Local Properties of Query Languages

Guozhu Dong
Dept of Computer Science
University of Melbourne
Parkville, Vic. 3052, Australia
Email: dong@cs.mu.oz.au

Leonid Libkin
Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974, USA
Email: libkin@bell-labs.com

Limsoon Wong
Kent Ridge Digital Labs
21 Heng Mui Keng Terrace
Singapore 119613
Email: limsoon@krdl.org.sg

Abstract

In this paper we study the expressiveness of *local* queries. By locality we mean — informally — that in order to check if a tuple belongs to the result of a query, one only has to look at a certain predetermined portion of the input. Examples include all relational calculus queries.

We start by proving a general result describing outputs of local queries. This result leads to many easy inexpressibility proofs for local queries. We then consider a closely related property, namely, the bounded degree property. It describes the outputs of local queries on structures that locally look "simple." Every query that is local is shown to have the bounded degree property. Since every relational calculus (first-order) query is local, the general results proved for local queries can be viewed as "off-the-shelf" strategies for proving inexpressibility results, which are often easier to apply than Ehrenfeucht-Fraïssé 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, which is essentially plain SQL, has the bounded degree property, thus answering a question that has been open for several years. Consequently, first-order queries with Härtig or Rescher quantifiers also have the bounded degree property. Finally, 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 auxiliary relations of moderate degree.

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-Fraïssé games [13, 17], locality [18], 0-1 laws [15], Hanf's technique [16, 24], the bounded degree property [36], etc. We are especially interested in local properties of queries, first introduced by Gaifman [18]. 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 one of the major motivations for research in finite model theory. However, most of those tools developed are modified Ehrenfeucht-Fraïssé 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 [16], infinitary logics [4], logics with counting [23], 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 [29, 40], first results on expressive power appeared in [7]. However, the results of [7] were based on the assumption that the deterministic and nondeterministic logspace are different, and thus questions on the expressive power of SQL-like languages remained open.

In the past few years, an intimate connection was discovered between relational languages with aggregate functions and languages whose main data structures are bags rather than sets. There was a flurry of activity in studying such languages, resulting in the thorough study of interdefinability of their primitives [3, 32, 21], complexity [21], optimization [6], equational theories [20] and, finally, the limitations of their expressive power [36, 35]. In particular, it was shown in [36] that the transitive closure of a graph remains inexpressible even when grouping and aggregation are added to the relational calculus. For a survey of the results in this area, see [22].

Since there was no tool available for studying languages with aggregate functions, the technique we tried to use in [36] was the following. We tried to find a property possessed by the queries in our language, which is not possessed by the transitive closure of a graph. The property we have in mind is this: Think of a query q that takes a graph as an input and returns a graph. 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 [36]. We have also demonstrated that it is a very convenient tool for establishing bounds on expressive power, often much easier to apply than the games or other tools. However, we were not able to prove in [36] that it extends to languages with aggregate functions. 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 very 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 [36]. 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 [18]. The locality result in [18] has two parts, and only one was used in our proof in [36]. 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 and, in particular, the expressiveness issues for such queries.

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. We do this in such a way that the presentation of the results about locality and bounded degree properties is language-independent, and can thus be applied to a number of languages, including first-order logic and some of its extensions. We give formal definitions of local queries, and generalize the definition of the bounded degree property to arbitrary queries. We also 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 expected generalizations of the bounded degree property. One of them, saying that the output of a query q cannot have more than c different in- and out-degrees, provided the input has at most k different degrees, and c depends only on q and k, was conjectured to be true for first-order queries. We show that, somewhat unexpectedly, there are first-order queries that violate this and even a slightly weaker property.

In Section 6 we introduce our 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, if there is no ordering on the domain elements. It also follows that first-order queries with Härtig and Rescher (equicardinality and majority) quantifiers [43] have the bounded degree property.

Finally, 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.

An extended abstract of this paper appeared in Proceedings of the 6th International Conference on

Database Theory [9].

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. So we denote the signature of graphs by τ_{gr}; this signature has one binary predicate, representing edges of the graph.

A structure will be written as $\mathcal{A} = \langle A, \overline{R}_1, \ldots, \overline{R}_l \rangle$, where A is a finite nonempty set called the universe of $\mathcal{A}$, and $\overline{R}_i$ is the interpretation of R_i, which is a subset of A^{p_i}. When it does not lead to confusion, we will write R_i in place of $\overline{R}_i$. We use the symbol $\cong$ to denote isomorphism of structures. The class of finite τ-structures is denoted by STRUCT$[\tau]$.

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 [12, 18]. 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. [16]. 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 $a \in A$, and $r \geq 0$, the r-**sphere** of a, denoted by $S_r(a)$, 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 universe of $N_r(\vec{t})$ is $S_r(\vec{t})$, the interpretation of the relations in τ is obtained by restricting them to the universe, 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 $N_d(a) \cong N_d(b)$. Note that since $N_d(a)$ and $N_d(b)$ are structures of the signature τ_1, any isomorphism $h : N_d(a) \rightarrow N_d(b)$ is required to satisfy $h(a) = b$.

We 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 2.1 Given a query $\psi(x_1, \ldots, x_m)$, its **locality rank** is the minimal number $r \in \mathbb{N}$ such that, for every $\mathcal{A} \in \mathrm{STRUCT}[\tau]$ 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 rank is ∞. A query is **local** if it has a finite locality rank. A language is **local** if every query in it is. $\square$

Are there any interesting examples of local queries? An answer to this is provided by Gaifman's locality theorem [18] which implies, in our terminology, the following fact.

Fact 2.2 *Every first-order (relational calculus) query is local.* $\square$

However, even the simplest fragment of second-order logic, monadic Σ^1_1, is not local. Consider a first-order formula $\psi_0(x)$ in the language of one binary relational symbol E and two unary symbols U and X that says the following: the interpretation of U is contained in the interpretation of X, no predecessor of an element of U (in terms of the E relation) is in X, X is closed under E-successors, and x is in X. Let $\psi(x)$ be $\exists X \psi_0(x)$; it defines a query on graphs with a distinguished set of nodes U. Assume that ψ is local, and its locality rank is r. Let G be the graph of a successor relation on d elements, where $d > 4r+5$ is odd. Let a be its middle element, and let b be an element that precedes a and is at the distance at least $r+1$ from the start node and a, and let c be an element that is preceded by a and is at the distance at least $r+1$ from the end node and a. If we interpret U as $\{a\}$, then $N_r(b) \cong N_r(c)$. At the same time, $\psi(c)$ holds, but $\psi(b)$ does not hold, proving that ψ is not local.

We shall see later that there are 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 $\mathrm{STRUCT}_k[\tau]$. 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 $\mathrm{STRUCT}_k[\tau]$, for any fixed k. We also see from this that first-order queries with Härtig and Rescher quantifiers [43] are local when restricted to the same structures.

As was mentioned before, a certain notion of uniform behavior of queries on $\mathrm{STRUCT}_k[\tau_{\mathrm{gr}}]$ was introduced earlier in [36]. 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 $deg(\Psi(G)) \leq f(k)$ for any $G \in \mathrm{STRUCT}_k[\tau_{\mathrm{gr}}]$. It was shown in [36] 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 that 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 $d \geq 0$, shows how many distinct small neighborhoods are realized in $\mathcal{A}$. The first result of this section shows the intimate 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 also be interpreted as saying that output of a local graph query cannot be much more complex than its input.

Theorem 3.1 *Let $\psi(x, y)$ be a graph query on τ-structures of finite locality rank 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. The key to our theorem is the observation that for any $m > 0$, when a large neighborhood of a fixed point a and a large neighborhood of another fixed point b are isomorphic, it is possible to find a permutation π on a smaller sphere S around a and b such that the m-neighborhoods of a and x and of b and $\pi(x)$ are isomorphic for all $x \in S$. This observation is formalized in the lemma below, whose proof is delayed until the end of the section.

Lemma 3.2 *Let r be an arbitrary positive integer, and let $d \geq 3r + 1$. Assume that $a \approx_d b$ in a τ-structure $\mathcal{A}$. 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 3.2 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 $d(a, x), d(b, x) > 2r + 1$. (This follows immediately from Claims 3.7 and 3.9 in the proof of Lemma 3.2.) Thus, $(a, x) \in E'$ iff $(b, x) \in E'$ by locality. Furthermore, by Lemma 3.2, 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. Thus, the number of possible indegrees of G' is at most $\mathsf{ntp}(d, G)$ and the number of possible outdegrees of G' is at most $\mathsf{ntp}(d, G)$. Hence $degset(G')$ has at most $2 \cdot \mathsf{ntp}(d, G)$ elements. $\square$

Before we give the proof of Lemma 3.2, let us give two simple applications to demonstrate Theorem 3.1's usefulness in establishing expressiveness bounds. The second of these will be generalized in the next section into a powerful result that lets us "compile away" Ehrenfeucht-Fraïssé games from many inexpressibility proofs.

Corollary 3.3 *No local query can define the transitive closure of a graph.*

Proof. Suppose $\psi(x, y)$ does define the transitive closure. Consider chains, which are graphs with the edge-set of the form $C_n = \{(a_0, a_1), \ldots, (a_{n-1}, a_n)\}$ where all a_is are distinct. Since ψ defines the transitive closure, $deg(\Psi(C_n)) = n + 1$. For every $d \geq 0$, there are at most $2d + 1$ non-isomorphic d-neighborhoods in a chain. Thus, if the locality rank of ψ is r, we obtain from Theorem 3.1 that $deg(\Psi(G))$ is at most $4(3r+1)+2$ for any chain graph G. Thus, ψ cannot define the transitive closure. □

Corollary 3.4 *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 3.1, we see that there is a bound on $deg(\Psi(G))$ that depends only on k and r, the locality rank of ψ, which implies the graph bounded degree property. □

The statement of Theorem 3.1 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 $\mathcal{A}$. Now we can prove the following.

Theorem 3.5 *Let $\psi(x_1, \ldots, x_n)$, $n \geq 2$, be a query on τ-structures of finite locality rank $r > 0$. 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,*

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

Proof. We prove this theorem by reduction to graph queries. Given a query $\psi(x_1, \ldots, x_n)$, $n > 2$, let $\psi'(x_1, \ldots, x_{n-1})$ be defined as follows. For a structure $\mathcal{A}$ with universe A, we let $\mathcal{A} \models \psi'(a_1, \ldots, a_{n-1})$ iff for some $a \in A$, and for some index $0 \leq i \leq n-1$, it is the case that $\mathcal{A} \models \psi(a_1, \ldots, a_i, a, a_{i+1}, \ldots, a_{n-1})$. Note that $i = 0$ means $\mathcal{A} \models \psi(a, a_1, \ldots, a_{n-1})$ and $i = n-1$ means $\mathcal{A} \models \psi(a_1, \ldots, a_{n-1}, a)$.

Our key lemma is:

Lemma 3.6 *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 $3r + 1$.*

We postpone the proof of this lemma until the end of the section, and now show how it implies 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 $\phi_0(x,y) = \psi_{n-2}(x,y)$. Let $\mathcal{A} \models \phi(x,y)$ iff $\mathcal{A} \models \phi_0(x,y)$ or $\mathcal{A} \models \phi_0(y,x)$. 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 3.6, we see that the locality rank of ϕ is $3^{n-2}r + \frac{1}{2}(3^{n-2}-1)$. The observation we made above about ψ^* shows that the query returning the Gaifman graph of the result of an n-ary query of locality rank r has locality rank $r_0 = 3^{n-2}r + \frac{1}{2}(3^{n-2}-1)$ for any $n \geq 2$.

