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

Dirk Leinders Jan Van den Bussche
Limburgs Universitair Centrum

ABSTRACT

We show that any expression of the relational division operator in the relational algebra with union, difference, projection, selection, and equijoins, 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); we link linear relational algebra expressions to expressions using only semijoins instead of joins; and we link these semijoin algebra expressions to the guarded fragment of first-order logic.

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 $R(A, B) \div S(C)$ 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* of $R(A, B)$ and $S(C, D)$, which returns

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

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 $=$ [19, 20]. Note that a set join with predicate "intersection nonempty" boils down to an ordinary equijoin!

It has long been observed that division is not well handled by classical query processing [13, 14]. 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, and equijoins (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 relation algebra that includes grouping, sorting, and aggregation operators [15, 17, 18]. 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]. Consequently, if a query is not expressible in the semijoin algebra, then its complexity in the relational algebra is at least quadratic. Furthermore, we characterize the class of semijoin algebra queries in turn as the queries expressible in the "guarded" fragment of first-order logic [3, 11, 12, 8]. An equivalence relation on structures, called guarded bisimilarity, is known to guarantee indistinguishability in the guarded fragment, and it is this tool that we will use to prove our complexity result.

One obvious problem that we leave open is to allow not just equijoins but theta-joins. Indeed, while we do allow selection comparisons involving not just equalities, but also order, we do not know yet how to cover join conditions involving order. As a matter of fact, our present characterization of linear relational algebra expressions by semijoin expression fails in the presence of order (or even nonequalities) in join conditions. Moreover, practical query processing uses a more powerful relational algebra including grouping, sorting, and aggregation operators. Proving complexity lower bounds in

[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.

such a rich setting seems very challenging to us.

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. Each relation name $R \in \mathbf{S}$ is a relational algebra expression. Its arity comes from $\mathbf{S}$.
2. If $E_1, E_2 \in \text{RA}$ 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 $E \in \text{RA}$ 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 $E \in \text{RA}$ 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 $E_1, E_2 \in \text{RA}$ have arities n and m, respectively, and θ is a conjunction of equalities of the form $\bigwedge_{s=1}^{k} x_{i_s} = y_{j_s}$, 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 and the join operator are as follows: (for relations r, r_1 and r_2)

$$\pi_{i_1,\ldots,i_k}(r) := \{(a_{i_1}, \ldots, a_{i_k}) \mid \bar{a} \in r\}$$
$$\sigma_{i=j}(r) := \{\bar{a} \in r \mid a_i = a_j\}$$
$$\sigma_{i<j}(r) := \{\bar{a} \in r \mid a_i < a_j\}$$
$$r_1 \bowtie_\theta r_2 := \{(\bar{a}, \bar{b}) \mid \bar{a} \in r_1,\ \bar{b} \in r_2,\ a_{i_s} = b_{j_s} \text{ for } s = 1, \ldots, k\}$$

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)

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

Example 3. Suppose $\mathbf{S}$ is Ullman's well-known example schema

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

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:

$$\pi_1\big(\text{Visits} \underset{x_2=y_1}{\ltimes} (\pi_1(\text{Serves}) - \pi_1(\text{Serves} \underset{x_2=y_2}{\ltimes} \text{Likes}))\big).$$

□

Note that SA expressions can only output "stored" tuples, defined as follows:

Definition 4 (stored tuple). A tuple is *stored* in database D if it belongs to some projection $\pi_{i_1,\ldots,i_p}(D(R))$ of one of the relations of D.

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

Definition 5 (guarded fragment, GF).

1. Atomic formulas of the form $x = y$ and $x < y$ 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 φ and ψ 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 φ do actually occur in α, 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 6. The query from Example 3 can be expressed by the following GF formula $\varphi(x)$:

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

□

The following strong correspondence between SA and GF is proved in the Appendix:

Theorem 7. *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 A,*

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

Conversely, for every GF formula $\varphi(x_1, \ldots, x_k)$, there exists an SA expression E_φ such that for every database A,

$$E_\varphi(A) = \{\bar{a} \text{ stored tuple in } A \mid A \models \varphi(\bar{a})\}$$

This 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 8 (guarded set). A set is *guarded* in database A if it is of the form $\{a_1,\dots,a_n\}$, where $(a_1,\dots,a_n) \in A(R)$ for some $R \in \mathbf{S}$.

