Querying Probabilistic Preferences in Databases

Batya Kenig
Technion
Haifa, Israel
batyak@cs.technion.ac.il

Benny Kimelfeld
Technion
Haifa, Israel
bennyk@cs.technion.ac.il

Haoyue Ping
Drexel University
Philadelphia, USA
hp354@drexel.edu

Julia Stoyanovich
Drexel University
Philadelphia, USA
stoyanovich@drexel.edu

ABSTRACT

We propose a novel framework wherein probabilistic preferences can be naturally represented and analyzed in a probabilistic relational database. The framework augments the relational schema with a special type of a relation symbol—a preference symbol. A deterministic instance of this symbol holds a collection of binary relations. Abstractly, the probabilistic variant is a probability space over databases of the augmented form (i.e., probabilistic database). Effectively, each instance of a preference symbol can be represented as a collection of parametric preference distributions such as Mallows. We establish positive and negative complexity results for evaluating Conjunctive Queries (CQs) over databases where preferences are represented in the Repeated Insertion Model (RIM), Mallows being a special case. We show how CQ evaluation reduces to a novel inference problem (of independent interest) over RIM, and devise a solver with polynomial data complexity.

CCS Concepts

•**Mathematics of computing** → **Probabilistic inference problems;** •**Information systems** → *Database design and models;*

Keywords

Probabilistic Databases; Probabilistic Preferences; Ranking Distributions; Repeated Insertion Model

1. INTRODUCTION

Preferences are statements about the relative quality or desirability of items. Ever larger amounts of preference information are being collected and analyzed in a variety of domains, including recommendation systems [4, 31, 33] and polling and election analysis [10, 15, 16, 29]. Preference datasets are also abundant in bioinformatics, where information about the relative expression levels of genes in a cell may provide evidence that groups of genes are involved in a common biological pathway [1, 22, 34].

Preferences are often inferred from indirect input (e.g., a ranked list may be inferred from individual choices), and are therefore uncertain in nature. This motivates a rich body of work on uncertain preference models in the statistics literature; see Marden [28] for a comprehensive reference. More recently, the machine learning community has been developing methods for effective modeling and efficient inference over preferences [3, 7, 11, 15, 17, 19, 23–25], with the Mallows model [27] receiving particular attention [11, 12, 25, 32].

In this paper, we take the position that preference modeling and analysis should be accommodated within a general-purpose probabilistic database framework. The framework presented here is based on a deterministic concept that we proposed in a past vision paper [18]. There, a *preference schema* is a relational schema with some relation symbols marked as *preference symbols* (and others as *ordinary symbols*). Figure 1 gives an example of an instance of a preference database, with the ordinary symbols **Candidates** and **Voters** and the preference symbol **Polls**. An instance (table) over a preference symbol represents a collection of preference relations, called *sessions*, over a set of elements, called *items*. The signature (attribute list) of a preference symbol distinguishes between the session identifier, the left-hand-side item and the right-hand-side item. For example, the tuple (`Ann`, `Oct-5`; `Sanders`; `Clinton`) in **Polls** denotes that in the session of the voter Ann on October 5th, the candidate Sanders is preferred to the candidate Clinton.

In our past proposal [18] we focused on representing deterministic preferences, and emphasized the importance of scalable and usable support for operations on preference data. These include relational operators and common analytics such as rank aggregation, clustering, and transitive closure, which we called "preference-to-preference functions." In the present work we focus on a different angle of the story—handing *uncertain* preferences. We develop a probabilistic representation of preferences within a *probabilistic preference database*, or *PPD* for short.

A PPD may store the probability space explicitly, or it may represent it compactly using a parametric model. The PPD framework leads to novel theoretical problems and algorithms, as we illustrate in a specific kind of PPDs, where each session is represented as an independent *Repeated Insertion Model* (RIM) [12]. RIM specifies a probabilistic ranking (linear order) via a generative process that reorders

PODS'17, May 14-19, 2017, Chicago, IL, USA

DOI: http://dx.doi.org/10.1145/3034786.3056111

a *reference ranking* by repeatedly inserting the items into the random ranking in a left-to-right scan. RIM generalizes various distributions over rankings, such as the well-known *Mallows* model [27], the *generalized Mallows* model [13] and the *multistage ranking* model [14].

In the PPD representations we explore, termed **RIM**-PPD, each session is associated with the parameters of a RIM model—the reference ranking and the dispersion parameter that determine the insertion probabilities. A **RIM**-PPD represents a probability space over preference databases, where a possible world is obtained by independently sampling a preference from the model of each session. Figure 2 gives an example of an instance of a **MAL**-PPD, which is a particular kind of a **RIM**-PPD. This representation compactly encodes the probability space of each session in **Polls** from Figure 1 by associating it with a Mallows model.

We study the data complexity of evaluating Conjunctive Queries (CQs) over **RIM**-PPDs. We focus on CQs to which we refer as *itemwise*. Intuitively, these are CQs where items are connected only through preferences. We first show a natural fragment of CQs where the itemwise CQs are *precisely* the CQs in which query evaluation can be done in polynomial time (under standard complexity assumptions). Here, *query evaluation* is meant in the sense of *probabilistic databases* [35], where the task is to compute the marginal probability of every possible answer. In the fragment we consider, we prove that every query that is *not* itemwise is actually #P-hard, and therefore, we establish a dichotomy in complexity.

The main analytical contribution of this paper is a polynomial-time algorithm for evaluating itemwise CQs. Interestingly, such CQs translate into a natural (and novel) inference problem over RIM. In this problem, every item is associated with one or more labels (e.g., "democratic" party or "comedy" genre), and the goal is to compute the probability that a graph pattern (or equivalently a partial order) over these labels *matches* the random ranking; that is, there exists a mapping from the nodes of the pattern to the items of the ranking such that labels (pattern nodes) and preferences (pattern edges) are preserved. The algorithm deploys dynamic programming, following a sequence of nontrivial reductions. We further describe how this algorithm can be extended to support claims about minimal and maximal indices of labels, and therefore compute the probabilities of events such as "the top-3 preferred movies include a Hitchcock movie" and "every young democratic candidate is preferred to every republican one." We are motivated by queries for recommendations that seek rankings subject to constraints on diversity.

As we illustrate in the paper, our labeled RIM model, and in particular our inference algorithm, are of interest independently from the PPD framework. We share motivation with recent research on probabilistic preferences, where the goal is in general to reason about distributions over rankings [8, 25, 26]. Existing inference algorithms apply to queries expressed as a partial order over individual items (e.g., compute the probability that `Sanders` is preferred to both `Clinton` and `Trump`, and that `Clinton` is preferred to `Trump`), but give only limited insight into preferences over groups of items (that we capture by labels). More recently, ranking distributions over labeled items have been introduced [17, 30] with the goal of using labels to enrich the expressiveness of distribution models.

There is a large body of work on supporting preference operators, such as top-k and partial orders, in relational databases. For example, Kießling [21] proposes to extend relational algebra with richer preference specifications than is supported by "ORDER BY" and "LIMIT", while Arvantis and Koutrika [2] demonstrate that preference operators can be implemented efficiently in scope of an RDBMS. In a nutshell, the goal of their work is to elegantly express and efficiently compute user preferences over the results of relational queries. In contrast, we focus on representing and querying databases in which preferences appear as data.

To summarize, in this paper we make the following contributions. First, we develop the framework of a probabilistic preference database. Second, we give a dichotomy in complexity for a fragment of CQs over **RIM**-PPDs. Third, we present an algorithm for evaluating itemwise CQs over **RIM**-PPDs. Fourth, we embark on the analysis of a novel and natural inference problem over RIM with labels. Lastly, we present an inference algorithm for querying labeled RIM.

2. PRELIMINARIES

In this section we set the basic terminology and notation that we use throughout the paper.

2.1 Database Model

A *relation signature* is a finite sequence α of distinct *attributes*. When there is no risk of ambiguity, we view α as a *set*, rather than *sequence*, of attributes. A *tuple* t over a relation signature α is a mapping from (the attribute set of) α to atomic values (e.g., numbers and strings). If $\alpha = (A_1, \ldots, A_k)$, then we identify a tuple t over α with the sequence $(t(A_1), \ldots, t(A_k))$. An *instance* over α is a finite set of tuples over α. A *relational schema* (or just *schema* for short) is a finite set $\mathbf{S}$ of *relation symbols*, where every relation symbol R is associated with a relation signature that we denote by $sig(R)$. We say that R is *k-ary* if $sig(R)$ consists of k attributes. A *database* D over a schema $\mathbf{S}$ associates with each relation symbol R an instance over $sig(R)$; we denote this instance by R^D.

A *query* Q over a schema $\mathbf{S}$ is associated with a relation signature that we denote by $sig(Q)$, and it maps every database D over $\mathbf{S}$ into an instance over $sig(Q)$; we denote this instance by $Q(D)$. A query Q is *k-ary* if $sig(Q)$ is of length k. A query Q of arity 0 is said to be *Boolean*, and in this case the result $Q(D)$ is either empty, corresponding to **false**, or the singleton $\{()\}$, corresponding to **true**. The statements $Q(D) = \{()\}$ and $Q(D) = \emptyset$ are also written as $D \models Q$ and $D \not\models Q$, respectively.

As a special case of a query, a *Conjunctive Query* (*CQ*) is represented by an expression of the form

$$Q(\mathbf{x}) \leftarrow \varphi_1(\mathbf{x}, \mathbf{y}), \ldots, \varphi_m(\mathbf{x}, \mathbf{y})$$

where $\mathbf{x}$ and $\mathbf{y}$ are disjoint sequences of distinct variables, and each $\varphi_i(\mathbf{x}, \mathbf{y})$ is an atomic formula. Here, an *atomic formula* (or just *atom* for short) has the form $R(t_1, \ldots, t_k)$, where R is a k-ary relation symbol, and each t_i, called *term*, is a variable (from $\mathbf{x}$ or $\mathbf{y}$) or a constant. The right hand side $\varphi_1(\mathbf{x}, \mathbf{y}), \ldots, \varphi_m(\mathbf{x}, \mathbf{y})$ is called the *body* of Q. We make the requirement that each variable of Q occurs at least once in the body. A *homomorphism* from Q to a database D (over the same schema) is a function h that maps every variable of Q to a constant, such that for each atom $R(t_1, \ldots, t_k)$, the tuple $(t_1, \ldots, t_k)$ is in R^D once each variable z is replaced

with $h(z)$. The signature $sig(Q)$ is $\mathbf{x}$ (hence, variables act as attributes). The set $Q(D)$ consists of the restrictions of all homomorphisms to $\mathbf{x}$; that is, if $\mathbf{x} = (x_1, \ldots, x_k)$ then $Q(D)$ is the set of all tuples $(h(x_1), \ldots, h(x_k))$ such that h is a homomorphism from Q to D.

2.2 Probability Spaces

All the probability spaces we refer to are discrete (and even finite). Formally, a *probability space* is a pair (Ω, π), where Ω is a countable set, called the *sample space*, and $\pi : \Omega \to [0, 1]$, called the *probability mass function*, is such that $\sum_{o \in \Omega} \pi(o) = 1$. Each element in the sample space is called a *sample*. Let $\mathcal{P} = (\Omega, \pi)$ be a probability space. As conventional, if $\psi : \Omega \to \{\mathbf{true}, \mathbf{false}\}$ is a Boolean condition over samples, then $\Pr(\psi(\mathcal{P}))$ denotes the sum of $\pi(o)$ over all the samples $o \in \Omega$ with $\psi(o) = \mathbf{true}$. (Note that this is a slight abuse of notation, as in $\psi(\mathcal{P})$ the symbol $\mathcal{P}$ denotes a random element from the sample space Ω and not the probability space.)

2.3 Orders and Rankings

A binary relation $\succ$ over a set $I = \{\sigma_1, \ldots, \sigma_n\}$ of *items* is a (strict) *partial order* if it is irreflexive and transitive; that is for all i, j and k we have $\sigma_i \not\succ \sigma_i$, and $\sigma_i \succ \sigma_j$ and $\sigma_j \succ \sigma_k$ imply $\sigma_i \succ \sigma_k$. A *linear* (or *total*) order is a partial order where every two items are comparable. By a slight abuse of notation, we often identify a linear order $\sigma_1 \succ \cdots \succ \sigma_n$ with the sequence $\langle \sigma_1, \ldots, \sigma_n \rangle$, and we call it a *ranking over* $\{\sigma_1, \ldots, \sigma_n\}$, or just *ranking* if the set of items is clear from the context. We denote by $rnk(I)$ the set of all rankings over I. If $\boldsymbol{\sigma} = \langle \sigma_1, \ldots, \sigma_n \rangle$ is a ranking, then $\text{items}(\boldsymbol{\sigma})$ denotes the set $\{\sigma_1, \ldots, \sigma_n\}$, and $\succ_{\boldsymbol{\sigma}}$ denotes the order that $\boldsymbol{\sigma}$ stands for; that is, $\sigma_i \succ_{\boldsymbol{\sigma}} \sigma_j$ whenever $i < j$. Finally, $\boldsymbol{\sigma}(\tau)$ denotes the position of an item τ in a ranking $\boldsymbol{\sigma}$; that is, if $\boldsymbol{\sigma} = \langle \sigma_1, \ldots, \sigma_n \rangle$ and $\tau = \sigma_i$, then $\boldsymbol{\sigma}(\tau) = i$.

EXAMPLE 2.1. Our running example is on individual preferences among US presidential candidates. Let I be the set $\{\texttt{Clinton}, \texttt{Rubio}, \texttt{Sanders}, \texttt{Trump}\}$. The ranking

$$\boldsymbol{\tau} = \langle \texttt{Clinton}, \texttt{Rubio}, \texttt{Sanders}, \texttt{Trump} \rangle$$

is an example member of $rnk(I)$. We have $\text{items}(\boldsymbol{\tau}) = I$ and $\succ_{\boldsymbol{\tau}}$ is the linear order given by

$$\texttt{Clinton} \succ_{\boldsymbol{\tau}} \texttt{Rubio} \succ_{\boldsymbol{\tau}} \texttt{Sanders} \succ_{\boldsymbol{\tau}} \texttt{Trump}\,.$$

In particular, we have $\texttt{Clinton} \succ_{\boldsymbol{\tau}} \texttt{Trump}$. □

2.4 The Repeated Insertion Model (RIM)

The *Repeated Insertion Model* [12], or *RIM* for short, is a generative process that gives rise to a family of distributions over rankings (linear orders over an itemset). RIM is a generalization of some well known statistical models for rankings such as *Mallows* [27] (defined later in this section), *generalized Mallows* [13], and *multistage ranking* [14].

A RIM model has two parameters, $\boldsymbol{\sigma}$ and Π, and is denoted by $\mathsf{RIM}(\boldsymbol{\sigma}, \Pi)$. The first parameter is a ranking $\boldsymbol{\sigma} = \langle \sigma_1, \ldots, \sigma_m \rangle$ referred to as a *reference ranking*. The model defines a probability distribution over the rankings of the items, or more formally, over the sample space $rnk(\text{items}(\boldsymbol{\sigma}))$. The probability of each ranking is defined by a generative insertion process that we describe below, using the second parameter. This parameter, Π, is called an *insertion probability function* (or just *insertion function* for short) and it maps every pair (i, j) of integers, with $1 \leq i \leq m$ and $1 \leq j \leq i$, into a probability $\Pi(i, j) \in [0, 1]$, so that for all $i = 1, \ldots, m$ we have $\sum_{j=1}^{i} \Pi(i, j) = 1$. (In particular, $\Pi(1, 1) = 1$.)

