A Dichotomy for Homomorphism-Closed Queries on Probabilistic Graphs

Antoine Amarilli
LTCI, France
Télécom Paris, France
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^∞. Our main result states that all *unbounded* queries in UCQ^∞, i.e., queries with infinitely many minimal models, are #P-hard for PQE. As *bounded* queries in UCQ^∞ are already classified by the dichotomy of Dalvi and Suciu [17], our results and theirs imply a complete dichotomy for $\mathrm{PQE}(\mathrm{UCQ}^\infty)$ on arity-two signatures. This dichotomy covers in particular all query languages contained in UCQ^∞ 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 either from counting the valuations of positive partitioned 2-DNF formulae (#PP2DNF), or from the source-to-target reliability problem in an undirected graph (#U-ST-CON), depending on properties of the minimal models of the query.

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

Keywords and phrases Tuple-independent database, #P-hardness, recursive query

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

The management of *uncertain and probabilistic data* is an important problem in many applications, e.g., automated knowledge base construction [26, 29, 19], data integration from diverse sources, predictive and stochastic modeling, applications based on (error-prone) sensor readings, etc. To represent probabilistic data, the most natural model is that of tuple-independent *probabilistic databases (TIDs)* [34]. 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.

In a breakthrough result, Dalvi and Suciu 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. In particular, they have shown that the probability of any UCQ can either be computed in polynomial time or it is #P-hard to compute. The queries that enjoy tractable PQE are called *safe*, and all other queries are called *unsafe*. This result has served as the foundation for many other subsequent works that investigated the complexity of PQE [21, 30, 31, 33, 13, 2, 28].

Despite the extensive research on TIDs, there is only little known for PQE for monotone

Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

query languages beyond UCQs. In particular, we are not aware of results concerning (negation-free) Datalog, which captures patterns that are not first-order definable, most notably *recursion* which is an essential ingredient in many applications. To this date, it remained unknown whether a dichotomy on PQE could be shown for (negation-free) Datalog, or for ontology-mediated queries [12].

In this work, we push the boundaries of the dichotomy for unions of conjunctive queries to a large class of *monotone* (i.e., positive) queries; namely, all queries that are closed under homomorphisms. This class can equivalently be viewed as the *infinite* union of conjunctive queries, abbreviated as UCQ^{∞}. Notably, UCQ^{∞} contains many well-known query languages such as (negation-free) disjunctive Datalog, regular path queries (RPQs), and a large class of ontology-mediated queries, as we elaborate in the discussion of related work. More specifically, we are interested in the following question: Does PQE admit a data complexity dichotomy on the class UCQ^{∞}? As any *bounded* UCQ^{∞} query is equivalent to a UCQ, it suffices to classify unbounded queries in UCQ^{∞}, i.e., queries with infinitely many minimal models.

The main result of this paper is to complete the dichotomy and show that PQE is #P-hard for *any unbounded* UCQ^{∞} *query*, in the case of an arity-two signature (where all relations are binary); the input to query evaluation is then a (labeled) *probabilistic graph*. This restricted setting captures many applications such as the ones dealing with ontologies and graph data, e.g., regular path queries (RPQs): we accordingly restrict to arity-two throughout the paper. It is of course not surprising that some queries in UCQ^{∞} are unsafe for similar reasons as unsafe UCQs, e.g., we can easily show that the RPQ $RS^{*}T$ is #P-hard by reducing from the well-known #P-hard query $Q_0 : R(x) \wedge S(x, y) \wedge T(y)$ from [16]. However, the challenge is to show hardness for *every* query in UCQ^{∞} that is not equivalent to a UCQ.

The proof proceeds by first showing a reduction from the problem of counting the valuations of positive partitioned 2-DNF formulae (#PP2DNF), which is also the problem used to show #P-hardness of the Q_0 query above. We show how this reduction can be applied to any query in UCQ^{∞} with a minimal model having a so-called *non-iterable edge*. Intuitively, the #PP2DNF reduction applies if we can find an edge to code the #PP2DNF problem, without having “back-and-forth” patterns that can make the query true.

When the #PP2DNF reduction fails, we show that we can instead use the unboundedness of the query to reduce from the source-to-target reliability problem in an undirected graph (#U-ST-CON). To do this, we must first study the minimal models of unbounded queries, and we show how to find one with a *tight edge*, i.e., an edge whose *dissociation* (replacing the edge by two copies) would make the query false. Picking one such edge which is iterable, we can code #U-ST-CON in a way that ensures that any source-to-target path will witness the existence of an iterate of the minimal model, making the query true. To argue that the query is false when there is no path, we must rely on a notion of *fine dissociation* that relies on some minimality assumptions that we can make on our choice of tight edge.

Related Work. Research on probabilistic databases is a well-established topic: we refer the reader to the book [34]. It was first shown in [24] that query evaluation for some queries on probabilistic databases is #P-hard, and Dalvi and Suciu [16] established a dichotomy result on queries: a self-join free conjunctive query is safe if it is *hierarchical*, and #P-hard if it is *non-hierarchical*. 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], and other results allow for disequality ($\neq$) joins in the queries [30] or for inequality ($<$) joins [31]. There is also a trichotomy result over queries with aggregation [33]. Some dichotomies are known for extended models, e.g., the dichotomy of Dalvi and Suciu on TIDs has been lifted to so-called *open-world probabilistic databases* [13]. However, we are not aware of dichotomies applying to Boolean queries beyond first-order.

Query evaluation on probabilistic databases has also been studied in the context of *ontology-mediated queries* (OMQs). In this context, the query also includes an *ontology*, i.e., (non-probabilistic) logical rules that can be used for inference. The complexity of evaluating such OMQs on probabilistic databases has been investigated for ontologies both in Description Logics [28] and in Datalog$^{\pm}$ [7, 8]. However, most of these classification results apply to OMQs that are *FO-rewritable*, i.e., the query with the ontology is in fact equivalent to a first-order query. The only exception that we know is the result on the Description Logic $\mathcal{ELI}$ [28], for which OMQs are rewritable to UCQ$^\infty$ but not to first-order. Our work generalizes their result (Theorem 6 of [28]) by showing hardness for any unbounded UCQ$^\infty$, not just the ones expressible in $\mathcal{ELI}$; and our techniques in Section 4 are related to theirs. However, their proof (Theorem 6 in [28] and Theorem 5.31 of [27]) has a gap due to a subtle problem concerning "back-and-forth matches"[1]. This problem only occurs because of inverse roles in $\mathcal{ELI}$, and thus would not occur in the case of the description logic $\mathcal{EL}$. Our own proof (second item of Proposition 4.8, and the material of Sections 5 and 6) addresses this problem, and provides a complete proof of Theorem 6 in [28], in addition to applying to all unbounded UCQ$^\infty$ (beyond the ones expressible in $\mathcal{ELI}$).

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 Appendix A), as is more common in some contexts such as description logics.

A σ*-fact* is an expression of the form $R(a, b)$ where R is a predicate and a, b are constants. 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 union of the domain of its facts.

Whenever we consider a σ-instance I, we can always see it as a $\sigma^{\leftrightarrow}$-instance defined as consisting of all the σ-facts in I, plus all the facts $R^-(b, a)$ for each fact $R(a, b)$ of I. Thus, whenever we consider a σ-instance I and say, for instance, that we consider all $\sigma^{\leftrightarrow}$-facts $F = R(a, b)$ of I that contain some $a \in \mathrm{dom}(I)$, we mean all the facts of the form $S(a, b)$ of I with $S \in \sigma$, and also all the facts of the form $S^-(a, b)$ for $S \in \sigma$, 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 $u, v \in \mathrm{dom}(I)$ that co-occur in some fact

[1] We have communicated the problem with the authors and they kindly confirmed.

