Noname manuscript No.
(will be inserted by the editor)

On matrices and K-relations

Robert Brijder · Marc Gyssens · Jan Van den Bussche

the date of receipt and acceptance should be inserted later

Abstract We show that the matrix query language MATLANG corresponds to a natural fragment of the positive relational algebra on K-relations. The fragment is defined by introducing a composition operator and restricting K-relation arities to two. We then proceed to show that MATLANG can express all matrix queries expressible in the positive relational algebra on K-relations, when intermediate arities are restricted to three. Thus we offer an analogue, in a model with numerical data, to the situation in classical logic, where the algebra of binary relations is equivalent to first-order logic with three variables.

Keywords Expressive power · Provenance semirings · Annotated relations · Data science

1 Introduction

Motivated by large-scale data science, there is recent interest in supporting linear algebra operations, such as matrix multiplication, in database systems. This has prompted investigations comparing the expressive power of common matrix operations with the operations on relations provided by the relational algebra and SQL [8,9,12,4].

For *Boolean* matrices, the connection between matrices and relations is very natural and well known. An $m \times n$ Boolean matrix A can be viewed as a binary relation $R \subseteq \{1, \ldots, m\} \times \{1, \ldots, n\}$, where R consists of those

This is a revised and extended version of the conference paper "On matrices and K-relations" presented at the 11th International Symposium on Foundations of Information and Knowledge Systems (FoIKS 2020), Dortmund, Germany, February 17–21, 2020 [5].

Robert Brijder has done this work as a Postdoctoral Fellow of the Research Foundation – Flanders (FWO).

Hasselt University, Faculty of Sciences, Martelarenlaan 42, 3500 Hasselt, Belgium. Corresponding author: Marc Gyssens. E-mail: marc.gyssens@uhasselt.be

pairs (i, j) for which $A_{i,j} = 1$. Boolean matrix multiplication then amounts to composition of binary relations. Composition is the central operation in the *algebra of binary relations* [13,16,17]. Besides composition, this algebra has operations such as converse, which corresponds to transposition of a Boolean matrix; union and complement, which correspond to disjunction and negation of Boolean matrices; and the empty and identity relations, which correspond to the zero and identity matrices.

A common theme in research in the foundations of databases is the expressive power of query languages [1]. When we decide to use a particular query language, we would like to understand as well as possible what we can do with this query language. Results that characterize expressiveness may be very helpful in this respect. An example of such a result is the classical Codd theorem, stating the equivalence between the standard relational algebra and first-order logic. Likewise, for the algebra of binary relations, a classical result [18] is that it has the same expressive power as the formulas with two free variables in FO(3), the three-variable fragment of first-order logic. In this sense, we understand quite well the expressive power of a natural set of operations on Boolean matrices.

What can now be said in this regard about more general matrices, with entries that are not just Boolean values? An $m \times n$ matrix with entries in some semiring K is essentially a mapping from $\{1, \dots, m\} \times \{1, \dots, n\}$ to K. This perfectly fits the data model of *K-relations* introduced by Green et al. [7]. In general, consider an infinite domain **dom** and a supply of attributes. In a database instance, we assign to each attribute a range of values, in the form of a finite subset of **dom**. Attributes can be declared to be compatible; compatible attributes have the same range. A relation schema S is a finite set of attributes. Tuples over S are mappings that assign to each attribute a value of the appropriate range. Now, a K-relation over S is a mapping that assigns to each tuple over S an element of K.

So, an $m \times n$ matrix X can be seen as a K-relation over two attributes, say, A and B, where the range of A is $\{1, \dots, m\}$ and the range of B is $\{1, \dots, n\}$. We can assume an order on all attributes and choose $A < B$ so that we know which values are row indices and which are column indices. If, furthermore, there is an $n \times k$ matrix Y at play, we can also model the latter as a K relation over two attributes, say, C and D, with $C < D$, but with the additional conditional that C is compatible with B to reflect that the number of columns of matrix X equals the number of rows of matrix Y. We can view vectors as K-relations over a single attribute, and scalars as K-relations over the empty schema. In general, a K-relation of arity r is essentially an r-dimensional tensor (multidimensional array). (Because we need not necessarily assume an order on **dom**, the tensor is unordered.)

Green et al. defined a generalization of the positive relation algebra working on K-relations, which we denote here by ARA.[1] When we restrict ARA to arities

[1] ARA stands for Annotated-Relation Algebra, as the elements from K that a K-relation assigns to its tuples are usually viewed as annotations.

of at most 3, which we denote by ARA(3), we obtain an analogue to FO(3) mentioned above. So, ARA provides a suitable scenario to reinvestigate, in a data model with numerical values, the equivalence between the algebra of binary relations and FO(3). In this paper, we make the following contributions.

1. We define a suitable generalization, to K-relations, of the composition operation of classical binary relations. When we add this composition operator to ARA, but restrict arities to at most two, we obtain a natural query language for matrices. We refer to this language here as "ARA(2) plus composition".
2. We show that ARA(2) plus composition actually coincides with the matrix query language MATLANG, introduced by two of the present authors with Geerts and Weerwag [4] in an attempt to formalize the set of common matrix operations found in numerical software packages.
3. We show that a matrix query is expressible in ARA(3) if and only if it is expressible in MATLANG, thus providing an analogue to the classical result about FO(3) and the algebra of binary relations. More generally, for any arity r, we show that an r-ary query over r-ary K-relations is expressible in ARA$(r+1)$ if and only if it is expressible in ARA(r) plus composition. For this result, we need the assumption that K is commutative. We stress that the proof is not a trivial adaptation of the proof of the classical result, because we can no longer rely on familiar classical properties like idempotence of union and join.

ARA has been a very influential vehicle for data provenance.[2] The elements from K are typically viewed as annotations, or as identifiers, and the semantics of ARA operations was originally designed to show how these annotations are propagated in the results of data manipulations. Other applications, apart from provenance, have been identified from the outset, such as security levels, or probabilities [7]. By doing the present work, we have understood that ARA can moreover serve as a fully-fledged query language for tensors (multidimensional arrays), and matrices in particular. This viewpoint is backed by the recent interest in processing Functional Aggregate Queries (FAQ [2,3], also known as AJAR [10]). Indeed, FAQ and AJAR correspond to the project-join fragment of ARA, without self-joins.

This is a revised and extended version of the conference paper "On matrices and K-relations" presented at FoIKS 2020 [5]. The conference version does not contain the proofs of the presented results. The current version is a fully self-contained version to which these proofs have been added. In addition, we have expanded the discussion on complexity issues connected to the translations between various languages considered in this work.

The paper is further organized as follows. Section 2 recalls the data model of K-relations and the associated query language ARA. Section 3 presents the result on ARA$(r+1)$ and ARA(r) plus composition. Section 4 relates ARA(2) plus composition to MATLANG. Section 5 draws conclusions, discusses related work, and proposes directions for further research.

[2] The paper by Green et al. [7] received the PODS 2017 test-of-time award.

2 Annotated-Relation Algebra

In this section, we start with some preliminaries before introducing the Annotated-Relation Algebra, or ARA for short. We also identify some identities on ARA expressions that are useful in later sections.

By *function* we will always mean a total function. For a function $f\colon X \to Y$ and $Z \subseteq X$, the *restriction* of f to Z, denoted by $f|_Z$, is the function $Z \to Y$ where $f|_Z(x) = f(x)$ for all $x \in Z$.

Recall that a *semiring* K is a set equipped with two binary operations, addition ($+$) and multiplication ($*$), such that (1) addition is associative, commutative, and has an identity element 0; (2) multiplication is associative, has an identity element 1, and has 0 as an annihilating element; and (3) multiplication distributes over addition. A semiring is called *commutative* if multiplication is commutative.

Proviso In the remainder of this paper, the presence of a semiring K is implicitly assumed. Unless where explicitly specified otherwise, K need not be commutative.

From the outset, we also fix countable infinite sets **rel**, **att**, and **dom**, the elements of which are called *relation names*, *attributes*, and *domain elements*, respectively. We assume the existence of an equivalence relation "$\sim$" on **att** with an infinite number of equivalence classes each of which is infinite. Let A and B be attributes. If $A \sim B$, we say that A and B are *compatible*. Intuitively, compatible attributes must be assigned the same domains, which will be formalized soon. A function $f\colon X \to Y$ with X and Y sets of attributes is called *compatible* if, for all $A \in X$, A and $f(A)$ are compatible.

A *relation schema* is a finite subset of **att**. A *database schema* is a function $\mathcal{S}\colon N \to \mathcal{P}_{\text{fin}}(\mathbf{att})$ from a finite set N of relation names to the set of all finite subsets of **att**, assigning a relation schema $\mathcal{S}(R)$ to each $R \in N$. We call the relation names in N also the *relation names of* $\mathcal{S}$. The *arity of a relation name* R of $\mathcal{S}$ is the cardinality $|\mathcal{S}(R)|$ of its schema. The *arity of the database schema* $\mathcal{S}$ is the largest arity among its relation names.

We now recursively define the expressions of the *Annotated-Relation Algebra*, abbreviated by ARA, syntactically. In the process, we assign a relation schema to each ARA expression by extending $\mathcal{S}$ from relation names to arbitrary ARA expressions. The ARA *expressions* over a database schema $\mathcal{S}$ is the smallest set of expressions that can be created using the following rules.

Relation name. A relation name R of $\mathcal{S}$ is an ARA expression over $\mathcal{S}$.
One. If e is an ARA expression over $\mathcal{S}$, then $\mathbf{1}(e)$ is an ARA expression over $\mathcal{S}$, and $\mathcal{S}(\mathbf{1}(e)) := \mathcal{S}(e)$.
Union. If e_1 and e_2 are ARA expressions over $\mathcal{S}$ with $\mathcal{S}(e_1) = \mathcal{S}(e_2)$ then $e_1 \cup e_2$ is an ARA expression over $\mathcal{S}$, and $\mathcal{S}(e_1 \cup e_2) := \mathcal{S}(e_1) = \mathcal{S}(e_2)$.
Projection If e is an ARA expression over $\mathcal{S}$ and $Y \subseteq \mathcal{S}(e)$, then $\pi_Y(e)$ is an ARA expression over $\mathcal{S}$, and $\mathcal{S}(\pi_Y(e)) := Y$.

Selection. If e is an ARA expression over $\mathcal{S}$, $Y \subseteq \mathcal{S}(e)$, and the elements of Y are mutually compatible, then $\sigma_Y(e)$ is an ARA expression over $\mathcal{S}$, and $\mathcal{S}(\sigma_Y(e)) := \mathcal{S}(e)$.

Renaming. If e is an ARA expression over $\mathcal{S}$ and $\varphi\colon \mathcal{S}(e) \to Y$ is a compatible one-to-one correspondence with $Y \subseteq \mathbf{att}$, then $\rho_\varphi(e)$ is an ARA expression over $\mathcal{S}$, and $\mathcal{S}(\rho_\varphi(e)) := Y$.

Join. If e_1 and e_2 are ARA expressions over $\mathcal{S}$, then $e_1 \bowtie e_2$ is an ARA expression over $\mathcal{S}$, and $\mathcal{S}(e_1 \bowtie e_2) := \mathcal{S}(e_1) \cup \mathcal{S}(e_2)$.

The *arity of an* ARA *expression* e over $\mathcal{S}$ is the cardinality $|\mathcal{S}(e)|$ of its schema.

Example 1 From a purely syntactical point of view, let $\mathcal{S}$ be a database schema on $N = \{\text{no\_courses}, \text{course\_fee}\}$ with $\mathcal{S}(\text{no\_courses}) = \{\text{student}, \text{dptm}\}$ and $\mathcal{S}(\text{course\_fee}) = \{\text{dptm}\}$. Hence, the arity of no_courses is 2 and the arity of course_fee is 1. We shall give meaning to the above in Example 2, after we look at databases and relations from a semantical point of view, below.

Let $\pi_{\{\text{student}\}}(\text{no\_courses} \bowtie \text{course\_fee})$ be an ARA expression over $\mathcal{S}$. This expression has schema $\{\text{student}\}$ and arity 1. We come back to the semantics of this expression in Example 3.

We now turn to semantics. A *domain assignment* is a function $D\colon \mathbf{att} \to \mathcal{D}$, where $\mathcal{D}$ is a set of nonempty finite subsets of $\mathbf{dom}$, such that, for compatible attributes A and B, $D(A) = D(B)$. Let X be a relation schema. A *tuple* over X with respect to D is a function $t\colon X \to \mathbf{dom}$ such that, for all $A \in X$, $t(A) \in D(A)$. We denote by $\mathcal{T}_D(X)$ the set of tuples over X with respect to D. Note that $\mathcal{T}_D(X)$ is finite. A *relation* r over X with respect to D is a function $r\colon \mathcal{T}_D(X) \to K$. So a relation annotates every tuple over X with respect to D with a value from K. If $\mathcal{S}$ is a database schema, then an *instance* $\mathcal{I}$ *of* $\mathcal{S}$ *with respect to* D is a function that assigns to every relation name R of $\mathcal{S}$ a relation $\mathcal{I}(R)\colon \mathcal{T}_D(\mathcal{S}(R)) \to K$.

Remark 1 In practice, a domain assignment need only be defined on the attributes that are used in the database schema (and on attributes compatible with these attributes). Thus, it can be specified finitely. While, here, we have chosen to keep the notions of domain assignment and instance separate, it may be argued that it is perhaps more natural to think of the domain assignment as being part of the instance.

Example 2 We wish to record for a university both the number of courses each student takes in each department and the unit fee for a course in each department. For that purpose, we shall use the database schema $\mathcal{S}$ defined in Example 1. Let K be the semiring of integers. If $\mathcal{I}$ is an instance of $\mathcal{S}$, then the relation $\mathcal{I}(\text{no\_courses})$ must annotate each pair of a student and a department with the number of courses taken by that student in that department. Likewise, the relation $\mathcal{I}(\text{course\_fee})$ must annotate each department with the unit fee for a course in that department. Let D be a domain assignment with $D(\text{student}) = \{\text{Alice}, \text{Bob}\}$ and $D(\text{dptm}) = \{\text{CS}, \text{Math}, \text{Bio}\}$. One particular database instance $\mathcal{I}$ of $\mathcal{S}$ with respect to D is shown in Figure 1.

$\mathcal{I}(\text{no\_courses}) =$

student	dptm	K
Alice	CS	5
Alice	Math	2
Alice	Bio	0
Bob	CS	2
Bob	Math	1
Bob	Bio	3

$\mathcal{I}(\text{course\_fee}) =$

dptm	K
CS	300
Math	250
Bio	330

Fig. 1 Example of a database instance.

We now define the relation $\mathbf{1}_X^D$, as well as how the generalizations of the classical operations from the positive relational algebra (which we encountered as the constructors in ARA expressions) work on relations.

One. Let X be a relation schema. The relation $\mathbf{1}_X^D \colon \mathcal{T}_D(X) \to K$ with schema X is defined by $\mathbf{1}_X^D(t) = 1$.

Union. Let $r_1, r_2 \colon \mathcal{T}_D(X) \to K$ be relations with the same schema X. The relation $r_1 \cup r_2 \colon \mathcal{T}_D(X) \to K$ with schema X is defined by $(r_1 \cup r_2)(t) = r_1(t) + r_2(t)$.

Projection. Let $r \colon \mathcal{T}_D(X) \to K$ be a relation with schema X, and let $Y \subseteq X$. The relation $\pi_Y(r) \colon \mathcal{T}_D(Y) \to K$ with schema Y is defined by

$$\big(\pi_Y(r)\big)(t) = \sum_{\substack{t' \in \mathcal{T}_D(X),\\ t'|_Y = t}} r(t').$$

Selection. Let $r \colon \mathcal{T}_D(X) \to K$ be a relation with schema X, and let $Y \subseteq X$, such that the attributes of Y are mutually compatible. The relation $\sigma_Y(r) \colon \mathcal{T}_D(X) \to K$ over X is defined by

$$\big(\sigma_Y(r)\big)(t) = \begin{cases} r(t) & \text{if } t(A) = t(B) \text{ for all } A, B \in Y; \\ 0 & \text{otherwise.} \end{cases}$$

Renaming. Let $r \colon \mathcal{T}_D(X) \to K$ be a relation with schema X, and let $\varphi \colon X \to Y$ be a compatible one-to-one correspondence. The relation $\rho_\varphi(r) \colon \mathcal{T}_D(Y) \to K$ over Y is defined by $\big(\rho_\varphi(r)\big)(t) = r(t \circ \varphi)$.

Join. Let $r_1 \colon \mathcal{T}_D(X_1) \to K$ and $r_2 \colon \mathcal{T}_D(X_2) \to K$ be relations with schemas X_1 and X_2, respectively. The relation $r_1 \bowtie r_2 \colon \mathcal{T}_D(X_1 \cup X_2) \to K$ over X is defined by $(r_1 \bowtie r_2)(t) = r_1(t|_{X_1}) * r_2(t|_{X_2})$.

