The Dichotomy of Conjunctive Queries on Probabilistic Structures

Nilesh Dalvi and Dan Suciu
University of Washington, Seattle.

ABSTRACT

We show that for every conjunctive query, the complexity of evaluating it on a probabilistic database is either PTIME or #P-complete, and we give an algorithm for deciding whether a given conjunctive query is PTIME or #P-complete. The dichotomy property is a fundamental result on query evaluation on probabilistic databases and it gives a complete classification of the complexity of conjunctive queries.

1. PROBLEM STATEMENT

Fix a relational vocabulary $R_1, \ldots, R_k$, denoted $\mathcal{R}$. A *tuple-independent probabilistic structure* is a pair $(\mathbf{A}, p)$ where $\mathbf{A} = (A, R_1^A, \ldots, R_k^A)$ is first order structure and p is a function that associates to each tuple t in $\mathbf{A}$ a rational number $p(t) \in [0, 1]$. A probabilistic structure $(\mathbf{A}, p)$ induces a probability distribution on the set of substructures $\mathbf{B}$ of $\mathbf{A}$ by:

$$p(\mathbf{B}) \quad = \quad \prod_{i=1}^{k} \Big(\prod_{t \in R_i^B} p(t) \times \prod_{t \in R_i^A - R_i^B} (1 - p(t)) \Big) \qquad (1)$$

where $\mathbf{B} \subseteq \mathbf{A}$, more precisely $\mathbf{B} = (A, R_1^B, \ldots, B_k^B)$ is s.t. $R_i^B \subseteq R_i^A$ for $i = 1, k$.

A *conjunctive query*, q, is a sentence of the form $\exists \bar{x}.(\varphi_1 \wedge \ldots \wedge \varphi_m)$, where each φ_i is a positive atomic predicate $R(t)$, called a *sub-goal*, and the tuple t consists of variables and/or constants. As usual, we drop the existential quantifiers and the $\wedge$, writing $q = \varphi_1, \varphi_2, \ldots, \varphi_m$. A *conjunctive property* is a property on structures defined by a conjunctive query q, and its probability on a probabilistic structure $(\mathbf{A}, p)$ is defined as:

$$p(q) \quad = \quad \sum_{\mathbf{B} \subseteq \mathbf{A} : \mathbf{B} \models q} p(\mathbf{B}) \qquad (2)$$

In this paper we study the data complexity of Boolean conjunctive properties on tuple independent probabilistic structures. (When clear from the context we blur the distinction between queries and properties).

More precisely, for a fixed vocabulary and a Boolean conjunctive query q we study the following problem:

EVALUATION For a given probabilistic structure $(\mathbf{A}, p)$, compute the probability $p(q)$.

The complexity is in the size of $\mathbf{A}$ and in the size of the representations of the rational numbers $p(t)$. This problem is trivially contained in #P, and we show conditions under which it is in PTIME, and conditions where it is #P-hard. The class #P [11] is the counting analogue of the class NP.

THEOREM 1.1. *(Dichotomy Theorem) Given any conjunctive query q, the complexity of* EVALUATION *is either* PTIME *or #P-complete.*

Background and motivation Dichotomy theorems are fundamental to our understanding of the structure of conjunctive queries. A widely studied problem, which can be viewed as the dual of our problem, is the *constraint satisfaction problem* (CSP) and is as follows: given a fixed relational structure, what is the complexity of evaluating conjunctive queries over the structure? Shaefer [10] has shown that over binary domains, CSP has a dichotomy into PTIME and NP-complete. Feder and Vardi [5] have conjectured that a similar dichotomy holds for arbitrary (non-binary) domains. Creignou and Hermann [3] showed that the counting version of the CSP problem has a dichotomy into PTIME and #P-complete. The problem we study in this paper seems different in nature, yet still interesting.

In addition to the pure theoretical interest we also have a practical motivation. Probabilistic databases are increasingly used to manage a wide range of imprecise data [12, 2]. But general purpose probabilistic database are difficult to build, because query evaluation is difficult: it is both theoretically hard (#P-hard [7, 4]) and plain difficult to understand. All systems reported in the literature have circumvented the full query evaluation problem by either severely restricting the queries [1], or by using a non-scalable (exponential) evaluation algorithm [6], or by using a weaker semantics based on intervals [8]. In our own system, MystiQ [2], we support arbitrary conjunctive queries as follows. For queries without self-joins, we test if they have a PTIME plan using the techniques in [9]; if not, then we run a Monte Carlo simulation algorithm. The query execution times between the two cases differ by one or two orders of magnitude (seconds v.s. minutes). The desire to improve MystiQ's query performance on arbitrary queries (i.e. with self-joins) has partially motivated this work.

1.1 Overview of Results

We summarize here our main results on the query evaluation problem. Some of this discussion is informal and is intended to introduce the major concepts needed to understand the evaluation of conjunctive queries on probabilistic structures.

Hierarchical queries: For a conjunctive query q, let $Vars(q)$ denote its set of variables, and, for $x \in Vars(q)$, let $sg(x)$ be the set of sub-goals that contain x.

DEFINITION 1.2. *A conjunctive query is* hierarchical *if for any two variables x, y, either $sg(x) \cap sg(y) = \emptyset$, or $sg(x) \subseteq sg(y)$, or $sg(y) \subseteq sg(x)$. We write $x \sqsubseteq y$ whenever $sg(x) \subseteq sg(y)$ and write $x \equiv y$ when $sq(x) = sg(y)$. A conjunctive property is hierarchical if it is defined by some hierarchical conjunctive query.*

It is easy to check that a conjunctive property is hierarchical if the minimal conjunctive query defining it is hierarchical. As an example, the query $q_{\text{hier}} = R(x), S(x, y)$ is hierarchical because $sg(x) = \{R, S\}$, $sg(y) = \{S\}$. On the other hand, the query $q_{\text{non-h}} = R(x), S(x, y), T(y)$ is not hierarchical because $sg(x) = \{R, S\}$ and $sg(y) = \{S, T\}$.

In prior work [4] we have studied the evaluation problem under the following restriction: every sub-goal of q refers to a different relation name. We say that q has no self-joins. The main result in [4], restated in the terminology used here, is:

THEOREM 1.3. *[4] Assume q has no self joins. Then: (1) If q is hierarchical, then it is in* PTIME. *(2) If q is not hierarchical then it is #*P*-hard.*

Moreover, the PTIME algorithm for a hierarchical query is the following simple recurrence on query's structure. Call a variable x *maximal* if for all y, $y \sqsupseteq x$ implies $x \sqsupseteq y$. Pick a maximal variable from each connected component of the query to obtain the set $x_1, \ldots, x_m$. Let $f_0, f_1(x_1), \ldots, f_m(x_m)$ be the connected components of q: f_0 contains all constant sub-goals, and $f_i(x_i)$ consists of all sub-goals containing x_i for $i = 1, m$. Then:

$$p(q) \quad = \quad p(f_0) \cdot \prod_{i=1,m} (1 - \prod_{a \in A} (1 - p(f_i[a/x_i]))) \qquad (3)$$

This formula is a recurrence on the query's structure (since each $f_i[a/x_i]$ is simpler than q) and it is correct because $f_i[a/x_i]$ is independent from $f_j[a'/x_j]$ whenever $i \neq j$ or $a \neq a'$. As an example, for query $q_{\text{hier}} = R(x), S(x, y)$, $p(q) = 1 - \prod_{a \in A}(1 - p(R(a))(1 - \prod_{b \in A}(1 - p(S(a, b)))))$.

In this paper we study arbitrary conjunctive queries (i.e. allowing self-joins), which turn out to be significantly more complex. The starting point is the following extension of Theorem 1.3 (2) (the proof is in the appendix):

THEOREM 1.4. *If q is not hierarchical then it is #*P*-hard.*

Thus, from now on we consider only hierarchical conjunctive queries in this paper, unless otherwise stated.

Inversions: As a first contact with the issues raised by self-joins, let us consider the following query:

$$q \quad = \quad R(x), S(x, y), S(x', y'), T(x')$$

We write it as $q = f_1(x)f_2(x')$, where $f_1(x) = R(x), S(x, y)$ and $f_2(x') = S(x', y'), T(x')$. The query is hierarchical, but it has a self-join because the symbol S occurs twice: as a consequence $f_1[a/x]$ is no longer independent from $f_2[a/x']$ (they share common tuples of the form $S(a, b)$), which prevents us from applying Equation (3) directly. Our approach here is to define a new query by equating $x = x'$, $f_3(x) = f_1(x)f_2(x) = R(x), S(x, y), S(x, y'), T(x)$ which is equivalent to $R(x), S(x, y), T(x)$. We show that the probability $p(q)$ can be expressed using recurrences over the probabilities of queries of the form $f_1[a/x_1]$, $f_2[a'/x_2]$, $f_3[a''/x_3]$, as a sum of a few formulas[1] in the same style as (3) (see Example 3.8). The correctness is based on the fact that $f_i[a/x_i]$ and $f_j[a'/x_j]$ are independent if $i \neq j$ or $a \neq a'$.

However, this approach fails when the query has an "inversion". Consider:

$$H_0 \quad = \quad R(x), S(x, y), S(x', y'), T(y')$$

This query is hierarchical, but the above approach no longer works. The reason is that the two sub-goals $S(x, y)$ and $S(x', y')$ unify, while $x \sqsupset y$ and $x' \sqsubset y'$: we call this an *inversion* (formal definition is in Sec. 2.2). If we write H_0 as $f_1(x)f_2(y')$ and attempt to apply a recurrence formula, the queries $f_1[a/x]$ and $f_2[a'/y']$ are no longer independent even if $a \neq a'$, because they share the common tuple $S(a, a')$.

Inversions can occur as a result of a chain of unifications:

$$\begin{aligned} H_k = \\ &R(x), S_0(x, y), \\ &\quad S_0(u_1, v_1), S_1(u_1, v_1) \\ &\quad\quad S_1(u_2, v_2), \ldots \\ &\quad\quad\quad S_{k-1}(u_k, v_k), S_k(u_k, v_k) \\ &\quad\quad\quad\quad S_k(x', y'), T(y') \end{aligned}$$

Here any two consecutive pairs of variables in the sequence $x \sqsupset y$, $u_1 \equiv v_1$, $u_2 \equiv v_2$, $\ldots$, $x' \sqsubset y'$ unify, and we also call this an inversion. We prove in the Appendix:

THEOREM 1.5. *For every $k \geq 0$, H_k is #*P*-hard.*

Thus, some hierarchical queries with inversions are #P-hard. We prove, however, that if q has no inversions, then it is in PTIME:

THEOREM 1.6. *If q is hierarchical and has no inversions, then it is in* PTIME.

The PTIME algorithm for inversion-free queries is a sum of recurrence formulas, each similar in spirit to (3). The proof is in Sec. 3.2.

Erasers The precise boundary between PTIME and #P-hard queries is more subtle than simply testing for inversions: some queries with inversion are #P-hard, while others are in PTIME, as illustrated below:

Example 1.7 Consider the hierarchical query q

$$\begin{aligned} q = &R(r, x), S(r, x, y), U(a, r), U(r, z), V(r, z) \\ &\quad S(r', x', y'), T(r', y'), V(a, r') \\ &\quad R(a, b), S(a, b, c), U(a, a) \end{aligned}$$

[1]This particular example admits an alternative, perhaps simpler PTIME solution, based on a dynamic programming algorithm on the domain A. For other, very simple queries, we are not aware of any algorithm that is simpler than ours (formula (9), Sec. 3.2), for example $R(x, y, y, x), R(x, y, x, z)$, or $R(y, x, y, x, y), R(y, x, y, z, x), R(x, x, y, z, u)$ (both are in PTIME because they have no inversions). To appreciate the difficulties even with such simple queries note that, by contrast, $R(y, x, y, x, y), R(y, y, y, z, x), R(x, x, y, z, u)$ is #P-hard. For additional challenging PTIME queries, see Fig. 1.

Here a, b, c are constants and the rest are variables. This query has an inversion between $x \sqsupset y$ and $x' \sqsubset y'$ (when unifying $S(r, x, y)$ with $S(r', x', y')$). Because of this inversion, one may be tempted to try to prove that it is #P-hard, using a reduction from H_0. Our standard construction starts by equating $r = r'$ to make q "like" H_0: call q' the resulting query (i.e. $q' = q[r/r']$). If one works out the details of the reduction, one gets stuck by the existence of the following homomorphism from $h : q \to q'$ that "avoids the inversion": it maps the variables r, x, y, z, r', x', y' to a, b, c, r, r, x', y' respectively, in particular sending $U(r, z), V(r, z)$ to $U(a, r)$, $V(a, r)$. Thus, h takes advantage of the two sub-goals $U(a, r)$, $V(a, r)$ in q' which did not exists in q, and its image does not contain the sub-goal $S(r, x, y)$, which is part of the inversion. We call such a homomorphism an *eraser* for this inversion: the formal definition is in Sec. 2.3. Because of this eraser, we cannot use the inversion to prove that the query is #P-hard. So far this discussion suggests that erasers are just a technical annoyance that prevent us from proving hardness of some queries with inversions. But, quite remarkably, erasers can also be used in the opposite direction, to derive a PTIME algorithm: they are used to cancel out (hence "erase") the terms in a certain expansion of $p(q)$ that correspond to inversions and that do not have polynomial size closed forms. Thus, our final result (proven in Sections 3 and 4) is:

THEOREM 1.8 (DICHOTOMY). *Let q be hierarchical.*
(1) If q has an inversion without erasers then q is #P-hard.
(2) If all inversions of q have erasers then q is in PTIME.

As a non-trivial application of (1) we show (Fig. 2 in Appendix A and in Example 4.1) that each of the following two queries are #P-hard, since each has an inversion between two isomorphic copies of itself:

$$q_{\text{2path}} = R(x, y), R(y, z)$$
$$q_{\text{marked-ring}} = R(x), S(x, y), S(y, x)$$

In general, the hardness proof is by reduction from the query H_k, where k is the length of an inversion without an eraser. The proof is not straightforward. It turns out that not every eraser-free inversion can be used to show hardness. Instead we show that if there is an eraser-free inversion then there is one that admits a reduction from H_k.

The PTIME algorithm in (2) is also not straightforward at all. It is quite different from the recurrence formula in Theorem 1.6, since we can no longer iterate on the structure of the query: in Example 1.7, the sub-query of q consisting of the first two lines is #P-hard (since without the third line there is no eraser), hence we cannot compute it separately from the third line. Our algorithm here computes $p(q)$ without recurrence, and thus is quite different from the inversion-free PTIME algorithm, but uses the latter as a subroutine.

2. AN EXPANSION FORMULA FOR CONJUNCTIVE QUERIES

In this section, we introduce the key terminology and prove an expansion formula for computing the probability of conjunctive queries that will be used to device PTIME algorithms for query evaluation. For the remainder of the paper, all queries are assumed to be hierarchical, as we know that non-hierarchical queries are #P-hard (Appendix B).

2.1 Coverage

We call an *arithmetic predicate* a predicate of the form $u = v$, $u \neq v$, or $u < v$ between a variable and a constant in C, or between two variables[2]. A *restricted arithmetic predicate* is an arithmetic predicate that is either between a variable and a constant, or between two variables u, v that co-occur in some sub-goal (equivalently $u \sqsupseteq v$ or $u \sqsubseteq v$). From now on, we will allow all conjunctive queries to have restricted arithmetic predicates.

DEFINITION 2.1. *A* coverage *for a query q is a set of conjunctive queries $\mathcal{C} = \{qc_1, \ldots, qc_n\}$ such that:*

$$q \equiv qc_1 \vee \ldots \vee qc_n$$

Each query in $\mathcal{C}$ is called a cover. *A* factor *of $\mathcal{C}$ is a connected component of some $qc_i \in C$. We denote the set of all factors in $\mathcal{C}$ by $\mathcal{F} = \{f_1, \ldots, f_k\}$.*

We alternatively represent a coverage by the pair $(\mathcal{F}, C)$, where $\mathcal{F}$ is a set of factors and C is a set of subsets of $\mathcal{F}$. Each element of C determines a cover consisting of the corresponding set of factors from $\mathcal{F}$.

For any query q the set $\mathcal{C} = \{q\}$ is a trivial coverage. We also define $\mathcal{C}^<(q)$, which we call the *canonical coverage*, obtained as follows. Consider all m pairs (u, v) of co-occurring variables u, v in q, or of a variable u and constant v. For each such pair choose one of the following predicates: $u < v$ or $u = v$ or $u > v$, and add it to q. This results in 3^m queries. Remove the unsatisfiable ones, then remove all redundant ones (i.e. remove qc_i if there exists another qc_j s.t. $qc_i \subset qc_j$). The resulting set $\mathcal{C}^<(q) = \{qc_1, \ldots, qc_n\}$ is the *canonical coverage* of q.

Unifiers

Let q, q' be two queries (not necessarily distinct). We rename their variables to ensure that $Vars(q) \cap Vars(q') = \emptyset$, and write qq' for their conjunction. Let g and g' be two sub-goals in q and q' respectively. The *most general unifier*, MGU, of g and g' (or the MGU of q, q' when g, g' are clear from the context) is a substitution θ for qq' s.t. (a) $\theta(g) = \theta(g')$, (b) for any other substitution θ' s.t. $\theta'(g) = \theta'(g')$ there exists ρ s.t. $\rho \circ \theta = \theta'$.

A *1-1 substitution* for queries q, q' is a substitution θ for qq' such that: (a) for any variable x and constant a $\theta(x) \neq a$, and (b) for any two distinct variables x, y in q (or in q'), $\theta(x) \neq \theta(y)$. The *set representation* of a 1-1 substitution θ is the set $\{(x, y) \mid x \in Vars(q), y \in Vars(q'), \theta(x) = \theta(y)\}$.

