Learning From Query-Answers: A Scalable Approach to Belief Updating and Parameter Learning

NICCOLÒ MENEGHETTI
OLIVER KENNEDY, University at Buffalo, USA
WOLFGANG GATTERBAUER, Northeastern University, USA

Tuple-independent and disjoint-independent probabilistic databases (TI- and DI-PDBs) represent uncertain data in a factorized form as a product of independent random variables that represent either tuples (TI-PDBs) or sets of tuples (DI-PDBs). When the user submits a query, the database derives the marginal probabilities of each output-tuple, exploiting the underlying assumptions of statistical independence. While query processing in TI- and DI-PDBs has been studied extensively, limited research has been dedicated to the problems of *updating or deriving the parameters from observations of query results*. Addressing this problem is the main focus of this paper. We first introduce *Beta Probabilistic Databases* (B-PDBs), a generalization of TI-PDBs designed to support both (i) *belief updating* and (ii) *parameter learning* in a principled and scalable way. The key idea of B-PDBs is to treat each parameter as a latent, Beta-distributed random variable. We show how this simple expedient enables both belief updating and parameter learning in a principled way, without imposing any burden on regular query processing. Building on B-PDBs, we then introduce *Dirichlet Probabilistic Databases* (D-PDBs), a generalization of DI-PDBs with similar properties. We provide the following key contributions for both B- and D-PDBs: (*i*) we study the complexity of performing Bayesian belief updates and devise efficient algorithms for certain tractable classes of queries; (*ii*) we propose a soft-EM algorithm for computing maximum-likelihood estimates of the parameters; (*iii*) we present an algorithm for efficiently computing conditional probabilities, allowing us to efficiently implement B- and D-PDBs via a standard relational engine; and (*iv*) we support our conclusions with extensive experimental results.

CCS Concepts: • **Information systems** → **Uncertainty**; • **Theory of computation** → *Incomplete, inconsistent, and uncertain databases*; • **Mathematics of computing** → Maximum likelihood estimation; Bayesian computation;

Additional Key Words and Phrases: Probabilistic Databases, uncertain data management, parameter learning, Bayesian updates, maximum likelihood estimation

ACM Reference Format:
Niccolò Meneghetti, Oliver Kennedy, and Wolfgang Gatterbauer. 2018. Learning From Query-Answers: A Scalable Approach to Belief Updating and Parameter Learning. *ACM Trans. Datab. Syst.* 1, 1, Article 1 (January 2018), 41 pages. https://doi.org/10.1145/3277503

1 INTRODUCTION

Uncertain data arises in numerous settings, including data exchange, ETL, approximate query processing, and more. In the last decade, the challenge of posing queries over uncertain data (data specified by a probability distribution) has received considerable attention by the database

Authors' addresses: Niccolò Meneghetti, niccolom@buffalo.edu; Oliver Kennedy, University at Buffalo, 338 Davis Hall, Buffalo, NY, 14260-2500, USA, okennedy@buffalo.edu; Wolfgang Gatterbauer, Northeastern University, 360 Huntington Avenue, Boston, MA, 02115-5000, USA, wolfgang@ccis.neu.edu.

0362-5915/2018/1-ART1 $15.00
https://doi.org/10.1145/3277503

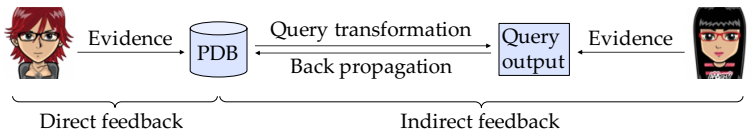


Fig. 1. In most classical probabilistic databases (PDBs), users directly provide probabilities for data in the PDB ("*Direct Feedback*"). In *Beta- and Dirichlet-Probabilistic Databases*, users provide probabilities for the outputs of queries over the database ("*Indirect Feedback*"). The methods developed in this paper give a principled approach for propagating this information back to the database and appropriately changing the probabilities.

community [2, 5, 7, 9, 19, 25, 32, 33, 39, 52, 55, 56], and querying uncertain data is relatively well understood. However, deriving the probability distribution behind a probabilistic database can be significantly harder [16, 17], particularly when facts about the data are noisy ("Ada *probably* works for HP"), or indirect ("Ada works for a tech company in Boston").

In this article we address the challenges of building a probabilistic database from a noisy, indirect signal. We propose two new models: *Beta-Probabilistic Databases (B-PDBs)* and *Dirichlet-Probabilistic Databases (D-PDBs)* that extend existing models of probabilistic databases with a principled process for updating the database's probability distribution ("belief updating"), or for deriving it from scratch ("parameter learning"). As illustrated in Figure 1, the information used to update or derive either of the new models may be indirect, on the output of a query. Concretely, our models can incorporate any information that can be expressed in terms of the probability of a Boolean expression holding over the database and — as an instance of "reverse data management" [45] — propagate this feedback back to the database. This information may also be noisy: The new information may itself be sampled, as in a poll or a vote, or crowdsourced. Most importantly, both new models are completely backwards compatible with existing, widely used models. B-PDBs generalize the Tuple-Independent model [5, 9, 11, 21] for probabilistic databases (TI-PDBs), while D-PDBs generalize both B-PDBs and the Disjoint-Independent model [2, 32] for probabilistic databases (DI-PDBs, also sometimes called X-Tuples). This backwards compatibility allows us to freely leverage query processing techniques developed for TI-PDBs and DI-PDBs, like probabilistic relational algebra (pRA) [21], Monte Carlo simulations [7, 33, 38], anytime approximations [13, 20], dissociations [8, 22, 23], lineage-based methods [26] and more. B- and D-PDBs enable a more powerful form of pay-as-you-go feedback on query results [34, 60], as well as probabilistic forms of in-database constraint programming [27, 37] and of virtualized experiments [15]. Above all else, B- and D-PDBs provide a systematic way of training any finite probabilistic relation instance [58].

Example 1.1 (Running example). Consider a hypothetical data aggregator combining information from a variety of sources like LinkedIn, Facebook, or Twitter, for example to make hiring decisions. Based on a tweet, the aggregator is able to infer that Ada is employed in Boston, but does not learn her employer. Given a table of employers `E(name,emp)` and locations `L(emp,loc)`, an equivalent assertion is given by the following existential query:

q_1 = `exists(select * from E natural join L where E.name='Ada' and L.loc='Boston')`

This information could then, in principle, be propagated directly into the database. However, there are at least two sources of uncertainty that could make this assertion unreliable: First, we may not be certain that the assertion is correct, for example as a result of ambiguity in the tweet it was extracted from. Secondly, the information extracted from this tweet may directly contradict prior evidence. Both of our proposed models make it possible to account for both factors when updating the underlying database: In the B-PDB (resp., D-PDB) model, each tuple (resp., x-tuple)

E^{p}

name	emp	*tid*	θ
Ada	HP	x_1	.6
Ada	IBM	x_2	.6
Bob	HP	x_3	.5

(a) TI-PDB

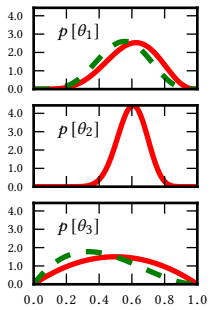
$E^{(a,b)}$ before observing $\neg(x_1 \vee x_3)$

name	emp	*tid*	a	b	$\langle\theta\rangle$	$h[\theta]$
Ada	HP	x_1	6	4	.6	−0.507
Ada	IBM	x_2	18	12	.6	−1.014
Bob	HP	x_3	2	2	.5	−0.125

after observing $\neg(x_1 \vee x_3)$

name	emp	*tid*	a	b	$\langle\theta\rangle$	$h[\theta]$
Ada	HP	x_1	6	5	.54	−0.528
Ada	IBM	x_2	18	12	.6	−1.014
Bob	HP	x_3	2	3	.4	−0.235

(b) B-PDB



(c) $p[\theta_i]$

Fig. 2. Running example (a): A simple TI-PDB with three tuples. (b, c): A corresponding B-PDB, *before* (solid red) and *after* (dashed green) observing the answer "no" to the query q_5 = `exists(select * from E where emp='HP')`. Notice that while tuples x_1 and x_2 exhibit the same marginal probability, the confidence on the estimate of $\mathbb{P}[x_2]$ is higher than the confidence on the estimate of $\mathbb{P}[x_1]$, as the differential entropy measured on θ_1 is larger than the differential entropy of θ_2 (columns theta and h shown for convenience only).

in the probabilistic database has a true, independent probability (resp., probability distribution). We know the tuples of the database, but only have an inaccurate estimate of their probabilities. Our goal is to learn the true probability based on repeated samples taken from queries over that database.

We initially focus exclusively on the Tuple-Independent model. In this model, a database is a set of n tuples $\{x_1, \ldots, x_n\}$, annotated with independent probabilities $\{\theta_1, \ldots, \theta_n\}$. It represents a standard relational database whose internal state is uncertain; the set of its plausible states (its "possible worlds") consists of the power-set of $\{x_1, \ldots, x_n\}$. The probability of a possible world w is simply the probability of selecting its tuples independently: $\mathbb{P}[w] = \prod_{x_i \in w} \theta_i \cdot \prod_{x_j \notin w}(1 - \theta_j)$. Given a Boolean query q, the probability of q being true is equal to the sum of the probabilities of the possible worlds that satisfy q: $\mathbb{P}[q] = \sum_{w \models q} \mathbb{P}[w]$.

Notice that our focus on independent input tuples (and later disjoint-independent tuples) does not limit the generality of our approach. It is known (e.g., [22, 51, 58]) that any correlation between probabilistic events can be captured by general Boolean functions over *independent events* only (not just *disjoint-independent events*). Hence, any finite set of possible worlds may be encoded as a view over a TI-PDB. For completeness, we present the full argument in appendix C.

Example 1.2 (Running example continued). Continuing example 1.1, we could use a TI-PDB to encode noisy knowledge about employment histories, for example from a source like Facebook (see Figure 2a). If we want to find out whether the person named Ada ever had an employer, we run the following Boolean query over the probabilistic employer relation E^{p}.

$$q_2 = \texttt{exists(select * from E}^{\mathsf{p}}\texttt{ where name='Ada')} \tag{1}$$

The answer is expected to be true ("Ada had at least one employer in the past") with probability 0.84 and false ("Ada never had a job") with probability 0.16.[1]

Let's now assume that there exists a TI-PDB $\mathcal{D}$ whose parameters are hidden, but that we would like to recover. To do so, we have a limited ability to gather (noisy) evidence (denoted $\mathcal{E}$) about

[1]The probabilities follow from $1 - [(1 - 0.6) \cdot (1 - 0.6)] = 0.84$. Also notice that we adopt here the *Closed-World Assumption* (CWA) as is common with TI-PDBs: any missing tuple is assumed to have probability 0.

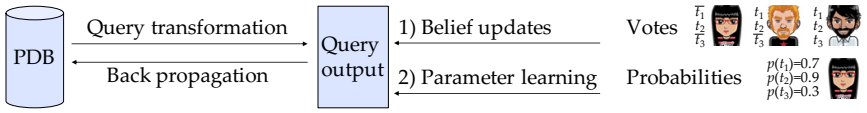


Fig. 3. Beta- and Dirichlet-Probabilistic Databases support both (1) *Belief updates* to merge new, sampled observations with the prior PDB state, and (2) *Parameter Learning* to derive a PDB that would produce the desired query probabilities.

$\mathcal{D}$. We encode $\mathcal{E}$ as (i) a set of Boolean queries $Q \stackrel{\text{def}}{=} \{q_1, \ldots, q_k\}$, and (ii) a finite set of query-answers $\mathcal{E} = \bigcup_i \mathcal{E}_{q_i}$ sampled from $\mathbb{P}[q_i]$, for each q_i in Q. Note that this approach generalizes to non-Boolean queries as well: A single non-boolean query with k possible results may be expanded into a set of k Boolean queries, individually testing the existence of each possible result.

Example 1.3 (Deriving Q). Our approach is agnostic to how the set of boolean queries Q are obtained for a given evidence set $\mathcal{E}$. However, to create context for the remainder of our work, we illustrate how Q would be derived in three potential applications of B-PDBs and D-PDBs: (i) Direct Feedback, (ii) Query Feedback, and (iii) Probabilistic Data Exchange.

Direct Feedback. Mimicking classical probabilistic databases, we might learn from (potentially noisy) direct evidence about tuples already in $\mathcal{D}$. In this case, Q selects the specific tuples we have evidence about. For example, given probabilities for tuples with tuple identifiers (TIDs) $t_1, \ldots t_k$ from relation $\mathtt{E}^p$, we would use: $Q = \left\{ \texttt{exists(select * from E}^p \texttt{ where tid=}t_i\texttt{)} \;\middle|\; i \in [1,k] \right\}$.

Query Feedback. In a second usage pattern, the user first queries the database, identifies potential errors in the result, and provides feedback about them. In this case, Q is defined to be (the Boolean expansion of) the user's query or queries.

Probabilistic Data Exchange. A third possibility is that we wish to learn from probabilities or votes regarding source facts that do not appear directly as tuples in the database (as in example 1.1). As in data exchange, we accomplish this by first defining the source facts in terms of a view over the target database. Q is then defined by the resulting view queries.

We assume each sample is drawn independently from all the others, and we use $\mathcal{E} = \bigcup_i \mathcal{E}_{q_i}$ to denote the "evidence", i.e. the whole set of samples. This article focuses mainly on two problems, as illustrated in Figure 3.

(1) **Belief updating**: given an initial hypothesis consisting of estimates of the hidden parameters of $\mathcal{D}$ and our confidence in each estimate, we show how to refine the hypothesis by incorporating the new evidence $\mathcal{E}$.
(2) **Parameter learning**: we show how to derive a new TI-PDB (resp., DI-PDB) from scratch, relying only on the given evidence.

Belief updating is useful when someone wants to improve an already reliable probabilistic model, exploiting some new, previously unseen, evidence. For example, let's assume we trust the information stored in relation E^p, but we want to improve our knowledge about Ada's work history. In order to do so, we submit a query q_3 "has Ada ever worked for IBM?"[2] to 10 independent data banks. If at least 7 of them answer "yes", then we may want to increase parameter θ_1 in E^p (whose initial value is 0.6) to reflect this additional information. Clearly, the extent of the adjustment will depend on how strong our initial confidence about the value of θ_1 was. Belief updating is about computing these adjustments to prior beliefs in a principled way.

[2]In SQL: q_3 = `exists(select * from E where name='Ada' and emp='IBM')`.

Parameter learning is about using evidence or expert inputs to build a new probabilistic model from scratch. In contrast to belief updating, we start with no prior and our aim is solely to find a model consistent with the available evidence. For example: if we are told that the query

$$q_4(emp) = \texttt{select emp from E where name='Ada'}$$

should return the answer {HP, IBM} ("Ada worked for both HP and IBM") with relative frequency 0.32 (or some other answer with relative frequency $1-0.32 = 0.68$), and should return the empty set $\emptyset$ ("Ada has not worked before") with relative frequency 0.12 (or some other non-empty answer with relative frequency $1-0.12 = 0.88$), then our goal becomes to choose θ_1 and θ_2 so that the model (E^{p}) exhibits the desired marginal probabilities for q. This is achieved either when $(\theta_1, \theta_2) = (0.4, 0.8)$ or when $(\theta_1, \theta_2) = (0.8, 0.4)$[3].

In the context of parameter learning, we will consider the option of collapsing multiple query-answers into a single update. For each update, we will assume a set of k distinct Boolean queries (e.g., the expansion of a non-Boolean query) and that each query is evaluated on exactly s samples of the ground truth database. We denote by $\boldsymbol{\tau} = \{\tau_1, \ldots, \tau_k\}$ the fraction of positive answers for each Therefore, we model the evidence $\mathcal{E}$ as a mapping that associates each query to its observed relative frequency of positive answers: $\mathcal{E} \stackrel{\text{def}}{=} \langle\{(q_1, \tau_1), (q_2, \tau_2), \ldots, (q_k, \tau_k)\},\ s\rangle$.

Example 1.4 (Running example continued). If we crawled the web and retrieved 25 LinkedIn profiles that are all plausible, equally likely matches for the entity named Ada, and all of them but 4 report some unspecified work experience, then we can associate the relative frequency $\tau = 0.84$ to query (1), and set $s = 25$. Evidence for other queries can be collected in a similar fashion.

The key idea behind B-PDBs is to model the parameters $\theta_i, i \in [n]$ as Beta-distributed latent random variables. This simple expedient allows us to (i) model *both* our current estimate of a probability and our confidence in a natural way, and (ii) to deploy a principled way to update those estimates in the presence of new evidence. We illustrate with Figure 2b (first table) a simple B-PDB, consisting of a single relation $E^{(\mathrm{a,b})}$, generalizing the TI-relation E^{p} we introduced in Figure 2a. Notice that for each tuple x_i the parameter θ_i has been replaced by *two* parameters, a_i and b_i. The symbol θ_i is now used to identify a $[0, 1]$-valued random variable, whose probability density is Beta-distributed:

$$p[\theta_i] \stackrel{\text{def}}{=} \theta_i^{a_i - 1} \cdot (1 - \theta_i)^{b_i - 1} \cdot B(a_i, b_i)^{-1} \tag{2}$$

Here $B(\cdot, \cdot)$ denotes the Beta function, which serves as a normalizing factor[4]. The three solid-red plots on the right in Figure 2 depict the density functions $p[\theta]$ for each tuple in the B-PDB. An intuitive way to understand Beta distributions is to think about their parameters a and b as votes. Under this interpretation, the first row for x_1 in the B-PDB represents a poll where the query "has Ada ever worked for HP?" has received $a_1 = 6$ positive answers and $b_1 = 4$ negative ones. Similarly, the second row for x_2 can be seen as a poll where the query "has Ada ever worked for IBM?" has received $a_2 = 18$ positive votes and $b_2 = 12$ negative ones. While the relative frequency of positive answers is the same for both the polls ($0.6 = \frac{6}{6+4} = \frac{18}{18+12}$), the second poll should be considered more "informative", as it involves more votes (30 vs. 10 votes). Consistently with this intuition, the plots of $p[\theta_1]$ and $p[\theta_2]$ both have a peak at 0.6, but the former exhibits a higher entropy, while the latter shows a stronger concentration around the expectation.

[3] Since $0.32 = 0.8 \cdot 0.4$ and $0.12 = (1-0.8)(1-0.4)$. Notice that q_4 is not a Boolean query. Nonetheless, the example is well defined: it is possible to write a Boolean query to verify whether the answer to $\mathbf{q}$ is {HP, IBM}, and another one to verify whether the answer is $\emptyset$.

[4] Chapter 25 of [35] is a good introduction to Beta distributions.

In B-PDBs, the *marginal probability* of a tuple x_i (that we denote with $\mathbb{P}[x_i]$) is equal to the *expected value* of the random variable θ_i, that we denote with $\langle\theta_i\rangle_{p[\theta_i]}$

$$\mathbb{P}[x_i] = \langle\theta_i\rangle_{p[\theta_i]} = \int_0^1 \theta_i \cdot p[\theta_i]\, d\theta_i \tag{3}$$

$$= \frac{a_i}{a_i + b_i} \tag{4}$$

It is well known [35] that the integral in eq. (3) admits the closed-form given in eq. (4). It follows that B-PDBs are *indistinguishable* from regular TI-PDBs when it comes to query processing: all existing inference techniques, both exact and approximate, that have been proposed in the past for TI-PDBs (e.g., [11, 20, 23, 38, 48, 50]) can be readily applied to B-PDBs. To do so, it is sufficient to convert the B-PDB into a TI-PDB by computing the expectation $\langle\theta_i\rangle_{p[\theta_i]}$ for each and every tuple in the database, in constant time for each tuple. Therefore, B-PDBs are a *conservative generalization* of TI-PDBs that add a principled way to update the parameters. Together with D-PDBs, they form the main focus of this article. The first table of Figure 2b shows the tuples' marginal probabilities inside column $\langle\boldsymbol{\theta}\rangle$. This column is shown for readers' convenience only; it is not explicitly stored in a B-PDB. It is immediate to verify that the given B-PDB is equivalent, in terms of query processing, to the relation E^{p} introduced in Figure 2a.

Beyond estimating tuples' probabilities (and unlike standard TI-PDBs), B-PDBs can also evaluate the *confidence* of such estimates. In the remainder of this article we adopt the *differential entropy* $h[\theta_i]$ as a metric for the confidence of the B-PDB's estimates of $\mathbb{P}[x_i]$[5]. By definition, the differential entropy of θ_i is:

$$h[\theta_i] \stackrel{\text{def}}{=} -\int_0^1 p[\theta_i] \cdot \ln(p[\theta_i])\, d\theta_i$$

The above integral admits a well-known [18, 43] closed form[6]. In contrast to the standard entropy, differential entropy $h[\theta_i]$ is defined between 0 and $-\infty$. Intuitively, the smaller $h[\theta_i]$ is, the higher is the confidence of the estimate of $\mathbb{P}[x_i]$, with 0 denoting a uniform distribution and uncertainty vanishing entirely at the limit of $-\infty$. For the readers' convenience, Figure 2b shows the differential entropies in the column $h[\boldsymbol{\theta}]$[7].

In the following sections, we will describe extensively how B-PDBs, and later how D-PDBs support belief updates. We introduce the former case here with a simple example. The second table in Figure 2b shows the effect of performing a belief update to incorporate the observation of a single, negative answer to the Boolean query

$$q_5 = \texttt{exists(select * from E where emp='HP')}\,.$$

Intuitively, the observation suggests that neither Ada nor Bob have worked in the past for HP. We react to this new information by adding a negative vote to both the first and the third tuple in the B-PDB. The adjusted probability densities of θ_1 and θ_3 are plotted on the right, in dashed green. Notice that the update has a greater impact on $p[\theta_3]$ than on $p[\theta_1]$, consistent with the fact that our confidence in θ_1 is higher: $h[\theta_1] < h[\theta_3]$. In the general case, belief updates may involve thousands of records, and affect the parameters of a B-PDB in a non-trivial way.

In the remainder of this article, we study (i) belief updates (Section 4) and (ii) parameter learning (Section 5), first in the context of B-PDBs, and later extending our approach to the more general class of D-PDBs (Section 7); we show how to perform both when we observe answers to an arbitrary set

[5] We use differential entropy and not standard deviation because $\mathbb{P}[x_i]$ is a probability.

[6] The closed solution is $h[\theta_i] = \ln(B(a_i, b_i)) - [(a_i - 1)\cdot(\psi(a_i) - \psi(a_i + b_i))] - [(b_i - 1)\cdot(\psi(b_i) - \psi(a_i + b_i))$, where $\psi(\cdot)$ denotes the Digamma function.

[7] As before, this information is given for readers' convenience only and does not need to be stored inside the B-PDB.

of conjunctive, self-join-free queries, and show efficient implementations for safe queries (Section 6). While the problem of parameter learning has been studied before, to the best of our knowledge the preliminary work on B-PDBs [46] was the first attempt to enable bayesian updates on probabilistic databases. This article extends that effort with a new model (D-PDBs) that generalizes the DI-PDB model for probabilistic databases; We also use the longer format to provide additional examples and experiments for clarity.

Our contributions include:

(1) **Bayesian belief updates.** Given a B-PDB and a set of queries' results, we show how to incorporate the new evidence into the B-PDB in a *Bayesian* fashion. We analyze the complexity of computing such Bayesian updates and provide efficient algorithms for tractable classes of queries.
(2) **Parameter learning.** We devise a soft-expectation-maximization algorithm for computing the maximum likelihood estimate of the parameters $\{\theta_1, \ldots, \theta_n\}$ w.r.t. some given queries' results.
(3) **Conditional probabilities.** We present an algorithm for efficiently computing the probability of a query result conditioned on observed evidence.
(4) **Generalization to DI-PDBs.** We introduce D-PDBs, an extension of B-PDBs that generalizes the DI-PDB model of probabilistic databases with support for bayesian belief updates and parameter learning.
(5) **Benchmarks.** We show how the algorithms we propose can be easily embedded into a standard relational engine, so to exploit its optimization features. We test our framework against real (YAGO2, IPUMS) and synthetic (TCP-H) data sets, annotated with probabilities.