The above operations provide semantics for ARA in a natural manner. Formally, let $\mathcal{S}$ be a database schema, and D a domain assignment. The semantics of an ARA expression e over $\mathcal{S}$ with respect to D is a mapping which associates to an instance $\mathcal{I}$ of $\mathcal{S}$ with respect to D the *output* relation $e(\mathcal{I})$ with schema $\mathcal{S}(e)$, defined by the following rules.

Relation name. If R is a relation name, then $R(\mathcal{I}) := \mathcal{I}(R)$.

One. If e is an ARA expression over $\mathcal{S}$, then $\big(\mathbf{1}(e)\big)(\mathcal{I}) := \mathbf{1}_{\mathcal{S}(e)}^D$.

$(\pi_{\text{student}}(\text{no\_courses} \bowtie \text{course\_fee}))(\mathcal{I}) =$

student	K
Alice	2000
Bob	1840

Fig. 2 The output relation for the ARA expression $\pi_{\text{student}}(\text{no\_courses} \bowtie \text{course\_fee})$ applied to the database instance $\mathcal{I}$ shown in Figure 1.

Union. If e_1 and e_2 are ARA expressions over $\mathcal{S}$ with $\mathcal{S}(e_1) = \mathcal{S}(e_2)$, then $(e_1 \cup e_2)(\mathcal{I}) := e_1(\mathcal{I}) \cup e_2(\mathcal{I})$.

Projection If e is an ARA expression over $\mathcal{S}$ and $Y \subseteq \mathcal{S}(e)$, then $\big(\pi_Y(e)\big)(\mathcal{I}) := \pi_X(e(\mathcal{I}))$.

Selection. If e is an ARA expression over $\mathcal{S}$, $Y \subseteq \mathcal{S}(e)$, and the attributes of Y are mutually compatible, then $\big(\sigma_Y(e)\big)(\mathcal{I}) := \sigma_Y(e(\mathcal{I}))$.

Renaming. If e is an ARA expression over $\mathcal{S}$ and $\varphi\colon \mathcal{S}(e) \to Y$ is a compatible one-to-one correspondence with $Y \subseteq \mathbf{att}$, then $\big(\rho_\varphi(e)\big)(\mathcal{I}) := \rho_\varphi(e(\mathcal{I}))$.

Join. If e_1 and e_2 are ARA expressions over $\mathcal{S}$, then $(e_1 \bowtie e_2)(\mathcal{I}) := e_1(\mathcal{I}) \bowtie e_2(\mathcal{I})$.

Example 3 We continue with Examples 1 and 2. Consider again the ARA expression $\pi_{\{\text{student}\}}(\text{no\_courses} \bowtie \text{course\_fee})$ over database scheme $\mathcal{S}$. Let $\mathcal{I}$ be an instance of $\mathcal{S}$ with respect to domain asignment D. From the above rules, it follows that $(\text{no\_courses} \bowtie \text{course\_fee})(\mathcal{I})$ annotates each pair of a student and a department with the product of the number of courses taken by that student in that department and the unit fee for a course in that department, i.e., with the total fee for the courses taken by that student in that department. The projection $\pi_{\{\text{student}\}}$ then aggregates these subtotals per student over all departments. Hence, $\big(\pi_{\{\text{student}\}}(\text{no\_courses} \bowtie \text{course\_fee})\big)(\mathcal{I})$ annotates each student with the total fee for all courses taken by that student. Figure 2 shows the output relation for the ARA expression $\pi_{\text{student}}(\text{no\_courses} \bowtie \text{course\_fee})$ applied to the database instance $\mathcal{I}$ shown in Figure 1.

Remark 2 The language ARA is a slight variation of the K-annotated relational algebra as originally defined by Green et al. [7] to better suit our purposes, i.e., comparing ARA to MATLANG in Section 4.

First, the original definition does not have a domain assignment $D\colon \mathbf{att} \to \mathcal{D}$ but instead a single domain common to all attributes (and it therefore also does not have a compatibility relation $\sim$). As such, the original definition corresponds to the case where database schemas and ARA expressions use only mutually compatible attributes, which is too restrictive for our purposes. Second, we focus on equality selections, while the original paper does not fix the allowed selection predicates. Third, and finally, we extended the original definition with one-relations.

The following observations, to the effect that some (but not all) classical relational-algebra equivalences carry over to the K-annotated setting, were originally made by Green et al. (They are not affected by the small differences between their formalism and ours, outlined in Remark 2.)

Proposition 1 ([7, Proposition 3.4]) *The following properties and identities hold, where, for each given identity, we assume that the left-hand side of that identity is well defined.*

- *Union is associative and commutative.*
- *Join is associative and distributive over union, i.e.,* $(r_1 \cup r_2) \bowtie r_3 = (r_1 \bowtie r_3) \cup (r_2 \bowtie r_3)$.
- *Any two selections commute.*
- *Projection and selection commute if projection retains the attributes on which selection takes place.*
- *Projection distributes over union, i.e.,* $\pi_Y(r_1 \cup r_2) = \pi_Y(r_1) \cup \pi_Y(r_2)$.
- *Selection distributes over union, i.e.,* $\sigma_Y(r_1 \cup r_2) = \sigma_Y(r_1) \cup \sigma_Y(r_2)$.
- *Selection and join commute in the sense that* $\sigma_Y(r_1) \bowtie r_2 = \sigma_Y(r_1 \bowtie r_2)$ *and* $r_1 \bowtie \sigma_Y(r_2) = \sigma_Y(r_1 \bowtie r_2)$.
- *If K is commutative, then join is commutative.*

Note that idempotence of union and of join, i.e., $r \bowtie r = r \cup r = r$, which holds for the classical relational algebra, does *not* in general hold for ARA.

We supplement Proposition 1 with the following properties.

Lemma 1 *Let* $r_1 \colon \mathcal{T}_D(X_1) \to K$ *and* $r_2 \colon \mathcal{T}_D(X_2) \to K$.

1. *If* $X_1 \cap X_2 \subseteq X \subseteq X_1 \cup X_2$, *then* $\pi_X(r_1 \bowtie r_2) = \pi_{X \cap X_1}(r_1) \bowtie \pi_{X \cap X_2}(r_2)$.
2. *If* $Y_1, Y_2 \subseteq X_1$ *where* $Y_1 \cap Y_2 \neq \emptyset$ *and the attributes of* Y_1 *and of* Y_2 *are mutually compatible, then* $\sigma_{Y_2}(\sigma_{Y_1}(r_1)) = \sigma_{Y_1 \cup Y_2}(r_1)$.
3. *If* $\varphi \colon X_1 \cup X_2 \to X$ *is a compatible one-to-one correspondence, then* $\rho_\varphi(r_1 \bowtie r_2) = \rho_{\varphi|_{X_1}}(r_1) \bowtie \rho_{\varphi|_{X_2}}(r_2)$. *If moreover* $X_1 = X_2$, *then* $\rho_\varphi(r_1 \cup r_2) = \rho_\varphi(r_1) \cup \rho_\varphi(r_2)$.
4. *If* $Y \subseteq X_1$ *and* $\varphi \colon X_1 \to X$ *is a compatible one-to-one correspondence, then* $\rho_\varphi(\sigma_Y(r_1)) = \sigma_{\varphi(Y)}(\rho_\varphi(r_1))$, *where* $\varphi(Y) = \{\varphi(y) \mid y \in Y\}$.

Proof 1. Both left- and right-hand side are functions from $\mathcal{T}_D(X)$ to K, as $(X \cap X_1) \cup (X \cap X_2) = X \cap (X_1 \cup X_2) = X$. To prove that they are equal, let t be any tuple in $\mathcal{T}_D(X)$. Then,

$$\big(\pi_X(r_1 \bowtie r_2)\big)(t) = \sum_{\substack{t' \in \mathcal{T}_D(X_1 \cup X_2), \\ t'|_X = t}} (r_1 \bowtie r_2)(t') = \sum_{\substack{t' \in \mathcal{T}_D(X_1 \cup X_2), \\ t'|_X = t}} r_1(t'|_{X_1}) * r_2(t'|_{X_2}).$$

Since $X_1 \cap X_2 \subseteq X$, all tuples t' in the latter sum agree on $X_1 \cap X_2$. Hence, we can apply distributivity of $*$ over $+$ to rewrite that sum to

$$\begin{aligned}
&\Big(\sum_{\substack{t'_1 \in \mathcal{T}_D(X_1), \\ t'_1|_{X \cap X_1} = t|_{X \cap X_1}}} r_1(t'_1)\Big) * \Big(\sum_{\substack{t'_2 \in \mathcal{T}_D(X_2), \\ t'_2|_{X \cap X_2} = t|_{X \cap X_2}}} r_2(t'_2)\Big) \\
&\qquad = \big(\pi_{X \cap X_1}(r_1)\big)(t|_{X \cap X_1}) * \big(\pi_{X \cap X_2}(r_2)\big)(t|_{X \cap X_2}) \\
&\qquad = \big(\pi_{X \cap X_1}(r_1) \bowtie \pi_{X \cap X_2}(r_2)\big)(t),
\end{aligned}$$

as was to be shown.

2. Both left- and right-hand side are functions from $\mathcal{T}_D(X_1)$ to K. To prove that they are equal, let t be any tuple in $\mathcal{T}_D(X_1)$. Then,
$$\big(\sigma_{Y_2}(\sigma_{Y_1}(r_1))\big)(t) = \begin{cases} \big(\sigma_{Y_1}(r_1)\big)(t) & \text{if } t(A) = t(B) \text{ for all } A, B \in Y_2; \\ 0 & \text{otherwise.} \end{cases}$$
Furthermore,
$$\big(\sigma_{Y_1}(r_1)\big)(t) = \begin{cases} r_1(t) & \text{if } t(A) = t(B) \text{ for all } A, B \in Y_1; \\ 0 & \text{otherwise.} \end{cases}$$
Combining the above, we see that
$$\begin{aligned} \big(\sigma_{Y_2}(\sigma_{Y_1}(r_1))\big)(t) &= \begin{cases} r_1(t) & \text{if } t(A) = t(B) \text{ for all } A, B \in Y_1 \cup Y_2; \\ 0 & \text{otherwise} \end{cases} \\ &= \big(\sigma_{Y_1 \cup Y_2}(r_1)\big)(t). \end{aligned}$$

3. In the identity involving join, both left- and right-hand side are functions from $\mathcal{T}_D(X)$ to K. To prove that they are equal, let t be any tuple in $\mathcal{T}_D(X)$. Let $u = t \circ \varphi$, which is a tuple in $\mathcal{T}_D(X_1 \cup X_2)$. Then,
$$\big(\rho_\varphi(r_1 \bowtie r_2)\big)(t) = (r_1 \bowtie r_2)(t \circ \varphi) = (r_1 \bowtie r_2)(u) = r_1(u|_{X_1}) * r_2(u|_{X_2}).$$
Obviously, $u|_{X_1} = t|_{\varphi|_{X_1}(X_1)} \circ \varphi|_{X_1}$ and $u|_{X_2} = t|_{\varphi|_{X_2}(X_2)} \circ \varphi|_{X_2}$. Furthermore, $\rho_{\varphi|_{X_1}}(r_1)$ is a function from $\mathcal{T}_D(\varphi|_{X_1}(X_1))$ to K and $\rho_{\varphi|_{X_2}}(r_2)$ is a function from $\mathcal{T}_D(\varphi|_{X_2}(X_2))$ to K. Hence,
$$\begin{aligned} r_1(u|_{X_1}) * r_2(u|_{X_2}) &= r_1(t|_{\varphi|_{X_1}(X_1)} \circ \varphi|_{X_1}) * r_2(t|_{\varphi|_{X_2}(X_2)} \circ \varphi|_{X_2}) \\ &= \big(\rho_{\varphi|_{X_1}}(r_1)\big)(t|_{\varphi|_{X_1}(X_1)}) * \big(\rho_{\varphi|_{X_2}}(r_2)\big)(t|_{\varphi|_{X_2}(X_2)}) \\ &= \big(\rho_{\varphi|_{X_1}}(r_1) \bowtie \rho_{\varphi|_{X_2}}(r_2)\big)(t) \end{aligned}$$
In the identity involving union, both left- and right-hand side are again functions from $\mathcal{T}_D(X)$ to K. To prove that they are equal, let t be as above. Since $X_1 = X_2$, φ is a function from $X_1 = X_2$ to X. Then,
$$\begin{aligned} \big(\rho_\varphi(r_1 \cup r_2)\big)(t) &= (r_1 \cup r_2)(t \circ \varphi) \\ &= r_1(t \circ \varphi) + r_2(t \circ \varphi) \\ &= \big(\rho_\varphi(r_1)\big)(t) + \big(\rho_\varphi(r_2)\big)(t) \\ &= \big(\rho_\varphi(r_1) \cup \rho_\varphi(r_2)\big)(t). \end{aligned}$$

4. Both left- and right-hand side of this identity are functions from $\mathcal{T}_D(X)$ to K. To prove that they are equal, let t be any tuple in $\mathcal{T}_D(X)$. Then,
$$\begin{aligned} \big(\rho_\varphi(\sigma_Y(r_1))\big)(t) &= \big(\sigma_Y(r_1)\big)(t \circ \varphi) \\ &= \begin{cases} r_1(t \circ \varphi) & \text{if } (t \circ \varphi)(A) = (t \circ \varphi)(B) \text{ for all } A, B \in Y; \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

The identity follows from the observations that $r_1(t \circ \varphi) = \big(\rho_\varphi(r_1)\big)(t)$, $(t \circ \varphi)(A) = t(\varphi(A))$, and $(t \circ \varphi)(B) = t(\varphi(B))$. Since φ is one-to-one, $t(\varphi(A)) = t(\varphi(B))$ for all $A, B \in Y$ is equivalent to $t(A') = t(B')$ for all $A', B' \in \varphi(Y)$.

□

We also use the derived operation of projecting away one attribute, $\hat{\pi}_A(e)$, which is a shorthand for $\pi_{\mathcal{S}(e)\setminus\{A\}}(e)$ if $A \in \mathcal{S}(e)$. Note that, conversely, $\pi_X(e) = (\hat{\pi}_{A_m} \cdots \hat{\pi}_{A_1})(e)$ where $X = \mathcal{S}(e) \setminus \{A_1, \dots, A_m\}$ and the A_i's are pairwise distinct. Hence, "standard" projection and projecting away one attribute are interchangable as far as the construction of ARA expressions is concerned. We shall take advantage of this in proofs, as projecting away one attribute has often the advantage of having to deal only with that particular attribute.

3 Annotated-Relation Algebra with Composition

In this section, we define an operation called k-composition and show that augmenting ARA by composition allows one to reduce the required arity of the relations that are computed in subexpressions. The intuition is to provide a generalization of classical composition of two binary relations to annotated relations, so that we can compose up to k relations of arity up to k. Specifically, the classical composition of a binary relation r with a binary relation s amounts to viewing these relations as relations over schemas $\{A, B\}$ and $\{A, C\}$, respectively, and performing $\hat{\pi}_A(r \bowtie s)$. Thus, we arrive at the following generalization.

Definition 1 Let k be a nonnegative integer and let $l \in \{1, \dots, k\}$. Let $r_i \colon \mathcal{T}_D(X_i) \to K$ for $i \in \{1, \dots, l\}$, let $X = X_1 \cup \cdots \cup X_l$, and let $A \in X_1 \cap \cdots \cap X_l$.

Define the *k-composition* $\zeta_{A,k}(r_1, \dots, r_l) \colon \mathcal{T}_D(X \setminus \{A\}) \to K$ as

$$\big(\zeta_{A,k}(r_1, \dots, r_l)\big)(t) = \big(\hat{\pi}_A(r_1 \bowtie \cdots \bowtie r_l)\big)(t),$$

for all $t \in \mathcal{T}_D(X \setminus \{A\})$.

Note that $\zeta_{A,k}$ takes at most k arguments. We emphasize that $\zeta_{A,k}$ is defined as a new operator (albeit one that can be defined by an ARA expression) and not as a shorthand for an ARA expression.

We denote by ARA $+ \zeta_k$ the language obtained by extending ARA with k-composition. That is, if $e_1, \dots, e_l$ are ARA $+ \zeta_k$ expressions with $l \le k$ and $A \in \mathcal{S}(e_1) \cap \cdots \cap \mathcal{S}(e_l)$, then $e = \zeta_{A,k}(e_1, \dots, e_l)$ is an ARA $+ \zeta_k$ expression. Also, we let $\mathcal{S}(e) := (\mathcal{S}(e_1) \cup \cdots \cup \mathcal{S}(e_l)) \setminus \{A\}$.

