December 1993

A Bounded Degree Property and Finite-Cofiniteness of Graph Queries

Leonid Libkin
University of Pennsylvania

Limsoon Wong
University of Pennsylvania

A Bounded Degree Property and Finite-Cofiniteness of Graph Queries

Abstract

We provide new techniques for the analysis of the expressive power of query languages for nested collections. These languages may use set or bag semantics and may be further complicated by the presence of aggregate functions. We exhibit certain classes of graphics and prove that properties of these graphics that can be tested in such languages are either finite or cofinite. This result settles that conjectures of Grumbach, Milo, and Paredaens that parity test, transitive closure, and balanced binary tree test are not expressible in bah languages like BALG of Grumbach and Milo and *BQL* of Libkin and Wong. Moreover, it implies that many recursive queries, including simple ones like test for a chain, cannot be expressed in a nested relational language even when aggregate functions are available. In an attempt to generalize the finite-cofiniteness result, we study the bounded degree property which says that the number of distinct in- and out-degrees in the output of a graph query does not depend on the size of the input if the input is "simple." We show that such a property implies a number of inexpressibility results in a uniform fashion. We then prove the bounded degree property for the nested relational language.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-93-95.

A Bounded Degree Property and Finite-Cofiniteness of Graph Queries

MS-CIS-93-95
LOGIC & COMPUTATION 75

Leonid Libkin
Limsoon Wong

University of Pennsylvania
School of Engineering and Applied Science
Computer and Information Science Department
Philadelphia, PA 19104-6389

December 1993

A Bounded Degree Property and Finite-Cofiniteness of Graph Queries

Leonid Libkin* Limsoon Wong†

Department of Computer and Information Science
University of Pennsylvania, Philadelphia, PA 19104-6389, USA
email: {libkin, limsoon}@saul.cis.upenn.edu

Abstract

We provide new techniques for the analysis of the expressive power of query languages for nested collections. These languages may use set or bag semantics and may be further complicated by the presence of aggregate functions. We exhibit certain classes of graphs and prove that properties of these graphs that can be tested in such languages are either finite or cofinite. This result settles the conjectures of Grumbach, Milo, and Paredaens that parity test, transitive closure, and balanced binary tree test are not expressible in bag languages like BALG of Grumbach and Milo and $\mathcal{BQL}$ of Libkin and Wong. Moreover, it implies that many recursive queries, including simple ones like test for a chain, cannot be expressed in a nested relational language even when aggregate functions are available. In an attempt to generalize the finite-cofiniteness result, we study the bounded degree property which says that the number of distinct in- and out-degrees in the output of a graph query does not depend on the size of the input if the input is "simple." We show that such a property implies a number of inexpressibility results in a uniform fashion. We then prove the bounded degree property for the nested relational language.

1 Introduction

As the relational algebra and the relational calculus are standard languages for relational databases, upon which most other approaches are based, there was a search for a standard language for nested relations and, more generally, nested collections. Several researchers have found such languages: Schek and Scholl [15]; Thomas and Fischer [16]; Colby [5]; Abiteboul and Kanellakis [1]; Breazu-Tannen, Buneman, and Wong [2]. All the discovered algebras and calculi have been proven to possess the same expressive power. So we can speak of *the* nested relational algebra or calculus.

The practical versions of these languages are further complicated by the presence of aggregate functions and arithmetic operations. Moreover, to implement these aggregate functions, they often use *bag* semantics. That is, duplicates are not removed. Many results on relational languages do not carry over to languages with bag semantics; see Chaudhuri and Vardi [4]. Adding aggregate functions to

*Supported in part by NSF Grant IRI-90-04137 and AT&T Doctoral Fellowship.
†Supported in part by NSF Grant IRI-90-04137 and ARO Grant DAALO3-89-C-0031-PRIME.

the nested relational algebra was studied by us in [11, 12]. In particular, we showed that the language thus obtained is equivalent to the nested *bag* algebra, called BALG in Grumbach and Milo [9] and $\mathcal{BQL}$ by us in [11]. This confirmed the intuition that bags essentially give us an ability to work with numbers.

There are many problems in analyzing these of languages. The flat relational algebra is an algebraization of first-order logic. Therefore, to answer questions about expressibility of the flat relational languages based on the relational algebra one can use a rich body of results known about first-order expressibility, as in Chandra and Harel [3] and Fagin [6]. However, calculi for nested relations are essentially higher-order logics, where very little is known about expressibility over finite structures.

Therefore, new techniques are needed for analyzing languages for nested collections. A difficulty seems to be that, in writing simple queries, one can increase the level of nesting in the intermediate data and then get a desired result by flattening or unnesting. Unless there is some restriction for doing this, there is very little hope for finding nice tools for analyzing expressiveness of such languages.

Fortunately, queries of the nested relational algebra were shown to be independent of the height of set nesting in the intermediate data. The first result of this kind was proved by Paredaens and Van Gucht [14] for queries over flat relations. It was later generalized by Wong [17] to arbitrary queries. Recently, we showed that it continues to hold in the presence of aggregate functions [12]. This property provides the simplifying tools we need to analyze our languages.