DEFINITION 2.2. *An MGU θ for two queries q, q' is called* strict *if it is a 1-1 substitution for qq'.*

For a trivial illustration, if $q = R(x, x, y, a, z)$ and $q' = R(u, v, v, w, w)$ and their MGU is θ, then $\theta(x) = \theta(y) = \theta(u) = \theta(v) = x'$, $\theta(w) = \theta(z) = a$, and the effect of the unification is $\theta(qq') = R(x', x', x', a, a)$. This is not strict: e.g. $\theta(x) = \theta(y)$ and also $\theta(z) = a$. We want to ensure that all unifications are strict.

DEFINITION 2.3. *(Strict coverage) Let $\mathcal{C}$ be a coverage and $\mathcal{F}$ be its factors. We say that $\mathcal{C}$ is* strict *if any MGU between any two factors $f, f' \in \mathcal{F}$ is strict.*

[2]As usual we require every variable to be range restricted, i.e. to occur in at least one sub-goal.

Example 2.4 Let $q = T(x), R(x,x,y), R(u,v,v)$. The trivial coverage $\mathcal{C} = \{q\}$ is not strict, as the MGU of the two R sub-goals of q equate x with y and u with v. Alternatively, consider the following three queries:

$$\begin{aligned} qc_1 &= T(x), R(x,x,x) \\ qc_2 &= T(x), R(x,x,y), R(u,u,u), x \neq y \\ qc_3 &= T(x), R(x,x,y), R(u,v,v), x \neq y, u \neq v \end{aligned}$$

One can show that $q \equiv qc_1 \vee qc_2 \vee qc_3$, hence $\mathcal{C} = \{qc_1, qc_2, qc_3\}$ is a coverage for q. The set of factors $\mathcal{F}$ consists of the connected components of these queries, which are

$$\begin{array}{llll} f_1 = T(x), R(x,x,x) & & f_2 = T(x), R(x,x,y), x \neq y \\ f_3 = R(u,u,u) & & f_4 = R(u,v,v), u \neq v \end{array}$$

and $C = \{\{f_1\}, \{f_2, f_3\}, \{f_2, f_4\}\}$. The coverage is strict, as a unifier cannot equate x with y or u with v in any query because of the inequalities. Similarly, the canonical coverage $\mathcal{C}^<(q)$, which has nine covers containing combinations of $x < y$, $x = y$, or $x > y$ with $u < v$, $u = v$, $u > v$, is also strict.

LEMMA 2.5. *The canonical coverage $\mathcal{C}^<(q)$ is always strict.*

2.2 Inversions

Fix a strict coverage $\mathcal{C}$ for q, with factors $\mathcal{F}$, and define the following undirected graph G. Its nodes are triples (f, x, y) with $f \in \mathcal{F}$ and $x, y \in Vars(f)$, and its edges are pairs $((f,x,y),(f',x',y'))$ s.t. there exists two sub-goals g, g' in f, f' respectively whose MGU θ satisfies $\theta(x) = \theta(x')$ and $\theta(y) = \theta(y')$. We call an edge in G a *unification edge*, and a path a *unification path*. Recall that for a preorder relation $\sqsupseteq$, the notation $x \sqsupset y$ means $x \sqsupseteq y$ and $x \not\sqsubseteq y$.

DEFINITION 2.6. *(Inversion-free Coverage) An inversion in $\mathcal{C}$ is a unification path from a node (f,x,y) with $x \sqsupset y$ to a node (f',x',y') with $x' \sqsubset y'$. An* inversion-free coverage *is a strict coverage that does not have an inversion. We say that q is inversion-free if it has at least one inversion-free coverage. Otherwise, we say that q has inversion.*

Obviously, to check whether $\mathcal{C}$ has an inversion it suffices to look for a path in which all intermediate nodes are of the form (f'', u, v) with $u \equiv v$, i.e. the $\sqsupset$ and $\sqsubset$ are only at the two ends of the path. The following result says that to check if a query has an inversion, it is enough to examine the canonical coverage.

PROPOSITION 2.7. *If there exists one coverage of q that does not contain inversion, then the canonical cover $\mathcal{C}^<(q)$ does not contain inversion.*

Example 2.8 We illustrate with two examples:

(a) Consider H_k in Theorem 1.5. The trivial coverage $\mathcal{C} = \{H_k\}$ is strict, and has factors $\mathcal{F} = \{f_0, f_1, \ldots, f_{k+1}\}$ (each line in the definition of H_k is one factor). The following is an inversion: (f_0, x, y), (f_1, u_1, v_1), $\ldots$, (f_k, u_k, v_k), (f_{k+1}, x', y'). This is an inversion because $x \sqsupset y$ and $x' \sqsubset y'$. The canonical coverage $\mathcal{C}^<$ also has an inversion, e.g. along the factors obtained by adding the predicates $x < y$, $u_1 < v_1$, $\ldots$, $u_k < v_k$, $x' < y'$.

(b) Consider the query $q = R(x), S(x,y), S(y,x)$. The trivial coverage $\mathcal{C} = \{q\}$ is strict, has one factor $\mathcal{F} = \{q\}$, and there is an inversion from (q,x,y) to (q,y,x) because $S(x,y)$ unifies with $S(y,x)$ (recall that we rename the variables before the unification, i.e. the unifier is between $R(x)$, $\underline{S(x,y)}$, $S(y,x)$ and its copy $R(x'), S(x',y'), \underline{S(y',x')}$). In the canonical coverage $\mathcal{C}^<$ there are three factors, corresponding to $x < y$, $x = y$, and $y < x$, and the inversion is between $x < y$ and $y < x$.

2.3 An Expansion Formula for Coverage

Given a conjunctive query q and a probabilistic structure $\mathbf{A} = (A, R_1^A, \ldots, R_k^A)$, we want to compute the probability $p(q)$. Our main tool is a generalized inclusion-exclusion formula that we apply to the coverage of a query.

DEFINITION 2.9. *(Expansion Variables) Let $\mathcal{C} = (\mathcal{F}, C)$ be a strict coverage, where $\mathcal{F} = \{f_1, \cdots, f_k\}$ is a set of factors and C is a set of subsets of $\mathcal{F}$. A set of* expansion variables *is a set $\bar{x} = \{\bar{x}_{f_1}, \cdots, \bar{x}_{f_k}\}$ such that*

1. *$\bar{x}_{f_i} \subseteq Vars(f_i)$ for $1 \leq i \leq k$.*
2. *If $x \in \bar{x}_f$ and $x \sqsubset y$, then $y \in \bar{x}_f$.*
3. *Any MGU of any two factors f_i and f_j equates an expansion variable to an expansion variable.*

We use $(\mathcal{F}, C, \bar{x})$ to denote a coverage where we have chosen the expansion variables.

DEFINITION 2.10. *(Unary coverage) A coverage $(\mathcal{F}, C, \bar{x})$ is called a* unary coverage *if for each $f \in F$, $\bar{x}_f$ consists of a single variable r_f. We call r_f the root variable in f.*

By definition of expansion variables, the root variable must be the maximal element under $\sqsubset$ order, i.e. must occur in all the sub-goals of the corresponding factor.

Our first PTIME algorithm (for inversion-free queries) uses a unary coverage: the discussion in the next few subsections is much easier to follow if one assumes all coverages to be unary. Our second PTIME algorithm (for queries with erasable inversions) uses a coverage in which all variables are expansion variables, i.e. $\bar{x}_f = Vars(f)$: for that reason our discussion below needs to be more complex.

For $f \in \mathcal{F}$, let $A_f = A^{|\bar{x}_f|}$, and for $\bar{a} \in A_f$, let $f(\bar{a})$ denote the query $f[\bar{a}/\bar{x}_f]$, i.e., the conjunctive query obtained by substituting the variables $\bar{x}_f$ with $\bar{a}$. The following follows simply from the definitions:

$$q = \bigvee_{c \in C} \bigwedge_{f \in c} \bigvee_{\bar{a} \in A_f} f(\bar{a}) \tag{4}$$

Our next step is to apply the inclusion/exclusion formula to (4). We need some notations. We call a subset $\sigma \subseteq \mathcal{F}$ a *signature*. Given $s \subseteq C$, its signature is $sig(s) = \bigcup_{c \in s} c$.

DEFINITION 2.11. *Given a set $\sigma \subseteq \mathcal{F}$, define*

$$N(\mathcal{C}, \sigma) = (-1)^{|\sigma|} \sum_{s \subseteq C : sig(s) = \sigma} (-1)^{|s|}$$

For example, if $C = \{c_1, c_2, c_3\}$ where $c_1 = \{f_1, f_2\}$,$c_2 = \{f_2, f_3\}$ and $c_3 = \{f_1, f_3\}$, then for signature $\sigma = \{f_1, f_2, f_3\}$ we have $N(\sigma) = (-1)^{|\{f_1,f_2,f_3\}|}((-1)^{|\{c_1,c_2\}|} + (-1)^{|\{c_1,c_3\}|} + (-1)^{|\{c_2,c_3\}|} + (-1)^{|\{c_1,c_2,c_3\}|}) = -2$.

Given k sets $\bar{T} = \{T_{f_1}, \ldots, T_{f_k}\}$, where $T_{f_i} \subseteq A_{f_i}$, we denote its signature $sig(\bar{T}) = \{f \mid T_f \neq \emptyset\}$, its cardinality $|\bar{T}| = \sum_i |T_{f_i}|$, and denote $\mathcal{F}(\bar{T})$ the query $\bigwedge_{f \in \mathcal{F}} \bigwedge_{a \in T_f} f(\bar{a})$.

DEFINITION 2.12. *(Expansion) Given a coverage $\mathcal{C}$, define its expansion as*

$$Exp(\mathcal{C}) = \sum_{\bar{T}} N(\mathcal{C}, sig(\bar{T}))(-1)^{|\bar{T}|} p(\mathcal{F}(\bar{T})) \quad (5)$$

We prove the following in the appendix, using the inclusion/exclusion formula on (4):

THEOREM 2.13. *(Expansion Theorem) If $\mathcal{C}$ is a coverage for q, then $p(q) = Exp(\mathcal{C})$.*

Of course, Equation (5) is of exponential size. To reduce it, our first goal is to express $p(\mathcal{F}(\bar{T}))$ as the product $\prod_f \prod_{\bar{a} \in T_f} p(f(\bar{a}))$. For that we need to ensure that any two queries $f(\bar{a})$, $\bar{a} \in A_f$ and $f'(\bar{a}')$, $\bar{a}' \in A_{f'}$ are independent, and this does not hold in general. We will enforce this by restricting the sets $\bar{T}$ in Eq. (5) to satisfy some extra conditions, which we call *independence predicates*. We first illustrate independence predicates on a running example, then present them in the general case. Then we will move to our second goal: finding a closed form for the sum of products.

2.4 Running Example

We give the basic intuition for independence predicates using the following example.

Example 2.14 Consider the following query

$$q = P(x), R(x, y), R(x', y'), S(x')$$

and a coverage $\mathcal{C} = (\mathcal{F}, C, \bar{x})$ where $\mathcal{F}$ consists of the following three queries:

$$\begin{aligned} f_1 &= P(x_1), R(x_1, y_1) \\ f_2 &= R(x_2, y_2), S(x_2) \\ f_3 &= P(x_3), R(x_3, y_3), S(x_3) \end{aligned}$$

and $C = \{\{f_1, f_2\}, \{f_3\}\}$ and the expansion variables are $\bar{x}_{f_1} = \{x_1\}, \bar{x}_{f_2} = \{x_2\}, \bar{x}_{f_3} = \{x_3\}$. It is easy to verify that $\mathcal{C}$ defined here is indeed a coverage. (Here f_3 is redundant, i.e. $\{\{f_1, f_2\}\}$ is already a coverage. The reason why we include f_3 will become clear later.) The function N on signatures is as follows: $N(\mathcal{C}, \{f_1, f_2\}) = 1$, $N(\mathcal{C}, \{f_3\}) = N(\mathcal{C}, \{f_1, f_2, f_3\}) = -1$ and $N(\mathcal{C}, \sigma) = 0$ for all other σ. Thus, the inclusion-exclusion formula in Theorem 2.13 gives:

$$p(q) = \sum_{\bar{T}} N(\mathcal{C}, sig(\bar{T}))(-1)^{|\bar{T}|} p(\mathcal{F}(\bar{T})) \quad (6)$$

where $\bar{T}$ is a triplet of sets $\{T_1, T_2, T_3\}$, $|\bar{T}| = |T_1| + |T_2| + |T_3|$ and $\mathcal{F}(\bar{T}) = f_1(T_1) f_2(T_2) f_3(T_3)$. Consider now three sets T_1, T_2, T_3, and let's examine the query $\mathcal{F}(\bar{T})$. If $T_1 \cap T_2 = T_1 \cap T_3 = T_2 \cap T_3 = \emptyset$ then $f_i(a)$ is independent from $f_j(a')$, for all $i \neq j$, or for $i = j$ and $a \neq a'$. In this case $p(\mathcal{F}(\bar{T}))$ is a product $\prod_{i=1,3} \prod_{a \in A} p(f_i(a))$. We will ensure that the sets T_i are disjoint in two steps. First we will show:

$$p(q) = \sum_{\bar{T} | T_1 \cap T_2 = \emptyset} N(\mathcal{C}, sig(\bar{T}))(-1)^{|\bar{T}|} p(\mathcal{F}(\bar{T})) \quad (7)$$

Starting from Eq.(6) we note that $N(\mathcal{C}, sig(\bar{T})) \neq 0$ for only three signatures, hence $p(q) = p_1 + p_2 + p_3$, where

$$\begin{aligned} p_1 &= \textstyle\sum_{T_1 \neq \emptyset, T_2 \neq \emptyset, T_3 = \emptyset} (-1)^{|\bar{T}|} p(\mathcal{F}(\bar{T})) \\ p_2 &= -\textstyle\sum_{T_1 = T_2 = \emptyset, T_3 \neq \emptyset} (-1)^{|\bar{T}|} p(\mathcal{F}(\bar{T})) \\ p_3 &= -\textstyle\sum_{T_1 \neq \emptyset, T_2 \neq \emptyset, T_3 \neq \emptyset} (-1)^{|\bar{T}|} p(\mathcal{F}(\bar{T})) \end{aligned}$$

Let p_1^I and p_3^I denote the same sums as p_1 and p_3, but where $\bar{T}$ is restricted to satisfy $T_1 \cap T_2 = \emptyset$. To prove Equation (7), all we need is to show is that $p_1 + p_3 = p_1^I + p_3^I$. In the sum defining p_3 denote $T_3' = T_3 - T_1 \cap T_2$, $T_3'' = T_3 \cap T_1 \cap T_2$ (hence $T_3 = T_3' \cup T_3''$) and $\bar{T}' = (T_1, T_2, T_3')$. We have $p_3 =$

$$\begin{aligned} &= -\sum_{\substack{\bar{T}' \mid T_1 \neq \emptyset, T_2 \neq \emptyset \\ T_3' \cap T_1 \cap T_2 = \emptyset}} \sum_{\substack{T_3'' \subseteq T_1 \cap T_2 \\ T_3' \cup T_3'' \neq \emptyset}} (-1)^{|\bar{T}|} p(\mathcal{F}(\bar{T})) \\ &= -\sum_{\substack{\bar{T}' \mid T_1 \neq \emptyset, T_2 \neq \emptyset \\ T_3' \cap T_1 \cap T_2 = \emptyset}} (-1)^{|\bar{T}'|} p(\mathcal{F}(\bar{T}')) \sum_{\substack{T_3'' \subseteq T_1 \cap T_2 \\ T_3' \cup T_3'' \neq \emptyset}} (-1)^{|T_3''|} \\ &= p_3^I + 0 + (p_1^I - p_1) \end{aligned}$$

The first line simply splits the summation into a sum where T_1, T_2, T_3' range over subsets of A, and an inner sum where T_3'' ranges over subsets of $T_1 \cap T_2$. The second line holds because the query $\mathcal{F}(\bar{T}) = f_1(T_1) f_2(T_2) f_3(T_3') f_3(T_3'')$ is logically equivalent to $f_1(T_1) f_2(T_2) f_3(T_3')$ since $\forall a \in T_3''$ $f_3(a)$ is $f_1(a) f_2(a)$ and a is in both T_1 and T_2. The last line follows by breaking the sum into three disjoint sums:

1. $T_1 \cap T_2 = \emptyset$. Then, T_3'' is only allowed to be the empty set and the inner sum is 1. The total contribution of such terms is exactly equal to p_3^I.
2. $T_1 \cap T_2 \neq \emptyset, T_3' \neq \emptyset$. Then the inner sum, $\sum_{T_3''} (-1)^{|T_3''|}$ is 0, because T_3'' ranges over all subsets of $T_1 \cap T_2$.
3. $T_1 \cap T_2 \neq \emptyset, T_3' = \emptyset$. Then the inner sum is -1, because T_3'' ranges over all subsets of $T_1 \cap T_2$ except $\emptyset$. The total contribution is $p_1^I - p_1$.

