Non-linear Prefixes in Query Languages

Non-Linear Prefixes in Query Languages

Antonio Badia*
Computer Engineering and
Computer Science department
University of Louisville
abadia@louisville.edu

Stijn Vansummeren†
Hasselt University and
Transnational University of Limburg
Belgium
stijn.vansummeren@uhasselt.be

ABSTRACT

In first order logic there are two main extensions to quantification: generalized quantifiers and non-linear prefixes. While generalized quantifiers have been explored from a database perspective, non-linear prefixes have not – most likely because of complexity concerns. In this paper we first illustrate the usefulness of non-linear prefixes in query languages by means of example queries. We then introduce the subject formally, distinguishing between two forms of non-linearity: *branching* and *cumulation*. To escape complexity concerns, we focus on monadic quantifiers. In this context, we show that branching does not extend the expressive power of first order logic when it is interpreted over finite models, while cumulation does not extend the expressive power when it is interpreted over bounded models. Branching and cumulation do, however, allow us to formulate some queries in a succinct and elegant manner. When branching and cumulation are interpreted over infinite models, we show that the resulting language can be embedded in an infinitary logic proposed by Libkin. We also discuss non-linear prefixes from an algorithmic point of view.

Categories and Subject Descriptors

H.2.3 [**Information Systems**]: Database Management—*Languages*

General Terms

Languages, Theory

Keywords

Generalized quantifiers, non-linear prefixes, branching, cumulation

*Supported by grant CAREER-0347555 from the National Science Foundation.
†Postdoctoral Fellow of the Research Foundation - Flanders (FWO).

PODS'07, June 11–14, 2007, Beijing, China.

1. INTRODUCTION

Modern, declarative query languages are based in logic, in particular, in First Order Logic (henceforth, FO) or some extension thereof. Thus, relational algebra is equivalent to the safe fragment of FO, while SQL can be seen as (safe) FO plus counters, and Datalog can be seen as (safe) FO with fixpoint. In the past century, FO has also been extended with respect to its quantification power, in two ways: with the introduction of generalized quantifiers (by Mostowski [30] and Lindstrom [28]) and with the introduction of non-linear prefixes of quantifiers (by Henkin [17]). While the idea of generalized quantification has been used as the basis for a query language [2, 13], the idea of non-linear prefixes has, to to the best of the authors' knowledge, never been applied to databases.

It is well known that in the standard *linear* quantifier prefixes used in FO any existential quantifier is under the scope of all the preceding universal quantifiers. For instance, the sentence $(\forall x)\,(\exists y)\,(\forall z)\,(\exists w)\,\varphi(x,y,z,w)$ can be converted, via Skolem functions, into the logically equivalent second order sentence

$$(\exists F_1)\,(\exists F_2)\,(\forall x)\,(\forall z)\,\varphi(x, F_1(x), z, F_2(x,z)) \qquad (1)$$

However, when non-linear *branching* quantifiers are allowed, formulas can be obtained which are not expressible in first order logic [6]. For instance, the sentence

$$\begin{pmatrix}(\forall x) & (\exists y)\\ (\forall z) & (\exists w)\end{pmatrix}\varphi(x,y,z,w) \qquad (2)$$

is intuitively read as stating that for each x we pick a y and for each z with pick a w, with w depending only on z and not on x. Through Skolemization, we obtain

$$(\exists F_1)\ (\exists F_2)\ (\forall x)\ (\forall z)\ \varphi(x, F_1(x), z, F_2(z)). \qquad (3)$$

Note that each existentially quantified variable is replaced by a Skolem function of *a single* variable in the formula above. Any linear reformulation of this sentence would lead to a Skolemization where at least one existentially quantified variable is replaced by a Skolem function depending on both x and z, as in (1). Depending on the exact statement of ϕ, the sentence (2) above hence cannot necessarily be expressed in FO.

Example 1. *As an example of the power of branching, the sentence*

$$\begin{pmatrix}(\forall x_1) & (\exists y_1)\\ (\forall x_2) & (\exists y_2)\end{pmatrix}(\exists z_1)\ (\exists z_2)\ (\exists z_3)\ (\phi_1 \wedge \phi_2 \wedge \phi_3),$$

where

$$\phi_1 = (x_1 = x_2) \rightarrow (y_1 = y_2)$$
$$\phi_2 = R(x_1, x_2) \rightarrow (y_1 \neq y_2)$$
$$\phi_3 = \bigwedge_{i \in \{1,2\}} \bigvee_{j \in \{1,2,3\}} (y_i = z_j)$$

expresses 3*-colorability of a graph* R*.*

In fact, a result in Blass and Gurevich [5] asserts that sentences like (2) above (with φ first order) capture existential second order logic which, by Fagin's theorem, captures NP. Hence, the reason that using non-linear prefixes have not been exploited in query languages may be complexity.

In this paper, we introduce the idea of non-linear prefixes in query languages.

- We first argue the usefulness of non-linear prefixes by giving examples of queries in English that are naturally and succinctly expressed as formulae using non-linear prefixes.
- We then introduce, FO($\mathcal{Q}$), the extension of FO with generalized quantifiers in $\mathcal{Q}$ and show how two forms of non-linearity, *branching* and *cumulation* can be expressed as operations *on* the quantifiers in $\mathcal{Q}$. As such, if $\mathcal{Q}^b$ and $\mathcal{Q}^c$ stand for the classes of generalized quantifiers obtained from $\mathcal{Q}$ by branching respectively cumulating quantifiers in $\mathcal{Q}$, then FO($\mathcal{Q}^b$) and FO($\mathcal{Q}^c$) is first order logic with branching and cumulation, respectively.
- Motivated by our examples and to avoid formulae like (2) that allow us to capture NP, we focus on FO($\mathcal{Q}^b$) and FO($\mathcal{Q}^c$) where all quantifiers in $\mathcal{Q}$ are *monadic*.
- We then show that in this setting FO($\mathcal{Q}^b$) does not extend the expressive power of FO, provided that all quantifiers in $\mathcal{Q}$ are evaluated on finite models (as is natural in query languages).
- We also show that FO($\mathcal{Q}^c$) does not add to the expressive power of FO provided that all quantifiers in $\mathcal{Q}$ are evaluated on finite models with a bound on their domain. We provide some partial insights into the expressiveness of FO($\mathcal{Q}^c$) when such a bound does not exist. In particular, we show that FO($\mathcal{Q} \cup \mathcal{Q}^b \cup \mathcal{Q}^c$) is embeddable in $\mathcal{L}_C$, an infinitary logic due to Libkin [27], even when generalized quantifiers in $\mathcal{Q}$ are evaluated on infinite models.
- Finally, we discuss branching and cumulation from an algorithmic point of view and close with directions for future work.

2. PRELIMINARIES

Let us first introduce the basic concepts underlying this work. Generalized Quantifiers (GQs) are extremely versatile and powerful; because we use only a limited version of the concept, and in the interest of brevity and clarity, we give here a simplified definition, fixing several parameters upfront.

Databases. From the outset we postulate a fixed, infinite universe $\mathcal{U}$ of atomic data values. Throughout the paper, we fix an arbitrary database schema $\mathbf{S}$. A database schema is a finite set of relation names, where each relation name R has an associated arity, denoted by arity(R). A database D over $\mathbf{S}$ is an assignment of a finite relation $D(R) \subseteq \mathcal{U}^k$ to each $R \in \mathbf{S}$, where k is the arity of R.

Generalized Quantifiers. A *generalized quantifier* of arity k is a non-empty set Q of k-ary relations over $\mathcal{U}$ that is closed under isomorphism (i.e., if $R \in Q$ and $S \subseteq \mathcal{U}^k$ is isomorphic to R, then also $S \in Q$). We write $Q\colon k$ to indicate that Q is a generalized quantifier of arity k. A generalized quantifier of arity 1 is called *monadic*.