Definition 9 (guarded bisimulation, guarded bisimilarity). A *guarded bisimulation* between two databases A and B is a non-empty set $\mathcal{I}$ of finite partial isomorphisms from A to B, such that the following back and forth conditions are satisfied:[2]

Forth. For every $f : X \to Y$ in $\mathcal{I}$ and for every guarded set X', there exists a partial isomorphism $g : X' \to Y'$ in $\mathcal{I}$ such that f and g agree on $X \cap X'$.

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

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

The following is a basic fact about GF [3]:

Proposition 10. *The guarded fragment is invariant under guarded bisimulation. Formally, if $A, \bar{a} \sim_g B, \bar{b}$, then for any GF formula $\varphi(\bar{x})$ we have:*

$$A \models \varphi(\bar{a}) \quad \Leftrightarrow \quad B \models \varphi(\bar{b}).$$

By Theorem 7 we obtain:

Corollary 11. *If $A, \bar{a} \sim_g B, \bar{b}$, then for any SA expression E 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 12. 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 Ω notation, we now define:[3]

Definition 13. 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 14. *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 15. *Every RA expression that is not quadratic, is equivalently expressible in SA.*

Note that the semijoin operator can be expressed in RA in a linear way; for example,

$$R(A,B) \underset{B=C}{\ltimes} S(C,D) = \pi_{A,B}(R \underset{B=C}{\bowtie} \pi_C(S)).$$

From the above theorems we therefore obtain:

Corollary 16. *A query is expressible by a linear RA expression if and only if it is expressible by an SA expression.*

We will prove Theorem 14 and 15 simultaneously. Our crucial lemma is Lemma 21. In order to state it, we need two definitions.

Definition 17. Let E be an RA expression of the form $E_1 \bowtie_\theta E_2$. We view $\theta \equiv \bigwedge_{s=1}^{k} x_{i_s} = y_{j_s}$ as the set of pairs $\{(i_s, j_s) \mid s = 1,\dots,k\}$. For $\ell = 1, 2$, the sets $\mathsf{constrained}_\ell(E)$ and their complements $\mathsf{unc}_\ell(E)$ are now defined as follows:

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

Example 18. For the expression $E = R \bowtie_{x_3 = y_1} S$, where R and S are ternary, we get:

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

□

Definition 19. Let D be a database and let E be an RA expression of the form $E_1 \bowtie_\theta E_2$. For any $\bar{a} \in E_1(D)$, we denote the set of elements occurring in $\bar{a}$ by $\mathsf{set}(\bar{a})$. We now define the set of *free values of* $\bar{a}$ as follows:

$$F_1^E(\bar{a}) := \mathsf{set}(\bar{a}) - \{a_i \mid i \in \mathsf{constrained}_1(E)\}$$

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

Example 20. Take again expression E from Example 18. Suppose that relation R contains the tuples $r_1 = (1,2,3)$

[2]For $X, Y \subseteq \mathbb{U}$, a mapping $f : X \to Y$ is a *partial isomorphism* from A to B if it is bijective, and for each $R \in \mathbf{S}$, of arity n, and all $x_1,\dots,x_n \in X$, we have $(x_1,\dots,x_n) \in A(R) \Leftrightarrow (f(x_1),\dots,f(x_n)) \in B(R)$, and moreover, for all $x, y \in X$, we have $x < y \Leftrightarrow f(x) < f(y)$.

[3]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].

and $r_2 = (4,5,4)$, and that relation S contains the tuples $s_1 = (3,4,5)$ and $s_2 = (3,3,3)$. Then:

$$\begin{aligned} F_1^E(r_1) &= \{1,2\} \qquad & F_2^E(s_1) &= \{4,5\} \\ F_1^E(r_2) &= \{5\} \qquad & F_2^E(s_2) &= \varnothing \end{aligned}$$

$\square$

We can now state the following crucial lemma:

Lemma 21. *Let $E = E_1 \bowtie_\theta E_2$, 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 $F_1^E(\bar{a}) \neq \varnothing \neq F_2^E(\bar{b})$. 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. *$|E_1 \bowtie_\theta E_2(D_n)| \geqslant n^2$.*

The proof uses the invariance of SA under guarded bisimilarity (Corollary 11), and is given in the Appendix.