Thus, we have shown Equation (7). Next, we introduce similar predicates between T_1, T_3, and T_2, T_3. This turns out to be much simpler: we write T_1 as $T_1' \cup T_1''$ where $T_1' = T_1 - T_3$ and $T_1'' = T_1 \cap T_3$. Similarly, we write T_2 as $T_2' \cup T_2''$ with $T_2' = T_2 - T_3$ and $T_2'' = T_2 \cap T_3$. The query $f_1(T_1) f_2(T_2) f_3(T_3)$ is logically equivalent to $f_1(T_1') f_2(T_2') f_3(T_3)$ since both f_1 and f_2 have a mapping to f_3. We now have independence predicates between T_1' and T_3 and T_2' and T_3. We replace $\bar{T}$ with $\bar{T}' = (T_1', T_2', T_3', T_1'', T_2'')$. Denoting $\mathtt{ip}(\bar{T}') = (T_1' \cap T_2' = T_1'' \cap T_2'' = T_1' \cap T_3' = T_2' \cap T_3' = \emptyset, T_1'' \subseteq T_3', T_2'' \subseteq T_3')$, we have:

$$\begin{aligned} p(q) &= \sum_{\mathtt{ip}(\bar{T}')} N(\mathcal{C}, sig(\bar{T}'))(-1)^{|\bar{T}|} p(\mathcal{F}(\bar{T}')) \\ &= \sum_{\mathtt{ip}(\bar{T}')} N(\mathcal{C}, sig(\bar{T}'))(-1)^{|\bar{T}|} \prod_{i=1,3} \prod_{a \in T_i'} p(f_i(a)) \quad (8) \end{aligned}$$

Note that the summation is over five sets $T_1', T_2', T_3', T_1'', T_2''$ but only T_1', T_2', T_3' are used in the compuation of p. The independence predicate $\mathtt{ip}$ allowed us to express $p(\mathcal{F}(\bar{T}))$ as a product. We will show later how to compute this sum. First, we need to show how to derive and use independence predicates in general. □

2.5 Independence Predicates

Our goal in this section is to define formally independence predicates. For unary coverages, an independence predicate is simply a statement $T_i \cap T_j \neq \emptyset$, but the non-unary case requires more formalism. We first introduce a new relational vocabulary, $\mathcal{T}$ consisting of the relation symbols $T_{f_1}, \cdots, T_{f_k}$ of arities $|x_{f_1}|, \ldots, |x_{f_k}|$ respectively. A

structure over this vocabulary is a k-tuple of sets $\bar{T}$; given a conjunctive query ϕ over the vocabulary $\mathcal{T}$, $\bar{T} \models \phi$ means that ϕ is true on $\bar{T}$. For a trivial illustration, assume T_{f_1}, T_{f_2} to be of arity 1, and $\phi = T_{f_1}(x), T_{f_2}(x)$. Then ϕ states that $T_{f_1} \cap T_{f_2} \neq \emptyset$.

Suppose we have have two factors f_i and f_j and θ is any 1-1 substitution on f_i, f_j, given in set representation, such that for all $(x_i, x_j) \in \theta$, x_i is an expansion variable of f_i and x_j is an expansion variable of f_j. Define

$$\begin{aligned}\theta^R(f_i, f_j) &= f_i, f_j, \bigwedge_{(x_i,x_j)\in\theta} x_i = x_j \\ \theta^T(f_i, f_j) &= T_{f_i}(\bar{x}_{f_i}), T_{f_j}(\bar{x}_{f_j}), \bigwedge_{(x_i,x_j)\in\theta} x_i = x_j\end{aligned}$$

Note that $\theta^R(f_i, f_j)$ is over the vocabulary $\mathcal{R}$ (same as the original query q), while $\theta^T(f_i, f_j)$ is over the vocabulary $\mathcal{T}$. We call them the *join query* and the *join predicate* respectively. We call the negation of join predicate, $\texttt{not}(\theta^T(f_i, f_j))$, an *independence predicate*.

Example 2.15 Consider factors f_1 and f_2 in Example 2.14, and let $\theta = \{(x_1, x_2)\}$. Then, $\theta^R(f_1, f_2) = P(x), R(x, y), S(x)$, $\theta^T(f_i, f_j) = T_1(x), T_2(x)$, and the independence predicate $\texttt{not}(\theta^T(f_i, f_j))$ says that T_1 and T_2 are disjoint.

The key property of independence predicates is the following: If T_i, T_j satisfy all independence predicates between f_i and f_j, then for all $\bar{a} \in T_i$ and $\bar{a}' \in T_j$, $f_i(\bar{a})$ and $f_j(\bar{a}')$ are independent.

2.6 Hierarchical Closure

Recall from Example 2.14 that, in order to introduce an independence predicate between two sets T_1, T_2 we needed to use the join query of their factors, $f_3(x) = f_1(x), f_2(x)$. In general, the join query between two factors in $\mathcal{F}$ is not necessarily in $\mathcal{F}$ (f_3 was redundant in Example 2.14). Thus, we will proceed as follows. Starting from a coverage $\mathcal{C}$ we will add join queries repeatedly until we obtain its *hierarchical closure*, denoted $\mathcal{C}^*$, then we will introduce independence predicates. Computing $\mathcal{C}^*$ is straightforward when $\mathcal{C}$ is an inversion-free coverage (which is the case for our first PTIME algorithm), but when $\mathcal{C}$ has inversions then some join queries are non-hierarchical and we cannot add them to $\mathcal{C}^*$. We define next $\mathcal{C}^*$ in the general case. Let $\mathcal{C} = (\mathcal{F}, C, \bar{x})$ be any coverage with a set of expansion variables $\bar{x}$.

Definition 2.16. Given two factors f_1 and f_2, with expansion variables $\bar{x}_{f_1}$ and $\bar{x}_{f_2}$, and a *MGU* given by the set representation θ, the *hierarchical unifer* θ_u is the maximal subset of θ such that:

1. $(x, y) \in \theta_u \Rightarrow x \in \bar{x}_{f_1}, y \in \bar{x}_{f_2}$
2. If $(x, y) \in \theta_u$ and (x', y') is such that $x \sqsubseteq x'$ or $y \sqsubseteq y'$ and $(x', y') \in \theta$, then $(x', y') \in \theta_u$.
3. The query $\theta_u^R(f_1, f_2)$ is hierarchical.

It can be shown that θ_u is uniquely determined. If θ_u is non-empty, we say that f_1 and f_2 can be *hierarchical joined* using θ and call the query $\theta_u^R(f_1, f_2)$ the *hierarchical join* of f_1 and f_2, and $\theta_u^T(f_1, f_2)$ the *hierarchical join predicate*.

Example 2.17 Let

$$\begin{aligned}f_1 = \quad & R(r, x), \quad S(r, x, y), U(a, r), U(r, z), V(r, z) \\ f_2 = \quad & \qquad S(r', x', y'), T(r', y'), V(a, r')\end{aligned}$$

and $\theta = \{(r, r'), (x, x'), (y, y')\}$ be the MGU of the two S sub-goals. Then, the hierarchical unifier is $\theta_u = \{(r, r')\}$. If we include any of (x, x') or (y, y'), we will have to include the other because $x \sqsubset y$ and $x' \sqsupset y'$, and then the join will not be hierarchical. The hierarchical join for this unifier is

$$\begin{aligned}\theta_u^R(f_1, f_2) = \quad & R(r, x), \quad S(r, x, y), U(a, r), U(r, z), V(r, z) \\ & \qquad S(r, x', y'), T(r, y'), V(a, r)\end{aligned}$$

and the set of expansion variables of the join is $\{r\}$. □.

Starting from the factors $\mathcal{F}$, we construct a set $\mathcal{H}$, a function *Factors* from $\mathcal{H}$ to subsets of $\mathcal{F}$, and a set of expansion variables $\bar{x}_h$ for $h \in \mathcal{H}$. This is done inductively as follows:

1. For each $f \in \mathcal{F}$, add f to $\mathcal{H}$ and let $\mathit{Factors}(f) = \{f\}$.
2. For any two queries h_1, h_2 in $\mathcal{H}$, and any MGU θ between h_1 and h_2, let $h = \theta_u^R(h_1, h_2)$ be their hierarchical join. Then add h to $\mathcal{H}$, define $\mathit{Factors}(h) = \mathit{Factors}(h_1) \cup \mathit{Factors}(h_2)$; define $\bar{x}_h = \theta_u(\bar{x}_{h_1} \cup \bar{x}_{h_2})$.

We need to show that $\mathcal{H}$ is finite. This follows from:

Lemma 2.18. *Given a fixed relational vocabulary $\mathcal{R}$ and a fixed set of constants C, the number of distinct hierarchical queries over $\mathcal{R}$ and C is finite.*

Define $\mathcal{F}^*$ to be the subset of $\mathcal{H}$ containing queries that are either inversion-free or in $\mathcal{F}$.

Definition 2.19. *(Hierarchical Closure) Given a coverage $\mathcal{C} = (\mathcal{F}, C, \bar{x})$, its* hierarchical closure *is $\mathcal{C}^* = (\mathcal{F}^*, C^*, \bar{x}^*)$ where $\mathcal{F}^*$, $\bar{x}^*$ are defined above and:*

$$C^* = \{c \mid c \subseteq \mathcal{F}^*, \bigcup_{f\in c} \mathit{Factors}(f) \in C\}$$

Note that $\mathcal{C}^*$ is indeed a coverage since the set $\mathcal{F}^*$ contains the set $\mathcal{F}$, the set C^* contains the set C, and the expansion variables satisfy the conditions in Def. 2.9. Let $\texttt{ip}(\mathcal{C}^*)$ be the conjunction of $\texttt{not}(jp)$, where jp ranges over all possible hierarchical join predicates in $\mathcal{F}^*$.

Lemma 2.20. *If $T \models \texttt{ip}(\mathcal{C}^*)$, then*

$$p(\mathcal{F}(q)) = \prod_{f\in\mathcal{F}^*} \prod_{a\in T_f} p(f(\bar{a}))$$

Finally, we look at conditions under which we can add the predicate $\texttt{ip}(\mathcal{C}^*)$ over $\bar{T}$. We divide the join predicates into two disjoint sets, *trivial* and *non-trivial*. A join predicate between factors h_i and h_j is called trivial if the join query is equivalent to either h_i or h_j, and is called non-trivial otherwise. We write $\texttt{ip}(\mathcal{C}^*)$ as $\texttt{ip}^n(\mathcal{C}^*) \wedge \texttt{ip}^t(\mathcal{C}^*)$, where $\texttt{ip}^n(\mathcal{C}^*)$ is the conjunction of $\texttt{not}(jp)$ over all non-trivial join predicates jp, and $\texttt{ip}^t(\mathcal{C}^*)$ is the conjunction over all trivial join predicates.

Definition 2.21. *(Eraser) Given a hierarchical join $jq = \theta_u^R(f_i, f_j)$, an* eraser *for jp is a set of factors $E \subseteq \mathcal{F}$ s.t.:*

1. $\forall q \in E$, *there is a homomorphism from* q *to* jq.

2. $\forall \sigma \subseteq \mathcal{F}$, $N(\mathcal{C}, \sigma \cup \{f_i, f_j\}) = N(\mathcal{C}, \sigma \cup \{f_i, f_j\} \cup E)$.

THEOREM 2.22. *Let q be a query such that every hierarchical join query $jq = \theta_u^R(f_i, f_j)$ between two factors in $\mathcal{F}^*$ has an eraser. Then,*

$$p(q) = \sum_{\bar{T} | T \models ip^n(\mathcal{C}^*)} N(\mathcal{C}^*, sig(\bar{T}))(-1)^{|\bar{T}|} p(\mathcal{F}(\bar{T}))$$

The theorem allows us to add all possible non-trivial independence predicates over the summation. If the hierarchical join query jp is inversion-free, then it belongs to $\mathcal{F}^*$ and it is its own eraser (i.e. $E = \{jp\}$ satisfies both conditions above). We can use it to separate T_i from T_j. In particular if q is inversion-free, then any hierarchical join query has an eraser, and all sets can be separated. But if jp has an inversion, then jp does not belong to $\mathcal{F}^*$ and we must find some different query (queries) in $\mathcal{F}^*$ that can be used to separate T_i from T_j.

Example 2.23 Let's revisit the query in Example 2.14. We had $q = P(x), R(x,y), R(x',y'), S(x')$. Suppose we start from the trivial coverage $\mathcal{C}_0 = \{q\}$, with two factors $\mathcal{F}_0 = \{f_1, f_2\}$ (see notations in Example 2.14), and suppose we chose a single expansion variable for f_1 and f_2, namely x_1, x_2 respectively. Its hierarchical closure adds the join query f_3 between f_1 and f_2. The coverage $\mathcal{C}_0^*$ contains the following covers: $\{f_1, f_2\}$, $\{f_3\}$, $\{f_1, f_3\}$, $\{f_2, f_3\}$ and $\{f_1, f_2, f_3\}$.

Thus, we have expressed the probability of a query $p(q)$ using the sum in Theorem 2.22. This is still exponential in size, and now we will show how compute a closed form for that sum. Here we will use different techniques for the two PTIME algorithm. In the first algorithm (for inversion-free queries) the coverage is unary, and all independence predicates are of the form $T_i \cap T_j \neq 0$: here we derive closed forms directly. In the second algorithm (for queries with erasable inversions) the independence predicates are more complex: in this case we will reduce the sum to the probability of an inversion-free query ϕ over the $\mathcal{T}$ vocabulary, thus bootstrapping the first PTIME algorithm.

3. PTIME ALGORITHMS

In this section, we establish one-half of the dichotomy by proving Theorem 1.8(2). We start by computing simple sums over functions on sets, then use it to give a PTIME algorithm for queries without inversion and finally give the general PTIME algorithm for queries that have erasers for all inversions.

3.1 Simple Sums

Let $A = \{1, \ldots, N\}$, $\bar{g} = (g_1, \ldots, g_k)$ be k functions $g_i : A \to \mathbf{R}$, $i = 1, \ldots, k$, and $\bar{T} = (T_1, \ldots, T_k)$ a k-tuple of subsets of A. Denote $\bar{g}(\bar{T}) = g_1(T_1) \cdots g_k(T_k)$, where $g_i(T_i) = \prod_{\bar{a} \in T_i} g_i(\bar{a})$. $\emptyset \neq \bar{T}$ abbreviates $\emptyset \neq T_1, \ldots, \emptyset \neq T_k$. Let ϕ be a conjunction of statements of the form $T_i \cap T_j = \emptyset$ or $T_i \subseteq T_j$, and define: $S_\phi = \{\sigma \mid \sigma \subseteq [k], \forall i, j \in \sigma, \phi \not\models T_i \cap T_j = \emptyset\} \cap \{\sigma \mid \sigma \subseteq [k], \forall i \in \sigma, j \notin \sigma, \phi \not\models T_i \subseteq T_j\}$.

DEFINITION 3.1. *Denote the following sums:*

$$\bigoplus\nolimits_\phi \bar{g} = \sum_{\bar{T} \subseteq A, \phi} \bar{g}(\bar{T})$$

$$\bigoplus\nolimits_\phi^+ \bar{g} = \sum_{\emptyset \neq \bar{T} \subseteq A, \phi} \bar{g}(\bar{T})$$

For $\sigma \subseteq [k]$, denote $\bar{g}_\sigma$ the family of functions $(g_i)_{i \in \sigma}$.

PROPOSITION 3.2. *The following closed forms hold:*

$$\bigoplus\nolimits_\phi \bar{g} = \prod_{a \in A} \sum_{\sigma \in S_\phi} \prod_{i \in \sigma} g_i(\bar{a})$$

$$\bigoplus\nolimits_\phi^+ \bar{g} = \sum_{\sigma \subseteq [k]} (-1)^{k - |\sigma|} \bigoplus \bar{g}_\sigma$$

Moreover, the expressions above have sizes $O(k2^k N)$ and $O(k2^{2k} N)$ respectively, hence all have an expression size that is linear in N.

Example 3.3 Consider four functions $g_i : A \to R$, $i = 1, 2, 3, 4$, and suppose we want to compute the following sum:

$$\sum_{T_1 \cap T_2 = \emptyset, T_2 \cap T_3 = \emptyset, T_4 \subseteq T_2} g_1(T_1) g_2(T_2) g_3(T_3) g_4(T_4)$$

In our notation, this is $\bigoplus_\phi \bar{g}$, where ϕ is $T_1 \cap T_2 = \emptyset \wedge T_2 \cap T_3 = \emptyset \wedge T_4 \subseteq T_2$. The set S_ϕ is $\{\emptyset, \{1\}, \{2\}, \{2,4\}, \{3\}, \{1,3\}\}$. Thus, the expression for the sum is

$$\prod_{a \in A} (1 + g_1(a) + g_2(a) + g_2(a) g_4(a) + g_3(a) + g_1(a) g_3(a))$$

The size of this expression is $8N$, where N is the size of A.

3.2 PTIME for Inversion-Free Queries

Let q be an inversion-free query. We give now a PTIME algorithm for computing q on a probabilistic structure.

THEOREM 3.4. *If q has no inversions then q has a unary coverage.*

This says that we can choose for each factor f a single *root* variable r_f s.t. any MGU between two (not necessarily distinct) factors f, f' maps r_f to $r_{f'}$: the proof in the Appendix uses the canonical coverage $\mathcal{C}^<$, considers for each factor f all maximal variables under $\sqsupseteq$, and chooses as root variable the maximum variable under $>$. Note that for queries with inversions Theorem 3.4 fails (recall the queries H_k).

Example 3.5 We illustrate Theorem 3.4 on two queries.

$$q_1 = R(x,y), S(x,y), S(x',y'), T(y')$$
$$q_2 = R(x,y), R(y,x)$$

In the trivial coverage $\mathcal{C} = \{q_1\}$ for q_1 the factors are

$$f_1 = R(x,y), S(x,y) \qquad f_2 = S(x',y'), T(y')$$

We see that $r_{f_1} = \{y\}$ and $r_{f_2} = \{y'\}$ satisfy the properties of Theorem 3.4 (there are two maximal variables for f_1, but we have to pick y because it unifies with y'). For q_2, the trivial coverage $\mathcal{C} = \{q_3\}$ does not work since there is a unifier that unifiers x with y, and exactly one of them can be the expansion variable. On the other hand, consider the following coverage:

$$f_1 = R(x_1, y_1), R(y_1, x_1), x_1 > y_1 \qquad f_2 = R(x,x)$$

now we can set $r_{f_1} = x_1$ and $r_{f_2} = x$. □

Now, let q be a query without inversion and $\mathcal{C} = (\mathcal{F}, C, \bar{x})$ be any unary coverage. Let $\mathcal{C}^* = (\mathcal{F}^*, C^*, \bar{x}^*)$ be the hierarchical closure of $\mathcal{C}$. Theorem 2.22 applied to this unary coverage gives:

$$p(q) = \sum_{\bar{T} | T \models \mathtt{ip}^n(\mathcal{C}^*)} N(\mathcal{C}^*, sig(\bar{T}))(-1)^{|\bar{T}|} p(\mathcal{F}^*(\bar{T})) p(f(\bar{a}))$$

All the sets in T have arity 1, since $\mathcal{C}^*$ is also unary, hence each join predicate has the form $T_i(x), T_j(x)$ which is equivalent to $T_i \cap T_j \neq \emptyset$, hence $\mathtt{ip}(\mathcal{C}^*)$ is a conjunction of predicates of the form $T_i \cap T_j = \emptyset$.