Let k be a nonnegative integer. We denote by ARA(k) the fragment of ARA in which the database schemas are restricted to arity at most k and each subexpression has arity at most k. In particular, join $e_1 \bowtie e_2$ is only allowed if $|\mathcal{S}(e_1 \bowtie e_2)| \le k$. The fragment (ARA $+ \zeta_k)(k)$ is defined similarly.

From Definition 1, it follows that $(\mathsf{ARA} + \zeta_k)(k)$ is subsumed by $\mathsf{ARA}(k + 1)$. Indeed, let e be an $(\mathsf{ARA} + \zeta_k)(k)$ expression. We obtain an equivalent ARA expression e' by subsequently replacing each subexpression of the form $\zeta_{A,k}(e_1, \dots, e_l)$ by $\hat{\pi}_A(e_1 \bowtie \cdots \bowtie e_l)$. Since the former has arity at most k, so has the latter. However, each such replacement introduces one subexpression not equivalent to a subexpression of e, namely $e_1 \bowtie \cdots \bowtie e_l$, of which we can only say that it has arity at most $k + 1$. Hence, e' is in $\mathsf{ARA}(k + 1)$.

One of our main results (Corollary 1) provides the converse inclusion, when the database schemas and outputs are restricted to arity at most k. To this end, we establish a normal form for ARA expressions (Theorem 1). First, we prove the following technical identity that we shall use to show Theorem 1.

Lemma 2 *Let $r_1, \dots, r_n$ be relations with relation schemas $X_1, \dots, X_n$, respectively, and with respect to a domain assignment D. Assume that $A, B \in X_1 \cup \cdots \cup X_n$ are distinct and compatible. Define, for $i \in \{1, \dots, n\}$,*

$$r_i' := \begin{cases} r_i & \text{if } A \notin X_i; \\ \rho_\varphi(r_i) & \text{if } A \in X_i, B \notin X_i; \\ \hat{\pi}_A(\sigma_{\{A,B\}}(r_i)) & \text{if } A, B \in X_i, \end{cases}$$

where φ is the one-to-one correspondence from X_i to $(X_i \setminus \{A\}) \cup \{B\}$ that maps A to B and keeps the remaining attributes fixed. Then,

$$\hat{\pi}_A(\sigma_{\{A,B\}}(r_1 \bowtie \cdots \bowtie r_n)) = r_1' \bowtie \cdots \bowtie r_n'.$$

Proof Let X be a finite set of attributes with $A, B \in X$ distinct and compatible. Let $r \colon \mathcal{T}_D(X) \to K$ be a relation and $t \in \mathcal{T}_D(X \setminus \{A\})$.

We have

$$\big(\hat{\pi}_A(\sigma_{\{A,B\}}(r))\big)(t) = \sum_{\substack{u \in \mathcal{T}_D(X), \\ u|_{X \setminus \{A\}} = t}} \big(\sigma_{\{A,B\}}(r)\big)(u) = \sum_{\substack{u \in \mathcal{T}_D(X), \\ u|_{X \setminus \{A\}} = t, \\ u(A) = u(B)}} r(u) = r(\tilde{t}), \tag{1}$$

where $\tilde{t} \in \mathcal{T}_D(X)$ is $\tilde{t} = t \cup \{(A, t(B))\}$. Thus, $\tilde{t}$ is obtained from t by adding attribute A with value $t(B)$. Indeed, the last summation of (1) is over a single tuple u, namely $u = \tilde{t}$.

In particular, applying (1) to $r_1 \bowtie \cdots \bowtie r_n$, we obtain

$$\big(\hat{\pi}_A(\sigma_{\{A,B\}}(r_1 \bowtie \cdots \bowtie r_n))\big)(t) = (r_1 \bowtie \cdots \bowtie r_n)(\tilde{t}) = r_1(\tilde{t}|_{X_1}) * \cdots * r_n(\tilde{t}|_{X_n}).$$

Denote the schemas of the relations $r_1', \dots, r_n'$ by $X_1', \dots, X_n'$, respectively. Let $i \in \{1, \dots, n\}$. We distinguish three cases.

- If $A \notin X_i$, then $\tilde{t}|_{X_i} = t|_{X_i}$. Hence $r_i(\tilde{t}|_{X_i}) = r_i'(t|_{X_i'})$.
- If $A \in X_i$ and $B \notin X_i$, then $\tilde{t}|_{X_i} = t|_{(X_i \setminus \{A\}) \cup \{B\}} \circ \varphi = t|_{X_i'} \circ \varphi$. Hence, $r_i(\tilde{t}|_{X_i}) = \big(\rho_\varphi(r_i)\big)(t|_{X_i'}) = r_i'(t|_{X_i'})$.
- If $A, B \in X_i$, then, by (1) but now applied to r_i and $t|_{X_i \setminus \{A\}}$, we have $r_i(\tilde{t}|_{X_i}) = \big(\hat{\pi}_A(\sigma_{\{A,B\}}(r_i))\big)(t|_{X_i \setminus \{A\}}) = r_i'(t|_{X_i'})$.

In all three cases, we obtain $r_i(\tilde{t}|_{X_i}) = r'_i(t|_{X'_i})$. Consequently,

$$r_1(\tilde{t}|_{X_1}) * \cdots * r_n(\tilde{t}|_{X_n}) = r'_1(t|_{X'_1}) * \cdots * r'_n(t|_{X'_n}) = (r'_1 \bowtie \cdots \bowtie r'_n)(t).$$

$\square$

We use the following terminology. Let $\mathcal{F}$ be any family of expressions. A *selection of $\mathcal{F}$-expressions* is an expression of the form $\sigma_{Y_n} \cdots \sigma_{Y_1}(f)$, where f is an $\mathcal{F}$-expression and $n \geq 0$. Note the slight abuse of terminology as we allow multiple selection operations. Furthermore, when we say that e is a *union of $\mathcal{F}$-expressions* or a *join of $\mathcal{F}$-expressions*, we allow e to be just a single expression in $\mathcal{F}$ (so union and join may be skipped).

We are now ready to state and prove one of the main results of this paper. This result is inspired by the classic equivalence of FO(3) and the algebra of binary relations [18]. (A compact proof of this equivalence is given by Marx and Venema [14, Theorem 3.4.5, Claim 2]; a self-contained exposition is also available [19].)

Two ARA expressions e_1 and e_2 over the same database schema are called *equivalent*, denoted $e_1 \equiv e_2$, if they yield the same output relation for every domain assignment and every database instance respecting that domain assignment.

Theorem 1 *Let $\mathcal{S}$ be a database schema of arity at most k, and assume that K is commutative. Every $\mathsf{ARA}(k+1)$ expression over $\mathcal{S}$ is equivalent to a union of selections of joins of $(\mathsf{ARA} + \zeta_k)(k)$ expressions over $\mathcal{S}$.*

Proof For brevity, if an expression is a union of selections of joins of $(\mathsf{ARA} + \zeta_k)(k)$ expressions over $\mathcal{S}$, then we say that this expression is in *normal form.*

We now prove that every $\mathsf{ARA}(k + 1)$ expression e is equivalent to an expression in normal form, by induction on the structure of e. (This approach works, since, by definition, a subexpression of an $\mathsf{ARA}(k+1)$ expression is also in $\mathsf{ARA}(k + 1)$.)

Relation names. Let e be a relation name R of $\mathcal{S}$. Since relation names of $\mathcal{S}$ have arity at most k, e is already in normal form.

One. Let e be $\mathbf{1}(e')$, where e' is equivalent to an expression in normal form. Let $e_1, \dots, e_n$ be $(\mathsf{ARA}+\zeta_k)(k)$ expressions over $\mathcal{S}$ such that $e_1 \bowtie \cdots \bowtie e_n$ is a subexpression of this expression in normal form. Since unions and selections do not change the schema of an expression, $\mathcal{S}(e') = \mathcal{S}(e_1 \bowtie \cdots \bowtie e_n)$, and, hence, $e = \mathbf{1}(e') \equiv \mathbf{1}(e_1 \bowtie \cdots \bowtie e_n) \equiv \mathbf{1}(e_1) \bowtie \cdots \bowtie \mathbf{1}(e_n)$. Since, for $i = 1, \dots, n$, $\mathcal{S}(\mathbf{1}(e_i)) = \mathcal{S}(e_i)$, $\mathbf{1}(e_i)$—just like e_i—is of arity at most k. We may therefore conclude that e is equivalent to an expression in normal form.

Union. Let e be $e_1 \cup e_2$, where e_1 and e_2 are equivalent to expressions in normal form. By definition, the union of expressions in normal form is again in normal form.

Join. Let e be $e_1 \bowtie e_2$, where e_1 and e_2 are equivalent to expressions in normal form. Since join distributes over union and since selection and join commute in the sense of Proposition 1, it follows that $e = e_1 \bowtie e_2$ is also equivalent to an expression in normal form.

Selection. Let e be $\sigma_Y(e')$, where e' is equivalent to an expression in normal form. Since selection distributes over union (Proposition 1), it follows that $e = \sigma_Y(e')$ is also equivalent to an expression in normal form.

Renaming. Let e be $\rho_\varphi(e')$, where e' is equivalent to an expression in normal form. Since renaming distributes over both union and join and since renaming and selection commute (Lemma 1), it follows that $e = \rho_\varphi(e')$ is equivalent to a union of selections of joins of renamings of $(\mathsf{ARA} + \zeta_k)(k)$ expressions. Since renaming preserves arity, renamings of $(\mathsf{ARA} + \zeta_k)(k)$ expressions are in turn $(\mathsf{ARA} + \zeta_k)(k)$ expressions. We may thus conclude that $e = \rho_\varphi(e')$ is equivalent to an expression in normal form.

Projection. Without loss of generality, let e be $\hat{\pi}_A(e')$, where e' is equivalent to an expression in normal form. Since projection distributes over union (Proposition 1), we may assume without loss of generality that e' is equivalent to $\sigma_{Y_m} \cdots \sigma_{Y_1}(f)$, with f a join of $(\mathsf{ARA} + \zeta_k)(k)$ expressions. By Proposition 1 and Lemma 1, we may assume that $Y_1, \dots, Y_m$ are pairwise disjoint. We may also assume that they are all of cardinality at least 2, since σ_Y on relations is the identity if $|Y| \leq 1$. We consider two cases.

1. $A \in Y_1 \cup \cdots \cup Y_m$. Since $Y_1, \dots, Y_m$ are pairwise disjoint, there is a unique $i \in \{1, \dots, m\}$ for which $A \in Y_i$. Since any two selections commute (Proposition 1), we may assume that $A \in Y_1$. Also by Proposition 1, $e = \hat{\pi}_A(e') \equiv \sigma_{Y_m} \cdots \sigma_{Y_2}(\hat{\pi}_A(\sigma_{Y_1}(f)))$. Since Y_1 is of cardinality at least 2, there exists $B \in Y_1$ distinct from A. Hence, by Lemma 1, $\sigma_{Y_1}(f) \equiv \sigma_{Y_1 \setminus \{A\}}(\sigma_{\{A,B\}}(f))$, and $\hat{\pi}_A(\sigma_{Y_1}(f)) \equiv \sigma_{Y_1 \setminus \{A\}}(\hat{\pi}_A(\sigma_{\{A,B\}}(f)))$. Now, by Lemma 2, $\hat{\pi}_A(\sigma_{\{A,B\}}(f))$ is equivalent to a join of $(\mathsf{ARA} + \zeta_k)(k)$ expressions, as was to be shown.
2. $A \notin Y_1 \cup \cdots \cup Y_m$. With a similar argument as in the former case, $e = \hat{\pi}_A(e') \equiv \sigma_{Y_m} \cdots \sigma_{Y_1}(\hat{\pi}_A(f))$. It remains to show that $\hat{\pi}_A(f)$ is a join of $(\mathsf{ARA} + \zeta_k)(k)$ expressions. Since e' is a subexpression of the $\mathsf{ARA}(k{+}1)$ expression e, e' in turn is in $\mathsf{ARA}(k{+}1)$, hence $|\mathcal{S}(e')| \leq k{+}1$. Moreover, $\mathcal{S}(f) = \mathcal{S}(e')$, hence $|\mathcal{S}(f)| \leq k + 1$. If $|\mathcal{S}(f)| \leq k$, then f itself is an $(\mathsf{ARA} + \zeta_k)(k)$ expression and so is $\hat{\pi}_A(f)$. So, assume that $|\mathcal{S}(f)| = k + 1$.

   Since join is commutative (because K is) and associative, we can regard f as a join of a *multi*set $F = \{f_1, \dots, f_n\}$ of $(\mathsf{ARA}+\zeta_k)(k)$ expressions. Now, for any two expressions e_1 and e_2 with $A \notin \mathcal{S}(e_1)$, we have that $\hat{\pi}_A(e_1 \bowtie e_2) \equiv e_1 \bowtie \hat{\pi}_A(e_2)$ (Lemma 1). Therefore, we may assume that, for all $i = 1, \dots, n$, $A \in \mathcal{S}(f_i)$. Let $\mathcal{S}_k^A(f)$ be the set of all k-element subsets of $\mathcal{S}(f)$ containing A. There exists a function $s\colon \{1, \dots, n\} \to \mathcal{S}_k^A(f)$ such that, for all $i = 1, \dots, n$, $\mathcal{S}(f_i) \subseteq s(i)$. Let $\mathcal{R}$ be the range of s. Thus, $|\mathcal{R}| \leq |\mathcal{S}_k^A(f)| = k$. Let, for $S \in \mathcal{R}$, $f_S := \bowtie_{i=1,\dots,n; s(i)=S} f_i$. In words, f_S is the join of all subexpressions f_i, $1 \leq i \leq n$, for which $s(i) = S$. Since $s(i) = S$ implies that $\mathcal{S}(f_i) \subseteq S$, it follows that $\mathcal{S}(f_S) \subseteq S$.

Hence, $|\mathcal{S}(f_S)| \leq |S| = k$. Therefore, f_S is an $(\mathsf{ARA}+\zeta_k)(k)$ expression. Since, for each $i = 1, \ldots, n$, f_i occurs in $f_{s(i)}$, it follows that $f \equiv \bowtie_{S\in\mathcal{R}} f_S$. Consequently, $\hat{\pi}_A(f) \equiv \hat{\pi}_A(\bowtie_{S\in\mathcal{R}} f_S) \equiv \zeta_{A,k}((f_S)_{S\in\mathcal{R}})$, by Definition 1. We thus obtain an $(\mathsf{ARA}+\zeta_k)(k)$ expression as desired.

□

Example 4 Assume that K is commutative and consider the $\mathsf{ARA}(3)$ expression $e = \pi_{\{B,C\}}(\sigma_{\{B,C\}}(R \bowtie R \bowtie S \bowtie T \bowtie \rho_\varphi(T)) \cup \sigma_{\{A,B\}}(R \bowtie S \bowtie T))$, where $\mathcal{S}(R) = \{A, B\}$, $\mathcal{S}(S) = \{B, C\}$, $\mathcal{S}(T) = \{A, C\}$ (A, B, C are pairwise distinct), and φ sends A to B and C to itself. The proof of Theorem 1 obtains an equivalent expression in normal form as follows.

$$\begin{aligned}
e &= \hat{\pi}_A(\sigma_{\{B,C\}}(R \bowtie R \bowtie S \bowtie T \bowtie \rho_\varphi(T)) \cup \sigma_{\{A,B\}}(R \bowtie S \bowtie T))\\
&\equiv \hat{\pi}_A(\sigma_{\{B,C\}}(R \bowtie R \bowtie S \bowtie T \bowtie \rho_\varphi(T))) \cup \hat{\pi}_A(\sigma_{\{A,B\}}(R \bowtie S \bowtie T))\\
&\equiv \sigma_{\{B,C\}}(\hat{\pi}_A(R \bowtie R \bowtie S \bowtie T \bowtie \rho_\varphi(T))) \cup \hat{\pi}_A(\sigma_{\{A,B\}}(R \bowtie S \bowtie T))\\
&\equiv \sigma_{\{B,C\}}(S \bowtie \rho_\varphi(T) \bowtie \hat{\pi}_A(R \bowtie R \bowtie T)) \cup \hat{\pi}_A(\sigma_{\{A,B\}}(R \bowtie S \bowtie T))\\
&\equiv \sigma_{\{B,C\}}(S \bowtie \rho_\varphi(T) \bowtie \zeta_{A,2}(R \bowtie R, T)) \cup \hat{\pi}_A(\sigma_{\{A,B\}}(R \bowtie S \bowtie T))\\
&\equiv \sigma_{\{B,C\}}(S \bowtie \rho_\varphi(T) \bowtie \zeta_{A,2}(R \bowtie R, T)) \cup \big(\hat{\pi}_A(\sigma_{\{A,B\}}(R)) \bowtie S \bowtie \rho_\varphi(T)\big).
\end{aligned}$$

The last expression is in the normal form since the subexpressions S, $\rho_\varphi(T)$, $\zeta_{A,2}(R \bowtie R, T)$, and $\hat{\pi}_A(\sigma_{\{A,B\}}(R))$ are all $(\mathsf{ARA}+\zeta_2)(2)$ expressions.

