Requêtes sur des données à ordre incomplet

Antoine Amarilli
Institut Mines–Télécom
Télécom ParisTech; CNRS LTCI

M. Lamine Ba
Institut Mines–Télécom
Télécom ParisTech; CNRS LTCI

Daniel Deutch
Blavatnik School of Computer Science
Tel Aviv University

Pierre Senellart
Institut Mines–Télécom; Télécom ParisTech; CNRS LTCI
& National University of Singapore; CNRS IPAL

ABSTRACT

Combiner des données ordonnées qui proviennent de différentes sources exige de recourir à un formalisme de *données à ordre incomplet* qui puisse représenter l'incertitude sur les différents ordres possibles. Des exemples d'application sont des listes d'établissements, tels des hôtels ou des restaurants, triés suivant une fonction inconnue représentant leur pertinence pour une requête, ou leur évaluation par des clients ; des documents édités de façon concurrente avec plusieurs manières possibles d'ordonner les contributions individuelles ; des résultats d'intégration de séquences d'événements comme des données de capteurs ou des entrées de fichiers journaux.

Le présent travail étend l'algèbre relationnelle positive aux données ordonées et aux données à ordre incomplet, et introduit un ensemble d'axiomes pour guider la conception d'une sémantique multiensembliste pour le langage. Nous introduisons deux telles sémantiques, et montrons que l'une d'entre elles est la plus générale possible pour notre ensemble d'axiomes. Nous construisons ensuite un système de représentation au sens fort pour ces sémantiques, qui s'appuie sur des ordres partiels interprétés suivant une sémantique des mondes possibles. Nous étudions l'expressivité de notre langage de requêtes, en le rattachant à des mesures de complexité sur les ordres partiels. Nous introduisons un opérateur de *top-k* pour extraire les *k* premiers éléments de l'ordre, et étudions la complexité de l'évaluation de requêtes, au sens des réponses possibles et des réponses certaines. Nous complétons finalement le langage avec un opérateur qui permet d'éliminer les doublons, et investiguons l'impact de cette modification sur nos résultats.

1. INTRODUCTION

Many application domains need to combine and transform ordered data from multiple sources. Examples include readings from multiple sensors or log entries from different applications or machines, that need to be combined to form a complete picture of events; rankings of restaurants and hotels based on various criteria (relevance, preference, or customer ratings); concurrent edits of shared documents, where the order of contributions made by different users needs to be merged. Even if the order of items from each individual source is usually known, the order of items across sources is often *uncertain*. For instance, even when sensor readings or log entries are provided with timestamps, these may be ill-synchronized across sensors or machines; rankings of hotels and restaurants may be biased by different preferences of different users; concurrent contributions to documents may be ordered in multiple reasonable ways. We say the resulting information is *order-incomplete*.

This paper studies query evaluation over order-incomplete data in a relational setting. We focus on the running example of lists of restaurants and hotels from a travel Web site, ranked according to an unknown function. An example query would be to obtain an ordered list of restaurant–hotel pairs that are in the same district, such that the restaurant features a particular cuisine, and perhaps limiting the output to the top-*k* such pairs. To evaluate such queries, we need a way to preserve order information through transformations, while accounting for its incompleteness.

To our knowledge, no previously proposed framework or system can be used to evaluate positive relational algebra queries in this sense; see Section 8 for a discussion of the related work. We therefore propose in this paper an extension of the positive relational algebra (PosRA) to ordered and order-incomplete data, and show how to design a bag semantics for it. We present which axioms should necessarily be satisfied, guiding the development of semantics in this context. We further propose different concrete semantics that respect them. We then describe a compact representation system for order-incomplete data based on partial orders, and study the expressiveness and complexity of the semantics over this representation system. Beyond standard relational algebra, we study semantics for enriched languages including order-based selection such as top-*k*, and an explicit operator for duplicate elimination. We next summarize our main contributions in more detail.

Order-incomplete databases (Section 3). We focus on the positive relational algebra, and capture ordered and order-incomplete data with the notion of order-incomplete relations (for short, oi-relations), which are sets of possible orders over tuples. To guide our definition of a semantics for such queries over such relations, we introduce a set of intuitive axioms: the semantics should faithfully generalize standard bag semantics; query evaluation should treat oi-relations as a representation system [17]; and existing order within input relations should be preserved. We then propose concrete semantics satisfying the desiderata: `GEN`, which we prove to be the most general (it imposes the minimal order constraints so that the axioms are satisfied), and `LEX`, based on a natural interpretation of the relational product as lexicographic ordering. We exemplify the usefulness of the semantics, and show some equational properties of positive relational algebra that continue to hold for them.

Partially-ordered databases (Section 4). Our abstraction of order-incomplete relations as sets of possible worlds is useful for defining query semantics, but not practical for representing query evaluation results. We accordingly define *partially-ordered relations* (or po-relations) which introduce a partial order [32] over their tuples. Each po-relation represents the oi-relation obtained as the set of linear extensions of its partial order; however, not all oi-relations may be represented as po-relations. We explain how po-relations can be used as a *strong representation system* [17] for the GEN and LEX semantics, and that query evaluation on po-relations can be performed in polynomial time in the size of the input po-relations.

Expressiveness: output and transformations (Section 5). We study the expressiveness of our semantics for PosRA. We first show that any po-relation can be obtained as the result of some GEN query (even if the query is restricted to have no input, i.e., only uses some constant relations introduced as part of the semantics). In contrast, the output of LEX queries is always *series-parallel* (defined following the corresponding notion on partial orders), assuming that the input also is (and in fact, all series-parallel po-relations can be obtained as LEX query results). We further characterize query outputs by connecting them to order-theoretical notions, bounding the dimension [10] (for GEN) and sp-height (for LEX) of the resulting partial orders as a function of the query size.

We also show that the two semantics are *incomparable* in the sense of the transformations they can capture: while the outputs of LEX are of a restricted form, GEN cannot express the concatenation operator while LEX can. We further show that for any semantics satisfying our axioms (not just GEN and LEX), all expressed transformations must be monotone but not all monotone transformations can be expressed.

Top-k: possibility and certainty (Section 6). We enrich the positive relational algebra with a top-k operator, where ranking is based on the underlying order. Top-k is typically (as in SQL) defined with respect to a total order, but our possible worlds interpretation allows its natural extension to order-incomplete relations. We show that the output of top-k may not be captured in general by oi-relations, and so we restrict our attention to queries where top-k is applied as the last operation. Here we show that the possible top-k results may be enumerated in polynomial time in data complexity, but the output size is generally exponential in k. This blowup motivates our study of *possibility* and *certainty* of a given candidate result for unbounded k, and more generally the problem of deciding possibility and certainty of a candidate output for general queries. We study the complexity of these problems as a function of the input database and candidate result: while certainty may be decided in PTIME, possibility is NP-complete in general but tractable in some restricted cases. We finally extend our results to other forms of order-based selection.

Duplicate elimination (Section 7). After having focused on bag semantics, we introduce an order-aware duplicate elimination operator to the language. For totally ordered relations, we define the semantics of this operator so that it eliminates duplicates only if they are contiguous, i.e., agree on their relative order with respect to all other tuples. If the result still contains duplicates, then we say that duplicate elimination fails (as our goal is to go back to an underlying set relation with no duplicates), and the output is an empty set of possible worlds. When extended to oi-relations, the intuition is that duplicate elimination is applied only to a subset of the possible worlds of the input, those for which all duplicates are "synchronized". We show po-relations continue to form a strong representation system for this semantics and that possibility and certainty may be decided in polynomial time. We contrast this with alternative semantics.

We next introduce basic definitions of classical concepts; related work is discussed in Section 8. Results are provided with complete proofs which are deferred to the appendix due to space constraints.

2. PRELIMINARIES

In this section, we introduce useful notions. We define our query language, a variant of the positive relational algebra, and recall its standard semantics on unordered data.

A *bag* or *multiset* over a set X is a function $B : X \to \mathbb{N}$. The *support* of a bag B is $B^{-1}(\mathbb{N}^+)$ and we write $x \in B$ if and only if $B(x) \neq 0$. We write a bag B with finite support as $B = \{\{b_1, \ldots, b_n\}\}$ where $n = \sum_{x \in X} B(x)$ is the *size* $|B|$ of B: for every $x \in X$, we have $B(x) = |\{1 \leqslant k \leqslant n \mid b_k = x\}|$. For any bag B and Boolean predicate φ on elements of X, the bag $B' = \{\{x \in B \mid \varphi(x)\}\}$ is the function that maps x to $B(x)$ if $\varphi(x)$ holds and maps x to 0 otherwise.

For any two bags B_1, B_2 over the same set X, $B_1 \uplus B_2$ is the bag over X defined by $x \mapsto B_1(x) + B_2(x)$. For any bag B over X and any function F from X to bags over X, $\biguplus_{x \in B} F(x)$ is the bag over X defined by $y \mapsto \sum_{x \in B} B(x) \cdot F(x)(y)$.

We fix a schema $\mathscr{S}$, which is a set of relation names along with their arities. Relations in the schema are defined over a countable set of values $\mathscr{D}$, in particular we assume $\mathbb{N} \subseteq \mathscr{D}$. A *tuple t over $\mathscr{D}$* of *arity* $\mathrm{a}(t) \in \mathbb{N}$ is an element of $\mathscr{D}^{\mathrm{a}(t)}$. The tuple itself is denoted $\langle v_1, \ldots, v_{\mathrm{a}(t)} \rangle$ (or occasionally simply v when the arity is 1); given two tuples t_1 and t_2, $\langle t_1, t_2 \rangle$ is the tuple of arity $\mathrm{a}(t_1) + \mathrm{a}(t_2)$ obtained by concatenating t_1 and t_2. A (bag) *relation R* over $\mathscr{D}$ is a *bag* of tuples over $\mathscr{D}$ that all have the same arity denoted $\mathrm{a}(R)$. Its *size*, or number of tuples, is the size $|R|$ of the bag. A (bag) *database D* is an instantiation of a finite subset of the relation names in $\mathscr{S}$, which we will sometimes view as a *mapping* from these relation names to relations; we impose that the relation mapped to a relation name has the arity imposed by the schema $\mathscr{S}$. We adopt the *unnamed perspective* where positions of tuples are indexed by integers, and write $t.k$ for the element of $\mathscr{D}$ at position k in a tuple t of arity $n \geqslant k$.

We then define the *Boolean formulas over tuples* that will be used for the selection operator:

DEFINITION 2.1. *A* tuple predicate *is a Boolean formula over atoms of the form ".m = .n" or ".$m = d$" where m,n are positive integers and $d \in \mathscr{D}$.*

A tuple predicate φ of the form ".m = .n" holds for a tuple t, *denoted $\varphi(t)$, if and only if $m \leqslant \mathrm{a}(t)$, $n \leqslant \mathrm{a}(t)$, and $t.m = t.n$. A tuple predicate φ of the form ".$m = d$"* holds for a tuple t, *denoted $\varphi(t)$, if and only if $m \leqslant \mathrm{a}(t)$ and $t.m = d$.*

The query language we consider is the positive fragment of the relational algebra [1], with the unnamed perspective. We now define its syntax:

DEFINITION 2.2. *We define inductively the language of* positive relational algebra *queries (or* PosRA*), over a countable set of constant expressions $\mathscr{C}$, as follows, where Q, Q_1, and Q_2 are PosRA queries:*

- *for any relation name $R \in \mathscr{S}$ or constant $C \in \mathscr{C}$, R and C are PosRA queries;*
- *for any tuple predicate φ, $\sigma_\varphi(Q)$ is a PosRA query of arity $\mathrm{a}(Q)$;*
- *for any sequence of (non-necessarily distinct) positive integers $(k_1, \ldots, k_p)$ with $k_i \leqslant \mathrm{a}(Q)$ for all i, $\Pi_{k_1 \ldots k_p}(Q)$ is a PosRA query of arity p;*
- *when $\mathrm{a}(Q_1) = \mathrm{a}(Q_2)$, $Q_1 \cup Q_2$ is a PosRA query of arity $\mathrm{a}(Q_1) = \mathrm{a}(Q_2)$;*
- *$Q_1 \times Q_2$ is a PosRA query of arity $\mathrm{a}(Q_1) + \mathrm{a}(Q_2)$.*

The set of relation names *of a PosRA query Q is the finite subset of relation names from $\mathscr{S}$ that occurs in Q.*

In what follows, we fix the constant expressions $\mathscr{C}$ to be:

restname	*district*
Gagnaire	8
TourArgent	5

(a) *Rest* table

restname	*district*
Tsukizi	6

(b) *Rest*$_2$ table

hotelname	*district*
Mercure	5
Balzac	8
Mercure	12

(c) *Hotel* table

Figure 1: Running Example

- $\emptyset$, representing the empty relation[1];
- $[t]$ for any tuple t over $\mathscr{D}$, of arity $\mathrm{a}(t)$, representing the singleton relation containing only tuple t;
- $\mathbb{N}^{\leqslant n}$ for $n \in \mathbb{N}$, of arity 1, representing the bag of the first $n+1$ natural integers.

We will mostly focus on queries under *bag semantics*, and address set semantics in Section 7. We recall the "classical" definition of bag semantics (see, e.g., [7]):

DEFINITION 2.3. *Let Q a PosRA query over $\mathscr{C}$ on relation names $\{R_1,\ldots,R_n\}$. Let D be a database over $\{R_1,\ldots,R_n\}$. The* evaluation of Q over D*, denoted $Q(D)$, is a relation over $\mathscr{D}$, according to the following inductive definition, where Q', Q_1, and Q_2 are PosRA queries:*

- *for any $1 \leqslant i \leqslant n$, R_i evaluates to $D(R_i)$;*
- *$\emptyset$ evaluates to the empty bag $\{\{\,\}\}$;*
- *for any tuple t, $[t]$ evaluates to the bag $\{\{t\}\}$;*
- *for any $n \in \mathbb{N}$, $\mathbb{N}^{\leqslant n}$ evaluates to $\{\{0,\ldots,n\}\}$;*
- *for any tuple predicate φ, $\sigma_\varphi(Q'(D))$ evaluates to the bag $\{\{t \in Q'(D) \mid \varphi(t)\}\}$;*
- *for any sequence of integers $(k_1,\ldots,k_p)$, $\Pi_{k_1\ldots k_p}(Q')$ evaluates to $\biguplus_{t\in Q'(D)}\{\{\langle t.k_1,\ldots,t.k_p\rangle\}\}$;*
- *if $S_1 := Q_1(D)$, $S_2 := Q_2(D)$, $(Q_1 \cup Q_2)(D) := S_1 \uplus S_2$;*
- *if $S_1 := Q_1(D)$, $S_2 := Q_2(D)$, $Q_1 \times Q_2$ evaluates to $\biguplus_{t_1\in S_1} \biguplus_{t_2\in S_2}\{\{\langle t_1,t_2\rangle\}\}$.*

In the following, we will always assume that queries are applied on databases of compatible schema.

3. ORDER-INCOMPLETE DATABASES

In this section, we introduce our notion of ordered and order-incomplete databases, the axioms that we impose on PosRA semantics for such databases, and our proposed semantics.

3.1 Totally Ordered Relations

Our first goal is to specify a semantics for the positive relational algebra on *ordered* relations, defined as follows.

DEFINITION 3.1. *A* totally ordered relation *L is a list of tuples over $\mathscr{D}$ with fixed arity $\mathrm{a}(L)$. We alternatively see L as a (bag) relation $\mathrm{Rel}(L)$ with a total order $<_L$ on its tuples. A* sublist *L' of a list L is a list of some of the tuples from L, in the order they appear in L.*

EXAMPLE 3.2. *Consider the two relations Rest and Hotel in Figure 1 whose tuples contain (simplified) information regarding restaurants and hotels in Paris, namely their name and district. We assume that customer ratings from a given travel Web site allow us to define an order on them, so that these relations are totally ordered from top to bottom. The column names are here to help readability but are not strictly speaking part of the schema, as we have adopted the unnamed perspective.*

We next consider semantics for the relational algebra, applied to totally ordered relations. First, we naturally define the ordered semantics of constants:

constants $\emptyset$ is the empty list $()$; for any tuple t, $[t]$ is the list (t); for any $n \in \mathbb{N}$, $\mathbb{N}^{\leqslant n}$ is the list $(0,\ldots,n)$.

Then, we define selection and projection on totally ordered relations in the natural way:

select For any tuple predicate φ, $\sigma_\varphi(L)$ is the sublist of tuples in L satisfying φ.

project For A a sequence of integers, $\Pi_A(L)$ is the list L' of the $\Pi_A(\{\{t\}\})$ for $t \in \mathrm{Rel}(L)$, in the same order $<_L$.

However, when we try to define the union or product of totally ordered relations, there is in general no natural way to represent the output as a totally ordered relation.

EXAMPLE 3.3. *Consider Rest and Rest$_2$ (Figure 1), two totally ordered relations describing different restaurants from different categories (French and International). We know nothing about the relative order of restaurants appearing in Rest and those appearing in Rest$_2$, and so there is no unique way to represent Rest$\cup$Rest$_2$ as a single totally ordered relation. Similarly, say that we join Rest and Hotel (based on location); there are multiple plausible ways to decide a relative order between pairs of restaurants and hotels.*

3.2 Order-Incomplete Relations

Intuitively, the semantics of queries involving union or product should allow multiple possible orders to be represented in the output. This motivates the notion of *order-incomplete relations*, which capture the orders through a *possible worlds* interpretation:

DEFINITION 3.4. *An* order-incomplete *relation (or, for short,* oi-relation*) is a set W of totally ordered relations called the* possible orders *of W, where the underlying relation $\mathrm{Rel}(L)$ of every $L \in W$ is the same, denoted $\mathrm{Rel}(W)$. If W consists of a single list L, then we say that the relation is* totally ordered*, and do not distinguish between the oi-relation $\{L\}$ and the totally ordered relation L. If W consists of all possible orders on $\mathrm{Rel}(L)$, then we say that W is* unordered*. For two oi-relations W and W', we write $W \subseteq W'$ to mean that every possible world of W is also a possible world of W'.*

An order-incomplete database *D, or oi-database, is a mapping from a finite subset of relation names from $\mathscr{S}$ to oi-relations. The underlying database $\mathrm{Rel}(D)$ of an oi-database is obtained by replacing each oi-relation W in D by its underlying relation $\mathrm{Rel}(W)$.*

Note that the definition of oi-relations does not impose a specific way to represent the possible orders. We will later consider an efficient way to do so. For now, however, we consider them abstractly as sets without worrying about the representation.

EXAMPLE 3.5. *In our running example, consider the following possible orders for the union of Rest and Rest$_2$, where we specify just the restaurant names for brevity: (*Tsukizi, Gagnaire, TourArgent*), (*Gagnaire, Tsukizi, TourArgent*), or (*Gagnaire, TourArgent, Tsukizi*). This set of worlds can be represented as an oi-relation.*

When defining the semantics of the positive relational algebra on oi-relations, we will impose two general principles. The first one is that the semantics that we define should always match the bag semantics in terms of values.

bag For every PosRA query Q and oi-database D, we require that $\mathrm{Rel}(Q(D)) = Q(\mathrm{Rel}(D))$ where Q on bag relations is applied according to the usual bag semantics for PosRA.

The second general principle is a standard requirement in the context of representation systems for incomplete information [17]. Namely, evaluating a query Q over an oi-database D should yield an oi-relation whose possible orders are the possible results of evaluating Q over the possible orders of D. More precisely:

consistency For any PosRA query Q and any oi-database $D = \{R_i \mapsto W_i \mid 1 \leqslant i \leqslant n\}$, we have:

[1]Formally, we have a distinct $\emptyset$ constant relation for every arity, but we will omit this distinction.

$$Q(D) = \bigcup_{L_1,\ldots,L_n \in W_1 \times \cdots \times W_n} Q(\{R_i \mapsto L_i \mid 1 \leqslant i \leqslant n\}).$$

Hence, our existing definition of projection and selection for totally ordered relations extends to oi-relations, and to define union and product it will suffice to define them on totally ordered relations.

Observe that sorting (such as the `ORDER BY` of SQL) and position-based selection (such as the `LIMIT` of SQL) are missing from the language. The former will be expressible using the existing PosRA operators for some of the semantics that we will introduce (see Section 5), and the latter will be studied in Section 6.

3.3 Union and Product

As opposed to projection and selection, union and product are operators which have multiple reasonable definitions for ordered relations. We will first impose reasonable desiderata on their behavior: neither union nor product change the order within their arguments, which means that if we extract from the union or product the individual arguments, they should be unchanged.

union Letting L_1, L_2 be two total orders and $W := L_1 \cup L_2$, all possible orders of W are interleavings of L_1 and L_2. Formally, for *every* possible world L of W, there is a partition of L into two disjoint sublists L'_1 and L'_2 such that $L_1 = L'_1$ and $L_2 = L'_2$.

product Letting $L_1 = (t_1^{(1)}, \ldots, t_{|L_1|}^{(1)})$ and $L_2 = (t_1^{(2)}, \ldots, t_{|L_2|}^{(2)})$ be two total orders and $W := L_1 \times L_2$, all possible orders of W can be decomposed, in a compatible fashion, in sublists that match L_1 on the attributes from L_1, and sublists that match L_2 on the attributes from L_2.
Formally, *every* possible world L of W has a one-to-one mapping $\psi : [\![1;|L_1|]\!] \times [\![1;|L_2|]\!] \to [\![1;|L_1| \cdot |L_2|]\!]$ such that for any $1 \leqslant k \leqslant |L_1|$, the sublist of L of positions $\{\psi(k,l) \mid 1 \leqslant l \leqslant |L_2|\}$ is $(\langle t_k^{(1)}, t_1^{(2)}\rangle, \ldots, \langle t_k^{(1)}, t_{|L_2|}^{(2)}\rangle)$; and for any $1 \leqslant l \leqslant |L_2|$, the sublist of L of positions $\{\psi(k,l) \mid 1 \leqslant k \leqslant |L_1|\}$ is $(\langle t_1^{(1)}, t_l^{(2)}\rangle, \ldots, \langle t_{|L_1|}^{(1)}, t_l^{(2)}\rangle)$.

