# On the complexity of division and set joins in the relational algebra

Dirk Leinders [a,*] Jan Van den Bussche [a]

[a] *Hasselt University and transnational University of Limburg, Agoralaan, gebouw D, 3590 Diepenbeek, Belgium*

**Abstract**

We show that any expression of the relational division operator in the relational algebra with union, difference, projection, selection, constant-tagging, and joins, must produce intermediate results of quadratic size. To prove this result, we show a dichotomy theorem about intermediate sizes of relational algebra expressions (they are either all linear, or at least one is quadratic), and we link linear relational algebra expressions to expressions using only semijoins instead of joins.

*Key words:* database, relational algebra, semijoin algebra, complexity

## 1 Introduction

Relational division, first identified by Codd [7] , is the prototypical example of a "set join". Set joins relate database elements on the basis of sets of values, rather than single values as in a standard natural join. Thus, the division $R(A, B) \div S(B)$ returns all $A$'s for which the set of $B$'s related to $A$ by $R$ contains the set $S$. There is also a variant of division, where the set of $B$'s must equal the set $S$. More generally, one has the *set-containment join* $R \underset{B \supseteq D}{\bowtie} S$ of $R(A, B)$ and $S(C, D)$, which returns

$$\Big\{(a, c) \mid \{b \mid R(a, b)\} \supseteq \{d \mid S(c, d)\}\Big\},$$

and again the analogous *set-equality join*. In principle, any other predicate on sets could as well be used in the place of $\supseteq$ or $=$ [18,19]. Note that a set join

* Corresponding author.
*Email addresses:* `dirk.leinders@uhasselt.be` (Dirk Leinders), `jan.vandenbussche@uhasselt.be` (Jan Van den Bussche).

Person

| pName | Symptom |
|---|---|
| An | headache |
| An | sore throat |
| An | neck pain |
| Bob | headache |
| Bob | sore throat |
| Bob | memory loss |
| Bob | neck pain |
| Carol | headache |

Disease

| dName | Symptom |
|---|---|
| flu | headache |
| flu | sore throat |
| Lyme | headache |
| Lyme | sore throat |
| Lyme | memory loss |
| Lyme | neck pain |

Symptoms

| Symptom |
|---|
| headache |
| neck pain |

$\text{Person} \bowtie_{\text{Person.Symptom} \supseteq \text{Disease.Symptom}} \text{Disease}$

| pName | dName |
|---|---|
| An | flu |
| Bob | flu |
| Bob | Lyme |

$\text{Person} \div \text{Symptoms}$

| pName |
|---|
| An |
| Bob |

Fig. 1. An illustration of set-containment join and division.

with predicate "intersection nonempty" boils down to an ordinary equijoin! An illustration is given in Figure 1.

It has long been observed that division is not well handled by classical query processing [11,12]. Indeed, while set joins are expressible in the relational algebra using combinations of equijoins and difference operators, the resulting expressions tend to be complex and inefficient. In this paper, we will confirm this phenomenon mathematically. Specifically, working in the relational algebra with union, difference, projections, selections, constant-tagging, and joins (cartesian product being a special case), we prove that any expression for the division operator must produce intermediate results of quadratic size. (The result holds both for containment- and equality-division, and then of course also for the more general set joins.)

Our work thus provides a formal justification of work done by various authors on implementing set joins directly as special-purpose operators, or on implementing them by compiling to the more powerful version of the relational algebra that includes grouping, sorting, and aggregation operators [13,16,17]. For instance, division (and set-equality join) can be implemented efficiently in time $O(n \log n)$ using sorting or counting tricks. [1] Note, however, that for set-containment join, no algorithm that is better than quadratic is known.

We will actually prove a number of more general results about relational algebra expressions which we believe are interesting on their own, and from which the result about division follows. Specifically, we will show that any expression that never produces intermediate results of quadratic size, will produce only intermediate results of linear size. Moreover, we will characterize the class of queries expressible by these "linear" expressions as the class of queries expressible by the *semijoin algebra*: this is the variant of the relational algebra where we replace the join operator by the semijoin operator [5,6]. Semijoin algebra expressions are linear by definition, and thus our result shows that a semantical restriction of relational algebra expressions (namely, linear) can be captured by a syntactical restriction (namely, semijoin). Consequently, if a query is not expressible in the semijoin algebra, then its complexity in the relational algebra is at least quadratic. To prove our complexity result, we use an equivalence relation on structures, called guarded bisimilarity, that is known to guarantee indistinguishability in the "guarded" fragment of first-order logic [3,9,10,8]. This guarded fragment precisely corresponds to the semijoin algebra [14].

The paper is organised as follows. In Section 2 we recall the definitions and known results on the semijoin algebra and the guarded fragment. Section 3 states and proves our dichotomy theorem. In Section 4 we show how the dichotomy theorem can be applied to prove complexity lower bounds for division and set joins.

## 2 Semijoin algebra and guarded fragment

From the outset, we assume an infinite, totally ordered universe $\mathbb{U}$ of basic data values. Throughout the paper, we fix an arbitrary database schema $\mathbf{S}$. A database schema is a finite set of relation names, where each relation name $R$ has an associated arity, denoted by $\text{arity}(R)$. A database $D$ over $\mathbf{S}$ is an assignment of a finite relation $D(R) \subseteq \mathbb{U}^n$ to each $R \in \mathbf{S}$, where $n$ is the arity of $R$.

To avoid misunderstanding, we define the relational algebra, as we will use it, formally.

**Definition 1 (relational algebra, RA)** *The syntax and semantics of the relational algebra are inductively defined as follows:*

[1] For set-equality join, where the result size alone can already be quadratic, we should really say in time $O(n \log n)$ plus output size.

