Beta Probabilistic Databases: A Scalable Approach to Belief Updating and Parameter Learning

Niccolò Meneghetti
HPE Vertica
niccolom@buffalo.edu

Oliver Kennedy
University at Buffalo
okennedy@buffalo.edu

Wolfgang Gatterbauer
Carnegie Mellon University
gatt@cmu.edu

ABSTRACT

Tuple-independent probabilistic databases (TI-PDBs) handle uncertainty by annotating each tuple with a probability parameter; when the user submits a query, the database derives the marginal probabilities of each output-tuple, assuming input-tuples are statistically independent. While query processing in TI-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 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. We use this model to provide the following key contributions: (i) we show how to scalably compute the posterior densities of the parameters given new evidence; (ii) we study the complexity of performing Bayesian belief updates, devising efficient algorithms for tractable classes of queries; (iii) we propose a soft-EM algorithm for computing maximum-likelihood estimates of the parameters; (iv) we show how to embed the proposed algorithms into a standard relational engine; (v) we support our conclusions with extensive experimental results.

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 community [3, 6, 8, 9, 16, 22, 29, 30, 35, 43, 45, 46], and querying uncertain data is relatively well understood. However, deriving the probability distribution behind a probabilistic database by observing its answers can be significantly harder [13, 14].

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

ISBN 978-1-4503-4197-4/17/05…$15.00

DOI: http://dx.doi.org/10.1145/3035918.3064026

In this paper, we address the challenge of building a probabilistic database from a noisy, indirect signal. Specifically, we propose a new model for probabilistic databases called *Beta-Probabilistic Databases (B-PDBs)* that enables principled approaches for deriving or updating the database's distribution. The information used to update or derive the B-PDB may be indirect; a B-PDB can incorporate any information that can be expressed in terms of the probability of a positive boolean expression holding over the database. This information may also be noisy — the information incorporated into a B-PDB may itself be sampled, as in a poll or a vote, or crowdsourced. Most importantly, B-PDBs are completely backwards compatible with the widely used Tuple-Independent model [6, 9, 10, 18] for probabilistic databases (TI-PDBs), allowing us to freely leverage query processing techniques developed for TI-PDBs, like pRA [18], Monte Carlo simulations [8, 30, 34], anytime approximations [17], dissociations [19, 20], lineage-based methods [23] and more. B-PDBs enable a more powerful form of pay-as-you-go feedback on query results [31, 48], as well as probabilistic forms of in-database constraint programming [24, 33] and of virtualized experiments [12].

Example 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 not her employer. Given a table of employers* `R(name,emp)` *and locations* `L(emp,loc)`*, an equivalent assertion is given in the following existential query:*

```
exists(select * from R natural join L
    where R.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 potentially make this assertion untrustworthy. First, ambiguous phrasing might leave the aggregator's natural-language information extraction uncertain. Secondly, information obtained from different sources may be contradictory. B-PDBs make it possible to account for both of these factors when updating the underlying database.

In the Tuple-Independent 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 powerset 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: $\mathbb{P}[q] = \sum_{w \models q} \mathbb{P}[w]$.

Example 1 (continued) *We could use a TI-PDB to encode noisy knowledge about employment histories:*

R^p			
name	emp	*tid*	$\boldsymbol{\theta}$
Ada	*HP*	x_1	.6
Ada	*IBM*	x_2	.6
Bob	*HP*	x_3	.5

If we want to find out whether the person named Ada ever had an employer, we run the following Boolean query

```
exists(select * from R^p where name='Ada')    (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 we are given (i) a TI-PDB $\mathcal{D}$ whose parameters are hidden, (ii) a set of Boolean queries $\mathcal{Q} \stackrel{\text{def}}{=} \{q_1, \ldots, q_k\}$, and (iii) a finite set of query-answers sampled from $\mathbb{P}[q]$, for each q in $\mathcal{Q}$. We denote by $\mathcal{E}$ (the "evidence") the whole set of samples and we assume each sample is drawn independently from all the others. This paper focuses mainly on two problems:

1. **Belief updates**: given an initial hypothesis about the hidden parameters of $\mathcal{D}$, we show how to refine such hypothesis as to incorporate the evidence $\mathcal{E}$.
2. **Parameter learning**: we show how to derive a new hypothesis from scratch, relying only on the given evidence.

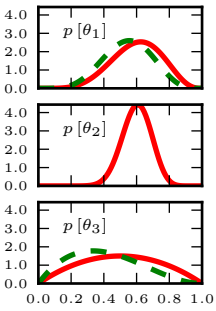
Without lack of generality, we assume each query is observed exactly s times, and we denote by τ the fraction of positive answers. Therefore, we model the evidence $\mathcal{E}$ as a mapping that associates each query to its observed relative frequency of positive answers: $\mathcal{E} \stackrel{\text{def}}{=} \{(q_1, \tau_1), (q_2, \tau_2), \ldots, (q_k, \tau_k)\}$.

Example 1 (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.*

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 R^p, but we want to improve our knowledge about Ada's work history. In order to do so, we submit the query "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 R^p (whose initial value is 0.6) as 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.

[1] The probabilities follow from $1 - [(1 - 0.6) \cdot (1 - 0.6)] = 0.84$. Also notice that we adopt here a closed-world assumption as is common with TI-PDBs: any missing tuple is assumed to have probability 0.

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



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

name	emp	*tid*	a	b	$\langle\boldsymbol{\theta}\rangle$	$h[\boldsymbol{\theta}]$
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	$\langle\boldsymbol{\theta}\rangle$	$h[\boldsymbol{\theta}]$
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

Figure 1: A simple B-PDB with three tuples, *before* (solid red) and *after* (dashed green) observing the answer "no" to the query `exists(select * from R where emp='HP')` .

Parameter learning is about using the experts' opinion to build a new probabilistic model from scratch. For example: if we are told that the query

$$\mathbf{q} = \texttt{select emp from R where name='Ada'}$$

should return the answer $\{\text{HP}, \text{IBM}\}$ ("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 (R^p) exhibits the desired marginal probabilities for $\mathbf{q}$. This is achieved either when $(\theta_1, \theta_2) = (0.4, 0.8)$ or when $(\theta_1, \theta_2) = (0.8, 0.4)$.[3]

The key idea behind B-PDBs is to model the parameters $\{\theta_1, \ldots, \theta_n\}$ as Beta-distributed latent random variables. This simple expedient allows us to (i) model *both* our current estimate of a probability and its 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 1 (first table) a simple B-PDB, consisting of a single relation, generalizing the TI-relation R^p we introduced in Example 1. 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:

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

Here $B(\cdot, \cdot)$ denotes the Beta function, which serves as a normalizing factor[4] . The three solid-red plots on the left in Figure 1 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 in the B-PDB represents a poll where the query "has Ada ever worked for HP?" has received 6 positive answers and 4 negative ones. Similarly, the second row can be seen as a poll where the query "has Ada ever worked for IBM?" has received 18 positive

[3] Since $0.32 = 0.8 \cdot 0.4$ and $0.12 = (1 - 0.8)(1 - 0.4)$. Notice that $\mathbf{q}$ 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 $\{\text{HP}, \text{IBM}\}$, and another one to verify whether the answer is $\emptyset$.

[4] Chapter 25 of [32] is a good introduction to Beta distributions.

votes and 12 negative ones. While the relative frequency of positive answers is the same for both the polls (0.6), the second poll should be considered more informative, as it involves more votes (30 against 10). Consistently with this intuition, the plots of $p[\theta_1]$ and $p[\theta_2]$ have both a peak on .6, but the former exhibits a higher entropy than the latter.