Intuitively, generalized quantifiers describe "generic" properties of relations [1]. For example, the set $\mathsf{three} := \{R \subseteq \mathcal{U} \mid |R| = 3\}$ of all sets of cardinality 3 is a generalized quantifier. Likewise, the set $\mathsf{even} := \{R \subseteq \mathcal{U} \mid |R| \text{ is even }\}$ of all sets of even cardinality is also a generalized quantifier. The set $\{\{a\}\}$ with a an arbitrary atomic data value is no a generalized quantifier, however, as is not closed under isomorphism (and hence break "genericity"). Another interesting example is the generalized quantifier $\mathsf{conn} := \{R \subseteq \mathcal{U}^2 \mid R \text{ is a connected graph}\}$ of all connected graphs.

Extending FO with generalized quantifiers. We can use generalized quantifiers to extend the power of first order logic as follows. Let $\mathcal{Q}$ be a class of generalized quantifiers. Then FO($\mathcal{Q}$) is the extension of first order logic with generalized quantifiers in $\mathcal{Q}$, whose formulae ϕ are given by the following grammar:

$$\begin{array}{rcl} \phi & ::= & x = y \mid R(x_1, \ldots, x_k) \mid \phi \wedge \phi \mid \neg\phi \mid (\exists x)\phi \\ & \mid & (Q\, x_1, \ldots, x_k)\, \phi. \end{array}$$

Here, x, y, and x_i (i a natural number) range over variables; $R \in \mathbf{S}$ is a relation name of arity k; and $Q \in \mathcal{Q}$ is a generalized quantifier of arity k.

The semantics of formulae is as follows. As usual, an *assignment* is a function from the set of all variables to $\mathcal{U}$. Let the satisfaction relation $D \models \phi[\alpha]$, denoting that formula ϕ is true in database D under assignment α, be defined as usual in FO [11] for all formulae except those of the form $(Q\, x_1, \ldots, x_k)\, \phi$. Then $D \models (Q\, x_1, \ldots, x_k)\phi[\alpha]$ iff

$$\{(a_1, \ldots, a_k) \in \mathcal{U}^k \mid D \models \phi[x_1 \colon a_1, \ldots, x_k \colon a_k, \alpha]\} \in Q.$$

Here, $x_1 \colon a_1, \ldots, x_k \colon a_k, \alpha$ is the assignment that equals α on all variables except that it maps x_1 to a_1, …, x_k to a_k.

We stress that $(Q\, x_1, \ldots, x_k)$ binds the variables $x_1, \ldots, x_k$ in a formula $(Q\, x_1, \ldots, x_k)\, \phi$. Formally, if the free variables FV(ϕ) of a formula ϕ, are defined as usual in FO [11] for all formulae except $(Q\, x_1, \ldots, x_k)\phi$, then

$$\mathrm{FV}((Qx_1, \ldots, x_k)\phi) = \mathrm{FV}(\phi) - \{x_1, \ldots, x_k\}.$$

We follow the standard convention and write $\phi(x, \ldots, y)$ to denote that every free variable of ϕ is among $x, \ldots, y$. Also, we write $\overline{x}$ for a sequence of variables $x_1, \ldots, x_k$.

EXAMPLE 2. *The formula* $\phi(x) = (\mathsf{three}\, y)\, R(x, y)$ *selects all nodes in* R *with out-degree* 3*. This query can clearly be simulated in* FO*. The query* $\psi(x) = (\mathsf{even}\, y)\, R(x, y)$ *that selects all nodes in* R *with an even out-degree cannot, however. As a final example, if we interpret a tuple* (x, y, z) *to signify that there is an* y*-labeled edge from node* x *to node* z*, then* $(\mathsf{conn}\, x, z)\, (\exists y)\, S(x, y, z)$ *is the query that decides whether the unlabeled graph underlying* S *is connected.*

At this point, it is worthwhile to note that the standard first order quantifiers $\exists$ and $\forall$ can also be expressed using generalized quantifiers. Indeed, if $\mathsf{exists} = \{R \subseteq \mathcal{U} \mid R \neq \emptyset\}$, and $\mathsf{all} = \{\mathcal{U}\}$, then it is readily seen that

$$(\exists x)\,\phi \equiv (\mathsf{exists}\,x)\,\phi \quad \text{and} \quad (\forall x)\,\phi \equiv (\mathsf{all}\,x)\,\phi.$$

However, as seen in the examples, there are many monadic quantifiers, like the aforementioned even, that are not FO-definable.

3. LINEAR AND NON-LINEAR PREFIXES

As in first order logic, formulae in FO($\mathcal{Q}$) have an equivalent formula in prenex normal form

$$\pm_1(Q_1\,\overline{x_1})\ \pm_2(Q_2\,\overline{x_2})\cdots\pm_n(Q_n\,\overline{x_n})\,\phi,$$

where every $\pm_i \in \{\neg, \neg\neg\}$, every $Q_i \in \{\exists\} \cup \mathcal{Q}$, and ϕ is quantifier-free. In particular, every variable of $\overline{x_i}$ in the formula above is in the scope of the variables of $\overline{x_1}, \ldots, \overline{x_{i-1}}$. The quantifier prefix $\pm_1(Q_1\,\overline{x_1})\ \pm_2(Q_2\,\overline{x_2})\cdots\pm_n(Q_n\,\overline{x_n})$ is therefore called *linear*.

Some queries in English, however, naturally involve *non-linear* quantifier prefixes [35, 37]. Consider, for example, the ternary relation LINEITEM whose tuples are of the form

$$(\text{orderkey}, \text{partkey}, \text{supkey}),$$

relating orders to their ordered parts, together with the parts' supplier. This relation is a simplification of the TPCH benchmark for decision support systems [39]. Now, consider the natural language query "List the orders where (at least) three suppliers supply (at least) five parts". There are various ways to interpret "three suppliers supply five parts":

1. There are three suppliers that supply five parts each. The parts may be the same or different for each supplier. This reading is expressed in FO(three, five) by

$$\varphi_1(x) := (\mathsf{three}\,y)\,(\mathsf{five}\,z)\ \text{LINEITEM}(x,y,z). \quad (4)$$

   The quantifier five is within the scope of the quantifier three. (Of course, this query can also be expressed in FO, albeit using a longer formula.)

2. There are five parts that are supplied by three suppliers each. The suppliers may or may not be the same for each part. This reading is expressed in FO(three, five) by

$$\varphi_2(x) := (\mathsf{five}\,z)\,(\mathsf{three}\,y)\ \text{LINEITEM}(x,y,z). \quad (5)$$

   The quantifier three is within the scope of the quantifier five.

3. There are three suppliers and five parts such that each supplier supplies all five parts and all five parts are supplied by each supplier. Observe that the suppliers and parts are "picked" from the universe independently of each other. In formal linguistics, this reading is expressed using *branching* quantification [6]:

$$\varphi_3(x) := \begin{pmatrix}\mathsf{three}\,y\\ \mathsf{five}\,z\end{pmatrix}\text{LINEITEM}(x,y,z). \quad (6)$$

   Here, each quantifier is independent of each other (being first or second has no significance).

4. Finally, it may be that the same three suppliers supply *a total* of five parts among the three of them (e.g., the first supplier supplies two parts, the second supplies two parts more, and the third supplier supplies one more part; or the first supplier supplies one part, the second supplies one part, and the third supplies three parts; or ...). In formal linguistics, this reading is expressed using *cumulation* quantification:

$$\varphi_5(x) := [\mathsf{three}\,y, \mathsf{five}\,z]\ \text{LINEITEM}(x,y,z) \quad (7)$$

   Again, the suppliers and parts are "picked" independently of each other and each quantifier is independent of each other (being first or second has no significance).

The important point of our example is that there are readings of some English queries that naturally involve non-linear quantification in terms of branching or cumulation. This suggests that branching and cumulation may be of interest in query languages. Let us therefore introduce these concepts formally.

We first note that we can view linear quantification as given by formulas of the form

$$\pm_1(Q_1\,\overline{x_1})\ \pm_2(Q_2\,\overline{x_2})\cdots\pm_n(Q_n\,\overline{x_n})\,\phi,$$

also as an iteration operation *on* generalized quantifiers [44].

