Local Properties of Query Languages

Guozhu Dong[1] **Leonid Libkin**[2] **Limsoon Wong**[3]

[1] Dept of Computer Science, University of Melbourne, Parkville, Vic. 3052, Australia, Email: dong@cs.mu.oz.au
[2] Bell Laboratories/Lucent Technologies, 600 Mountain Avenue, Murray Hill, NJ 07974, USA, Email: libkin@research.bell-labs.com
[3] BioInformatics Center & Institute of Systems Science, Singapore 119597, Email: limsoon@iss.nus.sg

Abstract. Expressiveness of database query languages remains the major motivation for research in finite model theory. However, most techniques in finite model theory are based on Ehrenfeucht-Fraisse games, whose application often involves a rather intricate argument. Furthermore, most tools apply to first-order logic and some of its extensions, but not to languages that resemble real query languages, like SQL.

In this paper we use *locality* to analyze expressiveness of query languages. A query is local if, to determine if a tuple belongs to the output, one only has to look at a certain predetermined portion of the input.

We study local properties of queries in a context that goes beyond the pure first-order case, and then apply the resulting tools to analyze expressive power of SQL-like languages. We first prove a general result describing outputs of local queries, that leads to many easy inexpressibility proofs. We then consider a closely related *bounded degree* property, which describes the outputs of queries on structures that locally look "simple," and makes inexpressibility proofs particularly easy. We prove that every local query has this property. Since every relational calculus (first-order) query is local, these results can be viewed as "off-the-shelf" strategies for inexpressibility proofs, which are often easier to apply than the games. We also show that some generalizations of the bounded degree property that were conjectured to hold, fail for relational calculus.

We then prove that the language obtained from relational calculus by adding grouping and aggregates (essentially plain SQL), has the bounded degree property, thus solving an open problem. Consequently, first-order queries with Härtig and Rescher quantifiers have the bounded degree property. Finally, we apply our results to show that SQL and relational calculus are incapable of maintaining the transitive closure view even in the presence of certain kinds of auxiliary data.

1 Introduction

One major issue in the study of database query languages is their expressive power. Given a query language, it is important to know if the language has enough power to express certain queries. Most database languages have limited power; for example, the relational calculus and algebra cannot express the transitive closure of a graph or the parity test. A large number of tools have been developed for first-order logic (or equivalently, the relational calculus); these include Ehrenfeucht-Fraisse games [1, 13], locality [13, 16], 0-1 laws [1, 13], Hanf's technique [15], the bounded degree property [23]. We are especially interested in local properties of queries, first introduced by Gaifman [16]. These state that the result of a query can be determined by looking at "small neighborhoods" of its arguments.

Expressiveness of database query languages remains the major motivation for research in finite model theory. However, most of those tools developed are modified Ehrenfeucht-Fraisse games, whose application often involves a rather intricate argument. Furthermore, most current tools are applicable only to first-order logic and some of its extensions (like fragments of second-order logic [15], infinitary logics [5], logics with counting [20], etc.); but they do not apply to languages that resemble real query languages, like SQL.

The goal of this paper is to give a thorough study of local properties of queries in a context that goes beyond the pure first-order case, and then apply the resulting tools to analyze expressive power of SQL-like languages.

Languages like SQL differ from the relational calculus in that they have grouping constructs (modeled by the SQL `GROUPBY`) and aggregate functions such as `COUNT` and `AVG`. After some initial investigation of extended relational languages was done in [21, 25], first results on expressive power appeared in [8]. However, the results of [8] were based on the assumption that the deterministic and nondeterministic logspace are different, and thus questions on expressive power of SQL-like languages remained open.

In the past few years, several researchers explored the connection between relational languages with aggregate functions and languages whose main data structures are bags rather than sets. Among the issues that were studied are interdefinability of their primitives [4, 22, 18], complexity [18], optimization [7], equational theories [17] and, finally and most recently, the limitations of their expressive power [23, 24]. In particular, it was shown in [23] that the transitive closure of a graph remains inexpressible even when grouping and aggregation are added to the relational calculus. For a survey of this area, see [19].

Since there was no tool available for studying languages with aggregate functions, in [23] we tried to find a property possessed by the queries in our language, which is not possessed by the transitive closure of a graph. Let a query q take a graph as an input and return a graph. Then we say that it has the *(graph) bounded degree property* if for any k, if all in- and out-degrees in an input graph G do not exceed k, then the number of distinct in- and out-degrees in the output graph $q(G)$ is bounded by some constant c, that depends only on k and q, and not on the graph G. It is clear that the transitive closure query violates this property: just look at the transitive closure of a chain graph.

We have been able to prove that the bounded degree property holds for every relational calculus graph query [23]. We have also demonstrated that it is a very convenient tool for establishing expressivity bounds, often much easier to apply than the games or other tools. However, we were not able to prove in [23] that it extends to languages with aggregation. Instead, we showed inexpressibility of the transitive closure in such a language by a direct brute-force argument, analyzing the properties of queries restricted to special classes of inputs (multicycles).

The question of whether relational calculus with grouping and aggregate functions has the bounded degree property was the main open problem left in [23]. We also mentioned a possible approach towards solving this problem. The proof of the bounded degree property for relational calculus was based on Gaifman's result that first-order formulae are *local*, in the sense as defined in [16]. The locality result in [16] has two parts, and only one was used in our proof in [23]. It says that in order to determine if a formula $\phi(\vec{x})$ is satisfied on a tuple $\vec{a}$, one only has to look at a small neighborhood of $\vec{a}$ of a predetermined size. (The second part deals with sentences, and is irrelevant for the discussion here.) Thus, we thought that it is of interest to give a general study of queries that satisfy this notion of locality.

The purpose of this paper is twofold. First, we give a general study of local queries, their expressive power, and more general notions of the bounded degree property. Second, we prove locality of certain queries in an SQL-like language and show that this is enough to confirm that it has the bounded degree property.

Organization In the next section, we introduce the notations in such a way that the presentation of the results about locality and bounded degree properties can be applied to a number of different languages, including first-order logic and some of its extensions. We give a formal definition of local queries, and note that every relational calculus query is local.

In Section 3, we prove the main result about expressiveness of local queries. We show that the number of different in- and out-degrees realized in the output of a graph query on an arbitrary structure is bounded above by the number of nonisomorphic neighborhoods realized in the input structure, such that the radius of these neighborhoods depends only on the query. We demonstrate some expressiveness bounds that immediately follow from this result.

