A Dichotomy for Homomorphism-Closed Queries on Probabilistic Graphs

Antoine Amarilli
LTCI, Télécom Paris, Institut Polytechnique de Paris, France

İsmail İlkan Ceylan
University of Oxford, United Kingdom

Abstract

We study the problem of probabilistic query evaluation (PQE) over probabilistic graphs, namely, tuple-independent probabilistic databases (TIDs) on signatures of arity two. Our focus is the class of queries that is closed under homomorphisms, or equivalently, the *infinite* unions of conjunctive queries, denoted UCQ$^\infty$. Our main result states that all *unbounded* queries in UCQ$^\infty$ are #P-hard for PQE. As *bounded* queries in UCQ$^\infty$ are already classified by the dichotomy of Dalvi and Suciu [17], our results and theirs imply a complete dichotomy on PQE for UCQ$^\infty$ queries over probabilistic graphs. This dichotomy covers in particular all fragments in UCQ$^\infty$ such as *negation-free (disjunctive) Datalog*, *regular path queries*, and a large class of *ontology-mediated queries* on arity-two signatures. Our result is shown by reducing from counting the valuations of positive partitioned 2-DNF formulae (#PP2DNF) for some queries, or from the source-to-target reliability problem in an undirected graph (#U-ST-CON) for other queries, depending on properties of minimal models.

2012 ACM Subject Classification Theory of computation → Database query processing and optimization (theory)

Keywords and phrases Tuple-independent database, #P-hardness, recursive queries, homomorphism-closed queries

Digital Object Identifier 10.4230/LIPIcs.ICDT.2020.5

Related Version A full version of the paper containing all missing proofs is available at [3], `https://arxiv.org/abs/1910.02048`.

Funding This work was supported by the UK EPSRC grant EP/R013667/1.

1 Introduction

The management of *uncertain and probabilistic data* is an important problem in many applications, e.g., automated knowledge base construction [19, 25, 28], data integration from diverse sources, predictive and stochastic modeling, applications based on (error-prone) sensor readings, etc. To represent probabilistic data, the most basic model is that of tuple-independent *probabilistic databases (TIDs)* [33]. In TIDs, every fact of the database is viewed as an independent random variable, and is either kept or discarded according to some probability. Hence, a TID induces a probability distribution over all *possible worlds*, that is, all possible subsets of the database. The central inference task for TIDs is then *probabilistic query evaluation* (PQE): Given a query Q, compute the probability of Q relative to a TID $\mathcal{I}$, i.e., the total probability of the possible worlds where Q is satisfied.

Dalvi and Suciu [17] obtained a dichotomy for PQE on *unions of conjunctive queries (UCQs)*, measured in *data complexity*, i.e., as a function of the input TID and with the query being fixed. They have shown that PQE can be solved in polynomial time for some UCQs (called *safe*), and that it is #P-hard for all other UCQs (called *unsafe*). Their result was the foundation of many other studies of the complexity of PQE [2, 14, 21, 27, 29, 30, 32].

Despite this extensive research on TIDs, there is little known about PQE for monotone query languages beyond UCQs. In particular, only few results are known for languages featuring *recursion*, which is an essential ingredient in many applications: it is unknown if PQE admits a dichotomy for Datalog, or for ontology-mediated queries [13].

In this work, we focus on a large class of queries beyond first-order: we study the queries that are *closed under homomorphisms*. We denote the class of such queries by UCQ^∞ as they are equivalent to *infinite* unions of conjunctive queries. We distinguish between *bounded* UCQ^∞ queries, which are logically equivalent to a UCQ, and *unbounded* UCQ^∞ queries, which cannot be expressed as a UCQ. Notably, UCQ^∞ captures (negation-free) disjunctive Datalog, regular path queries (RPQs) and a large class of ontology-mediated queries.

We study the framework of *probabilistic graphs*, i.e., probabilistic databases where all relations have *arity two*. Arity-two relations are the formalism used in description logics, and in works on knowledge graphs and information extraction such as NELL [28], Yago [25], and Google's Knowledge Vault [19]. In these contexts, we wish to evaluate unbounded queries on the data, e.g., RPQs or other UCQ^∞ queries, while taking into account its uncertainty. Therefore, we study the complexity of query evaluation on probabilistic graphs, and ask if PQE for the class UCQ^∞ admits a data complexity dichotomy in this case.

The main result of this paper is to show that PQE is #P-hard for *any unbounded* UCQ^∞ *query* over probabilistic graphs. Our result thus implies a dichotomy on PQE for UCQ^∞ over such graphs: as *bounded* UCQ^∞ queries are equivalent to UCQs, they are already classified by Dalvi and Suciu, and we show that all other UCQ^∞ queries are unsafe, i.e., the PQE problem is #P-hard for them. Of course, it is not surprising that *some* unbounded queries in UCQ^∞ are unsafe for similar reasons as unsafe UCQs, but the challenge is to show hardness for *every* unbounded UCQ^∞ query. We conjecture that the same result holds also on arbitrary arity signatures, but we leave this question open, as we explain in the Conclusion.

The proof has two main parts: First, we study UCQ^∞ queries with a model featuring a so-called *non-iterable edge*. For all such queries, we reduce from the problem of counting the valuations of positive partitioned 2-DNF formulae (#PP2DNF). Second, for other unbounded queries in UCQ^∞, we reduce from the source-to-target reliability problem in an undirected graph (#U-ST-CON): this second step is harder and relies on a study of minimal models.

Related work. Research on probabilistic databases is a well-established field; see e.g. [33]. The first dichotomy for queries on such databases was shown by Dalvi and Suciu [16]: a self-join-free conjunctive query is safe if it is *hierarchical*, and #P-hard otherwise. They then extended this result to a dichotomy on all UCQs [17]. Beyond UCQs, partial dichotomy results are known for some queries with negation [21], with disequality ($\neq$) joins in the queries [29], or with inequality ($<$) joins [30]. Some results are known for extended models, e.g., the dichotomy of Dalvi and Suciu has been lifted from TIDs to *open-world probabilistic databases* [14]. However, we are not aware of dichotomies in the probabilistic database literature that apply to Boolean queries beyond first-order logic, or to queries with fixpoints.

Query evaluation on probabilistic graphs has also been studied in the context of *ontology-mediated queries* (OMQs) [27, 9, 10]. An OMQ is a composite query that typically consists of a UCQ and an *ontology*, i.e., a logical theory on an arity-two signature. The only classification result on PQE for OMQs beyond FO-rewritable languages is given for the description logic $\mathcal{ELI}$ [27]. This result applies to a class of queries that go beyond first-order logic. Our work generalizes this result (Theorem 6 of [27]) by showing hardness for any unbounded UCQ^∞, not just the ones expressible as OMQs based on $\mathcal{ELI}$. Part of our techniques (Section 4) are

related to theirs, but the bulk of our proof (Sections 5 and 6) uses new techniques, the need for which had in fact been overlooked[1] in [26, 27]. Our proof thus completes the proof of Theorem 6 in [27], and generalizes it to all unbounded UCQ$^{\neq}$.

Paper structure. We introduce preliminaries in Section 2, and formally state our result in Section 3. We prove the result in the rest of the paper. We first deal in Section 4 with the case of queries having a model with a non-iterable edge (reducing from #PP2DNF), then argue in Section 5 that unbounded queries must have a model with a minimal tight edge, before explaining in Section 6 how to use this (when the edge is iterable) to reduce from #U-ST-CON. We then conclude in Section 7. Detailed proofs can be found in the full version [3].

2 Preliminaries

Vocabulary. We consider a *relational signature* σ which is a set of *predicates*. In this work, the signature is required to be *arity-two*, i.e., consist *only* of predicates of arity two. Our results can easily be extended to signatures with relations having predicates of arity one and two (see the full version [3]), as is more common in some contexts such as description logics.

A *σ-fact* is an expression of the form $F = R(a, b)$ where R is a predicate and a, b are constants. By a slight abuse of terminology, we call F a *unary* fact if $a = b$, and a *non-unary fact* otherwise. A *σ-atom* is defined in the same way with variables instead of constants. For brevity, we will often talk about a *fact* or an *atom* when σ is clear from context. We also speak of *R-facts* or *R-atoms* to specifically refer to facts or atoms that use the predicate R.

It will be convenient to write $\sigma^{\leftrightarrow}$ the arity-two signature consisting of the relations of σ and of the relations R^- for $R \in \sigma$, with a semantics that we define below.

Database instances. A *database instance over σ*, or a *σ-instance*, is a set of facts over σ. All instances considered in this paper are finite. The *domain* of a fact F, denoted $\mathrm{dom}(F)$, is the set of constants that appear in F, and the *domain* of an instance I, denoted $\mathrm{dom}(I)$, is the set of constants that appear in I, i.e., the union of the domains of its facts.

Every σ-instance I can be seen as a $\sigma^{\leftrightarrow}$-instance consisting of all the σ-facts in I, and all the facts $R^-(b, a)$ for each fact $R(a, b)$ of I. Thus, whenever we consider a σ-instance I, choose some $a \in \mathrm{dom}(I)$, and say, e.g., that we consider all $\sigma^{\leftrightarrow}$-facts of the form $F = R(a, b)$ in I, we mean all unary facts $S(a, a)$ with some $S \in \sigma$, all facts $S(a, b)$ of I with some $S \in \sigma$ and $b \in \mathrm{dom}(I)$, and also all facts $S^-(a, b)$ of I for some $S \in \sigma$ and $b \in \mathrm{dom}(I)$, that is, facts of the form $S(b, a)$. If we say that, for one such fact $F_0 = R(a, b_0)$, we create the fact $R(a', b_0)$ for some $a' \in \mathrm{dom}(I)$, it means that we create $S(a', b_0)$ if $F_0 = S(a, b_0)$ with $S \in \sigma$, and $S(b_0, a')$ if $F_0 = S^-(a, b_0)$ with $S \in \sigma$.