In B-PDBs the marginal probability of a tuple x_i is determined by the *expected value* of the random variable θ_i:

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

From now on we denote with $\langle\theta_i\rangle$ the expected value of θ_i. It is well known [32] that the integral in Equation (3) admits the following closed solution: $\mathbb{P}[x_i] = \langle\theta_i\rangle = a_i/(a_i + b_i)$. 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 [10, 17, 20, 34, 41, 42] can be readily applied to B-PDBs. To do so it is sufficient to convert the B-PDB into a TI-DB by computing the expectation $\langle\theta_i\rangle$ for each and every tuple in the database, in polynomial time. Therefore, B-PDBs are a conservative generalization of TI-PDBs that add a principled way to update the parameters, which is the main focus of our paper. The first table of Figure 1 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 R^{p} introduced in Example 1.

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

$$h[\theta_i] \stackrel{\text{def}}{=} -\int p[\theta_i] \cdot \ln(p[\theta_i]) d\theta_i$$

The above integral admits a well-known [39, 15] closed form[5]. Intuitively, the smaller $h[\theta_i]$ is the higher is the confidence of the estimate of $\mathbb{P}[x_i]$. For readers' convenience Figure 1 shows the differential entropies in the column $h[\boldsymbol{\theta}]$[6].

In the following sections we will describe extensively how B-PDBs support belief updates. We introduce the topic here with a simple example. The second table in Figure 1 shows the effect of performing a belief update to incorporate the observation of a single, negative answer to the Boolean query `exists(select * from R 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 left, in dashed green. Notice that the update has a greater impact on $p[\theta_3]$ rather than on $p[\theta_1]$, consistently with the fact that $h[\theta_1] < h[\theta_3]$. In the general case, belief updates may involve thousands of queries, and affect the parameters of a B-PDB in a non-trivial way.

In the remainder of this paper, we study (i) belief updates (Section 4) and (ii) parameter learning (Section 5) in great detail; we show how to perform both when we observe answers to an arbitrary set of conjunctive, self-join-free queries.

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. **Soft Expectation Maximization.** We devise a soft-EM algorithm for computing the maximum likelihood estimate of the parameters $\{\theta_1, \ldots, \theta_n\}$ w.r.t. some given queries' results.
3. **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) and synthetic (TCP-H) data sets, annotated with probabilities.

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:

$$\mathbf{q}(Z) = \exists X\ \exists Y\ R(X,Y) \wedge S(X,Z) \tag{4}$$

We denote by $\texttt{hvar}(\mathbf{q})$ the set of free ("head-") variables of $\mathbf{q}$, and by $\texttt{evar}(\mathbf{q})$ 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 of the possible grounding of the free variables to values in their *active domain* [2]. 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 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* [5, 7, 23] 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 $\texttt{hvar}(\mathbf{q}') = \{X\}$ 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 $q = q' \wedge q''$

[5]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.

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

- $\Phi_{\mathcal{D}}(q) = x_i$, when q is a ground atom of tuple x_i[7]

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: $\mathbf{q}(Z) :- R(X,Y), S(X,Z)$.

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 1 (continued) *The query* $\mathbf{q}$ *from Equation* (4) *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* R *and* S *are defined as follows:*

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

S		
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:*

$$q_1 :- R(X,Y), S(X, eng). \qquad q_2 :- R(X,Y), S(X, ita).$$

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:[8]

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

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

- If $P = R$ then $\Phi_{\mathcal{D}}(P) = x$ where x identifies the grounded tuple in R.
- If $P = P' \bowtie P''$ then $\Phi_{\mathcal{D}}(P) = \Phi_{\mathcal{D}}(P') \wedge \Phi_{\mathcal{D}}(P'')$.
- 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 1 (continued) *Both the following query plans*

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

compute the correct answer for query $\mathbf{q}$ *from Equation* (4)*, but they produce different lineage expressions:*

P'	
lng	$\Phi_{\mathcal{D}}$
eng	$((x_1 \vee x_2)\, x_4) \vee x_3x_5$
ita	x_3x_6

P''	
lng	$\Phi_{\mathcal{D}}$
eng	$x_1x_4 \vee x_2x_4 \vee x_3x_5$
ita	x_3x_6

Notice that all the lineage expressions produced by P' *are read-once and logically equivalent to the corresponding lineage expressions of* $\mathbf{q}$ *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) [18], a generalization of positive relational algebra that consists of three probabilistic operators: independent projection (π^{p}), independent join ($\bowtie^{\mathsf{p}}$) and selection (σ). These operators differ from standard relational algebra

[7] In this paper "atoms" are atomic first-order logic formulas. For example, the query from Equation (4) contains two atoms, $R(X,Y)$ and $S(X,Y)$. An atom is *ground* when it has no variables: $R(\texttt{'Ada'}, \texttt{'HP'})$.

[8] W.l.o.g. we assume the query to consist of only variables and don't write the constants. Selections can always be directly pushed into the database before executing the query.

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 $\mathsf{score}(P) = \theta_i$
- If $P = P' \bowtie^{\mathsf{p}} P''$ then $\mathsf{score}(P) = \mathsf{score}(P') \cdot \mathsf{score}(P'')$
- If $P = \sigma(P')$ then $\mathsf{score}(P) = \mathsf{score}(P')$
- If $P = \pi^{\mathsf{p}}_{\emptyset}(P')$ then

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

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:

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

Let's assume P' and P'' are two plans answering the Boolean queries q' and q'', respectively, and $\mathbb{P}[q'|\mathcal{D}] = \mathsf{score}(P')$ and $\mathbb{P}[q''|\mathcal{D}] = \mathsf{score}(P'')$. Notice that $\mathbb{P}[q' \wedge q''|\mathcal{D}] = \mathsf{score}(P' \bowtie^{\mathsf{p}} P'')$ 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 $\mathbb{P}[q' \vee q''|\mathcal{D}] = \mathsf{score}(\pi^{\mathsf{p}}_{\emptyset}(P))$ 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 only 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 1 *[9, 10, 21, 41] 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 anwering* $\mathbf{q}$ *is #P-hard.*

Example 1 (continued) *We can turn the relations R and S into a TI-PDB by annotating each tuple with a probability, that we store in a dedicated column named* $\boldsymbol{\theta}$*.*

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

S^{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:*

$$P' = \pi^{p}_{-X}(\pi^{p}_{-Y}(R^{p}) \bowtie^{p} S^{p}) \qquad P'' = \pi^{p}_{-XY}(R^{p} \bowtie^{p} S^{p}) \quad (5)$$

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 only when dealing with hierarchical queries. Dalvi and Suciu [9] developed a well-known algorithm to identify safe plans for hierarchical queries. Other techniques, like [8, 30, 34, 17], must be used to answer non-hierarchical queries.