Since $\mathcal{BQL}$ has built-in arithmetic, it is hard to find a logic that would capture it. Thus it is not clear which techniques can be used for proving results about its expressive power. There are several conjectures on $\mathcal{BQL}$ and the nested relational algebra, formulated by Grumbach and Milo [9] and Paredaens [13]:

Conjecture 1 (Grumbach and Milo) *Parity test is not definable in $\mathcal{BQL}$.*

Conjecture 2 (Grumbach and Milo) *Transitive closure is not definable in $\mathcal{BQL}$.*

Conjecture 3 (Paredaens) *Test for balanced binary trees is neither definable in the nested relational algebra nor in $\mathcal{BQL}$.*

BALG and $\mathcal{BQL}$ can be embedded in the nested relational language with aggregate functions [11]. Therefore, it suffices to solve conjectures 1, 2, and 3 in the latter. That conjecture 1 is true was shown by us in [11]. In this paper, among other things, we prove conjectures 2 and 3 as well.

Before we outline the main results of this paper, let us make a few observations. In most cases when people conjecture that something like transitive closure is not expressible in a language, they actually mean that a language is incapable of expressing recursive queries. Transitive closure just happens to be the most famous example of a recursive query, but it is not the simplest one. As Immerman [10] showed, the first-order logic with transitive closure captures the complexity class NLOGSPACE over ordered structures. There are (possibly) simpler classes and complete problems for them. For example, DLOGSPACE is captured by the first-order logic with *deterministic* transitive closure [10].

Therefore, if we could show that deterministic transitive closure is not expressible in the language that has at least the power of the first-order logic, then many other inexpressibility results will be obtained for free (for instance, connectivity and transitive closure). In fact, we exhibit two queries which are at most as hard as the deterministic transitive closure, one of them being a test for balanced binary trees, and show that they are not expressible in the languages we study.

We also face a lack of uniformity in proving inexpressibility results. There are well-known tools for proving first-order inexpressibility over finite structures, such as Ehrenfaucht-Fraïssé games. However, applying them to any query whose inexpressibility is to be proved, is a separate combinatorial problem, which is sometimes a nontrivial one. This is, perhaps, one of the reasons Paredaens formulated his balanced binary tree conjecture for the nested relational algebra as well. For the Paredaens conjecture, it may be easier to use another technique, Hanf's lemma (as presented in [7]), but it still requires some combinatorial proof which no longer works if we ask about balanced ternary, 4-ary etc. trees. In this paper we demonstrate a uniform technique for proving various inexpressibility results for the nested relational calculus that does not have this deficiency.

Organization. The rest of the paper is organized in four sections. In section 2 we describe the nested relational calculus $\mathcal{NRC}$ and its enhancement with aggregate functions $\mathcal{SQL}$ as in [12]. We choose this presentation of the nested relational calculus because of the simplicity of its syntax and the availability of tools for analyzing it. We formulate the conservativity results for $\mathcal{NRC}$ and $\mathcal{SQL}$.

In section 3 we define our sample queries and show that they are at most as hard as deterministic transitive closure. We then present the *bounded degree property* of a language and show how it implies various inexpressiblity results in a uniform fashion. We prove this property for the nested relational calculus.

In section 4 we study the expressive power of $\mathcal{SQL}$. Using conservativity, we first state that properties of naturals it can define are either finite or co-finite. Then we prove a much more involved result that properties of certain graphs definable in $\mathcal{SQL}$ are also *finite-cofinite.* From this result, we derive the inexpressibility of our sample queries in $\mathcal{SQL}$ and $\mathcal{BQL}$. Finally, we show that complete problems for a number of complexity classes below PTIME cannot be expressed in $\mathcal{SQL}$. Concluding remarks are given in section 5.

2 Languages $\mathcal{NRC}$ and $\mathcal{SQL}$ and conservative extension

In this section we define the nested relational calculus $\mathcal{NRC}$ as in Breazu-Tannen, Buneman and Wong [2] and its extension with aggregate functions as in Libkin and Wong [12].

A type in $\mathcal{NRC}$ is either a complex object type or is a function type $s \to t$ where s and t are complex object types. The complex object types are given by the following grammar:

$$s, t ::= b \mid \mathbf{B} \mid unit \mid s \times t \mid \{s\}$$

Here b ranges over some collection of unspecified base types. Objects of type $\mathbf{B}$ are the two boolean values *true* and *false*. Type *unit* has a unique object denoted by (). Objects of type $s \times t$ are pairs whose first components are objects of type s and second components are objects of type t. Objects of type $\{s\}$ are finite sets of objects of type s. Expressions of $\mathcal{NRC}$ are given in figure 1.

Lambda Calculus and Products

$$\frac{}{x^s : s} \qquad \frac{e : t}{\lambda x^s.e : s \to t} \qquad \frac{e_1 : s \to t \quad e_2 : s}{e_1\ e_2 : t}$$

$$\frac{}{() : \mathit{unit}} \qquad \frac{e_1 : s \quad e_2 : t}{(e_1, e_2) : s \times t} \qquad \frac{e : s \times t}{\pi_1\ e : s} \qquad \frac{e : s \times t}{\pi_2\ e : t}$$

Set Monad

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

Booleans

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

Figure 1: Expressions of $\mathcal{NRC}$

Semantics (see [2, 12]). The lambda calculus, product, and boolean constructs are standard. We briefly describe the meaning of the monad constructs here. $\{\}$ is the empty set. $\{e\}$ is the singleton set containing e. $e_1 \cup e_2$ is the union of sets e_1 and e_2. The construct $\bigcup\{e_1 \mid x \in e_2\}$ denotes the set obtained by first applying the function $\lambda x.e_1$ to elements of the set e_2 and then taking their big union. Hence $\bigcup\{e_1 \mid x \in e_2\} = f(o_1) \cup \ldots \cup f(o_n)$, where f is the function $\lambda x.e_1$ and $\{o_1, \ldots, o_n\}$ is the set e_2. It must be stressed that the $x \in e_2$ part in the construct $\bigcup\{e_1 \mid x \in e_2\}$ is not a membership test; it is the introduction of a new variable x whose scope is the subexpression e_1.

