HAL
open science

Fragments of bag relational algebra: Expressiveness and certain answers

Marco Console, Paolo Guagliardo, Leonid Libkin

To cite this version:

Marco Console, Paolo Guagliardo, Leonid Libkin. Fragments of bag relational algebra: Expressiveness and certain answers. Information Systems, 2022, 105, pp.101604. 10.1016/j.is.2020.101604 . hal-03934340

HAL Id: hal-03934340

https://inria.hal.science/hal-03934340v1

Submitted on 5 Dec 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ARTICLE IN PRESS

Information Systems xxx (xxxx) xxx

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is

Fragments of bag relational algebra: Expressiveness and certain answers[☆,☆☆]

Marco Console, Paolo Guagliardo[*], Leonid Libkin

School of Informatics, University of Edinburgh, United Kingdom

ARTICLE INFO

Article history:
Received 20 November 2019
Received in revised form 1 June 2020
Accepted 17 July 2020
Available online xxxx
Recommended by Gottfried Vossen

Keywords:
Bag semantics
Relational algebra
Expressiveness
Certain answers
Complexity

ABSTRACT

While all relational database systems are based on the bag data model, much of theoretical research still views relations as sets. Recent attempts to provide theoretical foundations for modern data management problems under the bag semantics concentrated on applications that need to deal with incomplete relations, i.e., relations populated by constants and nulls. Our goal is to provide a complete characterization of the complexity of query answering over such relations in fragments of bag relational algebra. The main challenges that we face are twofold. First, bag relational algebra has more operations than its set analog (e.g., additive union, max-union, min-intersection, duplicate elimination) and the relationship between various fragments is not fully known. Thus we first fill this gap. Second, we look at query answering over incomplete data, which again is more complex than in the set case: rather than certainty and possibility of answers, we now have numerical information about occurrences of tuples. We then fully classify the complexity of finding this information in all the fragments of bag relational algebra.

© 2020 Elsevier Ltd. All rights reserved.

1. Introduction

While all relational database management systems (DBMSs) use bags as the basis of their data model, much of relational database theory uses a model based on sets, thus disallowing repetitions of tuples. The presence of duplicates in real-life databases is a very important consideration that is reflected in practically all aspects of data management, such as querying, storing, and accessing data [2,3]. Theoretical research has raised this issue several times. By the early 1990s there was agreement on what the standard collection of bag relational algebra operations is [4], and in the mid 1990s their expressiveness and complexity were thoroughly studied [5,6], albeit in the context of the model of nested relations, or complex objects, which was the research focus back then [7,8]. Around the same time it was noticed that the well developed theory of query optimization, especially for conjunctive queries, does not apply to bag semantics [9], and despite many attempts and partial results [10,11], the key problem of the decidability of such optimizations remains unsolved [12]. Other languages, in particular those with aggregates and fixpoints in the spirit of Datalog, have been studied under bag semantics as well [6,13,14].

More recently, bag semantics has been considered in modern data management applications that combine traditional databases and reasoning tasks. In [15], fundamental problems of data integration and data exchange are studied under bag semantics and are shown to differ rather drastically from their set semantics counterparts. In [16], a similar program is carried out for ontology based data access (OBDA), where an ontology supplements information provided by a relational database in which duplicates are allowed. What is common to these applications is that in both of them one needs to query incomplete data, that is, databases with null values. The standard approach to querying such databases, which is used in data integration, data exchange, and OBDA applications, is based on the classical notion of certain answers [17].

However, when it comes to bags, the notion of certain answers becomes more complex than under set semantics. In general, an incomplete database D represents a collection $[\![D]\!] = \{D_1, D_2, \ldots\}$ of complete databases, obtained by interpreting incomplete data in D. A tuple $\bar{a}$ is a certain answer to a query q if it is in $q(D')$ for every $D' \in [\![D]\!]$; see [17,18]. Under bag semantics, we have more information: for each tuple, we know the number $\#\big(\bar{a}, q(D')\big)$ of occurrences of $\bar{a}$ in $q(D')$. Thus, as D' ranges over $[\![D]\!]$, we have a range of numbers that define an interval between

$$\min(\bar{a}, q, D) = \min_{D' \in [\![D]\!]} \#\big(\bar{a}, q(D')\big)$$

☆ Work supported by EPSRC, United Kingdom grants M025268 and N023056.

☆☆ This is the full version of a paper of the same title that appeared in the proceedings of ICDT 2019, the 22nd International Conference on Database Theory [1]. This work was also presented at AMW 2019, the 13th Alberto Mendelzon International Workshop on Foundations of Data Management, where it was invited to this special issue.

* Corresponding author.
E-mail addresses: mconsole@ed.ac.uk (M. Console), paolo.guagliardo@ed.ac.uk (P. Guagliardo), libkin@inf.ed.ac.uk (L. Libkin).

https://doi.org/10.1016/j.is.2020.101604
0306-4379/© 2020 Elsevier Ltd. All rights reserved.

and

$$\max(\bar{a}, q, D) = \max_{D' \in [\![D]\!]} \#\big(\bar{a}, q(D')\big).$$

Under set semantics, $\min(D, q, \bar{a}) = 1$ means that $\bar{a}$ is a certain answer to q on D, and $\max(D, q, \bar{a}) = 1$ means that $\bar{a}$ is a possible answer. On sets, these can be easily checked for positive relational algebra, but it is hard (coNP-complete and NP-complete, respectively) for full relational algebra [19]. This tells us that, in terms of relational algebra operations, selection, projection, Cartesian product and union are easy, but difference makes things hard. Our goal is to paint a similar picture for bags. The problems that we face are:

(1) there are more operations that are included in bag relational algebra, and we have less understanding of them;
(2) even for basic operations, there is very little knowledge of the complexity of answering queries over incomplete data.

We now explain these points in more detail.

Bag algebra fragments. Under set semantics, we have well understood fragments of relational algebra: SPC (select-project-Cartesian product) queries, positive relational algebra RA^+ that adds union, and full relational algebra RA that adds difference. Moreover, intersection is expressible as the natural join of two relations over the same attributes. Under bag semantics, however, the situation is different:

- SPC queries follow their set-theoretic analogs but they keep duplicates.
- For union, there are two options: *max-union*, which takes the maximum number of occurrences of a tuple, and *additive union*, which adds up multiplicities (and corresponds to `UNION ALL` in SQL).
- Intersection (SQL's `INTERSECT ALL`) takes the minimum number of occurrences of a tuple and it can no longer be expressed as a join in general.
- Difference (SQL's `EXCEPT ALL`) subtracts multiplicities of tuples up to zero.
- There is the *duplicate elimination* operation, which sets the multiplicities of tuples to 1.

When the multiplicities of tuples in the input relations are at most 1, the bag operations of difference, intersection and max-union coincide with set-theoretic difference, intersection and union, respectively. SQL's `EXCEPT`, `INTERSECT` and `UNION` (without the `ALL` modifier) are variants of difference, intersection and max-union, respectively, that treat input relations as sets by removing duplicates from them; `UNION` can also be seen as additive union followed by duplicate elimination.

The language RA of bag relational algebra consists of the following operations [4–6]:

- multiplicity-preserving versions of *selection* (σ), *projection* (π) and *Cartesian product* ($\times$), which form the class of SPC queries;
- *additive union* $\uplus$ that adds up multiplicities of tuples; together with SPC queries it forms the *positive relational algebra* RA^+;
- *max-union* $\cup$ that keeps the maximum number of occurrences of a tuple;
- *min-intersection* $\cap$ that keeps the minimum number of occurrences of a tuple;
- *difference* $-$ that subtracts the number of occurrences of a tuple up to zero, i.e., $\#(\bar{a}, R-R') = \max\big(\#(\bar{a}, R)-\#(\bar{a}, R'), 0\big)$;
- *duplicate elimination* ε that turns a bag into a set.

Example 1. For bags $R = \{a, a, b, b, c\}$ and $S = \{a, b, b, b\}$, we have:

$$R \uplus S = \{a, a, a, b, b, b, b, b, c\}\,; \qquad R - S = \{a, c\}\,;$$
$$R \cup S = \{a, a, b, b, b, c\}\,; \qquad \varepsilon(R) = \{a, b, c\}\,,$$
$$R \cap S = \{a, b, b\}\,; \qquad \varepsilon(S) = \{a, b\}\,.$$

We remark that, differently from additive union $\uplus$, the max-union operation $\cup$ is not explicitly implemented in any standard query language. To see how it could be used in practice, consider a scenario with two sensors, r and s, logging data about events a_1, a_2, etc. into unary relations R and S, as follows: whenever an event a_i is detected by r (resp., s), an occurrence of a_i is added to R (resp., S). Now assume that for any given event, one of the sensors can reliably detect all of its occurrences, while the other may, or may not. Then, $R \cup S$ correctly collates the data from the two sensors' logs, returning all the reliably detected occurrences of *every* event, but $R \uplus S$ does not.

To understand how query answering behaves in the fragments of RA, we first need to understand their relative expressiveness. It *might* appear that these questions have already been answered in [6,20]. However, this was done in the context of nested relations, and the results used the power of nesting in an essential way. For the usual bag algebra with non-nested relations, as implemented in all DBMSs, these basic results are surprisingly lacking. Thus, as our first task, we shall produce a full picture of the expressiveness of bag relational algebra fragments (which will indeed be different from the known results in the nested case).

Incomplete information and bags. There is a much bigger variety of relational algebra fragments for bags, but little is known about finding $\min(\bar{a}, q, D)$ and $\max(\bar{a}, q, D)$ for queries in those fragments. We know that min is easy to compute for RA^+ queries and that for full RA the problem is computationally hard: checking whether $\min(\bar{a}, q, D) \geq n$ is NP-complete [19,21]. Checking whether $\max(\bar{a}, q, D) \geq n$ is NP-complete even for SPC queries [21]. The complexity of actually computing min and max (or, in terms of a decision problem, checking whether $\min(D, q, D) = n$, and likewise for max) is still open.

Outline of the results. Our main results are summarized in Fig. 1.

Expressiveness We characterize the relative expressive power of RA fragments, as shown in the diagram. Furthermore, adding duplicate elimination to a fragment that does not have it results in a language that is strictly more expressive (than the original fragment), and incomparable with $\mathrm{RA}^+\{-\}$ (i.e., RA^+ with difference). The relative expressiveness of bag operations is indeed different from what was known in the nested relational case [5,6,20]. For example, over nested relations, adding min-intersection to the analog of RA^+ suffices to express max-union, but in the usual relational algebra over bags these two operations are incomparable in their expressiveness.

Complexity of min For fragments in the shaded area, computing $\min(\bar{a}, q, D)$ is tractable, and it can be done by evaluating the query naively on the incomplete database. For all fragments outside the shaded area, and all fragments with duplicate elimination (from $\mathrm{SPC}\{\varepsilon\}$ to the full RA), the complexity is intractable: checking whether $\min(\bar{a}, q, D)\ \theta\ n$ is NP-complete when θ is $\leq$, coNP-complete when θ is $\geq$, and DP-complete when θ is $=$. Recall that DP is the class of problems that are the intersection of an NP problem and a coNP problem [22].

Complexity of max For all the fragments, inside and outside the shaded area, and with or without duplicate elimination, computing max is intractable: checking $\max(\bar{a}, q, D)\ \theta\ n$ is NP-complete when θ is $\geq$, coNP-complete when θ is $\leq$, and DP-complete when θ is $=$.

Please cite this article as: M. Console, P. Guagliardo and L. Libkin, Fragments of bag relational algebra: Expressiveness and certain answers, Information Systems (2020) 101604, https://doi.org/10.1016/j.is.2020.101604.

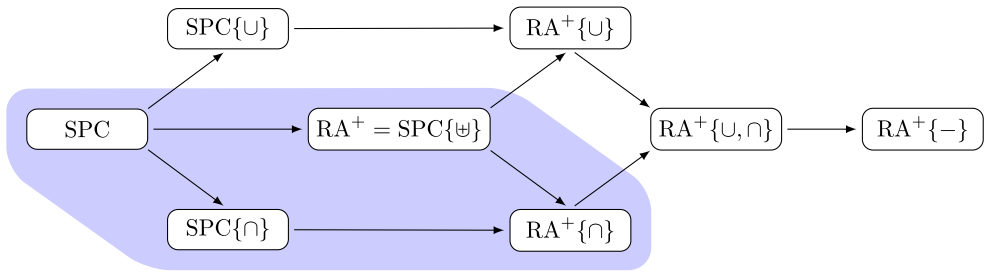

Fig. 1. Summary of the results. An edge indicates a more expressive fragment. Adding duplicate elimination makes every fragment more expressive, and incomparable with $\mathrm{RA}^+\{-\}$. Extending $\mathrm{RA}^+\{-\}$ with duplicate elimination results in a fragment that has the expressive power of full RA. The shaded area includes the fragments for which computing the minimum occurrences of certain answers is tractable, while this is intractable whenever duplicate elimination is added. Computing the maximum number of occurrences is intractable for all fragments.

Organization. Bag relational algebra is defined in Section 2, and the relative expressive power of its fragments is studied in Section 3. Query answering over bags with nulls is discussed in Section 4, and its complexity is classified in Section 5. Concluding remarks are given in Section 6.

2. Bag relational algebra

We now describe the standard operations of bag relational algebra and provide their semantics [4–6,20]. A bag is a collection of elements with associated multiplicities (numbers of occurrences); if an element b occurs n times in a bag B, we write $\#(b, B) = n$. If $\#(b, B) = 0$, it means that b does not occur in B. Sets are just a special case when $\#(b, B) \in \{0, 1\}$.

In a database D, each k-ary relation name R of the schema is associated with a bag R^D of k-tuples; we will omit the superscript D whenever it is clear from the context. We assume that the attributes of a k-ary relation are $1, \ldots, k$, i.e., we adopt the unnamed perspective [18].

Syntax. The syntax of relational algebra (RA) expressions is defined as follows:

$$
\begin{array}{rcll}
e, e' & ::= & R & \text{(base relations)} \\
& | & \sigma_{i=j}(e) & \text{(selection)} \\
& | & \pi_\alpha(e) & \text{(projection)} \\
& | & e \times e' & \text{(Cartesian product)} \\
& | & e \uplus e' & \text{(additive union)} \\
& | & e \cup e' & \text{(max-union)} \\
& | & e \cap e' & \text{(intersection)} \\
& | & e - e' & \text{(difference)} \\
& | & \varepsilon(e) & \text{(duplicate elimination)}
\end{array}
$$

where i and j in $\sigma_{i=j}(e)$ are positive integers, and α in $\pi_\alpha(e)$ is a possibly empty tuple of positive integers.