Note that we most likely cannot omit the "selections of" in Theorem 1. To see this for $k = 2$, consider the expression $\sigma_{\{A,C\}}(R \bowtie S)$ where R and S are relation names with $\mathcal{S}(R) = \{A, B\}$ and $\mathcal{S}(S) = \{B, C\}$. This expression is in $\mathsf{ARA}(3)$ and is also in normal form, as the relation names R and S constitute expressions in $(\mathsf{ARA}+\zeta_2)(2)$. We cannot see, however, how the selection operator $\sigma_{\{A,C\}}$ could be eliminated from this normal form.

Remark 3 Theorem 1 still holds if the **1** operator is omitted from the definition of ARA. Indeed, in the proof we can simply omit the case for the **1** operator, which is not used anywhere else.

In Theorem 1, we have shown that an $\mathsf{ARA}(k+1)$ expression is equivalent to a union of selections of joins of $(\mathsf{ARA}+\zeta_k)(k)$ expressions. Since union and selection do not change arity, a union of selections of joins of $(\mathsf{ARA}+\zeta_k)(k)$ expressions is an $(\mathsf{ARA}+\zeta_k)(k)$ expression if and only if these joins have arity at most k if and only the original $\mathsf{ARA}(k+1)$ expression has arity at most k. Hence, Theorem 1 yields the following corollary.

Corollary 1 *Let $\mathcal{S}$ be a database schema of arity at most k and assume that K is commutative. Every $\mathsf{ARA}(k+1)$ expression e over $\mathcal{S}$ of arity at most k is equivalent to an $(\mathsf{ARA}+\zeta_k)(k)$ expression over $\mathcal{S}$.*

Remark 4 Note that transforming an expression into the normal form of Theorem 1 may lead to an exponential increase in expression length. The reason is that the proof uses distributivity of join over union. Indeed, each time we replace an expression of the form $(e_1 \cup e_2) \bowtie e_3$ by $(e_1 \bowtie e_3) \cup (e_2 \bowtie e_3)$ there is a duplication of e_3. The proof of the classic translation of FO(3) to the algebra of binary relations also induces an exponential increase of expression length for similar reasons. A proof that this blowup is unavoidable remains open, both for our result and for the classical result (to the best of our knowledge).

4 Working with Matrices

In this section, we show that $(\mathsf{ARA} + \zeta_2)(2)$ is equivalent to a natural version of MATLANG [4]. As a consequence of Corollary 1, we then obtain that also ARA(3), with database schemas and output relations restricted to arity at most 2, is equivalent to MATLANG. We begin by recalling the definition of this language.

4.1 MATLANG

Let us fix the countable infinite sets **matvar** and **size**, where the latter has a distinguished element $1 \in \textbf{size}$. The elements of **matvar** are called *matrix variables* and the elements of **size** are called *size symbols*.

A *matrix schema* is a function $\mathcal{S}\colon V \to \textbf{size} \times \textbf{size}$ with $V \subseteq \textbf{matvar}$ both finite and nonempty. We write $(\alpha, \beta) \in \textbf{size} \times \textbf{size}$ also as $\alpha \times \beta$.

We now recursively define MATLANG expressions syntactically. In the process, we assign a matrix schema to each MATLANG expression by extending $\mathcal{S}$ from matrix variables to arbitrary MATLANG expressions. The MATLANG *expressions* over a matrix schema $\mathcal{S}$ is the smallest set of expressions that can be created by using the following rules.

Variable. A matrix variable M of $\mathcal{S}$ is a MATLANG expression over $\mathcal{S}$.

Transpose. If e is a MATLANG expression over $\mathcal{S}$ with $\mathcal{S}(e) = \alpha \times \beta$, then e^T is a MATLANG expression with $\mathcal{S}(e^T) := \beta \times \alpha$.

One-vector. If e is a MATLANG expression over $\mathcal{S}$ with $\mathcal{S}(e) = \alpha \times \beta$, then $\mathbf{1}(e)$ is a MATLANG expression with $\mathcal{S}(\mathbf{1}(e)) := \alpha \times 1$.

Diagonalization. If e is a MATLANG expression over $\mathcal{S}$ with $\mathcal{S}(e) = \alpha \times 1$, then $\mathsf{diag}(e)$ is a MATLANG expression with $\mathcal{S}(\mathsf{diag}(e)) := \alpha \times \alpha$.

Multiplication. If e_1 and e_2 are MATLANG expressions over $\mathcal{S}$ with $\mathcal{S}(e_1) = \alpha \times \beta$ and $\mathcal{S}(e_2) = \beta \times \gamma$, then $e_1 \cdot e_2$ is a MATLANG expression with $\mathcal{S}(e_1 \cdot e_2) := \alpha \times \gamma$.

Addition. If e_1 and e_2 are MATLANG expressions over $\mathcal{S}$ with $\mathcal{S}(e_1) = \mathcal{S}(e_2)$, then $e_1 + e_2$ is a MATLANG expression with $\mathcal{S}(e_1 + e_2) := \mathcal{S}(e_1) = \mathcal{S}(e_2)$.

Hadamard product. If e_1 and e_2 are MATLANG expressions over $\mathcal{S}$ with $\mathcal{S}(e_1) = \mathcal{S}(e_2)$, then $e_1 \circ e_2$ is a MATLANG expression with $\mathcal{S}(e_1 \circ e_2) := \mathcal{S}(e_1) = \mathcal{S}(e_2)$.

A *size assignment* is a function σ that assigns to each size symbol a strictly positive integer with $\sigma(1) = 1$. Let $\mathcal{M}$ be the set of all matrices over K. We say that $M \in \mathcal{M}$ *conforms* to $\alpha \times \beta \in \textbf{size} \times \textbf{size}$ by σ if M is a $\sigma(\alpha) \times \sigma(\beta)$-matrix.

$$\mathcal{I}(\text{no\_courses}) = \begin{pmatrix} 5 & 2 & 0 \\ 2 & 1 & 3 \end{pmatrix} \qquad \mathcal{I}(\text{course\_fee}) = \begin{pmatrix} 300 \\ 250 \\ 330 \end{pmatrix}$$

Fig. 3 An example of an instance of a matrix schema.

If $\mathcal{S}\colon V \to \textbf{size} \times \textbf{size}$ is a matrix schema, then an *instance of $\mathcal{S}$ with respect to σ* is a function $\mathcal{I}\colon V \to \mathcal{M}$ such that, for each $M \in V$, the matrix $\mathcal{I}(M)$ conforms to $\mathcal{S}(M)$ by σ.

Remark 5 In practice, a size assignment need only be defined on the size symbol that are used in the schema. Thus, it can be finitely specified. While, here, we have chosen to keep the notions of size assignment and instance separate, it may be argued that it is perhaps more natural to think of the size assignment as being part of the instance.

Example 5 Let K be the set of integers and let $\mathcal{S}$ be a matrix schema on the set of matrix variables $\{\text{no\_courses}, \text{course\_fee}\}$ with $\mathcal{S}(\text{no\_courses}) = \text{student} \times \text{dptm}$ and $\mathcal{S}(\text{course\_fee}) = \text{dptm} \times 1$. Now, let σ be a size assignment such that $\sigma(\text{student}) = 2$ and $\sigma(\text{dptm}) = 3$. An instance $\mathcal{I}$ of $\mathcal{S}$ with respect to σ is shown in Figure 3.

This matrix schema and instance may be compared to the database schema and instance exhibited in Examples 1 and 2 and Figure 1.

We now define how the operations which we encountered as the constructors in MATLANG expressions work on matrices.

Transpose, Multiplication, Addition. Matrix transpose, matrix multiplication, and matrix addition are defined in the usual way.

One-vector. Let M be a $m \times n$ matrix. Then, $\mathbf{1}(M)$ is the $m \times 1$ matrix (i.e., column vector) for which, for $i = 1, \dots, m$, $\big(\mathbf{1}(M)\big)_{i,1} = 1$.

Diagonalization. Let M be a $m \times 1$ matrix (i.e., a column vector). Then, $\mathsf{diag}(M)$ is the $m \times m$ (square) matrix for which, for $i, j = 1, \dots, m$,

$$\big(\mathsf{diag}(M)\big)_{i,j} = \begin{cases} M_{i,1} & \text{if } i = j; \\ 0 & \text{if } i \neq j. \end{cases}$$

Hadamard product Let M_1 and M_2 be $m \times n$ matrices. Then, $M_1 \circ M_2$ is the $m \times n$ matrix for which, for $i = 1, \dots, m$ and $j = 1, \dots, n$, $(M_1 \circ M_2)_{i,j} = (M_1)_{i,j} * (M_2)_{i,j}$.

The above operations provide semantics for ARA in a natural manner. Formally, let $\mathcal{S}$ be a matrix schema, and let σ be a size assignment. The semantics of a MATLANG expression over $\mathcal{S}$ with respect to σ is a mapping associating with an instance $\mathcal{I}$ of $\mathcal{S}$ with respect to σ the *output* matrix $e(\mathcal{I})$ conforming to $\mathcal{S}(e)$ by σ, defined by the following rules.

Variable. If M is a matrix variable, then $M(\mathcal{I}) := \mathcal{I}(M)$.

Mapping	MATLANG	$(\mathsf{ARA} + \zeta_2)(2)$
matrix variable $\rightarrow$ relation name	M	M
size symbol $\rightarrow$ attributes	α	row_α, col_α
element of **size** $\times$ **size** $\rightarrow$ relation schema	s	$\Gamma(s)$
matrix schema $\rightarrow$ database schema	$\mathcal{S}$	$\Gamma(\mathcal{S})$
size assignment $\rightarrow$ domain assignment	σ	$D(\sigma)$
matrix $\rightarrow$ relation	M	$\text{Rel}_{s,\sigma}(M)$
matrix instance $\rightarrow$ database instance	$\mathcal{I}$	$\text{Rel}_{s,\sigma}(\mathcal{I})$
MATLANG expression $\rightarrow$ $(\mathsf{ARA} + \zeta_2)(2)$ expression	e	$\Upsilon(e)$

Table 1 Symbol table for the simulation of MATLANG in $(\mathsf{ARA} + \zeta_2)(2)$.

Transpose. If e is a MATLANG expression over $\mathcal{S}$, then $(e^T)(\mathcal{I}) := \big(e(\mathcal{I})\big)^T$.

One-vector. If e is a MATLANG expression over $\mathcal{S}$, then $\big(\mathbf{1}(e)\big)(\mathcal{I}) := \mathbf{1}(e(\mathcal{I}))$.

Diagonalization. If e is a MATLANG expression over $\mathcal{S}$ with $\mathcal{S}(e) = \alpha \times 1$, then $\big(\mathsf{diag}(e)\big)(\mathcal{I}) := \mathsf{diag}(e(\mathcal{I}))$.

Multiplication If e_1 and e_2 are MATLANG expressions over $\mathcal{S}$ with $\mathcal{S}(e_1) = \alpha \times \beta$ and $\mathcal{S}(e_2) = \beta \times \gamma$, then $(e_1 \cdot e_2)(\mathcal{I}) := e_1(\mathcal{I}) \cdot e_2(\mathcal{I})$.

Addition If e_1 and e_2 are MATLANG expressions over $\mathcal{S}$ with $\mathcal{S}(e_1) = \mathcal{S}(e_2)$, then $(e_1 + e_2)(\mathcal{I}) := e_1(\mathcal{I}) + e_2(\mathcal{I})$.

Hadamard product If e_1 and e_2 are MATLANG expressions over $\mathcal{S}$ with $\mathcal{S}(e_1) = \mathcal{S}(e_2)$, then $(e_1 \circ e_2)(\mathcal{I}) := e_1(\mathcal{I}) \circ e_2(\mathcal{I})$.

Remark 6 Matrix addition and the Hadamard product are the pointwise applications of addition and product, respectively. The original definition of MATLANG [4] is more generally defined in terms of an arbitrary set Ω of allowed pointwise functions. So, MATLANG as defined above fixes Ω to $\{+, *\}$. This restriction was also considered by Geerts [6] (who also allows multiplication by constant scalars, but this is not essential).

Also, the original definition of MATLANG fixes K to the field of complex numbers and complex conjugate transpose is considered instead of transpose. By definition, the complex conjugate transpose of a matrix is obtained by pointwise application of complex conjugate to the transpose of that matrix. Hence, transpose can be expressed as pointwise application of complex conjugate to the complex conjugate transpose of a matrix. It follows that transpose and conjugate complex transpose are exchangable provided the set Ω of allowed pointwise functions contains complex conjugate.

In Sections 4.2 and 4.3, we provide simulations of MATLANG in $(\mathsf{ARA} + \zeta_2)(2)$ and of $(\mathsf{ARA} + \zeta_2)(2)$ in MATLANG.

4.2 Simulating MATLANG in $(\mathsf{ARA} + \zeta_2)(2)$

The notations used in this translation are summarized in Table 1 for easy reference. Examples 6 and 7, together with Figures 3 and 5, may also help to understand the translation.

For notational convenience, instead of fixing a one-to-one correspondence between **rel** and **matvar**, we assume that $\mathbf{rel} = \mathbf{matvar}$.

Let us now fix injective functions $\text{row}\colon \mathbf{size} \setminus \{1\} \to \mathbf{att}$ and $\text{col}\colon \mathbf{size} \setminus \{1\} \to \mathbf{att}$ with disjoint ranges such that, for all $\alpha \in \mathbf{size} \setminus \{1\}$, $\text{row}(\alpha)$ and $\text{col}(\alpha)$ are compatible. To reduce clutter, we write, for $\alpha \in \mathbf{size} \setminus \{1\}$, $\text{row}(\alpha)$ as row_α and $\text{col}(\alpha)$ as col_α.

Let $s \in \mathbf{size} \times \mathbf{size}$, $s = \alpha \times \beta$. We associate to s a relation schema

$$\mathit{\Gamma}(s) := \begin{cases} \{\text{row}_\alpha, \text{col}_\beta\} & \text{if } \alpha \neq 1 \neq \beta; \\ \{\text{row}_\alpha\} & \text{if } \alpha \neq 1 = \beta; \\ \{\text{col}_\beta\} & \text{if } \alpha = 1 \neq \beta; \\ \emptyset & \text{if } \alpha = 1 = \beta. \end{cases}$$

Observe that always $|\mathit{\Gamma}(s)| \leq 2$.

Let $\mathcal{S}$ be a matrix schema on a set of matrix variables V. We associate to $\mathcal{S}$ a database schema $\mathit{\Gamma}(\mathcal{S})$ on V as follows. For $M \in V$, we set $\big(\mathit{\Gamma}(\mathcal{S})\big)(M) := \mathit{\Gamma}(\mathcal{S}(M))$. Notice that, in the left-hand side, we interpret M as a relation name, whereas, in the right-hand side, we interpret M as a matrix variable. Of course, $\mathit{\Gamma}(S)$ is extended to ARA expressions in the usual way.

Let σ be a size assignment. We associate to σ a domain assignment $D(\sigma)$ where, for $\alpha \in \mathbf{size}$, $\big(D(\sigma)\big)(\text{row}_\alpha) = \big(D(\sigma)\big)(\text{col}_\alpha) := \{1, \ldots, \sigma(\alpha)\}$.

Let $M \in \mathcal{M}$ conform to s by σ. We associate to matrix M a relation $\text{Rel}_{s,\sigma}(M)\colon \mathcal{T}_{D(\sigma)}(\mathit{\Gamma}(s)) \to K$ as follows. For t in $\mathcal{T}_{D(\sigma)}(\mathit{\Gamma}(s))$, we have

$$\big(\text{Rel}_{s,\sigma}(M)\big)(t) := \begin{cases} M_{t(\text{row}_\alpha),t(\text{col}_\beta)} & \text{if } \alpha \neq 1 \neq \beta; \\ M_{t(\text{row}_\alpha),1} & \text{if } \alpha \neq 1 = \beta; \\ M_{1,t(\text{col}_\beta)} & \text{if } \alpha = 1 \neq \beta; \\ M_{1,1} & \text{if } \alpha = 1 = \beta. \end{cases}$$

Remark 7 We wish to make a short remark on 1×1 matrices over K, which by definition contain a single element of K. By the above, the schema of the relation representing a 1×1 matrix is the empty set, which has arity 0. The only tuple over this relation schema is the so-called empty tuple. To see that this tuple exists, recall that a tuple is function mapping the attributes of the relation schema to domain values. Mathematically, a function is a set of pairs, in this case pairs of attributes and domain values, exactly one for each attribute. Hence, if the relation schema is empty, the empty set technically satisfies the definition of tuple over that relation schema, and it is this tuple that we refer to as the empty tuple. If M is the 1×1 matrix with entry $a \in K$, then $\text{Rel}_{s,\sigma}(M)$ above maps the empty tuple to $a \in K$. Figure 4 visualizes the 1×1 matrix M and the 0-ary relation $\text{Rel}_{s,\sigma}(M)$. So, also for this corner case, the development above is sound.