The *Gaifman graph* of an instance I is the undirected graph having $\mathrm{dom}(I)$ as vertex set, and having an edge $\{u, v\}$ between any two $u \neq v$ in $\mathrm{dom}(I)$ that co-occur in some fact of I. An instance is *connected* if its Gaifman graph is connected. We then call $\{u, v\}$ an (undirected) *edge* of I, and the facts that *realize* the undirected edge e are the σ-facts of I whose domain is a subset of $\{u, v\}$. Note that a fact of the form $R(u, u)$ realizes all

[1] Specifically, we identified a gap in the proofs of Theorem 6 of [27] and Theorem 5.31 of [26] concerning a subtle issue of "back-and-forth" matches. We have communicated this with the authors of [26, 27], which they kindly confirmed. The problem is related to the use of inverse roles of $\mathcal{ELI}$, so we believe that it does not occur for the description logic $\mathcal{EL}$.

edges involving u. Slightly abusing notation, we say that an *ordered* pair $e = (u, v)$ is a (directed) *edge* of I if $\{u, v\}$ is an edge of the Gaifman graph. We then talk about the facts that *realize* the directed edge e as the $\sigma^{\leftrightarrow}$-facts of I defined as follows: all unary σ-facts of the form $R(u, u)$ and $R(v, v)$, all σ-facts $S(u, v)$ of I with $S \in \sigma$, and one fact $S^-(u, v)$ for every σ-fact $S(v, u)$ of I with $S \in \sigma$. Note that the facts that *realize* the directed edge (v, u) would correspond to the same σ-facts of I, but they are not the same $\sigma^{\leftrightarrow}$ facts: specifically, each relation $S \in \sigma$ has been exchanged with its reverse relation S^- in non-unary facts.

In the course of our proofs, we will often modify instances in a specific way, which we call *copying* an edge. Let I be an instance, let (u, v) be a directed edge of I, and let u', v' be any elements of I. If we say that we *copy* the edge e on (u', v'), it means that we modify I to add a copy of each fact realizing the edge e, but using u' and v' instead of u and v. Specifically, we create $S(u', v')$ for all σ-facts of the form $S(u, v)$ in I, we create $S(v', u')$ for all σ-facts of the form $S(v, u)$ in I, and we create $S(u', u')$ and $S(v', v')$ for all σ-facts respectively of the form $S(u, u)$ and $S(v, v)$ in I. Of course, if some of these facts already exist, they are not created again. Note that (u', v') is an edge of I after this process.

An instance I is a *subinstance* of another instance I' if $I \subseteq I'$, and I is a *proper subinstance* of I' if $I \subset I'$. Given a set $S \subseteq \mathrm{dom}(I)$ of domain elements, the subinstance of I *induced* by S is the instance formed of all the facts $F \in I$ such that $\mathrm{dom}(F) \subseteq S$.

A *homomorphism* from an instance I to an instance I' is a function h from $\mathrm{dom}(I)$ to $\mathrm{dom}(I')$ such that, for every fact $R(a, b)$ of I, the fact $R(h(a), h(b))$ is a fact of I'. In particular, whenever $I \subseteq I'$ then I has a homomorphism to I'. An *isomorphism* is a bijective homomorphism whose inverse is also a homomorphism.

Query languages. Throughout this work, we focus on Boolean queries. A (Boolean) *query* over a signature σ is a function from σ-instances to Booleans. An instance I *satisfies* a query Q (or Q *holds* on I, or I is a *model* of Q), written $I \models Q$, if Q returns true when applied to I; otherwise, I *violates* Q. We say that two queries Q_1 and Q_2 are *equivalent* if for any instance I, we have $I \models Q_1$ iff $I \models Q_2$. In this work, we study the class UCQ^∞ of queries that are *closed under homomorphisms* (also called *homomorphism-closed*), i.e., if I satisfies the query and I has a homomorphism to I' then I' also satisfies the query. Note that queries closed under homomorphisms are in particular *monotone*, i.e., if I satisfies the query and $I \subseteq I'$, then I' also satisfies the query.

One well-known subclass of UCQ^∞ is *bounded* UCQ^∞: every bounded query in UCQ^∞ is logically equivalent to a *union of conjunctive query* (UCQ), without negation or inequalities. Recall that a *conjunctive query* (CQ) is an existentially quantified conjunctions of atoms, and a UCQ is a disjunction of CQs. For brevity, we omit existential quantification when writing UCQs, and abbreviate conjunction with a comma. The other UCQ^∞ queries are called *unbounded*, and they can be seen as an infinite disjunction of CQs, with each disjunct corresponding to a model of the query.

A natural language captured by UCQ^∞ is *Datalog*, again without negation or inequalities. A Datalog program defines a signature of *intensional predicates*, including a 0-ary predicate Goal(), and consists of a set of *rules* which explain how intensional facts can be *derived* from other intensional facts and from facts of the instance (called *extensional*). The interpretation of the intensional predicates is defined by taking the (unique) least fixpoint of applying the rules, and the query holds iff the Goal() predicate can be derived. For formal definitions of this semantics, see, e.g., [1]. As Datalog queries are homomorphism-closed, we can see each Datalog program as a UCQ^∞, with the disjuncts intuitively corresponding to *derivation trees* for the program. However, note that there are homomorphism-closed queries that are not expressible in Datalog [18].

Ontology-mediated queries or OMQs [8] are another subclass of UCQ$^\infty$. An OMQ is a pair $(Q, \mathcal{T})$, where Q is a UCQ, and $\mathcal{T}$ is an ontology. A database instance I *satisfies* an OMQ $(Q, \mathcal{T})$ if the instance I and the logical theory $\mathcal{T}$ entail the query Q in the standard sense – see, e.g., [8], for details. There are ontological languages for OMQs based on description logics [5] and on existential rules [11, 12]. Many such OMQs can be equivalently expressed as a query in Datalog or in disjunctive Datalog on an arity-two signature [8, 20, 23], thus falling in the class UCQ$^\infty$. In particular, this is the case of any OMQ involving negation-free $\mathcal{ALCHI}$ (Theorem 6 of [8]), and of fragments of $\mathcal{ALCHI}$, e.g., $\mathcal{ELHI}$, and $\mathcal{ELI}$ as in [27].

Probabilistic query evaluation. We study the problem of probabilistic query evaluation over tuple-independent probabilistic databases. A *tuple-independent probabilistic database (TID)* over a signature σ is a pair $\mathcal{I} = (I, \pi)$ of a σ-instance I, and of a function π that maps every fact F to a probability $\pi(F)$, given as a rational number in $[0, 1]$. Formally, a TID $\mathcal{I} = (I, \pi)$ defines the following probability distribution over all *possible worlds* $I' \subseteq I$:

$$\pi(I') := \left(\prod_{F \in I'} \pi(F)\right) \times \left(\prod_{F \in I' \setminus I} (1 - \pi(F))\right).$$

Then, given a TID $\mathcal{I} = (I, \pi)$, the probability of a query Q relative to $\mathcal{I}$, denoted $\mathrm{P}_{\mathcal{I}}(Q)$, is given by the sum of the probabilities of the possible worlds that satisfy the query:

$$\mathrm{P}_{\mathcal{I}}(Q) := \sum_{I' \subseteq I, I' \models Q} \pi(I').$$

The *probabilistic query evaluation problem* (PQE) for a query Q, written PQE(Q), is then the task of computing $\mathrm{P}_{\mathcal{I}}(Q)$ given a TID $\mathcal{I}$ as input.

Complexity background. FP is the class of functions $f : \{0,1\}^* \mapsto \{0,1\}^*$ computable by a polynomial-time deterministic Turing machine. The class #P, introduced by Valiant [34], contains the computation problems that can be expressed as the number of accepting paths of a nondeterministic polynomial-time Turing machine. Equivalently, a function $f : \{0,1\}^* \mapsto \mathbb{N}$ is in #P if there exists a polynomial $p : \mathbb{N} \mapsto \mathbb{N}$ and a polynomial-time deterministic Turing machine M such that for every $x \in \{0,1\}^*$, it holds that $f(x) = |\{y \in \{0,1\}^{p(|x|)} \mid M \text{ answers 1 on the input } (x, y)\}|$.

For a query Q, we study the *data complexity* of PQE(Q), which is measured as a function of the input instance I, i.e., the signature and Q are fixed. For a large class of queries, in particular for any UCQ Q, the problem PQE(Q) is in the complexity class FP$^{\#P}$: we can use a nondeterministic Turing machine to guess a possible world according to the probability distribution of the TID (i.e., each possible world is obtained in a number of runs proportional to its probability), and then check in polynomial time data complexity if Q holds, with polynomial-time postprocessing to renormalize the number of runs to a probability. Our goal in this work is to show that the problem is also #P-hard.

To show #P-hardness, we use *polynomial-time Turing reductions* [15]. A function f is #P-complete under polynomial time Turing reductions if it is in #P and every $g \in$ #P is in FP^f. Polynomial-time Turing reductions are the most common reductions for the class #P and they are the reductions used to show #P-hardness in the dichotomy of Dalvi and Suciu [17], so we use them throughout this work.

Problems. We will show hardness by reducing from two well-known #P-hard problems. For some queries, we reduce from #PP2DNF [31], which is a standard tool to show hardness of unsafe UCQs. The original problem uses Boolean formulas; here, we give an equivalent rephrasing in terms of bipartite graphs.

▶ **Definition 2.1.** *Given a bipartite graph $H = (A, B, C)$ with edges $C \subseteq A \times B$, a* possible world *of H is a pair $\omega = (A', B')$ with $A' \subseteq A$ and $B' \subseteq B$. We call the possible world* good *if it is not an independent set, i.e., if one vertex of A' and one vertex of B' are adjacent in C; and call it* bad *otherwise. The* positive partitioned 2DNF problem (#PP2DNF) *is the following: Given a bipartite graph, compute how many of its possible worlds are good.*

It will be technically convenient to assume that H is connected. This is clearly without loss of generality, as otherwise the number of good possible worlds is simply obtained as the product of the number of good possible worlds of each connected component of H.