3. BETA PROBABILISTIC DATABASES

In this section we introduce *Beta probabilistic databases* (B-PDBs), our new 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 $\mathcal{B}e(a_i, b_i)$ 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:

$$p[\theta_i|\mathcal{H}] \stackrel{\text{def}}{=} \mathcal{B}e(a_i, b_i) \quad (6)$$

$$p[\boldsymbol{\theta}|\mathcal{H}] \stackrel{\text{def}}{=} \prod_{i=1}^{n} p[\theta_i|\mathcal{H}], \quad (7)$$

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*:

$$\theta_i \sim \mathrm{Beta}(a_i, b_i) \qquad x_i \sim \mathrm{Bernoulli}(\theta_i)$$

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 Boolean queries:

$$\mathbb{P}[x_i|\mathcal{H}] \stackrel{\text{def}}{=} \int_0^1 \theta_i \cdot \mathcal{B}e(a_i, b_i)\, d\theta_i \quad (8)$$

$$\mathbb{P}[w|\mathcal{H}] \stackrel{\text{def}}{=} \prod_{i=1}^{n} \mathbb{P}[x_i|\mathcal{H}]^{w[i]} \cdot \overline{\mathbb{P}[x_i|\mathcal{H}]}^{\overline{w[i]}} \quad (9)$$

$$\mathbb{P}[\varphi|\mathcal{H}] \stackrel{\text{def}}{=} \sum_{w:\ w \models \varphi} \mathbb{P}[w|\mathcal{H}] \quad (10)$$

Equations 8, 9 and 10 denote, respectively, the probability of a literal, the probability of a possible world, and the probability of a Boolean query. Notice that φ may represent the lineage of the 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 table at the top of Figure 1 with the probabilistic relation R^{p} discussed in Example 1). Notice that the marginal probability of x_i can be computed as [32]

$$\mathbb{P}[x_i|\mathcal{H}] = \int_0^1 \theta_i \cdot \mathcal{B}e(a_i, b_i)\, d\theta_i = \langle\theta_i\rangle_{\mathcal{H}} = \tfrac{a_i}{a_i+b_i} \qquad (11)$$

From Equation (11) it follows immediately that 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 [18], Monte Carlo simulations [8, 30, 34], anytime approximations [17], dissociations [19, 20], lineage-based methods [23] and many others.

Given two queries 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':

$$\mathbb{P}[\varphi|\varphi', \mathcal{H}] \stackrel{\text{def}}{=} \tfrac{\mathbb{P}[\varphi\wedge\varphi'|\mathcal{H}]}{\mathbb{P}[\varphi'|\mathcal{H}]} \qquad (12)$$

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

$$p[\boldsymbol{\theta}|\varphi, \mathcal{H}] \stackrel{\text{def}}{=} \frac{p[\boldsymbol{\theta}, \varphi|\mathcal{H}]}{\mathbb{P}[\varphi|\mathcal{H}]} = \frac{\mathbb{P}[\varphi|\boldsymbol{\theta}] \cdot p[\boldsymbol{\theta}|\mathcal{H}]}{\mathbb{P}[\varphi|\mathcal{H}]} \qquad (13)$$

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 Multiple independent observations

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 2 depicts model $\mathcal{H}^s$ using plate notation, and compares it with the model induced by TI-DBs. Within a model $\mathcal{H}^s$, we denote by $x_{\ell,i}$ and $\theta_{\ell,i}$ 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 $0 \le t \le s$, we denote by $\mathbb{P}[\varphi^t|\mathcal{H}^s]$ the probability of observing a set of s independent possible worlds from $\mathcal{H}$ where φ is satisfied exactly t times:

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

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

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

Notice that the above formula generalizes Equation (13). 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 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. Given a set of k distinct Boolean expressions $\{\varphi_1, \ldots, \varphi_k\}$ and k integers $\{t_1, \ldots, t_k\}$ between 0 and s, we 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

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

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

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

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

- $\mathbb{P}[\varphi_1 \wedge \varphi_2|\mathcal{H}] = \langle\theta_1\rangle_{\mathcal{H}} \cdot \langle\theta_2\rangle_{\mathcal{H}}$
- $\mathbb{P}[\varphi_1|\varphi_2, \mathcal{H}] = (\langle\theta_1\rangle_{\mathcal{H}} \cdot \langle\theta_2\rangle_{\mathcal{H}}) \cdot (\langle\theta_1\rangle_{\mathcal{H}} \otimes \langle\theta_2\rangle_{\mathcal{H}})^{-1}$
- $\mathbb{P}[\varphi_1|\mathcal{H}^2] = 2 \cdot (\langle\theta_1\rangle_{\mathcal{H}} \cdot \langle\theta_2\rangle_{\mathcal{H}}) \cdot (\langle\theta_1\rangle_{\mathcal{H}} \otimes \langle\theta_2\rangle_{\mathcal{H}})$
- $\mathbb{P}[\varphi_1, \varphi_2|\mathcal{H}^{1,2}] = (\langle\theta_1\rangle_{\mathcal{H}} \cdot \langle\theta_2\rangle_{\mathcal{H}}) \cdot (\langle\theta_1\rangle_{\mathcal{H}} \otimes \langle\theta_2\rangle_{\mathcal{H}})$

From Equation (11) it is straightforward to derive the following Lemma:

Lemma 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:*

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 $\mathcal{D}$.*

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 1 (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 prior $p[\theta_{(\cdot,\cdot,\cdot)}|\hat{\mathcal{H}}^{s,k}]$. Belief updating is discussed in Section 4.*

Definition 2 (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.*

4. BELIEF UPDATING

The goal of belief updating is to adjust the parameters $\mathbf{a}$ and $\mathbf{b}$ so to incorporate some new, previously unseen, evidence. As an analog, consider how one updates a Beta distribution directly. A single piece of evidence (a coin flip or "vote") increments either the a or b parameter by 1. Observe that the updating process for simple Beta distributions depends not only on the evidence itself (heads or tails), but also the weight of that evidence (a single unbiased sample). For example, if the evidence had consisted of two identical coin flips, we would increment the parameter by 2. We begin with a simple case of belief updates where the evidence consists of one vote ($s = 1$) regarding a single output tuple ($k = 1$). In the trivial case where feedback is applied directly to a ground atom x_i, the reader may see that our approach degenerates to incrementing either a_i or b_i.

We then parallelize this approach in two dimensions: We allow a single belief update to simultaneously learn from multiple votes ($s > 1$) about multiple tuples ($k > 1$). Specifically, if $\mathcal{H}$ denotes the current state of the 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 posterior $p[\boldsymbol{\theta}|\varphi, \mathcal{H}]$ and the prior $p[\boldsymbol{\theta}|\hat{\mathcal{H}}]$. Since every tuple of $\hat{\mathcal{H}}$ is independent from the others, it is sufficient to minimize the relative entropy between $p[\theta_i|\varphi, \mathcal{H}]$ and $p[\theta_i|\hat{\mathcal{H}}]$ for each and every tuple. The first step in this direction is to compute the marginal posterior $p[\theta_i|\varphi, \mathcal{H}]$. The following theorem addresses this specific problem.

Theorem 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 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. Consequently, the posterior of a Beta-Bernoulli compound distribution is guaranteed to be Beta-distributed. 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:

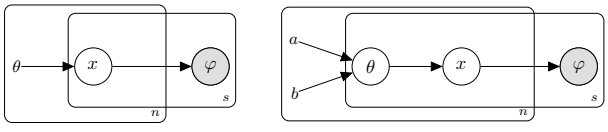


Figure 2: Comparison between a regular TI-PDB (left) and a B-PDB (right) when $k = 1$, using plate notation. TI-PDBs associate each tuple x_i with a single Boolean, Bernoulli-distributed random variable, whose probability mass function depends on a single parameter (θ_i). B-PDBs associate each tuple with *two* random variables: θ_i, which is $[0, 1]$-valued and Beta-distributed with hyper-parameters (a_i, b_i), and x_i, which is Boolean-valued and Bernoulli-distributed. For both TI-PDBs and B-PDBs the evidence consists of observed query answers, that here are modeled by the observable variable φ.