The main result of Section 4 is that every local query has the bounded degree property. We also show how this result can be used to establish expressiveness bounds in the presence of some auxiliary data.

In Section 5 we look at some generalizations of the bounded degree property that one migh expect to be true, and show that they fail even for first-order graph queries.

In Section 6, we introduce a theoretical SQL-like language that extends relational calculus with grouping and aggregate functions, and prove that it is local when restricted to unordered flat relations whose degrees are bounded by a constant. Therefore, the language has the bounded degree property over flat relations without ordering on the domain elements. This implies that it cannot express the transitive closure. It also follows that first-order queries with Härtig and Rescher (equicardinality and majority) quantifiers have the bounded degree property. In Section 7 we apply our results to incremental maintenance of views, and show that SQL and relational calculus are incapable of maintaining the transitive closure view even in the presence of certain kinds of auxiliary data.

Complete proofs of all the results can be found in [10].

2 Notations

We study queries on finite relational structures. A relational signature τ is a set of relation symbols $\{R_1, ..., R_l\}$, with an associated arity function. In what follows, $p_i(>0)$ denotes the arity of R_i. By τ_n we mean τ extended with n new constant symbols. We use graphs in many examples; we denote the signature of graphs by τ_{gr}, which consists of one binary predicate (for the edges).

A structure is written as $\mathcal{A} = \langle A, \overline{R}_1, \ldots, \overline{R}_l\rangle$, where A is a finite set called the carrier and $\overline{R}_i$ is the interpretation of R_i, which is a subset of A^{p_i}. The class of τ-structures is denoted by STRUCT$[\tau]$. When no confusion can arise, we write R_i in place of $\overline{R}_i$. We use the symbol $\cong$ to denote isomorphism of structures.

We would like to make our results general enough to apply to a variety of languages. To this end, we assume that a **query** is a formula $\psi(x_1, \ldots, x_m)$, where x_1, ..., x_m are free variables. We also assume the notion of $\models$ between structures and formulas. (You may think of ψ as a first-order formula in the language of τ, and $\models$ as the usual satisfaction relation.) Associated with a query $\psi(x_1, \ldots, x_m)$ is a mapping Ψ of structures from STRUCT$[\tau]$ to STRUCT$[S_m]$, where S_m is a symbol of arity m, defined by $\Psi(\mathcal{A}) = \langle A, \{(a_1, \ldots, a_m) \in A^m | \ \mathcal{A} \models \psi(a_1, \ldots, a_m)\}\rangle$. If $m = 2$, the output of a query is a graph, and we speak about **graph queries**. For convenience, queries are denoted by lower case Greek letters; the associated mappings of structures are denoted by the corresponding upper case Greek letters.

The following definitions are quite standard; see [13, 16]. Given a structure $\mathcal{A}$, its **graph** $\mathcal{G}(\mathcal{A})$ is defined as $\langle A, E\rangle$ where (a, b) is in E iff there is a tuple $\vec{t} \in \overline{R}_i$ for some i such that both a and b are in $\vec{t}$. It is also called the **Gaifman graph** of a structure, cf. [15]. The distance $d(a, b)$ is defined as the length of the shortest path from a to b in $\mathcal{G}(\mathcal{A})$. Note that the triangle inequality holds: $d(a, c) \leq d(a, b)+d(b, c)$. Given $a \in A$, its r**-sphere** $S_r(a)$ is $\{b \in A \mid d(a, b) \leq r\}$. Note that $a \in S_r(a)$. For a tuple $\vec{t}$, $S_r(\vec{t}) = \bigcup_{a\in\vec{t}} S_r(a)$.

Given a tuple $\vec{t} = (t_1, \ldots, t_n)$, its r**-neighborhood** $N_r(\vec{t})$ is defined as a τ_n structure

$$\langle S_r(\vec{t}), \overline{R}_1 \cap S_r(\vec{t})^{p_1}, \ldots, \overline{R}_k \cap S_r(\vec{t})^{p_k}, t_1, \ldots, t_n\rangle$$

That is, the carrier of $N_r(\vec{t})$ is $S_r(\vec{t})$, the interpretation of the relations in τ is obtained by restricting them to the carrier, and the n extra constants are the elements of $\vec{t}$.

Given a structure $\mathcal{A}$, we define an equivalence relation $a \approx_d b$ iff $N_d(a) \cong N_d(b)$. We also define $\mathsf{ntp}(d, \mathcal{A})$ to be the number of $\approx_d$ equivalence classes in $\mathcal{A}$. That is, $\mathsf{ntp}(d, \mathcal{A})$ is the number of isomorphism types of d-neighborhoods in $\mathcal{A}$.

Now we can give our main definition.

Definition 1. Given a query $\psi(x_1, \ldots, x_m)$, its **locality index** is a number $r \in \mathbb{N}$ such that, for every $\mathcal{A} \in$ STRUCT$[\tau]$ and for every two m-ary vectors $\vec{a}$, $\vec{b}$ of elements of A, it is the case that $N_r(\vec{a}) \cong N_r(\vec{b})$ implies $\mathcal{A} \models \psi(\vec{a})$ iff $\mathcal{A} \models \psi(\vec{b})$. If no such r exists, the locality index is ∞. A query is **local** if it has a finite locality index. A language is **local** if every query in it is. □

Are there any interesting examples of local queries? An answer to this is provided by Gaifman's locality theorem [16] which implies, in our terminology, the following fact.

Fact 1 *Every first-order (relational calculus) query is local.* □

However, even the simplest fragment of second-order logic, monadic Σ_1^1, is not local. It is not hard to construct a nonlocal query using connectivity test for undirected graphs, which is definable in monadic Σ_1^1 [3].

We shall see later that there are other interesting examples of local queries, though restricted to some classes of structures. We define these restricted classes of structures below. They play a central role in the paper.

For a graph G, its **degree set** $deg\_set(G)$ is the set of all possible in- and out-degrees that are realized in G. By $deg(G)$ we denote the cardinality of $deg\_set(G)$; that is, the number of different in- and out-degrees realized in G. We also define similar notions for arbitrary structures. Given a relation $\overline{R_i}$ in a structure $\mathcal{A}$, $degree_j(R_i, a)$ is the number of tuples in $\overline{R_i}$ whose jth component is a. Then $deg\_set(\mathcal{A})$ is defined as the set of all $degree_j(R_i, a)$ for $\overline{R_i} \in \mathcal{A}$ and $a \in A$. Finally, $deg(\mathcal{A})$ is the cardinality of $deg\_set(\mathcal{A})$.