The language $\mathcal{SQL}$ is obtained by adding the type of rationals $\mathbb{Q}$ and the following constructs to $\mathcal{NRC}$:

Arithmetic

$$\frac{e_1 : \mathbb{Q} \quad e_2 : \mathbb{Q}}{e_1 + e_2 : \mathbb{Q}} \qquad \frac{e_1 : \mathbb{Q} \quad e_2 : \mathbb{Q}}{e_1 \cdot e_2 : \mathbb{Q}}$$

$$\frac{e_1 : \mathbb{Q} \quad e_2 : \mathbb{Q}}{e_1 \div e_2 : \mathbb{Q}} \qquad \frac{e_1 : \mathbb{Q} \quad e_2 : \mathbb{Q}}{e_1 - e_2 : \mathbb{Q}}$$

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

The semantics of $\sum\{\!|e_1 \mid x^s \in e_2|\!\}$ is $f(o_1) + \ldots + f(o_n)$, where f is the function $\lambda x.e_1$ and $\{o_1, \ldots, o_n\}$

is the set e_2. All standard aggregate functions found in commercial databases can be expressed using $\sum$; see [12].

Remark. A sublanguage of $\mathcal{SQL}$ obtained by restricting $\mathbb{Q}$ to $\mathbb{N}$, removing $\div$, and using monus instead of minus is equivalent to the nested bag languages BALG and $\mathcal{BQL}$ of [9, 11]. Hence, any inexpressibility result for $\mathcal{SQL}$ implies a similar result for $\mathcal{BQL}$.

Let us now define the concept of *conservative extension.* The set height $ht(s)$ of a type s is defined by induction on the structure of type: $ht(unit) = ht(b) = 0$, $ht(s \times t) = ht(s \to t) = \max(ht(s), ht(t))$, and $ht(\{s\}) = 1 + ht(s)$. The set height of an expression e is defined as $ht(e) = \max\{ht(s) \mid s$ occurs in the unique type derivation of $e\}$. Let $\mathcal{L}_{i,o,h}$ denote the class of functions whose input has set height at most i, whose output has set height at most o, and which are definable in the language $\mathcal{L}$ using an expression whose set height is at most $h \geq \max(i, o)$. $\mathcal{L}$ is said to have the *conservative extension property* if $\mathcal{L}_{i,o,h} = \mathcal{L}_{i,o,h+1}$ for all i, o, and $h \geq \max(i, o)$. In other words, having the conservative extension property means that any query is independent of the height of intermediate data.

Fact 1 (Paredaens and Van Gucht [14], Wong [17]) *$\mathcal{NRC}$ has the conservative extension property. In particular, $\mathcal{NRC}$ queries on flat relations are exactly those expressible in the flat relational algebra.* □

Fact 2 (Libkin and Wong [12]) *$\mathcal{SQL}$ has the conservative extension property.* □

3 Bounded degree property and $\mathcal{NRC}$

In this section we first define two sample queries and show that in a language having at least the power of the relational algebra (first-order logic) they are at most as hard as deterministic transitive closure. Then we define the bounded degree property of a language and show how it implies a number of inexpressibility results in a uniform fashion. Finally we prove that this property holds in $\mathcal{NRC}$.

Definition 1

- *chain* : $\{s \times s\} \to \mathbb{B}$ *is a query that takes in a graph and returns* true *iff the graph is a chain, that is, a tree such that the out-degree of each node is at most 1.*
- *bbtree* : $\{s \times s\} \to \mathbb{B}$ *is a query that takes in a graph and returns* true *iff the graph is a balanced binary tree, that is, a binary tree in which all paths from the root to the leaves have the same length.*
- (see Immerman [10]) *dtc* : $\{s \times s\} \to \{s \times s\}$ *is the deterministic transitive closure. That is, if $G = \langle V, E\rangle$ is a digraph, then $dtc(G) = \langle V, E'\rangle$ where $(v_1, v_k) \in E'$ iff there is a path $(v_1, v_2) \in E, \ldots, (v_{k-1}, v_k) \in E$ such that v_{i+1} is a unique descendant of v_i, $i = 1, \ldots, k-1$.*

Proposition 1 *Let $\mathcal{L}$ be a language that has at least the power of the relational algebra. Then chain and bbtree are expressible in $\mathcal{L}(dtc)$*[1]*.* □

[1]Operations added to languages are listed explicitly in the brackets.