Corollary 1 *In a B-PDB $\mathcal{H}$ the conjugancy property holds if and only if the random variable x_i is fully observable from the evidence:*

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

Corollary 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 $\#\mathcal{P}$-complete 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 $\mathcal{H} \to \hat{\mathcal{H}}$ that minimizes the relative entropy between $p[\theta_i|\varphi, \mathcal{H}]$ and the marginal prior $p[\theta_i|\hat{\mathcal{H}}] \stackrel{\text{def}}{=} \mathcal{B}e(\hat{a}_i, \hat{b}_i)$. We denote such measure with $\mathsf{KL}_i^{\mathsf{div}}$:

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

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 $\mathsf{KL}_{\mathsf{div}}$ 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. $\mathbb{P}[\theta|\varphi, \mathcal{H}]$. This well-known criterion is formalized in the following definition and justified in Proposition 1[9]:

Definition 3 *We denote by $\mathsf{bfit}(a_i, b_i, \varphi)$ the pair of parameters $(\hat{a}_i^*, \hat{b}_i^*)$ satisfying the following two equations:*

$$\begin{cases} \int_0^1 \mathcal{B}e(\hat{a}_i^*, \hat{b}_i^*) \ln \theta_i \, d\theta_i = \int_0^1 p[\theta_i|\varphi, \mathcal{H}] \ln \theta_i \, d\theta_i \\ \int_0^1 \mathcal{B}e(\hat{a}_i^*, \hat{b}_i^*) \ln \overline{\theta_i} \, d\theta_i = \int_0^1 p[\theta_i|\varphi, \mathcal{H}] \ln \overline{\theta_i} \, d\theta_i \end{cases} \tag{14}$$

or, in terms of expectations:

$$\begin{cases} \langle \ln \theta_i \rangle_{\mathcal{B}e(\hat{a}_i^*, \hat{b}_i^*)} = \langle \ln \theta_i \rangle_{p[\theta_i|\varphi, \mathcal{H}]} \\ \langle \ln \overline{\theta}_i \rangle_{\mathcal{B}e(\hat{a}_i^*, \hat{b}_i^*)} = \langle \ln \overline{\theta}_i \rangle_{p[\theta_i|\varphi, \mathcal{H}]} \end{cases} \tag{15}$$

Proposition 1 *When $(\hat{a}_i, \hat{b}_i) = (\hat{a}_i^*, \hat{b}_i^*) = \mathsf{bfit}(a_i, b_i, \varphi)$ the relative entropy between the posterior $p[\theta_i|\varphi, \mathcal{H}]$ and the Beta distribution $\mathbb{P}[\theta_i|\hat{\mathcal{H}}]$ is minimized.*

Proof. The relative entropy between $p[\theta_i|\varphi, \mathcal{H}]$ and $p[\theta_i|\hat{\mathcal{H}}]$

[9] A similar, more general result is proved in [28] for all the distributions in the exponential family.

can be expressed as follows:

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

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

$$\nabla\mathsf{KL}_i^{\mathsf{div}} = \begin{bmatrix}\frac{\partial\mathsf{KL}_i^{\mathsf{div}}}{\partial\hat{a}_i}\\[2ex] \frac{\partial\mathsf{KL}_i^{\mathsf{div}}}{\partial\hat{b}_i}\end{bmatrix} = \begin{bmatrix}\langle\ln\theta_i\rangle_{p[\theta_i|\hat{\mathcal{H}}]} - \langle\ln\theta_i\rangle_{p[\theta_i|\varphi,\mathcal{H}]}\\[2ex] \langle\ln\overline{\theta_i}\rangle_{p[\theta_i|\hat{\mathcal{H}}]} - \langle\ln\overline{\theta_i}\rangle_{p[\theta_i|\varphi,\mathcal{H}]}\end{bmatrix}$$

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

$$\begin{aligned}
\mathrm{Hess}[\mathsf{KL}_i^{\mathsf{div}}] &= \nabla\nabla\mathsf{KL}_i^{\mathsf{div}} \\
&= \begin{bmatrix}\psi'(\hat{a}_i)-\psi'(\hat{a}_i+\hat{b}_i) & -\psi'(\hat{a}_i+\hat{b}_i)\\[2ex] -\psi'(\hat{a}_i+\hat{b}_i) & \psi'(\hat{b}_i)-\psi'(\hat{a}_i+\hat{b}_i)\end{bmatrix}
\end{aligned}$$

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 $\mathsf{KL}^{\mathsf{div}}$, is simply the sum of the tuple-wise KL divergences:

$$\mathsf{KL}^{\mathsf{div}} = \int\mathinner{..}\int p[\boldsymbol{\theta}|\varphi,\mathcal{H}]\cdot\ln\left(\frac{p[\boldsymbol{\theta}|\varphi,\mathcal{H}]}{p[\boldsymbol{\theta}|\hat{\mathcal{H}}]}\right)d\boldsymbol{\theta} = \sum_{i=1}^{n}\mathsf{KL}_i^{\mathsf{div}}$$

4.2 General case

First 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

$$p[\theta_{(\cdot,i)}|\hat{\mathcal{H}}^s] = \prod_{\ell=1}^{s} p[\theta_{\ell,i}|\hat{\mathcal{H}}] \tag{16}$$

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

$$p[\theta_{(\cdot,i)}|\varphi^t,\mathcal{H}^s] = \prod_{\ell=1}^{t} p[\theta_{\ell,i}|\varphi,\mathcal{H}]\cdot\prod_{\ell=t+1}^{s} p[\theta_{\ell,i}|\overline{\varphi},\mathcal{H}] \tag{17}$$

where $p[\theta_{\ell,i}|\varphi,\mathcal{H}]$ and $p[\theta_{\ell,i}|\overline{\varphi},\mathcal{H}]$ are computed according to Theorem 1. We now redefine $\mathsf{KL}_i^{\mathsf{div}}$ as the relative entropy between $p[\theta_{(\cdot,i)}|\varphi^t,\mathcal{H}^s]$ and $p[\theta_{(\cdot,i)}|\hat{\mathcal{H}}^s]$:

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

Definition 4 *We denote by* $\mathsf{bfit}(a_i,b_i,\{\varphi^t\},s)$ *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 2 *When* $(\hat{a}_i,\hat{b}_i)=(\hat{a}_i^*,\hat{b}_i^*)=\mathsf{bfit}(a_i,b_i,\{\varphi^t\},s)$ *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.

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

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

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

□

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

$$p[\theta_{(\cdot,\cdot,i)}|\hat{\mathcal{H}}^{s,k}] = \prod_{j=1}^{k} p[\theta_{(j,\cdot,i)}|\hat{\mathcal{H}}^s] \tag{18}$$

$$p[\theta_{(\cdot,\cdot,i)}|\mathcal{E},\mathcal{H}^{s,k}] = \prod_{j=1}^{k} p[\theta_{(j,\cdot,i)}|\varphi_j^{t_j},\mathcal{H}^s] \tag{19}$$

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 $p[\theta_{(\cdot,\cdot,i)}|\hat{\mathcal{H}}^{s,k}]$:

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

Definition 5 *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 3 *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 Proposition 3 mimics the one of Proposition 2. For the sake of conciseness we omit it. Before introducing our belief update algorithm, we observe that the equations from Definition 5 can be rewritten as follows:

$$\begin{cases} [\psi(\hat{a}_i^*) - \psi(\hat{a}_i^* + \hat{b}_i^*)] - [\psi(a_i) - \psi(a_i + b_i)] = \frac{r}{a_i} - \frac{1}{a_i + b_i} \\ [\psi(\hat{b}_i^*) - \psi(\hat{a}_i^* + \hat{b}_i^*)] - [\psi(b_i) - \psi(a_i + b_i)] = \frac{\overline{r}}{b_i} - \frac{1}{a_i + b_i} \end{cases} \tag{20}$$

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. From now on we denote by $\mathsf{bu}(a_i, b_i, r)$ the values $(\hat{a}^*, \hat{b}^*)$ that satisfy Equation (20).