The class of τ-structures $\mathcal{A}$ with $deg\_set(\mathcal{A}) \subseteq \{0, 1, \ldots, k\}$ is denoted by $\text{STRUCT}_k[\tau]$. We shall see that many queries in relational calculus augmented with grouping and arithmetic constructs (this is essentially plain SQL) are local when restricted to inputs from $\text{STRUCT}_k[\tau]$, for any fixed k. We also see from this that first-order queries with Härtig and Rescher quantifiers are local when restricted to the same structures.

As was mentioned before, a certain notion of uniform behavior of queries on $\text{STRUCT}_k[\tau_{\text{gr}}]$ was introduced earlier in [23]. We say that a graph query $\psi(x, y)$ has the **graph bounded degree property** if there exists a function $f : \mathbb{N} \to \mathbb{N}$ such that $deg(\Psi(G)) \le f(k)$ for any $G \in \text{STRUCT}_k[\tau_{\text{gr}}]$. It was shown in [23] that every first-order graph query has the graph bounded degree property.

3 Expressiveness of Local Queries

The goal of this section is to prove a general theorem characterizing outputs of local graph queries. Informally, our main result says this. If ψ is a local query, then the Gaifman graph of $\Psi(\mathcal{A})$ cannot be much more complex than the structure $\mathcal{A}$ itself. We first prove a theorem that states this result for graph queries. From this and a lemma that determines the locality rank of a query defining the Gaifman graph, we obtain our main result.

Recall that for any structure $\mathcal{A}$, the parameter $deg(\mathcal{A})$ shows how complex the structure looks globally. That is, how many different degrees are realized in it. The parameter $\mathsf{ntp}(d, \mathcal{A})$, for any fixed $d \ge 0$, shows how many distinct small neighborhoods are realized in $\mathcal{A}$. The first result of this section shows the connection between the parameter $\mathsf{ntp}(d, \cdot)$ on an input to a local *graph* query and the parameter $deg(\cdot)$ on the output. It can be interpreted as saying that output of a local graph query cannot be much more complex than its input.

Theorem 2. *Let $\psi(x, y)$ be a graph query on τ-structures of finite locality index r. Then for any $\mathcal{A} \in \mathrm{STRUCT}[\tau]$,*

$$deg(\Psi(\mathcal{A})) \;\leq\; 2 \cdot \mathsf{ntp}(3r+1, \mathcal{A})$$

In fact, the number of distinct in-degrees in $\Psi(\mathcal{A})$ is at most $\mathsf{ntp}(3r+1, \mathcal{A})$, and the number of distinct out-degrees in $\Psi(\mathcal{A})$ is at most $\mathsf{ntp}(3r+1, \mathcal{A})$.

Proof sketch. The key to our theorem is the following observation.

Lemma 3. *Let $r > 0$, $d \geq 3r+1$, and let $a \approx_d b$. Then there is a permutation π on $S_{d-r}(a, b)$ such that for every $x \in S_{d-r}(a, b)$, it is the case that $N_r(a, x) \cong N_r(b, \pi(x))$.*

To show how lemma 3 implies the theorem, let $G' = \langle V, E' \rangle$ be $\Psi(\mathcal{A})$. Let $d = 3r+1$. Let $a \approx_d b$. For every $x \notin S_{2r+1}(a, b)$, $N_r(a, x) \cong N_r(b, x)$, since $N_r(a) \cong N_r(b)$ and $d(a, x), d(b, x) > 2r+1$. Thus, $(a, x) \in E'$ iff $(b, x) \in E'$ by locality. Furthermore, by Lemma 3, for every $x \in S_{2r+1}(a, b)$, $(a, x) \in E'$ iff $(b, \pi(x)) \in E'$ by locality and the property of π. Hence a and b have the same outdegrees. A similar argument shows that a and b have the same indegrees. Hence $degset(G')$ has at most $2 \cdot \mathsf{ntp}(d, G)$ elements. $\square$

Let us give two simple applications to demonstrate the usefulness of Theorem 2 in establishing expressiveness bounds. The second of these will be generalized in the next section into a powerful result that lets us eliminate Ehrenfeucht-Fraisse games from many inexpressibility proofs.

Corollary 4. *No local query can define the transitive closure of a graph.*

Proof. Suppose $\psi(x, y)$ of locality index r defines the transitive closure. Consider chains, *i.e.* graphs of the form $C_n = \{(a_0, a_1), \ldots, (a_{n-1}, a_n)\}$ with all a_is distinct. Then $deg(\Psi(C_n)) = n+1$. For every $d \geq 0$, there are at most $2d$ non-isomorphic d-neighborhoods in a chain. Thus, $deg(\Psi(G)) \leq 4(3r+1)$, by Theorem 2. Hence, ψ cannot define the transitive closure. $\square$

Corollary 5. *Every local graph query has the graph bounded degree property.*

Proof. If all in- and out-degrees in G are bounded by k, then the maximum number of non-isomorphic d-neighborhoods depends only on k and d. Combining this with Theorem 2, we see that there is a bound on $deg(\Psi(G))$ that depends only on k and the locality index of ψ. $\square$

The statement of Theorem 2 is not completely satisfactory, since it only deals with graph queries. To generalize it to arbitrary queries, we look at the Gaifman graphs of the outputs. Recall that $\mathcal{G}(\mathcal{A})$ denotes the Gaifman graph of $\mathcal{A}$.

Theorem 6. *Let $\psi(x_1, \ldots, x_n)$, $n \geq 2$, be a query on τ-structures of finite locality index $r > 0$. Then there is a number m that depends only on n and r such that, for any $\mathcal{A} \in \mathrm{STRUCT}[\tau]$, the number of distinct degrees in the Gaifman graph of $\Psi(\mathcal{A})$ does not exceed $\mathsf{ntp}(m, \mathcal{A})$. In fact,*

$$deg(\mathcal{G}(\Psi(\mathcal{A}))) \;\leq\; \mathsf{ntp}(3^{n-1}r + (3^{n-1}-1)/2, \mathcal{A})$$

Proof sketch. We prove this theorem by reduction to graph queries. Given a query $\psi(x_1,\ldots,x_n)$, $n>2$, define $\psi'(x_1,\ldots,x_{n-1})$ by letting $\mathcal{A}\models\psi'(a_1,\ldots,a_{n-1})$ iff for some $a\in A$, and for some index $0\le i\le n-1$, it is the case that $\mathcal{A}\models\psi(a_1,\ldots,a_i,a,a_{i+1},\ldots,a_{n-1})$.