Now applying Theorem 3.1, we obtain that the number of different indegrees in $\mathcal{G}(\Psi(\mathcal{A}))$ is at most $\mathsf{ntp}(3r_0+1,\mathcal{A})$. Since $\mathcal{G}(\Psi(\mathcal{A}))$ is undirected, we obtain from this that $deg(\mathcal{G}(\Psi(\mathcal{A})))$ is at most $\mathsf{ntp}(3^{n-1}r + \frac{1}{2}(3^{n-1}-1),\mathcal{A})$, thus proving the theorem. $\square$

As a side remark, note that for the case $n = 2$, Theorem 3.5 yields $deg(\mathcal{G}(\Psi(\mathcal{A}))) \leq \mathsf{ntp}(3r+1,\mathcal{A})$, while Theorem 3.1 gives $deg(\Psi(\mathcal{A})) \leq 2 \cdot \mathsf{ntp}(3r+1,\mathcal{A})$. The reason for losing the factor of 2 is that in the former case we deal with undirected graphs, for which in-degree of each node equals its out-degree.

The remainder of this section is devoted to proving Lemmas 3.2 and 3.6.

Proof of Lemma 3.2. The proof requires several steps. Let us begin with a few general observations about neighborhoods.

Claim 3.7 *Let $N_m(a)$ and $N_m(b)$ be isomorphic and let h be an isomorphism between them. Then, for $l \leq m$, h restricted to $S_l(a)$ is an isomorphism between $N_l(a)$ and $N_l(b)$.*

Proof. It is enough to show that this restriction of h maps $S_l(a)$ onto $S_l(b)$; the rest will follow from the fact that h is an isomorphism. Let $x \in S_l(a)$; then we can find some elements $x_1,\ldots,x_i$ and tuples $\vec{t}_1,\ldots,\vec{t}_{i+1}$ such that $i < l$; $a, x_1 \in \vec{t}_1$; $x_1, x_2 \in \vec{t}_2; \ldots; x_i, x \in \vec{t}_{i+1}$ and each $\vec{t}_j \in \overline{R}_s$ for some s. Applying h, we get $b, h(x_1) \in h(\vec{t}_1)$; $h(x_1), h(x_2) \in h(\vec{t}_2); \ldots; h(x_i), h(x) \in h(\vec{t}_{i+1})$. Moreover, since h is an isomorphism between $N_m(a)$ and $N_m(b)$, we get that each $h(\vec{t}_j) \in \overline{R}_s \cap S_m(b)^{p_s}$ for some s. From this we immediately see that $h(x) \in S_l(b)$. Now, applying the same argument to h^{-1} we obtain that for each $y \in S_l(b)$, $h^{-1}(y) \in S_l(a)$, and thus h restricted to $S_l(a)$ maps $S_l(a)$ onto $S_l(b)$. $\square$

Claim 3.8 *Let h be an isomorphism between $N_m(a)$ and $N_m(b)$. Let $\vec{x}$ be a tuple from $S_l(a)$. Assume that $k + l \leq m$. Then $h(S_k(\vec{x})) = S_k(h(\vec{x}))$. In particular, $N_k(\vec{x})$ and $N_k(h(\vec{x}))$ are isomorphic.*

Proof. The proof above applies verbatim to show that for any x with $d(a,x) \leq l$, the isomorphism h maps $S_k(x)$ onto $S_k(h(x))$ for $k \leq m - l$. Thus, h maps $S_k(\vec{x})$ onto $S_k(h(\vec{x}))$. Using this together with the fact that h is an isomorphism and $S_k(\vec{x}) \subseteq S_m(a)$ and $S_k(h(\vec{x})) \subseteq S_m(b)$ we obtain as desired that $N_k(\vec{x})$ and $N_k(h(\vec{x}))$ are isomorphic. $\square$

We now return to proving Lemma 3.2. First, note the following. Assume $d(x, y) > 2r + 1$. Then, for any τ-relation in the structure $N_r(x, y)$, and any tuple t in that relation, either all components of t belong to $S_r(x)$, or all components of t belong to $S_r(y)$. Indeed, if there is a tuple with components $a \in S_r(x)$ and $b \in S_r(y)$, then $d(x, y) \leq d(x, a) + d(a, b) + d(b, y) \leq 2r + 1$. In such a case (that is, when $d(x, y) > 2r + 1$) we also say that $N_r(x, y)$ is the disjoint union of $N_r(x)$ and $N_r(y)$. Note that $N_r(x, y)$ is a τ_2-structure, but both $N_r(x)$ and $N_r(y)$ are τ_1-structures. The following claim will be used often in the proof.

Claim 3.9 *Assume that $d(x, y) > 2r + 1$ and $d(x', y') > 2r + 1$. Assume also that $N_r(x) \cong N_r(x')$ and $N_r(y) \cong N_r(y')$. Then $N_r(x, y) \cong N_r(x', y')$.* □

Indeed, using the observation above, we can define the isomorphism component-wise.

Now, let $d \geq 3r + 1$ (so that $d - r \geq 2r + 1$) and $a \approx_d b$. Fix an isomorphism $h : N_d(a) \to N_d(b)$; in particular $h(a) = b$. There are two cases.

Case 1: $S_{d-r}(a) \cap S_{d-r}(b) = \emptyset$. Then we define π as follows:

$$\pi(x) \quad = \quad \begin{cases} h(x) & \text{if } x \in S_{d-r}(a) \\ h^{-1}(x) & \text{if } x \in S_{d-r}(b) \end{cases}$$

If $x \in S_{d-r}(a)$, then $N_r(a, x) \subseteq N_d(a)$ and hence $N_r(a, x) \cong N_r(h(a), h(x)) = N_r(b, \pi(x))$. If $x \in S_{d-r}(b)$, then $N_r(a, x)$ is the disjoint union of $N_r(a)$ and $N_r(x)$ and hence is isomorphic to the disjoint union of $N_r(b)$ and $N_r(h^{-1}(x)) = N_r(\pi(x))$, that is, to $N_r(b, \pi(x))$. This proves Case 1.

Case 2: $S_{d-r}(a) \cap S_{d-r}(b) \neq \emptyset$. We need a few definitions first. Let $N^a = S_{d-r}(a) - S_{d-r}(b)$, $N^b = S_{d-r}(b) - S_{d-r}(a)$, and $X = str(a) \cap S_{d-r}(b)$. Then we define the following sets:

$$\begin{array}{rcl} A_0 & = & \{x \in N^a \mid h(x) \in X\} \\ A_1 & = & h(A_0) \subseteq X \\ B_0 & = & \{x \in N^b \mid h^{-1}(x) \in X\} \\ B_1 & = & h^{-1}(B_0) \subseteq X \\ M^a & = & N^a - A_0 \\ M^b & = & N^b - B_0 \\ X_0 & = & X - (A_1 \cup B_1) \end{array}$$

It is not hard to see that these 7 sets cover $S_{d-r}(a, b)$ and that in fact only A_1 and B_1 can have nonempty intersection.

We first note that if $x \in M^a$, then $h(x) \in M^b$. Indeed, since $h(x) \in S_{d-r}(b)$, we have $h(x) \in A_1 \cup X_0 \cup B_0 \cup B_1 \cup M^b$. Since $h(x) \notin X$ (otherwise we would have $x \in A_0$), we have $h(x) \in B_0 \cup M^b$. Assuming $h(x) \in B_0$, we get $x = h^{-1}(h(x)) \in B_1$, which contradicts the assumption. Hence, $h(x) \in M^b$. Similarly, if $y \in M^b$, then $h^{-1}(y) \in M^a$.

Claim 3.10 *For any $x \in A_0$ there is $m > 1$ such that $h^m(x) \in B_0$.*

Proof. We have $y = h(x) \in A_1$. By the above remark, $h(y) = h^2(x) \notin M^b$. If $h(y) \in A_1$, then $y \in A_0$, which is impossible since $A_0 \cap A_1 = \emptyset$. Thus, for $y \in A_1$, we have $h(y) \in X_0 \cup B_0 \cup B_1$; in particular, $h^2(x) \in X_0 \cup B_0 \cup B_1$. If $h^2(x) \in B_0$, we are done; if $h^2(x) \in B_1$ then $h^3(x) \in B_0$ and we are done. Otherwise we see that $h^3(x) \in X_0 \cup B_1$; so again if we have $h^3(x) \in B_1$, then $h^4(x) \in B_0$. Continuing, we see that the only possible way for $h^m(x)$ to be outside of B_0 is if we have $h^i(x) \in X_0$ for every $i > 1$. Since X_0 is finite, we have that $h^i(x) = h^j(x)$ for some $j > i > 1$; we assume that i is the minimal such. Then $h(h^{i-1}(x)) = h(h^{j-1}(x))$ but $h^{i-1}(x) \neq h^{j-1}(x)$, which contradict injectivity of h. This shows that $h^m(x) \in B_0$ for some m. □

Claim 3.11 *For any $y \in B_0$ there is $x \in A_0$ and $m > 1$ such that $h^m(x) = y$.*

Proof. The argument is just dual to the proof above. Apply the proof above to h^{-1} to get $x \in A_0$ by a number of applications of h^{-1}. □

Using Claims 3.10 and 3.11, we define a function $p : A_0 \to B_0$ by letting $p(x)$ be $h^m(x)$, where m is the minimum such that $h^m(x) \in B_0$.

Claim 3.12 *The function p is 1-1 and onto.*

Proof. It follows from Claim 3.11 that p is onto. To see that it is 1-1, assume that $p(x) = p(x')$ for some $x, x' \in A_0$. Then for some $m, m' > 1$, $p(x) = h^m(x)$ and $p(x') = h^{m'}(x')$. Assume without loss of generality that $m \geq m'$ and applying h^{-1} m' times, we obtain $h^{m-m'}(x) = x'$. Since no h-image of an element of A_0 can be in A_0, we get $m = m'$ and thus $x = x'$. □

Claim 3.13 *For every $x \in A_0$, $N_r(x) \cong N_r(p(x))$.*

Proof. Let $p(x) = h^m(x)$ for $m > 1$. It follows from the proof of Claim 3.10 that $x = h^0(x), h(x), \ldots, h^{m-1}(x) \in S_{d-r}(a)$. Thus, for every $0 \leq i \leq m-1$, $S_r(h^i(x)) \subseteq S_d(a)$ and hence h is defined on all these spheres. Applying Claim 3.8 we see $N_r(h^i(x)) \cong N_r(h^{i+1}(x))$ for any $i \leq m-1$. Thus $N_r(x) \cong N_r(h^m(x)) = N_r(p(x))$. □

Now we define the map π by cases:

$$\pi(x) \;=\; \begin{cases} h(x) & \text{if } x \in S_{d-r}(a) \\ h^{-1}(x) & \text{if } x \in M^b \\ p^{-1}(x) & \text{if } x \in B_0 \end{cases}$$

Claim 3.14 *π is a permutation on $S_{d-r}(a, b)$.*

Proof. It follows from the definition that π is defined everywhere on $S_{d-r}(a, b)$. To see that π is injective, note that each of its components is, so we only need to consider cases when two arguments correspond to different cases in the definition of π.

Now for the case where $x \in S_{d-r}(a)$ and $y \in M^b$, we have $\pi(x) = h(x) \in S_{d-r}(b)$ and $\pi(y) = h^{-1}(y) \in M^a$; hence $\pi(x) \neq \pi(y)$. For the case where $x \in S_{d-r}(a)$ and $y \in B_0$, we have again $\pi(x) \in S_{d-r}(b)$ and $\pi(y) = p^{-1}(y) \in A_0$; hence $\pi(x) \neq \pi(y)$. For the case where $x \in M^b$ and $y \in B_0$, we have $\pi(x) \in M^a$ and $\pi(y) \in A_0$ and again $\pi(x) \neq \pi(y)$.