We finally introduce Algorithm 1, that exploits Propositions 1 to 3 to perform s belief updates, in response to arbitrary evidence $\mathcal{E}$.

Algorithm 1: Belief Update

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 **for** $i \in \{1, \ldots, n\}$ **do**
4 $\quad r_i \leftarrow \sum_{j=1}^{k} \frac{1}{k}(\tau_j \mathbb{P}[x_i|\varphi_j, \mathcal{H}] + \overline{\tau_j}\mathbb{P}[x_i|\overline{\varphi_j}, \mathcal{H}])$
5 $\quad (\hat{a}_i^*, \hat{b}_i^*) \leftarrow \mathsf{bu}(a_i, b_i, r_i)$
6 **for** $i \in \{1, \ldots, n\}$ **do**
7 $\quad a_i \leftarrow \hat{a}_i^*$
8 $\quad b_i \leftarrow \hat{b}_i^*$

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 both in terms of queries and observed relative frequencies, 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.

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 observation that belief updates can only increase the likelihood $\mathbb{P}[\mathcal{E}|\mathcal{H}^{s,k}]$; it is immediate to derive a soft-EM [11, 27] algorithm that performs repeated belief updates until convergence. First we show how a single belief update $\mathcal{H} \rightarrow \hat{\mathcal{H}}^*$ affects $\mathbb{P}[\mathcal{E}|\mathcal{H}^{s,k}]$:

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

Notice that the pair $(\mathbf{a}^*, \mathbf{b}^*)$ from Definition 5 is the value of $\hat{\mathcal{H}}^{s,k}$ that maximizes the quantity

$$\langle \ln p[\mathcal{E}, \theta_{(\cdot,\cdot,\cdot)}|\hat{\mathcal{H}}^{s,k}]\rangle_{\ln p[\boldsymbol{\theta}|\mathcal{E},\mathcal{H}^{s,k}]}$$

which is a lower bound of the log-likelihood $\ln \mathbb{P}[\mathcal{E}|\hat{\mathcal{H}}^{s,k}]$. Therefore, it is possible to apply the considerations from [40] 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 $\quad\quad (\hat{a}_i^*, \hat{b}_i^*) \leftarrow \mathsf{bu}(a_i, b_i, r_i)$
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 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 3 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 the likelihood function may have several local maximums and even several global ones (as in Figure 3). The only guarantee offered by Algorithm 2 is to converge to a local maximum.

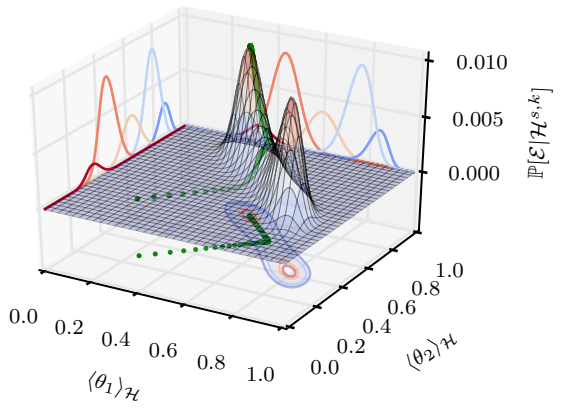


Figure 3: Convergence of Algorithm 2 towards a maximum likelihood estimate of (θ_1, θ_2). In general, the EM algorithm can converge towards a local optimum.

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 probability. The following Theorem states that the dichotomy identified by [10] (see Lemma 1) also applies to the computation of conditional probabilities in the form $\mathbb{P}[x_i|\varphi_j, \mathcal{H}]$.

Theorem 2 *Let φ_j represents 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. The first assertion is proven by observing that

$$\mathbb{P}[x_i|\varphi_j, \mathcal{H}] = \mathbb{P}[\varphi_j|x_i, \mathcal{H}] \cdot \mathbb{P}[x_i|\mathcal{H}]/\mathbb{P}[\varphi_j|\mathcal{H}] \qquad (21)$$

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 (21) takes polynomial time. We prove the second assertion by reduction: 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 (21), it is immediate to obtain

$$\mathbb{P}[\varphi_j|\mathcal{H}] = \mathbb{P}[x_i|\mathcal{H}] \cdot \mathbb{P}[(\varphi_j|_{x_i})|\mathcal{H}]/\mathbb{P}[x_i|\varphi_j, \mathcal{H}] \qquad (22)$$

Since Equation (22) holds for any literal in $\{x_1, \ldots, x_n\}$, we can apply it $n-1$ times and obtain:

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

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. □

From Lemma 1 and Theorem 2 it follows immediately that computing a single Bayesian update, to incorporate the answer to a hierarchical query, takes polynomial time in datasize. 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 (21). A plan performing such computation is called "*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:

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

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}, consisiting 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

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

Intuitively, R^{p} is obtained from R_0^{p} by projecting-away `tid`. R^{cp} associates each tuple x of R^{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 $\pi_{\mathbf{B}}^{\mathsf{c}}$ to the pair $(R^{\mathsf{p}}, R^{\mathsf{cp}})$, we obtain a pair of relations $(Q^{\mathsf{p}}, Q^{\mathsf{cp}})$, defined as follows:

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

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

Where $\Pi_{\mathsf{agg}}(\cdot)$ denotes the aggregate product and $\mathsf{cpexp} \stackrel{\text{def}}{=} 1 - (\overline{Q^{\mathsf{p}}.\mathtt{p}} \cdot \overline{R^{\mathsf{cp}}.\mathtt{cp}/R^{\mathsf{p}}.\mathtt{p}})$. 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:

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

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 $\mathbf{A}_i = \mathtt{hvar}(R_i^{\mathsf{p}}) \setminus \{\mathtt{p}\}$. 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 equivalent to the following RA plan:

$$Q^{\mathsf{p}}(\mathbf{A}, \mathtt{p}) \stackrel{\text{def}}{=} \pi_{\mathbf{A},(\Pi_{i=1}^m R_i^{\mathsf{p}}.\mathtt{p})}[\bowtie [R_1^{\mathsf{p}}, .., R_m^{\mathsf{p}}]]$$

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

$$Q^{\mathsf{cp}}(\mathbf{A}, \mathtt{cp}, \mathtt{lt}) \stackrel{\text{def}}{=} \uplus_{i=1}^m V_i$$

Here $\mathsf{cpexp} \stackrel{\text{def}}{=} (Q^{\mathsf{p}}.\mathtt{p} \cdot R_i^{\mathsf{cp}}.\mathtt{cp}/R_i^{\mathsf{p}}.\mathtt{p})$. 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 of the procedure by Dalvi and Suciu [9] for constructing safe plans. 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 $\mathit{evar}(\mathbf{q}) = \emptyset$ **then**
  return $CP(R) \bowtie^{\mathsf{c}} CP(S) \bowtie^{\mathsf{c}} \ldots$
else if $\mathbf{q} :- \mathbf{q}', \mathbf{q}''$ *and* $\mathit{evar}(\mathbf{q}') \cap \mathit{evar}(\mathbf{q}'') = \emptyset$ **then**
  return SafeCpPlan($\mathbf{q}'$) $\bowtie^{\mathsf{c}}$ SafeCpPlan($\mathbf{q}''$)
else if $X \in \mathit{evar}(\mathbf{q})$ *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 1 (continued) *If* $\mathbf{q}$ *denotes the hierarchical query from Equation* (4)*, then SafeCpPlan(*$\mathbf{q}$*) returns the plan:*

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

Notice that Q^{p} *is equivalent to the TI-relation returned by plan* P' *in Equation* (5)*.*

7. EXPERIMENTS

The goal of this section is to analyze the asymptotic behavior of Bayesian updates, and to experimentally verify what we formally proved in Section 5. Our second goal is to compare our approach against the existing literature on parameter learning in P-DBs, especially [13, 14].