Relative to the preliminary work [46], item 4 is entirely new material.

2 BACKGROUND

In this section, we review some background, notions, and contextualize B-PDBs w.r.t. previous work on probabilistic databases. For the sake of conciseness, we use the following notation: given a real number z we denote by $\overline{z}$ its complement $(1 - z)$. When φ is a Boolean expression, $\overline{\varphi}$ denotes, as usual, its negation $\neg\varphi$.

2.1 Relational Databases

A relational database consists of a finite collection of relations $\{R, S, T, \ldots\}$, over a finite set of n tuples $\{x_1, \ldots, x_n\}$. A conjunctive query $\mathbf{q}$ is a first-order formula in prenex normal form, respecting the following restrictions: (i) each predicate symbol represents a relation, (ii) all variables are either existentially quantified or quantifier-free, (iii) the formula is negation-free and (iv) disjunction-free. We use capital letters to denote first-order logic variables. For example:

$$\mathbf{q}(Z) = \exists X\ \exists Y\ E(X, Y) \wedge L(X, Z) \tag{5}$$

We denote by $\mathtt{hvar}(\mathbf{q})$ the set of free ("head-") variables of $\mathbf{q}$, and by $\mathtt{evar}(\mathbf{q})$ the set of existentially quantified variables. A conjunctive query is said to be *self-join-free* iff every relation name appears at most once; it is said to be *Boolean* iff there are no free variables. Given a database instance $\mathcal{D}$, every non-Boolean conjunctive query can be seen as a collection of Boolean queries, one for each possible grounding of the free variables to values in their *active domain* [1]. In the remainder of this paper we assume queries are always conjunctive and self-join-free. With limited abuse of notation we will denote non-Boolean queries as vectors $(\mathbf{q})$ and Boolean ones as an indexed vectors' components (q_j) that range over the groundings of $\mathbf{q}$. Given a database instance $\mathcal{D}$ and a Boolean

query q, we denote by $\Phi_{\mathcal{D}}(q)$ the *lineage* [4, 6, 26] of q, a propositional Boolean formula over the alphabet $\{x_1, \ldots, x_n\}$, built by the following recursive rules:

- $\Phi_{\mathcal{D}}(q) = \Phi_{\mathcal{D}}(q'_1) \vee \ldots \vee \Phi_{\mathcal{D}}(q'_k)$ when $q = \exists X\ \mathbf{q}'$, and $\mathtt{hvar}(\mathbf{q}') = \{X\}$ and $\{q'_1, \ldots, q'_k\}$ are the groundings of $\mathbf{q}'$ obtained by replacing X with one of the constants in its active domain
- $\Phi_{\mathcal{D}}(q) = \Phi_{\mathcal{D}}(q') \wedge \Phi_{\mathcal{D}}(q'')$ when $q = q' \wedge q''$
- $\Phi_{\mathcal{D}}(q) = x_i$, when q is a ground atom of tuple x_i[8]

A lineage expression φ is said to be *read-once* iff each literal appears at most once. It is straightforward to extend the definition of lineage to query answers: if φ is the lineage of q then the answer $\top$ has lineage φ, while the answer $\bot$ has lineage $\overline{\varphi}$. In the following we often identify Boolean queries with their lineage. For the sake of compactness we sometimes omit the $\wedge$ symbol in lineage expressions (therefore, x_1x_2 is an abbreviation for $x_1 \wedge x_2$) and use the common Datalog notation to express conjunctive queries; for example: $\mathbf{q}(Z) \coloneq E(X, Y), L(X, Z)$.

Given a variable X and a query $\mathbf{q}$, we denote by $\mathtt{at}(X, \mathbf{q})$ the set of $\mathbf{q}$'s atoms where X appears. Variables that appear in every atom of $\mathbf{q}$ are called *root variables*. We say that a query $\mathbf{q}$ is *hierarchical* iff, for any two existential variables (X, Y), either $\mathtt{at}(X, \mathbf{q}) \subseteq \mathtt{at}(Y, \mathbf{q})$ or $\mathtt{at}(Y, \mathbf{q}) \subseteq \mathtt{at}(X, \mathbf{q})$ or $\mathtt{at}(X, \mathbf{q}) \cap \mathtt{at}(Y, \mathbf{q}) = \emptyset$ holds.

Example 2.1 (continued). The query $\mathbf{q}$ from Equation (5) is hierarchical; the set of head-variables $\mathtt{hvar}(\mathbf{q})$ contains only Z, while $\mathtt{evar}(\mathbf{q})$ consists of $\{X, Y\}$. X is a root variable, but Y is not. Let $\mathcal{D}$ be a database instance where the relations E and L are defined as follows:

E		
name	emp	*tid*
Ada	HP	x_1
Ada	IBM	x_2
Bob	HP	x_3

L		
name	lng	*tid*
Ada	eng	x_4
Bob	eng	x_5
Bob	ita	x_6

Within $\mathcal{D}$ the active domain of Z is $\{eng, ita\}$. Therefore query $\mathbf{q}$ can be seen as a collection of two Boolean queries:

$$q_1 \coloneq E(X, Y), L(X, eng). \qquad q_2 \coloneq E(X, Y), L(X, ita).$$

Their lineage expressions are:

$$\Phi_{\mathcal{D}}(q_1) = x_1x_4 \vee x_2x_4 \vee x_3x_5 \qquad \Phi_{\mathcal{D}}(q_2) = x_3x_6$$

The lineage of q_2 is read-once, while the lineage of q_1 is not, as the literal x_4 is used twice (we will later show how to obtain a read-once expression for q_1).

We define *query plans* as sentences respecting the following grammar:

$$P ::= R \mid \pi_X(P') \mid \sigma(P') \mid \ \bowtie [P', P'', \ldots]$$

where R denotes an arbitrary relation name and projections (π), selections (σ) and natural joins ($\bowtie$) have the usual semantics. It is straightforward to extend the notation we use for queries to query plans: if P denotes a plan, then $\mathtt{hvar}(P)$ is the set of attributes in its output schema, while $\mathtt{evar}(P)$ denotes the set of attributes that are projected-away. In the following we write $\pi_{-X}(P)$ as short form for $\pi_{\mathtt{hvar}(P)\setminus\{X\}}(P)$. A plan P is Boolean when $\mathtt{hvar}(P)$ is empty. Given a database instance $\mathcal{D}$, a non-Boolean plan P can be seen as a collection of Boolean plans $\{P_1, \ldots, P_k\}$, one for each of its output-tuples. Each plan in $\{P_1, \ldots, P_k\}$ is obtained from P by substituting its head variables by the constants of the respective output-tuple. A query plan always corresponds to exactly one query, but one query may have multiple distinct query plans. Two query plans are logically equivalent if they answer the same query. Given a database instance $\mathcal{D}$ and a Boolean

[8] In this paper "atoms" are atomic first-order logic formulas. For example, the query from Equation (5) contains two atoms, $E(X, Y)$ and $L(X, Y)$. An atom is *ground* when it has no variables: $E(\mathtt{'Ada'}, \mathtt{'HP'})$.

plan P we denote by $\Phi_{\mathcal{D}}(P)$ the lineage of P, a Boolean expression built according to the following recursive rules:

- **Atom**: If $P = R$ then $\Phi_{\mathcal{D}}(P) = x$ where x identifies the grounded tuple in R.
- **Join**: If $P = P' \bowtie P''$ then $\Phi_{\mathcal{D}}(P) = \Phi_{\mathcal{D}}(P') \wedge \Phi_{\mathcal{D}}(P'')$.
- **Select/Project**: If $P = \pi_{\emptyset}(\sigma(P'))$ then $\Phi_{\mathcal{D}}(P) = \Phi_{\mathcal{D}}(P'_1) \vee \Phi_{\mathcal{D}}(P'_2) \vee \ldots \vee \Phi_{\mathcal{D}}(P'_k)$ assuming that $\{P'_1, P'_2, \ldots, P'_k\}$ are the *Boolean* plans corresponding to the output-tuples of $\sigma(P')$.

If plan P answers query q, then $\Phi_{\mathcal{D}}(P)$ is logically equivalent to $\Phi_{\mathcal{D}}(q)$, for every $\mathcal{D}$.

Example 2.2 (continued). Both the following query plans

$$P' = \pi_{-X}(\pi_{-Y}(E) \bowtie L) \qquad P'' = \pi_{-XY}(E \bowtie L)$$

compute the correct answer for query **q** from Equation (5), but they produce different lineage expressions:

P'	
lng	$\Phi_{\mathcal{D}}$
eng	$((x_1 \vee x_2)\, x_4) \vee x_3 x_5$
ita	$x_3 x_6$

P''	
lng	$\Phi_{\mathcal{D}}$
eng	$x_1 x_4 \vee x_2 x_4 \vee x_3 x_5$
ita	$x_3 x_6$

Notice that all the lineage expressions produced by P' are read-once and logically equivalent to the corresponding lineage expressions of **q** and P''.

2.2 Tuple-independent Probabilistic Databases

A tuple-independent probabilistic database (TI-PDB) is a regular relational database where each tuple represents an independent probabilistic event. Each tuple x_i is associated with a Bernoulli-distributed Boolean random variable, expected to be true with probability θ_i and false with probability $\overline{\theta_i}$. It represents the belief that tuple x_i belongs to the database. In slight abuse of notation we use x_i to denote both a tuple and its associated Boolean random variable; we use the vector notation $\boldsymbol{\theta}$ to denote the whole set of parameters $\{\theta_1, \ldots, \theta_n\}$. Unlike deterministic databases, the state of a TI-PDB is uncertain: the set of its plausible states (its "possible worlds") ranges over the power-set of its tuples. Hence, a possible world consists of a subset of tuples, generated by including each tuple x_i with probability θ_i. A TI-PDB $\mathcal{D}$ defines a probability measure $\mathbb{P}[\cdot]$ over possible worlds and Boolean queries. If w is a possible world, we denote by $w[i]$ a function that returns 1 when tuple x_i belongs to w, and 0 otherwise. The probability of w is $\mathbb{P}[w|\mathcal{D}] \stackrel{\text{def}}{=} \prod_{i:w[i]=1} \theta_i \cdot \prod_{i:w[i]=0} \overline{\theta_i}$, the probability of drawing its tuples independently; if q is a Boolean query, its marginal probability is the sum of all possible worlds where q is satisfied: $\mathbb{P}[q|\mathcal{D}] \stackrel{\text{def}}{=} \sum_{w \models q} \mathbb{P}[w]$. If φ is a lineage expression, we denote by $\mathbb{P}[\varphi|\mathcal{D}]$ the probability of φ being satisfied, given that each literal x_i is true with probability θ_i and false otherwise. Notice that $\mathbb{P}[q|\mathcal{D}] = \mathbb{P}[\varphi|\mathcal{D}]$ when $\varphi = \Phi_{\mathcal{D}}(q)$.

TI-PDBs are often associated with *Probabilistic Relational Algebra* (pRA) [21], a generalization of positive relational algebra that consists of three probabilistic operators: independent projection (π^{p}), independent join ($\bowtie^{\mathrm{p}}$) and selection (σ). These operators differ from standard relational algebra in the fact that they associate a score to each output tuple. Let P be the Boolean plan associated with an arbitrary output tuple; its score is computed according to the following recursive rules:

- If P identifies a tuple x_i then $\text{score}(P) = \theta_i$
- If $P = P' \bowtie^{\mathrm{p}} P''$ then $\text{score}(P) = \text{score}(P') \cdot \text{score}(P'')$
- If $P = \sigma(P')$ then $\text{score}(P) = \text{score}(P')$
- If $P = \pi^{\mathrm{p}}_{\emptyset}(P')$ then

$$\text{score}(P) = 1 - \left[(1 - \text{score}(P'_1)) \cdot \ldots \cdot (1 - \text{score}(P'_k))\right]$$

assuming that $\{P'_1, \ldots, P'_k\}$ are the plans corresponding to the output-tuples of P'. For the sake of conciseness we adopt the *independent-or* ($\otimes$) operator:

$$\text{score}(P) \stackrel{\text{def}}{=} \text{score}(P'_1) \otimes \ldots \otimes \text{score}(P'_k) \stackrel{\text{def}}{=} \bigotimes_{i \in \{1,\ldots,k\}} \left[\text{score}(P'_i)\right]$$

Let's assume P' and P'' are two plans answering the Boolean queries q' and q'', respectively, and $\mathbb{P}[q'|\mathcal{D}] = \text{score}(P')$ and $\mathbb{P}[q''|\mathcal{D}] = \text{score}(P'')$. Notice that $\mathbb{P}[q' \wedge q''|\mathcal{D}] = \text{score}(P' \bowtie^{\text{p}} P'')$ holds, but only if q' and q'' represent independent events. Similar considerations apply to π^{p}: if P' and P'' are the output-tuples of the plan P, then the equivalence $\mathbb{P}[q' \vee q''|\mathcal{D}] = \text{score}(\pi^{\text{p}}_{\emptyset}(P))$ holds only if q' and q'' represent independent events. The probabilistic independence between q' and q'' is guaranteed when their lineages do not share any literal. We can conclude that an arbitrary pRA plan P computes the correct marginal probabilities when all its intermediate results consist of pairwise independent events. A plan respecting such property is said to be *"safe"* and its lineage expressions are guaranteed to be read-once. The following Lemma summarizes a variety of results about probabilistic query processing over TI-PDBs

Lemma 2.3. *[9, 11, 24, 48] Let* $\mathbf{q}$ *be a self-join-free conjunctive query consisting of* k *Boolean queries* $\{q_1, \ldots, q_k\}$*. The following statements are equivalent:*

(1) Query $\mathbf{q}$ *is* hierarchical.
(2) For any $\mathcal{D}$*, query* $\mathbf{q}$ *admits a* safe *pRA plan.*
(3) For any $\mathcal{D}$ *and* $q_j \in \mathbf{q}$*, the lineage of* q_j *admits a read-once representation.*
(4) For any $\mathcal{D}$ *and* $q_j \in \mathbf{q}$*, computing* $\mathbb{P}[q_j|\mathcal{D}]$ *takes polynomial time in the size of* $\mathcal{D}$*.*

Deciding any (all) of the above properties (finding a certificate, if any exists) takes polynomial time in the size of $\mathbf{q}$*. If such test fails (i.e. no certificate exists), then answering* $\mathbf{q}$ *is #P-hard.*

Example 2.4 (continued). We can turn the relations E and L into a TI-PDB by annotating each tuple with a probability, that we store in a dedicated column named $\boldsymbol{\theta}$.

E^{p}			
name	emp	*tid*	$\boldsymbol{\theta}$
Ada	HP	x_1	0.6
Ada	IBM	x_2	0.6
Bob	HP	x_3	0.5

L^{p}			
name	lng	*tid*	$\boldsymbol{\theta}$
Ada	eng	x_4	0.4
Bob	eng	x_5	0.2
Bob	ita	x_6	0.6

We can rewrite the plans P' and P'' in terms of pRA:

$$P' = \pi^{\text{p}}_{-X}(\pi^{\text{p}}_{-Y}(E^{\text{p}}) \bowtie^{\text{p}} L^{\text{p}}) \qquad P'' = \pi^{\text{p}}_{-XY}(E^{\text{p}} \bowtie^{\text{p}} L^{\text{p}}) \tag{6}$$

They both produce the same output-tuples, but different scores:

P'	
lng	score
eng	$((\theta_1 \otimes \theta_2)\ \theta_4) \otimes \theta_3\theta_5 = 0.4024$
ita	$\theta_3\theta_6 = 0.3$

P''	
lng	score
eng	$\theta_1\theta_4 \otimes \theta_2\theta_4 \otimes \theta_3\theta_5 = 0.48016$
ita	$\theta_3\theta_6 = 0.3$

Notice that plan P' is safe, while P'' is not: the correct value of $\mathbb{P}[q_1|\mathcal{D}]$ is 0.4024, it is *not* 0.48016.

In conclusion, pRA is guaranteed to be sound whenever we evaluate a hierarchical query. Dalvi and Suciu [9] developed a well-known algorithm to identify safe plans for hierarchical queries. Other techniques, like [7, 20, 33, 38], must be used to answer non-hierarchical queries, or more general classes of query (e.g., Conjunctive queries with Inequalities [49]) Note that, although our focus is on hierarchical queries in this article, our approach may be easily adapted to any scheme capable of efficiently computing marginal and conditional probabilities.

3 BETA PROBABILISTIC DATABASES

In this section we recap *Beta probabilistic databases* (B-PDBs) from [46]. B-PDBs are a generalization of TI-PDBs based on the idea of imposing a prior distribution over the parameters $\boldsymbol{\theta}$. In the resulting model, each parameter θ_i becomes an independent random variable, whose probability density function follows a Beta distribution $\text{Beta}(a_i, b_i)$ determined by two hyper-parameters, a_i and b_i. We use $\mathbf{a}$ and $\mathbf{b}$ to denote the corresponding n-vectors of hyper-parameters, and $\mathcal{H} \stackrel{\text{def}}{=} (\mathbf{a}, \mathbf{b})$ to denote a B-PDB instance (the relational structure of $\mathcal{H}$ is assumed to be fixed and, for the sake of conciseness, it is never mentioned explicitly). Then the probability density functions are:

$$p[\theta_i|\mathcal{H}] \stackrel{\text{def}}{=} \mathcal{B}e(a_i, b_i) \tag{7}$$

$$p[\boldsymbol{\theta}|\mathcal{H}] \stackrel{\text{def}}{=} \prod_{i=1}^{n} p[\theta_i|\mathcal{H}], \tag{8}$$

where $\mathcal{B}e(a, b)$ denotes the probability density function of a Beta distribution:

$$\mathcal{B}e(a, b) \stackrel{\text{def}}{=} \theta^{a-1} \cdot \overline{\theta}^{b-1} \cdot B(a, b)^{-1}$$

In terms of graphical models, one can see a B-PDB as a collection of n independent Boolean variables, distributed according to n independent *Beta-Bernoulli compound distributions*:

$$\theta_i \sim \text{Beta}(a_i, b_i) \qquad x_i \sim \text{Bernoulli}(\theta_i) \tag{9}$$

Given an arbitrary function $f(\theta)$ and a distribution $p[\theta]$ we denote by $\langle f \rangle_{p[\theta]}$ the expected value of f, assuming θ is sampled from $p[\theta]$. Just like TI-PDBs, B-PDBs define a probability measure over possible worlds and lineage formulas:

$$\mathbb{P}[x_i|\mathcal{H}] \stackrel{\text{def}}{=} \int_0^1 \theta_i \cdot \mathcal{B}e(a_i, b_i)\, d\theta_i \tag{10}$$

$$\mathbb{P}[w|\mathcal{H}] \stackrel{\text{def}}{=} \prod_{i=1}^{n} \mathbb{P}[x_i|\mathcal{H}]^{w[i]} \cdot \overline{\mathbb{P}[x_i|\mathcal{H}]}^{\overline{w[i]}} \tag{11}$$

$$\mathbb{P}[\varphi|\mathcal{H}] \stackrel{\text{def}}{=} \sum_{w:\ w \models \varphi} \mathbb{P}[w|\mathcal{H}] \tag{12}$$

Equations 10, 11 and 12 denote, respectively, the probability of a literal, the probability of a possible world, and the probability of a Boolean query having lineage φ. Notice that φ may also represent the lineage of any one possible answer to a non-Boolean query. For example, if we submit a non-Boolean query $\mathbf{q}$ consisting of three Boolean queries $[\varphi_1, \varphi_2, \varphi_3]$, the probability of observing the answer $[\bot, \top, \bot]$ is $\mathbb{P}[\overline{\varphi}_1\varphi_2\overline{\varphi}_3|\mathcal{H}]$.

In practical terms, B-PDBs differ from TI-PDBs in that each tuple is annotated with *two* $\mathbb{R}^+$-valued parameters, rather than with a single probability measure (compare, for example, the relation $E^{(a,b)}$ in Figure 2b with the probabilistic relation E^{p} discussed in Example 2.4). Notice that the marginal probability of x_i can be computed as [35]

$$\mathbb{P}[x_i|\mathcal{H}] = \int_0^1 \theta_i \cdot \mathcal{B}e(a_i, b_i)\, d\theta_i = \langle \theta_i \rangle_{\mathcal{H}} = \frac{a_i}{a_i + b_i} \tag{13}$$

From Equation (13) it follows immediately that the vector $\langle \boldsymbol{\theta} \rangle_{\mathcal{H}}$ of expected tuple probabilities under $\mathcal{H}$ represents the parameters of a TI-PDB that behave identically to the B-PDB $\mathcal{H}$ when it comes to query processing. In other words, the mapping $\mathcal{H} \rightarrow \langle \boldsymbol{\theta} \rangle_{\mathcal{H}}$ allows us to immediately re-use all the standard query processing techniques developed for TI-PDBs, like pRA [21], Monte Carlo

simulations [7, 33, 38], anytime approximations [13, 20], dissociations [8, 22, 23], lineage-based methods [26] and many others.

Given two queries with lineage φ and φ', we denote by $\mathbb{P}[\varphi|\varphi', \mathcal{H}]$ the probability of observing φ being true in a possible world of $\mathcal{H}$ that already satisfies φ':

$$\mathbb{P}[\varphi|\varphi', \mathcal{H}] \stackrel{\text{def}}{=} \frac{\mathbb{P}[\varphi \wedge \varphi'|\mathcal{H}]}{\mathbb{P}[\varphi'|\mathcal{H}]} \tag{14}$$

We denote by $p[\boldsymbol{\theta}|\varphi, \mathcal{H}]$ the *posterior* probability density function of $\boldsymbol{\theta}$ w.r.t. φ:

$$p[\boldsymbol{\theta}|\varphi, \mathcal{H}] \stackrel{\text{def}}{=} \frac{p[\boldsymbol{\theta}, \varphi|\mathcal{H}]}{\mathbb{P}[\varphi|\mathcal{H}]} = \frac{\mathbb{P}[\varphi|\boldsymbol{\theta}] \cdot p[\boldsymbol{\theta}|\mathcal{H}]}{\mathbb{P}[\varphi|\mathcal{H}]} \tag{15}$$

Notice that $\mathbb{P}[\varphi|\boldsymbol{\theta}]$ represents the probability of observing φ being satisfied by a TI-PDB with parameters $\boldsymbol{\theta}$.

3.1 Modeling Independent Observations from $\mathcal{H}$

The model discussed so far assumes that query-answers are delivered in a deterministic fashion: if a user states that the answer to query q (with lineage φ) should be $\top$, then the posterior probability $p[\boldsymbol{\theta}|\varphi, \mathcal{H}]$ is well defined. The same applies to the case when the answer is negative ($\bot$), and the resulting posterior is $p[\boldsymbol{\theta}|\neg\varphi, \mathcal{H}]$. But what if the user states that q is true with probability 0.8? We now lay the groundwork for handling this very special case, that is of great importance in the context of maximum likelihood estimation. Before discussing the details, we just point out that some special handling is needed, as processing ten independent deterministic query-answers, say eight yeses and two noes, is *not* equivalent to processing ten non-deterministic answers with 0.8 bias. We propose the following solution: whenever a user states that q is true with probability 0.8 we pretend that she is reporting the results of a hidden poll with deterministic answers, where each answer is independent from the others and a fraction of 0.8 of the answers were positive. We follow a similar approach when a user delivers non-deterministic feedback about multiple queries (for example: "q_1 should be true with probability 0.32, but q_2 should be true with probability 0.88"). In such case we assume the user is reporting the results of two independent hidden polls. The discussion that follows formalizes this simple intuition.