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 e are the σ-facts of I that use both u and v. Slightly abusing notation, we will also 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 will then talk about the facts that *realize* e as the $\sigma^{\leftrightarrow}$-facts of the form $R(u, v)$, i.e., the $S(u, v)$ of I with $S \in \sigma$, and the $S^-(u, v)$ of I with $S \in \sigma$, corresponding to the σ-fact $S(v, u)$ of I. So if we say that we add a fresh element v' to $\mathrm{dom}(I)$ and create a copy of the facts of e on (u, v'), it means that we create $S(u, v')$ for all facts $S(u, v')$ realizing e with $S \in \sigma$, and we create $S(v', u)$ for all facts $S^-(u, v')$ realizing e with $S \in \sigma$.

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* if one of its elements (possibly both) is a leaf; otherwise, it is a non-leaf.

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 the domain of I to that of 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. We say that an instance I *satisfies* a query Q, that Q *holds* on I, or that 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, the query Q_1 holds on I iff Q_2 holds on I. All queries studied in this work 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. We call UCQ^∞ the class of all homomorphism-closed queries.

One well-known subclass of UCQ^∞ is *unions of conjunctive queries* (UCQs), without negation or inequalities. Formally, a *conjunctive query* (CQ) is an existentially quantified conjunctions of atoms, and a UCQ is a disjunction of CQs. For brevity, we will omit existential quantification when writing UCQs, and abbreviate the conjunction with a comma. For instance, the UCQ $R(x, y), S(x, z) \vee T(x, y)$ holds exactly when the instance contains a T-fact or when it contains an R-fact and an S-fact sharing the same first element. Note that queries in UCQ^∞ can be seen as an kind of infinite UCQs (hence the notation), i.e., a query in UCQ^∞ can always be seen as an infinite disjunction of CQs corresponding to the models of the query.

Another subclass of UCQ^∞ is *Datalog*, again without negation or inequalities. Intuitively, 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 now intensional facts can be *derived* from other intensional facts and from database facts (called *extensional*). The interpretation of the intensional predicates is defined by taking the (unique) least fixpoint of the rules, and the query holds iff the Goal() predicate can be derived. For a formal definition of the semantics, refer to [1]. Note that, when defining Datalog programs in our setting, the intensional relations can have arbitrary arity, i.e., they do not have to be arity-two. All Datalog queries are homomorphism-closed: intuitively, a Datalog program defines a UCQ^∞

having one disjunct for each possible *derivation tree* for the program. However, there are some homomorphism-closed queries that are not expressible in Datalog [18].

Another subclass of UCQ^∞ is the so-called *ontology-mediated queries* or OMQs [6], that is, database queries (typically, UCQs) coupled with an ontology, i.e., a set of logical constraints. More precisely, an OMQ is a pair $(Q, \mathcal{T})$, where Q is a UCQ, and $\mathcal{T}$ is an ontology in some logical formalism. A database instance I *satisfies* an OMQ $(Q, \mathcal{T})$ when the instance I and the logical constraints $\mathcal{T}$ entail the query Q in the standard sense – see, e.g., [6] for a formal definition. There is a large class of OMQ languages, mostly based on Description Logics (DLs) [3], and existential rules (also known as tuple-generating dependencies, or Datalog$^\pm$) [10, 11]. A prominent paradigm to evaluate OMQs is based on the notion of *rewritability*. For instance, an OMQ Q is *Datalog-rewritable* w.r.t. $\mathcal{T}$ if there is a Datalog query $Q_\mathcal{T}$ such that, for every database I consistent with $\mathcal{T}$, the query Q is entailed by I and $\mathcal{T}$ iff the rewriting $Q_\mathcal{T}$ holds in I. Many ontology languages admit efficient rewritings to Datalog (even on arity-two signatures) [23, 20]. Thus, the negation-free fragment of many ontology languages falls into the class UCQ^∞ on arity-two signatures.

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 $\mathrm{PQE}(Q)$, is then the task of computing $\mathrm{P}_\mathcal{I}(Q)$ for a given TID $\mathcal{I}$.

Complexity Background. FP is the class of functions $f : \{0,1\}^* \mapsto \{0,1\}^*$ computable by a polynomial-time deterministic Turing Machine, i.e., it is like the usual class P but for computation problems instead of decision problems. The class #P, introduced by Valiant [35], contains the computation problems that can be expressed as the number of accepting paths of a nondeterministic polynomial-time Turing machine. Formally, 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 $f(x) = |\{y \in \{0,1\}^{p(|x|)} \mid M \textit{ answers } y \textit{ on the input } x\}|$.

Several types of reductions exist for #P, while the most common being *polynomial-time Turing reductions* [14]. Informally, Turing reductions generalize the standard many-one reductions in the sense that they also allow access to an oracle. Thus, 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 ones used to show #P-hardness in the dichotomy of Dalvi and Suciu [17]. All of our reductions in this work are polynomial-time Turing reductions and more specifically *1-Turing reductions*, i.e, they require only a single oracle call to #P.

We study the *data complexity* of PQE(Q), which is measured only as a function of the input instance I, i.e., we assume that the signature and Q are fixed. It is immediate that the problem PQE(Q) is in the complexity class $\mathrm{FP}^{\#P}$ of computation problems that can be performed in polynomial time with access to a #P-oracle, as 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 the query holds, with polynomial-time normalization at the end to go from a number of runs to probabilities. Our focus in this work is to show that the problem is also #P-hard.

Hard problems. We will show hardness by reducing from two well-known #P-hard problems. For some queries, we will reduce from the *undirected st-connectivity problem* (#U-ST-CON) [32]:

▶ **Definition 2.1.** *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, determine the probability that there exists a path from s to t, where each graph edge has probability* 0.5.

In other cases, we will reduce from a more local problem, called #PP2DNF, which a standard tool to show hardness of unsafe UCQs. In our proof, we use it for unbounded queries. The original problem (given in [32]) uses Boolean formulas, but we give an equivalent rephrasing in terms of bipartite graphs.

▶ **Definition 2.2.** *Given a bipartite graph $H = (A, B, C)$ with edges $C \subseteq A \times B$, a* possible world *of H is a pair (A', B') with $A' \subseteq A$ and $B' \subseteq B$. We call the possible world* good *if one vertex of A' and one vertex of B' are adjacent in C, and* bad *otherwise. The* positive partitioned 2DNF problem (#PP2DNF) *is the following: Given a bipartite graph, compute how many of its possible worlds are good.*

Note that we can clearly assume without loss of generality that the bipartite graph H is *connected*, 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.

3 Result Statement

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

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

This dichotomy result holds for arbitrary arity queries, and characterizes the complexity of the PQE problem for UCQs. However, it does not apply to other homomorphism-closed languages beyond UCQs, as pointed out earlier. Our contribution, when restricting to the arity-two setting, is to generalize the dichotomy to UCQ^∞, i.e., to apply to *any* query closed under homomorphisms. Specifically, we show that all such queries are intractable unless they are equivalent to a UCQ.

▶ **Theorem 3.2.** *Let Q be a* UCQ^∞ *on an arity-two signature. Then,* PQE(Q) *is either in* FP *or it is* #P*-hard.*

Our result relies on the dichotomy of Dalvi and Suciu for queries that are equivalent to UCQs. The key point is then to show intractability for the remaining queries. Specifically, we speak of *unbounded* queries as queries which are closed under homomorphisms, but not equivalent to a UCQ, and show the following.

▶ **Theorem 3.3.** *Let Q be an unbounded* UCQ^∞ *on an arity-two signature. Then,* $\mathrm{PQE}(Q)$ *is* #P*-hard.*

Examples of unbounded queries are regular path queries such as RS^*T. Datalog queries can be either bounded (i.e., equivalent to a UCQ) or unbounded, as in the case, e.g., of the following program with one monadic intensional predicate U on extensional signature R, S, T which corresponds to the RPQ RS^*T:

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

Thus, for instance, the PQE problem for Datalog queries is #P-hard whenever the query is not equivalent to a UCQ, which is the case unless the Datalog program is nonrecursive or recursion is *bounded* [25].