We have now concluded the presentation of our axioms. In fact, the bag axiom is redundant given the other axioms:

PROPOSITION 3.6. *Axiom bag is implied by axioms constants, select, project, consistency, union, and product.*

We will thus see bag as a useful property implied by the axioms, rather than as an axiom itself. Our final set of axioms, denoted Ax, thus consists of constants, select, project, consistency, union, and product. We now claim:

PROPOSITION 3.7. *None of the axioms in Ax is implied by the others. In fact, if any of these axioms is discarded, there is a semantics consistent with the other axioms (and, except when removing axiom constants, with axiom bag), and a trivial query*[2] *which is not the identity over this semantics, namely one of: $\sigma_{\text{true}}(R)$, $R \cup \emptyset$, or $\Pi_{1\ldots n}(R \times \mathbb{N}^{\leqslant 0})$ assuming that the input relation R has arity n.*

This justifies that our set of axioms Ax is not redundant and that all axioms are helpful to forbid particularly unnatural choices of semantics. We now proceed to give concrete examples of semantics for union and product on oi-relations that satisfy Ax (in particular proving the consistency of Ax).

3.4 Concrete Semantics

We now define our semantics on totally ordered relations, extending them to oi-relations with axiom consistency.

[2]By "trivial" we mean that the query is the identity for the standard bag semantics. However, note that we usually do not want all trivial queries to be the identity for oi-relations (see details in appendix).

The `GEN` *semantics.* We define the *generic* union and product operators as follows:

generic union $L_1 \cup_{\texttt{GEN}} L_2$, with $\mathrm{a}(L_1) = \mathrm{a}(L_2)$, is the set of all *interleavings* of tuples from L_1 and L_2, i.e., *all* possible lists L' satisfying the criteria in the union axiom.

generic product $L_1 \times_{\texttt{GEN}} L_2$ is the set of *all* the lists L on $\{\{\langle t_1, t_2\rangle \mid t_1 \in L_1, t_2 \in L_2\}\}$ such that there are no tuples $t = \langle t_1, t_2\rangle$ and $t' = \langle t'_1, t'_2\rangle$ where t' strictly precedes t in L, but, for all i, t_i precedes or equals t'_i in L_i. (In this definition, the t_i's are tuples, not values.)

EXAMPLE 3.8. *The output of $Rest \cup_{\texttt{GEN}} Rest_2$ in our running example is the three possible orderings of the union that were presented in Example 3.5.*

The output of the product $Rest \times_{\texttt{GEN}} \sigma_{.2 \neq 12}(Hotel)$ is an oi-relation with two possible total orders:

$$(\langle \mathrm{G}, 8, \mathrm{M}, 5\rangle, \langle \mathrm{G}, 8, \mathrm{B}, 8\rangle, \langle \mathrm{TA}, 5, \mathrm{M}, 5\rangle, \langle \mathrm{TA}, 5, \mathrm{B}, 8\rangle),$$
$$(\langle \mathrm{G}, 8, \mathrm{M}, 5\rangle, \langle \mathrm{TA}, 5, \mathrm{M}, 5\rangle, \langle \mathrm{G}, 8, \mathrm{B}, 8\rangle, \langle \mathrm{TA}, 5, \mathrm{B}, 8\rangle).$$

The order on the product relations is in a sense the minimal order on pairs of hotels and restaurants that is consistent *with the order on the individual lists: we do not know how to order two pairs, except that they are comparable whenever both their hotels and their restaurants are comparable.*

Note that consequently the result of $Rest \bowtie Hotel$ (implemented as product followed by selection based on equality on district and by projecting out the extra district) includes both orders $(\langle \mathrm{G}, \mathrm{B}, 8\rangle, \langle \mathrm{TA}, \mathrm{M}, 5\rangle)$ *and* $(\langle \mathrm{TA}, \mathrm{M}, 5\rangle, \langle \mathrm{G}, \mathrm{B}, 8\rangle)$*, intuitively reflecting two different opinions regarding the ordering of pairs of restaurant and hotel in the same district.*

We call the semantics of PosRA on oi-relations that uses $\cup_{\texttt{GEN}}$ for union and $\times_{\texttt{GEN}}$ for product *the* `GEN` *semantics*. We show that this semantics satisfies our axioms:

PROPOSITION 3.9. *The* `GEN` *semantics satisfies Ax.*

We also observe that the `GEN` semantics is *the most general semantics* satisfying Ax, in the following sense:

PROPOSITION 3.10. *For any PosRA query Q and any oi-database D, letting W be the result of evaluating Q on D under the* `GEN` *semantics and W' be the result under a different semantics satisfying Ax, we have: $W' \subseteq W$.*

Lexicographic product. Another natural semantics that can be chosen for the product of ordered relations is that of *lexical ordering*, which is defined as follows:

lexicographic product $L_1 \times_{\texttt{LEX}} L_2$ is *the* list (a) consisting of all tuples $\langle t_1, t_2\rangle$ such that $t_1 \in L_1$ and $t_2 \in L_2$ and (b) where $\langle t_1, t_2\rangle$ precedes $\langle t'_1, t'_2\rangle$ if either $t_1 <_{L_1} t'_1$ or $t_1 =_{L_1} t'_1$ and $t_2 \leqslant_{L_2} t'_2$. (Again, in this definition, the t_i's are tuples, not values.)

EXAMPLE 3.11. *The $\times_{\texttt{LEX}}$ semantics enforces a lexicographic order on tuples in the result of a product. Hence, for instance, in our running example, the join $\Pi_{1,3,4}(\sigma_{.2=.4}(Rest \times_{\texttt{LEX}} \sigma_{.2 \neq 12}(Hotel)))$ includes only a single possible world; it is ordered according to the leftmost operand in the product, i.e.,, Rest:* $(\langle \mathrm{Gagnaire}, \mathrm{Balzac}, 8\rangle, \langle \mathrm{TourArgent}, \mathrm{Mercure}, 5\rangle)$.

It is easy to see that $\times_{\texttt{LEX}}$ satisfies our desiderata:

PROPOSITION 3.12. $\times_{\texttt{LEX}}$ *satisfies product.*

Concatenation. Another natural semantics for union is that of concatenation:

concatenation union $L_1 \cup_{\texttt{CAT}} L_2$, with $\mathrm{a}(L_1) = \mathrm{a}(L_2)$, is *the* set formed of the single list where all tuples of L_1 (in order) come before those of L_2 (in order).

It turns out that this useful semantics (that can easily be shown to satisfy the union axiom) can be simulated with $\cup_{\texttt{GEN}}$ and $\times_{\texttt{LEX}}$:

REMARK 3.13. *For oi-relations W_1 and W_2, $W_1 \cup_{\texttt{CAT}} W_2 = \Pi_{3\ldots n+2}\left(\sigma_{.1=.2}\left(\mathbb{N}^{\leqslant 1} \times_{\texttt{LEX}} (([0] \times_{\texttt{LEX}} W_1) \cup_{\texttt{GEN}} ([1] \times_{\texttt{GEN}} W_2))\right)\right)$ where $n = \mathrm{a}(W_1) = \mathrm{a}(W_2)$.*

Consequently, in the following we will mostly use the following overall semantics for PosRA queries, all of which satisfy Ax: `GEN`, `LEX` (with $\cup_{\texttt{GEN}}$ for the semantics of union and $\times_{\texttt{LEX}}$ for the semantics of product, and where $\cup_{\texttt{CAT}}$ can indirectly be expressed), and `GEN+LEX` (with both semantics for product allowed in different parts of the query).

3.5 Additional Properties

We have axiomatized our semantics for PosRA on totally ordered and order-incomplete relations, using a set of axioms that we showed to be non-redundant (Proposition 3.7), except for axiom bag (Proposition 3.6).

We now give more insight about these axioms and our semantics by presenting some *additional properties* which are standard for the positive relational algebra, and showing if they are consequences of the axioms, or if they are satisfied by the `LEX` and `GEN` semantics.

We consider the following properties, defined for all oi-relations W_1, W_2, W_3, tuple predicate φ, and sequences K_1 and K_2 of integers in $\{1,\ldots,\mathrm{a}(W_1)\}$ and $\{1,\ldots,\mathrm{a}(W_2)\}$ respectively:

union-associative $W_1 \cup (W_2 \cup W_3) = (W_1 \cup W_2) \cup W_3$

union-commutative $W_1 \cup W_2 = W_2 \cup W_1$

union-distributive $(W_1 \cup W_2) \times W_3 = (W_1 \times W_3) \cup (W_2 \times W_3)$ and $W_3 \times (W_1 \cup W_2) = (W_3 \times W_1) \cup (W_3 \times W_2)$

product-associative $W_1 \times (W_2 \times W_3) = (W_1 \times W_2) \times W_3$

selection-product $\sigma_\varphi(W_1 \times W_2) = W_1 \times \sigma_{\varphi'}(W_2)$ where all atoms of φ are of the form "$.m = .n$" or "$.m = d$" with $m > \mathrm{a}(W_1)$, $n > \mathrm{a}(W_1)$ and φ' is like φ except that all occurrences of $.m$ or $.n$ are replaced with $.(m - \mathrm{a}(W_1))$ or $.(n - \mathrm{a}(W_1))$; similarly, $\sigma_\varphi(W_1 \times W_2) = \sigma_\varphi(W_1) \times W_2$ where all atoms of φ are of the form "$.m = .n$" or "$.m = d$" with $m \leqslant \mathrm{a}(W_1)$, $n \leqslant \mathrm{a}(W_1)$

selection-projection $\Pi_{K_1}(\sigma_\varphi(W_1)) = \sigma_{\varphi'}(\Pi_{K_1}(W_1))$ if all atoms of φ are of the form "$.m = .n$" or "$.m = d$" with $m, n \in K_1$ and φ' is defined as φ except that all occurrences of $.m$ or $.n$ are replaced by some position, in K_1, where m or n occurs

projection-union $\Pi_{K_1}(W_1 \cup W_2) = \Pi_{K_1}(W_1) \cup \Pi_{K_1}(W_2)$

projection-product $\Pi_{1,\ldots,\mathrm{a}(W_1),K_2'}(W_1 \times W_2) = W_1 \times \Pi_{K_2}(W_2)$ where K_2' is as K_2 but replacing every k with $k + \mathrm{a}(W_1)$; similarly, $\Pi_{K_1,\mathrm{a}(W_1)+1,\ldots,\mathrm{a}(W_1)+\mathrm{a}(W_2)}(W_1 \times W_2) = \Pi_{K_1}(W_1) \times W_2$

Another property of a different nature, that is useful to have, is that the order of query results only depend of existing order, not on actual tuples values:

value-genericity For any Q and any oi-database D, letting λ be a bijective relabeling (i.e., a one-to-one mapping from $\mathscr{D}$ onto itself extended to an operation over oi-relations and oi-databases), then we have: $\lambda(Q(D)) = \lambda(Q)(\lambda(D))$, where $\lambda(Q)$ applies the relabeling to constant relations and values within selections.

We show that these additional properties hold for `GEN`, and that some of them hold for `LEX` as well:

PROPOSITION 3.14. *The following properties hold:*

(i) selection-projection *in any semantics that satisfies* Ax*;*
(ii) union-associative, union-commutative, *as well as* projection-union *for* $\cup_{\texttt{GEN}}$*;*
(iii) product-associative, selection-product, *as well as* projection-product *for* $\times_{\texttt{GEN}}$ *and* $\times_{\texttt{LEX}}$*;*
(iv) union-distributive *in* `GEN` *but not in* `LEX`*;*
(v) value-genericity *in* `GEN+LEX`.

Other semantics for union and product may not satisfy the same properties; for instance, it is clear that $\cup_{\texttt{CAT}}$ does not satisfy axiom union-commutative; value-genericity may also be violated (see details in appendix).

4. PARTIALLY-ORDERED DATABASES

While order-incomplete relations are useful as an abstract way to describe the semantics of PosRA, they are not a practical way to represent query evaluation results. Indeed, if we try to enumerate all the possible worlds of the oi-relation that represents the result of evaluating a query, we find that they can be far too numerous, even for a very simple query:

PROPOSITION 4.1. *The number of possible worlds of $R \cup_{\texttt{GEN}} S$ applied to totally ordered relations R and S can be exponential in the number of tuples of the input relations.*

This means that, if we were to represent explicitly the collection of possible worlds of oi-relations, query evaluation would be exponential in data complexity, which is clearly undesirable. Thus, in this section, we propose a natural representation of oi-relations through *partially ordered relations*, which (as we will show) can be used as a strong representation system for the `GEN+LEX` semantics.

DEFINITION 4.2. *A* partially ordered relation *(*po-relation*) over tuples of arity $n \in \mathbb{N}$ is a triple $R = (ID, T, <)$, where ID is a finite set of identifiers, T is a mapping of identifiers in ID to tuples of arity n, and $<$ is a* partial *order over ID. If $<$ is empty (i.e., imposes no order constraints), we say the po-relation is* unordered*. If $<$ is a total order, we say the po-relation is* totally ordered*.*

A partially ordered database *(or* po-database*) is a mapping from a finite subset of relation names in $\mathscr{S}$ to po-relations whose sets of tuple identifiers are pairwise disjoint. We say a po-database is* unordered *or* totally ordered *if all of its po-relations are.*

We now define the semantics of po-relations by constructing the oi-relation that each po-relation captures. To do so, we need to define more terminology about partial orders:

DEFINITION 4.3. *A* poset*, or partially ordered set, is a pair $P = (V, <)$ where $<$ is a partial order over a finite*[3] *set V; its* domain $\mathrm{dom}(P)$ *is the set V. For a po-relation $(ID, T, <)$, we say that $(ID, <)$ is its* underlying poset*.*

A linear extension *of a poset $(V, <)$ is a total order $(V, <')$ such that for all $x, y \in V$, if $x < y$ then $x <' y$.*

EXAMPLE 4.4. *Continuing our running example, the oi-relation of Example 3.8 obtained by evaluating Rest $\times_{\texttt{GEN}} \sigma_{.2\neq 12}$(Hotels) can be captured by the po-relation with 4 tuples represented by the Hasse diagram to the right: the order is represented from bottom to top, a path from a tuple t_1 below to a tuple t_2 above indicating that $t_1 \leqslant t_2$.*

$$\begin{tikzcd}
& (\mathrm{TA},5,\mathrm{B},8) & \\
(\mathrm{G},8,\mathrm{B},8) \arrow[ur] & & (\mathrm{TA},5,\mathrm{M},5) \arrow[ul] \\
& (\mathrm{G},8,\mathrm{M},5) \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

We can now define the oi-relation captured by a po-relation:

[3] We only consider finite posets.

DEFINITION 4.5. *A* possible world *of a partially ordered relation $R = (ID, T, <)$ is a po-relation $(ID, T, <')$ such that $(ID, <')$ is a linear extension of $(ID, <)$. We identify $(ID, T, <')$ with the corresponding totally ordered relation.*

The oi-relation captured by R, written $pw(R)$, is the set of its possible worlds. Similarly, for a po-database D, $pw(D)$ is the set of possible worlds obtained by picking a possible world for each relation of D.

It is then natural to ask whether *any* oi-relation can be captured by a po-relation, but the answer is negative:

EXAMPLE 4.6. *Consider the three hotel tuples from Figure 1, referred to, in order, as t_1, t_2, t_3. Let W be the oi-relation $\{(t_2, t_1, t_3), (t_3, t_1, t_2)\}$. This relation represents the hotel preferences of a user who plans to attend a conference in Paris but where it is not known yet whether it will take place in a Western or Eastern district (so the central 5th district of t_1 is always a compromise between the Western and Eastern t_2 and t_3).*

Now, assume by contradiction that there is a po-relation R such that $W = pw(R)$. No comparability pair can hold in the partial order of R, as for any two elements $x \neq y$ (for $x, y \in \{t_1, t_2, t_3\}$), there is one linear extension of W such that $x < y$ holds and one such that $y < x$ holds. Yet if R is unordered, then $pw(R)$ includes, e.g., the possible world $(t_1, t_2, t_3) \notin W$.

Are po-relations still useful, then? In fact, we now show that they form a strong representation system for the positive relational algebra under the full GEN+LEX semantics. In other words, starting with input oi-relations represented as po-relations, the output oi-relation is always representable as a po-relation. Furthermore, the output po-relation can be computed efficiently from the input in data complexity, so that it is practical to work with po-relations (rather than oi-relations).

THEOREM 4.7. *Let Q be a fixed PosRA query. Given a po-database D, we can compute in polynomial time a po-relation $Q(D)$ such that $pw(Q(D)) = Q(pw(D))$ under the* GEN+LEX *semantics.*

This important result will serve as the basis of our analysis of the expressiveness of PosRA, and of top-k query evaluation, in the following sections.

5. EXPRESSIVENESS

We now discuss the expressiveness of the proposed semantics for PosRA: what can be expressed, in terms of output or transformations, depending on the allowed inputs?

We first study which po-relations can be obtained as the output of queries, and show that any po-relation can be obtained as the result of some GEN query, even if the query is restricted to have no input, i.e. it only uses constant relations (Proposition 5.3). Likewise, any *series-parallel po-relation* can be obtained as the result of a LEX query with no input (Proposition 5.6). Conversely, we have already shown that, when the inputs are po-relations, the output of GEN queries is always a po-relation (Theorem 4.7). For LEX, we further show that if the underlying posets of the input relations are all series-parallel then the same applies to the query output.

We then show that the expressiveness of queries is further limited by the query size, in the sense that the size of GEN queries without inputs can be used to bound the *dimension* of the resulting partial orders (Proposition 5.7), and the size of LEX queries can be used to bound the *sp-height* of the resulting series-parallel partial orders, depending on that of the input partial orders (Proposition 5.9).

We next study the transformations that can be expressed by GEN and LEX, and show that the two semantics are incomparable in this sense (Corollary 5.14): the outputs of LEX are of a restricted form, but GEN cannot express the concatenation operator $\cup_{\texttt{CAT}}$ whereas LEX can (which we use to show that LEX can also express a *sort* operator). We conclude by giving limits on the expressiveness of *any* semantics satisfying our axioms: all expressed transformations must be monotone (Proposition 5.19) but some monotone transformations cannot be expressed (Proposition 5.20).

For the expressiveness results, we will need to formalize a notion of *isomorphism* on posets, and po-relations:

DEFINITION 5.1. *An* isomorphism *between posets $(P_1, <_1)$ and $(P_2, <_2)$ is a bijection $f : P_1 \to P_2$ which preserves order in both directions: for any $x, y \in P_1$, $x <_1 y$ iff $f(x) <_2 f(y)$.*

An isomorphism *between po-relations $R_1 = (ID_1, T_1, <_1)$ and $R_2 = (ID_2, T_2, <_2)$ is an isomorphism $f : ID_1 \to ID_2$ from the underlying poset $(ID_1, <_1)$ of R_1 to the underlying poset $(ID_2, <_2)$ of R_2, which additionally preserves values: for all $x \in ID_1$, $T_1(x) = T_2(f(x))$.*

We also need the notion of *realizer* and *dimension* of a partial order:

DEFINITION 5.2. *Letting $(P, <)$ be a poset, we say that a set of total orders $(P, <_1), \ldots, (P, <_n)$ is a* realizer *of $(P, <)$ if for every $x, y \in P$, $x < y$ iff $x <_i y$ for all i. The* dimension *[10] of P is the smallest n such that there exists a set of n total orders that realizes P.*

5.1 Possible Outputs

We start by studying the LEX and GEN semantics in terms of the possible relations that can be obtained as output. As we construct the query as a function of the desired output (rather than fixing the query and studying the transformation that it expresses), it will actually be sufficient to study queries with *no* inputs.

GEN *semantics.* We show that, following the GEN semantics, any po-relation can be obtained as output, even when no input is provided (a query with no input is a query where all relations are the constant expressions introduced in Section 3). This means that GEN, even with no inputs, matches the upper bound in expressive power implied by Theorem 4.7 (namely, that the output to queries taking po-relations as inputs will be po-relations).