Lemma 7. *Let $\psi(x_1,\ldots,x_n)$ be of locality rank $r>0$. Then $\psi'(x_1,\ldots,x_{n-1})$ is of locality rank $3r+1$.*

To prove the theorem, first note that if $\psi(x,y)$ is a graph query of locality rank r, and $\psi^*(x,y)$ is such that $\mathcal{A}\models\psi^*(a,b)$ iff $\mathcal{A}\models\psi(a,b)$ or $\mathcal{A}\models\psi(b,a)$, then ψ^* also has locality rank r.

For an arbitrary query $\psi(x_1,\ldots,x_n)$, $n>2$, define $\psi_1(x_1,\ldots,x_{n-1}) = \psi'(x_1,\ldots,x_{n-1})$, $\psi_2(x_1,\ldots,x_{n-2}) = \psi_1'(x_1,\ldots,x_{n-2})$, etc., until we obtain $\phi(x,y) = \psi_{n-2}(x,y)$. It is easy to see that $\mathcal{A}\models\phi(a,b)$ iff (a,b) is in the Gaifman graph of $\Psi(\mathcal{A})$. From Lemma 7, we see that the locality rank of ϕ is $3^{n-2}r+(3^{n-2}-1)/2$. Now the theorem follows from the observation made above, Theorem 2, and the fact that $\mathcal{G}(\Psi(\mathcal{A}))$ is undirected. □

4 Bounded Degree Property

A very convenient form of the locality property is called the *bounded degree property*. It says that for structures from $\mathrm{STRUCT}_k[\tau]$ (that is, τ-structures in which no degree exceeds k), there is an upper bound on $deg(\Psi(\mathcal{A}))$ that depends only on ψ and k. A special case of this property is the graph bounded degree property mentioned in Section 2. It was established for first-order graph queries in [23] (see also Corollary 5).

Definition 8. A query $\psi(x_1,\ldots,x_m)$ is said to have the **bounded degree property**, or **BDP**, if there is a function $f_\psi:\mathbb{N}\to\mathbb{N}$ such that $deg(\Psi(\mathcal{A}))\le f_\psi(k)$ for every $\mathcal{A}\in\mathrm{STRUCT}_k[\tau]$. □

This property can be used as an easy-to-apply tool for establishing expressiveness bounds of query languages. Assume that it is known that every query in a language $\mathcal{L}$ has the BDP. To show that some query q is not definable in $\mathcal{L}$, one has to find a number k and a class $\mathcal{C}$ of input structures in $\mathrm{STRUCT}_k[\tau]$ such that $q(\mathcal{A})$ can realize arbitrarily large degrees on structures $\mathcal{A}$ from $\mathcal{C}$. This is exactly the idea of the proof of Corollary 4. The usefulness of BDP for proving expressiveness bounds on first-order graph queries was demonstrated in [23].

The main result of this section is the following.

Theorem 9. *Every local query has the bounded degree property.*

Proof sketch. Fix a query $\psi(x_1,\ldots,x_m)$ of locality rank r. Fix a structure $\mathcal{A}$ in $\mathrm{STRUCT}_k[\tau]$. Without loss of generality assume $m>1$, $r>0$ and $A\ne\emptyset$. Let $p=\sum_i p_i$. Let $s_{\mathcal{A}}(d)$ be the maximum size of $S_d(a)$ for $a\in A$. Under these assumptions, we claim

Lemma 10. *Let $d = (2m-2)(2r+1)$. Suppose $a \approx_d b$ and $S_d(a) \cap S_d(b) = \emptyset$. Then $|degree_i(a) - degree_i(b)| \leq (2s_{\mathcal{A}}(d))^{m-1}$ for any $i \leq m$.* □

From this lemma we derive that $deg(\Psi(\mathcal{A})) \leq m \cdot s^m \cdot 2^{1+m+ls^p}$, where $s = s_{\mathcal{A}}((4m-4)(2r+1))$. Finally, since $deg\_set(\mathcal{A}) \subseteq \{0, \ldots, k\}$, there is an upper bound on $s_{\mathcal{A}}(n)$ that depends on n, k, and p only, from which the bounded degree property follows. □

Let us discuss some implications of this result. As a start, we note that the graph bounded degree property result from [23] applies only to queries from graphs to graphs. One may ask what happens in the presence of auxiliary information, such as the successor relation. Since the successor relation only adds 0 and 1 to the degree set, we obtain immediately

Corollary 11. *The graph bounded degree property of first-order queries continues to hold in the presence of a successor relation.* □

But what happens if relations more complex than the successor are allowed? For instance, auxiliary relations whose degrees are not bounded by any constant, but are still not very large? We can answer this question by using the (slightly modified) notion of moderate degree from [15], and the estimate on the number of in- and out-degrees obtained in the proof of Theorem 9.

Consider a class of structures $\mathcal{C} \subseteq \mathrm{STRUCT}[\tau]$ for some relational vocabulary τ. Define a function $s_{\mathcal{C}} : \mathbb{N} \to \mathbb{N}$ by letting $s_{\mathcal{C}}(n)$ be the maximal possible in- or out-degree in some n-element structure $\mathcal{A} \in \mathcal{C}$. Given an increasing function $g(n)$ such that $g(n)$ is not bounded by any constant, we say that $\mathcal{C}$ is of $g(n)$-**moderate degree** if $s_{\mathcal{C}}(n) \leq \log^{o(1)} g(n)$. That is, we have a function $\delta : \mathbb{N} \to \mathbb{N}$ such that $\lim_{n\to\infty} \delta(n) = 0$ and $s_{\mathcal{C}}(n) \leq \log^{\delta(n)} g(n)$. When g is the identity, we have the definition of moderate degree of [15].