Effectiveness and uniformity. We do not know whether we can effectively decide our dichotomy result in Theorem 3.2, i.e., given a query closed under homomorphisms, determine whether PQE(Q) is #P-hard or in FP. For UCQs, the dichotomy of Theorem 3.1 is effective using the super-exponential algorithm of [17], with the precise complexity being open. For more general query languages, the problem would depend on how the input query is represented. In the case of Datalog queries, for instance, we do not know if the problem is even decidable, because it is generally undecidable whether an input Datalog program is bounded [22]. Nevertheless, this does not imply undecidability in our context, as we could imagine a procedure that would, e.g., identify unsafe Datalog queries without needing to decide boundedness.

However, our dichotomy is effective for more restricted query languages for which we can decide boundedness, e.g., monadic Datalog or its generalization GN-Datalog [5], or C2RPQs for which boundedness was recently shown to be decidable [4].

For queries that we show to be #P-hard, we also do not focus on the question of whether the PTIME reduction can effectively be "found", i.e., given the #P-hard query, compute what the reduction is. All that matters is that, once the query is fixed, some PTIME reduction procedure exists. 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 which halts". We leave to future work the investigation of better-behaved query languages where we can bound the complexity (as a function of the query) of performing the reduction.

Proof outline. Theorem 3.3 is proven in the next three sections. In Section 4, we consider the case of queries for which we can find a model with a so-called *non-iterable edge*, intuitively 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 $Q_0 : R(w,x), S(x,y), T(y,z)$ 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 unbounded queries 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.

In Section 6, we use minimal tight patterns to cover 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 argue that the query is satisfied iff there is a path.

4 Hardness with Non-Iterable Edges

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

▶ **Definition 4.1.** *Let I be an instance and $e = (u, v)$ be a non-leaf edge of I. A* neighbor choice *of e is a pair of $\sigma^{\leftrightarrow}$-facts $N = (F_l, F_r)$ of I where F_l is of the form $R_l(l, u)$ and F_r is of the form $R_r(v, r)$ with $l \neq v$ and $r \neq u$. We write $I_{e,N}$ to denote an instance I with a non-leaf edge e relative to a neighbor choice N.*

Note that R_l and R_r are $\sigma^{\leftrightarrow}$-relations. Hence, we may have $R_l = R_r$, and we may have $l = r$. Let us illustrate the notion of neighbor choice on an example.

▶ **Example 4.2.** Given an instance $I = R(a, b), S(c, b), R(d, c)$, the edge (b, c) is non-leaf and a neighbor choice for it is $(R(a, b), R^{-}(c, d))$.

Note that every non-leaf edge (u, v) must have a neighbor choice, as we can pick F_l and F_r from the edges incident to u and v which are not e. We can now define the *iteration process*, which creates a path of the edge e, keeping the facts F_l and F_r at the beginning and end of the path, and copying all other incident facts:

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

- Copy non-incident facts: *Initialize $I^n_{e,N}$ as the induced subinstance of I relative to the domain $\mathrm{dom}(I) \setminus \{u, v\}$.*
- Copy incident facts F_l and F_r: *Add F_l and F_r to $I^n_{e,N}$, using u_1 and v_n, respectively.*
- Copy other facts incident to u: *For every $\sigma^{\leftrightarrow}$-fact $F'_l = R'_l(l', u)$ with $l' \neq v$ and $F'_l \neq F_l$, add the fact $R'_l(l', u_i)$ to $I^n_{e,N}$ for each $1 \leq i \leq n$.*
- Copy other facts incident to v: *For every $\sigma^{\leftrightarrow}$-fact $F'_r = R'_r(v, r')$ with $r' \neq u$ and $F'_r \neq F_r$, add the fact $R'_r(v_i, r')$ to $I^n_{e,N}$ for each $1 \leq i \leq n$.*
- Create copies of e: *For each $\sigma^{\leftrightarrow}$-fact $R(u, v)$ realizing e in I, add the $\sigma^{\leftrightarrow}$-fact $R(u_i, v_i)$ to $I^n_{e,N}$ for each $1 \leq i \leq n$, and add the $\sigma^{\leftrightarrow}$-fact $R(u_{i+1}, v_i)$ to $I^n_{e,N}$ for each $1 \leq i \leq n - 1$.*

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 edges as the original endpoints except that we have removed one fact in the label of one edge on each endpoint, as indicated by the neighbor choice. The reason why must choose two incident *facts* (not edges) in the neighbor choice is because in the PQE problem we give probabilities to single facts and not edges.

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

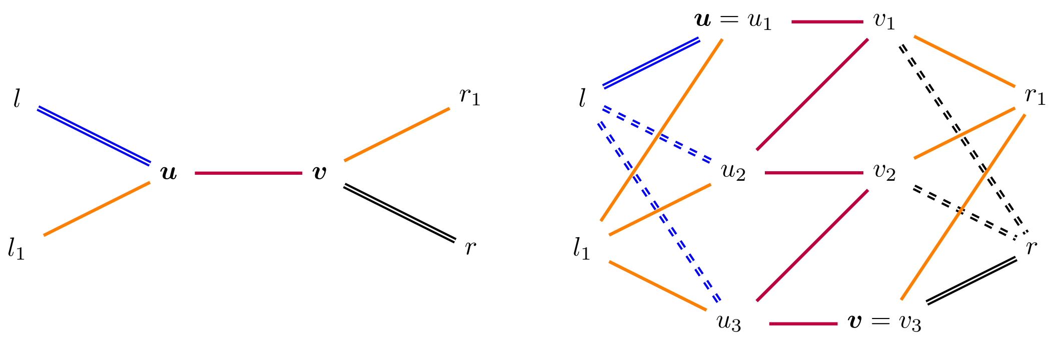


Figure 1 Example of iterating an edge in an instance (from left to right). Lines represent edges (realized by multiple $\sigma^{\leftrightarrow}$-facts). The iterated edge is the purple line at the middle; the double blue and black lines are the edges of F_l and F_r respectively; their dashed versions on the right are the same edges without the facts F_l and F_r respectively.

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

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 a neighbor choice, there are two possible regimes. Either all iterations still satisfy Q, or there is some iteration where Q is violated (and, by Observation 4.4, all subsequent iterations also violate Q). We formalize this as follows.

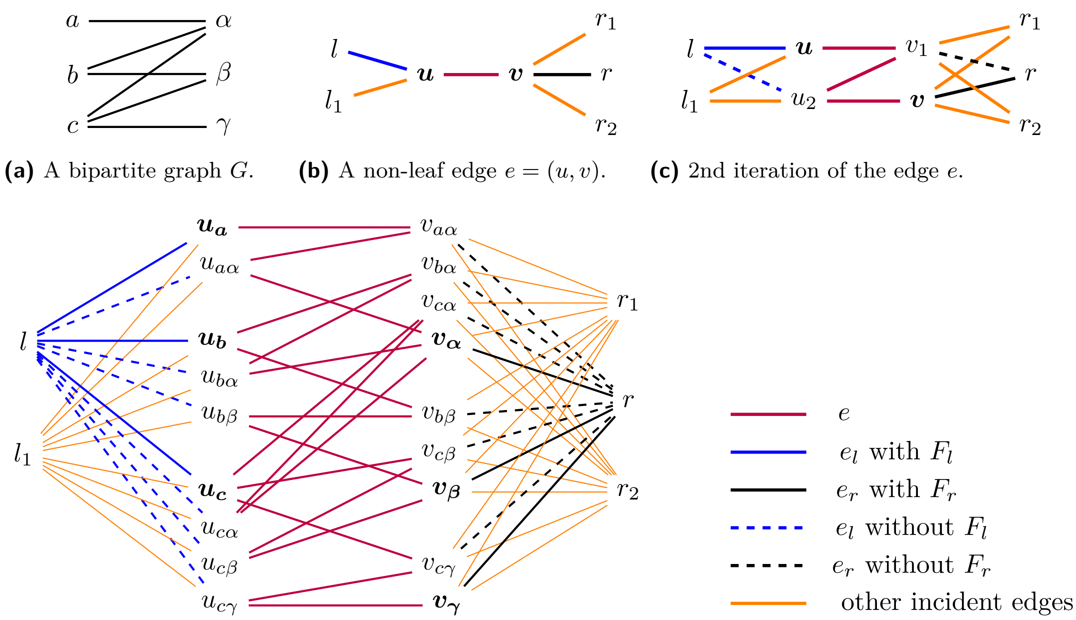
▶ **Definition 4.5.** *A non-leaf edge e of a model I of a query Q is* iterable *relative to a neighbor choice N if $I^n_{e,N}$ satisfies Q for each $n \geq 1$; otherwise, it is* non-iterable.