DEFINITION 3 (ITERATION). *If $Q\colon k$ and $P\colon l$ are generalized quantifiers then $Q \cdot P\colon k+l$ is the generalized quantifier*

$$\left\{R \subseteq \mathcal{U}^{k+l} \mid \left\{\overline{a} \in \mathcal{U}^k \mid \{\overline{b} \mid (\overline{a},\overline{b}) \in R\} \in P\right\} \in Q\right\}.$$

Indeed, it is readily verified that $(Q\,\overline{x})\,(P\,\overline{y})\,\phi$ is equivalent to $((Q \cdot P)\,\overline{x},\overline{y})\,\phi$.

In a similar vain, we can also define branching and cumulation as operations on generalized quantifiers. To simplify notation, we abandon the matrix-based notation for branching as we have used it up until here, and use a linear notation that facilitates integration of formulas in the surrounding text.

DEFINITION 4 (BRANCHING). *Let $Q_1\colon n_1, \ldots, Q_k\colon n_k$ be generalized quantifiers and let $n = \sum n_i$. Then the branching $(Q_1;\ldots;Q_k)\colon n$ of $Q_1,\ldots,Q_k$ is the generalized quantifier*

$$\{R \subseteq \mathcal{U}^n \mid \text{there is } S_1 \in Q_1, \ldots, S_n \in Q_k \text{ such that } S_1 \times \cdots \times S_k \subseteq R\}.$$

For example, it is readily verified that the third reading of our suppliers query above is indeed expressed by

$$((\mathsf{three};\mathsf{five})\,x,y)\ \text{LINEITEM}(x,y,z). \quad (8)$$

Moreover, the sentence

$$((\mathsf{all}\cdot\mathsf{exists};\mathsf{all}\cdot\mathsf{exists})\,x_1,y_1,x_2,y_2)\ (\exists z_1)(\exists z_2)(\exists z_3)\ \phi_1 \wedge \phi_2 \wedge \phi_3, \quad (9)$$

with all and exists the generalized quantifier forms of $\forall$ and $\exists$; and ϕ_1, ϕ_2, and ϕ_3 as in Example 1 expresses three-colorability.

DEFINITION 5 (CUMULATION). *Let $Q_1 : n_1, \ldots, Q_k : n_k$ be generalized quantifiers and let $n = \sum n_i$. Then the cumulation $[Q_1, \ldots, Q_k] : n$ of $Q_1, \ldots, Q_k$ is the generalized quantifier*

$$\{R \subseteq \mathcal{U}^n \mid \exists S \subseteq R \text{ such that } S_i \in Q_i \text{ for } 1 \leq i \leq k\},$$

where S_i is the projection of S on columns $1 + \sum_{j<i} n_i$ until $\sum_{j \leq i} n_j$:

$$S_i := \{\overline{a_i} \in \mathcal{U}^{n_i} \mid (\overline{a_1}, \ldots, \overline{a_k}) \in S \text{ for some } \overline{a_j} \in \mathcal{U}^{n_j}\}.$$

For example, it is readily verified that the fourth reading of our suppliers query above is expressed by

$$([\mathsf{three}, \mathsf{five}]\ y, z)\, \mathrm{LINEITEM}(x, y, z). \tag{10}$$

In what follows, we write $\mathcal{Q}^b$ for the class of generalized quantifiers obtained by branching quantifiers in $\mathcal{Q}$:

$$\mathcal{Q}^b := \{(Q_1; \ldots; Q_k) \mid k \geq 0 \text{ and every } Q_i \in \mathcal{Q}\}.$$

Similarly, we write $\mathcal{Q}^c$ for the class of generalized quantifiers obtained by cumulating quantifiers in $\mathcal{Q}$:

$$\mathcal{Q}^c := \{[Q_1; \ldots; Q_k] \mid k \geq 0 \text{ and every } Q_i \in \mathcal{Q}\}.$$

4. NON-LINEAR PREFIXES OF MONADIC QUANTIFIERS

As we have already noted in the Introduction, it is well-known that unrestricted quantifier branching gives a substantial rise in expressive power. For instance, Example 1 shows that branching allows us to express 3-colorability, an NP-hard property. In fact, Blass and Gurevich showed that the subset of $\mathrm{FO}(\{\mathsf{all} \cdot \mathsf{exists}\}^b)$ formulae of the form

$$\big((\mathsf{all} \cdot \mathsf{exists}; \mathsf{all} \cdot \mathsf{exists})\, x_1, y_1, x_2, y_2\big)\, \phi$$

with ϕ first order suffices to capture NP [5].

Since branching arbitrary quantifiers of arity 2 is therefore too powerful a feature to add to a query language, we will restrict our attention to branching and cumulation of monadic quantifiers (i.e., generalized quantifiers of arity 1). Our examples of Section 3 suggests that, apart from being theoretically interesting, branching and cumulation of such quantifiers is also useful in practice. In fact, research in formal linguistics strongly suggests that branching and cumulation may be needed to properly formalize some natural language sentences [35, 37].

As we are primarily interested in the application of non-linear prefixes to database queries – which test properties of *finite* relations – we will first focus on generalized quantifiers that themselves test only properties of finite relations.

DEFINITION 6. *A generalized quantifier is* finite-model *if it contains only finite relations.*

For instance, three, even, and exists are all finite-model, but all is not as it contains the infinite universe $\mathcal{U}$.

We define the range $\mathrm{rng}(Q)$ of a finite-model generalized quantifier Q, as $\mathrm{rng}(Q) := \{|R| \mid R \in Q\}$. Observe that for monadic generalized quantifiers, $R \in Q$ iff $|R| \in \mathrm{rng}(Q)$ since Q is closed under isomorphism. We abbreviate $\min(\mathrm{rng}(Q))$ by $\min(Q)$.

4.1 Branching

The following theorem then shows that branching monadic finite-model quantifiers does not add expressive power to first order logic.

PROPOSITION 7. *Let $\mathcal{Q}$ be a class of monadic, finite-model generalized quantifiers. Then* $\mathrm{FO}(\mathcal{Q}^b) \equiv \mathrm{FO}$ *and* $\mathrm{FO}(\mathcal{Q} \cup \mathcal{Q}^b) \equiv \mathrm{FO}(\mathcal{Q})$.

PROOF. Since $\mathrm{FO}(\mathcal{Q}^b)$ and $\mathrm{FO}(\mathcal{Q} \cup \mathcal{Q}^b)$ are at least as expressive as FO and $\mathrm{FO}(\mathcal{Q})$, respectively, it suffices to show that they are no more expressive.

Let $Q_1, \ldots, Q_k$ be quantifiers in $\mathcal{Q}$. First observe that a relation S is in $(Q_1; \ldots; Q_k)$ iff there exist S_i with $|S_i| = \min(Q_i)$ for $1 \leq i \leq k$ such that $S_1 \times \cdots \times S_k \subseteq S$. Indeed, the if direction follows by definition of branching. For the only-if direction, suppose that $S \in (Q_1; \ldots; Q_k)$. By definition, there exist $S_i' \in Q_i$ such that $S_1' \times \cdots \times S_k' \subseteq S$. Since $\min(Q_i)$ is the minimum of $\mathrm{rng}(Q_i)$, we know that the cardinality of S_i' is at least $\min(Q_i)$. Then fix, for every i, a subset $S_i \subseteq S_i'$ of cardinality $\min(Q_i)$. Since generalized quantifiers are closed under isomorphism and since there is a set of cardinality $\min(Q_i)$ in Q_i, also $S_i \in Q_i$. Moreover,

$$S_1 \times \cdots \times S_k \subseteq S_1' \times \cdots \times S_k' \subseteq S,$$

as desired.