Semantically, in $\mathsf{RIM}(\boldsymbol{\sigma}, \Pi)$ a ranking in $rnk(\text{items}(\boldsymbol{\sigma}))$ is generated by the following randomized process. Suppose that $\boldsymbol{\sigma} = \langle \sigma_1, \ldots, \sigma_m \rangle$. We begin with the empty ranking, and scan the items $\sigma_1, \ldots, \sigma_m$ in order, starting with σ_1. Each σ_i is inserted into a *random position* $j \in \{1, \ldots, i\}$ inside the current series $\langle \tau_1, \ldots, \tau_{i-1} \rangle$, pushing $\tau_j, \ldots, \tau_{i-1}$ forward, and resulting in the series

$$\langle \tau_1, \ldots, \tau_{j-1}, \sigma_i, \tau_j, \ldots, \tau_{i-1} \rangle\,.$$

An important property of this process is that the insertion position of each σ_i is probabilistically independent of the positions of the previous items σ_1, ..., σ_{i-1}. Moreover, an easy observation is that every insertion sequence gives rise to a unique ranking.

The above process defines a probability for each ranking $\boldsymbol{\tau}$ over $\text{items}(\boldsymbol{\sigma})$, and we denote this probability by $\Pi_{\boldsymbol{\sigma}}(\boldsymbol{\tau})$. The resulting probability space is (Ω, π), where Ω is $rnk(\text{items}(\boldsymbol{\sigma}))$ and π is $\Pi_{\boldsymbol{\sigma}}$. We denote this probability space by $[\![\mathsf{RIM}(\boldsymbol{\sigma}, \Pi)]\!]$.

EXAMPLE 2.2. Continuing with our running example, consider a model $\mathsf{RIM}(\boldsymbol{\sigma}, \Pi)$, where

$$\boldsymbol{\sigma} = \langle \texttt{Clinton}, \texttt{Sanders}, \texttt{Rubio}, \texttt{Trump} \rangle\,.$$

The following is a possible random execution of the generative process, where the result is the ranking $\boldsymbol{\tau}$ of Example 2.1. We begin with $\boldsymbol{\tau}$ being the empty ranking $\langle\rangle$, and insert items (candidates) as follows.

1. $\boldsymbol{\tau} = \langle \texttt{Clinton} \rangle$ (with probability $\Pi(1, 1) = 1$);
2. $\boldsymbol{\tau} = \langle \texttt{Clinton}, \texttt{Sanders} \rangle$ (with probability $\Pi(2, 2)$);
3. $\boldsymbol{\tau} = \langle \texttt{Clinton}, \texttt{Rubio}, \texttt{Sanders} \rangle$ (with probability $\Pi(3, 2)$);
4. $\boldsymbol{\tau} = \langle \texttt{Clinton}, \texttt{Rubio}, \texttt{Sanders}, \texttt{Trump} \rangle$ (with probability $\Pi(4, 4)$).

Therefore, the probability of $\boldsymbol{\tau}$ is given by $\Pi_{\boldsymbol{\sigma}}(\boldsymbol{\tau}) = \Pi(1, 1) \times \Pi(2, 2) \times \Pi(3, 2) \times \Pi(4, 4)$. □

An important and well known special case of RIM is the Mallows model, which we define next.

2.4.1 The Mallows Model

A *Mallows* model [27] is parameterized by a reference ranking $\boldsymbol{\sigma} = \langle \sigma_1, \ldots, \sigma_m \rangle$ and a *dispersion parameter* $\phi \in (0, 1]$, and is denoted by $\mathsf{MAL}(\boldsymbol{\sigma}, \phi)$. The model $\mathsf{MAL}(\boldsymbol{\sigma}, \phi)$ assigns to every ranking $\boldsymbol{\tau} \in rnk(\text{items}(\boldsymbol{\sigma}))$ a non-zero probability defined by

$$\Pr(\boldsymbol{\tau} \mid \boldsymbol{\sigma}, \phi) \stackrel{\text{def}}{=\!=} \frac{1}{Z} \phi^{d(\boldsymbol{\tau}, \boldsymbol{\sigma})}\,.$$

Here, $d(\boldsymbol{\tau}, \boldsymbol{\sigma})$ is *Kendall's tau* [20] distance between $\boldsymbol{\tau}$ and $\boldsymbol{\sigma}$ that counts the disagreements between $\boldsymbol{\tau}$ and $\boldsymbol{\sigma}$, and is formally given by

$$d(\boldsymbol{\tau}, \boldsymbol{\sigma}) \stackrel{\text{def}}{=\!=} \sum_{1 \leq i < j \leq m} \mathbb{1}[\boldsymbol{\tau}(\sigma_j) < \boldsymbol{\tau}(\sigma_i)]\,.$$

Candidates (o)

cand	party	sex	edu
Trump	R	M	BS
Clinton	D	F	JD
Sanders	D	M	BS
Rubio	R	M	JD

Voters (o)

voter	edu	sex	age
Ann	BS	F	25
Bob	BS	M	35
Cat	MS	F	40
Dave	MS	M	45
Eve	BS	F	55
Fred	JD	M	35

Polls (p)

voter	date	lcand	rcand
Ann	Oct-5	Sanders	Clinton
Ann	Oct-5	Sanders	Rubio
Ann	Oct-5	Sanders	Trump
Ann	Oct-5	Clinton	Rubio
Ann	Oct-5	Clinton	Trump
Ann	Oct-5	Rubio	Trump
Bob	Oct-5	Sanders	Rubio
Bob	Oct-5	Sanders	Clinton
Bob	Oct-5	Sanders	Trump
Bob	Oct-5	Rubio	Clinton
Bob	Oct-5	Rubio	Trump
Bob	Oct-5	Clinton	Trump
Dave	Nov-5	Clinton	Rubio
Dave	Nov-5	Clinton	Sanders
Dave	Nov-5	Clinton	Trump
Dave	Nov-5	Rubio	Sanders
Dave	Nov-5	Rubio	Trump
Dave	Nov-5	Sanders	Trump

Figure 1: An example of a preference database

The indicator function $\mathbb{1}$ assigns 1 to true statements and 0 to false ones. The *normalization constant* Z is the sum of $\phi^{d(\boldsymbol{\tau},\boldsymbol{\sigma})}$ over all $\boldsymbol{\tau} \in rnk(\text{items}(\boldsymbol{\sigma}))$.

Intuitively, the higher the distance of a ranking $\boldsymbol{\tau}$ is from the reference ranking $\boldsymbol{\sigma}$, the lower its probability under the Mallows model. Lower values of ϕ concentrate most of the probability mass around $\boldsymbol{\sigma}$, while $\phi = 1$ corresponds to the uniform probability distribution over $rnk(\text{items}(\boldsymbol{\sigma}))$.

Doignon [12] has shown that $\mathsf{MAL}(\boldsymbol{\sigma}, \phi)$ can be represented as the insertion model $\mathsf{RIM}(\boldsymbol{\sigma}, \Pi)$ where

$$\Pi(i,j) \stackrel{\text{def}}{=\!=} \frac{\phi^{i-j}}{1+\phi+\cdots+\phi^{i-1}}\,.$$

The reader can verify that Π is, indeed, an insertion function for RIM, as previously defined.

Example 2.3. Figure 2 depicts three Mallows models, aligned with the rows of the table (that we later discuss). The model in the top row, for instance, is $\mathsf{MAL}(\boldsymbol{\sigma}, \phi)$ where $\boldsymbol{\sigma} = \langle$`Clinton`, `Sanders`,`Rubio`,`Trump`$\rangle$ and $\phi = 0.3$. □

3. PROBABILISTIC PREFERENCE DATABASES

In this section we define the central concept of our framework—the *probabilistic preference database*. The definition builds on the concept of a "preference relation" that we introduced in a past vision paper [18]. We will illustrate the concepts of this section with our running example of a probabilistic preference database describing polling in a presidential election.

3.1 Preference Databases

Intuitively, a preference relation stores a collection of binary relations (called *preferences*) among a set of items, where each such binary relation is called a *session*. A preference relation instantiates a special relation symbol with a signature that distinguishes between three kinds of attributes: one corresponds to the session identifier, one to the left item, and one to the right item.

Formally, a *preference signature* is a relation signature of the form β, A_l, A_r (i.e., the concatenation of β, A_l and A_r from left to right), where β is a relation signature that we call the *session signature*, and A_l and A_r are attributes that we call the *left-hand-side* (*lhs*) attribute and *right-hand-side* (*rhs*) attribute, respectively. For presentation sake, we use semicolon (;) to distinguish between the different parts and write $(\beta; A_l; A_r)$.

Example 3.1. In our running example we use the preference signature (voter, date; lcand; rcand). Here the components β, A_l and A_r are (voter, date), lcand, and rcand, respectively. □

Let $(\beta; A_l; A_r)$ be a preference signature, and let r be an instance over $(\beta; A_l; A_r)$. Every value that occurs in either the lhs attribute or rhs attribute of r is called an *item*, and we denote by items(r) the set of items of r. Hence, items(r) can be identified with $\pi_{A_l}(r) \cup \pi_{A_r}(r)$. A *session* of an instance r is the β part of a tuple; that is, a tuple in the projection $\pi_\beta(r)$. We denote $\pi_\beta(r)$ by sessions(r). Observe that β can be empty, and then an instance r can store at most one session. Semantically, r maps every $\mathbf{s} \in$ sessions(r) into an order $\succ$ over items(r), where $\sigma \succ \tau$ if and only if r contains the tuple $(\mathbf{s}; \sigma; \tau)$.

Example 3.2. The table **Polls** of Figure 1 is an instance of the preference signature of Example 3.1. This instance contains three sessions:

- The session (`Ann`, `Oct-5`) is associated with the ranking ⟨`Sanders`, `Clinton`, `Rubio`, `Trump`⟩;
- The session (`Bob`, `Oct 5`) is associated with the ranking ⟨`Sanders`, `Rubio`, `Clinton`, `Trump`⟩;
- The session (`Dave`, `Nov 5`) is associated with the ranking ⟨`Clinton`, `Rubio`, `Sanders`, `Trump`⟩.

Polls represents each ranking explicitly, as an order, by listing all pairwise preferences. This is a conceptual representation and, in effect, when the orders are rankings (linear orders) a compact physical representation can be used, as we do later on in this paper. □

A *preference schema* $\mathbf{S}$ is a relational schema with two types of relation symbols: *ordinary* relation symbols R that are associated with ordinary signatures $sig(R)$, and *preference relation symbols* P that are associated with preference signatures $sig(P)$. For clarity and brevity, we refer to an ordinary and a preference relation symbol as an *o-symbol* and

Polls^E

voter	date	*Preference model* $\mathbf{MAL}(\boldsymbol{\sigma}, \phi)$
Ann	Oct-5	⟨`Clinton`, `Sanders`, `Rubio`, `Trump`⟩, 0.3
Bob	Oct-5	⟨`Trump`, `Rubio`, `Sanders`, `Clinton`⟩, 0.3
Dave	Nov-5	⟨`Clinton`, `Sanders`, `Rubio`, `Trump`⟩, 0.5

Figure 2: A **MAL**-instance over the p-symbol **Polls**

a *p-symbol*, respectively. A database D over a preference schema, called a *preference database*, associates with each o-symbol R an instance over $sig(R)$, and with each p-symbol P an instance over $sig(P)$; we call the former an *o-instance* and the latter a *p-instance*.

EXAMPLE 3.3. Our running example uses a preference schema $\mathbf{S}$ with three relation symbols:

- o-symbol **Candidates**(candidate, party, sex, edu) for information about political candidates.
- o-symbol **Voters**(voter, edu, sex, age) for information on individual voters.
- p-symbol **Polls**(voter, date; lcand; rcand) for polls of voters on specific dates.

Figure 1 depicts a preference database over $\mathbf{S}$. We write "(o)" and "(p)" next to each relation symbol to stress that it is an o-symbol or a p-symbol, respectively. □

In our running example so far, we assumed that we had exact information about voters' opinions on the candidates on specific dates. In the next section, we will make this knowledge probabilistic, interpreting the preference database of Figure 1 as one *possible world*.

3.2 Probabilistic Preference Databases

Let $\mathbf{S}$ be a preference schema. A *Probabilistic Preference Database* (abbrv. *PPD*) over $\mathbf{S}$ is a probability space over preference databases over $\mathbf{S}$. A PPD can be represented by explicitly specifying the entire sample space; however, we wish to allow for standard compact representations of preferences such as RIM and Mallows. In the remainder of this section we discuss a specific representation of this sort.

A *probabilistic preference model* is a (finite and typically compact) representation M of a probability space over partial orders $\succ$ over a finite set of items; we denote this probability space by $[\![M]\!]$. A *probabilistic preference model family*, or just *model family* for short, is a collection $\mathcal{M}$ of probabilistic preference models. As prominent examples, we define two model families.

- **RIM** is the family of RIM models $\mathsf{RIM}(\boldsymbol{\sigma}, \Pi)$.
- **MAL** is the family of Mallows models $\mathsf{MAL}(\boldsymbol{\sigma}, \phi)$.

Let $\mathbf{S}$ be a preference schema, and let $\mathcal{M}$ be a model family. Let P be a p-symbol of $\mathbf{S}$ and suppose that $sig(P) = (\beta; A_l; A_r)$. An *$\mathcal{M}$-instance* over P is a pair (r, μ), where r is an instance over β, and μ maps every tuple $\mathbf{s} \in r$ into a probabilistic preference model $M \in \mathcal{M}$. Each tuple in r is, as usual, referred to as a *session*.

EXAMPLE 3.4. Let $\mathbf{S}$ be the preference schema of Figure 1, as described in Example 3.3. Figure 2 depicts a **MAL**-instance (r, μ) over the p-symbol **Polls**. Note that r is an instance of the signature $\beta = (\mathsf{voter}, \mathsf{date})$, and in the figure it is the table on the left. The model $\mu(\mathbf{s})$ for each $\mathbf{s} \in r$ is shown to the right of $\mathbf{s}$. For example, the model $\mu(\texttt{Ann}, \texttt{Oct-5})$ is the one we discussed previously in Example 2.3. □

A *session-independent $\mathcal{M}$-PPD*, or just *$\mathcal{M}$-PPD* for short, over a preference schema $\mathbf{S}$ is a mapping that associates with each o-symbol R an instance over $sig(R)$, and with each p-symbol P an $\mathcal{M}$-instance over P; we denote these instances by R^E and P^E, respectively. An $\mathcal{M}$-PPD E over $\mathbf{S}$ defines a PPD over $\mathbf{S}$ by viewing each session as an independent probabilistic preference model. We denote this PPD by $[\![E]\!]$. More precisely, a sample (or *possible world*) D of $[\![E]\!]$ is constructed by the following generative process.

for all o-symbols R of $\mathbf{S}$ **do**
  $R^D := R^E$
for all p-symbol P of $\mathbf{S}$ **do**
  $P^D :=$ an empty p-instance over P
  let (r, μ) be the $\mathcal{M}$-instance P^E
  for all sessions $\mathbf{s} \in r$ **do**
    sample a partial order $\succ$ from $[\![\mu(\mathbf{s})]\!]$
    $P^D := P^D \cup \{(\mathbf{s}; \sigma; \tau) \mid \sigma \succ \tau\}$

EXAMPLE 3.5. Let D be the preference database of Figure 1. We define the **MAL**-PPD E as the one that consists of the following:

- The o-instances $\mathbf{Candidates}^D$ and $\mathbf{Voters}^D$ of Figure 1, now referred to as $\mathbf{Candidates}^E$ and $\mathbf{Voters}^E$, respectively.
- The **MAL**-instance $\mathbf{Polls}^E$ of Figure 2, and we denote it as (r, μ).