Proposition 12. *Let ψ be a local query. Let $\mathcal{C}$ be a class of structures of $g(n)$-moderate degree. Then there is $N \in \mathbb{N}$ such that for any $\mathcal{A} \in \mathcal{C}$ with $card(A) = n > N$, we have $deg(\Psi(\mathcal{A})) < g(n)$.* □

The transitive closure of a chain has as many distinct degrees as there are links in the chain. It is thus not definable by a local query even when auxiliary data of moderate degree are available. Now, using the fact that the transitive closure of a chain is FO-complete for DLOGSPACE [14], we obtain

Corollary 13. *Let P be a problem complete for DLOGSPACE under FO reductions. Then P is not definable by a local query even in the presence of relations of moderate degree.* □

The converse to Theorem 9 is not true. That is, there is a non-local query that has the bounded degree property. Indeed, let $\psi(x, y)$ be a graph query defined as follows. If G is the union of disjoint chains having a unique longest chain, then $G \models \psi(x, y)$ iff (x, y) is an edge in the unique longest chain in G; otherwise, $G \not\models \psi(x, y)$ for all x, y. It is clear that ψ has the bounded degree property but violates locality. Nevertheless, it should be pointed out that the relational algebra augmented with this query ψ does not have the bounded degree property.

5 Stronger Bounded Degree Properties

The reader may have noticed a certain asymmetry in the statement of the bounded degree property: We make an assumption about the degree *set* $deg\_set(\mathcal{A})$, and give a conclusion that there is an upper bound on the degree *count* $deg(\Psi(\mathcal{A}))$. So, the question arises: Can the bounded degree property be strengthened? In what follows, we present two most obvious attempts to strengthen it. It was conjectured that both of them hold for first-order logic, but we show that this is not the case. Consequently, not all local queries possess these stronger properties.

Definition 14. A query ψ has the **strong bounded degree property**, or SBDP, if there exists a function $f_\psi : \mathbb{N} \to \mathbb{N}$ such that $deg(\Psi(\mathcal{A})) \leq f_\psi(deg(\mathcal{A}))$ for any structure $\mathcal{A}$. □

Definition 15. A query ψ has the **interval bounded degree property**, or IBDP, if there exists a function $f_\psi : \mathbb{N} \to \mathbb{N}$ such that $deg(\Psi(\mathcal{A})) \leq f_\psi(k)$ for any structure $\mathcal{A}$ with $\max deg\_set(\mathcal{A}) - \min deg\_set(\mathcal{A}) \leq k$. □

It is easy to see that the SBDP implies the IBDP and the IBDP implies the BDP. It turns out, somewhat unexpectedly, that there are first-order graph queries that do not have them.

Theorem 16. *There are first-order graph queries that do not have the interval bounded degree property. Consequently, they do not have the strong bounded degree property either.*

Thus, in contrast to Theorem 9, we conclude that

Corollary 17. *There are local queries that do not possess the interval or the strong bounded degree properties.* □

In the remainder we sketch the main construction of Theorem 16. We need to construct a first-order graph query that does not have the IBDP. First fix $n > 3$, four disjoint sets $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_n\}$, $C = \{e_1, \ldots, e_n\}$, $D = \{d_1, \ldots, d_n\}$, and a permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$. Define the graph G_π as follows. Its set of nodes N is $X \cup Y \cup C \cup D \cup \{a, b, c\}$. Its edges are given as follows:

- There are loops (a, a), (b, b), (c, c) and also edges (b, c) and (c, b).
- For each $i < n$, there are edges (x_i, x_{i+1}) and (y_i, y_{i+1}).
- For each $i \leq n$, there is an edge $(x_i, y_{\pi(i)})$.
- For each $i \leq n$, there are edges (a, x_i), (x_i, a), (b, y_i), (y_i, b), (c, y_i), (y_i, c).
- For each $i \leq n$ and $j \leq n$, there are edges (x_i, e_j), (e_j, y_i), (y_i, d_j), (d_j, x_i).

Define the graph G_n as the disjoint union of G_π for all permutations π. That is, G_n has $n!$ connected components and $(4n + 3) \cdot n!$ nodes. It follows straightforwardly from the construction that $deg\_set(G_\pi) = \{n, n+1, n+2, n+3, n+4\}$.

Next, we define a query Ψ as follows: In some component G_π, in the output we get an edge from a to y_i iff we have $\pi(x_{l+1}) = y_{i+1}$ where $x_l = \pi^{-1}(y_i)$. One can now show that Ψ is first-order definable, but $deg(\Psi(G_n))$ depends on n. □

6 Aggregation, SQL, and the Bounded Degree Property

In this section, we investigate locality and the bounded degree property in the context of SQL-like languages. We start by briefly describing the syntax and semantics of the theoretical SQL-like language to be analyzed. Two main features that distinguish (plain) SQL from the relational calculus are grouping (the SQL `GROUPBY` operator) and aggregate functions (such as `COUNT` and `AVG`). Our languages incorporate these features in a clean analyzable way. We then show how the notions of locality and bounded degree extend to queries in our language. The main result is that queries naturally representing those on $\mathrm{STRUCT}_k[\tau]$ are local for every fixed k. Consequently, such queries have the BDP, and thus many inexpressibility proofs carry over from the first-order case to SQL.

Let us start with the syntax and semantics of our SQL-like language. The data types that can be manipulated in the language are given by the grammar:

$$s ::= b \mid \mathbb{B} \mid \mathbb{Q} \mid s_1 \times \cdots \times s_n \mid \{s\}$$

Elements of the base type b are drawn from an unspecified infinite domain. The type $\mathbb{B}$ contains the two Boolean objects *true* and *false*. The type Q contains the rational numbers. Elements of the product type $s_1 \times \cdots \times s_n$ are n-tuples whose ith component is of type s_i. Finally, elements of the set type $\{s\}$ are finite sets whose elements are of type s.