Then observe that the concrete elements of $S_1, \ldots, S_k$ do not matter since every Q_i is closed under isomorphism. As such, a formula $((Q_1; \ldots; Q_k)\, x_1, \ldots, x_k)\, \phi(x_1, \ldots, x_k, \overline{y})$ is equivalent to

$$(\exists z_1^1) \ldots (\exists z_{\min(Q_1)}^1) \ldots (\exists z_1^k) \ldots (\exists z_{\min(Q_k)}^k)$$
$$\left(\bigwedge_{1 \leq i < j \min(Q_1)} z_i^1 \neq z_j^1\right) \wedge \cdots \wedge \left(\bigwedge_{1 \leq i < j \min(Q_k)} z_i^k \neq z_j^k\right)$$
$$\wedge \left(\bigwedge_{\substack{1 \leq i_1 \leq \min(Q_1) \\ \cdots \\ 1 \leq i_k \leq \min(Q_k)}} \phi(z_{i_1}^1, \ldots, z_{i_k}^k, \overline{y})\right)$$

By using this translation inductively, every $\mathrm{FO}(\mathcal{Q}^b)$ formula is hence equivalent to a FO formula, and every $\mathrm{FO}(\mathcal{Q} \cup \mathcal{Q}^b)$ formula is hence equivalent to a $\mathrm{FO}(\mathcal{Q}^b)$ formula, as desired. □

4.2 Cumulation

For cumulation of monadic finite-model quantifiers, the expressiveness is less clear. The following proposition shows that no expressiveness is gained when only *bounded* generalized quantifiers are cumulated. Here, a generalized quantifiers is *bounded* if it is finite-model and $\mathrm{rng}(Q)$ is finite.

PROPOSITION 8. *Let $\mathcal{Q}$ be a class of bounded monadic generalized quantifiers. Then* $\mathrm{FO}(\mathcal{Q}^c) \equiv \mathrm{FO}$ *and* $\mathrm{FO}(\mathcal{Q} \cup \mathcal{Q}^c) \equiv \mathrm{FO}(\mathcal{Q})$.

PROOF. Since $\mathrm{FO}(\mathcal{Q}^c)$ and $\mathrm{FO}(\mathcal{Q} \cup \mathcal{Q}^c)$ are at least as expressive as FO and $\mathrm{FO}(\mathcal{Q})$ respectively, it suffices to show that they are no more expressive.

First observe that $([Q_1; \ldots; Q_k]x_1, \ldots, x_k)\, \phi(x_1, \ldots, x_k, \overline{y})$ with every $\mathrm{rng}(Q_i)$ a singleton $\{c_i\}$ is equivalent to the for-

mula $\psi_{c_1,\ldots,c_k}$ defined as follows.

$$(\exists z_1^1)\ldots(\exists z_{c_1}^1)\ldots(\exists z_1^k)\ldots(\exists z_{c_k}^k)$$
$$\left(\bigwedge_{1\leq i<j<c_1} z_i^1 \neq z_j^1\right)\wedge\cdots\wedge\left(\bigwedge_{1\leq i<j<c_k} z_i^k \neq z_j^k\right)$$
$$\wedge\left(\bigwedge_{1\leq m\leq k}\bigwedge_{1\leq j\leq c_m}\bigvee_{\substack{1\leq i_1\leq c_1\\ \ldots\\ 1\leq i_{m-1}\leq c_{m-1}\\ 1\leq i_{m+1}\leq c_{m+1}\\ \ldots\\ 1\leq i_k\leq c_k}}\phi(z_{i_1}^1,\ldots,z_{i_k}^k,\overline{y})\right)$$

Hence, $([Q_1';\ldots;Q_k']x_1,\ldots,x_k)\,\phi(x_1,\ldots,x_k,\overline{y})$ with $\mathrm{rng}(Q_i')$ not necessarily a singleton is equivalent to

$$\bigvee_{c_1\in\mathrm{rng}(Q_1')}\cdots\bigvee_{c_k\in\mathrm{rng}(Q_k')}\psi_{c_1,\ldots,c_k}.$$

This is clearly a finite formula when all $\mathrm{rng}(Q_i')$ are finite. By using this translation inductively, every $\mathrm{FO}(\mathcal{Q}^b)$ formula is hence equivalent to a FO formula, and every $\mathrm{FO}(\mathcal{Q}\cup\mathcal{Q}^b)$ formula is hence equivalent to a $\mathrm{FO}(\mathcal{Q}^b)$ formula, as desired. □

For the more general class of monadic finite-model monadic quantifiers, we provide the following partial insights for binary cumulations.

LEMMA 9 (RANGE REDUCTION). *The cumulation of a bounded with a finite-model quantifier is the same as the cumulation of two bounded quantifiers. That is, let Q_1 and Q_2 be two finite-model monadic generalized quantifiers such that Q_1 is bounded and Q_2 is not. Let l be the least element in $\mathrm{rng}(Q_2)$ such that $l\geq\max(\mathrm{rng}(Q_1))$. Then $[Q_1,Q_2]=[Q_1,Q_3]$ where $Q_3=\{R\in Q_2\mid |R|\leq l\}$.*

PROOF. It is clear that $[Q_1,Q_3]\subseteq[Q_1,Q_2]$. For the converse, suppose that $R\in[Q_1,Q_2]$. Then there exists $S\subseteq R$ such that $\pi_1(S)\in Q_1$ and $\pi_2(S)\in Q_2$, where π_1 and π_2 denote left and right projection of a binary relation, respectively. We discern the following cases:

- If $|\pi_2(S)|\leq l$ then $\pi_2(S)\in Q_3$. Hence, $R\in[Q_1,Q_3]$, as desired.
- If $|\pi_2(S)|>l$ then let $n=|\pi_1(S)|$. By definition of l, we have $n\leq l$. Then let S' be a subset of n tuples of S such that $\pi_1(S')=\pi_1(S)$ (clearly, such a subset must exist). Let B be a subset of $\pi_2(S)$ of cardinality $l-|\pi_2(S')|$, disjoint with $\pi_2(S')$. Such B must always exist since $|\pi_2(S)|>l$. Clearly, for each $b\in B$ there exists $a_b\in\pi_1(S)=\pi_1(S')$ such that $(a_b,b)\in S$. Then let $S''=S'\cup\{(a_b,b)\mid b\in B\}$. By construction, $S''\subseteq S\subseteq R$, $|\pi_1(S'')|=|\pi_1(S')|=|\pi_1(S)|\in\mathrm{rng}(Q_1)$, and $|\pi_2(S'')|=|\pi_2(S')|+|B|=|\pi_2(S')|+(l-|\pi_2(S')|)=l\in\mathrm{rng}(Q_3)$. As such, $\pi_1(S'')\in Q_1$ and $\pi_2(S'')\in Q_3$, and hence $R\in[Q_1,Q_3]$, as desired. □

It is not clear if and how this reduction lemma generalizes to arbitrary cumulations $[Q_1,\ldots,Q_k]$.

LEMMA 10. *The cumulation of two unbounded quantifiers whose ranges have an element in common is the same as the cumulation of two bounded quantifiers. That is, let Q_1 and Q_2 be two monadic finite-model generalized quantifiers such that $\mathrm{rng}(Q_1)\cap\mathrm{rng}(Q_2)$ is non-empty. Let $l=\max(\mathrm{rng}(Q_1)\cap\mathrm{rng}(Q_2))$, let $Q_1'=\{R\in Q_1\mid|R|\leq l\}$, and let $Q_2'=\{R\in Q_2\mid|R|\leq l\}$. Then $[Q_1,Q_2]=[Q_1',Q_2']$.*

PROOF. It is clear that $[Q_1',Q_2']\subseteq[Q_1,Q_2]$. For the converse, suppose that $R\in[Q_1,Q_2]$. Then there exists $S\subseteq R$ such that $\pi_1(S)\in\mathrm{rng}(Q_1)$ and $\pi_2(S)\in\mathrm{rng}(Q_2)$, where π_1 and π_2 denote left and right projection of a binary relation, respectively. We discern four cases:

- If $|\pi_1(S)|\leq l$ and $|\pi_2(S)|\leq l$ then $\pi_1(S)\in\mathrm{rng}(Q_1')$ and $\pi_2(S)\in\mathrm{rng}(Q_2')$. Hence, $R\in[Q_1',Q_2']$, as desired.
- If $|\pi_1(S)|>l$ and $|\pi_2(S)|\leq l$ then there are at least $|\pi_1(S)|>|\pi_2(S)|$ tuples in S. Let $n=|\pi_2(S)|$ and let S' be a subset of n tuples of S such that $\pi_2(S')=\pi_2(S)$. Furthermore, let $m:=|\pi_1(S')|\leq|S'|=n$ and let A be a subset of $\pi_1(S)$ of cardinality $l-m$, disjoint with $\pi_1(S')$. Such A must always exist since $|\pi_1(S)|>l$. Clearly, for each $a\in A$ there exists $b_a\in\pi_2(S)=\pi_2(S')$ such that $(a,b_a)\in S$. Then let $S''=S'\cup\{(a,b_a)\mid a\in A\}$. By construction, $S''\subseteq S\subseteq R$, $|\pi_1(S'')|=|\pi_1(S')|+|A|=m+(l-m)=l\in\mathrm{rng}(Q_1')$, and $|\pi_2(S'')|=|\pi_2(S')|=|\pi_2(S)|=n\in\mathrm{rng}(Q_2')$. As such, $\pi_1(S'')\in Q_1'$ and $\pi_2(S'')\in Q_2'$ and hence $R\in[Q_1',Q_2']$, as desired.
- If $|\pi_1(S)|\leq l$ and $|\pi_2(S)|>l$ then the reasoning is similar.
- If $|\pi_1(S)|>l$ and $|\pi_2(S)|>l$, then it suffices to show that there always exists $S'\subseteq S$ such that $|\pi_1(S')|=l$ and $|\pi_2(S')|\geq l$, as we can then use the reasoning in the previous cases to show that $R\in[Q_1',Q_2']$. But if both $|\pi_1(S)|>l$ and $|\pi_2(S)|>l$, then there must exist $S'\subseteq S$ such that S' is bijective (i.e., one-to-one and onto) and has cardinality l (if there is no such subset, then either $|\pi_1(S)|\leq l$ or $|\pi_2(S)|\leq l$). Hence, we are back in one of the previous cases. □

Again, it is not clear if and how this lemma generalizes to arbitrary cumulations $[Q_1,\ldots,Q_k]$.

4.3 Branching and cumulation of infinite-model quantifiers

Summarizing our results so far, we have shown that branching of finite-model monadic quantifiers does not add expressive power to FO, nor does cumulation of bounded monadic quantifiers. We now provide a rough upper bound on the expressiveness of branching and cumulation of *arbitrary* (possibly infinite-model) monadic quantifiers. In particular, we show that $\mathrm{FO}(\mathcal{Q}\cup\mathcal{Q}^b\cup\mathcal{Q}^c)$ with $\mathcal{Q}$ a class of monadic quantifiers can be embedded into $\mathcal{L}_C$, an infinitary logic due to Libkin [27]. Although the embedding by itself is rather straightforward, we hope that further research on connections of non-linear prefixes with infinitary logics may provide better insights into the exact expressiveness of $\mathrm{FO}(\mathcal{Q}\cup\mathcal{Q}^b\cup\mathcal{Q}^c)$.

The logic $\mathcal{L}_C$ is an extension of FO with (1) counting quantifiers $(\exists i\ x)$ for each natural number i, interpreted as "there are i x such that"; and (2) infinitary connectives: if

$(\varphi_i)_{i\in I}$ is a (possibly infinite) collection of formulae, then $\bigvee_{i\in I}\varphi_i$ is a formula; and the same for conjunction $\bigwedge_{i\in I}\varphi_i$. Libkin proves that an extension of FO with aggregates and grouping that captures SQL can be translated into this logic.

We should note that $\mathcal{L}_C$ is interesting in the sense that all formulae of finite rank express only *local* properties [27]. Hence, any language that can be translated into $\mathcal{L}_C$ using only finite rank formulae is also local. Unfortunately, however, our embedding below yields $\mathcal{L}_C$ of unbounded rank and it is hence currently not clear whether $\mathrm{FO}(\mathcal{Q}\cup\mathcal{Q}^b\cup\mathcal{Q}^c)$ is even a local language.

To be precise, the *rank* $\mathrm{rank}(\varphi)$ of a formula $\varphi\in\mathcal{L}_{aggr}$ defined as follows:

- If φ is atomic, then $\mathrm{rank}(\varphi)=0$.
- If $\varphi=\neg\psi$, then $\mathrm{rank}(\varphi)=\mathrm{rank}(\psi)$.
- If $\varphi=\bigvee_{i\in I}\psi_i$, then $\mathrm{rank}(\varphi)=\sup_{i\in I}\mathrm{rank}(\psi_i)$;
- If $\varphi=\bigwedge_{i\in I}\psi_i$, then $\mathrm{rank}(\varphi)=\sup_{i\in I}\mathrm{rank}(\psi_i)$;
- If $\varphi=(\exists i\,x)\psi$, then $\mathrm{rank}(\varphi)=\mathrm{rank}(\psi)+1$.

LEMMA 11. *Let $\mathcal{Q}$ be an arbitrary class of monadic quantifiers. For each formula $\varphi\in\mathrm{FO}(\mathcal{Q}\cup\mathcal{Q}^b\cup\mathcal{Q}^c)$ there exists an equivalent formula $\varphi'\in\mathcal{L}_C$.*

PROOF. The lemma is by induction on the syntax of φ.

- Formulae without quantifiers trivially have an equivalent in $\mathcal{L}_C$.
- If $\varphi=(Q\,x)\,\varphi_1$, then we take

$$\varphi' := \bigvee_{i\in rng(Q)}(\exists i\,x)\,\varphi_1'(x),$$

  where φ_1' is the $\mathcal{L}_C$ formula equivalent to φ_1, which exists by the induction hypothesis.

  Observe that, by the translation so far, if φ does not contain any branching or cumulation, φ' is always of finite rank.

- If $\varphi=\big((Q_1;Q_2)\,x_1,x_2\big)\,\varphi_1'(x_1,x_2,\overline{z})$, then ϕ' is the formula:

$$\bigvee_{i\in rng(Q_1)}\bigvee_{j\in rng(Q_2)}(\exists x_1)\ldots(\exists x_i)(\exists y_1)\ldots(\exists y_j)$$
$$\Big(\bigwedge_{1\le m<n\le i}x_m\neq x_n\Big)\wedge\Big(\bigwedge_{1\le m<n\le i}y_m\neq y_n\Big)$$
$$\wedge\Big(\bigwedge_{k\in\{1,\ldots,i\},l\in\{1,\ldots,j\}}\varphi_1'(x_k,y_l,\overline{z})\Big),$$

  where φ_1' is the $\mathcal{L}_C$ formula equivalent to φ_1, which exists by the induction hypothesis. This translation clearly generalizes to arbitrary branching formulae $\varphi=\big((Q_1;\ldots;Q_k)\,x_1,\ldots,x_k\big)\,\varphi_1'(x_1,\ldots,x_k,\overline{z})$.

- If $\varphi=([Q_1;Q_2]\,x_1,x_2)\,\varphi_1(x_1,x_2,\overline{z})$, then ϕ' is the formula

$$\bigvee_{i\in rng(Q_1)}\bigvee_{j\in rng(Q_2)}(\exists x_1)\ldots(\exists x_i)(\exists y_1)\ldots(\exists y_j)$$
$$\Big(\bigwedge_{1\le m<n\le i}x_m\neq x_n\Big)\wedge\Big(\bigwedge_{1\le m<n<j}y_m\neq y_n\Big)$$
$$\wedge\Big(\bigwedge_{1\le k\le i}\bigvee_{1\le l\le j}\varphi_1'(x_k,y_l,\overline{z})\Big)$$
$$\wedge\Big(\bigwedge_{1\le l\le j}\bigvee_{1\le k\le i}\varphi_1'(x_k,y_l,\overline{z})\Big),$$

  where φ_1' is the $\mathcal{L}_C$ formula equivalent to φ_1, which exists by the induction hypothesis. Again, this translation clearly generalizes to arbitrary cumulation formulae $\varphi=\big((Q_1;\ldots;Q_k)\,x_1,\ldots,x_k\big)\,\varphi_1'(x_1,\ldots,x_k,\overline{z})$. □

Note that the quantifier rank of the translating sentence for the last two cases depends on the quantifiers, i.e. it is unbounded for unbounded quantifiers. In contrast, the translation for formulae φ with only linear prefixes *always* yields an equivalent bounded-rank formula φ'. A better understanding of the expressiveness of $\mathrm{FO}(\mathcal{Q}\cup\mathcal{Q}^b\cup\mathcal{Q}^c)$ may therefore perhaps be gained from studying the unbounded rank formulae of $\mathcal{L}_C$ that have no equivalent formula of bounded rank.