Then D is a possible world of $[\![E]\!]$. Let $\mathbf{s}_1$, $\mathbf{s}_2$ and $\mathbf{s}_3$ be the three sessions of r, top down ($\mathbf{s}_1 = (\texttt{Ann}, \texttt{Oct-5})$, and so on). The probability of D in $[\![E]\!]$ is the product $p_1 \times p_2 \times p_3$, where p_1 is the probability of $\succ_{\mathbf{s}_1}$ in the model $\mu(\mathbf{s}_1)$, which is $\mathsf{MAL}(\boldsymbol{\sigma}, 0.3)$ for $\boldsymbol{\sigma} = \langle \texttt{Clinton}, \texttt{Sanders}, \texttt{Rubio}, \texttt{Trump} \rangle$, and p_2 and p_3 defined analogously for the sessions $\mathbf{s}_2$ and $\mathbf{s}_3$, respectively. □

3.3 Querying PPDs

We adopt the semantics of probabilistic databases [35] for query evaluation. Specifically, let $\mathbf{S}$ be a schema, let Q be a query, and let $\mathcal{D} = (\Omega, \pi)$ be a PPD. A *possible answer* for Q is a tuple $\mathbf{a}$ over $sig(Q)$ such that $\mathbf{a} \in Q(D)$ for some sample D of $\mathcal{D}$. We denote by $PosAns(Q, \mathcal{D})$ the set of all possible answers. Note that $PosAns(Q, \mathcal{D})$ is finite, since $\mathcal{D}$ is a finite probability space. The *confidence* of a possible answer $\mathbf{a} \in PosAns(Q, \mathcal{D})$, denoted $conf_Q(\mathcal{D}, \mathbf{a})$, is the probability of having $\mathbf{a}$ as an answer when querying a sample of $\mathcal{D}$:

$$conf_Q(\mathcal{D}, \mathbf{a}) \stackrel{\text{def}}{=\!=} \Pr(\mathbf{a} \in Q(\mathcal{D}))$$

As a special case, if Q is Boolean then $conf_Q(\mathcal{D})$ denotes the probability of true; that is,

$$conf_Q(\mathcal{D}) \stackrel{\text{def}}{=\!=} \Pr(Q(\mathcal{D}) = \mathbf{true}) .$$

In particular, if E is an $\mathcal{M}$-PPD for some model class $\mathcal{M}$, then *evaluating Q on E* is the task of computing the following (finite) set.

$$Q(E) \stackrel{\text{def}}{=\!=} \{(\mathbf{a}, conf_Q([\![E]\!], \mathbf{a})) \mid \mathbf{a} \in PosAns([\![E]\!])\}$$

When Q is Boolean, the evaluation of Q boils down to computing the number $conf_Q([\![E]\!])$.

EXAMPLE 3.6. We now present several Boolean CQs for our running example. We will refer to these queries in later sections of the paper. For clarity, we refer to every relational

symbol by its first letter (e.g., C for **Candidates**). We also follow the convention of using underscore (_) instead of variables that occur only once in the query; however, we index the underscores by subscripts for later reference. Recall the preference schema in Figure 1: $C(\mathsf{cand}, \mathsf{party}, \mathsf{sex}, \mathsf{edu})$, $V(\mathsf{voter}, \mathsf{edu}, \mathsf{sex}, \mathsf{age})$ and $P(\mathsf{voter}, \mathsf{date}, \mathsf{lcand}, \mathsf{rcand})$.

The query Q_1 asks whether there is a voter with a BS degree who prefers a male Democratic candidate to a female Democratic candidate.

$$Q_1() \leftarrow P(v, \__1; l; r), V(v, \mathtt{BS}, \__2, \__3), C(l, \mathtt{D}, \mathtt{M}, \__4), C(r, \mathtt{D}, \mathtt{F}, \__5)$$

The query Q_2 asks whether there is a voter who prefers a male candidate to a female candidate such that both candidates are of the same political party.

$$Q_2() \leftarrow P(\__1, \__2; l; r), C(l, p, \mathtt{M}, \__3), C(r, p, \mathtt{F}, \__4)$$

The query Q_3 asks whether there is a voter who prefers a female candidate to both Trump and Sanders.

$$Q_3() \leftarrow P(v, d; l; \mathtt{Trump}), P(v, d; l; \mathtt{Sanders}), C(l, \__1, \mathtt{F}, \__2)$$

Finally, the query Q_4 asks whether there is a voter who prefers a candidate of the same gender to a candidate of the same education.

$$Q_4() \leftarrow P(v, \__1; l; r), V(v, \__2, s, \__3), V(v, e, \__4, \__5) C(l, \__6, s, \__7), C(r, \__8, \__9, e)$$

Note that we can combine the atoms $V(v, \__2, s, \__3)$ and $V(v, e, \__4, \__5)$ into a single atom if we make the (natural) assumption that v is a key. We choose to use this phrasing to illustrate definitions from the next section.

When evaluating the above Boolean queries over a PPD, such as the **MAL**-PPD of Example 3.5, the goal is to compute the probability that the query evaluates to **true**. For example, by posing Q_1 we wish to compute the probability that at least one voter with a BS degree prefers a male Democratic candidate to a female Democratic candidate. □

4. CQ EVALUATION ON RIM-PPDS

In this section we establish preliminary complexity results on querying **RIM**-PPDs with CQs. All our results are on *data complexity*, where the schema and query are considered fixed. Recall that query evaluation entails computing the probability of every possible tuple in the result. Given that, it suffices to consider only *Boolean CQs* (for both lower and upper bounds), as the complexity of a general CQ can be determined by considering the Boolean CQ that is obtained by replacing the head variables with constant values.

4.1 Tractability of Itemwise CQs

In this section we define the notion of an *itemwise* CQ over a preference schema, and give a positive complexity result on their evaluation over **RIM**-PPDs. To define an itemwise CQ, we need some terminology.

Let **S** be a preference schema, and let Q be a CQ over **S**. An atomic formula of Q is called a *p-atom* if it is over a p-symbol, and an *o-atom* if it is over an o-symbol. The *Gaifman graph* (or *primal graph*) of Q, denoted $\mathcal{G}_Q$, is the graph that has the variables of Q as the nodes, and an edge between two variables x and y whenever x and y are different variables that occur in the same atomic formula of Q. The *Gaifman o-graph* of Q, denoted $\mathcal{G}^\circ_Q$, is defined similarly to $\mathcal{G}_Q$, except that edges due to p-atoms are avoided—there is an edge between x and y whenever x and y are different variables that occur in the same o-atom of Q.

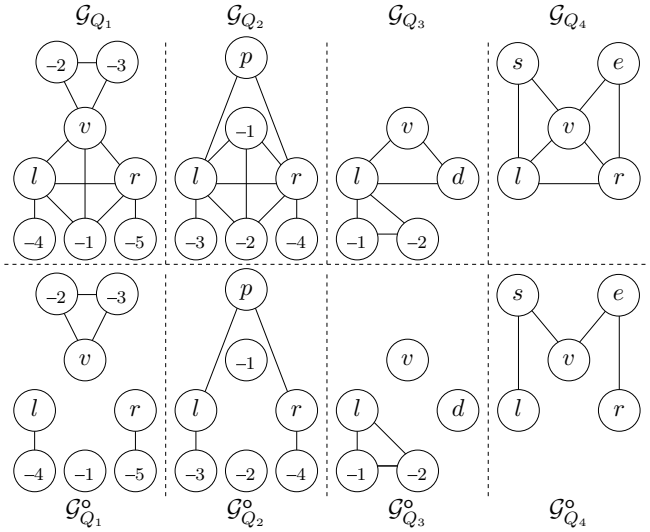

Figure 3: Gaifman graphs (top) and Gaifman o-graphs of the CQs of Example 3.6

Example 4.1. Figure 3 depicts the Gaifman graphs $\mathcal{G}_Q$ and the Gaifman o-graphs $\mathcal{G}^\circ_Q$ for the CQs Q of Example 3.6. To see the definition of $\mathcal{G}^\circ_Q$ in action, consider the leftmost Gaifman o-graph, $\mathcal{G}^\circ_{Q_1}$. The variable v is connected by an edge to the variable l in $\mathcal{G}^\circ_{Q_1}$, but not in $\mathcal{G}^\circ_{Q_1}$, since the only atom that contains both v and l is the p-atom $P(v, \__1; l; r)$. To simplify the figure, in $\mathcal{G}^\circ_{Q_4}$ and $\mathcal{G}^\circ_{Q_4}$ we do not include the underscore variables, since they do not impact the definitions that follow. □

Let $P(s_1, \ldots, s_k; t_l; t_r)$ be p-atom of Q. Each term s_i for $i = 1, \ldots, k$ is said to occur in a *session position*, and each of t_l and t_r is said to occur in an *item position*. A *session variable* of Q is a variable that occurs in a session position, and an *item variable* of Q is a variable that occurs in an item position. We say that Q is *sessionwise* if all p-atoms of Q refer to the same session; that is, if $P(s_1, \ldots, s_k; t_l; t_r)$ and $P'(s'_1, \ldots, s'_l; t'_l; t'_r)$ are p-atoms of Q, then $P = P'$ and $(s_1, \ldots, s_k) = (s'_1, \ldots, s'_l)$.

Example 4.2. All of the CQs of Example 3.6 are sessionwise. Indeed, queries Q_1, Q_2 and Q_4 involve a single p-atom (hence, they are sessionwise by definition), and in Q_3 both atoms have (v, d) in their session parts. □

Finally, let g be an undirected graph, and let U and W be sets of nodes of g. We say that U *completely separates* W if every path between two different nodes of W visits at least one node of U. We can now define an itemwise CQ.

Definition 1 (Itemwise CQ). *A CQ Q over a preference schema is* itemwise *if Q is sessionwise and the set of session variables completely separates the set of item variables in $\mathcal{G}^\circ_Q$.*

Intuitively, in an itemwise CQ the joins between item variables occur only inside the p-atoms, or through session variables. Put differently, in an itemwise CQ with a constant session, the o-atoms state properties of individual items but do not draw connections between the items.

EXAMPLE 4.3. We consider again the CQs of Example 3.6, and their Gaifman o-graphs in Figure 3. Recall from Example 4.2 that the CQs are all sessionwise. The CQ Q_1 is itemwise since there is no path between l and r in $\mathcal{G}^{\circ}_{Q_1}$. The CQ Q_3 is itemwise since there is a single item variable (namely l). The CQ Q_4 is itemwise since there is a single path between l and r in $\mathcal{G}^{\circ}_{Q_4}$, and that path visits the session variable v. Finally, Q_2 is *not* itemwise since the path l—p—r does not visit any session variable. □

We later prove the following theorem, which states that every itemwise Boolean CQ can be evaluated in polynomial time (under data complexity).

THEOREM 4.4. *Let* $\mathbf{S}$ *be a preference schema, and let* Q *be a Boolean CQ over* $\mathbf{S}$*. If* Q *is itemwise, then* Q *can be evaluated in polynomial time on* **RIM***-PPDs over* $\mathbf{S}$*.*

We discuss the evaluation algorithm in Section 4.4.

4.2 Hardness

The following theorem shows a restricted (yet natural) class of Boolean CQs where itemwise CQs are *precisely* the tractable ones (among the queries in the class), under conventional complexity assumptions. In other words, every Boolean CQ (in the class) that is not itemwise is necessarily hard to evaluate, and therefore, we establish a dichotomy.

THEOREM 4.5. *Let* $\mathbf{S}$ *be a preference schema, and let* Q *be a Boolean CQ over* $\mathbf{S}$ *such that* Q *has no self joins and* Q *has a single p-atom. If* Q *is not itemwise, then the evaluation of* Q *on* **RIM***-PPDs over* $\mathbf{S}$ *is* $\mathrm{FP}^{\#\mathrm{P}}$*-hard.*

Recall that a *self join* of a CQ is a pair of distinct atoms with the same relation symbol. Also recall that $\mathrm{FP}^{\#\mathrm{P}}$ is the class of functions that are efficiently computable using an oracle to some function in #P. A function f is $\mathrm{FP}^{\#\mathrm{P}}$-hard if there is a polynomial-time *Turing reduction*[1] (or *Cook reduction*) from every function in $\mathrm{FP}^{\#\mathrm{P}}$ to f.

The proof of Theorem 4.5 is in the appendix. The proof technique is similar to that of Dalvi and Suciu [9]. We first prove hardness of the simplest intractable CQ, which is the following.

$$Q_{\mathtt{h}}() \leftarrow R(x,y), P(x;y)$$

Here, R is an o-symbol and P is a p-symbol.

LEMMA 4.6. *The evaluation of* $Q_{\mathtt{h}}$ *on a given* **RIM***-PPD is* $\mathrm{FP}^{\#\mathrm{P}}$*-hard.*

We then reduce the evaluation of $Q_{\mathtt{h}}$ to the evaluation of every other CQ in the intractability side. We note in the proof of Lemma 4.6 we explain that the lower bound holds even for **MAL**-PPDs (using the same proof). Hence, Theorem 4.5 applies to **MAL**-PPDs, and so is the dichotomy in complexity.

[1]Using an oracle to a #P-hard (or $\mathrm{FP}^{\#\mathrm{P}}$-hard) function, one can solve in polynomial time every problem in the polynomial hierarchy [36].

4.3 Pattern Matching in Labeled Models

Our algorithm for evaluating itemwise CQs (Theorem 4.4) is a reduction to a certain type of inference over a generalized RIM model. In this section we introduce this generalized model and the type of queries we support, and in the next section we describe the reduction. In Section 5 we present an efficient algorithm for inference.

We assume an infinite set $\mathbf{\Lambda}$ of *item labels*. A *labeled RIM model* extends RIM by associating with each item a finite set of labels. More formally, labeled RIM is parameterized by $\boldsymbol{\sigma}$, Π and λ, where $\mathsf{RIM}(\boldsymbol{\sigma}, \Pi)$ is a RIM model and λ is a labeling function that maps every item σ_i in $\boldsymbol{\sigma}$ to a finite set $\lambda(\sigma_i)$ of labels in $\mathbf{\Lambda}$. We denote this model by $\mathsf{RIM}_{\mathsf{L}}(\boldsymbol{\sigma}, \Pi, \lambda)$. A *label pattern* (or just *pattern* for short) is a directed graph g where every node of g is a label in $\mathbf{\Lambda}$. We denote by $\mathrm{nodes}(g)$ and $\mathrm{edges}(g)$ the sets of nodes and edges of g, respectively.

Let $\boldsymbol{\tau} = \langle \tau_1, \ldots, \tau_m \rangle$ be a random ranking of a model $\mathsf{RIM}_{\mathsf{L}}(\boldsymbol{\sigma}, \Pi, \lambda)$, and let g be a pattern. Recall that $\succ_{\boldsymbol{\tau}}$ is the linear order that corresponds to $\boldsymbol{\tau}$; that is, $\tau_i \succ_{\boldsymbol{\tau}} \tau_j$ if and only if $i < j$. A *matching* of g in $\boldsymbol{\tau}$ (w.r.t. λ) is a function $\gamma : \mathrm{nodes}(g) \to \mathrm{items}(\boldsymbol{\sigma})$ that satisfies the following conditions.

