Oblivious Bounds on the Probability of Boolean Functions

WOLFGANG GATTERBAUER, Carnegie Mellon University
DAN SUCIU, University of Washington

This paper develops upper and lower bounds for the probability of Boolean functions by treating multiple occurrences of variables as independent and assigning them new individual probabilities. We call this approach dissociation and give an exact characterization of optimal oblivious bounds, i.e., when the new probabilities are chosen independent of the probabilities of all other variables. Our motivation comes from the weighted model counting problem (or, equivalently, the problem of computing the probability of a Boolean function), which is #P-hard in general. By performing several dissociations, one can transform a Boolean formula whose probability is difficult to compute, into one whose probability is easy to compute, and which is guaranteed to provide an upper or lower bound on the probability of the original formula by choosing appropriate probabilities for the dissociated variables. Our new bounds shed light on the connection between previous relaxation-based and model-based approximations and unify them as concrete choices in a larger design space. We also show how our theory allows a standard relational database management system (DBMS) to both upper and lower bound hard probabilistic queries in guaranteed polynomial time.

1. INTRODUCTION

Query evaluation on probabilistic databases is based on weighted model counting for positive Boolean expressions. Since model counting is #P-hard in general, today’s probabilistic database systems evaluate queries using one of the following three approaches: (1) incomplete approaches identify tractable cases (e.g., read-once formulas) either at the query-level [Dalvi and Suciu 2007; Dalvi et al. 2010] or the data-level [Olteanu and Huang 2008; Sen et al. 2010]; (2) exact approaches apply exact probabilistic inference, such as repeated application of Shannon expansion [Olteanu et al. 2009] or tree-width based decompositions [Jha et al. 2010]; and (3) approximate approaches either apply general purpose sampling methods [Jampani et al. 2008; Kennedy and Koch 2010; Re et al. 2007] or approximate the number of models of the Boolean lineage expression [Olteanu et al. 2010; Pink and Olteanu 2011].

This paper provides a new algebraic framework for approximating the probability of positive Boolean expressions. While our method was motivated by query evaluation on probabilistic databases, it is more general and applies to all problems that rely on weighted model counting, e.g., general probabilistic inference in graphical mod-
An important aspect of our method is that it is not model-based in the traditional sense. Instead, it enlarges the original variable space by treating multiple occurrences of variables as independent and assigning them new individual probabilities. We call this approach dissociation and explain where existing relaxation-based and model-based approximations fit into this larger space of approximations. We characterize probability assignments that lead to guaranteed upper or lower bounds on the original expression and identify the best possible oblivious bounds, i.e. after looking at only a limited scope of the expression. We prove that for every model-based bound there is always a dissociation-based bound that is as good or better. And we illustrate how a standard relational DBMS can both upper and lower bound hard probabilistic conjunctive queries without self-joins with appropriate SQL queries that use dissociation in a query-centric way.

We briefly discuss our results: We want to compute the probability $\Pr[\varphi]$ of a Boolean expression $\varphi$ when each of its Boolean variables $x_i$ is set independently to true with some given probability $p_i = \Pr[x_i]$. Computing $\Pr[\varphi]$ is known to be #P-hard in general [Valiant 1979] and remains hard to even approximate [Roth 1996]. Our approach is to approximate $\Pr[\varphi]$ with $\Pr[\varphi']$ that is easier to compute. The new formula $\varphi'$ is derived from $\varphi$ through a sequence of dissociation steps, where each step replaces $d$ distinct occurrences of some variable $x$ in $\varphi$ with $d$ fresh variables $x'_1, x'_2, \ldots, x'_d$. Thus,

\footnote{Note that weighted model counting is essentially the same problem as computing the probability $\Pr[\varphi]$ of a Boolean expression $\varphi$. Each truth assignment of the Boolean variables corresponds to one model whose weight is the probability of this truth assignment. Weighted model counting then asks for the sum of the weights of all satisfying assignments.}

\footnote{Dissociation is the breaking of an existing association between related, but not necessarily identical items.}
after applying dissociation repeatedly, we transform \( \varphi \) into another expression \( \varphi' \) and approximate \( \mathbb{P}[\varphi] \) with \( \mathbb{P}[\varphi'] \). The question that we address in this paper is: how should we set the probabilities of the dissociated variables \( x'_i \) in order to ensure that \( \mathbb{P}[\varphi'] \) is a good approximation of \( \mathbb{P}[\varphi] \)? In particular, we seek conditions under which \( \varphi' \) is guaranteed to be either an upper bound \( \mathbb{P}[\varphi'] \geq \mathbb{P}[\varphi] \) or a lower bound \( \mathbb{P}[\varphi'] \leq \mathbb{P}[\varphi] \).

Our main result can be summarized as follows: Suppose that \( x \) occurs positively in \( \varphi \). Dissociate \( x \) into two variables \( x'_1 \) and \( x'_2 \) such that the dissociated formula is \( \varphi' = \varphi'_1 \land \varphi'_2 \), where \( x'_1 \) occurs only in \( \varphi'_1 \) and \( x'_2 \) occurs only in \( \varphi'_2 \); in other words, \( \varphi = \varphi'_1 [x'/x] \land \varphi'_2 [x'/x] \) (we will later define this as “conjunctive dissociation”). Let \( p = \mathbb{P}[x], p'_1 = \mathbb{P}[x'_1], p'_2 = \mathbb{P}[x'_2] \) be their probabilities. Then (1) \( \mathbb{P}[\varphi'] \geq \mathbb{P}[\varphi] \) iff \( p'_1 \cdot p'_2 \geq p \), and (2) \( \mathbb{P}[\varphi'] \leq \mathbb{P}[\varphi] \) iff \( p'_1 \leq p \) and \( p'_2 \leq p \). In particular, the best upper bounds are obtained by choosing any \( p'_1, p'_2 \) that satisfy \( p'_1 \cdot p'_2 = p \), and the best lower bound is obtained by setting \( p'_1 = p'_2 = p \). The “only if” direction holds assuming \( \varphi' \) satisfies certain mild conditions (e.g., it should not be redundant), and under the assumption that \( p'_1, p'_2 \) are chosen obliviously, i.e. they are functions only of \( p = \mathbb{P}[x] \) and independent of the probabilities of all other variables. This restriction to oblivious probabilities guarantees the computation of the probabilities \( p'_1, p'_2 \) to be very simple. Our result extends immediately to the case when the variable \( x \) is dissociated into several variables \( x'_1, x'_2, \ldots, x'_m \), and also extends (with appropriate changes) to the case when the expressions containing the dissociated variables are separated by \( \lor \) rather than \( \land \) (Fig.1).

**Example 1.1 (2CNF Dissociation).** For a simple illustration of our main result, consider a Positive-Partite-2CNF expression with \(|E|\) clauses
\[
\varphi = \bigwedge_{(i,j) \in E} (x_i \lor y_j)
\]
for which calculating its probability is already \#P-hard [Provan and Ball 1983]. If we dissociate all occurrences of \( m \) variables \( x_i \), then the expression becomes:
\[
\varphi' = \bigwedge_{(i,j) \in E} (x'_{i,j} \lor y_j)
\]
which is equivalent to \( \bigwedge_j (y_j \lor \bigwedge_{i,j} x'_{i,j}) \). This is a read-once expression whose probability can always be computed in PTIME [Gurvich 1977]. Our main result implies the following: Let \( p_i = \mathbb{P}[x_i], i \in [m] \) be the probabilities of the original variables and denote \( p'_{i,j} = \mathbb{P}[x'_{i,j}] \) the probabilities of the fresh variables. Then (1) if \( \forall i \in [m] : p'_{i,j_1} \cdot p'_{i,j_2} \cdots p'_{i,j_{d_i}} = p_i \), then \( \varphi' \) is an upper bound \( \mathbb{P}[\varphi'] \geq \mathbb{P}[\varphi] \); (2) if \( \forall i \in [m] : p'_{i,j_1} = p'_{i,j_2} = \ldots = p'_{i,j_{d_i}} = p_i \), then \( \varphi' \) is a lower bound \( \mathbb{P}[\varphi'] \leq \mathbb{P}[\varphi] \). Furthermore, these are the best possible oblivious bounds, i.e. where \( p'_{i,j} \) depends only on \( p_i = \mathbb{P}[x_i] \) and is chosen independently of other variables in \( \varphi \).

We now explain how dissociation generalizes two other approximation methods in the literature (Fig.1 gives a high-level summary and Sect.5 the formal details).

**Relaxation & Compensation.** This is a framework by Choi and Darwiche [2009, 2010] for approximate probabilistic inference in graphical models. The approach performs exact inference in an approximate model that is obtained by relaxing equivalent oblivious bounds is inspired by the notion of oblivious routing algorithms [Valiant 1982] which use only local information and which can therefore be implemented very efficiently. Similarly, our oblivious framework forces \( p'_1, p'_2 \) to be computed only as a function of \( p_i \) without access to the rest of \( \varphi \). One can always find values \( p'_1, p'_2 \) for which \( \mathbb{P}[\varphi'] = \mathbb{P}[\varphi] \). However, to find those values in general, one has to first compute \( q = \mathbb{P}[\varphi] \), then find appropriate values \( p'_1, p'_2 \) for which the equality \( \mathbb{P}[\varphi'] = q \) holds. This is not practical, since our goal is to compute \( q \) in the first place.

ACM Transactions on Database Systems, Vol. V, No. N, Article 1, Publication date: January YYYY.
alence constraints in the original model, i.e. by removing edges. The framework allows one to improve the resulting approximations by compensating for the relaxed constraints. In the particular case of a conjunctive Boolean formula \( \varphi = \varphi_1 \land \varphi_2 \), relaxation refers to substituting any variable \( x \) that occurs in both \( \varphi_1 \) and \( \varphi_2 \) with two fresh variables \( x_1' \) in \( \varphi_1 \) and \( x_2' \) in \( \varphi_2 \). Compensation refers to then setting their probabilities \( p_1' = \mathbb{P}[x_1'] \) and \( p_2' = \mathbb{P}[x_2'] \) to \( p_1 = p \) and \( p_2 = \mathbb{P}[\varphi_1] \). This new probability assignment is justified by the fact that, if \( x \) is the only variable shared by \( \varphi_1 \) and \( \varphi_2 \), then compensation ensures that \( \mathbb{P}[\varphi'] = \mathbb{P}[\varphi] \) (we will show this claim in Prop. 5.1). In general, however, \( \varphi_1, \varphi_2 \) have more than one variable in common, and in this case we have \( \mathbb{P}[\varphi'] \neq \mathbb{P}[\varphi] \) for the same compensation. Thus in general, compensation is applied as a heuristics. Furthermore, it is then not known whether compensation provides an upper or lower bound.

Indeed, let \( p_1' = p, p_2' = \mathbb{P}[x|\varphi_1] \) be the probabilities set by the compensation method. Recall that our condition for \( \mathbb{P}[\varphi'] \) to be an upper bound is \( p_1' \cdot p_2' \geq p \), but we have \( p_1' \cdot p_2' = p \cdot \mathbb{P}[x|\varphi_1] \leq p \). Thus, the compensation method does not satisfy our oblivious upper bound condition. Similarly, because of \( p_1' = p \) and \( p_2' \geq p \), these values fail to satisfy our oblivious lower bound condition. Thus, relaxation is neither a guaranteed upper bound, nor a guaranteed lower bound. In fact, relaxation is not oblivious at all (since \( p_2' \) is computed from the probabilities of all variables, not just \( \mathbb{P}[x] \)). This enables it to be an exact approximation in the special case of a single shared variable, but fails to guarantee any bounds, in general.

Model-based approximations. Another technique for approximation described by Fink and Olteanu [2011] is to replace \( \varphi \) with another expression whose set of models is either a subset or superset of those of \( \varphi \). Equivalently, the upper bound is a formula \( \varphi_U \) such that \( \varphi \Rightarrow \varphi_U \), and the lower bound is \( \varphi_L \) such that \( \varphi_L \Rightarrow \varphi \). We show in this paper, that if \( \varphi \) is a positive Boolean formula, then all upper and lower model-based bounds can be obtained by repeated dissociation: the model-based upper bound is obtained by repeatedly setting probabilities of dissociated variables to 1, and the model-based lower bound by setting the probabilities to 0. While the thus generated model-based upper bounds for conjunctive expressions correspond to optimal oblivious dissociation bounds, the model-based lower bounds for conjunctive expressions are not optimal and can always be improved by dissociation.

Indeed, consider first the upper bound for conjunctions: the implication \( \varphi \Rightarrow \varphi_U \) holds iff there exists a formula \( \varphi_U \) such that \( \varphi \equiv \varphi_1 \land \varphi_2 \).

\[^3\] Pick a variable \( x \), denote \( p = \mathbb{P}[x] \) its probability, dissociate it into \( x_1' \) in \( \varphi_1 \) and \( x_2' \) in \( \varphi_2 \), and set their probabilities as \( p_1' = 1 \) and \( p_2' = p \). Thus, \( \varphi_U \) remains unchanged (except for the renaming of \( x \) to \( x_2' \)), while in \( \varphi_U \) we have set \( x_1 = 1 \). By repeating this process, we eventually transform \( \varphi_1 \) into true (Recall that our formula is monotone). Thus, model-based upper bounds are obtained by repeated dissociation and setting \( p_1' = 1 \) and \( p_2' = p \) at each step. Our results show that this is only one of many oblivious upper bounds as any choices with \( p_1' p_2' \geq p \) lead to an oblivious upper bound for conjunctive dissociations.