5. PRACTICAL CONSIDERATIONS

Let us now consider branching and cumulation from a query evaluation point of view. In particular, suppose that, for some given database D and some *fixed* monadic finite-model generalized quantifiers Q_1 and Q_2 we want to check that

$$D\models((Q_1;Q_2)\,x,y)\,R(x,y)$$

where R is a binary relation name. Let $m_1=\min(Q_1)$ and $m_2=\min(Q_2)$. By the reasoning in the proof of Proposition 7, this reduces to checking

$$D\models(\exists x_1)\ldots(\exists x_{m_1})\,(\exists y_1)\ldots(\exists y_{m_2})\,\varphi$$

where φ is the formula

$$\bigwedge_{1\le i<j\le m_1}(x_i\neq x_j)\;\wedge\bigwedge_{1\le i<j\le m_2}(y_i\neq y_j)\;\wedge\bigwedge_{\substack{1\le i\le m_2\\1\le j\le m_2}}R(x_i,y_j).$$

The naive way to evaluate this formula is by nested iteration over the active domain $\mathrm{adom}(D)$ of D: iterate over all $\overline{a}\in\mathrm{adom}(D)^{m_1}$ and all $\overline{b}\in\mathrm{adom}(D)^{m_2}$ and check that $D\models\varphi[\overline{a},\overline{b}]$. This method clearly requires $\Omega(n^{m_1+m_2})$ time, where n is the size of the relation $D(R)$ assigned to R by D.

There is, however, a more efficient way to evaluate this formula. Indeed, the following algorithm does so in time $O(n^{m_1}\log n^{m_1})$.

1. Compute

$$S:=\{(a_1,\ldots,a_{m_1},b)\mid\text{ every }(a_j,b)\in D(R)\}.$$

   This is essentially an m_1-times self-join of $D(R)$ on its second column,

$$S=\pi_{1,3,\ldots,2m_1-2,2m_1-1,2m_1}\big(\sigma_\theta(\underbrace{R\times\cdots\times R}_{m_1\text{ times}})\big)$$

with θ the condition

$$(2 = 2m_1) \wedge (4 = 2m_1) \wedge \cdots \wedge (2(m_1 - 1) = 2m_1).$$

As such, S can be computed in time $O(n^{m_1})$ using standard techniques [40].

2. Fix an arbitrary order $<$ on $\mathrm{adom}(D)$ and sort S lexicographically according to $<$. Using standard sorting techniques, this can be done in time $O(n^{m_1} \log n^{m_1})$ since the cardinality of S is at most n^{m_1}. After sorting, all records that agree on the first m_1 components of S are grouped together.

3. It is now easy to check the existence of distinct $a_1, \ldots, a_{m_1}$ and distinct $b_1, \ldots, b_{m_2}$ such that every $(a_i, b_j) \in D(R)$. Indeed, it suffices initialize a counter c to 0 and scan S: as long as the first m_1 components of the current tuple equal that of the previous tuple we increment c. Otherwise we reset $c := 1$. The check then succeeds iff c ever becomes m_2 during the scan. Since S contains at most $O(n^{m_1})$ tuples, this scan can be done in time $O(n^{m_1})$.

In conclusion, we can compute S in time $O(n^{m_1})$, sort S in time $O(n^{m_1} \log n^{m_1})$, and do the final scan in time $O(n^{m_1})$. Hence, we can check $D \models ((Q_1; Q_2)\, x, y)\, R(x, y)$ in time $O(n^{m_1} \log n^{m_1})$. Observe that, since $D \models ((Q_1; Q_2)\, x, y)\, R(x, y)$ iff $D \models ((Q_2; Q_1)\, y, x)\, R(x, y)$, the above algorithm implies that $D \models ((Q_1; Q_2)\, x, y)\, R(x, y)$ can be checked in time $O(n^m \log n^m)$ with $m = \min(m_1, m_2)$.

It is interesting to observe that this algorithm can also be mimicked in SQL. For example, if $m_1 = 3$, the following SQL query returns a non-empty answer iff $D \models ((Q_1; Q_2)\, x, y)\, R(x, y)$. Here, A denotes the first column of R and B the second.

SELECT *
FROM $R\ R_1$, $R\ R_2$, $R\ R_3$
WHERE $R_1.A$ <> $R_2.A$ AND $R_1.A$ <> $R_3.A$
  AND $R_2.A$ <> $R_3.A$ AND $R_1.B = R_2.B$
  AND $R_1.B = R_3.B$ AND $R_2.B = R_3.B$
GROUP BY $R_1.A$, $R_2.A$, $R_3.A$
HAVING COUNT(DISTINCT B)$\geq n$

Although the $O(n^{m_1} \log n^{m_1})$ algorithm provides a significant improvement on the naive $O(n^{m_1+m_2})$ algorithm, it is unclear whether it is usable in practice. For instance, if $D(R)$ is a fact table in a data warehouse an m_1-fold self join for $m_1 \geq 4$ is simply out of the question for current systems due to the enormous size of such tables. This hence raises the question whether a more efficient evaluation algorithm for branching quantification exists. We strongly suspect that this is not the case when the algorithm is to be expressed in SQL in the sense that any SQL expression checking $D \models ((Q_1; Q_2)\, x, y)\, R(x, y)$ must use at least $m = \min(m_1, m_2)$ self-joins, despite the fact that SQL expression can use grouping and aggregation.

The question of a more efficient evaluation algorithm is perhaps even more important for cumulation quantification. Indeed, by the reasoning in the proof of Proposition 8, checking whether $D \models ([Q_1; Q_2]\, x, y)\, R(x, y)$ with $\mathrm{rng}(Q_1) = \{m_1\}$ and $\mathrm{rng}(Q_2) = \{m_2\}$ reduces to checking

$$D \models (\exists x_1) \ldots (\exists x_{m_1})\, (\exists y_1) \ldots (\exists y_{m_2})\, \varphi'$$

where φ' equals

$$\varphi'(\overline{x}, \overline{y}) := \bigwedge_{1 \leq i < j \leq m_1} x_i \neq x_j \ \wedge \bigwedge_{1 \leq i < j \leq m_2} (y_i \neq y_j) \ \wedge \bigwedge_{1 \leq i \leq m_1} \bigvee_{1 \leq j \leq m_2} R(x_i, x_j) \ \wedge \bigwedge_{1 \leq j \leq m_2} \bigvee_{1 \leq i \leq m_1} R(x_i, x_j)$$

Again, the naive way to evaluate this formula by nested iteration runs in time $\Omega(n^{m_1+m_2})$. It is unclear, however, whether a more efficient evaluation algorithm exists.

6. RELATED WORK

The study of non-linear prefixes has a long tradition both in logic and in formal linguistics. We overview some of this work as it has provided the background for the present research. We also briefly touch on work in SQL optimization which is somewhat related to our paper. There is, of course, a very large body of research in descriptive complexity using generalized quantification [9, 14, 15, 26, 29, 41] which will not further be mentioned here.