The *arity* of RA expressions is defined as follows: for base relations, it is given by the schema; for $\sigma_{i=j}(e)$ and $\varepsilon(e)$, it is the arity of e; for $\pi_\alpha(e)$, it is the arity of α; for $e \times e'$, it is the sum of the arities of e and e'; for $e \star e'$ with $\star \in \{\cup, \uplus, \cap, -\}$, it is the arity of e.

We then say that an RA expression is well-formed w.r.t. a schema if: it mentions only relation names from the schema; i and j in $\sigma_{i=j}(e)$ are less than or equal to the arity of e; the elements of α in $\pi_\alpha(e)$ are less than or equal to the arity of e; the expressions e and e' in $e \star e'$, with $\star \in \{\cup, \uplus, \cap, -\}$, have the same arity. In the rest of the paper, we implicitly assume that we are always working with well-formed RA expressions.

Semantics. We give the semantics of (well-formed) RA expressions e by inductively defining the quantity $\#(\bar{a}, e, D)$, which is the number of occurrences of a tuple $\bar{a}$ (of appropriate arity) in the result of applying e to a database D. This is done as follows:

$$
\begin{aligned}
\#(\bar{a},\ R,\ D) &= \#\big(\bar{a}, R^D\big) \\
\#(\bar{a},\ \sigma_{i=j}(e),\ D) &= \begin{cases} \#(\bar{a}, e, D) & \text{if } \bar{a}.i = \bar{a}.j \\ 0 & \text{otherwise} \end{cases} \\
\#(\bar{a},\ \pi_\alpha(e),\ D) &= \sum_{\bar{a}' : \pi_\alpha(\bar{a}') = \bar{a}} \#(\bar{a}', e, D) \\
\#\big(\bar{a}\bar{a}',\ e \times e',\ D\big) &= \#\big(\bar{a}, e, D\big) \cdot \#\big(\bar{a}', e', D\big) \\
\#(\bar{a},\ e \uplus e',\ D) &= \#(\bar{a}, e, D) + \#(\bar{a}, e', D) \\
\#(\bar{a},\ e \cup e',\ D) &= \max\big\{\ \#(\bar{a}, e, D),\ \#(\bar{a}, e', D)\ \big\} \\
\#(\bar{a},\ e \cap e',\ D) &= \min\big\{\ \#(\bar{a}, e, D),\ \#(\bar{a}, e', D)\ \big\} \\
\#(\bar{a},\ e - e',\ D) &= \max\big\{\ \#(\bar{a}, e, D) - \#(\bar{a}, e', D),\ 0\ \big\} \\
\#(\bar{a},\ \varepsilon(e),\ D) &= \min\big\{\ \#(\bar{a}, e, D),\ 1\ \big\}
\end{aligned}
$$

where $\bar{a}.i$ denotes the ith element of $\bar{a}$, $\pi_{i_1,\ldots,i_n}(\bar{a})$ is the tuple $(\bar{a}.i_1, \ldots, \bar{a}.i_n)$, and the tuples $\bar{a}$ and $\bar{a}'$ in the rule for $e \times e'$ have the same arity as e and e', respectively.

Then, for an expression e and a database D, we define $e(D)$ as the bag of tuples $\bar{a}$ of the same arity as e so that $\#\big(\bar{a}, e(D)\big) = \#(\bar{a}, e, D)$.

Fragments. The two main fragments of RA we consider are SPC, consisting of selection (σ), projection (π) and Cartesian product ($\times$), and RA^+, which is SPC extended with additive union ($\uplus$). Given a fragment $\mathcal{L}$ of RA, we write $\mathcal{L}\{\mathsf{op}_1, \ldots, \mathsf{op}_n\}$ to denote the fragment obtained by adding the RA operations $\mathsf{op}_1, \ldots, \mathsf{op}_n$ to $\mathcal{L}$. Thus, for instance, RA^+ is $\mathrm{SPC}\{\uplus\}$.

A *query* is a mapping q from databases to bags of tuples. We always assume that queries are generic, that is, invariant under permutations of the domain [18]. A query q is *expressible* in a fragment $\mathcal{L}$ of RA if there is an expression e in that fragment so that $e(D) = q(D)$ for every database D.

Then, given two fragments $\mathcal{L}$ and $\mathcal{L}'$, we say that $\mathcal{L}'$ is *at least as expressive* as $\mathcal{L}$, and write $\mathcal{L} \subseteq \mathcal{L}'$, if every query expressible in $\mathcal{L}$ is also expressible in $\mathcal{L}'$. We say that $\mathcal{L}'$ is *more expressive* than $\mathcal{L}$, and write $\mathcal{L} \subsetneq \mathcal{L}'$, if $\mathcal{L}'$ is at least as expressive as $\mathcal{L}$ and there is a query that is expressible in $\mathcal{L}'$ but not in $\mathcal{L}$. Notice that if $\mathcal{L}'$ has all the operations of $\mathcal{L}$, then $\mathcal{L}'$ is at least as expressive as $\mathcal{L}$.

3. Expressive power of bag relational algebra fragments

In this section, we study the relative expressiveness of RA fragments. We present the results that justify the edges in Fig. 1, along with additional results that are not explicitly captured in that diagram.

We start by showing that extending positive relational algebra with max-union or intersection results in a more expressive fragment.

Proposition 1. $\mathrm{RA}^+ \subsetneq \mathrm{RA}^+\{\star\}$ *for* $\star \in \{\cup, \cap\}$.

Proof. Trivially $\mathrm{RA}^+\{\star\}$ is at least as expressive as RA^+, so we only need to show that there exists a query that is expressible in the former but not in latter. To this end, consider a schema consisting of two nullary (i.e., of arity 0) relation symbols R and S, and suppose that $R \star S$ is expressible in RA^+, i.e., there exists an RA^+ expression e equivalent to $R \star S$. For a database D, let $m = |R^D|$ and $n = |S^D|$; then, $|e(D)| = f_\star(m, n)$, where $f_\cup = \max$ and $f_\cap = \min$. Moreover, $|e(D)|$ is expressible by a polynomial $p_e \in \mathbb{N}[m, n]$, because $e \in \mathrm{RA}^+$ [23].

For $n_0 \in \mathbb{N}$, we define the polynomial $p \in \mathbb{N}[m]$ such that $p(m) = p_e(m, n_0)$ and distinguish the following two cases.

- When $\star = \cup$, let $n_0 = \deg(p_e) \geq \deg(p)$. Then, $p(m) = \max(m, n_0) = n_0$ for every $m \in \{0, \ldots, n_0\}$. But this is impossible, as the unique polynomial $\tilde{p} \in \mathbb{N}[m]$ of degree at most n_0 that interpolates the $n_0 + 1$ data points $\{(i, n_0) \mid i = 0, \ldots, n_0\}$ is the constant function $\tilde{p}(m) = n_0$, and $\tilde{p}(m) \neq p(m)$ for $m > n_0$.
- When $\star = \cap$, let n_0 be a positive integer. We have $\deg(p) \geq 1$, as $p(m) = \min(m, n_0) = n_0 > 0$ for every $m \geq n_0$, while $p(0) = 0$. Then p is strictly increasing in the interval $[0, +\infty)$, since its coefficients are in $\mathbb{N}$. But this is impossible because $p(m) = n_0$ for every $m \geq n_0$. □

The proof of Proposition 1 also applies to show the following.

Corollary 1. $\mathrm{SPC} \subsetneq \mathrm{SPC}\{\star\}$ *for* $\star \in \{\cup, \cap\}$.

We now show that additive union increases the expressive power of $\mathrm{SPC}\{\cap\}$ and $\mathrm{SPC}\{\cup\}$.

Proposition 2. $\mathrm{SPC}\{\star\} \subsetneq \mathrm{RA}^+\{\star\}$ *for* $\star \in \{\cup, \cap\}$.

The proof of the above result is based on the fact that, on databases consisting of nullary relations of size 1, there are RA^+ expressions whose results size is greater than 1, whereas every $\mathrm{SPC}\{\cup, \cap\}$ expression can only yield a (nullary) relation of size 1, as the following lemma shows.

Lemma 1. *Let D be a database over a schema consisting only of nullary relation names and such that $R^D = \{()\}$ for every relation name R in the schema. Then, for every $\mathrm{SPC}\{\cup, \cap\}$ expression e, it is the case that $e(D) = \{()\}$.*

Proof. Let e be an $\mathrm{SPC}\{\cup, \cap\}$ expression over the schema of D. Observe that all subexpressions of e are nullary and, since there are no attributes, none of them can be a selection. Moreover, projection will be over an empty list of attributes and so it can be discarded. Thus, assuming w.l.o.g. that there is no projection, we proceed by induction on the structure of e to show that $e(D) = \{()\}$.

The base case is when e is a relation name R; then trivially $e(D) = R^D = \{()\}$. For the inductive step, we have that $e = e_1 \star e_2$ with $\star \in \{\times, \cup, \cap\}$ and, by the inductive hypothesis, $e_1(D) = e_2(D) = \{()\}$. Then, from the semantics of $\times$, $\cap$ and $\cup$, we get $e(D) = e_1(D) \star e_2(D) = \{()\}$. □

Proof of Proposition 2. Let $\star \in \{\cup, \cap\}$. Trivially $\mathrm{RA}^+\{\star\}$ is at least as expressive as $\mathrm{SPC}\{\star\}$, thus we only need to show that there is a query expressible in the former but not in latter. To this end, consider a schema consisting of a nullary relation name R, and suppose that $e = R \uplus R$ (which, in particular, is an RA^+ expression) is expressible in $\mathrm{SPC}\{\star\}$, i.e., there exists an $\mathrm{SPC}\{\star\}$ expression e' equivalent to e. Towards a contradiction of this equivalence, let D be a database where $R^D = \{()\}$. Then, since e' is an $\mathrm{SPC}\{\cup, \cap\}$ expression, by Lemma 1 we have that $e'(D) = \{()\}$, whereas $e(D) = \{(), ()\}$. □

In particular, the proof of Proposition 2 also applies to show the following.

Corollary 2. $\mathrm{SPC} \subsetneq \mathrm{RA}^+$.

Next, we will show that $\cup$ increases the expressive power of $\mathrm{RA}^+\{\cap\}$, and $\cap$ increases the expressive power of $\mathrm{RA}^+\{\cup\}$. In turn, this implies that $\cup$ and $\cap$ are incomparable operations.

Proposition 3. $\mathrm{RA}^+\{\star\} \subsetneq \mathrm{RA}^+\{\cup, \cap\}$ *for* $\star \in \{\cup, \cap\}$.

To prove the above, we need some additional notions and lemmas. We start by introducing a special syntactic form for $\mathrm{RA}^+\{\cap\}$ expressions.

Definition 1. An $\mathrm{RA}^+\{\cap\}$ expression is in *intersection normal form* ($\cap$-NF) if it is of the form $e_1 \cap \cdots \cap e_n$ where each e_i is an RA^+ expression. Observe that this is well defined because $\cap$ is commutative and associative.

Lemma 2. *Every $\mathrm{RA}^+\{\cap\}$ expression over nullary relation names can be equivalently rewritten to $\cap$-NF.*

Proof. Let e be an $\mathrm{RA}^+\{\cap\}$ expression over nullary relation names. As before, note that all subexpressions of e are nullary and, since there are no attributes, none of them can be a selection. Moreover, projection will be over an empty list of attributes and therefore it can be discarded. Assuming w.l.o.g. that there is no projection, we proceed by induction on the structure of e.

Base case: e is R. Then $R \cap R$ is obviously equivalent to e and in $\cap$-NF.

Inductive step: e is $e' \star e''$ with $\star \in \{\uplus, \times\}$. By the induction hypothesis, there are $\cap$-NF expressions $e'_1 \cap \cdots \cap e'_m$ and $e''_1 \cap \cdots \cap e''_n$ equivalent to e' and e'', respectively. Then, the following $\cap$-NF expression:

$$\bigcap_{i=1}^{m} \bigcap_{j=1}^{n} (e'_i \star e''_j)$$

is equivalent to e because:

$$f_\star\big(\min(x_1, \ldots, x_m), \min(y_1, \ldots, y_n)\big) = \min\big(f_\star(x_1, y_1), \ldots, f_\star(x_1, y_n), \ldots, f_\star(x_m, y_1), \ldots, f_\star(x_m, y_n)\big)$$

where:

- $f_\times(x, y) = x \cdot y$ and $f_\uplus(x, y) = x + y$; and,
- for any given database D, we have that each $x_i = |e'_i(D)|$ and each $y_j = |e''_j(D)|$. □

The proofs of the next two lemmas can be found in Appendix.

Lemma 3. *Let D be a database over a schema with a nullary relation symbol R, and let e be an $\mathrm{RA}^+\{\cup\}$ expression that mentions only R. Then, $|e(D)| \geq |R^D|$.*

Lemma 4. *Let D be a database over a schema consisting of two nullary relation symbols R and S, and let e be an $\mathrm{RA}^+\{\cup\}$ expression that mentions both R and S. Then, $|e(D)| \geq |R^D|$ and $|e(D)| \geq |S^D|$.*

We are now ready to prove the result on the expressiveness of $\mathrm{RA}^+\{\cup, \cap\}$ stated earlier.

Please cite this article as: M. Console, P. Guagliardo and L. Libkin, Fragments of bag relational algebra: Expressiveness and certain answers, Information Systems (2020) 101604, https://doi.org/10.1016/j.is.2020.101604.

Proof of Proposition 3.