*(1)* *Each relation name* $R \in \mathbf{S}$ *is a relational algebra expression. Its arity comes from* $\mathbf{S}$*.*
*(2)* *If* $E_1, E_2 \in RA$ *have arity* $n$*, then also* $E_1 \cup E_2$ *(union),* $E_1 - E_2$ *(difference) belong to RA and are of arity* $n$*.*
*(3)* *If* $E \in RA$ *has arity* $n$ *and* $i_1, \ldots, i_k \in \{1, \ldots, n\}$*, then* $\pi_{i_1,\ldots,i_k}(E)$ *(projection) belongs to RA and is of arity* $k$*.*
*(4)* *If* $E \in RA$ *has arity* $n$ *and* $i, j \in \{1, \ldots, n\}$*, then* $\sigma_{i=j}(E)$ *and* $\sigma_{i<j}(E)$ *(selection) belong to RA and are of arity* $n$*.*
*(5)* *If* $E \in RA$ *has arity* $n$ *and* $c \in \mathbb{U}$*, then* $\tau_c(E)$ *(constant-tagging) belongs to RA and is of arity* $n+1$*.*
*(6)* *Let* $E_1, E_2 \in RA$ *with arities* $n$ *and* $m$*, respectively. Let* $\theta$ *be a conjunction of the form* $\bigwedge_{s=1}^{k} i_s \ \alpha_s \ j_s$ *with* $\alpha_s \in \{=, \neq, <, >\}$*, with* $i_s \in \{1, \ldots, n\}$*, and* $j_s \in \{1, \ldots, m\}$*. Then* $E_1 \bowtie_\theta E_2$ *(join) belongs to RA and is of arity* $n+m$*.*

*The semantics of the union and difference operators are the obvious set operators. The semantics of the projection, the selection, the constant-tagging and the join operator are as follows: (for relations* $r$*,* $r_1$ *and* $r_2$*)*

$$\begin{aligned}
&\pi_{i_1,\ldots,i_k}(r) := \{(a_{i_1}, \ldots, a_{i_k}) \mid \bar{a} \in r\} \\
&\sigma_{i=j}(r) := \{\bar{a} \in r \mid a_i = a_j\} \\
&\sigma_{i<j}(r) := \{\bar{a} \in r \mid a_i < a_j\} \\
&\tau_c(r) := \{(\bar{a}, c) \mid \bar{a} \in r\} \\
&r_1 \bowtie_\theta r_2 := \{(\bar{a}, \bar{b}) \mid \bar{a} \in r_1,\ \bar{b} \in r_2,\ \textit{and } a_{i_s}\ \alpha_s\ b_{j_s} \textit{ for } s = 1, \ldots, k\}
\end{aligned}$$

Note that selections of the form $\sigma_{i=\text{`}c\text{'}}(E)$, where $c \in \mathbb{U}$ and $E$ of arity $n$, can be expressed as $\pi_{1,\ldots,n}\Big(\sigma_{i=n+1}\tau_c(E)\Big)$.

We use $\mathrm{RA}^=$ to denote the variant of RA where only equijoins are allowed. More formally, in $\mathrm{RA}^=$, in every join condition $\theta$, every $\alpha_s$ is the symbol '='.

**Definition 2 (semijoin algebra, SA)** *The semijoin algebra is the variant of RA obtained by replacing the join operator* $E_1 \bowtie_\theta E_2$ *by the semijoin operator* $E_1 \ltimes_\theta E_2$*. The semantics of the semijoin operator is as follows: (for relations* $r_1$ *and* $r_2$*)*

$$r_1 \ltimes_\theta r_2 := \{\bar{a} \in r_1 \mid \exists \bar{b} \in r_2 : a_{i_s}\ \alpha_s\ b_{j_s} \textit{ for } s = 1, \ldots, k\}$$

*Again, we use* $SA^=$ *to denote the variant of SA where only equi-semijoins are allowed. So, in* $SA^=$*, in every semijoin condition* $\theta$*, every* $\alpha_s$ *is the symbol '='.*

**Example 3** *Suppose* $\mathbf{S}$ *is Ullman's well-known example schema [20]*

$$\{\textit{Likes(drinker,beer), Serves(bar,beer), Visits(drinker,bar)}\}.$$

$R$

| | | |
|---|---|---|
| a | b | c |
| d | e | f |

$S$

| | | |
|---|---|---|
| d | a | b |

$T$

| | |
|---|---|
| e | a |
| f | c |

Fig. 2. A database $D$ over the schema $\mathbf{S} = \{R, S, T\}$, where $R$ and $S$ are ternary and $T$ is binary, to illustrate the notion of "$C$-stored" tuple

*Let us call a bar lousy if it only serves beers nobody likes. The query that asks for the drinkers that visit a lousy bar can be expressed in SA as follows:*

$$\pi_1\Big(\text{Visits} \underset{2=1}{\ltimes} (\pi_1(\text{Serves}) - \pi_1(\text{Serves} \underset{2=2}{\ltimes} \text{Likes}))\Big).$$

*Note that this expression belongs to SA*$^=$. $\square$

If $C$ is a finite set of constants such that all constants in expression $E$ are in $C$, then we say that $E$ is an expression with constants in $C$. Note that SA expressions with constants in $C$ can only output "$C$-stored" tuples, defined as follows:

**Definition 4 ($C$-stored tuple)** *A tuple $\bar{d}$ is* $C$-stored *in database $D$ over schema* $\mathbf{S}$ *if the tuple obtained by deleting in $\bar{d}$ all values in $C$, belongs to some projection $\pi_{i_1,\ldots,i_p}(D(R))$ for some relation name $R$ in* $\mathbf{S}$.

**Example 5** *Let $D$ be the database over the schema $\mathbf{S} = \{R, S, T\}$ shown in Figure 2 and let $C$ be the singleton $\{a\}$. Tuple $(b, c)$ is $C$-stored in $D$, because $(b, c)$ is in projection $\pi_{2,3}(D(R))$; tuple $(a, f)$ is also $C$-stored in $D$, because the tuple obtained by deleting all $a$'s in $(a, f)$, i.e., $(f)$, is in $\pi_1(D(T))$. Tuples $(e, c)$ and $(g)$ are not $C$-stored in $D$.* $\square$

Next, we recall the definition of the guarded fragment of first-order logic [3,9,10,8]. When $\varphi$ stands for a formula, we follow the standard convention to write $\varphi(x_1, \ldots, x_k)$ to denote that every free variable of $\varphi$ is among $x_1, \ldots, x_k$.

**Definition 6 (guarded fragment, GF)** *(1) Atomic formulas of the form $x = y$ and $x < y$ and $x = c$, where $c \in \mathbb{U}$, are in GF.*

*(2) Relation atoms of the form $R(x_1, \ldots, x_k)$, with $R \in \mathbf{S}$ of arity $k$, are in GF.*

*(3) If $\varphi$ and $\psi$ are formulas of GF, then so are $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $\varphi \rightarrow \psi$ and $\varphi \leftrightarrow \psi$.*

*(4) If $\varphi(\bar{x}, \bar{y})$ is a formula of GF, and $\alpha(\bar{x}, \bar{y})$ is a relation atom such that all free variables of $\varphi$ do actually occur in $\alpha$, then $\exists\bar{y}(\alpha(\bar{x}, \bar{y}) \wedge \varphi(\bar{x}, \bar{y}))$ is a formula of GF.*

*The semantics of GF is that of first-order logic (or the relational calculus as we call it in database theory), interpreted over the active domain of the database [1].*