Consider now the lower bound: the implication \( \varphi_L \Rightarrow \varphi \) is equivalent to \( \varphi_L \equiv \varphi \land \varphi_2 \). Then there is a sequence of finitely many conjunctive dissociation steps, which transforms \( \varphi \) into \( \varphi \land \varphi_2 \) and thus into \( \varphi_L \). At each step, a variable \( x \) is dissociated into \( x_1' \) and \( x_2' \), and their probabilities are set to \( p_1' = p \) and \( p_2' = 0 \), respectively.\[^4\] According to our result, this choice is not optimal: instead one obtains a tighter bound by also setting \( p_2' = p \), which no longer corresponds to a model-based lower bound.

\[^3\] Fink and Olteanu [2011] describe their approach for approximating DNF expressions only. However, the idea of model-based bounds applies equally well to arbitrary Boolean expressions, including those in CNF.

\[^4\] The details here are more involved and are given in detail in Sect. 5.2.
Thus, model-based lower bounds for conjunctive expressions are not optimal and can always be improved by using dissociation.

Our dual result states the following for the case when the two formulas are connected with disjunction $\lor$ instead of conjunction $\land$: (1) the dissociation is an upper bound iff $p'_1 \geq p$ and $p'_2 \geq p$, and (2) it is a lower bound iff $(1 - p'_1)(1 - p'_2) \geq 1 - p$. We see that model-based approximation gives an optimal lower bound for disjunctions, because $(1 - p'_1)(1 - p'_2) = 1 \cdot (1 - p) = 1 - p$, however non-optimal upper bounds. Example 7.2 illustrates this asymmetry and the possible improvement through dissociation with a detailed simulation-based example.

**Bounds for hard probabilistic queries.** Query evaluation on probabilistic databases reduces to the problem of computing the probability of its lineage expression which is a a monotone, $k$-partite Boolean DNF where $k$ is fixed by the number of joins in the query. Computing the probability of the lineage is known to be $\#P$-hard for some queries [Dalvi and Suciu 2007], hence we are interested in approximating these probabilities by computing dissociated Boolean expressions for the lineage. We have previously shown in [Gatterbauer et al. 2010] that every query plan for a query corresponds to one possible dissociation for its lineage expression. The results in this paper show how to best set the probabilities for the dissociated expressions in order to obtain both upper bounds and lower bounds. We further show that all the computation can be pushed inside a standard relational database engine with the help of SQL queries that use User-Defined-Aggregates and views that replace the probabilities of input tuples with their optimal symmetric lower bounds. We illustrate this approach in [Sect. 6] and validate it on TPC-H data in [Sect. 7.5].

**Main contributions.** (1) We introduce an algebraic framework for approximating the probability of Boolean functions by treating multiple occurrences of variables as independent and assigning them new individual probabilities. We call this approach dissociation; (2) we determine the optimal upper and lower bounds for conjunctive and disjunctive dissociations under the assumption of oblivious value assignments; (3) we show how existing relaxation-based and model-based approximations fit into the larger design space of dissociations, and show that for every model-based bound there is at least one dissociation-based bound which is as good or tighter; (4) we apply our general framework to both upper and lower bound hard probabilistic conjunctive queries without self-joins in guaranteed PTIME by translating the query into a sequence of standard SQL queries; and (5) we illustrate and evaluate with several detailed examples the application of this technique. Note that this paper does not address the algorithmic complexities in determining alternative dissociations, in general.

**Outline.** Section 2 starts with some notational background, and Sect. 3 formally defines dissociation. Section 4 contains our main results on optimal oblivious bounds. Section 5 formalizes the connection between relaxation, model-based bounds and dissociation, and shows how both previous approaches can be unified under the framework of dissociation. Section 6 applies our framework to derive upper and lower bounds for hard probabilistic queries with standard relational database management systems. Section 7 gives detailed illustrations on the application of dissociation and oblivious bounds. Finally, Sect. 8 relates to previous work before Sect. 9 concludes.

### 2. GENERAL NOTATIONS AND CONVENTIONS

We use $[m]$ as short notation for \{1, \ldots, m\}, use the bar sign for the complement of an event or probability (e.g., $\bar{x} = \neg x$, and $\bar{p} = 1 - p$), and use a bold notation for sets (e.g., $s \subseteq [m]$) or vectors (e.g., $x = \langle x_1, \ldots, x_m \rangle$) alike. We assume a set $x$ of independent Boolean random variables, and assign to each variable $x_i$ a primitive event which is true with probability $p_i = P[x_i]$. We do not formally distinguish between the variable $x_i$ and the event $x_i$ that it is true. By default, all primitive events are assumed to be
3. DISSOCIATION OF BOOLEAN FUNCTIONS AND EXPRESSIONS

We define here dissociation formally. Let \( f(x, y) \) and \( f'(x', y) \) be two Boolean functions, where \( x, x' \) are disjoint sets of variables. Denote \( |x| = m, |x'| = m' \), and \( |y| = n \). We restrict \( f \) and \( f' \) to be positive in \( x \) and \( x' \), respectively [Crama and Hammer 2011, Def. 1.24].

**Definition 3.1 (Dissociation).** We call a function \( f' \) a dissociation of \( f \) if there exists a substitution \( \theta : x' \rightarrow x \) s.t. \( f'[\theta] = f \).

**Example 3.2 (CNF Dissociation).** Consider two functions \( f \) and \( f' \) given by CNF expressions

\[
\begin{align*}
\quad f &= (x_1 \lor y_1)(x_1 \lor y_2)(x_2 \lor y_1)(x_2 \lor y_3) \\
\quad f' &= (x'_1,1 \lor y_1)(x'_1,2 \lor y_2)(x'_2 \lor y_1)(x'_2 \lor y_3)
\end{align*}
\]

Then \( f' \) is a dissociation of \( f \) as \( f'[\theta] = f \) for the substitution \( \theta = \{(x'_1,1, x_1), (x'_1,2, x_1), (x'_2, x_2)\} \). Figure 2 shows the CNF expressions' primal graphs.

In practice, to find a dissociation for a function \( f(x, y) \), one proceeds like this: Choose any expression \( \varphi(x, y) \) for \( f \) and thus \( f = f_\varphi \). Replace the \( k_i \) distinct occurrences of variables \( x_i \) in \( \varphi \) with \( d_i \) fresh variables \( x'_i,1, x'_i,2, \ldots, x'_i,d_i \), with \( d_i \leq k_i \). The resulting expression \( \varphi' \) represents a function \( f' \) that is a dissociation of \( f \). Notice that we may obtain different dissociations by deciding for which occurrences of \( x_i \) to use distinct fresh variables, and for which occurrences to use the same variable. We may further obtain more dissociations by starting with different, equivalent expressions \( \varphi \) for the function \( f \). In fact, we may construct infinitely many dissociations this way. We also note that every dissociation of \( f \) can be obtained through the process outlined here. Indeed, let \( f'(x', y) \) be a dissociation of \( f(x, y) \) according to [Definition 3.1] and let \( \theta \) be the substitution for which \( f'[\theta] = f \). Then, if \( \varphi' \) is any expression representing \( f' \), the expression \( \varphi = \varphi'[\theta] \) represents \( f \). Then we can thus apply the described dissociation process to a certain expression \( \varphi \) and obtain \( f' \).

---

**Figure 2.** (a) Bipartite primal graph for CNF representing \( f \). (b) A dissociation \( f' \) where variable \( x_1 \) appearing \( k_1 = 2 \) times in \( f \) is dissociated into (replaced by) \( d_1 = 2 \) fresh variables in \( f' \).
Example 3.3 (Alternative Dissociations). Consider the two expressions:
\[
\varphi = (x \lor y_1)(x \lor y_2)(x \lor y_3)(y_4 \lor y_5) \\
\psi = xy_4 \lor xy_5 \lor y_1y_2y_3y_4 \lor y_1y_2y_3y_5
\]
Both are equivalent (\( \varphi \equiv \psi \)) and thus represent the same Boolean function (\( f_\varphi = f_\psi \)).
Yet each leads to a quite different dissociation in the variable \( x \):
\[
\varphi' = (x'_1 \lor y_1)(x'_2 \lor y_2)(x'_3 \lor y_3)(y_4 \lor y_5) \\
\psi' = x'_1y_4 \lor x'_2y_5 \lor y_1y_2y_3y_4 \lor y_1y_2y_3y_5
\]
Here, \( \varphi' \) and \( \psi' \) represent different functions (\( f_\varphi \neq f_\psi \)) and are both dissociations of \( f \) for the substitutions \( \theta_1 = \{(x'_1, x), (x'_2, x), (x'_3, x)\} \) and \( \theta_2 = \{(x'_1, x), (x'_2, x)\} \), respectively.

Example 3.4 (More alternative Dissociations). Consider the AND-function \( f(x, y) = xy \). It can be represented by the expressions \( xy \land y \land xy \land x \), \( \psi \) and \( \psi' \) are both dissociations of \( f \) for the substitutions \( \theta_1 = \{(x'_1, x), (x'_2, x), (x'_3, x)\} \) and \( \theta_2 = \{(x'_1, x), (x'_2, x)\} \), respectively.

4. Oblivious Bounds for Dissociated Event Expressions
Throughout this section, we fix two Boolean functions \( f(x, y) \) and \( f'(x', y) \) such that \( f' \) is a dissociation of \( f \). We are given the probabilities \( p = \mathbb{P}[x] \) and \( q = \mathbb{P}[y] \). Our goal is to find probabilities \( p' = \mathbb{P}[x'] \) of the dissociated variables so that \( \mathbb{P}[f'] \) is an upper or lower bound for \( \mathbb{P}[f] \). We first define oblivious bounds (Sec. 4.1), then characterize them, in general, through valuations (Sec. 4.2) and, in particular, for conjunctive and disjunctive dissociations (Sec. 4.3), then derive optimal bounds (Sec. 4.4), and end with illustrated examples for CNF and DNF dissociations (Sec. 4.5).

4.1. Definition of Oblivious Bounds
We use the subscript notation \( \mathbb{P}_{p,q}[f] \) and \( \mathbb{P}_{p',q}[f'] \) to emphasize that the probability space is defined by the probabilities \( p = (p_1, p_2, \ldots), q = (q_1, q_2, \ldots) \), and \( p' = (p'_1, p'_2, \ldots) \), respectively. Given \( p \) and \( q \), our goal is thus to find \( p' \) such that \( \mathbb{P}_{p',q}[f'] \geq \mathbb{P}_{p,q}[f] \) or \( \mathbb{P}_{p',q}[f'] \leq \mathbb{P}_{p,q}[f] \).

Definition 4.1 (Oblivious Bounds). Let \( f' \) be a dissociation of \( f \) and \( p = \mathbb{P}[x] \). We call \( p' \) an oblivious upper bound for \( p \) and dissociation \( f' \) of \( f \) iff \( \forall q : \mathbb{P}_{p',q}[f'] \geq \mathbb{P}_{p,q}[f] \). Similarly, \( p' \) is an oblivious lower bound iff \( \forall q : \mathbb{P}_{p',q}[f'] \leq \mathbb{P}_{p,q}[f] \).

In other words, \( p' \) is an oblivious upper bound if the probability of the dissociated function \( f' \) is bigger than that of \( f \) for every choice of \( q \). Put differently, the probabilities of \( x' \) depend only on the probabilities of \( x \) and not on those of \( y \).

An immediate upper bound is given by \( p' = 1 \), since \( f \) is monotone and \( f'[1/x'] = f[1/x] \). Similarly, \( p = 0 \) is a naive lower bound. This proves that the set of upper and lower bounds is never empty. Our next goal is to characterize all oblivious bounds and to then find optimal ones.

4.2. Characterization of Oblivious Bounds through Valuations
We will give a necessary and sufficient characterization of oblivious bounds, but first we need to introduce some notations. If \( f(x, y) \) is a Boolean function, let \( \nu : y \to \{0, 1\} \) be a truth assignment or valuation for \( y \). We use \( \nu \) for the vector \( \langle \nu(y_1), \ldots, \nu(y_n) \rangle \), and denote with \( f[\nu] \) the Boolean function obtained after applying the substitution \( \nu \). Note that \( f[\nu] \) depends on variables \( x \) only. Furthermore, let \( g \) be a Boolean functions,
over variables \( z \). We denote with \( g'' \) the Boolean function \( g'' = \bigwedge_j g''_j \), where \( g''_j = \bar{g}_j \) if \( \nu(y_j) = 0 \) and \( g''_j = g_j \) if \( \nu(y_j) = 1 \).

**Example 4.2 (Valuation Notation).** Assume \( g = (g_1, g_2) \) with \( g_1 = z_1 z_2 \) and \( g_2 = z_1 \bar{z}_2 \), and \( \nu = (0, 1) \). Then \( g'' = \neg(z_1 z_2) \land z_1 \bar{z}_2 = z_1 \bar{z}_2 \). Figure 3 illustrates our notation for this simple example with the help of Karnaugh maps. We encourage the reader to take a moment and carefully study the correspondences between \( g, \nu \), and \( g'' \). ■

Then, any function \( f(x, y) \) admits the following expansion by the \( y \)-variables:

\[
 f(x, y) = \bigvee_{\nu} \left( f[\nu] \land y^\nu \right)
\]  

Equation 3 gives us an alternative way to write \( f \) as disjunction over all \( 2^2 \) valuations of \( y \) as \( f = x(y_1 y_2) \lor x(y_1 \bar{y}_2) \lor x(\bar{y}_1 y_2) \lor y_1 y_2 \). ■

The following proposition is a necessary and sufficient condition for oblivious upper and lower bounds, based on valuations.

**Proposition 4.4 (Oblivious Bounds and Valuations).** Fix two Boolean functions \( f(x, y) \), \( f'(x', y) \) s.t. \( f' \) is a dissociation of \( f \), and let \( p \) and \( q' \) denote the probabilities of the variables \( x \) and \( x' \), respectively. Then \( p' \) is an oblivious upper bound iff \( \Pr_{p'}[f'[\nu]] \geq \Pr_p[f[\nu]] \) for every valuation \( \nu \) for \( y \). The proposition holds similarly for oblivious lower bounds.

