# Aggregate Functions, Conservative Extension, and Linear Orders


Leonid Libkin Limsoon Wong

Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104-6389, USA


## 1 Summary

Practical database query languages are usually equipped with some aggregate functions. For example, "find mean of column" can be expressed in SQL. However, the manner in which aggregate functions were introduced in these query languages leaves something to be desired. Breazu-Tannen, Buneman, and Wong [3] introduced a nested relational language $\mathcal{NRC}(=)$ based on monads [16, 24] and structural recursion [1, 2]. It was shown in Wong [27] that this language is equivalent to the nested relational algebras of Thomas and Fischer [22], Schek and Scholl [20], and Colby [4]. $\mathcal{NRC}(=)$ enjoys certain advantages over these languages: it is naturally embedded in functional languages, it is readily extensible, and it has a compact equational theory. Therefore, it is used in this report as a basis for investigating aggregate functions.

In section 2, the nested relational calculus $\mathcal{NRC}(=)$ is described. It is then endowed with rational numbers, rational arithmetic, and a summation operator. The augmented language, $\mathcal{NRC}(\mathbb{Q}, +, \cdot, -, \div, \sum, =)$, is able to express a variety of aggregate functions commonly found in real database query languages. The main results of this paper remain valid in a uniform way if any summation-like primitive, such as bounded product, is added to the language. This approach is more disciplined and general than those proposed by Klug [12], Ozsoyoglu, Ozsoyoglu, and Matos [18], and Klausner and Goodman [11].

In section 3, we prove that every function $f : s \to t$ expressible in $\mathcal{NRC}(\mathbb{Q}, +, \cdot, -, \div, \sum, =)$ can be computed without using any intermediate data whose depth of nesting of sets exceeds that of the input and output. This is known as the *conservative extension* property. Conservativity of nested relational query languages in the absence of aggregate functions was studied by Paredaens and Van Gucht [19] and Wong [26]. The former proved that it holds when input and output are flat relations. The latter generalized it to any input and output. Conservativity in the presence of aggregate functions was not previously studied.

In section 4, the conservative extension property is used to demonstrate the somewhat surprising fact that $\mathcal{NRC}(\mathbb{Q}, +, \cdot, -, \div, \sum, =)$ cannot express the usual linear ordering on rational numbers. As linear orders play a central role in fundamental data organization algorithms [14], this calls for special attention. We present a technique for lifting linear order at base types to linear order at all types. This technique yields linear orders that are expressible in $\mathcal{NRC}(\mathbb{Q}, +, \cdot, -, \div, \sum, =, \leq)$, which is the language obtained by augmenting $\mathcal{NRC}(\mathbb{Q}, +, \cdot, -, \div, \sum, =)$ with linear orders at base types. Linear order is known to increase expressive power in the context of database query languages [8, 23]. In our case, this is a major advantage. Queries such as "find maximum of column," "find mode of column" and "test parity of cardinality of a set" are expressible in $\mathcal{NRC}(\mathbb{Q}, +, \cdot, -, \div, \sum, =, \leq)$. More importantly, a function that assigns rank to elements of a set is now expressible.

This rank assignment function is used in section 5 to show that $\mathcal{NRC}(\mathbb{Q}, +, \cdot, -, \div, \sum, =, \leq)$ augmented with any combination of the transitive closure operator *tc*, the bounded fixpoint operator *bfix*, or the powerset operator *powerset* retains the conservative extension property. Hull and Su [7] showed that $\mathcal{NRC}(=, \mathit{powerset})$ is not conservative over flat input and output. This failure of conservativity for $\mathcal{NRC}(=, \mathit{powerset})$ was generalized to all input and output heights by Grumbach and Vianu [6]. In contrast, our result shows that conservativity can be repaired with very little extra. Suciu [21] showed that $\mathcal{NRC}(=, \mathit{bfix})$ is conservative over flat relations. His result is remarkable in that it did not need any arithmetic nor order. Furthermore, it is also valid when bounded fixpoint is replaced by bounded partial fixpoint operator. Our result uses arithmetic but holds for bounded fixpoint operator over any input and output. In fact, our proof of conservative extension holds *uniformly* for $\mathcal{NRC}(\mathbb{Q}, +, \cdot, -, \div, \sum, =, \leq, \prod, \odot, \iota)$ where $\prod$, $\iota$, and $\odot$ are any triple of additional primitives which are in a relationship like that between $\sum$, 0, and $+$.

## 2 Nested relational calculus with summation

The monad calculus of Breazu-Tannen, Buneman, and Wong [3] is denoted $\mathcal{NRC}$ here. In this section, it is extended with rational numbers, simple arithmetics, and a summation operator. The extended language is able to express many aggregate functions commonly found in commercial relational database query languages such as SQL.

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 grammar:

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

Objects of type $\mathbb{B}$ are the two boolean values *true* and *false*. The unique object of type *unit* is 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 $s$. We also include some uninterpreted base types $b$.

Expressions of $\mathcal{NRC}$ are constructed using the rules in the figure below. Note that [3] uses $ext(\lambda x^s.e_1)(e_2)$; but here we use the equivalent construct $\bigcup\{e_1 \mid x^s \in e_2\}$ instead. The language also contains some uninterpreted constants $c$ of base type $Type(c)$ and uninterpreted functions $p$ of function type

$Type(p)$. The type superscripts are omitted in the rest of the paper because they can be inferred [17, 10]. Throughout this paper we assume the usual convention that variables are distinct and that expressions are well formed.