For other queries, we reduce from the *undirected st-connectivity problem* (#U-ST-CON) [31]:

▶ **Definition 2.2.** *The* source-to-target undirected reachability problem (#U-ST-CON) *asks the following: Given an undirected graph G with two distinguished vertices s and t, where each graph edge has probability* 0.5*, determine the probability of obtaining a* good *possible world, i.e., a subgraph ω of G where there is a path from s to t.*

3 Result Statement

The goal of this paper is to extend the dichotomy of Dalvi and Suciu [17] on PQE for UCQs. Their result states:

▶ **Theorem 3.1** ([17])**.** *Let Q be a UCQ. Then,* PQE(Q) *is either in FP or it is #P-hard.*

We call a UCQ *safe* if PQE(Q) is in FP, and *unsafe* otherwise. This dichotomy characterizes the complexity of PQE for UCQs, but does not apply to other homomorphism-closed queries beyond UCQs. Our contribution, when restricting to the arity-two setting, is to generalize this dichotomy to UCQ$^\infty$, i.e., to *any* query closed under homomorphisms. Specifically, we show that all such queries are intractable unless they are equivalent to a safe UCQ.

▶ **Theorem 3.2.** *Let Q be a* UCQ$^\infty$ *on an arity-two signature. Then, either Q is equivalent to a safe UCQ and* PQE(Q) *is in FP, or it is not and* PQE(Q) *is #P-hard.*

Our result relies on the dichotomy of Dalvi and Suciu for UCQ$^\infty$ queries that are equivalent to UCQs. The key point is then to show intractability for *unbounded* UCQ$^\infty$ queries. Hence, our technical contribution is to show:

▶ **Theorem 3.3.** *Let Q be an unbounded* UCQ$^\infty$ *query on an arity-two signature. Then* PQE(Q) *is #P-hard.*

Examples of unbounded UCQ$^\infty$ queries include many Datalog queries, e.g., the following program with one monadic intensional predicate U on extensional signature R, S, T:

$$R(x, y) \to U(x) \qquad U(x), S(x, y) \to U(y) \qquad U(x), T(x, y) \to \text{Goal}()$$

Thus, our result implies that the PQE problem is #P-hard for all Datalog queries that are not equivalent to a UCQ, which is the case unless the Datalog program is nonrecursive or recursion is *bounded* [24]. Unbounded UCQ$^\infty$ queries also include many regular path queries, such as RS^*T which is equivalent to the Datalog program above.

Effectiveness and uniformity. We do not study if our dichotomy result in Theorem 3.2 is effective, i.e., given a query, the problem of determining whether it is safe or unsafe. The dichotomy of Theorem 3.1 on UCQs is effective via the algorithm of [17]: this algorithm has a super-exponential bound (in the query), with the precise complexity being open. Our dichotomy concerns the very general query language UCQ^∞, and its effectiveness depends on how the input is represented, which we can fix by restricting to a syntactically defined fragment. If we restrict to Datalog queries, it is not clear whether our dichotomy is effective, because it is generally undecidable whether an input Datalog program is bounded [22] – but this, on its own, does not imply undecidability for our dichotomy. However, our dichotomy is effective for more restricted query languages for which boundedness is decidable, e.g., monadic Datalog or its generalization GN-Datalog [7], or C2RPQs [6].

For unsafe queries, we also do not study the complexity of reduction *as a function of the query*, or whether this problem is even decidable. All that matters is that, once the query is fixed, some reduction procedure exists, which can be performed in polynomial time *in the input instance*. Such uniformity problems seem unavoidable, given that our language UCQ^∞ is very general and includes some queries for which non-probabilistic evaluation is not even decidable, e.g., "there is a path from R to T whose length is the index of a Turing machine that halts". We leave for future work the study of the query complexity of our reduction when restricting to better-behaved query languages such as Datalog or RPQs.

Proof outline. Theorem 3.3 is proven in the rest of the paper. There are two cases, depending on the query. We study the first case in Section 4, which covers queries for which we can find a model with a so-called *non-iterable edge*. Intuitively, this is a model where we can make the query false by replacing the edge by a back-and-forth path of some length between two neighboring facts that it connects. For such queries, we can show hardness by a reduction from #PP2DNF, essentially like the hardness proof for the query $Q_0 : R(w, x), S(x, y), T(y, z)$ which is the arity-two variant of the unsafe query of [16, Theorem 5.1]. This hardness proof covers some bounded queries (including Q_0) and some unbounded ones.

In Section 5, we present a new ingredient, to be used in the second case, i.e., when there is no model with a non-iterable edge. We show that any unbounded query must always have a model with a *tight edge*, i.e., an edge where we can make the query false by replacing it by two copies that disconnect its endpoints. What is more, we can find a model with a tight edge which is *minimal* in some sense, which we call a *minimal tight pattern*.

In Section 6, we use minimal tight patterns for the second case, covering unbounded queries that have a minimal tight pattern whose edge is iterable. This applies for all queries to which Section 4 did not apply (and also for some queries to which it did). Here, we reduce from the #U-ST-CON problem, intuitively using the iterable edge for a kind of reachability test, and using the minimality and tightness of the pattern to show the soundness and completeness of the reduction.

4 Hardness with Non-Iterable Edges

In this section, we present the hardness proof for the first case where we can find a model of the query with a *non-iterable edge*. This notion will be defined relative to an *incident pair* of a *non-leaf edge*:

▶ **Definition 4.1.** *Let I be an instance. We say that an element $u \in \mathrm{dom}(I)$ of I is a* leaf *if it occurs in only one undirected edge. We say that an edge (directed or undirected) is a* leaf edge *if one of its elements (possibly both) is a leaf; otherwise, it is a* non-leaf edge.

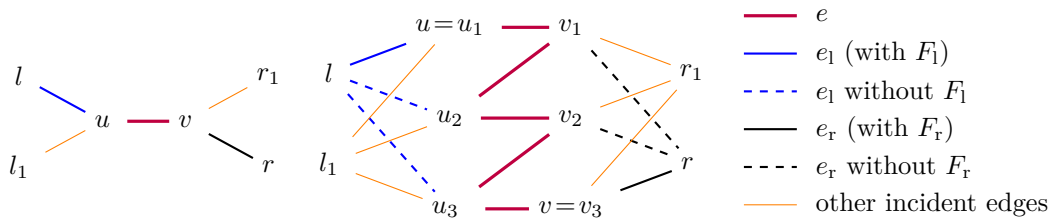


Figure 1 Example of iteration from an instance $I_{e,\Pi}$ (left) to $I^3_{e,\Pi}$ (middle). We write $\Pi = (F_\mathrm{l}, F_\mathrm{r})$ and call e_l and e_r the edges of F_l and F_r. Each line represents an edge, realized in general by multiple $\sigma^{\leftrightarrow}$-facts. A key is given at the right.

Let I be an instance and let $e = (u, v)$ be a non-leaf edge of I. A $\sigma^{\leftrightarrow}$-fact of I is left-incident *to e if it is of the form $R_\mathrm{l}(l, u)$ with $l \notin \{u, v\}$. It is* right-incident *to e if it is of the form $R_\mathrm{r}(v, r)$ with $r \notin \{u, v\}$. An* incident pair *of e is a pair of $\sigma^{\leftrightarrow}$-facts $\Pi = (F_\mathrm{l}, F_\mathrm{r})$, where F_l is left-incident to e and F_r is right-incident to e. We write $I_{e,\Pi}$ to denote an instance I with a non-leaf edge e and an incident pair Π of e in I.*

Note that an incident pair chooses two incident *facts* (not edges): this is intuitively because in the PQE problem, we will give probabilities to single facts and not edges. It is clear that every non-leaf edge e must have an incident pair, as we can pick F_l and F_r from the edges incident to u and v which are not e. Moreover, we must have $F_\mathrm{l} \neq F_\mathrm{r}$, and neither F_l nor F_r can be unary facts. However, as the relations R_l and R_r are $\sigma^{\leftrightarrow}$-relations, we may have $R_\mathrm{l} = R_\mathrm{r}$ or $R_\mathrm{l} = R_\mathrm{r}^-$, and the elements l and r may be equal if the edge (u, v) is part of a triangle with some edges $\{u, w\}$ and $\{v, w\}$.

Let us illustrate the notion of incident pair on an example.

▶ **Example 4.2.** Given an instance $I = R(a, b), T(b, b), S(c, b), R(d, c)$, the edge (b, c) is non-leaf and the only possible incident pair for it is $(R(a, b), R^-(c, d))$.

We can now define the *iteration process* on an instance $I_{e,\Pi}$, which intuitively replaces the edge e by a path of copies of e, keeping the facts of Π at the beginning and end of the path, and copying all other incident facts:

▶ **Definition 4.3.** *Let $I_{e,\Pi}$ be an instance where $e = (u, v)$, $\Pi = (F_\mathrm{l}, F_\mathrm{r})$, $F_\mathrm{l} = R_\mathrm{l}(l, u)$, $F_\mathrm{r} = R_\mathrm{r}(v, r)$, and let $n \geq 1$. The result of performing the* n-th iteration *of e in I relative to Π, denoted $I^n_{e,\Pi}$, is a σ-instance with domain $\mathrm{dom}(I^n_{e,\Pi}) := \mathrm{dom}(I) \cup \{u_2, \ldots, u_n\} \cup \{v_1, \ldots, v_{n-1}\}$, where the new elements are fresh, and where we use u_1 to refer to u and v_n to refer to v for convenience. The facts of $I^n_{e,\Pi}$ are defined by applying the following steps:*

- Copy non-incident facts: *Initialize $I^n_{e,\Pi}$ as the induced subinstance of I on $\mathrm{dom}(I) \setminus \{u, v\}$.*
- Copy incident facts F_l and F_r: *Add F_l and F_r to $I^n_{e,\Pi}$, using u_1 and v_n, respectively.*
- Copy other left-incident facts: *For each $\sigma^{\leftrightarrow}$-fact $F'_\mathrm{l} = R'_\mathrm{l}(l', u)$ of I that is left-incident to e (i.e., $l' \notin \{u, v\}$) and where $F'_\mathrm{l} \neq F_\mathrm{l}$, add to $I^n_{e,\Pi}$ the fact $R'_\mathrm{l}(l', u_i)$ for each $1 \leq i \leq n$.*
- Copy other right-incident facts: *For each $\sigma^{\leftrightarrow}$-fact $F'_\mathrm{r} = R'_\mathrm{r}(v, r')$ of I that is right-incident to e (i.e., $r' \notin \{u, v\}$) and where $F'_\mathrm{r} \neq F_\mathrm{r}$, add to $I^n_{e,\Pi}$ the fact $R'_\mathrm{r}(v_i, r')$ for each $1 \leq i \leq n$.*
- Create copies of e: *Copy the edge e (in the sense defined in the Preliminaries) on the following pairs: (u_i, v_i) for $1 \leq i \leq n$, and (u_{i+1}, v_i) for $1 \leq i \leq n - 1$.*

The iteration process is represented in Figure 1. Note that, for $n = 1$, we obtain exactly the original instance. Intuitively, we replace e by a path going back-and-forth between copies of u and v (and traversing e alternatively in one direction and another). The intermediate

vertices have the same incident facts as the original endpoints except that we have not copied the left-incident fact and the right-incident fact of the incident pair.