- $\mathrm{RA}^+\{\cup\} \subsetneq \mathrm{RA}^+\{\cap,\cup\}$
  Trivially $\mathrm{RA}^+\{\cap,\cup\}$ is at least as expressive as $\mathrm{RA}^+\{\cup\}$; therefore, we only need to show that there exists a query expressible in $\mathrm{RA}^+\{\cap,\cup\}$ that is not expressible in $\mathrm{RA}^+\{\cup\}$. To this end, consider a schema consisting of two nullary relation symbols R and S, and the $\mathrm{RA}^+\{\cap,\cup\}$ expression $e = R \cap S$. Towards a contradiction, suppose that e is expressible in $\mathrm{RA}^+\{\cup\}$, i.e., there exists an $\mathrm{RA}^+\{\cup\}$ expression e' equivalent to e.
  Let us first consider the case when e' mentions only either R or S; w.l.o.g., say R. Take D such that $|R^D| > |S^D|$; then $|e(D)| = |S^D|$, and $|e'(D)| \geq |R^D|$ by Lemma 3. Therefore, $|e'(D)| > |e(D)|$, which contradicts the equivalence of e and e'.
  When e' mentions both R and S, we can again take D such that $|R^D| > |S^D|$, which leads to the same contradiction, since $|e(D)| = |S^D|$, and $|e'(D)| \geq |R^D|$ by Lemma 4.
- $\mathrm{RA}^+\{\cap\} \subsetneq \mathrm{RA}^+\{\cap,\cup\}$
  Trivially $\mathrm{RA}^+\{\cap,\cup\}$ is at least as expressive as $\mathrm{RA}^+\{\cap\}$; therefore, we only need to show that there exists a query expressible in $\mathrm{RA}^+\{\cap,\cup\}$ that is not expressible in $\mathrm{RA}^+\{\cap\}$. To this end, consider a schema consisting of two nullary relation symbols R and S, and the $\mathrm{RA}^+\{\cap,\cup\}$ expression $e = R \cup S$. Towards a contradiction, suppose that e is expressible in $\mathrm{RA}^+\{\cap\}$, i.e., there exists an $\mathrm{RA}^+\{\cap\}$ expression e' equivalent to e. By Lemma 2, e' is equivalent to an expression of the form $e_1 \cap \cdots \cap e_k$ where each e_i is an RA^+ expression. Then, for every database D, whenever $m = |R^D|$ and $n = |S^D|$ we have that
  $$\underbrace{\max(m,n)}_{|e(D)|} = \underbrace{\min\big(p_1,\ldots,p_k\big)}_{|e'(D)|}$$
  where each p_i is a bivariate polynomial in $\mathbb{N}^+[m,n]$. Let d be the maximum degree of a polynomial among $p_1,\ldots,p_k$, and let $n_0 = (d+1)k+1$. Then for each $m < n_0$ we have $n_0 = \max(m,n_0) = \min\big(p_1(m,n_0),\ldots,p_k(m,n_0)\big)$. Consider now univariate polynomials $p'_i(m) = p_i(m,n_0) - n_0$ for $i \leq k$. We have $0 = n_0 - n_0 = \min\big(p'_1(m),\ldots,p'_k(m)\big)$ for all $m < n_0$. By pigeonhole, it means that there is a polynomial $p'_j(m)$ and at least $d+1$ distinct values $m_1,\ldots,m_{d+1}$ smaller than n_0 such that $p'_j(m_1) = \ldots = p'_j(m_{d+1}) = 0$. Since the degree of p'_j is at most d, this implies that $p'_j(m)$ is identically zero, and thus $p_j(m,n_0) = n_0$ for all m.
  Now, using $\max(m,n) = \min\big(p_1(m,n),\ldots,p_k(m,n)\big)$, we have
  $$\begin{aligned} n_0+1 &= \max(n_0+1,n_0) \\ &= \min\big(p_1(n_0+1,n_0),\ldots,p_k(n_0+1,n_0)\big) \\ &\leq p_j(n_0+1,n_0) = n_0\,, \end{aligned}$$
  which contradicts the assumption that $\cup$ is expressible in $\mathrm{RA}^+\{\cap\}$. $\square$

The following is a direct consequence of Proposition 3.

Corollary 3. *$\mathcal{L}\{\cup\}$ and $\mathcal{L}\{\cap\}$ are incomparable, for $\mathcal{L} \in \{\mathrm{SPC}, \mathrm{RA}^+\}$.*

Finally, we show that with additive union and difference one can express both intersection and max-union, therefore $\mathrm{RA}^+\{-\}$ is the most expressive fragment of RA without duplicate elimination.

Proposition 4. $\mathrm{RA}^+\{\cap,\cup\} \subsetneq \mathrm{RA}^+\{-\}$.

Proof. Let $e \in \mathrm{RA}^+\{\cap,\cup\}$; by induction on the structure of e we will show that there exists an expression $e' \in \mathrm{RA}^+\{-\}$ that is equivalent to e. The base case is when $e \in \mathrm{RA}^+$, which is trivially also in $\mathrm{RA}^+\{-\}$. For the inductive step, let $e = e_1 \star e_2$, with $e_1, e_2 \in \mathrm{RA}^+\{\cap,\cup\}$ and $\star \in \{\cap,\cup\}$. By the induction hypothesis, there exist $e'_1, e'_2 \in \mathrm{RA}^+\{-\}$ such that $e'_i \equiv e_i$ for $i = 1,2$. Consider the $\mathrm{RA}^+\{-\}$ expressions $e_\cap = e'_1 - (e'_1 - e'_2)$ and $e_\cup = e'_1 \uplus (e'_2 - e'_1)$; we will show that $e_\star \equiv e_1 \star e_2$ for $\star \in \{\cap,\cup\}$. To this end, let D be a database and let $\bar{a}$ be a tuple; moreover, let $m = \#(\bar{a}, e'_1, D)$ and $n = \#(\bar{a}, e'_2, D)$. Then, we have the following:

$$\begin{aligned} k_\cap &= \#(\bar{a}, e_\cap, D) = \#(\bar{a}, e'_1, D) \dot{-} \big(\#(\bar{a}, e'_1, D) \dot{-} \#(\bar{a}, e'_2, D)\big) \\ &= m \dot{-} (m \dot{-} n) \\ k_\cup &= \#(\bar{a}, e_\cup, D) = \#(\bar{a}, e'_1, D) + \big(\#(\bar{a}, e'_2, D) \dot{-} \#(\bar{a}, e'_1, D)\big) \\ &= m + (n \dot{-} m) \end{aligned}$$

If $m \geq n$, we have that $0 \leq m \dot{-} n = m - n \leq m$ and $n \dot{-} m = 0$, therefore $k_\cap = m - (m-n) = n$ and $k_\cup = m$. Otherwise, when $n > m$, we have that $m \dot{-} n = 0$ and $n \dot{-} m = n - m$, so $k_\cap = m \dot{-} 0 = m$ and $k_\cup = m + n - m = n$. Thus, $k_\cap = \min(m,n)$ and $k_\cup = \max(m,n)$.

Now, since $e'_1 \equiv e_1$ and $e'_2 \equiv e_2$ by the induction hypothesis, $\#(\bar{a}, e_1, D) = m$ and $\#(\bar{a}, e_2, D) = n$. Hence, for $\star \in \{\cap,\cup\}$, we have $k_\star = \#(\bar{a}, e_1 \star e_2, D)$ and, as D and $\bar{a}$ were chosen arbitrarily, $e_\star \equiv e_1 \star e_2$. This proves that $\mathrm{RA}^+\{\cap,\cup\} \subseteq \mathrm{RA}^+\{-\}$.

We are left to show that the containment is strict. This follows from the fact that every query q expressible in $\mathrm{RA}^+\{\cap,\cup\}$ is monotonic, that is, $q(D) \subseteq q(D')$ for all databases D and D' such that $D \subseteq D'$. However, in $\mathrm{RA}^+\{-\}$, one can express queries that are not monotonic. Indeed, consider $e = R - S$ over unary relation names R and S, and let $D = \{R(a), S(b)\}$ and $D' = \{R(a), S(a), S(b)\}$; then, $D \subseteq D'$ but $e(D) = \{R(a)\} \not\subseteq e(D') = \varnothing$. $\square$

Then, obviously, we immediately get the following.

Corollary 4. $\mathrm{RA}^+\{-,\varepsilon\}$ *and* RA *have the same expressive power.*

We conclude this section by showing that adding duplicate elimination to a fragment that does not already have it increases its expressive power.

Proposition 5. *$\mathcal{L} \subsetneq \mathcal{L}\{\varepsilon\}$ for every fragment $\mathcal{L}$ of* RA *without duplicate elimination.*

The proof of Proposition 5 is based on the fact that, on databases with relations of even size, every $\mathrm{RA}^+\{-\}$ expression can only yield a relation of even size, as the following lemma shows.

Lemma 5. *Let $\mathcal{L}$ be a fragment of bag relational algebra without duplicate elimination, let e be an $\mathrm{RA}^+\{-\}$ expression, and let D be a database such that, for every relation name R, the number of occurrences of every tuple in R^D is even. Then, the number of occurrences of every tuple in $e(D)$ is even.*

Proof. Let $\bar{a}$ be a tuple that occurs k times in $e(D)$, and let $k > 0$ (for $k = 0$ we have nothing to show). By induction on the structure of e, we will show that k is even.

Base case: e is a relation name R. Then, $k = \#(r, R^D)$ is even by assumption.

Inductive step:

- $e = \pi_\alpha(e')$. Then, k is the sum of the number of occurrences of all tuples $\bar{a}' \in e'(D)$ such that $\pi_\alpha(\bar{a}') = \bar{a}$. The number of occurrences of every tuple in $e'(D)$ is even by the induction hypothesis, hence k is even as well.

- $e = \sigma_{i=j}(e')$. Then $k = \#(\bar{a}, e', D)$, which is even by the induction hypothesis.
- $e = e_1 \times e_2$. Then, there are tuples $\bar{a}_1 \in e_1(D)$ and $\bar{a}_2 \in e_2(D)$ such that $\bar{a} = \bar{a}_1\bar{a}_2$ and $k = m \cdot n$, where $m = \#(\bar{a}_1, e_1, D)$ and $n = \#(\bar{a}_2, e_2, D)$. As $k > 0$, both m and n are positive and, by the induction hypothesis, even. Hence, k is even as well.
- $e = e_1 \uplus e_2$. Then, $k = m + n$ where $m = \#(\bar{a}, e_1, D)$ and $n = \#(\bar{a}, e_2, D)$. Since $k > 0$, at least one of m and n must be positive. Assume w.l.o.g. that $m > 0$; then, m is even by the induction hypothesis. If $n = 0$, then $k = m$ is even; otherwise, n is even by the induction hypothesis as well, and therefore k is the sum of two positive even numbers, which is obviously even.
- $e = e_1 - e_2$. Then, $k = m \dot{-} n$ where $m = \#(\bar{a}, e_1, D)$ and $n = \#(\bar{a}, e_2, D)$. As $k > 0$, we have that $m > n$; in turn, m is positive and, by the induction hypothesis, even. If $n = 0$, then $k = m$ is even; otherwise, n is even as well by the induction hypothesis, so $k = m - n$ is obviously even. □

Proof of Proposition 5. Let $\mathcal{L}$ be a fragment of bag relational algebra without duplicate elimination. Clearly, $\mathcal{L}\{\varepsilon\}$ is at least as expressive as $\mathcal{L}$, so we only need to show that there exists a query that is expressible in the former but not in latter. To this end, consider a schema consisting of a nullary relation name R, and take any database D such that $|R^D|$ is a positive even number. Then, for every $\mathcal{L}$ expression over R, we have that $|e(D)|$ is even by Lemma 5. Hence, the $\mathcal{L}\{\varepsilon\}$ expression $\varepsilon(R)$ is not expressible in $\mathcal{L}$, since $|\varepsilon(R^D)| = 1$. □

4. Certain and possible answers under bag semantics

Dealing with incomplete information is a recurring topic in many different areas of logic and computer science. In database theory, the main way to deal with the lack of information is via *incomplete databases*. Intuitively, an incomplete database D is a compact representation of a possibly infinite collection $[\![D]\!]$ of complete databases, which define the *semantics* of D.

In this paper, we use incomplete databases with *marked* (or *labeled*) *nulls*. This model of incompleteness is very common in the database literature [17,18] and naturally occurs in many different scenarios, e.g., in data exchange and integration (cf. [24–26]). In this model databases are populated by *constants* and *nulls*, coming from two disjoint and countably infinite sets denoted by Const and Null, respectively. More formally, a k-ary relation is a finite bag of k-ary tuples over $\mathsf{Const} \cup \mathsf{Null}$. A database D then maps each k-ary relation symbol R in the schema to a k-ary bag relation R^D. Given a database $D = \{R_1^D, \ldots, R_n^D\}$, we write $\mathsf{Const}(D)$ and $\mathsf{Null}(D)$ for the set of constants and nulls occurring in the R_i^Ds, respectively. The *active domain* of D is the set $\mathsf{Const}(D) \cup \mathsf{Null}(D)$, denoted by $\mathsf{adom}(D)$. We say that D is *complete* if $\mathsf{Null}(D) = \varnothing$.

The semantics $[\![D]\!]$ of a database D is defined by means of valuations. A valuation v is a map $v\colon \mathsf{Null}(D) \to \mathsf{Const}$, and the result of applying v to D is the complete database $v(D)$ obtained by replacing each null $\bot \in \mathsf{Null}(D)$ with $v(\bot)$. Observe that applying v to each relation R^D in D preserves multiplicities, i.e., for each relation name R in the schema and each tuple $\bar{c} \in R^{v(D)}$ the following equality holds:

$$\#\big(\bar{c}, R^{v(D)}\big) = \sum_{\bar{a}:v(\bar{a})=\bar{c}} \#\big(\bar{a}, R^D\big).$$

The set $[\![D]\!]$ is defined as $[\![D]\!] = \{v(D) \mid v \text{ is a valuation}\}$.

When relations are sets, the standard way to answer a query q on an incomplete database D is to compute *certain answers*, i.e., tuples that are in $q(D')$ for every $D' \in [\![D]\!]$, and *possible answers*, i.e., tuples that are in $q(D')$ for some $D' \in [\![D]\!]$. When relations are bags, however, one must also take multiplicities into account. In what follows, this is done by computing the minimum and maximum number of occurrences of a tuple in the answers across all databases in $[\![D]\!]$ (cf. [21]). Let D be a database, let q be a relational algebra expression of arity n, and let $\bar{a} \in \mathsf{Const}(D)^n$ be a tuple of constants, we define $\min(\bar{a}, q, D)$ and $\max(\bar{a}, q, D)$ as follows:

$$\min(\bar{a}, q, D) = \min_{D' \in [\![D]\!]} \#\big(\bar{a}, q(D')\big)\,; \tag{1a}$$

$$\max(\bar{a}, q, D) = \max_{D' \in [\![D]\!]} \#\big(\bar{a}, q(D')\big)\,. \tag{1b}$$

Intuitively, $\min(\bar{a}, q, D)$ and $\max(\bar{a}, q, D)$ are extensions of certain and possible answers to bag databases. Indeed, $\min(\bar{a}, q, D) \geq 1$ if and only if $\bar{a}$ is in $q(D')$ for every $D' \in [\![D]\!]$ (and thus it is a certain answer), and $\max(\bar{a}, q, D) \geq 1$ if $\bar{a}$ is in $q(D')$ for some $D' \in [\![D]\!]$ (and thus it is a possible answer).

Thus, from now on we assume $\min(\bar{a}, q, D)$ and $\max(\bar{a}, q, D)$ as our standard notion of query answers and study their complexity. More specifically, we will focus on *data complexity*, that is, computing $\min(\bar{a}, q, D)$ and $\max(\bar{a}, q, D)$ for a fixed query q. Depending on the type of comparison we use, several decision problems arise from the computation of min and max. These decision problems are defined as follows.