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. Formally:

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

Note that this result applies to arbitrary homomorphism-closed queries, whether they are bounded or not. If we consider for instance the query $R(w, x), S(x, y), T(y, z)$ (which is the arity-2 version of the prototypical hard query for TIDs [16]), then the model $R(a, b), S(b, c), T(c, d)$ has an edge $\{b, c\}$ which is non-leaf and non-iterable: its iteration with $n = 2$ relative to the only possible neighbor pair yields $R(a, b), S(b, c'), S(b', c'), S(b', c), T(c, d)$ which does not satisfy the query. Whenever we have a model of this kind, we will be able to show hardness by reducing from #PP2DNF (Definition 2.2).

Note that Theorem 4.6 also applies to bounded queries. On such queries, it generalizes the hardness part of the dichotomy result of [16] for *self-join-free* conjunctive queries (SJFQ), i.e., all non-hierarchical self-join free CQs are hard. Indeed, it is easy to see that whenever a SJFQ has *no* model with a non-leaf, non-iterable edge, then it must be hierarchical. However, Theorem 4.6 does not capture the hardness of some UCQs with self joins. For instance, the query $(R(w, x), S(x, y)) \lor (S(x, y), T(y, z))$ is #P-hard but does not have a model with a non-leaf and non-iterable edge: intuitively, we can evaluate this query just by looking at pairs of facts that share an element, which iteration does not affect. Theorem 4.6 will nevertheless be sufficient for our purposes of showing hardness for all *unbounded queries*, as we will do in the next sections.



(a) A bipartite graph G. **(b)** A non-leaf edge $e = (u, v)$. **(c)** 2nd iteration of the edge e.

(d) Coding of the bipartite graph G, where every edge is encoded by the 2nd iteration of e.

Figure 2 Example of a coding based on a bipartite graph G shown in Fig 2(a). An edge e (Fig 2(a)) is used in the encoding of G, and the result of iterating e is shown in Fig 2(c). The final encoding of the problem is illustrated in Fig 2(d). A key explains the colors (bottom right).

Thus, let us start presenting the construction needed to prove Theorem 4.6. In what follows, we define a coding that takes a bipartite graph and constructs a (probabilistic) instance in polynomial-time such that there is a bijective correspondence between the possible worlds of the bipartite graph and the possible worlds of the probabilistic instance.

▶ **Definition 4.7.** *Let $H = (A, B, C)$ be a connected bipartite graph. The* coding *of H relative to an instance I, to a non-leaf edge $e = (u, v)$ of I, to a neighbor choice $N = F_l, F_r$ of e with $F_l = R_l(l, u)$, $F_r = R_r(v, r)$, and to $n \geq 1$, is a probabilistic instance $\mathcal{I} = (J, \pi)$. The domain of J is $\mathrm{dom}(J) := (\mathrm{dom}(I) \setminus \{u, v\}) \cup \{u_a \mid a \in A\} \cup \{v_b \mid b \in B\} \cup \{u_{c,2}, \dots, u_{c,n} \mid c \in C\} \cup \{v_{c,1}, \dots, v_{c,n-1} \mid c \in C\}$. The facts of J and the mapping π are defined by the following steps:*

- 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: *For each $a \in A$ add the $\sigma^{\leftrightarrow}$-fact $R_l(u_a, l)$ to J, and similarly for each $b \in B$ add the $\sigma^{\leftrightarrow}$-fact $R_r(r, v_b)$ to J.*
- Copy other facts incident to u: *For each $\sigma^{\leftrightarrow}$-fact $F'_l = R'_l(u, r')$ of I with $r' \neq v$ and $F'_l \neq F_l$, add the $\sigma^{\leftrightarrow}$-facts $R'_l(u_a, r')$ for each $a \in A$ and $R'_l(u_{c,j}, r')$, for each $2 \leq j \leq i$ and $c \in C$.*
- Copy other facts incident to v: *For each $\sigma^{\leftrightarrow}$-fact $F'_r = R'_r(l', v)$ of I with $l' \neq u$ and $F'_r \neq F_r$, add the $\sigma^{\leftrightarrow}$-facts $R'_r(l', v_b)$, for each $b \in B$ and $R'_r(l', v_{c,j})$, for each $1 \leq j \leq i-1$ and $c \in C$.*
- Create copies of e: *For each $c \in C$ with $c = (a, b)$, create $2n - 1$ copies of e on the following new edges in J (i.e., copy all the $\sigma^{\leftrightarrow}$-facts of the edge e):*
  - $(u_a, v_{c,1})$,
  - $(u_{c,n}, v_b)$,

- $(u_{c,j}, v_{c,j})$ *for* $2 \leq j \leq n-1$, *and*
- $(u_{c,j+1}, v_{c,j})$ *for* $1 \leq j \leq n-1$

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 iteration: one other way to see the definition is that we code each edge of the bipartite graph as a copy of the n-th iteration of (u, v). Note also that there are only $|A| + |B|$ uncertain facts, by construction. It is clear that this coding is in polynomial time in H. The result of the coding is illustrated in Figure 2.

We now define the bijective function ϕ relating the possible worlds of the connected bipartite graph H to those of the probabilistic instance $\mathcal{I} = (J, \pi)$. For each vertex $a \in A$ we keep the copy of F_r incident to u_a if a is selected and we do not keep it otherwise, and we do the same for v_b and F_l. It is obvious that this correspondence is bijective and that all corresponding possible worlds have the same probability, namely, $0.5^{|A|+|B|}$. We can now prove the following reduction (independently from any particular query), recalling the notion of *good* and *bad* possible worlds of H from Definition 2.2:

▶ **Proposition 4.8.** *Let the probabilistic instance* $\mathcal{I} = (J, \pi)$ *be the coding of a connected bipartite graph* $H = (A, B, C)$ *relative to an instance* $I_{e,N}$*, 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, the following statements hold:*

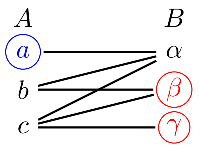
1. *For any* good *possible world* ω *of* H*,* $\phi(\omega)$ *has a homomorphism* from $I^n_{e,N}$*.*
2. *For any* bad *possible world* ω *of* H*,* $\phi(\omega)$ *has a homomorphism* to $I^{3n-1}_{e,N}$*.*

Proof. Observe that (1) corresponds to the soundness of the reduction, and (2) to the completeness. We start with the easier direction, and then prove the other direction.

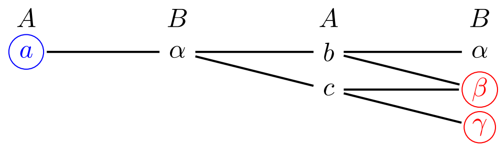
1. Let us assume that $\phi(\omega) = J'$. We more specifically claim that J' has a subinstance which is isomorphic to $I^n_{e,N}$. To see why, drop all copies of u from J' except u_a and the $u_{c,i}$, and all copies of v except v_b and the $v_{c,i}$, along with all facts where these elements appear. All of the original instance I except for the facts involving u and v can be found as-is in J'. Now, for the others, u_a has an incident copy of all edges incident to u in J' (including F_l), the same is true for v_b and v (including F_r), and we can use the $u_{e,i}$ and $v_{e,i}$ to witness the requisite path of copies of e.