PROPOSITION 5.3. *For any po-relation R, there is a PosRA query Q with no inputs such that the result of evaluating Q using the* GEN *semantics is R.*

LEX *semantics.* In contrast, the output of the LEX semantics is of a restricted kind, as we now explain. We say that a po-relation $R = (ID, T, <_R)$ is *series-parallel* if $(ID, <_R)$ is a series-parallel poset [32], defined as follows:

DEFINITION 5.4. *The class of* series-parallel posets *is the class including all single-element orders and closed under the* series *and* parallel *composition operations. Given two series-parallel posets P and Q with disjoint domains, the* series *composition of P and Q is the poset on $\text{dom}(P) \sqcup \text{dom}(Q)$ whose restriction to $\text{dom}(P)$ and $\text{dom}(Q)$ matches P and Q, and such that $p < q$ for any $p \in \text{dom}(P)$ and $q \in \text{dom}(Q)$. The* parallel *composition of P and Q is the poset on $\text{dom}(P) \sqcup \text{dom}(Q)$ whose restriction to $\text{dom}(P)$ and $\text{dom}(Q)$ matches P and Q, and such that no $p \in \text{dom}(P)$ and $q \in \text{dom}(Q)$ are comparable. A po-relation is* series-parallel *if its underlying poset is.*

We first show that any series-parallel po-relation can be obtained as output to a LEX query with no input relations:

PROPOSITION 5.5. *For any series-parallel po-relation R, there exists a PosRA query Q with no inputs such that the result of evaluat-*

ing Q under the `LEX` *semantics is R.*

We then show that the output of queries under the `LEX` semantics is always series-parallel (or empty, i.e., has no tuples), if the input po-relations are also series-parallel (in particular when there is no input).

PROPOSITION 5.6. *(Follows from [15].) For any query Q and po-database D of series-parallel po-relations, $Q(D)$ under the* `LEX` *semantics is either series-parallel or empty.*

Bounded query size. We have shown that `GEN` queries can have arbitrary po-relations as output, and that `LEX` queries can have arbitrary series-parallel po-relations as output. However, the construction builds queries whose size depends on the desired output po-relation.

We now show that, by contrast, if there is a bound on the size of the queries, then not all po-relations can be captured as output. Of course, this is not very surprising if no input relations are allowed (because there are only finitely many possible such queries, and infinitely many possible outputs). Yet we will show that, if the input relations are of restricted form, there are expressiveness limitations on query results no matter the size of the input relations.

We formalize this by showing bounds on certain poset complexity measures (poset dimension, sp-height of series-parallel posets) when the query size is bounded.

We can use poset dimension as a bound on the possible po-relations that can be obtained as output to a fixed query. Of course, we cannot obtain any bound if the input po-relations are allowed to be arbitrary (as the identity query could then have arbitrary outputs). We accordingly restrict to input po-relations that are unions of totally ordered relations. This is a natural restriction in our motivating scenarios where ordered data is combined, as the original will usually be totally ordered. We then have the following bound:

PROPOSITION 5.7. *For any $k \geqslant 1$, for any fixed query Q with no more than k product signs, for any input po-database D where each po-relation is a union of totally ordered relations, the dimension of the underlying poset of $Q(D)$ (under the* `GEN` *semantics) is at most $(k+1)$.*

Conversely, for any k-dimensional poset $(P,<)$ there exists a query Q with no more than k product signs, and a po-database D of totally ordered po-relations such that the underlying poset of $Q(D)$ (under the `GEN` *semantics) is $(P,<)$.*

In contrast, note that for the `LEX` semantics, by Proposition 5.6, even when allowing arbitrary series-parallel po-relations as input, the underlying poset of the query output is always series-parallel and thus has dimension at most 2 (by Proposition 2.2 of [27]).

We can show a result for `LEX` similar to the one above, using a different notion of complexity for series-parallel posets derived from the sp-height of an tree to construct this poset, called the *sp-tree*.

DEFINITION 5.8. *An* sp-tree *[5] is a rooted ordered tree whose internal nodes are labeled either "series" or "parallel", and leaf nodes are labeled with "singleton". The* decoding *of an sp-tree is a series-parallel poset (defined up to isomorphism) obtained in the following fashion:*

- *the decoding of a "singleton" node is the poset $(\{s\},\emptyset)$ where s is a fresh element;*
- *the decoding of a "series" node is the series composition of the posets obtained as the decoding of the children of this node, in the order in which they appear;*
- *likewise, the decoding of a "parallel" node is the parallel composition of the decoding of the children.*

We define the height *of an sp-tree in the usual manner. The* sp-height *of a series-parallel poset $(P,<)$ is the minimal height of an sp-tree that decodes to $(P,<)$ up to isomorphism.*

PROPOSITION 5.9. *For any $k \in \mathbb{N}$ and query Q, there is $k' \in \mathbb{N}$ (depending only on k and Q) such that for any po-database D of series-parallel po-relations of sp-height at most k, the underlying poset of $Q(D)$ (under the* `LEX` *semantics) is series-parallel with sp-height at most k', or empty.*

Of course, this result also applies if the input po-relations are series-parallel posets of a restricted form, e.g., totally ordered relations or unions thereof.

5.2 Possible Transformations

We now study the `LEX` and `GEN` semantics in terms of the possible *transformations* they can express on their input relations, leveraging results obtained thus far to show that they are incomparable. We limit our study to the case of input oi-relations which are po-relations, so that again the output oi-relations are po-relations by Theorem 4.7.

DEFINITION 5.10. *A* transformation *is defined as a mapping from po-databases to po-relations. A query Q* expresses *a transformation f (under semantics* `X`*) if, for any po-database D, $Q(D)$ (under semantics* `X`*) is isomorphic to $f(D)$.*

We note that, as long as value-genericity holds, queries can only express generic transformations, in the following sense. A transformation is *generic* except for values $\mathscr{V} \subseteq \mathscr{D}$ if, for any mapping $\nu : \mathscr{D} \to \mathscr{D}$ which is the identity on $\mathscr{V}$, and any po-database D, we have $f(\nu(D)) = \nu(f(D))$. A query Q may only express transformations that are generic except for values that occur in Q (in constant relations, or in selection predicates).

Comparing `LEX` *and* `GEN`. We may now compare the expressive power of `LEX` and `GEN` in terms of the transformations that they define on po-relations. It is a straightforward consequence of Proposition 5.6 that:

COROLLARY 5.11. *There are transformations expressible in* `GEN` *but not in* `LEX`.

Conversely, we show that `GEN` does not capture `LEX`, by showing that the concatenation operator $\cup_{\mathtt{CAT}}$ cannot be expressed in the `GEN` semantics:

PROPOSITION 5.12. *For any distinguished relation names R and S, there is no query Q such that, for any po-database D, $Q(D)$ evaluates to $R \cup_{\mathtt{CAT}} S$ under the* `GEN` *semantics.*

By contrast, $\cup_{\mathtt{CAT}}$ could be expressed in `LEX`, as explained in Remark 3.13. This immediately implies:

COROLLARY 5.13. *There are transformations expressible in* `LEX` *but not in* `GEN`.

Hence, from Corollaries 5.11 and 5.13, we observe that `GEN+LEX` is strictly more expressive than `LEX` or than `GEN`:

COROLLARY 5.14. *The* `GEN` *and* `LEX` *semantics are incomparable in terms of expressive power for transformations.*

Sorting. From the previous discussion, we observe that, as `LEX` can encode $\cup_{\mathtt{CAT}}$, it can also encode a *sort* operator, defined as follows:

DEFINITION 5.15. *For any integer $1 \leqslant i \leqslant \mathrm{a}(L)$, domain $U \subseteq \mathscr{D}$ and total order $<_U$ on U, we define the* sort operator *$\mathrm{Sort}_{i,<_U}$ as follows: for any totally ordered relation L such that $\Pi_i(L) \subseteq U$, $\mathrm{Sort}_{i,<_U}(L)$ is the totally ordered relation consisting of the tuples* *of L sorted according to the order $<_U$ on their values at position i, keeping the existing order of L between tuples which have the same value at position i (hence, this is a* stable *sort). Note that* $\mathrm{Sort}_{i,<_U}(R)$ *is undefined if R contains a tuple t such that $t.i \notin U$. We extend* $\mathrm{Sort}_{i,<_U}$ *to oi-relations using axiom* consistency *as usual.*

When U is finite, we say that $\mathrm{Sort}_{i,<_U}$ *is a* finite sort.

We now show that finite sorts can be expressed in LEX and directly encoded as a query:

PROPOSITION 5.16. *For any finite sort* $\mathrm{Sort}_{i,<_U}$, *there is a PosRA query Q with distinguished relation name R such that, for any po-database D, $Q(D)$ under the* LEX *semantics evaluates to* $\mathrm{Sort}_{i,<_U}(R)$ *when it is defined.*

But sorting is impossible in GEN by Proposition 5.12:

COROLLARY 5.17. *For any domain U of size $\geqslant 2$ and total order $<_U$ on U, for any distinguished relation name R with arity $n \geqslant 2$ and position $1 \leqslant i \leqslant n$, there is no query Q such that, for any po-database D, $Q(D)$ under the* GEN *semantics evaluates to* $\mathrm{Sort}_{i,<_U}(R)$ *when it is defined.*

Proposition 5.16 does not extend to infinite sort because the order on the infinite domain needs to be specified in some way. The result can be extended to any infinite sort if we assume that an infinite totally ordered relation is provided in the input po-database to represent the order on which to sort.

Intrinsic limits. Our expressiveness results so far were limited to the GEN and LEX semantics. However, while those two semantics are natural, our axioms Ax also allow different semantics, leaving open the question of which transformations can be expressed by them. We now present expressiveness limits for *any* PosRA semantics that satisfies Ax (in particular GEN+LEX). Formally, we show that every such semantics captures a strict subset of the monotone PTIME transformations on po-relations, and give an example of a natural transformation which is not captured.

Clearly the transformations are PTIME by Theorem 4.7. We now show that semantics that satisfy Ax are *monotone*:

DEFINITION 5.18. *A transformation f is* monotone *if for every po-databases D and D' we have $pw(f(D)) \subseteq pw(f(D'))$ whenever $pw(R_D) \subseteq pw(R_{D'})$ for every relation name R (R_D and $R_{D'}$ being the relations for R in D and D' respectively).*

PROPOSITION 5.19. *For any semantics* X *satisfying Ax and PosRA query Q, if a transformation f is expressed by Q under semantics* X, *then f is monotone.*

This is a coarse restriction, of course, and it is not at all the case that our semantics can capture *any* monotone query. We leave precise characterization of expressiveness for future work, but, for instance, we cannot "forget" input order:

PROPOSITION 5.20. *There is no semantics* X *satisfying Ax and query Q that capture the following monotone, generic transformation f: letting R be a distinguished relation name, for any po-database D, $f(D) = \mathrm{Rel}(R)$.*

6. TOP-*K*

The query language that we have studied so far does not allow selection based on the order on tuples. In this section we allow such selection by introducing a top-k operator, for which we will study possibility and certainty. As we will see at the end of the section, these problems generalize the natural questions of deciding instance certainty and instance possibility for po-relations (see Definition 6.16), for our possible worlds semantics.

6.1 Definition and General Results

As usual, we first introduce the semantics of top_k for totally ordered relations:

DEFINITION 6.1. *For every $k \in \mathbb{N}$ we introduce a unary operator* top_k *with the following semantics: for every totally ordered relation L,* $\mathrm{top}_k(L)$ *is the totally ordered relation which is the sublist of L of positions from 1 to $\min(k, |L|)$.*

We then use axiom consistency to extend top_k to work on oi-relations and po-relations. We note that oi-relations cannot represent the result of the top_k operator when the input relation is not totally ordered:

EXAMPLE 6.2. *Consider the query with no inputs: $Q = \mathrm{top}_1([a] \cup [b])$ for $a, b \in \mathscr{D}$. As the possible worlds of $[a] \cup [b]$ are (a, b) and (b, a), the possible query results should be (a) and (b). However, this is not representable by an oi- or po-relation, because the domains of the two results differ.*

We thus restrict our attention to cases where top_k is applied as the *last* operation of the query. In this case, we can show that, for fixed k:

PROPOSITION 6.3. *For any fixed query $Q := \mathrm{top}_k(Q')$ with Q' in the* GEN+LEX *semantics, one can compute the possible results of $Q(D)$ in PTIME in the input po-database D.*

However, the complexity may be exponential in the value of k (as well as in the size of Q). In this section, we study the feasibility of query evaluation when the top-k operator is added as a last operation and k is *not* fixed in the query. As the number of possible top-k results may then be exponential in k, we consider instead the two following *possibility* and *certainty* problems for top-k with unbounded:k:

DEFINITION 6.4. *For a PosRA query Q evaluated under semantics* X, *given an input po-database D, and a totally ordered po-relation L (called the* candidate possible world*), the* top-k possibility *problem asks whether there is a possible world of $Q(D)$ whose $|L|$ first elements are exactly L; in other words, whether L is a possible world of* $\mathrm{top}_{|L|}(Q(D))$.

The top-k certainty *problem is to determine whether* all *possible worlds of $Q(D)$ satisfy this condition.*

In our example application, if we assume a fixed query Q that integrates po-relations representing user preference on hotels, and whose output are names of hotel chains (with possible duplicates), the top-k possibility problem, given a po-database D representing the input hotels and a list L of hotel chain names, asks whether L can be the first elements of a possible world of $Q(D)$. For instance, we ask whether it is possible that all the top-3 hotels are "Mercure" hotels.

We now study the top-k possibility and certainty problems for the LEX, GEN and GEN+LEX semantics. Our results are summarized in Table 1. We always study *data complexity*: the query Q is fixed, and the input is the po-database D and the candidate possible world L (and we look at top-k for $k = |L|$, so k is not fixed).

The following results show that, in general, the certainty problem is in PTIME, but the possibility problem is NP-complete, even if we assume that all relations of the input po-database D are either unordered or totally ordered. Certainty is decided following a decomposition of the query result in a series composition, and the hardness of possibility is by a reduction from unary-3-partition.

THEOREM 6.5. *Top-k certainty is in PTIME for* GEN+LEX.

THEOREM 6.6. *Top-k possibility is NP for* GEN+LEX *and NP-hard for both* GEN *and* LEX, *even assuming that each input po-relation is either unordered or totally ordered.*

6.2 Tractability for Possibility

As we have shown that top-k possibility is NP-hard, we now show some restrictions that ensure tractability.

Unordered case. For any query Q under the GEN or LEX semantics, if we only allow unordered relations in the input po-database, then the top-k possibility problem can be solved in PTIME. This is fairly intuitive but is not entirely obvious, because order can be imposed on the input relations using constants in the query. However, we show that the resulting order is always of a certain restricted form, so that possibility is still in PTIME.

DEFINITION 6.7. *Given a poset $(P,<)$, a subset $S \subseteq P$ is an* undistinguishable antichain *if it is both an* antichain *(there are no $x,y \in S$ such that $x<y$) and it is an* undistinguishable set *(or* interval *[13]): for all $x,y \in S$ and $z \in P \backslash S$, $x<z$ iff $y<z$, and $z<x$ iff $z<y$.*

An undistinguishable antichain partition *(ua-partition) of a poset is a partition of its domain into undistinguishable antichains. A poset always has at least one partition, namely, the one where each element is put in its own partition. The* cardinality *of such a partition is its number of classes.*

PROPOSITION 6.8. *For any poset $(P,<)$, an ua-partition of minimal cardinality can be computed in PTIME.*

Accordingly, we define the *ua-width* of a poset as the cardinality of its smallest ua-partition.

PROPOSITION 6.9. *For any constant c, top-k possibility is in PTIME for the* GEN+LEX *semantics if we assume that all input po-relations have ua-width at most c.*

COROLLARY 6.10. *Top-k possibility is in PTIME for the* GEN+LEX *semantics if all input po-relations are unordered.*

Totally ordered case. We next consider the case where all input po-relations are totally ordered, which, as we already mentioned, is a natural restriction in our motivating scenarios. We show that in this case, top-k possibility remains NP-hard for GEN.

PROPOSITION 6.11. *Top-k possibility is NP-hard for the* GEN *semantics even if all input relations are assumed to be totally ordered.*

However, possibility becomes tractable for LEX under this hypothesis. In fact, we can show a stronger claim, with a suitable definition:

DEFINITION 6.12. *The* breadth *of an sp-tree is its number of topmost nodes that have no "parallel" descendants (and are not themselves "parallel").*

PROPOSITION 6.13. *For any constant $c \in \mathbb{N}$, top-k possibility is in PTIME for the* LEX *semantics if all input po-relations are series-parallel and each of their underlying posets has an sp-tree with breadth at most c.*

COROLLARY 6.14. *Top-k possibility is in PTIME for the* LEX *semantics if all input po-relations are totally ordered.*

No duplicates. A last way to ensure tractability is to assume that tuple values in the query result are all unique (with no duplicates). Intuitively, this makes it easy to decide which tuple from the candidate possible world should be matched to which tuple in the query result, because the unique tuple values ensure that only one match must be considered.

PROPOSITION 6.15. *Top-k possibility is in PTIME for the* GEN+LEX *semantics if we assume that all tuple values in $Q(D)$ are unique.*

Table 1: Summary of complexity results for top-k

Problem	Product	Input	Complexity	
Cert.	GEN+LEX	any	PTIME	(Thm. 6.5)
Poss.	GEN+LEX	any	NP	(Thm. 6.6)
Poss.[1]	GEN+LEX	any	PTIME	(Prop. 6.3)
Poss.[2]	GEN+LEX	any	PTIME	(Prop. 6.15)
Poss.	LEX	total	PTIME	(Cor. 6.14)
Poss.	GEN+LEX	unordered	PTIME	(Cor. 6.10)
Poss.	LEX	total+unordered	NP-hard	(Thm. 6.6)
Poss.	GEN	total	NP-hard	(Prop. 6.11)

[1] Assuming k is bounded
[2] Assuming no duplicate values in the query output

6.3 Other Problems

We now discuss some other natural questions that can be asked about a po-relation, but that cannot be expressed in our algebra (because their result would not be captured by our representation system).

Bottom-k. Of course, all of our study of top-k also applies to a corresponding bottom-k operator that returns the *last k* elements. The same tractability and intractability results apply as they are symmetric up to order reversal.

Instance possibility. A special case of interest is when the number of tuples in the candidate possible world L is exactly the number of tuples in the possible worlds of $Q(D)$, i.e., the top-k operator has no effect. We can accordingly define:

DEFINITION 6.16. *We define the* instance possibility *problem for a fixed query Q, given an input po-database D and a totally ordered candidate possible world L, as determining whether $L \in pw(Q(D))$. The* instance certainty *problem asks whether $pw(Q(D)) = \{L\}$.*

There is a reduction from instance possibility and certainty to the corresponding top-k problems (with the additional assumption that $|L| = |\mathrm{Rel}(Q(D))|$). Hence, all upper bounds in Table 1 also apply to instance possibility and certainty. What is more, it can be seen from the proofs that the lower bounds also extend to instance possibility and certainty.

We can also prove the following tractability result which specifically applies to instance possibility (once again we are studying data complexity):

PROPOSITION 6.17. *The instance possibility problem is in PTIME for the* GEN+LEX *semantics assuming that all input relations are either unordered or totally ordered, and that the fixed query does not use any product operator.*

Ranks. The problem of *rank possibility* and *rank certainty* is to decide, given an input rank k and tuple t, whether the tuple at rank k may be t, or is always t, in the possible worlds of a query result (or, more generally, of a po-relation).

PROPOSITION 6.18. *The problem of rank possibility and rank certainty on any po-relation R is in PTIME.*

Sublists. The problem of *sublist possibility* is to decide, given a totally ordered relation L and a po-relation R, whether there is a possible world L' of R such that L is a sublist of L'.

As there is a clear reduction from instance possibility to sublist possibility, and NP-membership is immediate by guessing a way to realize a suitable world L', we can show:

PROPOSITION 6.19. *The sublist possibility problem is NP-complete in the input po-relation R and candidate sublist L.*

However, following a similar idea as Proposition 6.3:

PROPOSITION 6.20. *We can solve the sublist possibility problem in PTIME in the input po-relation R and sublist L, if the length of L is assumed to be bounded by a constant.*

7. DUPLICATE ELIMINATION

So far we have focused on bag semantics, keeping duplicate tuples and treating them as different objects. Yet, in many motivating scenarios, tuples with the same values actually refer to the same object (e.g., the same restaurant reviewed on different Web sites), and in a set semantics spirit one would like to keep a single copy of them. The main difficulty is that tuples with the same value may differ in terms of their order relation with other tuples.

We next discuss what form of duplicate elimination can still take place. We define a dupElim operator, first on totally ordered relations and then (following axiom consistency) on po-relations.

7.1 Semantics for Totally Ordered Relations

We will rely on a notion of *undistinguishable subsets*:

DEFINITION 7.1. *Given any totally ordered relation $L = (t_1, \dots, t_n)$ and a subset S of tuples from L, we say that S is an* undistinguishable subset *(or* u-subset*) if it is a contiguous sub-sequence of duplicate tuples in W, i.e. for all $t_i, t_j \in S$, we have $t_i = t_j$ and for all $t \in W \backslash S$, $t < t_i$ iff $t < t_j$, and $t_i < t$ iff $t_j < t$.*

EXAMPLE 7.2. *In the totally ordered relation $\prod_1(Hotel)$, where Hotel is as in Figure 1, the two "Mercure" tuples do not form a u-subset since they do not agree on their relative ordering with respect to "Balzac". By contrast, in a totally ordered relation $L = (A, B, B, C)$ (where A, B, and C are tuples over $\mathscr{D}$), the two occurrences of B form an u-subset. Note that a singleton is always a u-subset.*

We use this notion to define a semantics for duplicate elimination on any totally ordered relation L, in the following way. First, check that for every tuple value in L, all the occurrences of this value form a u-subset. If this condition is satisfied, we set dupElim(L) to be the single possible world obtained by picking one representative element for each u-subset (clearly the result does not depend on the choice of representatives). If it is not respected, we say that duplicate elimination has *failed*, and set dupElim(L) to be an empty set of possible worlds.