Using Lemma 21, we can now prove Theorems 14 and 15. By structural induction, we will prove that any RA expression that is not quadratic, is linear and equivalently expressible in SA.

The base case is clear: R is not quadratic, is linear, and is in SA. For the case of selection, consider an expression of the form σE 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 SA as E'. We conclude that σE is linear and equivalently expressible in SA as $\sigma E'$. The cases of projection, union and difference are handled similarly.

The only nonstraightforward case is $E = E_1 \bowtie_\theta E_2$. Assume E is not quadratic. Then the conditions of Lemma 21 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 $E_2(D)$; if $F_2^E(\bar{b})$ is empty, $\bar{b}$ can be completely retrieved from $E_1(D)$. E can thus be written as $Z_1 \cup Z_2$, where

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

We can now express Z_1 and Z_2 in SA, as follows:

$$Z_2 = \bigcup_{f\,:\ \mathsf{unc}_2(E) \to \mathsf{constrained}_2(E)} \pi_{\bar{p}}(E_1 \ltimes_\theta \sigma_\varphi E_2)$$

Here,

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

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

$$g(j) = \begin{cases} \min\{i \mid (i,j) \in \theta\} & \text{if } j \in \mathsf{constrained}_2(E) \\ \min\{i \mid (i, f(j)) \in \theta\} & \text{if } j \in \mathsf{unc}_2(E) \end{cases}$$

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

Figure 1: Two databases A and B showing that division is inexpressible in SA.

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 our proof.

4. DIVISION, SET JOIN, AND FRIENDS

By Corollary 16, 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 11 as a tool.

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

Proposition 22. *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 22, we need to show that $R \div S$ is not expressible in SA. Thereto, consider the databases A and B shown in Figure 1. (Here, we take the natural numbers as our universe $\mathbb{U}$.) 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, $A, 1 \sim_g B, 1$, 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 $A, 1 \sim_g B, 1$, we invite the reader to verify that the following set $\mathcal{I}$ is a guarded bisimulation:

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

To handle the set-join version of Proposition 22, just insert a column into relation S (this will be the first column of the new relation), with always the same value 4. Then the above $\mathcal{I}$ is still a guarded bisimulation.

Other queries

Clearly, the applicability of the techniques we have developed in this paper is not restricted to division and set joins!

A

Visits(alex, pareto bar)
Serves(pareto bar, westmalle)
Likes(alex, westmalle)

B

Visits(alex, pareto bar)
Visits(bart, qwerty bar)
Serves(pareto bar, westmalle)
Serves(qwerty bar, westvleteren)
Likes(alex, westvleteren)
Likes(bart, westmalle)

Figure 2: 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 SA.

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 SA. Thereto, consider the databases A and B shown in Figure 2. (Here, we take the lexicographically ordered strings as our universe $\mathbb{U}$.) 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, $(A, \text{alex}) \sim_g (B, \text{alex})$, so any SA expression that returns alex on A will also return alex on B and therefore cannot express Q. To see that $(A, \text{alex}) \sim_g (B, \text{alex})$, we invite the reader to verify that the following set $\mathcal{I}$ is a guarded bisimulation:

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

5. DISCUSSION

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 σ, π, $\cup$ and $-$. In this connection, it has already been observed that non-recursive stratified datalog (NRSD) programs, in which every rule must be an acyclic join query, correspond to GF [9, 10]. By the well-known correspondence between acyclic join queries and semijoin programs, these acyclic NRSD programs also correspond to SA. Hence, the correspondence we have shown in Section 2 between SA and GF could also have been derived by combining these previous results. Nevertheless, the equivalence proof we give is direct and elementary.

As already mentioned in the Introduction, an obvious problem left open by us is to allow theta-joins with join conditions involving order or even other predicates. Of course, the conjecture is that division still requires quadratic expressions, even when theta-joins are allowed. Note, however, that Theorem 15 and Corollary 16 no longer literally hold in this context. For example, the RA expression over unary relations R and S:

$$(R - (R \ltimes_{x_1 \neq y_1} R)) \times (S - (S \ltimes_{x_1 \neq y_1} S))$$

is linear but not equivalently expressible in SA. With order conditions in semijoins, SA also becomes strictly more powerful than GF.