1. Labels match: $l \in \lambda(\gamma(l))$ for all nodes l of g;
2. Edges match: for all edges $l \to l'$ of g it holds that $\gamma(l) \succ_{\boldsymbol{\tau}} \gamma(l')$.

The ranking $\boldsymbol{\tau}$ *matches* g (w.r.t. λ), denoted $(\boldsymbol{\tau}, \lambda) \models g$, if there is at least one matching of g in $\boldsymbol{\tau}$. We denote by $\Gamma(g, \boldsymbol{\tau})$ the set of all the matchings of g in $\boldsymbol{\tau}$.

EXAMPLE 4.7. Consider the model $\mathsf{RIM}_{\mathsf{L}}(\boldsymbol{\sigma}, \Pi, \lambda)$ for

$$\boldsymbol{\sigma} \stackrel{\text{def}}{=\!=} \langle \mathtt{Sanders}, \mathtt{Clinton}, \mathtt{Rubio}, \mathtt{Trump}, \mathtt{Stein} \rangle .$$

The labeling λ is given alongside the pattern g of Figure 4a: next to each label v the candidates with the corresponding label are mentioned (e.g., $\lambda(\mathtt{Trump}) = \{l_{\mathtt{R}}, l_{\mathtt{BS}}\}$). The label $l_{\mathtt{R}}$ corresponds to "Republican," $l_{\mathtt{F}}$ to "Female," and $l_{\mathtt{BS}}$ to the "BS" academic degree. Consider the ranking $\boldsymbol{\tau}$ given by

$$\boldsymbol{\tau} \stackrel{\text{def}}{=\!=} \langle \mathtt{Rubio}, \mathtt{Clinton}, \mathtt{Sanders}, \mathtt{Trump}, \mathtt{Stein} \rangle .$$

Then, $\Gamma(g, \boldsymbol{\tau})$ contains the following matchings:

- $\gamma_1 \stackrel{\text{def}}{=\!=} \{l_{\mathtt{R}} \mapsto \mathtt{Rubio}, l_{\mathtt{BS}} \mapsto \mathtt{Sanders}, l_{\mathtt{F}} \mapsto \mathtt{Stein}\}$
- $\gamma_2 \stackrel{\text{def}}{=\!=} \{l_{\mathtt{R}} \mapsto \mathtt{Trump}, l_{\mathtt{BS}} \mapsto \mathtt{Sanders}, l_{\mathtt{F}} \mapsto \mathtt{Stein}\}$
- $\gamma_3 \stackrel{\text{def}}{=\!=} \{l_{\mathtt{R}} \mapsto \mathtt{Trump}, l_{\mathtt{BS}} \mapsto \mathtt{Trump}, l_{\mathtt{F}} \mapsto \mathtt{Stein}\}$

Note that in γ_3 two labels are mapped to Trump. □

The problem of *pattern matching* on labeled RIM models is the following. Given a model $\mathsf{RIM}_{\mathsf{L}}(\boldsymbol{\sigma}, \Pi, \lambda)$ and a pattern g, compute the probability that a random ranking matches g; more formally, the goal is to compute $\Pr(g \mid \boldsymbol{\sigma}, \Pi, \lambda)$ that we define as follows.

$$\Pr(g \mid \boldsymbol{\sigma}, \Pi, \lambda) \stackrel{\text{def}}{=\!=} \sum_{\substack{\boldsymbol{\tau} \in rnk(\mathrm{items}(\boldsymbol{\sigma})) \\ (\boldsymbol{\tau}, \lambda) \models g}} \Pi_{\boldsymbol{\sigma}}(\boldsymbol{\tau}) \qquad (1)$$

Note that this expression does not lend itself immediately to an efficient computation, since the number of possible

rankings $\boldsymbol{\tau}$ to consider can be factorial in m. In Section 5 we will present an algorithm for computing this probability, where the time is polynomial under data complexity (i.e., the pattern g is assumed fixed).

4.4 From CQs to Label Patterns

We now describe the reduction from evaluating an itemwise CQ on a **RIM**-PPD to computing the probability of pattern matching over labeled RIM. Throughout this section we fix a preference schema $\mathbf{S}$, an itemwise CQ Q over $\mathbf{S}$, and a **RIM**-PPD E.

Since Q is itemwise, it uses a single p-symbol (possibly multiple times), and we denote this p-symbol by P. We assume and that each p-atom has the session terms $t_1, \ldots, t_k$. Denote the **RIM**-instance P^E by (r, μ). Hence, μ maps every session $\mathbf{s} \in r$ to a RIM model, and we denote it by $\mathsf{RIM}(\boldsymbol{\sigma}^{\mathbf{s}}, \Pi^{\mathbf{s}})$. Recall that our goal is to compute the probability $\Pr([\![E]\!] \models Q)$.

Let r_Q be the set of tuples in r that can be obtained from $t_1, \ldots, t_k$ by assigning values to variables. For $\mathbf{s} = (a_1, \ldots, a_k) \in r_Q$, let $Q^{\mathbf{s}}$ be the CQ that is obtained from Q by replacing the terms $t_1, \ldots, t_k$ with the constants $a_1, \ldots, a_k$, respectively. The following proposition follows from the assumption that Q is sessionwise. The proof of the proposition is straightforward.

LEMMA 4.8. *The following hold.*

1. $\Pr([\![E]\!] \models Q) = \Pr\big(\bigvee_{\mathbf{s} \in r_Q} ([\![E]\!] \models Q^{\mathbf{s}})\big)$.
2. *No two item variables of $Q^{\mathbf{s}}$ belong to the same connected component of $\mathcal{G}^{\circ}_{Q^{\mathbf{s}}}$.*

Lemma 4.8 has several important consequences. From Part 1 we have the following due to probabilistic independence of the sessions.

$$\Pr([\![E]\!] \models Q) = 1 - \Pr\big(\wedge_{\mathbf{s} \in r_Q} ([\![E]\!] \not\models Q^{\mathbf{s}})\big) =$$
$$1 - \prod_{\mathbf{s} \in r_Q} \Pr([\![E]\!] \not\models Q^{\mathbf{s}}) = 1 - \prod_{\mathbf{s} \in r_Q} \big(1 - \Pr([\![E]\!] \models Q^{\mathbf{s}})\big)$$

Therefore, to compute $\Pr([\![E]\!] \models Q)$ it suffices to compute the probabilities $\Pr([\![E]\!] \models Q^{\mathbf{s}})$. Now, all p-atoms of Q use the same session terms, so to compute $\Pr([\![E]\!] \models Q^{\mathbf{s}})$ we can assume that P^E contains a single RIM model, namely $\mathsf{RIM}(\boldsymbol{\sigma}^{\mathbf{s}}, \Pi^{\mathbf{s}})$, since any other RIM model does not have its session matched in $Q^{\mathbf{s}}$.

We now consider Part 2 of Lemma 4.8. We split $Q^{\mathbf{s}}$ into several CQs. Denote by $Q^{\mathbf{s}}_p$ the CQ that consists of (all and only) the p-atoms of $Q^{\mathbf{s}}$, and by $Q^{\mathbf{s}}_o$ the CQ that consists of the remaining atoms (i.e., the o-atoms of $Q^{\mathbf{s}}$). Consider the graph G that contains the atoms of $Q^{\mathbf{s}}_o$ as nodes, and edges between every two atoms that share one or more variables. Let $C_1, \ldots, C_k$ be the connected components of G. For $i = 1, \ldots, k$ we denote by $Q^{\mathbf{s}}_i$ the CQ that consists of the atoms in C_i. Part 2 of Lemma 4.8 implies that each $Q^{\mathbf{s}}_i$ contains at most one item variable. Let x be an item variable of some $Q^{\mathbf{s}}_i$, and let $\sigma \in \text{items}(\boldsymbol{\sigma}^{\mathbf{s}})$. We say that σ is a *potential match* for x if at least one homomorphism from $Q^{\mathbf{s}}_i$ to E maps x to σ. In addition, σ is a *potential match* for an item variable y if y occurs only in p-atoms of $Q^{\mathbf{s}}$.

From the above development we get the following. A random ranking $\boldsymbol{\tau}$ of $\mathsf{RIM}(\boldsymbol{\sigma}^{\mathbf{s}}, \Pi^{\mathbf{s}})$ corresponds to a database D over $\mathbf{S}$ with $D \models Q^{\mathbf{s}}$ if and only if both of the following conditions hold.

1. There is a homomorphism from $Q^{\mathbf{s}}_i$ to D for all $i = 1, \ldots, k$.
2. There is a homomorphism μ from $Q^{\mathbf{s}}_p$ to P^D such that for all item variables x, the item $\mu(x)$ is a potential match for x.

So, we first check whether the first condition holds. If not, then $\Pr([\![E]\!] \models Q^{\mathbf{s}})$ is zero. Otherwise, we construct a labeling function λ over $\boldsymbol{\sigma}^{\mathbf{s}}$. The labels in the range of λ are the terms in the item positions of $Q^{\mathbf{s}}$. For each item variable x, we include the label x in $\lambda(\sigma_i)$ if σ_i is a potential match for x. And for every constant τ, we include the label τ in $\lambda(\sigma_i)$ if $\sigma_i = \tau$. Next, we construct the label pattern g, where (as expected) the nodes of g are the terms in the item positions of $Q^{\mathbf{s}}$. The pattern g contains an edge $t_1 \to t_2$ for every p-atom of $Q^{\mathbf{s}}$ of the form $(t'_1, \ldots, t'_k; t_1; t_2)$, having t_1 and t_2 on the lhs and rhs attributes, respectively. It follows from Condition 2 that $\Pr([\![E]\!] \models Q^{\mathbf{s}})$ is equal to the probability that $\mathsf{RIM}_{\mathsf{L}}(\boldsymbol{\sigma}^{\mathbf{s}}, \Pi^{\mathbf{s}}, \lambda)$ matches g.

EXAMPLE 4.9. Let Q be the query Q_3 of Example 3.6. We consider the model $\mathsf{RIM}(\boldsymbol{\sigma}^{\mathbf{s}}, \Pi^{\mathbf{s}})$ defined from the model $\mathsf{MAL}(\boldsymbol{\sigma}^{\mathbf{s}}, 0.3)$ for the session $\mathbf{s} = (\texttt{Ann}, \texttt{Oct-5})$ in the **MAL**-instance of Figure 2. The query $Q^{\mathbf{s}}$ is the following.

$$Q^{\mathbf{s}}_3() \leftarrow P(\texttt{Ann}, \texttt{Oct-5}; l; \texttt{Trump}),$$
$$P(\texttt{Ann}, \texttt{Oct-5}; l; \texttt{Sanders}), C(l, \_1, \texttt{F}, \_2)$$

The terms in the item positions are l, `Trump` and `Sanders`. Moreover, only `Clinton` is a potential match for l, since l stands for a female candidate. Hence, we translate the evaluation of $Q^{\mathbf{s}}$ into $\mathsf{RIM}_{\mathsf{L}}(\boldsymbol{\sigma}^{\mathbf{s}}, \Pi^{\mathbf{s}}, \lambda)$, where λ is defined as follows.

$$\lambda(\sigma) \stackrel{\text{def}}{=\!=} \begin{cases} \{\texttt{Trump}\} & \text{if } \sigma = \texttt{Trump}; \\ \{\texttt{Sanders}\} & \text{if } \sigma = \texttt{Sanders}; \\ \{\texttt{l}\} & \text{if } \sigma = \texttt{Clinton}; \\ \emptyset & \text{otherwise.} \end{cases}$$

Finally, the label pattern g is the one of Figure 4b, which asks for the existence of an l-item before both `Trump` and `Sanders`.

Now let Q be the query Q_4 of Example 3.6, and let $\mathbf{s}$ be the above session. The query $Q^{\mathbf{s}}$ is the following.

$$Q^{\mathbf{s}}_4() \leftarrow P(\texttt{Ann}, \texttt{Oct-5}; l; r), V(\texttt{Ann}, \_2, s, \_3),$$
$$V(\texttt{Ann}, e, \_4, \_5), C(l, \_6, s, \_7), C(r, \_8, \_9, e)$$

There are two terms in item positions: l and r. The potential matches of l are those that match the subquery

$$V(\texttt{Ann}, \_2, s, \_3), C(l, \_6, s, \_7)$$

(i.e., the same gender as `Ann`), and there is only one such candidate, namely `Clinton`. Similarly, the potential matches of r are those that match the subquery

$$V(\texttt{Ann}, e, \_4, \_5), C(r, \_8, \_9, e)$$

(i.e., the same education as `Ann`), namely `Trump` and `Sanders`. We construct $\mathsf{RIM}_{\mathsf{L}}(\boldsymbol{\sigma}^{\mathbf{s}}, \Pi^{\mathbf{s}}, \lambda)$, where λ is

$$\lambda(\sigma) \stackrel{\text{def}}{=\!=} \begin{cases} \{l\} & \text{if } \sigma = \texttt{Clinton}; \\ \{r\} & \text{if } \sigma \in \{\texttt{Trump}, \texttt{Sanders}\}; \\ \emptyset & \text{otherwise.} \end{cases}$$

Finally, g is simply the graph given by $l \to r$. $\square$

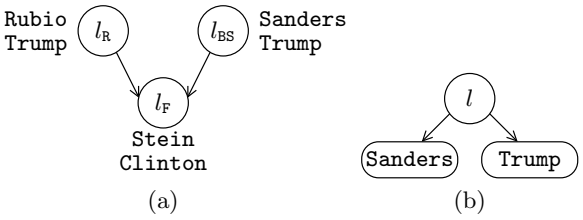

Figure 4: Examples of label patterns

Observe that the size of the pattern g we generated is determined by Q, and is independent of the **RIM**-PPD E. Hence, we establish Theorem 4.4 if we can efficiently compute the probability that a random sample of $\mathsf{RIM}_{\mathsf{L}}(\boldsymbol{\sigma}, \Pi, \lambda)$ matches a pattern g of a fixed size. We devise an algorithm for this task in the next section.

5. INFERENCE ON LABELED RIM

In this section we fix a model $\mathsf{RIM}_{\mathsf{L}}(\boldsymbol{\sigma}, \Pi, \lambda)$ and a pattern g, where $\boldsymbol{\sigma} = \langle \sigma_1, \ldots, \sigma_m \rangle$. We devise an algorithm for computing $\Pr(g \mid \boldsymbol{\sigma}, \Pi, \lambda)$, as defined in (1).

5.1 Top Matchings

Naturally, the challenge in computing $\Pr(g \mid \boldsymbol{\sigma}, \Pi, \lambda)$ is that the number of rankings in our space (and in particular those that match g) can be factorial in m. We are able to overcome this challenge by *partitioning* the space of all rankings $\boldsymbol{\tau} \in rnk(\boldsymbol{\sigma})$ into (pairwise-disjoint) subspaces, using the notion of a *top match* that we define next. The number of partitions is polynomial in m (assuming g is fixed), and hence, it will suffice to compute the probability of each partition. We then devise an algorithm for computing this probability.

Let $\boldsymbol{\tau}$ be a random ranking. For two matchings γ_1 and γ_2 in $\Gamma(g, \boldsymbol{\tau})$, we denote by $\gamma_1 \succeq_{\boldsymbol{\tau}} \gamma_2$ the fact that $\gamma_1(l) \succ_{\boldsymbol{\tau}} \gamma_2(l)$ for all nodes l of g for which $\gamma_1(l) \neq \gamma_2(l)$. A matching $\gamma \in \Gamma(g, \boldsymbol{\tau})$ is said to be a *top matching of g in $\boldsymbol{\tau}$* if $\gamma \succeq_{\boldsymbol{\tau}} \gamma'$ for all matchings $\gamma' \in \Gamma(g, \boldsymbol{\tau})$.