We first notice that larger iterates have homomorphisms back to smaller iterates:

▶ **Observation 4.4.** *For any instance I, for any non-leaf edge e of I, for any incident pair Π for e, and for any $1 \leq i \leq j$, it holds that $I^j_{e,\Pi}$ has a homomorphism to $I^i_{e,\Pi}$.*

Proof. Simply merge $u_i, \ldots, u_j$, and merge $v_i, \ldots, v_j$. ◀

Hence, choosing an instance I that satisfies Q, a non-leaf edge e of I, and an incident pair Π, there are two possible regimes. Either all iterations $I^n_{e,\Pi}$ satisfy Q, or there is some iteration $I^{n_0}_{e,\Pi}$ with $n_0 > 1$ that violates Q (and, by Observation 4.4, all subsequent iterations also do). We call e *iterable* relative to Π in the first case, and *non-iterable* in the second case.

▶ **Definition 4.5.** *A non-leaf edge e of a model I of a query Q is* iterable *relative to an incident pair Π if $I^n_{e,\Pi}$ satisfies Q for each $n \geq 1$; otherwise, it is* non-iterable.

The goal of this section is to show that if a query Q has a model with a non-leaf edge which is not iterable, then PQE(Q) is intractable:

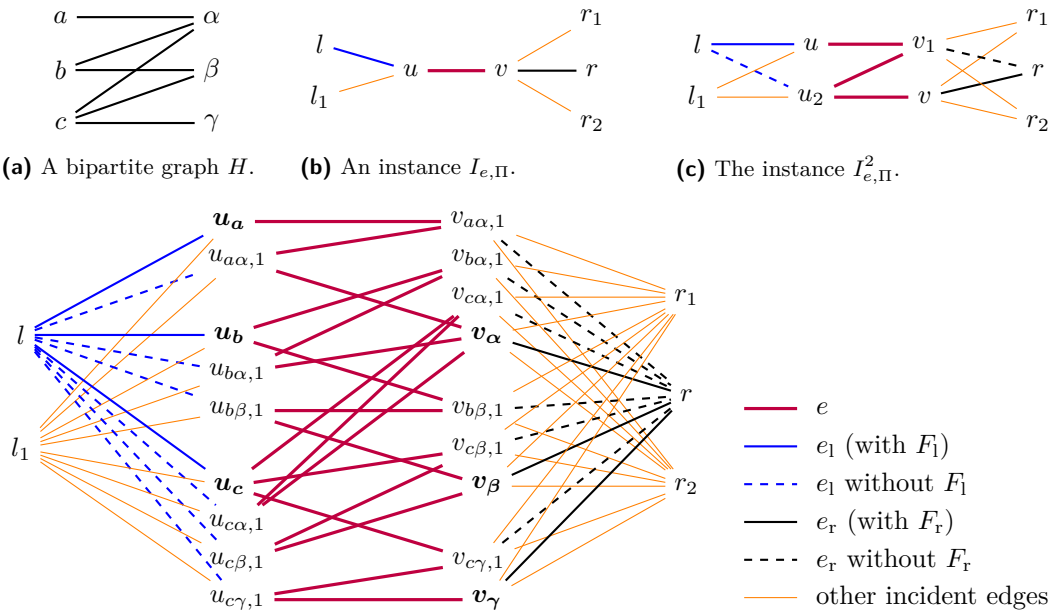
▶ **Theorem 4.6.** *For every* UCQ$^\infty$ *Q, if Q has a model I with a non-leaf edge e that is non-iterable relative to some incident pair, then* PQE(Q) *is #P-hard.*

Note that this result applies to arbitrary homomorphism-closed queries, whether they are bounded or not. Recall for instance the unsafe CQ $Q_0 : R(w,x), S(x,y), T(y,z)$. Then, the model $R(a,b), S(b,c), T(c,d)$ has an edge (b,c) which is non-leaf and non-iterable: indeed its iteration with $n = 2$ relative to the only possible incident pair yields $R(a,b), S(b,c'), S(b',c'), S(b',c), T(c,d)$ which does not satisfy the query. Thus, Theorem 4.6 also shows that PQE is #P-hard for this query. Of course, Theorem 4.6 is too coarse to show #P-hardness for all unsafe UCQs; for instance, it does not cover $Q'_0 : R(x,x), S(x,y), T(y,y)$, or $Q_1 : (R(w,x), S(x,y)) \vee (S(x,y), T(y,z))$. Theorem 4.6 will nevertheless be sufficient for our purpose of showing hardness for all *unbounded queries*, as we will do in the next sections.

Hence, in the rest of this section, we prove Theorem 4.6. Let $I_{e,\Pi}$ be the instance with the non-iterable edge, and let us take the smallest $n_0 > 1$ such that $I^{n_0}_{e,\Pi}$ violates the query. The idea is to use $I_{e,\Pi}$ and n_0 to show hardness of PQE by reducing from #PP2DNF (Definition 2.1). Thus, let us explain how we can use $I_{e,\Pi}$ to code a bipartite graph H in polynomial time into a TID $\mathcal{I}$. The definition of this coding does not depend on the query Q, but we will use the properties of $I_{e,\Pi}$ and n_0 to argue that it defines a reduction between #PP2DNF and PQE, i.e., there is a correspondence between the possible worlds of H and the possible worlds of $\mathcal{I}$, such that good possible worlds of H are mapped to possible worlds of $\mathcal{I}$ which satisfy Q. Let us first define the coding:

▶ **Definition 4.7.** *Let $I_{e,\Pi}$ be an instance where $e = (u,v)$, $\Pi = (F_{\mathrm{l}}, F_{\mathrm{r}})$, $F_{\mathrm{l}} = R_{\mathrm{l}}(l,u)$, $F_{\mathrm{r}} = R_{\mathrm{r}}(v,r)$, and let $n \geq 1$. Let $H = (A,B,C)$ be a connected bipartite graph. The* coding *of H relative to $I_{e,\Pi}$ and n is a TID $\mathcal{I} = (J,\pi)$ with domain* $\mathrm{dom}(J) := (\mathrm{dom}(I) \setminus \{u,v\}) \cup \{u_a \mid a \in A\} \cup \{v_b \mid b \in B\} \cup \{u_{c,2}, \ldots, u_{c,n} \mid c \in C\} \cup \{v_{c,1}, \ldots, v_{c,n-1} \mid c \in C\}$*, where the new elements are fresh. The facts of J and the probability mapping π are defined as follows:*

- Copy non-incident facts: *Initialize J as the induced subinstance of I on* $\mathrm{dom}(I) \setminus \{u,v\}$.
- Copy incident facts F_{l} and F_{r}: *Add to J the $\sigma^{\leftrightarrow}$-fact $R_{\mathrm{l}}(l,u_a)$ for each $a \in A$, and add to J the $\sigma^{\leftrightarrow}$-fact $R_{\mathrm{r}}(v_b,r)$ for each $b \in B$.*
- Copy other left-incident facts: *For each $\sigma^{\leftrightarrow}$-fact $F'_{\mathrm{l}} = R'_{\mathrm{l}}(l',u)$ of I that is left-incident to e (i.e., $l' \notin \{u,v\}$) and where $F'_{\mathrm{l}} \neq F_{\mathrm{l}}$, add to J the facts $R'_{\mathrm{l}}(l',u_a)$ for each $a \in A$, and add to J the facts $R'_{\mathrm{l}}(l',u_{c,j})$ for each $2 \leq j \leq n$ and $c \in C$.*



(a) A bipartite graph H. **(b)** An instance $I_{e,\Pi}$. **(c)** The instance $I^2_{e,\Pi}$.

(d) Coding of the bipartite graph H relative to $I^2_{e,\Pi}$. Bold elements correspond to vertices of H.

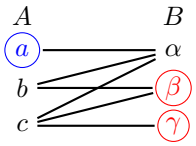
Figure 2 Example of the coding of a bipartite graph H shown in Figure 2a. We encode H relative to an instance $I_{e,\Pi}$ (Figure 2b), with a non-leaf edge e and an incident pair Π. The result $I^2_{e,\Pi}$ of iterating e in I with $n = 2$ (Definition 4.3) is shown in Figure 2c. The coding of H relative to $I_{e,\Pi}$ and $n = 2$ (Definition 4.7) is shown in Figure 2d, with the probabilistic facts being the copies of F_l and F_r in the edges in solid blue and black. A key explains the colors (bottom right).

