# A Dichotomy in the Intensional Expressive Power of Nested Relational Calculi augmented with Aggregate Functions and a Powerset Operator

Limsoon Wong
School of Computing
National University of Singapore
13 Computing Drive, Singapore 117417
wongls@comp.nus.edu.sg

## ABSTRACT

The extensional aspect of expressive power—i.e., what queries can or cannot be expressed—has been the subject of many studies of query languages. Paradoxically, although efficiency is of primary concern in computer science, the intensional aspect of expressive power—i.e., what queries can or cannot be implemented efficiently—has been much neglected. Here, we discuss the intensional expressive power of $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ powerset)$, a nested relational calculus augmented with aggregate functions and a powerset operation. We show that queries on structures such as long chains, deep trees, etc. have a dichotomous behaviour: Either they are already expressible in the calculus without using the powerset operation or they require at least exponential space. This result generalizes in three significant ways several old dichotomy-like results, such as that of Suciu and Paredaens that the complex object algebra of Abiteboul and Beeri needs exponential space to implement the transitive closure of a long chain. Firstly, a more expressive query language—in particular, one that captures SQL—is considered here. Secondly, queries on a more general class of structures than a long chain are considered here. Lastly, our proof is more general and holds for all query languages exhibiting a certain normal form and possessing a locality property.

## Categories and Subject Descriptors

H.2.3 [**Languages**]: Query languages; F.4.1 [**Mathematical logic**]: Model theory

## Keywords

Intensional expressive power; normal form; conservative extension property; locality property; dichotomy; nested relational calculus; SQL.


*PODS'13,* June 22–27, 2013, New York, New York, USA.


## 1. INTRODUCTION

A function $f$ that is expressible[1] in a query language can be implemented or executed in many different ways, each corresponding to a different algorithm. These algorithms may have different complexity. Some of these algorithms for $f$ may be expressible in the given query language, while some of them may not be expressible in it. Even though efficiency is a key issue in computer science, we seldom see results that study the power of query languages from this "intensional" perspective.

This lack of attention may be attributable to the logical vs physical separation, where a database system may use different execution plans for the same query depending on optimization factors such as what indices are available. Nevertheless, the syntax of a query naturally suggests an (unoptimized and possibly naive) implementation. Thus, as argued by Suciu, Paredeans, and Wong [14, 15], there is a so-called "natural operational semantics" for a query language; and intensional expressive power can be studied with respect to it. Yet it is significantly more difficult to study intensional expressive power and, in particular, there is a very limited repertoire of general techniques that can be applied.

Some of the papers that are in the spirit of intensional expressive power, with respect to the respective natural operational semantics of various query languages, include the followings:

- Colson [7] shows that while the function which computes the minimum of two integers in unary representation can be expressed using primitive recursion, all such expressions have higher than $O(min(m,n))$ complexity.
- Abiteboul and Vianu [2] show that while the parity query can be expressed by a generic machine, all such expressions have higher than PTIME complexity.
- Suciu and Wong [15] show that while sequential iteration queries (called *sri* queries in their paper) can be uniformly translated into data-parallel iteration queries (called *sru* queries in their paper), all such uniform translations must map some PTIME queries into exponential space ones.

[1]Throughout this paper, when we use the term "expressive power" without explicitly indicating whether it is the extensional aspect or the intensional aspect, we mean the extensional aspect.

- Suciu and Paredaens [14] show that while the transitive closure of a long chain can be expressed in the complex object algebra of Abiteboul and Beeri, all such expressions must use exponential space.

- Van den Bussche [6] shows that, when retricted to unary database schemas (i.e., when the input relations all have exactly one column), every query in the complex object algebra of Abiteboul and Beeri is either already expressible without using the powerset operator or must use exponential space.

- Biskup et al. [3] show that while the parity query can be expressed in what they called the equation algebra, all such expressions must use exponential space.

These are impressive results and their proofs are tour de force undertakings. Unfortunately, they are also query specific; and the proofs are complex and not easily "portable" to other queries. Therefore, more light ought to be shed on the structure of the query languages concerned or the structure of inefficient queries in these query languages that make the cause of the inefficiency clear.

In this paper, we study the intensional expressive power of $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ powerset)$, a nested relational calculus endowed with aggregate functions and a powerset operation. This calculus can express all the usual SQL queries such as group-by, count, average, etc. So it is a query language that is considerably more expressive than the complex object algebra of Abiteboul and Beeri [1]. Moreover, we study the intensional expressive power of $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ powerset)$ in a non-query-specific setting.

Our main result is a Dichotomy Theorem that, all queries on a general class of structures—which includes deep trees, long chains, etc.—are either already expressible in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ powerset)$ without using its powerset operation or must use an exponential amount of space. Since $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ powerset)$ is more general than the complex object algebra of Abiteboul and Beeri and the class of structures includes long chains, this result is a powerful generalization of the old result of Suciu and Paredaens [14] and also that of Van den Bussche [6]. Furthermore, our proof of this Dichotomy Theorem factors through a locality property [9] and the normal form induced by the conservative extension property [16]. Thus it holds also for any query language exhibiting a similar normal form and the locality property, offering a high potential for generalization to many other query languages. We demonstrate this last aspect on the equation algebra of Biskup and colleagues [3].

The organization of this paper is as follows. Section 2 presents the query language $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ powerset)$. Section 3 presents the conservative extension property, its associated rewrite rules, and the locality property. Section 4 proves our main result, the Dichotomy Theorem. Section 5 concludes the paper with an extensive discussion on how the Dichotomy Theorem generalizes the results of Suciu and Paredeans [14], Van den Bussche [6], and Biskup and colleagues [3].

## 2. $\mathcal{NRC}$, AGGREGATES, AND POWERSET

The ambient language for our study is $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ powerset)$. This query language is built upon the nested relational calculus $\mathcal{NRC}$ from Buneman et al. [5], by augmenting it with arithmetic operations, a summation operation, and a powerset operation. The base language $\mathcal{NRC}$ is equivalent to the usual nested relational algebra [5]. The extension of $\mathcal{NRC}$ by arithmetic operations and the summation operation makes it into $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$, which is able to express group-by and aggregate functions, and captures the expressive power of SQL [13].

The types and expressions in $\mathcal{NRC}$ are given in Figure 1. The type superscripts in the figure are conventionally omitted because they can be inferred. The semantics of a type is just a set of complex objects—i.e., a set of objects built up by nesting sets and records of base type objects:

- There are some base types $b$, as well as the usual base types *bool* (i.e., Booleans) and $Q$ (i.e., numbers).

- An object of type $s_1 \times \cdots \times s_n$ is a tuple (i.e., a record) whose $i$th component is an object of type $s_i$, for $1 \leq i \leq n$.

- An object of type $\{s\}$ is a finite set whose elements are objects of type $s$. An object of type $\{s\}$ is called a "relation". Moreover, if $s = b \times \cdots \times b$, then an object of type $\{s\}$ (or $s$) is called a "flat relation". However, if $s$ contains some set brackets, then an object of type $\{s\}$ is called a "nested relation". More generally, a type $s$ containing $n$ levels of nested set brackets is said to be of height $n$; e.g., $b \times b$ has height 0, $\{b \times b\}$ has height 1, and $\{b \times \{b\}\}$ has height 2.

The meaning of the expression constructs are described below:

- The expression $c$ denotes constants of a base type, i.e., the atomic objects.

- The expressions *true*, *false*, and *if* $e_1$ *then* $e_2$ *else* $e_3$ have their usual meaning.

- The expression $(e_1, \ldots, e_n)$ forms a tuple whose $i$th component is the object denoted by $e_i$, for $1 \leq i \leq n$.

- The expression $\pi_i\ e$ extracts the $i$th component of the tuple $e$.

- The expressions $\{\}$, $\{e\}$, and $e_1 \cup e_2$ have their usual meaning as set operations.

- The expression $\bigcup\{e_1 \mid x \in e_2\}$ forms the set obtained by first applying the function $f(x) = e_1$ to each object in the set $e_2$ and then taking their union. That is, $\bigcup\{e_1 \mid x \in e_2\} = f(C_1) \cup \ldots \cup f(C_n)$, where $f(x) = e_1$ and $\{C_1, \ldots, C_n\}$ is the set denoted by $e_2$. This construct is the sole means in $\mathcal{NRC}$ for iterating over a set.

- The expression $e_1 = e_2$ denotes an equality test between $e_1$ and $e_2$. Here, $e_1$ and $e_2$ denote objects of the same base type. With this construct, it is straightforward to define in $\mathcal{NRC}$ equality tests for all types.

- The expression *isempty* $e$ tests whether a set $e$ is empty. Here, $e$ denote a set of tuples of objects of base types. With this construct, it is straightforward to define in $\mathcal{NRC}$ emptiness tests for all set types.