2. As before, let us assume that $\phi(\omega) = J'$. Let us describe the homomorphism from J' to $I^{3n-1}_{e,N}$. To do this, first map all facts of J' that do not involve a copy of u or v to the corresponding facts of $I^{3n-1}_{e,N}$ using the identity mapping (which is possible as these facts are always untouched by our transformations). We will now explain how the copies of u and v are mapped to copies of u and v in $I^{3n-1}_{e,N}$: this clearly ensures that all incident facts to copies of u except F_r and the facts of the copies of e, and all incident facts to v except F_l and the facts of the copies of e. So all that remains is to map the copies of u and v as we said we would, in a way that ensures that we can map the copies of F_l, F_r, and e.

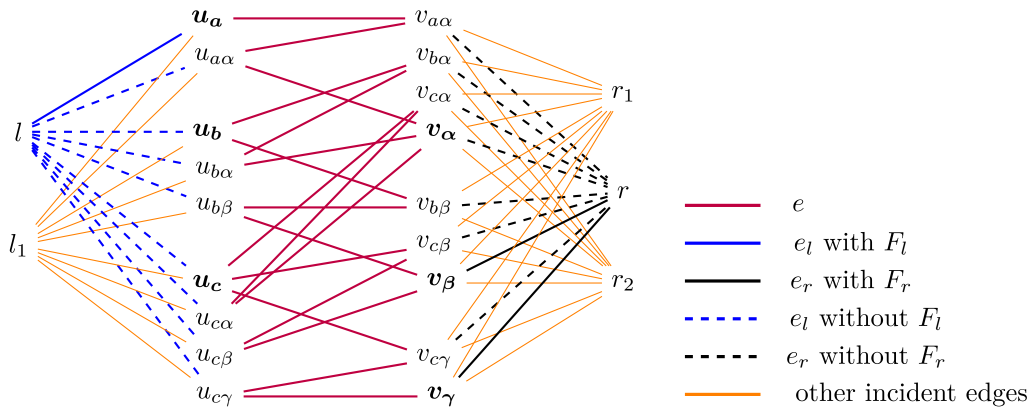
Our way to do this is illustrated in Figure 3. The first step is to take all copies of F_r in J', which correspond to vertices in $a \in A$ that were kept, and we map them all to the element u in $I^{3n-1}_{e,N}$, which is possible as it has the incident fact F_l. Now, we follow the paths of $2i-1$ copies of e back-and-forth until we reach vertices of the form v_b, and we map these paths to the first $2i-1$ edges of the path of copies of e from u to v in $I^{3n-1}_{e,N}$. From our assumption about the possible world J', none of the v_b reached at that stage have an incident copy of F_l, as we would otherwise have a witness to the fact that we kept two adjacent $a \in A$ and $b \in B$ in the possible world of H.



(a) A possible world H of G, where only circled nodes are in H.



(b) An unravelling of the possible world H up to paths of length 3.



(c) The edges (v_1, u_b), (v_1, u_c), and $(v_\alpha, u2)$ are changed to dashed lines, as their probabilistic facts are not in H.

Figure 3 Example of the backwards correctness of the proof: Fig 3(a) shows a possible world of the bipartite graph that violates the query. The unravelling of H is depicted in Fig 3(b). The possible world of the coding is given in Fig 3(c). A key explains the colors (bottom right).

The second step is to go back on the copies of e incident to these vertices that were not yet visited, and we follow a path of copies of e that were not yet mapped, which we map to the next $2i-1$ copies of e in the path from u to v in $I_{e,N}^{3n-1}$. We then reach elements of the form u_a, and they do not have any incident copies of F_r because all such edges and their outgoing paths were visited in the first step.

The third step is to go forwards on any outgoing edges to follow a path of copies of e that goes to vertices of the form v_b, mapping this to the last $2i-1$ edges of the path from u to v in $I_{e,N}^{3n-1}$. Some of these v_b may now be incident to copies of F_r, but the same is true of v in $I_{e,N}^{3n-1}$ and we have just reached it, so we can map these facts correctly.

The fourth step is to go backwards on any outgoing edges, going back on the path from u to v in $I_{e,N}^{3n-1}$, reaching vertices of the form u_a (which cannot be incident to any copy of F_r for the same reason as in the second step), and go forwards on any outgoing edges, going forward on the path from u to v and reaching again v, reaching vertices of the form v_b in J' that we map to b in $I_{e,N}^{3n-1}$, including the F_l-fact that may be incident to them. We repeat this process until everything reachable has been visited.

This means that everything was visited, as we have assumed without loss of generality that the bipartite graph was connected. Hence, we have mapped all elements in a homomorphic way, which concludes the description of the homomorphism and concludes the proof. ◀

We have established Proposition 4.8, which shows the required properties of our reduction, so we are ready to prove Theorem 4.6.

Proof of Theorem 4.6. Fix the query Q, the instance I, the non-leaf edge e of I which is non-iterable, and let us take the smallest $n > 1$ such that $I^n_{e,N}$ does not satisfy the query, but $I^{n-1}_{e,N}$ does.

We show #P-hardness by reducing from #PP2DNF (Definition 2.2). Let $H = (A, B, C)$ be an input connected bipartite graph. We apply the coding of Proposition 4.8 with $n - 1$ and obtain a probabilistic instance $\mathcal{I}$. This coding can be done in polynomial time.

Now let us use Proposition 4.8. We know that $I^{n-1}_{e,N}$ satisfies Q, but $I^{3(n-1)-1}_{e,N}$ does not, because $n > 1$ so $3(n-1) - 1 = 3n - 4 \geq n$, and as we know that $I^n_{e,N}$ violates Q, then so does $I^{3(n-1)-1}_{e,N}$ by Observation 4.4. Thus, Proposition 4.8 implies that the number of good possible worlds of H is the probability that Q is satisfied in a possible world of $\mathcal{I}$, multiplied by the constant factor $2^{|A|+|B|}$. Thus, the number of good possible worlds of H is $\mathrm{P}_{\mathcal{I}}(Q) \cdot 2^{|A|+|B|}$. This shows that the reduction is correct, and concludes the proof. ◀

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 open the case of unbounded queries for which, in all models, all non-leaf edges can be iterated.

In this section, we prove a general result on unbounded queries independent from the previous section: all unbounded queries must have a model with a *tight edge*, and we explain how to take it *minimal* in some sense. Tight edges and iterable edges are the two ingredients that we will use in Section 6 to show that unbounded queries are always hard.

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

▶ **Definition 5.1.** *The* dissociation *of a non-leaf edge* $\{a, b\}$ *in* I *is the instance* I' *where:*

- $\mathrm{dom}(I') = \mathrm{dom}(I) \cup \{a', b'\}$ *where* a' *and* b' *are fresh.*
- I' *is* I *where we remove the facts of the edge* $\{a, b\}$ *and add, for each such fact* $R(a, b)$ *(resp.,* $R(b, a)$*) the facts* $R(a, b')$ *and* $R(a', b)$ *(resp.,* $R(b, a'$ *and* $R(b', a)$*).*

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

▶ **Example 5.2.** Consider the instance $I = \{R(a, b), S(b, a), T(b, a), R(a, c), S(c, b), S(d, b)\}$. The edge $\{a, b\}$ is non-leaf, as witnessed by the edges $\{a, c\}$ and $\{b, c\}$, for instance. The result of the dissociation is then the instance

$$I' = \{R(a, b'), S(b, a'), T(b, a'), R(a', b), S(b', a), T(b', a), R(a, c), S(c, b), S(d, b)\},$$

where all facts which do not belong to the edge $\{a, b\}$ remain unchanged.

We then define the notion of a *tight* edge, which intuitively determines whether an edge is critical to make the query true.