Lambda Calculus and Products

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

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

Set Monad

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

Booleans

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

The semantics of $\mathcal{NRC}$ was described in [3]. The lambda calculus, product, and boolean constructs are standard. We briefly repeat 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 $e_2$. The shorthand $\{o_1, \ldots. o_n\}$ is used to denote $\{o_1\} \cup \ldots \cup \{o_n\}$. 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 $e_1$.

As it stands, $\mathcal{NRC}$ can merely express queries that are purely structural. It was shown in [3] that endowing $\mathcal{NRC}$ with equality test $=^s\colon s \times s \to \mathbb{B}$ at all types $s$ elevates $\mathcal{NRC}$ to a fully fledged nested relational language (which was shown by Wong [27] to be equivalent to classical nested relational algebras of Thomas and Fischer[22], Schek and Scholl [20], and Colby [4]). That is, operations such as nest, membership test, subset test, set intersection, set difference, etc. are expressible in $\mathcal{NRC}(=)$. (We write the additional primitive in brackets to distinguish various extensions of the language.) It should also be remarked that in [3], booleans are simulated by values of type $\{unit\}$ with $\{()\}$ for *true* and $\{\}$ for *false*. However, over the class of functions of type $s \to \{s_1\} \times \cdots \times \{s_n\}$, it does not matter which presentation of booleans is used — the resulting languages have the same expressive power.

Examples. $\bigcup\{\{x, 5\cdot x\} \mid x \in \{1,2,3\}\}$ evaluates to the set $\{1,2,3,5,10,15\}$. $\bigcup\{\bigcup\{\{(x,y)\} \mid x \in X\} \mid y \in Y\}$ forms the cartesian product of sets $X$ and $Y$. $\bigcup\{\bigcup\{\{(\pi_1\ x, y)\} \mid y \in \pi_2\ x\} \mid x \in X\}$ is the unnesting of the set $X$. $\bigcup\{\{(\pi_1\ x, \bigcup\{\mathit{if}\ \pi_1\ x = \pi_1\ y\ \mathit{then}\ \{\pi_2\ y\}\ \mathit{else}\ \{\} \mid y \in X\}\} \mid x \in X\}$ is the relational nesting of $X$.

Real database query languages frequently have to deal with queries such as "select average from column," "select maximum of column," "select count from column," etc. To handle this kind of queries, additional primitives must be added to $\mathcal{NRC}$. In this paper, we add rational numbers (whose type is denoted by $\mathbb{Q}$) and the following constructs:

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

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

where $+$, $\cdot$, $-$, and $\div$ are respectively addition, multiplication, subtraction, and division of rational numbers. The summation construct $\sum\{\!|e_1 \mid x^s \in e_2|\!\}$ denotes the rational obtained by first applying the function $\lambda x.e_1$ to every item in the set $e_2$ and then adding the results up. That is, $\sum\{\!|e_1 \mid x \in X|\!\}$ is $f(o_1) + \ldots + f(o_n)$ if $f$ is the function denoted by $\lambda x.e_1$ and $\{o_1, \ldots o_n\}$, with $o_1$, ..., $o_n$ all distinct, is the set denoted by $X$. It should be emphasized that the $\{\!|e_1 \mid x \in e_2|\!\}$ part of the construct $\sum\{\!|e_1 \mid x \in e_2|\!\}$ is not an expression of the language; hence $\sum\{\!|1 \mid x \in \{5,6\}|\!\}$ is 2 and not 1.

The extended language $\mathcal{NRC}(\mathbb{Q}, +, \cdot, -, \div, \sum, =)$ is capable of expressing many aggregate operations found in commercial databases. Here are some examples:

- "Count the number of records in $R$" is $\mathit{count}(R) \triangleq \sum\{\!|1|\ x \in R|\!\}$.
- "Total the first column of $R$" is $\mathit{total}(R) \triangleq \sum\{\!|\pi_1\ x \mid x \in R|\!\}$.
- "Average of the first column in $R$" is $\mathit{average}(R) \triangleq \mathit{total}(R) \div \mathit{count}(R)$.
- "Variance of the first column of $R$" is $\mathit{variance}(R) \triangleq (\sum\{\!|\mathit{sq}(\pi_1\ x) \mid x \in R|\!\} - (\mathit{sq}(\sum\{\!|\pi_1\ x \mid x \in R|\!\}) \div \mathit{count}(R))) \div \mathit{count}(R)$, where $\mathit{sq} \triangleq \lambda y.y \cdot y$.

Aggregate functions were first introduced into flat relational algebra by Klug [12]. He introduced these functions by repeating them for every column of a relation. That is, $\mathit{aggregate}_1$ is for column 1, $\mathit{aggregate}_2$ is for column 2, and so on. Ozsoyoglu, Ozsoyoglu, and Matos [18] generalized this approach to nested relations. Our use of the summation construct is more general. On the other hand, Klausner and Goodman [11] had "stand-alone" aggregate functions such as $\mathit{mean} : \{\mathbb{Q}\} \rightarrow \mathbb{Q}$. However, they had to rely on a notion of hiding to deal correctly with duplicates. Hiding is different from projection. Let

$R \triangleq \{(1,2),(2,3),(2,4)\}$. Projecting out the second column of $R$ gives us $R' \triangleq \{1,2\}$. Hiding the second column of $R$ gives us $R'' \triangleq \{(1,[2]),(2,[3]),(2,[4])\}$, where the hidden components are shown between square brackets. Observe that the former "eliminates" duplicates as sets have no duplicate by definition. The latter "retains" the duplicated 2 by virtue of tagging them with different hidden components. Then $mean(R'')$ produces the average of the first column of $R$, whereas $mean(R')$ does not compute the mean correctly. The use of hiding to retain duplicates is rather clumsy. Our use of the summation construct is simpler.