**Proof.** Remember that any two events in Eq. 3 are disjoint. The total probability theorem thus allows us to sum over the probabilities of all conjuncts:

\[
\Pr_{p, q}[f(x, y)] = \sum_{\nu} \left( \Pr_p[f[\nu]] \cdot \Pr_q[y^\nu] \right)
\]

The “if” direction follows immediately. For the “only if” direction, assume that \( p' \) is an oblivious upper bound. By definition, \( \Pr_{p', q}[f'] \geq \Pr_{p, q}[f] \) for every \( q \). Fix any valuation \( \nu : y \rightarrow \{0, 1\} \), and define the following probabilities \( q_i \): \( q_0 = 0 \) when \( \nu(y_1) = 0 \), and \( q_1 = 1 \) when \( \nu(y_1) = 1 \). It is easy to see that \( \Pr_{p, q}[f] = \Pr_p[f[\nu]] \), and similarly, \( \Pr_{p', q}[f'] = \Pr_{p'}[f'[\nu]] \), which proves \( \Pr_{p'}[f'[\nu]] \geq \Pr_p[f[\nu]] \). ■
A consequence of choosing \( p' \) obliviously is that it remains a bound even if we allow the variables \( y \) to be arbitrarily correlated. More precisely:

**Corollary 4.5 (Oblivious Bounds and Correlations).** Let \( f'(x', y) \) be a dissociation of \( f(x, y) \), let \( p' \) be an oblivious upper bound for \( p \), and let \( g = \langle g_1, \ldots, g_y \rangle \) be Boolean functions in some variables \( z \) with probabilities \( r = \mathbb{P}[z] \). Then:

\[
\mathbb{P}_{p', r}[f'(x', g(z))] \geq \mathbb{P}_{p, r}[f(x, g(z))].
\]

The result for oblivious lower bounds is similar.

The intuition is that, by substituting the variables \( y \) with functions \( g \) in \( f(x, y) \), we make \( y \) correlated. The corollary thus says that an oblivious upper bound remains an upper bound even if the variables \( y \) are correlated. This follows from folklore that any correlation between the variables \( y \) can be captured by general Boolean functions \( g \).

For completeness, we include the proof in Appendix B.

**Proof of Corollary 4.5.** We derive the probabilities of \( f \) and \( f' \) from Eq. 3:

\[
\mathbb{P}_{p, r}[f(x, g)] = \sum_{\nu} \left( \mathbb{P}_{p}[f[\nu]] \cdot \mathbb{P}_{r}[g[\nu]] \right)
\]

\[
\mathbb{P}_{p', r}[f'(x', g)] = \sum_{\nu} \left( \mathbb{P}_{p'}[f'[\nu]] \cdot \mathbb{P}_{r}[g[\nu]] \right)
\]

The proof follows now immediately from Prop. 4.4. 

### 4.3. Oblivious Bounds for Unary Conjunctive and Disjunctive Dissociations

A dissociation \( f'(x', y) \) of \( f(x, y) \) is called *unary* if \(|x| = 1\), in which case we write the function as \( f(x, y) \). We next focus on unary dissociations, and establish a necessary and sufficient condition for probabilities to be oblivious upper or lower bounds for the important classes of conjunctive and disjunctive dissociations. The criterion also extends as a sufficient condition to non-unary dissociations, since these can be obtained as a sequence of unary dissociations.

**Definition 4.6 (Conjunctive and Disjunctive Dissociation).** Let \( f'(x', y) \) be a Boolean function in variables \( x', y \). We say that the variables \( x' \) are *conjunctive* in \( f' \) if \( f'(x', y) = \bigwedge_{j \in |d|} f_j(x'_j, y) \), \( d = |x'| \). We say that a dissociation \( f'(x', y) \) of \( f(x, y) \) is conjunctive if \( x' \) are conjunctive in \( f' \). Similarly, we say that \( x' \) are *disjunctive* in \( f' \) if \( f'(x', y) = \bigvee_{j \in |d|} f_j(x'_j, y) \), and a dissociation is disjunctive if \( x' \) is disjunctive in \( f' \).

Thus in a conjunctive dissociation, each dissociated variable \( x'_j \) occurs in exactly one Boolean function \( f_j \) and these functions are combined by \( \wedge \) to obtain \( f' \). In practice, we start with \( f \) written as a conjunction, then replace \( x \) with a fresh variable in each conjunct:

\[
f(x, y) = \bigwedge f_j(x, y)
\]

\[
f'(x', y) = \bigwedge f_j(x'_j, y)
\]

Disjunctive dissociations are similar.

Note that if \( x' \) is conjunctive in \( f'(x', y) \), then for any substitution \( \nu : y \to \{0, 1\} \), \( f'[\nu] \) is either 0, 1, or a conjunction of variables in \( x' \): \( f'[\nu] = \bigwedge_{j \in |d|} x'_j \), for some set \( s \subseteq |d| \), where \( d = |x'| \). Similarly, if \( x' \) is disjunctive, then \( f'[\nu] \) is 0, 1, or \( \bigvee_{j \in |s|} x'_j \). 

---

Necessity does not always extend to non-unary dissociations. The reason is that an oblivious dissociation for \( x \) may set the probability of a fresh variable by examining all variables \( x \), while \( a \) in sequence of oblivious dissociations each new probability \( \mathbb{P}[x'_i, j] \) may depend only on the variable \( x_i \) currently being dissociated.

---

Note that for \( s = \emptyset \): \( f'[\nu] = \bigwedge_{j \in |s|} x'_j = 1 \) and \( f'[\nu] = \bigvee_{j \in |s|} x'_j = 0 \).

ACM Transactions on Database Systems, Vol. V, No. N, Article 1, Publication date: January YYYY.
We need one more definition before we state the main result in our paper.

**Definition 4.7 (Cover).** Let \( x' \) be conjunctive in \( f'(x', y) \). We say that \( f' \) covers the set \( s \subseteq [d] \) if there exists a substitution \( \nu \) s.t. \( f'[\nu] = \bigwedge_{j \in s} x'_j \). Similarly, if \( x' \) is disjunctive, then we say that \( f' \) covers \( s \) if there exists \( \nu \) s.t. \( f'[\nu] = \bigvee_{j \in s} x'_j \).

**Theorem 4.8 (Oblivious Bounds).** Let \( f'(x', y) \) be a conjunctive dissociation of \( f(x, y) \), and let \( p = \mathbb{P}[x] \), \( p' = \mathbb{P}[x'] \) be probabilities of \( x \) and \( x' \), respectively. Then:

1. If \( p'_j \leq p \) for all \( j \), then \( p' \) is an oblivious lower bound for \( p \), i.e. \( \forall q: \mathbb{P}_{p, q}[f'] \leq \mathbb{P}_{p, q}[f] \). Conversely, if \( p' \) is an oblivious lower bound for \( p \) and \( f' \) covers all singleton sets \( \{j\} \) with \( j \in [d] \), then \( p'_j \leq p \) for all \( j \).
2. If \( \prod_j p'_j \geq p \), then \( p' \) is an oblivious upper bound for \( p \), i.e. \( \forall q: \mathbb{P}_{p, q}[f'] \geq \mathbb{P}_{p, q}[f] \).
Conversely, if \( p' \) is an oblivious upper bound for \( p \) and \( f' \) covers the set \( [d] \), then \( \prod_j p'_j \geq p \).

Similarly, the dual result holds for disjunctive dissociations \( f'(x', y) \) of \( f(x, y) \):
3. If \( p'_j \geq p \) for all \( j \), then \( p' \) is an oblivious upper bound for \( p \). Conversely, if \( p' \) is an oblivious upper bound for \( p \) and \( f' \) covers all singleton sets \( \{j\} \), \( j \in [d] \), then \( p'_j \geq p \).
4. If \( \prod_j (1 - p'_j) \geq 1 - p \), then \( p' \) is an oblivious lower bound for \( p \). Conversely, if \( p' \) is an oblivious lower bound for \( p \) and \( f' \) covers the set \( [d] \), then \( \prod_j (1 - p'_j) \geq 1 - p \).

**Proof.** We make repeated use of [Prop. 4.4](#). We give here the proof for conjunctive dissociations only; the proof for disjunctive dissociations is dual and similar.

1. We need to check \( \mathbb{P}_{p}[f'[\nu]] \leq \mathbb{P}_{p}[f[\nu]] \) for every \( \nu \). Since the dissociation is unary, \( f[\nu] \) can be only 0, 1, or \( x \), while \( f'[\nu] \) is 0, 1, or \( \bigwedge_{j \in s} x'_j \) for some set \( s \subseteq [d] \).
   - **Case 1:** \( f[\nu] = 0 \). We will show that \( f'[\nu] = 0 \), which implies \( \mathbb{P}_{p}[f'[\nu]] = \mathbb{P}_{p}[f[\nu]] = 0 \). Recall that, by definition, \( f'(x', y) \) becomes \( f(x, y) \) if we substitute \( x \) for all variables \( x' \). Therefore, \( f'[\nu] = f'[\nu] x/x'_1, \ldots, x/x'_d = 0 \), which implies \( f'[\nu] = 0 \) because \( f' \) is monotone in the variables \( x' \).
   - **Case 2:** \( f[\nu] = 1 \). Then \( \mathbb{P}_{p}[f'[\nu]] \leq \mathbb{P}_{p}[f[\nu]] \) holds trivially.
   - **Case 3:** \( f[\nu] = x \). Then \( \mathbb{P}_{p}[f[\nu]] = p \), while \( \mathbb{P}_{p'}[f'[\nu]] = \prod_{j \in s} p'_j \). We prove that \( s \neq \emptyset \): this implies our claim, because \( \prod_{j \in s} p'_j \leq p'_j \leq p \), for any choice of \( j \in s \). Suppose otherwise, that \( s = \emptyset \), hence \( f[\nu] = 1 \). Substituting all variables \( x' \) with \( x \) transforms \( f' \) to \( f \). This implies \( f[\nu] = 1 \), which contradicts \( f[\nu] = x \).

   For the converse, assume that \( p' \) is an oblivious lower bound. Since \( f' \) covers \( \{j\} \), there exists a substitution \( \nu \) s.t. \( f'[\nu] = x'_j \), and therefore \( f[\nu] = x \). By [Prop. 4.4](#), we have \( p'_j = \mathbb{P}_{p'}[f'[\nu]] \leq \mathbb{P}_{p}[f[\nu]] = p \), proving the claim.

2. Here we need to check \( \mathbb{P}_{p'}[f'[\nu]] \geq \mathbb{P}_{p}[f[\nu]] \) for every \( \nu \). The cases when \( f[\nu] \) is either 0 or 1 are similar to the cases above, so we only consider the case when \( f[\nu] = x \). Then \( f'[\nu] = \bigwedge_{j \in s} x'_j \) and \( \mathbb{P}_{p'}[f'[\nu]] = \prod_{j \in s} p'_j \geq \prod_{j \in s} p_j \geq p = \mathbb{P}_{p}[f[\nu]] \).

   For the converse, assume \( p' \) is an oblivious upper bound, and let \( \nu \) be the substitution for which \( f'[\nu] = \bigwedge_j x'_j \) (which exists since \( f' \) is covers \( [d] \)). Then \( \mathbb{P}_{p'}[f'[\nu]] \geq \mathbb{P}_{p}[f[\nu]] \) implies \( p \leq \prod_j p_j \).

\[\square\]

### 4.4. Optimal Oblivious Bounds for Unary Conjunctive and Disjunctive Dissociations

We are naturally interested in the “best possible” oblivious bounds. Call a dissociation \( f' \) *non-degenerate* if it covers all singleton sets \( \{j\}, j \in [d] \) and the complete set \( [d] \).

[Theorem 4.8](#) then implies:
Corollary 4.9 (Optimal Oblivious Bounds). If \( f' \) is a conjunctive dissociation of \( f \) and \( f' \) is non-degenerate, then the optimal oblivious lower bound is \( p'_1 = p'_2 = \cdots = p \), while the optimal oblivious upper bounds are obtained whenever \( p'_1 p'_2 \cdots = p \). Similarly, if \( f' \) is a disjunctive dissociation of \( f \) and \( f' \) is non-degenerate, then the optimal oblivious upper bound is \( p'_1 = p'_2 = \cdots = p \), while the optimal oblivious lower bounds are obtained whenever \( (1 - p'_1) \cdot (1 - p'_2) \cdots = 1 - p \).

Notice that while optimal lower bounds for conjunctive dissociations and optimal upper bounds for disjunctive dissociations are uniquely defined with \( p'_1 = p \), there are infinitely many optimal bounds for the other directions (see Fig. 1). Let us call bounds symmetric if all dissociated variable have the same probability. Then optimal symmetric lower bounds for conjunctive dissociations are \( p'_1 = \sqrt{p} \), and optimal symmetric lower bounds for disjunctive dissociations \( p'_1 = 1 - \sqrt{1 - p} \).

We give two examples of degenerate dissociations. First, the dissociation \( f' = (x'_1 y_1 \lor y_1) \land (x'_2 y_2 \lor y_3) \) does not cover either \( \{1\} \) nor \( \{2\} \): no matter how we substitute \( y_1, y_2, y_3 \), we can never transform \( f' \) to \( x'_1 \). For example, \( f'[1/y_1, 0/y_2, 0/y_3] = 0 \) and \( f'[1/y_1, 0/y_2, 1/y_3] = 1 \). But \( f' \) does cover the set \( \{1, 2\} \) because \( f'[1/y_1, 1/y_2, 0/y_3] = x_1 \land x_2 \). Second, the dissociation \( f' = (x'_1 y_1 \lor y_1) \land (x'_2 y_2 \lor y_1) \) covers both \( \{1\} \) and \( \{2\} \), but does not cover the entire set \( \{1, 2\} \). In these cases the oblivious upper or lower bounds in [Theorem 4.8] are still hold, but are not necessarily optimal.