In this paper we have used the simplest type of quantifiers. In general, a *type* is a finite sequence of natural numbers $t = (k_1, \ldots, k_n)$. A global generalized quantifier of type t assigns, to each non-empty universe A a local generalized quantifier of type t. A local generalized quantifier of type $t = (k_1, \ldots, k_n)$ on A is a class of structures of type $< A, R_1, \ldots, R_n >$ with every R_i of arity k_i, that is closed under isomorphism. For instance, a type $(1, 2)$ means that the quantifier has two arguments: a set and a binary relation. Syntactically, the quantifier would bind 3 variables in 2 formulas, the first variable coming from the first formula and the last 2 variables from the second formula. Thus, type (1) is the simplest type, and is equivalent to having arity 1 – the type of quantifier we have used. In general, richer types yield more powerful quantifiers. The result of years of research in the subject has shown that there is a hierarchy of generalized quantifiers based on type. To be precise, we can define a *lexicographic* order on types:

$$(1) < (1,1) < \ldots < (2) < (2,1) < (2,1,1) < \ldots < (2,2) < \ldots < (3) < \ldots$$

Then for every type t there is a quantifier of type t that is not definable in $FO(\mathcal{Q})$ where all quantifiers in $\mathcal{Q}$ are of some type $t' < t$ [14, 15]. Quantifiers of types (1), (1, 1), $(1, 1, 1), \ldots$ are usually called *monadic* and are the only ones that can be captured by simple cardinality properties. Our choice of quantifiers of type (1) as an initial step in the study of non-linear prefixes is motivated by the fact that these are the simplest type, but at the same time it is already known that branching of iterations of such quantifiers leads to high complexity.

Branching quantifiers were introduced by Leon Henkin [17]. The classic theory was developed, among others, by [10, 42]. Early work concentrated on first order logic; even though it produced some well known results, such accomplishments are limited to combinations of $\forall, \exists$ prefixes. As stated in the Introduction, when non-linear *branching* quantifiers are allowed, formulas can be obtained that are not expressible in first order logic [6]. For instance, the formula

$$(\forall x\ \exists y; \forall z\ \exists w) x, y, z, w\ \varphi(x, y, z, w)$$

cannot be transformed into any linear prefix. The interest

in such formulas comes from the following three facts: such branching formulas seem to be needed to faithfully formalize some English sentences [6, 19]; allowing such branching formulas greatly increases the power of the language [42]; and branching formulas like the one above can be easily represented with generalized quantifiers -although not monadic ones. The formula above can be captured by the generalized quantifier H of type [4] defined by

$$\{S \in \mathcal{U}^4 \mid \exists f_1 : \mathcal{U} \to \mathcal{U}, \exists f_2 : \mathcal{U} \to \mathcal{U} \\ \forall a\ \forall b\ (a, f_1(a), b, f_2(b)) \in S\}$$

usually called the *Henkin quantifier*. Since this is also one of the simplest examples of non-first order definable properties, this quantifier and its properties has been studied in depth [18]. The main result about Henkin quantifiers is the Blass and Gurevich result [5], asserting that not only is every Henkin quantifier sentence equivalent to an existential second-order sentence, but also the converse holds. This means that known results about the logic of existential second-order sentences immediately apply to the logic of Henkin quantifier sentences: the Lowenheim-Skolem theorem, the compactness theorem, the definability of truth for sentences of this class by a sentence of the class, and more. Other early studies include [25], which proved the *Linear Prefix Theorem*: for each two different quantifier prefixes Q_1 and Q_2 of the same length there is a Q_1 sentence (i.e. a sentence made up of Q_1 as a prefix) that is not equivalent to any Q_2 sentence. In addition, [42] shows that in the case of first order logic prefixes are equivalent (in the sense of semantically indistinguishable) iff they only differ in variables, but order and type of quantifier are preserved. More precisely, given a linear prefix of quantifiers $Q_1 \ldots Q_n$, each Q_i equal to $\forall$ or $\exists$, call a subsequence of quantifiers of the same type a *block*. For instance, in

$$(\exists x_1)\,(\exists x_2)\,(\forall x_3)\,(\exists x_4)\,(\exists x_5)\,(\forall x_5)\,(\forall x_7)\,(\forall x_8)$$

there are 4 blocks, the first one including the first two quantifiers, the second one including only the third quantifier, the third block corresponding to the 4th and 5th quantifiers, and the fourth block corresponding to the last 3 quantifiers. Call two prefixes equivalent up to blocks if they have the same number and type of blocks, in the same order, and each block is of the same length as its corresponding block in the other prefix. Thus, quantifiers can be moved around only *within their block* (i.e. permutations inside a block are allowed). Then, two prefixes are equivalent iff they are equivalent up to blocks. Thus, the above prefix is equivalent to

$$(\exists x_2)\,(\exists x_1)\,(\forall x_3)\,(\exists x_5)\,(\exists x_4)\,(\forall x_7)\,(\forall x_8)\,(\forall x_5)$$

These results are extended to the case of generalized quantifiers in [22].

In formal linguistics, it was debated early whether nonlinear quantifiers are needed for representing natural language. Hintikka [19] was the first to argue so, but his examples were very controversial, and many linguists remained unconvinced. Barwise [6] gave examples that included generalized quantifiers (Hintikka's examples were restricted to $\forall$, $\exists$) and gathered more support. Nowadays, there is broad quorum that *cumulative readings* are indeed necessary [35]; even though it is still argued whether pure *branching readings* are needed, but most researchers seem to be of this opinion.

Westerstahl [44] studies *iterated quantifiers*, quantifiers that are defined by a prefix of generalized quantifiers. As an example, the sentence "Most critics reviewed two books" can be modeled by the sentence $(\mathsf{most}\, x)\,(\mathsf{two}\, y)\,\mathrm{Review}(x, y)$.[1] One can, however, define a new quantifier $\mathsf{most} \circ \mathsf{two}$ of type (2) as the composition of the two quantifiers above. Note that there is another linear reading of the example sentence, in which the same two parts are involved. This reading is expressed by permuting the quantifiers order (and hence reversing the scope) and creates another new quantifier. The interesting fact is that there are quantifiers that cannot be expressed as an iteration of simpler quantifiers; for instance "every boy likes a different girl" and "every student criticized himself" are two examples from [21] of complex quantifiers that cannot be broken down into two (a combination of "every" and "different" for the first example and of "every" and "himself" for the second)[2]). Keenan and Westerstahl [24] also study iteration of quantifiers from a formal linguistics perspective. A generalized quantifier is created by considering the iteration of n quantifiers to give rise (and define) a new quantifier. Here prefixes of generalized quantifiers are assumed to have the traditional (linear, left-to-right) scope. Several properties of iterations are studied; for instance, if all quantifiers in an iteration are (upward, downward) monotone, then the quantifier that the iteration defines is also (upward, downward) monotone. Spaan [38] gives conditions under which to distinguish cumulative and branching readings, but they are not necessary and sufficient – the problem remains open.

As a final note, we have pointed in Section 5 that the improved evaluation algorithm for branching based on sorting and grouping has a counterpart in SQL. Adding generalized quantifiers to SQL has been proposed [33, 20], although such proposals are focused on simplifying query formulation and use a fixed, finite set of type $(1, 1)$ quantifiers. Even in such cases, issues of safety and domain independence arise and must be dealt with [3]. Nevertheless, using generalized quantification in query languages offers interesting possibilities for query processing and optimization [2, 4].

7. CONCLUSION AND FURTHER WORK

We have taken the point of view that there are two natural extensions of FO with regard to its quantification power: by means of generalized quantifiers and by means of non-linear prefixes. We have intuitively argued that non-linear prefixes such as branching and cumulation are useful in query languages as they naturally occur in the succinct formalization of English queries. Furthermore, we have shown that branching and cumulation do not extend the power of FO when only finite-model respectively bounded monadic quantifiers are involved.

Clearly, we have only scratched the surface here. The exact expressiveness of cumulation of finite-model as opposed to bounded quantifiers is still to be determined, as is the expressiveness of $\mathrm{FO}(\mathcal{Q} \cup \mathcal{Q}^b \cup \mathcal{Q}^c)$ when $\mathcal{Q}$ contains arbitrary monadic quantifiers. Also, while our results imply that it

[1] The quantifier most can be defined in several ways; a very common one is to set it equal to $\{R \in \mathcal{U} \mid |R| > |\mathcal{U} - R|\}$.