So far we have only added the independence predicates $\mathtt{ip}^n(\mathcal{C}^*)$, i.e. independence predicates between those pairs h_i and h_j for which the join query is not equivalent to either h_i and h_j. Next, we add independence predicates between the remaining pairs. We generalize our technique of Example 2.14. We replace $\bar{T}$ with $\bar{T}'$, where $\bar{T}'$ contains all the sets in $\bar{T}$ along with some additional sets. For each h_i, h_j such that their hierarchical join is equivalent to h_j, $\bar{T}'$ contains an additional set $T_{i,j}$. Denote $\mathtt{ip}^l(\mathcal{C}^*)$ the conjunction of the following predicates

- A predicate $T_{i,j} \subseteq T_j$ for all $T_{i,j}$, T_j in $\bar{T}'$
- A predicate $T_{i_1,j} \cap T_{i_2,j} = \emptyset$ for all $T_{i_1,j}, T_{i_2,j}$ in $\bar{T}'$ such that there is a predicate $T_{i_1} \cap T_{i_2} = \emptyset$ in $\mathtt{ip}(\mathcal{C}^*)$.

Let $\mathtt{ip}'(\mathcal{C}^*)$ denote the conjunction of $\mathtt{ip}(\mathcal{C}^*)$ and $\mathtt{ip}^l(\mathcal{C}^*)$. Then, we obtain $p(q) =$

$$\sum_{\bar{T} | T \models \mathtt{ip}'(\mathcal{C}^*) \wedge \mathtt{ip}^l(\mathcal{C}^*)} N(\mathcal{C}^*, sig(\bar{T}))(-1)^{|\bar{T}|} \prod_{f \in \mathcal{F}^*} \prod_{\bar{a} \in T_f} p(f(\bar{a}))$$

Corresponding to each $T_i \in \bar{T}'$, let $g_i : A \to \mathbf{R}$ denote the function $g_i(\bar{a}) = -p(f_i(\bar{a}))$. Also, corresponding to each $T_{i,j} \in \bar{T}'$, let $g_{i,j}$ denote the function $g_{i,j}(\bar{a}) = 1$.

THEOREM 3.6. *Let q be inversion-free.*

1. *The probability of q is given by*

$$p(q) = \sum_{\sigma \subseteq \mathcal{F}^*} N(\mathcal{C}^*, \sigma) \bigoplus\nolimits^{+}_{i\mathtt{p}(\mathcal{C}^*) \wedge i\mathtt{p}^l(\mathcal{C}^*)} \bar{g}_\sigma \quad (9)$$

   where $\bigoplus^+$ ranges over all sets of the form $\bar{T}'$.

2. *For each $f \in \mathcal{F}^*$, $f(\bar{a})$ is an inversion-free query.*

We use Proposition 3.2 to write a closed-form expression for Equation (9) in terms of the probabilities $g_f(\bar{a}) = p(f(\bar{a}))$ for $f \in \mathcal{F}^*$. Since each of these queries is inversion-free, we recursively apply Equation (9) to compute their probabilities. For any query q, let $V(q)$ denote the maximum number of distinct variables in any single sub-goal of q. Clearly, for any factor f, $V(f(\bar{a})) < V(f) \leq V(q)$ (since $\bar{a}$ substitutes a variable in every sub-goal). Thus, the depth of the recursion is bounded by $V(q)$.

COROLLARY 3.7. *If q is an inversion-free query, then $p(q)$ can be expressed as a formula of size $O(N^{V(q)})$, where N is the size of the domain. In particular q is in* PTIME.

Example 3.8 Continuing our running example from Example 2.14, recall that $p(q)$ is given by Equation (8). Let $\bar{T}' = (T_1, T_2, T_3, T_{1,3}, T_{2,3})$. Denoting $g_i(a) = -p(f_i(a))$, for $i = 1, 2, 3$, $\phi \equiv (T_1 \cap T_2 = \emptyset)$ and $\psi \equiv (T_1 \cap T_2 = \emptyset) \wedge (T_1 \cap T_3 = \emptyset) \wedge (T_2 \cap T_3 = \emptyset) \wedge (T_{1,3} \cap T_{2,3} = \emptyset) \wedge (T_{1,3} \subseteq T_3) \wedge (T_{2,3} \subseteq T_3)$:

$$p(q) = \bigoplus\nolimits^{+}_{\phi}(g_1, g_2) + \bigoplus\nolimits^{+}(g_3) + \bigoplus\nolimits^{+}_{\psi}(g_1, g_2, g_3)$$

Now apply Prop. 3.2 to each expression, e.g. $\bigoplus^+_\psi(g_1, g_2, g_3) = \bigoplus_\psi(g_1, g_2, g_3) - \bigoplus_\psi(g_1, g_2) - \ldots$ Each sum in turn has a closed form. Furthermore, each $f_i(a)$ is a query with a single variable (y or y'), hence each $g_i(a) = p(f_i(a))$ can be computed inductively.

Appendix A gives example of inversion-free queries, showing several subtleties that were left out from the text.

Queries with Negated Subgoals The PTIME algorithm in this section can be extended to queries with negated sub-goals.

DEFINITION 3.9. *A* conjunctive query with negations *is a query $q = \exists \bar{x}.(\varphi_1 \wedge \ldots \wedge \varphi_k)$, where each φ_i is either a positive sub-goal $R(t)$, or a negative sub-goal $\mathtt{not}(R(t))$, or an arithmetic predicate. The query q is said to be inversion-free if the conjunctive query obtained by replacing each $\mathtt{not}(R(t))$ sub-goal with $R(t)$ sub-goal is inversion-free.*

DEFINITION 3.10. *(Inversion-free property) A property ϕ is called* inversion-free property *if it can be expressed as a Boolean combination of queries $\{q_1, \cdots, q_m\}$ such that each q_i is a conjunctive query with negation and the query $q_1 q_2 \cdots q_m$ is inversion-free.*

THEOREM 3.11. *Let ϕ be any inversion-free property. Then, computing $p(\phi)$ is in* PTIME.

PROOF. (Sketch) Consider a single inversion-free conjunctive query with negation. The same recurrence formula in Theorem 3.6 applies, the only difference is during recursion we will reach negated constant sub goals: $p(\mathtt{not}(R(a, b, c)))$ is simply $1 - p(R(a, b, c))$. For any general ϕ, use inclusion/exclusion formula to reduce it to conjunctive queries with negations, each of which is inversion-free. □

3.3 Complex Sums

In Section 3.2, we used simple sums to give a PTIME algorithm for inversion-free queries. Here, we show that the PTIME algorithm can be used to compute closed formulas for complex sums. We call this the *bootstrapping technique*.

Bootstrapping: Let $\bar{g} = (g_1, \ldots, g_k)$ be a family of functions, $g_i : A^{r_i} \to \mathbf{R}$, where the arity of g_i is r_i. We want to compute sums of the form $\mathtt{sum} = \sum_{\bar{S} | \phi} \bar{g}(\bar{S})$, where ϕ is a complex predicate. We cannot use the summations of Section 3.1, which only apply when g_i are unary. Instead, we use a bootstrapping technique to reduce this problem back to evaluating an inversion-free query on a probabilistic database, and use the PTIME algorithm of Section 3.2. The basic principle is that we can reduce the problem to the evaluation of ϕ over a probabilistic database. Create an probabilistic instance of $\mathcal{S}$, where, assuming $k = 1$ for simplicity, for each tuple $\bar{a} \in S$, set its probability to $p(\bar{a}) = g(\bar{a})/(1 + g(\bar{a}))$. Then, the probability of ϕ over this instance is $p(\phi) = \sum_S \prod_{\bar{a} \in S} p(\bar{a}) \prod_{\bar{a} \notin S} (1 - p(\bar{a})) = \prod_{\bar{a}} 1/(1 + g(\bar{a})) \sum_{S | \phi} g(S) = \prod_{\bar{a}} 1/(1 + g(\bar{a}))\mathtt{sum}$. Thus, we can compute $\mathtt{sum}$ in PTIME if we can evaluate the query ϕ in PTIME.

THEOREM 3.12. *Let ϕ be an inversion-free property. Then $\sum_{\bar{S} | \phi} \bar{g}(\bar{S})$ has a closed form polynomial in domain size.*

3.4 The General PTIME Algorithm

Let q be a conjunctive query and let $\mathcal{C} = (\mathcal{F}, C)$ be a strict coverage for q and let $\mathcal{H}$ be the set of hierarchical unifiers, as defined in Section 2.6. Suppose the following holds: for every hierarchical join predicate $jp = \theta^T(h_i, h_j)$ between two factors in $\mathcal{H}$, the join query $jq = \theta^R(f_i, f_j)$ has an eraser. We will show here that q is in PTIME, thus proving Theorem 1.8(2).

We set the expansion variables $\bar{x}$ to include all variables, i.e. $\bar{x}_f = Vars(f)$ for all $f \in \mathcal{F}$. Let $\mathcal{C}^* = (\mathcal{F}^*, C^*, \bar{x}^*)$ be the hierarchical closure of $\mathcal{C}$. By Theorem 2.22, we have $p(q) = Exp(\mathcal{C}^*)$, where

$$Exp(\mathcal{C}^*) = \sum_{\bar{T} | \mathtt{ip}^n(\mathcal{C}^*)} N(\mathcal{C}^*, sig(\bar{T}))(-1)^{|\bar{T}|} p(\mathcal{F}^*(\bar{T}))$$
$$= \sum_{\sigma} N(\mathcal{C}^*, \sigma) \sum_{\bar{T} | \mathtt{ip}^n(\mathcal{C}^*), sig(\bar{T}) = \sigma} (-1)^{|\bar{T}|} p(\mathcal{F}^*(\bar{T})) \quad (10)$$

Before we proceed, we illustrate with an example:

Example 3.13 Consider the query q in Example 1.7 Although q has an inversion (between the two S Subgoals) we have argued in Sec. 1.1 that it is in PTIME. Importantly, the third line of constants sub goals plays a critical role: if we removed it, the query becomes #P-hard.
Consider the coverage $\mathcal{C} = (\mathcal{F}, C, \bar{x})$, where $\mathcal{F}$ is[3]:

$$\begin{aligned} f_1 &= R(r,x), S(r,x,y), U(a,r), U(r,z), V(r,z), r \neq a \\ f_2 &= S(r',x',y'), T(r',y'), V(a,r'), r' \neq a \\ f_3 &= U(a,z'), V(a,z') \\ f_4 &= R(a), S(a,b,c), U(a,a) \end{aligned}$$

and $C = \{\{f_1, f_2, f_4\}, \{f_2, f_3, f_4\}\}$. We cannot simply take the root variables r, r', and z' as expansion variables and proceed with the recurrence formula in Th. 3.6, because the query $f_{12} = f_1(r)f_2(r)$ is #P-hard. We must keep all variables as expansion variables to avoid the inversion. Thus, the root unifiers $\mathcal{H}$ are (recall Example 2.17):

$$\begin{aligned} f_{12} &= f_1, f_2, r = r' \\ f_{23} &= f_2, f_3, r' = z' \\ f_{13} &= f_1, f_3, r = z' \\ f_{123} &= f_1, f_2, f_3, r = r' = z' \end{aligned}$$

Out of these, f_{12} and f_{123} have inversions, thus $\mathcal{F}^*(q) = \{f_1, f_2, f_3, f_4, f_{23}, f_{13}\}$. In the expansion $Exp(\mathcal{C}^*)$, there are sets $T_1, T_2, T_3, T_4, T_{23}, T_{13}$ but note that they are not unary, e.g. T_1 has arity 4 as $\bar{x}_{f_1} = \{r, x, y, z\}$. The critical question is how to separate now T_1 from T_2, since we don't have the factor f_{12}. Here we use the fact that there exists a homomorphism $f_3 \rightarrow f_{12}$, thus f_3 is an eraser between f_1 and f_2 and will use f_3 to separate T_1, T_2. The definition of an eraser (Def. 2.21) requires us to check $\forall \sigma$, $N(\mathcal{C}, \sigma \cup \{f_1, f_2\}) = N(\mathcal{C}, \sigma \cup \{f_1, f_2, f_3\})$. The only σ that makes both N's non-zero is $\{f_4\}$ (and supersets), and indeed the two numbers are equal to $+1$. It is interesting to note that, if we delete the last line from q, then we have the same set of factors but a new coverage $C' = \{\{f_1, f_2\}, \{f_2, f_3, f_4\}\}$: then f_3 is no longer an eraser because for $\sigma = \emptyset$ we have $N(\{f_1, f_2\}) = 1$ and $N(\{f_1, f_2, f_3\}) = 0$. Continuing the example, we conclude that, with aid from the eraser, we can now insert all independence predicates. We have to keep in mind, however, that these predicates are no longer simple disjointness conditions e.g. the predicate between T_1 and T_2 is the negation of the query $T_1(r,x,y,z), T_2(r,x',y')$. □

[3]Strictly speaking each constant sub-goal $R(a)$, $S(a,b,c)$, $U(a,a)$ should be a distinct factor.

We now focus on each of the inner sums in Equation (10). We want to reduce it to evaluation of an inversion-free property, but there are two problems. First, the predicate $\mathtt{ip}^n(\mathcal{C}^*)$ over $\bar{T}$ is not an inversion-free property. Second, we still need to add the predicates $\mathtt{ip}^t(\mathcal{C}^*)$ to make $p(\mathcal{F}^*(\bar{T}))$ multiplicative. To solve these problems, we apply a preprocessing step on Equation 10, which we call the *change of basis*. In this step, we group $\bar{T}$ that generate the same $\mathcal{F}^*(\bar{T})$ and sum over these groups.

Example 3.14 Consider a factor $f = R_1(x,y), R_2(y,z)$. We look at the set $T(x,y,z)$ corresponding to this factor, which is a ternary set since $\bar{x}_f = \{x,y,z\}$. For every T, define $S^0 = \pi_y(T)$, $S^1 = \pi_{xy}(T)$ and $S^2 = \pi_y(T)$, hence $T = S^0 \bowtie S^1 \bowtie S^2$ (natural join). Clearly, S^0, S^1, S^2 satisfy the predicate $S^0 = \pi_y(S^1) = \pi_y(S^2)$. Consider the sum

$$\sum_{\bar{T}} (-1)^{|\bar{T}|} p(f(\bar{T})) \quad (11)$$

We group all T that generate the same S^0, S^1, S^2 and show that the summation in Eq. 11 is equivalent to the following:

$$\sum_{\substack{S^1, S^2, S^0 \mid \\ S^0 = \pi_y(S^1) = \pi_y(S^2)}} (-1)^{|S^1|+|S^2|+|S^0|} p(R_1(S^1) R_2(S^2))$$

Thus, we have changed the basis of summation from T to S^0, S^1, S^2. □

The change of basis introduces some new predicates between sets, which we call the *link predicates*, e.g. predicates of the form $S^0 = \pi_y(S^1)$. But at the same time, as we shall see, the change of basis simplifies the independence predicates $\mathtt{ip}(\mathcal{C}^*)$, making them inversion-free, so that the computation of Equation (10) can be reduced to evaluation of inversion-free queries. We now formally define the change of basis. This consists of the following steps: (1) we change the summation basis from $\bar{T}$ to $\bar{S}$. (2) we translate the $\mathtt{ip}^n(\mathcal{C}^*)$ predicates from $\bar{T}$ to $\bar{S}$. (3) we introduce a new set of predicates, called the link predicates, on $\bar{S}$. (4) We add the remaining independence predicates, $\mathtt{ip}^t(\mathcal{C}^*)$, translated from $\bar{T}$ to $\bar{S}$, to $\bar{S}$.

Consider a factor $f \in \mathcal{F}^*$. It is a connected hierarchical query with the hierarchy relation $\sqsubseteq$ on $Vars(f)$. Given $x \in Vars(f)$, let $[x]$ denotes its equivalence class under $\sqsubseteq$ and let $\lceil x \rceil$ denote $\{y \mid y \sqsupseteq x\}$. Define a *hierarchy tree* for f as the tree where nodes are equivalence classes of variables, and edges are such that their transitive closure is $\sqsubseteq$. For instance, in Example 3.14, the hierarchy tree of f has nodes $\{x\}, \{y\}, \{z\}$ with $\{x\}$ as root and $\{y\}, \{z\}$ its children.

Define a new vocabulary, consisting of a relation $S_f^{[x]}$ for each $f \in \mathcal{F}^*$ and each node $[x]$ in the hierarchy tree of f, with arity equal to the size of $\lceil x \rceil$. Let $\bar{S}$ denote instances of this vocabulary. The intuition is that $S_f^{[x]}$ denotes $\pi_{\lceil x \rceil}(T_f)$ in the change of basis from $\bar{T}$ to $\bar{S}$. This completes step 1.

Let $\mathtt{ip}^n$ denote the set of independence predicates on $\bar{S}$, translated in a straightforward manner from the independence predicates $\mathtt{ip}^n(\mathcal{C}^*)$ on $\bar{T}$ (details in appendix). This is step 2.

Define a *link predicate* $S_f^{[x]} = \pi_{\lceil x \rceil}(S_f^{[y]})$ for every edge $([x],[y])$ in the hierarchy tree of f. Let $\mathtt{lp}$ be the set of all link predicates. This is step 3.