PROBLEM:	$\mathsf{MIN}^\theta[q]$, for $\theta \in \{>, =, <\}$ and a query q of arity n.
INPUTS:	an incomplete database D, a tuple $\bar{a} \in \mathsf{Const}(D)^n$, a non-negative integer k.
QUESTION:	is $\min(\bar{a}, q, D)\ \theta\ k$?

PROBLEM:	$\mathsf{MAX}^\theta[q]$, for $\theta \in \{>, =, <\}$ and a query q of arity n.
INPUTS:	an incomplete database D, a tuple $\bar{a} \in \mathsf{Const}(D)^n$, a non-negative integer k.
QUESTION:	is $\max(\bar{a}, q, D)\ \theta\ k$?

Whether the number k is represented in unary or binary form does not matter: the results will be the same regardless. All tractability results are shown assuming binary representation, and all matching hardness results will be proved for the case when k is represented in unary. The choice of inequalities, i.e., $<$ vs $\leq$ or $>$ vs $\geq$, is not important: since k is an integer, $\leq k$ is the same condition as $< k + 1$.

As for the case of certain and possible answers, the complexity of the above problems depends on the fragment of RA in which the query q is expressed. While for some of these fragments the problems can be proved to be intractable, there are fragments of RA for which computing $\min(\bar{a}, q, D)$ is tractable and can actually be done via *naive evaluation*. The naive evaluation of a query q of arity n on a bag database D is defined as the bag obtained by assuming that each null value in $\mathsf{Null}(D)$ is a distinct constant and evaluating q directly over D. In what follows, we will denote by $\mathsf{naive}(\bar{a}, q, D)$ the number of occurrences of a tuple $\bar{a} \in \mathsf{Const}(D)^n$ in the result of the naive evaluation of an RA query q on an incomplete database D. It is well known that $\mathsf{naive}(\bar{a}, q, D)$ can be computed in DLOGSPACE in data complexity [5,6].

Example 2. On a database with relations $R = \{a, \bot_1, \bot_1\}$ and $S = \{\bot_1, \bot_2\}$, the naive evaluation of $\sigma_{1=2}(R \times S)$, $\varepsilon(R)$ and $R \cap S$ gives $\{(\bot_1, \bot_1), (\bot_1, \bot_1)\}$, $\{a, \bot_1\}$ and $\{\bot_1\}$, respectively.

5. Complexity of certain and possible answers

We now turn our attention to the complexity of evaluating bag relational algebra expressions on incomplete databases, that is, solving the problems $\mathsf{MIN}^{\theta}[q]$ and $\mathsf{MAX}^{\theta}[q]$ for queries in various fragments of RA. As already explained, these problems are natural bag analogs of the notions of certainty and possibility over set databases.

In Section 5.1, we first provide upper and lower bounds for full RA. When the relation θ is $<$ or $>$, the results are easily derivable from the results for the set case in [19]; we complement them with the exact complexity for equality.

After introducing our technical framework in Section 5.2, we then focus – in Section 5.3 – on the problem $\mathsf{MIN}^{\theta}[q]$, proving the exact tractability boundary shown in Fig. 1. We start by showing that in all fragments up to $\mathrm{RA}^{+}\{\cap\}$, the value of min can be computed by naive evaluation of queries, which extends a result in [21]. We then show that beyond this fragment the problem becomes intractable: NP-complete for $<$, coNP-complete for $>$, and DP-complete for $=$. In particular, we show that the problem is intractable for all fragments containing $\mathrm{SPC}\{\cup\}$, and all fragments containing $\mathrm{SPC}\{\varepsilon\}$.

Next, in Section 5.4, we look at $\mathsf{MAX}^{\theta}[q]$. It was shown in [21] that for $>$ the problem is NP-complete, even for very simple queries. Here, we complete the picture and settle the case for $=$, even when q is a query that simply returns a relation from the database.

Finally, in Section 5.5, we discuss what happens with more complex selection conditions in queries.

5.1. Upper and lower bounds for full RA

Before delving into the complexity of the different fragments of RA, we briefly look at the complexity of evaluating general expressions. First, observe that RA queries are *generic*, i.e., invariant under permutations of the domain. In the bag case, this is stated as follows.

Proposition 6. *Let D and D' be complete databases, let ρ be a bijection between $\mathsf{adom}(D)$ and $\mathsf{adom}(D')$ such that $D' = \rho(D)$, and let $q \in \mathrm{RA}$ be a query of arity n. Then, $\#(\bar{a}, q, D) = \#(\rho\bar{a}, q, D')$ for every tuple $\bar{a} \in \mathsf{adom}(D)^n$.*

Proof. By induction on the structure of q.

Base case:

- $q = R$. Trivial.

Inductive step:

- $q = \sigma_{\gamma}(q')$. Suppose such ρ exists. Since ρ is a bijection, $\bar{a}$ satisfies γ if and only if $\rho\bar{a}$ does. If $\bar{a}$ does not satisfy γ then $\#(\bar{a}, q, D) = \#(\rho\bar{a}, q, D') = 0$. Suppose now that $\bar{a}$ satisfies γ. From the inductive hypothesis, we can conclude that $\#(\bar{a}, q', D) = \#(\rho\bar{a}, q', D')$ hence $\#(\bar{a}, q, D) = \#(\rho\bar{a}, q, D')$.
- $q = \pi_{\alpha}(q')$. Assume that q' has arity m and suppose that there exists a bijection ρ with the properties defined above. Let $\bar{b} \in \mathsf{adom}^m(D)$ be a generic tuple that is equal to $\bar{a}$ on the attributes in α, then $\#(\bar{a}, q, D) = \sum_{\bar{b}:\pi_{\alpha}(\bar{b})=\bar{a}} \#(\bar{b}, q', D)$.

  We now apply the inductive hypothesis and obtain that $\#(\bar{b}, q', D) = \#(\rho\bar{b}, q', D')$ for each such tuple. Clearly $\rho\bar{b}$ is equal to $\rho\bar{a}$ on the attributes in α, proving $\#(\bar{a}, q, D) \leq \#(\rho\bar{a}, q, D')$. To prove that this is actually an equality, we simply notice that ρ is a bijection, hence for each $\bar{u}' \in \mathsf{adom}^m(D')$ there exists a tuple $\bar{u} \in \mathsf{adom}^m(D)$ such that $\bar{u}' = \rho\bar{u}$ and $\bar{u}$ is equal to $\bar{a}$ on α if and only if $\bar{u}'$ is equal to $\rho\bar{a}$ on α.
- $q = q' \times q''$. Assume that q' has arity m', q'' has arity m'', and suppose that there exists a bijection ρ with the properties defined above. Let $\bar{a}_1$ and $\bar{a}_2$ be two tuples of arity m' and m'' respectively, such that $\bar{a} = \bar{a}_1 \circ \bar{a}_2$, then $\#(\bar{a}, q, D) = \#(\bar{a}_1, q', D) \cdot \#(\bar{a}_2, q'', D)$. From the inductive hypothesis, $\#(\bar{a}_1, q', D) = \#(\rho\bar{a}_1, q', D')$ and $\#(\bar{a}_2, q'', D) = \#(\rho\bar{a}_2, q'', D')$. The claim follows from $\rho\bar{a} = \rho\bar{a}_1 \circ \rho\bar{a}_2$.
- $q = \varepsilon(q')$. By the inductive hypothesis, we have that $\#(\bar{a}, q', D) = \#(\rho\bar{a}, q', D')$. Then the claim follows from the definition of ε.
- $q = q' \star q''$, for $\star \in \{\cap, \uplus, \cup, -\}$. By the inductive hypothesis, we have that $\#(\bar{a}, q', D) = \#(\rho\bar{a}, q', D')$ and $\#(\bar{a}, q'', D) = \#(\rho\bar{a}, q'', D')$. Then, the claim follows from the definition of $\cap$, $\uplus$, $\cup$, and $-$. $\square$

Intuitively, the above tells us that we need to take into account only finitely many valuations in order to compute the values in (1a) and (1b). Upper bounds of NP, coNP, and DP follow straightforwardly.

Proposition 7. *Let q be an expression in RA. Then:*

- $\mathsf{MIN}^{<}[q]$ *and* $\mathsf{MAX}^{>}[q]$ *are in* NP*;*
- $\mathsf{MIN}^{>}[q]$ *and* $\mathsf{MAX}^{<}[q]$ *are in* coNP*;*
- $\mathsf{MIN}^{=}[q]$ *and* $\mathsf{MAX}^{=}[q]$ *are in* DP.

Proof. Let D be an incomplete bag database, let $\bar{a}$ be a tuple in $\mathsf{adom}^n(D)$, and let k be a positive integer. Assume a set $\mathcal{C} \subseteq \mathsf{Const}$ such that $\mathcal{C}$ is disjoint with $\mathsf{adom}(D)$ and the cardinality of $\mathcal{C}$ is the same as the cardinality of $\mathsf{Null}(D)$. Let $\mathcal{V}$ be the set of valuations for $\mathsf{Null}(D)$ defined as follows: $\mathcal{V} = \{v \mid range(v) \subseteq (\mathcal{C} \cup \mathsf{Const}(D))\}$. Clearly, for each $D' \in [\![D]\!]$ there exists a valuation $v \in \mathcal{V}$ and a bijection ρ from $\mathsf{adom}(vD)$ to $\mathsf{adom}(D')$ such that ρ is the identity over $\mathsf{Const}(D)$ and $\rho(vD) = D'$.

Using this observation, to check whether $\min(\bar{a}, q, D) < k$ we can simply guess a valuation $v' \in \mathcal{V}$ such that $\#(v'\bar{a}, q, v'D) < k$. If such valuation exists, $v'D$ is itself a witness of $\min(\bar{a}, q, D) < k$. If such valuation does not exist, due to Proposition 6 we can conclude that there exists no $v'D \in [\![D]\!]$ such that $\#(v'\bar{a}, q, v'D) < k$. A similar technique can be used to check whether $\max(\bar{a}, q, D) > k$.

From these considerations we can conclude what follows. In order to solve $\mathsf{MIN}^{<}[q]$ and $\mathsf{MAX}^{>}[q]$ for input D and k, one can simply guess a valuation v for $\mathsf{Null}(D)$ and check whether $\#(v\bar{a}, q, vD) < k$ and $\#(v\bar{a}, q, vD) > k$ respectively. To solve $\mathsf{MIN}^{>}[q]$, one can guess a valuation v for $\mathsf{Null}(D)$ such that $\#(v\bar{a}, q, vD) < k + 1$ and use it as a counterexample for $min(\bar{a}, q, D) > k$. Similarly, $\mathsf{MAX}^{<}[q]$ can be solved by guessing a valuation v for $\mathsf{Null}(D)$ such that $\#(v\bar{a}, q, vD) > k-1$ and use it as a counterexample for $max(\bar{a}, q, D) < k$. Finally, to solve $\mathsf{MIN}^{=}[q]$, one can simply solve an instance of $\mathsf{MIN}^{<}[q]$ with input D, a, and $k+1$, and an instance of $\mathsf{MIN}^{>}[q]$ with input D, a, and $k-1$. The same technique can be used to solve $\mathsf{MAX}^{=}[q]$.

To prove the claim, we observe the following. For a given valuation v, checking whether $\#(v\bar{a}, q, vD)\theta k$ can be done in DLogSpace in the size of D for $q \in \mathrm{RA}$ and $\theta \in \{<, >\}$. Moreover, valuations for $\mathsf{Null}(D)$ can be guessed using a number of bits that is polynomial in the size of D. $\square$

For general RA expressions, all of these problems are complete in their respective classes.

Proposition 8. *Let q be an expression in RA. Then:*

- $\mathsf{MIN}^{<}[q]$ *and* $\mathsf{MAX}^{>}[q]$ *are* NP*-hard;*
- $\mathsf{MIN}^{>}[q]$ *and* $\mathsf{MAX}^{<}[q]$ *are* coNP*-hard;*
- $\mathsf{MIN}^{=}[q]$ *and* $\mathsf{MAX}^{=}[q]$ *are* DP*-hard.*

Proof. The results for $<$ and $>$ follow directly from the fact that set databases can be simulated by bag databases. Hence, the hardness results presented in [19] apply. For $=$, we can show a reduction from a very well known DP-complete problem; see Theorem 2 for the case of $\mathsf{MIN}^{=}[q]$ and Theorem 4 for the case of $\mathsf{MAX}^{=}[q]$. □

Despite these high bounds, one may expect that some fragments of RA will behave better; this is what we investigate next.

5.2. Technical tools

To prove our results, we will use reductions from two well-known decision problems: *satisfiability* (SAT) and *satisfiability–unsatisfiability* (SAT-UNSAT) of propositional formulae in conjunctive normal form (CNF). We use the following standard terminology:

- a *literal* is a propositional variable or its negation;
- a *clause* is a (non-empty) disjunction of literals;
- a k-CNF formula is a conjunction of clauses, where each clause consists of exactly k distinct literals.

We also assume that complementary literals (i.e., a propositional variable and its negation) do not appear in the same clause; this is without loss of generality, as such clauses can be removed without compromising equivalence to the original formula. Thus, we consider k-CNF formulae where each clause mentions precisely k distinct variables.

We write $\mathsf{Var}(\varphi)$ and $|\varphi|$ for the set of variables and the number of clauses of φ, respectively. A *truth assignment* for φ is a function from $\mathsf{Var}(\varphi)$ to either $\mathbf{t}$ (true) or $\mathbf{f}$ (false). The formula φ is *satisfiable* if there exists a truth assignment that makes all of its clauses true.

Example 3. The following 3-CNF formula

$$\varphi = \underbrace{(\neg x_1 \vee x_2 \vee x_4)}_{C_1} \wedge \underbrace{(x_1 \vee \neg x_2 \vee \neg x_3)}_{C_2} \tag{2}$$

has two clauses, C_1 and C_2, each consisting of three literals. We have $\mathsf{Var}(\varphi) = \{x_1, x_2, x_3, x_4\}$ and $|\varphi| = 2$. The formula is satisfiable because there is a truth assignment that makes both C_1 and C_2 true; e.g., the one mapping every variable in $\mathsf{Var}(\varphi)$ to $\mathbf{t}$.

We let k-SAT refer to the problem of checking whether a given k-CNF formula is satisfiable; this problem is well known to be NP-complete for $k \geq 3$. Given two k-CNF formulae φ and ψ, SAT-UNSAT is the problem of checking whether φ is satisfiable while ψ is not. For $k \geq 3$, this is known to be DP-complete [22].