Intuitively, duplicate elimination tries to reconcile (or "synchronize") order constraints for tuples sharing the same values, and fails when this cannot be done. We discuss other possibilities in Section 7.3.

EXAMPLE 7.3. *For* dupElim$(\prod_1(Hotel))$*, we can see that duplicate elimination fails. However, for L as in the previous example,* dupElim$(L) = (A, B, C)$.

7.2 Semantics for General oi-Relations

We now consider duplicate elimination for oi-relations, which may consist of multiple possible worlds. We define duplicate elimination via axiom consistency: the possible worlds are all results of duplicate elimination for each possible world of the input:

DEFINITION 7.4. *Given an oi-relation R, we let* dupElim(R) *be* $\bigcup_{W \in R}$ dupElim(W). *We say that* dupElim(R) completely fails *if* dupElim$(R) = \emptyset$.

The intuition is that we only keep possible worlds where duplicates agree, so duplicate elimination can be done "safely". We *completely fail* if dupElim fails on all possible worlds.

EXAMPLE 7.5. *Consider the totally ordered relation defined as $Rest_3 =$* (Tsukizi, Gagnaire)*, and consider the following query Q :* dupElim$(\prod_1(Rest) \cup_{\mathtt{GEN}} Rest_3)$. *The goal of query Q is to combine two different restaurant rankings, performing duplicate eliminations to consider different occurrences of the same restaurant name as referencing the same actual restaurant. The only possible world of the query output (and indeed, the only consistent way of consolidating both rankings) is (*Tsukizi, Gagnaire, TourArgent*), since duplicate elimination fails in the other possible worlds of the union.*

It is immediate that:

PROPOSITION 7.6. *For every oi-relation R, the following holds:* dupElim(R) *either completely fails or its result may be captured by a non-empty oi-relation.*

More interestingly, po-relations are still a strong representation system for queries with dupElim (up to complete failure). To show this, we need to define a notion of quotient of a po-relation by value equality:

DEFINITION 7.7. *For a po-relation $R = (ID, T, <)$, we define the* value-equality quotient *of R as the directed graph $\mathrm{G}_R = (ID', E)$ where: ID' is the quotient of ID by the equivalence relation $id_1 \sim id_2 \Leftrightarrow T(id_1) = T(id_2)$; and $E := \{(id'_1, id'_2) \in ID'^2 \mid id'_1 \neq id'_2 \wedge \exists (id_1, id_2) \in id'_1 \times id'_2, id_1 < id_2\}$.*

Cycles in the value-equality quotient of R precisely characterize complete failure of dupElim.

PROPOSITION 7.8. *For any po-relation R,* dupElim$(pw(R))$ *completely fails iff G_R has a cycle.*

COROLLARY 7.9. *For every po-relation R, one can determine in PTIME if* dupElim$(pw(R))$ *completely fails; if it does not, one can compute in PTIME a po-relation R' such that we have $pw(R') =$* dupElim$(pw(R))$.

Combined with Theorem 4.7, this means that po-relations may still serve as representation system for the extended language including dupElim, if we add the possibility of declaring failure of the entire query.

Top-k. Notice that the above result, together with Proposition 6.15, implies that when we perform dupElim as the last operation of a query (in the spirit of set semantics), then top-k and instance possibility and certainty may be decided in PTIME data complexity.

7.3 Alternative Semantics

In addition to the dupElim operator that we have defined, there are at least two plausible alternative semantics for duplicate elimination on totally ordered relations, which we can then extend to oi-relations using axiom consistency.

Weak duplicate elimination. A first possibility is to keep one representative element for each maximal u-subset (sequence of undistinguishable duplicates), even if multiple representatives are kept for each value. This duplicate elimination scheme is *weak* in the sense that duplicates may remain in the output.

EXAMPLE 7.10. *For $A \neq B$ tuples on $\mathscr{D}$, consider the totally ordered relation $L = (A, B, B, A)$. According to this semantics, we would have* dupElim$(L) = (A, B, A)$.

However, the analogue of Proposition 7.6 does not apply to this semantics: the result of duplicate elimination on an oi-relation (or even on a po-relation) may no longer be representable as an oi-relation:

EXAMPLE 7.11. *Consider po-relation $R = (\{a_1, b, a_2\}, T, <)$ with $T(a_1) = T(a_2) = A$ and $T(b) = B$, where $A \neq B$ are tuples over $\mathcal{D}$, and $<$ is defined by $a_1 < b$ and $a_1 < a_2$. We have $pw(R) = \{(A,B,A),(A,A,B)\}$, so that by axiom consistency we have that $\mathrm{dupElim}(pw(R)) = \{(A,B,A),(A,B)\}$ which is no longer an oi-relation (the underlying bag relations of the two possible worlds are different).*

Aggressive duplicate elimination. To design a semantics which is not weak but cannot fail, one alternative scheme is to define duplicate elimination on a totally ordered relation L as the oi-relation whose possible worlds are all possible totally ordered relations that can be obtained by choosing one representative element for each value, no matter whether the representatives are undistinguishable or not. In other words, we do not worry if we are not able to reconcile the order between duplicate tuples:

EXAMPLE 7.12. *Considering L as in Example 7.10, we obtain* $\mathrm{dupElim}(L) = \{(A,B),(B,A)\}$.

The result is then always representable as an oi-relation. However, the analogue of Corollary 7.9 does not hold: po-relations are no longer a strong representation system for this semantics:

EXAMPLE 7.13. *For A, B and C distinct tuples on $\mathcal{D}$, consider the totally ordered relation $L = (A,C,B,C,A)$. We then have that $\mathrm{dupElim}(L) = \{(A,C,B),(A,B,C),(B,C,A),(C,B,A)\}$, and there is no po-relation R such that $pw(R) = \mathrm{dupElim}(L)$ (not all six permutations of $\{A,B,C\}$ are possible worlds, yet each comparability pair is violated in a possible world).*

8. RELATED WORK

We next overview multiple lines of related work.

Order theory. Our development is naturally related to order theory; in particular, po-relations may be viewed as *labeled posets* [30], where labels are tuples. However, our focus on data management means that we study transformations based on queries, and study their complexity as a function of the inputs (data complexity), which is not a common perspective when studying partial orders (see an exception below). Hence, while we use the notions of poset dimension [10] and series-parallel posets [32], and our proofs of corresponding results leverage technical lemmas from [16, 29], to our knowledge our results are not covered by general results from (partial) order theory.

The most relevant work in this context is [14,15], bridging partial order theory and data management: they study queries on *pomsets*, which are labeled posets quotiented by isomorphism (i.e., renaming of identifiers). Their operators include $\cup_{\mathtt{GEN}}$ ("additive union"), $\cup_{\mathtt{CAT}}$ ("concatenation"), $\times_{\mathtt{LEX}}$ ("cartesian product") and generalizations of projection and selection ("restructuring"). Their Proposition 4.1 thus implies our Proposition 5.6 about the output of LEX queries being series-parallel, and we use their Proposition 4.6 as a tool in the proof of Theorem 6.5. However, our finer results about LEX (e.g., Proposition 5.9) are new, and our results about GEN are not covered by theirs as the $\times_{\mathtt{GEN}}$ operator is not proposed there. Beyond that, our semantic desiderata (not met by their semantics), our study of top-k possibility and certainty and our particular notion of duplicate elimination are all absent from [15] and novel to our knowledge. The "individual objects" operator of [15] is similar to our dupElim but removes conflicting order constraints rather than ensuring that they can be reconciled; also, it focuses on manipulating pomsets directly rather than seeing them as a representation system for transformations on their possible worlds.

Order theory has been also largely considered for handling preference data in database systems [3, 20, 21, 34] with concrete representation systems or query languages; our study differs from these in the sense that we propose a more general framework for data management in ordered domains.

Queries on ordered domains. In contrast to our work, which studies relations with ordered tuples, many works [6, 28] consider a different setting where there is an underlying order on *domain elements*. For instance, [36] studies a model where the universe is partially ordered and comparability relations between domain elements are seen as facts. Further, in the context of the expressive power of queries, the effect of having total ordering on the universe has been studied extensively [18, 37]. Yet, all these works differ from ours, because the underlying model is not the same and does not consider order between tuples.

Temporal data and data integration. A common use of order on facts are temporal databases [8,33] but they usually consider the order induced by timestamps, which is a total order. One exception is [12] which considers the partial order on facts induced by some facts being *more current* than others, and reasons about the possible total orders for the currency relation. However, the work does not consider how the order on facts could be manipulated via relational queries. What is more, because of the focus on currency, the possible worlds of the order are only completions for the facts associated to each entity, not for the overall facts, so they are not totally ordered relations in general: this differs from our semantics.

Data integration on ordered data has been considered in context of *rank aggregation* [34, 35] which study how to order integrated data according to user preferences. However, these works explore only specific semantics of ranking, in particular the ranking-join operator, and assume that the order on data is given by a score of a certain form; by contrast, we study the whole PosRA algebra and make no assumption about the order. The recent work [2] discusses the integration of temporal (hence, ordered) data, in a way that follows user preferences. Important differences with our approach include their focus on temporal data rather than arbitrary orders (although they can have multiple time domains), and their focus on a single preference-aware union operator, unlike our setting where we extend the full positive relational bag algebra.

Incomplete and probabilistic information. Our notion of oi-relation is inspired by the general question of incomplete data management [17] (which has also been extensively studied beyond relational settings, see e.g. [4, 26]) with our axiom consistency being a standard desideratum in this context. In the same way, our study of possibility and certainty in Section 6 is a common focus of works on incomplete information, unique here in the sense that uncertainty is on the order of tuples. Related work in this direction is the study of uncertain and probabilistic temporal and streaming data (e.g., [11, 23–25]); led by different motivations, these works focus on different semantic choices than ours (e.g. not focusing on relational algebra on partial orders, not considering duplicate elimination, having different notions of top-k, etc.). Top-k queries has been studied in various probabilistic contexts [9, 23, 31, 34, 39], but the semantics of ranking results for top-k is typically based on probability of results, on known values, or a combination of both (e.g. ranking based on expected value), rather than our ranking which is a combination of given rankings without a known underlying ordered domain.

Practical languages and implementations. XQuery and SQL are two examples of concrete query languages which can manipulate ordered data. In terms of specification, in standard SQL and some of its extensions [22], "ordering of the rows of the table specified by the query expression is guaranteed only for the query expression that immediately contains the `ORDER BY` clause" [19]. For XQuery, results are either ordered or unordered, but operations (e.g. union) will revert to the non-controllable document order ([38], section 3.4.2). Similar principles apply to other query languages. Practical implementations of SQL, when computing, e.g., the union of two ordered results, usually return one possible choice to combine their orders, though the specifics depend on which algorithm is used to compute the result. In contrast, our work proposes a more principled way of allowing to maintain all possible orderings that "make sense" (as made precise by our axiomatization).

9. CONCLUSION

We have proposed and studied possible worlds semantics for query evaluation, for the positive relational algebra enriched with constructs for order-based selection such as top-k, in presence of order-incomplete relational data. Other semantic choices are plausible and may fit our paradigm. Their exploration, involving further study of the tradeoff between expressiveness and complexity, is left for future work. As we have pointed out, this is particularly the case for duplicate elimination and set semantics.

An additional intriguing challenge for future work would be to combine "standard" incompleteness on data values, in addition to incompleteness of the order (possibly based on extending our possible worlds approach). Last, since our motivating scenarios involve data originating from multiple sources, effective provenance tracking in this context would be particularly relevant; we intend to study it as future work.

10. REFERENCES

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

[2] B. Alexe, M. Roth, and W.-C. Tan. Preference-aware integration of temporal data. Technical Report UCSC-SOE-14-04, UCSC, 2014.

[3] A. Arvanitis and G. Koutrika. PrefDB: Supporting Preferences as First-Class Citizens in Relational Databases. *IEEE TKDE*, 26(6):1–1, 2014.

[4] P. Barceló, L. Libkin, A. Poggi, and C. Sirangelo. XML with incomplete information. *J. ACM*, 58(1):4:1–4:62, Dec. 2010.

[5] H. L. Bodlaender and B. van Antwerpen–de Fluiter. *Parallel algorithms for series parallel graphs*. Springer, 1996.

[6] P. Buneman, A. Jung, and A. Ohori. Using powerdomains to generalize relational databases. *Th. Comp. Sci.*, 91(1), 1991.

[7] S. Chaudhuri and M. Y. Vardi. Optimization of *Real* conjunctive queries. In *PODS*, 1993.

[8] J. Chomicki and D. Toman. Time in database systems. In *Handbook of Temporal Reasoning in Artificial Intelligence*. Elsevier, 2005.

[9] G. Cormode, F. Li, and K. Yi. Semantics of ranking queries for probabilistic data and expected ranks. In *ICDE*, 2009.

[10] B. Dushnik and E. W. Miller. Partially ordered sets. *Amer. J. Math.*, 63(3), 1941.

[11] T. Emrich, H. Kriegel, N. Mamoulis, M. Renz, and A. Züfle. Querying uncertain spatio-temporal data. In *ICDE*, pages 354–365, 2012.

[12] W. Fan, F. Geerts, and J. Wijsen. Determining the currency of data. *TODS*, 37(4):25, 2012.

[13] R. Fraïssé. L'intervalle en théorie des relations; ses genéralisations, filtre intervallaire et clôture d'une relation. *North-Holland Math. Stud.*, 99, 1984.

[14] S. Grumbach and T. Milo. An algebra for pomsets. In *ICDT*, 1995.

[15] S. Grumbach and T. Milo. An algebra for pomsets. *Inf. Comput.*, 150(2), 1999.

[16] T. Hiraguchi. On the dimension of orders. *Sci. rep. Kanazawa Univ.*, 4(01), 1955.

[17] T. Imieliński and W. Lipski. Incomplete information in relational databases. *JACM*, 31(4), 1984.

[18] N. Immerman. Relational queries computable in polynomial time. *Inf. Control*, 68(1-3), 1986.

[19] ISO. ISO 9075:2008: SQL, 2008.

[20] M. Jacob, B. Kimelfeld, and J. Stoyanovich. A system for management and analysis of preference data. *VLDB Endow.*, 7(12):1255–1258, 2014.

[21] W. Kiessling. Foundations of preferences in database systems. In *VLDB*, pages 311–322. VLDB Endowment, 2002.

[22] W. Kießling, M. Endres, and F. Wenzel. The preference SQL system - an overview. *IEEE Data Eng. Bull.*, 34(2):11–18, 2011.

[23] B. Kimelfeld and C. Ré. Transducing Markov sequences. *J. ACM*, 61(5):32, 2014.

[24] M. Koubarakis. Database models for infinite and indefinite temporal information. *Inf. Syst.*, 19(2), 1994.

[25] J. Letchner, C. Ré, M. Balazinska, and M. Philipose. Lahar demonstration: Warehousing Markovian streams. *PVLDB*, 2(2):1610–1613, 2009.

[26] L. Libkin. Data exchange and incomplete information. In *PODS*, pages 60–69, 2006.

[27] R. H. Möhring. Computationally tractable classes of ordered sets. In *Algorithms and Order*. Springer, 1989.

[28] W. Ng. An extension of the relational data model to incorporate ordered domains. *TODS*, 26(3), 2001.

[29] O. Øre. *Theory of Graphs*, chapter 10. AMS, 1962.

[30] V. R. Pratt. The pomset model of parallel processes: Unifying the temporal and the spatial. In *Seminar on Concurrency*, 1984.

[31] C. Re, N. N. Dalvi, and D. Suciu. Efficient top-k query evaluation on probabilistic data. In *ICDE*, 2007.

[32] B. Schröder. *Ordered Sets: An Introduction*. Birkhäuser, 2003.

[33] R. T. Snodgrass, J. Gray, and J. Melton. *Developing time-oriented database applications in SQL*. Morgan Kaufmann, 2000.

[34] M. Soliman and I. Ilyas. Ranking with uncertain scores. In *ICDE*, pages 317–328, 2009.

[35] M. A. Soliman, I. F. Ilyas, D. Martinenghi, and M. Tagliasacchi. Ranking with uncertain scoring functions: semantics and sensitivity measures. In *SIGMOD*, 2011.

[36] R. van der Meyden. The complexity of querying indefinite data about linearly ordered domains. *JCSS*, 54(1), 1997.

[37] M. Y. Vardi. The complexity of relational query languages (extended abstract). In *STOC*, 1982.

[38] W3C. XQuery 3.0: An XML query language, 2014.

[39] X. Zhang and J. Chomicki. Semantics and evaluation of top-k queries in probabilistic databases. *DAPD*, 26(1), 2009.

APPENDIX

A. PROOFS FOR SECTION 3 (ORDER-INCOMPLETE DATABASES)

PROPOSITION 3.6. *Axiom bag is implied by axioms constants, select, project, consistency, union, and product.*

Proof. The proof is naturally by induction on the structure of a PosRA query (where Q, Q_1, Q_2 are PosRA queries) over $\{R_1, \ldots, R_n\}$ and $\mathscr{C}$:

- if the query is of the form R_i, this is trivial;
- the constants axiom implies that $\mathrm{Rel}(\emptyset) = \mathrm{Rel}(()) = \{\{\ \}\}$;
- for any tuple t, the constants axiom implies that $\mathrm{Rel}([t]) = \mathrm{Rel}((t)) = \{\{t\}\}$;
- for any $n \in \mathbb{N}$, the constants axiom directly implies that $\mathrm{Rel}((0, \ldots, n)) = \{\{0, \ldots, n\}\}$;
- for any tuple predicate φ, the consistency and select axioms imply that any possible world of the oi-relation $\sigma_\varphi(Q(D))$ is formed of the tuples that satisfy φ, with their original multiplicity in $Q(D)$;
- for any sequence of integers $(k_1, \ldots, k_p)$, the consistency and project axiom imply that any possible world of $\Pi_{k_1 \ldots k_p}(Q(D))$ is formed of all tuples of the form $\Pi_A(\{\{t\}\})$ for $t \in \mathrm{Rel}(Q(D))$, with the sum of the multiplicities of such tuples t;
- the union and consistency axioms imply that any possible world of $Q_1(D) \cup Q_2(D)$ contains exactly the tuples of $Q_1(D)$ and of $Q_2(D)$, with the sum of their multiplicities;
- the product (especially, the requirement of the mapping being one-to-one) and consistency axioms finally imply that any possible world of $Q_1(D) \times Q_2(D)$ has for underlying bag relation the product of these two relations. □

PROPOSITION 3.7. *None of the axioms in Ax is implied by the others. In fact, if any of these axioms is discarded, there is a semantics consistent with the other axioms (and, except when removing axiom constants, with axiom bag), and a trivial query[4] which is not the identity over this semantics, namely one of: $\sigma_{\mathrm{true}}(R)$, $R \cup \emptyset$, or $\Pi_{1 \ldots n}(R \times \mathbb{N}^{\leqslant 0})$ assuming that the input relation R has arity n.*

Proof. First, let us prove that with the axioms Ax, each of the trivial queries that we consider is indeed the identity:

$\sigma_{\mathrm{true}}(R)$: by axiom select, this is the identity on totally ordered relations; by axiom consistency, this is also the identity on oi-relations.

$R \cup \emptyset$: by axioms union and constants, this is the identity on totally ordered relations; by axiom consistency, this is also the identity on oi-relations.

$\Pi_{1 \ldots n}(R \times \mathbb{N}^{\leqslant 0})$: we assume n to be the arity of the relation mapped to R. By axiom product and constants, $R \times \mathbb{N}^{\leqslant 0}$ results in an identical totally ordered relation, except that its arity has increased by one. Then, by axiom project, we retrieve the initial totally ordered relation. By axiom consistency, this query is also the identity on oi-relations.

We now consider each axiom in turn, showing that dropping it would allow a semantics that still satisfies axiom bag (except when the removed axiom is axiom constants) but where one of these trivial queries is not the identity. This also implies that the set of axioms is not redundant.