Let $\mathcal{S}\colon V \to \mathbf{size} \times \mathbf{size}$ be a matrix schema, and let $\mathcal{I}$ be a matrix instance of $\mathcal{S}$ with respect to σ. We associate to $\mathcal{I}$ an instance $\text{Rel}_{\mathcal{S},\sigma}(\mathcal{I})$ of database schema $\mathit{\Gamma}(\mathcal{S})$ with respect to $D(\sigma)$ as follows. For $M \in V$, we set $\big(\text{Rel}_{\mathcal{S},\sigma}(\mathcal{I})\big)(M) := \text{Rel}_{\mathcal{S}(M),\sigma}(\mathcal{I}(M))$.

$M = (a)$ $\mathrm{Rel}_{s,\sigma}(M) =$

K
a

Fig. 4 The 1×1 matrix considered in Remark 7 and its representation as a 0-ary relation.

$\mathcal{I}(\text{no\_courses}) =$

$\text{row}_{\text{student}}$	col_{dptm}	K
1	1	5
1	2	2
1	3	0
2	1	2
2	2	1
2	3	3

$\mathcal{I}(\text{course\_fee}) =$

row_{dptm}	K
1	300
2	250
3	330

Fig. 5 Matrix instance from Figure 3 represented as a database instance. The tuple entries are the row and column indices of the corresponding matrices.

Example 6 Consider again matrix schema $\mathcal{S}$, size assignment σ, and matrix instance $\mathcal{I}$ of $\mathcal{S}$ with respect to σ from Example 5 and Figure 3. We have that $\big(\Gamma(\mathcal{S})\big)(\text{no\_courses}) = \{\text{row}_{\text{student}}, \text{col}_{\text{dptm}}\}$ and $\big(\Gamma(\mathcal{S})\big)(\text{course\_fee}) = \{\text{row}_{\text{dptm}}\}$. The database instance $\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I})$ is shown in Figure 5.

We now show that every MATLANG expression can be simulated by an $(\mathsf{ARA} + \zeta_2)(2)$ expression.

Lemma 3 *For every* MATLANG *expression e over a matrix schema $\mathcal{S}$, there exists an* $(\mathsf{ARA} + \zeta_2)(2)$ *expression $\Upsilon(e)$ over database schema $\Gamma(\mathcal{S})$ such that*

1. $\big(\Gamma(\mathcal{S})\big)(\Upsilon(e)) = \Gamma(\mathcal{S}(e))$*; and*
2. *for all size assignments σ and matrix instances $\mathcal{I}$ of $\mathcal{S}$ with respect to σ,* $\big(\Upsilon(e)\big)(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I})) = \mathrm{Rel}_{\mathcal{S}(e),\sigma}(e(\mathcal{I}))$.

Proof We construct the translation recursively on the structure of the MATLANG expression. The basis of the inductive proof that the translation satisfies the desired properties is in the translation of matrix variables. The inductive steps are a straightforward albeit sometimes tedious application of definitions and the induction hypothesis. For most operations, we therefore only provide some intuition. Only for two more elaborate cases, diagonalization and matrix product, we provide full proofs in the most general subcase.

Variable. If M is a matrix variable, then

$$\Upsilon(M) := M.$$

Both properties are trivially satisfied, as left- and right-hand side of the first property both reduce to $\Gamma(\mathcal{S}(M))$ and left- and right-hand side of the second property both reduce to $\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I}(M))$.

Transpose. If e is a MATLANG expression over $\mathcal{S}$ with $\mathcal{S}(e) = \alpha \times \beta$, then

$$\Upsilon(e^T) := \begin{cases} \rho_{\text{row}_\alpha \to \text{col}_\alpha, \text{col}_\beta \to \text{row}_\beta}(\Upsilon(e)) & \text{if } \alpha \neq 1 \neq \beta; \\ \rho_{\text{row}_\alpha \to \text{col}_\alpha}(\Upsilon(e)) & \text{if } \alpha \neq 1 = \beta; \\ \rho_{\text{col}_\beta \to \text{row}_\beta}(\Upsilon(e)) & \text{if } \alpha = 1 \neq \beta; \\ \Upsilon(e) & \text{if } \alpha = 1 = \beta. \end{cases}$$

Transposing a matrix involves swapping rows and columns, and, hence, in the translation, row attributes must become column attributes and column attributes must become row attributes. This is most obvious in the first of the four cases above, where $\alpha \neq 1 \neq \beta$. The three other cases are specializations of the first case for column vectors, row vectors, and 1×1 matrices, respectively.

One-vector. If e is a $\mathsf{MATLANG}$ expression over $\mathcal{S}$ with $\mathcal{S}(e) = \alpha \times \beta$, then

$$\varUpsilon(\mathbf{1}(e)) := \begin{cases} \mathbf{1}(\pi_{\{\mathrm{row}_\alpha\}}(\varUpsilon(e))) & \text{if } \alpha \neq 1; \\ \mathbf{1}(\pi_\emptyset(\varUpsilon(e))) & \text{if } \alpha = 1. \end{cases}$$

Applying the one-vector operation to a matrix can be simulated by applying the one operation to an appropriate projection of the relational representation of that matrix.

Diagonalization. If e is a $\mathsf{MATLANG}$ expression over $\mathcal{S}$ with $\mathcal{S}(e) = \alpha \times 1$, then

$$\varUpsilon(\mathsf{diag}(e)) := \begin{cases} \sigma_{\{\mathrm{row}_\alpha, \mathrm{col}_\alpha\}}(\varUpsilon(e) \bowtie \mathbf{1}(\rho_{\mathrm{row}_\alpha \to \mathrm{col}_\alpha}(\varUpsilon(e)))) & \text{if } \alpha \neq 1; \\ \varUpsilon(e) & \text{if } \alpha = 1. \end{cases}$$

The latter case reflects that the diagonalization of a 1×1 matrix is that matrix itself.

For diagonalization, we formally prove that both properties of this Lemma are satisfied in the case $\alpha \neq 1 \neq \beta$ as the case $\alpha = 1$ is straightforward, using the above observation. As induction hypothesis, we assume that $\big(\varGamma(\mathcal{S})\big)(\varUpsilon(e)) = \varGamma(\mathcal{S}(e))$, which equals $\{\mathrm{row}_\alpha\}$, and that, for all size assignments σ and matrix instances $\mathcal{I}$ of $\mathcal{S}$ with respect to σ, $\big(\varUpsilon(e)\big)(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I})) = \mathrm{Rel}_{\mathcal{S}(e),\sigma}(e(\mathcal{I}))$.

As for the first property, we have, on the one hand, $\big(\varGamma(\mathcal{S})\big)(\varUpsilon(\mathsf{diag}(e))) = \big(\varGamma(\mathcal{S})\big)(\sigma_{\{\mathrm{row}_\alpha, \mathrm{col}_\alpha\}}(\varUpsilon(e) \bowtie \mathbf{1}(\rho_{\mathrm{row}_\alpha \to \mathrm{col}_\alpha}(\varUpsilon(e))))) = \{\mathrm{row}_\alpha, \mathrm{col}_\alpha\}$, by the induction hypothesis. On the other hand, $\varGamma(\mathcal{S}(\mathsf{diag}(e))) = \{\mathrm{row}_\alpha, \mathrm{col}_\alpha\}$, since $\mathcal{S}(\mathsf{diag}(e)) = \alpha \times \alpha$. Hence, left- an right-hand side in the first property are equal.

As for the second property, let t be a tuple in $\mathcal{T}_{D(\sigma)}(\{\mathrm{row}_\alpha, \mathrm{col}_\alpha\})$.

On the one hand,

$$\begin{aligned} &\Big(\big(\varUpsilon(\mathsf{diag}(e))\big)(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I}))\Big)(t) \\ &\quad = \Big(\big(\sigma_{\{\mathrm{row}_\alpha, \mathrm{col}_\alpha\}}(\varUpsilon(e) \bowtie \mathbf{1}(\rho_{\mathrm{row}_\alpha \to \mathrm{col}_\alpha}(\varUpsilon(e))))\big)(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I}))\Big)(t) \\ &\quad = \Big(\sigma_{\{\mathrm{row}_\alpha, \mathrm{col}_\alpha\}}\big(\big(\varUpsilon(e) \bowtie \mathbf{1}(\rho_{\mathrm{row}_\alpha \to \mathrm{col}_\alpha}(\varUpsilon(e)))\big)(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I}))\big)\Big)(t). \end{aligned}$$

We see that, if $t(\mathrm{row}_\alpha) \neq t(\mathrm{col}_\alpha)$, then

$$\Big(\big(\varUpsilon(\mathsf{diag}(e))\big)(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I}))\Big)(t) = 0.$$

Let us therefore assume in the remainder of the calculation that $t(\mathrm{row}_\alpha) = t(\mathrm{col}_\alpha)$. Then, using the induction hypothesis in the fourth equality,

$$\begin{aligned}
&\Big(\sigma_{\{\mathrm{row}_\alpha,\mathrm{col}_\alpha\}}\big((\varUpsilon(e) \bowtie \mathbf{1}(\rho_{\mathrm{row}_\alpha\to\mathrm{col}_\alpha}(\varUpsilon(e))))(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I}))\big)\Big)(t)\\
&\quad= \Big(\big(\varUpsilon(e) \bowtie \mathbf{1}(\rho_{\mathrm{row}_\alpha\to\mathrm{col}_\alpha}(\varUpsilon(e)))\big)(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I}))\Big)(t)\\
&\quad= \Big(\big(\varUpsilon(e)\big)(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I}))\Big)(t|_{\{\mathrm{row}_\alpha\}}) *\\
&\qquad\qquad\qquad \Big(\big(\mathbf{1}(\rho_{\mathrm{row}_\alpha\to\mathrm{col}_\alpha}(\varUpsilon(e)))\big)(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I}))\Big)(t|_{\{\mathrm{col}_\alpha\}})\\
&\quad= \Big(\big(\varUpsilon(e)\big)(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I}))\Big)(t|_{\{\mathrm{row}_\alpha\}}) *\\
&\qquad\qquad\qquad \Big(\mathbf{1}\big((\rho_{\mathrm{row}_\alpha\to\mathrm{col}_\alpha}(\varUpsilon(e)))(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I}))\big)\Big)(t|_{\{\mathrm{col}_\alpha\}})\\
&\quad= \big(\mathrm{Rel}_{\mathcal{S}(e),\sigma}(e(\mathcal{I}))\big)(t|_{\{\mathrm{row}_\alpha\}}) * \mathbf{1}_{\{\mathrm{col}_\alpha\}}(t|_{\{\mathrm{col}_\alpha\}})\\
&\quad= \big(e(\mathcal{I})\big)_{t|_{\{\mathrm{row}_\alpha\}}(\mathrm{row}_\alpha)} * 1\\
&\quad= \big(e(\mathcal{I})\big)_{t(\mathrm{row}_\alpha)}.
\end{aligned}$$

On the other hand,

$$\Big(\mathrm{Rel}_{\mathcal{S}(e),\sigma}\big((\mathsf{diag}(e))(\mathcal{I})\big)\Big)(t) = \Big((\mathsf{diag}(e))(\mathcal{I})\Big)_{t(\mathrm{row}_\alpha),t(\mathrm{col}_\alpha)}.$$

If $t(\mathrm{row}_\alpha) \neq t(\mathrm{col}_\alpha)$, then the latter expression equals 0; if $t(\mathrm{row}_\alpha) = t(\mathrm{col}_\alpha)$, then it equals $\big(e(\mathcal{I})\big)_{t(\mathrm{row}_\alpha)}$. Hence, left- an right-hand side in the second property are also equal.

Multiplication If e_1 and e_2 are MATLANG expressions over $\mathcal{S}$ with $\mathcal{S}(e) = \alpha\times\beta$ and $\mathcal{S}(e) = \beta\times\gamma$, then

$$\varUpsilon(e_1\cdot e_2) = \begin{cases}\zeta_{C,2}(\rho_{\varphi_1}(\varUpsilon(e_1)), \rho_{\varphi_2}(\varUpsilon(e_2))) & \text{if } \beta \neq 1;\\ \varUpsilon(e_1) \bowtie \varUpsilon(e_2) & \text{if } \beta = 1,\end{cases}$$

where C is an attribute different from both row_α and col_γ, $\varphi_1(\mathrm{col}_\beta) = \varphi_2(\mathrm{row}_\beta) = C$, and φ_1 and φ_2 are the identity elsewhere.

Also for matrix multiplication, we formally prove that both properties of this Lemma are satisfied in the case where $\beta \neq 1$ and $\alpha \neq 1 \neq \gamma$. The other cases are then straightforward. As induction hypothesis, we assume that $\big(\varGamma(\mathcal{S})\big)(\varUpsilon(e_1)) = \varGamma(\mathcal{S}(e_1))$, which equals $\{\mathrm{row}_\alpha, \mathrm{col}_\beta\}$, $\big(\varGamma(\mathcal{S})\big)(\varUpsilon(e_2)) = \varGamma(\mathcal{S}(e_2))$, which equals $\{\mathrm{row}_\beta, \mathrm{col}_\gamma\}$, and that, for all size assignments σ and matrix instances $\mathcal{I}$ of $\mathcal{S}$ with respect to σ and for $i = 1, 2$, $\big(\varUpsilon(e_i)\big)(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I})) = \mathrm{Rel}_{\mathcal{S}(e_i),\sigma}(e_i(\mathcal{I}))$.

As for the first property, we have, on the one hand, $\big(\varGamma(\mathcal{S})\big)(\varUpsilon(e_1\cdot e_2)) = \big(\varGamma(\mathcal{S})\big)(\zeta_{C,2}(\rho_{\varphi_1}(\varUpsilon(e_1)), \rho_{\varphi_2}(\varUpsilon(e_2)))) = \{\mathrm{row}_\alpha, \mathrm{col}_\gamma\}$, by the induction hypothesis. On the other hand, $\varGamma(\mathcal{S}(e_1\cdot e_2)) = \{\mathrm{row}_\alpha, \mathrm{col}_\gamma\}$, since $\mathcal{S}(e_1\cdot e_2) = \alpha\times\gamma$. Hence, left- an right-hand side in the first property are equal.

As for the second property, let t be a tuple in $\mathcal{T}_{D(\sigma)}(\{\mathrm{row}_\alpha, \mathrm{col}_\gamma\})$.