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[10], a system developed by Dylla and Theobald [13, 14], and compare the results. From now on we denote with $\mathcal{T}$ the parameters of the ground-truth TI-PDB, with $\mathcal{Q}$ a fixed set of conjunctive hierarchical queries, with $\mathcal{E}$ the set of marginal probabilities of $\mathcal{Q}$ 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 [13], we measure both systems' performances in terms of their mean squared error: $\mathsf{MSE} \stackrel{\text{def}}{=} \frac{1}{|\mathcal{Q}|} \sum_{\varphi \in \mathcal{Q}} (\mathbb{P}[\varphi|\mathcal{H}] - \mathbb{P}[\varphi|\mathcal{T}])^2$. When the MSE equals zero, the likelihood $\mathbb{P}[\mathcal{E}|\mathcal{H}]$ is maximized. We mimic [13] 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 [13], and reproduce the "scalability" experiment over the YAGO2[11] knowledge base. As in [13], 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 4a to 4c report our findings. Both systems were executed in single-threaded mode. The execution time is measured in seconds, while 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 [13] 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 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.

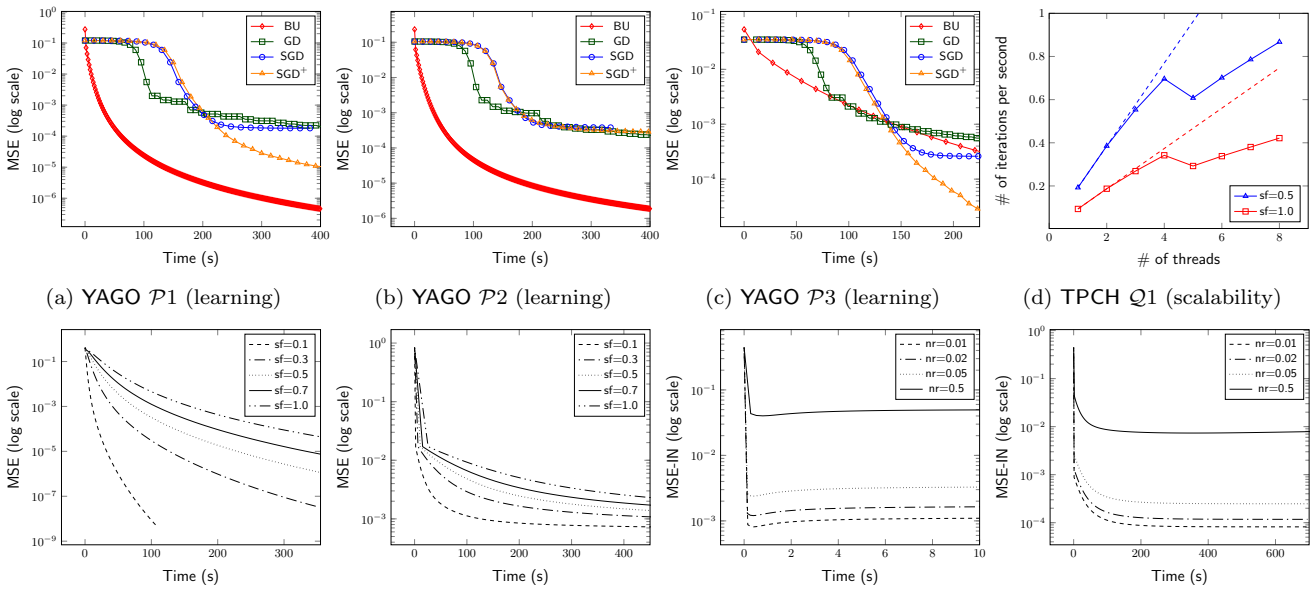
Experiment 2. In this experiment we analyze the behavior of B-PDBs under stress conditions. We use the `dbgen` utility [1] to generate a set of relations, that we annotate with synthetic probabilities. We use two query-sets, $\mathcal{Q}1$ and $\mathcal{Q}2$. The former consists of queries Q_3, Q_4 and Q_6 from the TPC-H benchmark (this choice mirrors [4, 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 4e and 4f 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 4d shows the speedup achieved by multi-threading.

Experiment 3. In this experiment we adopt a different metric: $\mathsf{MSE\text{-}IN} \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} (\mathbb{P}[x_i|\mathcal{H}] - \mathbb{P}[x_i|\mathcal{T}])^2$. 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 3, 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 $\mathsf{sf} = 0.1$), but we jump-start the BU algorithm by setting up a fraction of the B-PDB's parameters so to

[10] http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/software/tpdblearn/
[11] http://www.mpi-inf.mpg.de/yago-naga/yago/

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	$\mathcal{P}1$	$\mathcal{P}2$	$\mathcal{P}3$	$\mathcal{Q}1$	$\mathcal{Q}2$	$\mathcal{Q}1$	$\mathcal{Q}2$	$\mathcal{Q}1$	$\mathcal{Q}2$
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.



(a) YAGO $\mathcal{P}1$ (learning) (b) YAGO $\mathcal{P}2$ (learning) (c) YAGO $\mathcal{P}3$ (learning) (d) TPCH $\mathcal{Q}1$ (scalability)

(e) TPCH $\mathcal{Q}1$ (learning) (f) TPCH $\mathcal{Q}2$ (learning) (g) TPCH $\mathcal{Q}1$ (cleaning) (h) TPCH $\mathcal{Q}2$ (cleaning)

Figure 4: Experimental results: (a)-(c): Experiment 1. (d)-(f): Experiment 2. (g),(h): Experiment 3

mirror $\mathcal{T}$, in a low-entropy configuration ($a+b=10^6$). The results are shown in Figures 4g and 4h, 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 $\mathcal{Q}1$). On the other hand, we observe that better priors lead to better estimates of $\mathcal{T}$.

8. RELATED WORK

Stoyanovich et al. [47] derive probability distributions for the missing parts of incomplete databases, using the complete parts as evidence. Dylla and Theobald [13] 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 [25] or by maximum likelihood estimation [26]. It is also a central feature for many knowledge-based model construction (KBMC) frameworks, including Probabilistic Relational Models [37], Markov Logic [44], Multi-entity Bayesian Networks [38] 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 [10]. Koch and Olteanu [36] were the first to address the problem of conditioning in probabilistic databases. Their work relies on U-databases, while ours focuses on TI-PDBs and hierarchical queries.

9. 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 sampled 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 distributions as 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 work by dropping the assumption on tuple independence. We also plan to test alternative inference methods to handle non-hierarchical queries, such as [19].

Acknowledgements: We like to thank our reviewers for their helpful feedback. This work was supported by a gift from Oracle, NPS Award #N00244-16-1-0022, and NSF Awards SaTC-1409551 and CAREER IIS-1553547. The conclusions and opinions in this work are solely those of the authors and do not represent the views of Oracle, The Naval Postgraduate School, or the National Science Foundation.