# 3 Conservative extension

Let us first 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 $ht(\{s\}) = 1 + ht(s)$. Every expression of our language has a unique typing derivation. Hence the set height of expression $e$ is defined as $ht(e) = \max\{ht(s) \mid s$ occurs in the 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 with fixed constant* $k$ if $\mathcal{L}_{i,o,h} = \mathcal{L}_{i,o,h+1}$ for all $i$, $o$, and $h \geq \max(i,o,k)$. Note that if $\mathcal{L}$ has the conservative extension property with constant $k$, then for any additional primitive $p : s \to t$, $\mathcal{L}(p)$ has it with constant at most $\max(ht(p),k) = \max(ht(s \to t),k)$.

In this section, we present a rewrite system for $\mathcal{NRC}(\mathbb{Q},+,\cdot,-,\div,\sum,=)$ that is strongly normalizing. The normal forms induced by this rewriting are then used to prove that every definable function is definable using operators whose set height is at most the set height of the input/output of the function. The theorem implies that $\mathcal{NRC}(\mathbb{Q},+,\cdot,-,\div,\sum,=)$ has the conservative extension property with fixed constant 0. Consequently, the class $\mathcal{NRC}(\mathbb{Q},+,\cdot,-,\div,\sum,=)_{i,o,h}$ is *independent* of $h$. Hence using intermediate data structure of great height does not increase the horsepower of the language (though it frequently makes programs more elegant).

We proceed using the strategy developed by Wong [26]. First, observe that any equality test $=^s \colon s \times s \to \mathbb{B}$ can be implemented in terms of equality tests at base types $=^b \colon b \times b \to \mathbb{B}$. Hence, in the rest of the report, we assume that $=^s$, where $s$ is not a base type, is a syntactic sugar as implemented in the proposition below.

**Proposition 3.1** *Any equality test* $=^s \colon s \times s \to \mathbb{B}$ *can be implemented in terms of equality tests at base types* $=^b \colon b \times b \to \mathbb{B}$*, using* $\mathcal{NRC}(\mathbb{Q},+,\cdot,-,\div,\sum,=)$ *as the ambient language.*

**Proof.** Proceed by induction on $s$.

- $=^b$ is the given equality test at base type $b$.
- $x =^{s \times t} y \triangleq$ *if* $\pi_1\ x =^s \pi_1\ y$ *then* $\pi_2\ x =^t \pi_2\ y$ *else false*

- $X =^{\{s\}} Y \triangleq \mathit{if}\ X \subseteq^s Y\ \mathit{then}\ Y \subseteq^s X\ \mathit{else}\ \mathit{false}$, where
- $X \subseteq^s Y \triangleq ((\sum\{\!|\mathit{if}\ x \in^s Y\ \mathit{then}\ 0\ \mathit{else}\ 1 \mid x \in X|\!\}) =^{\mathbb{Q}} 0)$
- $x \in^s Y \triangleq (\sum\{\!|\mathit{if}\ x =^s y\ \mathit{then}\ 1\ \mathit{else}\ 0 \mid y \in Y|\!\}) =^{\mathbb{Q}} 1$. $\square$

The next step toward proving the conservative extension property for $\mathcal{NRC}(\mathbb{Q}, +, \cdot, -, \div, \sum, =)$ is a rewrite system adapted from Wong [26]. Let $e[e'/x]$ stands for the expression obtained by replacing all free occurrences of $x$ in $e$ by $e'$, provided the free variables in $e'$ are not captured during the substitution. Now, consider the rules below.

- $(\lambda x.e)(e') \leadsto e[e'/x]$
- $\pi_i(e_1, e_2) \leadsto e_i$
- $\pi_i(\mathit{if}\ e_1\ \mathit{then}\ e_2\ \mathit{else}\ e_3) \leadsto \mathit{if}\ e_1\ \mathit{then}\ \pi_i\ e_2\ \mathit{else}\ \pi_i\ e_3$
- $\bigcup\{e \mid x \in \{\}\} \leadsto \{\}$
- $\bigcup\{\{\} \mid x \in e\} \leadsto \{\}$
- $\bigcup\{e \mid x \in \{e'\}\} \leadsto e[e'/x]$
- $\bigcup\{e \mid x \in \mathit{if}\ e_1\ \mathit{then}\ e_2\ \mathit{else}\ e_3\}$
  $\leadsto \mathit{if}\ e_1\ \mathit{then}\ \bigcup\{e \mid x \in e_2\}\ \mathit{else}\ \bigcup\{e \mid x \in e_3\}$
- $\bigcup\{e_1 \mid x \in \bigcup\{e_2 \mid y \in e_3\}\} \leadsto \bigcup\{\bigcup\{e_1 \mid x \in e_2\} \mid y \in e_3\}$
- $\bigcup\{e \mid x \in e_1 \cup e_2\} \leadsto \bigcup\{e \mid x \in e_1\} \cup \bigcup\{e \mid x \in e_2\}$
- $\sum\{\!|e \mid x \in \{\}|\!\} \leadsto 0$
- $\sum\{\!|e \mid x \in \{e'\}|\!\} \leadsto e[e'/x]$
- $\sum\{\!|e \mid x \in e_1 \cup e_2|\!\} \leadsto \sum\{\!|e \mid x \in e_1|\!\} + \sum\{\!|\mathit{if}\ x \in e_1\ \mathit{then}\ 0\ \mathit{else}\ e \mid x \in e_2|\!\}$
- $\sum\{\!|e \mid x \in \mathit{if}\ e_1\ \mathit{then}\ e_2\ \mathit{else}\ e_3|\!\}$
  $\leadsto \mathit{if}\ e_1\ \mathit{then}\ \sum\{\!|e \mid x \in e_2|\!\}\ \mathit{else}\ \sum\{\!|e \mid x \in e_3|\!\}$
- $\sum\{\!|e \mid x \in \bigcup\{e_1 \mid y \in e_2\}|\!\}$
  $\leadsto \sum\{\!|\sum\{\!|(e \div \sum\{\!|\sum\{\!|\mathit{if}\ x = v\ \mathit{then}\ 1\ \mathit{else}\ 0 \mid v \in e_1|\!\} \mid y \in e_2|\!\}) \mid x \in e_1|\!\} \mid y \in e_2|\!\}$