**Example 7** *The query from Example 3 can be expressed by the following GF formula* $\varphi(x)$*:*

$$\exists y\Big(\text{Visits}(x, y) \wedge \neg\exists z\,(\text{Serves}(y, z) \wedge \exists w\,\text{Likes}(w, z))\Big).$$

$\square$

There is a strong correspondence between SA$^=$ and GF: one can be translated into the other. The following theorem was proven in our previous work [14]:

**Theorem 8** *For every SA$^=$ expression $E$ of arity $k$, there exists a GF formula $\varphi_E(x_1, \ldots, x_k)$ such that for every database $D$,*

$$\{\bar{d} \in \mathbb{U}^k \mid D \models \varphi_E(\bar{d})\} = E(D)$$

*Conversely, for every GF formula $\varphi(x_1, \ldots, x_k)$ with constants in $C$, there exists an SA$^=$ expression $E_\varphi$ such that for every database $D$,*

$$E_\varphi(D) = \{\bar{d}\ C\text{-stored tuple in } D \mid D \models \varphi(\bar{d})\}$$

In our previous work [14], this correspondence between SA$^=$ and GF was proved for the setting without constants. Nevertheless, an easy adaptation of that proof shows that the correspondence still holds for the setting with constants of this paper.

The correspondence between SA$^=$ and GF is very useful because it allows us to apply the notion of "guarded bisimulation", originally developed in the context of GF, to SA$^=$. We recall the definition next.

**Definition 9 (guarded set)** *A set is* guarded *in database $D$ if it is of the form $\{d_1, \ldots, d_n\}$, where $(d_1, \ldots, d_n) \in D(R)$ for some $R \in \mathbf{S}$.*

**Definition 10 ($C$-partial isomorphism)** *Let $A$ and $B$ be databases over schema $\mathbf{S}$ and let $X, Y, C \subseteq \mathbb{U}$. A mapping $f : X \to Y$ is a $C$-*partial isomorphism *from $A$ to $B$ if it is bijective, and for each $R \in \mathbf{S}$, of arity $n$, and all $x_1, \ldots, x_n \in X$, we have $(x_1, \ldots, x_n) \in A(R) \Leftrightarrow (f(x_1), \ldots, f(x_n)) \in B(R)$, and moreover, for all $x, y \in X$ and for all $c \in C$, we have $x < y \Leftrightarrow f(x) < f(y)$ and $x = c \Leftrightarrow f(x) = c$.*

**Definition 11 ($C$-guarded bisimulation, $C$-guarded bisimilarity)** *A $C$-*guarded bisimulation *between two databases $A$ and $B$ is a non-empty set $\mathcal{I}$ of finite $C$-partial isomorphisms from $A$ to $B$, such that the following back and forth conditions are satisfied:*

***Forth.*** *For every $f : X \to Y$ in $\mathcal{I}$ and for every guarded set $X'$ of $A$, there*

$A$

$R$

| | |
|---|---|
| 1 | 2 |
| 2 | 3 |

$S$

| | |
|---|---|
| 1 | 2 |

$T$

| | |
|---|---|
| 2 | 3 |

$B$

$R$

| | |
|---|---|
| 6 | 7 |
| 7 | 8 |
| 9 | 10 |
| 10 | 11 |

$S$

| | |
|---|---|
| 6 | 7 |
| 9 | 10 |

$T$

| | |
|---|---|
| 7 | 8 |
| 10 | 11 |

Fig. 3. Databases $A$ and $B$ to illustrate the notion of guarded bisimulation.

*exists a partial isomorphism $g\colon X' \to Y'$ in $\mathcal{I}$ such that $f$ and $g$ agree on $X \cap X'$.*

***Back.*** *For every $f\colon X \to Y$ in $\mathcal{I}$ and for every guarded set $Y'$ of $B$, there exists a partial isomorphism $g\colon X' \to Y'$ in $\mathcal{I}$ such that $f^{-1}$ and $g^{-1}$ agree on $Y \cap Y'$.*

*Now let $C$ be a set of constants and let $A$ be a database and $\bar{a}$ a $C$-stored tuple in $A$, and let $B, \bar{b}$ be another such pair. We say that $A, \bar{a}$ and $B, \bar{b}$ are* $C$-guarded bisimilar*—denoted by $A, \bar{a} \sim_g^C B, \bar{b}$—if there exists a $C$-guarded bisimulation $\mathcal{I}$ between them that contains the partial isomorphism $\bar{a} \mapsto \bar{b}$.*

**Example 12** *Let $A$ and $B$ be the databases shown in Figure 2. Let $C$ be the empty set. The following set of $\varnothing$-partial isomorphisms is a $\varnothing$-guarded bisimulation between $A$ and $B$:*

$$
\begin{array}{ll}
(1,2) \mapsto (6,7) & \qquad (2,3) \mapsto (7,8) \\
(1,2) \mapsto (9,10) & \qquad (2,3) \mapsto (10,11)
\end{array}
$$

*Let us check the back property for one particular partial isomorphism $f\colon (1,2) \mapsto (6,7)$. We consider all guarded sets $Y'$ of $B$: if $Y'$ is $(6,7)$, we choose $g$ as $f$; if $Y'$ is $(9,10)$, we also choose $g$ as $f$ (the intersection of $Y$ and $Y'$ is empty, so any $g$ will do); if $Y'$ is $(7,8)$, we choose $(2,3) \mapsto (7,8)$ for $g$ (the intersection of $Y$ and $Y'$ is $\{7\}$ and $f^{-1}$ and $g^{-1}$ both map $7$ to $2$); finally, if $Y'$ is $(10,11)$, we choose $(2,3) \mapsto (10,11)$ for $g$ (the intersection of $Y$ and $Y'$ is $\{10\}$ and $f^{-1}$ and $g^{-1}$ both map $10$ to $2$). The other properties can be checked analogously.* $\square$

A basic fact about GF is that GF formulas can not distinguish between inputs that are guarded bisimilar [3]:

**Proposition 13** *The guarded fragment is invariant under guarded bisimulation. Formally, if $A, \bar{a} \sim_g^C B, \bar{b}$, then for any GF formula $\varphi(\bar{x})$ with constants in $C$ we have $A \models \varphi(\bar{a}) \quad \Leftrightarrow \quad B \models \varphi(\bar{b})$.*

Andréka et al. [3] proved this result for the setting without constants. Nevertheless, an easy adaptation of that proof shows that the result still holds for the setting with constants of this paper.