However, most cases of practical interest result in dissociations that are non-degenerate, in which case the bounds in [Theorem 4.8] are tight. We explain this here. Consider the original function, pre-dissociation, written in a conjunctive form:

\[
f(x, y) = g_0 \land \bigwedge_{j \in [d]} (x \lor g_j) = g_0 \land \bigwedge_{j \in [d]} f_j
\]

where each \( g_j \) is a Boolean function in the variables \( y \), and where we denoted \( f_j = x \lor g_j \). For example, if \( f \) is a CNF expression, then each \( f_j \) is a clause containing \( x \), and \( g_0 \) is the conjunction of all clauses that do not contain \( x \). Alternatively, we may start with a CNF expression, and group the clauses containing \( x \) in equivalence classes, such that each \( f_j \) represents one equivalence class. For example, starting with four clauses, we group into two functions \( f = [(x \lor y_1) (x \lor y_2)] \land [(x \lor y_3) (x \lor y_4)] = (x \lor y_1 y_2) \land (x \lor y_3 y_4) = f_1 \land f_2 \). Our only assumption about [Eq. 4] is that it is non-redundant, meaning that none of the expressions \( g_0 \) or \( f_j \) may be dropped. Then we prove:

Proposition 4.10 (Non-degenerate Dissociation). Suppose the function \( f \) in [Eq. 4] is non-redundant. Define \( f' = g_0 \land \bigwedge_{j \in [d]} (x'_j \lor g_j) \). Then \( f' \) covers every singleton set \( \{j\} \). Moreover, if the implication \( g_0 \Rightarrow \bigvee_j g_j \) does not hold, then \( f' \) also covers the set \( [d] \). Hence \( f' \) is non-degenerate. A similar result holds for disjunctive dissociations if the dual implication \( g_0 \Leftarrow \bigwedge_j g_j \) does not hold.

Proof. We give here the proof for conjunctive dissociations only; the proof for disjunctive dissociations follows from duality. We first prove that \( f' \) covers any singleton set \( \{j\} \), for \( j \in [d] \). We claim that the following logical implication does not hold:

\[
g_0 \land \bigwedge_{i \neq j} g_i \Rightarrow g_j
\]

Indeed, suppose the implication holds for some \( j \). Then the following implication also holds: \( g_0 \land \bigwedge_{i \neq j} (x \lor g_i) \Rightarrow (x \lor g_j) \), since for \( x = 0 \) it is the implication above, while for \( x = 1 \) it is a tautology. Therefore, the function \( f_j \) is redundant in [Eq. 4], which contradicts our assumption. Hence, the implication [Eq. 5] does not hold. Let \( \nu \) be any
We next give two examples that illustrate optimal oblivious bounds for conjunctive and disjunctive dissociations in some detail.

**Example 4.11 (CNF Dissociation).** Consider the function $f_c$ given by an CNF expression and its dissociation $f'_c$:

\[
\begin{align*}
f_c &= (x \lor y_1) (x \lor y_2) (x \lor y_3) y_4 \\
f'_c &= (x'_1 \lor y_1) (x'_2 \lor y_2) (x'_3 \lor y_3) y_4
\end{align*}
\]

There are $2^4 = 16$ valuations for $y = \langle y_1, y_2, y_3, y_4 \rangle$. Probabilities $p'_1 = \langle p'_{11}, p'_{12}, p'_{13} \rangle$ are thus an oblivious upper bound exactly if they satisfy the 16 inequalities given under “CNF dissociation” in Fig. 4a. For valuations with $x_3 = 0$ (and thus $f'_c[\nu] = 0$) or all

![Diagram](image-url)
\[\nu_j = 1 \text{ (and thus } f'_i[\nu] = 1) \text{ the inequalities trivially hold. For the remaining } 7 \text{ non-trivial inequalities, } p_j^1p_j^2p_j^3 \geq p \text{ implies all others. Figure 4b shows the partial order between the non-trivial valuations, with } x'_jx'_jx'_j \text{ implying all others. Since } f_c \text{ and } f'_c \text{ are positive in } x \text{ and } x', \text{ respectively, it follows that optimal oblivious upper bounds are given by } p_j^1p_j^2p_j^3 = p, \text{ e.g., by setting } p'_i = \sqrt[p]{p} \text{ for symmetric bounds.}

Oblivious lower bounds are given by the 16 inequalities after inverting the inequality sign. Here we see that the three inequalities } p'_j \leq p \text{ together imply the others. Hence, oblivious lower bounds are those that satisfy all three inequalities. The only optimal oblivious upper bounds are then given by } p'_j = p. \]

**Example 4.12 (DNF Dissociation).** Consider the function } f_d \text{ given by an DNF expression and its dissociation } f'_d:

\[
\begin{align*}
f_d &= xy_1 \lor xy_2 \lor xy_3 \lor y_4 \\
 f'_d &= x'_1y_1 \lor x'_2y_2 \lor x'_3y_3 \lor y_4
\end{align*}
\]

An oblivious upper bound } p' = (p_1^1, p_2^1, p_3^1) \text{ must thus satisfy the 16 inequalities given under “DNF dissociation” in Fig. 4a. For valuations with } \nu_4 = 1 \text{ (and thus } f_d[\nu] = 1) \text{ or } \nu_4 = 0 \text{ (and thus } f_d[\nu] = 0) \text{ the inequalities trivially hold. For the remaining inequalities we see that the elements of set } \{x_1^1, x_2^1, x_3^1\} \text{ together imply all others, and that } x_1^1 \lor x_2^1 \lor x_3^1 \text{ is implied by all others (Fig. 4c shows the partial order between the non-trivial valuations). Thus, an oblivious upper bound must satisfy } p'_j \geq p, \text{ and the optimal one is given by } p'_j = p. \text{ Analogously, an oblivious lower bound } \text{ must satisfy } \bar{p}_1^1\bar{p}_2^1\bar{p}_3^1 \geq \bar{p}. \text{ Optimal ones are given for } \bar{p}_1^1\bar{p}_2^1\bar{p}_3^1 = \bar{p}, \text{ e.g., by setting } p'_j = 1 - \sqrt[\bar{p}]{\bar{p}}. \]

5. RELAXATION AND MODEL-BASED BOUNDS AS DISSOCIATION

This section formalizes the connection between relaxation, model-based bounds and dissociation that was outlined in the introduction. In other words, we show how both previous approaches can be unified under the framework of dissociation.

5.1. Relaxation & Compensation

The following proposition shows relaxation & compensation as conjunctive dissociation and was brought to our attention by [Choi and Darwiche 2011].

**Proposition 5.1 (Compensation and Conjunctive Dissociation).** Let } f_1, f_2 \text{ be two monotone Boolean functions which share only one single variable } x. \text{ Let } f \text{ be their conjunction, and } f' \text{ be the dissociation of } f \text{ on } x, \text{ i.e.}

\[
\begin{align*}
f &= f_1 \land f_2 \\
f' &= f_1[x'_1/x] \lor f_2[x'_2/x]
\end{align*}
\]

Then } \mathbb{P}[f] = \mathbb{P}[f'] \text{ for } \mathbb{P}[x'_1] = \mathbb{P}[x] \text{ and } \mathbb{P}[x'_2] = \mathbb{P}[x|f_1].

**Proof of Prop. 5.1.** First, note that } \mathbb{P}[f] = \mathbb{P}[f_1]\mathbb{P}[f_2|f_1]. \text{ On the other hand, } \mathbb{P}[f'] = \mathbb{P}[f'_1]\mathbb{P}[f'_2] \text{ as } f'_1 \text{ and } f'_2 \text{ are independent after dissociating on the only shared variable } x. \text{ We also have } \mathbb{P}[f_1] = \mathbb{P}[f'_1] \text{ since } \mathbb{P}[x] = \mathbb{P}[x'_1]. \text{ It remains to be shown that}

\[\text{Remember that the probability of a disjunction of two independent events is } \mathbb{P}[x'_1 \lor x'_2] = 1 - \bar{p}_1^1\bar{p}_2^1. \]
\[ P[f_2'] = P[f_2|f_1]. \] Indeed:

\[
P[f_2'] = P[x_2' \cdot f_2'(1/x_2', x_2') \cdot f_2'(0/x_2')]
= P[x_2'] \cdot P[f_2'(1/x_2')] + P[x_2'] \cdot P[f_2'(0/x_2')]
= P[x_2'] \cdot P[f_2(1/x)] + P[x_2'] \cdot P[f_2(0/x)]
= P[x_2] \cdot P[f_2(1/x)|f_1] + P[x_2] \cdot P[f_2(0/x)|f_1]
= P[f_2|f_1]
\]

which proves the claim. □

Note that compensation is not oblivious, since the probability \( p_2' \) depends on the other variables occurring in \( \varphi_1 \). Further note that, in general, \( \varphi_1, \varphi_2 \) have more than one variable in common, and in this case we have \( P[\varphi'] \neq P[\varphi] \) for the same compensation. Thus in general, compensation is applied as a heuristics, and it is then not known whether it provides an upper or lower bound.

The dual result for disjunctions holds by replacing \( f_1 \) with its negation \( \bar{f}_1 \) in \( P[x_2'] = P[x|f_1] \). This result is not immediately obvious from the previous one and has, to our best knowledge, not been stated or applied anywhere before.

**Proposition 5.2** ("Disjunctive Compensation"). Let \( f_1, f_2 \) be two monotone Boolean functions which share only one single variable \( x \). Let \( f \) be their disjunction, and \( f' \) be the dissociation of \( f \) on \( x \), i.e. \( f = f_1 \lor f_2 \), and \( f' = f_1[x'/x] \lor f_2[x'/x] \). Then \( P[f] = P[f'] \) for \( P[x_1] = P[x] \) and \( P[x_2] = P[x|f_1] \).

**Proof of Prop. 5.2** Let \( g = f, g_1 = f_1, \) and \( g_2 = f_2 \). Then \( f = f_1 \lor f_2 \) is equivalent to \( g = g_1 \land g_2 \). From Prop. 5.1 we know that \( P[g] = P[g'], \) and thus \( P[f] = P[f'] \), for \( P[x_1] = P[x] \) and \( P[x_2] = P[x|g_1] = P[x|f_1] \). □

### 5.2. Model-based Approximation

The following proposition shows that all model-based bounds can be derived by repeated dissociation. However, not all dissociation-bounds can be explained as models since dissociation is in its essence an *algebraic* and not a model-based technique (dissociation creates more variables and thus changes the probability space). Therefore, dissociation can improve any existing model-based approximation approach. Example 7.2 will illustrate this with a detailed simulation-based example.

**Proposition 5.3** (Model-based Bounds as Dissociations). Let \( f, f_U \) be two monotone Boolean functions over the same set of variables, and for which the logical implication \( f \Rightarrow f_U \) holds. Then: (a) there exists a sequence of optimal conjunctive dissociations that transform \( f \) to \( f_U \), and (b) there exists a sequence of non-optimal disjunctive dissociations that transform \( f \) to \( f_U \). The dual result holds for the logical implication \( f_L \Rightarrow f \): (c) there exists a sequence of optimal disjunctive dissociations that transform \( f \) to \( f_L \), and (d) there exists a sequence of non-optimal conjunctive dissociations that transform \( f \) to \( f_L \).

**Proof of Prop. 5.3**. We focus here on the implication \( f \Rightarrow f_U \). The proof for the results \( f_L \Rightarrow f \) then follows from duality.

(a) The implication \( f \Rightarrow f_U \) holds iff there exists a positive function \( f_2 \) such that \( f = f_U \land f_2 \). Pick a set of variables \( x \) s.t. \( f_2(1/x) = 1 \), and dissociate \( f \) on \( x \) into \( f' = f_U[x/x'] \land f_2[x'/x]. \) By setting the probabilities of the dissociated variables to \( p_1 = p \) and \( p_2 = 1 \), the bounds become optimal (\( p'_1 p'_2 = p \)). Further more, \( f_U \) remains unchanged (except for the renaming of \( x \) to \( x' \)), whereas \( f_2 \) becomes true. Hence, we get \( f' = f_U \). Thus, all model-based upper bounds can be obtained by conjunctive dissociation and choosing optimal oblivious bounds at each dissociation step.
(b) The implication \( f \Rightarrow f_U \) also holds if \( f_U \) is a positive function such that \( f_U = f \lor f_d \). Let \( m \) be the positive minterm or elementary conjunction involving all variables of \( f \). The function \( f_d \) can be then written as DNF \( f_d = c_1 \lor c_2 \lor \ldots \), with products \( c_i \subseteq m \). Since \( f \) is monotone, we know \( m \Rightarrow f \), and thus also \( m f_d \Rightarrow f \). We can therefore write \( f = f \lor m f_d \) or as
\[
 f = f \lor m c_1 \lor m c_2 \lor \ldots 
\]
Let \( x_i \) be the set of all variables in \( m \) that do not occur in \( c_i \) and denote with \( m_i \) the conjunction of \( x_i \). Then then each \( m c_i \) can instead be written as \( m_i c_i \) and thus:
\[
 f = f \lor m_1 c_1 \lor m_2 c_2 \lor \ldots 
\]
WLOG, we now separate one particular conjunct \( m_i c_i \) and dissociate on the set \( x_i \)
\[
 f' = \frac{f \lor m_1 c_1 \lor m_2 c_2 \lor \ldots x_i'}{f_1} \lor \frac{m_i c_i[x_i']}{f_2} 
\]
By setting the probabilities of the dissociated variables to the non-optimal upper bounds \( p_1 = p \) and \( p_2 = 1 \), \( f_1 \) remains unchanged (except for the renaming of \( x_i \) to \( x_i' \)), whereas \( f_2 \) becomes \( c_i \). Hence, we get \( f' = f \lor m_1 c_1 \lor m_2 c_2 \lor \ldots \lor c_i \). We can now repeat the same process for all conjuncts \( m c_i \) and receive after a finite number of dissociation steps
\[
 f'' = f \lor (c_1 \lor c_2 \lor \ldots) = f \lor f_d 
\]
Hence \( f'' = f_U \). Thus, all model-based upper bounds can be obtained by disjunctive dissociation and choosing non-optimal bounds at each dissociation step. 