[2] Recall that generalized quantifiers have been used in formal linguistics to model the behavior and properties of English determiners and other words classes that may work in a similar function. Hence, for a linguist, "different" and "himself" can also be seen as generalized quantifiers.

is in principle possible to add branching and cumulation (of bounded quantifiers) as syntactic sugar to practical query languages, there is still the question whether more efficient evaluation algorithms exist that can be expected to work well in practice. While this leaves the current article with many questions, we hope nevertheless to have been able to convince the reader that non-linear prefixes are interesting from a database point of view.

8. REFERENCES

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

[2] Badia, A., *A Family of Query Languages with Generalized Quantifiers: its Definition, Properties and Expressiveness*, Ph.d. thesis, Indiana University, 1997.

[3] Badia, A. *Safety, Domain Independence and Generalized Quantification*, *Data and Knowledge Engineering*, vol. 38, n. 2, August 2001, pages 147-172, Elsevier Publishers.

[4] Badia, A. and Cao, B. *Processing and Optimization of Generalized Quantification Queries*, under submission.

[5] A. Blass and Y. Gurevich *Henkin quantifiers and complete problems*, Annals of Pure and Applied Logic, 32(1):1-16, 1986.

[6] Barwise, J. *On Branching Quantifiers in English*, *Journal of Philosophical Logic*, v. 8, 1979, pages 47-80

[7] Barwise, J. and R. Cooper, *Generalized Quantifiers and Natural Language*, *Linguistic and Philosophy*, volume 4, 1981.

[8] *Generalized Quantifiers in Natural Language*, van Benthem, J. and ter Meulen, A., eds. Foris Publications, 1985.

[9] Dawar, A. and Hella, L., *The Expressive Power of Finitely Many Generalized Quantifiers*, in Proceedings of the 9th IEEE Symposium on Logic In Computer Science, 1994.

[10] Enderton, H. *Finite Partially-Ordered Quantifiers*, in *Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik*, vol. 16 (1970), pp. 393-397.

[11] Enderton, H. *A Mathematical Introduction to Logic*, Academic Press, 1972.

[12] Grumbach, S. and Tollu, C., *Query Languages with Counters*, in Proceedings of the International Conference on Database Theory, Lecture Notes in Computer Science, number 333, Springer-Verlag, 1992.

[13] Gyssens, M., Van Gucht, D. and Badia, A., *Query Languages with Generalized Quantifiers*, in *Application of Logic Databases*, Ramakrishnan, Ragu ed., Kluwer Academic Publishers, 1995.

[14] Hella, L., *Definability Hierarchies of Generalized Quantifiers*, Annals of Pure and Applied Logic, volume 43, 1989.

[15] Hella, L., Luosto, K. and Vaananen, J. *The Hierarchy Theorem for Generalized Quantifiers*, J. Symbolic Logic, 61(3), p. 802-817, 1996.

[16] Hella, L., Libkin, L., Nurmonen, J. and Wong, Limsoon *Logics with Aggregate Operators*, Journal of the ACM, 48(4), 2001, pages 880-907.

[17] Leon Henkin, *Some Remarks on Infinitely Long Formulas*, in *Infinitistic Methods*, Warsaw, 1959, pp. 167-183.

[18] Herre, L., Krynicki, M., Pinus, A. and Vaananen, J. *The Hartig Quantifier: A Survey*, The Journal of Symbolic Logic, 56(4), 1991, pages 1153-1183.

[19] Hintikka, J. *Quantifiers vs. Quantification Theory*, Linguistic Inquiry, 5(2), 1974, pages 153-177.

[20] Hsu, P. Y. and Parker, D. S., *Improving SQL with Generalized Quantifiers*, in Proceedings of the Tenth International Conference on Data Engineering, 1995.

[21] Keenan, E. *Beyond the Frege Boundary*, *Linguistics and Philosophy*, v. 15, 1992.

[22] Keenan, E. *Natural Language, Sortal Reducibility and Generalized Quantifiers*, *Journal of Symbolic Logic*, v. 58, n. 1, 1993, pages 314-325.

[23] Keenan, E. and Moss, L., *Generalized Quantifiers and the Expressive Power of Natural Language*, in [8].

[24] Keenan, E. and Westerstahl, D., *Generalized Quantifiers in Linguistics and Logic*, in [8].

[25] Keisler, J. and Walkoe, W. *The Diversity of Quantifier Prefixes*, *Journal of Symbolic Logic*, v. 38, pages 79-85, 1973.

[26] Kolaitis, P. and Väänänen, J., *Generalized Quantifiers and Pebble Games on Finite Structures*, Annals of Pure and Applied Logic, Elsevier, v. 74, 1995, pages 23-75.

[27] Libkin, L. *Expressive Power of SQL*, Theoretical Computer Science, 296 (2003), pages 379-404.

[28] Lindstrom, P., *First Order Predicate Logic with Generalized Quantifiers*, Theoria, volume 32, 1966.

[29] Luosto, K. *Hierarchies of Monadic Generalized Quantifiers*, Journal of Symbolic Logic, 65(3), 2000, pages 1241-1263.

[30] Mostowski, A., *On a Generalization of Quantifiers*, Fundamenta Mathematica, volume 44, 1957.

[31] Pirotte, A., *High Level Data Base Query Languages*, in *Logic and Databases*, Gallaire and Minker, eds., Plenum Press, 1978, pages 409-436.

[32] Quantz, J. *how to Fit Generalized Quantifiers into Terminological Logics*, Proceedings of the 10th European Conference on Artificial Intelligence, 1992.

[33] Rao, S. and Van Gucht, D. and Badia, A., *Providing Better Support for a Class of Decision Support Queries*, Proceedings of the ACM SIGMOD International Conference on Management n of Data, Montreal, Canada, 1996, pages 217–227.

[34] Ross, K. and Chatziantoniou, D., *Groupwise Processing of Relational Queries*, in Proceedings of the 23rd VLDB Conference, 1997.

[35] Scha, R. *Distributive, Collective and Cumulative Quantification*, in *Truth, Interpretation and Information; selected papers from the 3rd Amsterdam Colloquium*, Dordrecht, Holland, 1984, pages 131-158.

[36] M. Sevenster, *Henkin Quantifiers: Logic, Games, and Computation*, The bulletin of the EATCS no

89, pages 136-155, June 2006.

[37] Sher, G. *Ways of Branching Quantifiers*, Linguistics and Philosophy, v. 13, 1990.

[38] Spaan, M. *Parallel Quantification*, in J. van der Does and J. van Eyck (eds.) *Generalized Quantifier Theory and Applications*, CSLI Lecture Notes, 1993.

[39] The TPC-H Benchmark, online at `http:\\www.tpc.org`.

[40] Ullman, J. D., *Principles of Database and Knowledge Base Systems*, Computer Science Press, 1988.

[41] Vaananen, J. *Unary Quantifiers on Finite Models*, Journal of Logic, Language and Information, v. 6, 1997, pages 275-304.

[42] Walkoe, W. *Finite Partially-Ordered Quantification*, *Journal of Symbolic Logic*, v. 35, n. 4, pages 535-555, 1970.

[43] Westerstahl, D., *Quantifiers in Formal and Natural Languages*, in *Handbook of Philosophical Logic*, volume IV, Gabbay, D. and Guenther, F., eds., Reidel Publishing Company, 1989.

[44] Westerstahl, D. *Iterated Quantifiers*, in *Dynamics, Polarity and Quantification*, Kanazawa and Pinon, eds., CSLI, Stanford University, 1994, pages 173-209.

[45] Westerstahl, D. *Branching Quantifiers and Natural Language*, in Garderfons, P. (ed.) *Generalized Quantifiers: Linguistic and Logical Approaches*, Dordrecht, 1987.