This system of rewrite rules preserves the meanings of expressions. The last rule deserves special attention. 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 $\{o_1, o_2\}$. Suppose $e_1[o_1/y]$ and $e_1[o_2/y]$ both evaluate to $\{o_3\}$. Suppose $e[o_3/x]$ 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 is used to handle duplicates properly.

**Proposition 3.2 (Soundness)** *If $e_1 \leadsto e_2$, then $e_1 = e_2$. That is, $e_1 \leadsto e_2$ implies $e_1$ and $e_2$ denote the same value.*

**Proof.** Straightforward. $\square$

A system of rewrite rules is said to be strongly normalizing if any sequence of applications of these rules is guaranteed to terminate.

**Proposition 3.3 (Strong normalization)** *The above rewrite system is strongly normalizing.*

**Proof.** While the last three rules seem to increase the "character count" of expressions, it should be remarked that $\sum\{\!|e \mid x \in e'|\!\}$ is always rewritten by these three rules to an expression that decreases in the $e'$ position. This is the key to the proof. The detail can be found in the appendix of Libkin and Wong [15]. $\square$

Hence every expression can be rewritten to some normal form. These normal forms have the following property:

**Theorem 3.4 (Conservative extension)** *Let $e : s$ be an expression of $\mathcal{NRC}(\mathbb{Q}, +, \cdot, -, \div, \sum, =)$ in normal form. Then $ht(e) \leq \max(\{ht(s)\} \cup \{ht(t) \mid t$ is the type of a free variable occurring in $e\})$. Therefore, $\mathcal{NRC}(\mathbb{Q}, +, \cdot, -, \div, \sum, =)$ has the conservative extension property with fixed constant 0.*

**Proof.** By a fairly routine structural induction on $e$. $\square$

Conservativity for $\mathcal{NRC}(=)$ was studied by Paredaens and Van Gucht [19] and by Wong [26]. The former proved that $\mathcal{NRC}(=)_{i,o,h} = \mathcal{NRC}(=)_{i,o,h+1}$ for $i = o = 1$. The latter generalized it to all $i$ and $o$. However conservativity in the presence of aggregate functions was not studied. The above theorem implies that $\mathcal{NRC}(\mathbb{Q}, +, \cdot, -, \div, \sum, =)_{i,o,h} = \mathcal{NRC}(\mathbb{Q}, +, \cdot, -, \div, \sum, =)_{i,o,h+1}$ for any $i$, $o$, $h \geq \max(i, o)$. Hence we have generalized the results of [19] and [26] to the case where aggregate functions are present.

The theorem has practical significance. Some databases are designed to support nested sets up to a fixed depth of nesting. For example, Jaeschke and Schek [9] designed a statistical database whose relations are those having height at most 2. Another example is the commercially successful SQL which supports just flat relations. Both of these systems have a suitable collection of aggregate functions. "$\mathcal{NRC}(\mathbb{Q}, +, \cdot, -, \div, \sum, =)$ restricted to height 2 or 1" is a natural query language for such databases. But knowing that $\mathcal{NRC}(\mathbb{Q}, +, \cdot, -, \div, \sum, =)$ is conservative at all set heights, one can instead provide the user with the entire language $\mathcal{NRC}(\mathbb{Q}, +, \cdot, -, \div, \sum, =)$ as a more convenient query language for these databases, so long as queries have input/output height not exceeding 2 or 1.

# 4 Linear ordering on nested relations

The conservative extension property can be used to study many properties of languages (see Libkin and Wong [15] for some examples). In this section, we use it to demonstrate that $\mathcal{NRC}(\mathbb{Q},+,\cdot,-,\div,\sum,=)$ is incapable of expressing the usual linear ordering $\leq^{\mathbb{Q}}: \mathbb{Q}\times\mathbb{Q}\to\mathbb{B}$ on rational numbers. So we introduce linear order for base types. Then a technique for lifting linear order at base types to all types is presented.

**Proposition 4.1** *$\mathcal{NRC}(\mathbb{Q},+,\cdot,-,\div,\sum,=)$ cannot express $\leq^{\mathbb{Q}}$.*

**Proof.** It is enough to show that the following function cannot be expressed: $g(x)=0$ if $x\leq 1$ and $g(x)=1$ if $x>1$. Observe that $g:\mathbb{Q}\to\mathbb{Q}$ has height 0. By the conservative extension property, it must be definable using an expression of height 0. However, we can prove the following claim:

*Claim.* Let $g(x):\mathbb{Q}$ be an expression defined wholely in terms of $+$, $-$, $\cdot$, $\div$, $=^b$, *if-then-else*, constants, and the variable $x:\mathbb{Q}$. Then there are two polynomials $p(x)$ and $q(x)$ with rational coefficients such that $g(x)$ coincides with $p(x)\div q(x)$ almost everywhere. That is, $g(x)\neq p(x)\div q(x)$ for only finitely many $x\in\mathbb{Q}$.