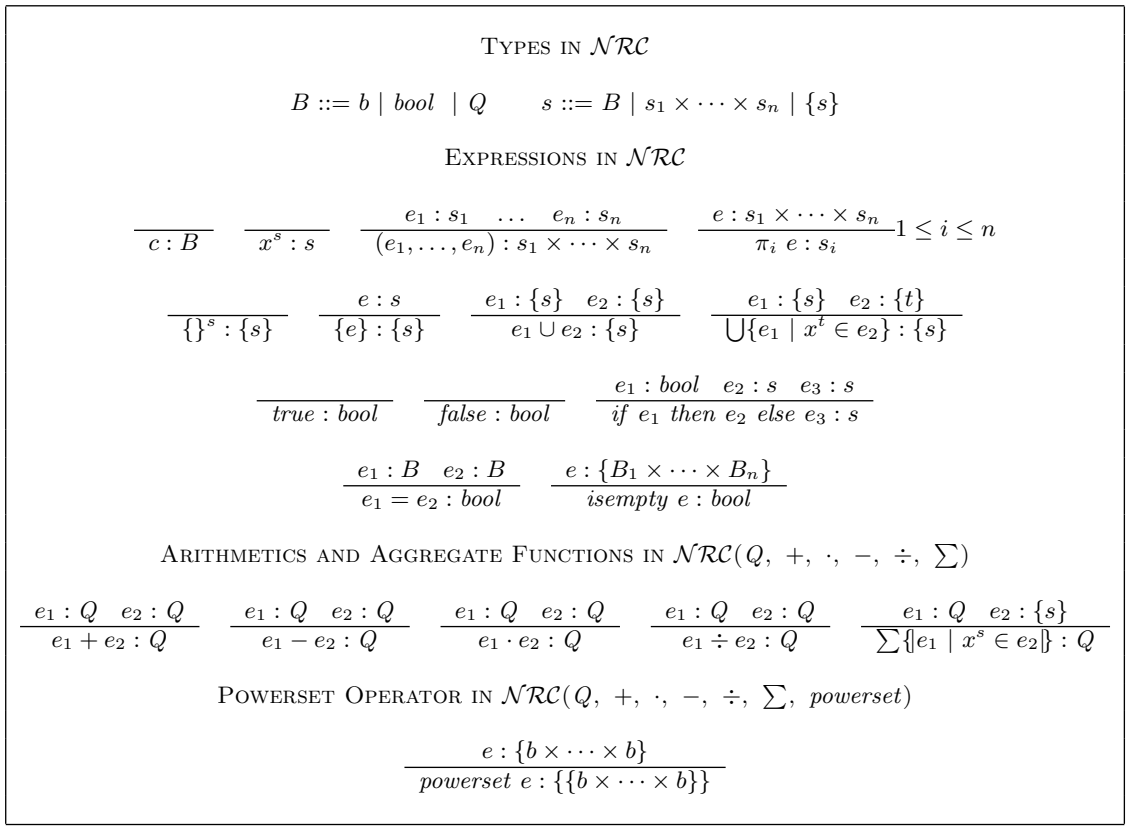


**Figure 1: $\mathcal{NRC}$ and its extensions $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$ and $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ powerset)$.**

There is a straightforward translation between this syntax and the comprehension syntax [4] of the form $\{e \mid \delta_1, \ldots, \delta_n\}$ where each $\delta_i$ either has the form $x_i \in e_i$ or the form $e_i$. The translation is given by:

- $\{e \mid x_1 \in e_1, \Delta\} =_{df} \bigcup\{\{e \mid \Delta\} \mid x_1 \in e_1\}$;
- $\{e \mid e_1, \Delta\} =_{df} if\ e_1\ then\ \{e \mid \Delta\}\ else\ \{\}$; and
- $\{e \mid \} =_{df} \{e\}$.

We use the comprehension syntax to write examples, but the reader should understand these examples as syntactic sugars of the actual $\mathcal{NRC}$ expressions.

*Example 1.* Let $X : \{employee \times salary \times dept\}$ be a relation that records the annual salary of employees in a company. Let $Y : \{employee \times \{employee\}\}$ be a relation that records the set of immediate reportees of each manager.

- The managers in the company can be expressed in $\mathcal{NRC}$ as $\Pi_1(Y) =_{df} \{\pi_1\ y \mid y \in Y\}$.
- A flat version of the manager-reportee relation can be expressed as $unnest(Y) =_{df} \{(\pi_1\ y,\ x) \mid y \in Y, x \in \pi_2\ y\}$.
- The annual salary of managers in the company can be expressed as $join(X, Y) =_{df} \{(y, \pi_2\ x) \mid y \in \Pi_1(Y), x \in X, y = \pi_1\ x\}$.
- A nested version of the salary relation where employees are grouped by department is $nest(X) =_{df} \{(\pi_3\ x, \{(\pi_1\ y, \pi_2\ y) \mid y \in X, \pi_3\ y = \pi_3\ x\}) \mid x \in X\}$.

While all the operations of the usual nested relational algebra can be expressed in $\mathcal{NRC}$ [5], aggregate functions usually encountered in SQL queries are not expressible in $\mathcal{NRC}$. So we augment $\mathcal{NRC}$ with the usual arithmetic operations $+$, $-$, $\cdot$ and $\div$ and a summation construct $\sum\{\!|e_1 \mid x \in e_2|\!\}$ to give the query language $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$. The summation construct $\sum\{\!|e_1 \mid x \in e_2|\!\}$ first applies the function $f(x) = e_1$ to each object in the set $e_2$ and sums the resulting numbers; that is, $\sum\{\!|e_1 \mid x \in e_2|\!\} = f(C_1) + \cdots + f(C_n)$, where $f(x) = e_1$ and $\{C_1, \ldots, C_n\}$ is the set denoted by $e_2$. Here are some illustrative examples showing how this construct captures the usual group-by and aggregate functions in SQL queries.

*Example 2.* As before, let $X : \{employee \times salary \times dept\}$ be a relation that records the annual salary of employees in a company.

- The number of employees in the company can be expressed in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$ as $count(X) =_{df} \sum\{\!|1 \mid x \in X|\!\}$.
- The annual EOM budget of the company can be expressed as $sum(X) =_{df} \sum\{\!|\pi_2\ x \mid x \in X|\!\}$.
- The mean annual salary of employees of the company can be expressed as $ave(X) =_{df} sum(X) \div count(X)$.
- Finally, the mean annual salary of each department can be produced by $dept\_ave(X) =_{df} \{(\pi_1\ x, ave(\pi_2\ x)) \mid x \in nest(X)\}$.

Let us use the notation $e[\vec{R}]$ to mean the an expression $e$ with free variables $\vec{R}$. However, when it is not important to explicitly list the free variables, we write it simply as $e$. For a list of objects $\vec{O}$ that conform to the types of $\vec{R}$, we use the notation $e[\vec{O}/\vec{R}]$ for the expression obtained by substituting $\vec{O}$ for $\vec{R}$. We think of the expression $e[\vec{R}]$ as a "query" where $\vec{R}$ are its input; equivalently, we can think of it as a function $f(\vec{R}) = e[\vec{R}]$. The expression $e[\vec{R}]$ is said to be a "flat relational query" if each $R$ in $\vec{R}$ is a flat relation and $e[\vec{R}] : \{b \times \cdots \times b\}$. Recall that a flat relation can have type $\{b \times \cdots \times b\}$ or type $b \times \cdots \times b$. So, we use the notation $e[\vec{R}, \vec{x}]$ when it is important to explicitly separate the two kinds of variables in a flat relational query.

Below are some well-known results [5, 16, 13, 8, 10].

PROPOSITION 1. *1. $\mathcal{NRC}$ is in PTIME.*

*2. $\mathcal{NRC}$ is equivalent to the classical nested relational algebra.*

*3. $\mathcal{NRC}$ restricted to flat relational queries is equivalent to the classical relational algebra.*

*4. $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum)$ is in PTIME.*

*5. $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum)$ restricted to flat relational queries is equivalent to the aggregate logic $L_{aggr}(Q, +, \cdot, -, \div, \sum)$.*

*6. $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum)$ cannot express recursive queries such as transitive closure.*

Due to the equivalence of $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum)$ and $L_{aggr}(Q, +, \cdot, -, \div, \sum)$ on flat relational queries, every flat relational query $e[\vec{R}, \vec{x}]$ in $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum)$ can be translated to a logic formula $\varphi_e^{\vec{R}}(\vec{x}, y)$ in $L_{aggr}(Q, +, \cdot, -, \div, \sum)$. So, for any objects $\vec{O}$, $\vec{o}$, $\{o'\}$ conforming to the types of $\vec{R}$, $\vec{x}$, and $e$, it is the case that $o' \in e[\vec{O}/\vec{R}, \vec{o}/\vec{x}]$ if and only if $[\vec{O}/\vec{R}, \vec{o}/\vec{x}, o'/y] \models \varphi_e^{\vec{R}}(\vec{x}, y)$.