For every case, we start with the `GEN` semantics (see Section 3.4) and modify it for one particular operator. We also rely on suitable modifications of Proposition 3.6 to show that the bag axiom is still satisfied by this modified semantics.

constants: we can set the semantics of $\emptyset$ to be $\langle 0 \rangle$. The bag axiom is violated, but no other axiom constrains the semantics of $\emptyset$. Then, by axiom union, $\emptyset \cup \emptyset$ has at least one possible world, and the only possible world that is consistent with the axiom is $(0,0)$. Therefore $\emptyset \cup \emptyset \neq \emptyset$.

select: set the semantics of $\sigma_\varphi(L)$ to be the set of all permutations of $\sigma_\varphi(\mathrm{Rel}(L))$. This change does not affect the validity of the bag axiom, and no other axiom constrains the semantics of selection on totally ordered relations. Then:

$$\sigma_{\mathrm{true}}(\{(0,1)\}) = \{(0,1),(1,0)\} \neq \{(0,1)\}.$$

project: set the semantics of $\Pi_A(L)$ to be the set of all permutations of $\Pi_A(\mathrm{Rel}(L))$. This change does not affect the validity of the bag axiom, and no other axiom constrains the semantics of projection on totally ordered relations. Then:

$$\Pi_{1,2}(\{(0,1)\} \times \mathbb{N}^{\leqslant 0}) = \{(0,1),(1,0)\} \neq \{(0,1)\}.$$

consistency: set the semantics of union on oi-relations when at least one operand has at least two possible worlds to be the set of all permutations of the bag semantics of union. This change does not affect the validity of the bag axiom, nor does it change anything to the semantics of union on totally ordered relations as constrained by the union axiom. Then:

$$\begin{aligned}&\{(0,1,2),(0,2,1)\} \cup \emptyset \\ =&\{(0,1,2),(0,2,1),(1,0,2),(1,2,0),(2,0,1),(2,1,0)\} \\ \neq&\{(0,1,2),(0,2,1)\}.\end{aligned}$$

union: set the semantics of $L_1 \cup L_2$ to be the set of all permutations of $\mathrm{Rel}(L_1) \cup \mathrm{Rel}(L_2)$. This change does not affect the validity of the bag axiom, and no other axiom constrains the semantics of union on totally ordered relations. Then:

$$\{(0,1)\} \cup \emptyset = \{(0,1),(1,0)\} \neq \{(0,1)\}.$$

[4]By "trivial" we mean that the query is the identity for the standard bag semantics. However, note that we usually do not want all trivial queries to be the identity for oi-relations (see details in appendix).

product: set the semantics of $L_1 \times L_2$ to be the set of all permutations of $\mathrm{Rel}(L_1) \times \mathrm{Rel}(L_2)$. This change does not affect the validity of the bag axiom, and no other axiom constrains the semantics of product on totally ordered relations. Then:

$$\Pi_{1,2}(\{(0,1)\} \times \mathbb{N}^{\leqslant 0}) = \{(0,1),(1,0)\}) \neq \{(0,1)\}. \qquad \square$$

As a complement to Proposition 3.7, we present an example of a query whose semantics is the identity for bag relational algebra but where we do *not* wish that the query is the identity for oi-relations.

EXAMPLE A.1. *Assume a user favors Mercure hotels. The user issues* $Q := \sigma_{.1=\mathrm{Mercure}}(\mathit{Hotel}) \cup \sigma_{.1=\mathrm{Mercure}}(\mathit{Hotel})$ *with the intent of retrieving all hotels, but treating Mercure hotels in a special way. The semantics of* Q *in the bag relational algebra is the identity. However, for oi-relations, it may not be the identity: intuitively, the order between tuples satisfying* ".1 = Mercure" *and those that do not is lost and could be recreated in a different way. For example, the user may want to put all Mercure hotels first; that is the behavior of the* $\cup_{\texttt{CAT}}$ *operator, that we introduce further.*

PROPOSITION 3.9. *The* GEN *semantics satisfies Ax.*

Proof. By construction, GEN satisfies the constants, select, project, and consistency axioms.

The fact that GEN satisfies the union axiom is clear: all possible worlds in the union are interleavings since the possible worlds are precisely all interleavings.

Let us now prove that the product axiom is also satisfied. Let L be a possible world of $L_1 \times L_2$. Let us write $L_1 = (t_1^{(1)}, \ldots, t_p^{(1)})$ and $L_2 = (t_1^{(2)}, \ldots, t_q^{(2)})$. The totally ordered relation L is defined to contain, for every $1 \leqslant k \leqslant p$ and $1 \leqslant l \leqslant q$ a tuple $t_{k,l} = \langle t_k^{(1)}, t_l^{(2)} \rangle$; we define ψ as the function that maps (k,l) to the position of tuple $t_{k,l}$ in L. For a fixed k, the projection of $\psi(k, [\![1;q]\!])$ onto the last $\mathrm{a}(L_2)$ attributes has for elements (ignoring order) $t_1^{(2)}, t_2^{(2)}, \ldots t_q^{(2)}$, i.e., exactly those of L_2. We just need to show that order is also preserved. Assume it is not the case; then there exists $1 \leqslant i < j \leqslant q$ such that $t_{k,j}$ precedes $t_{k,i}$ in L. This contradicts the semantics of product in GEN. Similarly, we show that for a fixed l, the projection of $\psi([\![1;p]\!], l)$ onto the first $\mathrm{a}(L_1)$ attributes is exactly L_1. $\square$

PROPOSITION 3.10. *For any PosRA query* Q *and any oi-database* D*, letting* W *be the result of evaluating* Q *on* D *under the* GEN *semantics and* W' *be the result under a different semantics satisfying Ax, we have:* $W' \subseteq W$.

Proof. We prove this by induction on the structure of Q. Since the axioms fully constrain constants, selection, and projection, we can limit the study to union and product. Letting X be some other semantics satisfying the axioms, we will write $\cup_{\texttt{X}}$ and $\times_{\texttt{X}}$ for unions and products performed under X.

Assume that $Q = Q_1 \cup Q_2$ and, letting W_1 and W_2 be the results of evaluating Q_1 and Q_2 under the GEN semantics, and W_1 and W_2 be the results under the X semantics, let $W := W_1 \cup_{\texttt{GEN}} W_2$ and $W' := W_1' \cup_{\texttt{X}} W_2'$. By induction hypothesis we have $W_1' \subseteq W_1$ and $W_2' \subseteq W_2$, and we wish to show that $W' \subseteq W$. Let L be one of the possible worlds of W'. By the consistency axiom, there exists a possible world L_1 of W_1' and L_2 of W_2' such that $L \in L_1 \cup L_2$, and as $W_1' \subseteq W_1$ and $W_2' \subseteq W_2$ we have $L_1 \in W_1$ and $L_2 \in W_2$. Also, by the union axiom, L is an interleaving of L_1 and L_2. Since the GEN semantics of union consists of *all* interleavings of the operands, in particular, $L \in W$.

Now assume that $Q = Q_1 \times Q_2$, and define W_1, W_2, W_1', W_2', W, and W', similarly as above, replacing unions by products. As above, let L be a possible world of W', and we must show that it is a possible world of W; define L_1 and L_2 as above. The axiom product requires the existence of a mapping ψ from pairs of positions on L_1, L_2 to positions on L. As there may be several of them, we choose ψ to be one that maximizes the following quantity: $\sum_{1 \leqslant k \leqslant |L_1|} \sum_{1 \leqslant l \leqslant |L_2|} k \times l \times \psi(k,l)$.

We first note that for any $1 \leqslant k \leqslant |L_1|$, $1 \leqslant l \leqslant |L_2|$, the tuple at position $\psi(k,l)$ in L is $\langle t_k^{(1)}, t_l^{(2)} \rangle$ where $t_j^{(i)}$ is the tuple at position j in L_i. This is a direct consequence of two applications of the product axioms.

Let $1 \leqslant k < k' \leqslant |L_1|$, $1 \leqslant l < l' \leqslant |L_2|$. We show that $\psi(k,l) \leqslant \psi(k',l')$, which will conclude the proof since W contains all such lists. Assume this is not the case, i.e., $\psi(k,l) > \psi(k',l')$.

First, consider the case where $k = k'$, and let l, l' be the lowest two positions such that $l \leqslant l'$ and $\psi(k,l) > \psi(k,l')$, which means $\psi(k,l')$ is the l-th position among all $\psi(k,m)$ for $1 \leqslant m \leqslant |L_2|$.

Then by the axiom product, the sublist of L with positions $\{\psi(k,m) \mid 1 \leqslant m \leqslant |L_2|\}$ is equal (once projected on the last $\mathrm{a}(L_2)$ attributes) to L_2. In particular, this means $\langle t_k^{(1)}, t_{l'}^{(1)} \rangle$ (the tuple at position $\psi(k,l')$) is equal to $\langle t_k^{(1)}, t_l^{(1)} \rangle$ (the l-th tuple in the sublist of L) with positions $\{\psi(k,m) \mid 1 \leqslant m \leqslant |L_2|\}$ and therefore $t_{l'}^{(2)} = t_l^{(2)}$.

Now, consider the mapping $\psi' : [\![1;|L_1|]\!] \times [\![1;|L_2|]\!] \to [\![1;|L_1| \cdot |L_2|]\!]$, defined to be identical to ψ except that $\psi'(k,l) := \psi(k,l')$ and $\psi'(k,l') := \psi(k,l)$. ψ is obviously one-to-one. Furthermore, ψ' satisfies the conditions of the product axiom: the sublist of L of positions $\{\psi'(k,m) \mid 1 \leqslant m \leqslant |L_2|\}$ is the same as for ψ, and since $t_{l'}^{(2)} = t_l^{(2)}$ the sublist of L of positions $\{\psi'(m,l) \mid 1 \leqslant m \leqslant |L_1|\}$ and $\{\psi'(m,l) \mid 1 \leqslant m \leqslant |L_1|\}$ are equal. Now:

$$\begin{aligned}
&\sum_{1 \leqslant p \leqslant |L_1|} \sum_{1 \leqslant m \leqslant |L_2|} pm\psi'(p,m) \\
&= \left(\sum_{1 \leqslant p \leqslant |L_1|} \sum_{1 \leqslant m \leqslant |L_2|} pm\psi(p,m) \right) + k(l'-l)(\psi(k,l) - \psi(k,l')) \\
&> \sum_{1 \leqslant p \leqslant |L_1|} \sum_{1 \leqslant m \leqslant |L_2|} pm\psi(m,m)
\end{aligned}$$

which contradicts the maximality of ψ and concludes the case $k = k'$.

The case where $l = l'$ is exactly symmetrical and we show in the same way that ψ cannot reach the maximal value of the sum.

In the case where $k < k'$, $l < l'$ and $\psi(k',l') < \psi(k,l)$, consider $\psi(k,l')$: either we have $\psi(k,l') < \psi(k',l')$, which means $\psi(k,l') < \psi(k,l)$ and we are back in the case $k = k'$, or $\psi(k',l') < \psi(k,l')$ and we are back in the case $l = l'$. □

PROPOSITION 3.12. $\times_{\texttt{LEX}}$ *satisfies product.*

Proof. The possible worlds of a $\times_{\texttt{LEX}}$ expression are always a subset of the same expression for the $\times_{\texttt{GEN}}$ product, so we conclude because $\times_{\texttt{GEN}}$ satisfies product by Proposition 3.9 and removing possible worlds cannot make product false. □

PROPOSITION 3.14. *The following properties hold:*

(i) *selection-projection in any semantics that satisfies Ax;*
(ii) *union-associative, union-commutative, as well as projection-union for* $\cup_{\texttt{GEN}}$*;*
(iii) *product-associative, selection-product, as well as projection-product for* $\times_{\texttt{GEN}}$ *and* $\times_{\texttt{LEX}}$*;*
(iv) *union-distributive in* GEN *but not in* LEX*;*
(v) *value-genericity in* GEN+LEX.

Proof. First observe that it suffices to show all properties but value-genericity hold when W_1, W_2 and W_3 are totally ordered relations. Indeed, all properties are of the form $Q_1(D) = Q_2(D)$ for a database D containing oi-relations W_1, W_2 and W_3 and Q_1, Q_2 some queries; by axiom consistency, $Q_1(D)$ is the union over all possible worlds L_1, L_2, L_3 of W_1, W_2 and W_3 of $Q_1(D_{L_1,L_2,L_3})$ where D_{L_1,L_2,L_3} is like D except that W_i is replaced with L_i. If the properties holds for lists, we thus have that $Q_1(D)$ is the union over all possible worlds L_1, L_2, L_3 of $Q_2(D_{L_1,L_2,L_3})$, which is by axiom consistency again $Q_2(D)$.

We now consider each property in turn, for totally ordered relations L_1, L_2, and L_3, and φ, φ', K_1, K_2 as in the statement of every property. Note that by the bag property, which holds in any semantics that satisfies Ax, the underlying bag relation on the left-hand side and right-hand side of the equation of every property are the same.

union-associative: Let $L \in L_1 \cup_{\texttt{GEN}} (L_2 \cup_{\texttt{GEN}} L_3)$. L can be partitioned into sublists L'_1 and L', which can in turn be partitioned into sublists L'_2 and L'_3, with $L_1 = L'_1$, $L_2 = L'_2$, $L_3 = L'_3$. Calling L'' the merging of the sublists L'_1 and L'_2, L can thus be partitioned into sublists L'' and L'_3, which means $L \in (L_1 \cup \texttt{GEN} L_2) \cup \texttt{GEN} L_3)$.

union-commutative: An interleaving of L_1 and L_2 is also an interleaving of L_2 and L_1.

union-distributive: Let us first show this does not hold in LEX. Consider totally ordered relations $L_1 = (0)$, $L_2 = (1)$, and $L_3 = (2,3)$. Then $(L_1 \cup_{\texttt{GEN}} L_2) \times_{\texttt{LEX}} L_3 = \{(0,1),(1,0)\} \times_{\texttt{LEX}} (2,3) = \{(\langle 0,2\rangle,\langle 0,3\rangle,\langle 1,2\rangle,\langle 1,3\rangle),(\langle 1,2\rangle,\langle 1,3\rangle,\langle 0,2\rangle,\langle 0,3\rangle),\}$. On the other hand, $(L_1 \times_{\texttt{LEX}} L_3) \cup (L_2 \times_{\texttt{LEX}} L_3) = \{(\langle 0,2\rangle,\langle 0,3\rangle)\} \cup \{(\langle 1,2\rangle,\langle 1,3\rangle\}$. Thus, for example, the possible order $(\langle 0,2\rangle,\langle 1,2\rangle,\langle 0,3\rangle,\langle 1,3\rangle)$ is in $(L_1 \times_{\texttt{LEX}} L_3) \cup (L_2 \times_{\texttt{LEX}} L_3)$ but not in $(L_1 \cup_{\texttt{GEN}} L_2) \times_{\texttt{LEX}} L_3$.

Let us now show that for any three lists, $(L_1 \cup_{\texttt{GEN}} L_2) \times_{\texttt{GEN}} L_3 = (L_1 \times_{\texttt{GEN}} L_3) \cup_{\texttt{GEN}} (L_2 \times_{\texttt{GEN}} L_3)$. The symmetric property is proved in a completely symmetric manner.

Let L be a possible order of $(L_1 \cup_{\texttt{GEN}} L_2) \times_{\texttt{GEN}} L_3$. Let $\psi : [\![1;|L_1|+|L_2|]\!] \times [\![1;|L_3|]\!] \to [\![1;(|L_1|+|L_2|)|L_3|]\!]$ be the mapping from the product axiom. Let (K_1,K_2) the partition of $[\![1;|L_1|+|L_2|]\!]$ that indicates the position, in $L_1 \cup_{\texttt{GEN}} L_2$, of the tuples from L_1 and L_2 given by the union axiom. Consider the sublist L_{13} of L of the positions in $\psi(K_1,[\![1;|L_3|]\!])$ and L_{23} the sublist of all other tuples of L. L_{13} and L_{12} are possible orders of $L_1 \times_{\texttt{GEN}} L_2$ and $L_1 \times_{\texttt{GEN}} L_3$, respectively, as they satisfy the conditions of $\times_{\texttt{GEN}}$. Since L_{13} and L_{23} are disjoint sublists of L, we have $L \in L_{13} \cup L_{23} \subseteq (L_1 \times_{\texttt{GEN}} L_3) \cup_{\texttt{GEN}} (L_2 \times_{\texttt{GEN}} L_3)$ by the consistency axiom.

In the other direction, let $L \in (L_1 \times_{\texttt{GEN}} L_3) \cup_{\texttt{GEN}} (L_2 \times_{\texttt{GEN}} L_3)$. L is an interleaving between a list L_{13} from $L_1 \times_{\texttt{GEN}} L_3$ and a list L_{23} from $L_2 \times_{\texttt{GEN}} L_3$ by the union axiom. Let ψ_1 and ψ_2 be the mappings given by the product axiom for L_{13} and L_{23} respectively. If $L_3 = \emptyset$, the result trivially holds. Otherwise, observe that L_1 is the projection on the first $\mathrm{a}(L_1)$ attributes of the elements of L_{13} at positions $\psi_1([\![1;|L_1|]\!]1)$. Similarly, L_2 is the projection on the first $\mathrm{a}(L_2)$ attributes of the elements of L_{23} at positions $\psi_2([\![1;|L_2|]\!],1)$. Thus, the projection on the first $\mathrm{a}(L_1)$ attributes of the elements of L corresponding to that of L_1 and L_2 in L_{13} and L_{23} is a list L_{12} that is an interleaving between L_1 and L_2, thus in $L_1 \cup_{\texttt{GEN}} L_2$. Now, $L_{12} \times_{\texttt{GEN}} L_3$ is exactly L, which means $L \in (L_1 \cup_{\texttt{GEN}} L_2) \times L_3$.

product-associative: Let $L \in (L_1 \times_{\texttt{GEN}} L_2) \times_{\texttt{GEN}} L_3$. Two arbitrary tuples of L of the form $t = \langle t_1,t_2,t_3\rangle$, $t' = \langle t'_1,t'_2,t'_3\rangle$ with $t \leqslant_L t'$ satisfy $t_3 \leqslant_{L_3} t'_3$ or $\langle t_1,t_2\rangle \leqslant_{L_{12}} \langle t'_1,t'_2\rangle$ for some $L_{12} \in L_1 \times_{\texttt{GEN}} L_2$. But $\langle t_1,t_2\rangle \leqslant_{L_{12}} \langle t'_1,t'_2\rangle$ means that $t_1 \leqslant_{L_1} t'_1$ or $t_2 \leqslant_{L_2} t'_2$. Thus, tuples $t \leqslant_L t'$ satisfy $t_1 \leqslant_{L_1} t'_1$, $t_2 \leqslant_{L_2} t'_2$, or $t_3 \leqslant_{L_3} t'_3$. But then, $L_1 \times (L_2 \times_{\texttt{GEN}} L_3)$ is similarly all lists L' such that whenever $t \leqslant_{L'} t'$, meaning $t_1 \leqslant_{L_1} t'_1$ or $t_2 \leqslant_{L_2} t'_2$ or $t_3 \leqslant_{L_3} t'_3$. Thus $L \in L_1 \times (L_2 \times_{\texttt{GEN}} L_3)$. The other direction is exactly symmetric.

Now, let $L = (L_1 \times_{\texttt{LEX}} L_2) \times_{\texttt{LEX}} L_3$ and $t \leqslant_L t'$. We decompose t and t' as above. We have either $\langle t_1,t_2\rangle =_{L_1 \times_{\texttt{LEX}} L_2} \langle t'_1,t'_2\rangle$ and $t_3 \leqslant_{L_3}$; or $\langle t_1,t_2\rangle <_{L_1 \times_{\texttt{LEX}} L_2} t'_1,t'_2$. In the former case, this means that $t_1 =_{L_1} t'_1$ and $t_2 =_{L_2} t'_2$, which, together with $t_3 \leqslant_{L_3} t'_3$ means that $\langle t_2,t_3\rangle \leqslant_{L_2 \times_{\texttt{LEX}} L_3} \langle t'_2,t'_3\rangle$ and thus $t \leqslant_{L_1 \times_{\texttt{LEX}} (L_2 \times_{\texttt{LEX}} L_3)} t'$. In the latter case, this means that $t_1 <_{L_1} t'_1$ or $t_1 =_{L_1} t'_1$ and $t_2 <_{L_2} t'_2$. But then if $t_1 <_{L_1} t'_1$, $t \leqslant_{L_1 \times_{\texttt{LEX}} (L_2 \times_{\texttt{LEX}} L_3)} t'$; and if $t_2 <_{L_2} t'_2$, $\langle t_2,t_3\rangle \leqslant_{L_2 \times_{\texttt{LEX}} L_3} \langle t'_2,t'_3\rangle$ and thus since $t_1 =_{L_1} t'_1$, $t \leqslant_{L_1 \times_{\texttt{LEX}} (L_2 \times_{\texttt{LEX}} L_3)} t'$. We thus have the fact that in all cases $t \leqslant_L t' \Rightarrow t \leqslant_{L_1 \times_{\texttt{LEX}} (L_2 \times_{\texttt{LEX}} L_3)} t'$. Since $<_L$ and $<_{L_1 \times_{\texttt{LEX}} (L_2 \times_{\texttt{LEX}} L_3)}$ are total orders over the same set of elements, they are the same and $L = L_1 \times_{\texttt{LEX}} (L_2 \times_{\texttt{LEX}} L_3)$.

selection-product: Let $L \in \sigma_\varphi(L_1 \times_{\texttt{GEN}} L_2)$ and $t \leqslant_L t'$ in L. We pose $t = \langle t_1,t_2\rangle$, $t' = \langle t'_1,t'_2\rangle$. By axiom select, there exists a list $L' \in L_1 \times_{\texttt{GEN}} L_2$ such that L is the sublist of L' of all tuples where φ holds. This means $t \leqslant_{L'} t'$ and thus either $t_1 \leqslant_{L_1} t'_1$ or $t_2 \leqslant_{L_2} t'_2$. Since both $\varphi(t)$ and $\varphi(t')$ hold, by definition of φ', both $\varphi'(t_2)$ and $\varphi'(t'_2)$ also hold, thus t_2 and t'_2 are kept in $\sigma_\varphi(L_2)$. Since for arbitrary $\langle t_1,t_2\rangle \leqslant_L \langle t'_1,t'_2\rangle$ we have either $t_1 \leqslant_{L_1} t'$ or $t_2 \leqslant_{\sigma_{\varphi'}(L_2)} t'_2$, by the definition of $\times_{\texttt{GEN}}$, L is a sublist of a list in $L_1 \times_{\texttt{GEN}} \sigma_{\varphi'}(L_2)$. But since both lists have the same underlying relation, they are equal.