Example 5.1. Consider the model $\mathsf{RIM}_{\mathsf{L}}(\boldsymbol{\sigma}, \Pi, \lambda)$ of Example 4.7, and the pattern g of Figure 4a. The example also defines a ranking $\boldsymbol{\tau}$ and two matchings γ_1 and γ_2 in $\Gamma(g, \boldsymbol{\tau})$. Then we have $\gamma_1 \succeq_{\boldsymbol{\tau}} \gamma_2$ and, in fact, γ_1 is a top matching of g in $\boldsymbol{\tau}$. □

Example 5.2. Consider the model $\mathsf{RIM}_{\mathsf{L}}(\boldsymbol{\sigma}^{\mathbf{s}}, \Pi^{\mathbf{s}}, \lambda)$ constructed in Example 4.9, and let g be the label pattern of Figure 4b. Let $\boldsymbol{\tau}$ be given by

$$\boldsymbol{\tau} \stackrel{\text{def}}{=\!=} \langle \texttt{Clinton}\,\texttt{Trump}, \texttt{Rubio}, \texttt{Sanders} \rangle .$$

Then a top matching of g in $\boldsymbol{\tau}$ is the following:

$$\{l \mapsto \texttt{Clinton}\,\texttt{Trump}\,,\ \texttt{Trump} \mapsto \texttt{Trump}\,,\ \texttt{Sanders} \mapsto \texttt{Sanders}\}$$

We denote this matching by γ_0 for later reference. □

The following lemma shows that if there is any matching of g in $\boldsymbol{\tau}$, then there is a *unique* top matching. Recall that $(\boldsymbol{\tau}, \lambda) \models g$ denotes that $\Gamma(g, \boldsymbol{\tau})$ is nonempty.

Lemma 5.3. *For all random rankings $\boldsymbol{\tau}$, if $(\boldsymbol{\tau}, \lambda) \models g$ then there is precisely one top matching of g in $\boldsymbol{\tau}$.*

Intuitively, Lemma 5.3 allows us to partition the set of rankings that match g according to their top match. This enables us to reduce the inference task to the one of calculating the probability of the set of possible top matchings.

5.2 Strategy

For a function $\gamma : \text{nodes}(g) \to \text{items}(\boldsymbol{\sigma})$ we denote by $rnk(\boldsymbol{\sigma}, \gamma)$ the set of all rankings $\boldsymbol{\tau} \in rnk(\boldsymbol{\sigma})$ such that $(\boldsymbol{\tau}, \lambda) \models g$ and γ is the top matching of g in $\boldsymbol{\tau}$. By Lemma 5.3 we have that

$$\Pr(g \mid \boldsymbol{\sigma}, \Pi, \lambda) = \sum_{\substack{\gamma:\text{nodes}(g)\to \\ \text{items}(\boldsymbol{\sigma})}} \sum_{\substack{\boldsymbol{\tau}\in \\ rnk(\boldsymbol{\sigma},\gamma)}} \Pi_{\boldsymbol{\sigma}}(\boldsymbol{\tau}) . \tag{2}$$

As there is only a polynomial number of functions from nodes(g) to items$(\boldsymbol{\sigma})$, it suffices to be able to efficiently compute $\sum_{\boldsymbol{\tau} \in rnk(\boldsymbol{\sigma},\gamma)} \Pi_{\boldsymbol{\sigma}}(\boldsymbol{\tau})$, which we denote by p_γ.

$$p_\gamma \stackrel{\text{def}}{=\!=} \sum_{\boldsymbol{\tau} \in rnk(\boldsymbol{\sigma},\gamma)} \Pi_{\boldsymbol{\sigma}}(\boldsymbol{\tau}) \tag{3}$$

In the rest of this section we show how to compute p_γ. Hence, we fix $\gamma : \text{nodes}(g) \to \text{items}(\boldsymbol{\sigma})$, and present an algorithm for computing p_γ. To shorten the notation, in the remainder of this section we will denote $rnk(\boldsymbol{\sigma}, \gamma)$ by $\mathcal{P}$.

We assume that g is acyclic, and that all nodes of g are labels that occur in the image of λ; otherwise, $p_\gamma = 0$. Let l be a node of g. A *child* of l is a node l' such that g has the edge $l \to l'$, and a *parent* of l is a node l' such that g has the edge $l' \to l$. We denote by $ch_g(l)$ and $pa_g(l)$ the sets of all children and all parents of l, respectively.

Denote by $img(\gamma)$ the item set $\{\gamma(l) \mid l \in \text{nodes}(g)\}$. The position of the items in $img(\gamma)$ within a random ranking is described by a mapping $\delta\colon \text{nodes}(g) \to [1, m]$, where $\delta(l)$ represents the position of item $\gamma(l)$ in the ranking. We say that δ is *consistent with g* (or just consistent) if for all edges $l' \to l$ of g we have that $\delta(l') < \delta(l)$. We denote by $\mathcal{R}$ the set of all mappings that are consistent with g. We say that δ is *realized* in a ranking $\boldsymbol{\tau}$, or that $\boldsymbol{\tau}$ *realizes* δ, if $\delta(l) = \boldsymbol{\tau}(\gamma(l))$ for every $l \in \text{nodes}(g)$.

The algorithm is based on the following lemma.

Lemma 5.4. *For all $\boldsymbol{\tau} \in rnk(\boldsymbol{\sigma})$ we have $\boldsymbol{\tau} \in \mathcal{P}$ if and only if there exists a consistent mapping δ such that:*

1. *$\boldsymbol{\tau}$ realizes δ;*
2. *For every $l \in \text{nodes}(g)$ and $\sigma \in \text{items}(\boldsymbol{\tau})$, if $l \in \lambda(\sigma)$ and $\boldsymbol{\tau}(\sigma) < \delta(l)$, then $pa_g(l) \neq \emptyset$ and*

$$\boldsymbol{\tau}(\sigma) < \max_{l' \in pa_g(l)} \delta(l') .$$

In particular, if $pa_g(l) = \emptyset$ then $\gamma(l)$ is the highest ranking item of type l in $\boldsymbol{\tau}$, that is, $\gamma(l) \succ_{\boldsymbol{\tau}} \sigma$ for all σ such that $l \in \lambda(\sigma)$.

Example 5.5. Consider again the $\mathsf{RIM}_{\mathsf{L}}(\boldsymbol{\sigma}^{\mathbf{s}}, \Pi^{\mathbf{s}}, \lambda)$ model constructed in Example 4.9. Let g be the label pattern of Figure 4b. Recall the top matching γ_0 of g in $\boldsymbol{\tau}$ from Example 5.2, and assume that γ is γ_0. Let δ be the mapping $\{l \mapsto 1\,,\ \texttt{Trump} \mapsto 2\,,\ \texttt{Sanders} \mapsto 4\}$. Then δ is consistent with g, and δ is realized in $\boldsymbol{\tau}$. The reader can verify that Lemma 5.4 holds with respect to $\boldsymbol{\tau}$ and δ. □

We denote by $p(\delta)$ the probability that the generated random ranking $\boldsymbol{\tau}$ is such that $\boldsymbol{\tau} \in \mathcal{P}$ and $\boldsymbol{\tau}$ realizes δ:

$$p(\delta) \stackrel{\text{def}}{=\!=} \sum_{\substack{\boldsymbol{\tau}\in\mathcal{P},\\ \boldsymbol{\tau} \text{ realizes } \delta}} \Pi_{\boldsymbol{\sigma}}(\boldsymbol{\tau}) \tag{4}$$

Then, from Lemma 5.3 we conclude that

$$p_\gamma = \sum_{\delta\in\mathcal{R}} p(\delta)\,. \tag{5}$$

Next, we describe an algorithm for computing the $p(\delta)$ for all $\delta \in \mathcal{R}$. The algorithm follows the stochastic insertion process of RIM, with the following twist. We begin the RIM process not with the empty ranking, but rather with one that contains $img(\gamma)$ in an ordering that is consistent with g. Since there can be multiple such orderings, we track all of them simultaneously.

For a formal description, some notation is needed. For $t = 0, \ldots, m$ we denote by the cardinality of the item set $\{\sigma_1, \ldots, \sigma_t\} \cup img(\gamma)$ by s_t, and the set of all consistent mappings $\delta\colon \text{nodes}(g) \to [1, s_t]$ by $\mathcal{R}_t$. In particular, $s_m = m$ and $\mathcal{R}_m = \mathcal{R}$. As usual, the algorithm iterates through the items of $\boldsymbol{\sigma}$ in order, from σ_1 to σ_m. At iteration t, it inserts σ_t into a prefix that contains $\{\sigma_1, \ldots, \sigma_{t-1}\} \cup img(\gamma)$ and realizes a consistent mapping in $\mathcal{R}_{t-1}$, thereby generating a new prefix that realizes a mapping in $\mathcal{R}_t$. If σ_t happens to be in $img(\gamma)$, then the insertion is avoided.

Next, we denote by $\widehat{\mathcal{P}}_t$ the set of all rankings $\boldsymbol{\tau}$ over $\{\sigma_1, \ldots, \sigma_t\} \cup img(\gamma)$ such that for some $\delta \in \mathcal{R}_t$ the two conditions of Lemma 5.4 are satisfied. For $\boldsymbol{\tau} \in \widehat{\mathcal{P}}_t$, we denote by $\boldsymbol{\tau}_{|t}$ the projection of $\boldsymbol{\tau}$ onto $\{\sigma_1, \ldots, \sigma_t\}$. In other words, $\boldsymbol{\tau}_{|t}$ is obtained from $\boldsymbol{\tau}$ by removing the items σ_i of $img(\gamma)$ with $i > t$. In particular, $\boldsymbol{\tau}_{|m} = \boldsymbol{\tau}$.

Finally, for $\delta \in \mathcal{R}_t$ we denote by $\widehat{p}_t(\delta)$ the sum of probabilities given by the (classic) RIM distribution function to all rankings in $\widehat{\mathcal{P}}_t$ that realize δ, projected onto the members of $\{\sigma_1, \ldots, \sigma_t\}$. Formally:

$$\widehat{p}_t(\delta) \stackrel{\text{def}}{=\!=} \sum_{\substack{\boldsymbol{\tau}\in\widehat{\mathcal{P}}_t,\\ \boldsymbol{\tau} \text{ realizes } \delta}} \Pi^t_{\boldsymbol{\sigma}}(\boldsymbol{\tau}_{|t}) \tag{6}$$

where $\Pi^t_{\boldsymbol{\sigma}}(\boldsymbol{\tau}')$ is the probability of generating the ranking $\boldsymbol{\tau}' \in rnk(\{\sigma_1, \ldots, \sigma_t\})$ at time t (i.e., the product of the corresponding insertion probabilities).

From the definition of $p(\delta)$ in Equation (4), the above discussion, and the definition of $\widehat{p}_t(\delta)$ in Equation (6) we conclude that

$$p(\delta) = \widehat{p}_m(\delta)\,. \tag{7}$$

Our algorithm computes $\widehat{p}_t(\delta)$ for all $t \in \{1, \ldots, m\}$ and $\delta \in \mathcal{R}_t$; in particular, the algorithm computes $\widehat{p}_m(\delta)$, and hence the required $p(\delta)$.

5.2.1 Summary

So far we proved that every ranking that matches g has a unique top matching, hence computing $\Pr(g \mid \boldsymbol{\sigma}, \Pi, \lambda)$ reduces, by (2), to computing the probability that a given mapping γ is a top matching for a random ranking $\boldsymbol{\tau}$, denoted p_γ (3). Next, in Lemma 5.4 we translated having γ as a top matching into consistency conditions over the function δ that maps $img(\gamma)$ into specific indices. That allowed us to reduce the computation of p_γ, by (5), to the computation of $p(\delta)$, which the probability that δ is realized in the random $\boldsymbol{\tau}$ (4). Finally, we defined in (6) the function $\widehat{p}_t(\delta)$ that tracks insertions over a series augmented with $img(\gamma)$, so that $p(\delta)$ is precisely $\widehat{p}_m(\delta)$. Hence, computing $p(\delta)$ reduces to computing $\widehat{p}_t(\delta')$ for every $\delta' \in \mathcal{R}_t$. Our algorithm, described in the next section, accomplishes this task and returns p_γ.

5.3 Algorithm Description

We begin by defining the insertion probability function $\Upsilon[i, j, \delta]$ for our algorithm, which returns the probability of inserting item σ_i into index j of a subranking containing items $\{\sigma_1, \ldots, \sigma_{i-1}\} \cup img(\gamma)$, where the members of $img(\gamma)$ are positioned as in $\delta \in \mathcal{R}_{i-1}$. We denote by $img_{>t}(\gamma)$ the items in $img(\gamma)$ with an index larger than t, that is:

$$img_{>t}(\gamma) \stackrel{\text{def}}{=\!=} img(\gamma) \cap \{\sigma_{t+1}, \ldots, \sigma_m\}$$

Intuitively, our insertion function accounts only for inconsistencies that arise between item σ_i and those with a lower index in $\boldsymbol{\sigma}$. Therefore, we can ignore all items of $img_{>i}(\gamma)$ in the subranking by appropriately adjusting the insertion index as follows:

$$\begin{aligned}\Upsilon[i, j, \delta] \stackrel{\text{def}}{=\!=}& \\ &\Pi\left(i, j - |\{\gamma(l) \in img_{>i}(\gamma) \mid \delta(l) < j\}|\right)\end{aligned} \tag{8}$$

That is, $\Upsilon[i, j, \delta]$ is the probability $\Pi[i, j']$, returned by the classic RIM insertion function, for inserting item σ_i into the modified index j' that accounts only for those items whose index is smaller than i.

Recall that $\Pi^t_{\boldsymbol{\sigma}}(\boldsymbol{\tau})$ denotes the probability that RIM generated the subranking $\boldsymbol{\tau}$, at time t. Analogously, we denote by $\Upsilon^t_{\boldsymbol{\sigma}}(\boldsymbol{\tau})$ the probability of the ranking $\boldsymbol{\tau}$ induced by the new insertion function (8). For every consistent mapping $\delta \in \mathcal{R}_t$, we compute the probability

$$q_t(\delta) = \sum_{\substack{\boldsymbol{\tau}\in\mathcal{P}_t,\\ \boldsymbol{\tau} \text{ realizes } \delta}} \Upsilon^t_{\boldsymbol{\sigma}}(\boldsymbol{\tau})\,.$$

The algorithm TopProb (Figure 5) constructs the sets of consistent mappings $\mathcal{R}_1, \ldots, \mathcal{R}_m$ by extending the set of consistent mappings from the previous iteration. The initial set $\mathcal{R}_0$ is simply the set of all rankings over $img(\gamma)$ that are consistent with g, and for every $\delta \in \mathcal{R}_0$, we define $q_0(\delta) = 1$.

At iteration t, a consistent mapping $\delta \in \mathcal{R}_{t-1}$ is extended by inserting item σ_t into every possible *legal* position, as defined by the second condition of Lemma 5.4 (line 3). The subroutine Range implements this constraint based on the values in δ (lines 5-7).