10. REFERENCES

[1] TPC-H benchmark. http://www.tpc.org/tpch/.

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

[3] P. Agrawal, O. Benjelloun, A. D. Sarma, C. Hayworth, S. U. Nabar, T. Sugihara, and J. Widom. Trio: A system for data, uncertainty, and lineage. In *VLDB*, 2006.

[4] L. Antova, T. Jansen, C. Koch, and D. Olteanu. Fast and simple relational processing of uncertain data. In *ICDE*, 2008.

[5] O. Benjelloun, A. D. Sarma, A. Y. Halevy, and J. Widom. ULDBs: Databases with uncertainty and lineage. In *VLDB*, 2006.

[6] J. Boulos, N. N. Dalvi, B. Mandhani, S. Mathur, C. Ré, and D. Suciu. MYSTIQ: a system for finding more answers by using probabilities. In *SIGMOD*, 2005.

[7] P. Buneman, S. Khanna, and W. C. Tan. Why and where: A characterization of data provenance. In *ICDT*, 2001.

[8] Z. Cai, Z. Vagena, L. L. Perez, S. Arumugam, P. J. Haas, and C. M. Jermaine. Simulation of database-valued Markov chains using SimSQL. In *SIGMOD*, 2013.

[9] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *VLDB*, 2004.

[10] N. N. Dalvi and D. Suciu. The dichotomy of probabilistic inference for unions of conjunctive queries. *J. ACM*, 59(6):30, 2012.

[11] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *JRSS-B*, pages 1–38, 1977.

[12] J. Duggan and M. L. Brodie. Hephaestus: Data reuse for accelerating scientific discovery. In *CIDR*, 2015.

[13] M. Dylla and M. Theobald. Learning tuple probabilities in probabilistic databases. Technical report, Max-Planck-Institut für Informatik, 2014.

[14] M. Dylla, M. Theobald, and I. Miliaraki. Querying and learning in probabilistic databases. In *Reasoning Web.*, 2014.

[15] A. Erdélyi, W. Magnus, F. Oberhettinger, and F. G. Tricomi. *Tables of Integral Transforms: Vol.: 1*. McGraw-Hill Book Company, Incorporated, 1954.

[16] R. Fink, A. Hogue, D. Olteanu, and S. Rath. SPROUT^2: A squared query engine for uncertain web data. In *SIGMOD*, 2011.

[17] R. Fink, J. Huang, and D. Olteanu. Anytime approximation in probabilistic databases. *VLDB J.*, 22(6):823–848, 2013.

[18] N. Fuhr and T. Rölleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM TOIS*, 15(1):32–66, 1997.

[19] W. Gatterbauer and D. Suciu. Oblivious bounds on the probability of Boolean functions. *ACM TODS*, 39(1):5, 2014.

[20] W. Gatterbauer and D. Suciu. Approximate lifted inference with probabilistic databases. *PVLDB*, 8(5):629–640, 2015.

[21] M. C. Golumbic, A. Mintz, and U. Rotics. Factoring and recognition of read-once functions using cographs and normality and the readability of functions associated with partial k-trees. *DAM*, 154(10):1465–1477, 2006.

[22] T. J. Green, G. Karvounarakis, Z. G. Ives, and V. Tannen. Provenance in ORCHESTRA. *IEEE Data Eng. Bull.*, 33(3):9–16, 2010.

[23] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *PODS*, 2007.

[24] N. Gupta, L. Kot, G. Bender, S. Roy, J. Gehrke, and C. Koch. Coordination through querying in the Youtopia system. In *SIGMOD*, 2011.

[25] B. Gutmann, A. Kimmig, K. Kersting, and L. D. Raedt. Parameter learning in probabilistic databases: A least squares approach. In *ECML/PKDD*, pages 473–488, 2008.

[26] B. Gutmann, I. Thon, and L. D. Raedt. Learning the parameters of probabilistic logic programs from interpretations. In *ECML/PKDD*, pages 581–596, 2011.

[27] H. Hartley. Maximum likelihood estimation from incomplete data. *Biometrics*, 14(2):174–194, 1958.

[28] R. Herbrich. Minimising the Kullback–Leibler divergence. Technical report, Microsoft Research, 2005.

[29] J. Huang, L. Antova, C. Koch, and D. Olteanu. MayBMS: A probabilistic database management system. In *SIGMOD*, 2009.

[30] R. Jampani, F. Xu, M. Wu, L. L. Perez, C. M. Jermaine, and P. J. Haas. MCDB: A Monte Carlo approach to managing uncertain data. In *SIGMOD*, 2008.

[31] S. R. Jeffery, M. J. Franklin, and A. Y. Halevy. Pay-as-you-go user feedback for dataspace systems. In *SIGMOD*, 2008.

[32] N. L. Johnson, S. Kotz, and N. Balakrishnan. Continuous univariate distributions, vol. 2, 1995.

[33] P. C. Kanellakis and D. Q. Goldin. Constraint programming and database query languages. In *International Symposium on Theoretical Aspects of Computer Software*, pages 96–120. Springer, 1994.

[34] R. M. Karp, M. Luby, and N. Madras. Monte-Carlo approximation algorithms for enumeration problems. *J. Algorithms*, 10(3):429–448, 1989.

[35] O. Kennedy and C. Koch. PIP: A database system for great and small expectations. In *ICDE*, 2010.

[36] C. Koch and D. Olteanu. Conditioning probabilistic databases. *PVLDB*, 1(1):313–325, 2008.

[37] D. Koller. Probabilistic relational models. In *ILP-99*, pages 3–13, 1999.

[38] K. B. Laskey. MEBN: A language for first-order bayesian knowledge bases. *Artif. Intell.*, 172(2-3):140–178, 2008.

[39] A. C. G. V. Lazo and P. N. Rathie. On the entropy of continuous probability distributions (corresp.). *IEEE TOIT*, 24(1):120–122, 1978.

[40] R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.

[41] D. Olteanu and J. Huang. Using OBDDs for efficient query evaluation on probabilistic databases. In *SUM*, 2008.

[42] D. Olteanu, J. Huang, and C. Koch. Approximate confidence computation in probabilistic databases. In *ICDE*, 2010.

[43] L. D. Raedt, A. Kimmig, and H. Toivonen. ProbLog: A probabilistic prolog and its application in link discovery. In *IJCAI*, pages 2462–2467, 2007.

[44] M. Richardson and P. M. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

[45] P. Sen, A. Deshpande, and L. Getoor. PrDB: managing and exploiting rich correlations in probabilistic databases. *VLDB J.*, 18(5):1065–1090, 2009.

[46] S. Singh, C. Mayfield, S. Mittal, S. Prabhakar, S. E. Hambrusch, and R. Shah. Orion 2.0: native support for uncertain data. In *SIGMOD*, 2008.

[47] J. Stoyanovich, S. B. Davidson, T. Milo, and V. Tannen. Deriving probabilistic databases with inference ensembles. In *ICDE*, pages 303–314, 2011.

[48] Y. Yang, N. Meneghetti, R. Fehling, Z. H. Liu, and O. Kennedy. Lenses: An on-demand approach to ETL. *PVLDB*, 8(12):1578–1589, 2015.