The formula $\varphi_e^{\vec{R}}(\vec{x}, y)$ in $L_{aggr}(Q, +, \cdot, -, \div, \sum)$ actually only has variables on atomic objects. For ease of understanding and cross-referencing:

- We use the convention of treating each variable $x_i$ in $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum)$ as a tuple of variables $x_{i,1}$, ..., $x_{i,n}$ in $L_{aggr}(Q, +, \cdot, -, \div, \sum)$, where each $x_{i,j}$ is to take on the value of the $j$th component of the value taken on by $x_i$.
- We adopt the convention of using $R_1$, $R_2$, ... to name the input relations in both $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum)$ and $L_{aggr}(Q, +, \cdot, -, \div, \sum)$.
- We use $x_1$, $x_2$, ... to name the free variables in a query in $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum)$, which represent the input variables of the query.
- We use $y$ (actually, $y_{,1}$, $y_{,2}$, ...) to name the free variables in a translated logic formula $\varphi_e^{\vec{R}}(\vec{x}, y)$ that corresponds to the outputs of the original query $e[\vec{R}, \vec{x}]$.
- We write $\varphi_e$ and even $\varphi$ instead of $\varphi_e^{\vec{R}}(\vec{x}, y)$ when there is no confusion.

As noted in Proposition 1, recursive queries such as transitive closure are inexpressible in $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum)$. In an influential paper [1], Abiteboul and Beeri suggest augmenting $\mathcal{NRC}$ with a powerset operation, resulting in the query language $\mathcal{NRC}(powerset)$, to enable such queries to be expressed without resorting to explicit recursion. So we also augment $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum)$ with a powerset operation on flat relations to obtain the query language $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum, powerset)$, as shown in Figure 1. Here, *powerset* $e$ produces a set containing all the subsets of the set denoted by $e$, provided $e$ is a flat relation. $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum, powerset)$ can express recursive queries such as transitive closure, as illustrated in the following example.

*Example 3.* Let $Y : \{employee \times employee\}$ be a relation that the direct reportees of each employee. The direct and indirect reportees of each employee can be expressed in $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum, powerset)$ as

$$tc(Y) =_{df} \bigcap\{X \mid \begin{array}{l} X \in powerset\ cp(dom(Y)), \\ subset(Y, X),\ closed(X)\}, \end{array}$$

where

- $dom(Y) =_{df} \{\pi_1\ y \mid y \in Y\} \cup \{\pi_2\ y \mid y \in Y\}$,
- $cp(Z) =_{df} \{(u, v) \mid u \in Z, v \in Z\}$,
- $subset(Y, X) =_{df} isempty\ \{y \mid y \in Y,\ isempty\ \{y \mid x \in X, y = x\}\}$,
- $closed(Z) =_{df} subset(\{\pi_1\ u,\ \pi_2\ v) \mid u \in Z,\ v \in Z,\ \pi_2\ u = \pi_1\ v\},\ Z)$, and
- $\bigcap(Z) = \{u \mid u \in Z,\ subset(\{subset(u, v) \mid v \in Z\}, \{true\})\}$.

As our interest is in the intensional aspect of expressive power, we need to know how each expression in $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum, powerset)$ is executed. We specify this explicitly in Figure 2, as a call-by-value operational semantics. A call-by-value operational semantics is widely adopted in programming languages and has also been used for several variations of $\mathcal{NRC}$ in earlier works [14, 15] on intensional expressive power.

In Figure 2, the notation $e \Downarrow C$ means the closed expression $e$ is evaluated to the object $C$. The notation $C_1 \cup \cdots \cup C_n$ means the set of objects obtained by the union of the sets $C_1, \ldots, C_n$. The notations $C_1 + C_2$, $C_1 - C_2$, $C_1 \cdot C_2$, $C_1 \div C_2$ mean the objects obtained as the sum, difference, product, and division of $C_1$ and $C_2$ respectively. This evaluation is sound in the sense that, when $e : s$ and $e \Downarrow C$, then $C$ is an object of type $s$ and $e = C$. Hence each $e : s$ evaluates to a unique $C$. We use the notation $e \Downarrow$ to refer to the unique evaluation tree of $e$.

Here, we do not define the space complexity $sizeof(e \Downarrow)$ of an evaluation in terms of the size of the evaluation tree. Instead, we define it in terms of the size of the largest object in the evaluation tree—viz., $sizeof(e \Downarrow) = \max\{sizeof(C) \mid$ the object $C$ occurs in the evaluation tree $e \Downarrow\}$. The size of an object is the number of atomic objects (i.e., objects of base type $b$) that it contains. This way of defining the complexity of evaluation has also been used in earlier works on intensional expressive power [14, 15].

Analogously, we define the time complexity $timeof(e \Downarrow)$ of an evaluation in terms of time complexity of the largest node

$$\frac{}{c \Downarrow c} \qquad \frac{e_1 \Downarrow C_1 \quad \ldots \quad e_n \Downarrow C_n}{(e_1, \ldots, e_n) \Downarrow (C_1, \ldots, C_n)} \qquad \frac{e \Downarrow (C_1, \ldots, C_n)}{\pi_i\ e \Downarrow C_i} 1 \leq i \leq n$$

$$\frac{}{\{\} \Downarrow \{\}} \qquad \frac{e \Downarrow C}{\{e\} \Downarrow \{C\}} \qquad \frac{e_1 \Downarrow C_1 \quad e_2 \Downarrow C_2}{e_1 \cup e_2 \Downarrow C_1 \cup C_2}$$

$$\frac{e_2 \Downarrow \{C_1, \ldots, C_n\} \quad e_1[C_1/x] \Downarrow C_1' \quad \cdots \quad e_1[C_n/x] \Downarrow C_n'}{\bigcup\{e_1 \mid x \in e_2\} \Downarrow C_1' \cup \cdots \cup C_n'}$$

$$\frac{}{\mathit{true} \Downarrow \mathit{true}} \qquad \frac{}{\mathit{false} \Downarrow \mathit{false}} \qquad \frac{e_1 \Downarrow \mathit{true} \quad e_2 \Downarrow C}{\mathit{if}\ e_1\ \mathit{then}\ e_2\ \mathit{else}\ e_3 \Downarrow C} \qquad \frac{e_1 \Downarrow \mathit{false} \quad e_3 \Downarrow C}{\mathit{if}\ e_1\ \mathit{then}\ e_2\ \mathit{else}\ e_3 \Downarrow C}$$

$$\frac{e_1 \Downarrow C_1 \quad e_2 \Downarrow C_2}{e_1 = e_2 \Downarrow \mathit{true}} C_1 = C_2 \qquad \frac{e_1 \Downarrow C_1 \quad e_2 \Downarrow C_2}{e_1 = e_2 \Downarrow \mathit{false}} C_1 \neq C_2$$

$$\frac{e \Downarrow C}{\mathit{isempty}\ e \Downarrow \mathit{true}} C = \{\} \qquad \frac{e \Downarrow C}{\mathit{isempty}\ e \Downarrow \mathit{false}} C \neq \{\}$$

$$\frac{e_1 \Downarrow C_1 \quad e_2 \Downarrow C_2}{e_1 + e_2 \Downarrow C_1 + C_2} \qquad \frac{e_1 \Downarrow C_1 \quad e_2 \Downarrow C_2}{e_1 - e_2 \Downarrow C_1 - C_2} \qquad \frac{e_1 \Downarrow C_1 \quad e_2 \Downarrow C_2}{e_1 \cdot e_2 \Downarrow C_1 \cdot C_2} \qquad \frac{e_1 \Downarrow C_1 \quad e_2 \Downarrow C_2}{e_1 \div e_2 \Downarrow C_1 \div C_2}$$

$$\frac{e_2 \Downarrow \{C_1, \ldots, C_n\} \quad e_1[C_1/x] \Downarrow C_1' \quad \cdots \quad e_1[C_n/x] \Downarrow C_n'}{\sum\{\!|e_1 \mid x \in e_2|\!\} \Downarrow C_1' + \cdots + C_n'}$$

$$\frac{e \Downarrow \{C_1, \ldots, C_n\}}{\mathit{powerset}\ e \Downarrow \{C_1', \ldots, C_{2^n}'\}}$$
where $C_1', \ldots, C_{2^n}'$ are the subsets of $\{C_1, \ldots, C_n\}$

**Figure 2: A call-by-value operational semantics of $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ \mathit{powerset})$.**

in the evaluation tree—viz., $\mathit{timeof}(e \Downarrow) = \max\{\mathit{timeof}(e' \Downarrow C') \mid$ the node $e' \Downarrow C'$ occurs in the evaluation tree of $e \Downarrow \}$. Here, the time complexity $\mathit{timeof}(e' \Downarrow C')$ of a node is defined in general as the number of branches that the node has. For example, in Figure 2, $\mathit{timeof}(e_1 + e_2 \Downarrow C_1 + C_2) = 2$ and $\mathit{timeof}(\bigcup\{e_1 \mid x \in e_2\} \Downarrow C_1 \cup \cdots \cup C_n') = n+1$. However, for primitive operations, we assign them their expected time complexity. For example, $\mathit{timeof}(\mathit{powerset}\ e \Downarrow C) = 2^n$, where $n$ is the cardinality of $e$.