Given a B-PDB $\mathcal{H}$ and a positive integer s, we denote by $\mathcal{H}^s$ the distribution obtained by replicating the model of $\mathcal{H}$ exactly s times. In other words, $\mathcal{H}^s$ represents the distribution of a set of s possible worlds drawn independently from $\mathcal{H}$. Figure 4 depicts model $\mathcal{H}^s$ using plate notation (Figure 4b), and compares it with the model induced by TI-DBs (Figure 4a). Within a model $\mathcal{H}^s$, we denote by $\theta_{\ell,i}$ and $x_{\ell,i}$ the pairs of random variables associated with the i-th tuple of the ℓ-th possible world. Therefore

$$\theta_{\ell,i} \sim \text{Beta}(a_i, b_i) \qquad x_{\ell,i} \sim \text{Bernoulli}(\theta_{\ell,i})$$

We denote by $x_{(\cdot,i)}$ the s-vector $(x_{1,i}, \ldots, x_{s,i})$, by $x_{(\ell,\cdot)}$ the n-vector $(x_{\ell,1}, \ldots, x_{\ell,n})$ and by $x_{(\cdot,\cdot)}$ the s-by-n matrix containing all the Boolean random variables of the model. We adopt similar conventions for defining the semantics of $\theta_{(\cdot,i)}$, $\theta_{(\ell,\cdot)}$ and $\theta_{(\cdot,\cdot)}$. Given an integer t such that $0 \leq t \leq s$, we denote by $\mathbb{P}[\varphi^t|\mathcal{H}^s]$ the probability of observing a set of s independent possible worlds from $\mathcal{H}$ among which φ is satisfied exactly t times:

$$\mathbb{P}[\varphi^t|\mathcal{H}^s] \stackrel{\text{def}}{=} \binom{s}{t} \cdot \mathbb{P}[\varphi|\mathcal{H}]^t \cdot \mathbb{P}[\overline{\varphi}|\mathcal{H}]^{s-t} \tag{16}$$

The posterior probability density of $\theta_{(\cdot,\cdot)}$ w.r.t. $\mathcal{H}^s$ and evidence φ^t may be rewritten as:

$$p[\theta_{(\cdot,\cdot)}|\varphi^t,\mathcal{H}^s] = \frac{\mathbb{P}[\varphi^t|\theta_{(\cdot,\cdot)}]\cdot p[\theta_{(\cdot,\cdot)}|\mathcal{H}^s]}{\mathbb{P}[\varphi^t|\mathcal{H}^s]}$$
$$= \frac{\binom{s}{t}\prod_{\ell=1}^{t}\mathbb{P}[\varphi|\theta_{(\ell,\cdot)}]\cdot\prod_{\ell=t+1}^{s}\mathbb{P}[\overline{\varphi}|\theta_{(\ell,\cdot)}]\cdot\prod_{\ell=1}^{s}p[\theta_{(\ell,\cdot)}|\mathcal{H}]}{\mathbb{P}[\varphi^t|\mathcal{H}^s]}$$

Notice that the above formula generalizes Equation (15). Given a positive integer k, we denote by $\mathcal{H}^{s,k}$ the probability distribution obtained by replicating $\mathcal{H}^s$ exactly k times. Equivalently, $\mathcal{H}^{s,k}$ represents the distribution of a set of $s\cdot k$ possible worlds *of the database* sampled independently from $\mathcal{H}$. We extend our notation accordingly, denoting by $x_{j,\ell,i}$ and $\theta_{j,\ell,i}$ the random variables associated with the $(js+\ell)$-th possible world. Recall that we previously defined evidence as a set of queries associated with responses: $\mathcal{E}\stackrel{\text{def}}{=}\langle\{(q_1,\tau_1),(q_2,\tau_2),\ldots,(q_k,\tau_k)\},\ s\rangle$. Observe that the set of possible tuples in $\mathcal{D}$ is fixed, and consequently that the lineage of any query result is also fixed. Hence, we can model evidence instead by a set of k distinct Boolean expressions $\{\varphi_1,\ldots,\varphi_k\}$, where $\varphi_i=\Phi(q_i)$, and k integers $\{t_1,\ldots,t_k\}$ between 0 and s, where $t_i=\tau_i\cdot s$. We abuse notation and denote by $\mathcal{E}=\{\varphi_1^{t_1},\ldots,\varphi_k^{t_k}\}$ the event of observing each φ_j in $\{\varphi_1,\ldots,\varphi_k\}$ being satisfied exactly t_j over the s possible worlds $\{x_{(j,1,\cdot)},\ldots,x_{(j,s,\cdot)}\}$. Its likelihood is

$$\mathbb{P}[\mathcal{E}|\mathcal{H}^{s,k}] = \prod_{j=1}^{k}\mathbb{P}[\varphi_j^{t_j}|\mathcal{H}^s] \tag{17}$$

The posterior probability density of $\theta_{(\cdot,\cdot,\cdot)}$ w.r.t. $\mathcal{H}^{s,k}$ and evidence $\mathcal{E}=\{\varphi_1^{t_1},\ldots,\varphi_k^{t_k}\}$ is

$$p[\theta_{(\cdot,\cdot,\cdot)}|\mathcal{E},\mathcal{H}^{s,k}]\stackrel{\text{def}}{=}\prod_{j=1}^{k}p[\theta_{(j,\cdot,\cdot)}|\varphi_j^{t_j},\mathcal{H}^s]$$

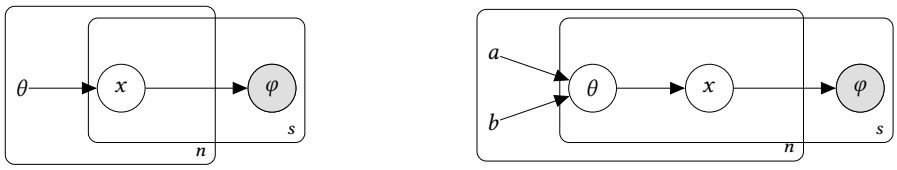
Due to the tight coupling of boolean queries and their lineage formulas, moving forward we will use φ interchangeably to refer to both.

Example 3.1. Let's assume we have a B-PDB $\mathcal{H}$ with two tuples, x_1 and x_2, and $k=2$ queries with lineage: $\varphi_1=x_1\wedge x_2$ and $\varphi_2=x_1\vee x_2$. Then:

- The probability of x_1 (resp x_2) conditioned on $\mathcal{H}$ is the expected value of its parameter
  $\mathbb{P}[x_1|\mathcal{H}]=\langle\theta_1\rangle_{\mathcal{H}}$ $\quad$ $\mathbb{P}[x_2|\mathcal{H}]=\langle\theta_2\rangle_{\mathcal{H}}$
- The probability of φ_1 (resp., φ_2) based on the probability of its parameters.
  $\mathbb{P}[\varphi_1|\mathcal{H}]=\langle\theta_1\rangle_{\mathcal{H}}\cdot\langle\theta_2\rangle_{\mathcal{H}}$ $\quad$ $\mathbb{P}[\varphi_2|\mathcal{H}]=\langle\theta_1\rangle_{\mathcal{H}}\otimes\langle\theta_2\rangle_{\mathcal{H}}$
- φ_1 subsumes φ_2, so:
  $\mathbb{P}[\varphi_1\wedge\varphi_2|\mathcal{H}]=\langle\theta_1\rangle_{\mathcal{H}}\cdot\langle\theta_2\rangle_{\mathcal{H}}$
- Also conditioning on φ_2 (unsurprisingly) means normalizing over cases where φ_2 is true.
  $\mathbb{P}[\varphi_1|\varphi_2,\mathcal{H}]=(\langle\theta_1\rangle_{\mathcal{H}}\cdot\langle\theta_2\rangle_{\mathcal{H}})\cdot(\langle\theta_1\rangle_{\mathcal{H}}\otimes\langle\theta_2\rangle_{\mathcal{H}})^{-1}$
- Evaluating φ_1 on exactly two samples of $\mathcal{H}$, φ_1 will be true exactly once if it is true in the first sample and false in the second, or visa versa.
  $\mathbb{P}[\varphi_1^1|\mathcal{H}^2]=2\cdot(\langle\theta_1\rangle_{\mathcal{H}}\cdot\langle\theta_2\rangle_{\mathcal{H}})\cdot(\langle\overline{\theta_1}\rangle_{\mathcal{H}}\otimes\langle\overline{\theta_2}\rangle_{\mathcal{H}})$
- Evaluating φ_1 and φ_2 on two independent samples (one sample per query) creates two independent events. Note the difference from $\mathbb{P}[\varphi_1\wedge\varphi_2|\mathcal{H}]$
  $\mathbb{P}[\varphi_1^1,\varphi_2^1|\mathcal{H}^{1,2}]=(\langle\theta_1\rangle_{\mathcal{H}}\cdot\langle\theta_2\rangle_{\mathcal{H}})\cdot(\langle\theta_1\rangle_{\mathcal{H}}\otimes\langle\theta_2\rangle_{\mathcal{H}})$

From Equation (13) it is straightforward to derive the following Lemma:

Lemma 3.2. *For any arbitrary evidence $\mathcal{E}=\{\varphi_1^{t_1},\ldots,\varphi_k^{t_k}\}$ the likelihood function $\mathbb{P}[\mathcal{E}|\mathcal{H}^{s,k}]$ respects the following properties:*



(a) (b)

Fig. 4. Comparison between a regular TI-PDB (a) and a B-PDB (b) when the evidence is observed against a single query φ (hence $k = 1$), using plate notation. TI-PDBs associate each tuple $i \in [n]$ with a single Boolean, Bernoulli-distributed random variable x_i, whose probability mass function depends on a single parameter θ_i. In contrast, B-PDBs model θ_i not as a parameter but as a random variable that is $[0, 1]$-valued and Beta-distributed with hyper-parameters (a_i, b_i). In other words, B-PDBs associate each tuple with *two* random variables: θ_i and x_i. For both TI-PDBs and B-PDBs the evidence consists of observed query answers, that here are modeled by the observable variable φ.

(1) If $\mathcal{H}_a$ and $\mathcal{H}_b$ are two B-PDBs such that $\langle\boldsymbol{\theta}\rangle_{\mathcal{H}_a} = \langle\boldsymbol{\theta}\rangle_{\mathcal{H}_b}$ then $\mathbb{P}[\mathcal{E}|\mathcal{H}_a] = \mathbb{P}[\mathcal{E}|\mathcal{H}_b]$.
(2) Let $\mathcal{H}$ be a B-PDB and $\mathcal{D}$ a TI-PDB with parameters $\boldsymbol{\theta}^*$: if $\boldsymbol{\theta}^* = \langle\boldsymbol{\theta}\rangle_{\mathcal{H}}$ then $\mathbb{P}[\mathcal{E}|\mathcal{H}^{s,k}] = \mathbb{P}[\mathcal{E}|\mathcal{D}^{s,k}]$, where $\mathcal{D}^{s,k}$ denotes the distribution obtained by drawing $s \cdot k$ independent samples from $\mathcal{D}$.

Since model $\mathcal{H}^{s,k}$ generalizes $\mathcal{H}$ (as $\mathcal{H} = \mathcal{H}^{1,1}$), in the following we will often use the former as a placeholder for the latter. The same applies to the associated notations ($\theta_{(\cdot,\cdot,\cdot)}$, $\varphi_j^{t_j}$). The reader should nonetheless keep in mind that updating $\mathcal{H}^{s,k}$ once is not equivalent to updating $\mathcal{H}$ for $s \cdot k$ times.

The next two sections are dedicated to two specific operations supported by B-PDBs, that involve the computation of the posterior $p[\theta_{(\cdot,\cdot,\cdot)}|\mathcal{E}, \mathcal{H}^{s,k}]$: *belief updating* and *maximum likelihood estimation*. We introduce their formal definition here:

Definition 3.3 (Belief updating). Given a B-PDB $\mathcal{H}$ and an evidence event $\mathcal{E} = \{\varphi_1^{t_1}, \ldots, \varphi_k^{t_k}\}$, belief updating is the process of identifying the B-PDB $\hat{\mathcal{H}}$ that minimizes the relative entropy between the posterior $p[\theta_{(\cdot,\cdot,\cdot)}|\mathcal{E}, \mathcal{H}^{s,k}]$ and the new prior $p[\theta_{(\cdot,\cdot,\cdot)}|\hat{\mathcal{H}}^{s,k}]$. Belief updating is discussed in Section 4.

As we will see, new evidence can create dependencies that can not be represented with a B-PDB. We are looking for an new B-PDB whose implied possible worlds are closest the exact posterior.

Definition 3.4 (Maximum likelihood estimation). Given some evidence $\mathcal{E} = \{\varphi_1^{t_1}, \ldots, \varphi_k^{t_k}\}$, maximum likelihood estimation is the problem of identifying a local maximum of the likelihood function $\mathbb{P}[\mathcal{E}|\mathcal{H}^{s,k}]$. Maximum likelihood estimation is discussed in Section 5.

Note that this likelihood is unaffected if we scale the hyper-parameter vectors $(\mathbf{a}, \mathbf{b})$. That is, for any vector $\boldsymbol{\mu}$ of positive reals, $\mathbb{P}[\mathcal{E}|(\mathbf{a}, \mathbf{b})] = \mathbb{P}[\mathcal{E}|(\boldsymbol{\mu}\mathbf{a}^T, \boldsymbol{\mu}\mathbf{b}^T)]$. Hence the output of maximum likelihood estimation is a TI-PDB given by the probability vector $\boldsymbol{\theta}$, and not the supporting hyper-parameters. Rather, the hyper-parameters fill a role analogous to temperature in simulated annealing, stabilizing probabilities as they are supported by more evidence.

4 BELIEF UPDATING

The goal of belief updating is to adjust the parameters $\mathbf{a}$ and $\mathbf{b}$ as to incorporate some new, previously unseen, evidence. Following the notation introduced in the previous section, we begin with a simple

case of belief updates where the evidence consists of a deterministic query-answer ($s = 1$) regarding a single boolean query ($k = 1$). We then parallelize this approach in two dimensions: We allow a single belief update to simultaneously learn from noisy feedback ($s > 1$) about multiple Boolean queries ($k > 1$). Specifically, if $\mathcal{H}$ denotes the current state of our B-PDB and if $\mathcal{E} = \{\varphi_1^{t_1}, \ldots, \varphi_k^{t_k}\}$ is the new evidence we observe, our goal is to identify a new state $\hat{\mathcal{H}} \stackrel{\text{def}}{=} (\hat{\mathbf{a}}, \hat{\mathbf{b}})$ that minimizes the relative entropy between $p[\theta_{(\cdot,\cdot,\cdot)}|\mathcal{E}, \mathcal{H}^{s,k}]$ and $p[\theta_{(\cdot,\cdot,\cdot)}|\hat{\mathcal{H}}]$.

4.1 Simple case: $s = k = 1$

When $s = k = 1$ holds and $\mathcal{E} = \{\varphi\}$, $\hat{\mathcal{H}}$ is supposed to minimize the relative entropy between the exact posterior distribution $p[\boldsymbol{\theta}|\varphi, \mathcal{H}]$ and the new state of the database $p[\boldsymbol{\theta}|\hat{\mathcal{H}}]$. Since every tuple of $\hat{\mathcal{H}}$ is independent from the others, it is sufficient to minimize the relative entropy between the marginal distributions $p[\theta_i|\varphi, \mathcal{H}]$ and $p[\theta_i|\hat{\mathcal{H}}]$ for each and every tuple. To accomplish this, we need to first derive the value of the integral $p[\theta_i|\varphi, \mathcal{H}] = \int_0^1 .. \int_0^1 p[\boldsymbol{\theta}|\varphi, \mathcal{H}] d\theta_1 .. d\theta_{i-1} d\theta_{i+1} .. d\theta_n$. The following theorem does so.

Theorem 4.1. *The marginal posterior probability of random variable θ_i, given hypothesis $\mathcal{H}$ and evidence $\mathcal{E} = \{\varphi\}$, can be computed as follows:*

$$p[\theta_i|\varphi, \mathcal{H}] = \mathbb{P}[x_i|\varphi, \mathcal{H}] \cdot \mathcal{B}e(a_i + 1, b_i) + \mathbb{P}[\overline{x_i}|\varphi, \mathcal{H}] \cdot \mathcal{B}e(a_i, b_i + 1)$$

Proof. The proof is given in Appendix B. □

Theorem 4.1 states that $p[\theta_i|\varphi, \mathcal{H}]$ is a mixture of Beta distributions. It is well known that the Beta distribution is the conjugate prior of the Bernoulli distribution: if $\theta_i \sim \text{Beta}(a_i, b_i)$ and $x_i \sim \text{Bernoulli}(\theta_i)$ then the posterior of θ_i is $\mathcal{B}e(a_i + 1, b_i)$ when x_i is observed to be true, and $\mathcal{B}e(a_i, b_i + 1)$ if x_i is observed to be false. Within a B-PDB we can exploit such property whenever we can infer the value of some random variable x_i from the evidence; this is formalized by the following Corollary:

Corollary 4.2. *In a B-PDB $\mathcal{H}$ the marginal posterior of θ_i respects the following property:*

$$p[\theta_i|\varphi, \mathcal{H}] = \begin{cases} \mathcal{B}e(a_i + 1, b_i) & \text{if and only if } \varphi \models x_i \\ \mathcal{B}e(a_i, b_i + 1) & \text{if and only if } \varphi \models \overline{x_i} \end{cases}$$

Theorem 4.1 suggests a very intuitive interpretation of the marginal posterior $p[\theta_i|\varphi, \mathcal{H}]$: it can be seen as a random process in which we first make a guess about the value taken by x_i in the evidence and then we select the appropriate conjugate prior. Notice that the complexity of computing $p[\theta_i|\varphi, \mathcal{H}]$ depends on the complexity of computing conditional probabilities in the form $\mathbb{P}[x|\varphi, \mathcal{H}]$. This problem is discussed extensively in Section 6; for the moment we just observe that it is $\#\mathcal{P}$-hard in the general case.

Now that we know how to compute the marginal posterior $p[\theta_i|\varphi, \mathcal{H}]$, we can move our attention to the problem of computing an update $\mathcal{H} \rightarrow \hat{\mathcal{H}}$ that minimizes the relative entropy between $p[\theta_i|\varphi, \mathcal{H}]$ and the i-th prior $p[\theta_i|\hat{\mathcal{H}}] \stackrel{\text{def}}{=} \mathcal{B}e(\hat{a}_i, \hat{b}_i)$. We denote such measure, also known as *Kullback-Leibler* divergence, with KL_i^{div}:

$$\text{KL}_i^{\text{div}} = \int_0^1 p[\theta_i|\varphi, \mathcal{H}] \cdot \ln\left(\frac{p[\theta_i|\varphi, \mathcal{H}]}{p[\theta_i|\hat{\mathcal{H}}]}\right) d\theta_i$$

Notice that $p[\theta_i|\hat{\mathcal{H}}]$ belongs to the exponential family, and its sufficient statistics are $\ln \theta_i$ and $\ln \overline{\theta_i}$. Therefore, if we want to minimize the relative entropy KL_{div} we have to choose $(\hat{a}_i, \hat{b}_i)$ so that

the expected value of $(\ln\theta_i, \ln\overline{\theta}_i)$ w.r.t. $\mathbb{P}[\theta_i|\hat{\mathcal{H}}]$ matches the expected value of the same statistics computed w.r.t. probability mass function $p[\theta|\varphi,\mathcal{H}]$. This well-known [31] criterion is formalized in the following definition and justified in Prop. 4.4[9]:

Definition 4.3. We denote by $\text{bfit}(a_i, b_i, \varphi)$ the pair of parameters $(\hat{a}_i^*, \hat{b}_i^*)$ satisfying the following two equations:

$$\begin{cases} \int_0^1 \mathcal{B}e(\hat{a}_i^*, \hat{b}_i^*) \ln\theta_i \; d\theta_i = \int_0^1 p[\theta_i|\varphi,\mathcal{H}] \ln\theta_i \; d\theta_i \\ \int_0^1 \mathcal{B}e(\hat{a}_i^*, \hat{b}_i^*) \ln\overline{\theta_i} \; d\theta_i = \int_0^1 p[\theta_i|\varphi,\mathcal{H}] \ln\overline{\theta_i} \; d\theta_i \end{cases} \tag{18}$$

or, in terms of expectations:

$$\begin{cases} \langle \ln\theta_i \rangle_{\mathcal{B}e(\hat{a}_i^*, \hat{b}_i^*)} = \langle \ln\theta_i \rangle_{p[\theta_i|\varphi,\mathcal{H}]} \\ \langle \ln\overline{\theta}_i \rangle_{\mathcal{B}e(\hat{a}_i^*, \hat{b}_i^*)} = \langle \ln\overline{\theta}_i \rangle_{p[\theta_i|\varphi,\mathcal{H}]} \end{cases} \tag{19}$$

Proposition 4.4. *When $(\hat{a}_i, \hat{b}_i) = (\hat{a}_i^*, \hat{b}_i^*) = bfit(a_i, b_i, \varphi)$, the relative entropy between the posterior $p[\theta_i|\varphi,\mathcal{H}]$ and the Beta distribution $p[\theta_i|\hat{\mathcal{H}}]$ is minimized.*

Proof. The relative entropy between $p[\theta_i|\varphi,\mathcal{H}]$ and $p[\theta_i|\hat{\mathcal{H}}]$ can be expressed as follows:

$$\begin{aligned} \mathsf{KL}_i^{\text{div}} &= \int_0^1 p[\theta_i|\varphi,\mathcal{H}] \cdot \ln\left(\frac{p[\theta_i|\varphi,\mathcal{H}]}{p[\theta_i|\hat{\mathcal{H}}]}\right) d\theta_i \\ &= h\left[p[\theta_i|\varphi,\mathcal{H}]\right] - \int_0^1 p[\theta_i|\varphi,\mathcal{H}] \cdot \ln\left(p[\theta_i|\hat{\mathcal{H}}]\right) d\theta_i \\ &= h\left[p[\theta_i|\varphi,\mathcal{H}]\right] + \ln(B(\hat{a}_i,\hat{b}_i)) - \int_0^1 p[\theta_i|\varphi,\mathcal{H}] \cdot \ln\left(\theta_i^{\hat{a}_i-1} \cdot \overline{\theta}_i^{\hat{b}_i-1}\right) d\theta_i \\ &= h\left[p[\theta_i|\varphi,\mathcal{H}]\right] + \ln(B(\hat{a}_i,\hat{b}_i)) - \int_0^1 p[\theta_i|\varphi,\mathcal{H}] \cdot (\hat{a}_i-1) \cdot \ln\theta_i \; d\theta_i \\ &\quad - \int_0^1 p[\theta_i|\varphi,\mathcal{H}] \cdot (\hat{b}_i-1) \cdot \ln\overline{\theta_i} \; d\theta_i \\ &= h\left[p[\theta_i|\varphi,\mathcal{H}]\right] + \ln(B(\hat{a}_i,\hat{b}_i)) - (\hat{a}_i-1) \cdot \langle \ln\theta_i \rangle_{p[\theta_i|\varphi,\mathcal{H}]} - (\hat{b}_i-1) \cdot \langle \ln\overline{\theta_i} \rangle_{p[\theta_i|\varphi,\mathcal{H}]} \end{aligned}$$

The gradient of $\mathsf{KL}_i^{\text{div}}$ w.r.t. $(\hat{a}_i, \hat{b}_i)$ is:

$$\begin{aligned} \nabla\mathsf{KL}_i^{\text{div}} &= \begin{bmatrix} \frac{\partial \mathsf{KL}_i^{\text{div}}}{\partial \hat{a}_i} \\ \\ \frac{\partial \mathsf{KL}_i^{\text{div}}}{\partial \hat{b}_i} \end{bmatrix} = \begin{bmatrix} [\frac{\partial B(\hat{a}_i,\hat{b}_i)}{\partial \hat{a}_i} \cdot B(\hat{a}_i,\hat{b}_i)^{-1}] - \langle \ln\theta_i \rangle_{p[\theta_i|\varphi,\mathcal{H}]} \\ \\ [\frac{\partial B(\hat{a}_i,\hat{b}_i)}{\partial \hat{b}_i} \cdot B(\hat{a}_i,\hat{b}_i)^{-1}] - \langle \ln\overline{\theta_i} \rangle_{p[\theta_i|\varphi,\mathcal{H}]} \end{bmatrix} \\ &= \begin{bmatrix} [\psi(\hat{a}_i) - \psi(\hat{a}_i+\hat{b}_i)] - \langle \ln\theta_i \rangle_{p[\theta_i|\varphi,\mathcal{H}]} \\ \\ [\psi(\hat{b}_i) - \psi(\hat{a}_i+\hat{b}_i)] - \langle \ln\overline{\theta_i} \rangle_{p[\theta_i|\varphi,\mathcal{H}]} \end{bmatrix} \\ &= \begin{bmatrix} \langle \ln\theta_i \rangle_{p[\theta_i|\hat{\mathcal{H}}]} - \langle \ln\theta_i \rangle_{p[\theta_i|\varphi,\mathcal{H}]} \\ \\ \langle \ln\overline{\theta_i} \rangle_{p[\theta_i|\hat{\mathcal{H}}]} - \langle \ln\overline{\theta_i} \rangle_{p[\theta_i|\varphi,\mathcal{H}]} \end{bmatrix} \end{aligned}$$

[9] A similar, more general result is proved in [31] for all the distributions in the exponential family.