Corollary 1 *Let $\mathcal{L}$ be a language that has at least the power of the relational algebra. If chain is not expressible in $\mathcal{L}$, then none of the following is expressible in $\mathcal{L}$: dtc, transitive closure, tests for connectivity of directed and undirected graphs, test whether a graph is a tree, test for acyclicity.* $\Box$

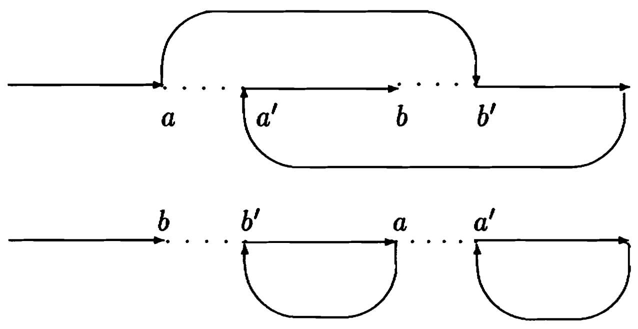
Let $G = \langle V, E\rangle$ be a graph. Define $\mathit{in\text{-}deg}(v) = \mathit{card}(\{v' \mid (v', v) \in E\})$ and $\mathit{out\text{-}deg}(v) = \mathit{card}(\{v' \mid (v, v') \in E\})$. The *degree set* of G, $\mathit{deg}(G)$, is defined as $\{\mathit{in\text{-}deg}(v) \mid v \in V\} \cup \{\mathit{out\text{-}deg}(v) \mid v \in V\} \subseteq \mathbb{N}$. One of the reasons why most recursive queries are not first-order definable is that they may take in a graph[2] whose degree set contains only small integers and may return a graph whose degree set is large. The definition below captures this intuition.

Definition 2 *Let $\mathcal{L}$ be a language. It is said to have the* bounded degree property *(at type s) if, for any $f : \{s \times s\} \to \{s \times s\}$ that is definable in $\mathcal{L}$ and for any number k there exists a number c, depending on f and k only, such that $\mathit{card}(\mathit{deg}(f(G))) \leq c$ for any graph G satisfying $\mathit{deg}(G) \subseteq \{0, 1, \ldots, k\}$.*

The bounded degree property can be used to prove various inexpressibility results.

Theorem 1 *Let $\mathcal{L}$ be a language that has at least the power of the relational algebra. Then, if $\mathcal{L}$ has the bounded degree property at type s, then neither chain : $\{s \times s\} \to \mathbf{B}$ nor bbtree : $\{s \times s\} \to \mathbf{B}$ is expressible in $\mathcal{L}$.*

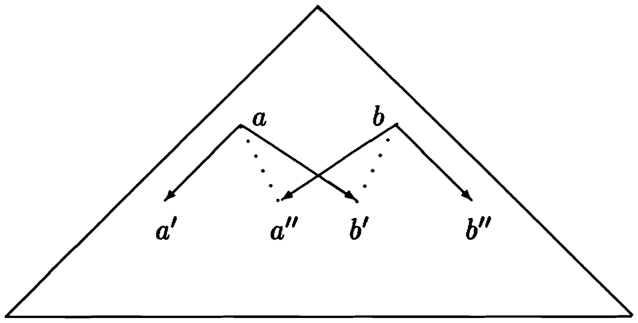
Proof. We offer a proof by picture. Assume *chain* is definable; then it is possible to define an expression that, when given a chain as an input, returns its transitive closure. As shown below, using chain it is possible to determine if a precedes b by re-arranging two edges and checking if the resulting graph is a chain:



But this contradicts the bounded degree property as we started with an n-node graph whose degree set is $\{0, 1\}$ and ended up with $\{0, 1, \ldots, n\}$.

If *bbtree* is definable, it is possible to determine if two nodes in a balanced binary tree are at the same level by re-arranging two edges as follows and checking if the result is still a balanced binary tree:

[2]We use graphs for the simplicity of exposition. Relational structures of arbitrary finite arity can be used.



Again, we started with an $n-$node graph whose degree set is $\{0, 1, 2\}$ and, making cliques of the nodes at the same level, ended up with a graph whose degree set has cardinality $\log_2(n + 1)$. □

The main reason we study this property is that it holds in $\mathcal{NRC}$.

Theorem 2 *$\mathcal{NRC}$ has the bounded degree property at base types.*

Proof sketch. Let $f : \{b \times b\} \rightarrow \{b \times b\}$ be an $\mathcal{NRC}$ expression where b is a base type. Then, by conservativity, f is equivalent to a relational algebra expression. Therefore, if $G' = f(G)$, then for some first-order expression F we have $\forall a \forall b E'(a, b) \leftrightarrow F(a, b, E)$. According to Gaifman [8], F is a Boolean combination of certain sentences and formulae with a, b as free variables in which all quantifiers are bounded to some neighborhoods of a and b. Moreover, the maximal radius of those neighborhoods, r, is determined by F. If $deg(G) \subseteq \{0, \ldots, k\}$, then it is possible to find the number of all neighborhoods of radius up to r, which depends only on F and k. Let c_0 be that number. Then, for all nodes a and b which have the same neighborhoods of radius up to r, we have $\forall c : E'(a, c) \leftrightarrow E'(b, c)$. Hence, there are at most 2^{c_0} elements in $deg(G')$. □

This settles the first part of conjecture 3.

Corollary 2

- *The flat relational algebra has the bounded degree property.*
- *chain, bbtree and other queries listed in corollary 1 are not expressible in $\mathcal{NRC}$.* □

4 Finite-cofiniteness and $\mathcal{SQL}$

To the best of our knowledge, there is no logic capturing the language $\mathcal{SQL}$, nor its flat fragment. The proof of the bounded degree property is based on Gaifman's result about local formulae [8]. That result was proved by quantifier elimination. This poses a problem if we try to prove the bounded degree property for flat types in $\mathcal{SQL}$.