The deep result below is due to Suciu and Paredaens [14].

Proposition 2. *Let $e[R]$ be a query that implements the transitive closure of an input flat relation $R : \{b \times b\}$ in $\mathcal{NRC}(\mathit{powerset})$. Let $O$ be a sufficiently long chain of type $\{b \times b\}$. Then $\mathit{sizeof}(e[O/R] \Downarrow)$ is $\Omega(2^{|O|})$. Thus every implementation of transitive closure in $\mathcal{NRC}(\mathit{powerset})$ requires exponential space.*

One may say that, since transitive closure is inexpressible in $\mathcal{NRC}$, any implementation of it in $\mathcal{NRC}(\mathit{powerset})$ must use the powerset operation and therefore must take exponential space. This is naive because there may exist a clever implementation that uses the powerset operation only on small intermediate data and, thus, achieves an overall PTIME complexity. Proposition 2 rules this out.

The proof of this result by Suciu and Paredaens also contains an implicit proof of another result, which Van den Bussche [6] proves explicitly in a later paper, that queries in $\mathcal{NRC}(\mathit{powerset})$ on unary database schemas are either already expressible in $\mathcal{NRC}$ (i.e., do not use the powerset operation) or must use exponential space (i.e., must use the powerset operation in a non-trivial way). It is known from Biskup et al. [3] that the parity query, which tests whether an input set has even cardinality, must use exponential space in a so-called equation algebra. The equation algebra is equivalent to $\mathcal{NRC}(\mathit{powerset})$, and every expression in $\mathcal{NRC}(\mathit{powerset})$ can be translated to an expression in the equation algebra having the same space complexity [3]. Thus every query can be implemented in the equation algebra with equal or better space complexity than in $\mathcal{NRC}(\mathit{powerset})$. Combining these two results of Van den Bussche and Biskup et al., we have the following result.

Proposition 3. *1. Every flat relational query on unary database schemas in $\mathcal{NRC}(\mathit{powerset})$ is either already expressible in $\mathcal{NRC}$ or requires exponential space.*

*2. In particular, every implementation of the parity query in $\mathcal{NRC}(\mathit{powerset})$ requires exponential space.*

In the rest of this paper, we aim to generalize these two results to a Dichotomy Theorem on a large class of flat relational queries expressible in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ \mathit{powerset})$. In particular, these flat relational queries expressible in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ \mathit{powerset})$ are shown here to be dichotomous in the sense that either they are already expressible in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$ or they require at least exponential space. Hence, the extra expressive power that the powerset operation buys for $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ \mathit{powerset})$ comes strictly with an exponential cost.

# 3. TWO POWERFUL PROPERTIES

We need the conservative extension property and the locality property to prove our main results.

## 3.1 Conservative Extension

The conservative extension property and its system of rewrite rules, in Figure 3, were introduced by Wong [16] and generalized by Libkin and Wong [12, 13]. The last rule deserves a special mention. Consider the incorrect equation: $\sum\{\!|e \mid x \in \bigcup\{e_1 \mid y \in e_2\}|\!\} = \sum\{\!|\sum\{\!|e \mid x \in e_1|\!\} \mid y \in e_2|\!\}$. Suppose $e_2$ evaluates to a set of two distinct objects $\{o_1, o_2\}$; $e_1[o_1/y]$ and $e_1[o_2/y]$ both evaluate to $\{o_3\}$; and $e[o_3/x]$ evaluates to 1. Then the left-hand-side of the "equation" returns 1 but the right-hand-side yields 2. The division operation in the last rule in Figure 3 is used to handle duplicates properly. The following properties of this system of rewrite rules are well known [16, 12, 13].

PROPOSITION 4 (CONSERVATIVE EXTENSION).

1. *This system of rewrite rules is sound.*
2. *This system of rewrite rules is strongly normalizing.*
3. *Let $e$ be an expression in $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum,$ powerset) that is in normal form with respect to this system of rewrite rules. That is, no rule can be applied to further rewrite $e$. Let $e'[\vec{R}] : s$ be a subexpression in $e$. Suppose $\vec{R}$ have types whose height is at most $h$, and the type $s$ has height $h'$. Then all the types appearing in the type derivation of $e'[\vec{R}] : s$ have height at most $\max(h, h')$, if the powerset operation does not appear in $e'[\vec{R}]$; or, they have height at most $\max(h, h', 2)$, if the powerset operation appears in $e'[\vec{R}]$.*

This system of rewrite rules does not increase the complexity of evaluation.

PROPOSITION 5. *Let $e[\vec{R}] \mapsto e'[\vec{R}]$. Let $\vec{O}$ be a list of objects conforming to the types of $\vec{R}$. Then*

1. *sizeof$(e[\vec{O}/\vec{R}] \Downarrow) \geq$ sizeof$(e'[\vec{O}/\vec{R}] \Downarrow)$.*
2. *timeof$(e[\vec{O}/\vec{R}] \Downarrow) \geq$ timeof$(e'[\vec{O}/\vec{R}] \Downarrow)$.*

## 3.2 Locality

The second powerful machinery needed to prove our dichotomy result is the locality property. We need some understanding of "$\tau$ structure", "Gaifman graph", "$r$-sphere", and "$r$-neighbourhood", before we can explain the locality property [11].

A signature $\tau$ is a list of symbols $\vec{R}$, where $\vec{R}$ is to be regarded as input for a query. Each symbol $R_i$ in $\vec{R}$ has type of the form $\{b \times \cdots \times b\}$. A $\tau$ structure $\mathcal{A} = \langle A, \vec{O}\rangle$ has a universe $A$ (which is a finite nonempty set of objects of type $b$) and a list of objects $\vec{O}$ (and each object $O_i$ in $\vec{O}$ has the type of $R_i$ and is the interpretation of $R_i$). Each $O_i$ in $\vec{O}$ is a set of tuples, and each element $o_{i,j}$ in a tuple $o_i \in O_i$ is in the universe $A$. Each object in the universe is required to be in some $O_i$. We use STRUCT$[\tau]$ to denote the class of $\tau$ structures, and the symbol $\simeq$ to denote the isomorphism of $\tau$ structures. Also, we use the symbol $\tau_m$ to denote the signature obtained by extending the signature $\tau$ with $m$ new constant symbols.

The Gaifman graph $\mathcal{G}(\mathcal{A})$ of a $\tau$ structure $\mathcal{A} = \langle A, \vec{O}\rangle$ is a graph whose edges are pairs $(a, b)$ where there is a tuple $o_i \in O_i$, for some $O_i$ in $\vec{O}$, such that both $a$ and $b$ are elements in $o_i$. The distance $d^{\mathcal{A}}(a, b)$ is the length of the shortest path from $a$ to $b$ in $\mathcal{G}(\mathcal{A})$. Given a tuple $\vec{a} = (a_1, \ldots, a_m)$ of objects in $A$, and some $r \geq 0$, the $r$-sphere of $\vec{a}$ is defined as $S_r^{\mathcal{A}}(\vec{a}) = \bigcup_{1 \leq i \leq m} S_r^{\mathcal{A}}(a_i)$, where $S_r^{\mathcal{A}}(a_i) = \{b \in A \mid d^{\mathcal{A}}(a_i, b) \leq r\}$. The $r$-neighbourhood of $\vec{a}$ is the $\tau_m$ structure $N_r^{\mathcal{A}}(\vec{a}) = \langle S_r^{\mathcal{A}}(\vec{a}), \vec{O}|_{S_r^{\mathcal{A}}(\vec{a})}, a_1, \ldots, a_m\rangle$, meaning that $N_r^{\mathcal{A}}(\vec{a})$ is obtained by restricting $\mathcal{A}$ to the universe $S_r^{\mathcal{A}}(\vec{a})$ and adding some extra constants which are the elements of $\vec{a}$.

As shown by Gaifman [9], the result of any first-order query can be determined by considering "small neighbourhoods" of its input—the locality property. As shown by Dong et al. [8], flat relational queries expressible in $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum)$ also enjoy this kind of locality property, modulo a mild restriction on input relations. It is also shown by Hella et al. [10] and Libkin [11] that flat relational queries in $L_{aggr}(Q, +, \cdot, -, \div, \sum)$ enjoy the locality property even when it is further augmented with any collection $\Omega$ of functions on numbers[2] and any collection $\Theta$ of aggregate functions.[3] Since $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum)$ and $L_{aggr}(Q, +, \cdot, -, \div, \sum)$ have equivalent expressive power in terms of flat relational queries, see Proposition 1, this means that flat relational queries in $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum)$ actually enjoy the locality property without any restriction, even when the query language is further augmented with any collection $\Omega$ of functions on numbers and any collection $\Theta$ of aggregate functions.