Now $p(x)\div q(x)=1$ iff $p(x)-q(x)=0$. Since $p(x)-q(x)=0$ is a polynomial equation, it has finitely many roots. Hence $g(x)$ cannot coincide with $p(x)\div q(x)$ almost everywhere. Consequently, $g$ is not expressible. $\square$

Therefore, we propose to augment $\mathcal{NRC}(\mathbb{Q},+,\cdot,-,\div,\sum,=)$ with a linear order $\leq^b: b\times b\to b$ for each base type $b$. Many important data organization functions such as sorting algorithms and duplicate detection/elimination algorithms rely on linear orders. In the remainder of this section, we show how to lift linear order at base types to linear order at all types. First recall that the Hoare ordering $\sqsubseteq^\flat$ on the subsets of an ordered set is defined as $X\sqsubseteq^\flat Y$ iff for every $x\in X$ there is $y\in Y$ such that $x\sqsubseteq y$. Then

**Proposition 4.2** *Let $(D,\sqsubseteq)$ be a partially ordered set. Define an order $\lesssim^\flat$ on the finite subsets of $D$ as follows: $X\lesssim^\flat Y$ iff either $X\sqsubseteq^\flat Y$ and $Y\not\sqsubseteq^\flat X$, or $X\sqsubseteq^\flat Y$ and $Y\sqsubseteq^\flat X$ and $X-Y\sqsubseteq^\flat Y-X$. Then $\lesssim^\flat$ is a partial order. Moreover, if $\sqsubseteq$ is a linear order, then so is $\lesssim^\flat$.*

**Proof.** See Libkin and Wong [15]. $\square$

Kupert, Saake, and Wegner [14] gave three linear orderings on collection types in their study of duplicate detection and elimination. The ordering defined above coincides with one of them and is in fact a particular case of an order well known in universal algebra and combinatorics [13, 25]. An important feature of our technique of lifting linear orders is that the resulting linear orders are readily seen to be computable by our very limited language. Hence in the rest of the report, we assume that $\leq^s$, where $s$ is not a base type, is a syntactic sugar as implemented in the theorem below.

**Theorem 4.3 (Linear order)** *$\mathcal{NRC}(\mathbb{Q}, +, \cdot, -, \div, \sum, =)$ augmented with linear order $\leq^b$: $b \times b \to b$ at every base type $b$ can express a linear order $\leq^s$: $s \times s \to s$ at every type $s$.*

**Proof.** Proceed by induction on $s$.

- $\leq^b$ is the given linear order on base type $b$.
- $x \leq^{s \times t} y \triangleq$ *if* $\pi_1\ x \leq^s \pi_1\ y$ *then* (*if* $\pi_1\ x =^s \pi_1\ y$ *then* $\pi_2\ x \leq^t \pi_2\ y$ *else true*) *else false*
- $X \leq^{\{s\}} Y \triangleq$ *if* $X \sqsubseteq^\flat_s Y$ *then* (*if* $Y \sqsubseteq^\flat_s X$ *then* $X \lesssim^\flat_s Y$ *else true*) *else false*
- $X \sqsubseteq^\flat_s Y \triangleq (\sum\{\!|(\mathit{if}\ (\sum\{\!|(\mathit{if}\ x \leq^s y\ \mathit{then}\ 1\ \mathit{else}\ 0) \mid y \in Y|\!\}) = 0\ \mathit{then}\ 1\ \mathit{else}\ 0) \mid x \in X|\!\}) = 0$
- $X \lesssim^\flat Y \triangleq (\sum\{\!|\mathit{if}\ x \in^s Y\ \mathit{then}\ 0\ \mathit{else}\ (\mathit{if}\ (\sum\{\!|\mathit{if}\ y \in^s X\ \mathit{then}\ 0\ \mathit{else}\ (\mathit{if}\ x \leq^s y\ \mathit{then}\ 1\ \mathit{else}\ 0) \mid y \in Y|\!\}) = 0\ \mathit{then}\ 1\ \mathit{else}\ 0) \mid x \in X|\!\}) = 0.$ □

Hence we denote the language endowed with linear order at base types by $\mathcal{NRC}(\mathbb{Q}, +, \cdot, -, \div, \sum, =, \leq)$. Several other queries commonly encountered in practical database environments, as well as some unusual ones, are now easily expressed:

- "Rows of $R$ whose first column value is the maximum of the column" is $\mathit{maxrows}(R) \triangleq \bigcup\{\mathit{if}\ (\sum\{\!|\mathit{if}\ \pi_1(x) = \pi_1(y)\ \mathit{then}\ 0\ \mathit{else}\ \mathit{if}\ \pi_1(y) \leq \pi_1(x)\ \mathit{then}\ 1\ \mathit{else}\ 0 \mid x \in R|\!\} = 0)\ \mathit{then}\ \{y\}\ \mathit{else}\ \{\} \mid y \in R\}$.
- "Rows of $R$ whose first column value is the mode of the column" is $\mathit{moderows}(R) \triangleq \mathit{maxrows}(\bigcup\{\{(\ \sum\{\!|\mathit{if}\ f(y) = f(x)\ \mathit{then}\ 1\ \mathit{else}\ 0 \mid y \in R|\!\},\ x)\} \mid x \in R\})$.
- "Parity of the cardinality of a set $R$" is $\mathit{odd}(R) \triangleq \bigcup\{\mathit{if}\ \sum\{\!|\mathit{if}\ x \leq y\ \mathit{then}\ 1\ \mathit{else}\ 0 \mid y \in R|\!\} = \sum\{\!|\mathit{if}\ y \leq x\ \mathit{then}\ 1\ \mathit{else}\ 0 \mid y \in R|\!\}\ \mathit{then}\{()\}\ \mathit{else}\ \{\} \mid x \in R\} = \{()\}$.