▶ **Definition 5.3.** *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 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 three edges $\{u, a\}, \{a, b\}, \{b, v\}$ (possibly $u = v$) such that disconnecting the facts with a dissociation makes the query false. So, in a sense, a tight pattern also resembles the prototypical unsafe CQ Q_0 : $R(w, x), S(x, y), T(y, z)$ from [16] – but again for bounded queries they do not handle all cases, e.g., they do not handle $Q_1 : (R(w, x), S(x, y)) \vee (S(x, y), T(y, z))$.

For our purposes, we will not only need tight patterns, but *minimal tight pattern*, as we now define.

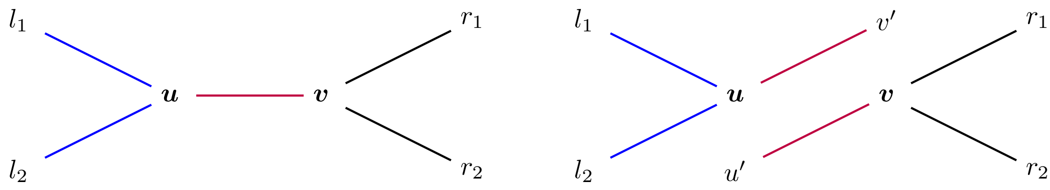


Figure 4 An instance on the left with a non-leaf edge (u, v), and the result of dissociating this edge on the right.

▶ **Definition 5.4.** *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. The* side weight *of e is the number of facts in I that involve a and not b, plus the number of facts in I that involve b and not a. 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 5.5.** *Every unbounded query Q has a minimal tight pattern.*

The intuition of how to find tight patterns is relatively straightforward: the only instances without non-leaf edges are intuitively disjoint union of star-shaped patterns. If a query is unbounded, then its validity cannot be determined simply by looking at star patterns (e.g., as Q_1 above), so it must intuitively have a model where performing dissociations iteratively will eventually make the query false, so that we must eventually find a tight edge. We will formalize this intuition below. Once we know that there is a tight edge, then it is simple to argue that we can take one that is minimal in the sense that we require.

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 the 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 are no longer any edges that we can dissociate, i.e., all edges are leaf edges.

▶ **Observation 5.6.** *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. It is sufficient to show that an application of dissociation decreases the number of non-leaf edges by 1. To do so, we consider an instance I with a non-leaf edge e, and show that the dissociation I' of e in I, has $n - 1$ non-leaf edges.

Let us assume $e = \{a, b\}$. The new elements a' and b' in I' are leaf elements, and for any other element of the domain of I', it is a leaf in I' iff it was a leaf in I: this is clear for elements that are not a and b as they occur exactly in the same edges, and for a and b we know that they were not leaves in I (they occurred in $e = \{a, b\}$ and in some other edge), and they are still not leaves in I' (they occur in the same other edge and in $\{a, b'\}$ and $\{b, a'\}$, respectively).

Thus, the edges of I' that are not $\{a, b'\}$ or $\{a', b\}$ are leaf edges in I' iff they were in I. So, in terms of non-leaf edges the only difference between I and I' is that we removed the non-leaf edge $\{a, b\}$ from I and we added the two edges $\{a, b'\}$ and $\{a', b\}$ in I' which are leaf edges because a' and b' are leaves. Thus, we conclude the claim. ◀

Hence, if we start with an instance I and perform an iterative dissociation process, then after n steps (n being the number of non-leaf edges of I), the process terminates and reaches an instance that consists only of leaf edges.

Let us now consider which instances have no non-leaf edges. They are intuitively union of stars, and in particular they homomorphically map to some constant-sized subset of their facts, as will be crucial when we turn back to our unbounded query.

▶ **Proposition 5.7.** *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. We first prove the result for connected instances I. In this case, we define the constant $k'_\sigma := 2^{2\times|\sigma|}$. There are two cases. The first case is when all elements of I are leaves: then, as I is connected, it must consist of a single edge and consists of at most $2\,|\sigma|$, facts; so, taking $I' = I$ and the identity homomorphism concludes the proof. The second case is when I contains a non-leaf element a. In this case, consider all edges $\{a, b_1\}, \ldots, \{a, b_n\}$ incident to a, with $n > 1$, as a is not a leaf. Each of the b_i must be leaves: if some b_i is not a leaf then $\{a, b_i\}$ would be a non-leaf edge because neither a nor b_i would be leaves. We then define an equivalence relation $\sim$ on the b_i by writing $b_i \sim b_j$ if the edges $\{a, b_i\}$ and $\{a, b_j\}$ contain the exact same set of facts (up to the isomorphism mapping b_i to b_j): there are at most k'_σ equivalence classes. The requisite subset of I and the homomorphism can thus be obtained by picking one representative of each equivalence class, keeping the edges incident to these representatives, and mapping each b_i to the chosen representative of its class.

We now extend the proof to instances I that are not necessarily connected. Letting I be such an instance, we consider its connected components $I_1, \ldots, I_m$. Each of these is connected and has no non-leaf edges, so there are subsets $I'_1, \ldots, I'_m$ with $\leq k'_\sigma$ facts each and a homomorphism of each I_i to its I'_i. Now, there are only constantly many instances with $\leq k'_\sigma$ facts up to isomorphism: let k''_σ be their number, and let $k_\sigma := k''_\sigma \times k'_\sigma$. The requisite subinstance and homomorphism is obtained by again picking one representative for each isomorphism equivalence class of the I'_i (at most k''_σ of them, so at most k_σ facts in total) and mapping each I_i to the I'_j which is the representative for I'_i. This concludes the proof. ◀

We can now prove our theorem by appealing to the unboundedness of the query, which we rephrase as having *minimal models* of arbitrarily large size.

▶ **Definition 5.8.** *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 since it is closed under homomorphisms. Conversely, if Q is equivalent to a UCQ, then it has finitely many minimal models which are obtained directly from the UCQ disjuncts, by eliminating the ones that are redundant. This obviously means the following:

▶ **Observation 5.9.** *Any unbounded query Q has a minimal model I with $> k$ facts for any $k \in \mathbb{N}$.*

We are ready to show Theorem 5.5:

Proof of Theorem 5.5. We first show the first part of the claim: any unbounded query has a tight pattern. Let k_σ be the bound from Proposition 5.7. By Observation 5.9, let I_0 be a minimal model with $> k_\sigma$ facts. Set $I := I_0$ and let us apply an iterative dissociation process: while I has edges that are non-leaf but not tight, perform the dissociation, yielding I', and let $I := I'$.

By Observation 5.6 this process must terminate after at most n steps, where n is the number of non-leaf edges of I_0. Let I_n be the result of this process. If I_n has a non-leaf edge e which is tight, then we are done as we have found a tight pattern (I, e). Otherwise, let us reach a contradiction.

First notice that, throughout the rewriting process, it has remained true that I is a model of Q. Indeed, if performing a dissociation breaks this, then the dissociated edge was tight. Also notice that, throughout the rewriting, it has remained true that I has a homomorphism to I_0: it is true initially, with the identity homomorphism, and when we dissociate I to I' then I' has a homomorphism to I defined by mapping the fresh elements a' and b' to the original elements a and b and as the identity otherwise. Hence, I_n is a model of Q having a homomorphism to I_0.

Note that I_n has no non-leaf edges. Thus, Proposition 5.7 tells us that I_n admits a homomorphism to some subset I'_n of size at most k_σ. This homomorphism witnesses that I'_n also satisfies Q. But now, I'_n is a subset of I_n so it has a homomorphism to I_n, which has a homomorphism to I_0. Let $I'_0 \subseteq I_0$ be the image of I'_n by the composed homomorphism. It has at most k_σ facts, because I'_n does; and it satisfies Q because I'_n does. But as I_0 had $> k_\sigma$ facts, I'_0 is a strict subset of I_0 that satisfies Q. This contradicts the minimality of I_0, which leads us to conclude the first part of the claim.