By Theorem 8 we obtain:

**Corollary 14** *If $A, \bar{a} \sim_g^C B, \bar{b}$, then for any SA$^=$ expression $E$ with constants in $C$ we have $\bar{a} \in E(A) \quad \Leftrightarrow \quad \bar{b} \in E(B)$.*

## 3 A dichotomy theorem

Before we can state the theorem we need precise definitions of what we mean by "linear" and "quadratic" expressions. Beware that "linear" is an upper-bound notion, while "quadratic" is a lower-bound notion.

**Definition 15** *The* size *of a relation is defined as its cardinality. The* size *of a database $D$, denoted by $|D|$, is the sum of the sizes of its relations.*

Using the familiar $O$ and $\Omega$ notation, we now define: [2]

**Definition 16** *For any RA expression $E$, define the function*

$$c(E) : \mathbb{N} \to \mathbb{N} : n \mapsto \max\{|E(D)| : |D| = n\}.$$

*Then $E$ is called*

- linear *if for each subexpression $E'$ of $E$, $c(E') = O(n)$;*
- quadratic *if for some subexpression $E'$ of $E$, $c(E') = \Omega(n^2)$.*

We will prove:

**Theorem 17** *Every RA expression is either linear or quadratic.*

In other words, intermediate complexities such as $O(n \log n)$ are not achievable in RA. Anyone who has played long enough with RA expressions will intuitively know that, but we have never seen a proof. Moreover, we also have the following variant:

**Theorem 18** *Every RA expression that is not quadratic, is equivalently expressible in SA$^=$.*

[2] For a function $f : \mathbb{N} \to \mathbb{N}$, recall that $f = O(n)$ if for some $c > 0$ and some $n_0$, $f(n) \leqslant cn$ for all $n \geqslant n_0$; and $f = \Omega(n^2)$ if for some $c > 0$, $f(n) \geqslant cn^2$ infinitely often [2].

Note that the equi-semijoin operator can be expressed in RA in a linear way; for example, if $R$ and $S$ have arity two, then

$$R \underset{2=1}{\ltimes} S = \pi_{1,2}(R \underset{2=1}{\bowtie} \pi_1(S)).$$

From the above theorems we therefore obtain:

**Corollary 19** *A query is expressible by a linear RA expression if and only if it is expressible by an $SA^{=}$ expression.*

We will prove Theorem 17 and 18 simultaneously. Our crucial lemma is Lemma 24. In order to state it, we need two definitions.

**Definition 20** *Let $E$ be an RA expression of the form $E_1 \bowtie_\theta E_2$. For $\alpha \in \{=, \neq, <, >\}$, we define $\theta^\alpha$ as the following conjunction*

$$\bigwedge_{\{s \in \{1,\dots,k\} \mid \alpha_s \textit{ is } \alpha\}} i_s \ \alpha \ j_s.$$

*We also view $\theta^\alpha$ as the set of pairs $\{(i_s, j_s) \mid \alpha_s \text{ is } \alpha,\ \ s = 1, \dots, k\}$. For $\ell = 1, 2$, the sets $\mathsf{constrained}_\ell(E)$ and their complements $\mathsf{unc}_\ell(E)$ are now defined as follows:*

$$\begin{aligned}
&\mathsf{constrained}_1(E) := \{i \mid \exists j : (i, j) \in \theta^{=}\}\\
&\mathsf{unc}_1(E) := \{1, \dots, \mathit{arity}(E_1)\} - \mathsf{constrained}_1(E)\\
&\mathsf{constrained}_2(E) := \{j \mid \exists i : (i, j) \in \theta^{=}\}\\
&\mathsf{unc}_2(E) := \{1, \dots, \mathit{arity}(E_2)\} - \mathsf{constrained}_2(E)
\end{aligned}$$

**Example 21** *For the expression $E = R \bowtie_{3=1} S$, where $R$ and $S$ are ternary, we get:*

$$\begin{array}{ll}
\theta^{=} = \{(3, 1)\} & \\
\mathsf{constrained}_1(E) = \{3\} & \mathsf{unc}_1(E) = \{1, 2\}\\
\mathsf{constrained}_2(E) = \{1\} & \mathsf{unc}_2(E) = \{2, 3\}.
\end{array}$$

$\square$

In the next definition and in the proof of Theorem 17 and Theorem 18, we will use intervals. For $a, b \in \mathbb{U}$, recall the interval notation $[a, b]$ for the set $\{x \in \mathbb{U} \mid a \leqslant x \leqslant b\}$.

**Definition 22** *Let $D$ be a database and let $E$ be an RA expression of the form $E_1 \bowtie_\theta E_2$ with constants in $C$. We assume that $C = \{c_1, \dots, c_k\}$ with $c_1 < \cdots < c_k$. For any $\bar{d} \in E_1(D)$, we denote the set of elements occurring in*

*$\bar{d}$ by* $\mathsf{set}(\bar{d})$*. We now define the set of* free values of *$\bar{d}$ as follows:*

$$F_1^E(\bar{d}) := \mathsf{set}(\bar{d}) - \{d_i \mid i \in \mathit{constrained}_1(E)\} - C - \bigcup_{\substack{i\in\{1,\dots,k-1\}\\ [c_i,c_{i+1}]\ \mathit{finite}}} [c_i, c_{i+1}]$$

*The set $F_2^E(\bar{d})$ of free values of a tuple $\bar{d} \in E_2(D)$ is defined analogously.*

**Example 23** *Let $\mathbb{U}$ be $\mathbb{Z}$. Consider expression $E = \sigma_{2=`2'}R \bowtie_{3=1} \sigma_{3=`5'}S$, where $R$ and $S$ are ternary. So, $C$ equals $\{2, 5\}$. Suppose that relation $R$ contains the tuples $r_1 = (1, 2, 3)$ and $r_2 = (4, 6, 3)$, and that relation $S$ contains the tuples $s_1 = (3, 5, 6)$ and $s_2 = (1, 1, 1)$. Then:*

$$F_1^E(r_1) = \{1\} \qquad F_2^E(s_1) = \{6\}$$
$$F_1^E(r_2) = \{6\} \qquad F_2^E(s_2) = \varnothing$$

□

We can now state the following crucial lemma:

**Lemma 24** *Let $E = E_1 \bowtie_\theta E_2$ with constants in $C$ and where $E_1$ and $E_2$ are $SA^=$ expressions. Assume there exists a database $D$ and a tuple $(\bar{a}, \bar{b}) \in E_1 \bowtie_\theta E_2(D)$ such that $F_1^E(\bar{a}) \neq \varnothing$ and $F_2^E(\bar{b}) \neq \varnothing$. Then there exists a sequence $(D_n)_{n\geqslant 1}$ of databases such that for some constant $c > 0$ and for all $n$:*

*(1) $|D_n| \leqslant cn$, and*
*(2) $|E_1 \bowtie_\theta E_2(D_n)| \geqslant n^2$.*

Before we prove this lemma, we define the notion of "tuple space" used in the proof.