- Copy other right-incident facts: *For each $\sigma^{\leftrightarrow}$-fact $F'_\mathrm{r} = R'_\mathrm{r}(v, r')$ of I that is right-incident to e (i.e., $r' \notin \{u, v\}$) and where $F'_\mathrm{r} \neq F_\mathrm{r}$, add to J the facts $R'_\mathrm{r}(v_b, r')$ for each $b \in B$ and add to J the facts $R'_\mathrm{r}(v_{c,j}, r')$ for each $1 \leq j \leq n-1$ and $c \in C$.*
- Create copies of e: *For each $c \in C$ with $c = (a, b)$, copy e on the following pairs: $(u_{c,i}, v_{c,i})$ for $1 \leq i \leq n$, and $(u_{c,i+1}, v_{c,i})$ for $1 \leq i \leq n-1$, where we use $u_{c,1}$ to refer to u_a and $v_{c,n}$ to refer to v_b.*

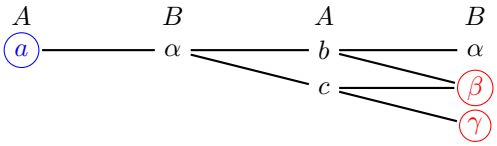
Finally, we define the function π such that it maps all the facts created in the step "Copy incident facts F_l and F_r" to 0.5, and all other facts to 1.

Observe how this definition relates to the definition of iteration (Definition 4.3): we intuitively code each edge of the bipartite graph as a copy of the path of copies of e in the definition of the n-th iteration of (u, v). Note also that there are exactly $|A| + |B|$ uncertain facts, by construction. It is clear that, for any choice of $I_{e,\Pi}$ and n, this coding is in polynomial time in H. The result of the coding is illustrated in Figure 2.

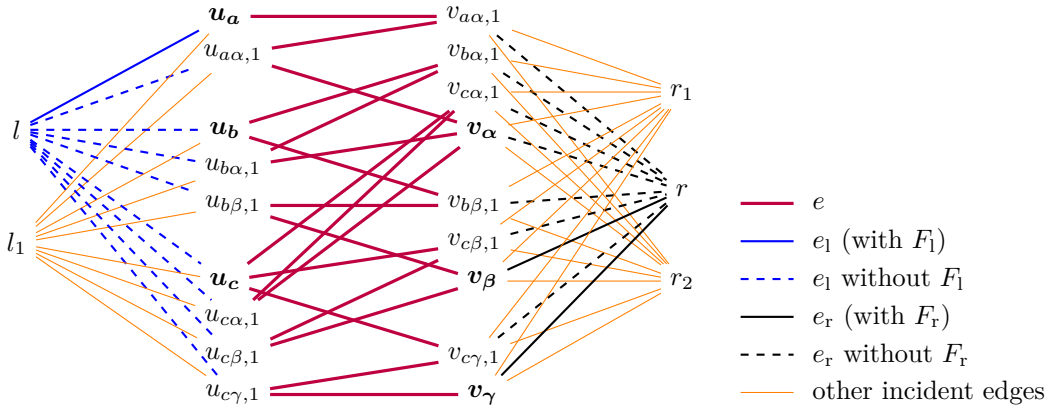
We now define the bijection ϕ, mapping each possible world ω of the connected bipartite graph H to a possible world of the TID $\mathcal{I}$. For each vertex $a \in A$, we keep the copy of F_r incident to u_a in $\phi(\omega)$ if a is kept in ω, and we do not keep it otherwise; we do the same for v_b, and F_l. It is obvious that this correspondence is bijective, and that all possible worlds have the same probability, namely, $0.5^{|A|+|B|}$. Furthermore, we can use ϕ to define a reduction, thanks to the following property:



(a) A possible world ω of H from Figure 2a, containing all circled nodes.



(b) The way H is considered in the completeness proof of Proposition 4.8.



(c) The possible world $\phi(\omega)$ of the coding (Figure 2d) for ω. The edges (l, u_b), (l, u_c), and (v_α, r) are changed to dashed lines, as they correspond to vertices of H that are not kept in ω.

Figure 3 Example for the completeness direction of the proof of Proposition 4.8. Figure 3a shows a bad possible world ω of the bipartite graph. The corresponding possible world of the coding of Figure 2d (using the instance $I^2_{e,\Pi}$ of Figure 2b) is given in Figure 3c. In the proof, we explore H as depicted in Figure 3b to argue that Figure 3c has a homomorphism to $I^5_{e,\Pi}$. A key explains the colors (bottom right).

▶ **Proposition 4.8.** *Let the TID $\mathcal{I} = (J, \pi)$ be the coding of a connected bipartite graph $H = (A, B, C)$ relative to an instance $I_{e,\Pi}$ and to $n \geq 1$ as described in Definition 4.7, and let ϕ be the bijective function defined above from the possible worlds of H to those of $\mathcal{I}$. Then:*

1. *For any* good *possible world ω of H, $\phi(\omega)$ has a homomorphism* from $I^n_{e,\Pi}$.
2. *For any* bad *possible world ω of H, $\phi(\omega)$ has a homomorphism* to $I^{3n-1}_{e,\Pi}$.

Proof sketch. The first direction is because $\phi(\omega)$ then contains a subinstance isomorphic to $I^n_{e,\Pi}$: keep the facts of the path corresponding to any edge witnessing that ω is good.

The harder part is the second direction as illustrated in Figure 3: when ω is bad, we can show how to "fold back" $\phi(\omega)$, going from the copies of F_l to the copies of F_r, into the iterate $I^{3n-1}_{e,\Pi}$. This uses the fact that ω is bad, so the copies of F_l and F_r must be sufficiently far from one another. ◀

Proposition 4.8 allows us to conclude the proof of Theorem 4.6. Indeed, we can take $I_{e,\Pi}$ which satisfies Q, and choose the smallest $n_0 > 1$ such that $I^{n_0}_{e,\Pi}$ violates Q. Hence, $I^{n_0-1}_{e,\Pi}$ satisfies Q, but $I^{3(n_0-1)-1}_{e,\Pi}$ does not. Then, by Proposition 4.8, good possible worlds of H give a possible world of $\mathcal{I}$ that satisfies Q, and bad possible worlds of H give a possible world of $\mathcal{I}$ that does not satisfy Q. This argument concludes the proof of Theorem 4.6.

5 Finding a Minimal Tight Pattern

In the previous section, we have shown hardness for queries (bounded or unbounded) that have a model with a non-iterable edge; leaving the case of unbounded queries open, for which, in all models, all non-leaf edges can be iterated. We first note that such queries indeed exist:

▶ **Example 5.1.** Consider the following Datalog program:

$$R(x,y) \to A(y), \quad A(x), S(x,y) \to B(y), \quad B(x), S(y,x) \to A(y), \quad T(x,y), B(x) \to \mathrm{Goal}().$$

This program is unbounded, as it tests if the instance contains a path of the form $R(a, a_1)$, $S(a_1, a_2), S^-(a_2, a_3), \ldots, S(a_{2n+1}, a_{2n+2}), T(a_{2n+2}, b)$. However, it has no model with a non-iterable edge: in every model, the query is satisfied by a path of the form above, and we cannot break such a path by iterating an edge (i.e., this yields a longer path of the same form).

If we tried to reduce from #PP2DNF for this query as in the previous section, then the reduction would fail because the edge is iterable: in possible worlds of the bipartite graph, where we have not retained two adjacent vertices, we would still have matches of the query in the corresponding possible world of the probabilistic instance, where we go from a chosen vertex to another by going *back-and-forth* on the copies of e that code the edges of the bipartite graph. These are the "back-and-forth matches" which were missed in [26, 27].

In light of this, we handle the case of such queries in the next two sections. In this section, we prove a general result for unbounded queries (independent from the previous section): all unbounded queries must have a model with a *tight edge*, which is additionally *minimal* in some sense. Tight edges and iterable edges will then be used in Section 6 to show hardness for unbounded queries which are not covered by the previous section.

Let us start by defining this notion of *tight edge*, via a rewriting operation on instances called a *dissociation*.

▶ **Definition 5.2.** *The* dissociation *of a non-leaf edge $e = (u, v)$ in I is the instance I' where:*

- $\mathrm{dom}(I') = \mathrm{dom}(I) \cup \{u', v'\}$ *where u' and v' are fresh.*
- *I' is I where we create a copy of the edge e on (u, v') and on (u', v), and then remove all non-unary facts that realize e in I'.*

Dissociation is illustrated in the following example (see also Figure 4).

▶ **Example 5.3.** Consider the instance $I = \{R(a,b), S(b,a), T(b,a), R(a,c), S(c,b), S(d,b), U(a,a), U(b,b)\}$. The edge (a,b) is non-leaf, as witnessed by the edges $\{a,c\}$ and $\{b,c\}$. The result of the dissociation is $I' = \{R(a,b'), S(b',a), T(b',a), R(a',b), S(b,a'), T(b,a'), R(a,c), S(c,b), S(d,b), U(a,a), U(a',a'), U(b,b), U(b',b')\}$.

We then call an edge *tight* in a model of Q if dissociating it makes Q false.

▶ **Definition 5.4.** *Let Q be a query and I be a model of Q. An edge e of I is* tight *if it is non-leaf, and the result of the dissociation of e in I does not satisfy Q. A* tight pattern *for the query Q is a pair (I, e) of a model I of Q and of an edge e of I that is tight.*

Intuitively, a tight pattern is a model of a query containing at least three edges $\{u,a\}, \{a,b\}, \{b,v\}$ (possibly $u = v$) such that performing a dissociation makes the query false. For instance, for the unsafe CQ $Q_0 : R(w,x), S(x,y), T(y,z)$ from [16], a tight pattern would be $R(a,b), S(b,c), T(c,d)$ with the edge (b,c). Again, not all unsafe CQs have a tight pattern, e.g., Q_0' and Q_1 from Section 4 do not.

For our purposes, we will not only need tight patterns, but *minimal tight patterns*:

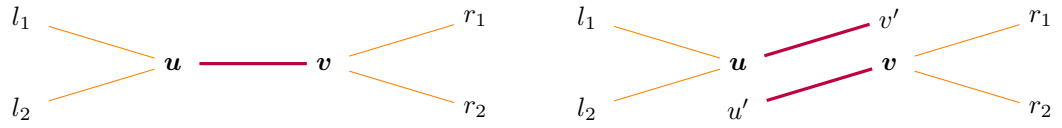


Figure 4 An instance (left) with a non-leaf edge (u, v), and the result (right) of dissociating (u, v).