Conversely, let $L \in L_1 \times_{\texttt{GEN}} \sigma_{\varphi'}(L_2)$ and $t \leqslant_L t'$ in L with $t = \langle t_1, t_2 \rangle$, $t' = \langle t'_1, t'_2 \rangle$. Then either $t_1 \leqslant_{L_1} t'_1$ or $t_2 \leqslant_{\sigma_{\varphi'}(L_2)} (t'_2)$. But $\sigma_{\varphi'}(L_2)$ is a sublist of L_2 by axiom select. This means $t_2 \leqslant_{L_2} t'_2$. Thus, L is a sublist of a list in $L_1 \times L_2$. Since all tuples of L satisfy φ, it is also a sublist of a list in $\sigma_\varphi(L_1 \times_{\texttt{GEN}} L_2)$. But since both lists have the same underlying relation, they are equal.

We now move to $\times_{\texttt{LEX}}$ and consider $L = \sigma_\varphi(L_1 \times_{\texttt{GEN}} L_2)$, let $t \leqslant_L t'$ with the same notations as before. Thus either either $t_1 <_{L_1} t'_1$ or $t_1 =_{L_1} t'_1$ and $t_2 \leqslant_{L_2} t'_2$. But then as before t_2 and t'_2 are kept in $\sigma_{\varphi'}(L_2)$. We thus have either $t_1 <_{L_1} t'_1$ or $t_1 =_{L_1} t'_1$ and $t_2 \leqslant_{\sigma_{\varphi'}(L_2)} t'_2$. Thus $t_1 \leqslant_{L_1 \times_{\texttt{LEX}} \sigma_{\varphi'}(L_2)} t_2$. Since $<_L$ and $<_{L_1 \times_{\texttt{LEX}} \sigma_{\varphi'}(L_2)}$ are total orders over the same set of elements, they are the same and $L = L_1 \times_{\texttt{LEX}} \sigma_{\varphi'}(L_2)$.

We show the symmetric property for GEN and LEX in the exact same manner.

selection-projection: the fact that selection-projection is implied by Ax, precisely by consistency, select, and project is clear. Indeed, the underlying relations are the same and neither selection nor projection changes the order between tuples.

projection-union: This is a simple consequence of the fact that an interleaving of projections is a projections of the interleavings; order is not changed by projection.

projection-product: We let $K'_1 = K_1, \mathrm{a}(W_1)+1, \dots, \mathrm{a}(W_1)+\mathrm{a}(W_2)$. We show $\Pi_{K'_1}(L_1 \times L_2) = \Pi_{K_1}(L_1) \times L_2$, the symmetric property being proved analogously.

Let $L \in \Pi_{K'_1}(L_1 \times_{\texttt{GEN}} L_2)$. There exists $L' \in L_1 \times_{\texttt{GEN}} L_2$ such that $L = \Pi_{K'_1}(L')$. Let $t = \langle t_1, t_2 \rangle, t' = \langle t'_1, t'_2 \rangle \in L$, with $t \leqslant_{L'} t'$, meaning $\Pi_{K'_1}(t) \leqslant_L \Pi_{K'_1}(t')$. We have either $t_1 \leqslant_{L_1} t'_1$ or $t_2 \leqslant_{L_2} t'_2$. By axiom project, we thus have either $\Pi_{K_1}(t_1) \leqslant_{\Pi_{K_1}(L_1)} \Pi_{K_1}(t'_1)$ or $t_2 \leqslant_{L_2} t'_2$. Thus $L \in \Pi_{K_1}(L_1) \times L_2$.

Conversely, let $L \in \Pi_{K_1}(L_1) \times_{\texttt{GEN}} L_2$. We take $t \leqslant_L t'$ with $t = \langle t_1, t_2 \rangle$, $t' = \langle t'_1, t'_2 \rangle \in L$. There exists $t_0, t'_0 \in L_1$ such that $t_1 = \Pi_{K_1}(t_0)$ and $t'_1 = \Pi_{K_1}(t_0)$. Now, either $t_1 \leqslant_{\Pi_{K_1}(L_1)} t'_1$ and then $t_0 \leqslant_{L_1} t'_0$, or $t_2 \leqslant_{L_2} t'_2$. Thus, $\langle t_0, t_2 \rangle \leqslant_{L_1 \times_{\texttt{GEN}} L_2} \langle t'_0, t'_2 \rangle$. By axiom project, $t \leqslant_{\Pi_{K'_1}(L_1 \times_{\texttt{GEN}} L_2)} t'$. Thus $L \in \Pi_{K'_1}(L_1 \times_{\texttt{GEN}} L_2)$.

Let now $L \in \Pi_{K'_1}(L_1 \times_{\texttt{LEX}} L_2)$. There exists $L' \in L_1 \times_{\texttt{LEX}} L_2$ such that $L = \Pi_{K'_1}(L')$. Let $t = \langle t_1, t_2 \rangle, t' = \langle t'_1, t'_2 \rangle \in L$, with $t \leqslant_{L'} t'$, meaning $\Pi_{K'_1}(t) \leqslant_L \Pi_{K'_1}(t')$. We have either $t_1 <_{L_1} t'_1$ or $t_1 =_{L_1} t'_1$ and $t_2 \leqslant_{L_2} t'_2$. Thus, either $\Pi_{K_1}(t_1) <_{\Pi_{K_1}(L_1)} \Pi_{K_1}(t'_1)$ or $\Pi_{K_1}(t_1) =_{\Pi_{K_1}(L_1)} \Pi_{K_1}(t'_1)$ and $t_2 \leqslant_{L_2} t'_2$. Thus $L = \Pi_{K_1}(L_1) \times L_2$.

value-genericity: We prove this axiom by induction on the structure of Q. The definition requires value-genericity to hold for constants and selections, value-genericity clearly holds for projection, and trivially holds for relation names. We now show that $\cup_{\texttt{GEN}}$, $\times_{\texttt{GEN}}$, and $\times_{\texttt{LEX}}$ also satisfy value-genericity.

Assume that $Q = Q_1 \odot Q_2$ for $\odot \in \{\cup_{\texttt{GEN}}, \times_{\texttt{GEN}}, \times_{\texttt{LEX}}\}$ and let Ω_1, Ω_2 be the oi-relations obtained by evaluating Q_1, Q_2 on the input relations. We have, using the fact that $\lambda(Q)$ only impacts selection and constants, and using the induction hypothesis:

$$\begin{aligned} &\lambda(Q)(\lambda(W_1), \dots, \lambda(W_n)) \\ &= (\lambda(Q_1) \odot \lambda(Q_2))(\lambda(W_1), \dots, \lambda(W_n)) \\ &= \lambda(Q_1)(\lambda(W_1), \dots, \lambda(W_n)) \odot \lambda(Q_2)(\lambda(W_1), \dots, \lambda(W_n)) \\ &= \lambda(Q(W_1, \dots, W_n)) \odot \lambda(Q(W_2, \dots, W_n))) = \lambda(\Omega_1) \odot \lambda(\Omega_2). \end{aligned}$$

We thus simply need to verify that $\lambda(\Omega_1) \odot \lambda(\Omega_2) = \lambda(\Omega_1 \odot \Omega_2)$.

Union only looks at the order of tuples: a list L is an interleaving of L_1 and L_2 if and only if the list $\lambda(L)$ is an interleaving of $\lambda(L_1)$ and $\lambda(L_2)$.

For $\times_{\texttt{GEN}}$ and $\times_{\texttt{LEX}}$, observe that they are defined in terms of order among tuples, not in term of values: L is a possible world of $L_1 \times_{\texttt{GEN}} L_2$ or $L_1 \times_{\texttt{LEX}} L_2$ if and only if $\lambda(L)$ is a possible world of $\lambda(L_1) \times_{\texttt{GEN}} \lambda(L_2)$ or $\lambda(L_1) \times_{\texttt{LEX}} \lambda(L_2)$, respectively. □

As a complement to Proposition 3.14, we show an example where value-genericity is violated:

EXAMPLE A.2. *Consider again the query $Rest \cup Rest_2$. We give the following semantics to union: $L_1 \cup L_2$ is the single list where all tuples that are at the top of list L_2 and satisfy ".2 = 6" come first, followed by tuples of L_1, followed by all other tuples of L_2. This semantics would correspond to the (bizarre) preference of a user for international restaurants only if they are close to the user's home (hence, in the 6th district) and better than any farther international restaurants, French restaurants being preferred to international restaurants otherwise. This is reminiscent of the preference rules of [ART14]. As it results in an interleaving of L_1 and L_2, the union axiom is satisfied. However, the value-genericity axiom is clearly violated, since the special treatment of the value "6" does not appear in the query but is hidden in the semantics of union.*

B. PROOFS FOR SECTION 4 (PARTIALLY-ORDERED DATABASES)

PROPOSITION 4.1. *The number of possible worlds of $R \cup_{\texttt{GEN}} S$ applied to totally ordered relations R and S can be exponential in the number of tuples of the input relations.*

Proof. Let R and S be totally ordered relations with n tuples. The number of possible worlds of the result is the number of possible interleavings, namely, $\binom{2n}{n}$, which is (by Stirling's formula) asymptotically equivalent to $\frac{4^n}{\sqrt{\pi n}} = \Omega(2^n)$. □

THEOREM 4.7. *Let Q be a fixed PosRA query. Given a po-database D, we can compute in polynomial time a po-relation $Q(D)$ such that $pw(Q(D)) = Q(pw(D))$ under the* GEN+LEX *semantics.*

Proof. We show this by induction on the structure of Q.

- If Q is a relation name R, $Q(D) := D(R)$ is a po-relation that can be obtained in time linear in D.
- If $Q = \emptyset$, we let $Q(D)$ be the empty po-relation.
- If $Q = [t]$, $Q(D)$ is the po-relation on the singleton tuple (t).
- If $Q = \mathbb{N}^{\leqslant n}$, $Q(D) := (\llbracket 0;n \rrbracket, k \mapsto (k), <)$ where $<$ is the total order over integers. This has size constant in D.
- If $Q = \sigma_\varphi(Q')$ and $Q'(D) = (ID', T', <')$, let ID be the set of all $\iota \in ID'$ such that $\varphi(T'(\iota))$ holds. Then $Q(D) := (ID, T'_{|ID}, <'_{|ID})$, which is constructible in time linear in $Q'(D)$.
- If $Q = \Pi_{k_1 \ldots k_p}(Q')$ and $Q'(D) = (ID', T', <')$, let $T : \iota \mapsto \Pi_{k_1 \ldots k_p}(T'(\iota))$. Then $Q(D) := (ID', T, <')$ is constructible in time linear in $Q'(D)$.
- If $Q = Q_1 \cup_{\texttt{GEN}} Q_2$, let, for $i \in \{1,2\}$, $Q_i(D) = (ID_i, T_i, <_i)$. If ID_1 and ID_2 are not disjoint, we rename identifiers from one of them to fresh identifiers, redefining T_i and $<_i$ accordingly, which is linear in D. Hence, we assume without loss of generality that ID_1 and ID_2 are disjoint. We define $Q(D) := (ID_1 \cup ID_2, T_1 \cup T_2, <_1 \cup <_2)$. This construction is linear in $Q_1(D)$ and $Q_2(D)$. We will now prove that this gives the right semantics, using the fact that a linear extension of the union of two disjoint partial orders is an arbitrary interleaving of linear extensions of the two partial orders.

  For the forward direction, let L be a possible world of $Q(D)$. By our remark above $Q(D)$, there is a possible world L_1 of $Q(D_1)$ and L_2 of $Q(D_2)$ such that L is an interleaving of L_1 and L_2. By the induction hypothesis, $L_1 \in Q_1(pw(D))$ and $L_2 \in Q_2(pw(D))$. Since $(Q_1 \cup_{\texttt{GEN}} Q_2)(pw(D))$ is formed of all interleavings of $Q_1(pw(D))$ and $Q_2(pw(D))$, $L \in (Q_1 \cup_{\texttt{GEN}} Q_2)(pw(D)) = Q(pw(D))$.

  For the backward direction, let $L \in Q(pw(D))$. This is an interleaving of a $L_1 \in Q_1(pw(D))$ and a $L_2 \in Q_2(pw(D))$. By the induction hypothesis, $L_1 \in pw(Q_1)$ and $L_2 \in pw(Q_2)$, which means L_1 is a possible world of $Q_1(D)$ and L_2 is a possible world of $Q_2(D)$. Thus, L is a possible world of $Q(D)$.
- If $Q = Q_1 \times_{\texttt{GEN}} Q_2$, let, for $i \in \{1,2\}$, $Q_i(D) = (ID_i, T_i, <_i)$ as given by the induction hypothesis. We define $Q(D) := (ID_1 \times ID_2, T, <)$ where $T : (\iota_1, \iota_2) \mapsto \langle T(\iota_1), T(\iota_2) \rangle$ and $<$ is defined as the minimal order relation such that $(\iota_1, \iota_2) < (\iota'_1, \iota'_2)$ whenever there are $i \neq j \in \{1,2\}$ such that $\iota_i < \iota'_i$ and $\iota_j < \iota'_j$. This can be constructed in time polynomial in the product of the size of $Q_1(D)$ and $Q_2(D)$, hence, in time polynomial in D: to construct the order, enumerate all pairs that are as above, and then complete the set of constraints into an order via transitive closure.

  Now, to prove correctness, let L be a possible world of $Q(D)$. For arbitrary $i \in \{1,2\}$, the definition of $<$ ensures there is no $(\iota_1, \iota_2) < (\iota'_1, \iota'_2)$ if $\iota'_i \leqslant \iota_i$. This means $L \in Q(pw(D))$. Conversely, if $L \in Q(pw(D))$, L does not violate any of the constraints of $<$, and is therefore a possible world of $Q(D)$.
- If $Q = Q_1 \times_{\texttt{LEX}} Q_2$, for $i \in \{1,2\}$, $Q_i(D) = (ID_i, T_i, <_i)$. We define $Q(D) := (ID_1 \times ID_2, T, <)$ where T is as in the previous case and $<$ is the lexicographic product of the orders $<_1$ and $<_2$. This is constructible in linear time in the size of the product of $Q_1(D)$ and $Q_2(D)$, and the definition of the `LEX` semantics of product ensures that possible worlds of $Q(D)$ are exactly possible outcomes of Q over $pw(D)$. □

C. PROOFS FOR SECTION 5 (EXPRESSIVENESS)

C.1 Possible Outputs (Section 5.1)

PROPOSITION 5.3. *For any po-relation R, there is a PosRA query Q with no inputs such that the result of evaluating Q using the* `GEN` *semantics is R.*

We use the definitions of Definition 5.2. We first state the following useful lemma (Theorem 9.6 of [Hir55], see also [Øre62]) that we will use throughout, and restate its proof in our terms, for the sake of being self-contained:

LEMMA C.1. *A poset $(P, <_P)$ has a realizer $(L_1, \ldots, L_n)$ of size n iff P is isomorphic to a subset S of $R = \mathbb{N}^{\leqslant k_1} \times_{\texttt{GEN}} \cdots \times_{\texttt{GEN}} \mathbb{N}^{\leqslant k_n}$ for some integers $k_1, \ldots, k_n$ (the order on S being the restriction on that of R).*

Proof. For the forward direction, we identify each element x of P to $f(x) := (n_1^x, \ldots, n_n^x)$, where n_i^x is the position where x occurs in L_i, and take $k_i := |P|$ for all i. Now, for any $x, y \in P$, we have $x <_P y$ iff $n_i^x < n_i^y$ for all $1 \leqslant i \leqslant n$ (that is, $x <_{L_i} y$), hence iff $f(x) <_R f(y)$. Hence, taking S to be the image of f (which is injective), S is indeed isomorphic to P.

For the converse direction, let $k_1, \ldots, k_n$ be the integers, R be as defined, and S be the subset. For each $1 \leqslant i \leqslant n$, we construct L_i that enumerates the elements of P as follows: take for L_i the reverse image by the isomorphism from P to S of the total order on S obtained by sorting the elements of S by their i-th coordinate, and then by their coordinates $1, 2, \ldots, i-1, i+1, \ldots, n$, in lexicographic order. To see that $(L_1, \ldots, L_n)$ is indeed a realizer of P, consider $x, y \in P$, we have $x <_P y$ iff, letting t^x and t^y be the corresponding elements in S, $t_i^x \leqslant_S t_i^y$ for all $1 \leqslant i \leqslant n$ but $t_j^x <_S t_j^y$ for some $1 \leqslant j \leqslant n$. If this holds, then indeed $x <_{L_i} y$: it is obviously true if they differed for coordinate i, otherwise they were tied for all coordinates until one coordinate differed and broke the tie in the correct direction. Conversely, if this does not hold then $y <_{L_j} x$, where j is the coordinate where it does not hold. □

We now prove the main claim:

Proof. We first argue that to prove the claim, it suffices to show that for any poset $(P,<)$, there exists a query Q such that the tuples of $Q()$ (the result of evaluating Q) all have unique values, and the underlying order of the po-relation $Q()$ is $(P,<)$. Indeed, to prove the desired result from this claim, first build a po-relation $Q()$ with the desired partial order and unique values, and then replace the values by the desired values by performing the join (i.e., product, selection, projection) with a union of singleton constant expressions that map each unique tuple value of $Q()$ to the desired value of the corresponding tuple in the desired po-relation R.

We now prove the weaker claim. Fix the desired poset $(P,<)$. Let d be the order dimension of P (it is necessarily finite). We now use Lemma C.1 to argue that, for some tuple predicate φ, P can be obtained (up to the tuple values) as $\sigma_\varphi(\mathbb{N}^{\leqslant k_1} \times_{\texttt{GEN}} \cdots \times_{\texttt{GEN}} \mathbb{N}^{\leqslant k_d})$ for some $k_1,\ldots,k_n$. □

PROPOSITION 5.5. *For any series-parallel po-relation R, there exists a PosRA query Q with no inputs such that the result of evaluating Q under the* LEX *semantics is R.*

Proof. We prove the claim by a straightforward induction on R:

- If R is a singleton (t), then we take $Q := [t]$
- If R is the parallel composition of R_1 and R_2, then, applying the induction hypothesis to obtain Q_1 and Q_2 such that $R_1 = Q_1()$ and $R_2 = Q_2()$, we take $Q := Q_1 \cup Q_2$.
- If R is the series composition of R_1 and R_2, with the same notations and reasoning we take $Q := Q_1 \cup_{\texttt{CAT}} Q_2$. □

PROPOSITION 5.6. *(Follows from [GM99].) For any query Q and po-database D of series-parallel po-relations, $Q(D)$ under the* LEX *semantics is either series-parallel or empty.*

Proof. We prove the claim by induction (in fact we prove a slightly stringer claim, allowing for relations of D to also be empty). The relations of D are either series-parallel po-relations or are empty, and the expressions $\emptyset$, $[t]$ and $\mathbb{N}^{\leqslant n}$ are series-parallel or empty, which proves the base case. For the induction, the union of two series-parallel or empty po-relations of compatible arity is a series-parallel or empty po-relation (and the underlying poset is the parallel composition of the two original posets), the projection of a series-parallel or empty po-relation is still series-parallel or empty (the underlying poset does not change), and the selection of a series-parallel or empty po-relation is either the empty po-relation or its underlying poset is a non-empty restriction of a series-parallel poset, which is still series-parallel [BGR97].

It remains to show the claim for $R \times_{LEX} S$. If either of R or S are empty, then the resulting po-relation is empty as well. Otherwise, its underlying poset P is defined as the lexicographic product of P_1 and P_2, which are that of R and S, and are series-parallel. To see why P is series-parallel, consider any sp-trees T_1 and T_2 representing P_1 and P_2 (see Definition 5.8). Clearly, the result of replacing every "singleton" node of T_1 by a copy of T_2 is an sp-tree for P. Hence, P is series-parallel. □

PROPOSITION 5.7. *For any $k \geqslant 1$, for any fixed query Q with no more than k product signs, for any input po-database D where each po-relation is a union of totally ordered relations, the dimension of the underlying poset of $Q(D)$ (under the* GEN *semantics) is at most $(k+1)$.*

Conversely, for any k-dimensional poset $(P,<)$ there exists a query Q with no more than k product signs, and a po-database D of totally ordered po-relations such that the underlying poset of $Q(D)$ (under the GEN *semantics) is $(P,<)$.*

Proof. Let us first deal with the case of totally ordered input relations. Use properties selection-projection, projection-union, and projection-product to rewrite Q so that projections are the outermost operators. Use properties selection-product and selection-union to move selections outwards. Now, use union-distributive so that Q is rewritten to an equivalent projection of selections of unions of products of the base relations and constant relations.

Clearly the products in this form cannot be more than k-ary. Hence, the resulting partial order is a union of products of (at most) $k+1$ po-relations which are totally ordered (whether they are the input relations or constant relations).