We point out that in previous work, we have already generalized the notion of guarded bisimilarity to SA with order-semijoins [16], but how to apply this to generalize the results of the present paper remains open.

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.

6. 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] J. Flum, M. Frick, and M. Grohe. Query evaluation via tree-decompositions. *Journal of the ACM*, 49(6):716–752, 2002.

[10] G. Gottlob, E. Grädel, and H. Veith. Datalog lite: a deductive query language with linear time model checking. *ACM Transactions on Computational Logic*, 3(1):42–79, 2002.

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

[12] 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.

[13] 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.

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

[15] 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.

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

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

[18] 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.

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

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

APPENDIX

A. SEMIJOIN ALGEBRA AND GUARDED FRAGMENT

From SA to GF

Lemma 23. *For every SA expression E of arity k, for every database A and for every tuple $\bar{a} = (a_1, \dots, a_k)$ in $E(A)$, there exists R in $\mathbf{S}$, a tuple $\bar{t}$ in $A(R)$, and a function $f : \{1, \dots, k\} \to \{1, \dots, arity(R)\}$ such that $a_i = t_{f(i)}$ for $i = 1, \dots, k$.*

Proof. By structural induction on expression E. ☐

We now prove that for every SA expression E of arity k, there exists a GF formula $\varphi_E(x_1, \dots, x_k)$ such that for every database A,

$$E(A) = \{\bar{a} \in \mathbb{U}^k \mid A \models \varphi_E(\bar{a})\}.$$

The proof is by structural induction on E.

- if E is R, then $\varphi_E(x_1, \dots, x_k) := R(x_1, \dots, x_k)$.
- if E is $E_1 \cup E_2$, then

  $$\varphi_E(x_1, \dots, x_k) := \varphi_{E_1}(x_1, \dots, x_k) \vee \varphi_{E_2}(x_1, \dots, x_k).$$
- if E is $E_1 - E_2$, then

  $$\varphi_E(x_1, \dots, x_k) := \varphi_{E_1}(x_1, \dots, x_k) \wedge \neg\varphi_{E_2}(x_1, \dots, x_k).$$
- if E is $\sigma_{i=j}(E_1)$, then $\varphi_E(x_1, \dots, x_k) := \varphi_{E_1}(x_1, \dots, x_k) \wedge x_i = x_j$.
- if E is $\sigma_{i<j}(E_1)$, then $\varphi_E(x_1, \dots, x_k) := \varphi_{E_1}(x_1, \dots, x_k) \wedge x_i < x_j$.
- if E is $\pi_{i_1,\dots,i_k}(E_1)$ with E_1 of arity n, then, by induction, we have a formula $\varphi_{E_1}(z_1, \dots, z_n)$. By Lemma 23, $\varphi_{E_1}(\bar{z})$ is equivalent to the formula obtained by replacing in $\psi :=$

  $$\bigvee_{R \in \mathbf{S}} \bigvee_{f:\{1,\dots,n\}\to\{1,\dots,\mathrm{arity}(R)\}} \exists(t_j)_{j\in Q}\big(R(\bar{t}) \wedge \varphi_{E_1}(t_{f(1)}, \dots, t_{f(n)})\big)$$

  each $t_{f(i)}$ by z_i, $i = 1, \dots, n$. In this formula, Q is $\{1, \dots, \mathrm{arity}(R)\} - f(\{1, \dots, n\})$. Formula φ_E should now only select components $i_1, \dots, i_k$ out of this formula. To this end, we modify ψ by using $Q' = \{1, \dots, \mathrm{arity}(R)\} - f(\{i_1, \dots, i_k\})$ instead of Q, and replacing $t_{f(i_l)}$ by x_l, for $l = 1, \dots, k$. Thus $\varphi_E(x_1, \dots, x_k)$ is obtained.
- if E is $E_1 \ltimes_\theta E_2$ with $\theta = \bigwedge_{l=1}^{s} x_{i_l} = y_{j_l}$ and E_2 of arity n, then, by induction, we have formulas $\varphi_{E_1}(x_1, \dots, x_k)$ and $\varphi_{E_2}(z_1, \dots, z_n)$. Using Lemma 23, $\varphi_E(x_1, \dots, x_k)$ is obtained by replacing in formula $\chi :=$