More significantly, the rank assignment function can be expressed. The rank assignment function leads to a few rather surprising results to be discussed shortly.

**Proposition 4.4** *A rank assignment $\mathit{sort}^s : \{s\} \to \{s \times \mathbb{Q}\}$ is the function such that $\mathit{sort}\{o_1, \ldots, o_n\} = \{(o_1, 1), \ldots, (o_n, n)\}$ where $o_1 < \ldots < o_n$. $\mathcal{NRC}(\mathbb{Q}, +, \cdot, -, \div, \sum, =, \leq)$ can define $\mathit{sort}^s$.*

**Proof.** $\mathit{sort}(R) \triangleq \bigcup\{\{(x,\ \sum\{\!|\mathit{if}\ y \leq x\ \mathit{then}\ 1\ \mathit{else}\ 0 \mid y \in R|\!\})\} \mid x \in R\}$. □

# 5 More conservative extension results

The ability to compute a linear order and a rank assignment function at every type proves to be an asset. In this final section, we present a few more conservative extension results. First, let us consider the following primitives:

$$\frac{g : \{s\} \quad f : \{s\} \to \{s\}}{\mathit{bfix}^s(f, g) : \{s\}}$$

$$\frac{}{tc^s : \{s \times s\} \to \{s \times s\}} \qquad \frac{}{\mathit{powerset}^s : \{s\} \to \{\{s\}\}}$$

where $tc(R)$ is the transitive closure of $R$; $\mathit{bfix}(f, g)$ is the bounded fixpoint of $f$ with respect to $g$; that is, it is the least fixpoint of the equation $f(R) = g \cap (R \cup f(R))$; and $\mathit{powerset}(R)$ is the powerset of $R$.

**Corollary 5.1** *The followings have the conservative extension property:*

- $\mathcal{NRC}(\mathbb{Q}, +, \cdot, -, \div, \sum, =, \leq, tc)$ *with fixed constant* $1$.
- $\mathcal{NRC}(\mathbb{Q}, +, \cdot, -, \div, \sum, =, \leq, \mathit{bfix})$ *with fixed constant* $1$.
- $\mathcal{NRC}(\mathbb{Q}, +, \cdot, -, \div, \sum, =, \leq, \mathit{powerset})$ *with fixed constant* $2$.

**Proof.** We provide the proof for the first one, the other two are straightforward adaptation of the same technique. First observe that $\mathcal{NRC}(\mathbb{Q}, +, \cdot, -, \div, \sum, =, \leq, tc^{\mathbb{Q}})$, where we restrict computation of transitive closure to binary relations of rational numbers, has the conservative extension property with constant 1. Therefore, it suffices for us to show that $tc^s$ is expressible in it for any $s$. This can be achieved by exploiting the rank assignment function *sort* by defining

- $tc(R) \triangleq \mathit{decode}(tc^{\mathbb{Q}}(\mathit{encode}(R, \mathit{sort}(\mathit{dom}(R)))), \mathit{sort}(\mathit{dom}(R)))$, where
- $\mathit{dom}(R) \triangleq \bigcup\{\{\pi_1\ x\} \mid x \in R\} \cup \bigcup\{\{\pi_2\ x\} \mid x \in R\}$,
- $\mathit{encode}(R, C) \triangleq \bigcup\{\bigcup\{\bigcup\{\mathit{if}\ \pi_1\ x = \pi_1\ y\ \mathit{then}\ \mathit{if}\ \pi_2\ x = \pi_1\ z\ \mathit{then}\ \{(\pi_2\ y, \pi_2\ z)\}\ \mathit{else}\ \{\}\ \mathit{else}\ \{\} \mid z \in C\} \mid y \in C\} \mid x \in R\}$, and
- $\mathit{decode}(R, C) \triangleq \bigcup\{\bigcup\{\bigcup\{\mathit{if}\ \pi_1\ x = \pi_2\ y\ \mathit{then}\ \mathit{if}\ \pi_2\ x = \pi_2\ z\ \mathit{then}\ \{(\pi_1\ y, \pi_1\ z)\}\ \mathit{else}\ \{\}\ \mathit{else}\ \{\} \mid z \in C\} \mid y \in C\} \mid x \in R\}$. □

Conservativity of $\mathcal{NRC}(=, \mathit{powerset})$ was considered by Hull and Su [7] and Grumbach and Vianu [6]. The former showed that $\mathcal{NRC}(=, \mathit{powerset})_{i,o,h} \neq \mathcal{NRC}(=, \mathit{powerset})_{i,o,h+1}$ for any $h$ and $i = o = 1$. This implies the failure of conservative extension for $\mathcal{NRC}(=, \mathit{powerset})$ with respect to flat relations. The latter generalized this result to any $i$ and $o$. The corollary above showed that the failure at higher heights can be repaired by augmenting $\mathcal{NRC}(=, \mathit{powerset})$ with a summation operator.

More recently, Suciu [21] showed, using a technique related to that of Van den Bussche [5], that $\mathcal{NRC}(=, \mathit{bfix})_{i,o,h} = \mathcal{NRC}(=, \mathit{bfix})_{i,o,h+1}$ for $i = o = 1$. This is remarkable because he did not need any arithmetic operation. The corollary above showed that the conservativity of bounded fixpoint can be extended to all input and output in the presence of arithmetics.