It only remains to show the second part of the claim: there exists a minimal tight pattern. We already concluded that Q has a tight pattern (I, e), and e has some finite weight $w_1 > 0$ in I. Pick the minimal $0 < w'_1 \leq w_1$ such that Q has a tight pattern (I', e') where e' has weight w'_1. Now, e' has some finite side weight $w_2 \geq 2$ in I'. Pick the minimal $2 \leq w'_2 \leq w_2$ such that Q has a tight pattern (I'', e'') where e' has weight w'_1 and has side weight w'_2. We can then see that (I'', e'') is a minimal tight pattern by minimality of w'_1 and w'_2. This concludes the proof. ◀

6 Hardness with Tight Iterable Edges

In this section, we conclude the proof of our main result (Theorem 3.3) by showing that a minimal tight pattern which is iterable can be used to show hardness. We first comment that this part of the proof is indeed necessary, i.e., there are some unbounded queries that were not covered by Theorem 4.6.

▶ **Example 6.1.** Consider the following Datalog program:

- $R(x, y) \to A(y)$
- $A(x), S(x, y) \to B(y)$
- $B(x), S(y, x) \to A(y)$
- $T(x, y), B(x) \to \text{Goal}()$

This program accepts instances containing paths of the form $R(a, a_1), S(a_1, a_2), S^-(a_2, a_3), \ldots, S(a_{2n+1}, a_{2n+2}), T(a_{2n+2}, b)$, so the query is unbounded. However, it has no model with a non-iterable edge. Indeed, in every model the query is made true because of a path of the form above, and we cannot break such a path by iterating an edge (we will obtain either the same path or a longer path).

If we tried to reduce from #PP2DNF for this query, as in the proof of Theorem 4.6, 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 TID 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.

To conclude the proof of Theorem 3.3, we show the following result:

▶ **Theorem 6.2.** *For every query Q, if we have a minimal tight pattern (I, e) where the edge e is iterable, then* $\mathrm{PQE}(Q)$ *is #P-hard.*

Observe that this result indeed suffices to conclude the proof of our main result (Theorem 3.3).

Proof of Theorem 3.3. Let Q be an unbounded UCQ^∞. If we have a model of Q with a non-iterable edge, then we conclude by Theorem 4.6 that $\mathrm{PQE}(Q)$ is #P-hard. Otherwise, by Theorem 5.5, 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 6.2. ◀

Hence, in this section, we show Theorem 6.2. The idea of the proof is again simple: if we have an iterable edge, then all iterates of the edge satisfy the query, so we can use it to reduce from the U-ST-CON problem: we code the input undirected graph using probabilistic edges so that possible worlds of the undirected graph with a source-to-target path feature some iterate of the instance, and possible worlds without any such path do not. This latter part of the proof will again be the most challenging, and we will show it by leveraging the tightness and minimality of the pattern to argue that possible worlds without a path have a homomorphism to a so-called *fine dissociation* of the edge of the pattern, which we argue cannot satisfy the query.

Before we start presenting our reduction, let us define this notion of fine dissociation: it is relative to a neighborhood choice *and* to a fact of the chosen edge, which is intuitively because we can only assign probabilities to single facts:

▶ **Definition 6.3.** *Let I be an instance, let $e = (u, v)$ be a non-leaf edge in I, let $F_l = R_l(l, u), F_r = R_r(v, r)$ be a neighbor choice of e in I, and let F_m be a fact of the edge e. The result of performing the* fine dissociation *of e in I relative to F_l, F_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 relative to domain* $\mathrm{dom}(I) \setminus \{u, v\}$
- Copy incident facts F_l and F_r: *Add F_l and F_r to I'*
- Copy other facts incident to u: *For every fact $F'_l = R'_l(l', u)$ with $l' \neq v$ and $F'_l \neq F_l$, add the fact $R'_l(l', u')$ to I'.*
- Copy other facts incident to v: *For every fact $F'_r = R'_r(v, r')$ with $r' \neq u$ and $F'_r \neq F_r$, add the fact $R'_r(v', r')$ to I'.*
- Create the copies of e: *For each fact $F'_m = R(u, v)$ of e in I:*
  - *add the facts $R(u, v')$ and $R(u', v)$ to I', and*
  - *if $F'_m \neq F_m$, add the facts $R(u, v)$ and $R(u', v')$ to I'.*

The result of a *fine dissociation* is illustrated in Figure 5. In the case where the edge e is realized by one single fact, then notice that in the dissociation there is no edge $\{u, v\}$ left. If

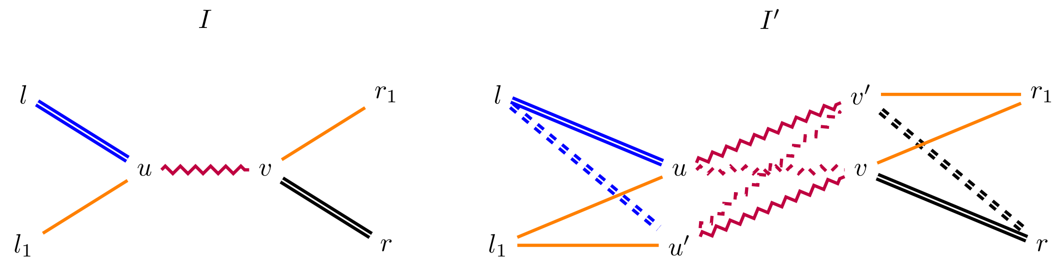


Figure 5 Example of the fine dissociation of an edge in an instance (from I to I'). Lines represent edges. The iterated edge is the zigzag purple line; its dashed version in I' is the same edge without the fact F_m; the double blue and black lines are the edges of F_l and F_r respectively; their dashed versions in I' are the same edges without the facts F_l and F_r respectively.

there are more facts that realize in e, however, the result is more complicated because of the edges $\{u, v\}$ and $\{u', v'\}$. Intuitively, fine dissociation is a more complicated variant of dissociation where (like iteration) the new elements are connected to all incident facts to u and v not in the edge e, where (like dissociation) we create two copies of e that are not connected, and where (unlike dissociation or iteration) we also create two copies of e with one less fact.

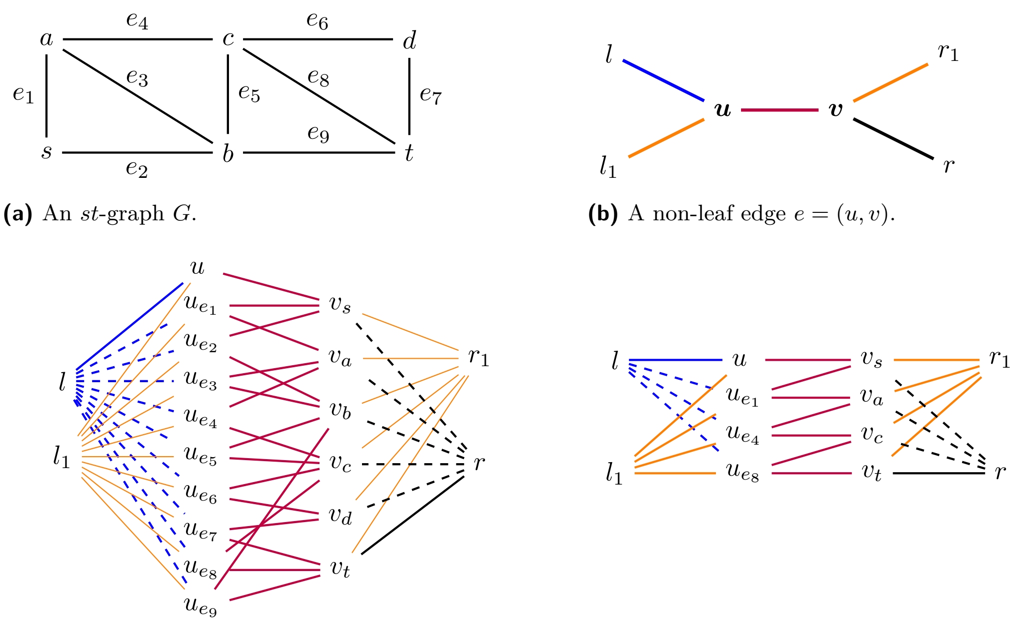
We will study later in the proof the properties of the fine dissociation. For now, let us start the proof of Theorem 6.2 by describing the coding: given an st-graph, i.e., an undirected graph G with source s and target t, we define a probabilistic instance in polynomial time such that there is a bijective correspondence between the possible worlds of the st-graph and the possible worlds of the probabilistic instance.