It remains to show that π is onto. First, all $S_{d-r}(b)$ is covered since h is an isomorphism. Let $x \in M^a$. Then $y = h(x) \in M^b$ and $x = \pi(y) = h^{-1}(h(x))$. Finally, if $x \in A_0$, then for $y = p(x) \in B_0$ we have $x = \pi(y)$. □

Claim 3.15 *For any $x \in S_{d-r}(a) \cup S_{d-r}(b)$, $N_r(a, x) \cong N_r(b, \pi(x))$.*

Proof. We need to consider three cases, corresponding to the definition of π. The first case is when $x \in S_{d-r}(a)$. Then $S_r(a, x) \subseteq S_d(a)$ and we have, by Claim 3.8, $N_r(a, x) \cong N_r(h(a), h(x)) = N_r(b, \pi(x))$. The second case is when $x \in M^b$. Then $N_r(a, x)$ is the disjoint union of $N_r(a)$ and $N_r(x)$. Since $\pi(x) = h^{-1}(x) \in M^a$, $N_r(b, \pi(x))$ is the disjoint union of $N_r(b)$ and $N_r(\pi(x))$ and we get $N_r(a, x) \cong N_r(b, \pi(x))$ from $N_r(x) \cong N_r(h^{-1}(x))$. The third and final case is when $x \in B_0$. Here we know that for $y = p^{-1}(x) = \pi(x)$, $N_r(y) \cong N_r(x)$. Thus, $N_r(a, x)$ is the disjoint union of $N_r(a)$ and $N_r(x)$, and is thus isomorphic to the disjoint union of $N_r(b)$ and $N_r(y)$, which is $N_r(b, \pi(x))$. □

This finishes the proof of Case 2, and thus the lemma.

Proof of Lemma 3.6. Fix $\mathcal{A} \in \text{STRUCT}[\tau]$. Let $\vec{a} = (a_1, \ldots, a_{n-1})$ and $\vec{b} = (b_1, \ldots, b_{n-1})$ be such that $N_{3r+1}(\vec{a}) \cong N_{3r+1}(\vec{b})$. Let f be an isomorphism. To prove the lemma, we must show that $\mathcal{A} \models \psi'(\vec{a})$ implies $\mathcal{A} \models \psi'(\vec{b})$.

Let $\mathcal{A} \models \psi'(\vec{a})$. Then $\mathcal{A} \models \psi(\vec{a}')$ where $\vec{a}'$ is obtained from $\vec{a}$ by inserting a new element a as one of the components. Without loss of generality, we assume that $\mathcal{A} \models \psi(a_1, \ldots, a_{n-1}, a)$ for some $a \in A$. We now show that there exists $b \in A$ such that $\mathcal{A} \models \psi(b_1, \ldots, b_{n-1}, b)$.

First, we consider the case when $d(a, a_i) \leq 2r + 1$ for some a_i; that is, $a \in S_{2r+1}(\vec{a})$. Then $S_r(a) \subseteq S_{3r+1}(\vec{a})$, and from this we conclude that $N_r(a_1, \ldots, a_{n-1}, a) \cong N_r(b_1, \ldots, b_{n-1}, f(a))$. Thus, b can be taken to be $f(a)$.

Now assume that $d(a, a_i) > 2r + 1$ for all $i = 1, \ldots, n-1$. Then $N_r(a_1, \ldots, a_{n-1}, a)$ is the disjoint union of $N_r(\vec{a})$ and $N_r(a)$ in the same sense as defined in the proof of Lemma 3.2. Now we claim that there exists a $b \in A$ such that $b \notin S_{2r+1}(\vec{b})$ and $N_r(b) \cong N_r(a)$. Note that this is sufficient to conclude the lemma: for such an element b, we have that $N_r(b_1, \ldots, b_{n-1}, b)$ is the disjoint union of $N_r(\vec{b})$ and $N_r(b)$ and thus, by Claim 3.9, it is isomorphic to $N_r(a_1, \ldots, a_{n-1}, a)$. Thus, $\mathcal{A} \models \psi(b_1, \ldots, b_{n-1}, b)$.

To prove the existence of b, first notice that if $a \notin S_{2r+1}(\vec{b})$, then we can just take b to be a. Thus, we assume $a \in S_{2r+1}(\vec{b})$. Therefore, $S_r(a) \subseteq S_{3r+1}(\vec{b})$, and thus for $b_0 = f^{-1}(a)$ we have $N_r(b_0) \cong N_r(a)$. Notice that $b_0 \in S_{2r+1}(\vec{a})$ since f^{-1} is the isomorphism of $N_{3r+1}(\vec{b})$ and $N_{3r+1}(\vec{a})$. Now, if $b_0 \notin S_{2r+1}(\vec{b})$, then we are done.

Assume $b_0 \in S_{2r+1}(\vec{b})$ and define $b_1 = f^{-1}(b_0)$. As before, $N_r(b_0) \cong N_r(b_1)$ (and thus $N_r(b_1) \cong N_r(b)$ and $b_1 \in S_{2r+1}(\vec{a})$. If $b_1 \notin S_{2r+1}(\vec{b})$, we are done; otherwise we continue this process by constructing $b_2 = f^{-1}(b_1), b_3 = f^{-1}(b_2)$, etc. One possibility is that this process never ends, that is, for each i and $b_i \in S_{2r+1}(\vec{a}) \cap S_{2r+1}(\vec{b})$ we have that $b_{i+1} = f^{-1}(b_i)$ is again in $S_{2r+1}(\vec{b})$ (and also in $S_{2r+1}(\vec{a})$). Since $S_{2r+1}(\vec{a}) \cap S_{2r+1}(\vec{b})$ is finite, we can find the lexicographically minimal pair (i, j) with $j > i$ such that $b_j = b_i$. If $i = 0$, then $a = f(b_0) = f(b_j) = b_{j-1} \in S_{2r+1}(\vec{a})$, which contradicts $a \notin S_{2r+1}(\vec{a})$. If $i > 0$, then $b_{i-1} = f(b_i) = f(b_j) = b_{j-1}$, contradicting the minimality of (i, j).

Thus, the process of constructing the sequence $b_0, b_1, \ldots$ eventually stops when we have $b_i \in S_{2r+1}(\vec{a}) \cap S_{2r+1}(\vec{b})$ such that $b_{i+1} = f^{-1}(b_i) \notin S_{2r+1}(\vec{b})$. Since $N_r(b_{i+1}) \cong N_r(b_i) \cong \ldots \cong N_r(b_0) \cong N_r(a)$, we find an element $b = b_{i+1}$ such that $b \notin S_{2r+1}(\vec{b})$ and $N_r(b) \cong N_r(a)$. This concludes the proof. $\Box$

4 Bounded Degree Property

A very convenient form of the locality property is called the *bounded degree property.* It says that for structures from $\text{STRUCT}_k[\tau]$ (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. This special case was established for all first-order queries from graphs to graphs in [36] (see also Corollary 3.4).

Definition 4.1 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 $deg(\Psi(\mathcal{A})) \leq f_\psi(k)$ for every $\mathcal{A} \in \text{STRUCT}_k[\tau]$. $\Box$

This property can be used as an easy-to-apply tool for establishing expressiveness bounds of queries. 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 $\text{STRUCT}_k[\tau]$ such that $q(\mathcal{A})$ can realize arbitrarily large sets of degrees on structures $\mathcal{A}$ from $\mathcal{C}$. This is exactly the idea of the proof of Corollary 3.3.

The usefulness of BDP as a tool for proving expressiveness bounds on first-order graph queries was demonstrated in [36]. In this section we prove that every local query has the BDP. From this we can derive generalizations of the result of [36]. For instance, we show that we can use essentially the technique outlined above in the presence of some auxiliary relations, such as the successor relation, or relations of moderate degree [16].

Theorem 4.2 *Every local query has the bounded degree property.*

The proof of this result, which is a generalization of Corollary 3.4, is delayed until the end of the section. For now let us discuss some implications of this result. As a start, we note that the graph bounded degree property result from [36] 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 4.3 *The graph bounded degree property of first-order queries continues to hold in the presence of a successor relation.* $\square$

But what happens if relations more complex than the successor are allowed? For instance, what happens if we allow 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 [16].

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 $g : \mathbb{N} \to \mathbb{R}$ such that g is not bounded by any constant, we say that $\mathcal{C}$ is of **g-moderate degree** if $s_{\mathcal{C}}(n) \leq \log^{o(1)} g(n)$ for all 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 [16].

Proposition 4.4 *Let ψ be a local query. Let $\mathcal{C}$ be a class of structures of g-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*

$$deg(\Psi(\mathcal{A})) \quad < \quad g(n).$$

Proof. According to the proof of Theorem 4.2 to be presented shortly, for any $\mathcal{A} \in \mathcal{C}$ of cardinality n, and for appropriately chosen constants c and d,

$$deg(\Psi(\mathcal{A})) \quad \leq \quad 2^{c \cdot s_{\mathcal{C}}(n)^d}$$

Since $g(n)$ is not bounded by any constant, for each pair of constants C,$D > 0$, we have $\log^{D\delta(n)-1} g(n) < C$ for large enough n. Applying this to $D = d$ and $C = 1/c$ we get, for large enough n,

$$\sqrt[d]{\log^{d\delta(n)-1} g(n)} \quad < \quad \frac{1}{\sqrt[d]{c}}$$

Hence, $\log^{\delta(n)} g(n) < \frac{1}{\sqrt[d]{c}} \cdot \log^{\frac{1}{d}} g(n)$, which implies $s_{\mathcal{C}}(n) \quad < \quad \frac{1}{\sqrt[d]{c}} \cdot \log^{\frac{1}{d}} g(n)$. It follows that $cs_{\mathcal{C}}(n)^d < \log g(n)$ and hence $2^{cs_{\mathcal{C}}(n)^d} < g(n)$. Then $deg(\Psi(\mathcal{A})) \leq 2^{c \cdot s_{\mathcal{C}}(n)^d}$ implies $deg(\Psi(\mathcal{A})) < g(n)$. $\square$

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. We thus have an example of a problem complete for DLOGSPACE [28] that cannot be definable by a local query even in the presence of relations of moderate degree.

More applications of the BDP in the presence of auxiliary relations are given in Section 7. For now, let us provide the proof of Theorem 4.2. We need to show that given a local query $\psi(x_1, \ldots, x_m)$, there is a function $f_\psi : \mathbb{N} \to \mathbb{N}$ such that $deg(\Psi(\mathcal{A})) \leq f_\psi(k)$ for every $\mathcal{A} \in \mathrm{STRUCT}_k[\tau]$.

Fix a $\mathrm{STRUCT}_k[\tau]$ structure $\mathcal{A}$. Fix a local query $\psi(x_1, \ldots, x_m)$. Assume $m > 1$; otherwise the output is a unary relation and $deg(\Psi(\mathcal{A}))$ is at most 2. Assume that each relation symbol R_i in τ has arity p_i, $1 \le i \le l$. Let $p = \sum_i p_i$. Let r be the locality rank of $\psi(x_1, \ldots, x_m)$. Assume without loss of generality that $r > 0$. Let $s_{\mathcal{A}}(d)$ be the maximum size of $S_d(a)$ for $a \in A$. Let $degree_i(x)$ be the ith degree of x in the output of ψ. Under these assumptions, we claim

Lemma 4.5 *Let $d = (2m-2)(2r+1)$. Suppose $a \approx_d b$ and $S_d(a) \cap S_d(b) = \emptyset$. Then $|degree_1(a) - degree_1(b)| \le (2s_{\mathcal{A}}(d))^{m-1}$.*

Proof. We define a permutation π on the set of $(m-1)$-vectors $\vec{t}$ from $A^{m-1} - S_d(a,b)^{m-1}$ such that $\mathcal{A} \models \psi(a, \vec{t})$ iff $\mathcal{A} \models \psi(b, \pi(\vec{t}))$. By $\psi(a, \vec{t})$, where $t = (t_1, \ldots, t_{m-1})$, we mean $\psi(a, t_1, \ldots, t_{m-1})$. If we can find such π, then the maximal difference between $degree_1(a)$ and $degree_1(b)$ is the maximal number of $(m-1)$-tuples having all their components in $S_d(a,b)$. Such a number is at most $(2s_{\mathcal{A}}(d))^{m-1}$.