In this section we use another technique to overcome this difficulty. It is well known that properties of cardinalities of finite models which can be tested in the first-order logic are either finite or co-finite.

For example, parity cannot be tested. Using conservativity, we present two results of the same kind for $\mathcal{SQL}$. First, properties of natural numbers expressible in $\mathcal{SQL}$ are either finite or co-finite. That answers conjecture 1. Second, for certain families of graphs the same finiteness-cofiniteness property holds. Then we derive inexpressibility of *chain* and *bbtree* from that. This answers conjecture 2 and the second part of conjecture 3, since $\mathcal{BQL}$ can be embedded in $\mathcal{SQL}$.

4.1 Expressive power of $\mathcal{SQL}$ at base types

Let $\mathcal{U}$ be a property of natural numbers, that is, $\mathcal{U} \subseteq \mathbf{N}$. By a test for $\mathcal{U}$ we mean a function $p : \mathbf{Q} \to \mathbf{B}$ such that for all $n \in \mathbf{N}$, $p(n)$ is *true* iff $n \in \mathcal{U}$.

Theorem 3 (see Libkin and Wong [11]) *Let $\mathcal{U} \subseteq \mathbf{N}$ be a property of natural numbers. Then membership test for $\mathcal{U}$ is definable in $\mathcal{SQL}$ iff $\mathcal{U}$ is either finite or co-finite.* □

A similar result can be shown for new base types, provided we do not have powerful functions on them. It is also based on the conservative extension property.

Corollary 3 *Let b be a base type with a countably infinite domain D. Assume that only equality test and a linear order $\leq^b$ isomorphic to ω are available for b. Then a test for $D' \subseteq D$ is definable in $\mathcal{SQL}(b, \leq^b)$ iff D' is either finite or co-finite.* □

4.2 Expressive power of $\mathcal{SQL}$ over flat relations

A binary relation $O : \{b \times b\}$ is called a *k-multi-cycle* if it is nonempty and is of the form

$$\left\{ \begin{array}{ccc} (o_1^1, o_2^1), (o_2^1, o_3^1), & \ldots, & (o_{h-1}^1, o_h^1), (o_h^1, o_1^1), \\ \vdots & & \vdots \\ (o_1^m, o_2^m), (o_2^m, o_3^m), & \ldots, & (o_{h-1}^m, o_h^m), (o_h^m, o_1^m) \end{array} \right\}$$

where $h \geq k$ and o_i^j are all distinct. That is, it is a graph containing $m \geq 1$ unconnected cycles of equal length $h \geq k$.

Define $\mathit{distance}_c(o, o', O)$ to be a predicate that holds iff the distance from node $\pi_1 o$ to node $\pi_2 o'$ in k-multi-cycle O is c. Note that $\mathit{distance}_c$ is definable in $\mathcal{SQL}$ for each constant c.

Define a *d-state* S with respect to variables $R : \{b \times b\}$, x_1, ..., x_m : $b \times b$ to be a conjunction of formulae of the form $\mathit{distance}_c(x_i, x_j, R)$ or the form $\neg\mathit{distance}_c(x_i, x_j, R)$. Moreover for each $0 \leq c \leq d$, $1 \leq i, j \leq m$, either $\mathit{distance}_c(x_i, x_j, R)$ or $\neg\mathit{distance}_c(x_i, x_j, R)$ must appear in it. Also S has to be satisfiable in the sense that some k-multi-cycle O and edges o_1, ..., o_m in O can be found so that $S[O/R, o_1/x_1, ..., o_m/x_m]$ holds.

Theorem 4 *Let $G : \{b \times b\} \to \mathbf{B}$ be a function expressible in $\mathcal{SQL}$. Then there is some k such that for all k-multi-cycles O, it is the case that $G(O)$ is true; or for all k-multi-cycles O, it is the case that $G(O)$ is false.*

$$\sum \left\{ \left| \cdots \sum \left\{ \left| \begin{array}{l} \textit{if } P_1 \\ \textit{then } f_1 \\ \vdots \\ \textit{else if } P_h \\ \textit{then } f_h \\ \textit{else } f_0 \end{array} \right| x_{m+1} \in R \right\} \right| \cdots \left| x_{m+n} \in R \right\} \right\}$$

Figure 2: Special form of $\mathcal{SQL}$ query

Before we present the proof of this theorem, let us observe that if we identify isomorphic k-multi-cycles, then for any $m \geq 1$, properties of k-multi-cycles consisting of at most m components are either finite or co-finite. In $\mathcal{SQL}(chain)$ it is possible to distinguish k-multi-cycles containing one cycle from those containing two. Therefore,

Corollary 4 *chain is not expressible in $\mathcal{SQL}$.* □

The proof of theorem 4 uses the following property of k-multi-cycles.