6. QUERY-CENTRIC DISSOCIATION BOUNDS FOR PROBABILISTIC QUERIES

Our previous work [Gatterbauer et al. 2010] has shown how to upper bound the probability of conjunctive queries without self-joins by issuing a sequence of SQL statements over a standard relational DBMS. This section illustrates such dissociation-based upper bounds and also complements them with new lower bounds. We use the Boolean query \( Q := R(X), S(X, Y), T(Y) \), for which the probability computation problem is known to be \#P-hard, over the following database instance \( D \):

\[
\begin{array}{ccc}
R & | & A \\
\hline
x_1 & | & T \\
x_2 & | & z_2 \\
\end{array} \quad \begin{array}{ccc}
S & | & A & B \\
\hline
z_1 & | & 1 & 1 \\
z_2 & | & 2 & 1 \\
\end{array} \quad \begin{array}{ccc}
T & | & B \\
y_1 & | & 1 \\
y_2 & | & 2 \\
\end{array}
\]

Thus, relation \( S \) has three tuples \((1,1), (2,1), (2,2)\), and both \( R \) and \( T \) have two tuples \((1) \) and \((2) \). Each tuple is annotated with a Boolean variable \( x_1, x_2, z_1, \ldots, y_2 \), which represents the independent event that the corresponding tuple is present in the database. The lineage expression \( \varphi \) is then the DNF that states which tuples need to be present in order for the Boolean query \( Q \) to be true:
\[
\varphi = x_1 z_1 y_1 \lor x_2 z_2 y_1 \lor x_2 z_3 y_2 
\]
Calculating \( \mathbb{P}[\varphi] \) for a general database instance is \#P-hard. However, if we treat each occurrence of a variable \( x_i \) in \( \varphi \) as different (in other words, we dissociate \( \varphi \) eagerly on all tuples \( x_i \) from table \( R \)), then we get a read-once expression
\[
\varphi' = x_1 z_1 y_1 \lor x'_{2,1} z_2 y_1 \lor x'_{2,2} z_3 y_2 \\
= (x'_{1} z_1 \lor x'_{2,1} z_2) y_1 \lor x'_{2,2} z_3 y_2 
\]
select IOR(Q3.P) as P 
from (select T.B, T.P*Q2.P as P 
from T, 
(select Q1.B, IOR(Q1.P) as P 
from (select S.A, S.B, S.P*R.P as P 
from R, S, T 
where R.A=S.A) as Q1 
group by Q1.B) as Q2 
where T.B = Q2.B) as Q3

(a) Query \( P_R \)

create view VR as 
select R.A, 
1-power(1-R.P,1e0/count(*)) as P 
from R, S, T 
where R.A=S.A 
and S.B=T.B 
group by R.A, R.P

(b) View \( V_R \) for lower bound with \( P_R \)

\[
\text{Fig. 5.} \hspace{1em} \text{(a) SQL query corresponding to plan} \ P_R \hspace{1em} \text{for deriving an upper bound for the hard probabilistic Boolean query} \ Q = R(X), S(X,Y), T(Y) \hspace{1em} \text{Table} \ R \hspace{1em} \text{needs to be replaced with the view} \ V_R \hspace{1em} \text{for deriving a lower bound.} \hspace{1em} \text{IOR is a user-defined aggregate explained in the text and stated in Appendix C.}
\]

Writing \( p_1, q_1, r_1 \) for the probabilities of variables \( x_i, y_1, z_i \), respectively, we can calculate

\[
\text{P}[\varphi] = (((p_1^1 \cdot r_1) \otimes (p_2^1 \cdot r_2)) \cdot q_1 \otimes (p_2^2 \cdot r_3 \cdot q_2)
\]

where “\( \otimes \)” stands for multiplication and “\( \otimes \)” for independent-or.\(^{10}\)

We know from Theorem 4.8 (3) that \( \text{P}[\varphi] \) is an upper bound to \( \text{P}[\varphi] \) by assigning the original probabilities to the dissociated variables. Furthermore, we have shown in \( \text{Gatterbauer et al. 2010} \) that \( \text{P}[\varphi] \) can be calculated with a probabilistic query plan

\[
P_R = \pi^P_0 \otimes^P_1 \otimes^P_2 \otimes^P_3 \text{[} R(X), S(X,Y), T(Y) \text{]}
\]

where the probabilistic join operator \( \otimes^P \) (in prefix notation) and the probabilistic project operator with duplicate elimination \( \pi^P \) compute the probability assuming that their inputs are independent [Fuhr and Rolleke 1997]. Thus, when the join operator joins two tuples with probabilities \( p_1 \) and \( p_2 \), respectively, the output has probability \( p_1 \cdot p_2 \). When the independent project operator eliminates \( k \) duplicate records with probabilities \( p_1, \ldots, p_k \), respectively, the output has probability \( 1 - p_1 \cdots p_k \). This connection between read-once formulas and query plans was first observed by [Oteanu and Huang 2006]. We write here \( P_R \) to emphasize that this plan dissociates tuples in table \( R \).\(^{11}\)

Figure 5a shows the corresponding SQL query assuming that each of the input tables has one additional attribute \( P \) for the probability of a tuple. The query deploys a user-defined aggregate (UDA) IOR that calculates the independent-or for the probabilities of the tuples grouped together, i.e. \( \text{IOR}(p_1, p_2, \ldots, p_n) = 1 - p_1 \cdots p_n \). Appendix C states the UDA definition for PostgreSQL.

We also know from Theorem 4.8 (4) that \( \text{P}[\varphi] \) is a lower bound to \( \text{P}[\varphi] \) by assigning new probabilities \( 1 - \sqrt{1 - p_1} \) to \( x_{2,1} \) and \( x_{2,2} \) (or more generally, any probabilities \( p_{2,1} \) and \( p_{2,2} \) with \( p_{2,1}^2 + p_{2,2}^2 \geq 1 \)). Because of the connection between the read-once expression \( \varphi \) and the query plan \( P_R \), we can calculate the lower bound by using the same SQL query from Fig. 5a after exchanging the table \( R \) with a view \( V_R \) (Fig. 5b); \( V_R \) is basically a copy of \( R \) that replaces the probability \( p_i \) of a tuple \( x_i \) appearing in \( \varphi \) with \( 1 - \sqrt{1 - p_i} \) where \( d_i \) is the number of times that \( x_i \) appears in the lineage of \( Q \). The view joins tables \( R, S \) and \( T \), groups the original input tuples \( x_i \) from \( R \), and assigns each \( x_i \) the new probability \( 1 - \sqrt{1 - p_i} \) calculated as \( 1\text{-power}(1-T.P,1e0/count(*)) \).

\(^{10}\)The independent-or combines two probabilities as if calculating the disjunction between two independent events. It is defined as \( p_1 \otimes p_2 := 1 - p_1 \cdot p_2 \).

\(^{11}\)Note we use the notation \( P_R \) for both the probabilistic query plan and the corresponding SQL query.
Alternatively to $\varphi'$, if we treat each occurrence of a variable $y_j$ in $\varphi$ as different (in other words, we dissociate $\varphi$ eagerly on all tuples $y_j$ from table $T$), then we get another read-once expression

$$\varphi'' = x_1z_1y_{1,1} \lor x_2z_2y_{1,2} \lor x_2z_3y_2$$

$$= x_1z_1y_{1,1} \lor x_2(z_2y_{1,2} \lor z_3y_2)$$

$\mathbb{P}[\varphi'']$ is an upper bound to $\mathbb{P}[\varphi]$ by assigning the original probabilities to the dissociated variables. In turn, $\mathbb{P}[\varphi'']$ can be calculated with another probabilistic query plan that dissociates all tuples from table $T$ instead of $R$:

$$P_T = \pi_{y_{1,1}}^{p_{y_{1,1}}} \pi_{y_{1,2}}^{p_{y_{1,2}}} R(X) \pi_{S}^{p_{S}} [S(X,Y), T(Y)]$$

Similarly to before, $\mathbb{P}[\varphi'']$ is a lower bound to $\mathbb{P}[\varphi]$ by assigning new probabilities $1 - \sqrt{1 - q_j}$ to $y_{1,1}$ and $y_{1,2}$. And we can calculate this lower bound with the same query $P_T$ after exchanging $T$ with a view $V_T$ that replaces the probability $q_j$ of a tuple $y_j$ with $1 - \sqrt{1 - q_j}$ where $d_j$ is the number of times that $y_j$ appears in the lineage of $Q$.

Note that both query plans will calculate upper and lower bounds to query $Q$ over any database instance $D$. In fact, all possible query plans give upper bounds to the true query probability. As we have illustrated here, by replacing the input tables with appropriate views, we can use the same query plans to derive lower bounds. We refer the reader to [Gatterbauer et al. 2010] where we develop the theory of the partial dissociation order among all possible query plans and give a sound and complete algorithm that returns a set of query plans which are guaranteed to give the tightest bounds possible in a query-centric way for any conjunctive query without self-joins. For our example hard query $Q$, plans $P_R$ and $P_T$ are the best possible plans. We further refer to [Gatterbauer and Suciu 2013] for more details and an extensive discussion on how to speed up the resulting multi-query evaluation.

Also note that these upper and lower bounds can be derived with the help of any standard relational database, even cloud-based databases which commonly do not allow users to define their own UDAs. To our best knowledge, this is the first technique that can upper and lower bound hard probabilistic queries without any modifications to the database engine nor performing any calculations outside the database.

7. Illustrations of Oblivious Bounds

In this section, we study the quality of oblivious bounds across varying scenarios: We study the bounds as function of correlation between non-dissociated variables (Sect. 7.1), compare dissociation-based with model-based approximations (Sect. 7.2), illustrate a fundamental asymmetry between optimal upper and lower bounds (Sect. 7.3), show that increasing the number of simultaneous dissociations does not necessarily worsen the bounds (Sect. 7.4), and apply our framework to approximate hard probabilistic queries over TPC-H data with a standard relational database management system (Sect. 7.5).

7.1. Oblivious Bounds as Function of Correlation between Variables

Example 7.1 (Oblivious Bounds and Correlations). Here we dissociate the DNF $\varphi_d = x \lor x B$ and the analogous CNF $\varphi_c = (x \lor A) (x \lor B)$ on $x$ and study the er-

---

12 The UDA IOR can be expressed with standard SQL aggregates, e.g., "$1-\exp(\text{sum}(\text{case } Q3.P \text{ when 1 then 1E-307 else 1-Q3.P end}))$" on Microsoft SQL Azure.
Clearly, the bounds also depend on the probabilities of the variables $x, A, B$. Let $p = \Pr[x]$ and assume $A$ and $B$ have the same probability $q = \Pr[A] = \Pr[B]$. We set $p' = \Pr[x_1] = \Pr[x_2]$ according to the optimal symmetric bounds from Corollary 4.9.

In a few steps, one can calculate the probabilities as

\[
\begin{align*}
\Pr[\varphi_d] &= 2pq - p\Pr[AB] \\
\Pr[\varphi_d'] &= 2p'q - p'\Pr[AB] \\
\Pr[\varphi_c] &= p + (1 - p)\Pr[AB] \\
\Pr[\varphi_c'] &= 2p'q + p'^2(1 - 2q) + (1 - p'^2)\Pr[AB]
\end{align*}
\]

**Results:** Figure 6 shows the probabilities of the expressions $\Pr[\varphi]$ (full lines) and those of their dissociations $\Pr[\varphi']$ (border of shaded areas) for various values of $p, q$ and as function of the correlation $\rho(A, B)$. For example, Fig.6d shows the case for DNF when $\Pr[x]$ is $p = 0.8$ and $A, B$ have the same probability $q$ of either 0.8, 0.5, or 0.2. When $A, B$

\[13\text{Note that this simplified example also illustrates the more general case } \psi_d = xA \lor xB \lor C \text{ when } C \text{ is independent of } A \text{ and } B, \text{ and thus } \Pr[\psi_d] = \Pr[\psi_d](1 - \Pr[C]) + \Pr[C]. \text{ As a consequence, the graphs in Fig.6 for } \Pr[C] \neq 0 \text{ would be vertically compressed and the bounds tighter in absolute terms.}
\]

\[14\text{The correlation } \rho(A, B) \text{ between Boolean events } A \text{ and } B \text{ is defined as } \rho(A, B) = \frac{\text{cov}(A, B)}{\text{var}(A)\text{var}(B)}, \text{ with covariance } \text{cov}(A, B) = \Pr[AB] - \Pr[A]\Pr[B] \text{ and variance } \text{var}(A) = \Pr[A] - (\Pr[A])^2. \text{ Hence, notice that}
\]
are not correlated at all (ρ = 0), then the upper bound is a better approximation when q is small, and the lower bound is a better approximation when q is large. On the other hand, if A, B are not correlated, then there is no need to dissociate the two instances of x as one can then compute \( P(x \lor A)(x \lor B) \) simply as \( p + pP[A]P[B] \). The more interesting case is when A, B are positively correlated (\( P[AB] \geq P[A]P[B] \)), e.g., positive Boolean functions of other independent variables z, such as the provenance for probabilistic conjunctive queries). The right of the vertical dashed line of Fig.6d shows that, in this case, dissociation offers very good upper and lower bounds, especially when the formula has a low probability. The graph also shows the effect of dissociation when A, B are negatively correlated (left of dashed line). Notice that the correlation cannot always be −1 (e.g., two events, each with probability > 0.5, can never be disjunct). The graphs also illustrate why these bounds are obliviously optimal, i.e. without knowledge of A, B: for every choice of p, there are some A, B for which the upper or lower bound becomes tight.