▶ **Definition 5.5.** *Given an instance I with a non-leaf edge $e = (a, b)$, the* weight *of e is the number of facts that realize e in I (including unary facts). The* side weight *of e is the number of $\sigma^{\leftrightarrow}$-facts in I that are left-incident to e, plus the number of $\sigma^{\leftrightarrow}$-facts in I that are right-incident to e. Given a query Q, we say that a tight pattern (I, e) is* minimal *if:*

- *Q has no tight pattern (I', e') where the weight of e' is strictly less than that of e; and*
- *Q has no tight pattern (I', e') where the weight of e' is equal to that of e and the side weight of e' is strictly less than that of e.*

We can now state the main result of this section:

▶ **Theorem 5.6.** *Every unbounded query Q has a minimal tight pattern.*

The idea of how to find tight patterns is as follows. We first note that the only instances without non-leaf edges are intuitively disjoint unions of star-shaped subinstances. Now, if a query is unbounded, then its validity cannot be determined simply by looking at such subinstances (unlike Q'_0 or Q_1 above), so there must be a model of the query with an edge that we cannot dissociate without breaking the query, i.e., a tight pattern. Once we know that there is a tight pattern, then it is simple to argue that we can find a model with a tight edge that is minimal in the sense that we require.

To formalize this intuition, let us first note that any *iterative dissociation process*, i.e., any process of iteratively applying dissociation to a given instance, will necessarily terminate. More precisely, an *iterative dissociation process* is a sequence of instances starting at an instance I and where each instance is defined from the previous one by performing the dissociation of some non-leaf edge. We say that the process *terminates* if it reaches an instance, where there is no edge left to dissociate, i.e., all edges are leaf edges.

▶ **Observation 5.7.** *For any instance I, the iterative dissociation process will terminate in n steps, where n is the number of non-leaf edges in I.*

Proof sketch. Each dissociation decreases the number of non-leaf edges by 1. ◀

Let us now consider instances with no non-leaf edges. They are intuitively disjoint unions of star-shaped subinstances, and in particular they homomorphically map to some constant-sized subset of their facts, as will be crucial when studying our unbounded query.

▶ **Proposition 5.8.** *For every signature σ, there exists a bound $k_\sigma > 0$, ensuring the following: For every instance I on σ having no non-leaf edge, there exists an instance $I' \subseteq I$ such that I has a homomorphism to I' and such that we have $|I'| < k_\sigma$.*

Proof sketch. Connected instances where we cannot perform a dissociation can have at most one non-leaf element, with all edges using this element and a leaf. Now, each edge can be described by the set of facts that realize it, for which there are finitely many possibilities (exponentially many in the signature size). We can thus show the result by collapsing together edges having the same set of facts.

Disconnected instances where we cannot perform a dissociation are unions of the connected instances of the form above, so the number of possibilities up to homomorphic equivalence is finite (exponential in the number of possible connected instances). We can then conclude by collapsing together connected components that are isomorphic. ◀

We can now prove Theorem 5.6 by appealing to the unboundedness of the query. To do this, we will rephrase unboundedness in terms of *minimal models*:

▶ **Definition 5.9.** *A* minimal model *of a query Q is an instance I that satisfies Q and such that every proper subinstance of I violates Q.*

We can rephrase the unboundedness of a UCQ$^\infty$ Q in terms of minimal models: Q is unbounded iff it has infinitely many minimal models. Indeed, if a query Q has finitely many minimal models, then it is clearly equivalent to the UCQ formed from these minimal models, because it is closed under homomorphisms. Conversely, if Q is equivalent to a UCQ, then it has finitely many minimal models which are obtained as homomorphic images of the UCQ disjuncts. Thus, we can clearly rephrase unboundedness as follows:

▶ **Observation 5.10.** *A* UCQ$^\infty$ *query Q is unbounded iff it has a minimal model I with $> k$ facts for any $k \in \mathbb{N}$.*

We can now show Theorem 5.6. We first show how to find a tight pattern, which is not necessarily minimal. To do this, take a sufficiently large minimal model I_0 of the query by Observation 5.10, and perform an iterative dissociation process, while it is possible to dissociate edges without breaking the query. By Observation 5.7, this process eventually terminates. If the result I_n of the process has a non-leaf edge which we did not dissociate, then dissociating this edge breaks the query, so it is tight and we are done. Otherwise, we reach a contradiction: as there are only leaf edges in I_n, Proposition 5.8 implies that I_n has a homomorphism to a constant-sized subset I'_n, which also satisfies Q. Now, I'_n has a homomorphism back into I_n (as a subset), then into I_0 (by undoing the dissociations). This identifies a constant-sized subset of I_0 that satisfies the query, which contradicts the definition of I_0 as a large minimal model.

Having found a tight pattern, we find a minimal tight pattern simply by minimizing first on the weight, then on the side weight, which concludes the proof of Theorem 5.6.

6 Hardness with Tight Iterable Edges

In this section, we conclude the proof of Theorem 3.3 by showing that a minimal tight pattern can be used to show hardness when it is iterable. Formally:

▶ **Theorem 6.1.** *For every query Q, if we have a minimal tight pattern (I, e) where the edge e is iterable, then* PQE(Q) *is #P-hard.*

This covers all the queries to which Section 4 did not apply, and concludes the proof of Theorem 3.3:

Proof of Theorem 3.3. Let Q be an unbounded UCQ$^\infty$. If we have a model of Q with a non-iterable edge, then we conclude by Theorem 4.6 that PQE(Q) is #P-hard. Otherwise, by Theorem 5.6, we have a minimal tight pattern, and its edge is then iterable (otherwise the first case would have applied), so that we can apply Theorem 6.1. ◀

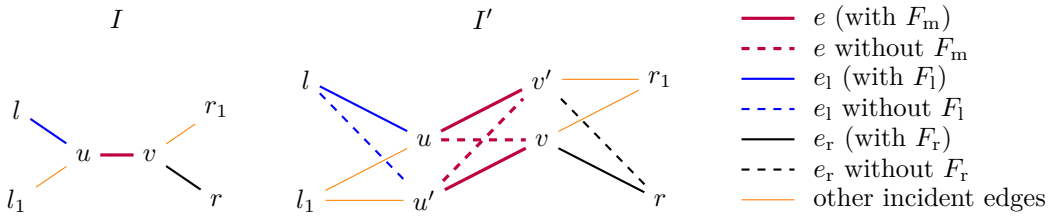


Figure 5 Example of fine dissociation from an instance I (left) to I' (middle) for a choice of e, of $\Pi = (F_\mathrm{l}, F_\mathrm{r})$, and of F_m. We call e_l and e_r the edges of F_l and F_r. A key is given at the right.

Thus, it only remains to show Theorem 6.1. The idea is to use the iterable edge e of the minimal tight pattern (I, e) for some incident pair Π to reduce from the undirected st-connectivity problem #U-ST-CON (Definition 2.2). Given an input st-graph G for #U-ST-CON, we will code it as a TID $\mathcal{I}$ built using $I_{e,\Pi}$, with one probabilistic fact per edge of G. To show a reduction, we will argue that good possible worlds of G correspond to possible worlds J' of $\mathcal{I}$ containing some iterate of the instance $I^n_{e,\Pi}$ (with n being the length of the path), and J' then satisfies Q because e is iterable. Conversely, we will argue that bad possible worlds of G correspond to possible worlds J' of $\mathcal{I}$ that have a homomorphism to a so-called *fine dissociation* of e in I, and we will argue that this violates Q query thanks to our choice of (I, e) as a minimal tight pattern. Let us first define this notion of *fine dissociation*:

▶ **Definition 6.2.** *Let I be an instance, let $e = (u, v)$ be a non-leaf edge in I, let $F_\mathrm{l} = R_\mathrm{l}(l, u)$ and $F_\mathrm{r} = R_\mathrm{r}(v, r)$ be an incident pair of e in I, and let F_m be a non-unary fact realizing the edge e. The result of performing the* fine dissociation *of e in I relative to $F_\mathrm{l}, F_\mathrm{r}$ and F_m is an instance I' on the domain $\mathrm{dom}(I') = \mathrm{dom}(I) \cup \{u', v'\}$, where the new elements are fresh. It is obtained by applying the following steps:*

- Copy non-incident facts: *Initialize I' as the induced subinstance of I on $\mathrm{dom}(I) \setminus \{u, v\}$.*
- Copy incident facts F_l and F_r: *Add the facts F_l and F_r to I'.*
- Copy other left-incident facts: *For every $\sigma^{\leftrightarrow}$-fact $F'_\mathrm{l} = R'_\mathrm{l}(l', u)$ of I that is left-incident to e (i.e., $l' \notin \{u, v\}$) and where $F'_\mathrm{l} \neq F_\mathrm{l}$, add to I' the fact $R'_\mathrm{l}(l', u')$.*
- Copy other right-incident facts: *For every $\sigma^{\leftrightarrow}$-fact $F'_\mathrm{r} = R'_\mathrm{r}(v, r')$ of I that is right-incident to e (i.e., $r' \notin \{u, v\}$) and where $F'_\mathrm{r} \neq F_\mathrm{r}$, add to I' the fact $R'_\mathrm{r}(v', r')$.*
- Create the copies of e: *Copy e on the pairs (u, v') and (u', v) of I', and copy e* except the fact *F_m on the pairs (u, v) and (u', v') of I'.*

The result of a *fine dissociation* is illustrated in Figure 5. If the only non-unary fact realizing the edge e in I is F_m, then (u, v) and (u', v') are not edges in the result of the fine dissociation; otherwise, they are edges but with a smaller weight than e. Observe that fine dissociation is related both to dissociation (Section 5) and to iteration (Section 4). We will study later when fine dissociation can make the query false.

We can now start the proof of Theorem 6.1 by describing the coding, which depends on our choice of $I_{e,\Pi}$ and of a fact F_m, but does not depend on the query Q. Given an input st-graph, i.e., an undirected graph G with source s and target t, we construct a TID $\mathcal{I}$ whose possible worlds will have a bijection to those of G.