In our reductions, we will use CNF formulae encoded as relations. To this end, we assume that formulae are built from a countably infinite and linearly ordered set Var of propositional variables. Assuming a bijection ρ from Var to Null, for each variable $x \in \mathsf{Var}$ we define the pairs:

$$\bar{u}^{\mathbf{t}}_x = \big(0, \rho(x)\big)\,, \tag{3a}$$

$$\bar{u}^{\mathbf{f}}_x = \big(\rho(x), 1\big)\,. \tag{3b}$$

Observe that $\bar{u}^{\mathbf{t}}_x$ and $\bar{u}^{\mathbf{f}}_x$ do not unify: there exists no valuation v of nulls such that $v(\bar{u}^{\mathbf{t}}_x) = v(\bar{u}^{\mathbf{f}}_x) = (0, 1)$. We denote concatenation of tuples, in particular of pairs of the above form, by juxtaposition; for example, $\bar{u}^{\mathbf{t}}_x \bar{u}^{\mathbf{f}}_y \bar{u}^{\mathbf{f}}_z$ is the tuple $\big(0, \rho(x), \rho(y), 1, \rho(z), 1\big)$.

Given a k-CNF formula φ, with each clause C of φ we associate a relation $\mathsf{enc}(C)$ of arity $2k$ constructed as follows:

(1) For each truth assignment τ that makes C true, we add a single occurrence of the tuple $\bar{u}_1 \cdots \bar{u}_k$, where each $\bar{u}_i$ is equal to $\bar{u}^{\mathbf{t}}_{x_i}$ if $\tau(x_i) = \mathbf{t}$ and to $\bar{u}^{\mathbf{f}}_{x_j}$ if $\tau(x_i) = \mathbf{f}$, and the variables are considered in the order we assumed on Var.
(2) We then remove duplicates from the resulting relation, which amounts to considering only truth assignments that are unique with respect to what they assign to the variables mentioned in the clause C we are encoding.

If the clauses of φ are $C_1, \ldots, C_n$, we define $R_\varphi = \biguplus_{j=1}^{n} \mathsf{enc}(C_j)$. Note that the number of tuples in each $\mathsf{enc}(C_j)$ is exactly $2^k - 1$ and so the total number of tuples in R_φ is $|\varphi| \cdot (2^k - 1)$. Thus, when k is considered fixed, the size of R_φ is polynomial w.r.t. the size of φ.

Example 4. The encoding R_φ of the 3-CNF formula (2), assuming the ordering $x_1 < x_2 < x_3 < x_4$ and the mapping $\rho(x_i) = \bot_i$ for $i \in \{1, \ldots, 4\}$, is as follows:

$$\underbrace{\begin{array}{|c|c|c|c|c|c|}\hline 0 & \bot_1 & 0 & \bot_2 & 0 & \bot_4 \\ \hline 0 & \bot_1 & 0 & \bot_2 & \bot_4 & 1 \\ \hline 0 & \bot_1 & \bot_2 & 1 & 0 & \bot_4 \\ \hline 0 & \bot_1 & \bot_2 & 1 & \bot_4 & 1 \\ \hline \bot_1 & 1 & 0 & \bot_2 & \bot_4 & 1 \\ \hline \bot_1 & 1 & \bot_2 & 1 & 0 & \bot_4 \\ \hline \bot_1 & 1 & \bot_2 & 1 & \bot_4 & 1 \\ \hline \end{array}}_{\mathsf{enc}(C_1)}$$

$$\biguplus \underbrace{\begin{array}{|c|c|c|c|c|c|}\hline 0 & \bot_1 & 0 & \bot_2 & 0 & \bot_3 \\ \hline 0 & \bot_1 & 0 & \bot_2 & \bot_3 & 1 \\ \hline 0 & \bot_1 & \bot_2 & 1 & 0 & \bot_3 \\ \hline \bot_1 & 1 & 0 & \bot_2 & 0 & \bot_3 \\ \hline \bot_1 & 1 & 0 & \bot_2 & \bot_3 & 1 \\ \hline \bot_1 & 1 & \bot_2 & 1 & 0 & \bot_3 \\ \hline \bot_1 & 1 & \bot_2 & 1 & \bot_3 & 1 \\ \hline \end{array}}_{\mathsf{enc}(C_2)}$$

The relations R_φ enjoy the following properties that will be central in our proofs.

Lemma 6. *For every k-CNF formula φ, all of the following hold:*

(a) *Let R be the relation that encodes a clause of φ. Then, for every valuation v of nulls, $\#\big((0, 1)^k, v(R)\big) \leq 1$.*
(b) *Let v be a valuation with range in $\{0, 1\}$, and let τ be the truth assignment such that, for each $x \in \mathsf{Var}(\varphi)$, $\tau(x) = \mathbf{t}$ iff $v\big(\rho(x)\big) = 1$. Then, τ makes a clause C of φ true if and only if $\#\big((0, 1)^k, v(R)\big) = 1$, where $R = \mathsf{enc}(C)$.*
(c) *There is a truth assignment satisfying exactly m clauses of φ if and only if there is a valuation v with range in $\{0, 1\}$ such that $\#\big((0, 1)^k, v(R_\varphi)\big) = m$.*
(d) *For every valuation v, there is a valuation v' with range in $\{0, 1\}$ such that $\#\big((0, 1)^k, v'(R_\varphi)\big) \geq \#\big((0, 1)^k, v(R_\varphi)\big)$.*

Proof.

(a) By construction, all tuples in R are of the form

$$\bar{u}^{t_1}_{x_1} \cdots \bar{u}^{t_k}_{x_k}\,, \quad \text{with } x_1, \ldots, x_k \in \mathsf{Var}(\varphi) \text{ and } t_1, \ldots, t_k \in \{\mathbf{t}, \mathbf{f}\}\,. \tag{4}$$

Moreover, for any two distinct tuples there exists i such that one of them is of the form $\bar{u}^{t_1}_{x_1} \cdots \bar{u}^{\mathbf{t}}_{x_i} \cdots \bar{u}^{t_k}_{x_k}$ while the other has the form $\bar{u}^{t_1}_{x_1} \cdots \bar{u}^{\mathbf{f}}_{x_i} \cdots \bar{u}^{t_k}_{x_k}$. Let v be a valuation; clearly, it cannot be that $v(\bar{u}^{\mathbf{t}}_{x_i}) = v(\bar{u}^{\mathbf{f}}_{x_i}) = (0, 1)$, so no two distinct tuples in R both unify with $(0, 1)^k$. Thus, $\#\big((0, 1)^k, v(R)\big) \leq 1$.

(b) By construction, τ makes C true if and only if there is a tuple $\bar{a} \in R$ of the form (4) such that $t_i = \tau(x_i)$ for each $i \in \{1, \dots, k\}$. In turn, by definition of τ and as v has range in $\{0, 1\}$, this is the case iff $v\big(\bar{u}_{x_i}^{\tau(x_i)}\big) = (0, 1)$ for each $i \in \{1, \dots, k\}$, and so iff $v(\bar{a}) = (0, 1)^k$. By Lemma 6(a), this is the case iff $\#\big((0, 1)^k, v(R)\big) = 1$.

(c) **"if".** Let v be a valuation with range in $\{0, 1\}$ such that $\#\big((0, 1)^k, v(R_\varphi)\big) = m$. Then, by Lemma 6(b), there are exactly m distinct clauses $C_1, \dots, C_m$ of φ such that, for each $i \in \{1, \dots, k\}$, we have $\#\big((0, 1)^k, v(R_i)\big) = 1$, where $R_i \subseteq R_\varphi$ is the relation encoding C_i. Consider the truth assignment τ such that, for each $x \in \mathsf{Var}(\varphi)$, $\tau(x) = \mathbf{t}$ iff $v\big(\rho(x)\big)$. Then, by Lemma 6(b), τ satisfies $C_1, \dots, C_m$ but no other clause of φ. To see this, let C be a clause of φ distinct from $C_1, \dots, C_m$ and encoded by $R \subseteq R_\varphi$, and suppose that τ makes C true. But then $\#\big((0, 1)^k, v(R)\big) = 1$ and so $\#\big((0, 1)^k, v(R_\varphi)\big) > m$, against the initial assumption.
"only if". Let $\varphi = \{C_1, \dots, C_m, C_{m+1}, \dots, C_n\}$, and let R_i be the relation that encodes clause C_i, for each $i \in \{1, \dots, n\}$. Let τ be a truth assignment that τ makes $C_1, \dots, C_m$ true and $C_{m+1}, \dots, C_n$ false. Consider the valuation v such that, for each $\bot \in \mathsf{Null}$,

$$v(\bot) = \begin{cases} 1 & \text{if } \rho^{-1}(\bot) = x \in \mathsf{Var}(\varphi) \text{ and } \tau(x) = \mathbf{t}\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

Then, due to Lemma 6(a) and Lemma 6(b), for $i \in \{1, \dots, n\}$ we have that $\#\big((0, 1)^k, v(R_i)\big)$ is 1 if $i \leq m$, and 0 otherwise. Thus, $\#\big((0, 1)^k, v(R_\varphi)\big) = \sum_{i=1}^{n} \#\big((0, 1)^k, v(R_i)\big) = m$.

(d) For any given valuation v, we can always define the valuation v' such that, for every $\bot \in \mathsf{Null}$, $v'(\bot) = v(\bot)$ if $v(\bot) \in \{0, 1\}$, and $v'(\bot) = 0$ otherwise. As v' agrees with v on the values assigned to the nulls in the preimage of $\{0, 1\}$ under v, we have that $v(\bar{a}) = v'(\bar{a})$ whenever $v(\bar{a})$ unifies with $(0, 1)^k$. Thus, $\#\big((0, 1)^k, v'(R_\varphi)\big) \geq \#\big((0, 1)^k, v(R_\varphi)\big)$. □

We also define a variant of the encoding described above, by associating each clause with an identifier in the relation that encodes the formula. To make this formal, let φ be a k-CNF formula, and let ι be an injection from the clauses of φ to Const; for each clause C of φ, we then define

$$\mathsf{enc}(C, \iota) = \big(B \times \mathsf{enc}(C)\big) \uplus B'$$

where B and B' are tables consisting of a single occurrence of the tuples $\big(\iota(C)\big)$ and $\big(\iota(C)(0, 1)^k\big)$, respectively. Intuitively, we compute the relation $\mathsf{enc}(C)$, add one occurrence of $(0, 1)^k$, and then prefix every tuple with the value $\iota(C)$. Note that, for distinct clauses C and C', no tuple of $\mathsf{enc}(C, \iota)$ unifies with any of the tuples in $\mathsf{enc}(C', \iota)$. For $\varphi = C_1 \wedge \dots \wedge C_n$, we let $\mathsf{enc}(\varphi, \iota) = \biguplus_{i=1}^{n} \mathsf{enc}(C_i, \iota)$, whose size is exactly $2^k \cdot |\varphi|$.

Example 5. Consider the encoding of the 3-CNF formula (2) in Example 4. If $\iota(C_1) = a \in \mathsf{Const}$, then $\mathsf{enc}(C_1, \iota)$ is the following table:

a	0	$\bot_1$	0	$\bot_2$	0	$\bot_4$
a	0	$\bot_1$	0	$\bot_2$	$\bot_4$	1
a	0	$\bot_1$	$\bot_2$	1	0	$\bot_4$
a	0	$\bot_1$	$\bot_2$	1	$\bot_4$	1
a	$\bot_1$	1	0	$\bot_2$	$\bot_4$	1
a	$\bot_1$	1	$\bot_2$	1	0	$\bot_4$
a	$\bot_1$	1	$\bot_2$	1	$\bot_4$	1
a	0	1	0	1	0	1

Our second technical tool is a $(k+1)$-CNF formula derived from two k-CNF formulae, to check whether one is satisfiable while the other is not. To this end, for a k-CNF formula φ and a literal $\ell \in \{x, \neg x\}$ with $x \notin \mathsf{Var}(\varphi)$, we denote by φ^ℓ the $(k+1)$-CNF formula obtained by adding ℓ to each clause of φ. Moreover, for a (finite) set Σ of propositional variables, we let $\varphi_{\mathbf{f}}(\Sigma)$ denote the $|\Sigma|$-CNF formula consisting of all maxterms over Σ,[1] except $\bigvee_{x \in \Sigma} x$. Observe that $\varphi_{\mathbf{f}}(\Sigma)$ has size $2^{|\Sigma|} - 1$, and it is satisfied only by the truth assignment that maps all of the variables in Σ to $\mathbf{f}$.

Then, given two k-CNF formulae f and g, we define the $(k+1)$-CNF formula

$$\begin{aligned} h(f, g) \;=\; & f^x \wedge g^{\neg x} \wedge g^{\neg y} \wedge \varphi_{\mathbf{f}}(\{z_1, \dots, z_{k+1}\}) \\ & \wedge (x \vee z_2 \vee \dots \vee z_{k+1}) \\ & \wedge (\neg x \vee y \vee z_3 \vee \dots \vee z_{k+1})\,, \end{aligned} \tag{5}$$

where none of the variables $x, y, z_1, \dots, z_{k+1}$ appears in $\mathsf{Var}(f) \cup \mathsf{Var}(g)$. This formula has the following property:

Lemma 7. *Let f and g be k-CNF formulae. Then, f is satisfiable and g is unsatisfiable if and only if the maximum number of clauses of $h(f, g)$ that can be satisfied by a single truth assignment is exactly $|h(f, g)| - 1$.*

Proof. We denote by $\|\varphi\|$ the maximum number of clauses that can be satisfied in φ by a single truth assignment, while $|\varphi|$ denotes the number of clauses in φ. To avoid the clutter, we let $\varphi_{\mathbf{f}}$ refer to $\varphi_{\mathbf{f}}(\{z_1, \dots, z_{k+1}\})$ in (5), and h refer to $h(f, g)$ itself; note that $|h| = |f| + 2 \cdot |g| + 2^{k+1} + 1$.

"if". Assume that $\|h\| = |h| - 1$. We will prove that f is satisfiable while g is unsatisfiable by showing that all other possible combinations lead to a contradiction.

Let us start by supposing that there exists a satisfying truth assignment τ for g. Then, the extension of τ that assigns $\mathbf{t}$ to all the variables in $\mathsf{Var}(f) \cup \{x, y\}$ and $\mathbf{f}$ to those in $\mathsf{Var}(\varphi_{\mathbf{f}})$ satisfies h, in contradiction of the initial assumption.