PROPOSITION 6 (LOCALITY). *Every flat relational query $e[\vec{R}]$ in $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum)$ has the locality property. That is, there is a finite natural number $r$ such that, for every $\mathcal{A} = \langle A, \vec{O}\rangle \in STRUCT[\vec{R}]$, for every two $m$-ary vectors $\vec{a}$ and $\vec{b}$ of elements of $A$, it is the case that $N_r^{\mathcal{A}}(\vec{a}) \simeq N_r^{\mathcal{A}}(\vec{b})$ implies $\vec{a} \in e[\vec{O}/\vec{R}]$ if and only if $\vec{b} \in e[\vec{O}/\vec{R}]$.*

So for every flat relational query expressible in $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum)$, there is some number $r$ such that, for every pair $(\vec{a}, \vec{b})$ whose neighbourhoods are isomorphic up to radius $r$, they must either be both in the result of the query or both not in the result of the query. We call the smallest such number $r$ the "locality index" of the query.

An equivalence relation $\vec{a} \approx_r^{\mathcal{A}} \vec{b}$ is induced by $N_r^{\mathcal{A}}(\vec{a}) \simeq N_r^{\mathcal{A}}(\vec{b})$. Each resulting isomorphism type, also called a neighbourhood type, induced by this equivalence relation is definable by a first-order formula $\xi(\vec{u})$ such that $\vec{a} \approx_r^{\mathcal{A}} \vec{b}$ if and only if $\mathcal{A}, [\vec{b}/\vec{u}] \models \xi(\vec{u})$. This formula can be thought of as a "diagram" showing how objects in this neighbourhood type are "connected" to each other and to the reference objects (i.e., $\vec{u}$). If a restriction is imposed so that $\mathcal{G}(\mathcal{A})$ has degree at most $k$, the number of resulting isomorphism types realised for each $r > 0$ is finite. This restriction, together with the locality property, implies that the result of any flat relational query $e[\vec{R}]$ in $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum)$ is completely characterized by a finite number of neighbourhood types.

[2] A function $f \in \Omega$ has type of the form $Q \times \cdots \times Q \to Q$.
[3] An aggregate function $g \in \Theta$ has type of the form $\{\!|Q|\!\} \to Q$, where $\{\!|Q|\!\}$ denotes the type of multiset of numbers.

$$
\begin{array}{rcl}
\bigcup\{e \mid x \in \{\}\} & \mapsto & \{\} \\
\bigcup\{e_1 \mid x \in \{e_2\}\} & \mapsto & e_1[e_2/x] \\
\bigcup\{e \mid x \in (e_1 \cup e_2)\} & \mapsto & \bigcup\{e \mid x \in e_1\} \cup \bigcup\{e \mid x \in e_2\} \\
\bigcup\{e_1 \mid x \in \bigcup\{e_2 \mid y \in e_3\}\} & \mapsto & \bigcup\{\bigcup\{e_1 \mid x \in e_2\} \mid y \in e_3\} \\
\bigcup\{e \mid x \in (\mathit{if}\ e_1\ \mathit{then}\ e_2\ \mathit{else}\ e_3)\} & \mapsto & \mathit{if}\ e_1\ \mathit{then}\ \bigcup\{e \mid x \in e_2\}\ \mathit{else}\ \bigcup\{e \mid x \in e_3\} \\
\pi_i(e_1, \ldots, e_2) & \mapsto & e_i \\
\pi_i\ (\mathit{if}\ e_1\ \mathit{then}\ e_2\ \mathit{else}\ e_3) & \mapsto & \mathit{if}\ e_1\ \mathit{then}\ \pi_i\ e_2\ \mathit{else}\ \pi_i\ e_3 \\
\mathit{if\ true\ then}\ e_2\ \mathit{else}\ e_3 & \mapsto & e_2 \\
\mathit{if\ false\ then}\ e_2\ \mathit{else}\ e_3 & \mapsto & e_3 \\
\sum\{\!|e \mid x \in \{\}|\!\} & \mapsto & 0 \\
\sum\{\!|e \mid x \in \{e'\}|\!\} & \mapsto & e[e'/x] \\
\sum\{\!|e \mid x \in \mathit{if}\ e_1\ \mathit{then}\ e_2\ \mathit{else}\ e_3|\!\} & \mapsto & \mathit{if}\ e_1\ \mathit{then}\ \sum\{\!|e \mid x \in e_2|\!\}\ \mathit{else}\ \sum\{\!|e \mid x \in e_3|\!\} \\
\sum\{\!|e \mid x \in e_1 \cup e_2|\!\} & \mapsto & \sum\{\!|e \mid x \in e_1|\!\} + \sum\{\!|\mathit{if}\ x \in e_1\ \mathit{then}\ 0\ \mathit{else}\ e \mid x \in e_2|\!\} \\
\sum\{\!|e \mid x \in \bigcup\{e_1 \mid y \in e_2\}|\!\} & \mapsto & \sum\{\!|\sum\{\!|(e \div \sum\{\!|\sum\{\!|\mathit{if}\ x = v\ \mathit{then}\ 1\ \mathit{else}\ 0 \mid v \in e_1|\!\} \\
 & & \mid y \in e_2|\!\})| \ x \in e_1|\!\} \mid y \in e_2|\!\}
\end{array}
$$

**Figure 3: A system of rewrite rules for $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ \mathit{powerset})$.**

The following proposition on objects that are connected in a neighbourhood type is easily proved.

PROPOSITION 7. *Given a neighbourhood type $\xi(u_1, \ldots, u_m)$ induced by some $r$-neighbourhood. Suppose $u_i$ and $u_j$ are connected to each other in $\xi(u_1, \ldots, u_m)$. Then for any $\tau_m$ structure $\mathcal{A} = \langle A, \vec{O}, o_1, \ldots, o_m\rangle$ realizing $\xi(u_1, \ldots, u_m)$, it is the case that $d^{\mathcal{A}}(o_i, o_j) \leq 2mr + 1$.*

# 4. COMPLEXITY OF QUERIES ON SEVERELY DICHOTOMOUS STRUCTURES

Given a signature $\tau$. A "motif" of radius $r$ is a first-order formula $\rho(u)$ with a single free variable $u$ and has locality index $r$ on all $\tau$ structures. We say a $\tau$ structure $\mathcal{A}$ is "bounded" at threshold $g$ by a motif $\rho(u)$ of radius $r$ if there are at most $rg$ elements in the universe of $\mathcal{A}$ that make $\rho(u)$ true—i.e., $|\{a \in A \mid \mathcal{A}, [a/u] \models \rho(u)\}| \leq rg$. We say a class $\mathcal{C}$ of $\tau$ structures is "bounded" at threshold $g$ by a motif $\rho(u)$ if $\rho(u)$ bounds all structures in $\mathcal{C}$ at threshold $g$. On the other hand, we say $\mathcal{C}$ is "unbounded" by $\rho(u)$ if for every $g > 0$, there is $\mathcal{A} \in \mathcal{C}$ that is not bounded at threshold $g$ by $\rho(u)$.

Now we define the class of dichotomous structures.

*Definition 1.*

1. A class $\mathcal{C}$ of $\tau$ structures is called "dichotomous" at threshold $g$ if and only if (i) $\mathcal{C}$ is unbounded by some motifs, and (ii) $\mathcal{C}$ is bounded by all other motifs at threshold $g$.
2. A dichotomous class $\mathcal{C}$ is "deep" if it is unbounded by some motifs of radius $r$ at every $r$.
3. A dichotomous class $\mathcal{C}$ has "severity" $\ell$ if for every motif $\rho(u)$ that unbounds $\mathcal{C}$, there is a sequence of structures $\mathcal{A}_1, \mathcal{A}_2, \ldots$, in $\mathcal{C}$ having universe of increasing size, and the ratio $|\{a \in A_i \mid \mathcal{A}_i, [a/u] \models \rho(u)\}|/|A_i|$ tends to $\ell$ as $i$ tends to infinity.
4. A dichotomous class is "severely dichotomous" if it has severity 1.

Given a $\tau$ structure $\mathcal{A} = \langle A, \vec{O}\rangle$, we define its size as the size of its universe: $|\mathcal{A}| = |A|$. We can now state and prove our Dichotomy Theorem for $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ \mathit{powerset})$.

THEOREM 1 (DICHOTOMY). *Let $e[\vec{R}] : \{b \times \cdots \times b\}$ be a flat relational query in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ \mathit{powerset})$ and the input $\vec{R}$ comes from a class $\mathcal{C}$ where (i) $\mathcal{C}$ is severely dichotomous, and (ii) the Gaifman graphs of its structures have degree at most $k$. Then either $e[\vec{R}]$ is expressible in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$; or there is a sequence of structures $\mathcal{A}_i = \langle A_i, \vec{O}_i\rangle \in \mathcal{C}$ of increasing size, such that $\mathit{sizeof}(e[\vec{O}_i/\vec{R}] \Downarrow)$ is $\Omega(2^{|\mathcal{A}_i|})$.*

PROOF. Let $\mathcal{C}$ be severely dichotomous at threshold $g$, and the Gaifman graphs of structures in it have degree at most $k$. Let $\mathcal{A} = \langle A, \vec{O}\rangle \in \mathcal{C}$ be the input to the query $e[\vec{R}]$.