The Hessian of $\mathsf{KL}_i^{\text{div}}$ w.r.t. $(\hat{a}_i, \hat{b}_i)$ is positive semidefinite:

$$\text{Hess}[\mathsf{KL}_i^{\text{div}}] = \nabla\nabla \mathsf{KL}_i^{\text{div}} \begin{bmatrix} \psi'(\hat{a}_i) - \psi'(\hat{a}_i + \hat{b}_i) & -\psi'(\hat{a}_i + \hat{b}_i) \\ -\psi'(\hat{a}_i + \hat{b}_i) & \psi'(\hat{b}_i) - \psi'(\hat{a}_i + \hat{b}_i) \end{bmatrix}$$

Where $\psi'(\cdot)$ denotes the Trigamma function. The thesis follows immediately. □

Finally, we observe that the relative entropy between $p[\boldsymbol{\theta}|\varphi, \mathcal{H}]$ and $p[\boldsymbol{\theta}|\hat{\mathcal{H}}]$, that we denote with KL^{div}, is minimized when $\hat{\mathcal{H}} = \text{argmin} \sum_{i=1}^{n} \mathsf{KL}_i^{\text{div}}$:

$$\begin{aligned} \mathsf{KL}^{\text{div}} &= \int p[\boldsymbol{\theta}|\varphi, \mathcal{H}] \cdot \ln\left(\frac{p[\boldsymbol{\theta}|\varphi, \mathcal{H}]}{p[\boldsymbol{\theta}|\hat{\mathcal{H}}]}\right) d\boldsymbol{\theta} \\ &= h\left[p[\boldsymbol{\theta}|\varphi, \mathcal{H}]\right] - \int p[\boldsymbol{\theta}|\varphi, \mathcal{H}] \cdot \ln\left(p[\boldsymbol{\theta}|\hat{\mathcal{H}}]\right) d\boldsymbol{\theta} \\ &= h\left[p[\boldsymbol{\theta}|\varphi, \mathcal{H}]\right] - \sum_{i=1}^{n} \int p[\boldsymbol{\theta}|\varphi, \mathcal{H}] \cdot \ln\left(p[\theta_i|\hat{\mathcal{H}}]\right) d\boldsymbol{\theta} \\ &= h\left[p[\boldsymbol{\theta}|\varphi, \mathcal{H}]\right] - \sum_{i=1}^{n} \int_0^1 p[\theta_i|\varphi, \mathcal{H}] \cdot \ln\left(p[\theta_i|\hat{\mathcal{H}}]\right) d\theta_i \end{aligned}$$

4.2 Intermediate case: $k = 1, s > 1$

Next we address the case where $k = 1$, $s > 1$ and $\mathcal{E} = \{\varphi^t\}$. Under these assumptions the goal of belief updating is to minimize the KL divergence between $p[\theta_{(\cdot,\cdot)}|\varphi^t, \mathcal{H}^s]$ and $p[\theta_{(\cdot,\cdot)}|\hat{\mathcal{H}}^s]$. Since the tuples of $\hat{\mathcal{H}}$ are pairwise independent, it is sufficient to minimize the relative entropy between $p[\theta_{(\cdot,i)}|\varphi^t, \mathcal{H}^s]$ and $p[\theta_{(\cdot,i)}|\hat{\mathcal{H}}^s]$ for every i in $\{1, \ldots, n\}$. As usual $\theta_{(\cdot,i)}$ denotes the vector $(\theta_{1,i}, \ldots, \theta_{s,i})$. Its probability density w.r.t. $\hat{\mathcal{H}}^s$ is

$$p[\theta_{(\cdot,i)}|\hat{\mathcal{H}}^s] = \prod_{\ell=1}^{s} p[\theta_{\ell,i}|\hat{\mathcal{H}}] \tag{20}$$

while its posterior density w.r.t. $\mathcal{H}^s$ and evidence $\{\varphi^t\}$ is

$$p[\theta_{(\cdot,i)}|\varphi^t, \mathcal{H}^s] = \prod_{\ell=1}^{t} p[\theta_{\ell,i}|\varphi, \mathcal{H}] \cdot \prod_{\ell=t+1}^{s} p[\theta_{\ell,i}|\overline{\varphi}, \mathcal{H}] \tag{21}$$

where $p[\theta_{\ell,i}|\varphi, \mathcal{H}]$ and $p[\theta_{\ell,i}|\overline{\varphi}, \mathcal{H}]$ are computed according to Theorem 4.1. We now redefine $\mathsf{KL}_i^{\text{div}}$ as the relative entropy between $p[\theta_{(\cdot,i)}|\varphi^t, \mathcal{H}^s]$ and $p[\theta_{(\cdot,i)}|\hat{\mathcal{H}}^s]$:

$$\mathsf{KL}_i^{\text{div}} = \int_0^1 .. \int_0^1 p[\theta_{(\cdot,i)}|\varphi^t, \mathcal{H}^s] \ln\left(\frac{p[\theta_{(\cdot,i)}|\varphi^t, \mathcal{H}^s]}{p[\theta_{(\cdot,i)}|\hat{\mathcal{H}}^s]}\right) d\theta_{(\cdot,i)}$$

Definition 4.5. We denote by $\text{bfit}(a_i, b_i, \{\varphi^t\}, s)$ the pair of parameters $(\hat{a}_i^*, \hat{b}_i^*)$ satisfying the following two equations:

$$\begin{cases} \langle \ln \theta_i \rangle_{\mathcal{B}e(\hat{a}_i^*, \hat{b}_i^*)} = \frac{t}{s} \langle \ln \theta_i \rangle_{p[\theta_i|\varphi, \mathcal{H}]} + \frac{s-t}{s} \langle \ln \theta_i \rangle_{p[\theta_i|\overline{\varphi}, \mathcal{H}]} \\ \langle \ln \overline{\theta}_i \rangle_{\mathcal{B}e(\hat{a}_i^*, \hat{b}_i^*)} = \frac{t}{s} \langle \ln \overline{\theta}_i \rangle_{p[\theta_i|\varphi, \mathcal{H}]} + \frac{s-t}{s} \langle \ln \overline{\theta}_i \rangle_{p[\theta_i|\overline{\varphi}, \mathcal{H}]} \end{cases}$$

Proposition 4.6. *When $(\hat{a}_i, \hat{b}_i) = (\hat{a}_i^*, \hat{b}_i^*) = \text{bfit}(a_i, b_i, \{\varphi^t\}, s)$, the relative entropy between $p[\theta_{(\cdot,i)}|\varphi^t, \mathcal{H}^s]$ and $p[\theta_{(\cdot,i)}|\hat{\mathcal{H}}^s]$ is minimized.*

Proof.

$$\begin{aligned}
\mathsf{KL}_i^{\mathsf{div}} &= \int_0^1 .. \int_0^1 p[\theta_{(\cdot,i)}|\varphi^t, \mathcal{H}^s] \ln\left(\frac{p[\theta_{(\cdot,i)}|\varphi^t, \mathcal{H}^s]}{p[\theta_{(\cdot,i)}|\hat{\mathcal{H}}^s]}\right) d\theta_{(\cdot,i)} \\
&= \sum_{\ell=1}^{t}\left[\int_0^1 p[\theta_{\ell,i}|\varphi, \mathcal{H}] \ln\left(\frac{p[\theta_{\ell,i}|\varphi, \mathcal{H}]}{p[\theta_{\ell,i}|\mathcal{H}]}\right) d\theta_{\ell,i}\right] \\
&\quad + \sum_{\ell=t+1}^{s}\left[\int_0^1 p[\theta_{\ell,i}|\overline{\varphi}, \mathcal{H}] \ln\left(\frac{p[\theta_{\ell,i}|\overline{\varphi}, \mathcal{H}]}{p[\theta_{\ell,i}|\mathcal{H}]}\right) d\theta_{\ell,i}\right] \\
&= h\left[p[\theta_{(\cdot,i)}|\varphi^t, \mathcal{H}^s]\right] + s \cdot \ln(B(\hat{a}_i, \hat{b}_i)) \\
&\quad - (\hat{a}_i - 1) \cdot \left[t \cdot \langle \ln \theta_i \rangle_{p[\theta_i|\varphi,\mathcal{H}]} + (s-t)\langle \ln \theta_i \rangle_{p[\theta_i|\overline{\varphi},\mathcal{H}]}\right] \\
&\quad - (\hat{b}_i - 1) \cdot \left[t \cdot \langle \ln \overline{\theta}_i \rangle_{p[\theta_i|\varphi,\mathcal{H}]} + (s-t)\langle \ln \overline{\theta}_i \rangle_{p[\theta_i|\overline{\varphi},\mathcal{H}]}\right]
\end{aligned}$$

The gradient $\nabla \mathsf{KL}_i^{\mathsf{div}}$ is zero when

$$\begin{cases}
\langle \ln \theta_i \rangle_{p[\theta_i|\hat{\mathcal{H}}]} = \frac{t}{s}\langle \ln \theta_i \rangle_{p[\theta_i|\varphi,\mathcal{H}]} + \frac{s-t}{s}\langle \ln \theta_i \rangle_{p[\theta_i|\overline{\varphi},\mathcal{H}]} \\
\langle \ln \overline{\theta}_i \rangle_{p[\theta_i|\hat{\mathcal{H}}]} = \frac{t}{s}\langle \ln \overline{\theta}_i \rangle_{p[\theta_i|\varphi,\mathcal{H}]} + \frac{s-t}{s}\langle \ln \overline{\theta}_i \rangle_{p[\theta_i|\overline{\varphi},\mathcal{H}]}
\end{cases}$$

□

4.3 General case: $k > 1, s > 1$

We can finally address the general case where $k > 1$, $s > 1$ and $\mathcal{E} = \{\varphi_1^{t_1}, \ldots, \varphi_k^{t_k}\}$. Under these assumptions the goal of belief updating is to minimize the KL divergence between $p[\theta_{(\cdot,\cdot,\cdot)}|\mathcal{E}, \mathcal{H}^{s,k}]$ and $p[\theta_{(\cdot,\cdot,\cdot)}|\hat{\mathcal{H}}^{s,k}]$, and is achieved by minimizing the KL divergence between $p[\theta_{(\cdot,\cdot,i)}|\mathcal{E}, \mathcal{H}^{s,k}]$ and $p[\theta_{(\cdot,\cdot,i)}|\hat{\mathcal{H}}^{s,k}]$, for every i in $\{1, \ldots, n\}$, where

$$p[\theta_{(\cdot,\cdot,i)}|\hat{\mathcal{H}}^{s,k}] = \prod_{j=1}^{k} p[\theta_{(j,\cdot,i)}|\hat{\mathcal{H}}^{s}] \tag{22}$$

$$p[\theta_{(\cdot,\cdot,i)}|\mathcal{E}, \mathcal{H}^{s,k}] = \prod_{j=1}^{k} p[\theta_{(j,\cdot,i)}|\varphi_j^{t_j}, \mathcal{H}^{s}] \tag{23}$$

Therefore, we can redefine $\mathsf{KL}_i^{\mathsf{div}}$ as the relative entropy between $p[\theta_{(\cdot,\cdot,i)}|\mathcal{E}, \mathcal{H}^{s,k}]$ and $p[\theta_{(\cdot,\cdot,i)}|\hat{\mathcal{H}}^{s,k}]$:

$$\mathsf{KL}_i^{\mathsf{div}} = \int_0^1 .. \int_0^1 p[\theta_{(\cdot,\cdot,i)}|\mathcal{E}, \mathcal{H}^{s,k}] \ln\left(\frac{p[\theta_{(\cdot,\cdot,i)}|\mathcal{E}, \mathcal{H}^{s,k}]}{p[\theta_{(\cdot,\cdot,i)}|\hat{\mathcal{H}}^{s,k}]}\right) d\theta_{(\cdot,\cdot,i)}$$

Definition 4.7. We denote by $\mathsf{bfit}(a_i, b_i, \mathcal{E}, s, k)$ the pair of parameters $(\hat{a}_i^*, \hat{b}_i^*)$ satisfying the following two equations:

$$\begin{cases}
\langle \ln \theta_i \rangle_{\hat{\mathcal{H}}^*} = \sum_j \frac{t_j}{ks}\langle \ln \theta_i \rangle_{p[\theta_i|\varphi_j,\mathcal{H}]} + \frac{s-t_j}{ks}\langle \ln \theta_i \rangle_{p[\theta_i|\overline{\varphi}_j,\mathcal{H}]} \\
\langle \ln \overline{\theta}_i \rangle_{\hat{\mathcal{H}}^*} = \sum_j \frac{t_j}{ks}\langle \ln \overline{\theta}_i \rangle_{p[\theta_i|\varphi_j,\mathcal{H}]} + \frac{s-t_j}{ks}\langle \ln \overline{\theta}_i \rangle_{p[\theta_i|\overline{\varphi}_j,\mathcal{H}]}
\end{cases}$$

Proposition 4.8. *When $(\hat{a}_i, \hat{b}_i) = \mathsf{bfit}(a_i, b_i, \mathcal{E}, s, k)$ the relative entropy between $p[\theta_{(\cdot,\cdot,i)}|\mathcal{E}, \mathcal{H}^{s,k}]$ and $p[\theta_{(\cdot,\cdot,i)}|\hat{\mathcal{H}}^{s,k}]$ is minimized.*

The proof of Prop. 4.8 mimics the one of Prop. 4.6. For the sake of conciseness we omit it. Before introducing our belief update algorithm, we observe that the equations from Definition 4.7 can be rewritten as follows:

$$\begin{cases} [\psi(\hat{a}_i^*) - \psi(\hat{a}_i^* + \hat{b}_i^*)] - [\psi(a_i) - \psi(a_i + b_i)] = \frac{r}{a_i} - \frac{1}{a_i + b_i} \\ [\psi(\hat{b}_i^*) - \psi(\hat{a}_i^* + \hat{b}_i^*)] - [\psi(b_i) - \psi(a_i + b_i)] = \frac{\overline{r}}{b_i} - \frac{1}{a_i + b_i} \end{cases} \tag{24}$$

where $r = \frac{1}{k} \sum_{j=1}^{k} \left[\frac{t_j}{s} \mathbb{P}[x_i | \varphi_j, \mathcal{H}] + \frac{s - t_j}{s} \mathbb{P}[x_i | \overline{\varphi}_j, \mathcal{H}] \right]$ and $\psi(\cdot)$ denotes the Digamma function. Notice that r is simply the evidence-conditioned probability of literal x_i averaged over $s \cdot k$ evidence samples. From now on we denote by $\text{bu}(a_i, b_i, r)$ a procedure that computes the values $(\hat{a}^*, \hat{b}^*)$ that satisfy Equation (24) for a given hypothesis $\mathcal{H}^{s,k}$ and evidence $\mathcal{E}$. We finally introduce Algorithm 1, that exploits Prop. 4.4, 4.6 and 4.8 to perform a belief update in response to arbitrary evidence $\mathcal{E}$. Notice that Algorithm 1 shows how to perform belief updates for three different, semantically distinct models: $\mathcal{H}$, $\mathcal{H}^s$ and $\mathcal{H}^{s,k}$. Choosing one model over the others means choosing a specific way to model the evidence, as either a query-answer ($\mathcal{H}$), a query-marginal ($\mathcal{H}^s$) or a set of queries with marginal probabilities ($\mathcal{H}^{s,k}$). Interestingly, Algorithm 1 allows us to update a B-PDB in an *incremental* fashion: if the evidence is provided as a stream of query-answers, dynamically changing over time, a B-PDB can incorporate such information by performing a new belief update every time a new chunk of evidence becomes available. The idea of performing repeated Bayesian updates is discussed in detail in the next section.

Algorithm 1: Belief Update

Data: Model $\mathcal{H}$, evidence $\mathcal{E} = \{\varphi_1^{t_1}, \ldots, \varphi_k^{t_k}\}$

for $j \in \{1, \ldots, k\}$ **do**
  $\tau_j \leftarrow t_j / s$;
for $i \in \{1, \ldots, n\}$ **do**
  $r_i \leftarrow \sum_{j=1}^{k} \frac{1}{k} (\tau_j \mathbb{P}[x_i | \varphi_j, \mathcal{H}] + \overline{\tau_j} \mathbb{P}[x_i | \overline{\varphi_j}, \mathcal{H}])$
  $(\hat{a}_i^*, \hat{b}_i^*) \leftarrow \text{bu}(a_i, b_i, r_i)$
for $i \in \{1, \ldots, n\}$ **do**
  $a_i \leftarrow \hat{a}_i^*$
  $b_i \leftarrow \hat{b}_i^*$

5 PARAMETER LEARNING (MLE)

In this section we show how to exploit our belief update procedures to identify a local maximum of the likelihood function $\mathbb{P}[\mathcal{E} | \mathcal{H}^{s,k}]$. Our approach relies on the simple observation that a belief update triggered by evidence $\mathcal{E}$ can only increase the likelihood of observing $\mathcal{E}$ again; it is immediate to derive a soft-EM [12, 30] algorithm that performs repeated belief updates until convergence.

First we introduce the concept of *posterior predictive* distribution. Let's assume we have observed evidence $\mathcal{E}$ and devised the corresponding posterior distribution $p[\theta_{(\cdot,\cdot,\cdot)} | \mathcal{E}, \mathcal{H}^{s,k}]$. Rather than deriving a new hypothesis $\hat{\mathcal{H}}$ that minimizes the KL divergence against it, we instead use the posterior itself to compute the probability of observing some new evidence $\mathcal{E}'$ in the future. Such probability is given by

$$\int_0^1 .. \int_0^1 \mathbb{P}[\mathcal{E}' \mid \theta_{(\cdot,\cdot,\cdot)}] \cdot p[\theta_{(\cdot,\cdot,\cdot)} \mid \mathcal{E}, \mathcal{H}^{s,k}] \, d\theta_{(\cdot,\cdot,\cdot)} \tag{25}$$

In the literature the resulting distribution is called posterior predictive. Our next goal is to show that the posterior predictive assigns to the event of observing $\mathcal{E}$ (again) a probability that is larger or equal to the one assigned by the original hypothesis $\mathcal{H}$. In other words, we want to prove the following proposition

Proposition 5.1. *For every evidence $\mathcal{E}$ and model $\mathcal{H}^{s,k}$ the following holds*

$$\int_0^1 .. \int_0^1 \mathbb{P}[\mathcal{E} \mid \theta_{(\cdot,\cdot,\cdot)}] \cdot p[\theta_{(\cdot,\cdot,\cdot)} \mid \mathcal{E}, \mathcal{H}^{s,k}] \, d\theta_{(\cdot,\cdot,\cdot)} \geq \mathbb{P}[\mathcal{E} \mid \mathcal{H}^{s,k}] \tag{26}$$

Proof. The inequality in eq. (26) can be rewritten as

$$\int_0^1 .. \int_0^1 \mathbb{P}[\mathcal{E} \mid \theta_{(\cdot,\cdot,\cdot)}] \cdot \frac{\mathbb{P}[\mathcal{E} \mid \theta_{(\cdot,\cdot,\cdot)}] \cdot p[\theta_{(\cdot,\cdot,\cdot)} \mid \mathcal{H}^{s,k}]}{\mathbb{P}[\mathcal{E} \mid \mathcal{H}^{s,k}]} \, d\theta_{(\cdot,\cdot,\cdot)} \geq \mathbb{P}[\mathcal{E} \mid \mathcal{H}^{s,k}] \tag{27}$$

Since $\mathbb{P}[\mathcal{E} \mid \mathcal{H}^{s,k}]$ is positive

$$\int_0^1 .. \int_0^1 \mathbb{P}[\mathcal{E} \mid \theta_{(\cdot,\cdot,\cdot)}]^2 \cdot p[\theta_{(\cdot,\cdot,\cdot)} \mid \mathcal{H}^{s,k}] \, d\theta_{(\cdot,\cdot,\cdot)} \geq \left[\mathbb{P}[\mathcal{E} \mid \mathcal{H}^{s,k}]\right]^2 \tag{28}$$

The same can be expressed in terms of expectations

$$\langle \mathbb{P}[\mathcal{E} \mid \theta_{(\cdot,\cdot,\cdot)}]^2 \rangle_{p[\theta_{(\cdot,\cdot,\cdot)} \mid \mathcal{H}^{s,k}]} \geq \left[\langle \mathbb{P}[\mathcal{E} \mid \theta_{(\cdot,\cdot,\cdot)}] \rangle_{p[\theta_{(\cdot,\cdot,\cdot)} \mid \mathcal{H}^{s,k}]}\right]^2 \tag{29}$$

The thesis is proved by making two observations:

$$\langle \mathbb{P}[\mathcal{E} \mid \theta_{(\cdot,\cdot,\cdot)}]^2 \rangle_{p[\theta_{(\cdot,\cdot,\cdot)} \mid \mathcal{H}^{s,k}]} \geq \int_0^1 .. \int_0^1 \mathbb{P}[\mathcal{E} \mid \theta_{(\cdot,\cdot,\cdot)}]^2 \cdot p[\theta_{(\cdot,\cdot,\cdot)} \mid \mathcal{H}^{s,k}]^2 \, d\theta_{(\cdot,\cdot,\cdot)} \tag{30}$$

$$\left[\langle \mathbb{P}[\mathcal{E} \mid \theta_{(\cdot,\cdot,\cdot)}] \rangle_{p[\theta_{(\cdot,\cdot,\cdot)} \mid \mathcal{H}^{s,k}]}\right]^2 \leq \int_0^1 .. \int_0^1 \mathbb{P}[\mathcal{E} \mid \theta_{(\cdot,\cdot,\cdot)}]^2 \cdot p[\theta_{(\cdot,\cdot,\cdot)} \mid \mathcal{H}^{s,k}]^2 \, d\theta_{(\cdot,\cdot,\cdot)} \tag{31}$$

The first inequality holds because $p[\theta_{(\cdot,\cdot,\cdot)} \mid \mathcal{H}^{s,k}] \in (0, 1]$, the second follows from the Cauchy-Bunyakovsky-Schwarz inequality. □

Prop. 5.1 shows that the posterior predictive distribution is always biased towards the observed evidence $\mathcal{E}$. We now move our attention to the belief update $\mathcal{H} \rightarrow \hat{\mathcal{H}}^*$ and show how it relates to the posterior predictive. As usual, $\hat{\mathcal{H}}^*$ denotes the value of $\hat{\mathcal{H}}$ that minimize the following measure:

$$\begin{aligned}
\mathsf{KL}^{\mathsf{div}} &= \int_0^1 .. \int_0^1 p[\theta_{(\cdot,\cdot,\cdot)} | \mathcal{E}, \mathcal{H}^{s,k}] \ln \left[\frac{p[\theta_{(\cdot,\cdot,\cdot)} | \mathcal{E}, \mathcal{H}^{s,k}]}{p[\theta_{(\cdot,\cdot,\cdot)} | \hat{\mathcal{H}}^{s,k}]}\right] d\theta_{(\cdot,\cdot,\cdot)} \\
&= h\left[p[\theta_{(\cdot,\cdot,\cdot)} | \mathcal{E}, \mathcal{H}^{s,k}]\right] - \int_0^1 .. \int_0^1 p[\theta_{(\cdot,\cdot,\cdot)} | \mathcal{E}, \mathcal{H}^{s,k}] \ln p[\theta_{(\cdot,\cdot,\cdot)} | \hat{\mathcal{H}}^{s,k}] d\theta_{(\cdot,\cdot,\cdot)} \\
&= h\left[p[\theta_{(\cdot,\cdot,\cdot)} | \mathcal{E}, \mathcal{H}^{s,k}]\right] - \langle \ln p[\theta_{(\cdot,\cdot,\cdot)} | \hat{\mathcal{H}}^{s,k}] \rangle_{p[\theta_{(\cdot,\cdot,\cdot)} | \mathcal{E}, \mathcal{H}^{s,k}]}
\end{aligned}$$

It is easy to conclude that

$$\hat{\mathcal{H}}^* = \underset{\hat{\mathcal{H}}}{\operatorname{argmin}} \, \mathsf{KL}^{\mathsf{div}} = \underset{\hat{\mathcal{H}}}{\operatorname{argmax}} \langle \ln p[\theta_{(\cdot,\cdot,\cdot)} | \hat{\mathcal{H}}^{s,k}] \rangle_{p[\theta_{(\cdot,\cdot,\cdot)} | \mathcal{E}, \mathcal{H}^{s,k}]} \tag{32}$$

In other words, $\hat{\mathcal{H}}^*$ is a maximum likelihood estimator for the posterior predictive distribution, which in turn is biased towards the evidence $\mathcal{E}$ that triggered the update $\mathcal{H} \rightarrow \hat{\mathcal{H}}^*$. Therefore, we can devise an EM algorithm by simply repeating multiple belief updates until convergence. The above conclusion, together with the considerations from [47], can be used to justify several variants of the EM algorithm. In the following we provide the pseudo-code of the classic, fully

Bayesian, soft-EM (here named Algorithm 2). Intuitively, the "E-step" consists of the computation of the posterior $p[\theta_{(\cdot,\cdot,\cdot)}|\mathcal{E}, \mathcal{H}^{s,k}]$, while the "M-step" consists of the belief update $\mathcal{H} \rightarrow \hat{\mathcal{H}}^*$.