**Definition 25** *Let $D$ be a database over database schema $\mathbf{S}$. The tuple space $T_D$ of database $D$ is defined as $\bigcup\{D(R) \mid R \in \mathbf{S}\}$.*

From the definition of guarded set, it is clear that for each tuple $\bar{d} \in T_D$, $\mathsf{set}(\bar{d})$ is guarded and conversely, for each guarded set $X$ there is a tuple $\bar{d} \in T_D$ with $\mathsf{set}(\bar{d}) = X$.

**PROOF.** We give a proof by construction.

The desired sequence is constructed as follows. For $D_1$ we take $D$. For $k \geqslant 1$, we construct $D_{k+1}$ from $D_k$ as follows:

(1) for each $x \in F_1^E(\bar{a})$ and for each $x \in F_2^E(\bar{b})$, we make a fresh new domain

element $\mathsf{new}^{(k)}(x)$ that has the same relative order in the domain as $x$; if it is not possible to create such a new domain element, we create an isomorphic copy $D'_k$ of $D_k$ such that for any two values $r$, $s$ in $D'_k$ with $r < x < s$, there exists $u \in \mathbb{U}$ different from $x$ such that $r < u < s$. This is possible because to the left of the minimum of $C$, we can translate all elements in $D_k$. Similarly for the elements in $D_k$ to the right of the maximum of $C$, and similarly for the elements in $D_k$ in an infinite interval $[c_i, c_{i+1}]$. So, we assume w.l.o.g. that we can always create these new domain elements satisfying the specified condition;

(2) for each tuple $\bar{t} = (t_1, \ldots, t_n) \in T_D$ satisfying $\mathsf{set}(\bar{t}) \cap F_1^E(\bar{a}) \neq \varnothing$, we construct a tuple $f_1^{(k)}(\bar{t}) = (r_1, \ldots, r_n)$ with

$$r_i = \begin{cases} \mathsf{new}^{(k)}(t_i) & \text{if } t_i \in F_1^E(\bar{a}) \\ t_i & \text{else} \end{cases}$$

We put this tuple in precisely the same relations as $\bar{t}$. Note that by construction $\bar{t} \mapsto f_1^{(k)}(\bar{t})$ is a $C$-partial isomorphism.

(3) for each tuple $\bar{t} = (t_1, \ldots, t_n) \in T_D$ satisfying $\mathsf{set}(\bar{t}) \cap F_2^E(\bar{b}) \neq \varnothing$, we construct a tuple $f_2^{(k)}(\bar{t}) = (r_1, \ldots, r_n)$ with

$$r_i = \begin{cases} \mathsf{new}^{(k)}(t_i) & \text{if } t_i \in F_2^E(\bar{b}) \\ t_i & \text{else} \end{cases}$$

We put this tuple in precisely the same relations as $\bar{t}$. Note that by construction $\bar{t} \mapsto f_2^{(k)}(\bar{t})$ is a $C$-partial isomorphism.

To illustrate this construction, let database $D$ be the one shown in the upper part of Figure 4 and let expression $E$ be $(R \ltimes_{1=2} T) \bowtie_{3=1} (S \ltimes_{2=1} T)$. Let $\bar{a}$ be $(1, 2, 3)$ and let $\bar{b}$ be $(3, 4, 5)$. Then, $F_1^E(\bar{a}) = \{1, 2\}$ and $F_2^E(\bar{b}) = \{4, 5\}$. For each $i \in F_1^E(\bar{a}) \cup F_2^E(\bar{b})$, we denote $\mathsf{new}^{(1)}(i)$ by $i'$ and $\mathsf{new}^{(2)}(i)$ by $i''$. We assume the following order on the domain of $D_3$: $1 < 1' < 1'' < 2 < 2' < 2'' < 3 < \ldots < 9 < 10$. Databases $D_2$ and $D_3$ are shown in the lower part of Figure 4.

Now take $c := 2|D|$. Because in each step at most $2|D|$ tuples are added, the first requirement for the sequence holds.

We now check the second requirement. First, we show that for each $n$ and $k$ with $1 \leqslant k \leqslant n - 1$

$$D, \bar{a} \ \sim_g^C \ D_n, f_1^{(k)}(\bar{a})$$

Take an arbitrary $n$ and consider the set $\mathcal{I} = \{g_{\bar{t}}^{(k)} \mid \bar{t} \in T_D \text{ with } \mathsf{set}(\bar{t}) \cap F_1^E(\bar{a}) \neq \varnothing, 1 \leqslant k \leqslant n - 1\} \cup \{h_{\bar{t}} \mid \bar{t} \in T_D\}$, where

- $g_{\bar{t}}^{(k)} : \bar{t} \mapsto f_1^{(k)}(\bar{t})$, and

$D$

$R$

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 8 | 9 | 10 |

$S$

| | | |
|---|---|---|
| 3 | 4 | 5 |

$T$

| | |
|---|---|
| 6 | 1 |
| 4 | 7 |

$D_2$

$R$

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 8 | 9 | 10 |
| $1'$ | $2'$ | 3 |

$S$

| | | |
|---|---|---|
| 3 | 4 | 5 |
| 3 | $4'$ | $5'$ |

$T$

| | |
|---|---|
| 6 | 1 |
| 4 | 7 |
| 6 | $1'$ |
| $4'$ | 7 |

$D_3$

$R$

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 8 | 9 | 10 |
| $1'$ | $2'$ | 3 |
| $1''$ | $2''$ | 3 |

$S$

| | | |
|---|---|---|
| 3 | 4 | 5 |
| 3 | $4'$ | $5'$ |
| 3 | $4''$ | $5''$ |

$T$

| | |
|---|---|
| 6 | 1 |
| 4 | 7 |
| 6 | $1'$ |
| $4'$ | 7 |
| 6 | $1''$ |
| $4''$ | 7 |

Fig. 4. Databases $D = D_1$, $D_2$ and $D_3$ in the construction for $E = (R \ltimes_{1=2} T) \bowtie_{3=1} (S \ltimes_{2=1} T)$.

- $h_{\bar{t}} : \bar{t} \mapsto \bar{t}$.

In our running example, $\mathcal{I} = \{(1,2,3) \mapsto (1',2',3),\ (1,2,3) \mapsto (1'',2'',3),\ (3,4,5) \mapsto (3,4',5'),\ (3,4,5) \mapsto (3,4'',5''),\ (6,1) \mapsto (6,1'),\ (6,1) \mapsto (6,1''),\ (7,4) \mapsto (7,4'),\ (7,4) \mapsto (7,4'')\} \cup \{(1,2,3) \mapsto (1,2,3),\ (3,4,5) \mapsto (3,4,5),\ (6,1) \mapsto (6,1),\ (7,4) \mapsto (7,4),\ (8,9,10) \mapsto (8,9,10)\}$.