On the one hand, using the induction hypothesis in the ninth equality,

$$
\begin{aligned}
&\Big(\big(\Upsilon(e_1\cdot e_2)\big)(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I}))\Big)(t)\\
&\quad= \Big(\big(\zeta_{C,2}(\rho_{\varphi_1}(\Upsilon(e_1)),\rho_{\varphi_2}(\Upsilon(e_2)))\big)(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I}))\Big)(t)\\
&\quad= \Big(\big(\pi_{\{\mathrm{row}_\alpha,\mathrm{col}_\beta\}}(\rho_{\varphi_1}(\Upsilon(e_1))\bowtie\rho_{\varphi_2}(\Upsilon(e_2)))\big)(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I}))\Big)(t)\\
&\quad= \Big(\pi_{\{\mathrm{row}_\alpha,\mathrm{col}_\beta\}}\big((\rho_{\varphi_1}(\Upsilon(e_1))\bowtie\rho_{\varphi_2}(\Upsilon(e_2)))(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I}))\big)\Big)(t)\\
&\quad= \sum_{t'\in\mathcal{T}_{D(\sigma)}(\{\mathrm{row}_\alpha,C,\mathrm{col}_\gamma\}),\ t'|_{\{\mathrm{row}_\alpha,\mathrm{col}_\gamma\}}=t}\Big(\big(\rho_{\varphi_1}(\Upsilon(e_1))\bowtie\rho_{\varphi_2}(\Upsilon(e_2))\big)(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I}))\Big)(t')\\
&\quad= \sum_{t'\in\mathcal{T}_{D(\sigma)}(\{\mathrm{row}_\alpha,C,\mathrm{col}_\gamma\}),\ t'|_{\{\mathrm{row}_\alpha,\mathrm{col}_\gamma\}}=t}\Big(\big(\rho_{\varphi_1}(\Upsilon(e_1))\big)(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I}))\bowtie\big(\rho_{\varphi_2}(\Upsilon(e_2))\big)(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I}))\Big)(t')\\
&\quad= \sum_{t'\in\mathcal{T}_{D(\sigma)}(\{\mathrm{row}_\alpha,C,\mathrm{col}_\gamma\}),\ t'|_{\{\mathrm{row}_\alpha,\mathrm{col}_\gamma\}}=t}\Big(\rho_{\varphi_1}\big((\Upsilon(e_1))(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I}))\big)\bowtie\rho_{\varphi_2}\big((\Upsilon(e_2))(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I}))\big)\Big)(t')\\
&\quad= \sum_{t'\in\mathcal{T}_{D(\sigma)}(\{\mathrm{row}_\alpha,C,\mathrm{col}_\gamma\}),\ t'|_{\{\mathrm{row}_\alpha,\mathrm{col}_\gamma\}}=t}\Big(\rho_{\varphi_1}\big((\Upsilon(e_1))(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I}))\big)\Big)(t'|_{\{\mathrm{row}_\alpha,C\}})\,*\\
&\qquad\qquad\qquad\Big(\rho_{\varphi_2}\big((\Upsilon(e_2))(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I}))\big)\Big)(t'|_{\{C,\mathrm{col}_\gamma\}})\\
&\quad= \sum_{t'\in\mathcal{T}_{D(\sigma)}(\{\mathrm{row}_\alpha,C,\mathrm{col}_\gamma\}),\ t'|_{\{\mathrm{row}_\alpha,\mathrm{col}_\gamma\}}=t}\Big(\big(\Upsilon(e_1)\big)(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I}))\Big)(t'|_{\{\mathrm{row}_\alpha,C\}}\circ\varphi_1)\,*\\
&\qquad\qquad\qquad\Big(\big(\Upsilon(e_2)\big)(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I}))\Big)(t'|_{\{C,\mathrm{col}_\gamma\}}\circ\varphi_2)\\
&\quad= \sum_{t'\in\mathcal{T}_{D(\sigma)}(\{\mathrm{row}_\alpha,C,\mathrm{col}_\gamma\}),\ t'|_{\{\mathrm{row}_\alpha,\mathrm{col}_\gamma\}}=t}\big(\mathrm{Rel}_{\mathcal{S}(e_1),\sigma}(e_i(\mathcal{I}))\big)(t'|_{\{\mathrm{row}_\alpha,C\}}\circ\varphi_1)\,*\\
&\qquad\qquad\qquad\big(\mathrm{Rel}_{\mathcal{S}(e_2),\sigma}(e_i(\mathcal{I}))\big)(t'|_{\{C,\mathrm{col}_\gamma\}}\circ\varphi_2)\\
&\quad= \sum_{t'\in\mathcal{T}_{D(\sigma)}(\{\mathrm{row}_\alpha,C,\mathrm{col}_\gamma\}),\ t'|_{\{\mathrm{row}_\alpha,\mathrm{col}_\gamma\}}=t}\big(e_1(\mathcal{I})\big)_{t'(\varphi_1(\mathrm{row}_\alpha)),t'(\varphi_1(\mathrm{col}_\beta))}*\big(e_2(\mathcal{I})\big)_{t'(\varphi_2(\mathrm{row}_\beta)),t'(\varphi_2(\mathrm{col}_\gamma))}\\
&\quad= \sum_{t'\in\mathcal{T}_{D(\sigma)}(\{\mathrm{row}_\alpha,C,\mathrm{col}_\gamma\}),\ t'|_{\{\mathrm{row}_\alpha,\mathrm{col}_\gamma\}}=t}\big(e_1(\mathcal{I})\big)_{t'(\mathrm{row}_\alpha),t'(C)}*\big(e_2(\mathcal{I})\big)_{t'(C),t'(\mathrm{col}_\gamma)}\\
&\quad= \sum_{k=1}^{\sigma(\beta)}\big(e_1(\mathcal{I})\big)_{t(\mathrm{row}_\alpha),k}*\big(e_2(\mathcal{I})\big)_{k,t(\mathrm{col}_\gamma)}\,.
\end{aligned}
$$

On the other hand,

$$
\begin{aligned}
\big(\mathrm{Rel}_{\mathcal{S}(e),\sigma}((e_1\cdot e_2)(\mathcal{I}))\big)(t) &= \big((e_1\cdot e_2)(\mathcal{I})\big)_{t(\mathrm{row}_\alpha),t(\mathrm{col}_\gamma)}\\
&= \big(e_1(\mathcal{I})\cdot e_2(\mathcal{I})\big)_{t(\mathrm{row}_\alpha),t(\mathrm{col}_\gamma)}\\
&= \sum_{k=1}^{\sigma(\beta)}\big(e_1(\mathcal{I})\big)_{t(\mathrm{row}_\alpha),k}*\big(e_2(\mathcal{I})\big)_{k,t(\mathrm{col}_\gamma)}\,.
\end{aligned}
$$

Hence, left- an right-hand side in the second property are also equal.

Addition If e_1 and e_2 are $\mathsf{MATLANG}$ expressions over $\mathcal{S}$ with $\mathcal{S}(e_1) = \mathcal{S}(e_2)$, then

$$\Upsilon(e_1 + e_2) := \Upsilon(e_1) \cup \Upsilon(e_2).$$

Since union involves adding values of corresponding tuples, this operation provides the appropriate translation for matrix addition.

Hadamard product If e_1 and e_2 are $\mathsf{MATLANG}$ expressions over $\mathcal{S}$ with $\mathcal{S}(e_1) = \mathcal{S}(e_2)$, then

$$\Upsilon(e_1 \circ e_2) := \Upsilon(e_1) \bowtie \Upsilon(e_2).$$

Since joining relations with the same scheme involves multiplying values of corresponding tuples, this operation provides the appropriate translation for the Hadamard product.

□

Example 7 Consider again matrix schema $\mathcal{S}$, size assignment σ, and matrix instance $\mathcal{I}$ of $\mathcal{S}$ with respect to σ from Example 5 and Figure 3. Consider the $\mathsf{MATLANG}$ expression $e = \text{no\_courses} \cdot \text{course\_fee}$ over $\mathcal{S}$. We have $\mathcal{S}(e) = \text{student} \times 1$ and

$$e(\mathcal{I}) = \begin{pmatrix} 2000 \\ 1840 \end{pmatrix} \qquad \text{and} \qquad \text{Rel}_{\mathcal{S}(e),\sigma}(e(\mathcal{I})) =$$

$\text{row}_{\text{student}}$	K
1	2000
2	1840

.

By Lemma 3 and its proof, we have that $\text{Rel}_{\mathcal{S}(e),\sigma}(e(\mathcal{I}))$ equals $e'(\text{Rel}_{\mathcal{S},\sigma}(\mathcal{I}))$ with

$$e' = \zeta_{C,2}(\rho_{\varphi_1}(\text{no\_courses}), \rho_{\varphi_2}(\text{course\_fee})),$$

where $\varphi_1(\text{col}_\gamma) = \varphi_2(\text{row}_\gamma) = C \notin \{\text{row}_\alpha, \text{col}_\beta\}$ and φ_1 and φ_2 are the identity elsewhere.

4.3 Simulating $(\mathsf{ARA} + \zeta_2)(2)$ in $\mathsf{MATLANG}$

The notations used in this translation are summarized in Table 2 for easy reference. Examples 8 and 9, together with Figures 1 and 3, may also help to understand the translation.

In order to simulate $(\mathsf{ARA} + \zeta_2)(2)$ in $\mathsf{MATLANG}$, we equip **att** with some linear ordering $<$. Note that $<$ is an ordering on attributes, not on domain elements. Only an ordering on domain elements can have an impact on the expressive power of query languages [1].

Again, we assume that **rel** = **matvar**. Let us fix an injective function $\Psi\colon \textbf{att} \to \textbf{size} \setminus \{1\}$.

Let X be a relation schema with $|X| \leq 2$. We associate to X an element $\Theta(X) \in \textbf{size} \times \textbf{size}$ as follows:

$$\Theta(X) := \begin{cases} \Psi(A_1) \times \Psi(A_2) & \text{if } X = \{A_1, A_2\} \text{ and } A_1 < A_2; \\ \Psi(A) \times 1 & \text{if } X = \{A\}; \\ 1 \times 1 & \text{if } X = \emptyset. \end{cases}$$

Mapping	$(\mathsf{ARA} + \zeta_2)(2)$	$\mathsf{MATLANG}$
relation name $\to$ matrix variable	R	R
attribute $\to$ size symbol	A	$\Phi(A)$
relation schema $\to$ element of **size** $\times$ **size**	X	$\Theta(X)$
database schema $\to$ matrix schema	$\mathcal{S}$	$\Theta(\mathcal{S})$
domain assignment $\to$ size assignment	D	$\sigma(D)$
relation $\to$ matrix	r	$\mathrm{Mat}_D(r)$
database instance $\to$ matrix instance	$\mathcal{I}$	$\mathrm{Mat}_D(\mathcal{I})$
$(\mathsf{ARA} + \zeta_2)(2)$ expression $\to$ $\mathsf{MATLANG}$ expression	e	$\Phi(e)$

Table 2 Symbol table for the simulation of $(\mathsf{ARA} + \zeta_2)(2)$ in $\mathsf{MATLANG}$.

Let $\mathcal{S}$ be a database schema on a set N of relation names with arity at most 2. We associate to $\mathcal{S}$ a matrix schema $\Theta(\mathcal{S})$ on N as follows. For $R \in N$, we set $\big(\Theta(\mathcal{S})\big)(R) := \Theta(\mathcal{S}(R))$.

Let D be a domain assignment. We associate to D a size assignment $\sigma(D)$ where, for $A \in \mathbf{att}$, $\big(\sigma(D)\big)(\Psi(A)) = |D(A)|$. If every domain in the range of a domain assignment D is of the form $\{1, \dots, n\}$ for some integer $n \geq 1$, then we say that D is *consecutive*.

Let D be a consecutive domain assignment. Given a relation $r \colon \mathcal{T}_D(X) \to K$ with $|X| \leq 2$, we associate a matrix $\mathrm{Mat}_D(r)$ conforming to $\Theta(X)$ by $\sigma(D)$. We distinguish three cases:

1. If $X = \{A_1, A_2\}$ with $A_1 < A_2$, then $\mathrm{Mat}_D(r)$ is a $|D(A_1)| \times |D(A_2)|$ matrix. For $i = 1, \dots, |D(A_1)|$ and $j = 1, \dots, |D(A_2)|$, $\big(\mathrm{Mat}_D(r)\big)_{i,j} = r(t)$, where $t \in \mathcal{T}_D(X)$ is defined by $t(A_1) := i$ and $t(A_2) := j$.
2. If $X = \{A\}$, then $\mathrm{Mat}_D(r)$ is a $|D(A)| \times 1$ matrix (i.e., a column vector). For $i = 1, \dots, |D(A)|$, $\big(\mathrm{Mat}_D(r)\big)_{i,1} = r(t)$, where $t \in \mathcal{T}_D(X)$ is defined by $t(A) := i$.
3. If $X = \emptyset$, then $\mathrm{Mat}_D(r)$ is a 1×1 matrix. We have that $\big(\mathrm{Mat}_D(r)\big)_{1,1} = r(t)$, where t is the empty tuple (see Remark 7).

Let $\mathcal{S} \colon N \to \mathcal{P}_{\mathrm{fin}}(\mathbf{att})$ be a database schema such that all relation names in N have arity at most 2, and let $\mathcal{I}$ be a database instance of $\mathcal{S}$ with respect to D. We associate to $\mathcal{I}$ a matrix instance $\mathrm{Mat}_D(\mathcal{I})$ of $\mathrm{Mat}(\mathcal{S})$ with respect to $\sigma(D)$ as follows. For $R \in N$, we set $\big(\mathrm{Mat}_D(\mathcal{I})\big)(R) := \mathrm{Mat}_D(\mathcal{I}(R))$.

Example 8 Consider again database schema $\mathcal{S}$, domain assignment D, and database instance $\mathcal{I}$ of $\mathcal{S}$ with respect to D from Examples 1 and 2 and Figure 1. To reduce clutter, assume that $\mathbf{att} = \mathbf{size} \setminus \{1\}$ and that Ψ is the identity function. Take student $<$ dptm. We have that $\big(\Theta(\mathcal{S})\big)(\text{no\_courses}) = \text{student} \times \text{dptm}$ and $(\Theta(\mathcal{S}))(\text{course\_fee}) = \text{dptm} \times 1$. Consider domain assignment D' and database instance $\mathcal{I}'$ obtained from D and $\mathcal{I}$, respectively, by replacing Alice by 1, Bob by 2, CS by 1, Math by 2, and Bio by 3. Note that D' is consecutive. The instance $\mathrm{Mat}_{D'}(\mathcal{I}')$ is shown in Figure 3.

We now show that every $(\mathsf{ARA} + \zeta_2)(2)$ expression can be simulated by a $\mathsf{MATLANG}$ expression.

Lemma 4 *For every* $(\mathsf{ARA}+\zeta_2)(2)$ *expression* e *over a database schema* $\mathcal{S}$ *of arity at most* 2*, there exists a* $\mathsf{MATLANG}$ *expression* $\Phi(e)$ *over matrix schema* $\Theta(\mathcal{S})$ *such that*

1. $\big(\Theta(\mathcal{S})\big)(\Phi(e)) = \Theta(\mathcal{S}(e))$*; and*
2. *for all consecutive*[3] *domain assignments* D *and database instances* $\mathcal{I}$ *with respect to* D*,* $\big(\Phi(e)\big)(\mathrm{Mat}_D(\mathcal{I})) = \mathrm{Mat}_D(e(\mathcal{I}))$*.*

Proof We construct the translation recursively on the structure of the $(\mathsf{ARA}+\zeta_2)(2)$ expression. The basis of the inductive proof that the translation satisfies the desired properties is in the translation of relation names. The inductive steps are a straightforward albeit sometimes tedious application of definitions and the induction hypothesis. For most operations, we therefore only provide some intuition. Only for two more elaborate cases, projection and selection, we provide full proofs in the most general subcase.

Relation name. If M is a relation name with $|\mathcal{S}(M)| \leq 2$, then

$$\Phi(M) := M.$$

Both properties are trivially satisfied, as left- and right-hand side of the first property both reduce to $\Theta(\mathcal{S}(M))$ and left- and right-hand side of the second property both reduce to $\mathrm{Mat}_D(\mathcal{I}(M))$.

One. If e is an $(\mathsf{ARA}+\zeta_2)(2)$ expression over $\mathcal{S}$, then

$$\Phi(\mathbf{1}(e)) := \begin{cases} \mathbf{1}(\Phi(e)) \cdot \Big(\mathbf{1}\big(\big(\Phi(e)\big)^T\big)\Big)^T & \text{if } |\mathcal{S}(e)| = 2; \\ \mathbf{1}(\Phi(e)) & \text{if } |\mathcal{S}(e)| < 2. \end{cases}$$

Applying the one operation to a null-ary or unary relation can be simulated simply by applying the one-vector operation to the matrix representation of that relation. In case of a binary relation, we must also apply the one-vector operation to the transpose of that matrix, and then multiply the former with the transpose of the latter to obtain a matrix of the same dimensions as the original one.

Union. If e_1 and e_2 are $(\mathsf{ARA}+\zeta_2)(2)$ expressions over $\mathcal{S}$ with $\mathcal{S}(e_1) = \mathcal{S}(e_2)$, then

$$\Phi(e_1 \cup e_2) := \Phi(e_1) + \Phi(e_2).$$

Since matrix addition was translated in the union of their relational representations in Lemma 3, union of at most binary relations must be translated in the sum of their matrix representations.

[3] Recall that we required consecutive domain assignments to obtain a matrix representation of a relation. (In case of arbitrary domain assignments, the representation is still possible, of course, if we provide mappings of all domains into initial segments of the strictly positive integers.)

Projection It is convenient in this case to consider projecting out an attribute rather than standard projection. We already argued that these are interchangable. If e is an $(\mathsf{ARA} + \zeta_2)(2)$ expression over $\mathcal{S}$ and $A \in \mathcal{S}(e)$, then

$$\Phi(\hat{\pi}_A(e)) := \begin{cases} \Phi(e) \cdot \mathbf{1}\big(\big(\Phi(e)\big)^T\big) & \text{if } \mathcal{S}(e) = \{A_1, A_2\},\ A_1 < A = A_2; \\ \big(\Phi(e)\big)^T \cdot \mathbf{1}(\Phi(e)) & \text{otherwise.} \end{cases}$$

In essence, the two cases distinguish between projecting out the second attribute (if there is one) and projecting out the first attribute (or the only attribute).

For this operation, we formally prove that both properties of this Lemma are satisfied in the former case. As induction hypothesis, we assume that $\big(\Theta(\mathcal{S})\big)(\Phi(e)) = \Theta(\mathcal{S}(e))$, which equals $\Psi(A_1) \times \Psi(A_2)$, and that, for all consecutive domain assignments D and database instances $\mathcal{I}$ with respect to D, $\big(\Phi(e)\big)(\mathrm{Mat}_D(\mathcal{I})) = \mathrm{Mat}_D(e(\mathcal{I}))$.