▶ **Definition 6.3.** *Let $I_{e,\Pi}$ be an instance where $e = (u, v)$, $\Pi = (F_\mathrm{l}, F_\mathrm{r})$, $F_\mathrm{l} = R_\mathrm{l}(l, u)$, $F_\mathrm{r} = R_\mathrm{r}(v, r)$ and let F_m be a non-unary fact of I realizing e. Let $G = (W, C, s, t)$ be an undirected graph with source and target. The* coding *of G relative to $I_{e,\Pi}$ and F_m is a TID $\mathcal{I} = (J, \pi)$ with domain $\mathrm{dom}(J) := \mathrm{dom}(I) \cup \{u_c \mid c \in C\} \cup \{v_w \mid w \in W \setminus \{t\}\}$, where the new elements are fresh, and where we use v_t to refer to v for convenience. The facts of J and the probability mapping π are defined as follows:*

- Copy non-incident facts: *Initialize J as the induced subinstance of I on* $\mathrm{dom}(I) \setminus \{u, v\}$.
- Copy incident facts F_{l} and F_{r}: *Add the facts F_{l} and F_{r} to J.*
- Copy other left-incident facts: *For every $\sigma^{\leftrightarrow}$-fact $F'_{\mathrm{l}} = R'_{\mathrm{l}}(l', u)$ of I that is left-incident to e (i.e., $l' \notin \{u, v\}$) and where $F'_{\mathrm{l}} \neq F_{\mathrm{l}}$, add to J the facts $R'_{\mathrm{l}}(l', u_c)$ for each edge $c \in C$.*
- Copy other right-incident facts: *For every $\sigma^{\leftrightarrow}$-fact $F'_{\mathrm{r}} = R'_{\mathrm{r}}(v, r')$ of I that is right-incident to e (i.e., $r' \notin \{u, v\}$) and where $F'_{\mathrm{r}} \neq F_{\mathrm{r}}$, add to J the facts $R'_{\mathrm{r}}(v_w, r')$ for each $w \in W$.*
- Create copies of e: *Copy e on the pair (u, v_s) of J, and for each edge $c = \{a, b\}$ in C, copy e on the pairs (u_c, v_a) and (u_c, v_b) of J.*

Finally, we define the function π as follows. For each edge c of C, π maps the copy of the fact F_{m} in the edge (u_c, v_w) to 0.5*, for an arbitrary choice of $w \in c$. All other facts are mapped to* 1 *by π.*

The coding is exemplified in Figure 6. It is important to note that the edges are coded by paths of length 2. This choice is critical, because the source graph to the reduction is undirected, but the facts on edges are directed; so, intuitively, we symmetrize by having two copies of the edge in opposite directions in order to traverse them in both ways. The choice on how to orient the edges (i.e., the choice of $w \in c$ when defining π) has no impact in how the edges can be traversed when their probabilistic fact is present, but it has an impact when the probabilistic fact is missing. Indeed, this is the reason why fine dissociation includes two copies of e with one missing fact.

It is easy to see that the given coding is in polynomial time in the input G for every choice of $I_{e,\Pi}$ and F_{m}. Let us now define the bijection ϕ, mapping each possible world ω of G to a possible world of the TID $\mathcal{I}$ as follows. For each edge $c \in C$, we keep the probabilistic fact incident to u_c in the instance $\phi(\omega)$ if c is kept in the possible world ω, and we do not keep it, otherwise. It is obvious that this correspondence is bijective and that all possible worlds have the same probability $0.5^{|C|}$. We can now explain why ϕ defines a reduction:

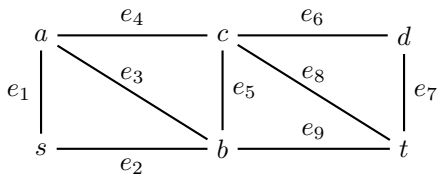
▶ **Proposition 6.4.** *Let the TID $\mathcal{I} = (J, \pi)$ be the coding of an undirected st-graph G relative to an instance $I_{e,\Pi}$ and to F_{m} as described in Definition 6.3. Let ϕ be the bijective function defined above from the possible worlds of G to those of $\mathcal{I}$. Then:*

1. *For any* good *possible world ω of G with a witnessing simple s,t-path traversing n edges, $\phi(\omega)$ has a homomorphism* from $I_{e,\Pi}^{n+1}$.
2. *For any* bad *possible world ω of G, $\phi(\omega)$ has a homomorphism* to *the result of finely dissociating e in I relative to* Π *and F_{m}.*

Proof sketch. For the forward direction, we find $I_{e,\Pi}^{n+1}$ as a subinstance of $\phi(\omega)$ by following the image in J of the witnessing path in ω.

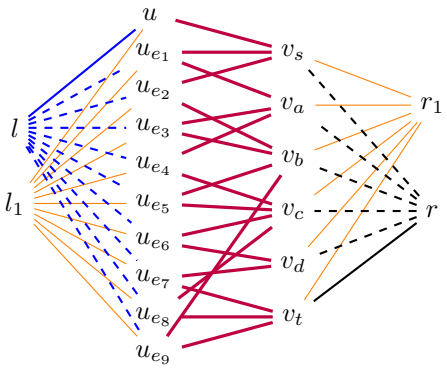
The backward direction is again more challenging. We consider a *cut* in ω between s and t. Then, any two vertices on different sides of the cut can only be connected by two successive copies of e with one of them missing the fact F_{m} (it can be the first or second copy depending on the orientation choice). We then construct the homomorphism to the fine dissociation (Figure 5) by mapping the vertex u to u, mapping vertices on the s-side of the cut (including v_s) to v', mapping the edges between these vertices back-and-forth to $(v'u)$ and (u, v'), mapping all vertices on the t-side (including $v_t = v$) to v, mapping edges between them back-and-forth to (v, u') and (u', v), and mapping the edges across the cut to either (v', u') and (u', v) or to (v', u) and (u, v), depending on the orientation choice. ◀

Proposition 6.4 leads us to a proof of Theorem 6.1: good possible worlds of G give a possible world of $\mathcal{I}$ that satisfies Q thanks to the iterability of e, and bad possible worlds of G give a possible world of $\mathcal{I}$ having a homomorphism to the fine dissociation. The only

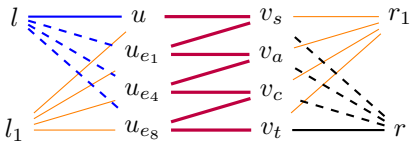


(a) An st-graph G.

(b) An instance $I_{e,\Pi}$.



(c) Coding of the graph G relative to $I_{e,\Pi}$ and some F_{m}.



(d) The image of an s-t path in the coding.

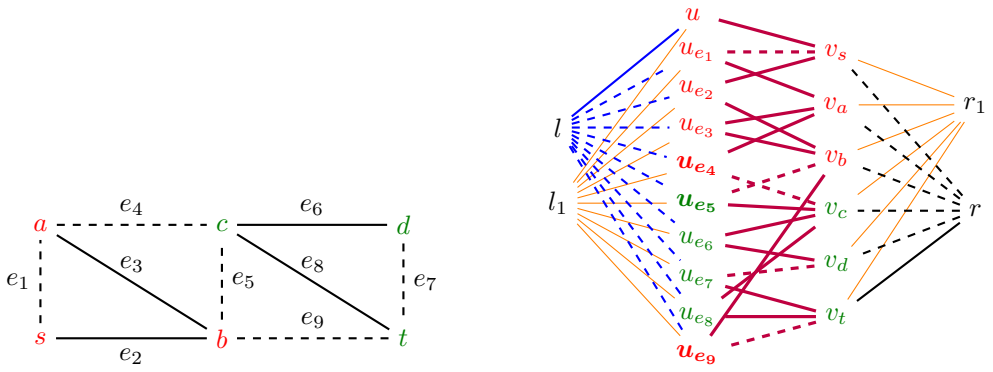
Figure 6 Example of the coding on an st-graph G shown in Figure 6a. We encode G relative to an instance $I_{e,\Pi}$ (Figure 6b) and to some choice of a non-unary fact F_{m} realizing e. The coding of G relative to $I_{e,\Pi}$ and F_{m} is shown in Figure 6c, with the probabilistic facts being *exactly one* copy of F_{m} for *one* of every pair of purple edges adjacent to an element in $\{u_{e_1}, \ldots, u_{e_9}\}$. Each st-path in G gives rise to a subinstance in the coding: consider for instance the st-path which is via the edges e_1, e_4, e_8. The corresponding subinstance in the coding for this path is shown in Figure 6d, which is an iterate of the form $I_{e,\Pi}^{n+1}$, where n is the number of edges on the path, and hence $n = 3$, in this case.

missing piece is to argue that the fine dissociation does not satisfy the query. We can do this using the minimality and tightness of the pattern:

▶ **Lemma 6.5.** *Let Q be a query, let (I, e) be a minimal tight pattern for Q, let Π be an arbitrary incident pair of e in I, and let F_{m} be an arbitrary non-unary fact realizing e in I. Then, the result of the fine dissociation of e in I relative to Π and F_{m} does not satisfy Q.*

Proof sketch. We assume that the fine dissociation I_1 satisfies Q, and show a contradiction by rewriting it in several steps. The process of the proof is illustrated as Figure 8. We first dissociate the copies of e in I_1 with F_{m} missing: as their weight is strictly less than e, the minimality of e ensures that they are not tight, so the result I_2 still satisfies Q. We then homomorphically fold the dissociated edges into the copies of e, and obtain I_3, which still satisfies Q: it is like I_1 but without the copies of e with F_{m} missing. Now, the copies of e in I_3 have a smaller side weight than in I_1, so the minimality of e ensures that they are not tight. We can dissociate them again, yielding I_4, which still satisfies Q. We can now homomorphically fold the dissociated edges and obtain I_5, which still satisfies Q, and is homomorphic to the dissociation of e in I_1. As e was tight, I_5 should not satisfy the query, so we have reached a contradiction. ◀

This concludes the proof of Theorem 6.1, and thus of our main theorem (Theorem 3.3).



(a) A possible world ω of G with no s,t-path (dashed edges are the ones that are not kept): the vertices are colored in red or green depending on their side of the cut.