Proposition 2 *Let E be an expression of $\mathcal{SQL}$ having $R : \{b \times b\}$, $N : \mathbb{Q}$, x_1, ..., $x_m : b \times b$ as free variables such that E has the special form given in figure 2, where f_i's are ratios of polynomials in terms of N, P_i's are Boolean combinations of formulae of the form $\pi_i x_{i'} = \pi_j x_{j'}$, $\pi_i x_{i'} \neq \pi_j x_{j'}$, $\neg distance_c(x_i, x_j, R)$, or $distance_c(x_i, x_j, R)$. Let $d \geq n + m + C$ where C is the sum of the c's for each $distance_c(x_i, x_j, R)$ or $\neg distance(x_i, x_j, R)$ in the P_i's. Let S be any d-state with respect to R, x_1, ..., x_m. Then there is a number k and a ratio e of polynomials in terms of N such that for any k-multi-cycle O, and edges o_1, ..., o_m in O making $S[O/R, o_1/x_1, ..., o_m/x_m]$ true, it is the case that $E[O/R, o_1/x_1, ..., o_m/x_m, card(O)/N] = e[card(O)/N]$.*

Proof sketch. By the probability p for a predicate P of n free variables to hold with respect to a graph O, we mean the proportion of the instantiations of the free variables to edges in O that make P true. The key to the proof of this proposition is in realizing that the probability p_i for P_i to hold and $P_{j<i}$ to fail can be determined in the case of k-multi-cycle when k is large. (By convention, p_0 is the probability for every P_i to fail. That is, it is the probability for the last branch to be executed.) Moreover p_i can be expressed as a ratio of two polynomials of N. Thus e can be defined as $N^n \cdot (p_0 \cdot f_0 + \ldots + p_h \cdot f_h)$.

Each probability p_i can be calculated as follows. First, generate all possible d-states D_j's with respect to the variables R, x_1, ..., x_{m+n}. Second, determine the probability q_j of D_j given the certainty of S; this can be calculated using the procedure given in the next paragraph. Third, eliminate those D_j's that are inconsistent with the conjunction of S, P_i, and the negation of $P_{j<i}$. Finally, calculate p_i by summing the q_i's corresponding to those remaining d-states.

It remains to show that each q_i can be expressed as a ratio of two polynomials in N. Partition the positive leaves of the corresponding D_i into groups so that the variables in each group are connected between themselves and are unconnected with those in other groups. (Variables x and y are said to be connected in D_i if there is a positive leaf $distance_c(x, y, R)$ in D_i.) Note that the negative leaves merely assert that these groups are unconnected. Then we proceed by induction on the number of groups.

The base case is when we have just one group. In such a situation, all the variables lie on the same cycle. Then a lower bound on k can be determined from the group to force the variables to lie on a line. Let u is the number of free variables amongst x_{m+1}, ..., x_{m+n} appearing in the group; in this case $u = n$. Then $q_i = N \div N^u$ if no variables amongst x_1, ..., x_m appear in the group. Otherwise, $q_i = 1 \div N^u$. In either condition, q_i is a ratio of polynomials in N.

For the induction case, suppose we have more than one group. The independent probability of each group can be calculated as in the base case. Then q_i is the difference between the product of these independent probabilities and the sum of the probabilities where these groups are made to overlap in all possible ways. These groups are made to overlap by turning some negative leaves in D_i into positive ones. Notice that when groups overlap, the number of groups strictly decreases. Hence the induction hypothesis can be applied to obtain these probabilities as ratios of polynomials in N. Consequently, q_i can be expressed as a ratio of polynomials in N as desired. □

Now we return to the proof of theorem 4.

Proof sketch of theorem 4. Suppose $G : \{b \times b\} \to \mathbf{B}$ is implemented by the $\mathcal{SQL}$ expression $\lambda R.E$. Without loss of generality, E can be assumed to be a normal form with respect to the rewrite system used by us in the proof of the conservative extension theorem [12]. We note that such an E contains no subexpression of the form $\bigcup\{e_1 \mid x \in e_2\}$. Furthermore, all occurrences of summation in E must be of the form $\sum\{\!|e \mid x \in R|\!\}$.

Let us temporarily enrich our language with the usual logical operators $\vee$, $\wedge$, $\neg$, $\neq$, $\not\leq$, as well as $distance_c$. Also introduce a new variable $N : \mathbf{Q}$ which is to be interpreted as the cardinality of R. Rewrite all summations into the special form given in figure 2, so that each f_i has the form $h_i \div g_i$, where h_i is a polynomial in terms of N and g_i is either a polynomial in terms of N or is again a subexpression of the same special form. Also, the P_i's are formulae whose leaves are of the following form: $\pi_i x_{i'} = \pi_j x_{j'}$, $\pi_i x_{i'} \neq \pi_j x_{j'}$, $distance_c(x_i, x_j, R)$, $\neg distance_c(x_i, x_j, R)$, $U =^{\mathbf{Q}} V$, $U \neq^{\mathbf{Q}} V$, $U \leq V$, or $U \not\leq V$, where U and V also have the same special form.

Let the resultant expression be F. The rewriting should be such that for all sufficiently long k-multi-cycles O, $F[O/R, card(O)/N]$ holds iff $E[O/R]$ holds. This can be accomplished by using rules like:

- $\mathit{if}\ e_1\ \mathit{then}\ \sum\{\!|e_2 \mid x \in R|\!\}\ \mathit{else}\ e_3 \leadsto \sum\{\!|\ \mathit{if}\ e_1\ \mathit{then}\ e_2\ \mathit{else}\ e_3 \div N \mid x \in R|\!\}$
- $\mathit{if}\ e_1\ \mathit{then}\ e_2\ \mathit{else}\ \sum\{\!|e_3 \mid x \in R|\!\} \leadsto \sum\{\!|\ \mathit{if}\ e_1\ \mathit{then}\ e_2 \div N\ \mathit{else}\ e_3 \mid x \in R|\!\}$
- $e_1 \cdot \sum\{\!|e_2 \mid x \in R|\!\} \leadsto \sum\{\!|e_1 \cdot e_2 \mid x \in R|\!\}$
- $\sum\{\!|e_1 \mid x \in R|\!\} \cdot e_2 \leadsto \sum\{\!|e_1 \cdot e_2 \mid x \in R|\!\}$
- $\sum\{\!|e_1 \mid x \in R|\!\} \div e_2 \leadsto \sum\{\!|e_1 \div e_2 \mid x \in R|\!\}$
- $\sum\{\!|e_1 \mid x \in R|\!\} + e_2 \leadsto \sum\{\!|e_1 + (e_2 \div N) \mid x \in R|\!\}$