We now consider the case when both f and g are unsatisfiable. To examine $\|h\|$, let h_1 denote $f^x \wedge g^{\neg x} \wedge g^{\neg y}$, and h_2 consist of all clauses of h that are not in h_1, so that $h = h_1 \wedge h_2$. Note that $\|h\| \leq \|h_1\| + \|h_2\|$; moreover, h_2 depends on the variables in $\mathsf{Var}(\varphi_{\mathbf{f}})$ while h_1 does not. We denote $\|h_2\|$ by M when all variables in $\mathsf{Var}(\varphi_{\mathbf{f}})$ are mapped to $\mathbf{f}$, and by N otherwise. Then, we examine $\|h_1\|$, M, and N for each of the four possible truth assignments for x and y:

Assignment		$\|h_1\|$	M	N
$x \mapsto \mathbf{f}$	$y \mapsto \mathbf{f}$	$= \|f\| + \|g\| + \|g\|$	$= \|\varphi_{\mathbf{f}}\| + 1$	$< \|\varphi_{\mathbf{f}}\| + 2$
$x \mapsto \mathbf{f}$	$y \mapsto \mathbf{t}$	$\leq \|f\| + \|g\| + \|g\|$	$= \|\varphi_{\mathbf{f}}\| + 1$	$< \|\varphi_{\mathbf{f}}\| + 2$
$x \mapsto \mathbf{t}$	$y \mapsto \mathbf{f}$	$= \|f\| + \|g\| + \|g\|$	$= \|\varphi_{\mathbf{f}}\| + 1$	$< \|\varphi_{\mathbf{f}}\| + 2$
$x \mapsto \mathbf{t}$	$y \mapsto \mathbf{t}$	$= \|f\| + \|g\| + \|g\|$	$= \|\varphi_{\mathbf{f}}\| + 2$	$< \|\varphi_{\mathbf{f}}\| + 2$

As f and g are unsatisfiable, $\|f\| < |f|$ and $\|g\| < |g|$. Therefore, in all of the above cases, both $\|h_1\| + M$ and $\|h_1\| + N$ are strictly less than $|f| + 2 \cdot |g| + |\varphi_{\mathbf{f}}| + 1$, and so $\|h\| < |h| - 1$, in contradiction of the initial assumption.

"only if". Let τ be a satisfying assignment for f, and assume that g is unsatisfiable. The extension of τ that maps all variables in $\mathsf{Var}(g) \cup \{x, y\} \cup \mathsf{Var}(\varphi_{\mathbf{f}})$ to $\mathbf{f}$ clearly satisfies all clauses of h but one, namely $(x \vee z_2 \vee \dots \vee z_{k+1})$. Now, suppose there exists a truth assignment τ' satisfying all the clauses of h. Then, τ' satisfies $g^{\neg y}$ and, since g is unsatisfiable, $\tau'(y) = \mathbf{f}$. Moreover, τ' also satisfies $\varphi_{\mathbf{f}}$ and thus $\tau'(z_i) = \mathbf{f}$ for every $i \in \{1, \dots, k+1\}$. In turn, to make the clause $(\neg x \vee y \vee z_3 \vee \dots \vee z_{k+1})$ true, we must have $\tau'(x) = \mathbf{f}$. Therefore, the clause $(x \vee z_2 \vee \dots \vee z_{k+1})$ of h is not satisfied by τ', which is a contradiction. □

[1] That is, all disjunctions where each variable in Σ appears exactly once, in either positive or negated form.

The following result is a critical lemma that will allow us to show the hardness of min problems by exhibiting a polynomial-time procedure that constructs a database for every CNF formula (based on the encodings given above), and query with specific properties.

Lemma 8. *Let q be an* RA *query of arity* 0.[2] *Let* dbenc *be a procedure that for each k-*CNF *formula φ builds, in polynomial time w.r.t. the size of φ, a database D_φ such that the following properties hold:*

(P1) *There exists a polynomially computable number χ, dependent only on k and $|\varphi|$, for which: there is a truth assignment that satisfies exactly m clauses of φ if and only if there is a valuation v with range in $\{0, 1\}$ such that $\#\big((), q, v(D_\varphi)\big) = \chi - m$.*

(P2) *For every valuation v, there exists a valuation v' with range in $\{0, 1\}$ such that $\#\big((), q, v(D_\varphi)\big) \geq \#\big((), q, v'(D_\varphi)\big)$.*

Then, (a) $\mathsf{MIN}^{<}[q]$ *is* NP-*hard,* (b) $\mathsf{MIN}^{>}[q]$ *is* coNP-*hard, and* (c) $\mathsf{MIN}^{=}[q]$ *is* DP-*hard.*

Proof. Observe that (P2) directly implies the following:

(P3) $\min\big((), q, D_\varphi\big) = \min \big\{ \#\big((), q, v(D_\varphi)\big) \mid v$ is a valuation with range in $\{0, 1\} \big\}$.

(a) Let $k \geq 3$. We show that dbenc provides a reduction from k-SAT to $\mathsf{MIN}^{<}[q]$. To this end, let φ be a k-CNF formula, and let $D_\varphi = \mathsf{dbenc}(\varphi)$.
By (P3), $\min\big((), q, D_\varphi\big) = \chi - |\varphi|$ iff there exists a valuation v with range in $\{0, 1\}$ such that $\#\big((), q, v(D_\varphi)\big) = \chi - |\varphi|$. In turn, by (P1), this is the case iff there is a truth assignment satisfying $|\varphi|$ clauses of φ. Therefore, φ is satisfiable iff D_φ, $()$ and $\chi - |\varphi|$ form a positive instance of $\mathsf{MIN}^{<}[q]$.

(b) Since $\mathsf{MIN}^{<}[q]$ is the complement of $\mathsf{MIN}^{>}[q]$, its complexity follows directly from the previous point.

(c) We show that dbenc and (5) together give a reduction from SAT-UNSAT to $\mathsf{MIN}^{<}[q]$. To this end, let f and g be two k-CNF formulae with $k \geq 3$, let φ denote the formula $h(f, g)$ defined as in (5), and let $D_\varphi = \mathsf{dbenc}(\varphi)$.
By Lemma 7, f is satisfiable and g is unsatisfiable iff the maximum number of clauses of φ that can be satisfied by the same truth assignment is exactly $|\varphi| - 1$, which is the case iff φ is unsatisfiable and there is a truth assignment that satisfies $|\varphi| - 1$ clauses of φ. In turn, by (P1), this is the case iff both of the following hold:

- there is no valuation v' with range in $\{0, 1\}$ such that $\#\big((), q, v'(D_\varphi)\big) = \chi - |\varphi|$, and
- there exists a valuation v with range in $\{0, 1\}$ for which $\#\big((), q, v(D_\varphi)\big) = \chi - (|\varphi| - 1) = \chi - |\varphi| + 1$.

By (P3), the above hold (together) iff $\min\big((), q, D_\varphi\big) = \chi - |\varphi| + 1$. Therefore, f is satisfiable and g is unsatisfiable if and only if D_φ, $()$ and $\chi - |\varphi| + 1$ form a positive instance of $\mathsf{MIN}^{=}[q]$. □

5.3. *Computing* min*: intractability beyond* RA$^+\{\cap\}$

While computing min is hard in the general case, there exists a large fragment of RA for which min can be computed via naive evaluation. In the set case this fragment is well known to be positive relational algebra [17] consisting of selection, projection, Cartesian product, and union. Notice that over sets the intersection of two relations is expressible by join, but over bags this is no longer the case. It turns out that the good behavior of join with respect to certain answers extends to bags, when we add intersection explicitly. Indeed, for RA$^+\{\cap\}$, one can compute min simply by using naive evaluation.

Theorem 1. *Let q be an* RA$^+\{\cap\}$ *expression of arity n, let D be a database, and let $\bar{a} \in \mathsf{Const}(D)^n$. Then,* $\min(\bar{a}, q, D) = \mathsf{naive}(\bar{a}, q, D)$.

Proof. This is an immediate consequence of a more general result that we shall see later on (Proposition 9). □

From the fact that bag relational algebra queries are in DLogSpace with respect to data complexity [5,6], we then immediately get the following:

Corollary 5. *For expressions q in* RA$^+\{\cap\}$*, the problems* $\mathsf{MIN}^{<}[q]$, $\mathsf{MIN}^{>}[q]$ *and* $\mathsf{MIN}^{=}[q]$ *are all in* DLogSpace. □

As a matter of fact, RA$^+\{\cap\}$ is the best fragment for which we can compute certain answers efficiently under bag semantics. To show this, from the diagram in Fig. 1, it suffices to prove that the problem is intractable for SPC$\{\cup\}$, as well as for all fragments with duplicate elimination. This is what we do next; in particular, we shall see that in all of these fragments the intractable complexity is exactly the same, namely NP-complete, coNP-complete, and DP-complete, when θ is $<$, $>$, and $=$, respectively.

Hardness for SPC$\{\cup\}$. We first study the max-union operator $\cup$. Recall that, in the set case, adding union is harmless and preserves the property that certain answers can be found by naive evaluation. But under bag semantics, adding $\cup$ to SPC gives rise to a substantial increase in the complexity of computing min.

Theorem 2. *There exists an* SPC$\{\cup\}$ *query q for which* $\mathsf{MIN}^{<}[q]$ *is* NP-*hard,* $\mathsf{MIN}^{>}[q]$ *is* coNP-*hard, and* $\mathsf{MIN}^{=}[q]$ *is* DP-*hard.*

Proof. Let φ be a k-CNF formula, and define $\mathsf{dbenc}(\varphi)$ as the database with relations R and T, where R is the encoding R_φ of φ, and T consists of precisely $|\varphi|$ occurrences of $(0, 1)^k$. Consider the query $q = \pi_{()}(R \cup T)$; we will show that dbenc and q satisfy the properties of Lemma 8.

Let v be a valuation, let $A = \mathsf{adom}\big(v(D_\varphi)\big)^{2k}$, and let $\bar{u} = (0, 1)^k$. First, we will show that

$$\#\big((), q, v(D_\varphi)\big) = 2^k \cdot |\varphi| - \#\big(\bar{u}, v(R_\varphi)\big) \geq (2^k - 1) \cdot |\varphi| \tag{6}$$

By definition of projection, $\#\big((), \pi_{()}(R \cup T), v(D_\varphi)\big)$ is equal to

$$\sum_{\bar{a} \in A} \#\big(\bar{a}, R \cup T, v(D_\varphi)\big) . \tag{6.1}$$

As $R = R_\varphi$ and T does not contain any tuple distinct from $\bar{u}$, (6.1) equals

$$\sum_{\bar{a} \in A : \bar{a} \neq \bar{u}} \#\big(\bar{a}, v(R_\varphi)\big) + \underbrace{\#\big(\bar{u}, R \cup T, v(D_\varphi)\big)}_{(\dagger)} . \tag{6.2}$$

The summation above amounts to $|v(R_\varphi)| - \#\big(\bar{u}, v(R_\varphi)\big)$, and by definition of $\cup$ we have that $(\dagger)$ is equal to $\max \big\{ \#\big(\bar{u}, v(R_\varphi)\big), \#\big(\bar{u}, v(T)\big) \big\}$. By construction, T consists of exactly $|\varphi|$ occurrences of the constant tuple $\bar{u}$, so $\#\big(\bar{u}, v(T)\big) = \#(\bar{u}, T) = |\varphi|$. In turn, since $|v(R_\varphi)| = |R_\varphi| = (2^k - 1) \cdot |\varphi|$, we have that (6.2) is equal to

$$(2^k - 1) \cdot |\varphi| - \#\big(\bar{u}, v(R_\varphi)\big) + \underbrace{\max\big\{\#\big(\bar{u}, v(R_\varphi)\big), |\varphi|\big\}}_{(\dagger\dagger)} . \tag{6.3}$$

[2] We will consider queries that project onto the empty tuple, so the only possible answers are either the empty bag, or bags containing one or more occurrences of $()$. This is only for technical convenience: the results hold even if we disallow projection over the empty tuple.

Obviously $\#(\bar{u}, v(R_\varphi)) \leq |\varphi|$ by Lemma 6, so (††) $= |\varphi|$. Thus, from (6.3) we directly get (6).

By Lemma 6, there exists a truth assignment that satisfies m clauses of φ iff there exists a valuation v with range in $\{0, 1\}$ such that $\#(\bar{u}, v(R_\varphi)) = m$. In turn, from (6), this is the case iff $\#((), q, v(D_\varphi)) = 2^k \cdot |\varphi| - m$, thus proving that (P1) holds for $\chi = 2^k \cdot |\varphi|$. In addition, from (6) and Lemma 6(d) it directly follows that (P2) holds as well. □

Handling duplicate elimination. In terms of tractability, no fragment survives the addition of duplicate elimination. Indeed, for all fragments from SPC{ε} to full relational algebra RA, the decision problems are NP-complete, coNP-complete, and DP-complete for $<$, $>$ and $=$, respectively.

Theorem 3. *There exists an* SPC{ε} *query q for which* $\mathsf{MIN}^{<}[q]$ *is* NP-*hard,* $\mathsf{MIN}^{>}[q]$ *is* coNP-*hard, and* $\mathsf{MIN}^{=}[q]$ *is* DP-*hard.*

Proof. Let φ be a k-CNF formula, define $\mathsf{dbenc}(\varphi)$ as the database D_φ consisting of the relation $S = \mathsf{enc}(\varphi, \iota)$, for some injection ι from φ to Const, and consider the query $q = \pi_{()}(\varepsilon(S))$.[3] We will show that dbenc and q satisfy the properties of Lemma 8.

By construction and by Lemma 6(a), for every valuation v, the number of occurrences of $\iota(C)(0, 1)^k$ in $v(\mathsf{enc}(C, \iota))$ is either 1 or 2, and any other tuple occurs at most once. Recall also that, for distinct clauses C and C', no tuple in $\mathsf{enc}(C, \iota)$ unifies with any tuple in $\mathsf{enc}(C', \iota)$.

From Lemma 6(c) it follows that (P1) holds for $\chi = 2^k \cdot |\varphi|$ if we show that, for every valuation v,

$$\#((), q, v(D_\varphi)) = 2^k \cdot |\varphi| - m \iff \#((0, 1)^k, v(R_\varphi)) = m\,, \qquad (7)$$

where $R_\varphi = \mathsf{enc}(\varphi)$. In addition, from (7) and Lemma 6(d) it also follows that (P2) holds.

Let v be a valuation, and let S_j denote the set of tuples that occur precisely j times in $v(S)$. Then,

$$\#((), q, v(D_\varphi)) = \sum_{\bar{a} \in S_1} 1 + \sum_{\bar{a} \in S_2} 1 = |v(S)| - \sum_{\bar{a} \in S_2} 1.$$

Since $|v(S)| = |S| = 2^k \cdot |\varphi|$, we have that $\#((), q, v(D_\varphi)) = 2^k \cdot |\varphi| - m$ if and only if $\sum_{\bar{a} \in S_2} 1 = m$. In turn, this is the case iff

(†) there exist exactly m distinct clauses $C_1, \ldots, C_m$ in φ such that, for every $i \in \{1, \ldots, m\}$, $\#(\iota(C_i)(0, 1)^k, v(\mathsf{enc}(C_i, \iota)))$ is equal to 2.