We present the language incrementally. We start from $\mathcal{NRC}(=)$, which is equivalent to the usual nested relational algebra [2, 6]. To obtain our SQL-like language we add arithmetic and a summation operation to model aggregation. The syntax of $\mathcal{NRC}(=)$ is given below.

$$\frac{}{x^s : s} \qquad \frac{}{c : \mathbb{Q}}$$

$$\frac{}{\mathit{true} : \mathbb{B}} \qquad \frac{}{\mathit{false} : \mathbb{B}} \qquad \frac{e_1 : \mathbb{B} \quad e_2 : s \quad e_3 : s}{\mathit{if}\ e_1\ \mathit{then}\ e_2\ \mathit{else}\ e_3 : s} \qquad \frac{e_1 : s \quad e_2 : s}{e_1 = e_2 : \mathbb{B}}$$

$$\frac{e : s_1 \times \cdots \times s_n}{\pi_i\ e : s_i} \qquad \frac{e_1 : s_1 \quad \cdots \quad e_n : s_n}{(e_1, \ldots, e_n) : s_1 \times \cdots \times s_n}$$

$$\frac{}{\{\}^s : \{s\}} \qquad \frac{e : s}{\{e\} : \{s\}} \qquad \frac{e_1 : \{s\} \quad e_2 : \{s\}}{e_1 \cup e_2 : \{s\}} \qquad \frac{e_1 : \{t\} \quad e_2 : \{s\}}{\bigcup\{e_1 \mid x^s \in e_2\} : \{t\}}$$

We often omit the type superscripts as they can be inferred. Let us briefly recall the semantics, cf. [6]. Variables x^s are available for each type s. Every rational constant is available. The operations for Booleans, tupling and projections are standard. $\{\}$ forms the empty set. $\{e\}$ forms the singleton set containing e. $e_1 \cup e_2$ unions the two sets e_1 and e_2. Finally, $\bigcup\{e_1 \mid x \in e_2\}$ maps the function $f = \lambda x.e_1$ over all elements in e_2 and then returns their union; thus if e_2 is the set $\{o_1, \ldots, o_n\}$, the result of this operation would be $f(o_1) \cup \cdots \cup f(o_n)$. For example, $\bigcup\{\{(x,x)\} \mid x \in \{1,2\}\}$ evaluates to $\{(1,1),(2,2)\}$.

Given a type s, the **height** of s is defined as the nesting depth of set brackets in s. For example, the usual flat relations (sets of tuples of base types) have height 1. Given an expression e, the **height** of e is defined as the maximal height of all types that appear in the typing derivation of e. For example, $\bigcup\{\bigcup\{\{(x,y)\} \mid x \in R\} \mid y \in S\}$ is an expression of height 1 if both R and S are flat relations. It is known [26, 28] that when restricted to expressions of height 1, $\mathcal{NRC}(=)$ is equivalent to the usual relational algebra. We also write $\mathcal{NRC}(=_b)$ when the equality test is restricted to base types b, $\mathbb{B}$, and $\mathbb{Q}$. We sometimes list the free variables in an expression in brackets like: $e(R, x)$.

As was mentioned, the practical database language SQL extends the relational calculus by having arithmetic operations, a group-by operation, and various aggregate functions such as `AVG`, `COUNT`, `SUM`, `MIN`, and `MAX`. It is known [6] that the group-by operator can already be simulated in $\mathcal{NRC}(=)$. The others need to be added. The arithmetic operators are the standard ones: $+$, $-$, $\cdot$, and $\div$ of type $\mathbb{Q} \times \mathbb{Q} \to \mathbb{Q}$. We also add the order on the rationals: $\leq_{\mathbb{Q}}$: $\mathbb{Q} \times \mathbb{Q} \to \mathbb{B}$. As to aggregate functions, we add just the following construct

$$\frac{e_1 : \mathbb{Q} \quad e_2 : \{s\}}{\sum\{\!|e_1 \mid x^s \in e_2|\!\} : \mathbb{Q}}$$

The semantics is this: map the function $f = \lambda x.e_1$ over all elements of e_2 and then add up the results. Thus, if e_2 is the set $\{o_1, \ldots, o_n\}$, it returns $f(o_1)+\cdots+f(o_n)$. For example, $\sum\{\!|1 \mid x \in X|\!\}$ returns the cardinality of X. Note that this is different from adding up the values in $\{f(o_1), \ldots, f(o_n)\}$; in the example above, doing so yields 1 as no duplicates are kept. To emphasize that duplicate values of f are being added up, we use bag (multiset) brackets $\{\!|\ |\!\}$ in this construct.

We denote this theoretical reconstruction of SQL by $\mathcal{NRC}^{\text{aggr}}$. That is, $\mathcal{NRC}^{\text{aggr}}$ has all the constructs of $\mathcal{NRC}(=)$, the arithmetic operations $+, -, \cdot$ and $\div$, the summation construct $\sum$ and the linear order on the rationals.

Let us provide two examples to demonstrate how typical SQL queries involving aggregate functions can be implemented in $\mathcal{NRC}^{\text{aggr}}$. For the first example, consider the query that computes the total expenditure on male employees in various departments in a company. Let $EMP : \{name \times salary \times sex \times dept\}$ be a relation that tabulates the name, salary, sex, and department of employees. The query in SQL is `SELECT dept, SUM(salary) FROM EMP WHERE sex = 'male' GROUPBY dept`. It can be expressed in $\mathcal{NRC}^{\text{aggr}}$ as $\bigcup\{\{(\pi_{dept}\ x,\ \sum\{\!|\mathit{if}\ \pi_{dept}\ x = \pi_{dept}\ y\ \mathit{then}\ \mathit{if}\ \pi_{sex}\ y = \mathit{'male'}\ \mathit{then}\ \pi_{salary}\ y\ \mathit{else}\ 0\ \mathit{else}\ 0 \mid y\ \in EMP|\!\})\} \mid x \in EMP\}$. For the second example, consider the query that computes the number of distinct salaries of male employees in various departments in the same company. The query in SQL is `SELECT dept, COUNT(distinct salary) FROM EMP WHERE sex = 'male' GROUPBY dept`. Note that in this query, duplicate salary figures in a department are eliminated before counting. It can be expressed in $\mathcal{NRC}^{\text{aggr}}$ as $\bigcup\{\{(\pi_{dept}\ x,\ \sum\{\!|1 \mid y \in \bigcup\{\mathit{if}\ \pi_{dept}\ z\ = \pi_{dept}\ x\ \mathit{then}\ \mathit{if}\ \pi_{sex}\ z = \mathit{'male'}\ \mathit{then}\ \{\pi_{salary}\ z\}\ \mathit{else}\ \{\}\ \mathit{else}\ \{\} \mid z \in EMP\}|\!\})\} \mid x \in EMP\}$.

In fact, it is known [23] that all possible nested applications of all SQL aggregate functions mentioned above can be implemented in $\mathcal{NRC}^{\text{aggr}}$. It is also known [23] that $\mathcal{NRC}^{\text{aggr}}$ has the conservative extension property and thus its expressive power depends only on the height of input and output and is independent of the height of intermediate data. So to conform to SQL, it suffices to restrict our input and output to height at most one.