To define such a map π, we have to partition each vector $\vec{t} = (t_1, \ldots, t_{m-1})$ that does not belong to $S_d(a,b)^{m-1}$ into two subvectors, whose respective $2r+1$-spheres do not intersect. This will allow us to give a definition by cases. The partition is achieved by means of the following construction that uses a sequence of embedded spheres within $S_d(a,b)$.

Let $h : N_d(a) \to N_d(b)$ be an isomorphism. We define the map $h^* : S_d(a,b) \to S_d(a,b)$ by letting $h^*(x) = x$ for $x \in S_d(a)$ and $h^*(x) = h^{-1}(x)$ for $x \in S_d(b)$ (recall that $S_d(a) \cap S_d(b) = \emptyset$). Next, define S_x^1 to be $S_{2r+1}(x)$, and let $S_x^i = S_{i(2r+1)}(x) - S_{(i-1)(2r+1)}(x)$ for $i > 1$.

First we consider the case when $S_a^i = \emptyset$ for some $i \le 2m-2$. If this is so, then $S_{(i-1)(2r+1)}(a)$ is the set of nodes of a connected component in $\mathcal{G}(\mathcal{A})$. From this and $a \approx_d b$ we conclude that $S_{(i-1)(2r+1)}(b)$ is the set of nodes of a connected component in $\mathcal{G}(\mathcal{A})$, and $S_{(i-1)(2r+1)}(a) = S_{d'}(a)$ and $S_{(i-1)(2r+1)}(b) = S_{d'}(b)$ for any $d' \ge (i-1)(2r+1)$. Let $\vec{t}$ be any vector not contained in $S_d(a,b)^{m-1}$. Let $\vec{t_a}$ denote the components of $\vec{t}$ that belong to $S_d(a)$, $\vec{t_b}$ denote the components of $\vec{t}$ that belong to $S_d(b)$, and $\vec{t_0}$ denote the remaining components. Then we see that $S_{2r+1}(\vec{t_a})$, $S_{2r+1}(\vec{t_b})$ and $S_{2r+1}(\vec{t_0})$ are pairwise disjoint. Thus, for each such $\vec{t}$, we define $\pi(\vec{t})$ by applying h^* on the components of $\vec{t_a}$ and $\vec{t_b}$ and the identity function on $\vec{t_0}$. It is easy to see that π is a permutation, and it follows from Claim 3.9 that $N_r(a, \vec{t}) \cong N_r(b, \pi(\vec{t}))$.

Now we consider the case when none of S_a^i and S_b^i is empty for $i \le 2m-2$. We claim that for any vector $\vec{t} = (t_1, \ldots, t_{m-1})$ that does not belong to $S_d(a,b)^{m-1}$, there exists $i \le 2m-2$ such that no t_j is in $S_a^i \cup S_b^i$. Indeed, since $\vec{t} \notin S_d(a,b)^{m-1}$, we have that at most $m-2$ of its components belong to $S_d(a) \cup S_d(b)$. Since $S_d(a)$ is the disjoint union $\bigcup_{j \le 2m-2} S_a^j$ and similarly $S_d(b)$ is the disjoint union $\bigcup_{j \le 2m-2} S_b^j$, we see that at least m of S_a^j's do not contain any element of $\vec{t}$, and at least m of S_b^j's do not contain any element of $\vec{t}$. Thus, there is a j such that neither S_a^j nor S_b^j contains an element of $\vec{t}$. So we define the set $I_{\vec{t}} = \{j \le 2m-2 \mid \vec{t} \cap (S_a^j \cup S_b^j) = \emptyset\}$. Since $I_{\vec{t}} \ne \emptyset$, define $i_{\vec{t}}$ as the minimum element of this set.

For any vector $\vec{t}$, we define $\vec{t_0}$ as its subvector consisting of those components that belong to $\bigcup_{j < i_{\vec{t}}} (S_a^j \cup S_b^j)$, and $\vec{t_1}$ as the vector containing the remaining components of $\vec{t}$. Note that for any $\vec{t} \notin S_d(a,b)^{m-1}$, $\vec{t_1}$ is nonempty.

We are now ready to define the map π. Given a vector $\vec{t}$, if $i_{\vec{t}} = 1$, then $\pi(\vec{t})$ is defined to be $\vec{t}$.

Otherwise, $\pi(\vec{t})$ is obtained by applying h^* to each component of $\vec{t_0}$, and leaving $\vec{t_1}$ intact. It is easy to see that on vectors with some components not in $S_d(a,b)^{m-1}$, the mapping π is injective. Since h is an isomorphism and $S_d(a) \cap S_d(b) = \emptyset$, there exists an inverse to h^*. This shows that π is onto: for any $\vec{t} = \vec{t_0} \cup \vec{t_1}$, apply the inverse of h^* to $\vec{t_0}$ to obtain a new vector $\vec{s_0}$. Then $\vec{s} = \vec{s_0} \cup \vec{t_1}$ is mapped by π onto $\vec{t}$. Indeed, since h is an isomorphism, $i_{\vec{s}} = i_{\vec{t}}$, and thus $\pi(\vec{s}) = \vec{t}$.

Finally, we show that for any $\vec{t} \notin S_d(a,b)^{m-1}$, $N_r(a,\vec{t})$ is isomorphic to $N_r(b,\pi(\vec{t}))$. From this by locality we obtain $\mathcal{A} \models \psi(a,\vec{t})$ iff $\mathcal{A} \models \psi(b,\pi(\vec{t}))$. By definition of $\vec{t_0}$ and $\vec{t_1}$, their components are at least at the distance $2r+1$, and hence $N_r(a,\vec{t})$ is the disjoint union of $N_r(a,\vec{t_0})$ and $N_r(\vec{t_1})$. Since h is an isomorphism, every element of $S = \bigcup_{j<i_{\vec{t}}}(S_a^j \cup S_b^j)$ is mapped onto an element of S. Hence, $N_r(b,\pi(\vec{t}))$ is the disjoint union of $N_r(b,h^*(\vec{t_0}))$ and $N_r(\vec{t_1})$. Let $\vec{t_{01}}$ denote the components of $\vec{t_0}$ in $S_d(a)$, and $\vec{t_{02}}$ denote the components of $\vec{t_0}$ in $S_d(b)$. Then $N_r(a,\vec{t_0})$ is the disjoint union of $N_r(\vec{a},\vec{t_{01}})$ and $N_r(\vec{t_{02}})$, and $N_r(b,h^*(\vec{t_0}))$ is the disjoint union of $N_r(\vec{b},h(\vec{t_{01}}))$ and $N_r(h^{-1}(\vec{t_{02}}))$. Since $N_r(a,\vec{t_0}) \cong N_r(\vec{b},h(\vec{t_{01}}))$ and $N_r(\vec{t_{02}}) \cong N_r(h^{-1}(\vec{t_{02}}))$, we obtain that $N_r(a,\vec{t_0}) \cong N_r(b,h^*(\vec{t_0}))$ and thus $N_r(a,\vec{t})$ is isomorphic to $N_r(b,\pi(\vec{t}))$. □

Under the same assumptions as Lemma 4.5, we claim

Proposition 4.6 *Let $s = s_{\mathcal{A}}((4m-4)(2r+1))$. Then $deg(\Psi(\mathcal{A})) \leq m \cdot s^m \cdot 2^{1+m+ls^p}$.*

Proof. Let $d = (2m-2)(2r+1)$. It follows from Lemma 4.5 that for any $a \in A$,

$$card(\{degree_1(b) \mid b \approx_d a\}) \;\leq\; 2(2s_{\mathcal{A}}(d))^{m-1} + 1 + s_{\mathcal{A}}(2d).$$

Indeed, for any $b \approx_d a$ such that $b \notin S_{2d}(a)$, we have $S_d(a) \cap S_d(b) = \emptyset$, and thus by Lemma 4.5 the difference between $degree_1(a)$ and $degree_1(b)$ is at most $(2s_{\mathcal{A}}(d))^{m-1}$. Hence, elements outside of $S_{2d}(a)$ contribute at most $2(2s_{\mathcal{A}}(d))^{m-1}+1$ elements to the set $\{degree_1(b) \mid b \approx_d a\}$, from which the observation follows. Multiplying this by m, we obtain the number of different degrees for each isomorphism type of d-neighborhoods. Thus,

$$deg(\Psi(\mathcal{A})) \quad < \quad m \cdot \mathsf{ntp}(d,\mathcal{A}) \cdot (2(2s_{\mathcal{A}}(d))^{m-1} + 1 + s_{\mathcal{A}}(2d))$$

The number $\mathsf{ntp}(d,\mathcal{A})$ is bounded above by the number of nonisomorphic structures of signature τ_1 that have at most $s_{\mathcal{A}}(d)$ elements. That is, $\mathsf{ntp}(d,\mathcal{A}) \leq s_{\mathcal{A}}(d)\prod_{i=1}^{l} 2^{s_{\mathcal{A}}(d)^{p_i}} \leq s_{\mathcal{A}}(d) \cdot 2^{ls_{\mathcal{A}}(d)^p}$. Let $s = s_{\mathcal{A}}(2d)$. Since $s_{\mathcal{A}}(d) \leq s$ and $s \geq 1$ (because $A \neq \emptyset$), we obtain $deg(\Psi(\mathcal{A})) \leq ms2^{ls^p}(2(2s)^{m-1} + 1 + s) \leq ms2^{ls^p}(2^{m+1}s^{m-1}) = m \cdot s^m \cdot 2^{1+m+ls^p}$. □

Finally, we can complete the proof of Theorem 4.2. By assumption, $deg\_set(\mathcal{A}) \subseteq \{0,\ldots,k\}$. Thus $s_{\mathcal{A}}(d) \leq (mkp+1)^d$. Let $f_\psi(k) = m \cdot (mkp+1)^{(4m-4)(2r+1)m} \cdot 2^{1+m+l(mkp+1)^{p(4m-4)(2r+1)}}$. Then we apply Proposition 4.6 and conclude $deg(\Psi(\mathcal{A})) \leq f_\psi(k)$ as desired.

Thus, all local queries have the bounded degree property. However, the converse 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 adding this ψ to first-order logic destoys the bounded degree property of the latter.

5 Stronger Bounded Degree Properties

The astute 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 5.1 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 5.2 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 5.3 *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 4.2, we conclude that

Corollary 5.4 *There are local queries that do not possess the interval or the strong bounded degree properties.* □

The remainder of this section is devoted to proving Theorem 5.3. We need to construct a first-order graph query that does not have the IBDP. First fix $n > 3$, four disjoint sets $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_n\}$, $C = \{c_1, \ldots, c_n\}$, $D = \{d_1, \ldots, d_n\}$, 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, c_j), (c_j, y_i), (y_i, d_j), (d_j, x_i).
- There are no other edges.

It follows straightforwardly from the construction that $deg\_set(G_\pi) = \{n, n+1, n+2, n+3, n+4\}$.