7.2. Oblivious Bounds versus Model-based Approximations

Example 7.2 (Disjunctive Dissociation and Models). This example compares the approximation of our dissociation-based approach with the model-based approach by [Fink and Olteanu 2011] and illustrates how dissociation-based bounds are tighter, in general, than model-based approximations. For this purpose, we consider again the hard Boolean query \( Q : R(X), S^d(X, Y), T(Y) \) over the database D from Sect. 6. We now only assume that the table S is deterministic, as indicated by the superscript d in \( S^d \). The query-equivalent lineage formula is then

\[
φ = x_1 y_1 \lor x_2 y_1 \lor x_2 y_2
\]

for which Fig. 7a shows the bipartite primal graph. We use this instance as its primal graph forms a \( P_4 \), which is the simplest 2-partite lineage that is not read-once. In order to compare the approximation quality, we need to limit ourselves to an example which is tractable enough so we can generate the whole probability space. In practice, we allow each variable to have any of 11 discrete probabilities \( D = \{0, 0.1, 0.2, \ldots, 1\} \) and consider all \( 11^4 = 14641 \) possible probability assignments \( υ : (p_1, p_2, q_1, q_2) \to D^2 \) with \( p = P[x] \) and \( q = P[y] \). For each \( υ \), we calculate both the absolute error \( δ^υ = P[φ^∗] - P[φ] \) and the relative error \( ε^υ = \frac{δ^υ}{P[φ^∗]} \), where \( P[φ^∗] \) stands for any of the approximations, and the exact probability \( P[φ] \) is calculated by the Shannon expansion on \( y_1 \) as \( φ \equiv y_1 (x_1 \lor x_2) \lor \neg y_1 (x_2 y_2) \) and thus \( P_p,q[φ] = (1 - (1 - p_1)(1 - p_2))q_1 + (1 - q_1)p_2 q_2 \).

Models: We use the model-based approach by [Fink and Olteanu 2011] to approximate \( φ \) with lowest upper bound (LUB) formulas \( φ_{U1} \) and greatest lower bound (GLB) formulas \( φ_{L1} \), for which \( φ_{U1} \Rightarrow φ \) and \( φ \Rightarrow φ_{L1} \), and neither of the upper (lower) bounds implies another upper (lower) bound. Among all models considered, we focus on only read-once formulas. Given the lineage \( φ \), the 4 LUBs are \( φ_{U1} = x_1 y_1 \lor x_2 \), \( φ_{U2} = y_1 \lor x_2 y_2 \), \( φ_{U3} = (x_1 \lor x_2)y_1 \lor y_2 \), and \( φ_{U4} = x_1 \lor x_2(y_1 \lor y_2) \). The 3 GLBs are \( φ_{L1} = (x_1 \lor x_2) y_2 \), \( φ_{L2} = x_1(y_1 \lor y_2) \), and \( φ_{L3} = x_1 y_1 \lor x_2 y_2 \). For each \( υ \), we choose \( \min_i(p_i[φ_{U1}]) \) and \( \max_i(p_i[φ_{L1}]) \) as the best upper and lower model-based bounds, respectively.

\[
ρ(A, B) = \frac{P[AB] - q^2}{q^2 - q^2ρ} \quad \text{and, hence: } P[AB] = ρ(A, B) \cdot (q - q^2) + q^2. \]

Further, \( P[AB] = 0 \) (i.e. disjointness between A and B) is not possible for q > 0.5, and from \( P[A \lor B] \leq 1 \), one can derive \( P[AB] \geq 2q - 1 \). In turn, \( ρ = -1 \) is not possible for q < 0.5, and it must hold \( P[AB] \geq 0 \). From both together, one can derive the condition \( ρ_{\min}(q) = \max(-\frac{q}{2q - 1}, -\frac{1 - q^2 - 2q}{q^2 - q^2}) \) which gives the minimum possible value for ρ, and which marks the left starting point of the graphs in Fig.6 as function of q.

A path \( P_i \) is a graph with vertices \( \{x_1, \ldots, x_n\} \) and edges \( \{x_1, x_2 x_3, \ldots, x_{n-1} x_n\} \).
Figure 7. Example 7.2: (a)-(c): Bipartite primal graphs for DNF \( \varphi \) and two dissociations. Notice that the primal graphs of \( \varphi' \) and \( \varphi'' \) are forests and thus correspond to read-once expressions. (d): For a given dissociation \( d \), there is only one optimal oblivious upper bound but infinitely many optimal lower bounds. We evaluate \( \mathbb{P}[\varphi'] \) for three of the latter (two of which coincide with models \( m \)) and keep the maximum as the best oblivious lower bound. (e): In comparison, dissociation gives substantially better upper bounds than model-based bounds (0.73\% vs. 4.55\% average and 6.7\% vs. 298.3\% worst-case relative errors), yet lower bounds are only slightly better. (f): Relative errors of 4 approximations for individual data points sorted by the dissociation error for upper bounds and for lower bounds separately; this is why the dissociation errors show up as smooth curves (red) while the model-based errors are unsorted and thus ragged (blue). (g): Here we sorted the errors for each approximation individually; this is why all curves are smooth.

**Dissociation**: Analogously, we consider the possible dissociations into read-once formulas. For our given \( \varphi \), those are \( \varphi'_1 = (x_1 \lor x'_2,1) y_1 \lor x'_2,2 y_2 \) and \( \varphi'_2 = x_1 y'_{1,1} \lor x_2 (y'_{1,2} \lor y_2) \), with Fig. 7b and Fig. 7c illustrating the dissociated read-once primal graphs. From Corollary 4.9 we know that \( \mathbb{P}_{p,q}[\varphi'_1] \geq \mathbb{P}_{p,q}[\varphi] \) for the only optimal oblivious upper bounds \( p'_{2,1} = p'_{2,2} = p_2 \) and \( \mathbb{P}_{p,q}[\varphi'_1] \leq \mathbb{P}_{p,q}[\varphi] \) for any \( p'_2 \) with \( p'_{2,1} p'_{2,2} = p_2 \). In particular, we choose 3 alternative optimal oblivious lower bounds \( p'_3 \in \{(p_2,0), (1 - \sqrt{1-p_2},1 - \sqrt{1-p_2}), (0, p_2)\} \) (see Fig. 7d). Analogously \( \mathbb{P}_{p,q}[\varphi'_2] \geq \mathbb{P}_{p,q}[\varphi] \) for \( q'_{1,1} = q'_{1,2} = q_1 \) and \( \mathbb{P}_{p,q}[\varphi'_2] \leq \mathbb{P}_{p,q}[\varphi] \) for \( q'_1 \in \{(q_1,0), (1 - \sqrt{1-q_1},1 - \sqrt{1-q_1}), (0, q_1)\} \). For each \( \varphi \), we choose the minimum among the 2 upper bounds and the maximum among the 6 lower bounds as the best upper and lower dissociation-based bounds, respectively.

**Results**: Figures 7e-g show that dissociation-based bounds are always better or equal to model-based bounds. The reason is that all model-based bounds are a special case of oblivious dissociation bounds. Furthermore, dissociation gives far better upper bounds, but only slightly better lower bounds. The reason is illustrated in Fig. 7d: the single dissociation-based upper bound \( p' = (p,p) \) always dominates the two model-based upper bounds, whereas the two model-based lower bounds are special cases of infinitely many optimal oblivious lower dissociation-based bounds. As extreme case, it is therefore possible for a model-based lower bound to coincide with the best among all optimal oblivious

Note that we consider here dissociation on both \( x \)- and \( y \)-variables, thus do not treat them as distinct.
oblivious lower bound dissociations. For our example, we evaluate three oblivious lower bounds, two of which coincide with models.

### 7.3. Conjunctive versus Disjunctive Dissociation

**Example 7.3 (Disjunctive and Conjunctive Dissociation).** This example illustrates an interesting asymmetry: optimal upper bounds for disjunctive dissociations and optimal lower bounds for conjunctive dissociations are not only unique but also better, on average, than optimal upper bounds for conjunctive dissociations and optimal lower bounds for disjunctive dissociations, respectively (see Figure 1). We show this by comparing the approximation of a function by either dissociating a conjunctive or a disjunctive expression for the same function.

We re-use the setup from Example 7.2 where we had a function expressed by a disjunctive expression \( \varphi \). Our DNF \( \varphi \) can be written as CNF \( \psi = (x_1 \lor x_2)(y_1 \lor x_2)(y_1 \lor y_2) \) with \( f_2 = f_1 \) and two conjunctive dissociations \( \psi'_1 = (x_1 \lor x'_2)(y_1 \lor x'_2)(y_1 \lor y_2) \) and \( \psi'_2 = (x_1 \lor x'_2)(y'_1 \lor x_2)(y'_1 \lor y_2) \) (Figures 8a-c shows the primal graphs). Again from Corollary 4.9 we know that \( P_{p,q}[\varphi'_1] \leq P_{p,q}[\psi] \) for the only optimal oblivious lower bounds \( p_{2,1} = p_{2,2} = p_2 \) and \( P_{p,q}[\varphi'_1] \geq P_{p,q}[\varphi] \) for any \( p_2 \) with \( p_{2,1} = p_{2,2} = p_2 \). In particular, we choose 3 alternative optimal oblivious lower bounds \( p_{2} \in \{ p_2, 1 \} \) (see Fig. 7d). Analogously \( P_{p,q}[\varphi'_2] \leq P_{p,q}[\varphi] \) for

\[17\]

Notice that this transformation from DNF to CNF is hard, in general. We do not focus on algorithmic aspects in this paper, but rather show the potential of this new approach.

---

ACM Transactions on Database Systems, Vol. V. No. N, Article 1, Publication date: January YYYY.
Fig. 9. Examples of different query correspondences for the four DNFs 
(see Example 7.2). Figures 8e-g show that optimal disjunctive upper bounds are, in general but not consistently, better than optimal conjunctive upper bounds for the same function (≈ 83.5% of those data points with different approximations are better for conjunctive dissociations). The dual result holds for lower bounds. This duality can be best seen in the correspondences of absolute errors between upper and lower bounds.

7.4. Multiple Dissociations at Once

Here we investigate the influence of the primal graph and number of dissociations on the tightness of the bounds. Both examples correspond to the lineage of the standard unsafe query $Q : - R(X), S(X, Y), T(Y)$ over two different database instances.

Example 7.4 (Path $P_n$ as Primal Graph). This example considers a DNF expression whose primal graph forms a $P_n$, i.e. a path of length $n$ (see Fig. 9a). Note that this is a generalization of the path $P_3$ from Example 7.2 and corresponds to the lineage of the same unsafe query over larger database instance with $2n – 1$ tuples:

$$\varphi_n = x_1 y_1 \lor x_1 y_2 \lor x_2 y_2 \lor \ldots \lor x_{n-1} y_n \lor x_n y_n$$

$$\varphi'_n = x'_{1,1} y_1 \lor x'_{1,2} y_2 \lor x'_{2,1} y_2 \lor \ldots \lor x'_{n-1,2} y_n \lor x'_{n,1} y_n$$

Exact: In the following, we assume the probabilities of all variables to be $p$ and use the notation $p_n := P[\varphi_n]$ and $p'_n := P[\varphi'_n]$, where $\varphi'_n$ corresponds to the formula $\varphi_n$ without the last conjunct $x_n y_n$. We can then express $p_n$ as function of $p'_n$, $p_{n-1}$ and $p'_{n-1}$ by recursive application of Shannon’s expansion to $x_n$ and $y_n$:

$$p_n = P[x_n] (P[y_n] + P[y_n] p_{n-1}) + P[x_n] p'_n$$

$$p'_n = P[y_n] (P[x_{n-1}] + P[x_{n-1}] p'_{n-1}) + P[y_n] p_{n-1}$$

We thus get the linear recurrence system

$$p_n = A_1 p_{n-1} + B_1 p'_{n-1} + C_1$$

$$p'_n = A_2 p_{n-1} + B_2 p'_{n-1} + C_2$$
with $A_1 = \bar{p}$, $B_1 = pp^2$, $C_1 = p^2(2 - p)$, $A_2 = \bar{p}$, $B_2 = \bar{p}p$, and $C_2 = p^2$. With a few manipulations, this recurrence system can be transformed into a linear non-homogenous recurrence relation of second order $p_n = Ap_{n-1} + Bp_{n-2} + C$ where $A = A_1 + B_2 = p(1 + p)$, $B = A_2B_1 - A_1B_2 = -p^2p^2$, and $C = B_1C_2 + C_1(1 - B_2) = p^2(p^2 + (2 - p)(1 - p))$.

Thus we can recursively calculate $\mathbb{P}[\varphi_n]$ for any probability assignment $p$ starting with initial values $p_1 = p^2$ and $p_2 = 3p^2 - 2p^3$.

**Dissociation:** Figure 9b shows the primal graph for the dissociation $\varphi'_{n}$. Variables $x_1$ to $x_{n-1}$ are dissociated into two variables with same probability $p'$, whereas $x_n$ into one with original probability $p$. In other words, with increasing $n$, there are more variables dissociated into two fresh ones each. The probability $\mathbb{P}[\varphi'_{n}]$ is then equal to the probability that at least one variable $x_i$ is connected to one variable $y_j$:

$$\mathbb{P}[\varphi'_{n}] = 1 - (1 - pp')(1 - p(1 - \bar{p}^2))^{n - 2}(1 - p(1 - \bar{p}p'))$$

We set $p' = p$ for upper bounds, and $p' = 1 - \sqrt{1 - p}$ for lower bounds.