Finally, we add the trivial independence predicates $\mathtt{ip}^t$. For this, we expand the basis of summation from $\bar{S}$ to $\bar{S}'$ by adding the following sets. We add a new set $S^{i,j}_{x_i,x_j}$ corresponding to each pair $S_{f_i}^{[x_i]}$, $S_{f_j}^{[x_j]}$ such that (i) f_i and f_j have a hierarchical join query which is equivalent to f_j and (ii) there are sub-goals g_i in h_i and g_j in h_j referring to the same relation such that $Vars(g_i) = \lceil x_i \rceil$ and $Vars(g_j) = \lceil x_j \rceil$. For each such $S^{i,j}_{x_i,x_j}$, $\mathtt{ip}^t$ contains the following conjuncts: $S_{f_i}^{[x_i]} \cap S_{f_j}^{[x_j]} = \emptyset$, $S^{i,j}_{x_i,x_j} \subseteq S_{f_j}^{[x_j]}$. This describes the step 4.

Finally, we put it all together. We define a function $G(\bar{S}')$ on $\bar{S}'$ as follows. Consider a relation $S_f^{[x]}$, and let p be the number of children of $[x]$ in the hierarchy tree. For a tuple t in $S_f^{[x]}$, let

$$G(t) = (-1)^{p+1} \prod_{g \in sg(f) | Vars(g) = \lceil x \rceil} p(g(t))$$

Define $G(\bar{S}') = \prod_{t \in \bar{S}'} G(t)$.

Denote $sig(\bar{S}')$ the set $\{f \mid S_f^{[r_f]} \neq \emptyset\}$, where $[r_f]$ denotes the root of the hierarchy tree of f.

THEOREM 3.15. *With $\mathtt{ip}^t$, $\mathtt{ip}^n$, $\mathtt{lp}$, sig and G as defined above,*

$$\sum_{\bar{T} | \mathtt{ip}(\mathcal{C}^*), sig(\bar{T}) = \sigma} (-1)^{|\bar{T}|} p(\mathcal{F}^*(\bar{T})) = \sum_{\bar{S}' | \mathtt{ip}^n, \mathtt{ip}^t, \mathtt{lp}, sig(\bar{S}') = \sigma} G(\bar{S}')$$

Finally, we use the bootstrapping principle to reduce the problem of computing the summation to the evaluation of the query $\phi = (\mathtt{ip}^n \wedge \mathtt{ip}^t \wedge \mathtt{lp} \wedge sig(\bar{S}') = \sigma)$.

LEMMA 3.16. *The query ϕ defined above is an inversion-free property.*

By using Theorem 3.12, we get the following:

THEOREM 3.17. *Suppose for every hierarchical join predicate $jp = \theta^T(h_i, h_j)$ between two factors in $\mathcal{H}$, the join query $jq = \theta^R(f_i, f_j)$ has an eraser. Then, q is* PTIME.

4. #P-HARD QUERIES

Here we show the other half of Theorem 1.8, i.e., if q has an inversion without an eraser, then q is #P-hard.

Let $\mathcal{C} = (\mathcal{F}, C, \bar{x})$ be any strict coverage for q, $\mathcal{C}^* = (\mathcal{F}^*, C^*, \bar{x}^*)$ its closure and $\mathcal{H}$ the set of hierarchical join queries over $\mathcal{F}$.

Suppose there are factors $h, h' \in \mathcal{H}$ such that the join query $hj = \theta^T(h, h')$ has an inversion, but not an eraser. Among all such hj, we will pick a specific one and use it to show that q is #P-hard. Note that if there is no such hj, then the query is in PTIME by Theorem 3.17.

Let the inversion in hj consist of a unification path of length k from (f, x, y) with $x \sqsubset y$ to (f', x', y') with $x' \sqsupset y'$. Then, we will prove the #P-hardness of q using a reduction from the chain query H_k, which is #P-hard by Theorem 1.5.

Given an instance of H_k, we create an instance of q. The basic idea is as follows: take the unification path in hj that has the inversion and completely unify it. We get a non-hierarchical query (due to the inversion) with two distinguished variables x and y (the inversion variables), $k+2$ distinguished sub-goals (that participated in the inversion), plus other sub-goals in the factor. Use the structure of this query and the contents of the $k+2$ relations in the instance of H_k to create an instance for q. We skip the formal description of the reduction, but instead illustrate it on examples.

Example 4.1 Consider $q = U(x), V(x,y), V(y,x)$ and the coverage $\mathcal{C} = (\mathcal{F}, C)$ where $\mathcal{F} = \{f\}$ with $f = U(x), V(x,y)$, $V(y,x), x \neq y$ and $C = \{\{f\}\}$. The coverage has a single factor and a single cover. The first V sub-goal of factor f unifies with the second sub-goal of another copy of f to give an inversion between $x \sqsupset y$ and their copy $y' \sqsubset x'$. If we unify the two sub-goals in two copies of f, we get the query:

$$q_u = \underline{U}(x), \underline{V}(x,y), V(y,x), \underline{U}(y)$$

We have underlined the sub-goals taking part in the inversion. Now we give a reduction from the query $H_0 = R(x), S(x,y), S(x',y'), T(y')$. Given any instance of R, S, T for H_0 construct an instance of U, V as follows. We map the R, S, T relations in H_0 to the U, V, U underlined sub goals of q_u as follows: for each tuple $R(a)$, create a tuple $U(a)$ with same probability. For each $S(a,b)$, create $V(a,b)$ with the same probability. For each $T(a)$, create $U(a)$ with same probability. Also, for each $S(a,b)$, create $V(b,a)$ with probability 1 (this corresponds to the non-underlined sub-goal).

There is a natural 1-1 correspondence between the substructures of U, V and the substructures of R, S, T with the same probability. It can be shown that q is true on a substructure iff the query $R(x), S(x,y) \vee S(x',y'), T(y')$ is true on the corresponding substructure. Thus, we can compute the probability of the query $R(x), S(x,y) \vee S(x',y'), T(y')$, and hence, the probability of H_0, by applying inclusion-exclusion.

Next, we show why a hardness reduction fails if the inversion has an eraser.

Example 4.2 We revisit the query q in Example 3.13. There is an inversion between $x \sqsubset y$ in f_1 and $x' \sqsupset y'$ in f_2. However, their hierarchical join, f_{12} have an eraser. The unified query consists of $q_u =$
$\underline{R}(r,x), \underline{S}(r,x,y), U(a,r), U(r,z), V(r,z), V(a,r), \underline{T}(r,x)$
$R(a), S(a,b,c), U(a,a)$

We construct an instance $RSTUV$ for q from an instance $R'S'T'$ for H_0 as in previous example. However, there is a bad mapping from q to q_u, corresponding to the eraser, which is $\{r \to a, x \to b, y \to c, x' \to x, y' \to y, z \to r\}$, which avoids the $\underline{R}$ sub-goal. The effect is that q is true on a world iff the query $S'(x',y')T'(y')$ (rather that H_0) is true on the corresponding world. So the reduction from H_0 fails. In fact, we know that this query q is in PTIME.

The final example shows that if there are multiple inversions without erasers, we need to pick one carefully, which makes the hardness reduction challenging.

Example 4.3 Consider the following variation of the query in previous example:

$$\begin{aligned} q = & R(x), S(x,y), U(x,y,a,b), U(z_1,z_2,x,y), V(z_1,z_2,x,y) \\ & S(x',y'), T(y'), V(x',y',a,b) \\ & R(a), S(a,b), U(a,b,a,b) \end{aligned}$$

Let f_1 and f_2 denote the factors corresponding to the first two lines of q. There is an inversion from $x \sqsupset y$ in f_1 to $x' \sqsubset y'$ in f_2 via the two S sub-goals, and it does not have an eraser. But if we unify the two S sub-goals to obtain S, there is a "bad mapping" from q to q_u that maps x, y to a, b and z_1, z_2 to x, y. However, as it turns out, there is another inversion in q that we can use for hardness. The inversion is from $x \sqsupset y$ to $z_1 \equiv z_2$ to x', y' through the following unification path: $U(\underline{x}, \underline{y}, x, y)$ unifies with (a copy of) $U(\underline{z}_1, \underline{z}_2, x, y)$ and $V(\underline{z}_1, \underline{z}_2, x, y)$ unifies with $V(\underline{x}', \underline{y}', a, b)$. We can show that this inversion works for the hardness reduction.

By formalizing these ideas, we prove:

THEOREM 4.4. *Suppose there are $h, h' \in \mathcal{H}^*(q)$ such that their hierarchical join hj has an inversion without an eraser. Then, q is #P-complete.*

5. CONCLUSIONS

We show that every conjunctive query has either PTIME or #P-complete complexity on a probabilistic structure. As part of the analysis required to establish this result we have introduced new notions such as hierarchical queries, inversions, and erasers. Future work may include several research directions: a study whether the hardness results can be sharpened to counting the number of substructures (i.e. when all probabilities are 1/2); an analysis of the query complexity; extensions to richer probabilistic models (e.g. to probabilistic databases with disjoint and independent tuples [9]); and, finally, studies for making our PTIME algorithm practical for probabilistic database systems.

6. REFERENCES

[1] Daniel Barbará, Hector Garcia-Molina, and Daryl Porter. The management of probabilistic data. *IEEE Trans. Knowl. Data Eng.*, 4(5):487–502, 1992.

[2] Jihad Boulos, Nilesh Dalvi, Bhushan Mandhani, Shobhit Mathur, Chris Re, and Dan Suciu. Mystiq: a system for finding more answers by using probabilities. In *SIGMOD*, pages 891–893, 2005.

[3] Nadia Creignou and Miki Hermann. Complexity of generalized satisfiability counting problems. *Inf. Comput.*, 125(1):1–12, 1996.

[4] Nilesh Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. In *VLDB*, 2004.

[5] Tomas Feder and Moshe Y. Vardi. Monotone monadic snp and constraint satisfaction. In *STOC*, pages 612–622, 1993.

[6] Norbert Fuhr and Thomas Rolleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Trans. Inf. Syst.*, 15(1):32–66, 1997.

[7] Erich Gradel, Yuri Gurevich, and Colin Hirch. The complexity of query reliability. In *PODS*, pages 227–234, 1998.

[8] Laks V. S. Lakshmanan, Nicola Leone, Robert Ross, and V. S. Subrahmanian. Probview: a flexible probabilistic database system. *ACM Trans. Database Syst.*, 22(3):419–469, 1997.

[9] Christopher Re, Nilesh Dalvi, and Dan Suciu. Query evaluation on probabilistic databases. *IEEE Data Engineering Bulletin*, 29(1):25–31, 2006.

[10] Thomas J. Schaefer. The complexity of satisfiability problems. In *STOC*, pages 216–226, 1978.

[11] L. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8:410–421, 1979.

[12] Jennifer Widom. Trio: A system for integrated management of data, accuracy, and lineage. In *CIDR*, 2005.

APPENDIX

A. EXAMPLES OF INVERSIONS

We illustrate in Fig. 1 several subtleties of inversion-free queries that were left out from the text. Fig. 2 illustrates some queries with inversions; all are #P-hard.

Query. The trivial coverage is non-strict and has an "inversion"	Fragment of a strict coverage (Unification chain underlined)	Comments
$R(x)S_1(\underline{x}, y, \underline{y})$ $S_1(\underline{u}, v, \underline{w}), S_2(\underline{u}, v, \underline{w})$ $S_2(\underline{x'}, x', \underline{y'}), T(y')$	$qc_1 = R(x), S_1(\underline{x}, y, \underline{y}), x \neq y,$ $S_1(\underline{u}, v, \underline{v}), S_2(\underline{u}, v, \underline{v}), u \neq v$ $S_2(x', x', y'), T(y'), x' \neq y'$ $qc_2 = R(x), S_1(x, y, y), x \neq y$ $S_1(\underline{u}, u, \underline{w}), S_2(\underline{u}, u, \underline{w}), u \neq w$ $S_2(\underline{x'}, x', \underline{y'}), T(y'), x' \neq y'$	Illustrates the need for a strict coverage. The unification path forming an inversion in q in the trivial cover (which is non-strict) is interrupted when we add $\neq$ predicates to make the cover strict.
$R(x_1, x_2), S(\underline{x_1}, x_2, \underline{y}, y),$ $S(x_1, x_1, x_2, x_2)$ $S(\underline{x'}, x', \underline{y'}, y'), T(y')$	$qc = R(x, x), S(\underline{x}, x, \underline{y}, y),$ $S(x, x, x, x), x \neq y$ $S(\underline{x'}, x', \underline{y'}, y'), T(y'), x' \neq y'$ $= R(x, x), S(x, x, x, x),$ $S(x', x', y', y'), T(y'), x' \neq y'$	This illustrates the need to minimize covers. The inversion disappears after minimizing qc.
$R(x_1, x_2), S(\underline{x_1}, x_2, \underline{y}, y)$ $S(x_1, x_2, x_1, x_2)$ $S(\underline{x'}, x', \underline{y'_1}, y'_2), T(y'_1, y'_2)$	$qc_1 = R(x, x), S(\underline{x}, x, \underline{y}, y), x \neq y$ $S(\underline{x'}, x', \underline{y'}, y'), T(y', y'), x' \neq y'$ $S(x, x, x, x)$ $qc_2 = R(x, x), S(x, x, x, x),$ $S(x', x', y', y'), T(y', y'), x' \neq y'$	This shows that we should not consider redundant coverages. There is an inversion in qc_1, but this cover is contained in qc_2 so it is redundant and after we remove qc_1 from the coverage there is no more inversion.

Figure 1: Inversion-free queries: all are in PTIME.

B. PROOF OF THEOREM 1.4

Let P be a conjunctive formula and $\mathbf{A}$ be a structure. We say that P is *decisive* w.r.t. $\mathbf{A}$ if there exists a function $c : A \to Var(P)$ s.t. for any homomorphism $h : P \to \mathbf{A}$ there exists an automorphism $i : P \to P$ s.t. denoting $h' = h \circ i$ we have $c \circ h' = id_P$. The function c, which we call a *choice* function, "chooses" for each node u in A a variable $x = c(u)$ in P such that any homomorphism from P to $\mathbf{A}$ maps x to u, up to renaming of variables in P. Let S be a class of structures. We say that P is decisive w.r.t. S if it is decisive w.r.t. to each structure in S.

In the sequel we will make use of the following two classes of graphs. A *4-partite graph* has nodes partitioned into four classes V_i, $i = 1, 2, 3, 4$, and edges are subsets of $\bigcup_{i=1}^{3} V_i \times V_{i+1}$. A *triangled-graph* has a distinguished node v_0 and two disjoint sets of nodes V_1, V_2 s.t. edges are subsets of $(\{v_0\} \times V_1) \cup (V_1 \times V_2) \cup (V_2 \times \{v_0\})$.

Example B.1 The query below checks if a graph has a chain of length 3:

```
P_3 =  E(x,y), E(y,z), E(z,u)
```

Then P_3 is decisive on the set of 4-partite graphs. To see this, the choice function simply chooses to map V_1 to x, V_2 to y, V_3 to z and V_4 to u.

Example B.2 The query below checks if the graph has a triangle:

```
T = E(x,y), E(y,z), E(z,x)
```

Then T is decisive on the class of triangled graphs. To see this, consider a triangled graph G and define c to map v_0 to x, V_1 to y and V_2 to z. A homomorphism $h : T \to G$ may map x to some other node than v_0, but after a proper rotation (automorphism) we transform h into a homomorphism $h \circ i$ that is consistent with c.

Note that T is not decisive on the class of all graphs. For example it is not decisive on the complete graph K_4.

Our interest in the two queries above and their associated classes of decisive structures comes from the fact that their complexity is #P-complete:

Query	Fragment of a strict coverage (inversion underlined)	Comments
$R(x,y), R(y,z)$	$qc = R(x,y), R(\underline{y},\underline{z})$ $qc = R(\underline{x'},\underline{y'}), R(y',z')$	Here and the inversion is between $y \sqsupset z$ and $x' \sqsubset y'$ in a copy of itself.
$R(x), S_1(x,y),$ $S_1(u_1,v_1), S_2(u_1,v_1)$ $S_2(u_2,v_2), S_2(v_2,u_2)$	$qc_1 = R(x), S_1(\underline{x},\underline{y}), x > y,$ $S_1(\underline{u_1},\underline{v_1}), S_2(\underline{u_1},\underline{v_1}), u_1 > v_1$ $S_2(\underline{u_2},\underline{v_2}), S_2(v_2,u_2), u_2 > v_2$ $qc_2 = R(x), S_1(\underline{x},\underline{y}), x < y,$ $S_1(\underline{u_1},\underline{v_1}), S_2(\underline{u_1},\underline{v_1}), u_1 < v_1$ $S_2(u_2,v_2), S_2(v_2,u_2), u_2 < v_2$	Here $x \sqsupset y$, $u_1 \equiv v_1$, $u_2 \equiv v_2$ and the inversion path goes twice through each factor. We call this an *open marked ring*.
$R(x), S(x,y), S(y,x)$	$qc_1 = R(x), S(\underline{x},\underline{y}), S(y,x), x < y$ $qc_2 = R(x'), S(x',y'), S(\underline{y'},\underline{x'}), x' > y'$	Here $x \sqsupset y$ and the inversion is between x, y and their copy y', x'. We call this a *marked ring*.
$R(x), S_1(x,y),$ $S_1(u_1,v_1), S_2(u_1,v_1)$ $S_2(u_2,v_2), S_2(v_2,u_2)$	$qc_1 = R(x), S(\underline{x},\underline{y},y), x \neq y,$ $T(\underline{u},\underline{v}), S(\underline{u},\underline{v},v), u \neq v,$ $U(y'), S(x',y',x'), x' \neq y'$ $qc_2 = R(x), S(x,y,y), x \neq y$ $T(\underline{w},\underline{v}), S(\underline{w},\underline{v},w), w \neq v,$ $U(y'), S(\underline{x'},\underline{y'},x'), x' \neq y'$	Here the inversion path goes twice through the subgoal $S(u,v,w)$ using different pairs of variables.

Figure 2: Queries with inversions: all are #P-hard

PROPOSITION B.3. *Let P_3 be the 3-chain property in Example B.1. The complexity of computing $\mathbf{P}[P_3]$ on 4-partite graphs is #P-complete.*

Let T be the triangle property in Example B.2. The complexity of computing $\mathbf{P}[T]$ on triangled graphs is #P-complete.

