in $\mathrm{PFP}(\mathbf{Q})$ and in $\Sigma^{1,\omega}_2(\mathbf{Q}) \cap \Pi^{1,\omega}_2(\mathbf{Q})$. On the other hand we have also shown that there is a polynomial time quantifier Q such that $<^{k,Q}$ is provably not definable in $\mathrm{IFP}(Q)$. We do not yet have any characterization of the sets $\mathbf{Q}$ of quantifiers on which an ordering $<^{k,\mathbf{Q}}$ is definable in $\mathrm{IFP}(\mathbf{Q})$.

For some specific quantifiers, we proved some results relating separation of fixpoint logics to the separation of complexity classes. These are analogous to the Abiteboul-Vianu theorem. The techniques used seem to have some limitations and it is not clear how to overcome these.

Acknowledgements: Research of the first and the second author was supported by a British–Finnish ARC grant from the British Council and CIMO.

A special note: A large part of the results of this paper were proved independently by the third author and by the first two authors, without the knowledge of each other's work. Two separate papers containing these results, one by the first two authors (Anuj Dawar and Lauri Hella) and one by the third author (Anil Seth) were accepted to LICS'98. The present paper is the result of merging these two papers, following the recommendation of the programme committee.

References

[1] S. Abiteboul, M. Y. Vardi, and V. Vianu. Fixpoint logics, relational machines, and computational complexity. *J. ACM*, 44:30–56, 1997.

[2] S. Abiteboul and V. Vianu. Datalog extensions for database queries and updates. *J. of Computer and System Sciences*, 43:62–124, 1991.

[3] S. Abiteboul and V. Vianu. Generic computation and its complexity. In *Proc. 23rd ACM Symposium on the Theory of Computing*, 1991.

[4] S. Abiteboul and V. Vianu. Computing with first-order logic. *J. of Computer and System Sciences*, 50(2):309–335, 1995.

[5] J. Barwise. On Moschovakis closure ordinals. *Journal of Symbolic Logic*, 42:292–296, 1977.

[6] J. Y. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.

[7] A. Chandra and D. Harel. Computable queries for relational databases. *J. of Computer and System Sciences*, 21:156–178, 1980.

[8] A. Chandra and D. Harel. Structure and complexity of relational queries. *J. of Computer and System Sciences*, 25:99–128, 1982.

[9] A. Dawar. *Feasible Computation through Model Theory.* PhD thesis, University of Pennsylvania, Philadelphia, 1993.

[10] A. Dawar. A restricted second order logic for finite structures. In D. Leivant, editor, *Logic and Computational Complexity*, volume 960 of *LNCS*, pages 393–413. Springer, 1995.

[11] A. Dawar and L. Hella. The expressive power of finitely many generalized quantifiers. *Information and Computation*, 123(2):172–184, 1995.

[12] A. Dawar, S. Lindell, and S. Weinstein. Infinitary logic and inductive definability over finite structures. *Information and Computation*, 119(2):160–175, 1995.

[13] H. Ebbinghaus and J. Flum. *Finite Model Theory.* Perspectives in Mathematical Logic. Springer, 1995.

[14] E. Grädel and M. Otto. Inductive definability with counting on finite structures. In *Computer Science Logic (CSL 92).* Springer-Verlag, Lecture Notes in Computer Science, 702, 1993.

[15] Y. Gurevich and S. Shelah. Fixed-point extensions of first-order logic. *Annals of Pure and Applied Logic*, 32:265–280, 1986.

[16] L. Hella. Logical hierarchies in PTIME. *Information and Computation*, 129:1–19, 1996.

[17] L. Hella, P. G. Kolaitis, and K. Luosto. How to define a linear order on finite models. In *Annals of Pure and Applied Logic*, 87:241-267, 1997.

[18] N. Immerman. Relational queries computable in polynomial time. *Information and Control*, 68:86–104, 1986.

[19] P. G. Kolaitis and J. A. Väänänen. Generalized quantifiers and pebble games on finite structures. *Annals of Pure and Applied Logic*, 74(1):23–75.

[20] P. G. Kolaitis and M. Y. Vardi. Fixpoint logic vs. infinitary logic in finite-model theory. In *Proc. 7th IEEE Symp. on Logic in Computer Science*, pages 46–57, 1992.

[21] P. G. Kolaitis and M. Y. Vardi. Infinitary logics and 0-1 laws. *Information and Computation*, 98(2):258–294, 1992.

[22] M. Otto. *Bounded Variable Logics and Counting*, volume 9 of *Lecture Notes in Logic.* Springer, 1997.

[23] E. Rosen, 1993. Personal Communication.

[24] A. Seth. When do fixed point logics capture complexity classes. In *Tenth Annual IEEE Symposium on Logic in Computer Science*, pages 353–363, 1995.

[25] A. Seth. Sharper results on the expressive power of generalized quantifiers. In *Proceedings of 17th FST & TCS conference, Springer LNCS 1346*, pages 200–219, 1997.

[26] M. Y. Vardi. The complexity of relational query languages. In *Proc. 14th ACM Symposium on the Theory of Computing*, pages 137–146, 1982.