  $$\varphi_{E_1}(x_1, \dots, x_k) \wedge \bigvee_{R \in \mathbf{S}} \bigvee_{f:\{1,\dots,n\}\to\{1,\dots,\mathrm{arity}(R)\}} \exists(t_j)_{j\in Q}\big(R(\bar{t}) \wedge \varphi_{E_2}(t_{f(1)}, \dots, t_{f(n)})\big)$$

  each $t_{f(j_l)}$ by x_{i_l}, $l = 1, \dots, s$. In this formula, Q is $\{1, \dots, \mathrm{arity}(R)\} - f(\{j_1, \dots, j_s\})$. Note that condition θ is enforced by repetition of variables x_{i_l}. ☐

From GF to SA

Definition 24. Let A be a database over database schema $\mathbf{S}$. The set of stored tuples in A is defined by the SA expression

$$G_k = \bigcup_{R \in \mathbf{S}} \{\pi_{i_1,\dots,i_k} R \mid 1 \leqslant i_1, \dots, i_k \leqslant \mathrm{arity}(R)\}.$$

We now prove that for every GF formula $\varphi(x_1, \dots, x_k)$, there exists an SA expression E such that for every database A,

$$\{\bar{a} \text{ stored tuple in } A : A \models \varphi(\bar{a})\} = E(A).$$

By structural induction on φ, we construct the desired semijoin expression E_φ.

- if $\varphi(x_1,\ldots,x_k)$ is $R(x_{i_1},\ldots,x_{i_l})$ then $E_\varphi := G_k \ltimes_\theta R$, where θ is $(x_{i_1} = y_1) \wedge (x_{i_2} = y_2) \wedge \ldots \wedge (x_{i_l} = y_l)$;
- if $\varphi(x_1,\ldots,x_k)$ is $(x_i = x_j)$ then $E_\varphi := \sigma_{i=j}(G_k)$;
- if $\varphi(x_1,\ldots,x_k)$ is $(x_i < x_j)$ then $E_\varphi := \sigma_{i<j}(G_k)$;
- if $\varphi(x_1,\ldots,x_k)$ is $\psi(x_1,\ldots,x_k) \vee \xi(x_1,\ldots,x_k)$ then $E_\varphi := E_\psi \cup E_\xi$;
- if $\varphi(x_1,\ldots,x_k)$ is $\neg\psi(x_1,\ldots,x_k)$ then $E_\varphi := G_k - E_\psi$;
- suppose $\varphi(x_1,\ldots,x_k)$ is $\exists\bar{z}(\alpha(\bar{x},\bar{z}) \wedge \psi(\bar{x},\bar{z}))$. Let x_{i_1}, $\ldots, x_{i_r}$ be the different occurrences of variables among $x_1,\ldots,x_k$ in α. Now, $E_\varphi := G_k \ltimes_{\theta_1} (E_\alpha \ltimes_{\theta_2} E_\psi)$ where condition θ_1 is $(x_{i_1} = y_{f(1)}) \wedge (x_{i_2} = y_{f(2)}) \wedge \ldots \wedge (x_{i_r} = y_{f(r)})$ with $f(j)$ the position of x_{i_j} in α. Condition θ_2 equates the $\bar{x}$- and $\bar{z}$-variables in ψ to their occurrences in α. $\square$