PROOF. By reduction from the problem of computing the probability of bipartite 2DNF formulas. Let $X = \{x_1, \ldots, x_m\}$ and $Y = \{y_1, \ldots, y_n\}$ be two disjoint sets of Boolean variables, and consider a bipartite 2DNF formula:

$$\Phi = \bigvee_{k=1,t} x_{i_k} \wedge y_{j_k} \tag{12}$$

Construct the following 4-partite graph: $V_0 = \{u\}$, $V_1 = X$, $V_2 = Y$, $V_4 = \{v\}$, where u, v are two new nodes. All edges from u to x_i are present and their probability if $\mathbf{P}[x_i]$; for each clause $x_{i_k} \wedge y_{j_k}$ in (12) there is an edge (x_{i_k}, y_{j_k}) with probability 1, and all edges (y_j, v) are present and have probability $\mathbf{P}[y_j]$. Clearly the probability that this graph has a path of length 3 is precisely $\mathbf{P}[\phi]$. This proves the hardness of P_3. The hardness of T is obtained similarly, by merging u and v into a single node. □

THEOREM B.4. *Let Q be a conjunctive formula, which is minimal, and let P be subformula. If there exists a class of structures S s.t. (1) P is decisive on S and (2) P is #P-complete on S, then Q is #P-complete on the class of all structures.*

PROOF. We reduce the problem of evaluating P on some structure in S to the problem of evaluating Q on an arbitrary structure. Let $\mathbf{A} \in S$, and $c : A \to Var(P)$ be a choice function. We construct a new structure $\mathbf{B}$ as follows. First define $H = \{h : P \to \mathbf{A} \mid c \circ h = id_P\}$ to be the set of homomorphism from P to $\mathbf{A}$ that are consistent with the choice function. Note that H is polynomial in the size of $\mathbf{A}$ since P is fixed. Define the new structure $\mathbf{B}$ as follows. Its nodes, B are obtained as follows. First define the set $N = \{(x,h) \mid x \in Var(Q), h \in H\}$; next define the equivalence relation $(x,h) \equiv (x',h')$ if $(x,h) = (x',h')$, or if $x = x' \in Var(P)$ and $h(x) = h'(x)$ (i.e. collapse multiple copies of the same variable from P if they are mapped to the same node in A). The nodes in $\mathbf{B}$ are equivalence classes $[(x,h)]$, i.e. $B = N/\equiv$. The relations in $\mathbf{B}$ are of the form $R([(x_1,h)], \ldots, [(x_k,h)])$, where $R(x_1, \ldots, x_k)$ appears in Q, and $h \in H$. One can think of $\mathbf{B}$ as consisting of multiple copies of Q, one for each possible way of mapping P into $\mathbf{A}$, but such that all copies of the same P-variable that are mapped to the same node $u \in A$ are merged into a single node. The latter are precisely the nodes of the form $[(x,h)]$ for $x \in Var(P)$, and we call them the *special nodes* in $\mathbf{B}$. Thus, the special nodes in $\mathbf{B}$ form a substructure that is isomorphic to some substructure $\mathbf{A}_0$ of $\mathbf{A}$, which is large enough to contain the image of all homomorphism from P to $\mathbf{A}$. The probabilities are as follows. If $x_1, \ldots, x_k \in Var(P)$ then

$$\mathbf{P}_B(R([(x_1,h)], \ldots, [(x_k,h)])) = \mathbf{P}_A(R(h(x_1), \ldots, h(x_k)))$$

; otherwise $\mathbf{P}_B(R([(x_1,h)],\ldots,[(x_k,h)])) = 1$. Note that there is a 1-to-1 correspondence between the worlds W_A of $\mathbf{A}$ and the worlds W_B of $\mathbf{B}$, and $\mathbf{P}[W_A] = \mathbf{P}[W_B]$.

Claim 1. Let W_A be a world of $\mathbf{A}$ s.t. $W_A \models P$. Then, denoting W_B the corresponding world of $\mathbf{B}$, we have $W_B \models Q$. Indeed, let $h : P \to \mathbf{A}$ be a homomorphism whose image uses only tuples in W_A. We can assume w.l.o.g. that it is consistent with the choice function, i.e. $c \circ h = id_P$ (otherwise simply compose it with the automorphism i), hence $h \in H$. Extended it to a homomorphism $\bar{h} : Q \to \mathbf{B}$ by defining $\bar{h}(x) = [(x,h)]$: it clearly only uses tuples in W_B.

Claim 2. Let W_B be a world of $\mathbf{B}$ s.t. $W_B \models Q$. Then, denoting W_A the corresponding world of $\mathbf{A}$ we have $W_A \models P$. Let $\bar{h} : Q \to \mathbf{B}$ be a homomorphism. If $\bar{h}$ maps $Var(P)$ only to the special nodes in $\mathbf{B}$, then we are done; but this may not necessarily be the case. We will prove instead that there exists some automorphism $g : Q \to Q$ s.t. $\bar{h} \circ g$ maps $Var(P)$ to the special nodes in $\mathbf{B}$.

Define the function $f : B \to Var(Q)$ to be $f([x,h]) = x$; one can check that it is a homomorphism from $\mathbf{B}$ to Q, and that all special nodes and only these are mapped to $Var(P)$. Consider the composition $f \circ \bar{h} : Q \to Q$, which is an isomorphism (since Q is minimal); in particular $\bar{h}^{-1}$ is functional, i.e. $|\bar{h}^{-1}(u)| \leq 1$. Define $g = (f \circ \bar{h})^{-1}$ to be its inverse. Then $\bar{h} \circ g$ maps $Var(P)$ to the special nodes in $\mathbf{B}$. Indeed, for any variable $x \in Var(P)$, $f^{-1}(x)$ consists only of special nodes, hence $\bar{h}(g(x)) = \bar{h}(\bar{h}^{-1}(f^{-1}(x))) = Dom(h^{-1}) \cap f^{-1}(x)$ is a special node. □

THEOREM B.5. *Let $P = R_1(\bar{v}_1), R_2(\bar{v}_2), R_3(\bar{v}_3)$ be a conjunctive property, which is minimal, and for which there exists two variables x, y s.t. $x \in \bar{v}_1, x \in \bar{v}_2, x \notin \bar{v}_3$ and $y \notin \bar{v}_1, y \in \bar{v}_2, y \in \bar{v}_3$. Then there exists a class of structures S s.t. (a) P is decisive w.r.t. S and (b) P is #P-complete on structures in S. Note that R_1, R_2, R_3 may be any relation names, possibly the same relation name.*

PROOF. By reduction from partitioned 2DNF. Consider Eq.(12), and recall that the variables are $X = \{x_1,\ldots,x_m\}$, $Y = \{y_1,\ldots,y_n\}$. Let $U = \{u_1, u_2, \ldots, u_k\}$ be all the variables occurring in P in addition to x and y, and C be the set of constants. Define the structure $\mathbf{A}$ s.t. $A = X \cup Y \cup U \cup C$, and the relations are defined as follows:

$$\begin{aligned} R_1^A &= \{R_1(\bar{v}_1[x_i/x]) \mid i = 1,m\} \\ R_2^A &= \{R_1(\bar{v}_1[x_{i_k}/x, y_{j_k}]/y) \mid k = 1,t\} \\ R_3^A &= \{R_3(\bar{v}_3[y_j/y]) \mid j = 1,n\} \end{aligned}$$

Thus, the tuples in the first set correspond to the Boolean variables x_i, those in the second set correspond to clauses $x_{i_k} \wedge y_{j_k}$, and those in the third set correspond to the Boolean variables y_j. Note that the three sets defined on the right are disjoint: if two or more of the relation names R_1, R_2, R_3 are the same, then their interpretation in $\mathbf{A}$ consists of the union of the corresponding right hand definitions above. The tuple probabilities are as follows: those in R_1^A are precisely $\mathbf{P}(x_i)$, those in R_2^A are 1, and those in R_3^A are precisely $\mathbf{P}(y_j)$.

We first show that P is decisive on $\mathbf{A}$. Define the choice function $c : A \to Var(P)$ to be $c(x_i) = x$ for $i = 1,m$, $c(y_j) = y$ for $j = 1,n$ and $c(u_p) = u_p$ for $p = 1,k$. We need to prove that every homomorphism $h : P \to \mathbf{A}$ is, up to isomorphism, consistent with the choice function. For that we note that the choice function itself is a homomorphism $c : \mathbf{A} \to P$, hence $c \circ h : P \to P$ is an automorphism (since P is minimal), and we denote $i = (c \circ h)^{-1}$. We show now that $h' = h \circ i$ is consistent with c. Indeed: $c \circ h' = c \circ h \circ (c \circ h)^{-1} = id_P$.

Next we prove that the probability of P being true on $\mathbf{A}$ is the same as the probability that Φ is true. There is an obvious one-to-one correspondence between worlds W_A of $\mathbf{A}$ and truth assignment for Φ: the tuple in R_1^A corresponding to x_i occurs in W_A iff $x_i = true$, and similarly for R_3^A and the y_j's. Clearly if the truth assignment makes Φ true, then P is true on W_A: simply pick two variables x_i and y_j that are both true under the truth assignment, and note that P can be mapped to the three tuples corresponding to x_i, to the clause $x_i \wedge y_j$ and to y_j respectively. Conversely, suppose P is true on W_A, i.e. there exists a homomorphism $h : P \to \mathbf{A}$ whose image is contained in W_A. Since P is decisive on $\mathbf{A}$ there exists another homomorphism $h' : P \to \mathbf{A}$ that is consistent with c, i.e. it maps x to some x_i and y to some y_j. Then $Im(h')$ consists of three tuples $R_1^A(\bar{v}_1[x_i/x])$, $R_2(\bar{v}_2[x_{i_k}/x, y_{j_k}/y])$, and $R_3^A(\bar{v}_3[y_j/y])$, and, moreover $x_{i_k} \wedge y_{j_k}$ is a clause in Φ, which is true under the truth assignment corresponding to W_A.

□

COROLLARY B.6. *Let Q be a non-hierarchical conjunctive query. Then Q is #P-hard.*

PROOF. Consider the minimal conjunctive query defined by Q. Since Q is non-hierarchical, there must be two variables x and y such that $sg(x) \cap sg(y) \neq \emptyset$, $sg(x) - sg(y) \neq \emptyset$ and $sg(y) - sg(x) \neq \emptyset$. Thus, the minimal query must contain a subformula $P = R_1(\bar{v}_1), R_2(\bar{v}_2), R_3(\bar{v}_3)$ s.t. $x \in \bar{v}_1, x \in \bar{v}_2, x \notin \bar{v}_3$ and $y \notin \bar{v}_1, y \in \bar{v}_2, y \in \bar{v}_3$.

It follows from the previous two results that Q is #P-hard. □

C. PROOF OF THEOREM 1.5

We will prove here that for every $k \geq 0$, H_k is #P-hard. Recall that

$H_k =$
$R(x)$,$S_0(x,y)$,
$\quad S_0(u_1,v_1)$,$S_1(u_1,v_1)$

$S_1(u_2, v_2)$,...
$S_{k-1}(u_k, v_k)$,$S_k(u_k, v_k)$
$S_k(x', y')$, $T(y')$

Define queries $\phi_0, \cdots, \phi_{k+1}$, where

$$\begin{aligned} \phi_0 &= R(x), S_0(x, y) \\ \phi_i &= S_{i-1}(u, v), S_i(u, v) \text{ for } 1 \leq i \leq k \\ \phi_{k+1} &= S_k(x', y'), T(y') \end{aligned}$$

Thus, $H_k = \bigwedge_{i \in [k]} \phi_i$. For any proper subset S of $[k]$, the query $\bigwedge_{i \in S}$ is in PTIME(this follows from a result we prove later that every inversion-free query is in PTIME). Using the principle of inclusion-exclusion, to show the hardness of H_k, it is enough to show the hardness of the query $\bigvee_{i \in [k]} \phi_i$ is hard, or equivalently, its negation $q = \bigwedge_{i \in [k]} (NOT \phi_i)$.

We give a reduction from the problem of computing the probability of bipartite 2DNF formulas. Let $X = \{x_1, \ldots, x_m\}$ and $Y = \{y_1, \ldots, y_n\}$ be two disjoint sets of Boolean variables, and consider a bipartite 2DNF formula:

$$\Phi = \bigvee_{h=1,t} x_{i_h} \wedge y_{j_h} \tag{13}$$

We construct an instance for relations $R, S_0, \cdots, S_k, T$. For each variable $x_i \in X$, create a tuple $R(x_i)$ and assign it a probability 1/2. For each $y_i \in Y$, create a tuple $T(y_i)$ and assign it a probability 1/2. For each clause (x_{i_h}, y_{j_h}), and for each $l \in [k]$, create a tuple $S_l((x_{i_h}, y_{j_h})$ and assign it a probability p_1 for $l = 0, k$ and a probability of p_2 for $1 \leq l \leq k-1$.

Let $T_{i,j}$ be the number of assignments of Φ such that i clauses have both variables true and j clauses have no variables true. Thus, $(t - i - j)$ have exactly 1 variable true, where t is the number of clauses.

There is a canonical mapping between the truth assignments of X, Y and worlds of relations S, T where $x \in X$ is true iff $S(x)$ is present and $y \in Y$ is true iff $T(y)$ is present.

Consider some fixed assignment where i clauses have both variables true and j clauses have no variables true. Fix relations R, T accordingly and consider all possible worlds of $S_1, \cdots, S_k$ such that q is true on the worlds. For each (x_{i_h}, y_{j_h}), consider all tuples of the form $S_l(x_{i_h}, y_{j_h})$:

1. If x_{i_h} and y_{i_h} are true, the tuples $S_0(x_{i_h}, y_{j_h})$ must be both out, and other edges do not matter. Its probability is $(1 - p_1)^2$
2. If one of them is true, one of the tuples $S_0(x_{i_h}, y_{j_h})$ must be out (depending on which variable is true), and other edges do not matter. Its probability is $(1 - p_1)$.
3. If x_{i_h} and y_{i_h} are both false, the only requirement is that not all $S_l(x_{i_h}, y_{j_h})$ are in. Its probability is $(1 - p_1^2 p_2^{k-2})$.

Thus, its total probability of all worlds corresponding to this fixed assignment is

$$(1/2)^{|X|+|Y|}[(1 - p_1)^2]^i [(1 - p_1^2 p_2^{k-2})]^j [(1 - p_1)]^{c-i-j}$$

This can be written as KA^iB^j, where $K = (1/2)^{|X|+|Y|}(1 - p)^c$, $A = (1 - p_1)$ and $B = (1 - p_1^2 p_2^{k-2})/(1 - p_1)$.

Thus $Pr[q] = \sum_{i,j:i+j \leq t} T_{i,j} K.A^i.B^j$

This is a linear equation in variables $T_{i,j}$. We put different values of p_1, p_2 to get different values of A,B and get a system of linear equations. The coefficient matrix of this set of equations is the Vandemonte matrix which is known to be invertible. By inverting the matrix, we solve for each $T_{i,j}$. Finally, we can compute the number of satisfying assignments of ϕ using $\sum_{i,j|i+j \leq t, j \neq t} T_{i,j}$. This gives a polynomial time reduction from the problem of computing H_k to counting the number of satisfying assignments of a bipartite DNF formula. Hence, H_k is #P-hard.

D. PROOF OF THEOREM 2.7

Consider some probability space. Let $\boldsymbol{U} = (U_1, \cdots, U_k)$ be a vector consisting of k sets. For each $i \in [k]$ and each $x \in U_i$, let $E(i, x)$ be an event in the probability space. Define $E(i) = \bigvee_{x \in U_i} E(i, x)$. Let Q be a CNF formula over events $E(1), \cdots E(k)$, i.e., let ψ be a set of subsets of $[k]$ and let

$$Q = \bigvee_{S \in \psi} \bigwedge_{i \in S} E(i) \tag{14}$$

We will derive an expression for $Pr[Q]$ in terms of the probabilities of the events $E(i, x)$. We need some notations. A *signature* is simply a subset of $[k]$. Given a vector of sets $\boldsymbol{S} = (S_1, \cdots, S_k)$, the signature of $\boldsymbol{S}$, denoted $sig(\boldsymbol{S})$, is the set $\{i \mid S_i \neq \emptyset\}$. $E(\boldsymbol{S})$ is defined as the event $\bigwedge_{i \in [k]} E(i, S_i)$. The size of $\boldsymbol{U}$ is defined as $|\boldsymbol{U}| = |U_1| + \cdots + |U_k|$. Also, given vectors $\boldsymbol{S}$ and $\boldsymbol{T}$, we say that $\boldsymbol{S} \subseteq \boldsymbol{T}$ iff for all $i \in [k]$, $S_i \subseteq T_i$.

Define the upward closure of ψ as $\mathtt{up}(\psi) = \{sg \mid sg \subseteq [k], \exists sg_0 \in \psi \ s.t. \ sg_0 \subseteq sg\}$. Define the minimal elements of ψ as $Factors(\psi) = \{sg \mid sg \in \psi, \forall sg_0 \in \psi.\ sg_0 \subseteq sg \Rightarrow sg_0 = sg\}$. For a set of signatures G, let $sig(G) = \cup_{sg \in G} sig(sg)$. Given a signature sg, define

$$N(sg) = (-1)^{|sg|} \sum_{G | G \subseteq Factors(\psi), sig(G) = sg} (-1)^{|G|}$$

Our main result is follows:

THEOREM D.1. *With $\boldsymbol{U}$, ψ and Q as defined above,*

$$Pr[Q] = \sum_{\boldsymbol{S} \subseteq \boldsymbol{U}} N(sig(\boldsymbol{S}))(-1)^{|\boldsymbol{S}|}$$

We will need the following result later which gives an alternate formula for $N(sg)$.

LEMMA D.2. $N(sg) = \sum_{\{sg_0 | sg_0 \subseteq sg, sg_0 \notin UP(\psi)\}} (-1)^{|sg_0|}$.