There is a first order formula $A(\cdot)$ in the language of graphs, which has only a binary predicate $E(\cdot, \cdot)$, that is true in G_π only for the node a: This is so because a is the only node with loop that does not have an edge to another node with loop. Looking for other nodes with loops we get that there is a formula $BC(\cdot)$ that is only true of b and c. From this we conclude that there are formulae $X(\cdot)$ true only of x_i's (these have edges to and from a) and $Y(\cdot)$ true only of y_i's (these have edges to and from b and c). Note that the edges of the graph of the function π are the only edges between x's and y's.

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.

For any finite number of variables $z_1, \ldots, z_m$, there is a formula $same_m(z_1, \ldots, z_m)$ true only if z_i's are in the same component: This is true because the transitive-symmetric closure of G_n can be constructed in 4 iterations.

Now define the $\psi(z, y)$ as follows:

$$\begin{aligned}
&A(z) \wedge Y(y) \wedge \\
&(\exists x \exists x' \exists y'. same_5(z, y, x, x', y') \wedge X(x) \wedge X(x') \wedge Y(y') \wedge \\
&E(x, x') \wedge E(y, y') \wedge \\
&E(x, y) \wedge E(x', y'))
\end{aligned}$$

and finally define the first-order graph query Ψ as $\Psi(G) = \{(z, v) \mid G \models \psi(z, v)\}$. The two claims below give us a family of graphs G_n such that each G_n has a degree set consisting of 5 consecutive integers, but $deg(\Psi(G)) \geq n - 3$. The theorem follows immediately.

Claim 5.5 $deg\_set(G_n) = \{n, n+1, n+2, n+3, n+4\}$

Proof. Immediate by construction, because taking disjoint union of G_π's we cannot introduce more in- and out-degrees. □

Claim 5.6 *For any $i < n - 2$, $i \in deg\_set(\Psi(G_n))$.*

Proof. For each $i < n - 2$, consider a permutation π that does the following: for every $j \leq i + 1$, $\pi(j) = j$, and for every $j > i + 1$, $\pi(j) = n - j + i + 2$. Then on the nodes of G_π with such a π, we get that exactly the pairs (a, y_j), where $j \leq i$, can satisfy ψ. So in $\Psi(G_n)$ for this π the node a has outdegree i. This finishes the claim and thus the theorem. □

As a closing remark, note that if we only want to show that there are first-order queries that do not have the SBDP, we can simplify the construction above. Instead of G_π, consider G'_π with $X \cup Y \cup \{a\}$ as the set of nodes and edges (x_i, x_{i+1}), (y_i, y_{i+1}) for $i < n$, (a, x_i) and $(x_i, y_{\pi(i)})$ for $i \le n$, and (a, a). Define G'_n as the disjoint union of G'_πs. We can still test for the a, x or y nodes, and if a number of nodes are in the same component. Now we see that $\mathit{deg\_set}(G'_n) = \{0, 1, 2, n\}$, but again for each $i \le n-2$ we get that $i \in \mathit{deg\_set}(\Psi(G'_n))$ for the same ψ as before.

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 $\mathbb{Q}$ 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, 5]. To obtain our SQL-like language we add arithmetic and a summation operation to model aggregation. The syntax and typing rules of $\mathcal{NRC}(=)$ is given below, using the standard notations of programming language theory [19].

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

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

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

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

We often omit the type superscripts as they can be inferred. Let us briefly recall the semantics, cf. [5]. 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 the union of the results; 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 [41, 44] 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 [5] 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 $\mathbb{Q} \times \mathbb{Q} \to \mathbb{Q}$. Note that as we consider only well-defined queries, we will not encounter the situation of dividing by zero using $\div$. We also add the order on the rationals: $\leq_{\mathbb{Q}}\colon \mathbb{Q} \times \mathbb{Q} \to \mathbb{B}$. 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' GROUP BY dept`. It can be expressed in $\mathcal{NRC}^{\text{aggr}}$ as $\bigcup\{\{(\pi_{dept}\ x,\ \sum\{\!|\mathit{if}\ \pi_{dept}\ x = \pi_{dept}\ y\ \mathit{then}\ \mathit{if}\ \pi_{sex}\ y = \text{'}male\text{'}\ \mathit{then}\ \pi_{salary}\ y\ \mathit{else}\ 0\ \mathit{else}\ 0 \mid y \in EMP|\!\})\} \mid x \in EMP\}$. 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' GROUP BY 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 $\bigcup\{\{(\pi_{dept}\ x,\ \sum\{\!|1 \mid y \in \bigcup\{\mathit{if}\ \pi_{dept}\ z = \pi_{dept}\ x\ \mathit{then}\ \mathit{if}\ \pi_{sex}\ z = \mathit{'male'}\ \mathit{then}\ \{\pi_{salary}\ z\}\ \mathit{else}\ \{\}\ \mathit{else}\ \{\} \mid z \in \mathit{EMP}\}|\!\})\} \mid x \in \mathit{EMP}\}$.

In fact, it was shown in [33, 36] that all (nested) applications of SQL aggregate functions mentioned above can be implemented in $\mathcal{NRC}^{\text{aggr}}$. It is also known [33, 36] that $\mathcal{NRC}^{\text{aggr}}$ has the conservative extension property. A language is said to have the conservative extension property if its expressive power depends only on the height of input and output and is independent of the height of intermediate data. Since $\mathcal{NRC}^{\text{aggr}}$ has this nice property, 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 universe 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 6.1 *For any fixed k, every relational query in $\mathcal{NRC}^{\text{aggr}}$ is local on* STRUCT$_k[\tau]$.

From here, applying verbatim the proof of Theorem 4.2, we conclude

Corollary 6.2 *Relational queries in $\mathcal{NRC}^{\text{aggr}}$ have the bounded degree property.* □

Before we prove Theorem 6.1, let us state some corollaries. We immediately conclude from Corollary 6.2 that

Corollary 6.3 (cf. [36]) $\mathcal{NRC}^{\text{aggr}}$ *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 [43] are two generalized quantifiers for equal cardinality and bigger cardinality respectively. Since these tests can be done in $\mathcal{NRC}^{\text{aggr}}$, and also since every first-order query is $\mathcal{NRC}^{\text{aggr}}$-definable, we obtain:

Corollary 6.4 *Every first-order query with Härtig and Rescher quantifiers has the bounded degree property.* □

In the rest of the section we prove Theorem 6.1. We fix a vocabulary τ, and use $\vec{R}$ to denote a τ-structure, that is, a vector of relations of type of the form $\{b \times \cdots \times b\}$, with the ith one having arity p_i. We first give some technical definitions. Then we develop a normal form result from which the desired theorem drops out readily.

6.1 New definitions

It is a fact that all first-order logic formulas can be rephrased as expressions of $\mathcal{NRC}^{\text{aggr}}$. So for the sake of convenience, in the definitions below we will mix notations from $\mathcal{NRC}^{\text{aggr}}$ and first-order logic, with the understanding that the first-order logic formulas in such mixed notations can be replaced by equivalent expressions of $\mathcal{NRC}^{\text{aggr}}$. Also, recall that in an $\mathcal{NRC}^{\text{aggr}}$ expression such as $\bigcup\{e_1 \mid x \in R\}$, the variable x ranges over objects in R. Thus, if R is a relation of arity p, then x ranges over the tuples of arity p in R. That is, $\mathcal{NRC}^{\text{aggr}}$ uses tuple variables. Note that individual components of tuples can be accessed in $\mathcal{NRC}^{\text{aggr}}$ by using the projection operation. For example, the ith component of a tuple t can be obtained as $\pi_i\ t$. For consistency sake, we will also use tuple variables in our first-order logic formulas below.

Definition 6.5 Let $\vec{R}$ denote a vector of relations of type of the form $\{b \times \cdots \times b\}$. Let $\vec{x}$ denote a vector of tuples of type of the form $b \times \cdots \times b$ appearing in these relations. A **neighborhood formula** is an expression $M(\vec{R}, \vec{x}) : \mathbb{B}$ of $\mathcal{NRC}^{\text{aggr}}$ that is equivalent to a first-order formula of the form given below and moreover it must be satisfiable in the sense that there are sets $\vec{R}$ and tuples $\vec{x}$ such that $M(\vec{R}, \vec{x})$ is true and each tuple in $\vec{x}$ is in some set amongst $\vec{R}$.

$$\begin{aligned}\exists \vec{y} \in \bigcup \vec{R}.\ & \Psi(\vec{x}, \vec{y}) \wedge \\ & \Omega(\vec{R}, \vec{x}, \vec{y}) \wedge \\ & \Delta(\vec{R}, \vec{x}, \vec{y}) \wedge \\ & \Phi(\vec{x}, \vec{y}) \wedge \\ & \forall z \in \bigcup \vec{R}. \Theta(\vec{x}, \vec{y}, z)\end{aligned}$$

where all of the following must be satisfied.

- $\Psi(\vec{x}, \vec{y})$ is a quantifier-free formula that specifies the exact connections between the components in tuples in $\vec{x}$ and $\vec{y}$. In other words, $\Psi(\vec{x}, \vec{y})$ specifies the equality type of tuples in $\vec{x}$ and $\vec{y}$.

  That is, $\Psi(\vec{x}, \vec{y})$ is a conjunction: For each tuple t in $\vec{x}$ or $\vec{y}$, for each tuple t' in $\vec{x}$ or $\vec{y}$, for each component z in t, and for each component z' in t', either $z = z'$ is a conjunct of $\Psi(\vec{x}, \vec{y})$ or $z \neq z'$ is a conjunct of $\Psi(\vec{x}, \vec{y})$. Moreover, $\Psi(\vec{x}, \vec{y})$ has no other conjunct. (In the notations of $\mathcal{NRC}^{\text{aggr}}$, the test $z = z'$ can be written as $\pi_i t = \pi_{i'} t'$, assuming that z is the ith component of t and z' is the i'th component of t'. The test $z \neq z'$ can be similarly expressed.)

- $\Omega(\vec{R}, \vec{x}, \vec{y})$ is a quantifier-free formula that specifies exactly which tuples in $\vec{x}$ and $\vec{y}$ are in which of $\vec{R}$; each of $\vec{x}$ and $\vec{y}$ must be in some $\vec{R}$.

  That is, $\Omega(\vec{R}, \vec{x}, \vec{y})$ is a conjunction: For each tuple t in $\vec{x}$ or $\vec{y}$, and for each relation R in $\vec{R}$, either $R(t)$ is a conjunct of $\Omega(\vec{R}, \vec{x}, \vec{y})$, or $\neg R(t)$ is a conjunct of $\Omega(\vec{R}, \vec{x}, \vec{y})$; and for each t in $\vec{x}$ or $\vec{y}$, there is a R in $\vec{R}$ such that $R(t)$ is a conjunct of $\Omega(\vec{R}, \vec{x}, \vec{y})$.

- $\Delta(\vec{R}, \vec{x}, \vec{y})$ is a formula that specifies the degrees of the components of $\vec{x}$ and $\vec{y}$ in $\vec{R}$.

  That is, the following must be specified for each tuple t amongst $\vec{x}$ and $\vec{y}$, for each component z of t, and for each possible combination of positions ps: the number of tuples t' in $\vec{x}$ such that t' is equal to z at every position listed in ps, the number of tuples t' in $\vec{y}$ such that t' is equal to z at every position listed in ps, and for each relation R, the number of tuples t' in R that is equal to z at every position listed in ps. That is, $\Delta(\vec{R}, \vec{x}, \vec{y})$ is concerned only with the number of connections that the components of $\vec{x}$ and $\vec{y}$ can have; it does not care about other tuples in $\vec{R}$.