As for the first property, we have, on the one hand, $\big(\Theta(\mathcal{S})\big)(\Phi(\hat{\pi}_{A_2}(e))) = \big(\Theta(\mathcal{S})\big)\big(\Phi(e) \cdot \mathbf{1}\big(\big(\Phi(e)\big)^T\big)\big) = |D(A_1)| \times 1$, by the induction hypothesis. On the other hand, $\Theta(\mathcal{S}(\hat{\pi}_{A_2}(e))) = |D(A_1)| \times 1$, since $\mathcal{S}(\hat{\pi}_{A_2}(e)) = \{A_1\}$. Hence, left- an right-hand side in the first property are equal.

As for the second property, let $i = 1, \ldots, |D(A_1)|$.

On the one hand, using the induction hypothesis in the fifth equality,

$$\begin{aligned}
&\Big(\big(\Phi(\hat{\pi}_{A_2}(e))\big)(\mathrm{Mat}_D(\mathcal{I}))\Big)_{i,1} \\
&= \Big(\Big(\Phi(e) \cdot \mathbf{1}\big(\big(\Phi(e)\big)^T\big)\Big)(\mathrm{Mat}_D(\mathcal{I}))\Big)_{i,1} \\
&= \Big(\big(\Phi(e)\big)(\mathrm{Mat}_D(\mathcal{I})) \cdot \Big(\mathbf{1}\big(\big(\Phi(e)\big)^T\big)\Big)(\mathrm{Mat}_D(\mathcal{I}))\Big)_{i,1} \\
&= \Big(\big(\Phi(e)\big)(\mathrm{Mat}_D(\mathcal{I})) \cdot \mathbf{1}\Big(\Big(\big(\Phi(e)\big)^T\Big)(\mathrm{Mat}_D(\mathcal{I}))\Big)\Big)_{i,1} \\
&= \Big(\big(\Phi(e)\big)(\mathrm{Mat}_D(\mathcal{I})) \cdot \mathbf{1}\Big(\Big(\big(\Phi(e)\big)(\mathrm{Mat}_D(\mathcal{I}))\Big)^T\Big)\Big)_{i,1} \\
&= \Big(\mathrm{Mat}_D(e(\mathcal{I})) \cdot \mathbf{1}\big(\big(\mathrm{Mat}_D(e(\mathcal{I}))\big)^T\big)\Big)_{i,1} \\
&= \sum_{j=1}^{|D(A_2)|} \big(\mathrm{Mat}_D(e(\mathcal{I}))\big)_{i,j} * \Big(\mathbf{1}\big(\big(\mathrm{Mat}_D(e(\mathcal{I}))\big)^T\big)\Big)_{j,1} \\
&= \sum_{j=1}^{|D(A_2)|} \big(e(\mathcal{I})\big)(t_{i,j}) * 1 \\
&= \sum_{j=1}^{|D(A_2)|} \big(e(\mathcal{I})\big)(t_{i,j}),
\end{aligned}$$

where $t_{i,j} \in \mathcal{T}_D(\{A_1, A_2\})$ is defined by $t(A_1) = i$ and $t(A_2) = j$.

On the other hand, for $t_i \in \mathcal{T}_D(\{A_1\})$ defined by $t(A_1) = i$,

$$\begin{aligned}\Big(\mathrm{Mat}_D\big((\hat{\pi}_{A_2}(e))(\mathcal{I})\big)\Big)_{i,1} &= \Big(\big(\hat{\pi}_{A_2}(e)\big)(\mathcal{I})\Big)(t_i)\\ &= \big(\hat{\pi}_{A_2}(e(\mathcal{I}))\big)(t_i)\\ &= \sum_{t\in\mathcal{T}_D(\{A_1,A_2\}),\ t|_{A_1}=t_i}\big(e(\mathcal{I})\big)(t)\\ &= \sum_{j=1}^{|D(A_2)|}\big(e(\mathcal{I})\big)(t_{i,j}),\end{aligned}$$

where $t_{i,j} \in \mathcal{T}_D(\{A_1, A_2\})$ is defined by $t(A_1) = i$ and $t(A_2) = j$. Hence, left- an right-hand side in the second property are also equal.

Selection. If e is an $(\mathsf{ARA} + \zeta_2)(2)$ expression over $\mathcal{S}$, $Y \subseteq \mathcal{S}(e)$, and the attributes of Y are mutually compatible, then

$$\Phi(\sigma_Y(e)) := \begin{cases}\Phi(e)\circ \mathsf{diag}(\mathbf{1}(\Phi(e))) & \text{if } |\mathcal{S}(e)| = |Y| = 2;\\ \Phi(e) & \text{otherwise.}\end{cases}$$

The latter case reflects that selection has no effect unless at least two attributes are involved in it. We formally prove that both properties of this Lemma are satisfied in the former case. Therefore, assume that $\mathcal{S}(e) = Y = \{A_1, A_2\}$. As induction hypothesis, we assume that $\big(\Theta(\mathcal{S})\big)(\Phi(e)) = \Theta(\mathcal{S}(e))$, which equals $|D(A_1)| \times |D(A_2)| = |D(A_1)| \times |D(A_1)|$, since A_1 and A_2 are mutually compatible. We also assume that, for all consecutive domain assignments D and database instances $\mathcal{I}$ with respect to D, $\big(\Phi(e)\big)(\mathrm{Mat}_D(\mathcal{I})) = \mathrm{Mat}_D(e(\mathcal{I}))$.
As for the first property, we have, on the one hand,

$$\big(\Theta(\mathcal{S})\big)(\Phi(\sigma_{\{A_1,A_2\}}(e))) = \big(\Theta(\mathcal{S})\big)(\Phi(e)\circ \mathsf{diag}(\mathbf{1}(\Phi(e)))),$$

which equals $|D(A_1)|\times|D(A_1)|$, by the induction hypothesis. On the other hand, $\Theta(\mathcal{S}(\sigma_{\{A_1,A_2\}}(e))) = |D(A_1)| \times |D(A_1)|$, since $\mathcal{S}(\sigma_{\{A_1,A_2\}}(e)) = \{A_1, A_2\}$, and A_1 and A_2 are compatible. Hence, left- an right-hand side in the first property are equal.
As for the second property, let $i, j = 1, \ldots, |D(A_1)| = |D(A_2)|$.
On the one hand, using the induction hypothesis in the last equality,

$$\begin{aligned}&\Big(\big(\Phi(\sigma_{\{A_1,A_2\}}(e))\big)(\mathrm{Mat}_D(\mathcal{I}))\Big)_{i,j}\\ &\quad= \Big(\big(\Phi(e)\circ \mathsf{diag}(\mathbf{1}(\Phi(e)))\big)(\mathrm{Mat}_D(\mathcal{I}))\Big)_{i,j}\\ &\quad= \Big(\big(\Phi(e)\big)(\mathrm{Mat}_D(\mathcal{I}))\circ\big(\mathsf{diag}(\mathbf{1}(\Phi(e)))\big)(\mathrm{Mat}_D(\mathcal{I}))\Big)_{i,j}\\ &\quad= \Big(\big(\Phi(e)\big)(\mathrm{Mat}_D(\mathcal{I}))\Big)_{i,j} * \Big(\big(\mathsf{diag}(\mathbf{1}(\Phi(e)))\big)(\mathrm{Mat}_D(\mathcal{I}))\Big)_{i,j}\\ &\quad= \big(\mathrm{Mat}_D(e(\mathcal{I}))\big)_{i,j} * ((\mathsf{diag}(\mathbf{1}(\Phi(e))))(\mathrm{Mat}_D(\mathcal{I})))_{i,j}.\end{aligned}$$

We now distinguish two cases. If $i \neq j$, then the second factor in the K-product above equals 0, and, hence,

$$\Big(\big(\Phi(\sigma_{\{A_1,A_2\}}(e))\big)(\mathrm{Mat}_D(\mathcal{I}))\Big)_{i,j} = 0.$$

If $i = j$, then

$$\begin{aligned}
&\big(\mathrm{Mat}_D(e(\mathcal{I}))\big)_{i,i} * \Big(\big(\mathsf{diag}(\mathbf{1}(\Phi(e)))\big)(\mathrm{Mat}_D(\mathcal{I}))\Big)_{i,i}\\
&\quad= \big(\mathrm{Mat}_D(e(\mathcal{I}))\big)_{i,i} * \Big(\big(\mathbf{1}(\Phi(e))\big)(\mathrm{Mat}_D(\mathcal{I}))\Big)_{i,1}\\
&\quad= \big(\mathrm{Mat}_D(e(\mathcal{I}))\big)_{i,i} * \Big(\mathbf{1}\big(\big(\Phi(e)\big)(\mathrm{Mat}_D(\mathcal{I}))\big)\Big)_{i,1}\\
&\quad= \big(\mathrm{Mat}_D(e(\mathcal{I}))\big)_{i,i} * 1\\
&\quad= \big(\mathrm{Mat}_D(e(\mathcal{I}))\big)_{i,i}\\
&\quad= \big(e(\mathcal{I})\big)(t_{i,i}),
\end{aligned}$$

where $t_{i,i} \in \mathcal{T}_D(\{A_1, A_2\})$ is defined by $t(A_1) = i = t(A_2)$.
On the other hand, for $t_{i,j} \in \mathcal{T}_D(\{A_1, A_2\})$ defined by $t(A_1) = i$ and $t(A_2) = j$,

$$\begin{aligned}
\Big(\mathrm{Mat}_D\big(\big(\sigma_{\{A_1,A_2\}}(e)\big)(\mathcal{I})\big)\Big)_{i,j} &= \Big(\big(\sigma_{\{A_1,A_2\}}(e)\big)(\mathcal{I})\Big)(t_{i,j})\\
&= \big(\sigma_{\{A_1,A_2\}}(e(\mathcal{I}))\big)(t_{i,j}),
\end{aligned}$$

where $t_{i,j} \in \mathcal{T}_D(\{A_1, A_2\})$ is defined by $t(A_1) = i$ and $t(A_2) = j$. We again distinguish two cases. If $i \neq j$, then $t(A_i) = i \neq j = t(A_j)$ and, consequently,

$$\Big(\mathrm{Mat}_D\big(\big(\sigma_{\{A_1,A_2\}}(e)\big)(\mathcal{I})\big)\Big)_{i,j} = 0.$$

If $i = j$, then $t(A_i) = t(A_j)$ and

$$\Big(\mathrm{Mat}_D\big(\big(\sigma_{\{A_1,A_2\}}(e)\big)(\mathcal{I})\big)\Big)_{i,i} = \big(e(\mathcal{I})\big)(t_{i,i}).$$

Hence, left- an right-hand side in the second property are also equal.

Renaming. If e is an $(\mathsf{ARA} + \zeta_2)(2)$ expression over $\mathcal{S}$ and $\varphi\colon \mathcal{S}(e) \to Y$ is a compatible one-to-one correspondence with $Y \subseteq \mathbf{att}$, then

$$\Phi(\rho_\varphi(e)) := \begin{cases} \big(\Phi(e)\big)^T & \text{if } S(e) = \{A_1, A_2\},\ A_1 < A_2,\ \text{and } \varphi(A_1) > \varphi(A_2);\\ \Phi(e) & \text{otherwise.}\end{cases}$$

Since attribute names are not reflected in the matrix representation of relations, renaming has no effect on the translation, unless the renaming reverses the order of the attributes, which results in swapping rows and columns, i.e., in transpose.

Join. If e_1 and e_2 are $(\mathsf{ARA} + \zeta_2)(2)$ expressions over $\mathcal{S}$, then $\Phi(e_1 \bowtie e_2)$ equals

$$\begin{cases} \Phi(e_1) \circ \Phi(e_2) & \text{if } \mathcal{S}(e_1) = \mathcal{S}(e_2); \\ s(\Phi(e_1), \Phi(e_2)) \circ \Phi(e_2) & \text{if } \mathcal{S}(e_1) = \emptyset; \\ \Phi(e_1) \circ s(\Phi(e_2), \Phi(e_1)) & \text{if } \mathcal{S}(e_2) = \emptyset; \\ \Phi(e_1) \cdot \big(\Phi(e_2)\big)^T & \text{if } \mathcal{S}(e_1) = \{A_1\} \text{ and } \mathcal{S}(e_2) = \{A_2\}; \\ \Big(\Phi(e_1) \cdot \big(\Phi(e_2)\big)^T\Big)^T & \text{if } \mathcal{S}(e_1) = \{A_2\} \text{ and } \mathcal{S}(e_2) = \{A_1\}; \\ \operatorname{diag}(\Phi(e_1)) \cdot \Phi(e_2) & \text{if } \mathcal{S}(e_1) = \{A_1\} \text{ and } \mathcal{S}(e_2) = \{A_1, A_2\}; \\ \Big(\big(\Phi(e_1)\big)^T \cdot \operatorname{diag}(\Phi(e_2))\Big)^T & \text{if } \mathcal{S}(e_1) = \{A_1, A_2\} \text{ and } \mathcal{S}(e_2) = \{A_1\}; \\ \Phi(e_1) \cdot \operatorname{diag}(\Phi(e_2)) & \text{if } \mathcal{S}(e_1) = \{A_1, A_2\} \text{ and } \mathcal{S}(e_2) = \{A_2\}; \\ \Big(\operatorname{diag}(\Phi(e_1)) \cdot \big(\Phi(e_2)\big)^T\Big)^T & \text{if } \mathcal{S}(e_1) = \{A_2\} \text{ and } \mathcal{S}(e_2) = \{A_1, A_2\}, \end{cases}$$

where $A_1 < A_2$ in each case where both attributes occur, and s is an abbreviation defined as follows. If f_1 and f_2 are two $\mathsf{MATLANG}$ expressions with $\mathcal{S}(f_1) = 1 \times 1$ and $\mathcal{S}(f_2) = \alpha \times \beta$, then $s(f_1, f_2) := \mathbf{1}(f_2) \cdot f_1 \cdot \big(\mathbf{1}({f_2}^T)\big)^T$. Notice that $\mathcal{S}(s(f_1, f_2)) = \alpha \times \beta$. Furthermore, if $\mathcal{J}$ is a matrix instance over $\mathcal{S}$ for which $f_1(\mathcal{J})$ is a 1×1 matrix with entry a and $f_2(\mathcal{J})$ is an $m \times n$ matrix for some $m, n \geq 1$, then $\big(s(f_1, f_2)\big)(\mathcal{J})$ is the $m \times n$ matrix in which all entries equal a.

The many cases above reflect all possible combination where $|\mathcal{S}(e_1)| \leq 2$, $|\mathcal{S}(e_2)| \leq 2$, and $|\mathcal{S}(e_1 \bowtie e_2)| \leq 2$, taking into account the order of the attributes. In the first case, where $\mathcal{S}(e_1) = \mathcal{S}(e_2)$, join reduces to pointwise multiplication of the tuple values, which is reflected by the Hadamard product in the translation. The second and third cases concern a join with a null-ary relation, which reduces to a scalar multiplication of the value of the empty tuple with the tuple values of the other relation. In the translation, this scalar multiplication is simulated using the abbreviation s explained above and the Hadamard product. Observe that we need to distinguish two cases, since K-multiplication is not necessarily commutative. The fourth and fifth cases concern the join of two disjoint unary relations, which results in considering all possible combinations of tuples and associating with them the product of their values. In the translation, this is simulated by a matrix product. Depending on the order of the attributes involved, a transpose may be in order, which is why there are again two cases here. The remaining cases involve the join of a binary relation and a unary relation over one of the attributes of the binary relation. To obtain the correct result, the value of each tuple of the binary relation must be multiplied, in the correct order, with the value of the corresponding tuple of the unary relation. In the translation, this is simulated using diagonalization and matrix multiplication. We need to distinguish four cases, since the order of the join matters (binary with unary or unary with binary), and since it also matters whether the unary relation is defined over the first or the second attribute (with respect to their mutual order) of the binary relation.

Composition. If e_1 and e_2 are $(\mathsf{ARA}+\zeta_2)(2)$ expressions over $\mathcal{S}$ with $\mathcal{S}(e_1) = \{A_1, A_3\}$, and $\mathcal{S}(e_2) = \{A_2, A_3\}$, and A_1, A_2, A_3 pairwise different, then

$$\Phi(\zeta_{A_3,2}(e_1,e_2)) := \Big(\big(\Phi(e_1)\big)^{T(A_1,A_3)} \cdot \big(\Phi(e_2)\big)^{T(A_3,A_2)}\Big)^{T(A_1,A_2)},$$

where, for attributes A and B and $\mathsf{MATLANG}$ expression f,

$$f^{T(A,B)} := \begin{cases} f & \text{if } A < B; \\ f^T & \text{if } A > B. \end{cases}$$

As may be expected, this operation can be simulated straightforwardly with matrix multiplication. Depending on the order of the attributes, however, transpositions may be in order.
Notice that we have only covered the case where $|\mathcal{S}(e_1) \triangle \mathcal{S}(e_2)| = 2$, where $\triangle$ denotes symmetric difference. If $|\mathcal{S}(e_1) \triangle \mathcal{S}(e_2)| \leq 1$, then $\zeta_{A_3,2}(e_1,e_2) \equiv \hat{\pi}_{A_3}(e_1 \bowtie e_2)$ is expressible in $\mathsf{ARA}(2)$ (since then $|\mathcal{S}(e_1 \bowtie e_2)| \leq 2$) . In this case, we first replace $\zeta_{A_3,2}(e_1,e_2)$ by its defining expression $\hat{\pi}_{A_3}(e_1 \bowtie e_2)$ and then proceed using the cases for projection and join above.