We now use Lemma C.1 for each product in the union to argue that its dimension is at most $k+1$, and conclude using the fact that, for any $d \geqslant 2$, any finite union $R = R_1 \cup_{\texttt{GEN}} \cdots \cup_{\texttt{GEN}} R_n$ of po-relations of dimension $\leqslant d$ has dimension $\leqslant d$. Indeed, create a realizer Z for the union by making the first total order of Z a concatenation of the first total order of the realizers for each R_i for $1 \leqslant i \leqslant n$ in increasing order, and making the other total orders of Z the concatenation of the other total orders of the realizers for each R_i for $n \geqslant i \geqslant 1$ in decreasing order. The correctness of this construction is immediate as order between two elements in any unioned relation R_i are correctly represented by the realizer of the union as they were correctly represented in the realizer for R_i; and any two elements in R_i and R_j for $i \neq j$ are incomparable as evidenced by the first and second total orders of Z (this is where we use the fact that $d \geqslant 2$).

If the input relations are unions of total orders instead of total orders, we can rewrite Q to replace the input relations with unions of input relations which we assume are total, and the same proof works.

Conversely, the fact that any such poset can be realized is a direct consequence of Lemma C.1. □

PROPOSITION 5.9. *For any $k \in \mathbb{N}$ and query Q, there is $k' \in \mathbb{N}$ (depending only on k and Q) such that for any po-database D of series-parallel po-relations of sp-height at most k, the underlying poset of $Q(D)$ (under the* LEX *semantics) is series-parallel with sp-height at most k', or empty.*

Proof. This claim follows immediately from the proof of Proposition 5.6, noting that an sp-tree T for the result of query evaluation can be obtained from sp-trees $T_1,\ldots,T_n$ from the input po-relations, and that the sp-height of T depends only on that of the T_i and on the query Q (and not, e.g., on the size of the T_i). □

C.2 Possible Transformations (Section 5.2)

COROLLARY 5.11. *There are transformations expressible in* GEN *but not in* LEX.

Proof. By Proposition 5.6 any transformation expressed by a LEX query is such that the image of a po-database of totally ordered relations is a series-parallel po-relation. So, to show that some transformations can be expressed by GEN but not by LEX, it suffices to provide an example of a query Q and po-database D such that $Q(D)$ is not a series-parallel po-relation when evaluated under the GEN semantics.

Consider Q the query $\sigma_\varphi(\mathbb{N}^{\leqslant 1} \times_{\texttt{GEN}} \mathbb{N}^{\leqslant 2})$ and D the empty po-database, where φ is the tuple predicate:

$$(.1 = 1 \wedge .2 = 0) \vee (.1 = 1 \wedge .2 = 1) \vee (.1 = 0 \wedge .2 = 1) \vee (.1 = 0 \wedge .2 = 2)$$

It is easily verified that $Q(D)$ is a po-relation with four tuples t_1, t_2, t_3 and t_4, with respective values $\langle 1,0\rangle$, $\langle 1,1\rangle$, $\langle 0,1\rangle$ and $\langle 0,2\rangle$, such that exactly the following comparability relations hold: $t_1 < t_2$, $t_3 < t_2$, $t_3 < t_4$. But this is exactly the N-shaped poset of [Möh89] which is an example of a non-series-parallel poset. Hence, $Q(D)$ is not series-parallel, proving the desired result. ☐

PROPOSITION 5.12. *For any distinguished relation names R and S, there is no query Q such that, for any po-database D, $Q(D)$ evaluates to $R \cup_{\texttt{CAT}} S$ under the* GEN *semantics.*

To prove Proposition 5.12, we first show the following lemma:

LEMMA C.2. *Let $v \in \mathscr{D}\backslash\mathbb{N}$ be a value. For any query Q that does not mention v, the following holds. Assume that no relation of a po-database D contains two tuples $t_1 < t_2$ such that either $t_1.i = v$ and $t_2.i \neq v$, or $t_1.i \neq v$ and $t_2.i = v$. Then $Q(D)$ under the* GEN *semantics has the same property.*

Proof. Let v_0 be any value of $\mathscr{D}\backslash\mathbb{N}$ that does not occur in Q We show the claim by induction on the query Q.

The base cases are the following:

- For the base relations, the claim is trivial by our hypothesis on D.
- For the empty and singleton constant expressions, the claim is trivial as they contain less than two tuples.
- For the $\mathbb{N}^{\leqslant i}$ constant expressions, the claim is immediate as $v \notin \mathbb{N}$.

Now for the induction step:

- For selection, the claim is immediate as the property to prove is maintained when taking subsets of tuples
- For projection, the claim is also immediate as the property to prove is maintained when reordering, copying or deleting attributes
- For union, the property is preserved as any comparability relation between tuples in the union implies that the two tuples must come from the same input relation, where the comparability relation preexists between the same tuples.
- We now show that the property is preserved for product (under the GEN semantics). Consider $Q(D) = Q_1(D) \times_{\texttt{GEN}} Q_2(D)$ where $Q_1(D)$ and $Q_2(D)$ satisfy the conditions, and assume that there are two tuples $(t_1,t_2) < (t_1',t_2')$ in $Q(R)$. We distinguish on whether $1 \leqslant i \leqslant \mathrm{a}(Q_1)$ or $\mathrm{a}(Q_1) < i \leqslant \mathrm{a}(Q_1) + \mathrm{a}(Q_2)$. For the first case, we thus have $t_1.i \neq t_1'.i$, as one of them is v and the other is $\neq v$. Thus, by definition of the product order in the GEN semantics, as $(t_1,t_2) < (t_1',t_2')$ we must have $t_1 \leqslant t_1'$, so necessarily $t_1 < t_1'$ in $Q_1(D)$, contradicting our assumption about $Q_1(D)$. The second case is symmetric. ☐

We now conclude with the proof of Proposition 5.12.

Proof. Let us assume that there exists a query Q capturing $\cup_{\texttt{CAT}}$. Consider a po-database D formed of $R = (\langle r\rangle)$ and $S = (\langle s\rangle)$ where r and s are distinct values of $\mathscr{D}$ that do not occur in Q. By our assumption about Q, we know that $Q(D)$ has only one possible world, namely $(\langle r\rangle, \langle s\rangle)$. However, by Lemma C.2, taking $v = r$, D had the desired property, but $Q(D)$ does not. So we have a contradiction, and $\cup_{\texttt{CAT}}$ cannot be thus captured. ☐

PROPOSITION 5.16. *For any finite sort $\mathrm{Sort}_{i,<_U}$, there is a PosRA query Q with distinguished relation name R such that, for any po-database D, $Q(D)$ under the* LEX *semantics evaluates to $\mathrm{Sort}_{i,<_U}(R)$ when it is defined.*

Proof. Consider a totally ordered relation R_U that describes the total order $(U, <_U)$. As U is finite, R_U is finite, and it is series-parallel because it is totally ordered, so by Proposition 5.5 there is a query Q_U with no inputs whose output is R_U.

Consider now the query Q: $\Pi_{2,\ldots,\mathrm{a}(R)+1}(\sigma_{.1=.(i+1)}(Q_U \times_{\texttt{LEX}} R))$. It is straightforward that $Q(D)$ is the desired relation: the selection and product (corresponding to a join) sort tuples first by their values at position i according to $Q_U()$ (under the assumption that the sort is defined), that is, R_U, with a stable order in case of ties, and the projection ensures that the tuples values are correct. ☐

COROLLARY 5.17. *For any domain U of size $\geqslant 2$ and total order $<_U$ on U, for any distinguished relation name R with arity $n \geqslant 2$ and position $1 \leqslant i \leqslant n$, there is no query Q such that, for any po-database D, $Q(D)$ under the* GEN *semantics evaluates to $\mathrm{Sort}_{i,<_U}(R)$ when it is defined.*

Proof. Fix U, $<_U$, n, and i. We show how $\cup_{\texttt{CAT}}$ on input relations of arity 1 can be implemented using Q, so the impossibility of implementing $\cup_{\texttt{CAT}}$ in the GEN semantics (Proposition 5.12) implies a contradiction (noting that, from the proof of this proposition, $\cup_{\texttt{CAT}}$ cannot be implemented even if we assume that the input relations have arity 1).

Indeed, let R and S be two relation names with arity 1, and let $a < b$ be two distinct values of U, and consider the query Q': $\Pi_{A'}(Q(\Pi_A([a] \times R) \cup_{\texttt{GEN}} \Pi_A([b] \times S)))$, where A is the sequence $s_1, \ldots, s_n$ with $s_j = 2$ for all $j \neq i$ and $s_i = 1$, and A' is the singleton sequence i_0 for some

$i_0 \neq i$. It is easy to see that $Q(D)$ under the GEN semantics evaluates to $R \cup_{\texttt{CAT}} S$. Indeed, $\Pi_A([a] \times R)$ is the relation R except that each tuple $t = \langle v \rangle$ is replaced by $\langle v, \dots, v, a, v, \dots, v \rangle$ with a at position i, and likewise for $\Pi_A([b] \times S)$. Now sorting ensures that we obtain $R \cup_{\texttt{CAT}} S$ up to the change of values, and the right values are recovered thanks to $\Pi_{A'}$.

Hence, we have implemented $\cup_{\texttt{CAT}}$ in GEN, and reached the desired contradiction. ☐

PROPOSITION 5.19. *For any semantics* X *satisfying* Ax *and PosRA query Q, if a transformation f is expressed by Q under semantics* X*, then f is monotone.*

Proof. Immediate by induction and by axiom consistency. ☐

PROPOSITION 5.20. *There is no semantics* X *satisfying* Ax *and query Q that capture the following monotone, generic transformation f: letting R be a distinguished relation name, for any po-database D,* $f(D) = \mathrm{Rel}(R)$.

Proof. Let us fix X and Q, assuming by contradiction that they have the desired property, and let us construct a counterexample po-database D consisting of a single relation R. We let R be the totally ordered relation $(\langle a \rangle, \langle a \rangle)$ where a is a constant not in $\mathbb{N}$ and that does not occur in Q.

We now show by induction that in the po-relation which is the result of evaluating any subexpression of Q, either a does not occur, or there are two comparable tuples that have the same value. For the base cases:

- The claim holds for the input relations: R has two comparable tuples that have the same value.
- The claim holds for the constant expressions, as a does not occur in them (remember that Q does not contain a).

Now, for the induction:

- The claim is immediate for the one possible semantics for projection dictated by axiom project: if a does not occur then a still does not occur after projection; if there are two comparable tuples with the same value then this is still the case after projection.
- The claim is immediate for the one possible semantics for selection dictated by axiom select: if a does not occur then this is preserved by selection, otherwise the two comparable tuples with the same value are treated in the same way by the selection.
- For the union $R_1 \cup R_2$, assuming that R_1 and R_2 satisfy the induction hypothesis, either a occurs in none of R_1 and R_2 and, by property bag, this is still true of the result. Otherwise, considering the two comparable tuples with the same value in one of the input relations (say R_1), we observe that by axiom union those two tuples must still be comparable (and have the same value) in the union, so $R_1 \cup R_2$ satisfies the property.
- For the product $R_1 \times R_2$, assuming again the induction hypothesis on R_1 and R_2, if neither of R_1 and R_2 contains a then the product does not by property bag. If one of them does, then if the other relation is empty, the output does not mention a by property bag. We distinguish on whether R_1 or R_2 contains a (and thus contains two comparable tuples with the same value); we assume it is R_1, as the case where it is R_2 is symmetric. Let k_1, k_2 be the position of the two comparable tuples of R_1. Consider the mapping ψ of axiom product: in every possible world of $R_1 \times R_2$, we must have $\psi(k_1, 1) < \psi(k_2, 1)$ and those two tuples have the same value. Hence, $R_1 \times R_2$ satisfies the property.

This proves that $Q(D)$ under X is not suitable, as $Q(D)$ should be $\mathrm{Rel}(R)$, which should both contain a and not have two comparable tuples. ☐

D. PROOFS FOR SECTION 6 (TOP-K)

D.1 Definition and General Results (Section 6.1)

PROPOSITION 6.3. *For any fixed query* $Q := \mathrm{top}_k(Q')$ *with* Q' *in the* GEN+LEX *semantics, one can compute the possible results of* $Q(D)$ *in PTIME in the input po-database D.*

Proof. Compute the po-relation $R := Q'(D)$ in PTIME (Theorem 4.7). Now, consider every list L of k tuples of R; for fixed Q and k, the number of such lists is polynomial in D. Now, for every such list, we can decide in PTIME if there is a linear extension of R whose first k elements match L, by adding the order constraints $l_1 < l_2$, $l_2 < l_3$, ..., $l_{k-1} < l_k$, and $l_k < x$ for all x in the domain of R that is not in L, and checking if the result is still a poset (i.e., if there is no cycle, which can be done in PTIME). ☐

THEOREM 6.5. *Top-k certainty is in PTIME for* GEN+LEX.

Proof. We compute in PTIME $R := Q(D)$ using Theorem 4.7 and we show that we can determine in PTIME whether L is the only possible world of $\mathrm{top}_k(R)$.

We say that the po-relation R is *serial* if there are two po-relations R_1 and R_2 such that R is the series composition of R_1 and R_2. We use Proposition 4.6 of [GM99] to compute in polynomial time a decomposition of R as the series composition of $R_1, \dots, R_n$, where the R_i are non-serial po-relations. In particular, if R was not serial, we have $n = 1$ and the decomposition is just $R_1 = R$.

Now, following this decomposition, write L as the series composition of $L_1, \dots, L_n$, such that each L_i contains as many elements as R_i (except for L_n which may contain strictly less, but at least one). Clearly, to solve the certainty problem for R and L, it suffices to solve it for each R_i and L_i. So we may assume without loss of generality that R is non-serial and $|L|$ contains at most as many elements as R.

Write $R = (ID, T, <)$. We say that an element $x \in ID$ is a *root* of R if there is no $y \in ID$ such that $y < x$. Let $l := T(r)$ be the value of a root r of R. If the value of all elements of R by T is l (in particular, if $ID = \{r\}$), it is easy to solve certainty: just check if L contains only value l. Otherwise, define the *ancestors* of an element $x \in ID$ as the set A_x of the elements of ID that precede them, and define the *ancestor number* of x as $n_x := |A_x|$. Let $y \in ID$ be an element with value $l' := T(y)$ such that $l' \neq l$, whose ancestor number n_y is minimal. We now prove that L is the only possible world for $\text{top}_{|L|}(R)$ iff L contains only l and $|L| \leqslant n_y$.

Clearly, if L is of the prescribed form, then it is a possible world (obtained by enumerating the ancestors of y in a suitable order), and it is the only possible world (if a different possible world can be achieved, then the first enumerated element whose value is not l violates the minimality of y). Conversely, assume that L is not of the prescribed form. If the first $\min(n_y, |L|)$ positions of L contain a different value than l, then it is not a possible world by minimality of n_y, as before. Otherwise, if $|L| > n_y$, we claim that L is not the only possible world, because there are two different possible worlds for $\text{top}_{|L|}(R)$, as follows.

We first deal with the case where $n_y = 0$. This means y is a root of R, so that R has one root with value l and one root with value l'. It is then clear that if $|L| > 0$ then L is not the only possible world.

Hence, assume that A_y is non-empty. We now justify that the set S_y of elements incomparable to y is not empty. Indeed, if it were, then R would be serial: it would be the serial composition of A_y, y itself, and its descendants if any. Now consider the set S'_y of the minimal elements of S_y, which is also non-empty; all immediate predecessors of elements of S'_y must be ancestors of y. We now claim that there is $z \in S'_y$ whose value $T(z)$ is l. To see why, assume by contradiction that it is not the case. By minimality of y, the ancestor number of all elements of S'_y must be at least n_y. But then, as all ancestors of elements of S'_y are in A_y, this means that for all $z \in S'_y$, $A_z = A_y$. This implies that R is serial, as it is the serial composition of the non-empty A_y on the one hand, and $\{y\} \cup S'_y$ and their descendants on the other hand: any element of A_y is less than y, less than elements of S'_y, less than descendants of $\{y\}$ because it is less than $\{y\}$, and less than elements of S_y because it is less than the elements of S'_y which are the minimal elements of S_y.

Hence we conclude that there is $z \in S'_y$ with $T(z) = l$. We thus obtain two possible worlds of $\text{top}_{|L|}(R)$: l repeated $n_y + 1$ times, by enumerating $A_y \cup \{z\}$, and l repeated n_y times and l', by enumerating $A_y \cup \{y\}$. This concludes the proof. ☐

THEOREM 6.6. *Top-k possibility is NP for* GEN+LEX *and NP-hard for both* GEN *and* LEX*, even assuming that each input po-relation is either unordered or totally ordered.*

Proof. We start by showing NP membership. Observe that we can just evaluate the fixed query Q on the input D, which is in PTIME by Theorem 4.7, and then guess a permutation of the elements of the resulting po-relation $R := Q(D)$ and check that it is indeed a linear extension of R and that its first $|L|$ values are exactly L. Of course the same argument applies if R is an arbitrary po-relation rather than the result of evaluating a query.

We now show hardness, making the assumption that relations of D are either totally ordered or unordered. Further, in our constructions, the candidate possible worlds L will always have a number of elements that is exactly the number of elements of $Q(D)$, and not less (i.e., we are showing hardness of the instance possibility problem as in Definition 6.16, which implies that of the top-k possibility problem for unbounded k).

We first assume that $Q(D)$ is evaluated according to the GEN semantics. The reduction is from the UNARY-3-PARTITION problem [GJ79]: given $3m$ integers $E = (n_1, \ldots, n_{3m})$ written in unary and a number B, decide if the integers can be partitioned in triples such that the sum of each triple is B. We reduce an instance $\mathscr{I}$ of UNARY-3-PARTITION to an instance of the possibility problem for a certain po-relation R and a certain totally ordered relation L.

Let $\mathsf{s}, \mathsf{n}, \mathsf{e}$ be three distinct values from $\mathscr{D}$ (they stand for *start* values, *inner* values, and *end* values). We set R' to be a unary totally ordered relation with tuples encoding E in the following fashion: for $1 \leqslant i \leqslant 3m$, one tuple t^i_1 with value s, n_i tuples t^i_j (with $2 \leqslant j \leqslant n_i + 1$) with value n, and one tuple $t^i_{n_i+2}$ with value e (R' is the total order formed by concatenating the $3m$ sequences of length $n_i + 2$). We alternatively number the tuples of R' as $t'_1, \ldots, t'_{2\times 3m + \sum_{1 \leqslant i \leqslant 3m} n_i}$. We consider the query $Q : \Pi_2(\mathbb{N}^{\leqslant 3m-1} \times_{\texttt{GEN}} R')$, and denote by R the po-relation obtained by evaluating Q. Note that as all tuples of the unary query result have value in $\{\mathsf{s}, \mathsf{n}, \mathsf{e}\}$.

We create a first totally ordered sequence L_1 in the following way: for $1 \leqslant i \leqslant 3m$, $3m - i - 1$ repetitions of the sublist formed of s, n_i tuples with value n, and one tuple with value e.

We write $L_1^{\leqslant k}$ for the prefix of L_1 of length k, for $0 \leqslant k \leqslant |L_1|$. We say that $L_1^{\leqslant k}$ is a *whole prefix* if either $k = 0$ (that is, the empty prefix) or the k-th symbol of L_1 has value e. We say that a linear extension L' of R *realizes* $L_1^{\leqslant k}$ if the sequence of its k-th first values is $L_1^{\leqslant k}$, and that it realizes L_1 if it realizes $L_1^{\leqslant |L_1|}$. When L' realizes $L_1^{\leqslant k}$, we call the *matched* elements the elements of R that occur in the first k positions of L', and say that the other elements are *unmatched*. We call the *group-i elements* the elements whose first component before projection were $i - 1$.

We first observe that for any linear extension L' realizing $L_1^{\leqslant k}$, for all i, the group-i unmatched elements must be all of the form t'_j for $j > k_i$ for some k_i, with $\sum_i k_i = k$. Indeed, if they did not form a suffix, then some order constraint of R would have been violated in the construction of L'.

Second, we say that we are in a *whole situation* if for all i, the value of element t'_{k_i+1} is either undefined (i.e., there are no group-i unmatched elements, which means $k_i = |R'|$) or it is s. In such a situation, we observe that k_i is of the form $\sum_{i=1}^{l} (n_i + 2)$ for some l and we set $S_i := \biguplus_{1 \leqslant i \leqslant l} \{\{n_i\}\}$ to be the bag of *group-i consumed integers*. The *group-i remaining integers* are $E \backslash S_i$ (seeing E as a multiset).

We now prove the following claim: for any linear extension L' realizing L_1, we are in a whole situation, and the multiset union $\biguplus_{1 \leqslant i \leqslant 3m} S_i$ is equal to the multiset obtained by repeating integer n_i of E $3m - i$ times for all $1 \leqslant i \leqslant 3m$.

We prove the first part of the claim by showing it for all whole prefixes $L_1^{\leqslant k}$, by induction on k. It is certainly the case for $L_1^{\leqslant 0}$ (the empty prefix). Now, assuming that it holds for prefixes of length up to l, to realize a whole prefix $L^{\leqslant l'}$ with $l' > l$, you must first realize a strictly shorter whole prefix $L^{\leqslant l''}$ with $l'' \leqslant l$ (take it to be of maximal length), so by induction hypothesis you are in a whole situation when realizing $L^{\leqslant l''}$. Now to realize the whole prefix $L^{\leqslant l'}$, the sequence of additional values to realize is s, a certain number of n's, and e, and it is easily seen that this must bring you from a whole situation to a whole situation: since there is only one s in this sequence of additional values, there is only one group-i such that an s value becomes matched; now, to match the additional n's and e, only this particular group-i can be used, as any first unmatched element of a group-j with $j \neq i$ is s. Hence the claim is proven.