From the construction it follows that each of these functions is a $C$-partial isomorphism between $D$ and $D_n$. Now we check the back and forth properties of $\mathcal{I}$.

**Forth.** Take an arbitrary partial isomorphism $f$ in $\mathcal{I}$ and an arbitrary guarded set $X'$ in $D$. Let $\bar{t}'$ be a tuple in $T_D$ such that $\mathsf{set}(\bar{t}') = X'$. Suppose $f$ is $g_{\bar{t}}^{(k)}$ for some $\bar{t}$ and $k$. We distinguish 2 cases: *i)* $X' \cap F_1^E(\bar{a}) \neq \varnothing$. Then, $f$ agrees with partial isomorphism $g_{\bar{t}'}^{(k)}$ on $\mathsf{set}(\bar{t}) \cap X'$. Indeed, they both map values $x \in F_1^E(\bar{a})$ onto $\mathsf{new}^{(k)}(x)$ and they map values $y \notin F_1^E(\bar{a})$ onto $y$. *ii)* $X' \cap F_1^E(\bar{a}) = \varnothing$. Then, $f$ agrees with $h_{\bar{t}'}$ on $\mathsf{set}(\bar{t}) \cap X'$. When $f$ is $h_{\bar{t}}$ for some $\bar{t}$, $f$ clearly agrees with $h_{\bar{t}'}$ on $\mathsf{set}(\bar{t}) \cap X'$.

**Back.** Take an arbitrary partial isomorphism $f$ in $\mathcal{I}$ and an arbitrary guarded set $Y'$ in $D_n$. We distinguish 2 cases: *i)* $Y' = \mathsf{set}(f_1^{(l)}(\bar{u}))$ for some $1 \leqslant l \leqslant n-1$ and $\bar{u} \in T_D$; and *ii)* $Y' = \mathsf{set}(\bar{t}')$ for some $\bar{t}' \in T_D \cap T_{D_n}$. In case *i)*, $f^{-1}$ agrees with $(g_{\bar{u}}^{(l)})^{-1}$ on $\mathsf{set}(f(\bar{t})) \cap Y'$. In case *ii)*, $f^{-1}$ agrees with $(h_{\bar{t}'})^{-1}$ on $\mathsf{set}(f(\bar{t})) \cap Y'$.

Furthermore, for each $1 \leqslant k \leqslant n-1$, $\bar{a} \mapsto f_1^{(k)}(\bar{a})$ is an element of $\mathcal{I}$. A similar argument leads to

$$D, \bar{b} \ \sim_g^C \ D_n, f_2^{(k)}(\bar{b})$$

for each $1 \leqslant k \leqslant n-1$.

By Corollary 14 we have that for each $0 \leqslant k, l \leqslant n-1$: $f_1^{(k)}(\bar{a}) \in E_1(D_n)$ and $f_2^{(k)}(\bar{b}) \in E_2(D_n)$, where for simplicity we define $f_1^{(0)}$ and $f_2^{(0)}$ as the identity function.

In our running example, only $(1,2,3)$ satisfies $R \ltimes_{1=2} T$ in $D$, but in $D_3$ also $(1',2',3)$ and $(1'',2'',3)$ satisfy this expression; also in $D_3$ the tuples $(3,4,5)$, $(3,4',5')$ and $(3,4'',5'')$ satisfy $S \ltimes_{2=1} T$.

We now show that each pair of tuples $(f_1^{(k)}(\bar{a}), f_2^{(l)}(\bar{b}))$ with $1 \leqslant k, l \leqslant n-1$ satisfies $\theta$. We first show that $(f_1^{(k)}(\bar{a}), f_2^{(l)}(\bar{b}))$ satisfies $\theta^{=}$. Let $(i,j) \in \theta^{=}$. Then, $i$ is in $\mathsf{constrained}_1(E)$, and therefore the $i$-th component of $f_1^{(k)}(\bar{a})$ is $a_i$. Analogously, the $j$-th component of $f_2^{(l)}(\bar{b})$ is $b_j$. Because $(\bar{a}, \bar{b})$ satisfies $\theta$, it satisfies $\theta^{=}$, and therefore $a_i = b_j$.

The pair of tuples $(f_1^{(k)}(\bar{a}), f_2^{(l)}(\bar{b}))$ also satisfies $\theta^{<}$. Let $(i,j) \in \theta^{<}$. By construction, the $i$-th component of $f_1^{(k)}(\bar{a})$ equals either $a_i$ or $\mathsf{new}^{(k)}(a_i)$, and, analogously, the $j$-th component of $f_2^{(l)}(\bar{b})$ equals either $b_j$ or $\mathsf{new}^{(l)}(b_j)$. Because $(\bar{a}, \bar{b})$ satisfies $\theta$, we have $a_i < b_j$. By choosing $\mathsf{new}^{(k)}(a_i)$ and $\mathsf{new}^{(l)}(b_j)$ with the same relative order in the domain as $a_i$ and $b_j$, respectively, we also have $\mathsf{new}^{(k)}(a_i) < b_j$, $a_i < \mathsf{new}^{(l)}(b_j)$, and $\mathsf{new}^{(k)}(a_i) < \mathsf{new}^{(l)}(b_j)$.

The arguments that $(f_1^{(k)}(\bar{a}), f_2^{(l)}(\bar{b}))$ satisfies $\theta^{\neq}$ and $\theta^{>}$ are similar. So, each pair of tuples $(f_1^{(k)}(\bar{a}), f_2^{(l)}(\bar{b}))$ with $1 \leqslant k, l \leqslant n-1$ satisfies $\theta$, and we thus obtain at least $n^2$ tuples in $E_1 \bowtie_\theta E_2(D_n)$, which completes the proof.

Using Lemma 24, we can now prove Theorems 17 and 18. By structural induction, we will prove that any RA expression that is not quadratic, is linear and equivalently expressible in $\mathrm{SA}^=$.

The base case is clear: $R$ is not quadratic, is linear, and is in $\mathrm{SA}^=$. For the case of selection, consider an expression of the form $\sigma E$ that is not quadratic (the actual selection condition does not matter here). Then $E$ is not quadratic either, and by induction, $E$ is linear and equivalently expressible in $\mathrm{SA}^=$ as $E'$. We conclude that $\sigma E$ is linear and equivalently expressible in $\mathrm{SA}^=$ as $\sigma E'$. The cases of projection, union, difference, and constant-tagging are handled similarly.