In the rest of the section, we prove this theorem.

Let $*$ be an element such that $* \notin U_i$ for all i and define $U_i^* = U_i \cup \{*\}$. Given an element $x \in U_1^* \times \cdots \times U_k^*$, the *signature* of x, denoted $sig(x)$, is a subset of $[k]$ given by $\{i \mid \pi_i(x) \neq *\}$. Given a vector of sets $\boldsymbol{S} = (S_1, \cdots, S_k)$ where $S_i \subseteq U_i$, define

$$\Pi_\psi(\boldsymbol{S}) = \{x \mid x \in (S_1 \cup \{*\}) \times \cdots (S_k \cup \{*\}), sig(x) \in \psi\}$$

Given a vector $\boldsymbol{x} \in \Pi_\psi(\boldsymbol{U})$, define $E(\boldsymbol{x}) = \bigwedge_{i \in sig(x)} E(i, \pi_i(x))$. Then, from Eq (14), it follows that

$$Q = \bigvee_{\boldsymbol{x} \in \Pi_\psi(\boldsymbol{U})} E(\boldsymbol{x})$$

Using inclusion-exclusion, we obtain

$$Pr[Q] = \sum_{T \subseteq \Pi_\psi(\boldsymbol{U})} (-1)^T Pr[\bigwedge_{x \in T} E(x)] \tag{15}$$

For a set $T \subseteq \Pi_\psi(\boldsymbol{U})$, define $\pi_i(T) = \{\pi_i(x) \mid x \in T, \pi_i(x) \neq *\}$. Also, define $E(i, S) = \bigwedge_{s \in S} E(i, s)$. Then, $\bigwedge_{x \in T} E(x) = E(1, \pi_1(T)) \wedge \cdots \wedge E(k, \pi_k(T))$.

In Eq (15, we group the T based on their projection to obtain

$$Pr[Q] = \sum_{S_1, \cdots, S_k} Pr[\bigwedge_{i \in [k]} E(i, S_i)] * (\sum_{T \subseteq \Pi_\psi(\boldsymbol{U}), \pi_i(T) = S_i} (-1)^T) \tag{16}$$

Let $N(S_1, \cdots, S_k)$ denote the sum $\sum_{T \subseteq \Pi_\psi(\boldsymbol{U}), \pi_i(T) = S_i} (-1)^T$. Thus,

$$Pr[Q] = \sum_{S_1, \cdots, S_k} N(S_1, \cdots, S_k) Pr[\bigwedge_{i \in [k]} E(i, S_i)]$$

The main result of this section is an expression for the quantity $N(S_1, \cdots, S_k)$. Given a vector $\boldsymbol{S} = (S_1, \cdots, S_k)$, define the signature of $\boldsymbol{S}$, denoted $sig(\boldsymbol{S})$, as the set $\{i \mid D_i \neq \emptyset\}$.

In an ordered set $(X, <)$, an *ideal* is a set of the form $\{x \mid x \leq a\}$, for a fixed element $a \in X$, which we denote by $[a]$.

LEMMA D.3. *If $[A]$ is an ideal in $\mathcal{P}(\mathcal{U})$, then $\sum_{\{T | T \in [A]\}} (-1)^T = 0$ if A is nonempty, and is 1 if A is empty. Note that $T \in [A]$ means $T \subseteq A$.*

For $\boldsymbol{S} = (S_1, \cdots, S_k)$, denote

$$ND(S) = \sum_{T \subseteq \Pi_\psi(\boldsymbol{S})} (-1)^{|T|}$$

LEMMA D.4. $N(\boldsymbol{S}) = \sum_{\boldsymbol{R} \subseteq \boldsymbol{S}} (-1)^{|S-R|} ND(R)$. *Here $\boldsymbol{R} = (R_1, \cdots, R_k)$ and $\boldsymbol{R} \subseteq \boldsymbol{S}$ means $R_i \subseteq S_i$ for all i.*

PROOF. Direct inclusion-exclusion applied to $N(\boldsymbol{S})$. □

Define $\mathtt{up}(\psi) = \{sg \mid sg \subseteq [k], \exists sg' \in \psi s.t. sg' \subseteq sg\}$.

LEMMA D.5. *1. If $sig(\boldsymbol{R}) \in \mathtt{up}(\psi)$, then $ND(\boldsymbol{R}) = 0$.*

2. If $sig(\boldsymbol{R}) \notin \mathtt{up}(\psi)$, then $ND(\boldsymbol{R}) = 1$.

PROOF. Follows from the fact that $sig(\boldsymbol{R}) \in \mathtt{up}(\psi)$ iff $\Pi_\psi(\boldsymbol{R}) \neq \emptyset$ and from the fact that $ND(\boldsymbol{R})$ sums $(-1)^{|T|}$, where T ranges over the ideal defined by $\Pi_\psi(\boldsymbol{R})$. □

Hence, $N(\boldsymbol{S}) = (-1)^{|\boldsymbol{S}|} \sum_{\{\boldsymbol{R} \subseteq \boldsymbol{S} : sig(\boldsymbol{R}) \notin \mathtt{up}(\psi)\}} (-1)^{\boldsymbol{R}}$.

Let sg be a signature, i.e. $sg \subseteq [k]$. Denote the quantity $M(S, sg) = \sum_{R \subseteq S : sig(R) = sg} (-1)^R$. Thus, we have:

$$N(S) = (-1)^S * \sum_{sg \notin \mathtt{up}(\psi)} M(S, sg)$$

For a signature $sg' \subseteq [k]$, denote:

$$MD(S, sg') = \sum_{R \subseteq S : sig(R) \subseteq sg'} (-1)^R$$

LEMMA D.6. $M(S, sg) = \sum_{sg' \subseteq sg} (-1)^{|} sg - sg' | * MD(S, sg')$

PROOF. Again inclusion/exclusion formula applied to the set sg. □

LEMMA D.7. *1. If $sg' \cap sig(S) \neq \emptyset$, then $MD(S, sg') = 0$.*
2. If $sg' \cap sig(S) = \emptyset$, then $MD(S, sg') = 1$.

PROOF. Follows from the fact that the set $\{R \mid R \subseteq S, sig(R) \subseteq sg'\}$ is an ideal, and it is nonempty iff $sg' \subseteq sig(S)$. □

Next, we manipulate the expression $M(S, sg)$ as follows. We have $M(S, sg) = (-1)^{sg} M'(S, sg)$, where:

$$\begin{aligned} M'(S, sg) &= \sum_{sg' \subseteq sg} (-1)^{sg'} MD(S, sg') \\ &= \sum_{sg' \subseteq sg, sg' \cap sig(S) = \emptyset} (-1)^{sg'} \\ &= \sum_{sg' \subseteq (sg - sig(S))} (-1)^{sg'} \end{aligned}$$

This is a sum over the ideal generated by $sg - sig(S)$. This ideal contains only the empty set when $sg \subseteq sig(S)$, hence:

LEMMA D.8. *1. If $sg \subseteq sig(S)$, then $M(S, sg) = (-1)^{sg}$*
2. If $sg \not\subseteq sig(S)$, then $M(S, sg) = 0$.

Hence,

$$\begin{aligned} N(S) &= (-1)^S * \sum_{sg \notin \mathtt{up}(\psi)} M(S, sg) \\ &= (-1)^S * \sum_{sg \notin \mathtt{up}(\psi), sg \subseteq sig(S)} (-1)^s g \\ &= (-1)^S * \sum_{sg \subseteq sig(S)} (-1)^{sg} - (-1)^S * \sum_{sg \in \mathtt{up}(\psi), sg \subseteq sig(S)} (-1)^{sg} \\ &= -(-1)^S * \sum_{sg \in UP(\psi), sg \subseteq sig(S)} (-1)^{sg} \end{aligned}$$

The last equality holds because we assume $sig(S) \neq \emptyset$, hence $sg \subseteq sig(S)$ is a non-empty ideal.

THEOREM D.9. $N(S) = -(-1)^S * \sum_{sg \in \mathtt{up}(\psi), sg \subseteq sig(S)} (-1)^{sg}$.

Next, assume that $\mathtt{up}(\psi)$ is generated by the set $\psi = \{\phi_1, \cdots, \phi_p\}$, where each factor ϕ_i is a subset of $[k]$. Then we apply inclusion exclusion to (N6):
$N(S) = -(-1)^S \sum_{G \subseteq [p]} (-1)^{|G-1|} \sum_{\cup_{i \in G} \phi_i \subseteq sg \subseteq sig(S)} (-1)^{sg}$.
In the inner sum sg ranges over the interval $[\cup_{i \in G} \phi_i, sig(S)]$, hence the sum is $(-1)^{sig(S)}$ when $\cup_{i \in G} = sig(S)$ and 0 otherwise. It follows:

THEOREM D.10. $N(S) = (-1)^S (-1)^{sig(S)} \sum_{sig(G) = sig(S)} (-1)^G$.

E. PROOF OF THE DICHOTOMY THEOREM

E.1 Unifiers

In this section, we define a set $\mathcal{H}(q)$, called the set of hierarchical unifiers of q, by starting from the factors of q and unifying them in certain way.

DEFINITION E.1. (*Hierarchical join predicate*) Let q_1 and q_2 be two strict hierarchical queries with disjoint sets of variables and let $g_1 \in$ *subgoals*(q_1) and $g_2 \in$ *subgoals*(q_2) be any two sub-goals that are unifiable. Thus, g_1 and g_2 have same arity, say a. Let $m_u : Vars(g_1) \to Vars(g_2)$ be the most general unifier of g_1 and g_2, which is a bijection. Let $x_1 \sqsubseteq \cdots \sqsubseteq x_a$ be all the variables in g_1 and $y_1 \sqsubseteq \cdots \sqsubseteq y_a$ be all the variables in g_2. Let w be the largest integer such that $m_u(x_i) \equiv y_i$ for $1 \leq i \leq w$. A *hierarchical join predicate* between q_1 and q_2 is the set $\{(x_i, m_u(x_i)) \mid 1 \leq i \leq w\}$

DEFINITION E.2. (*Hierarchical Unifier*) Let q_1 and q_2 be two strict hierarchical queries with disjoint sets of variables and let jp be some hierarchical join predicate between them. A hierarchical unifier of q_1 and q_2 is a query obtained by considering

$$q_u \leftarrow q_1, q_2, \bigwedge_{(x_i, x_j) \in jp} (x_i = x_j)$$

and removing all $=$ predicates by substituting.

LEMMA E.3. *Let q_u be a hierarchical unifier of two strict hierarchical queries q_1 and q_2. Then, q_u is a strict hierarchical query.*

PROOF. TBD. □

The above result justifies the name "hierarchical unifier", because such unifiers are always hierarchical. Next we define a set $\mathcal{H}(q)$, called the set of hierarchical unifiers of q, along with a function *Factors* from $\mathcal{H}(q)$ to subsets of $\mathcal{F}(q)$. They are constructed inductively as follows:

1. For each $q \in \mathcal{F}(q)$, add q to $\mathcal{H}(q)$ and let $Factors(q) = \{q\}$.
2. If q_1, q_2 are in $\mathcal{H}(q)$, and q_u is their hierarchical unifier, add q_u to $\mathcal{H}(q)$ if it is not logically equivalent to any existing query in $\mathcal{H}(q)$. Also, define $Factors(q_u)$ to be $Factors(q_1) \cup Factors(q_2)$

LEMMA E.4. *The set $\mathcal{H}(q)$ is finite.*

PROOF. All queries in $\mathcal{H}$ are hierarchical by Lemma E.3. There are only finitely many hierarchical queries up to equivalence on a given set of relations and given set of constants. [[Expand this proof]]. □

E.2 The Polynomial Time Algorithm

Let $\mathcal{H}^*(q)$ be the subset of $\mathcal{H}(q)$ containing queries which are either inversion-free or in $\mathcal{F}(q)$.

E.2.1 Query expansion

Let $\mathcal{H}^*(q) = \{qh_1, qh_2, \cdots, qh_k\}$. Define

$$\psi = \{S \mid S \subseteq [k], qc_i \subseteq (\bigcup_{i \in S} Factors(qh_i)) \text{ for some } qc_i \in \mathcal{C}(q)\}$$

Thus, ψ contains all combinations of hierarchical unifiers that make q true. Let $Factors(\psi)$ be the minimal elements of ψ.

LEMMA E.5. *With ψ as defined above,*

$$q \equiv \bigvee_{S \in \psi} \bigwedge_{i \in S} qh_i$$

PROOF. The $\Leftarrow$ direction is obvious from the definition of ϕ. For the $\Rightarrow$ direction, consider any mapping η of q into the database. Consider the factor corresponding to that mapping and the set of its connected components. This set is in in ϕ and hence $\bigvee_{S \in \psi} \bigwedge_{i \in S} qh_i$ is true on the database. □

We then apply the generalized inclusion-exclusion formula from Sec D to obtain:

$$Pr[q] = \sum_{G \subseteq Factors(\psi)} (-1)^{|G|+|sig(G)|} \sum_{\{T \mid sig(T)=sig(G)\}} (-1)^T Pr[qh(T)]$$

where $qh(T) = qh_1(\pi_1(T)), qh_2(\pi_2(T)), \cdots, qh_k(\pi_k(T))$.

Define $coeff(sg) = (-1)^{|sg|} \sum_{G \subseteq Factors(\psi), sig(G)=sg} (-1)^{|G|}$. The sum can alternatively be rewritten as:

$$Pr[q] = \sum_{sg \subseteq [K]} F(sg)$$

where

$$F(sg) = coeff(sg) \sum_{\{T \mid sig(T)=sg\}} (-1)^{|T|} Pr[qh(T)]$$

E.2.2 Adding Independence Predicates

Let $\bar{x_1}, \cdots, \bar{x_k}$ be the set of variables of $qh_1, \cdots, qh_k$. Define new relational symbols $S_1, \cdots, S_k$ where the arity of S_i equals $|\bar{x_i}|$. Given any join predicate jp between qh_i and qh_j, consider the following conjunctive query:

$$q_{jp}() \rightarrow S_i(\bar{x_i}), S_j(\bar{x_j}), \bigwedge_{(x,y) \in jp} (S_i.x = S_j.y)$$

Given any set $T = (T_1, \cdots, T_k)$, let $q_{jp}(T)$ be the predicate which is true if q_{jp} is true when evaluated on T, i.e. by setting T_i to be the instance of S_i.

An *independence predicate* is simply the negation of a join predicate. Let Q_{ip} be the set of all independence predicates, i.e., $Q_{ip} = \{\mathtt{not}(q_{jp}) \mid q_{jp} \in Q_{jp}\}$.

We divide the join predicates into two disjoint sets, *trivial* and *non-trivial*. A join predicate between factors h_i and h_j is called trivial if the join query is equivalent to either h_i or h_j, and is called non-trivial otherwise. We write $\mathtt{ip}(\mathcal{C}^*)$ as $\mathtt{ip}^n(\mathcal{C}^*) \wedge \mathtt{ip}^t(\mathcal{C}^*)$, where $\mathtt{ip}^n(\mathcal{C}^*)$ is the conjunction of $\mathtt{not}(jp)$ over all non-trivial join predicates jp, and $\mathtt{ip}^t(\mathcal{C}^*)$ is the conjunction over all trivial join predicates.

For a signature sg, let $\mathtt{ip}^n(sg)$ denote the subset of Q_{ip} consisting of independence predicates between all S_i and S_j such that $i, j \in sg$. Let π be a function that maps each signature sg to a set of independence predicates $\pi(sg) \subseteq \mathtt{ip}^n(sg)$. Denote:

Let π be any predicate on T, i.e. a query over the relations $S_1, \cdots, S_k$. Define

$$sum(\pi) = \sum_{T:\pi(T)} N(sig(T))(-1)^T Pr[qh(T)]$$

Thus, the probability of q is simply $sum(\emptyset)$, where $\emptyset$ is the predicate that is identically true. Define $\mathtt{ip}^n$ to be the conjunction of all independence predicates between queries in $\mathcal{H}^*(q)$, i.e. $\mathtt{ip}^n = \bigvee_{jp} \mathtt{not}(jp)$, where jp ranges over all join predicates between all $q_i, q_j \in \mathcal{H}^*(q)$.

We will prove that when the query satisfies the PTIME conditions, then $sum(\emptyset) = sum(\mathtt{ip}^n)$.

Definition E.6. (*Eraser*) Let qh_i and qh_j be any two strict hierarchical queries in $\mathcal{H}^*(q)$ and let q_{ij} be their unifier corresponding to some join predicate jp. An *eraser* for the unifier q_{ij} is a set of queries $E \subseteq \mathcal{H}^*(q)$ such that:

1. For all $q \in E$, $q \rightarrow q_{ij}$
2. For all $sg \subseteq [k]$, $N(sg \cup \{i, j\}) = N(sg \cup \{i, j\} \cup \{k \mid qh_k \in E\})$.

Theorem E.7. *Suppose for every q_i, q_j, q_{ij} such that $q_i, q_j \in \mathcal{H}^*(q)$ and q_{ij} is a hierarchical unifier of q_i and q_j, either $q_{ij} \in \mathcal{H}^*(q)$ or it has an eraser. Then, $sum(\emptyset) = sum(\mathtt{ip}^n)$.*

We will prove Theorem E.7 in the rest of this section.

Let N be the size of the domain for the database. Let $\mathcal{S}$ denote the vocabulary $S_1, \cdots, S_k$. Let $\mathcal{Q}_{N,\mathcal{S}}(k)$ be the set of conjunctive queries of arity k over $\mathcal{S}$ that are equivalent on domain of size N. For each $q \in \mathcal{Q}_{N,\mathcal{S}}(k)$, define the following

$$q^* = \exists \bar{x}.q(\bar{x}) \wedge (\bigwedge_{\{q' | q' \in \mathcal{Q}_{N,\mathcal{S}}(k), q' \text{ contains } q\}} \mathtt{not}(q'(\bar{x}))$$

Let $\mathcal{Q}^*_{N,\mathcal{S}}(k) = \{q^* \mid q \in \mathcal{Q}_{N,\mathcal{S}}(k)\}$ and let $\mathcal{Q}^*_{N,\mathcal{S}} = \cup_{k \geq 0} \mathcal{Q}^*_{N,\mathcal{S}}(k)$. Each of the query in $\mathcal{Q}^*_{N,\mathcal{S}}$ is Boolean, hence it contains only finitely many queries up to equivalence on domains of size N, which we denote $\{qs^*_1, qs^*_2, \cdots, qs^*_t\}$. For each qs^*_i, qs_i denotes the conjunctive query which is the positive part of qs^*_i.