To prove the second part of the claim, observe that whenever we go from a whole prefix to a whole prefix by additionally matching s, n_j times n, and e, then we add to S_i the integer n_j. So the claim holds by construction of L_1.

We now create a second totally ordered sequence L_2 in the same way as L_1, except we replace the $3m-i$ repetitions of the sublist by $i-1$ repetitions. A similar argument shows that for any linear extension of R whose first $|L_1|$ tuples achieve L_1 and last $|L_2|$ tuples achieve L_2, the group-i unmatched elements are a contiguous sequence t'_j for $k_i < j < m_i$ for some m_i. In addition, if $k_i < m_i - 1$, t'_{k_i} has value e and t'_{k_i} has value e, and the unmatched values R_i (defined in an analogous fashion) are a multiset corresponding exactly to $\{\{n_1, \ldots, n_{3m}\}\}$. So the unmatched elements are formed of $3m$ totally ordered sequences of length $n_i + 2$ for $1 \leqslant i \leqslant 3m$, of the form s, n_i times n, and e, with a certain order relation between the sequences.

But we now notice that we can clearly achieve L_1 by picking the following, in that order: for $1 \leqslant j \leqslant 3m$, for $1 \leqslant i \leqslant j-1$, all the t^j_l's of the group-i. Similarly, for L_2, we can pick the following, in *reverse* order: for $3m \geqslant j \geqslant 1$, for $3m \geqslant i \geqslant 3m - j + 2$, all the t^j_l's of the group-i. When picking elements this way, the unmatched elements are $3m$ totally ordered sequences (one for each group-i, and one for each j) that are incomparable with each other. We denote these elements u^j_l with $1 \leqslant j \leqslant 3m$ iterating over totally ordered sequences, and $1 \leqslant l \leqslant n_j + 2$ iterating within each sequence. Let T be the sub-po-relation of R that consists of exactly these elements: it is the parallel composition of $3m$ total orders, one for each j, consisting of the u^j_l's.

We claim that for any sequence L', $L := L_1 L' L_2$ is a possible world of R if and only if L' is a possible world of T. The "only if" direction can be proved with the construction above. The "if" direction comes from the fact that T is the *least constrained* po-relation for the unmatched sequences, since the order within each sequence is known to be total.

We now consider the sequence L' that consists of the following tuples, in order, repeated m times: three tuples with value s, B tuples with value n, three tuples with value e. We claim that L' is a possible world of T iff $\mathscr{I}$ is a positive instance to the UNARY-3-PARTITION problem, concluding the reduction. To see why, observe that there is a bijection between 3-partitions and linear extensions of T which achieve L', in the following sense: from a 3-partition (s^i_1, s^i_2, s^i_3) for $1 \leqslant i \leqslant m$ (with $n_{s^i_1} + n_{s^i_2} + n_{s^i_3} = B$ for all i, and each n_l being used exactly once), realize L' by picking successively, for $1 \leqslant i \leqslant n$, $u^{s^i_1}_1$, $u^{s^i_2}_1$ and $u^{s^i_3}_1$ that have value s, and then the B tuples for $1 \leqslant p \leqslant 3$, $2 \leqslant j \leqslant n_{s^i_p} + 1$, $u^{s^i_p}_j$ that have value n, and last the $t^{s^i_p}_{n_{s^i_p}+2}$ for $1 \leqslant p \leqslant 3$ that have value e. Conversely, it is easy to build a 3-partition from any linear extension to achieve L' from T. The construction of L and R is clearly in polynomial time, which concludes the proof.

We last show that hardness also holds for `LEX`. The reduction is the same, except that we take relation R_1 to be *unordered* rather than totally ordered. The proof adapts, as in the proof for `GEN` we only used the fact that $t'_j < t'_k$ for $j < k$ within a group-i, but never the comparability across groups. □

D.2 Tractability for Possibility (Section 6.2)

PROPOSITION 6.8. *For any poset $(P,<)$, an ua-partition of minimal cardinality can be computed in PTIME.*

We first show the following lemma:

LEMMA D.1. *For any poset $(P,<)$ and undistinguishable antichains A_1, A_2 such that $A_1, A_2 \subseteq P$ and $A_1 \cap A_2 \neq \emptyset$, $A_1 \cup A_2$ is an undistinguishable antichain.*

Proof. We first show it is an antichain. Proceed by contradiction, and let $x, y \in A_1 \cup A_2$ such that $x < y$. As A_1 and A_2 are antichains, we must have $x \in A_1 \backslash A_2$ and $y \in A_2 \backslash A_1$, or vice-versa. Assume the first case, the second case is symmetric. As A_1 is an undistinguishable set, letting $z \in A_1 \cap A_2$ (which exists by our hypothesis), as $x < y$ and $x \in A_1$, we have $z < y$. But $z \in A_2$ and $y \in A_2$, which contradicts the fact that A_2 is an antichain.

We next show it is an undistinguishable set. Let $x, y \in A_1 \cup A_2$ and $z' \in P \backslash (A_1 \cup A_2)$, assume that $x < z'$ and show that $y < z'$. As A_1 and A_2 are undistinguishable sets, this is immediate unless $x \in A_1 \backslash A_2$ and $y \in A_2 \backslash A_1$, or vice-versa. We again assume the first case as the second one is symmetric. Now considering again $z \in A_1 \cap A_2$, we know that $z < z'$ as A_1 is an undistinguishable set, so that $y < z'$ as A_2 is an undistinguishable set, proving the result. The fact that $z' < x$ implies $z' < y$ is proved in a similar fashion. □

We now prove the main claim:

Proof. Start with the trivial partition, and for every pair of items, see if their current classes can be merged (i.e., merge them, and check in PTIME if it is an antichain, and if it is an undistinguishable set). The process is in PTIME.

Now assume that there is a partition of strictly smaller cardinality. There has to be a class c of this partition which intersects two different classes $c_1 \neq c_2$ of the original partition, otherwise it is a refinement of the previous partition and so has a higher number of classes. But now $c \cup c_1$ and $c \cup c_2$, and thus $c \cup c_1 \cup c_2$, hence $c_1 \cup c_2$, are undistinguishable antichains, contradicting the fact that c_1 and c_2 could not be merged. □

PROPOSITION 6.9. *For any constant c, top-k possibility is in PTIME for the `GEN+LEX` semantics if we assume that all input po-relations have ua-width at most c.*

Proof. The proof is by observing that, for any such c and Q, there is a constant c_Q depending only on Q and c such that, for any input po-database D whose relations have ua-width $\leqslant c$, $Q(D)$ has ua-width at most c_Q.

We compute the bound by induction. For the base cases: the input relations have ua-width at most c, the constant relations have constant ua-width with the trivial ua-partition. For the induction: projection clearly does not change anything, selection may only reduce the ua-width as it removes tuples, the union of two relations with bounds c_1 and c_2 has bound $c_1 + c_2$, and for product the bound is $\prod c_i$ where c_i is the bound for the i-th input relation. (The observation is that the product can be computed on the relations quotiented by the indistinguishability equivalence relation, and then re-populating the relations with the members of the classes; this works both for the `GEN` and `LEX` semantics.)

We now evaluate $Q(D)$ in PTIME by Theorem 4.7, and it suffices to show we can solve the possibility problem in PTIME for a po-relation R and candidate possible world L, under the assumption that R has ua-width at most c. Let P be a ua-partition of width c of R. We complete L using "wildcard" elements that can match any tuple, such that $|L|$ is the number of elements of R. (Of course, if $|L|$ is strictly larger than the number of elements of R, we can immediately reject.)

If there is a way to realize L as a possible world of R, we call the *finishing order* the permutation π of $\{1, \ldots, c\}$ obtained by considering, for each class c_i of P, the largest position n_i of $\{1, \ldots, |L|\}$ to which an element of c_i is mapped, and sorting the class indexes by ascending finishing order. We say we can realize L with finishing order π if there is a realization of L whose finishing order is π. Hence, it suffices to check, for every possible permutation π, whether L can be realized from R with finishing order π, and as the number of finishing orders depends only on c, this is only a constant factor in the complexity in R.

We now claim that to determine whether L can be realized with finishing order π, the following greedy approach works. Read L linearly. At any point, maintain the set of elements of R which have already been used (distinguish the *used* and *unused* elements; initially all elements are unused), and distinguish permutation classes in *exhausted classes*, the ones where all elements have been mapped; *open classes*, the ones where all smaller elements have been mapped; and *blocked classes*, the ones where some smaller element is not mapped (initially the open classes are those which are roots in the poset obtained from the underlying poset of R by quotienting by the equivalence relation induced by P; and the others are blocked).

When reading a value v from L, consider all open classes. If none of these classes have an unused element with value v (or, if v is a wildcard, any element), reject. Otherwise, take the open class with the lowest finishing time (i.e., appears the earliest in π) that has such an element, and use an arbitrary suitable element from it. (Update the class to be *exhausted* if it is, in which case update from *blocked* to *open* the classes that must be). Once L is read, accept iff all elements are used (i.e., all classes are exhausted).

It is clear by construction that if this greedy algorithm accepts then it has found a way to match L in R; indeed all matches that it performs satisfy the values and the order relations of R. It must now be proved that if L can be matched in R with finishing order π, then the algorithm accepts when considering π. To do so, we must show that if there is such a match, then there is such a match where all elements are mapped, following what the greedy algorithm does, to a suitable element in the open class with smallest finishing time (we call this a *minimal* element); if we do, then we justify the existence of a match that the algorithm will construct.

Now, to see why this is possible, consider a match m and take the smallest element t of L mapped to a non-minimal element s in class c in R. Consider a minimal element s' in class c' instead in R, with the same value, and t' the element to which it is mapped (and $t <_L t'$). Consider the match m' obtained by mapping t to s' and t' to s. The new match m' still satisfies conditions on the values, now, let us assume that an order constraint was violated. It must be for $t <_L t'' <_L t'$, and s'' in R to which t'' is mapped must be $> s$ (this is the only possible violation). Now if s'' was thus mapped it means that the class c of s was exhausted when reaching t'' in L (so we could match t'' to s''), but because t' was not reached yet, the class c' of s' was not exhausted yet, but this contradicts the fact that c' finishes before c (according to π). Hence, we can rewrite m to be of the form of what is performed by the greedy algorithm. This concludes the proof. □

PROPOSITION 6.11. *Top-k possibility is NP-hard for the* `GEN` *semantics even if all input relations are assumed to be totally ordered.*

Proof. The proof is the same as the hardness proof in Theorem 6.6, noting that all input relations used in the reduction are totally ordered. □

PROPOSITION 6.13. *For any constant $c \in \mathbb{N}$, top-k possibility is in PTIME for the* `LEX` *semantics if all input po-relations are series-parallel and each of their underlying posets has an sp-tree with breadth at most c.*

Proof. The first part of the claim is to observe that we can obtain in PTIME an sp-tree for $Q(D)$ that has constant breadth (i.e., depending only on Q and c). This is done as in the proof of Proposition 5.6.

The second part of the claim is to show that the possibility problem can be solved in PTIME on any input series-parallel po-relation provided with an sp-tree T of constant breadth. Call "critical" a topmost node that has no "parallel" descendant. The order on the elements of the subtree rooted at such nodes is therefore a total order, so we see T as a constant-size tree whose leaves are total orders.

We solve the possibility problem by a dynamic algorithm whose state is, for every such total order, the size of the prefix of the total order that has already been matched to elements of L. At each state, we say the problem can be solved if it can be solved by matching the first remaining element of L to some first remaining element in one of the total orders where this is possible according to the constraints of T. □

PROPOSITION 6.15. *Top-k possibility is in PTIME for the* `GEN+LEX` *semantics if we assume that all tuple values in $Q(D)$ are unique.*

Proof. We compute $Q(D)$ in PTIME using Theorem 4.7. It suffices to test if L is a prefix of a linear extension of $R := Q(D)$, and this can be tested in linear time by creating R' obtained by adding all the minimal comparability pairs imposed by L (of which there are linearly many) to R, as in the proof of Proposition 6.3, noting that the unique values means that all comparability pairs of L can be matched in L in a unique way. L is a possible world of R if and only if R' has no cycles (= its transitive closure is still antisymmetric), which can be tested in linear time by computing the strongly connected components of R' and checking that they are all trivial. □

D.3 Other Problems (Section 6.3)

PROPOSITION 6.17. *The instance possibility problem is in PTIME for the* GEN+LEX *semantics assuming that all input relations are either unordered or totally ordered, and that the fixed query does not use any product operator.*

Proof. Use properties selection-union, selection-projection and projection-union to rewrite Q as a projection of a selection of a union of relations of D and constant relations. Now, evaluate $R := Q(D)$ in PTIME by Theorem 4.7, and represent it as a union of a constant number of totally ordered relations $L_1, \ldots, L_n$, and one unordered relation P.

We must now decide whether L is a possible world of the result of the query evaluation, namely, $L \in pw(R)$ (using the fact that $|L| = |\mathrm{Rel}(R)|$, so that $\mathrm{top}_k(R) = R$). We first check whether the domains of L and of R are the same (this is what would not work if we had $|L| < |\mathrm{Rel}(R)|$).

We now use a dynamic algorithm. The state of the algorithm is the position $(p_1, \ldots, p_n, p)$ indicating the number of tuples of $L_1, \ldots, L_n$ matched to L and the number of tuples of L that have been considered. When reading one tuple from L, we increment p, and we can progress in one of the L_i's the totally ordered relations if the tuple value matches (increasing the corresponding p_i), or make no progress at all in the R_i's (which corresponds to matching the tuple from L with an element of U). The initial state is $(0, \ldots, 0, 0)$: we have read no tuple from L and matched none of the tuples in the L_i's. We succeed in the state $(p_1, \ldots, p_n, p)$ with $p = |L|$ and $p_i = |L_i|$ for all i: this implies that all tuples of L were read and all tuples of the L_i's were matched, and by cardinality and equality of the domains this implies that all tuples of U were successfully matched to L as well. As the number of states is polynomial in the input L and $Q(D)$ which is polynomial in D, the dynamic algorithm runs in polynomial time. □

PROPOSITION 6.18. *The problem of rank possibility and rank certainty on any po-relation R is in PTIME.*

Proof. Given a po-relation R, we can compute in PTIME, for every element x, its *earliest index* $i^-(x)$, which is its number of ancestors plus one, and its *latest index* $i^+(x)$, which is the number of elements of R minus the number of descendants of x. It is easily seen that for any element x, there is a linear extension of R where x appears at position $i^-(x)$, or at position $i^+(x)$, or in fact at any position of $[i^-(x), i^+(x)]$, the *interval* of x.

Hence, rank possibility and rank certainty for tuple t and rank k can be decided by checking whether some element of the order whose interval contains k has value t, or whether all such elements have value t. □

PROPOSITION 6.19. *The sublist possibility problem is NP-complete in the input po-relation R and candidate sublist L.*

Proof. For NP membership, it suffices to guess a mapping from L to R, add the corresponding order constraints to R: letting $t_1, \ldots, t_n$ be the elements of R to which elements $1, 2, \ldots, n$ of L are mapped, we add the order constraints $t_1 < \ldots < t_n$. Now, check if the result is still a poset. If it is, then it has a linear extension that witnesses that L is a sublist of a possible world of R.

For NP-hardness, considering an algorithm $\mathscr{A}$ to solve the sublist possibility problem for an input R and L, we can solve the instance possibility problem for a fixed query Q and an input po-database D and L by computing $R := Q(D)$ in PTIME by Theorem 4.7 and checking whether L is a possible sublist of R using $\mathscr{A}$: L is a possible sublist of R iff it is a possible world of R as $|L| = |\mathrm{Rel}(R)|$. □

PROPOSITION 6.20. *We can solve the sublist possibility problem in PTIME in the input po-relation R and sublist L, if the length of L is assumed to be bounded by a constant.*

Proof. As in Proposition 6.3, there are polynomially many choices of tuples $t_1, \ldots, t_n$ in R to be matched to L, so we can test each one of them, and it suffices to check whether the addition of the order constraints $t_1 < \cdots < t_n$ still yields a poset. □

E. PROOFS FOR SECTION 7 (DUPLICATE ELIMINATION)

PROPOSITION 7.6. *For every oi-relation R, the following holds:* $\mathrm{dupElim}(R)$ *either completely fails or its result may be captured by a non-empty oi-relation.*

Proof. We need to show that the obtained possible worlds share an underlying (unordered) relation. Since every world in the result contains exactly a single copy of every tuple in R, this property is clearly satisfied: for every $W \in R$, $\mathrm{Rel}(W)$ is the result of "standard" duplicate elimination on the domain of R. □

PROPOSITION 7.8. *For any po-relation R,* $\mathrm{dupElim}(pw(R))$ *completely fails iff* G_R *has a cycle.*

Proof. We first show that the existence of a cycle implies complete failure of dupElim. Let $id'_1, \ldots, id'_n, id'_1$ be a simple cycle of G_R. For all $1 \leqslant i \leqslant n$, there exists $id_{1i}, id_{2i} \in id'_1$ such that $id_{2i} < id_{1(i+1)}$ and $T(id_{2i}) \neq T(id_{1(i+1)})$ (with the convention $id_{1(n+1)} = id_{11}$).

Let L be a possible world of R. Assume by contradiction that for all $1 \leqslant i \leqslant n$, id'_i forms a u-subset of L. Let us show by recurrence on j that for all $1 \leqslant j \leqslant n$, $id_{21} \leqslant_L id_{2j}$. The base case is trivial. Assume this holds for j and let us show it for $j+1$. Since $id_{2j} < id_{1(j+1)}$, we have $id_{21} \leqslant id_{2j} <_L id_{1(j+1)}$. If $id_{2(j+1)} <_L id_{21}$, then $id_{2(j+1)} <_L id_{21} <_L id_{1(j+1)}$ with $T(id_{2(j+1)}) = T(id_{1(j+1)}) \neq T(id_{21})$ (the non-equality is because the cycle is simple), which contradicts the fact that $id'_{(j+1)}$ is a subset). This proves the induction case. Now $id_{21} \leqslant_L id_{2j} <_L id_{1_n}$, which contradicts the fact that id'_1 is a u-subset. Thus, dupElim fails in every possible world of R.

Conversely, let us assume G_R is acyclic. Consider a topological sort of G_R as $id'_1, \ldots, id'_n$. For $1 \leqslant j \leqslant n$, let L_j be a linear extension of the poset $(id'_j, <_{id'_j})$. Let L be the concatenation of $L_1, \ldots L_n$. We claim L is a linear extension of R in which dupElim does not fail; this latter fact is clear by construction of L. Now, let $id_1 < id_2$ in R. Either for some $1 \leqslant j \leqslant n$, $id_1, id_2 \in id'_j$ and then $id_1 <_{L_j} id_2$ by construction which means $id_1 <_L id_2$; or they are in different classes id'_{j_1} and id'_{j_2} and this is reflected in G_R, which means that $j_1 < j_2$ and $id_1 <_L id_2$. □

COROLLARY 7.9. *For every po-relation R, one can determine in PTIME if* $\mathrm{dupElim}(pw(R))$ *completely fails; if it does not, one can compute in PTIME a po-relation* R' *such that we have* $pw(R') = \mathrm{dupElim}(pw(R))$.

Proof. We first observe that G_R can be constructed in PTIME, and that testing that G_R is acyclic is also done in PTIME. Thus, we can determine in PTIME whether $\mathrm{dupElim}(pw(R))$ fails.

If it does not, we let $\mathrm{G}_R = (ID', E)$ and construct R' as $(ID', T', <')$ where $T'(id')$ is the unique $T'(id)$ for $id \in id'$ and $<'$ is the transitive closure of E, which is asymmetric because G_R is acyclic. Observe that $\mathrm{Rel}(R')$ is the set of all tuples within the bag $\mathrm{Rel}(R)$.

Now, it is easy to check that $pw(R') = \mathrm{dupElim}(pw(R))$. Indeed, any possible world L of R' can be achieved in $\mathrm{dupElim}(pw(R))$ by considering, as in the proof of Proposition 7.8, the possible world of R obtained following the topological sort of G_R defined by L. Conversely, any possible world L of $\mathrm{dupElim}(pw(R))$ is the result of dupElim on a possible world L' of R where, for every tuple value, all occurrences of that value in L' are u-subsets; so, as they are contiguous and the order relations reflected in G_R must be respected, they are defined by a topological sort of G_R, which can also be obtained as the corresponding linear extension of R'. □

F. REFERENCES

[ART14] B. Alexe, M. Roth, and W.-C. Tan. Preference-aware integration of temporal data. Technical Report UCSC-SOE-14-04, UCSC, 2014.

[BGR97] D. Bechet, P. d. Groote, and C. Retoré. A complete axiomatisation for the inclusion of series-parallel partial orders. In *RTA*, 1997.

[GJ79] M. R. Garey and D. S. Johnson. *Computers And Intractability. A Guide to the Theory of NP-completeness*. W. H. Freeman, 1979.

[GM99] S. Grumbach and T. Milo. An algebra for pomsets. *Inf. Comput.*, 150(2), 1999.

[Hir55] T. Hiraguchi. On the dimension of orders. *Sci. rep. Kanazawa Univ.*, 4(01), 1955.

[Möh89] R. H. Möhring. Computationally tractable classes of ordered sets. In *Algorithms and Order*. Springer, 1989.

[Øre62] O. Øre. *Theory of Graphs*, chapter 10. AMS, 1962.