In the case where $\sigma_t \in img(\gamma)$ (that is, $\gamma(l) = \sigma_t$ for some $l \in \text{nodes}(g)$), there is only a single legal position for σ_t, namely $\delta(l)$. Therefore, the mapping does not change (i.e., $\delta' = \delta$). Otherwise, the mapping is updated according to the insertion position j as follows. We denote by δ_{+j} the assignment obtained from δ by inserting an item into index j in the random prefix; that is, $\delta_{+j}(l) = \delta(l) + 1$ whenever $\delta(l) \geq j$, and otherwise $\delta_{+j}(l) = \delta(l)$. The probability of the resulting mapping δ', combining the previous mapping δ and the insertion position j, is calculated in line 8.

In the following section we show that for every consistent mapping $\delta \in \mathcal{R}_t$ it holds that $q_t(\delta) = \widehat{p}_t(\delta)$, proving that the algorithm indeed computes $\widehat{p}_m(\delta)$, and thus $p(\delta)$, for every $\delta \in \mathcal{R}_m$. This allows us to compute p_γ (Eq. (5)).

Algorithm TopProb($\mathsf{RIM_L}(\boldsymbol{\sigma}, \Pi, \lambda), g, \gamma$)

1: **for** $i = 1, \ldots, m$ **do**
2: **for all** $\delta \in \mathcal{R}_{i-1}$ **do**
3: **for all** $j \in \mathsf{Range}(\delta, g, \gamma, \sigma_i)$ **do**
4: **if** $\sigma_i \in img(\gamma)$ **then**
5: $\delta' := \delta$
6: **else**
7: $\delta' := \delta_{+j}$
8: $q_i(\delta') \mathrel{+}= q_{i-1}(\delta) \times \Upsilon(i, j, \delta)$
9: $\mathcal{R}_i := \mathcal{R}_i \cup \{\delta'\}$
10: **return** $\sum_{\delta \in \mathcal{R}_m} q_m(\delta)$

Subroutine $\mathsf{Range}(\delta, g, \gamma, \sigma_i)$

1: **if** $\sigma_i \in img(\gamma)$: $\sigma_i = \gamma(l)$ **then**
2: **return** $\delta(l)$
3: $s_i := i + |img_{>i}(\gamma)|$ {prefix size}
4: range $:= \{1, \ldots, s_i\}$
5: **for all** $\{l \in \lambda(\sigma_i)\}$ **do**
6: $l_p := \arg\max_{l' \in pa_g(l)} \delta(l')$
7: range := range $\setminus \{\delta(l_p) + 1, \ldots, \delta(l)\}$
8: **return** range

Figure 5: An algorithm for computing p_γ

5.4 Correctness

Let $t > 1$ be a position in $\boldsymbol{\sigma}$, and let $\boldsymbol{\tau} \in \widehat{\mathcal{P}}_{t-1}$. Then $\boldsymbol{\tau}_{+j}$ is the ranking that results from inserting σ_t into index j. If $\sigma_t \in img(\gamma)$ then $\boldsymbol{\tau}_{+j}$ is simply $\boldsymbol{\tau}$. Otherwise, $\boldsymbol{\tau}_{+j}[i] = \boldsymbol{\tau}[i-1]$ for all $i > j$, $\boldsymbol{\tau}_{+j}[j] = \sigma_t$, and $\boldsymbol{\tau}_{+j}[i] = \boldsymbol{\tau}[i]$ in case $i < j$. As before, $\boldsymbol{\tau}_{|t}$ is the ranking projected onto the items with index at most t. We begin with the following lemma.

LEMMA 5.6. *The following hold for all iterations t of* TopProb.

1. *For all legal indices j returned by* Range *and $\boldsymbol{\tau} \in \widehat{\mathcal{P}}_{t-1}$ we have $\boldsymbol{\tau}_{+j} \in \widehat{\mathcal{P}}_t$.*
2. *For every $\boldsymbol{\tau} \in \widehat{\mathcal{P}}_t$ there exists a ranking $\boldsymbol{\tau}' \in \widehat{\mathcal{P}}_{t-1}$ and a legal index j returned by* Range *such that $\tau'_{+j} = \tau$.*

Lemma 5.6 implies that the algorithm correctly constructs the $\mathcal{R}_t$. The lemma is needed for the next one.

LEMMA 5.7. *In the execution of* TopProb *we have $q_i(\delta) = \widehat{p}_i(\delta)$ for all $i \in \{1, \ldots, m\}$ and $\delta \in \mathcal{R}_i$.*

From Lemma 5.7 we conclude that $q_m(\delta) = \widehat{p}_m(\delta) = p(\delta)$ for all $\delta \in \mathcal{R}$. Therefore, from (5) we get the desired probability, as stated in the following theorem.

THEOREM 5.8. *Algorithm* TopProb *returns p_γ (Eq. (5)).*

Finally, the following theorem gives an upper bound on the execution cost of the algorithm.

THEOREM 5.9. *(Complexity) Algorithm* TopProb *runs in time $O(m^{|\text{nodes}(g)|+2})$.*

PROOF. The number of consistent mappings that the algorithm processes is in $O(m^{|\text{nodes}(g)|})$. For each mapping, all possible insertion positions are inspected, and this is repeated for every index $i \in \{1, \ldots, m\}$, overall the complexity of the algorithm is $O(m^{|\text{nodes}(g)|+2})$. □

Therefore, we establish the main result of this section.

THEOREM 5.10. *For every fixed label pattern g, the probability $\Pr(g \mid \boldsymbol{\sigma}, \Pi, \lambda)$ can be computed in polynomial time on a given $\mathsf{RIM_L}(\boldsymbol{\sigma}, \Pi, \lambda)$ model.*

In particular, from the reduction of Section 4.4 and Theorem 5.10 we get the correctness of Theorem 4.4.

5.5 Incorporating Min/Max Conditions

Consider a labeled RIM model $\mathsf{RIM_L}(\boldsymbol{\sigma}, \Pi, \lambda)$. We now describe an extension of the algorithm TopProb to handle conditions over the *minimal* and *maximal* index where a label occurs in the random ranking. In the context of our running example, this allows us to compute probabilities of the following kinds of events.

1. Every Democratic candidate is preferred to every Republican candidate.
2. Every female Republican is preferred to every male Democrat.
3. Hillary Clinton is among the top 3 choices.
4. A Libertarian is among the bottom 3 choices.
5. Every Green candidate is ranked higher than every Republican and lower than every Democrat.

We support such queries as follows. Let $\mathsf{RIM_L}(\boldsymbol{\sigma}, \Pi, \lambda)$ be a labeled RIM model. We denote by $\boldsymbol{\Lambda}_\lambda$ the (finite) set of all labels that occur in the image of λ. The positions of the minimal and maximal items associated with label $l \in \boldsymbol{\Lambda}_\lambda$, within a random ranking, is described by the following pair of mappings:

$$\alpha \colon \boldsymbol{\Lambda}_\lambda \to [1, m] \tag{9}$$
$$\beta \colon \boldsymbol{\Lambda}_\lambda \to [1, m] \tag{10}$$

That is, $\alpha(l)$ and $\beta(l)$ represent the positions of the highest and lowest ranked items of type l in a random ranking, respectively. We say that α and β are *realized* in a ranking $\boldsymbol{\tau}$, or that $\boldsymbol{\tau}$ *realizes* α and β, if for every $l \in \boldsymbol{\Lambda}_\lambda$:

$$\alpha(l) = \min\{i \mid l \in \lambda(\tau_i)\}$$
$$\beta(l) = \max\{i \mid l \in \lambda(\tau_i)\}$$

To phrase the above examples as conditions over minimal and maximal indices of labels, suppose that the labels l_D, l_G, l_L and l_R stand for Democratic, Green, Libertarian and Republican parties, respectively. The first event states that $\beta(l_D) < \alpha(l_R)$, the fourth event states that $\beta(l_L) \geq m-2$ (m being the number of candidates), and the fifth event states that $\alpha(l_R) > \beta(l_G)$ and $\alpha(l_G) > \beta(l_D)$.

For a fixed number of labels, there is only a polynomial number of possible mappings α and β (see (9) and (10)). Our extension of the algorithm TopProb computes the marginal probability of each pair of such mappings. Hence, the algorithm can evaluate *every* fixed condition over these mappings, as long as the numerical operations are assumed to

Algorithm $\mathsf{TopProbMinMax}(\mathsf{RIM_L}(\boldsymbol{\sigma}, \Pi, \lambda), g, \gamma, \varphi)$

```
1: for i = 1, ..., m do
2:   for all ⟨δ, α, β⟩ ∈ R_{i-1} do
3:     for all j ∈ Range(δ, g, γ, σ_i) do
4:       if σ_i ∈ img(γ) then
5:         ⟨δ', α', β'⟩ := ⟨δ, α, β⟩
6:       else
7:         ⟨δ', α', β'⟩ := ⟨δ_{+j}, α_{+j}, β_{+j}⟩
8:       q_i(⟨δ', α', β'⟩) += q_{i-1}(⟨δ, α, β⟩) × Υ(i, j, δ)
9:       R_i := R_i ∪ ⟨δ', α', β'⟩
10:      for all l ∈ Λ_λ | l ∈ λ(σ_i) do
11:        α'(l) := min(α'(l), j)
12:        β'(l) := max(β'(l), j)
13: return Σ_{⟨δ,α,β⟩ ∈ R_m | ⟨α,β⟩ ⊨ φ} q_m(⟨δ, α, β⟩)
```

Figure 6: A dynamic programming algorithm for computing $p_{\gamma,\varphi}$

take constant time, as in the *Real RAM* model [5]. (And for ordinary Turing machines, the complexity should be adjusted to incorporate bitwise operations.)

We now state the formal conclusion. Let $\boldsymbol{\Lambda}_\lambda = \{l_1, \ldots, l_k\}$. By *computable min/max condition* we refer to any Boolean recursive (computable) condition $\varphi(x_1, \ldots, x_k, y_1, \ldots, y_k)$ over the natural numbers where x_i and y_i represent the values of $\alpha(l_i)$ and $\beta(l_i)$ respectively. The mappings α and β are essentially an *assignment* to the variables of φ. We say that the mappings α and β *satisfy* a min/max condition φ, denoted $\langle \alpha, \beta \rangle \models \varphi$ if substituting the values of these mappings to the corresponding variables causes φ to evaluate to true.

Given a label pattern g, we denote by $\Pr(g \wedge \varphi \mid \boldsymbol{\sigma}, \Pi, \lambda)$ the probability that a random ranking $\boldsymbol{\tau}$ matches g (i.e., $(\boldsymbol{\tau}, \lambda) \models g$), and realizes a pair of mappings α and β that satisfy φ (i.e., $\langle \alpha, \beta \rangle \models \varphi$). In particular, we denote by $p_{\gamma,\varphi}$ the probability that γ is the top matching of g in a random ranking (see (3)) that also realizes a pair of mappings α and β, which satisfy φ.

THEOREM 5.11. *For every fixed label pattern g and fixed computable min/max condition φ, the probability* $\Pr(g \wedge \varphi \mid \boldsymbol{\sigma}, \Pi, \lambda)$ *is computable in polynomial time on Real RAM for given labeled RIM models* $\mathsf{RIM_L}(\boldsymbol{\sigma}, \Pi, \lambda)$.

Theorem 5.11 is realized by the dynamic programming algorithm TopProbMinMax (Figure 6), which is an extension of the algorithm TopProb (Figure 5), and explained below.

The constraints defined by a min/max condition φ are incorporated into the dynamic programming algorithm (Figure 6) as follows. We extend the set of mappings maintained by the algorithm to a triple $\langle \delta, \alpha, \beta \rangle$ where δ remains unchanged (recall that $\delta\colon \text{nodes}(g) \mapsto [1, m]$ represents the position of the item $\gamma(l)$ in a random ranking), and mappings α and β are as previously described (see (9) and (10)). A triple $\langle \delta, \alpha, \beta \rangle$ is *consistent* if δ is consistent (that is, for all edges $l' \to l$ of g the inequality $\delta(l') < \delta(l)$ holds).

The new algorithm TopProbMinMax, supporting min/max conditions is depicted in Figure 6. The subroutine Range is the same as in algorithm TopProb, and is therefore omitted. The updating of the values in mappings α and β, for each label $l \in \boldsymbol{\Lambda}_\lambda$, is performed in lines 10-12. The rest of the algorithm follows in a straightforward manner from algorithm TopProb.

The extended algorithm maintains and processes a larger state space, leading to increased complexity. Nevertheless, the number of possible consistent mappings $\langle \delta, \alpha, \beta \rangle$ is limited to $O(m^{3|\boldsymbol{\Lambda}_\lambda|})$. While larger than the space maintained by the original algorithm (TopProb), this is still exponential only in the size of the query, preserving the polynomial data complexity.

6. CONCLUDING REMARKS

In our past vision paper [18] we made the case for incorporating preferences into the relational model, to allow for queries that involve popular operations over preferences. In this paper we proposed a novel angle to this framework—querying probabilistic preferences by utilizing the expressive power of the relational model. Our investigation of **RIM**-PPDs illustrates that our framework gives rise to new challenges in two domains: databases and statistical inference. In particular, we showed how the evaluation of CQs leads to a novel and natural inference problem over RIM models, and we presented a nontrivial algorithm for solving this problem. We conclude this paper by proposing several directions for future research.

Theoretical directions include the generalization of our results in various directions: more general query languages (e.g., larger fragments of FO), a more general dichotomy in the complexity of CQs, approximate query evaluation (e.g., PTAS on Boolean CQ evaluation), and additional statistical models of preferences (e.g., Choi et al. [8]). Furthermore, it is important to establish general conditions where tractability can be attained under *combined* (i.e., query-and-data) complexity.

Practical directions are, naturally, abundant. While our algorithm provides the guarantee of polynomial time, translating it into a practical implementation over preferences of realistic volumes presents significant challenges. Our long-term goal is to build a database system that incorporates preferences in the PPD model, and allows for both non-trivial preference-specific operations (e.g., rank aggregation) and statistical inference.

Towards this goal, we will propose efficient physical representations of preferences. Further, we will develop efficient algorithms for exact answering of probabilistic preference queries that leverage CPU parallelism and data parallelism in distributed architectures. Finally, we will explore performance optimization opportunities that can be derived from approximate query answering. Our ongoing and future systems work will build on the theoretical insights developed in this paper.

Acknowledgments

This work was supported in part by the US National Science Foundation (NSF) Grants No. 1464327 and 1539856, by the US-Israel Binational Science Foundation (BSF) Grant No. 2014391, and by the Israel Science Foundation (ISF) Grant No. 1295/15. Benny Kimelfeld is a Taub Fellow, supported by the Taub Foundation.

7. REFERENCES

[1] S. Aerts, D. Lambrechts, S. Maity, P. V. Loo, B. Coessens, F. D. Smet, L.-C. Tranchevent, B. D. Moor, P. Marynen, B. Hassan, P. Carmeliet, and Y. Moreau. Gene prioritization through genomic data fusion. *Nature Biotechnology*, 24(5):537–544, 2006.

[2] A. Arvanitis and G. Koutrika. Prefdb: Supporting preferences as first-class citizens in relational databases. *IEEE Trans. Knowl. Data Eng.*, 26(6):1430–1446, 2014.

[3] P. Awasthi, A. Blum, O. Sheffet, and A. Vijayaraghavan. Learning mixtures of ranking models. In *NIPS*, pages 2609–2617, 2014.

[4] S. Balakrishnan and S. Chopra. Two of a kind or the ratings game? adaptive pairwise preferences and latent factor models. *Frontiers of Computer Science*, 6(2):197–208, 2012.