We call each such query a *cell*. A *cell signature* is any subset of $\mathcal{Q}^*_{N,\mathcal{S}}$. Given a cell signature $csig$, it defines the following query

$$(\bigwedge_{q \in csig} q) \wedge (\bigwedge_{q \notin csig} \mathtt{not}(q))$$

Given a set T, we say $T \models csig$ if T satisfies the query defined by $csig$. The cell signatures partition the sets of all T. Thus, we have

$$\begin{aligned} Pr[q] &= \sum_T N(sig(T))(-1)^T Pr[qh(T)] \\ &= \sum_{csig} \sum_{\{T | T \models csig\}} N(sig(T))(-1)^T Pr[qh(T)] \end{aligned}$$

We say that a cell signature $csig$ contains a join predicate if their is a cell $qs^*_i \in csig$ and a join predicate query q_{jp} such that $qs^*_i \subseteq q_{jp}$.

Lemma E.8. *Let q be the union of all cell signatures that do not contain any join predicate. Then $T \models q$ iff T satisfies all the independence predicates.*

Proof. □

To prove Theorem E.7, we only need to show that the total contribution of all cell signatures that contain at least one join predicate is 0. We will show this by grouping cell signatures into groups of three.

Let $F(csig)$ denote the quantity $\sum_{T|T \models csig} N(sig(T))(-1)^T Pr[qh(T)]$. Let qh_i, qh_j be any two hierarchical queries with unifier q_u corresponding to the join predicate jp. q_{jp} is the join predicate query on the S vocabulary. Let $E = \{qh_{l_1}, \cdots, qh_{l_m}\}$ be its eraser. Thus, there is a mapping $h : qh_{l_1}, \cdots, qh_{l_m} \rightarrow q_u$. Let $q_{jp,E} = h(S_{l_1}), \cdots, h(S_{l_m}), q_{jp}$.

Let qs_m be any query that contains q_{jp} but not $q_{jp,E}$. Thus, there is a mapping $g : q_{jp}$ to qs_m. Let $f = h \circ g$ and let $qs'_m = f(S_{l_1}), \cdots, f(S_{l_m}), qs_m$.

Let $csig_0$ be any subset of Q_S that does not contain qs_m and qs'_m.

Lemma E.9. *With $csig_0$, qs_m and qs'_m as defined above,*

$$F(csig_0 \cup \{qs_m\}) + F(csig_0 \cup \{qs'_m\}) + F(csig_0 + \{qs_m, qs'_m\}) = 0$$

PROOF. Consider any $T0$ that satisfies either of the three cells. Then, $T0$ satisfies the query qs_m (note that qs'_m contains the query qs_m). Let H_{l_i} be the set of tuples obtained for S_{l_i} from $qs_m(T0)$ using the mapping f.

Let T' be obtained from $T0$ by removing tuples H_{l_i} from T_{l_i} for all i in the eraser. Now we fix T' and look at all the T satisfying either of the three cells and which gives rise to the same T'. Every such T is obtained by adding some subset of H_{l_i} to T'_{l_i}.

Claim: Every possible T obtained from T' by adding some subset of H_{l_i} to T'_{l_i} satisfies one of the three cells.

Further, for a fixed T', each T gives rise to the same query $qh(T)$. Thus, when we sum over all such T, we get an ideal which is 0. Summing over all T', we get that the total contribution of the three cells is 0 □

LEMMA E.10. *The set of all cell signatures that contain at least one join predicate can be partitioned into groups of three of the form in Lemma E.9.*

PROOF. Each triplet is defined by (i) a join predicate q_{jp} with an eraser E, (ii) a pair of queries qs_a and qs_b where qs_a contains q_{jp} but not $q_{jp,E}$ and qs_b is obtained from qs_a by attaching E, and (iii) a subset of cells $csig_0$. The triplet is then given by: $csig_0 \cup \{qs_a\}$, $csig_0 \cup \{qs_b\}$ and $csig_0 \cup \{qs_a, qs_b\}$.

Now, given any $csig$ containing a join predicate, define q_{jp}, qs_a, qs_b and $csig_0$ as follows. Order the set of all join predicates and the set of cells and pick a canonical eraser for each join predicate. Let q_{jp} be the smallest join predicate in $csig$. Let E be the canonical eraser for qs_i and let $q_{jp,E}$ be the query as described above.

Let qs_m be the smallest cell in $csig$ that contains q_{jp}. If qs_m does not contain $q_{jp,E}$, let $qs_a = qs_m$ and define qs_b appropriately. Note that $csig$ may not contain qs_b. If qs_m contains $q_{jp,E}$ let $qs_b = qs_m$ and define qs_a appropriately. Again, $csig$ may not contain qs_a. Let $csig_0$ be all the cells in $csig$ except qs_a and qs_b. This defines the triplet for qs_m.

Claim: every cell signature containing a join predicate belongs to a unique triplet.

This follows from Lemma E.11 □

LEMMA E.11. *Let q_i and q_j be two queries with a join predicate q_u that has an inversion. Suppose E is an eraser for q_u, such that there is a mapping $h : E \to q_u$. Then, for any $q_l \in E$, $h(q_l), q_i$ is hierarchical.*

PROOF. Suppose on the contrary there is an inversion between $R(x), S(x,y) \in q_l$ and $S(x',y'), T(y') \in q_i$ such that $h(x) = x'$, $h(y) = y'$, where $R(x)$ is some subgoal that contains x but not y, $S(x,y)$ is some subgoal containing both x and y, $S(x',y')$ is a subgoal containing both x' and y' and $T(y')$ is a subgoal containing y' but not x'.

There are two cases: the join predicate between q_i and q_j does not touch variable x' in q_j. Then, no subgoal of q_i in q_u contains the variable x'. So, h maps $R(x)$ to some subgoal in q_j itself. Thus, q_j is not hierarchical, which is a contradiction.

Hence, the join predicate between q_i and q_j uses the variable x'. It also uses y' because $x' \sqsubseteq x$. Now, since q_l and q_i have inversion, there must be an eraser E' that has a mapping to $h(q_l), q_i$. This eraser only uses a portion of the partial unifier of q_i, q_j, hence there is a mapping $E' \to q_u$. □

E.2.3 Change of Basis

We have

$$Pr[q] = \sum_{T:\mathtt{ip}^n(T)} N(sig(T))(-1)^T Pr[qh(T)]$$

For each i, we expand $qh_i(T_i)$ into the relations it contains. We group all the T that result in the same $qh(T)$.

Each qh_i is a connected hierarchical query. Let $\sqsubset$ be the hierarchy relation on $Vars(qh_i)$. Define a *hierarchy tree* for qh_i as follows. The nodes of the trees are certain subsets of $Vars(qh_i)$. For each subset of the set of subgoals of qh_i, there is a node in the hierarchy tree consisting of the intersection of variables of those subgoals. A node n is a child of n' if $n \subset n'$ and there is no n'' such that $n \subset n'' \subset n'$.

For each node in the hierarchy tree of qh_i, we define a new relational symbol whose attributes are the variables in that node. Let $\mathcal{S}^i = \{S^i_0, S^i_1, \cdots\}$ be the set of new relational symbols and let $\{X^i_0, X^i_0, \cdots\}$ be the corresponding sets of variables.

Consider any vector $U_i = (U_{i_1}, U_{i_2}, \cdots)$, where $U_{i_j} \subseteq A^{Arity(S^i_j)}$. We say that $T \models U$ if for all i, j, U_{i_j} is the projection of T_i on the variables X^i_j. Define $F_i(U_i) = \prod_j \prod_{g|Vars(g)=X^i_j} Pr[qh_i(U_{i_j})]$ and let $F(U) = F_1(U_1) \times \cdots \times F_k(U_k)$. Then, if $T \models U$ and T satisfies all the independence predicates, we have $Pr[qh(T) = F(U)$.

We rewrite $Pr[q]$ as

$$Pr[q] = \sum_U \sum_{\{T|U\models T, \mathtt{ip}^n(T)\}} N(sig(T))(-1)^T Pr[qh(T)]$$

The signature of T can be determined by the signature of U in straightforward way, and we write $N(sig(T))$ as $N(sig'(U))$. Also, we write $Pr[qh(T)]$ as $F(U)$. We have

$$Pr[q] = \sum_U N(sig'(U))F(U) \sum_{\{T|U\models T, \mathtt{ip}^n(T)\}} (-1)^T$$

Next, we note that $\mathtt{ip}^n(T)$ is independent of T for a given U, and we move the independence predicates to U as follows: For each independence predicate between sub-goal g_1 of qh_{i_1} and sub-goal g_2 of qh_{i_2}, we add the independence predicate

$\texttt{not}(S^{i_1}_{j_1}(\bar{x}), S^{i_2}_{j_1}(\bar{x}))$, where $S^{i_1}_{j_1}$ is the relation corresponding to $Vars(g_1)$ and $S^{i_2}_{j_2}$ is the relation corresponding to $Vars(g_12$. Let $\texttt{ip}^n(U)$ be the conjunction of all such predicates. Then,

$$Pr[q] = \sum_{\{U|\texttt{ip}^n(U)\}} N(sig'(U))F(U) \sum_{\{T|U\models T\}} (-1)^T$$

Not all possible U have a possible T. For instance, if relations $S^i_{j_1}$ and $S^i_{j_2}$ share a set of variable X, then U must have $\pi_X(S^i_{j_1}) = \pi_X(S^i_{j_2})$. We use the hierarchy tree to determine when a U has a possible T. For each $S^i_{j_1}$ and $S^i_{j_2}$ such that $S^i_{j_1}$ is a child of $S^i_{j_2}$, define the predicate $S^i_{j_1} = \pi_X(S^i_{j_2}$, where X is the set of variables in $S^i_{j_1}$. Let ϕ be the conjunction of all such predicates on U.

LEMMA E.12. *Let $f(U^i_j)$ be a function which is $(-1)^{|U^i_j|}$ if S^i_j has even number of children in the hierarchy tree of qh_i and 1 otherwise. Then, $\sum_{\{T|U\models T\}}(-1)^{|T|} = \prod_{i,j} f(U^i_j)$ if U satisfies ϕ and 0 otherwise.*

Using the above lemma, we get

$$\begin{aligned} Pr[q] &= \sum_{\{U|\texttt{ip}^n(U),\phi(U)\}} N(sig'(U))f(U)F(U) \\ &= \sum_{sig} N'(sig) \sum_{\{U|\texttt{ip}^n(U),\phi(U),sig(U)=sig\}} f(U)F(U) \end{aligned}$$

Next, we add the remaining independence predicates, namely $\texttt{ip}^t$. Consider all pairs $S^{i_1}_{j_1}$ and $S^{i_2}_{j_2}$ in the query that refer to the same predicate and which have not been separated using $\texttt{ip}^n$. Fix an ordering on the subgoals of the query, and let $g(U^i_j)$ be a function which is $(-1)^{|U^i_j|}$ if there are odd number of subgoal less than S^i_j that need to be separated from S^i_j and 1 otherwise. Then,

$$Pr[q] = \sum_{sig} N'(sig) \sum_{\{U|\texttt{ip}^n(U),\texttt{ip}^t(U),\phi(U),sig(U)=sig\}} g(U)f(U)F(U)$$

We observe that computing the inner sum is equivalent to evaluating the query $(\texttt{ip}^n(U) \wedge \texttt{ip}^t(U) \wedge \phi(U) \wedge sig(U) = sig)$ on a probabilistic database with schema S^i_j and instance U^i_j and probabilities given by $Pr[t \in S^i_j] = g(t)f(t)F(t)/(1+g(t)f(t)F(t))$.

Finally, to evaluate $(\texttt{ip}^n(U) \wedge \texttt{ip}^t(U) \wedge \phi(U) \wedge sig(U) = sig)$, we negate $\texttt{ip}^n$ and use inclusion-exclusion to represent it as probabilities of finite number of conjunctive queries (with negated subgoals due to ϕ. Each such conjunctive query is inversion-free [[need to give more details here]], because the $\texttt{ip}^n \wedge \texttt{ip}^t$ part consists of a bunch of join predicates corresponding to hierarchical unifiers, and the ϕ part also contains the same join predicates (but with negated sub-goals). So the resulting query is inversion-free and can be evaluated in PTIME.

E.3 Hardness Proof

The main result of this section is that if there is a hierarchical unifier that contains an inversion but does not have an eraser, then the query is $\#P$-hard. This shows that the PTIME condition and the hardness condition complement each other.

THEOREM E.13. *Let $q_i, q_j \in \mathcal{H}^*(q)$ and let q_k be their hierarchical unifier q_k such that*

1. *q_k contains an inversion.*
2. *q_k does not have any eraser.*

Then, q is $\#P$-complete.

We prove this in the rest of this section. First, we need some definitions and results.

DEFINITION E.14. (*Redundant Set of Covers*) A set of covers $qc_1, \cdots, qc_k$ is *strictly redundant* if there exists a mapping $h : qc \to qc_1, \cdots, qc_k$, where qc is not among $qc_1, \cdots, qc_k$. A set of covers is *redundant* if it contains a strictly redundant subset of covers.

DEFINITION E.15. Let $qc_0, \cdots, qc_k$ be a non-redundant set of covers. Let $qcs \leftarrow qc_0, \cdots, qc_k$ and define the *cover-set* query to be the minimization of qcs :

$$qcs' = \text{minimize}(qcs) = qc'_0, \cdots, qc'_k$$

where each qc'_i is a subset of subgoals of qc_i. Denote the *inclusions* and the *projection* homomorphisms:

$$\begin{aligned} in_i &: qc_i \to qcs \quad i = 0, 1, \cdots, k \\ in &: qcs' \to qcs \\ pr &: qcs \to qcs' \end{aligned}$$

Note that $pr \circ in$ is the identity mapping on qcs'.

DEFINITION E.16. *The mappings $h_i : q \to qcs'$ obtained by composing $h : q \to qc_i$ (the cover mapping), in_i (the i^{th} inclusion) and pr (the projection) are called* canonical mappings.

LEMMA E.17. *If F is a non-redundant set of covers, then every mapping from q to the cover-set query of F is canonical upto isomorphism.*

DEFINITION E.18. (*Extension*) Let $qh \in \mathcal{H}(q)$ be any hierarchical unifier with $Factors(qh) = \{qf_1, \cdots, qf_k\}$. Let $qc_1, \cdots, qc_k$ be a multiset of covers such that qc_i contains the factor qf_i. An *extension* of qh is a query qce' obtained by minimizing $qce = qc_1, \cdots, qc_k, qh$. Define the inclusion homomorphism $in : qce' \to qce$, the i^{th} inclusion homomorphism $in_i : qc_i \to qce$ and the projection homomorphism $pr : qce \to qce'$ in the natural way. Also, define canonical mappings for extensions as we defined it for cover-sets above.

Now some hardness results.

LEMMA E.19. *Let $C = \{qc_1, \cdots, qc_k\}$ be a non-redundant set of covers such that their cover-set qcs has an inversion. Then, q is #P-hard.*

PROOF. Without loss of generality, we can assume that for any proper subset of C, the cover-set does not have an inversion (otherwise we replace C with the smaller set and repeat the argument).

Let the inversion in qcs consist of

$$g_0(x), h_0(x, y), g_1(u_1, v_1), h_1(u_1, v_1), \cdots, g_{n-1}(u_{n-1}, v_{n-1}), h_{n-1}(u_{n-1}, v_{n-1}), g_n(x', y'), h_n(y')$$

where subgoals h_i and g_{i+1} refer to the same relation. For each $qc_i \in C$, define the type of qc_i as the subset of $[n]$ consisting of all t such that the image of qc_i under the pr homomorphism contains the subgoals g_t, h_t.

Claim: for each qc_i, its type contains at least one t which is not present in any other type.

This follows from the minimality of the set F, because if qc_i does not contribute any unique t, then we can remove if from F and still get an inversion in the cover-set query.

[[Next use the inclusion-exclusion on the types, and argue that exactly one conjunct of types is #P-hard (namely, one that contains all the types. Use this to give a reduction from RSSS..ST query]]

□

LEMMA E.20. *Let $qh \in \mathcal{H}(q)$ be a hierarchical unifier that has an extension qce such that all the mappings from $q \to qce$ are canonical. Then q is #P-hard.*

PROOF. We use the extension qce to find a non-redundunct set of covers whose cover-set has an inversion.

Let $\{qc_1, \cdots, qc_k\}$ be the set of covers used in the extension of qce. Let h be a mapping that maps each variable that does not participate in the inversion to a unique contant. Construct a new set of covers $F' = \{qc'_1, \cdots, qc'_k\}$ where $qc'_i = h(qc_i)$. Note that the resulting queries are indeed covers. We will show that F' is a non-redundunct set of covers whose cover-set has an inversion.

It is easy to see that the cover-set of F' is precisely the query $h(qce)$. Since h does not touch the variables that participate in the inversion, $h(qce)$ also contains an inversion, and so does $h(qce)$. To prove that F' is non-reduntant, we note that a non-canonical mapping from some cover qc to the cover-set of F' gives a non-canonical mapping from a different cover qc' (obtained by replacing the new constants back by varaibles) to the extension qce. This is a contradiction, hence all mappings into the cover-set of F' are canonical. So F' is non-reduntant. □

LEMMA E.21. *Let $qh \in \mathcal{H}(q)$ be a hierarchical unifier that does not have an eraser. Then, there is an extension qce such that all the mappings from $q \to qce$ are canonical.*