(b) Possible world of the coding in Figure 6c for the possible world of G at the left. Copies of e are dashed when they are missing the fact F_{m}. Vertices u_{e_i} corresponding to edges across the cut are in bold.

Figure 7 Illustration of a possible world (Figure 7a) of the graph G from Figure 6a, and the corresponding possible world (Figure 7b) of the coding (Figure 6c). The homomorphism of Figure 7b to the fine dissociation is given by the vertex colors: the red u-vertices are mapped to u, the red v-vertices are mapped to v', the green u-vertices are mapped to u', and the green v-vertices are mapped to v. The vertex colors are determined by the cut (Figure 7a) except for the bold vertices where it depends on the orientation choice.

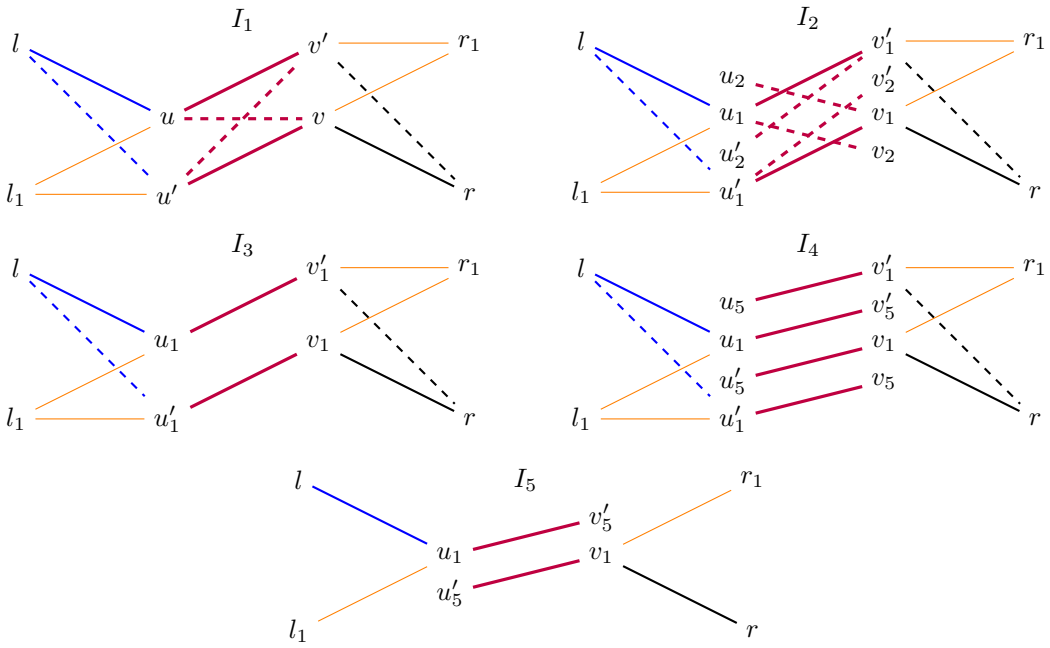


Figure 8 Illustration of the proof of Lemma 6.5, with I_1 being the fine dissociation I' of Figure 5, and I_5 being isomorphic to the dissociation on Figure 4.

7 Conclusion

We have shown that PQE is #P-hard for any unbounded UCQ$^\neq$ on an arity-two signature, and hence proved a dichotomy on PQE for all UCQ$^\neq$ queries: either they are unbounded and PQE is #P-hard, or they are bounded and the dichotomy by Dalvi and Suciu applies. Our result captures many query languages; in particular disjunctive Datalog over binary signatures, regular path queries, and all ontology-mediated queries closed under homomorphisms.

There are three natural directions to extend our result. First, we could study queries that are *not* homomorphism-closed, e.g., with disequalities or negation. We believe that this would require different techniques as the problem is still open for UCQs (beyond the results of [21]). Second, we could lift the arity restriction and work on signatures of arbitrary arity: we conjecture that PQE is still #P-hard for any unbounded UCQ$^\neq$ in that case. Much of our proof techniques may adapt, but we do not know how to extend the definitions of dissociation, fine dissociation, and iteration. In particular, dissociation on a fact is difficult to adapt because incident facts on arbitrary arity signatures may intersect in complicated ways. For this reason, we leave the extension to arbitrary-arity signatures to future work. Third, a natural question for future work is whether our hardness result on unbounded homomorphism-closed queries also applies to the *(unweighted) model counting problem*, where all facts of the TID must have probability 0.5: the hardness of this problem has only been shown on the class of self-join free CQs [4].

References

1 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of databases*. Addison-Wesley, 1995.

2 Antoine Amarilli, Pierre Bourhis, and Pierre Senellart. Tractable lineages on treelike instances: Limits and extensions. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS-16)*, pages 355–370. ACM, 2016.

3 Antoine Amarilli and İsmail İlkan Ceylan. A dichotomy for homomorphism-closed queries on probabilistic graphs, 2020. Full version with proofs: `https://arxiv.org/abs/1910.02048`.

4 Antoine Amarilli and Benny Kimelfeld. Model counting for conjunctive queries without self-joins. *CoRR*, abs/1908.07093, 2019.

5 Franz Baader, Diego Calvanese, Deborah L McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic handbook*. Cambridge University Press, 2007.

6 Pablo Barceló, Diego Figueira, and Miguel Romero. Boundedness of conjunctive regular path queries, 2019. URL: `https://hal.archives-ouvertes.fr/hal-02056388/`.

7 Michael Benedikt, Balder Ten Cate, Thomas Colcombet, and Michael Vanden Boom. The complexity of boundedness for guarded logics. In *2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 293–304. IEEE, 2015.

8 Meghyn Bienvenu, Balder Ten Cate, Carsten Lutz, and Frank Wolter. Ontology-based data access: A study through disjunctive Datalog, CSP, and MMSNP. *ACM Transactions on Database Systems (TODS)*, 39(4):33:1–33:44, 2014.

9 Stefan Borgwardt, İsmail İlkan Ceylan, and Thomas Lukasiewicz. Ontology-mediated queries for probabilistic databases. In *Proceedings of the 31th AAAI Conference on Artificial Intelligence (AAAI-17)*, pages 1063–1069. AAAI Press, 2017.

10 Stefan Borgwardt, İsmail İlkan Ceylan, and Thomas Lukasiewicz. Ontology-mediated query answering over log-linear probabilistic data. In *Proceedings of the 33rd National Conference on Artificial Intelligence (AAAI-19)*. AAAI Press, 2019.

11 Andrea Calì, Georg Gottlob, and Michael Kifer. Taming the infinite chase: Query answering under expressive relational constraints. *JAIR*, 48:115–174, 2013.

12 Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. A general Datalog-based framework for tractable query answering over ontologies. *J. Web Sem.*, 14:57–83, 2012.

13 İsmail İlkan Ceylan. *Query answering in probabilistic data and knowledge bases*. Doctoral thesis, TU Dresden, 2017.

14 İsmail İlkan Ceylan, Adnan Darwiche, and Guy Van den Broeck. Open-world probabilistic databases. In *Proceedings of the 15th International Conference on Principles of Knowledge Representation and Reasoning (KR-16)*, pages 339–348. AAAI Press, 2016.

15 Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC-71)*, pages 151–158. ACM, 1971.

16 Nilesh Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. *The VLDB Journal*, 16(4):523–544, 2007.

17 Nilesh Dalvi and Dan Suciu. The dichotomy of probabilistic inference for unions of conjunctive queries. *J. ACM*, 59(6), 2012.

18 Anuj Dawar and Stephan Kreutzer. On Datalog vs. LFP. In *International Colloquium on Automata, Languages, and Programming*, pages 160–171. Springer, 2008.

19 Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge Vault: A Web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 601–610. ACM, 2014.

20 Thomas Eiter, Magdalena Ortiz, Mantas Šimkus, Trung-Kien Tran, and Guohui Xiao. Query rewriting for Horn-$\mathcal{SHIQ}$ plus rules. In *AAAI*, 2012.

21 Robert Fink and Dan Olteanu. Dichotomies for queries with negation in probabilistic databases. *ACM Transactions on Database Systems (TODS)*, 41(1):4:1–4:47, 2016.

22 Haim Gaifman, Harry Mairson, Yehoshua Sagiv, and Moshe Y. Vardi. Undecidable optimization problems for database logic programs. *J. ACM*, 40(3):683–713, July 1993.

23 Georg Gottlob and Thomas Schwentick. Rewriting ontological queries into small nonrecursive Datalog programs. In *KR*, 2012.

24 Gerd G Hillebrand, Paris C Kanellakis, Harry G Mairson, and Moshe Y Vardi. Undecidable boundedness problems for Datalog programs. *The Journal of Logic Programming*, 25(2):163–190, 1995.

25 Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence*, 194:28–61, 2013.

26 Jean Christoph Jung. *Reasoning in many dimensions: uncertainty and products of modal logics*. PhD thesis, University of Bremen, 2014.

27 Jean Christoph Jung and Carsten Lutz. Ontology-based access to probabilistic data with OWL QL. In *Proceedings of the 11th International Conference on The Semantic Web - Volume Part I*, pages 182–197. Springer-Verlag, 2012.

28 T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. Never-ending learning. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI-15)*, pages 2302–2310, 2015.

29 Dan Olteanu and Jiewen Huang. Using OBDDs for efficient query evaluation on probabilistic databases. In *Proceedings of the 2nd International Conference on Scalable Uncertainty Management (SUM-08)*, volume 5291 of *LNCS*, pages 326–340, 2008.

30 Dan Olteanu and Jiewen Huang. Secondary-storage confidence computation for conjunctive queries with inequalities. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, pages 389–402. ACM, 2009.

31 J. Scott Provan and Michael O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, 12(4), 1983 .

32 Christopher Ré and Dan Suciu. The trichotomy of HAVING queries on a probabilistic database. *The VLDB Journal*, 18(5):1091–1116, 2009.

33 Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. *Probabilistic databases*, volume 3. Morgan-Claypool, 2011.

34 Leslie Gabriel Valiant. The complexity of computing the permanent. *TCS*, 8(2):189–201, 1979.