By construction, we have that $\#(\iota(C)(0, 1)^k, v(\mathsf{enc}(C, \iota))) = 2$ iff $\#((0, 1)^k, v(\mathsf{enc}(C))) = 1$, for every $C \in \varphi$. So, by Lemma 6(a), (†) holds iff $\#((0, 1)^k, v(R_\varphi)) = m$. □

5.4. Computing max: hardness for simple queries

The problem of computing max is hard already for very simple queries: in fact, $\mathsf{MAX}^{>}[q]$ is NP-complete for a query q that returns a base relation [21]. Here, we complete the picture by proving that deciding $\mathsf{MAX}^{=}[q]$ for the same class of queries is complete for DP.

Theorem 4. *The problem* $\mathsf{MAX}^{=}[q]$ *is* DP-*hard even for a query q that returns a base relation.*

Proof. We provide a reduction from SAT-UNSAT. To this end, let f and g be two 3-CNF formulae, let φ denote $h(f, g)$, let D_φ be the database consisting of the single relation $R = R_\varphi$, and consider the query $q = R$. Observe that φ is a 4-CNF formula and, for every valuation v, we have that $\#((0, 1)^4, q, v(D_\varphi)) = \#((0, 1)^4, v(R_\varphi))$.

By Lemma 7, f is satisfiable and g is unsatisfiable iff the maximum number of clauses of φ that can be satisfied by a single truth assignment is $|\varphi| - 1$. By Lemma 6(c), this is the case iff both of the following hold:

(†) there exists a valuation v with range in $\{0, 1\}$ such that $\#((0, 1)^4, v(R_\varphi)) = |\varphi| - 1$, and

(‡) there is no valuation v' with range in $\{0, 1\}$ such that $\#((0, 1)^4, v'(R_\varphi)) = |\varphi|$.

Observe that Lemma 6(d) directly implies that

$$\max_{v \in V}\{\#((0, 1)^k, v(R_\varphi))\} = \max_{v \in V'}\{\#((0, 1)^k, v(R_\varphi))\},$$

where V is the set of all valuations, and $V' \subseteq V$ consists of all valuations with range in $\{0, 1\}$.

Thus, (†) and (‡) hold together iff $\max((0, 1)^4, q, D_\varphi) = |\varphi| - 1$. Therefore, f is satisfiable and g is unsatisfiable iff D_φ, $(0, 1)^k$, and $|\varphi| - 1$ are a positive instance of $\mathsf{MAX}^{=}[q]$ for $q = R$. □

5.5. Complex selection conditions

In our definition of relational algebra, we assumed that selection conditions are equalities of the form $i = j$, where i and j denote positions within a table. Selection conditions γ can be generalized to arbitrary Boolean combinations of equalities, according to following the grammar:

$$\gamma \;:=\; (i = j) \mid \gamma \wedge \gamma \mid \gamma \vee \gamma \mid \neg\gamma$$

We briefly describe how more complex conditions affect our results.

If we only add conjunction (that is, selection conditions are conjunctions of equalities), there is no change at all. This is because the cascade-of-selections equivalence,

$$\sigma_{\gamma \wedge \gamma'}(e) \equiv \sigma_\gamma(\sigma_{\gamma'}(e)),$$

applies under bag semantics as well.

If we add disjunction, it might at first appear that this leads to intractability, because the following equivalence

$$\sigma_{\gamma \vee \gamma'}(e) \equiv \sigma_\gamma(e) \cup \sigma_{\gamma'}(e)$$

holds under bag semantics, and we have seen that the fragment SPC{$\cup$} is intractable. However, adding disjunction to selection conditions is weaker than adding max-union, and preserves tractability.

Proposition 9. *Let q be an* $\mathsf{RA}^{+}\{\cap\}$ *query of arity n, where selection conditions are positive Boolean combinations of equalities; then,* $\mathsf{MIN}^{<}[q]$ *and* $\mathsf{MIN}^{>}[q]$ *can be solved in* DLogSpace. *More precisely, for every database D and for every n-tuple $\bar{a}$ of constants,* $\min(\bar{a}, q, D) = \mathsf{naive}(\bar{a}, q, D)$.

Proof. For every valuation v, we have $v(\bar{a}) = \bar{a}$ and

$$\mathsf{naive}(\bar{a}, q, D) \leq \sum_{\bar{a}' : v(\bar{a}') = \bar{a}} \mathsf{naive}(\bar{a}', q, D) \leq \#(\bar{a}, q, v(D))\,, \qquad (8)$$

where the rightmost inequality is by Lemma 9, which is given in Appendix. Let us now define a valuation $\tilde{v}$ whose restriction to $\mathsf{Null}(D)$ is injective and such that the image of $\mathsf{Null}(D)$ under it is disjoint with $\mathsf{Const}(D)$. In other words, each null in D is mapped to a distinct constant that does not appear in D.

From the definition of naive evaluation, it then follows that $\#(\bar{a}, q, \tilde{v}(D)) = \mathsf{naive}(\bar{a}, q, D)$. This, together with (8), proves that $\mathsf{naive}(\bar{a}, q, D) = \min(\bar{a}, q, D)$. □

[3] The output of this query is always either the empty bag, or the bag consisting of a single occurrence of the empty tuple ().

The problems become intractable when conditions are unrestricted Boolean combinations of equalities, even when these are added to the SPC fragment.

Proposition 10. *In the extension of* SPC *that allows arbitrary Boolean combinations of equalities as selection conditions, there is a query q for which* $\mathsf{MIN}^{<}[q]$ *is* NP-*complete,* $\mathsf{MIN}^{>}[q]$ *is* coNP-*complete, and* $\mathsf{MIN}^{=}[q]$ *is* DP-*complete.*

Proof. Let φ be a k-CNF formula, and define $\mathsf{dbenc}(\varphi)$ as the database D_φ with relations R and T, where R is the encoding R_φ of φ and T consists of a single occurrence of $(0,1)^k$. Consider the query $q = \pi_{()}\big(\sigma_\gamma(R \times T)\big)$, where γ is the disjunction of all inequalities $i \neq j$ for every $i \in \{1, \dots, 2^k\}$ and $j = 2^k + 1$. We will show that dbenc and q satisfy the properties of Lemma 8.

For a valuation v, the answer to q on $v(D_\varphi)$ consists of as many occurrences of $()$ as there are tuples different from $(0,1)^k$ in R_φ. More precisely,

$$\begin{aligned}\#\big((), q, v(D_\varphi)\big) &= \sum_{\bar{a} \neq (0,1)^k} \#\big(\bar{a}, v(R_\varphi)\big) \\ &= |v(R_\varphi)| - \#\big((0,1)^k, v(R_\varphi)\big)\,.\end{aligned} \tag{9}$$

Therefore, (P1) holds for $\chi = (2^k - 1) \cdot |\varphi| = |R_\varphi| = |v(R_\varphi)|$ by Lemma 6(c). In addition, from (9) and Lemma 6(d) it also follows that (P2) holds. □

6. Conclusions

We have provided two complete classifications: of the expressive power of fragments of bag relational algebra, and of the complexity of computing certain and possible answers in those fragments. For the complexity of certain answers, we have a dichotomy: either they can be computed efficiently by naive evaluation, or their complexity is intractable, which means NP-complete, or coNP-complete, or DP-complete (depending on how the problem is turned into a decision problem).

Directions for future work are motivated by the recent work on bag semantics in data management applications where incompleteness naturally occurs, such as data exchange [15] and OBDA [16]. Notice that we have primarily concentrated on the closed-world semantics, which as of late has been actively studied in those contexts; see, e.g., [27–31]. Thus we believe our results could be relevant to understanding the complexity of these applications under the closed-world assumption. As another direction for future work, we would like to study the complexity of finding certain and possible answers in the fragments of bag relational algebra under the open-world assumption. The general case is of course undecidable, but the picture for the fragments studied here is not clear. Finally, we would like to use our results as the starting point for the study of answering queries with grouping and aggregation over incomplete data, as such queries rely on bag semantics.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank the anonymous reviewers for their constructive comments and suggestions, which have contributed to the improvement of this work.

Appendix. Additional proofs

Proof of Lemma 3. We proceed by induction on the structure of e. Note that all subexpressions of e are nullary and, since there are no attributes, none of them can be a selection. Moreover, according to the syntax of bag relational algebra defined in Section 2, *every* subexpression of e *must* mention R at least once (and it does not mention any other symbol of the schema due to the assumptions of the lemma).

Base case: e is R. Then, $|e(D)| = |R^D|$.

Inductive step:

- e is $\pi_{()}(e')$. Then, $|e(D)| = |e'(D)|$ and by the induction hypothesis $|e'(D)| \geq |R^D|$.
- e is $e' \times e''$. Then, $|e(D)| = |e'(D)| \cdot |e''(D)|$. By the induction hypothesis $|e'(D)|$ and $|e''(D)|$ are both $\geq |R^D|$, and thus their product is as well.
- e is $e' \cup e''$. Then, $|e(D)| = \max\big\{|e'(D)|, |e''(D)|\big\} \geq |e'(D)|$ and by the induction hypothesis $|e'(D)| \geq |R^D|$.
- e is $e' \uplus e''$. Then, $|e(D)| = |e'(D)| + |e''(D)| \geq |e'(D)|$. By the induction hypothesis both $|e'(D)|$ and $|e''(D)|$ are $\geq |R^D|$, and therefore their sum is as well. □

Proof of Lemma 4. We proceed by induction on the structure of e. Note that all subexpressions of e are nullary and, since there are no attributes, none of them can be a selection.

Base case: e is $e_R \star e_S$, where $\star \in \{\times, \uplus, \cup\}$ and e_R, e_S are $\mathsf{RA}^+\{\cup\}$ expressions such that e_R mentions only R and e_S mentions only S. Then, we have $|e(D)| = f_\star\big(|e_R(D)|, |e_S(D)|\big)$, where $f_\times(m,n) = m \cdot n$, $f_\uplus(m,n) = m + n$ and $f_\cup(m,n) = \max(m,n)$. Clearly, for every $m, n \in \mathbb{N}$ and for each $\star \in \{\times, \uplus, \cup\}$, we have that $f_\star(m,n) \geq m$ and $f_\star(m,n) \geq n$. Thus, since $|e_R(D)| \geq |R^D|$ and $|e_S(D)| \geq |S^D|$ by Lemma 3, the claim follows trivially.

Inductive step: Straightforward application of the induction hypothesis as in Lemma 3. □

Lemma 9. *Let q be a n-ary expression in* $\mathsf{RA}^+\{\cap\}$ *and let D be a bag semantics database. For every tuple $\bar{a} \in \mathsf{adom}^n(D)$ and every valuation v for* $\mathsf{Null}(D)$ *the following inequality holds:*

$$\sum_{\substack{\{\bar{b} \in \mathsf{adom}^n(D) \\ \mid\ v\bar{b} = v\bar{a}\}}} \mathsf{naive}(\bar{b}, q, D) \leq \#(v\bar{a}, q, vD)$$

Proof. We proceed by induction on the structure of q.

Base case:

- $q = R$. Trivially, each occurrence of $\bar{a}$ in R^D results in one occurrence of $v(\bar{a})$ in $R^{v(D)}$.

Inductive step:

- $q = \pi_\alpha(q')$. Assume that q' has arity m. Due to the definition of Projection, the following equality holds for every valuation v for $\mathsf{Null}(D)$:

$$\#(v\bar{a}, \pi_\alpha(q'), vD) = \sum_{\substack{\{\bar{c} \in \mathsf{adom}^m(vD) \\ \mid\ \pi_\alpha(\bar{c}) = v\bar{a}\}}} \#(\bar{c}, q', vD)\,. \tag{A.1}$$

Notice that $\bar{c}$ in Eq. (A.1) is a tuple of constants, hence from the definition of semantics for incomplete databases we can conclude that there exists at least one tuple $\bar{u} \in \mathsf{adom}^m(D)$ such that $v\bar{u} = \bar{c}$. We now apply the inductive hypothesis

to q' and $\bar{u}$ and derive the following:

$$\sum_{\substack{\{\bar{b}\in \mathsf{adom}^m(D) \\ \mid\ v\bar{u}=v\bar{b}\}}} \mathsf{naive}(\bar{b}, q', D) \leq \#(v\bar{u}, q', vD)\,. \tag{A.2}$$

By assumption, $v\bar{u}$ in Eq. (A.2) is equal to $\bar{c}$. For this reason, we can substitute $v\bar{u}$ with $\bar{c}$ obtaining the following inequality:

$$\sum_{\substack{\{\bar{b}\in \mathsf{adom}^m(D) \\ \mid\ v\bar{b}=\bar{c}\}}} \mathsf{naive}(\bar{b}, q', D) \leq \#(\bar{c}, q', vD)\,. \tag{A.3}$$

Observe now that v is a function, hence for every $\bar{b} \in \mathsf{adom}^m(D)$ there exists one and only one tuple $\bar{c}$ of constants such that $v\bar{b} = \bar{c}$. In light of this consideration, we can apply Eq. (A.3) to each tuple $\bar{c}$ in the right-hand side of Eq. (A.1) and obtain the following inequality.

$$\begin{aligned}&\sum_{\substack{\{\bar{c}\in \mathsf{adom}^m(vD) \\ \mid\ \pi_\alpha(\bar{c})=v\bar{a}\}}} \left(\sum_{\substack{\{\bar{b}\in \mathsf{adom}^m(D) \\ \mid\ v\bar{b}=\bar{c}\}}} \mathsf{naive}(\bar{b}, q', D) \right) \leq \\ &\sum_{\substack{\{\bar{c}\in \mathsf{adom}^m(vD) \\ \mid\ \pi_\alpha(\bar{c})=v\bar{a}\}}} \#(\bar{c}, q', vD)\,.\end{aligned} \tag{A.4}$$

The double summation on the left-hand side of Eq. (A.4) can now be re-written as a single summation over all those tuples $\bar{b}$ such that $v\bar{b}$ is equal to $v\bar{a}$ on α. Hence:

$$\sum_{\substack{\{\bar{b}\in \mathsf{adom}^m(D) \\ \mid\ \pi_\alpha(v\bar{b})=v\bar{a}\}}} \mathsf{naive}(\bar{b}, q', D) \leq \sum_{\substack{\{\bar{c}\in \mathsf{adom}^m(vD) \\ \mid\ \pi_\alpha(\bar{c})=v\bar{a}\}}} \#(\bar{c}, q', vD)\,. \tag{A.5}$$

Consider now $\mathsf{naive}(\bar{a}, \pi_\alpha(q'), D)$, and let $\bar{z}$ be a generic tuple in $\mathsf{adom}^m(D)$ such that $\bar{z}$ is equal to $\bar{a}$ on the attributes in α. Since

$$Z = \{\bar{z} \in \mathsf{adom}^m(D) \ \mid\ \bar{z} \text{ is equal to } \bar{a} \text{ on } \alpha\}$$

is a subset of

$$B = \{\bar{b} \in \mathsf{adom}^m(D) \ \mid\ v\bar{b} \text{is equal to } v\bar{a} \text{ on } \alpha\}\,,$$

the following inequality holds:

$$\sum_{\bar{z}\in Z} \mathsf{naive}(\bar{z}, q', D) \leq \sum_{\bar{b}\in B} \mathsf{naive}(\bar{b}, q', D)\,.$$

The claim now follows from the definition of projection, that is,

$$\mathsf{naive}(\bar{a}, \pi_\alpha(q'), D) = \sum_{\bar{z}\in Z} \mathsf{naive}(\bar{z}, q', D)\,.$$

- $q = \sigma_\gamma(q')$, where γ is a positive Boolean combination of equalities. W.l.o.g. we assume γ to be in disjunctive normal form, that is, $\gamma = \bigvee_i(\bigwedge_j(a_{i,j} = b_{i,j}))$.