Before, we assumed queries to be formulae $\psi(x_1, \ldots, x_m)$, mapping structures of some relational vocabulary τ into m-ary relations, defined by $\Psi(\mathcal{A}) = \langle A, \{(a_1, \ldots, a_m) \mid a_1, \ldots, a_m \in A, \mathcal{A} \models \psi(a_1, \ldots, a_m)\}\rangle$. Now we have to show how $\mathcal{NRC}^{\text{aggr}}$-expressions correspond to queries. After this, we shall be able to transfer the notions of locality and bounded degree to $\mathcal{NRC}^{\text{aggr}}$.

First, we model τ-structures as tuples of objects of types of the form $\{b \times \ldots \times b\}$, with the arities corresponding to those of the symbols in τ. We shall abbreviate $b \times \ldots \times b$, m times, as b^m. A **relational query** over STRUCT$[\tau]$ in $\mathcal{NRC}^{\text{aggr}}$ is an $\mathcal{NRC}^{\text{aggr}}$ expression e of type $\{b^m\}$, whose free variables have types $\{b^{p_1}\}, \ldots, \{b^{p_l}\}$, where p_i is the arity of the ith symbol in τ. Given such an expression, which we write as $e(R_1, \ldots, R_l)$ or $e(\vec{R})$, it can be considered as a query ψ_e as follows. We let, for a τ-structure $\mathcal{A}$ over the domain of type b,

$$\mathcal{A} \models \psi_e(a_1, \ldots, a_m) \text{ iff } (a_1, \ldots, a_m) \in e(\mathcal{A})$$

In other words, the Ψ_e corresponding to the query ψ_e is precisely e. (This is true because $(a_1, \ldots, a_m) \in e(\mathcal{A})$ implies that all a_is are in the carrier of $\mathcal{A}$.)

Now, for each relational query e, we say that it is local if ψ_e is, and e's locality rank is that of ψ_e. Similarly, we define the bounded degree property of relational queries in $\mathcal{NRC}^{\text{aggr}}$. Finally, we say that a query is local on a class of structures $\mathcal{C} \subset$ STRUCT$[\tau]$ if the condition in the definition of locality is satisfied on every structure from $\mathcal{C}$ (but not necessarily on every structure in STRUCT$[\tau]$).

Our main result is:

Theorem 18. *For any fixed k, every relational query in $\mathcal{NRC}^{\text{aggr}}$ is local on* STRUCT$_k[\tau]$.

Proof sketch. The proof relies on the following key lemma which gives us a very convenient 'normal form' of $\mathcal{NRC}^{\text{aggr}}$ queries when restricted to structures of degrees at most k. The normal form is a chain of *if-then-else* statements where each branch is a relational calculus expression, and all uses of aggregate functions can only appear in the conditions of these *if-then-else* statements.

Lemma 19. *Let $\vec{R}$ denote a vector of relations of degree at most k, $e(\vec{R}) : s$ be an $\mathcal{NRC}^{\text{aggr}}$-expression, with s of height at most 1. Then $e(\vec{R})$ is equivalent to an expression of the form if $\mathcal{P}_1(\vec{R})$ then $e_1(\vec{R})$ ... else if $\mathcal{P}_d(\vec{R})$ then $e_d(\vec{R})$ else $e_{d+1}(\vec{R})$, where each $e_j(\vec{R})$ is in $\mathcal{NRC}(=_b)$ and d depends only on k and e.* □

This normal form result gets complicated aggregate functions out of the way. We can now prove our theorem. Let $\vec{R}$ denote a structure in STRUCT$_k[\tau]$ whose elements are of base type b. Let $e(\vec{R})$ be a relational query in $\mathcal{NRC}^{\text{aggr}}$. By Lemma 19, we can assume that $e(\vec{R})$ has the form *if* $\mathcal{P}_1(\vec{R})$ *then* $e_1(\vec{R})$ ... *else if* $\mathcal{P}_d(\vec{R})$ *then* $e_d(\vec{R})$ *else* $e_{d+1}(\vec{R})$, where each $e_i(\vec{R})$ is in $\mathcal{NRC}(=_b)$. Since $\mathcal{NRC}(=)$ enjoys the conservative extension property [28], each e_i can be defined in relational calculus. By Fact 1, every ψ_{e_i} has some finite locality index r_i. From this we immediately conclude that ψ_e has locality index $\max_i r_i$. □

From here, applying verbatim the proof of Theorem 9, we conclude

Corollary 20. *Relational queries in $\mathcal{NRC}^{\mathrm{aggr}}$ have the bounded degree property.*

We immediately conclude from Corollary 20 that

Corollary 21. (cf. [23]) *$\mathcal{NRC}^{\mathrm{aggr}}$ cannot express the following queries: (deterministic) transitive closure of a graph, connectivity test, testing for a (binary, ternary, etc.) tree. This continues to hold when a built-in successor relation or any other built-in relations whose degrees do not exceed a fixed number k are available on the nodes.* □

Recall that Härtig and Rescher quantifiers are two generalized quantifiers for equal cardinality and bigger cardinality respectively. Since these tests can be done in $\mathcal{NRC}^{\mathrm{aggr}}$, we obtain:

Corollary 22. *Every first-order query with Härtig and Rescher quantifiers has the bounded degree property.* □

7 Applications to Incremental Recomputation

Since relational calculus has a limited expressive power and cannot compute queries such as transitive closure, one often stores the results of these queries as materialized database views. Once the underlying database changes, the changes must be propagated to the views as well. In the case when a view is defined in relational calculus, or at least in the same language in which update propagations are specified, the problem of incremental maintenance has been studied thoroughly. However, few papers [11, 9, 12, 27] addressed the issue of maintaining queries such as the transitive closure in first-order or $\mathcal{NRC}^{\mathrm{aggr}}$.

It was shown [9] that, in the absence of auxiliary data, recursive queries such as transitive closure and same generation cannot be maintained in relational calculus or even in SQL. It was conjectured in [9, 12] that this continues to be true in the presence of auxiliary data. Using the results developed in previous sections, we can address this question partially. In particular, we now show that maintenance of some recursive queries remains impossible even if auxiliary data of moderate or low degree are available.

We also consider the same-generation query over a graph having two label symbols A and B. Such a graph can be conveniently represented by two relations, one for edges labeled A and the other for B, which need not be disjoint. We use A and B to name these two relations. Then x and y are in the same generation with respect to A and B iff there is a z such that there is a walk from x to z in A and a walk from z to y in B that are equal in length.