Algorithm 2: Greedy-MLE

Data: Model $\mathcal{H}$, evidence $\mathcal{E} = \{\varphi_1^{t_1}, \ldots, \varphi_k^{t_k}\}$

1 **for** $j \in \{1, \ldots, k\}$ **do**
2 $\quad \tau_j \leftarrow t_j/s$;
3 **repeat**
4 $\quad$ **for** $i \in \{1, \ldots, n\}$ **do**
5 $\quad\quad r_i \leftarrow \sum_{j=1}^{k} \frac{1}{k}(\tau_j \mathbb{P}[x_i|\varphi_j, \mathcal{H}] + \overline{\tau_j}\mathbb{P}[x_i|\overline{\varphi_j}, \mathcal{H}])$
6 $\quad\quad (\hat{a}_i^*, \hat{b}_i^*) \leftarrow \mathrm{bu}(a_i, b_i, r_i)$
7 $\quad$ **for** $i \in \{1, \ldots, n\}$ **do**
8 $\quad\quad a_i \leftarrow \hat{a}_i^*$
9 $\quad\quad b_i \leftarrow \hat{b}_i^*$
10 **until convergence**;

Example 5.2 (continued). Let's assume we are given a B-PDB with two tuples, x_1 and x_2, and $k = 2$ queries: $\varphi_1 = x_1 \wedge x_2$ and $\varphi_2 = x_1 \vee x_2$. The initial state of the database, $\mathcal{H}_0$, is

$$a_1 = 1 \qquad a_2 = 1 \qquad b_1 = 3 \qquad b_2 = 3$$

Query φ_1 is observed to be satisfied $t_1 = 32$ times over $s = 100$ samples, while φ_2 is observed to be true $t_2 = 88$ times. Hence $\mathcal{E} = \{\varphi_1^{t_1}, \varphi_2^{t_2}\}$, and the target marginal probabilities for φ_1 and φ_2 are, respectively, $\tau_1 = 0.32$ and $\tau_2 = 0.88$. Given an arbitrary B-PDB $\mathcal{H}$, the likelihood of observing $\mathcal{E}$ being generated by $\mathcal{H}$ is $\mathbb{P}[\mathcal{E}|\mathcal{H}^{100,2}] =$

$$= \binom{100}{32}\mathbb{P}[\varphi_1|\mathcal{H}]^{32}\mathbb{P}[\overline{\varphi}_1|\mathcal{H}]^{68} \cdot \binom{100}{88}\mathbb{P}[\varphi_2|\mathcal{H}]^{88}\mathbb{P}[\overline{\varphi}_2|\mathcal{H}]^{12}$$

The likelihood is maximized when $\mathbb{P}[\varphi_1|\mathcal{H}] = \tau_1$ and $\mathbb{P}[\varphi_2|\mathcal{H}] = \tau_2$. There are two values of $\boldsymbol{\theta}$ that satisfy these conditions: either $\boldsymbol{\theta} = (0.8, 0.4)$, or $\boldsymbol{\theta} = (0.4, 0.8)$. Figure 5 shows how Algorithm 2 converges to one of the optimal values for $\boldsymbol{\theta}$: the green dots represent the state of the database (in terms of $\langle\theta_1\rangle_{\mathcal{H}}$ and $\langle\theta_2\rangle_{\mathcal{H}}$) after each iteration of the cycle at lines 3-10. The starting point is $\langle\boldsymbol{\theta}\rangle_{\mathcal{H}} = (0.25, 0.25)$.

It is important to notice that, for the sake of performing MLE, we are not strictly required to process the whole available evidence in every iteration of lines 3-10. Under reasonable assumptions, it is safe to modify Algorithm 2 so that, at each iteration, only a small portion of the evidence is processed, as long as we ensure that, in the long term, each portion is processed with uniform frequency and, as a consequence, it is equally represented. For example, we could modify Algorithm 2 so to process one query at-a-time (and use the update rule from Definition 4.5) or to process one sample at-a-time (and use the update rule from Definition 4.3). The query/sample to be processed next could be chosen with a deterministic policy (for example: round-robin) or with a randomized one (for example: uniform sampling). While all these variants will result into a sound greedy MLE algorithm, they are not semantically equivalent from the point of view of belief updates. To understand why, consider the following observation: performing a single BU step on model $\mathcal{H}^2$ for evidence $\{\varphi^1\}$ will not have necessarily the same outcome as performing a sequence of two distinct BU steps on model $\mathcal{H}$, one with evidence $\{\varphi\}$ and the other with evidence $\{\overline{\varphi}\}$. As shown in Figure 5 the likelihood function may have several local maximums and even several global ones.

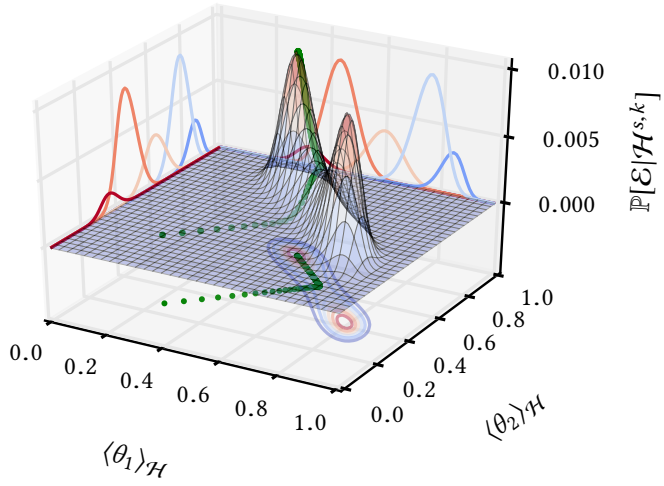


Fig. 5. Convergence of Algorithm 2 towards a maximum likelihood estimate of (θ_1, θ_2).

In the general case Algorithm 2, and its variations discussed above, will only converge to a *local* optimum.

6 COMPUTING CONDITIONAL PROBABILITIES

Computing conditional probabilities in the form $\mathbb{P}[x_i|\varphi_j, \mathcal{H}]$ is a central requirement for both Algorithm 1 and Algorithm 2. As discussed in Section 3, $\mathbb{P}[x_i|\varphi_j, \mathcal{H}]$ denotes the probability of observing tuple x_i being present in a possible world sampled from $\mathcal{H}$ that satisfies φ_j. In this Section we study the computational complexity of deriving such probabilities.

Theorem 6.1. *Let φ_j represent the lineage of a Boolean conjunctive query, and x_i be one of its literals. In general, computing the conditional probabilities $\mathbb{P}[x_i|\varphi_j, \mathcal{H}]$ (or $\mathbb{P}[x_i|\overline{\varphi_j}, \mathcal{H}]$) is #$\mathcal{P}$-hard but it is in PTIME when φ_j is read-once.*

Proof. By Turing reduction. First we prove that computing $\mathbb{P}[x_i|\varphi_j, \mathcal{H}]$ takes polynomial time in the size of $\mathcal{D}$ when φ_j is read-once. We start by observing that

$$\mathbb{P}[x_i|\varphi_j, \mathcal{H}] = \mathbb{P}[\varphi_j|x_i, \mathcal{H}] \cdot \mathbb{P}[x_i|\mathcal{H}]/\mathbb{P}[\varphi_j|\mathcal{H}] \tag{33}$$

Notice that $\mathbb{P}[\varphi_j|x_i, \mathcal{H}]$ represents the marginal probability of the formula obtained by replacing x_i with $\top$ in φ_j. We denote by $(\varphi_j|_{x_i})$ such formula, therefore $\mathbb{P}[(\varphi_j|_{x_i})|\mathcal{H}] = \mathbb{P}[\varphi_j|x_i, \mathcal{H}]$. If φ_j is read-once then so is $(\varphi_j|_{x_i})$, therefore computing the right-hand side of Equation (33) takes polynomial time. We now analyze the worst-case complexity of computing $\mathbb{P}[x_i|\varphi_j, \mathcal{H}]$: let φ_j be a non-read-once expression, and $\{x_1, \ldots, x_n\}$ the literals appearing in it; we want to reduce the problem of computing $\mathbb{P}[\varphi_j|\mathcal{H}]$ to the problem of computing conditional probabilities in the form $\mathbb{P}[x|\varphi, \mathcal{H}]$. From Equation (33), it is immediate to obtain

$$\mathbb{P}[\varphi_j|\mathcal{H}] = \mathbb{P}[x_i|\mathcal{H}] \cdot \mathbb{P}[(\varphi_j|_{x_i})|\mathcal{H}]/\mathbb{P}[x_i|\varphi_j, \mathcal{H}] \tag{34}$$

Since Equation (34) holds for any literal in $\{x_1, \ldots, x_n\}$, we can apply it $n-1$ times and obtain:

$$\mathbb{P}[\varphi_j|\mathcal{H}] = \frac{\mathbb{P}[x_1|\mathcal{H}]}{\mathbb{P}[x_1|\varphi_j, \mathcal{H}]} \cdot \frac{\mathbb{P}[x_2|\mathcal{H}]}{\mathbb{P}[x_2|(\varphi_j|_{x_1}), \mathcal{H}]} \cdots \frac{\mathbb{P}[x_{n-1}|\mathcal{H}]}{\mathbb{P}[x_{n-1}|(\varphi_j|_{x_1 \ldots x_{n-2}}), \mathcal{H}]} \cdot \mathbb{P}[(\varphi_j|_{x_1 \ldots x_{n-1}})|\mathcal{H}]$$

Notice that the last factor consists of the probability of a read-once Boolean formula, as the expression $(\varphi_j|_{x_1..x_{n-1}})$ depends only on the literal x_n. Therefore, if we have an oracle able to compute conditional probabilities in the form $\mathbb{P}[x|\varphi,\mathcal{H}]$, we can compute $\mathbb{P}[\varphi_j|\mathcal{H}]$ in polynomial time, by making $(n-1)$ calls[10]. □

From Lemma 2.3 and Theorem 6.1 it follows immediately that computing a single Bayesian update, to incorporate the answer to a hierarchical query, takes polynomial time in the data-size. We point out that Bayesian updating is not restricted to hierarchical queries: if φ_j is not read-once it is always possible to approximate $\mathbb{P}[x_i|\varphi_j,\mathcal{H}]$ with standard inference techniques. In the next Section we adapt the well-known algorithm by Dalvi and Suciu [9] to the goal of computing Bayesian updates extensionally, by means of "CP-plans".

6.1 CP-plans: Extensional Evaluation of $\mathbb{P}[x_i|\varphi_j,\mathcal{H}]$ for Hierarchical Queries

Let $\mathbf{q} = [\varphi_1, .., \varphi_k]$ be a non-Boolean hierarchical query. Our goal is to compute $\mathbb{P}[x_i|\varphi_j,\mathcal{H}]$ for every Boolean query φ_j in $\{\varphi_1, .., \varphi_k\}$ and every literal x_i appearing in φ_j. We first show how to compute $\mathbb{P}[\varphi_j|\mathcal{H}]$ and $\mathbb{P}[\varphi_j|x_i,\mathcal{H}]$ for every φ_j and x_i, extensionally. Once $\mathbb{P}[\varphi_j|\mathcal{H}]$ and $\mathbb{P}[\varphi_j|x_i,\mathcal{H}]$ are known, it is immediate to obtain $\mathbb{P}[x_i|\varphi_j,\mathcal{H}]$ by Equation (33). A plan performing such computation is called a"*CP-plan*". In order to represent CP-plans compactly, we introduce a simple extension of pRA. In our algebra, a CP-plan (P^{cp}) is a sentence respecting the following grammar:

$$P^{\mathsf{cp}} ::= CP(R_0^{\mathsf{p}}) \mid \pi^{\mathsf{c}}_X(P') \mid \sigma^{\mathsf{c}}(P') \mid \ \bowtie^{\mathsf{c}} [P', P'', \ldots]$$

R_0^{p} represents an arbitrary TI-relation, where each tuple has a unique identifier `tid` and is associated with a marginal probability `p`. Let's assume $\mathbf{A}$ is a key for R_0^{p}, consisting of all the attributes except for `tid` and `p`. The operator $CP(R^{\mathsf{p}})$ turns a TI-relation R_0^{p} into a pair $(R^{\mathsf{p}}, R^{\mathsf{cp}})$, where

$$R^{\mathsf{p}}(\mathbf{A},\mathsf{p}) \stackrel{\text{def}}{=} \pi_{\mathbf{A},\mathsf{p}}(R_0^{\mathsf{p}}) \qquad R^{\mathsf{cp}}(\mathbf{A},\mathsf{cp},\mathtt{lt}) \stackrel{\text{def}}{=} \pi_{\mathbf{A},1,\mathtt{tid}}(R_0^{\mathsf{p}})$$

Intuitively, R^{p} is obtained from R_0^{p} by projecting-away `tid`. R^{cp} associates each tuple x of R_0^{p} with the conditional probability $\mathbb{P}[x|x]$, which is, by definition, equal to 1. All the other operators of our algebra process pairs of relations in the form $(R^{\mathsf{p}}, R^{\mathsf{cp}})$. Let $\mathbf{B}$ be a strict subset of $\mathbf{A}$. If we apply the projection operator $\pi^{\mathsf{c}}_{\mathbf{B}}$ to the pair $(R^{\mathsf{p}}, R^{\mathsf{cp}})$, we obtain a pair of relations $(Q^{\mathsf{p}}, Q^{\mathsf{cp}})$, defined as follows:

$$Q^{\mathsf{p}}(\mathbf{B},\mathsf{p}) \stackrel{\text{def}}{=} \pi_{\mathbf{B},(1-\Pi_{\text{agg}}(\overline{R^{\mathsf{p}}.\mathsf{p}}))}(R^{\mathsf{p}})$$

$$Q^{\mathsf{cp}}(\mathbf{B},\mathsf{cp},\mathtt{lt}) \stackrel{\text{def}}{=} \pi_{\mathbf{A},\mathsf{cpexp},R^{\mathsf{cp}}.\mathtt{lt}}[(R^{\mathsf{p}} \bowtie_{\mathbf{A}} R^{\mathsf{cp}}) \bowtie_{\mathbf{B}} Q^{\mathsf{p}}]$$

Here, $\Pi_{\text{agg}}(\cdot)$ denotes the aggregate product and $\mathsf{cpexp} \stackrel{\text{def}}{=} 1 - (\overline{Q^{\mathsf{p}}.\mathsf{p}} \cdot \overline{R^{\mathsf{cp}}.\mathsf{cp}}/\overline{R^{\mathsf{p}}.\mathsf{p}})$. The selection operator (σ^{c}) simply applies the selection predicate to both R^{p} and R^{cp}. Therefore, the statement $(Q^{\mathsf{p}}, Q^{\mathsf{cp}}) = \sigma^{\mathsf{c}}(R^{\mathsf{p}}, R^{\mathsf{cp}})$ is equivalent to the following RA plan:

$$Q^{\mathsf{p}}(\mathbf{A},\mathsf{p}) \stackrel{\text{def}}{=} \sigma(R^{\mathsf{p}}) \qquad Q^{\mathsf{cp}}(\mathbf{A},\mathsf{cp},\mathtt{lt}) \stackrel{\text{def}}{=} \sigma(R^{\mathsf{cp}})$$

Let's now assume we are given a collection of m relation pairs $\{(R_1^{\mathsf{p}}, R_1^{\mathsf{cp}}), .., (R_m^{\mathsf{p}}, R_m^{\mathsf{cp}})\}$ and $\mathbf{A}_i = \mathsf{hvar}(R_i^{\mathsf{p}}) \setminus \{\mathsf{p}\}$. Let's define $\mathbf{A} = \cup_{i=1}^{m}\mathbf{A}_i$. The statement $(Q^{\mathsf{p}}, Q^{\mathsf{cp}}) = \bowtie^{\mathsf{c}} [(R_1^{\mathsf{p}}, R_1^{\mathsf{cp}}), .., (R_m^{\mathsf{p}}, R_m^{\mathsf{cp}})]$ is

[10] A similar conclusion can be drawn from the work of Kanagal et al. [36] by noticing that the *sensitivity* $\frac{\partial \mathbb{P}[\varphi_j]}{\partial \theta_i}$ can be computed as the difference $\mathbb{P}[\varphi_j|x_i] - \mathbb{P}[\varphi_j|\overline{x_i}]$.

equivalent to the following RA plan:

$$Q^{\mathsf{p}}(\mathbf{A}, \mathsf{p}) \stackrel{\text{def}}{=} \pi_{\mathbf{A}, (\Pi_{i=1}^{m} R_i^{\mathsf{p}}.\mathsf{p})}(\bowtie [R_1^{\mathsf{p}}, .., R_m^{\mathsf{p}}])$$
$$V_i(\mathbf{A}, \mathsf{cp}, \mathtt{lt}) \stackrel{\text{def}}{=} \pi_{\mathbf{A}, \mathsf{cpexp}, R_i^{\mathsf{cp}}.\mathtt{lt}}(\bowtie_{\mathbf{A}_i} [Q^{\mathsf{p}}, R_i^{\mathsf{p}}, R_i^{\mathsf{cp}}]) \; \forall i \in \{1..m\}$$
$$Q^{\mathsf{cp}}(\mathbf{A}, \mathsf{cp}, \mathtt{lt}) \stackrel{\text{def}}{=} \uplus_{i=1}^{m} V_i$$

Here $\mathsf{cpexp} \stackrel{\text{def}}{=} (Q^{\mathsf{p}}.\mathsf{p} \cdot R_i^{\mathsf{cp}}.\mathsf{cp} / R_i^{\mathsf{p}}.\mathsf{p})$. Now that we have defined all the operators of our algebra, we can show how to build a CP-plan for a given hierarchical query. The method we propose (Algorithm 3) is a straightforward adaptation to conditional probabilities of the procedure by Dalvi and Suciu [9] for constructing safe plans for computing marginal probabilities. The first component of the plan (Q^{p}) is constructed exactly as in Dalvi and Suciu's method. The second component of the plan (Q^{cp}) also closely mirrors this method, excepting only the leaves, where $R^{\mathsf{cp}} = \pi_{R^{\mathsf{p}}.\mathbf{A}, 1, R^{\mathsf{p}}.\mathtt{tid}}(R^{\mathsf{p}})$, which is equivalent for the purposes of safety. In other words, Algorithm 3 can be thought of as two independent instances of the Dalvi and Suciu method run in parallel. Their algorithm is known to be sound and complete; Algorithm 3 inherits both properties.

Algorithm 3: SafeCpPlan

Data: Hierarchical query $\mathbf{q}(..) :- R(..), S(..), \ldots$

if $\mathtt{evar}(\mathbf{q}) = \emptyset$ **then**
  return $CP(R) \bowtie^{\mathsf{c}} CP(S) \bowtie^{\mathsf{c}} \ldots$
else if $\mathbf{q} :- \mathbf{q}', \mathbf{q}''$ *and* $\mathtt{evar}(\mathbf{q}') \cap \mathtt{evar}(\mathbf{q}'') = \emptyset$ **then**
  return SafeCpPlan($\mathbf{q}'$) $\bowtie^{\mathsf{c}}$ SafeCpPlan($\mathbf{q}''$)
else if $\exists X \in \mathtt{evar}(\mathbf{q})$: X *is a root variable* **then**
  return π^{c}_{-X}(SafeCpPlan($\mathbf{q}'(X, \mathtt{hvar}(\mathbf{q})) :- R(..), S(..), \ldots$))

Let $(Q^{\mathsf{p}}, Q^{\mathsf{cp}})$ be the result of a CP-plan generated by Algorithm 3: if φ_j is the lineage of a tuple in Q^{p} and its literals are $\{x_1, .., x_m\}$, then Q^{cp} is guaranteed to contain m copies of such tuple, each copy being associated with a conditional probability $\mathbb{P}[\varphi_j | x_i]$, for every x_i in $\{x_1, .., x_m\}$.

Example 6.2 (continued). If $\mathbf{q}$ denotes the hierarchical query from Equation (5), then SafeCpPlan($\mathbf{q}$) returns the plan:

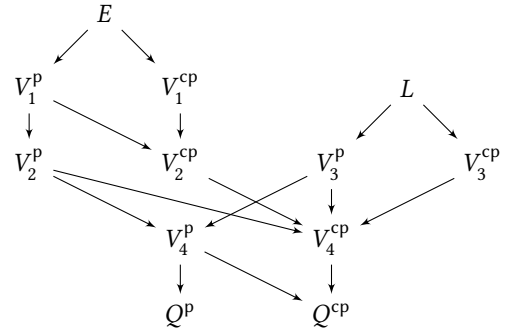
$$(Q^{\mathsf{p}}, Q^{\mathsf{pc}}) = \pi^{\mathsf{c}}_{-X}(\pi^{\mathsf{c}}_{-Y}(CP(E)) \bowtie^{\mathsf{c}} CP(L))$$

Let's split it into several views, in a bottom-up fashion:

$$(V_1^{\mathsf{p}}, V_1^{\mathsf{cp}}) = CP(E)$$
$$(V_2^{\mathsf{p}}, V_2^{\mathsf{cp}}) = \pi^{\mathsf{c}}_{-Y}(V_1^{\mathsf{p}}, V_1^{\mathsf{cp}})$$
$$(V_3^{\mathsf{p}}, V_3^{\mathsf{cp}}) = CP(L)$$
$$(V_4^{\mathsf{p}}, V_4^{\mathsf{cp}}) = (V_2^{\mathsf{p}}, V_2^{\mathsf{cp}}) \bowtie^{\mathsf{c}} (V_3^{\mathsf{p}}, V_3^{\mathsf{cp}})$$
$$(Q^{\mathsf{p}}, Q^{\mathsf{cp}}) = \pi^{\mathsf{c}}_{-X}(V_4^{\mathsf{p}}, V_4^{\mathsf{cp}})$$

The content of each view is presented in Figure 6. For the sake of clarity we have annotated each tuple with its lineage; with little abuse of notation, some lineage formulas contain conditional expressions; translating them into regular Boolean formulas is straightforward. Notice that the CP-plan does not materialize these formulas.