- $\sum\{\!|e_1 \mid x \in R|\!\} - e_2 \leadsto \sum\{\!|e_1 - (e_2 \div N) \mid x \in R|\!\}$
- $e_1 - \sum\{\!|e_2 \mid x \in R|\!\} \leadsto \sum\{\!|(e_1 \div N) - e_2 \mid x \in R|\!\}$
- $e_1 + \sum\{\!|e_2 \mid x \in R|\!\} \leadsto \sum\{\!|(e_1 \div N) + e_2 \mid x \in R|\!\}$

Having obtained F in this special form, we continue the proof using the following steps.

Step 1. If F is already in the form required by proposition 2, we can transform it according to proposition into a ratio of polynomials in terms of N (finding a lowerbound for k in the process). If F does not have the right form, proceed to the remaining steps.

Step 2. Look for an innermost subexpression of F that has the special form required by proposition 2. Let this subexpression be F' and its free variables be y_1, ..., y_m, R and N. Let the number of summation in F' be n. Generate all possible d-states (where d is the smallest one suggested by proposition 2) with respect to these free variables of F'. Let S_0, S_1, ..., S_h be these d-states, with $S_0 = \neg S_1 \wedge \cdots \wedge \neg S_h$. ($S_0$ is one of the d-states because $d < k$.) Apply proposition 2 to F' with respect to each S_i to obtain expressions e_i (finding a lower bound for k in the process). Then F' is equivalent to *if* S_1 *then* e_1 *else* ...*if* S_h *then* e_h *else* e_0. Note that each e_i is a ratio of polynomials of N.

Step 3. To maintain the same special form, we need to push the S_i up one level to the expression in which F' is nested. This is done using rules like:

- $(\textit{if } S_1 \textit{ then } e_1 \ \ldots \textit{if } S_h \textit{ then } e_h \textit{ else } e_0) =^{\mathbf{Q}} V \leadsto (S_0 \wedge e_0 = V) \vee \cdots \vee (S_h \wedge e_h = V)$
- $\textit{if } P \textit{ then } (f \div (\textit{if } S_1 \textit{ then } e_1 \textit{ else } \ldots \textit{if } S_h \textit{ then } e_h \textit{ else } e_0)) \textit{ else } e \leadsto \textit{if } P \wedge S_0 \textit{ then } f \div e_0 \ \ldots \textit{if } P \wedge S_h \textit{ then } f \div e_h \textit{ else } e$

Step 4. After step 3, some expression having the form $U =^{\mathbf{Q}} V$, $U \leq V$, or their negation can become an (in)equation of ratios of polynomials of N. Such an expression can be replaced either by *true* or by *false*. For illustration, we explain the case of $U =^{\mathbf{Q}} V$; the other cases are similar. First $U =^{\mathbf{Q}} V$ is readily transformed into a polynomial $P = 0$ with N being its only free variable. Check if P is identically 0. If this is the case, replace $U =^{\mathbf{Q}} V$ by *true*. If P is not identically 0, we use the fact that a polynomial has a finite number of roots. By choosing a sufficiently large lower bound for k, we can ensure that N always exceeds the largest root of P. Thus, in this case we replace $U =^{\mathbf{Q}} V$ by *false*.

Observe that in step 2 we have reduced the number of summations and in step 4 we have reduced the number of equality and inequality tests. By repeating these steps, we must eventually reach the base case and arrive at an expression where step 1 is applicable. When we are finished, the resultant expression is clearly a boolean formula containing no free variable. Therefore its value does not depend on R. Consequently the theorem holds for any k not smaller than the lower bound determined by the above process. □

The proof of theorem 4 relies on two things: satisfiability of d-states is easy to decide for k-multicycles and probabilities are easy to calculate and express as ratios of polynomials in terms of the size of graphs for k-multi-cycles. There is another class of graphs having these two properties: k-strict-binary-trees. A k-strict-binary-tree is a nonempty tree where each node has either 0 or 2 decendents and the distance from the root to any leaf is at least k.