- $\Phi(\vec{x}, \vec{y})$ is a quantifier-free formula that says tuples in $\vec{y}$ are distinct and that they are distinct from those in $\vec{x}$.

- $\Theta(\vec{x}, \vec{y}, z)$ is a quantifier-free formula that says z has a component different from all components of $\vec{x}$ and $\vec{y}$ whenever z is not equal to any of these tuples. In other words, if z is not equal to any tuple in $\vec{x}$ and $\vec{y}$, then z must contain a component that is "new." □

A neighborhood formula $M(\vec{R}, \vec{x})$ can be thought of as a complete description (diagram) of a small neighborhood of $\vec{x}$ in $\vec{R}$. The "completeness" of the description is provided by the Θ part of the formula $M(\vec{R}, \vec{x})$. The components that are "new" in the z in Θ are those objects not in the neighborhood.

Definition 6.6 A neighborhood formula $M(\vec{R}, \vec{x})$ is said to have **radius** r if the following two conditions hold:

- All components of tuples in $\vec{y}$ are at most r connections away from some components of tuples in $\vec{x}$. The formula that expresses this fact is implied by the $\Psi(\vec{x}, \vec{y})$ part of $M(\vec{R}, \vec{x})$. Note that the components of tuples in $\vec{y}$ are not required to be close to the same tuple in $\vec{x}$. (A component of a tuple t_1 is said to be r connections away from a component of a tuple t_{r+1} if there are tuples

t_2, ..., t_r such that each pair of tuples t_i and t_{i+1} have a common component, for $1 \leq i \leq r$. This is a straightforward generalization of the notion of path length of between nodes in a graph. For this reason, we use the term "endpoint" to mean the same thing as a "component" of a tuple.)

- All components of tuples in $\vec{x}$ and $\vec{y}$ that are less than r connections away from any endpoints of $\vec{x}$ must have as many connections in $\Psi(\vec{x}, \vec{y})$ as their degrees specified by the $\Delta(\vec{R}, \vec{x}, \vec{y})$ part of $M(\vec{R}, \vec{x})$. This condition ensures that every object within r connections away from $\vec{x}$ appears within $\Psi(\vec{x}, \vec{y})$. □

Here are a few facts about neighborhood formulas. These facts are used implicitly in the rewriting required in Theorem 6.11.

- If each relation in $\vec{R}$ has degree at most k, then for any vector of tuples $\vec{x}$ and for any r, the number of possible (non-equivalent) neighborhood formulas of these tuples having radius r is bounded.
- If two neighborhood formulas of the same tuples $\vec{x}$ in $\vec{R}$ have the same radius r and are consistent with each other, then they are equivalent. (Two such formulas are consistent with each other if they can be satisfied by the same $\vec{x}$ and $\vec{R}$.)
- If two neighborhood formulas of the same tuples in $\vec{R}$ have different radii but are consistent with each other, then the one with the longer radius implies the one with the shorter radius.

Now we define topological parameters of multiple relations. These are defined in terms of the relations and do not refer to any particular tuples. Note that they can be expressed in $\mathcal{NRC}^{\mathrm{aggr}}$.

Definition 6.7 A **topological parameter** of a relation R in $\vec{R}$ with respect to a neighborhood formula $M(\vec{R}, x)$ having radius r is the number of x in R satisfying $M(\vec{R}, x)$. It is a number expressed in $\mathcal{NRC}^{\mathrm{aggr}}$ as $\sum\{\!|\, \mathit{if}\ M(\vec{R}, x)\ \mathit{then}\ 1\ \mathit{else}\ 0 \mid x \in R \,|\!\}$. □

Definition 6.8 A **topological polynomial** $Q(\vec{R})$ is a "polynomial" defined in terms of topological parameters of the R's in $\vec{R}$. That is, it is built up from numeric constants, topological parameters $f_i(\vec{R})$, and arithmetic operators $+$, $-$, and $\cdot$. For example, $Q(\vec{R})$ can be $2 \cdot f_1(\vec{R}) \cdot f_1(\vec{R}) + 3 \cdot f_2(\vec{R}) + 4$. □

Definition 6.9 A **topological predicate** $\mathcal{P}(\vec{R})$ is a Boolean combination of polynomial (in)equations defined in terms of topological parameters of the R's in $\vec{R}$. For example, $\mathcal{P}(\vec{R})$ can be $2 \cdot f_1(\vec{R}) \cdot f_1(\vec{R}) + 3 \cdot f_2(\vec{R}) + 4 \leq 0$. □

6.2 Normal form for relational queries in $\mathcal{NRC}^{\mathrm{aggr}}$

In this subsection we develop a normal form for SQL-like queries on unordered structures whose degrees are bounded by a constant k. Using this normal form, we transfer many powerful results on relational calculus to SQL-like languages. In particular, $\mathcal{NRC}^{\text{aggr}}$ is shown to be local on these structures and to possess the bounded degree property. To simplify the presentation, we look at the situation of having multiple unordered input relations of arbitrary fixed arity. (The results generalize easily to the situation where the relations are of different arities.)

The normal form to be developed shortly basically says that nested use of aggregate functions can be eliminated from all queries provided the input structure has low degree. Thus to develop this normal form, we need a technique for eliminating the nested use of aggregate functions. The essence of this technique is captured by the following result.

Lemma 6.10 *Let $e(\vec{R}, \vec{x}) : \mathbb{Q}$ be an expression of $\mathcal{NRC}^{\text{aggr}}$ of the form*

$$\sum \{\!| \textit{ if } M(\vec{R}, x, \vec{x}) \wedge \mathcal{P}(\vec{R}) \textit{ then } Q(\vec{R}) \textit{ else } 0 \mid x \in R |\!\}$$

where R is one of the relation in $\vec{R}$, $M(\vec{R}, x, \vec{x})$ is a neighborhood formula having radius r, $\mathcal{P}(\vec{R})$ is a topological predicate, and $Q(\vec{R})$ is a topological polynomial. Let every relation in $\vec{R}$ be of degree at most k and $\vec{x}$ be restricted to tuples in these relations. Suppose $M'(\vec{R}, \vec{x})$ is a neighborhood formula having radius $r' > 2 \cdot r$ that is consistent with $M(\vec{R}, x, \vec{x})$. That is, there are sets $\vec{R}$, tuples $\vec{x}$ in sets $\vec{R}$, and tuple x in the set R such that both $M(\vec{R}, x, \vec{x})$ and $M'(\vec{R}, \vec{x})$ are true. Then there is a topological polynomial $Q'(\vec{R})$ such that $e(\vec{R}, \vec{x})$ is equivalent to $Q'(\vec{R}) \cdot Q(\vec{R})$ whenever $M'(\vec{R}, \vec{x})$ and $\mathcal{P}(\vec{R})$ hold.

Proof. The $Q'(\vec{R})$ that we need to construct is simply the number of tuples x in R that satisfy $M(\vec{R}, x, \vec{x})$, given that $M'(\vec{R}, \vec{x})$ and $\mathcal{P}(\vec{R})$ hold. There are four cases to consider.

The first case is when $M(\vec{R}, x, \vec{x})$ specifies that x is not in R. Since x comes from R by definition, this case is never true. Then necessarily $Q'(\vec{R}) = 0$. For the remaining cases, we assume that $M(\vec{R}, x, \vec{x})$ specifies that x is in R.

The second case is when $M(\vec{R}, x, \vec{x})$ specifies that x is equal to one of the elements of $\vec{x}$. Then $Q'(\vec{R}) = 1$ is forced.

The third case is when $M(\vec{R}, x, \vec{x})$ specifies that x is different from all of $\vec{x}$ but is at most r connections away from some of $\vec{x}$. Let $M'(\vec{R}, \vec{x})$ be $\exists \vec{y}.A$. Suppose the vector $\vec{y}$ consists of these tuple variables: $t_1, \ldots, t_m$. Then x can be instantiated to any t_i such that $\exists \vec{y}.A \wedge M(\vec{R}, t_i, \vec{x}) \wedge R(t_i)$ is consistent. Then $Q'(\vec{R})$ is the number of such t_i, which we can easily read off from the given neighborhood formulas.

The fourth case is when $M(\vec{R}, x, \vec{x})$ specifies that x is different from all of $\vec{x}$ and is not within r connections of any $\vec{x}$. Since $M(\vec{R}, x, \vec{x})$ is a neighborhood formula of radius r, we can derive from it a neighborhood formula $M''(\vec{R}, x)$ of x in R having radius r. This can be done by deleting from $M(\vec{R}, x, \vec{x})$ all subformulas involving $\vec{x}$ and all subformulas involving elements of $\vec{y}$ that are not within r connections of x. Let $f(\vec{R}) = \sum \{\!| \textit{ if } M''(\vec{R}, w) \textit{ then } 1 \textit{ else } 0 \mid w \in R |\!\}$; that is, $f(\vec{R})$ is the topological parameter of $\vec{R}$ that tells us how many w in R satisfy the neighborhood formula $M''(\vec{R}, w)$ of radius r. These w's have neighborhoods identical to that specified for x and are thus potential candidates for x. Note that some of these w's may turn out to be "bad" candidates because they are within r connections of some elements of $\vec{x}$. Thus we cannot take $Q'(\vec{R})$ to be $f(\vec{R})$. We must first subtract from $f(\vec{R})$ the number of those w's that are bad. In order to compute the number of such bad w's, we do the following. Let $M'(\vec{R},\vec{x})$ be $\exists\vec{y}.A$. Let $X \subseteq \vec{x}$ denote a maximal subset of $\vec{x}$ satisfying the following two conditions. First, for each tuple t in X, $M'(\vec{R},\vec{x})$ says that t is in R. Second, for any two syntactically distinct tuples t and t' in X, $M'(\vec{R},\vec{x})$ says that they disagree on at least one component. Let $Y \subseteq \vec{y}$ denote the subset of $\vec{y}$ that $M'(\vec{R},\vec{x})$ specifies to be in R. Let D denote the number of $w \in X \cup Y$ such that $\exists\vec{y}.A \wedge M''(\vec{R},w)$ is consistent and that w is within r connections of some $\vec{x}$. The check on w above is possible because $M'(\vec{R},\vec{x})$ has radius $r' > 2 \cdot r$. These w's are those tuples in R that x is not allowed to take. Note that D can be easily read off from the given neighborhood formulas. Then $Q'(\vec{R}) = f(\vec{R}) - D$. This completes the proof. □

We can now provide a normal form result: A query in $\mathcal{NRC}^{\text{aggr}}$ on a structure whose degree is bounded by k can always be rewritten to a form consisting of a chain of *if-then-else* statements where each condition is a topological predicate and each branch is a relational calculus expression. Thus all uses of aggregate functions are at the outermost level of the normal form.

Theorem 6.11 *Let $\vec{R}$ denote a vector of relations of degree at most k. Let $e(\vec{R}) : s$ be an expression of $\mathcal{NRC}^{\text{aggr}}$ with s a type 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 $\mathcal{P}_j(\vec{R})$ is a topological predicate, each $e_j(\vec{R})$ is in $\mathcal{NRC}(=_b)$, and d depends only on k and e.*

Proof sketch. Let $\vec{R}$ denote a structure of degree at most k. Let $e(\vec{R}) : s$ be an arbitrary query in $\mathcal{NRC}^{\text{aggr}}$ with type s of height at most 1. We know that $\mathcal{NRC}^{\text{aggr}}$ has the conservative extension property [33]. So we can assume that $e(\vec{R})$ is a normal form with respect to the rewriting done in the proof of the conservative extension property [33]. Thus it does not use nested sets and that all summations in it have the form $\sum\{\!|e' \mid y \in \pi_i(\vec{R})|\!\}$ and all big unions in it have the form $\bigcup\{e' \mid y \in \pi_i(\vec{R})\}$.