APPENDIX

A. NOMENCLATURE

Symbol	Meaning
$\mathcal{H}$	Beta Probabilistic DB
$\mathcal{D}$	Tuple-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	Parameters of the Beta distribution
$B(\cdot)$	Beta function
$\Gamma(\cdot)$	Gamma function
$\psi(\cdot)$	Digamma function
$\psi'(\cdot)$	Trigamma function
w	Possible world
$x_1, \ldots, x_n$	Tuples / Boolean random variables
$\theta_1, \ldots, \theta_n$	Tuples' marginal probabilities
$\boldsymbol{\theta}$	Vector $(\theta_1, \ldots, \theta_n)$
$\overline{\theta_i}$	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
$R, S, T, \ldots$	Relations' names
$X, Y, Z, \ldots$	First-order logic variables
$\mathsf{ADom}(\cdot)$	Active domain
$\mathtt{hvar}(q_j)$	The head variables of query q_j
$\mathtt{evar}(q_j)$	The existential variables of query q_j
$\mathcal{T}$	Ground-truth

Table 2: Nomenclature

B. PROOFS

Proof of Theorem 1. From the Bayes' rule we have that

$$p[\boldsymbol{\theta}|\varphi, \mathcal{H}] = \frac{p[\boldsymbol{\theta}, \varphi|\mathcal{H}]}{\mathbb{P}[\varphi|\mathcal{H}]} = \frac{\mathbb{P}[\varphi|\boldsymbol{\theta}] \cdot p[\boldsymbol{\theta}|\mathcal{H}]}{\mathbb{P}[\varphi|\mathcal{H}]}$$

Where

$$\mathbb{P}[\varphi|\boldsymbol{\theta}] = \sum_{w:w\models\varphi} \mathbb{P}[w|\boldsymbol{\theta}] = \sum_{w:w\models\varphi} \left[\prod_{i=1}^{n} \theta_i^{w[i]} \cdot \overline{\theta_i}^{\overline{w[i]}}\right]$$

$$p[\boldsymbol{\theta}|\mathcal{H}] = \prod_{i=1}^{n} p[\theta_i|\mathcal{H}] = \prod_{i=1}^{n} \frac{\theta_i^{(a_i-1)} \cdot \overline{\theta_i}^{(b_i-1)}}{B(a_i, b_i)}$$

$$\mathbb{P}[\varphi|\mathcal{H}] = \sum_{w:w\models\varphi} \left[\prod_{i=1}^{n} \left(\frac{a_i}{a_i+b_i}\right)^{w[i]} \cdot \left(\frac{b_i}{a_i+b_i}\right)^{\overline{w[i]}}\right]$$

Hence

$$p[\boldsymbol{\theta}|\varphi, \mathcal{H}] = \frac{\mathbb{P}[\varphi|\boldsymbol{\theta}] \cdot p[\boldsymbol{\theta}|\mathcal{H}]}{\mathbb{P}[\varphi|\mathcal{H}]} = \tfrac{1}{\mathbb{P}[\varphi|\mathcal{H}]} \cdot \sum_{w:w\models\varphi} (\mathbb{P}[w|\boldsymbol{\theta}] \cdot p[\boldsymbol{\theta}|\mathcal{H}])$$

First we want to prove that

$$\mathbb{P}[w|\boldsymbol{\theta}] \cdot p[\boldsymbol{\theta}|\mathcal{H}] = \mathbb{P}[w|\mathcal{H}] \cdot \prod_{i=1}^{n} \mathcal{B}e(a_i + w[i], b_i + \overline{w[i]})$$

Expanding $\mathbb{P}[w|\boldsymbol{\theta}]$ and $p[\boldsymbol{\theta}|\mathcal{H}]$ in the LHS we obtain

$$\mathbb{P}[w|\boldsymbol{\theta}] \cdot p[\boldsymbol{\theta}|\mathcal{H}] = \prod_{i=1}^{n} \theta_i^{w[i]} \cdot \overline{\theta_i}^{\overline{w[i]}} \cdot \prod_{i=1}^{n} \frac{\theta_i^{(a_i-1)} \cdot \overline{\theta_i}^{(b_i-1)}}{B(a_i, b_i)}$$

$$= \prod_{i=1}^{n} \frac{\theta_i^{(a_i+w[i])-1} \cdot \overline{\theta_i}^{(b_i+\overline{w[i]})-1}}{B(a_i, b_i)}$$

$$= \prod_{i=1}^{n} \mathcal{B}e(a_i + w[i], b_i + \overline{w[i]}) \cdot \frac{B(a_i + w[i], b_i + \overline{w[i]})}{B(a_i, b_i)}$$

$$= \prod_{i=1}^{n} \mathcal{B}e(a_i + w[i], b_i + \overline{w[i]}) \cdot \frac{(a_i)^{w[i]} \cdot (b_i)^{\overline{w[i]}} \cdot \Gamma(a_i + b_i)}{\Gamma(a_i + b_i + 1)}$$

$$= \prod_{i=1}^{n} \mathcal{B}e(a_i + w[i], b_i + \overline{w[i]}) \cdot \frac{(a_i)^{w[i]} \cdot (b_i)^{\overline{w[i]}}}{a_i + b_i}$$

$$= \prod_{i=1}^{n} \mathcal{B}e(a_i + w[i], b_i + \overline{w[i]}) \cdot \left(\frac{a_i}{a_i+b_i}\right)^{w[i]} \cdot \left(\frac{b_i}{a_i+b_i}\right)^{\overline{w[i]}}$$

$$= \mathbb{P}[w|\mathcal{H}] \cdot \prod_{i=1}^{n} \mathcal{B}e(a_i + w[i], b_i + \overline{w[i]})$$

Therefore we can express the posterior $p[\boldsymbol{\theta}|\varphi, \mathcal{H}]$ as follows:

$$p[\boldsymbol{\theta}|\varphi, \mathcal{H}] = \tfrac{1}{\mathbb{P}[\varphi|\mathcal{H}]} \cdot \sum_{w:w\models\varphi} \mathbb{P}[w|\mathcal{H}] \cdot \prod_{i=1}^{n} \mathcal{B}e(a_i + w[i], b_i + \overline{w[i]})$$

$$= \sum_{w:w\models\varphi} \mathbb{P}[w|\varphi, \mathcal{H}] \cdot \left(\prod_{i=1}^{n} \mathcal{B}e(a_i + w[i], b_i + \overline{w[i]})\right)$$

Notice that $\sum_{w:w\models\varphi} \mathbb{P}[w|\varphi, \mathcal{H}] = 1$, therefore the posterior $p[\boldsymbol{\theta}|\varphi, \mathcal{H}]$ is a convex combination (a *mixture*) of products of Beta distributions. When we marginalize it w.r.t. the parameter θ_i, we obtain the following:

$$p[\theta_i|\varphi, \mathcal{H}] = \int_0^1 .. \int_0^1 p[\boldsymbol{\theta}|\varphi, \mathcal{H}]\ d\theta_1..d\theta_{i-1}d\theta_{i+1}..d\theta_m$$

$$= \sum_{w:w\models\varphi} \mathbb{P}[w|\varphi, \mathcal{H}] \cdot \mathcal{B}e(a_i + w[i], b_i + \overline{w[i]}) \cdot$$

$$\cdot \int_0^1 .. \int_0^1 \prod_{j\in\{1,..,m\}\setminus\{i\}} \mathcal{B}e(a_j + w[j], b_j + \overline{w[j]})\ d\theta_1..d\theta_{i-1}d\theta_{i+1}..d\theta_m$$

$$= \sum_{w:w\models\varphi} \mathbb{P}[w|\varphi, \mathcal{H}] \cdot \mathcal{B}e(a_i + w[i], b_i + \overline{w[i]}) \cdot 1$$

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

This concludes our proof. □