Notice that Q^{p} is equivalent to the TI-relation returned by plan P' in Equation (6).



V_1^{p}

name	emp	p	(lin.)
Ada	HP	0.6	x_1
Ada	IBM	0.6	x_2
Bob	HP	0.5	x_3

V_1^{cp}

name	emp	cp	lt	(lin.)
Ada	HP	1	x_1	$x_1 \mid x_1$
Ada	IBM	1	x_2	$x_2 \mid x_2$
Bob	HP	1	x_3	$x_3 \mid x_3$

V_2^{p}

name	p	(lin.)
Ada	0.84	$x_1 \vee x_2$
Bob	0.5	x_3

V_2^{cp}

name	cp	lt	(lin.)
Ada	1	x_1	$x_1 \vee x_2 \mid x_1$
Ada	1	x_2	$x_2 \vee x_2 \mid x_2$
Bob	1	x_3	$x_3 \mid x_3$

V_3^{p}

name	lng	p	(lin.)
Ada	eng	0.4	x_4
Bob	eng	0.2	x_5
Bob	ita	0.6	x_6

V_3^{cp}

name	lng	cp	lt	(lin.)
Ada	eng	1	x_4	$x_4 \mid x_4$
Bob	eng	1	x_5	$x_5 \mid x_5$
Bob	ita	1	x_6	$x_6 \mid x_6$

V_4^{p}

name	lng	p	(lin.)
Ada	eng	0.34	$(x_1 \vee x_2)x_4$
Bob	eng	0.1	x_3x_5
Bob	ita	0.3	x_3x_6

V_4^{cp}

name	lng	cp	lt	(lin.)
Ada	eng	0.4	x_1	$(x_1 \vee x_2)x_4 \mid x_1$
Ada	eng	0.4	x_2	$(x_1 \vee x_2)x_4 \mid x_2$
Ada	eng	0.84	x_4	$(x_1 \vee x_2)x_4 \mid x_4$
Bob	eng	0.2	x_3	$x_3x_5 \mid x_3$
Bob	eng	0.5	x_5	$x_3x_5 \mid x_5$
Bob	ita	0.6	x_3	$x_3x_6 \mid x_3$
Bob	ita	0.5	x_6	$x_3x_6 \mid x_6$

Q^{p}

lng	p	(lin.)
eng	0.4024	$((x_1 \vee x_2)x_4) \vee x_3x_5$
ita	0.3	x_3x_6

Q^{cp}

name	cp	lt	(lin.)
eng	0.46	x_1	$((x_1 \vee x_2)x_4) \vee x_3x_5 \mid x_1$
eng	0.46	x_2	$((x_1 \vee x_2)x_4) \vee x_3x_5 \mid x_2$
eng	0.4688	x_3	$((x_1 \vee x_2)x_4) \vee x_3x_5 \mid x_3$
eng	0.856	x_4	$((x_1 \vee x_2)x_4) \vee x_3x_5 \mid x_4$
eng	0.668	x_5	$((x_1 \vee x_2)x_4) \vee x_3x_5 \mid x_5$
ita	0.6	x_3	$x_3x_6 \mid x_3$
ita	0.5	x_6	$x_3x_6 \mid x_6$

Fig. 6. CP-plan data dependencies

7 EXTENSION TO MUTUALLY EXCLUSIVE EVENTS

In this section we relax the assumption that each tuple is stochastically independent from all the others, allowing some tuples to be mutually exclusive to others. We first introduce the *disjoint-independent* model, a well-known [10] generalization of TI-PDBs that is able to model mutually exclusive events. Later we generalize our own framework in the same spirit, and show how to handle Bayesian updates and parameter learning in the new model.

7.1 Disjoint-Independent PDBs

A disjoint-independent probabilistic database (DI-PDB) is a TI-PDB where each relation is annotated with a *key*. In the context of DI-PDBs, a key is simply a collection of attributes, with the property that every two tuples in a relation that agree on the key attributes must represent mutually exclusive events. If two tuples do not agree on the key attributes, then they represent stochastically

independent events. The set of all tuples in a relation that agree on some key value form a *block*. Since a block represents a collection of mutually exclusive events, the sum of the probabilities of its tuples must not be greater than 1. For convenience we identify each tuple in a DI-PDB with a lineage literal $x_{u,v}$, where index u identifies a specific block (x_u) and index v a specific tuple within the block. We adopt the same indexing strategy to identify tuples' probabilities (hence, each tuple $x_{u,v}$ is associated with a probability $\theta_{u,v}$ and a DI-relation always satisfies the constraint $\forall u: \ \sum_v \theta_{u,v} \leq 1$). For simplicity, we assume that every block has exactly $c \geq 1$ tuples.

A *possible world* w is a subset of tuples where no two elements belong to the same block. We can model it as a function $w(u)$ that maps each block to either one of its tuples (when that tuple belongs to w) or the value ε (when no tuple from the block appears in w). A disjoint-independent database $\mathcal{D}$ defines a probability measure over both possible worlds and Boolean queries:

$$\mathbb{P}[w(u)|\mathcal{D}] = \begin{cases} \theta_{u,v} & \text{if } w(u) = x_{u,v} \\ 1 - \sum_v \theta_{u,v} & \text{if } w(u) = \varepsilon \end{cases} \tag{35}$$

$$\mathbb{P}[w|\mathcal{D}] = \prod_u \mathbb{P}[w(u)|\mathcal{D}] \tag{36}$$

$$\mathbb{P}[q|\mathcal{D}] = \sum_{w \models q} \mathbb{P}[w|\mathcal{D}] \tag{37}$$

The lineage expression $x_u = x_{u,v}$ (or simply $x_{u,v}$) denotes the event of being in a possible world w where $w(u) = x_{u,v}$ With little abuse of notation we denote by $x_u = x_{u,\varepsilon}$ (or simply $x_{u,\varepsilon}$) the lineage of the event of being in a possible world where block x_u is associated with the value ε, and by $\theta_{u,\varepsilon} = 1 - \sum_v \theta_{u,v}$ its probability.

Example 7.1 (continued). Let's revisit our running example, imposing the constraint that each person can work for at most one company. The resulting probabilistic model can be expressed as a DI-PDB:

E^{p}			
name	emp	*tid*	θ
Ada	HP	$x_{1,1}$	0.6
Ada	IBM	$x_{1,2}$	0.3
Bob	HP	$x_{2,1}$	0.5

L^{p}			
name	lng	*tid*	θ
Ada	eng	$x_{3,1}$	0.4
Bob	eng	$x_{4,1}$	0.2
Bob	ita	$x_{5,1}$	0.6

Key attributes are underlined. Relation E^{p} and L^{p} have, respectively, six and eight possible worlds. Notice that DI-relations can model TI-relations, as is the case with L^{p}. The probability that Ada is unemployed ($x_{1,\varepsilon}$) is 0.1. The Boolean query $q_1 :\!-\, E(X, Y), L(X, eng)$ has lineage $((x_{1,1} \vee x_{1,2})\ x_{3,1}) \vee x_{2,1}x_{4,1}$. Its probability is $\mathbb{P}[q_1|\mathcal{D}] = ((\theta_{1,1} + \theta_{1,2})\ \theta_{3,1}) \otimes \theta_{2,1}\theta_{4,1}$.

While hierarchical queries are not guaranteed to be tractable in the DI model, Dalvi and Suciu [10] developed a polynomial time algorithm to identify safe extensional plans for all conjunctive, self-join free queries that admit one. They derived a dichotomy theorem, showing that every conjunctive, self-join-free query either admits an extensional plan or is intractable (#P-hard).

7.2 Dirichlet-Probabilistic Databases (D-PDBs)

We are now ready to generalize the DI model so as to support Bayesian updates. Each block x_u in a DI-PDB can be seen as a categorically-distributed random variable, taking values in a domain of cardinality $c + 1$ (c tuples plus ε). From now on we denote by $\boldsymbol{\theta}_{\mathbf{u}}$ a vector of $c + 1$ parameters associated with the random variable x_u: $\boldsymbol{\theta}_{\mathbf{u}} \stackrel{\text{def}}{=} (\theta_{u,\varepsilon}, \theta_{u,1}, .., \theta_{u,c})$. Notice that $\boldsymbol{\theta}_{\mathbf{u}}$, by definition, ranges in the probabilistic simplex $C \stackrel{\text{def}}{=} \{\mathbf{x} \in (\mathbb{R}^+)^{c+1} : \sum_i x_i = 1\}$. Rather than taking $\boldsymbol{\theta}_{\mathbf{u}}$ as a given, known

quantity, we model it as a latent random vector, distributed according to a Dirichlet distribution with concentration parameters $\boldsymbol{\alpha}_u \stackrel{\text{def}}{=} (\alpha_{u,\varepsilon}, \alpha_{u,1}, .., \alpha_{u,c})$. Therefore, its marginal probability is

$$p[\boldsymbol{\theta}_u|\boldsymbol{\alpha}_u] = \frac{\theta_{u,\varepsilon}^{\alpha_{u,\varepsilon}-1} \cdot \theta_{u,1}^{\alpha_{u,1}-1} \cdots \theta_{u,c}^{\alpha_{u,c}-1}}{B(\boldsymbol{\alpha}_u)} \tag{38}$$

where $B(\boldsymbol{\alpha}_u)$ denotes the *generalized* Beta function

$$B(\boldsymbol{\alpha}_u) = \int_C \left(\theta_{u,\varepsilon}^{\alpha_{u,\varepsilon}-1} \cdot \theta_{u,1}^{\alpha_{u,1}-1} \cdots \theta_{u,c}^{\alpha_{u,c}-1}\right) d\boldsymbol{\theta}_u = \frac{\Gamma(\alpha_{u,\varepsilon}) \cdot \Gamma(\alpha_{u,1}) \cdots \Gamma(\alpha_{u,c})}{\Gamma(\alpha_{u,\varepsilon}) + \Gamma(\alpha_{u,1}) + \cdots + \Gamma(\alpha_{u,c})} \tag{39}$$

From now on we denote by $\mathcal{D}ir(\boldsymbol{\alpha}_u)$ the right-end-side of Equation (38). From the above definitions it follows immediately that the random variable x_u is distributed according to a *Dirichlet-Categorical compound distribution*, and its marginal probability is given by

$$\mathbb{P}[x_{u,y}|\boldsymbol{\alpha}_u] = \int_C \mathbb{P}[x_{u,y}|\boldsymbol{\theta}_u] \cdot p[\boldsymbol{\theta}_u|\boldsymbol{\alpha}_u] d\boldsymbol{\theta}_u = \frac{\alpha_{u,y}}{\alpha_{u,\varepsilon} + \alpha_{u,1} + \cdots + \alpha_{u,c}} \tag{40}$$

where y ranges in $\{\varepsilon, 1, \ldots, c\}$ and $\mathbb{P}[x_{u,y}|\boldsymbol{\theta}_\mathbf{u}] = \theta_{u,y}$. If we denote by $\mathcal{H}$ the set of all the hyper-parameters $\{\boldsymbol{\alpha}_u\}_u$, it is easy to see that $\mathcal{H}$ defines a probability measure over possible worlds (w) and Boolean queries (q):

$$\mathbb{P}[w|\mathcal{H}] = \prod_u \mathbb{P}[w(u)|\mathcal{H}] = \prod_u \mathbb{P}[w(u)|\boldsymbol{\alpha}_u] \tag{41}$$

$$\mathbb{P}[q|\mathcal{H}] = \sum_{w \models q} \mathbb{P}[w|\mathcal{H}] \tag{42}$$

Similarly, if φ represents the lineage of query q, then $\mathbb{P}[\varphi|\mathcal{H}] = \sum_{w\models\varphi} \mathbb{P}[w|\mathcal{H}]$ and conditional probabilities are well-defined as per Equation (14). In addition, Equations 16 and 17 can be readily applied to generalize the model to multiple observations and define the semantics of $\mathcal{H}^s$ and $\mathcal{H}^{s,k}$. For the sake of conciseness we skip the details. From now on we call $\mathcal{H}$ a *Dirichlet Probabilistic Database* (D-PDB). It is easy to verify that a B-PDB is a special case of D-PDB where $c = 1$ for each block or, equivalently, where every attribute belongs to its relation key.

Theorem 7.2. *If our current hypothesis is $\mathcal{H}$ and we observe a possible world where the lineage formula φ is satisfied, then, for each block x_u, the marginal posterior of random vector $\boldsymbol{\theta}_u$ is given by*

$$p[\boldsymbol{\theta}_u|\varphi, \mathcal{H}] = \sum_{y\in\{\varepsilon,1,..,c\}} \mathbb{P}[x_{u,y}|\varphi, \mathcal{H}] \cdot \mathcal{D}ir(\boldsymbol{\alpha}_u + \mathbf{e}_y) \tag{43}$$

where $\mathbf{e}$ denotes the standard natural basis of $\boldsymbol{\alpha}_u$.

Proof. The proof is given in Appendix B. □

In the general case the marginal posterior $p[\boldsymbol{\theta}_u|\varphi, \mathcal{H}]$ is not a Dirichlet distribution, but rather a Dirichlet mixture. Nonetheless, minimizing the relative entropy between $p[\boldsymbol{\theta}_u|\varphi, \mathcal{H}]$ and the prior $p[\boldsymbol{\theta}_u|\mathcal{H}]$ is straight-forward, as the latter belongs to the exponential family.

Definition 7.3. We denote by $\text{dfit}(\boldsymbol{\alpha}_u, \varphi)$ the vector $\boldsymbol{\alpha}_u^*$ that satisfies the following $c+1$ constraints:

$$\forall y \in \{\varepsilon, 1, .., c\} \quad \int_0^1 \mathcal{D}ir(\boldsymbol{\alpha}_u^*) \ln \theta_{u,y}\ d\theta_{u,y} = \int_0^1 p[\theta_{u,y}|\varphi, \mathcal{H}] \ln \theta_{u,y}\ d\theta_{u,y} \tag{44}$$

In other words $\boldsymbol{\alpha}^*_u$ is built so that $\mathcal{D}ir(\boldsymbol{\alpha}^*_u)$ agrees with $p[\theta_{u,y}|\varphi,\mathcal{H}]$ on the expected value of $\ln(\theta_{u,y})$, for each tuple $x_{u,y}$ in the block x_u. By Theorem 7.2 and the properties of the Dirichlet distribution the above equation can be rewritten as

$$\forall y \in \{\varepsilon, 1, .., c\} \quad [\psi(\alpha^*_{u,y}) - \psi(|\boldsymbol{\alpha}^*_u|_1)] - [\psi(\alpha_{u,y}) - \psi(|\boldsymbol{\alpha}_u|_1)] = \frac{\mathbb{P}[x_{u,y}|\varphi,\mathcal{H}]}{\alpha_{u,y}} - \frac{1}{|\boldsymbol{\alpha}_u|_1} \tag{45}$$

where $|\cdot|_1$ denotes the L-1 norm.

From [31] it follows immediately that $\mathcal{D}ir(\boldsymbol{\alpha}^*_u)$ is the Dirichlet distribution that minimizes the KL divergence w.r.t. the marginal posterior $p[\theta_{u,y}|\varphi,\mathcal{H}]$. Let's now assume that evidence $\mathcal{E}$ consists of $s \cdot k$ samples over k distinct Boolean queries, with s samples per query. In other words $\mathcal{E} = \{\varphi_1^{t_1}, \ldots, \varphi_k^{t_k}\}$. In order to minimize the relative entropy between the priors of $\mathcal{H}^{s,k}$ and the marginal posteriors computed w.r.t. $\mathcal{E}$, we must choose a set of hyper-parameters that satisfies, for each block x_u, the following constraints

$$\forall y \in \{\varepsilon, 1, .., c\} \quad [\psi(\alpha^*_{u,y}) - \psi(|\boldsymbol{\alpha}^*_u|_1)] - [\psi(\alpha_{u,y}) - \psi(|\boldsymbol{\alpha}_u|_1)] = \frac{r_y}{\alpha_{u,y}} - \frac{1}{|\boldsymbol{\alpha}_u|_1} \tag{46}$$

Where $r_y = \frac{1}{k}\sum_{j=1}^{k}\left[\frac{t_j}{s}\mathbb{P}[x_{u,y}|\varphi_j,\mathcal{H}] + \frac{s-t_j}{s}\mathbb{P}[x_{u,y}|\overline{\varphi}_j,\mathcal{H}]\right]$. Notice that r_y is the evidence-conditioned probability of tuple $x_{u,y}$ averaged over $s \cdot k$ evidence samples. Algorithm 4 shows how to perform a Bayesian Update in a D-PDB. By $\mathrm{du}(\boldsymbol{\alpha}_u, \mathbf{r})$ we denote a procedure that computes the hyper-parameters vector $\boldsymbol{\alpha}^*_u$ that satisfies Equation (46) for a given hypothesis $\mathcal{H}^{s,k}$ and evidence $\mathcal{E}$.

Algorithm 4: Belief Update (DI model)

Data: Model $\mathcal{H}$, evidence $\mathcal{E} = \{\varphi_1^{t_1}, \ldots, \varphi_k^{t_k}\}$

```
for j ∈ {1, ..., k} do
    τ_j ← t_j / s;
for u ∈ {1, ..., n} do
    for y ∈ {ε, 1, ..., c} do
        r_y ← Σ_{j=1}^{k} (1/k)(τ_j ℙ[x_{u,y}|φ_j, ℋ] + τ̄_j ℙ[x_{u,y}|φ̄_j, ℋ])
    α*_u ← du(α_u, r)
for u ∈ {1, ..., n} do
    α_u ← α*_u
```

It is easy to verify that the same considerations we introduced in Section 5 in the context of B-PDBs remain true for D-PDBs: by simply repeating multiple iterations of Algorithm 4 it is possible to design a greedy MLE algorithm for the DI model. We omit the details.

8 EXPERIMENTS

This section is divided in two parts. In the first part we provide experimental evaluation of the Bayesian updating technique proposed in section 4, in combination with the Dirichlet model presented in section 7. The second part is dedicated to maximum likelihood estimation.

Experimental Setup All the experiments discussed here were performed on a 3.4 GHz Intel Core i7-2600 with 32GB of DDR3-1333 MHz RAM and a 750GB SSD disk, running Ubuntu 14.04.5 LTS.

8.1 Bayesian updates and D-PDBs

We extended MayBMS [3], a publicly available[11] block-independent probabilistic database, with the support for Bayesian updating. Our implementation[12], that we call Beta-MayBMS, extends the original system in two ways: (i) it imposes Dirichlet priors on each parameter and (ii) it enables Bayesian updating w.r.t. arbitrary query answers. The standard, SQL-like syntax of MayBMS is augmented with the following statement

```
update <bi-tbl> given evidence ( <query> ) is [not] empty;
```

The `<query>` used as evidence can be any query supported by MayBMS; in other words, Bayesian updating is not restricted to hierarchical queries. More details about the design of Beta-MayBMS are given in Appendix D. Here we describe our experimental evaluation of it against a simple data cleaning task.

We obtained a copy of the 5% 1990 sample from the IPUMS-USA data set [54]. The sample consists of 12,501,046 records of census data, from which we extracted the following variables: `EDUC`, `MARST`, `CHBORN`, `EMPSTAT`, `VETSTAT`, `YRIMMIG`, `AGE`. We randomly selected a fraction of 0.0001% of the data and introduced noise over the attribute `AGE` by simulating typing errors. Each affected record is replaced with a block of mutually exclusive tuples, consisting of the original `AGE` value together with up to two noisy copies. The noise is introduced by randomly replacing (or dropping) a digit in the record's `AGE` value; these variations mimic the confusion matrix of a basic SVD classifier[13] trained against the MNIST [44] dataset of handwritten digits.

Our benchmark consists in asserting a number of query-answers that involve the attribute `AGE` and measure the evolution of the mean square error (MSE) over the noisy records after each belief update. We chose query-answers that express general knowledge about the data set; they are listed in Figure 7. Each query answer is asserted up to four times, for a total of 100 belief updates. As shown in Figure 8g, each Bayesian update effectively reduces the noise over the attribute `AGE`, and the largest improvement is gained during the first run of 25 assertions. Afterwards the MSE converges to the value of 165.

8.2 Maximum Likelihood Estimation and B-PDBs

This section is dedicated to the experimental evaluation of the maximum likelihood estimation algorithm proposed in Section 5, in the context of TI-PDBs. We compare our approach against the existing literature on parameter learning in probabilistic databases. Dylla and Theobald [16, 17] were the first to address this problem. In their work they model parameter learning as a constraint satisfaction problem: given a set of Boolean queries, labeled with marginal probabilities, their aim is to find a parameter vector $\boldsymbol{\theta}$ so that the resulting TI-PDB exhibits the desired marginal probabilities, for each Boolean query in the training set. They first address the problem from a theoretical perspective, characterizing its satisfiability and giving bounds on its computational complexity. They then show how to solve it using well-established optimization techniques, adopting several variants of the stochastic gradient descent algorithm. As discussed in Section 5, our Algorithm 2 can be used to perform a similar optimization task. The work of Dylla and Theobald is probably the closest to ours amongst the existing literature, and the most suitable for an experimental comparison. Concretely, our experiments aim to verify the following three conjectures: (i) our MLE algorithm has comparable performance to the one proposed by Dylla and Theobald; (ii) our algorithm scales well when used on large datasets; (iii) our algorithm can be polarized towards

[11] http://maybms.sourceforge.net/
[12] https://gitlab.odin.cse.buffalo.edu/niccolom/beta-maybms-public
[13] https://github.com/antononcube/MathematicaVsR/blob/master/Projects/HandwrittenDigitsClassificationByMatrixFactorization/R/HandwrittenDigitsClassificationByMatrixFactorization.pdf

```
(1) (select * from ipums where AGE<=2  and EDUC>0) is empty
(2) (select * from ipums where AGE<=9  and EDUC>1) is empty
(3) (select * from ipums where AGE<=13 and EDUC>2) is empty
(4) (select * from ipums where AGE<=19 and EDUC>(AGE-11)) is empty
(5) (select * from ipums where AGE<=23 and EDUC>10) is empty
(6) (select * from ipums where AGE>=8  and EDUC=0) is empty
(7) (select * from ipums where AGE>=14 and AGE<=50 and EDUC<=1) is empty
(8) (select * from ipums where AGE<=16 and MARST<>6) is empty
(9) (select * from ipums where AGE>=74 and MARST=3) is empty
(10) (select * from ipums where AGE<40  and MARST=5) is empty
(11) (select * from ipums where AGE<=20 and MARST between 3 and 5) is empty
(12) (select * from ipums where AGE<=14 and CHBORN<>0) is empty
(13) (select * from ipums where AGE<=17 and CHBORN>1) is empty
(14) (select * from ipums where AGE<=19 and CHBORN>2) is empty
(15) (select * from ipums where AGE<=22 and CHBORN>3) is empty
(16) (select * from ipums where AGE<=30 and CHBORN>4) is empty
(17) (select * from ipums where AGE<=35 and CHBORN>5) is empty
(18) (select * from ipums where AGE<=45 and CHBORN>6) is empty
(19) (select * from ipums where AGE<=15 and EMPSTAT<>0) is empty
(20) (select * from ipums where AGE>=16 and EMPSTAT=0 ) is empty
(21) (select * from ipums where AGE>=62 and EMPSTAT=2 ) is empty
(22) (select * from ipums where AGE>=86 and EMPSTAT=1 ) is empty
(23) (select * from ipums where AGE<=15 and VETSTAT<>0) is empty
(24) (select * from ipums where AGE>=16 and VETSTAT=0 ) is empty
(25) (select * from ipums where YRIMMIG>0 and YRIMMIG<1990 and AGE<(1990-YRIMMIG))) is empty
```

Fig. 7. Query-answers for the IPUMS data set

a specific maximum of the likelihood, when needed, by setting the priors accordingly. Unlike the previous section, all the experiments presented here use TI datasets and hierarchical queries. Before discussing the experiments in details, we briefly describe our prototype implementation of Algorithm 2.