Theorem 23. *Neither transitive closure nor same-generation can be maintained in the relational calculus when auxiliary data of moderate degree are available.*

Proof sketch. The main idea of the proof of non-maintainability of both transitive closure and same-generation [9] is essentially this: Suppose there is an expression $g(I, I^+, t)$ that, given an input I, the result of a query I^+ on I, and a tuple t in I, produces the output of the query on $I - \{t\}$. Then both proofs in [9] show how to use this assumption to produce an expression in first-order logic plus g that computes the transitive closure of a chain. Since the construction of [9] does not assume any auxiliary data, we can apply it here to obtain that, if either query is maintainable in first-order in the presence of auxiliary data of moderate degree, then with such auxiliary data the transitive closure of a chain is computable, which contradicts Corollary 13. $\square$

Using essentially the same argument, but employing Corollary 21 in place of Corollary 13, we can also prove that

Corollary 24. *Neither transitive closure nor same-generation can be maintained in $\mathcal{NRC}^{\text{aggr}}$ in the presence of auxiliary data whose degrees are bounded by a constant.* $\square$

8 Future Work

There are many open questions we would like to address in the future. We are interested in developing techniques for proving languages local. So far, there appears to be no commonality between Gaifman's proof of locality for first-order [16] and our proof of (restricted) locality of $\mathcal{NRC}^{\text{aggr}}$. We also believe that this restriction can be eliminated, but we have not been able to prove it.

Conjecture 1 *Every relational query in $\mathcal{NRC}^{\text{aggr}}$ is local.*

The previous results do not seem to apply to ordered structures: indeed, by taking any input and returning the graph of the underlying linear order, we violate the bounded degree property. Thus, it does not hold in $\mathcal{NRC}^{\text{aggr}}(\leq_b)$, which is $\mathcal{NRC}^{\text{aggr}}$ augmented with a linear order on type b. However, we still believe that the bounded degree property can be partially recovered:

Conjecture 2 *Every relational query in $\mathcal{NRC}^{\text{aggr}}(\leq_b)$ that is order-independent has the bounded degree property.*

Acknowledgements. We thank Moshe Vardi for suggesting the extension from Theorem 2 to Theorem 6, and Tim Griffin for a careful reading of the manuscript. Part of this work was done while Wong was visiting the University of Melbourne and Bell Laboratories. Wong would like to thank these organizations and fellow coauthors for their hospitality during this work.

References

1. S. Abiteboul, R. Hull, V. Vianu, *Foundations of Databases*, Addison Wesley, 1995.
2. S. Abiteboul, P. Kanellakis. Query languages for complex object databases. *SIGACT News*, 21(3):9–18, 1990.
3. M. Ajtai and R. Fagin. Reachability is harder for directed than for undirected graphs. *Journal of Symbolic Logic*, 55(1):113–150, March 1990.

4. J. Albert. Algebraic properties of bag data types. In *VLDB'91*, pages 211–219.
5. J. Barwise et al eds., *Model-Theoretic Logics*. Springer-Verlag, 1985.
6. P. Buneman, S. Naqvi, V. Tannen, L. Wong. Principles of programming with complex objects and collection types. *Theoretical Computer Science*, 149 (1995), 3–48.
7. S. Chaudhuri, M. Y. Vardi, Optimization of *real* conjunctive queries, In *PODS'93*.
8. M.P. Consens, A.O. Mendelzon, Low complexity aggregation in GraphLog and Datalog, *Theoretical Computer Science* 116 (1993), 95–116.
9. G. Dong, L. Libkin, L. Wong. On impossibility of decremental recomputation of recursive queries in relational calculus and SQL. In *Database Progr. Lang.'95*, Springer Electronic Workshops in Computing, 1996.
10. G. Dong, L. Libkin, L. Wong. Local properties of query languages, Tech. Memo, Bell Labs, 1995.
11. G. Dong and J. Su. Incremental and Decremental Evaluation of Transitive Closure by First-Order Queries. *Information and Computation*, 120(1):101–106, 1995.
12. G. Dong and J. Su. Space-bounded FOIES. In *PODS'95*, pages 139–150.
13. H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer Verlag, 1995.
14. K. Etessami, Counting quantifiers, successor relations, and logarithmic space, *in* Conf. on Structure in Complexity Theory, 1995.
15. R. Fagin, L. Stockmeyer, M. Vardi, On monadic NP vs monadic co-NP, *Information and Computation*, 120 (1994), 78–92.
16. H. Gaifman, On local and non-local properties, *in* Logic Colloquium '81, North Holland, 1982.
17. T. Griffin, L. Libkin, Incremental maintenance of views with duplicates, In *SIGMOD'95*, pages 319–330.
18. S. Grumbach, T. Milo, Towards tractable algebras for bags, *Journal of Computer and System Sciences*, 52 (1996), ??–??.
19. S. Grumbach, L. Libkin, T. Milo and L. Wong. Query languages for bags: expressive power and complexity. *SIGACT News*, 27 (1996), 30–37.
20. S. Grumbach and C. Tollu. On the expressive power of counting. *Theoretical Computer Science* 149(1): 67–99, 1995.
21. A. Klug, Equivalence of relational algebra and relational calculus query languages having aggregate functions, *Journal of the ACM* **29**, No. 3 (1982), 699–717.
22. L. Libkin, L. Wong, Some properties of query languages for bags, In *DBPL'93*, Springer, 1994.
23. L. Libkin, L. Wong, Query languages for bags and aggregate functions. *JCSS*, to appear. Extended abstract in *PODS'94*, pages 155–166.
24. L. Libkin, L. Wong, On representation and querying incomplete information in databases with bags, *Information Processing Letters* **56** (1995), 209–214.
25. G. Ozsoyoglu, Z. M. Ozsoyoglu, V. Matos, Extending relational algebra and relational calculus with set-valued attributes and aggregate functions, *ACM Transactions on Database Systems* **12**, No. 4 (1987), 566–592.
26. J. Paredaens and D. Van Gucht. Converting nested relational algebra expressions into flat algebra expressions. *ACM TODS*, 17(1):65–93, March 1992.
27. S. Patnaik and N. Immerman. Dyn-FO: A parallel dynamic complexity class. In *PODS'94*, pages 210–221.
28. L. Wong, Normal forms and conservative properties for query languages over collection types, *JCSS* 52 (1996), 495–505.

This article was processed using the LaTeX macro package with LLNCS style