So we can use Lemma 6.10 to remove summation operation from $e(\vec{R})$. This removal can be achieved by applying the lemma starting from summations that are innermost in $e(\vec{R})$ and working outwards. Note that some tedious but straightforward rewriting, similar to those used in the proof of the finite-cofiniteness of $\mathcal{NRC}^{\text{aggr}}$ on multicycles [36], might be necessary before each application of Lemma 6.10. Those facts about neighborhood formulas given in Section 6 are used to justify the rewriting here. The above is done by repeating the main steps below until all summations have been eliminated.

Step 1. We need to prepare, if necessary, the innermost summation in our expression so that it has the form required by Lemma 6.10. For example, the *else*-branch may not be 0. In this case we can use the identity:

- $\sum\{\!|\textit{if } C \textit{ then } E_1 \textit{ else } E_2 \mid x \in R|\!\} = \sum\{\!|\textit{if } C \textit{ then } E_1 \textit{ else } 0 \mid x \in R|\!\} + \sum\{\!|\textit{if } \neg C \textit{ then } E_2 \textit{ else } 0 \mid x \in R|\!\}$.

Another possibility is that the *then*-branch may not be a topological polynomial. In this case, the *then*-branch must have a subexpression involving an *if-then-else*. We need to push it as far out as possible so that it can be absorbed using the identity given above. To do this "pushing," we can apply identities such as:

- *if* E_1 *then* (*if* E_2 *then* E_3 *else* E_4) *else* E_5 = *if* $E_1 \wedge E_2$ *then* E_3 *else* (*if* $E_1 \wedge \neg E_2$ *then* E_4 *else* E_5).
- E_1 *op* (*if* E_2 *then* E_3 *else* E_4) = *if* E_2 *then* E_1 *op* E_3 *else* E_1 *op* E_4, where *op* $\in \{+, -, \cdot, \div\}$.
- (*if* E_2 *then* E_3 *else* E_4) *op* E_1 = *if* E_2 *then* E_3 *op* E_1 *else* E_4 *op* E_1, where *op* $\in \{+, -, \cdot, \div\}$.

A final possibility is that the condition of the *if-then-else* of our innermost summation may not be of the form $M(\vec{R}, x, \vec{x}) \wedge \mathcal{P}(\vec{R})$. Using standard identities of logical connectives, we can assume without loss of generality that the condition is of the form $C \wedge \mathcal{P}(\vec{R})$. We can exploit the fact that the summation is innermost and thus C must be a Boolean combination whose literals are either equality or inequality tests of the components of x and $\vec{x}$. Such a C is equivalent to a finite disjunction of mutually exclusive neighborhood formulas $M_1(\vec{R}, x, \vec{x})$, ..., $M_n(\vec{R}, x, \vec{x})$ of a sufficiently large radius. A simple upper bound for the radius is the number of symbols in C. Thus we can use the following identity to deal with the problem:

- $\sum\{\!|$ *if* $C \wedge \mathcal{P}(\vec{R})$ *then* E *else* $0 \mid x \in R|\!\} = \sum\{\!|$ *if* $M_1(\vec{R}, x, \vec{x}) \wedge \mathcal{P}(\vec{R})$ *then* E *else* $0 \mid x \in R|\!\} + \cdots + \sum\{\!|$ *if* $M_n(\vec{R}, x, \vec{x}) \wedge \mathcal{P}(\vec{R})$ *then* E *else* $0 \mid x \in R|\!\}$.

Step 2. Having made the preparation in Step 1, we can assume that we now have a a summation $E(\vec{R}, \vec{x})$ in $e(\vec{R})$ that has the form $\sum\{\!|$ *if* $M(\vec{R}, x, \vec{x}) \wedge \mathcal{P}(\vec{R})$ *then* $Q(\vec{R})$ *else* $0 \mid x \in R|\!\}$, where $M(\vec{R}, x, \vec{x})$ is a neighborhood formula having radius r, $\mathcal{P}(\vec{R})$ is a topological predicate, and $Q(\vec{R})$ is a topological polynomial. Let $M_1(\vec{R}, \vec{x})$, ..., $M_n(\vec{R}, \vec{x})$ be all the neighborhood formulas of radius $2r + 1$ that are consistent with $M(\vec{R}, x, \vec{x})$. There is only a finite number of such (non-equivalent) neighborhood formulas. By Lemma 6.10, we know that for each $M_i(\vec{R}, \vec{x})$, there is a topological polynomial $Q_i(\vec{R})$ such that $E(\vec{R}, \vec{x})$ is equivalent to $Q_i(\vec{R}) \cdot Q(\vec{R})$ whenever $M_i(\vec{R}, \vec{x})$ and $\mathcal{P}(\vec{R})$ both hold. Thus $E(\vec{R}, \vec{x})$ is equivalent to $E'(\vec{R}, \vec{x})$, which is the following expression: *if* $M_1(\vec{R}, \vec{x}) \wedge \mathcal{P}(\vec{R})$ *then* $Q_1(\vec{R}) \cdot Q(\vec{R})$ *else* ... *else if* $M_n(\vec{R}, \vec{x}) \wedge \mathcal{P}(\vec{R})$ *then* $Q_n(\vec{R}) \cdot Q(\vec{R})$ *else* 0.

Step 3. The application of Step 2 produces a chain of *if-then-else* statements in $E'(\vec{R}, \vec{x})$, which is not in a form to which Lemma 6.10 is applicable. Fortunately, the following identity can be used to rewrite the expression into the appropriate form:

- $\sum\{\!|$ *if* C_1 *then* E_1 *else* ... *else* C_n *then* E_n *else* $0 \mid x \in R|\!\} = \sum\{\!|$ *if* C_1 *then* E_1 *else* $0 \mid x \in R|\!\} + \cdots + \sum\{\!|$ *if* C_n *then* E_n *else* $0 \mid x \in R|\!\}$, if C_1, ..., C_n are mutually exclusive conditions.

This identity is applicable because the $M_i(\vec{R}, \vec{x})$'s above are mutually exclusive.

Step 4. The above rewritings will eventually lead to summations having the form $\sum\{\!|$ *if* $M(\vec{R}, x) \wedge \mathcal{P}(\vec{R})$ *then* $Q(\vec{R})$ *else* $0 \mid x \in R|\!\}$, where the neighborhood formula $M(\vec{R}, x)$ does not mention any additional fixed tuples. Such a summation can be rewritten immediately to *if* $\mathcal{P}(\vec{R})$ *then* $Q'(\vec{R}) \cdot Q(\vec{R})$ *else* 0, where $Q'(\vec{R})$ is the topological parameter defined as $\sum\{\!|$ *if* $M(\vec{R}, x)$ *then* 1 *else* $0 \mid x \in R|\!\}$.

The above 4-step process is repeated until all summations are replaced by topological parameters. The result of rewriting is an expression $e'(\vec{R})$ of $\mathcal{NRC}^{\text{aggr}}$ that does not use the $\sum$ operator, except in the implementation of topological parameters of $\vec{R}$. Note that all these topological parameters must appear inside some topological predicates. We can move all topological predicates in $e'(\vec{R})$ as far out as possible using the identity: $E_1(\vec{R}) = \mathit{if}\ \mathcal{P}(\vec{R})\ \mathit{then}\ E_2(\vec{R})\ \mathit{else}\ E_3(\vec{R})$, where $E_2(\vec{R})$ and $E_3(\vec{R})$ are obtained from $E_1(\vec{R})$ by replacing all occurrences of the topological predicate $\mathcal{P}(\vec{R})$ with *true* and *false* respectively.

The result of these moves is an expression $e''(\vec{R})$ of $\mathcal{NRC}^{\text{aggr}}$ of the form $\mathit{if}\ \mathcal{P}_1(\vec{R})\ \mathit{then}\ e_1(\vec{R})\ ...\ \mathit{else}\ \mathit{if}\ \mathcal{P}_d(\vec{R})\ \mathit{then}\ e_d(\vec{R})\ \mathit{else}\ e_{d+1}(\vec{R})$, where each $\mathcal{P}_i(\vec{R})$ is a topological predicate and each $e_i(\vec{R})$ is in $\mathcal{NRC}(=_b)$. Note that d does not depend on the value of $\vec{R}$. The theorem is thus proved. □

As an illustration of the proof of this theorem, let us consider the query $Q(R) = \bigcup\{\mathit{if}\ \mathit{indeg}(x,R) = \mathit{outdeg}(x,R)\ \mathit{then}\ \{x\}\ \mathit{else}\ \{\} \mid x \in R\}$, where $\mathit{indeg}(x,R) = \sum\{\!|\mathit{if}\ \pi_2(y) = \pi_1(x)\ \mathit{then}\ 1\ \mathit{else}\ 0 \mid y \in R|\!\}$ and $\mathit{outdeg}(x,R) = \sum\{\!|\mathit{if}\ \pi_1(y) = \pi_2(x)\ \mathit{then}\ 1\ \mathit{else}\ 0 \mid y \in R|\!\}$. This query returns those edges in the graph R whose in-degrees equal their out-degrees. We demonstrate the theorem on the smallest interesting bound on the degree of R, namely $k = 1$. According to the proof, we begin on one of the innermost summation, $\mathit{indeg}(x,R)$. This first step is to put it into a form so that Lemma 6.10 applies. It is straightforward to see that $\mathit{indeg}(x,R) = \sum\{\!|\mathit{if}\ M_1^1(R,y,x)\ \mathit{then}\ 1\ \mathit{else}\ 0 \mid y \in R|\!\} + \cdots + \sum\{\!|\mathit{if}\ M_m^1(R,y,x)\ \mathit{then}\ 1\ \mathit{else}\ 0 \mid y \in R|\!\}$, where $M^1_{1\le i\le m}$ are the finite number of neighbourhood formula of radius 1 and each of them specifies that $\pi_2(y) = \pi_1(x)$ and that every node in R has degree at most $k = 1$. The second step is to apply Lemma 6.10 to each $\sum\{\!|\mathit{if}\ M_i^1(R,y,x)\ \mathit{then}\ 1\ \mathit{else}\ 0 \mid y \in R|\!\}$ above. Let $M_{i,j}^3(R,x)$ denote a neighbourhood formula of radius 3 that is consistent with $M_i^1(R,y,x)$ and $Q_{i,j}(R)$ be the topological polynomial corresponding to $Q'(R)$ given by Lemma 6.10. Since R has degree at most 1 and $M_i^1(R,y,x)$ specifies $\pi_2(y) = \pi_1(x)$, it follows that $Q'(R)$ and thus $Q_{i,j}(R)$ equals 1. So each $\sum\{\!|\mathit{if}\ M_i^1(R,y,x)\ \mathit{then}\ 1\ \mathit{else}\ 0 \mid y \in R|\!\}$ above is replaced by $(\mathit{if}\ M_{i,1}^3(R,x)\ \mathit{then}\ 1\ \mathit{else}\ 0) + \cdots + (\mathit{if}\ M_{i,m_i}^3(R,x)\ \mathit{then}\ 1\ \mathit{else}\ 0)$, where $M^3_{i,1\le j\le m_i}(R,x)$ are all the mutually exclusive neighbourhood formula of radius 3 that are consistent with $M_i^1(R,y,x)$. Thus $\mathit{indeg}(x,R) = (\mathit{if}\ M_{1,1}^3(R,x)\ \mathit{then}\ 1\ \mathit{else}\ 0) + \cdots + \mathit{if}\ M_{m,m_m}^3(R,x)\ \mathit{then}\ 1\ \mathit{else}\ 0)$. Note that $\mathit{indeg}(x,R)$ now does not contain any summation. Applying similar transformations, $\mathit{outdeg}(x,R)$ is also reduced to an expression that contains no summation. We can stop at this point, as Step 3 and Step 4 of the proof of the theorem are not needed for this example: $Q(R)$ is already in a form that uses no summation and is in $\mathcal{NRC}(=_b)$.