By Proposition 5, the system of rewrite rules in Figure 3 does not increase complexity. By Proposition 4, it is sound and strongly normalizing. Thus we assume $e[\vec{R}]$ is an expression in normal form with respect to this system of rewrite rules.

If the powerset operation does not appear in $e[\vec{R}]$, then the theorem trivially holds. So, let it contain some occurrences of the powerset operation. Suppose $\mathit{powerset}\ e'[\vec{R}, \vec{x}]$ is the earliest occurrence of the powerset operation to be evaluated when $e[\vec{R}]$ is evaluated according to the operational semantics given in Figure 2.

There are only two ways to introduce a new variable in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ \mathit{powerset})$, namely the $\bigcup\{e_1 \mid x \in e_2\}$ construct and the $\sum\{\!|e_1 \mid x \in e_2|\!\}$ construct. So, each new free variable $x_i$ in $\vec{x}$ must have been introduced in an enclosing expression of the form $\bigcup\{\cdots \mathit{powerset}\ e'[\vec{R}, \vec{x}]\cdots \mid x_i \in E\}$ or the form $\sum\{\!|\cdots \mathit{powerset}\ e'[\vec{R}, \vec{x}]\cdots \mid x_i \in E|\!\}$. As the entire expression $e[\vec{R}]$ is in normal form, and $e'[\vec{R}, \vec{x}]$ is the earliest instance of the powerset operation to be evaluated, $E$ must be one of the $R_i$ in $\vec{R}$, which is a flat relation. Hence, $x_i$ must have height 0 and has a type of the form $b \times \cdots \times b$. As the powerset operation requires its input to be a flat relation, we know $e'[\vec{R}, \vec{x}]$ has type of the form $\{b \times \cdots \times b\}$. Therefore, $e'[\vec{R}, \vec{x}]$ is a flat relational query in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$.

We need to inspect the entire expression $e[\vec{R}]$ to extract

all the conditions $\psi(\vec{x})$ that must hold on $\vec{x}$ before $e'[\vec{R},\vec{x}]$ is evaluated. Let the notation $\odot$ represents a subexpression that contains the occurrence of the expression $e'[\vec{R},\vec{x}]$, and recall that $\varphi_E$ denotes the formula in $L_{aggr}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$ that the expression $E$ in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$ translates to. We define the extraction function $\overrightarrow{e[\vec{R}]}$ by induction below.

- $\overrightarrow{\bigcup\{\mathit{if}\ E\ \mathit{then}\ \odot\ \mathit{else}\ F \mid x \in R\}} = (x, \vec{x}' : R, \vec{R}' : \varphi_E \wedge \psi)$, where $\overrightarrow{\odot} = (\vec{x}' : \vec{R}' : \psi)$.
- $\overrightarrow{\bigcup\{\mathit{if}\ E\ \mathit{then}\ F\ \mathit{else}\ \odot \mid x \in R\}} = (x, \vec{x}' : R, \vec{R}' : \neg\varphi_E \wedge \psi)$, where $\overrightarrow{\odot} = (\vec{x}' : \vec{R}' : \psi)$.
- $\overrightarrow{\bigcup\{\mathit{if}\ \odot\ \mathit{then}\ E\ \mathit{else}\ F \mid x \in R\}} = (x, \vec{x}' : R, \vec{R}' : \psi)$, where $\overrightarrow{\odot} = (\vec{x}' : \vec{R}' : \psi)$.
- $\overrightarrow{\odot \cup E} = \overrightarrow{\odot}$,
- $\overrightarrow{E \cup \odot} = \overrightarrow{\odot}$,
- the remaining rules are analogous; we omit them to avoid tedium.

By the conservative extension property (Proposition 4), all the types that appear in the typing derivation of $e'[\vec{R},\vec{x}]$ have height at most 1 (i.e., must be flat). We note that, by Proposition 1, $e'[\vec{R},\vec{x}]$ is equivalent to a formula $\varphi(\vec{x},y)$ in $L_{aggr}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$. We extract all the conditions $\psi(\vec{x})$ that must hold on $\vec{x}$ before $e'[\vec{R},\vec{x}]$ is evaluated—i.e., $\overrightarrow{e[\vec{R}]} = (\vec{x} : \vec{R} : \psi(\vec{x}))$. Let $C[\vec{x}]$ be the expression in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$ that is equivalent to the formula $\psi(\vec{x})$ in $L_{aggr}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$. Now we define $\phi(\vec{x},y) =_{df} \bigwedge_i R_i(x_i) \wedge \psi(\vec{x}) \wedge \varphi(\vec{x},y)$, which is a formula in $L_{aggr}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$ and corresponds to the query $\{(\vec{x},y) \mid x_1 \in R_1, ..., x_n \in R_n, C[\vec{x}], y \in e'[\vec{R},\vec{x}]\}$ in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$.

By the locality property (Proposition 6) of $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$, let $\phi(\vec{x},y)$'s locality index be $r$. Since the Gaifman graph of our input structure has degree at most $k$, there is a finite number of $r$-neighbourhood types $\xi_h(\vec{x},y)$ such that $\neg(\xi_h(\vec{x},y) \Rightarrow \neg\phi(\vec{x},y))$. Thus $\bigvee_h \xi_h(\vec{x},y)$ if and only if $\phi(\vec{x},y)$. For convenience, let us refer to these neighbourhood types as the "qualifying neighbourhood types".

Let $x_{i,j}$ be the $j$th component of $x_i$ in $\vec{x}$, $y_{,l}$ be the $l$th component of $y$, and $\pi_l o'$ the $l$th component of a tuple $o'$ of objects in $\mathcal{A}$. Suppose for each $y_{,l}$ and qualifying neighbourhood type $\xi_h(\vec{x},y)$, we are able to determine a number $H_{h,l}$ in a manner that is independent of $\mathcal{A}$, such that given any $\vec{o}$ of the appropriate types, it is the case that $|\{\pi_l o' \mid \mathcal{A}, [\vec{o}/\vec{x}, o'/y] \models \xi_h(\vec{x},y)\}| < H_{h,l}$. Then we can make the following conclusions. There are at most $H_{h,l}$ distinct instantiations of $y_{,l}$ that make $\xi_h(\vec{x},y)$ true. Thus, there are at most $H^*_h = \prod_l H_{h,l}$ distinct instantiations of $y$ that make $\xi_h(\vec{x},y)$ true. Thus, there are at most $H^* = \sum_h \prod_l H_{h,l}$ distinct instantiations of $y$ that make $\phi(\vec{x},y)$ true. By definition of $\phi(\vec{x},y)$, there are at most $H^*$ tuples in the result of evaluating $e'[\vec{O}'/\vec{R}, \vec{o}/\vec{x}]$ in the context of $e[\vec{R}]$. In this case, this powerset operation can be eliminated by replacing it with an expression $\mathit{powerset}_{H^*}\ e'[\vec{R},\vec{x}]$. Here, $\mathit{powerset}_{H^*}$ is a function for producing subsets of size at most $H^*$ and is expressible in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$. We repeat the entire process above as many times as necessary. If all occurrences of the powerset operation are eliminated, then the original query $e[\vec{R}]$ must already be expressible in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$.

Now, let us show how to determine the crucial number $H_{h,l}$. We only need to consider three situations when we compute $H_{h,l}$. The first situation is when $y_{,l}$ is connected to some $x_{i,j}$ in $\xi_h(\vec{x},y)$. Since $\xi_h(\vec{x},y)$ describes a $r$-neighbourhood, by Proposition 7, the distance between $x_{i,j}$ and $y_{,l}$ is at most $2mr+1$, where $m$ is the length of the tuple of variables denoted by $\vec{x},y$. Since the Gaifman graph of our input structure has degree at most $k$, for any instantiation $\vec{o}$ for $\vec{x}$, there are at most $k^{2mr+1}$ distinct instantiations for $y_{,l}$. So we can set $H_{h,l} = k^{2mr+1}$ for this first situation.

On the other hand, $y_{,l}$ may not be connected to any $x_{i,j}$ in $\xi_h(\vec{x},y)$. Let $\xi'_h(y_{,l}) =_{df} \exists\vec{x},\ y_{,1},\ ...\ y_{,l-1},\ y_{,l+1},\ ...\ y_{,m}.\xi_h(\vec{x},y)$. By the locality property (Proposition 6), let the locality index of $\xi'_h$ be $r'$. Since the Gaifman graph has degree at most $k$, there is a finite number of $r'$-neighbourhood types $\rho_{h,d}(y_{,l})$ such that $\neg(\rho_{h,d}(y_{,l}) \Rightarrow \neg\xi'_h(y_{,l}))$. Thus $\bigvee_d \rho_{h,d}(y_{,l})$ if and only if $\xi'_h(y_{,l})$. Suppose we are able to determine a number $H'_{h,l,d}$, in a manner that is independent of $\mathcal{A}$, such that $|\{a \mid \mathcal{A}, [a/y_{,l}] \models \rho_{h,d}(y_{,l})\}| < H'_{h,l,d}$. Then we can set $H_{h,l} = \sum_d H'_{h,l,d}$.