**Results:** Figures 9c-d shows the interesting result that the disjunctive upper bounds become tight for increasing size of the primal graph, and thus increasing number of dissociations. This can be best seen in Fig. 9d for which $p$ is chosen as to keep $\mathbb{P}[\varphi] = 0.5$ constant for varying $n$ and we have $\lim_{n \to \infty} \mathbb{P}[\varphi'_n] = \mathbb{P}[\varphi_n] = 0.5$ for upper bounds. In contrast, the disjunctive lower bounds become weaker but still have a limit value $\lim_{n \to \infty} \mathbb{P}[\varphi'_n] \approx 0.2929$ (derived numerically).

**Example 7.5 (Complete bipartite graph $K_{n,n}$ as Primal Graph).** This example considers a DNF whose primal graph forms a complete bipartite graph of size $n$, i.e. each variable $x_i$ is appearing in one clause with each variable $y_j$ (see Fig. 10a). Note that this example corresponds to lineage for the standard unsafe query over a database instance with $O(n^2)$ tuples:

$$\varphi_n = \bigvee_{(i,j) \in [n]^2} x_i y_j$$

$$\varphi'_n = \bigvee_{j \in [n]} \left( y_j \bigvee_{i \in [n]} x'_{i,j} \right)$$

**Exact:** We again assume that all variables have the same probability $p = \mathbb{P}[x_i] = \mathbb{P}[y_j]$. $\mathbb{P}[\varphi'_n]$ is then equal to the probability that there is at least one tuple $x_i$ and at
select distinct s.nationkey
from Supplier, Partsupp, Part
where s.supkey = ps.supkey
and ps.partkey = p.partkey
and s.supkey <= $1
and p.name like "$2"

(b) Relevant TPC-H schema

(a) Deterministic query Q(a)

Fig. 11. Example 7.6 Parameterized SQL query Q(a) and relevant portion of the TPC-H schema.

least one tuple $y_i$:

$$P[\varphi_n] = (1 - (1 - p)^n)^2 \quad (6)$$

**Dissociation:** Figure 10b shows the primal graph for the dissociation $\varphi'_n$. Each variable $x_i$ is dissociated into $n$ fresh variables with same probability $p'$, i.e. there are $n^2$ fresh variables in total. The probability $P[\varphi'_n]$ is then equal to the probability that at least one variable $y_i$ is connected to one variables $x_{i,j}$:

$$P[\varphi'_n] = 1 - \left(1 - p(1 - (1 - p')^n)\right)^n$$

We will again choose $p$ as to keep $r := P[\varphi_n]$ constant with increasing $n$, and then calculate $P[\varphi'_n]$ as function of $r$. From Eq. 6, we get $p = 1 - \sqrt[5]{1 - \sqrt[r]}$ and then set $p' = p$ for upper bounds, and $p' = 1 - \sqrt[5]{1 - p}$ for lower bounds as each dissociated variable is replaced by $n$ fresh variables. It can then be shown that $P[\varphi'_n]$ for the upper bound is monotonically increasing for $n$ and bounded below 1 with the limit value:

$$\lim_{n \to \infty} P[\varphi'_n] = 1 - (1 - \sqrt{r})^{\sqrt{r}}$$

**Results:** Figure 10d keeps $P[\varphi] = 0.5$ constant (by decreasing $p$ for increasing $n$) and shows the interesting result that the optimal upper bound is itself upper bounded and reaches a limit value, although there are more variables dissociated, and each variable is dissociated into more fresh ones. This limit value is 0.5803 for $r = 0.5$. However, lower bounds are not useful in this case.

7.5. Dissociation with a Standard Relational Database Management System

**Example 7.6 (TPC-H).** Here we apply the theory of dissociation to bound hard probabilistic queries with the help of PostgreSQL 9.2, an open-source relational DMBS. We use the TPC-H DBGEN data generator to generate a 1GB database. We then add a column $P$ to each table, and assign to each tuple either the probability $p = 0.1$, or $p = 0.5$, or a random probability from the set $\{0.01, 0.02, \ldots, 0.5\}$ ($p = \text{rand} 0.5$). Choosing tuple probabilities $p \leq 0.5$ helps us avoid floating-point errors for queries with very large lineages whose query answer probabilities would otherwise be too close to 1.\(^2\) Our experiments use the following parameterized query (Fig. 11):

$$Q(a) \leftarrow S(s, a), PS(s, u), P(u, n), s \leq $1, n like "$2$$

Relations $S$, $PS$ and $P$ represent tables Supplier, Partsupp and Part, respectively. Variable $a$ stands for attribute nationkey (“answer tuple”), $s$ for suppkey, $u$ for partkey (“unit”),

\(^1\)http://www.postgresql.org/download/
\(^2\)Compare Fig. 13b and Fig. 13e to observe the impact of the choice of input tuple probabilities $p$ on the probabilities of query answers for the same query.
create view VP as
select p.partkey, s.nationkey,
    1-power(1-P.P.1e6/count(1)) as P
from Part P, Partsupp PS, Supplier S
where p.partkey=ps.partkey
and ps.supplykey = s.supplykey
and s.name like $1
and p.name like $2

group by p.partkey, s.nationkey, P.P

select s.nationkey, IOR(Q3.P) as P
from (select S.s.nationkey, S.P*Q2.P as P
    from Supplier S,
        (select Q1.ps.supplykey, s.nationkey, IOR(Q1.P) as P
            from Partsupp PS, VP
            where ps.partkey = p.partkey
                  and ps.supplykey <= $1)
        as Q1
    group by Q1.s.nationkey) as Q2

where s.supplykey = Q2.ps.supplykey) as Q3

group by Q3.s.nationkey

(a) View VP for lower bounds with $P_P^*$

(b) Query $P_P^*$

Fig. 12. Example 7.6. (a) View definition VP and (b) and query $P_P^*$ for deriving the lower bounds by dissociating table $P$. Note the inclusion of the attribute nationkey in VP for reasons explained in the text.

and n for name. The probabilistic version of this query asks which nations (as determined by the attribute nationkey) are most likely to have suppliers with supplykey ≤ $1 that supply parts with a name like $1 when all tuples in Supplier, Partsupp, and Part are probabilistic. Parameters $1$ and $2$ allow us to reduce the number of tuples that can participate from tables Supplier and Part, respectively, and to thus study the effects of the lineage size on the predicted dissociation bounds and running times. By default, tables Supplier, Partsupp and Part have 10k, 800k and 200k tuples, respectively, and there are 25 different numeric attributes for nationkey. For parameter $1$, we choose a value $\in \{500, 1000, \ldots, 10000\}$ which selects the respective number of tuples from table Supplier. For parameter $2$, we choose a value $\in \{\%,'\%red','\%red%green%'\}$ which selects 200k, 11k or 251 tuples in table Part, respectively.

**Translation into SQL**: Note that the lineage for each individual query answer corresponds to the lineage for the Boolean query $Q$ from Sect. 6, which we know is hard, in general. We thus bound the probabilities of the query answers by evaluating four different queries that correspond to the query-centric dissociation bounds from Sect. 6 dissociating either table Supplier or table Part, and calculating either upper and lower bounds. To get the final upper (lower) bounds, we take the minimum (maximum) of the two (upper) bounds for each answer tuple. The two query plans are as follows:

\[
P_P^*(a) = \pi_{\text{partkey}}^u \times_P^\pi [S(s, a), PS(s, u), s \leq $1], P(u, n), n \text{ like $2$}
\]

\[
P_P^*(a) = \pi_{\text{nationkey}}^u \times_P^\pi [S(s, a), PS(s, u), s \leq $1], P(u, n), n \text{ like $2$}
\]

Notice one technical detail for determining the lower bounds with plan $P_P$: Any tuple $t$ from table Part may appear a different number of times in the lineages for different query answers. Thus, for every answer $a$ that has $t$ in its lineage, we need to create a distinct copy of $t$ in the view $V_P$, with a probability that depends only on the number of times that $a$ appears in the lineage for $a$. Thus, the view definition for $V_P$ needs to include the attribute nationkey (Fig. 12a) and $P_P$ needs to be adapted as follows (Fig. 12b):

\[
P_P^*(a) = \pi_{\text{nationkey}}^u \times_P^\pi [S(s, a), PS(s, u), s \leq $1], V_P(u, a)]
\]

21Lower bounds by dissociating Supplier are easier since the table includes the query answer attribute nationkey. As consequence, any tuple from Supplier may appear in the lineage of one query answer only.

22We could actually use the total number of times the tuple appears in all lineages and still get lower bounds. However, the resulting lower bounds would be weaker and not obliviously optimal.
In order to speed up the resulting multi-query evaluation, we apply a deterministic semi-join reduction on the input tables and then reuse intermediate query results across all four subsequent queries. Since query optimization is not the focus of this paper, we do not show these techniques in Fig. 12 and instead refer to [Gatterbauer and Suciu 2013] for a detailed discussion of techniques to speed up query dissociation. In addition, the exact SQL statements that allow the interested reader to repeat our experiments with PostgreSQL are available on the LaPushDB project page.²

²http://LaPushDB.com/
Ground truth: To compare our bounds against ground truth, we construct the lineage DNF for each query answer and use DeMorgan to write it as “dual lineage CNF” without exponential increase in size. For example, the DNF $\varphi = x_1 x_3 \lor x_1 x_2$ can be written as CNF $\bar{\varphi} = (\bar{x}_1 \lor \bar{x}_3) \land (\bar{x}_1 \lor \bar{x}_2)$ with $P[\bar{\varphi}] = 1 - P[\varphi]$. The problem of evaluating a probabilistic CNF can further be translated into the problem of computing the partition function of a propositional Markov random field, for which the AI community has developed sophisticated solvers. SampleSearch [Gogate and Domingos 2010; Gogate and Dechter 2011] has been developed using this approach.

Quality Results: Figure 13 shows the top 10 query answers, as predicted by the upper dissociation bounds for varying query parameters $1$ and $2$, as well as varying tuple probabilities $p$. The crosses show the ground truth probabilities as determined by SampleSearch. The red intervals show the interval between upper and lower dissociation bounds. Recall that the final dissociation interval is the intersection between the interval from dissociation on Supplier (left of the red interval) and on Part (right of the red interval). The graphs suggest that the upper dissociation bounds are very close to the actual probabilities, which is reminiscent of Sect. 7.4 having shown that upper bounds for DNF dissociations are commonly closer to the true probabilities than lower bounds. For Fig. 13f, we have no ground truth as the lineage for the top tuple (11) has size 32899 (i.e. the corresponding DNF has 32899 clauses), which is too big for exact probabilistic inference. However, extrapolating from Fig. 13d and Fig. 13e to Fig. 13f, we speculate that upper dissociation bounds give good approximations here as well.

The different interval sizes (i.e. quality of the bounds) arise from different numbers of dissociated tuples in the respective lineages. We illustrate with 4 cases (Fig. 13a):

(A) If there is a plan that does not dissociate any tuple, then both upper and lower bounds coincide. This scenario is also called data-safe [Jha et al. 2010]. For example, the lineage for answer (2) in Fig. 13c has 40 unique tuples from table Part, out of which 3 (7.5%) are dissociated into 2 fresh ones with $P_S$. However, all of the 43 tuples from table Supplier that appear in the lineage appear only once. Therefore, $P_S$ dissociates no tuple when calculating the answer probability for (2), and as a result gives us the exact value.

(B) The lineage for the top-ranked answer (6) in Fig. 13a has 42 unique tuples from table Part, out of which 5 (= 11.9%) are dissociated into 2 fresh ones with $P_S$. In contrast, the lineage has 53 unique tuples from table Supplier, out of which only 4 (= 7.5%) are dissociated into 2 fresh ones with $P_S$. Intuitively, $P_S$ should give us therefore tighter bounds, which is confirmed by the results.

(C) The lineage for the top-ranked answer (11) in Fig. 13b and Fig. 13e (both figures show results for the same query, but for different tuple probabilities $p = 0.1$ or $p = \text{rand} 0.5$) has 1830 unique tuples from table Part, out of which $\approx 5.8\%$ are dissociated into 2 or 3 fresh ones with $P_S$. In contrast, the lineage has only 434 unique tuples from table Supplier, out of which nearly $95.6\%$ are dissociated into 2 to 11 fresh variables. Thus, $P_S$ gives far tighter bounds in this case.

(D) The lineage for the same answer (11) in Fig. 13f has 32899 unique tuples from table Part, out of which $\approx 6.3\%$ are dissociated into 2-4 fresh ones with $P_S$. In contrast, the lineage has only 438 unique tuples from table Supplier, out of which all are dissociated into 80 fresh ones with $P_S$ (this is an artifact of the TPC-H random database generator). Thus, the bounds for $P_S$ are very poor.

Importantly, relevance ranking of the answer tuples by upper dissociation bounds approximates the ranking by query reliability very well. For example, for the case of $p = 0.1$ and $2 = \text{%red%}$ (Fig. 13d and Fig. 13e), the ranking given by the minimum upper bounds was identical to the ranking given by the ground truth for all parameter

---

2 http://www.hlt.utdallas.edu/~vgogate/SampleSearch.html
Example 7.6: Timing results for queries with varying parameters $\$1$ and $\$2$: For small lineage (\(<\ 10000$), dissociation bounds can be calculated in a small multiple (\(<\ 4$) of the time needed for a standard deterministic query. For large lineages (\(>\ 10000$), calculation scales linearly in the size of the total lineage.

Timing Results: Figure 14 compares the times for evaluating the deterministic query (Fig. 11a) with the times for calculating the dissociation bounds for changing parameters $\$1$ and $\$2$. As experimental platform, we use PostgreSQL 9.2 on a 2.5 Ghz Intel Core i5 with 16G of main memory. We run each query 5 times and take the average execution time. Figure 14d also shows the size of the total lineage of a query (which is the same as the number of query results for the deterministic query without projection) and the times needed by SampleSearch to evaluate the ground truth, if possible. Since table Supplier contains exactly 10k tuples with suppkey $\in\{1,\ldots,10000\}$, any choice of $\$1\geq10000$ has no effect on the query. We show separate graphs for the time needed to calculate the upper bounds only (which our theory and experiments suggest give better absolute approximations) and the time for both upper and lower bounds (lower bounds are more expensive due to the required manipulation of the input tuples). We also show the times for retrieving the lineage with a lineage query. Any probabilistic approach that evaluates the probabilities outside of the database engine needs to issue this query to retrieve the DNF for each answer. The time needed for the lineage query thus serves as minimum benchmark for any probabilistic approximation.