This normal form theorem gets complicated aggregate functions out of the way. Using it, we can now prove Theorem 6.1.

Proof of Theorem 6.1. Let $\vec{R}$ denote a structure in $\mathrm{STRUCT}_k[\tau]$ whose elements are of base type b. Let $e(\vec{R})$ be a relational query in $\mathcal{NRC}^{\text{aggr}}$. By Theorem 6.11, we can assume that $e(\vec{R})$ has the form $\mathit{if}\ \mathcal{P}_1(\vec{R})\ \mathit{then}\ e_1(\vec{R})\ ...\ \mathit{else}\ \mathit{if}\ \mathcal{P}_d(\vec{R})\ \mathit{then}\ e_d(\vec{R})\ \mathit{else}\ e_{d+1}(\vec{R})$, where each $\mathcal{P}_i(\vec{R})$ is a topological predicate and each $e_i(\vec{R})$ is in $\mathcal{NRC}(=_b)$. Since $\mathcal{NRC}(=)$ enjoys the conservative extension property [44], each e_i can be defined in relational algebra. Hence, by Fact 2.2, every ψ_{e_i} is local and has some finite locality rank r_i. From this we immediately conclude that ψ_e has locality rank $\max_i r_i$, thus proving the theorem. □

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 [10, 8, 11, 42] addressed the issue of maintaining queries such as the transitive closure in first-order or $\mathcal{NRC}^{\text{aggr}}$.

It was shown [8] 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 [8, 11] 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.

In addition to the transitive closure query, 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 7.1 *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 [8] is essentially this: Suppose there is an expression $g(I, I^+, t)$ that, given an input I, the result of a query (transitive closure or same-generation) I^+ on I, and a tuple t in I, produces the output of the query on $I - \{t\}$. (In the case of same-generation, one tuple is removed from A and one from B.) Then both proofs in [8] show how to use this assumption to produce an expression in first-order plus g that computes the transitive closure of a chain. Since the construction of [8] does not assume any auxiliary data, we can apply it here to obtain that, if either transitive closure or same-generation 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. However, this contradicts the remark made after the proof of Proposition 4.4. $\Box$

Using essentially the same argument, but employing Corollary 6.3 we can also prove that

Corollary 7.2 *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.* $\Box$

8 Conclusion

In the past several years, a number of papers dealing with locality in finite-model theory answered most of the questions raised by the conference version of this paper. Thus, in this concluding section, we briefly describe the problems posed by the ICDT'97 version of this paper [9], and give pointers to solutions.

One of the problems posed by [9] was the following: extend results that describe outputs of local queries in terms of $\mathsf{ntp}(d, \mathcal{A})$ from graph queries to arbitrary ones. In this paper, the only extension of this kind was for the Gaifman graph of the output. It turns out that an analog of Theorem 3.1 can be proved for queries of arbitrary arity, with d depending on both locality rank and the arity. For details, see [31].

Another problem mentioned in [9] was to develop techniques for proving languages local. One such technique was proposed in [30] which showed that queries in any reasonable logic that satisfies an analog of Hanf's theorem [24, 16] are local. Using this, and results of [25, 38], the paper [30] showed that first-order logic extended with unary generalized quantifiers is local. In [31], a technique was presented that allows one to prove locality without a recourse to Hanf's theorem. The same paper showed a version of infinitary logic that can define every numerical property, but expresses only local queries when restricted to finite relational structures.

Two problems related to aggregate query languages were posed by [9]. The first one was to prove that every relational query in $\mathcal{NRC}^{\text{aggr}}$ is local. This was done in [37] by using the following technique. For every relational query Q in $\mathcal{NRC}^{\text{aggr}}$, [37] shows how to construct another query Q' with the following two properties: (1) Q is local iff Q' is local, and (2) Q' can be defined in first-order logic extended with counting quantifiers. Since the latter only expresses local queries, as shown in [30], the locality of relational queries in $\mathcal{NRC}^{\text{aggr}}$ follows.

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. It was conjectured by [9] that the bounded degree property can be partially recovered for this language. That is, the conjecture of [9] was that every relational query in $\mathcal{NRC}^{\text{aggr}}(\leq_b)$ that is order-independent has the bounded degree property. This conjecture was recently disproved by L. Hella; the proof can be found in [26].

Acknowledgements. We are greteful to anonymous reviewers for numerous comments and improvements. We thank Moshe Vardi suggesting the extension from Theorem 3.1 to Theorem 3.5. Part of this work was done while Wong was visiting the University of Melbourne and Bell Laboratories at Murray Hill. Wong would like to thank these organizations and fellow coauthors Dong and Libkin 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] J. Albert. Algebraic properties of bag data types. In *Proceedings of 17th International Conference on Very Large Data Bases*, pages 211–219, 1991.

[4] J. Barwise, S. Feferman, editors. *Model-Theoretic Logics*. Springer-Verlag, 1985.

[5] P. Buneman, S. Naqvi, V. Tannen, L. Wong. Principles of programming with complex objects and collection types. *Theoretical Computer Science*, 149(1):3–48, September 1995.

[6] S. Chaudhuri, M. Y. Vardi, Optimization of *real* conjunctive queries, *in* "Proceedings of 12th ACM Symposium on Principles of Database Systems," Washington, D. C., May 1993.

[7] M.P. Consens, A.O. Mendelzon, Low complexity aggregation in GraphLog and Datalog, *Theoretical Computer Science* **116**, No. 1 (1993), 95–116.

[8] G. Dong, L. Libkin, L. Wong. On impossibility of decremental recomputation of recursive queries in relational calculus and SQL. In *Proceedings of 5th International Workshop on Database Programming Languages*, Gubbio, Italy, September 1995. Springer Electronic Workshops in Computing. Available from Springer EWiC server: http: //www.springer.co.uk /eWiC /Workshops /DBPL5.html.

[9] G. Dong, L. Libkin, L. Wong. Local properties of query languages. In *Proceedings of International Conference on Database Theory (ICDT'97)*, Springer LNCS vol. 1186, pages 141–154.

[10] G. Dong and J. Su. Incremental and Decremental Evaluation of Transitive Closure by First-Order Queries. *Information and Computation*, 120(1):101–106, 1995.

[11] G. Dong and J. Su. Space-bounded FOIES. In *Proceedings of 14th ACM Symposium on Principles of Database Systems*, pages 139–150, San Jose CA, May 1995.

[12] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer Verlag, 1995.

[13] A. Ehrenfeucht. An application of games to the completeness problem of formalized theories. *Fundamenta Mathematicae*, 49 (1961), 129–141.

[14] K. Etessami, Counting quantifiers, successor relations, and logarithmic space, *Journal of Computer and System Sciences*, 54(3):400-411, 1997.

[15] R. Fagin. Probabilities on finite models. *Journal of Symbolic Logic*, 41 (1976), 50–58.

[16] R. Fagin, L. Stockmeyer, M. Vardi, On monadic NP vs monadic co-NP, *Information and Computation*, 120 (1994), 78–92.

[17] R. Fraïssé . Sur quelques classifications des systèmes de relations. *Université d'Alger, Publications Scientifiques, Série A*, 1 (1954), 35–182.

[18] H. Gaifman, On local and non-local properties, *in* "Proceedings of the Herbrand Symposium, Logic Colloquium '81," North Holland, 1982.

[19] C. A. Gunter, *Semantics of Programming Languages: Structures and Techniques.* MIT Press, 1992.

[20] T. Griffin, L. Libkin, Incremental maintenance of views with duplicates, *in* "Proceedings of ACM-SIGMOD International Conference on Management of Data," San Jose, CA, May 1995.

[21] S. Grumbach, T. Milo, Towards tractable algebras for bags, *Journal of Computer and System Sciences*, 55(3):570–588, 1996.

[22] S. Grumbach, L. Libkin, T. Milo and L. Wong. Query languages for bags: expressive power and complexity. *SIGACT News*, 27(2):30–37, 1996.

[23] S. Grumbach and C. Tollu. On the expressive power of counting. *Theoretical Computer Science* 149(1): 67–99, 1995.

[24] W. Hanf. Model-theoretic methods in the study of elementary logic. In J.W. Addison et al, eds, *The Theory of Models*, North Holland, 1965, pages 132–145.

[25] L. Hella. Logical hierarchies in PTIME. *Information and Computation*, 129 (1996), 1–19.

[26] L. Hella, L. Libkin and J. Nurmonen. Notions of locality and their logical characterizations over finite models. *Journal of Symbolic Logic*, to appear.

[27] N. Immerman and E. Lander. Describing graphs: A first order approach to graph canonization. In *"Complexity Theory Retrospective"*, Springer Verlag, Berlin, 1990.

[28] N. Immerman, Languages that capture complexity classes, *SIAM Journal of Computing* **16** (1987), 760–778.

[29] A. Klug, Equivalence of relational algebra and relational calculus query languages having aggregate functions, *Journal of the ACM* **29**, No. 3 (1982), 699–717.

[30] L. Libkin. On the forms of locality over finite models. In *Proceedings of 12th IEEE Symposium on Logic in Computer Science*, 204–215, Warsaw, Poland, 1997.

[31] L. Libkin. On counting logics and local properties. In *Proceedings of 13th IEEE Symposium on Logic in Computer Science*, 501–512, Indianapolis, Indiana, 1998.

[32] L. Libkin, L. Wong, Some properties of query languages for bags, *in* "Proceedings of 4th International Workshop on Database Programming Languages," Manhattan, New York, August 1993.

[33] L. Libkin, L. Wong, Aggregate functions, conservative extension, and linear orders, *in* "Proceedings of 4th International Workshop on Database Programming Languages," Manhattan, New York, August 1993.

[34] L. Libkin, L. Wong, Conservativity of nested relational calculi with internal generic functions, *Information Processing Letters* **49** (1994), 273–280.

[35] L. Libkin, L. Wong, On representation and querying incomplete information in databases with bags, *Information Processing Letters* **56** (1995), 209–214.

[36] L. Libkin, L. Wong, Query languages for bags and aggregate functions. *Journal of Computer and System Sciences*, 55(2):241–272, 1997.

[37] L. Libkin and L. Wong, On the power of aggregation in relational query languages. *in* "Proceedings of 6th International Workshop on Database Programming Languages," Springer LNCS vol. 1369, 1998, pages 260–280.

[38] J. Nurmonen. On winning strategies with unary quantifiers. *J. Logic and Computation*, 6 (1996), 779–798.

[39] J. Nurmonen. Unary quantifiers and finite structures. PhD Thesis, University of Helsinki, 1996.

[40] 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(4):566–592, 1987.

[41] J. Paredaens and D. Van Gucht. Converting nested relational algebra expressions into flat algebra expressions. *ACM Transaction on Database Systems*, 17(1):65–93, March 1992.

[42] S. Patnaik and N. Immerman. Dyn-FO: A parallel dynamic complexity class. *Journal of Computer and System Sciences*, 55(2):199–209, 1997.

[43] J. Väänänen, Generalized quantifiers. *Bulletin of the EATCS*, 62:115–136, 1997.

[44] L. Wong, Normal forms and conservative properties for query languages over collection types, *Journal of Computer and System Sciences*, 52(3):495–505, 1996.