[5] L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: *np*- completeness, recursive functions and universal machines. *Bull. Amer. Math. Soc. (N.S.)*, 21(1):1–46, 07 1989.

[6] G. R. Brightwell and P. Winkler. Counting linear extensions is #P-complete. In *STOC*, pages 175–181, 1991.

[7] L. M. Busse, P. Orbanz, and J. M. Buhmann. Cluster analysis of heterogeneous rank data. In *ICML*, pages 113–120, 2007.

[8] A. Choi, G. V. den Broeck, and A. Darwiche. Tractable learning for structured probability spaces: A case study in learning preference distributions. In *IJCAI*, pages 2861–2868. AAAI Press, 2015.

[9] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *VLDB*, pages 864–875. Morgan Kaufmann, 2004.

[10] P. Diaconis. A generalization of spectral analysis with applications to ranked data. *Annals of Statistics*, 17(3):949–979, 1989.

[11] W. Ding, P. Ishwar, and V. Saligrama. Learning mixed membership mallows models from pairwise comparisons. *CoRR*, abs/1504.00757, 2015.

[12] J.-P. Doignon, A. Pekeč, and M. Regenwetter. The repeated insertion model for rankings: Missing link between two subset choice models. *Psychometrika*, 69(1):33–54, 2004.

[13] M. A. Fligner and J. S. Verducci. Distance based ranking models. *Journal of the Royal Statistical Society. Series B (Methodological)*, 48(3):359–369, 1986.

[14] M. A. Fligner and J. S. Verducci. Multistage ranking models. *Journal of the American Statistical Association*, 83(403):892–901, 1988.

[15] I. C. Gormley and T. B. Murphy. A latent space model for rank data. In *ICML*, 2006.

[16] I. C. Gormley and T. B. Murphy. A mixture of experts model for rank data with applications in election studies. *The Annals of Applied Statistics*, 2(4):1452–1477, 12 2008.

[17] J. Huang, A. Kapoor, and C. Guestrin. Riffled independence for efficient inference with partial rankings. *J. Artif. Intell. Res. (JAIR)*, 44:491–532, 2012.

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

[19] T. Kamishima and S. Akaho. Supervised ordering by regression combined with thurstone's model. *Artif. Intell. Rev.*, 25(3):231–246, 2006.

[20] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

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

[22] R. Kolde, S. Laur, P. Adler, and J. Vilo. Robust rank aggregation for gene list integration and meta-analysis. *Bioinformatics*, 28(4):573–580, 2012.

[23] G. Lebanon and J. D. Lafferty. Cranking: Combining rankings using conditional probability models on permutations. In *ICML*, pages 363–370, 2002.

[24] G. Lebanon and Y. Mao. Non-parametric modeling of partially ranked data. *Journal of Machine Learning Research*, 9:2401–2429, 2008.

[25] T. Lu and C. Boutilier. Effective sampling and learning for mallows models with pairwise-preference data. *J. Mach. Learn. Res.*, 15(1):3783–3829, Jan. 2014.

[26] T. Lukasiewicz, M. V. Martinez, D. Poole, and G. I. Simari. Probabilistic models over weighted orderings: Fixed-parameter tractable variable elimination. In *KR*, pages 494–504. AAAI Press, April 2016.

[27] C. L. Mallows. Non-null ranking models. i. *Biometrika*, 44(1-2):114–130, June 1957.

[28] J. I. Marden. *Analyzing and Modeling Rank Data*. Chapman & Hall, 1995.

[29] G. McElroy and M. Marsh. Candidate gender and voter choice: Analysis from a multimember preferential voting system. *Political Research Quarterly*, 63(4):pp. 822–833, 2010.

[30] C. Meek and M. Meila. Recursive inversion models for permutations. In *NIPS*, pages 631–639. Curran Associates, Inc., 2014.

[31] A. D. Sarma, A. D. Sarma, S. Gollapudi, and R. Panigrahy. Ranking mechanisms in twitter-like forums. In *WSDM*, pages 21–30, 2010.

[32] J. Stoyanovich, L. Ilijasic, and H. Ping. Workload-driven learning of mallows mixtures with pairwise preference data. In *WebDB*, page 8, 2016.

[33] J. Stoyanovich, M. Jacob, and X. Gong. Analyzing crowd rankings. In *WebDB*, pages 41–47, 2015.

[34] J. M. Stuart, E. Segal, D. Koller, and S. K. Kim. A gene-coexpression network for global discovery of conserved genetic modules. *Science*, 302:249–255, 2003.

[35] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[36] S. Toda and M. Ogiwara. Counting classes are at least as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 21(2), 1992.

APPENDIX

A. ADDITIONAL PROOFS

A.1 Proof of Lemma 4.6

LEMMA 4.6. *The evaluation of Q_h on a given* **RIM**-*PPD is* $\mathrm{FP}^{\#\mathrm{P}}$-*hard.*

PROOF. The proof is by reduction from the problem of counting the number of linear extensions of a partial order. The input for this problem consists of a set $A = \{\sigma_1, \ldots, \sigma_m\}$ of items and a partial order $\succ$ over A. The goal is to compute $|rnk(A|\succ)|$, where $rnk(A|\succ)$ is the set of all rankings $\boldsymbol{\sigma} \in rnk(A)$ that satisfy $\sigma_i \succ_{\boldsymbol{\sigma}} \sigma_j$ whenever $\sigma_i \succ \sigma_j$. This problem is known to be #P-complete [6]. Given the input A and $\succ$, we construct the following **RIM**-PPD E. The relation R^E is the inverse of $\succ$, that is, it consists of all the pairs (σ_j, σ_i) where $\sigma_i \succ \sigma_j$. The single model of P^E is $\mathsf{RIM}(\boldsymbol{\sigma}, \Pi)$, where $\boldsymbol{\sigma}$ is the ranking $\langle \sigma_1, \ldots, \sigma_m \rangle$ and Π is the insertion function that defines a uniform distribution over the permutations of A; that is: $\Pi(i, j) = \frac{1}{i}$ whenever $1 \leq i \leq m$ and $1 \leq j \leq i$. Note that $\mathsf{RIM}(\boldsymbol{\sigma}, \Pi)$ is the same as $\mathsf{MAL}(\boldsymbol{\sigma}, 1)$.

Observe that every possible world W of the PPD $[\![E]\!]$ consists of R^W, which is the same as R^E, and a linear order over A that is encoded in P^W. Also note that a possible world satisfies Q_h if and only if P^W shares at least one pair with R^W. From the construction of R^W it follows that W satisfies Q_h if and only if P^W is *not* a linear extension of $\succ$. Finally, we observe that every possible world W has the same probability, namely $\frac{1}{m!}$ (where m is the number of items in A). Therefore:

$$conf_{Q_\mathsf{h}}([\![E]\!]) = \frac{m! - |rnk(A|\succ)|}{m!}\,.$$

In particular, we can compute $|rnk(A|\succ)|$ by multiplying $1 - conf_{Q_\mathsf{h}}([\![E]\!])$ by $m!$. □

A.2 Proof of Theorem 4.5

THEOREM 4.5. *Let* **S** *be a preference schema, and let Q be a Boolean CQ over* **S** *such that Q has no self joins and Q has a single p-atom. If Q is not itemwise, then the evaluation of Q on* **RIM**-*PPDs over* **S** *is* $\mathrm{FP}^{\#\mathrm{P}}$-*hard.*

PROOF. We will prove the hardness of evaluating Q over RIM-PPDs by a reduction from the evaluation of the CQ $Q_\mathsf{h}() \leftarrow R(x, y), P(x; y)$, which is shown to be $\mathrm{FP}^{\#\mathrm{P}}$-hard in Lemma 4.6. Let E_h be an input **RIM**-PPD instance over the schema of Q_h. Then E_h maps R to an instance R^{E_h} over R, and P to a **RIM**-instance that consists of a single **RIM**-model, which we denote by M_h.

Let $P_Q(\beta; A_l; A_r)$ be the single p-symbol that appears in Q, and let $P_Q(s_1, \ldots, s_k; x_Q; y_Q)$ be the single p-atom of Q. Observe that x_Q and y_Q must be distinct, and neither is a session variable; otherwise, Q is itemwise by definition, contradicting our assumption. We fix a constant c, and denote by $(c_1, \ldots, c_k)$ the tuple that is obtained from $(s_1, \ldots, s_k)$ by replacing every variable with c. Hence, the tuple $(c_1, \ldots, c_k)$ consists of the constants in $\{s_1, \ldots, s_k\}$ and, possibly, the constant c.

We now construct a **RIM**-instance E over the schema of Q. The **RIM**-model P^E_Q is the pair (r, μ), where the relation r consists of the single tuple $(c_1, \ldots, c_k)$ defined above, and μ maps this tuple to the above model M_h. The o-instances of E are constructed by the following procedure. For each tuple (a, b) of R^{E_h} and o-atom $S(t_1, \ldots, t_n)$ of Q, we add to the S-instance of E the tuple $(d_1, \ldots, d_n)$, where for $i = 1, \ldots, n$ we have:

$$d_i = \begin{cases} t_i & \text{if } t_i \text{ is a constant;} \\ \mathsf{c} & \text{if } t_i \text{ is a session variable;} \\ a & \text{if } t_i = x_Q; \\ b & \text{otherwise.} \end{cases}$$

Observe that every variable z of Q that is neither x_Q nor a session variable is replaced with b. In particular, y_Q is replaced with b. Let p be a path of $\mathcal{G}^\circ_Q$ that does not visit any session variable. The existence of p is guaranteed, since Q is not itemwise. Then all the variables along p are replaced with b, except for x_Q that is replaced with a. In particular, there is an edge in p such that one incident node is replaced with a and one with b.

This completes the construction of E. To complete the proof, we will show that

$$\Pr(Q([\![E]\!]) = \mathbf{true}) \;=\; \Pr(Q_\mathsf{h}([\![E_h]\!]) = \mathbf{true})\,. \qquad (11)$$

Let Q_o be the CQ that is obtained from Q by removing the single p-atom. From the construction of E we get the following.

1. For every tuple (a, b) of R^{E_h} there is a homomorphism from Q_o to E that maps x_Q and y_Q to a and b, respectively.
2. For every homomorphism from Q_o to E that maps x_Q and y_Q to a and b, respectively, the tuple (a, b) is in R^{E_h}.

We mention here that the second item uses the fact that there are no self joins, and is not necessarily true otherwise.

Finally, from the construction of P^E_Q it follows that there is a bijection φ between the possible worlds of $[\![E_h]\!]$ and $[\![E]\!]$ such that for every possible world D of $[\![E_h]\!]$ we have that $Q_\mathsf{h}(D) = Q(\varphi(D))$, and moreover, D and $\varphi(D)$ have the same probability. Hence, Equation (11) immediately follows, as required. □

A.3 Proof of Lemma 5.3

LEMMA 5.3. *For all random rankings $\boldsymbol{\tau}$, if $(\boldsymbol{\tau}, \lambda) \models g$ then there is precisely one top matching of g in $\boldsymbol{\tau}$.*

PROOF. If $(\boldsymbol{\tau}, \lambda) \models g$ then there exists a matching of g in $\boldsymbol{\tau}$. This means that there is a mapping from the nodes of g to the items of $\boldsymbol{\tau}$ that is consistent with the edges of g. Since $\boldsymbol{\tau}$ defines a total order over its items and in particular over the items mapped by the matching, then g must be acyclic.

We prove the claim by induction on $|\text{nodes}(g)|$. If g contains a single node l, then $\gamma^*(l) = \tau_k$ where $k = \min\{i \in \{1, \ldots, m\} \mid l \in \lambda(\tau_i)\}$. Since $\boldsymbol{\tau}$ matches g, then k exists, and by the minimality condition is unique.

Assume correctness for patterns with n nodes, and we prove the claim for a pattern with $n + 1$ nodes. Consider a node $l \in \text{nodes}(g)$ with no outgoing edges in g. Since g is acyclic such a node must exist. Now consider the pattern induced by $\text{nodes}(g) \setminus \{l\}$ (i.e., without l and its incoming edges), defined over n nodes. Since $\boldsymbol{\tau}$ matches g it must also match the pattern induced by $\text{nodes}(g) \setminus \{l\}$, denoted g^n. By the induction hypothesis, $\boldsymbol{\tau}$ has a single, unique top matching γ^*_n corresponding to g^n. Let τ_k be an item in $\boldsymbol{\tau}$

that is associated with l (i.e., $l \in \lambda(\tau_k)$). We define k as follows. If $pa_g(l) \neq \emptyset$ then τ_k is the highest ranked item of type l that appears after $\gamma_n^*(u)$ for each $u \in pa_g(l)$:

$$k = \min \left\{ i > \max_{u \in pa_g(l)} \boldsymbol{\tau}(\gamma_n^*(u)) \mid l \in \lambda(\tau_i) \right\} \quad (12)$$

Recall that $\boldsymbol{\tau}(a)$ denotes the index of item a in the ranking. Otherwise, if $pa_g(l) = \emptyset$, then:

$$k = \min \{ i > 0 \mid l \in \lambda(\tau_i) \}$$

Since $\boldsymbol{\tau}$ matches g, such an index k must exist, and due to the minimality condition, is unique. Define the matching γ_{n+1} as follows:

$$\gamma_{n+1}(u) = \begin{cases} \gamma_n^*(u) & \text{if } u \neq l \\ \tau_k & \text{otherwise} \end{cases}$$

We now show that γ_{n+1} is both minimum and unique. By the induction hypothesis γ_n^* is both minimum and unique for the pattern g excluding the node l. Therefore, the top matching for g must be identical to γ_n^* for all nodes in $\text{nodes}(g) \setminus \{l\}$. By the minimality and uniqueness of the index k, we get that γ_{n+1} is also minimum and unique, as required. □

A.4 Proof of Lemma 5.4

LEMMA 5.4. *For all $\boldsymbol{\tau} \in rnk(\boldsymbol{\sigma})$ we have $\boldsymbol{\tau} \in \mathcal{P}$ if and only if there exists a consistent mapping δ such that:*

1. *$\boldsymbol{\tau}$ realizes δ;*
2. *For every $l \in \text{nodes}(g)$ and $\sigma \in \text{items}(\boldsymbol{\tau})$, if $l \in \lambda(\sigma)$ and $\boldsymbol{\tau}(\sigma) < \delta(l)$, then $pa_g(l) \neq \emptyset$ and*

$$\boldsymbol{\tau}(\sigma) < \max_{l' \in pa_g(l)} \delta(l') .$$

PROOF. We prove each direction separately.

The "only if" direction. Let $\boldsymbol{\tau} \in \mathcal{P}$. Hence, γ is a top matching for g in $\boldsymbol{\tau}$. We will show that there exists a consistent mapping δ such that $\boldsymbol{\tau}$ and δ meet the conditions of the lemma. Define the mapping $\delta\colon \text{nodes}(g) \to [1, m]$ such that $\delta(l)$ is $\boldsymbol{\tau}(\gamma(l))$, that is, the index of $\gamma(l)$ in $\boldsymbol{\tau}$. By this definition, $\boldsymbol{\tau}$ clearly realizes δ, and δ is consistent with g.