How do we determine $H'_{h,l,d}$? Each $\rho_{h,d}(y_{,l})$ is a motif that either bounds $\mathcal{C}$ at threshold $g$ or unbounds $\mathcal{C}$. This brings us into the second and the third situation respectively.

The second situation is when $\rho_{h,d}(y_{,l})$ bounds $\mathcal{C}$ at threshold $g$. Recall that the locality index of $\xi'_h$ is $r'$. So the locality index of $\rho_{h,d}(y_{,l})$ is also $r'$. Then, by definition of bounding motifs, there are at most $r'g$ number of instantiations for $y_{,l}$ that make $\rho_{h,d}(y_{,l})$ true. So we can set $H_{h,l,d} = r'g$.

The third and last situation is when $\rho_{h,d}(y_{,l})$ unbounds $\mathcal{C}$. As $\mathcal{C}$ is severely dichotomous, there is a sequence of structures $\mathcal{A}_1 = \langle A_1, \vec{O}_1\rangle$, $\mathcal{A}_2 = \langle A_2, \vec{O}_2\rangle$, ... in $\mathcal{C}$ having universe of increasing size, such that the ratio $|\{a \in A_i \mid \mathcal{A}_i, [a/y_{,l}] \models \rho_{h,d}(y_{,l})\}|/|A_i|$ tends to 1. Note that all of the objects $a$ in $\{a \in A_i \mid \mathcal{A}_i, [a/y_{,l}] \models \rho_{h,d}(y_{,l})\}$ must be used to instantiate $y_{,l}$ as required by the locality property, because $\rho_{h,d}(y_{,l})$ is a neighbourhood type. That is, the number of instantions for $y_{,l}$ is essentially $|A_i|$. In this case, we cannot set $H_{h,l,d}$ (and thus $H_{h,l}$) to a finite value in a manner that is independent of the input structure to our query $e[\vec{R}]$. Therefore, the powerset operation in $\mathit{powerset}\ e'[\vec{R},\vec{x}]$ cannot be eliminated in this situation. On the other hand, the number of instantiations for $y$ is $\Omega(|A_i|)$ since one of its components, $y_{,l}$, has $|A_i|$ instantiations. So, $e'[\vec{R},\vec{x}]$ has $\Omega(|A_i|)$ elements. Thus, $\mathit{sizeof}(e[\vec{O}_i/\vec{R}] \Downarrow)$ is $\Omega(2^{|A_i|})$ as desired.

By the way, if $\mathcal{C}$'s severity level was some $\ell < 1$, the number of instantiations for $y_{,l}$ would be $\ell|A_i|$. In this case, $\mathit{sizeof}(e[\vec{O}_i/\vec{R}] \Downarrow)$ would be $\Omega(2^{\ell|A_i|})$. □

# 5. REMARKS

Let us close this paper by discussing how the Dichotomy Theorem generalizes the old results of Suciu, Paredaens, Van den Bussche, Biskup, and others [14, 6, 3].

## 5.1 On the result of Suciu and Paredaens

Suciu and Paredaens [14] have shown earlier that all implementations of the transitive closure of a single long chain in the nested relational algebra of Abiteboul and Beeri [1], which is equivalent to $\mathcal{NRC}(\mathit{powerset})$, require exponential

space. In this paper, we have improved on this result in several significant ways.

Firstly, our Dichotomy Theorem is not limited to a single specific query such as the transitive closure of a chain. It works equally well for all flat relational queries (on severely dichotomous structures) that are inexpressible in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$ but expressible in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ powerset)$, such as the transitive closure of a set of $k$ single long chains, a set of $k$ long circles, a deep full binary tree, and many more.

COROLLARY 1. *For any flat relational query on any severely dichotomous class of structures whose Gaifman graphs have degree at most $k$, if it is inexpressible in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$ but is expressible in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ powerset)$, then all of its implementations in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ powerset)$ need exponential space.*

Secondly, our Dichotomy Theorem is not limited to $\mathcal{NRC}(powerset)$. We have already shown it for a more powerful query language, $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ powerset)$, which has aggregate functions and better captures queries in SQL.

Thirdly, our proof technique is more general. So long as the query language has the locality property before the *powerset* operation is added to it and, after the *powerset* operation is added to it, has a normal form induced by the conversative extension property, our proof works. For example, let $\Omega$ comprise the functions $\otimes$, $\iota$, and $\oslash$; and $\Theta$ comprise the aggregate function $\Pi$ as defined below:

- $\otimes : Q \times Q \to Q$ is a commutative associative function;
- $\iota : Q$ is its identity;
- $\oslash : Q \times Q \to Q$ is a "duplicate compensator" function satisfying
$$\overbrace{(a \oslash n) \otimes \cdots \otimes (a \oslash n)}^{n \text{ times}} = a;$$
- $\Pi\{\!|e_2 \mid x \in e_1|\!\}$ is an aggregate function that applies the function $f(x) = e_2$ to every object in the set $e_1$ and aggregate the results using $\otimes$, and so satisfying $\Pi\{\!|e_2 \mid x \in e_1|\!\} = f(o_1) \otimes \cdots \otimes f(o_m) \otimes \iota$ where $e_1$ is the set $\{o_1, ..., o_m\}$.

Then we can prove the following more general Dichotomy Theorem.

THEOREM 2. *For any flat relational query on any severely dichotomous class of structures whose Gaifman graphs have degree at most $k$, if it is inexpressible in $\mathcal{NRC}(\Omega,\ \Theta,\ Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$ but is expressible in $\mathcal{NRC}(\Omega,\ \Theta,\ Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ powerset)$, then all of its implementations in $\mathcal{NRC}(\Omega,\ \Theta,\ Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ powerset)$ need exponential space.*

PROOF. (Sketch) It is easy to show that $\mathcal{NRC}(\Omega,\ \Theta,\ Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ powerset)$ has the conservative extension property by adding the following rules to the rewrite system:

- $\Pi\{\!|e \mid x \in \bigcup\{e_1 \mid y \in e_2\}|\!\} \mapsto \Pi\{\!|\Pi\{\!|(e \oslash \sum\{\!|\sum\{\!|\mathit{if}\ x = v\ \mathit{then}\ 1\ \mathit{else}\ 0 \mid v \in e_1|\!\} \mid y \in e_2|\!\})| \ x \in e_1|\!\} \mid y \in e_2|\!\}$,
- the other rules are analogous to those for $\sum$ and are omitted.

The result of Hella et al. [10] also implies $\mathcal{NRC}(\Omega,\ \Theta,\ Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$ has the locality property. So the proof of our Dichotomy Theorem can be applied verbatim to conclude this proposition. □

We have also gained some further insights on the use of *powerset* operation. If a function $f$ on a severely dichotomous class of structures of degree $\leq k$ is expressible in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$, but an implementation $e$ of it in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ powerset)$ uses the *powerset* operation, our Dichotomy Theorem actually does not guarantee the removal of *powerset* operation in the implementation $e$. It is perfectly reasonable for $f$ to have an inefficient implementation in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ powerset)$; i.e., the Dichotomy Theorem is not a clever optimizer. On the other hand, the Dichotomy Theorem is sufficient for us to conclude that, if $f$ is inexpressible in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$, then all of its implementation in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ powerset)$ have to use the *powerset* operation at least once in a non-trivial way.

## 5.2 On the result of Van den Bussche

Definition 1 divides the severely dichotomous class of structures into two subclasses: the class of structures that are deep and the class of structures that are not deep. The deep class has motifs of increasingly large radius that unbounds the class, while in the non-deep class, all motifs that unbound the class have small radius. Structures in the non-deep class are basically things like an arbitrarily large set of short chains—the class is unbounded by the number of short chains rather than the length of these chains.

We have so far encountered examples of deep severely dichotomous classes of structures. Let us now give an example of non-deep severely dichotomous classes of structures, viz., the class of sets of type $\{b\}$ having an arbitrarily large number of elements! The following result is a simple consequence of Corollary 1.

COROLLARY 2. *1. For any flat relational query on unary database schemas, if it is inexpressible in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$, but is expressible in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ powerset)$, then all of its implementations in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ powerset)$ need exponential space.*

*2. In particular, all implementations of the parity query in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ powerset)$ need exponential space.*

PROOF. The first part follows immediately from Corollary 1 and the fact that the class of sets of type $\{b\}$ having a large number elements is severely dichotomous.