The only nonstraightforward case is $E = E_1 \bowtie_\theta E_2$. Suppose $E$ uses constants in $C = \{c_1, \ldots, c_k\}$ with $c_1 < \cdots < c_k$. Assume $E$ is not quadratic. Then the conditions of Lemma 24 cannot be satisfied, because otherwise $E$ would be quadratic. Hence, we know that for each database $D$ and each joining pair of tuples $(\bar{a}, \bar{b})$ in $E_1(D) \bowtie_\theta E_2(D)$, either $F_1^E(\bar{a})$ or $F_2^E(\bar{b})$ is empty (or both). If $F_1^E(\bar{a})$ is empty, $\bar{a}$ can be completely retrieved from $E_2(D)$, from the constants in $C$, and from the intervals $[c_i, c_{i+1}]$ with $i \in \{1, \ldots, k-1\}$ that are finite; if $F_2^E(\bar{b})$ is empty, $\bar{b}$ can be completely retrieved from $E_1(D)$, from the constants in $C$, and from the intervals $[c_i, c_{i+1}]$ with $i \in \{1, \ldots, k-1\}$ that are finite. The expression $E$ can thus be written as $Z_1 \cup Z_2$, where

$$Z_1 = \{(\bar{a}, \bar{b}) \in E_1 \bowtie_\theta E_2 \mid F_1^E(\bar{a}) = \varnothing\}$$
$$Z_2 = \{(\bar{a}, \bar{b}) \in E_1 \bowtie_\theta E_2 \mid F_2^E(\bar{b}) = \varnothing\}$$

We can now express $Z_1$ and $Z_2$ in $\mathrm{SA}^=$. First let

$$C \cup \bigcup_{\substack{i \in \{1,\ldots,k-1\} \\ [c_i, c_{i+1}] \text{ finite}}} [c_i, c_{i+1}] = \{v_1, \ldots, v_m\}$$

and let us write $\tau_{v_1 \cdots v_m}$ as a shorthand for $\tau_{v_m} \cdots \tau_{v_1}$. Now we can write $Z_2$ as

$$\bigcup_{\substack{f \colon \mathsf{unc}_2(E) \to \mathsf{constrained}_2(E) \\ \cup \{\mathrm{arity}(E_2)+1, \ldots, \mathrm{arity}(E_2)+m\}}} \pi_{\bar{p}} \Big( \sigma_\psi \tau_{v_1 \cdots v_m} (E_1 \ltimes_{\theta^=} \sigma_\varphi \tau_{v_1 \cdots v_m} E_2) \Big),$$

where $f$ ranges over all possible mappings from $\mathsf{unc}_2(E)$ to $\mathsf{constrained}_2(E) \cup \{\mathrm{arity}(E_2)+1, \ldots, \mathrm{arity}(E_2)+m\}$, and where

$$\varphi \equiv \bigwedge_{j \in \mathsf{unc}_2(E)} j = f(j),$$

$$\psi \equiv \bigwedge_{\alpha \in \{\neq, <, >\}} \bigwedge_{(i,j) \in \theta^\alpha} i \; \alpha \; g(j),$$

and $\bar{p} = 1, \ldots, \text{arity}(E_1), g(1), \ldots, g(\text{arity}(E_2))$ where

$$g(j) = \begin{cases} \min\{i \mid (i,j) \in \theta^{=}\} & \text{if } j \in \mathsf{constrained}_2(E) \\ \min\{i \mid (i,f(j)) \in \theta^{=}\} & \text{if } j \in \mathsf{unc}_2(E) \text{ and } f(j) \in \mathsf{constrained}_2(E) \\ \text{arity}(E_1) + \ell & \text{if } j \in \mathsf{unc}_2(E) \text{ and } f(j) = \text{arity}(E_2) + \ell \end{cases}$$

The use of the minimum function is arbitrary here; any function that chooses an element out of a set will do.

The SA$^=$ expression for $Z_1$ is entirely analogous. Since SA$^=$ expressions are always linear, it also follows that $E$ is linear, as desired. This concludes the proof of Theorems 17 and 18.

## 4 Division, set join, and friends

By Corollary 19, to prove that a query can only be expressed in the relational algebra by quadratic expressions, it suffices to show that it is not expressible in SA$^=$. And to show nonexpressibility in SA$^=$, we have Corollary 14 as a tool.

We are thus fully armed now to return to the division operator and set joins from the beginning of this article, and show:

**Proposition 26** *Division is expressible in RA only by quadratic expressions. Furthermore, every RA expression that is empty if and only if the set join is empty, must be quadratic.*

Note that it would not be very interesting to claim that the set join itself can only be expressed by quadratic expressions, because the output size of the set join is already quadratic.

To prove Proposition 26, we need to show that $R \div S$ is not expressible in SA$^=$ using constants in a fixed finite set $C$. Thereto, consider the databases $A$ and $B$ shown in Figure 5. (Here, we take the natural numbers as our universe $\mathbb{U}$.) We assume that the values in $A$ and $B$ are not in the set $C$. Then $R \div S$ equals $\{1, 2\}$ in $A$, but is empty in $B$ (regardless of whether we use the set containment, or the set equality variant of division). Nevertheless, $A, 1 \sim_g^C B, 1$, so any SA$^=$ expression that returns 1 on $A$ will also return 1 on $B$ and therefore cannot express $R \div S$. To see that $A, 1 \sim_g^C B, 1$, we invite the reader to verify that the following set $\mathcal{I}$ is a $C$-guarded bisimulation:

$$\mathcal{I} = \{1 \mapsto 1\} \cup \{\bar{a} \mapsto \bar{b} \mid \bar{a} \in A(R) \text{ and } \bar{b} \in B(R), \text{ or } \bar{a} \in A(S) \text{ and } \bar{b} \in B(S)\}$$

To handle the set join version of Proposition 26, just insert a column into

$A$

$R$

| | |
|---|---|
| 1 | 7 |
| 1 | 8 |
| 2 | 7 |
| 2 | 8 |

$S$

| |
|---|
| 7 |
| 8 |

$B$

$R$

| | |
|---|---|
| 1 | 7 |
| 1 | 8 |
| 2 | 8 |
| 2 | 9 |
| 3 | 7 |
| 3 | 9 |

$S$

| |
|---|
| 7 |
| 8 |
| 9 |

Fig. 5. Two databases $A$ and $B$ showing that division is inexpressible in $\mathrm{SA}^=$.

relation $S$ (this will be the first column of the new relation), with always the same value 4, which we assume is also not in $C$. Then the above $\mathcal{I}$ is still a $C$-guarded bisimulation.