□

Observe that, in the above proof, the number of cases in the expression of $\Phi(e_1 \bowtie e_2)$ can be significantly reduced if the semiring K is commutative (in which case join is commutative).

Example 9 Consider again database schema $\mathcal{S}$, domain assignment D, and database instance $\mathcal{I}$ of $\mathcal{S}$ with respect to D from Examples 1 and 2 and Figure 1. Consider the $(\mathsf{ARA}+\zeta_2)(2)$ expression $e = \text{no\_courses} \bowtie \text{course\_fee}$ over $\mathcal{S}$. We have $\mathcal{S}(e) = \{\text{student}, \text{dptm}\}$ and

$e(\mathcal{I}) =$

student	dptm	K
1	1	1500
1	2	500
1	3	0
2	1	600
2	2	250
2	3	990

and $\mathrm{Mat}_D(e(\mathcal{I})) = \begin{pmatrix} 1500 & 500 & 0 \\ 600 & 250 & 990 \end{pmatrix}.$

By Lemma 3 and its proof, we have that $\mathrm{Mat}_D(e(\mathcal{I}))$ is equal to $e'(\mathrm{Mat}_D(\mathcal{I}))$ with $e' = \text{no\_courses} \cdot \mathsf{diag}(\text{course\_fee})$.

4.4 Relationship with ARA(3)

Corollary 1, Lemma 3, and Lemma 4 together establish the equivalence of $\mathsf{MATLANG}$ with the language $\mathsf{ARA}(3)$ restricted to database schemas and output relations of arity at most 2.

Theorem 2 *If K is commutative, then, for each* ARA(3) *expression of arity at most 2 over a database schema $\mathcal{S}$ of arity at most 2, there exists a* MATLANG *expression e' such that $\mathrm{Mat}_D(e(\mathcal{I})) = e'(\mathrm{Mat}_D(\mathcal{I}))$ for all consecutive domain assignments D and instances $\mathcal{I}$ with respect to $\mathcal{S}$ over D.*

Conversely, for each MATLANG *expression e over a matrix schema $\mathcal{S}$, there exists an* ARA(3) *expression e' such that $\mathrm{Rel}_{\mathcal{S}(e),\sigma}(e(\mathcal{I})) = e'(\mathrm{Rel}_{\mathcal{S},\sigma}(\mathcal{I}))$ for all size assignments σ and matrix instances $\mathcal{I}$ of $\mathcal{S}$ with respect to σ.*

4.5 Complexity of the translations

Now that we have established translations from MATLANG to ARA(3) via $(\mathsf{ARA}+\zeta_2)(2)$ as an intermediate step, we also want to look at the complexity of these translations, in terms of the lengths of the expressions involved.

First of all, we note that the translations Υ from MATLANG to $(\mathsf{ARA}+\zeta_2)(2)$ and Φ from $(\mathsf{ARA}+\zeta_2)(2)$ to MATLANG provided in the proofs of Lemmas 3 and 4, respectively, are strictly speaking exponential. They can, however, be readily adapted to become linear, provided the schema is considered to be fixed. (If the schema is not fixed and considered to be part of the input, these adaptations result in quadratic translations.) The required adaptations are the following.

We first consider the translation from MATLANG to $(\mathsf{ARA}+\zeta_2)(2)$. For a MATLANG expression e with $\mathcal{S}(e) = \alpha \times 1$ with $\alpha \neq 1$, there is a constant-length expression Tp_α with $\mathcal{S}(\mathrm{Tp}_\alpha) = \alpha \times 1$. Indeed, since α is a size symbol of $\mathcal{S}(e)$ distinct from 1, there is a matrix variable M with $\mathcal{S}(M)$ equal to either $\alpha \times \beta$ or $\beta \times \alpha$ for some β. Taking $\mathrm{Tp}_\alpha := \mathbf{1}(M)$ in the former case and $\mathrm{Tp}_\alpha := \mathbf{1}(M^T)$ in the latter case, we have $\mathcal{S}(\mathrm{Tp}_\alpha) = \alpha \times 1$ as desired. The only source of exponential growth in Lemma 3 is in the expression $\sigma_{\{\mathrm{row}_\alpha,\mathrm{col}_\alpha\}}(\Upsilon(e) \bowtie \mathbf{1}(\rho_{\mathrm{row}_\alpha\to\mathrm{col}_\alpha}(\Upsilon(e))))$ appearing in the diagonalization case, which is equivalent to $\sigma_{\{\mathrm{row}_\alpha,\mathrm{col}_\alpha\}}(\Upsilon(e) \bowtie \rho_{\mathrm{row}_\alpha\to\mathrm{col}_\alpha}(\Upsilon(\mathrm{Tp}_\alpha)))$.

For the converse translation, from $(\mathsf{ARA}+\zeta_2)(2)$ to MATLANG, we observe that, for an ARA expression e with $\mathcal{S}(e) := X \subseteq \{A_1, A_2\}$, there is a constant-length expression Tp_X with $\mathcal{S}(\mathrm{Tp}_X) = X$. Indeed, if $A \in \mathcal{S}(e)$, then there exists $A' \in \mathcal{S}(R_{A'})$ for some relation name $R_{A'}$ such that A' is compatible with A. Taking $\mathrm{Tp}_X := \bowtie_{A\in X} \rho_{A'\to A}(\pi_{\{A'\}}(R_{A'}))$ if $X \neq \emptyset$ and $\mathrm{Tp}_\emptyset := \pi_\emptyset(R)$ for some relation name R, we have $\mathcal{S}(\mathrm{Tp}_X) = X$ as desired. Replacing each occurrence of $\mathbf{1}(\Phi(e))$ by the equivalent expression $\mathbf{1}(\Phi(\mathrm{Tp}_{\mathcal{S}(e)}))$ and each occurrence of $\mathbf{1}\big(\big(\Phi(e)\big)^T\big)$ by the equivalent expression $\mathbf{1}\big(\big(\Phi(\mathrm{Tp}_{\mathcal{S}(e)})\big)^T\big)$ in the proof of Lemma 4 avoids exponential growth.

We remind the reader that $(\mathsf{ARA}+\zeta_2)(2)$ is subsumed by ARA(3), and, hence, the "translation" from $(\mathsf{ARA}+\zeta_2)(2)$ to ARA(3) comes for free. The translation from ARA(3) to $(\mathsf{ARA}+\zeta_2)(2)$ is exponential, however (Remark 4). In summary, we have provided linear translations in the following directions: MATLANG $\leftrightarrow$ $(\mathsf{ARA}+\zeta_2)(2) \to$ ARA(3).

4.6 Indistinguishability

Using a recent result by Geerts on indistinguishability in $\mathsf{MATLANG}$ [6], we can also relate $\mathsf{ARA}(3)$ to C^3, the three-variable fragment of first-order logic with counting [15]. Let A_1 and A_2 be matrices of the same dimensions $m \times n$. We view A_1 and A_2 as instances of a schema $\mathcal{S}$ on a single matrix name M with $\mathcal{S}(M) = \alpha \times \beta$, with respect to the size assignment σ that maps α to m and β to n. We say that A_1 and A_2 are *indistinguishable* in $\mathsf{MATLANG}$, denoted by $A_1 \equiv_{\mathsf{MATL}} A_2$, if for each $\mathsf{MATLANG}$ expression e over $\mathcal{S}$ with $\mathcal{S}(e) = 1 \times 1$, we have $e(A_1) = e(A_2)$. Similarly, one can define indistinguishability of binary K-relations r_1 and r_2 in $\mathsf{ARA}(3)$, denoted by $r_1 \equiv_{\mathsf{ARA}(3)} r_2$. This leads to the following corollary to Theorem 2.

Corollary 2 $A_1 \equiv_{\mathsf{MATL}} A_2$ *if and only if* $\mathrm{Rel}_{s,\sigma}(A_1) \equiv_{\mathsf{ARA}(3)} \mathrm{Rel}_{s,\sigma}(A_2)$.

Geerts's result concerns finite undirected graphs G_1 and G_2 with the same number of nodes. Recall that G_1 and G_2 are called indistinguishable in C^3, denoted by $G_1 \equiv_{\mathsf{C}^3} G_2$, if each C^3-sentence over a single binary relation variable has the same truth value on G_1 and G_2. Denote the adjacency matrix of G by $\mathsf{Adj}(G)$.

Theorem 3 ([6]) *If K is the field of complex numbers, then* $\mathsf{Adj}(G_1) \equiv_{\mathsf{MATL}} \mathsf{Adj}(G_2)$ *if and only if* $G_1 \equiv_{\mathsf{C}^3} G_2$.

We can immediately conclude the following, for suitable s and σ:

Corollary 3 *If K is the field of complex numbers, then $G_1 \equiv_{\mathsf{C}^3} G_2$ if and only if* $\mathrm{Rel}_{s,\sigma}(\mathsf{Adj}(G_1)) \equiv_{\mathsf{ARA}(3)} \mathrm{Rel}_{s,\sigma}(\mathsf{Adj}(G_2))$.

5 Conclusion

In related work, Yan, Tannen, and Ives consider provenance for linear algebra operators [20]. In that approach, provenance tokens represent not the matrix entries (as in our work), but the matrices themselves. Polynomial expressions (with matrix addition and matrix multiplication) are derived to show the provenance of linear algebra operations applied to these matrices.

Our result that every matrix query expressible in $\mathsf{ARA}(3)$ is also expressible in $\mathsf{MATLANG}$ provides a partial converse to the observation already made in the original paper [4], to the effect that $\mathsf{MATLANG}$ can be expressed in $\mathcal{L}_{\mathrm{Aggr}}(3)$: the relational calculus with summation and numerical functions [11], restricted to three base variables.[4] This observation was made in the extended setting of $\mathsf{MATLANG}$ that allows arbitrary pointwise functions (Remark 6). For the language considered here, $\mathsf{ARA}(3)$ provides a more appropriate upper bound for comparison, and $\mathsf{ARA}(3)$ is still a natural fragment of $\mathcal{L}_{\mathrm{Aggr}}(3)$.

[4] $\mathcal{L}_{\mathrm{Aggr}}$ is a two-sorted logic with base variables and numerical variables.

When allowing arbitrary pointwise functions in MATLANG, we actually move beyond the positive relational algebra, as queries involving negation can be expressed. For example, applying the function $x \wedge \neg y$ pointwise to the entries of two $n \times n$ Boolean matrices representing two binary relations R and S on $\{1, \dots, n\}$, we obtain the set difference $R \setminus S$. It is an interesting research question to explore expressibility of queries in MATLANG in this setting. For example, consider the following $\mathcal{L}_{\mathrm{Aggr}}(3)$ query on two matrices M and N:

$$\forall i \exists j \forall k \forall x \big(M(i,k,x) \to \exists i\, N(j,i,x)\big)$$

Here, $M(i,k,x)$ means that $M_{i,k} = x$, and similarly for $N(j,i,x)$.

The above query, which does not even use summation, reuses the base variable i and checks whether each row of M, viewed as a set of entries, is included in some row of N, again viewed as a set of entries. We conjecture that the query is not expressible in MATLANG with arbitrary pointwise functions. Developing techniques for proving this conjecture is an interesting direction for further research.

Finally, recall that our main result Corollary 1 assumes that K is commutative. It should be investigated whether or not this result still holds in the noncommutative case.

Acknowledgement

We thank Floris Geerts for inspiring discussions.

We also thank the anonymous reviewers for their comments which were very helpful in improving this paper.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Abo Khamis, M., Ngo, H., Rudra, A.: FAQ: Questions asked frequently. In: T. Milo, W. Tan (eds.) Proceedings 35th ACM IGMOD-SIGACT-SIGAI Symposium on Principles of Databases, San Francisco, CA, USA, June 26–July 01, 2016, pp. 13–28. ACM (2016)
3. Abo Khamis, M., Ngo, H., Rudra, A.: Juggling functions inside a database. SIGMOD Record **46**(1), 6–13 (2017)
4. Brijder, R., Geerts, F., Van den Bussche, J., Weerwag, T.: On the expressive power of query languages for matrices. In: B. Kimelfeld, Y. Amsterdamer (eds.) Proceedings 21st International Conference on Database Theory, Vienna, Austria, March 26–29, 2018, *LIPIcs*, vol. 98, pp. 10:1–10:17. Schloss Dagstuhl-Leibniz Center for Informatics (2018)
5. Brijder, R., Gyssens, M., Van den Bussche, J.: On matrices and K-relations. In: A. Herzig, J. Kontinen (eds.) Proceedings 11th International Symposium on Foundations of Information and Knowledge Systems, Dortmund, Germany, February 17–21, 2020, *Lecture Notes in Computer Science*, vol. 12012, pp. 42–57. Springer (2020)
6. Geerts, F.: On the expressive power of linear algebra on graphs. In: P. Barcelo, M. Calautti (eds.) Proceedings 22nd International Conference on Database Theory, Lisbon, Portugal, March 26–28, 2019, *LIPIcs*, vol. 127, pp. 7:1–7:19. Schloss Dagstuhl–Leibniz Center for Informatics (2019)

7. Green, T., Karvounarakis, G., Tannen, V.: Provenance semirings. In: L. Libkin (ed.) Proceedings 26th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Beijing, China, June 11–13, 2007, pp. 31–40 (2007)
8. Hutchison, D., Howe, B., Suciu, D.: LaraDB: A minimalist kernel for linear and relational algebra computation. In: F. Afrati, J. Sroka (eds.) Proceedings 4th ACM SIGMOD Workshop on Algorithms and Systems for MapReduce and Beyond, pp. 2:1–2:10 (2017)
9. Jananthan, H., Zhou, Z., Gadepally, V., Hutchison, D., Kim, S., Kepner, J.: Polystore mathematics of relational algebra. In: J.Y. Nie, Z. Obradovic, T. Suzumura, R. Ghosh, R. Nambiar, C. Wang, H. Zang, R. Baeza-Yates, X. Hu, J. Kepner, A. Cuzzocrea, J. Tang, M. Toyoda (eds.) Proceedings 2017 IEEE International Conference on Big Data, Boston, MA, USA, December 11–14, 2017, pp. 3180–3189. IEEE (2017)
10. Joglekar, M., Puttagunta, R., Ré, C.: AJAR: Aggregations and joins over annotated relations. In: T. Milo, W. Tan (eds.) Proceedings 35th ACM IGMOD-SIGACT-SIGAI Symposium on Principles of Databases, San Francisco, CA, USA, June 26–July 01, 2016, pp. 91–106. ACM (2016)
11. Libkin, L.: Expressive power of SQL. Theoretical Computer Science **296**(3), 379–404 (2003)
12. Luo, S., Gao, Z., Gubanov, M., Perez, L., Jermaine, C.: Scalable linear algebra on a relational database system. SIGMOD Record **47**(1), 24–31 (2018)
13. Maddux, R.: The origin of relation algebras in the development and axiomatization of the calculus of relations. Studia Logica **50**(3/4), 421–455 (1991)
14. Marx, M., Venema, Y.: Multi-Dimensional Modal Logic. Springer (1997)
15. Otto, M.: Bounded Variable Logics and Counting: A Study in Finite Models, *Lecture Notes in Logic*, vol. 9. Springer (1997)
16. Pratt, V.: Origins of the calculus of binary relations. In: Proceedings 7th Annual IEEE Symposium on Logic in Computer Science, Santa Cruz, CA, USA, June 22–25, 1992, pp. 248–254 (1992)
17. Tarski, A.: On the calculus of relations. Journal of Symbolic Logic **6**, 73–89 (1941)
18. Tarski, A., Givant, S.: A Formalization of Set Theory Without Variables, *AMS Colloquium Publications*, vol. 41. American Mathematical Society (1987)
19. Van den Bussche, J.: FO^3 and the algebra of binary relations. `https://databasetheory.org/node/94`. Retrieved 22 July 2019
20. Yan, Z., Tannen, V., Ives, Z.: Fine-grained provenance for linear algebra operators. In: S.C. Boulakia (ed.) 8th USENIX Workshop on the Theory and Practice of Provenance, Washington, DC, USA, June 8–9, 2016 (2016)