While Beta-MayBMS is able compute belief updates for arbitrary queries, it requires to process each update as a full SQL statement, going through the usual steps of parsing, planning, optimization and execution of each query. This is not optimal for iterative processes like MLE, where the same queries, plans and resources are reused many times until the algorithm converges. We decided to provide an alternative implementation, targeted at the specific task of running MLE over B-PDBs for tractable, hierarchical queries. We call our prototype BetaMLE: it consists of two components, a *query planner* and an *execution engine*. The main purpose of the planner is to build, for a given set of queries, a compact representation of the lineage formulas associated with their marginal and conditional probabilities. For each query, the planner builds a CP-plan, by running Algorithm 3, and then uses a standard SQL DBMS to ground it. It then computes the lineage formulas of the grounded tuples and feeds them to the execution engine. The purpose of the execution engine is to run Algorithm 2. At each iteration it uses the information provided by the planner to compute the marginal and conditional probabilities for all the queries. It then applies Equation (33) to obtain conditional probabilities in the form $\mathbb{P}[x|\varphi, \mathcal{H}]$. It eventually computes the Bayesian Updates, as explained in Section 4. This decoupled architecture has several advantages: first it ensures that the SQL optimizer is called only once per problem instance, even if the same set of queries is used over many iterations of Algorithm 2; secondly it greatly simplifies locking, parallelism and memory allocation in the execution engine. The query planner is implemented in Java; its output consists of a SQL script that generates the ground CP-plans. The execution engine is implemented in C++11

data set	YAGO	YAGO	YAGO	TPCH	TPCH	TPCH	TPCH	TPCH	TPCH
scaling factor	-	-	-	0.1	0.1	0.5	0.5	1.0	1.0
query set	$\mathcal{P}1$	$\mathcal{P}2$	$\mathcal{P}3$	$Q1$	$Q2$	$Q1$	$Q2$	$Q1$	$Q2$
total # of Bool. queries	228,179	132,277	465,418	20,109	3,697,683	94,840	$18.3 \cdot 10^6$	187,532	$36.5 \cdot 10^6$
max lineage size	2,333	2,333	262	5,545	5,545	28,029	28,029	55,386	55,386
avg lineage size	2.220	2.715	3.745	7.156	3.304	8.617	3.340	8.977	3.346
total # of literals	217,860	217,860	1,742,928	765,572	765,572	3,824,671	3,824,671	7,651,215	7,651,215
total # of active literals	217,860	217,860	1,742,928	138,972	760,572	787,593	3,799,669	1,622,379	7,601,211
max # of inst. of a literal	13	2	1	4	610	7	663	7	693
avg # of inst. of a literal	2.325	1.648	1.0	1.0356	16.0662	1.0376	16.0916	1.0377	16.0981

Table 1. Statistics of the data- and query-sets used. A literal is said to be *active* when it appears in the lineage of at least one query.

(g++ 4.8.4). It uses OpenMP [14] for parallelism and CMinPack [14] for numerical optimization. As a comparison point we used TPDB, the prototype system developed and made available[15] by Dylla and Theobald [16, 17].

MLE Experiment 1. This experiment focuses on parameter learning. We adopt a TI-PDB with known parameters as ground truth and process a fixed set of queries to generate the evidence. We then incrementally update our B-PDB, and observe how well it models the evidence over time. We also run TPDB and compare the results. From now on we denote with $\mathcal{T}$ the parameters of the ground-truth TI-PDB, with Q a fixed set of conjunctive hierarchical queries, with $\mathcal{E}$ the set of marginal probabilities of Q w.r.t. $\mathcal{T}$ (the evidence), and with $\mathcal{H}$ the set of parameters learned by either the B-PDB ($\mathbf{a}$, $\mathbf{b}$) or the TPDB ($\boldsymbol{\theta}$). Following [16], we measure both systems' performances in terms of their mean squared error: $\text{MSE} \stackrel{\text{def}}{=} \frac{1}{|Q|} \sum_{\varphi \in Q} (\mathbb{P}[\varphi|\mathcal{H}] - \mathbb{P}[\varphi|\mathcal{T}])^2$. When the MSE equals zero, the likelihood $\mathbb{P}[\mathcal{E}|\mathcal{H}]$ is maximized. We mimic [16] for the choice of the data- and query-sets; we use the query-sets $\mathcal{P}1$, $\mathcal{P}2$ and $\mathcal{P}3$ as defined in Appendix A of [16], and reproduce the "scalability" experiment over the YAGO2[16] knowledge base. As in [16], the ground-truth $\mathcal{T}$ is set up by annotating each YAGO2 fact with a random probability, uniformly chosen in $[0, 1]$. The properties of the resulting lineage formulas are summarized in Table 1. TPDB provides three parameter learning algorithms, based on direct minimization of the MSE objective function: gradient descent (GD), stochastic gradient descent (SGD), and stochastic gradient descent with per-tuple learning rate (SGD$^+$). We tested all of them, measuring the evolution of the MSE over time. Figures 8a to 8c report our findings. Both the systems were executed in single-threaded mode. The execution time is measured in seconds and does not include planning and grounding time, the MSE is reported in log-scale, so to emphasize the trajectories near their convergence point. Our Bayesian-updating algorithm is denoted as BU. Each point in the plot represents the execution of one iteration of Algorithm 1, over all the available evidence. Over the three experiments we observe a common behavior: BU follows a L-shape trajectory, characterized by a fast-start where the MSE is greatly reduced in few iterations; the algorithm then slows progressively down, as it reaches its steady state. The three gradient-based methods, on the other hand, exhibit a limited improving rate in the first few iterations, followed by a speed-up as soon as they reach a steep sector of the objective function surface. They then slow down as they reach a local minimum of the MSE. GD is consistently the fastest of the three in reaching the fast-converging phase of the trajectory, but the less efficient afterwards. Dylla and Theobald [16] analyze this behavior in great detail. Overall, this experiment shows that the strict semantics of Bayesian updates does not pose a burden on performing MLE efficiently. Additionally, BU offers good MLE performance without requiring the user to fine-tune any parameter (like GD's initial learning rate, for example). Looking at the good performance of

[14] http://www.openmp.org

[15] http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/software/tpdblearn/

[16] http://www.mpi-inf.mpg.de/yago-naga/yago/

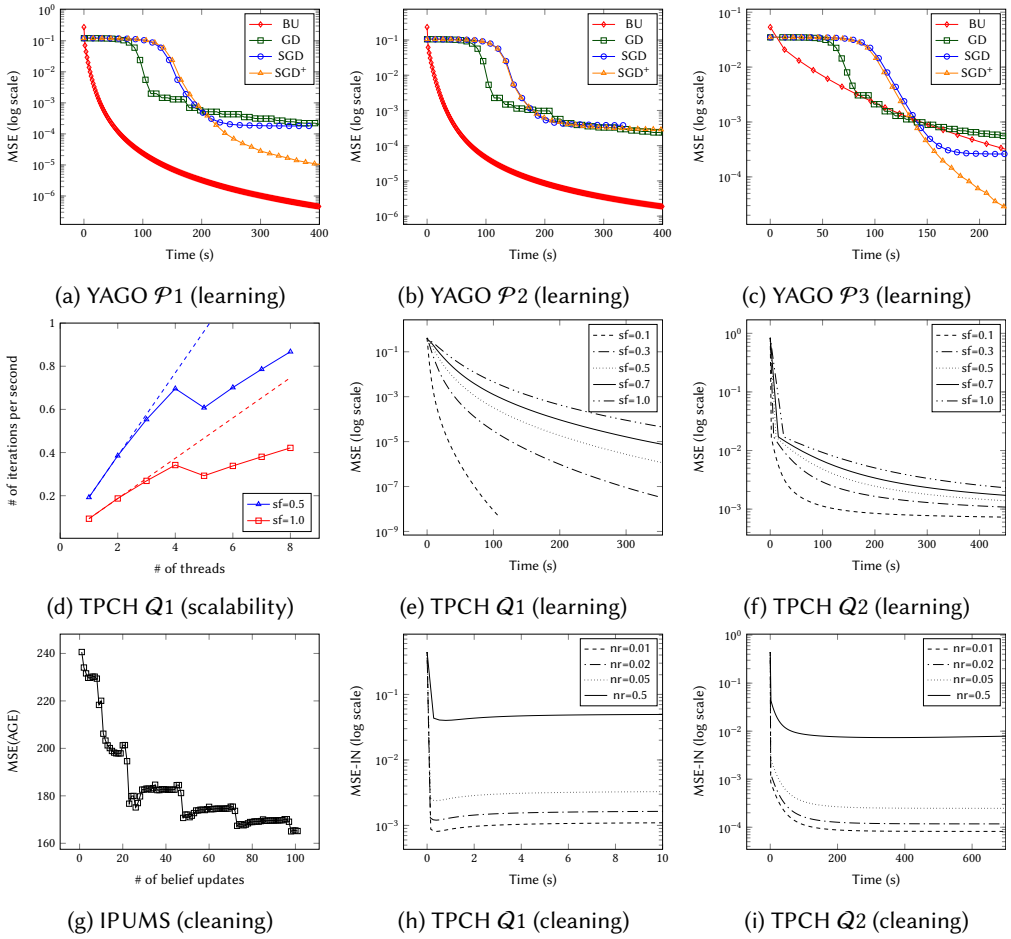


Fig. 8. Experimental results: (a)-(c): MLE Experiment 1. (d)-(f): MLE Experiment 2. (g): Bayesian updating. (h),(i): MLE Experiment 3

SGD$^+$ on $\mathcal{P}3$, we feel it would be interesting to develop a stochastic variant of BU, where only a randomly chosen portion of the evidence is processed at each iteration.

MLE Experiment 2. In this experiment we analyze the behavior of B-PDBs under stress conditions. We use the dbgen utility [59] to generate a set of relations, that we annotate with synthetic probabilities. We use two query-sets, $\mathcal{Q}1$ and $\mathcal{Q}2$. The former consists of queries Q_3, Q_4 and Q_6 from the TPC-H benchmark (this choice mirrors [3, 9]); the latter extends $\mathcal{Q}1$ with 12 additional join/group-by queries, devoid of any selection predicate. By varying the dbgen's scaling factor parameter (sf) between 0.1 and 1.0, we build several instances of $\mathcal{T}$, whose sizes range between 100 MB and 1 GB. Table 1 summarizes the properties of the resulting lineage formulas. We designed this experiment to test several corner-cases in the parameter learning problem: (i) having large lineage formulas in $\mathcal{E}$, (ii) having literals that appear in many formulas, (iii) having a large number of formulas in $\mathcal{E}$. We replicate the same measurements as in Experiment 1, but we run our prototype in multi-threaded mode. Figures 8e and 8f show that the behavior of BU is very consistent over multiple tests, as it follows the usual L-shaped trajectory, drifting towards a local minimum of the MSE. Figure 8d shows the speedup achieved by multi-threading.

MLE Experiment 3. In this experiment we adopt a different metric:

$$\text{MSE-IN} \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n}(\mathbb{P}[x_i|\mathcal{H}] - \mathbb{P}[x_i|\mathcal{T}])^2$$

Our goal is to measure the ability of a B-PDB to rebuild the ground-truth $\mathcal{T}$ by only looking at $\mathcal{E}$. As exemplified in Figure 5, identifying a global maximum of the likelihood does not guarantee the ability of deriving $\mathcal{T}$, as $\mathcal{E}$ may be implicitly ambiguous. One way to circumvent this problem is to polarize the B-PDB's priors towards $\mathcal{T}$. The goal of this experiment is to simulate such process. We repeat Experiment 2 (with sf = 0.1), but we jump-start the BU algorithm by setting up a fraction of the B-PDB's parameters so to mirror $\mathcal{T}$, in a low-entropy configuration ($a + b = 10^6$). The results are shown in Figures 8h and 8i, where nr (noise ratio) denotes the fraction of parameters that are set to random values. Overall we observe that Bayesian updates do not guarantee a steady decrease in MSE-IN, especially when the evidence is too ambiguous (as in $Q1$). On the other hand, we observe that better priors lead to better estimates of $\mathcal{T}$.

9 RELATED WORK

Stoyanovich et al. [57] derive probability distributions for the missing parts of incomplete databases, using the complete parts as evidence. Dylla and Theobald [16] study the problem of deriving the parameters of a TI-PDB from a set of Boolean queries, labeled with their marginal probabilities. They prove the problem is $\#\mathcal{P}$-hard in the general case, and provide a sound criterion to identify problem instances that admit a solution. Rather than computing a maximum-likelihood estimate of the parameters, like we advocate in this paper, they propose to derive them by direct minimization of the mean squared error. Their approach does not consider Bayesian updates. Parameter learning has been proposed in the context of Probabilistic Logic Programming, either by minimizing the mean squared error [28] or by maximum likelihood estimation [29]. It is also a central feature for many knowledge-based model construction (KBMC) frameworks, including Probabilistic Relational Models [41], Markov Logic [53], Multi-entity Bayesian Networks [42] and many others. All the above approaches rely on probabilistic models that are significantly more sophisticated than TI-PDBs, but without the complexity guarantees provided by the dichotomy theorem [11]. Koch and Olteanu [40] were the first to address the problem of conditioning in probabilistic databases. Their work relies on U-databases, while ours focuses on TI-/BI-PDBs and hierarchical queries.

10 CONCLUSIONS AND FUTURE WORK

"Where do the probabilities come from?" is an often-asked question related to probabilistic databases. The approach suggested in this paper is to learn the parameters from query answers. We devise a method for incorporating new evidence in an incremental fashion, by performing belief updates as soon as new query answers are observed. A fundamental ingredient to our approach is the use of Beta/Dirichlet priors: we show how to derive the posterior distribution in closed form and how to update the parameters in a principled way. In the future we propose to generalize our framework as to model continuous domains. We also plan to test alternative inference methods to handle non-hierarchical conjunctive queries [22] and even non-monotone queries [8].

Acknowledgements: The authors would like to thank the anonymous reviewers for their constructive comments. This work was supported by a gift from Oracle and NSF Awards ACI-1640864, CAREER IIS-1750460, and CAREER IIS-1762268. The conclusions and opinions in this work are solely those of the authors and do not represent the views of Oracle or the National Science Foundation.

REFERENCES

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of Databases*. Addison-Wesley. http://webdam.inria.fr/Alice/

[2] Parag Agrawal, Omar Benjelloun, Anish Das Sarma, Chris Hayworth, Shubha U. Nabar, Tomoe Sugihara, and Jennifer Widom. 2006. Trio: A System for Data, Uncertainty, and Lineage. In *VLDB*. http://dl.acm.org/citation.cfm?id=1164231

[3] Lyublena Antova, Thomas Jansen, Christoph Koch, and Dan Olteanu. 2008. Fast and Simple Relational Processing of Uncertain Data. In *ICDE*. https://doi.org/10.1109/ICDE.2008.4497507

[4] Omar Benjelloun, Anish Das Sarma, Alon Y. Halevy, and Jennifer Widom. 2006. ULDBs: Databases with Uncertainty and Lineage. In *VLDB*. http://dl.acm.org/citation.cfm?id=1164209

[5] Jihad Boulos, Nilesh N. Dalvi, Bhushan Mandhani, Shobhit Mathur, Christopher Ré, and Dan Suciu. 2005. MYSTIQ: a system for finding more answers by using probabilities. In *SIGMOD*. https://doi.org/10.1145/1066157.1066277

[6] Peter Buneman, Sanjeev Khanna, and Wang Chiew Tan. 2001. Why and Where: A Characterization of Data Provenance. In *ICDT*. https://doi.org/10.1007/3-540-44503-X_20

[7] Zhuhua Cai, Zografoula Vagena, Luis Leopoldo Perez, Subramanian Arumugam, Peter J. Haas, and Christopher M. Jermaine. 2013. Simulation of database-valued Markov chains using SimSQL. In *SIGMOD*. https://doi.org/10.1145/2463676.2465283

[8] Li Chou, Wolfgang Gatterbauer, and Vibhav Gogate. 2018. Dissociation-Based Oblivious Bounds for Weighted Model Counting. In *UAI*. http://auai.org/uai2018/proceedings/papers/312.pdf

[9] Nilesh N. Dalvi and Dan Suciu. 2004. Efficient Query Evaluation on Probabilistic Databases. In *VLDB*. http://www.vldb.org/conf/2004/RS22P1.PDF

[10] Nilesh N. Dalvi and Dan Suciu. 2007. Management of probabilistic data: foundations and challenges. In *SIGMOD*. https://doi.org/10.1145/1265530.1265531

[11] Nilesh N. Dalvi and Dan Suciu. 2012. The dichotomy of probabilistic inference for unions of conjunctive queries. *J. ACM* 59, 6 (2012), 30. https://doi.org/10.1145/2395116.2395119

[12] Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *JRSS-B* (1977), 1–38.

[13] Maarten Van den Heuvel, Floris Geerts, Wolfgang Gatterbauer, and Martin Theobald. 2018. A General Framework for Anytime Approximation in Probabilistic Databases. In *8th International Workshop on Statistical Relational (StarAI)*. https://arxiv.org/pdf/1806.10078

[14] Frédéric Devernay. 2007. C/C++ Minpack. http://devernay.free.fr/hacks/cminpack/. (2007).

[15] Jennie Duggan and Michael L Brodie. 2015. Hephaestus: Data Reuse for Accelerating Scientific Discovery.. In *CIDR*.

[16] Maximilian Dylla and Martin Theobald. 2014. *Learning Tuple Probabilities in Probabilistic Databases*. Technical Report. Max-Planck-Institut für Informatik. http://pubman.mpdl.mpg.de/pubman/item/escidoc:2028353/component/escidoc:2028364/MPI-I-2014-5-001.pdf

[17] Maximilian Dylla, Martin Theobald, and Iris Miliaraki. 2014. Querying and Learning in Probabilistic Databases. In *Reasoning Web*. https://doi.org/10.1007/978-3-319-10587-1_8

[18] Arthur Erdélyi, Wilhelm Magnus, Fritz Oberhettinger, and Francesco G Tricomi. 1954. *Tables of Integral Transforms: Vol.: 1*. McGraw-Hill Book Company, Incorporated.

[19] Robert Fink, Andrew Hogue, Dan Olteanu, and Swaroop Rath. 2011. SPROUT^2: A squared query engine for uncertain web data. In *SIGMOD*. https://doi.org/10.1145/1989323.1989481

[20] Robert Fink, Jiewen Huang, and Dan Olteanu. 2013. Anytime approximation in probabilistic databases. *VLDB J.* 22, 6 (2013), 823–848. https://doi.org/10.1007/s00778-013-0310-5

[21] Norbert Fuhr and Thomas Rölleke. 1997. A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems. *ACM TOIS* 15, 1 (1997), 32–66. https://doi.org/10.1145/239041.239045

[22] Wolfgang Gatterbauer and Dan Suciu. 2014. Oblivious bounds on the probability of Boolean functions. *ACM TODS* 39, 1 (2014), 5. https://doi.org/10.1145/2532641

[23] Wolfgang Gatterbauer and Dan Suciu. 2015. Approximate Lifted Inference with Probabilistic Databases. *PVLDB* 8, 5 (2015), 629–640. http://www.vldb.org/pvldb/vol8/p629-gatterbauer.pdf

[24] Martin Charles Golumbic, Aviad Mintz, and Udi Rotics. 2006. Factoring and recognition of read-once functions using cographs and normality and the readability of functions associated with partial k-trees. *DAM* 154, 10 (2006), 1465–1477. https://doi.org/10.1016/j.dam.2005.09.016

[25] Todd J. Green, Grigoris Karvounarakis, Zachary G. Ives, and Val Tannen. 2010. Provenance in ORCHESTRA. *IEEE Data Eng. Bull.* 33, 3 (2010), 9–16. http://sites.computer.org/debull/A10sept/green.pdf

[26] Todd J. Green, Gregory Karvounarakis, and Val Tannen. 2007. Provenance semirings. In *PODS*. https://doi.org/10.1145/1265530.1265535

[27] Nitin Gupta, Lucja Kot, Gabriel Bender, Sudip Roy, Johannes Gehrke, and Christoph Koch. 2011. Coordination Through Querying in the Youtopia System. In *SIGMOD*. 4. https://doi.org/10.1145/1989323.1989490

[28] Bernd Gutmann, Angelika Kimmig, Kristian Kersting, and Luc De Raedt. 2008. Parameter Learning in Probabilistic Databases: A Least Squares Approach. In *ECML/PKDD*. 473–488. https://doi.org/10.1007/978-3-540-87479-9_49
[29] Bernd Gutmann, Ingo Thon, and Luc De Raedt. 2011. Learning the Parameters of Probabilistic Logic Programs from Interpretations. In *ECML/PKDD*. 581–596. https://doi.org/10.1007/978-3-642-23780-5_47
[30] HO Hartley. 1958. Maximum likelihood estimation from incomplete data. *Biometrics* 14, 2 (1958), 174–194.
[31] Ralf Herbrich. 2005. *Minimising the Kullback–Leibler divergence*. Technical Report. Microsoft Research.
[32] Jiewen Huang, Lyublena Antova, Christoph Koch, and Dan Olteanu. 2009. MayBMS: A probabilistic database management system. In *SIGMOD*. https://doi.org/10.1145/1559845.1559984
[33] Ravi Jampani, Fei Xu, Mingxi Wu, Luis Leopoldo Perez, Christopher M. Jermaine, and Peter J. Haas. 2008. MCDB: A Monte Carlo approach to managing uncertain data. In *SIGMOD*. https://doi.org/10.1145/1376616.1376686
[34] Shawn R. Jeffery, Michael J. Franklin, and Alon Y. Halevy. 2008. Pay-as-you-go user feedback for dataspace systems. In *SIGMOD*. https://doi.org/10.1145/1376616.1376701
[35] Norman L Johnson, Samuel Kotz, and N Balakrishnan. 1995. Continuous univariate distributions, vol. 2. (1995).
[36] Bhargav Kanagal, Jian Li, and Amol Deshpande. 2011. Sensitivity analysis and explanations for robust query evaluation in probabilistic databases. In *SIGMOD*. https://doi.org/10.1145/1989323.1989411
[37] Paris C Kanellakis and Dina Q Goldin. 1994. Constraint programming and database query languages. In *International Symposium on Theoretical Aspects of Computer Software*. Springer, 96–120.
[38] Richard M. Karp, Michael Luby, and Neal Madras. 1989. Monte-Carlo Approximation Algorithms for Enumeration Problems. *J. Algorithms* 10, 3 (1989), 429–448. https://doi.org/10.1016/0196-6774(89)90038-2
[39] Oliver Kennedy and Christoph Koch. 2010. PIP: A database system for great and small expectations. In *ICDE*. https://doi.org/10.1109/ICDE.2010.5447879
[40] Christoph Koch and Dan Olteanu. 2008. Conditioning probabilistic databases. *PVLDB* 1, 1 (2008), 313–325. http://www.vldb.org/pvldb/1/1453894.pdf
[41] Daphne Koller. 1999. Probabilistic Relational Models. In *ILP-99*. 3–13. https://doi.org/10.1007/3-540-48751-4_1
[42] Kathryn B. Laskey. 2008. MEBN: A language for first-order Bayesian knowledge bases. *Artif. Intell.* 172, 2-3 (2008), 140–178. https://doi.org/10.1016/j.artint.2007.09.006
[43] Aida C. G. Verdugo Lazo and Pushpa N. Rathie. 1978. On the entropy of continuous probability distributions (Corresp.). *IEEE TOIT* 24, 1 (1978), 120–122. https://doi.org/10.1109/TIT.1978.1055832
[44] Yann LeCun and Corinna Cortes. 2010. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. (2010). http://yann.lecun.com/exdb/mnist/
[45] Alexandra Meliou, Wolfgang Gatterbauer, and Dan Suciu. 2011. Reverse Data Management. *PVLDB* 4, 11 (2011), 1490–1493. http://www.vldb.org/pvldb/vol4/p1490-meliou.pdf
[46] Niccolò Meneghetti, Oliver Kennedy, and Wolfgang Gatterbauer. 2017. Beta Probabilistic Databases: A Scalable Approach to Belief Updating and Parameter Learning. In *SIGMOD*. ACM, New York, NY, USA, 573–586. https://doi.org/10.1145/3035918.3064026
[47] Radford M Neal and Geoffrey E Hinton. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*. Springer, 355–368.
[48] Dan Olteanu and Jiewen Huang. 2008. Using OBDDs for Efficient Query Evaluation on Probabilistic Databases. In *SUM*. https://doi.org/10.1007/978-3-540-87993-0_26
[49] Dan Olteanu and Jiewen Huang. 2009. Secondary-storage confidence computation for conjunctive queries with inequalities. In *SIGMOD*. ACM, 389–402. https://doi.org/10.1145/1559845.1559887
[50] Dan Olteanu, Jiewen Huang, and Christoph Koch. 2010. Approximate confidence computation in probabilistic databases. In *ICDE*. https://doi.org/10.1109/ICDE.2010.5447826
[51] David Poole. 1993. Probabilistic Horn Abduction and Bayesian Networks. *Artif. Intell.* 64, 1 (1993), 81–129.
[52] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. 2007. ProbLog: A Probabilistic Prolog and Its Application in Link Discovery. In *IJCAI*. 2462–2467. http://dli.iiit.ac.in/ijcai/IJCAI-2007/PDF/IJCAI07-396.pdf
[53] Matthew Richardson and Pedro M. Domingos. 2006. Markov logic networks. *Machine Learning* 62, 1-2 (2006), 107–136. https://doi.org/10.1007/s10994-006-5833-1
[54] Steven Ruggles, Sarah Flood, Ronald Goeken, Josiah Grover, Erin Meyer, Jose Pacas, and Matthew Sobek. 2018. IPUMS USA: Version 8.0 [dataset]. In *IPUMS*. https://doi.org/10.18128/D010.V8.0
[55] Prithviraj Sen, Amol Deshpande, and Lise Getoor. 2009. PrDB: managing and exploiting rich correlations in probabilistic databases. *VLDB J.* 18, 5 (2009), 1065–1090. https://doi.org/10.1007/s00778-009-0153-2
[56] Sarvjeet Singh, Chris Mayfield, Sagar Mittal, Sunil Prabhakar, Susanne E. Hambrusch, and Rahul Shah. 2008. Orion 2.0: native support for uncertain data. In *SIGMOD*. https://doi.org/10.1145/1376616.1376744
[57] Julia Stoyanovich, Susan B. Davidson, Tova Milo, and Val Tannen. 2011. Deriving probabilistic databases with inference ensembles. In *ICDE*. 303–314. https://doi.org/10.1109/ICDE.2011.5767854