**Other queries** Clearly, the applicability of the techniques we have developed in this paper is not restricted to division and set joins! For example, over the beer-drinkers database schema from Example 3, consider the following query $Q$:

List all drinkers that visit a bar that serves a beer they like.

Any RA expression of this query must be quadratic.

To see this, we show again that $Q$ is not expressible in $\mathrm{SA}^=$ using constants in a fixed finite set $C$. Thereto, consider the databases $A$ and $B$ shown in Figure 6. (Here, we take the lexicographically ordered strings as our universe $\mathbb{U}$.) We assume that the values in $A$ and $B$ are not in the set $C$. In $A$, Alex visits the Pareto bar, which serves Westmalle, which he likes. But in $B$ no drinker visits a bar that serves a beer he likes. Nevertheless, $(A, \text{alex}) \sim_g^C (B, \text{alex})$, so any $\mathrm{SA}^=$ expression that returns alex on $A$ will also return alex on $B$ and therefore cannot express $Q$. To see that $(A, \text{alex}) \sim_g^C (B, \text{alex})$, we invite the reader to verify that the following set $\mathcal{I}$ is a $C$-guarded bisimulation:

$$\begin{aligned}\mathcal{I} = {} & \{\text{alex} \mapsto \text{alex}\} \\ & \cup \bigcup \Big\{ \{\bar{a} \mapsto \bar{b} \mid \bar{a} \in A(R) \text{ and } \bar{b} \in B(R)\} \mid R = \text{Visits}, \text{Serves}, \text{Likes} \Big\}\end{aligned}$$

| $A$ | $B$ |
| --- | --- |
| Visits(alex, pareto bar) | Visits(alex, pareto bar) |
| Serves(pareto bar, westmalle) | Visits(bart, qwerty bar) |
| Likes(alex, westmalle) | Serves(pareto bar, westmalle) |
| | Serves(qwerty bar, westvleteren) |
| | Likes(alex, westvleteren) |
| | Likes(bart, westmalle) |

Fig. 6. Two databases $A$ and $B$ showing that the query "give all drinkers that visit a bar that serves a beer they like" is not expressible in $\mathrm{SA}^{=}$.

## 5 Concluding remarks

The attentive reader will note that the beer-drinkers query $Q$ from the previous section is a typical example of a "cyclic" join query, and such joins are already long known not to be computable by semijoins only [5,6,4]. But note that the semijoin programs that were considered in the theory of join dependencies can use *only* semijoins, while SA expressions can also use $\sigma$, $\pi$, $\cup$ and $-$.

On the technical side, our work leaves open the generalisation where the universe of data elements is not merely equipped with a total order, but where arbitrary predicates are present which can be used in join conditions. One cannot expect our Theorem 18 to hold in all such cases, as this will depend on the predicates at hand. A related issue is to investigate the impact of integrity constraints on our results.

Practical query processing uses a more powerful relational algebra including grouping, sorting, and aggregation operators. Proving complexity lower bounds in such a rich setting seems very challenging to us. However, containment-division can be expressed by the linear expression

$$\pi_A\Big(\gamma_{A,\mathrm{count}(B)}(R \ltimes_{B=C} S) \underset{\mathrm{count}(B)=\mathrm{count}(C)}{\ltimes} \gamma_{\varnothing,\mathrm{count}(C)} S\Big)$$

using grouping ($\gamma$) and aggregation (counting). Equality-division can be expressed by an analogous linear RA expression with grouping and counting [11,12].

## Acknowledgment

We thank Jerzy Tyszkiewicz for helpful discussions on the relationship between the semijoin algebra and the guarded fragment. The second author would also like to thank Bart Goethals for inspiring discussions on the semijoin algebra, and Dirk Van Gucht for inspiring discussions on the complexity of set joins.

## References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] A. Aho, J.E. Hopcroft, and J.D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, 1983.

[3] H. Andréka, I. Németi, and J. van Benthem. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998.

[4] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database schemes. *Journal of the ACM*, 30(3):479–513, 1983.

[5] P.A. Bernstein and D.W. Chiu. Using semi-joins to solve relational queries. *Journal of the ACM*, 28(1):25–40, 1981.

[6] P.A. Bernstein and N. Goodman. Power of natural semijoins. *SIAM Journal on Computing*, 10(4):751–771, 1981.

[7] E.F. Codd. Relational completeness of data base sublanguages. In R. Rustin, editor, *Data Base Systems*, pages 65–98. Prentice-Hall, 1972.

[8] H. de Nivelle and M. de Rijke. Deciding the guarded fragments by resolution. *Journal of Symbolic Computation*, 35(1):21–58, 2003.

[9] E. Grädel. On the restraining power of guards. *Journal of Symbolic Logic*, 64(4):1719–1742, 1999.

[10] E. Grädel, C. Hirsch, and M. Otto. Back and forth between guarded and modal logics. *ACM Transactions on Computational Logic*, 3(3):418–463, 2002.

[11] G. Graefe. Relational division: four algorithms and their performance. In *Proceedings of the 5th International Conference on Data Engineering*, pages 94–101. IEEE Computer Society, 1989.

[12] G. Graefe and R.L. Cole. Fast algorithms for universal quantification in large databases. *ACM Transactions on Database Systems*, 20(2):187–236, 1995.

[13] S. Helmer and G. Moerkotte. Evaluation of main memory join algorithms for joins with set comparison join predicates. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, pages 386–395. Morgan Kaufmann Publishers Inc., 1997.

[14] D. Leinders, M. Marx, J. Tyszkiewicz, and J. Van den Bussche. The semijoin algebra and the guarded fragment. *Journal of Logic, Language and Information*, 14(3):331–343, June 2005.

[15] D. Leinders, J. Tyszkiewicz, and J. Van den Bussche. On the expressive power of semijoin queries. *Information Processing Letters*, 91(2):93–98, 2004.

[16] N. Mamoulis. Efficient processing of joins on set-valued attributes. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 157–168. ACM Press, 2003.

[17] K. Ramasamy, J.M. Patel, J.F. Naughton, and R. Kaushik. Set containment joins: the good, the bad and the ugly. In *Proceedings of the 26th International Conference on Very Large Data Bases*, pages 351–362. Morgan Kaufmann Publishers Inc., 2000.

[18] S.G. Rao, A. Badia, and D. Van Gucht. Providing better support for a class of decision support queries. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 217–227. ACM Press, 1996.

[19] S. Sarawagi and A. Kirpal. Efficient set joins on similarity predicates. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 743–754. ACM Press, 2004.

[20] J.D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume 1. Computer Science Press, 1988.