Our timing results show that, for small lineages ($\(<\ 10000$), calculating upper and lower bounds can be achieved in a time that is only a small multiple ($\(<\ 4$) of the time needed for an equivalent deterministic query. For large lineages ($\(>\ 10000$), calculating the bounds scales linearly with the size of the lineage (Fig. 14c), whereas determin-

\footnote{The reported times are for evaluating all answer DNFs without the overhead for the lineage query.}
Oblivious Bounds on the Probability of Boolean Functions

Probabilistic query evaluation can scale in sublinear time (recall that the cardinality of the answer set is maximal 25 across all queries since the database contains only 25 different values for the answer attribute nationkey). Importantly, scalability for dissociation is independent of the tractability of the data instance, e.g. maximum treewidth of the lineage for any query answer. In contrast, SampleSearch quickly takes too long for increasing lineage. For example, SampleSearch needs 108 sec for calculating the ground truth for Fig. 13e (The maximum lineage size among all 25 query answers is 1941 in this scenario). Upper dissociation bounds can be calculated in only 3 sec and give the same ranking (and almost the same probabilities).

Key take-aways: Overall, our quality and timing experiments suggest that dissociation bounds (in particular, the upper bounds) can provide a good approximation of the actual probabilities and provide a ranking of the query answers that is often identical to the ranking for their actual probabilities. These bounds can be calculated with guaranteed polynomial scalability in the size of the data. In particular for queries with small lineage sizes (< 10000), calculating the bounds took only a small multiple (< 4) of the time needed to evaluate standard deterministic queries.

8. RELATED WORK AND DISCUSSION

Dissociation is related to a number of recent approaches in the graphical models and constraint satisfaction literature which approximate an intractable problem with a tractable relaxed version after treating multiple occurrences of variables or nodes as independent or ignoring some equivalence constraints: Choi et al. [2007] approximate inference in Bayesian networks by “node splitting,” i.e. removing some dependencies from the original model. Ramirez and Geffner [2007] treat the problem of obtaining a minimum cost satisfying assignment of a CNF formula by “variable renaming,” i.e. replacing a variable that appears in many clauses by many fresh new variables that appear in few. Pipatsrisawat and Darwiche [2007] provide lower bounds for MaxSAT by “variable splitting,” i.e. compiling a relaxation of the original CNF. Andersen et al. [2007] improve the relaxation for constraint satisfaction problems by “refinement through node splitting,” i.e. making explicit some interactions between variables. Choi and Darwiche [2009] relax the maximum a posteriori (MAP) problem in probabilistic graphical models by dropping equivalence constraints and partially compensating for the relaxation. Our work provides a general framework for approximating the probability of Boolean functions with both upper and lower bounds. These bounds shed light on the connection between previous relaxation-based and model-based approximations and unify them as concrete choices in a larger design space. We thus refer to all of the above approaches as instances of dissociation-based approximations.

Another line of work that is varying called “discretization,” “bucketing,” “binning,” or “quantization” proposes relaxations by merging or partitioning states or nodes (instead of splitting them), and to then perform simplified calculations over those partitions. Dechter and Rish [2003] approximate a function with high arity by a collection of smaller-arity functions with a parameterized scheme called “mini-buckets.” As example, the sum \( \sum_i f(x_i) \cdot g(x_i) \) for non-negative functions \( f \) and \( g \) can be upper bounded by the summation \( (\max_i f(x_i)) \cdot \sum_i g(x_i) \), i.e. all different values of \( f(x_i) \) are replaced by the single maximum value \( \max_i f(x_i) \), which simplifies the calculations. Similarly, the sum can be lower bounded by \( (\min_i f(x_i)) \cdot \sum_i g(x_i) \). St-Aubin et al. [2000] use Algebraic Decision Diagrams (ADDs) and reduce the sizes of the intermediate value functions generated by replacing the values at the terminals of the ADD with ranges of values. Bergman et al. [2011], [2013] construct relaxations of Multivalued Decision Diagrams (MDDs) by merging vertices when the size of the partially constructed MDD grows too large. Gogate and Domingos [2011] compress potentials computed during the execution of variable elimination by “quantizing” them, i.e. replacing a number...
of distinct values in the range of a function, we collectively refer to them as quantization-based approximations. A more detailed literature overview in this space is given by Gogate and Domingos [2013].

Note that dissociation-based approaches (that split nodes) and quantization-based approaches (that merge nodes) are not inverse operations, but are rather two complementary approaches that may be combined to yield improved methods. The inverse of dissociation is what we refer to as assimilation [26]. Consider the Boolean formula \( \phi = x_1 y_1 \lor x_2 y_2 \lor x_2 y_2 \) which is a dissociation of the much simpler formula \( \phi^* = x(y_1 \lor y_2) \). Hence, we know from dissociation that \( \mathbb{P}[\phi^*] \leq \mathbb{P}[\phi] \) for \( p \leq \min(p_1, p_2) \) and that \( \mathbb{P}[\phi^*] \geq \mathbb{P}[\phi] \) for \( p \geq 1 - \bar{p}_1 \bar{p}_2 \). Note that for quantization, we can always choose \( \max(p_1, p_2) \) or \( \min(p_1, p_2) \) to get a guaranteed upper or lower bound. In contrast, for assimilation, we may have to choose a different value to get a guaranteed bound. Also, for the case \( p_1 = p_2 \), assimilation will generally still be an approximation, whereas quantization would be exact. Thus, these are two different approaches.

Existing approaches for query processing over probabilistic databases that are both general and tractable are either: (1) simulation-based approaches that adapt general purpose sampling methods [Jampani et al. 2008; Kennedy and Koch 2010; Re et al. 2007]; or (2) model-based approaches that approximate the original number of models with guaranteed lower or upper bounds [Olteanu et al. 2010; Fink and Olteanu 2011]. We have show in this paper that, for every model-based bound, there exists a dissociation bound that is at least as good or better.

Our work on dissociation originated while generalizing propagation-based ranking methods from graphs [Detwiler et al. 2009] to hypergraphs and conjunctive queries. In [Gatterbauer et al. 2010], we applied dissociation in a query-centric way to upper bound hard probabilistic queries, and showed the connection between propagation on graphs and dissociation on hypergraphs (see [Gatterbauer and Suciu 2013] for all details). In this paper, we provide the theoretical underpinnings of these results in a generalized framework with both upper and lower bounds. A previous version of this paper was made available as [Gatterbauer and Suciu 2011].

9. OUTLOOK

We introduced dissociation as a new algebraic technique for approximating the probability of Boolean functions. We applied this technique to derive obliviously optimal upper and lower bounds for conjunctive and disjunctive dissociations and proved that dissociation always gives equally good or better approximations than models. We did not address the algorithmic complexities of exploring the space of alternative dissociations, but rather see our technique as a basic building block for new algorithmic approaches. Such future approaches can apply dissociation at two conceptual levels: (1) at the query-level, i.e. at query time and before analyzing the data, or (2) at the data-level, i.e. while analyzing the data.

The advantage of query-centric approaches is that they can run in guaranteed polynomial time [27] yet at the cost of no general approximation guarantees [28]. Here, we envision that the query-centric, first-order logic-based view of operating on data by the

---

26 We prefer the word “assimilation” as inverse of dissociation over of the more natural choice of “association” as it implies correctly that two items are actually merged and not merely associated.

27 Recall from our experiments [Fig. 14c] that query-centric dissociation scales linearly in the size of the lineage, independent of intricacies in the data, such as the treewidth of the lineage.

28 However, also recall from our experiments [Fig. 13] that query-centric approaches may work well in practice. Notice here the similarity to loopy belief propagation [Frey and MacKay 1997], which is applied widely and successfully, despite lacking general performance guarantees.
database community can also influence neighboring communities, in particular those working on lifted inference (see e.g. [Van den Broeck et al. 2011]).

The advantage of data-centric approaches is that exact solutions can be arbitrarily approximated, yet at the cost of no guaranteed runtime [Roth 1996]. Here, we envision a range of new approaches (that may combine dissociation with quantization) to compile an existing intractable formula into a tractable target language, e.g., read-once formulas or formulas with bounded treewidth. For example, one can imagine an approximation scheme that adds repeated dissociation to Shannon expansion in order to avoid the intermediate state explosion.

ACKNOWLEDGMENTS

We would like to thank Arthur Choi and Adnan Darwiche for helpful discussions on Relaxation & Compensation, and for bringing [Prop. 5.1] to our attention [Choi and Darwiche 2011]. We would also like to thank Vibhav Gogate for helpful discussions on quantization and guidance for using his tool SampleSearch, Alexandra Meliou for suggesting the name “dissociation,” and the reviewers for their careful reading of this manuscript and their detailed feedback. This work was partially supported by NSF grants IIS-0915054 and IIS-1115188. More information about this research, including the PostgreSQL statements to repeat the experiments on TPC-H data, can be found on the project page: http://LaPushDB.com/

REFERENCES


A. NOMENCLATURE

\[ x, y, z \] independent Boolean random variables
\[ \varphi, \psi \] Boolean formulas, probabilistic event expressions
\[ f, g, f_0 \] Boolean function, represented by an expression \( \varphi \)
\[ \mathbb{P}[x], \mathbb{P}[\varphi] \] probability of an event or expression
\[ p_i, q_i, r_i \] probabilities \( p_i = \mathbb{P}[x_i], q_i = \mathbb{P}[y_i], r_i = \mathbb{P}[z_i] \)
\[ x, g, p \] sets \( \{x_1, \ldots, x_k\} \) or vectors \( (x_1, \ldots, x_k) \) of variables, functions or probabilities
\[ \mathbb{P}_{p,q}[f] \] probability of function \( f(x, y) \) for \( p = \mathbb{P}[x], q = \mathbb{P}[y] \)
\[ \bar{x}, \bar{y}, \bar{p} \] complements \( \neg x, \neg y, 1 - p \)
\[ f', \varphi' \] dissociation of a function \( f \) or expression \( \varphi \)
\[ \theta \] substitution \( \theta : x' \mapsto x \); defines a dissociation \( f' \) of \( f \) if \( f'(\theta) = f \)
\[ f[x'/x] \] substitution of \( x' \) for \( x \) in \( f \)
\[ m, m', n \] \( m = |x|, m' = |x'|, n = |y| \)
\[ d_i \] number of new variables that \( x_i \) is dissociated into
\[ \nu \] valuation or truth assignment \( \nu : y \rightarrow \{0, 1\} \) with \( y_i = \nu_i \)
\[ f[v], \varphi[v] \] function \( f \) or expression \( \varphi \) with valuation \( \nu \) substituted for \( y \)
\[ g^* \] \[ g^* = \bigwedge_i g_i^* \] where \( g_i^* = g_i \) if \( \nu_i = 0 \) and \( g_i^* = \bar{g}_i \) if \( \nu_i = 1 \)

B. REPRESENTING COMPLEX EVENTS (DISCUSSION OF COROLLARY 4.5)

It is known from Poole’s independent choice logic [Poole 1993] that arbitrary correlations between events can be composed from disjoint-independent events only. A disjoint-independent event is represented by a non-Boolean independent random variable \( y \) which takes either of \( k \) values \( v = (v_1, \ldots, v_k) \) with respective probabilities \( q = (q_1, \ldots, q_k) \) and \( \sum_i q_i = 1 \). Poole writes such a “disjoint declaration” as \( y(v_1, \ldots, v_k) \).

In turn, any \( k \) disjoint events can be represented starting from \( k-1 \) independent Boolean variables \( z = (z_1, \ldots, z_{k-1}) \) and probabilities \( \mathbb{P}[z] = (q_1, q_2, q_3, \ldots, q_{k-1}) \), by assigning the disjoint-independent event variable \( y \) its value \( v_i \) whenever event \( A_i \) is true, with \( A_i \) defined as:

\[
(y = v_1) \equiv A_1 : z_1 \\
(y = v_2) \equiv A_2 : z_1 z_2 \\
\vdots \\
(y = v_{k-1}) \equiv A_{k-1} : z_1 \ldots z_{k-2} z_{k-1} \\
(y = v_k) \equiv A_k : z_1 \ldots z_{k-2} z_{k-1} .
\]

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

It follows that arbitrary correlations between events can be modeled starting from independent Boolean random variables alone. For example, two complex events \( A \) and \( B \) with \( \mathbb{P}[A] = \mathbb{P}[B] = q \) and varying correlation (see Sect. 7.1) can be represented as composed events \( A := z_1 z_2 \lor z_3 \lor z_4 \) and \( B := z_1 z_2 \lor z_3 \lor z_5 \) over the primitive events \( z \) with varying probabilities \( \mathbb{P}[z] \). Events \( A \) and \( B \) become identical for \( \mathbb{P}[z] = (0, 0, q, 0, 0) \), independent for \( \mathbb{P}[z] = (0, 0, 0, q, q) \), and disjoint for \( \mathbb{P}[z] = (0.5, q, 0, 0, 0) \) with \( q \leq 0.5 \).
Here we show the User-defined Aggregate (UDA) IOR in PostgreSQL:

```sql
create or replace function ior_sfunc(float, float) returns float as 'select $1 * (1.0 - $2)' language SQL;

create or replace function ior_finalfunc(float) returns float as 'select 1.0 - $1' language SQL;

create aggregate ior (float)(
    sfunt = ior_sfunc,
    stype = float,
    finalfunc = ior_finalfunc,
    initcond = '1.0');
```