[58] Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. 2011. *Probabilistic Databases*. Morgan & Claypool Publishers.
[59] TPC. 2017. TPC-H Benchmark. http://www.tpc.org/tpch/. (2017).
[60] Ying Yang, Niccolò Meneghetti, Ronny Fehling, Zhen Hua Liu, and Oliver Kennedy. 2015. Lenses: An On-demand Approach to ETL. *PVLDB* 8, 12 (2015), 1578–1589. https://doi.org/10.14778/2824032.2824055

A NOMENCLATURE

Symbol	Meaning
$\mathcal{H}$	Beta Probabilistic DB / Dirichlet Probabilistic Database
$\mathcal{D}$	Tuple-Independent Probabilistic DB / Disjoint-Independent Probabilistic DB
$p[\cdot]$	Probability density function
$\mathbb{P}[\cdot]$	Probability measure
$\langle f(\theta)\rangle_{p[\theta]}$	Expected value of $f(\theta)$ when $\theta \sim p[\cdot]$
$h[\cdot]$	differential entropy
$\mathcal{E}$	Evidence
$\mathcal{B}e(a_i, b_i)$	P.d.f. of a Beta distribution
a, b	Shape parameters of the Beta distribution
$\mathcal{D}ir(\boldsymbol{\alpha})$	P.d.f. of a Dirichlet distribution
$\boldsymbol{\alpha}$	Concentration parameters of the Dirichlet distribution
$B(\cdot)$	(Generalized) Beta function
$\Gamma(\cdot)$	Gamma function
$\psi(\cdot)$	Digamma function
$\psi'(\cdot)$	Trigamma function
w	Possible world
$x_1, \ldots, x_n$	Tuples (TI model) / Blocks (DI model)
$x_{u,1}, \ldots, x_{u,c}$	Tuples in a given block x_u (DI model)
$\theta_1, \ldots, \theta_n$	Tuples' marginal probabilities (TI model)
$\theta_{u,1}, \ldots, \theta_{u,c}$	Tuples' marginal probabilities for a given block x_u (DI model)
$\boldsymbol{\theta}$	Vector $(\theta_1, \ldots, \theta_n)$ (TI model)
$\boldsymbol{\theta}_u$	Vector $(\theta_{u,1}, \ldots, \theta_{u,c})$ (DI model)
$\overline{\theta_i}$	Abbreviation for $(1 - \theta_i)$
$q_1, \ldots, q_k$	Conjunctive queries
$\varphi_1, \ldots, \varphi_k$	Lineage formulas
$\overline{\varphi_j}$	Abbreviation for $\neg\varphi_j$
t_j	Observed frequency of positive answers to φ_j
τ_j	Observed relative freq. of positive answers to φ_j
k	Number of queries
s	Number of samples per query
n	Number of tuples (TI model) / Number of blocks (DI model)
$R, S, T, \ldots$	Relations' names
$X, Y, Z, \ldots$	First-order logic variables
$\mathsf{ADom}(\cdot)$	Active domain
$\mathsf{hvar}(q_j)$	The head variables of query q_j
$\mathsf{evar}(q_j)$	The existential variables of query q_j
$\mathcal{T}$	Ground-truth

Table 2. Nomenclature

B PROOFS

Theorem 4.1 and Theorem 7.2 show how to compute the marginal posteriors for B-PDBS and D-PDBs, respectively. In this section we first provide the proof for Theorem 7.2. Since B-PDBs represent a special case of D-PDBs, obtaining a proof for Theorem 4.1 is straightforward, and will be addressed as a corollary.

Proof of Theorem 7.2. Let $\mathcal{H}$ be a D-PDB and φ be a lineage formula defined over its tuples. Let's denote by $\boldsymbol{\Theta}$ the set of all the latent random vectors in $\mathcal{H}$, i.e. $\boldsymbol{\Theta} \stackrel{\text{def}}{=} \cup_{u=1}^{n} \boldsymbol{\theta}_u$. Our goal is to

prove that

$$p[\boldsymbol{\theta}_u|\varphi,\mathcal{H}] = \sum_{y\in\{\varepsilon,1,..,c\}} \mathbb{P}[x_{u,y}|\varphi,\mathcal{H}]\cdot p[\boldsymbol{\theta}_u|x_{u,y},\mathcal{H}] \tag{47}$$

By definition the u-th marginal of $p[\boldsymbol{\Theta} \mid \varphi,\mathcal{H}]$ is

$$p[\boldsymbol{\theta}_u \mid \varphi,\mathcal{H}] = \int .. \int p[\boldsymbol{\Theta} \mid \varphi,\mathcal{H}]\, d\boldsymbol{\theta}_1 \ldots d\boldsymbol{\theta}_{u-1} d\boldsymbol{\theta}_{u+1} \ldots d\boldsymbol{\theta}_n \tag{48}$$

To be more concise, we denote by $\boldsymbol{\Theta}^*$ the set $\boldsymbol{\Theta} \setminus \{\boldsymbol{\theta}_u\}$, hence

$$p[\boldsymbol{\theta}_u \mid \varphi,\mathcal{H}] = \int p[\boldsymbol{\Theta} \mid \varphi,\mathcal{H}]\, d\boldsymbol{\Theta}^* \tag{49}$$

Applying the Bayes' rule we can write

$$p[\boldsymbol{\theta}_u \mid \varphi,\mathcal{H}] = \int \frac{p[\boldsymbol{\Theta},\varphi \mid \mathcal{H}]}{\mathbb{P}[\varphi \mid \mathcal{H}]}\, d\boldsymbol{\Theta}^* \tag{50}$$

Since event φ is independent from $\mathcal{H}$ given a certain $\boldsymbol{\Theta}$, we have that

$$p[\boldsymbol{\theta}_u \mid \varphi,\mathcal{H}] = \int \frac{\mathbb{P}[\varphi \mid \boldsymbol{\Theta}]\cdot p[\boldsymbol{\Theta} \mid \mathcal{H}]}{\mathbb{P}[\varphi \mid \mathcal{H}]}\, d\boldsymbol{\Theta}^* \tag{51}$$

Since $\mathbb{P}[\varphi \mid \mathcal{H}]$ does not depend on $\boldsymbol{\Theta}$ we can write

$$p[\boldsymbol{\theta}_u \mid \varphi,\mathcal{H}] = \frac{1}{\mathbb{P}[\varphi \mid \mathcal{H}]}\cdot \int \mathbb{P}[\varphi \mid \boldsymbol{\Theta}]\cdot p[\boldsymbol{\Theta} \mid \mathcal{H}]\, d\boldsymbol{\Theta}^* \tag{52}$$

We can expand the content of the integral, noticing that

$$\mathbb{P}[\varphi \mid \boldsymbol{\Theta}] = \sum_{w:w\models\varphi} \mathbb{P}[w \mid \boldsymbol{\Theta}] \tag{53}$$

$$p[\boldsymbol{\Theta} \mid \mathcal{H}] = \prod_{l=1}^{n} p[\boldsymbol{\theta}_l \mid \boldsymbol{\alpha}_l] \tag{54}$$

Hence we can write

$$p[\boldsymbol{\theta}_u \mid \varphi,\mathcal{H}] = \frac{1}{\mathbb{P}[\varphi \mid \mathcal{H}]}\cdot \int \sum_{w:w\models\varphi} \mathbb{P}[w \mid \boldsymbol{\Theta}]\cdot \prod_{l=1}^{n} p[\boldsymbol{\theta}_l \mid \boldsymbol{\alpha}_l]\, d\boldsymbol{\Theta}^* \tag{55}$$

Since $\mathbb{P}[w \mid \boldsymbol{\Theta}] = \prod_{l=1}^{n} \mathbb{P}[w(l) \mid \boldsymbol{\theta}_l]$ and $\mathbb{P}[w(l) \mid \boldsymbol{\theta}_l]$ is a categorical distribution, we can write

$$p[\boldsymbol{\theta}_u \mid \varphi,\mathcal{H}] = \frac{1}{\mathbb{P}[\varphi \mid \mathcal{H}]}\cdot \int \sum_{w:w\models\varphi} \prod_{l=1}^{n} \mathbb{P}[w(l) \mid \boldsymbol{\theta}_l]\cdot p[\boldsymbol{\theta}_l \mid \boldsymbol{\alpha}_l]\, d\boldsymbol{\Theta}^* \tag{56}$$

We now partition the worlds that satisfy φ w.r.t. the value they assign to block x_u; we denote by φ_y the conjunction $\varphi \wedge x_{u,y}$. Notice that if $y' \neq y''$ then $\varphi_{y'}$ and $\varphi_{y''}$ are mutually exclusive.

$$p[\boldsymbol{\theta}_u \mid \varphi,\mathcal{H}] = \frac{1}{\mathbb{P}[\varphi \mid \mathcal{H}]}\cdot \sum_{y\in\{\varepsilon,1,..,c\}} \left[\int \sum_{w:w\models\varphi_y} \prod_{l=1}^{n} \mathbb{P}[w(l) \mid \boldsymbol{\theta}_l]\cdot p[\boldsymbol{\theta}_l \mid \boldsymbol{\alpha}_l]\, d\boldsymbol{\Theta}^* \right] \tag{57}$$

If we denote by L^* the set $\{1,\ldots,n\} \setminus \{u\}$ then we can write

$$p[\boldsymbol{\theta}_u \mid \varphi,\mathcal{H}] = \frac{1}{\mathbb{P}[\varphi \mid \mathcal{H}]}\cdot \sum_{y\in\{\varepsilon,1,..,c\}} \left[\sum_{w:w\models\varphi_y} \mathbb{P}[x_{u,y} \mid \boldsymbol{\theta}_u]\cdot p[\boldsymbol{\theta}_u \mid \boldsymbol{\alpha}_u]\cdot \prod_{l\in L^*} \mathbb{P}[w(l) \mid \boldsymbol{\alpha}_l] \right] \tag{58}$$

Since $x_{u,y}$ and $\boldsymbol{\alpha}_u$ are independent given $\boldsymbol{\theta}_u$, we can write

$$p[\boldsymbol{\theta}_u \mid \varphi, \mathcal{H}] = \frac{1}{\mathbb{P}[\varphi \mid \mathcal{H}]} \cdot \sum_{y \in \{\varepsilon, 1, .., c\}} \left[\sum_{w : w \models \varphi_y} \mathbb{P}[x_{u,y}, \boldsymbol{\theta}_u \mid \boldsymbol{\alpha}_u] \cdot \prod_{l \in L^*} \mathbb{P}[w(l) \mid \boldsymbol{\alpha}_l] \right] \tag{59}$$

Or equivalently

$$p[\boldsymbol{\theta}_u \mid \varphi, \mathcal{H}] = \frac{1}{\mathbb{P}[\varphi \mid \mathcal{H}]} \cdot \sum_{y \in \{\varepsilon, 1, .., c\}} \left[\sum_{w : w \models \varphi_y} \mathbb{P}[\boldsymbol{\theta}_u \mid x_{u,y}, \boldsymbol{\alpha}_u] \cdot p[x_{u,y} \mid \boldsymbol{\alpha}_u] \cdot \prod_{l \in L^*} \mathbb{P}[w(l) \mid \boldsymbol{\alpha}_l] \right] \tag{60}$$

We can simplify to

$$\begin{aligned} p[\boldsymbol{\theta}_u \mid \varphi, \mathcal{H}] &= \frac{1}{\mathbb{P}[\varphi \mid \mathcal{H}]} \cdot \sum_{y \in \{\varepsilon, 1, .., c\}} \left[\sum_{w : w \models \varphi_y} p[\boldsymbol{\theta}_u \mid x_{u,y}, \boldsymbol{\alpha}_u] \cdot \mathbb{P}[w \mid \mathcal{H}] \right] && (61) \\ &= \frac{1}{\mathbb{P}[\varphi \mid \mathcal{H}]} \cdot \sum_{y \in \{\varepsilon, 1, .., c\}} p[\boldsymbol{\theta}_u \mid x_{u,y}, \boldsymbol{\alpha}_u] \cdot \mathbb{P}[\varphi_y \mid \mathcal{H}] && (62) \\ &= \sum_{y \in \{\varepsilon, 1, .., c\}} p[\boldsymbol{\theta}_u \mid x_{u,y}, \boldsymbol{\alpha}_u] \cdot \mathbb{P}[x_{u,y} \mid \varphi, \mathcal{H}] && (63) \end{aligned}$$

This proves Equation (47). To prove Theorem 7.2 it is sufficient to notice that $p[\boldsymbol{\theta}_u \mid x_{u,y}, \boldsymbol{\alpha}_u]$ is the standard posterior of a Dirichlet-Categorical compound distribution. Hence $p[\boldsymbol{\theta}_u \mid x_{u,y}, \boldsymbol{\alpha}_u]$ is a Dirichlet distribution with the same parameters as the prior $\mathcal{D}ir(\boldsymbol{\alpha}_u)$ except for parameter $\alpha_{u,y}$, that is increased by one. In other words: $p[\boldsymbol{\theta}_u \mid x_{u,y}, \boldsymbol{\alpha}_u] = \mathcal{D}ir(\boldsymbol{\alpha}_u + \mathbf{e}_y)$. □

Corollary. We can use Equation (47) to prove Theorem 4.1. First we observe that a B-PDB is just a D-PDB where each block contains exactly one tuple. If we set $\boldsymbol{\alpha}_u = (a_u, b_u)$ then $\mathcal{D}ir(\boldsymbol{\alpha}_u) = \mathcal{B}e(a_u, b_u)$. Therefore Equation (47) can be rewritten as the sum of two terms

$$p[\boldsymbol{\theta}_u | \varphi, \mathcal{H}] = \mathbb{P}[x_{u,1} | \varphi, \mathcal{H}] \cdot p[\boldsymbol{\theta}_u | x_{u,1}, \mathcal{H}] + \mathbb{P}[x_{u,\varepsilon} | \varphi, \mathcal{H}] \cdot p[\boldsymbol{\theta}_u | x_{u,\varepsilon}, \mathcal{H}] \tag{64}$$

Or, using B-PDBs notation,

$$p[\theta_i | \varphi, \mathcal{H}] = \mathbb{P}[x_i | \varphi, \mathcal{H}] \cdot p[\theta_i | x_i, \mathcal{H}] + \mathbb{P}[\overline{x_i} | \varphi, \mathcal{H}] \cdot p[\theta_i | \overline{x_i}, \mathcal{H}] \tag{65}$$

Where $p[\theta_i | x_i, \mathcal{H}]$ and $p[\theta_i | \overline{x_i}, \mathcal{H}]$ are just the standard posterior probabilities for a Beta-Bernoulli compound distribution; hence $p[\theta_i | x_i, \mathcal{H}] = \mathcal{B}e(a_i + 1, b_i)$ and $p[\theta_i | \overline{x_i}, \mathcal{H}] = \mathcal{B}e(a_i, b_i + 1)$. □

C REPRESENTING COMPLEX EVENTS FROM INDEPENDENT VARIABLES

We show next that any correlation between variables can be captured by general Boolean functions over *independent events* only (not just *disjoint-independent events*). We repeat here the argument that was presented in this form in [22], but is well known folklore.

It is known from Poole's independent choice logic [51] that arbitrary correlations between events can be composed from *disjoint-independent events* only. A disjoint-independent event is represented by a non-Boolean independent random variable y which takes either of k values $\mathbf{v} = \langle v_1, \ldots, v_k \rangle$ with respective probabilities $\mathbf{q} = \langle q_1, \ldots, q_k \rangle$ and $\sum_i q_i = 1$. Poole writes such a "disjoint declaration" as $y([v_1 : q_1, \ldots, v_k : q_k])$.

In turn, any k disjoint events can be represented starting from $k - 1$ independent Boolean variables $\mathbf{z} = \langle z_1, \ldots, z_{k-1} \rangle$ and probabilities $\mathbb{P}[\mathbf{z}] = \langle q_1, \frac{q_2}{\bar{q}_1}, \frac{q_3}{\bar{q}_1 \bar{q}_2}, \ldots, \frac{q_{k-1}}{\bar{q}_1 \cdots \bar{q}_{k-2}} \rangle$, by assigning the

disjoint-independent event variable y its value v_i whenever event A_i is true, with A_i defined as:

$$\begin{aligned}(y = v_1) &\equiv A_1 \coloneqq z_1\\ (y = v_2) &\equiv A_2 \coloneqq \bar{z}_1 z_2\\ &\vdots\\ (y = v_{k-1}) &\equiv A_{k_1} \coloneqq z_1 \ldots \bar{z}_{k-2} z_{k-1}\\ (y = v_k) &\equiv A_k \coloneqq \bar{z}_1 \ldots \bar{z}_{k-2} \bar{z}_{k-1}\ .\end{aligned}$$

For example, a disjoint-independent event $y(v_1 : \frac{1}{5}, v_2 : \frac{1}{2}, v_3 : \frac{1}{5}, v_4 : \frac{1}{10})$ can be represented with three independent Boolean variables $\mathbf{z} = (z_1, z_2, z_3)$ and $\mathbb{P}[\mathbf{z}] = (\frac{1}{5}, \frac{5}{8}, \frac{2}{3})$.

It follows that *arbitrary correlations between events* can be modeled starting from *independent Boolean random variables* alone. For example, two *complex events* A and B with $\mathbb{P}[A] = \mathbb{P}[B] = q$ and varying correlation can be represented as *composed events* $A \coloneqq z_1 z_2 \vee z_3 \vee z_4$ and $B \coloneqq \bar{z}_1 z_2 \vee z_3 \vee z_5$ over the *primitive events* $\mathbf{z}$ with varying probabilities $\mathbb{P}[\mathbf{z}]$. Events A and B become identical for $\mathbb{P}[\mathbf{z}] = (0, 0, q, 0, 0)$, independent for $\mathbb{P}[\mathbf{z}] = (0, 0, 0, q, q)$, and disjoint for $\mathbb{P}[\mathbf{z}] = (0.5, q, 0, 0, 0)$ with $q \leq 0.5$.

D BETA-MAYBMS

MayBMS [3] is a state-of-the-art probabilistic database built on top of PostgreSQL 8.3.3. We developed Beta-MayBMS, an extension of MayBMS that supports Bayesian updating. In this Section we briefly highlight the differences between the two systems. The first difference lies in the data model: rather than storing straight probabilities for each record, Beta-MayBMS associates each block of records with a Dirichlet prior. Let's assume a Beta-MayBMS instance contains the following deterministic relation:

```
# select * from edu;
person | level | hyp
--------+-------+-----
Ada    | phd   |   7
Ada    | ms    |   2
Ada    | bs    |   1
Bob    | phd   |   3
Bob    | ms    |   3
Bob    | bs    |   4
Carl   | ms    |   3
Carl   | bs    |   7
(8 rows)
```

As in MayBMS, the above can be turned into a block-independent probabilistic relation by running a `repair key` statement:

```
# create table eduP as repair key person in edu weight by hyp;
# select * from eduP;
 person | level | hyp | _hyp | _v0 | _d0 | _p0
--------+-------+-----+------+-----+-----+-----
 Ada    | phd   |   7 |    7 |   1 |   5 | 0.7
 Ada    | ms    |   2 |    2 |   1 |   8 | 0.2
 Ada    | bs    |   1 |    1 |   1 |   3 | 0.1
 Bob    | phd   |   3 |    3 |   2 |   6 | 0.3
 Bob    | ms    |   3 |    3 |   2 |   4 | 0.3
 Bob    | bs    |   4 |    4 |   2 |   1 | 0.4
```

```
 Carl   | ms    |   3 |    3 |   3 |   7 | 0.3
 Carl   | bs    |   7 |    7 |   3 |   2 | 0.7
(8 rows)
```

Similarly to MayBMS, each block is associated with a unique identifier (_v0) and each tuple is associated with a probability and an identifier (_d0 and _p0). Beta-MayBMS uses an additional column (_hyp) to store the parameters of the Dirichlet prior. The user can compute a Bayesian update by running a `posterior of` statement:

```
# posterior of eduP given (select * from eduP where level='phd') is not empty;
 _v0 | _d0 |       _hyp        |        _p0
-----+-----+-------------------+--------------------
   1 |   5 |  7.56053753051323 |  0.716805615782907
   1 |   8 |  1.99066547181542 |  0.188732637538479
   1 |   3 | 0.996339265815391 | 0.0944617466786136
   2 |   6 |  3.06707966333329 |  0.306833717581594
   2 |   4 |  2.97023626766526 |   0.29714540740456
   2 |   1 |   3.9585857173191 |  0.396020875013846
(6 rows)
```

The following statement computes the above relation and updates the relevant hyper-parameters.

```
# update eduP given evidence (select * from eduP where level='phd') is not empty;
```

Internally, Beta-MayBMS reuses the MayBMS query-engine for computing conditional probabilities in the form $\mathbb{P}[x_i|\varphi_j, \mathcal{H}]$, through query rewriting. New internal functions were added for computing the actual Bayesian update and identify the new set of hyper-parameters that minimize the relative entropy w.r.t. the posterior. Beta-MayBMS does not use CP-plans. All the querying capabilities of MayBMS are maintained, without performance loss. For example, the following query will return the probability of someone having a PhD:

```
# select conf() from (select * from eduP where level='phd') sq;
 conf
------
 0.79
(1 row)
```

Received December 2017; revised August 2018; accepted September 2018