Assume, by way of contradiction, that the second item of the lemma is not met. That is, there exists a label $l \in \text{nodes}(g)$, and an item τ_i such that *(a)* $l \in \lambda(\tau_i)$, *(b)* $\tau_i \succ_{\boldsymbol{\tau}} \gamma(l)$, *and (c)* either $pa_g(l) = \emptyset$ or $i > \max_{l' \in pa_g(l)} \delta(l')$. In this case, the mapping $\gamma'\colon \text{nodes}(g) \to [1, m]$ defined as:

$$\gamma'(v) = \begin{cases} \gamma(v) & \text{if } v \neq l \\ \tau_i & \text{otherwise} \end{cases}$$

is a matching for $\boldsymbol{\tau}$ (i.e., $\gamma' \in \Gamma(g, \boldsymbol{\tau})$), and $\gamma' \succeq_{\boldsymbol{\tau}} \gamma$. Therefore, we arrive at a contradiction that γ is a top matching for g in $\boldsymbol{\tau}$.

The "if" direction. Now assume that $\boldsymbol{\tau}$ realizes a consistent mapping δ such that $\boldsymbol{\tau}$ and δ meet the conditions of the lemma. We will show that $\boldsymbol{\tau} \in \mathcal{P}$. The fact that $\boldsymbol{\tau}$ realizes δ means that δ maps the items of $img(\gamma)$ to their positions in $\boldsymbol{\tau}$. Therefore, the matching γ is defined by $\gamma(l) = \boldsymbol{\tau}(\delta(l))$. Since δ is consistent, we get that γ is a matching of g in $\boldsymbol{\tau}$ (i.e., $\gamma \in \Gamma(g, \boldsymbol{\tau})$).

Assume, by way of contradiction, that there exists a matching γ' of g in $\boldsymbol{\tau}$ such that $\gamma' \neq \gamma$ and $\gamma' \succeq_{\boldsymbol{\tau}} \gamma$. (Recall from Lemma 5.3 that the top matching is unique.) This means that there exists a label $l \in \text{nodes}(g)$ such that $\gamma'(l) \succ_{\boldsymbol{\tau}} \gamma(l)$. We can assume that g is acyclic, since otherwise $\Gamma(g, \boldsymbol{\tau})$ is empty. Let $l \in \text{nodes}(g)$ be the smallest node in some topological order over g, for which $\gamma'(l) \succ_{\boldsymbol{\tau}} \gamma(l)$.

If $pa_g(l) = \emptyset$, then we immediately arrive at a contradiction of the second condition of the lemma. So, we assume that $pa_g(l) \neq \emptyset$. Since γ' is a matching, it holds that $\gamma'(l)$ appears after all items $\gamma'(l_p)$ where $l_p \in pa_g(l)$. Moreover, the fact that l was chosen to be first in some topological order over g implies that $\gamma'(l_p) = \gamma(l_p)$ for every node $l_p \in pa_g(l)$. Therefore, the label l and the item $\sigma = \gamma'(l)$ are in violation of the second condition of the lemma, proving the claim. □

A.5 Proof of Lemma 5.6

We denote by $\boldsymbol{\tau}_{-j}$ the ranking that results from removing item $\boldsymbol{\tau}[j]$ from the ranking $\boldsymbol{\tau}$. Specifically, $\boldsymbol{\tau}_{-j}[i] = \boldsymbol{\tau}[i]$ for all $i < j$, and $\boldsymbol{\tau}_{-j}[i] = \boldsymbol{\tau}[i+1]$ otherwise.

LEMMA 5.6. *The following hold for all iterations t of* TopProb.

1. *For all legal indices j returned by* Range *and $\boldsymbol{\tau} \in \widehat{\mathcal{P}}_{t-1}$ we have $\boldsymbol{\tau}_{+j} \in \widehat{\mathcal{P}}_t$.*
2. *For every $\boldsymbol{\tau} \in \widehat{\mathcal{P}}_t$ there exists a ranking $\boldsymbol{\tau}' \in \widehat{\mathcal{P}}_{t-1}$ and a legal index j returned by* Range *such that $\tau'_{+j} = \tau$.*

PROOF. We recall that $\widehat{\mathcal{P}}_t$ is the set of all rankings $\boldsymbol{\tau}$ over $\{\sigma_1, \ldots, \sigma_t\} \cup img(\gamma)$ such that for some $\delta \in \mathcal{R}_t$ the two conditions of Lemma 5.4 are satisfied.

We first prove the claim for the case where $\sigma_t \in img(\gamma)$. In this case, the rankings of both $\widehat{\mathcal{P}}_{t-1}$ and $\widehat{\mathcal{P}}_t$ are defined over the items in $\{\sigma_1, \ldots, \sigma_t\} \cup img_{>t}(\gamma)$. For this reason we also have that $\mathcal{R}_{t-1} = \mathcal{R}_t$. Hence, we get that $\widehat{\mathcal{P}}_{t-1} = \widehat{\mathcal{P}}_t$, proving both claims. In the remainder of the proof we assume that $\sigma_t \notin img(\gamma)$.

We first prove Part 1. Assume that $\boldsymbol{\tau} \in \widehat{\mathcal{P}}_{t-1}$, and let $\delta \in \mathcal{R}_{t-1}$ be the mapping realized by $\boldsymbol{\tau}$. Then for every legal index j returned by Range, $\boldsymbol{\tau}_{+j}$ realizes the mapping δ_{+j}, which is consistent because δ is consistent. Therefore, $\delta_{+j} \in \mathcal{R}_t$. Furthermore, according to Range, the position j is guaranteed to ensure the second condition of Lemma 5.4 with respect to the mapping δ_{+j}. Therefore, by definition of $\widehat{\mathcal{P}}_t$ we get that $\boldsymbol{\tau}_{+j} \in \widehat{\mathcal{P}}_t$.

We now prove Part 2. Assume that $\boldsymbol{\tau} \in \widehat{\mathcal{P}}_t$, and let $\delta \in \mathcal{R}_t$ such that $\boldsymbol{\tau}$ realizes δ. Therefore, every item in $\boldsymbol{\tau}$ is in a legal position with respect to δ. In particular, σ_t is in a legal position j of $\boldsymbol{\tau}$ (i.e., $\tau_j = \sigma_t$). We consider the ranking $\boldsymbol{\tau}_{-j}$. Since $\boldsymbol{\tau} \in \widehat{\mathcal{P}}_t$ realizes $\delta \in \mathcal{R}_t$, then $\boldsymbol{\tau}_{-j}$ realizes the mapping δ_{-j}, which is consistent because δ is consistent, thus $\delta_{-j} \in \mathcal{R}_{t-1}$. Since $\boldsymbol{\tau}$ and $\delta \in \mathcal{R}_t$ meet the conditions of Lemma 5.4, then $\boldsymbol{\tau}_{-j}$ and $\delta_{-j} \in \mathcal{R}_{t-1}$ do so as well, because the $-j$ operator cannot cause a violation of the lemma's second condition. From that, we get that $\boldsymbol{\tau}_{-j} \in \widehat{\mathcal{P}}_{t-1}$. □

A.6 Proof of Lemma 5.7

LEMMA 5.7. *In the execution of* TopProb *we have $q_i(\delta) = \widehat{p}_i(\delta)$ for all $i \in \{1, \ldots, m\}$ and $\delta \in \mathcal{R}_i$.*

PROOF. The proof of the lemma follows an inductive argument on the index i. As the basis of the induction, consider any consistent mapping $\delta \in \mathcal{R}_1$. The probability calculated by the algorithm (line 8) is

$$q_1(\delta) = q_0(\delta_0) \times \Upsilon(1, j, \delta) = 1$$

for any $j \in \{1, \ldots, |img(\gamma)| + 1\}$. This is because $q_0(\delta_0) = 1$ by definition, and $\Upsilon(1, j, \delta) = \Pi(1, 1) = 1$ for any $j \in \{1, \ldots, |img(\gamma)|+1\}$ (see (8)). We now show that $\widehat{p}_1(\delta) = 1$, proving the basis. Let $\boldsymbol{\tau} \in \widehat{\mathcal{P}}_1$. Since $\boldsymbol{\tau}_{|1} = \langle \sigma_1 \rangle$, we get, by (6), that $\widehat{p}_1(\delta) = \Pi^1_{\boldsymbol{\sigma}}(\boldsymbol{\tau}_{|1}) = 1$. This proves that $q_1(\delta) = \widehat{p}_1(\delta)$ as required.

We now assume correctness for $i-1$, and prove that $q_i(\delta) = \widehat{p}_i(\delta)$. Assume that $\sigma_i \in img(\gamma)$. In this case, let $l \in \text{nodes}(g)$ be such that $\sigma_i = \gamma(l)$. Therefore, σ_i is inserted into position $\delta(l)$ (line 2 of Range), and the resulting realization $\delta' \in \mathcal{R}_i$ is identical to $\delta \in \mathcal{R}_{i-1}$ (line 5 of TopProb). Therefore,

$$q_i(\delta) = q_{i-1}(\delta) \times \Upsilon(i, \delta(l), \delta)\,.$$

By the induction hypothesis $q_{i-1}(\delta) = \widehat{p}_{i-1}(\delta)$, and combined with (6), we get that

$$\begin{aligned} q_i(\delta) &= \widehat{p}_{i-1}(\delta) \times \Upsilon(i, \delta(l), \delta) \\ &= \sum_{\substack{\boldsymbol{\tau} \in \widehat{\mathcal{P}}_{i-1}, \\ \boldsymbol{\tau} \text{ realizes } \delta}} \Pi^{i-1}_{\boldsymbol{\sigma}}(\boldsymbol{\tau}_{|i-1}) \times \Upsilon(i, \delta(l), \delta)\,. \end{aligned} \tag{13}$$

Now, consider any ranking $\boldsymbol{\tau}_{|i-1}$, where $\boldsymbol{\tau} \in \widehat{\mathcal{P}}_{i-1}$. Since $\sigma_i \in img(\gamma)$ is positioned at index $\delta(l)$ of $\boldsymbol{\tau}$, the ranking that results from inserting item σ_i into position $\delta(l)$ in the ranking $\boldsymbol{\tau}_{|i-1}$ is exactly $\boldsymbol{\tau}_{|i}$. Therefore,

$$\Pi^i_{\boldsymbol{\sigma}}(\boldsymbol{\tau}_{|i}) = \Pi^{i-1}_{\boldsymbol{\sigma}}(\boldsymbol{\sigma}, \boldsymbol{\tau}_{|i-1}) \times \Upsilon(i, \delta(l), \delta)\,. \tag{14}$$

Furthermore, since $\sigma_i \in img(\gamma)$, $\widehat{\mathcal{P}}_{i-1}$ and $\widehat{\mathcal{P}}_i$ are defined over the same set of items $\{\sigma_1, \ldots, \sigma_i\} \cup img_{>i}(\gamma)$, and for the same reason $\mathcal{R}_{i-1} = \mathcal{R}_i$. Therefore, $\widehat{\mathcal{P}}_{i-1} = \widehat{\mathcal{P}}_i$, and in particular,

$$\{\boldsymbol{\tau} \in \widehat{\mathcal{P}}_{i-1} \mid \boldsymbol{\tau} \text{ realizes } \delta\} = \{\boldsymbol{\tau} \in \widehat{\mathcal{P}}_i \mid \boldsymbol{\tau} \text{ realizes } \delta'\}\,. \tag{15}$$

Substituting (14) and (15) into (13) proves the inductive step.

For the second case we assume that $\sigma_i \notin img(\gamma)$. Note that in this case, for every ranking $\boldsymbol{\tau} \in \widehat{\mathcal{P}}_{i-1}$ we have that $\boldsymbol{\tau}_{|i-1}$ is identical to $\boldsymbol{\tau}_{|i}$. We denote by $I(i, \delta)$ the set of legal indexes returned by the procedure Range$(\delta, g, \gamma, \sigma_i)$. Finally, for brevity, we denote by $\widehat{\mathcal{P}}_t(\delta)$ to be the set of rankings in $\widehat{\mathcal{P}}_t$ that realize δ. We start by expressing the probability $q_i(\delta)$ computed by TopProb.

$$q_i(\delta) = \sum_{\delta' \in \mathcal{R}_{i-1}} q_{i-1}(\delta') \sum_{\substack{j \in I(i,\delta), \\ \delta'_{+j} = \delta}} \Upsilon(i, j, \delta) \tag{16}$$

By the induction hypothesis we have that $q_{i-1}(\delta') = \widehat{p}_{i-1}(\delta')$, and therefore,

$$q_i(\delta) = \sum_{\delta' \in \mathcal{R}_{i-1}} \widehat{p}_{i-1}(\delta') \sum_{\substack{j \in I(i,\delta), \\ \delta'_{+j} = \delta}} \Upsilon(i, j, \delta)\,. \tag{17}$$

We substitute the exact definition of $\widehat{p}_{i-1}(\delta')$ by (6) and get the following.

$$q_i(\delta) = \sum_{\delta' \in \mathcal{R}_{i-1}} \sum_{\boldsymbol{\tau} \in \widehat{\mathcal{P}}_{i-1}(\delta')} \Pi^{i-1}_{\boldsymbol{\sigma}}(\boldsymbol{\tau}_{|i-1}) \sum_{\substack{j \in I(i,\delta), \\ \delta'_{+j} = \delta}} \Upsilon(i, j, \delta) \tag{18}$$

As we remarked, since $\sigma_t \notin img(\gamma)$, then $\boldsymbol{\tau}_{|i-1} = \boldsymbol{\tau}_{|i}$. Therefore, we can write:

$$q_i(\delta) = \sum_{\delta' \in \mathcal{R}_{i-1}} \sum_{\boldsymbol{\tau} \in \widehat{\mathcal{P}}_{i-1}(\delta')} \sum_{\substack{j \in I(i,\delta), \\ \delta'_{+j} = \delta}} \Pi^{i-1}_{\boldsymbol{\sigma}}(\boldsymbol{\tau}_{|i}) \cdot \Upsilon(i, j, \delta) \tag{19}$$

According to (8), $\Upsilon(i, j, \delta) = \Pi(i, j')$, where j' is the modified index that considers only the items $\{\sigma_1, \ldots, \sigma_{i-1}\}$. That is, j' is the index that corresponds to j for the ranking $\boldsymbol{\tau}_{|i}$. By the definition of $\boldsymbol{\tau}_{+j}$, we have that

$$\Pi^{i-1}_{\boldsymbol{\sigma}}(\boldsymbol{\tau}_{|i}) \cdot \Upsilon(i, j, \delta) = \Pi^i_{\boldsymbol{\sigma}}((\boldsymbol{\tau}_{|i})_{+j})\,.$$

By substituting and reversing the summation order we get that

$$q_i(\delta) = \sum_{\delta' \in \mathcal{R}_{i-1}} \sum_{\substack{j \in I(i,\delta), \\ \delta'_{+j} = \delta}} \sum_{\boldsymbol{\tau} \in \widehat{\mathcal{P}}_{i-1}(\delta')} \Pi^i_{\boldsymbol{\sigma}}((\boldsymbol{\tau}_{|i})_{+j})\,. \tag{20}$$

So now, we only need to show that the set of rankings aggregated in (20) is precisely $\widehat{\mathcal{P}}_i(\delta)$. We denote this set of rankings by $\widehat{\mathcal{Q}}_i(\delta)$ and express it as follows.

$$\widehat{\mathcal{Q}}_i(\delta) = \{\boldsymbol{\tau} \mid \boldsymbol{\tau} \text{ realizes } \delta \text{ and } \exists j \in I(i, \delta) \text{ s.t. } \boldsymbol{\tau}_{-j} \in \mathcal{P}_{i-1}\}$$

From Lemma 5.6 we conclude that $\widehat{\mathcal{Q}}_i(\delta) = \widehat{\mathcal{P}}_i(\delta)$, as required. □