  If this is the case, a tuple $\bar{a}$ satisfies γ if it satisfies all the equalities in one of the disjuncts of γ.
  Let A be the set of n-tuples that unify with $\bar{a}$; for every valuation v we have

$$\begin{aligned}\sum_{\bar{a}'\in A} \mathsf{naive}\big(\bar{a}', \sigma_\gamma(q'), D\big) &\leq \sum_{\bar{a}'\in A} \mathsf{naive}\big(\bar{a}', q', D\big) \\ &\leq \#\big(v(\bar{a}), q', v(D)\big)\,,\end{aligned} \tag{A.6}$$

  where the left inequality is by definition of the selection operation, and the right one is by the inductive hypothesis. If $v(\bar{a})$ satisfies γ, then

$$\#\big(v(\bar{a}), \sigma_\gamma(q'), v(D)\big) = \#\big(v(\bar{a}), q', v(D)\big)$$

  and so the claim follows by Eq. (A.6). Otherwise, when $v(\bar{a})$ does not satisfy γ, for each disjunct of γ there is an equality $i = j$ not satisfied by $v(\bar{a})$. Then, from the definition of selection we have $\#\big(v(\bar{a}), \sigma_\gamma(q'), v(D)\big) = 0$.
  Observe now that valuations are functions, hence $v\bar{a}.i \neq v\bar{a}.j$ implies that $\bar{a}.i \neq \bar{a}.j$. In turn, this implies all the tuples $\bar{b} \in \mathsf{adom}^n(D)$ such that $v\bar{b} = v\bar{a}$ do not satisfy the equality conditions violated by $v\bar{a}$. From this consideration, we can conclude that $\sum_{\{\bar{b}\in \mathsf{adom}^n(D) \ \mid\ v\bar{b}=v\bar{a}\}} \mathsf{naive}(\bar{b}, \sigma_\gamma(q'), D) = 0$, proving the claim.
- $q = q_1 \uplus q_2$. From the definition of $\uplus$ we have the following:

$$\begin{aligned}&\sum_{\substack{\{\bar{b}\in \mathsf{adom}^n(D) \\ \mid\ v\bar{b}=v\bar{a}\}}} \mathsf{naive}(\bar{b}, q_1 \uplus q_2, D) = \\ &\sum_{\substack{\{\bar{b}\in \mathsf{adom}^n(D) \\ \mid\ v\bar{b}=v\bar{a}\}}} \mathsf{naive}(\bar{b}, q_1, D) + \sum_{\substack{\{\bar{b}\in \mathsf{adom}^n(D) \\ \mid\ v\bar{b}=v\bar{a}\}}} \mathsf{naive}(\bar{b}, q_2, D)\,.\end{aligned} \tag{A.7}$$

  Applying the inductive hypothesis to q_1 and q_2 we obtain:

$$\sum_{\substack{\{\bar{b}\in \mathsf{adom}^n(D) \\ \mid\ v\bar{b}=v\bar{a}\}}} \mathsf{naive}(\bar{b}, q_1, D) \leq \#(v\bar{a}, q_1, vD)\,; \tag{A.8}$$

$$\sum_{\substack{\{\bar{b}\in \mathsf{adom}^n(D) \\ \mid\ v\bar{b}=v\bar{a}\}}} \mathsf{naive}(\bar{b}, q_2, D) \leq \#(v\bar{a}, q_2, vD)\,. \tag{A.9}$$

  Finally, we apply the definition of additive union to vD and obtain the following:

$$\#(v\bar{a}, q_1 \uplus q_2, vD) = \#(v\bar{a}, q_1, vD) + \#(v\bar{a}, q_2, vD)\,. \tag{A.10}$$

  The claim now follows from Eqs. (A.7), (A.8), (A.9), (A.10).
- $q = q_1 \times q_2$. Suppose that the arity of q_1 is a and the arity of q_2 is b, and assume two tuples $\bar{a}_1 \in \mathsf{adom}^a(D)$ and $\bar{a}_2 \in \mathsf{adom}^a(D)$ such that $\bar{a} = \bar{a}_1 \circ \bar{a}_2$. From the definition of Cartesian product we obtain the following.

$$\#(v\bar{a}, q_1 \times q_2, vD) = \#(v\bar{a}_1, q_1, vD) \cdot \#(v\bar{a}_2, q_2, vD)\,. \tag{A.11}$$

  Applying the inductive hypothesis on $v\bar{a}_1$ and $v\bar{a}_2$, we obtain the following inequalities:

$$\sum_{\substack{\{\bar{b}_1\in \mathsf{adom}^a(D) \\ \mid\ v\bar{b}_1=v\bar{a}_1\}}} \mathsf{naive}(\bar{b}_1, q_1, D) \leq \#(v\bar{a}_1, q_1, vD)\,; \tag{A.12}$$

$$\sum_{\substack{\{\bar{b}_2\in \mathsf{adom}^b(D) \\ \mid\ v\bar{b}_2=v\bar{a}_2\}}} \mathsf{naive}(\bar{b}_2, q_2, D) \leq \#(v\bar{a}_2, q_2, vD)\,. \tag{A.13}$$

  We now apply Eqs. (A.12) and (A.13) to Eq. (A.11), and we derive the following:

$$\begin{aligned}&\sum_{\substack{\{\bar{b}_1\in \mathsf{adom}^a(D) \\ \mid\ v\bar{b}_1=v\bar{a}_1\}}} \mathsf{naive}(\bar{b}_1, q_1, D) \cdot \\ &\sum_{\substack{\{\bar{b}_2\in \mathsf{adom}^b(D) \\ \mid\ v\bar{b}_2=v\bar{a}_2\}}} \mathsf{naive}(\bar{b}_2, q_2, D) \leq \#(v\bar{a}, q_1 \times q_2, vD)\,.\end{aligned} \tag{A.14}$$

  Let $\mathcal{S} = \{\bar{b} \in \mathsf{adom}^n(D) \ \mid\ v\bar{b} = v\bar{a}\}$. From the definition of Cartesian product we can derive the following equality:

$$\begin{aligned}&\sum_{\bar{b}\in\mathcal{S}} \mathsf{naive}(\bar{b}, q_1 \times q_2, D) = \\ &\sum_{\substack{\{\bar{b}_1,\bar{b}_2 \\ \mid\ \bar{b}_1\circ\bar{b}_2\in\mathcal{S}\}}} \mathsf{naive}(\bar{b}_1, q_1, D) \cdot \mathsf{naive}(\bar{b}_2, q_2, D)\,.\end{aligned} \tag{A.15}$$

To prove the claim, observe that the left-hand side of Eq. (A.14) is always greater or equal to the right-hand side of Eq. (A.15).

- $q = q_1 \cap q_2$. From the definition of $\cap$ we can derive the following:

$$\sum_{\substack{\{\bar{b}\in\mathsf{adom}^n(D)\\ \mid\ v\bar{b}=v\bar{a}\}}} \mathsf{naive}(\bar{b}, q_1 \cap q_2, D) = \sum_{\substack{\{\bar{b}\in\mathsf{adom}^n(D)\\ \mid\ v\bar{b}=v\bar{a}\}}} \min\{\,\mathsf{naive}(\bar{b}, q_1, D),\ \mathsf{naive}(\bar{b}, q_2, D)\,\}\,. \tag{A.16}$$

We now apply the inductive hypothesis to q_1 and q_2 and obtain the following:

$$\sum_{\substack{\{\bar{b}\in\mathsf{adom}^n(D)\\ \mid\ v\bar{b}=v\bar{a}\}}} \mathsf{naive}(\bar{b}, q_1, D) \leq \#(v\bar{a}, q_1, vD)\,; \tag{A.17}$$

$$\sum_{\substack{\{\bar{b}\in\mathsf{adom}^n(D)\\ \mid\ v\bar{b}=v\bar{a}\}}} \mathsf{naive}(\bar{b}, q_2, D) \leq \#(v\bar{a}, q_2, vD)\,. \tag{A.18}$$

Finally, we apply the definition of intersection to vD and obtain the following.

$$\#(v\bar{a}, q_1 \cap q_2, vD) = \min\{\#(v\bar{a}, q_1, vD), \#(v\bar{a}, q_2, vD)\}\,. \tag{A.19}$$

We now observe that the right-hand side of Eq. (A.16) is always lesser or equal to the left-hand side of Eqs. (A.17) and (A.18). This is due to the application of the *min* operator. In turn, together with Eq. (A.19) this proves that the following inequality holds:

$$\sum_{\substack{\{\bar{b}\in\mathsf{adom}^n(D)\\ \mid\ v\bar{b}=v\bar{a}\}}} \mathsf{naive}(\bar{b}, q_1 \cap q_2, D) \leq \#(v\bar{a}, q_1 \cap q_2, vD)\,.$$

This in turn proves the claim. □

References

[1] M. Console, P. Guagliardo, L. Libkin, Fragments of bag relational algebra: Expressiveness and certain answers, in: ICDT, in: LIPIcs, vol. 127, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019, pp. 8:1–8:16.
[2] C.J. Date, H. Darwen, A Guide to the SQL Standard, Addison-Wesley, 1996.
[3] R. Ramakrishnan, J. Gehrke, Database Management Systems, McGraw-Hill, 2003.
[4] J. Albert, Algebraic properties of bag data types, in: VLDB, 1991, pp. 211–219.
[5] S. Grumbach, T. Milo, Towards tractable algebras for bags, J. Comput. System Sci. 52 (3) (1996) 570–588.
[6] L. Libkin, L. Wong, Query languages for bags and aggregate functions, J. Comput. System Sci. 55 (2) (1997) 241–272.
[7] P. Buneman, S.A. Naqvi, V. Tannen, L. Wong, Principles of programming with complex objects and collection types, Theoret. Comput. Sci. 149 (1) (1995) 3–48.
[8] R.G.G. Cattell, The Object Database Standard: ODMG-93, Morgan Kaufmann, 1993.
[9] S. Chaudhuri, M.Y. Vardi, Optimization of real conjunctive queries, in: PODS, 1993, pp. 59–70.
[10] T.S. Jayram, P.G. Kolaitis, E. Vee, The containment problem for real conjunctive queries with inequalities, in: PODS, 2006, pp. 80–89.
[11] S. Cohen, Equivalence of queries combining set and bag-set semantics, in: PODS, 2006, pp. 70–79.
[12] P.G. Kolaitis, The query containment problem: Set semantics vs. bag semantics, in: AMW, 2013.
[13] L.E. Bertossi, G. Gottlob, R. Pichler, Datalog: Bag semantics via set semantics, in: ICDT, 2019, pp. 16:1–16:19.
[14] L.S. Colby, L. Libkin, Tractable iteration mechanisms for bag languages, in: ICDT, 1997, pp. 461–475.
[15] A. Hernich, P.G. Kolaitis, Foundations of information integration under bag semantics, in: LICS, IEEE Computer Society, 2017, pp. 1–12.
[16] C. Nikolaou, E.V. Kostylev, G. Konstantinidis, M. Kaminski, B.C. Grau, I. Horrocks, The bag semantics of ontology-based data access, in: IJCAI, 2017, pp. 1224–1230.
[17] T. Imielinski, W. Lipski, Incomplete information in relational databases, J. ACM 31 (4) (1984) 761–791.
[18] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison-Wesley, 1995.
[19] S. Abiteboul, P. Kanellakis, G. Grahne, On the representation and querying of sets of possible worlds, Theoret. Comput. Sci. 78 (1) (1991) 158–187.
[20] S. Grumbach, L. Libkin, T. Milo, L. Wong, Query languages for bags: expressive power and complexity, SIGACT News 27 (2) (1996) 30–44.
[21] M. Console, P. Guagliardo, L. Libkin, On querying incomplete information in databases under bag semantics, in: IJCAI, ijcai.org, 2017, pp. 993–999.
[22] C.H. Papadimitriou, M. Yannakakis, The complexity of facets (and some facets of complexity), J. Comput. System Sci. 28 (2) (1984) 244–259.
[23] T.J. Green, G. Karvounarakis, V. Tannen, Provenance semirings, in: PODS, ACM, 2007, pp. 31–40.
[24] M. Arenas, P. Barceló, L. Libkin, F. Murlak, Foundations of Data Exchange, Cambridge University Press, 2014.
[25] M. Bienvenu, M. Ortiz, Ontology-mediated query answering with data-tractable description logics, in: Reasoning Web, 2015, pp. 218–307.
[26] M. Lenzerini, Data integration: a theoretical perspective, in: PODS, 2002, pp. 233–246.
[27] S. Ahmetaj, M. Ortiz, M. Simkus, Polynomial datalog rewritings for expressive description logics with closed predicates, in: IJCAI, 2016, pp. 878–885.
[28] G. Amendola, N. Leone, M. Manna, P. Veltri, Enhancing existential rules by closed-world variables, in: IJCAI, 2018, pp. 1676–1682.
[29] A. Hernich, Answering non-monotonic queries in relational data exchange, Log. Methods Comput. Sci. 7 (3) (2011).
[30] A. Hernich, L. Libkin, N. Schweikardt, Closed world data exchange, ACM Trans. Database Syst. 36 (2) (2011) 14:1–14:40.
[31] C. Lutz, I. Seylan, F. Wolter, Ontology-mediated queries with closed predicates, in: IJCAI, 2015, pp. 3120–3126.

Please cite this article as: M. Console, P. Guagliardo and L. Libkin, Fragments of bag relational algebra: Expressiveness and certain answers, Information Systems (2020) 101604, https://doi.org/10.1016/j.is.2020.101604.