Immerman [8] showed that first-order logic with least fixpoint and order is equivalent to PTIME. This may imply $\mathcal{NRC}(\mathbb{Q},+,\cdot,-,\div,\sum,=,\leq,\mathit{lfp})_{1,1,h} = \mathcal{NRC}(\mathbb{Q},+,\cdot,-,\div,\sum,=,\leq,\mathit{lfp})_{1,1,h+1}$. In which case, $\mathcal{NRC}(\mathbb{Q},+,\cdot,-,\div,\sum,=,\leq,\mathit{lfp})$ is conservative over flat relations. This should be contrasted with the corollary above. The languages in the corollary do not necessarily give us all PTIME queries over flat relations. Furthermore, conservativity holds for them over any input and output.

The technique used in our proof of conservative extension has an intrinsic uniformity. To illustrate this, let us introduce three partially interpreted primitives $\iota$, $\odot$ and $\prod$ to $\mathcal{NRC}(\mathbb{Q},+,\cdot,-,\div,\sum,=,\leq)$,

$$\frac{}{\iota : b} \qquad \frac{e_1 : b \quad e_2 : b}{e_1 \odot e_2 : b} \qquad \frac{e_1 : b \quad e_2 : \{s\}}{\prod\{\!|e_1 \mid x^s \in e_2|\!\} : b}$$

where $b$ is some fixed type, $\odot : b \times b \to b$ is a commutative associative binary operation, $\iota : b$ is the identity for $\odot$, and $\prod\{\!|e \mid x^s \in \{o_1,\ldots,o_n\}|\!\} = e[o_1/x^s] \odot \ldots \odot e[o_n/x^s] \odot \iota$ for any set $\{o_1,\ldots,o_n\}$, with $o_1$, ..., $o_n$ all distinct, of type $\{s\}$. As an example, take $\odot$ to be $\cdot$ and $b$ to be $\mathbb{Q}$, then $\iota$ becomes 1 and $\prod$ becomes a sort of bounded product.

**Proposition 5.2** *For every $i$, $o$, and $h \geq \max(i,o,ht(b))$, $\mathcal{NRC}(\mathbb{B},\mathbb{Q},+,\cdot,-,\div,\sum,=,\leq,\odot,\prod,\iota)_{i,o,h} = \mathcal{NRC}(\mathbb{B},\mathbb{Q},+,\cdot,-,\div,\sum,=,\leq,\odot,\prod,\iota)_{i,o,h+1}$.*

**Proof.** It suffices to append the rules below to the rewrite system of section 3. Note the use of the linear ordering $\leq$. (If $\odot$ is also idempotent, simpler rules can be used.)

- $\prod\{\!|e \mid x \in \{\}|\!\} \leadsto \iota$
- $\prod\{\!|e \mid x \in \{e'\}|\!\} \leadsto e[e'/x]$
- $\prod\{\!|e \mid x \in e_1 \cup e_2|\!\} \leadsto \prod\{\!|e \mid x \in e_1|\!\} \odot \prod\{\!|\mathit{if}\ x \in e_1\ \mathit{then}\ \iota\ \mathit{else}\ e \mid x \in e_2|\!\}$
- $\prod\{\!|e \mid x \in \mathit{if}\ e_1\ \mathit{then}\ e_2\ \mathit{else}\ e_3|\!\}$*newline* $\leadsto \mathit{if}\ e_1\ \mathit{then}\ \prod\{\!|e \mid x \in e_2|\!\}\ \mathit{else}\ \prod\{\!|e \mid x \in e_3|\!\}$
- $\prod\{\!|e \mid x \in \bigcup\{e_1 \mid y \in e_2\}|\!\} \leadsto \prod\{\!|\prod\{\!|\mathit{if}\ (\sum\{\!|\mathit{if}\ x \in e_1[w/y]\ \mathit{then}\ (\mathit{if}\ w = y\ \mathit{then}\ 0\ \mathit{else}\ (\mathit{if}\ w \leq y\ \mathit{then}\ 1\ \mathit{else}\ 0))\ \mathit{else}\ 0 \mid w \in e_2|\!\}) = 0\ \mathit{then}\ e\ \mathit{else}\ \iota \mid x \in e_1|\!\} \mid y \in e_2|\!\}$. □

# 6 Conclusion and future work.

The conservative extension property of nested relational calculi is studied in the presence of aggregate functions and linear orders. We showed that this property is retained by the nested relational calculus $\mathcal{NRC}(=)$ when very simple arithmetics and a summation operator are added to the language. We proved also that the presence of linear orders at base types leads to a more uniform and perhaps unexpected demonstration of the conservative extension property of several nested relational calculi. In particular, the well-known failure of conservativity of $\mathcal{NRC}(=, powerset)$ is shown to be repairable at higher heights when very simple arithmetics, bounded summation, and linear orders are available. These results have many consequences, including an interesting finite-cofiniteness property of the bag query language of Libkin and Wong [15]; we hope to present them in detail in a future report.

It is known that the presence of a linear order adds power to first-order query languages [8, 23]. Our nested set language has enough power to express a linear order at all types. It is a good framework for investigating the impact of linear orders on nested collections. Also, other kinds of linear orders on nested collections such as those in [14] should be studied.

We were able to demonstrate the conservative extension property for the nested set language with aggregate functions and additional primitives such as transitive closure, bounded fixpoint and powerset by reducing these primitives to the corresponding ones on rational numbers. What is the general property of these primitives that allowed this reduction?

The nested relational language with summation seems to be adequate for statistical databases. Does it have sufficient expressive power for querying databases for other advanced applications such as spatial databases, geographic databases, and genome databases?