B. PROOF OF LEMMA 21

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

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

Proof of Lemma 21. 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 $x \in F_1(\bar{a})$ and for each $x \in F_2(\bar{b})$, 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 $\forall r,s \in \mathrm{dom}(D'_k)$ $\exists t \in \mathbb{U}$: $r < t < s$. So, we assume w.l.o.g. that we can always create these new domain elements satisfying the specified condition;
2. we set $\mathrm{dom}(D_{k+1})$ to $\mathrm{dom}(D_k)$ extended with the newly created domain elements;
3. for each tuple $\bar{t} = (t_1,\ldots,t_n) \in T_D$ satisfying $\mathsf{set}(\bar{t}) \cap F_1(\bar{a}) \neq \varnothing$, we construct a tuple $f_1^{(k)}(\bar{t}) = (r_1,\ldots,r_n)$ with
$$r_i = \begin{cases} \mathsf{new}^{(k)}(t_i) & \text{if } t_i \in F_1(\bar{a}) \\ t_i & \text{else} \end{cases}$$
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 partial isomorphism.
4. for each tuple $\bar{t} = (t_1,\ldots,t_n) \in T_D$ satisfying $\mathsf{set}(\bar{t}) \cap F_2(\bar{b}) \neq \varnothing$, we construct a tuple $f_2^{(k)}(\bar{t}) = (r_1,\ldots,r_n)$ with
$$r_i = \begin{cases} \mathsf{new}^{(k)}(t_i) & \text{if } t_i \in F_2(\bar{b}) \\ t_i & \text{else} \end{cases}$$
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 partial isomorphism.

$D(R)$		
1	2	3
8	9	10

$D(S)$		
3	4	5

$D(T)$	
6	1
4	7

$D_2(R)$		
1	2	3
8	9	10
$1'$	$2'$	3

$D_2(S)$		
3	4	5
3	$4'$	$5'$

$D_2(T)$	
6	1
4	7
6	$1'$
$4'$	7

$D_3(R)$		
1	2	3
8	9	10
$1'$	$2'$	3
$1''$	$2''$	3

$D_3(S)$		
3	4	5
3	$4'$	$5'$
3	$4''$	$5''$

$D_3(T)$	
6	1
4	7
6	$1'$
$4'$	7
6	$1''$
$4''$	7

Figure 3: Databases $D = D_1$, D_2 and D_3 in the construction for $E = (R \ltimes_{x_1=y_2} T) \bowtie_{x_3=y_1} (S \ltimes_{x_2=y_1} T)$.

5. we set the tuple space $T_{D_{k+1}}$ to T_{D_k} extended with the newly constructed tuples.

To illustrate this construction, let database D be the one shown in the upper part of Figure 3 and let expression E be $(R \ltimes_{x_1=y_2} T) \bowtie_{x_3=y_1} (S \ltimes_{x_2=y_1} T)$. Let $\bar{a}$ be $(1,2,3)$ and let $\bar{b}$ be $(3,4,5)$. Then, $F_1(\bar{a}) = \{1,2\}$ and $F_2(\bar{b}) = \{4,5\}$. For each $i \in F_1(\bar{a}) \cup F_2(\bar{b})$, 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 3.

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 $1 \leqslant k \leqslant n-1$
$$D, \bar{a} \ \sim_g \ D_n, f_1^{(k)}(\bar{a})$$
Take an arbitrary n and consider the set $\mathcal{I} = \{g_{\bar{t}}^{(k)} \mid \bar{t} \in T_D$ with $\mathsf{set}(\bar{t}) \cap F_1(\bar{a}) \neq \varnothing$, $1 \leqslant k \leqslant n-1\} \cup \{h_{\bar{t}} \mid \bar{t} \in T_D\}$, where

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

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 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)* $X' \cap F_1(\bar{a}) \neq \varnothing$. Then, f agrees with partial isomorphism $g_{\bar{t}'}^{(k)}$ on $\mathsf{set}(\bar{t}) \cap X'$. Indeed, they both map values $x \in F_1(\bar{a})$ onto $\mathsf{new}^{(k)}(x)$ and they map values $y \notin F_1(\bar{a})$ onto y. *ii)* $X' \cap F_1(\bar{a}) = \varnothing$. 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

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

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

By Corollary 11 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 $R \ltimes_{x_1=y_2} T$ 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 $S \ltimes_{x_2=y_1} T$.

Note that for each $i \in \mathsf{constrained}_1(E)$, the i-th component of $f_1^{(k)}(\bar{a})$ equals a_i and for each $j \in \mathsf{constrained}_2(E)$, the j-th component of $f_2^{(l)}(\bar{b})$ equals b_j. Because $(\bar{a}, \bar{b})$ satisfies condition θ, each pair of tuples $(f_1^{(k)}(\bar{a}), f_2^{(l)}(\bar{b}))$ with $1 \leqslant k, l \leqslant n-1$ also satisfies θ. This gives us at least n^2 tuples in $E_1 \bowtie_\theta E_2(D_n)$, which completes the proof. $\square$