For the second part, we observe that if the parity query is expressible in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$, then it is also possible to express a query in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$ to test whether a single long chain has an even number of nodes. By the finite-cofiniteness property of $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$ established by Libkin and Wong [13], the latter query is inexpressible in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$. Thus, the parity query is inexpressible in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum)$. Since the simplest instance of the parity query is an example of a query on unary database schemas, it follows from the first part that exponential space is needed to implement the parity query in $\mathcal{NRC}(Q,\ +,\ \cdot,\ -,\ \div,\ \sum,\ powerset)$. □

This corollary straightforwardly generalizes the old result of Van den Bussche (i.e., Proposition 3) to a more powerful query language.

## 5.3 On the result of Biskup et al.

It is possible to capture the key feature of the equation algebra of Biskup et al. [3] by augmenting $\mathcal{NRC}$ with the following construct:

$$\frac{e_1 : \{\{b \times \cdots \times b\}\} \qquad e_2 : \{b \times \cdots \times b\}}{\bigcup\{e_1 \mid x^{\{b\times\cdots\times b\}} \subseteq e_2\} : \{\{b \times \cdots \times b\}\}}$$

with the following call-by-value operational semantics:

$$\frac{\begin{array}{c} e_2 \Downarrow \{C_1, \ldots, C_n\} \\ e_1[C_1'/x] \Downarrow C_1'' \quad \ldots \quad e_1[C_{2^n}'/x] \Downarrow C_{2^n}'' \end{array}}{\begin{array}{c} \bigcup\{e_1 \mid x \subseteq e_2\} \Downarrow C_1'' \cup \cdots \cup C_{2^n}'' \\ \text{where } C_1', \ldots, C_{2^n}' \text{ are the subsets of } \{C_1, \ldots, C_n\} \end{array}}$$

The meaning of the construct $\bigcup\{e_1 \mid x \subseteq e_2\}$ is the set $f(C_1') \cup \cdots \cup f(C_{2^n}')$, where $f(x) = e_1$ and $C_1'$, ..., $C_{2^n}'$ are all the subsets of $e_2$. Operationally, as in Biskup et al. [3], this construct is executed by enumerating each subset $C_i'$ of $e_2$ and inserting $f(C_i)'$ into the result set one by one. If most of the $f(C_i)'$ are empty or identical, the evaluation of this construct should take only polynomial space even though the time complexity is exponential.

We denote the extensions of $\mathcal{NRC}$ and $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum)$ with this contruct by $\mathcal{NRC}(eqn)$ and $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum, eqn)$ respectively.

It is easy to see $\bigcup\{e_1 \mid x \subseteq e_2\} = \bigcup\{e_1 \mid x \in powerset\ e_2\}$ and $powerset\ e = \bigcup\{\{x\} \mid x \subseteq e\}$. Thus, $\mathcal{NRC}(powerset)$ and $\mathcal{NRC}(eqn)$ have the same expressive power and, similarly, $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum, powerset)$ and $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum, eqn)$ have the same expressive power.

If an upperbound $n$ on the cardinality of $e_2$ is known in advance, we can replace $\bigcup\{e_1 \mid x \subseteq e_2\}$ by an expression $\bigcup\{e_1 \mid x \in powerset_n\ e_2\}$. Here, $powerset_n\ e_2$ is an expression that enumerates all subsets of $e_2$ upto size $k$, which is obviously definable in $\mathcal{NRC}$.

It follows easily from the work of Biskup et al. [3] that the transitive closure query can be implemented in $\mathcal{NRC}(eqn)$ using polynomial space. Therefore, the Dichotomy Theorem in terms of space complexity does not hold in $\mathcal{NRC}(eqn)$ and $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum, eqn)$.

PROPOSITION 8. *1. There is a flat relational query on a severely dichotomous class of structures of degree at most k that cannot be implemented in $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum)$, but can be implemented in $\mathcal{NRC}(eqn)$ using polynomial space.*

*2. In particular, transitive closure is such a query.*

Nevertheless, it is possible to prove a general Dichotomy Theorem in terms of time complexity for $\mathcal{NRC}(eqn)$ and $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum, eqn)$. We sketch a proof for the latter.

THEOREM 3. *Let $e[\vec{R}] : \{b \times \cdots \times b\}$ be a flat relational query in $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum, eqn)$ and the input $\vec{R}$ comes from a class $\mathcal{C}$ where (i) $\mathcal{C}$ is severely dichotomous, and (ii) the Gaifman graphs of its structures have degree at most k. Then either $e[\vec{R}]$ is expressible in $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum)$; or there is a sequence of structures $\mathcal{A}_i = \langle A_i, \vec{O}_i \rangle \in \mathcal{C}$ of increasing size such that $timeof(e[\vec{O}_i/\vec{R}_i] \Downarrow)$ is $\Omega(2^{|\mathcal{A}_i|})$.*

PROOF. (Sketch) It can be shown that $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum, eqn)$ has the conservative extension property. Moreover, the associated rewrite rules do not increase time complexity; see Proposition 5. So $e[\vec{R}]$ is assumed to be in normal form. Now, we look for the first expression of the form $\bigcup\{e_1 \mid x \subseteq e_2\}$ in $e[\vec{R}]$ that is to be executed. Then, by an argument similar to that in Theorem 4, we know $e_2$ is a flat relational query in $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum)$. Since $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum)$ has the locality property, again by an argument similar to that in Theorem 4, we can either determine an upper bound $n$ on the size of $e_2$ independent of the universe of the input structure or show that its size is as large as the universe of the input structure. In the first situation, the expression can be replaced by $\bigcup\{e_1 \mid x \in powerset_n\ e_2\}$, which is expressible in $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum)$; so we make the replacement and repeat the whole process. In the second situation, we know that this occurrence of $\bigcup\{e_2 \mid x \subseteq e_2\}$ takes exponential time to evaluate. □

Since parity, transitive closure, etc. are inexpressible in $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum)$, and their input can be easily restricted to severely dichotomous classes of structures having degree at most $k$, all their implementations in $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum, eqn)$ require exponential time.

COROLLARY 3. *1. All implementations of the parity query in $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum, eqn)$ need exponential time.*

*2. All implementations of the transitive closure query in $\mathcal{NRC}(Q, +, \cdot, -, \div, \sum, eqn)$ need exponential time.*

# 6. ACKNOWLEDGEMENTS

I have not done work on query language theory for more than a decade. I re-started on the subject when Val Tannen asked me to contribute a book chapter to a festschrift for Peter Buneman last year. I am grateful to both of them for their mentorship when I was a student at UPenn and for re-triggering my interest in query language theory. I dedicate this paper to Peter Buneman.

I am also thankful to an anonymous referee who pointed out the relevant works of Van den Bussche and Biskup et al. This led me to expand this paper to include new results on the parity query and the equation algebra. These inclusions help provide a more complete appreciation of the Dichotomy Theorem.

This work was supported in part by a Singapore Ministry of Education grant MOE-T1-251RES1206.

# 7. REFERENCES

[1] S. Abiteboul and C. Beeri. The power of languages for the manipulation of complex values. *VLDB Journal*, 4(4):727–794, 1995.

[2] S. Abiteboul and V. Vianu. Generic computation and its complexity. In *Proc. 23rd ACM Symp. Theory of Computing*, pages 209–219, 1991.

[3] J. Biskup, J. Paredaens, T. Schwentick, and J. Van den Bussche. Solving equations in the relational algebra. *SIAM Journal on Computing*, 33(5):1052–1055, 2004.

[4] P. Buneman, L. Libkin, D. Suciu, V. Tannen, and L. Wong. Comprehension syntax. *SIGMOD Record*, 23(1):87–96, 1994.

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

[6] J. Van den Bussche. Simulation of the nested relational algebra by the flat relational algebra, with an application to the complexity of evaluating powerset algebra expressions. *Theoretical Computer Science*. 254(1–2):363–377, 2001.

[7] L. Colson. About primitive recursive algorithms. *Theoretical Computer Science*, 83:57–69, 1991.

[8] G. Dong, L. Libkin, and L. Wong. Local properties of query languages. *Theoretical Computer Science*, 239:277–308, 2000.

[9] H. Gaifman. On local and non-local properties. In *Proc. Herbrand Symp., Logic Colloq. '81*, pages 105–135, 1982.

[10] L. Hella, L. Libkin, J. Nurmonen, and L. Wong. Logics with aggregate operators. *Journal of the ACM*, 48(4):880–907, 2001.

[11] L. Libkin. On forms of locality over finite models. In *Proc. 12th IEEE Symp. Logic in Computer Science*, pages 204–215, 1997.

[12] L. Libkin and L. Wong. Conservativity of nested relational calculi with internal generic functions. *Information Processing Letters*, 49(6):273–280, 1994.

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

[14] D. Suciu and J. Paredaens. The complexity of the evaluation of complex algebra expressions. *Journal of Computer and Systems Sciences*, 55(2):322–343, 1997.

[15] D. Suciu and L. Wong. On two forms of structural recursion. In *Proc. of 5th Intl. Conf. on Database Theory*, pages 111–124, 1995.

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