**Acknowledgements.** Discussions with Peter Buneman, Val Breazu-Tannen, and especially Dan Suciu directly resulted in this paper. We thank them for their encouragement and insights. We are also grateful to Anthony Kosky and Paula Ta-Shma for their valuable comments. Support for Leonid Libkin is provided in part by National Science Foundation Grant IRI-90-04137 and a AT&T Doctoral Fellowship. Support for Limsoon Wong is provided in part by National Science Foundation Grant IRI-90-04137 and Army Research Office Grant DAAL03-89-C-0031-PRIME.

# References

[1] V. Breazu-Tannen, P. Buneman, and S. Naqvi. Structural recursion as a query language. In *Proceedings of 3rd International Workshop on Database Programming Languages, Naphlion, Greece*, pages 9–19. Morgan Kaufmann, August 1991.

[2] V. Breazu-Tannen and R. Subrahmanyam. Logical and computational aspects of programming with Sets/Bags/Lists. In *LNCS 510: Proceedings of 18th International Colloquium on Automata, Languages, and Programming, Madrid, Spain, July 1991*, pages 60–75. Springer Verlag, 1991.

[3] Val Breazu-Tannen, Peter Buneman, and Limsoon Wong. Naturally embedded query languages. In *LNCS 646: Proceedings of International Conference on Database Theory, Berlin, Germany, October, 1992*, pages 140–154. Springer-Verlag, October 1992.

[4] Latha S. Colby. A recursive algebra for nested relations. *Information Systems*, 15(5):567–582, 1990.

[5] Jan Van den Bussche. Complex object manipulation through identifiers: An algebraic perspective. technical Report 92-41, University of Antwerp, Department of Mathematics and Computer Science, Universiteitsplein 1, B-2610 Antwerp, Belgium, September 1992.

[6] Stephane Grumbach and Victor Vianu. Playing games with objects. In *LNCS 470: 3rd International Conference on Database Theory, Paris, France, December 1990*, pages 25–39. Springer-Verlag, 1990.

[7] Richard Hull and Jianwen Su. On the expressive power of database queries with intermediate types. *Journal of Computer and System Sciences*, 43:219–267, 1991.

[8] Neil Immerman. Relational queries computable in polynomial time. *Information and Control*, 68:86–104, 1986.

[9] G. Jaeschke and H. J. Schek. Remarks on the algebra of nonfirst normal form relations. In *Proceedings ACM Symposium on Principles of Database Systems*, pages 124–138, Los Angeles, California, March 1982.

[10] L. A. Jategaonkar and J. C. Mitchell. ML with extended pattern matching and subtypes. In *Proceedings of ACM Conference on LISP and Functional Programming*, pages 198–211, Snowbird, Utah, July 1988.

[11] Aviel Klausner and Nathan Goodman. Multirelations: Semantics and languages. In *Proceedings of 11th International Conference on Very Large Databases, Stockholm, August 1985*, pages 251–258, Los Altos, CA, August 1985. Morgan Kaufmann.

[12] Anthony Klug. Equivalence of relational algebra and relational calculus query languages having aggregate functions. *Journal of the ACM*, 29(3):699–717, July 1982.

[13] J. B. Kruskal. The theory of well-quasi-ordering: A frequently discovered concept. *Journal of Combinatorial Theory Series A*, 13:297–305, 1972.

[14] K. Kupert, G. Saake, and L. Wegner. Duplicate detection and deletion in the extended $\mathrm{NF}^2$ data model. In *LNCS 367: Foundation of Data Organization and Algorithms*, pages 83–101. Springer-Verlag, June 1989.

[15] Leonid Libkin and Limsoon Wong. Query languages for bags. Technical Report MS-CIS-93-36/L&C 59, University of Pennsylvania, Philadelphia, PA 19104, March 1993.

[16] Eugenio Moggi. Notions of computation and monads. *Information and Computation*, 93:55–92, 1991.

[17] A. Ohori, P. Buneman, and V. Breazu-Tannen. Database programming in Machiavelli: A polymorphic language with static type inference. In *Proceedings of ACM International Conference on Management of Data*, pages 46–57, Portland, Oregon, June 1989.

[18] G. Ozsoyoglu, Z. M. Ozsoyoglu, and V. Matos. Extending relational algebra and relational calculus with set-valued attributes and aggregate functions. *ACM Transactions on Database Systems*, 12(4):566–592, December 1987.

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

[20] H.-J. Schek and M. H. Scholl. The relational model with relation-valued attributes. *Information Systems*, 11(2):137–147, 1986.

[21] Dan Suciu. Fixpoints and bounded fixpoints for complex objects. Technical Report MS-CIS-93-32/L&C 58, University of Pennsylvania, Philadelphia, PA 19104, March 1993.

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

[23] M. Y. Vardi. The complexity of relational query languages. In *Proceedings of 14th ACM Symposium on Theory of Computing*, pages 137–146, 1982.

[24] Philip Wadler. Comprehending monads. In *Proceedings of ACM Conference on Lisp and Functional Programming*, Nice, June 1990.

[25] W. Wechler. *Universal Algebra for Computer Scientists*, volume 25 of *EATCS Monograph on Theoretical Computer Science*. Springer-Verlag, Berlin, 1992.

[26] Limsoon Wong. Normal forms and conservative properties for query languages over collection types. In *Proceedings of 12th ACM Symposium on Principles of Database Systems*, pages 26–36, Washington, D. C., May 1993.

[27] Limsoon Wong. Query languages over collection types. Manuscript available from Limsoon@Saul.CIS.UPenn.EDU, June 1993.