Theorem 5 *Let $G : \{b \times b\} \to \mathbb{B}$ be a function expressible in $\mathcal{SQL}$. Then there is some k such that for all k-strict-binary-trees O, it is the case that $G(O)$ is true; or for all k-strict-binary-trees O, it is the case that $G(O)$ is false.*

Proof sketch. It is easy to decide if a d-state is satisfiable by some k-strict-binary-trees. The probability calculation is also simple. The only problem is that the probability must be expressed wholely as a ratio of polynomials of the number of edges in the tree. This is dealt with by observing that in k-strict-binary-trees, the number of internal nodes is 1 less half the number of edges and the number of leaves is equal to 2 plus the number of internal nodes. The theorem follows by repeating verbatim the proof for k-multi-cycles. □

So, if we identify isomorphic k-strict-binary-trees, then their properties recognizable in $\mathcal{SQL}$ are either finite or co-finite. In $\mathcal{SQL}(bbtree)$, for any $k > 0$, one can distingush a balanced binary tree of height k from any other k-strict-binary-tree. Therefore,

Corollary 5 *bbtree is not definable in $\mathcal{SQL}$.* □

In summary, we obtain

Corollary 6 *All the queries listed in corollary 1 are not expressible in $\mathcal{SQL}$.* □

4.3 Complete problems and $\mathcal{SQL}$

Some of the problems considered above are known to be complete for various complexity classes under first-order reductions. For example, the graph reachability problem is first-order complete for NLOGSPACE and its restriction to graphs with outdegree 1 is first-order complete for DLOGSPACE. Using the results of Immerman [10] on first-order completeness, the fact that $\mathcal{NRC}$ and $\mathcal{SQL}$ are closed under first-order reductions, and the inexpressiblity results proved in this paper, we get

Corollary 7 *Let P be a problem that is complete with respect to first-order reductions for one of the following classes: DLOGSPACE, Sym-LOGSPACE, NLOGSPACE, PTIME. Then P can not be solved by $\mathcal{SQL}$.* □

5 Conclusion and future work

We have considered the problem of analyzing the expressive power of nested relational and bag languages. We have shown that the conservativity property of these languages is a very powerful technique in analyzing their expressive power. We looked at the nested relational calculus $\mathcal{NRC}$ and presented a new technique for proving a number of inexpressibility results for it in a uniform way. We then looked at $\mathcal{SQL}$, which is obtained from $\mathcal{NRC}$ by adding aggregate functions, and proved a finite-cofiniteness property of some graph queries. This property ensures that our sample recursive queries remain inexpressible in $\mathcal{BQL}$, solving conjectures 1,2 and 3.

There are a few problems that we would like to work on. The most important is the following.

Conjecture 4 *$\mathcal{SQL}$ has the bounded degree property.*

Answering the following questions may shed some light on this conjecture.

1. What is a logic that captures (the first-order fragment of) $\mathcal{SQL}$?
2. Which logics have the bounded degree property? Observe that we used only a part of Gaifman's result to prove the bounded degree property for the first-order logic. Hence we believe there is a chance to find its generalizations for other logics.

It was shown in [11] that in order to fill the gap between set and bag languages with structural recursion one has to add a new primitive to the set language: $gen(n) = \{0, 1, \ldots, n\}$. With such a primitive, the bounded degree property does not hold, and the techniques for proving inexpressibility in $\mathcal{SQL}$ do not work. But we still believe that recursive queries like *chain* and *bbtree* are not definable. Proving this remains open.

Acknowledgments. We would like to thank Val Breazu-Tannen, Peter Buneman, Jan Paredaens, Dan Suciu, and Scott Weinstein for helpful discussions.

References

[1] S. Abiteboul and P. Kanellakis, Query languages for complex object databases, *SIGACT News* 21 (1990), 9–18.

[2] V. Breazu-Tannen, P. Buneman, and L. Wong, Naturally embedded query languages. In *LNCS 646: Proc. ICDT, Berlin, Germany, October, 1992*, pages 140–154. Springer-Verlag, October 92.

[3] A. Chandra and D. Harel, Structure and complexity of relational queries, *JCSS* 25 (1982), 99–128.

[4] S. Chaudhuri and M. Vardi, Optimization of *real* conjunctive queries, *Proceedings of the 12th Conference on Principles of Database Systems*, Washington DC, 1993, pages 59–70.

[5] L. Colby, A recursive algebra for nested relations, *Inform. Systems* 15 (1990), 567–582.

[6] R. Fagin, Finite model theory – a personal perspective, *TCS* 116 (1993), 3–31.

[7] R. Fagin, L. Stockmeyer and M. Vardi, On monadic NP vs. monadic co-NP, In *Proc. 8th IEEE Conf. on Structure in Complexity Theory*, May 1993, pages 19–30.

[8] H. Gaifman, On local and non-local properties, in: *Proceedings of the Herbrand Symposium*, Logic Colloquim '81, North Holland, 1982, pages 105–135.

[9] S. Grumbach, T. Milo, Towards tractable algebras for bags, *Proceedings of the 12th Conference on Principles of Database Systems*, Washington DC, 1993, pages 49–58.

[10] N. Immerman, Languages that capture complexity classes, *SIAM J. Comput.* 16 (1987), 760–778.

[11] L. Libkin and L. Wong, Some properties of query languages for bags, In *Proceedings of the 4th International Workshop on Database Programming Languages*, Springer Verlag, 1993, to appear.

[12] L. Libkin and L. Wong, Aggregate functions, conservative extension and linear orders, In *Proceedings of the 4th International Workshop on Database Programming Languages*, Springer Verlag, 1993, to appear.

[13] J. Paredaens, Private communication, Collection Types Workshop, February 1993.

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

[15] H.-J. Schek and M. Scholl, The relational model with relation-valued attributes, *Inform. Systems* 11 (1986), 137–147.

[16] S.J. Thomas and P. Fischer, Nested relational structures, in P. Kanellakis editor, *Advances in Computing Research: The Theory of Databases*, pages 269–307, JAI Press, 1986.

[17] L. Wong. Normal forms and conservative properties for query languages over collection types. In *PODS 93*, pages 26–36, Washington, D. C., May 1993.