▶ **Definition 6.4.** *Let $G = (W, C, s, t)$ be an undirected graph with source and sink. The* coding *of G relative to an instance I, to a non-leaf edge $e = (u, v)$ of I, to a neighbor choice $N = F_l, F_r$ with $F_l = R_l(l, u)$ and $F_r = R_r(v, r)$, and to a fact F_m of e, is a probabilistic instance $\mathcal{I} = (J, \pi)$. The domain of J is that of I plus a fresh element u_c for each $c \in C$ and a fresh element v_w for each $w \in W$; we identify v_t to v. The facts of J and the mapping π are defined by the following steps:*

- Copy non-incident facts: *Initialize J as the induced subinstance of J_1 relative to the domain* $\mathrm{dom}(I) \setminus \{u, v\}$
- Copy incident facts F_l and F_r: *Add the facts $R_l(l, u)$ and $R_r(v, r)$ to J*
- Copy other facts incident to u: *For every fact $F'_l = R'_l(l', u)$ with $l' \neq v$ and $F'_l \neq F_l$, add the facts $R'_l(l', u_c)$ to J for each $c \in C$.*
- Copy other facts incident to v: *For every fact $F'_r = R'_r(v, r')$ with $r' \neq u$ and $F'_r \neq F_r$, add the facts $R'_r(v_w, r')$ to J for each $w \in W$.*
- Create copies of e: *we create the following copies of e (i.e., of all its facts) in J:*
  - (u, v_s)
  - (u_c, v_a) *and* (u_c, v_b) *for each edge* $c = \{a, b\}$ *of* C

Finally, we define 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 reduction is exemplified in Figure 6. As this coding is somewhat complicated, and the reason for this may not be apparent, let us intuitively explain. The edges are coded by paths of length 2 because the source graph to the reduction is undirected but the facts on edges are directed, so we symmetrize by having two copies of the edge in opposite directions so that we can traverse them in both ways. (The choice that we make in how to orient the



(a) An st-graph G.

(b) A non-leaf edge $e = (u, v)$.

(c) Coding of the graph G

(d) An illustration of a successful s-t path in the coding.

Figure 6 Example of a coding based on an st-graph G shown in Fig 6(a). An edge e (Fig 6(b)) is used in the encoding of G, and the result of the coding is shown in Fig 6(c). Each st-path in G is reflected in the coding as an iterate of e, e.g., in Figure 6(d).

edges, i.e., the choice of which $w \in c$ we pick when defining π, has no impact in how the edges can be traversed when their probabilistic fact is kept; but it has an impact in how the edge looks like when the probabilistic fact is missing. Specifically, it is the reason why there are two copies of e with one missing fact in the fine dissociation, as will later become clear.)

It is clear that the coding is in polynomial time. Let us now define the function ϕ relating the possible worlds of the connected graph G to those of the probabilistic instance (J, π). For each edge $c \in C$ we keep the probabilistic fact incident to u_c if c is kept. It is obvious that this correspondence is bijective and that all possible worlds have probability $0.5^{|C|}$.

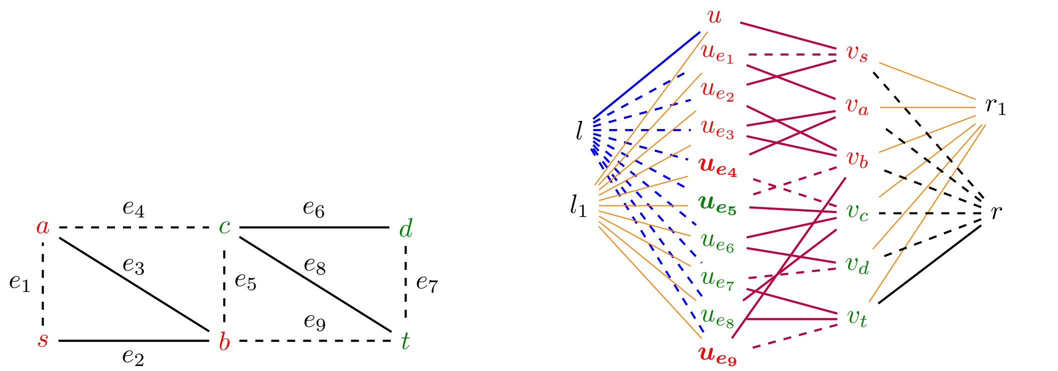
We can now state the following result for the coding, which is again independent from the query.

▶ **Proposition 6.5.** *Let the probabilistic instance $\mathcal{I} = (J, \pi)$ be the coding of an undirected st-graph G relative to an instance I, a non-leaf edge e of I, a neighbor choice N, and a fact F_m realizing e, as described in Definition 6.4. Let ϕ be the bijective function from the possible worlds of G to those of $\mathcal{I}$ defined above. Then the following statements hold:*

1. *For any possible world ω of G where there is a path from s to t of length n, $\phi(\omega)$ has a homomorphism* from $I_{e,N}^{n+1}$.
2. *For any possible world ω of G where there is no path from s is to t, $\phi(\omega)$ has a homomorphism* to *the result of finely dissociating e in I relative to N and F_m.*

Proof. As before, we start with the easier forward direction (1), and then prove the backward direction (2).

1. Consider a witnessing path $s = w_1, \ldots, w_n = t$ in the possible world of G, and assume without loss of generality that the path is simple, i.e., it traverses each vertex at most once. We claim that $\phi(\omega)$ actually has a subinstance isomorphic to $I_{e,N}^{n+1}$. See Figure 6(d) for an



(a) A possible world of G with no s–t path (dashed edges are the ones that are not kept), the left and right vertices in the cut are colored.

(b) Possible world of the coding for the possible world of G at the left. Copies of e are dashed when they have lost one fact. Vertices u_{e_i} corresponding to edges of the cut are in bold, and these vertices are colored depending on their image. The vertex colors describe a homomorphism to the fine dissociation

Figure 7 Illustration of a possible world (Figure 7(a)) of the graph G from Figure 6(a), and the corresponding possible world (Figure 7(b)) of the coding (Figure 6(c)) which has a homomorphism to the fine dissociation (Figure 5)

illustration of a possible $I^{n+1}_{e,N}$. To see why this is true, we take as usual the facts of $\phi(\omega)$ that do not involve any copy of u or v and keep them as-is, because they occur in $\phi(\omega)$ as they do in $I^{n+1}_{e,N}$.

We start by taking the one copy of F_l leading to u and the copy of e leading to v_s. We now follow the path which gives a path of copies of e: for each edge $c = \{w_j, w_{j+1}\}$ of the path, we have two successive copies of e between v_{w_j} and u_c, and between u_c and $v_{w_{j+1}}$. Note that, as the path uses edge c, it was kept in the possible world of G under consideration, so all the copies of e in question have all their facts, i.e., neither of the copies of F_m can be missing. The assumption that the path is simple ensures that we do not visit the same vertex multiple times. After traversing these $2i$ copies of e in alternating directions, we reach $v_t = v$, and finally we use the fact F_r which is incident to b. So, we have indeed found a subinstance of $\phi(\omega)$ which is isomorphic to $I^{n+1}_{e,N}$.

2. Let us write J' in place of $\phi(\omega)$. Let us denote by I' the result of finely dissociating in I the edge e relative to the neighbor choice N and the fact F_m. Suppose that $e = (u, v)$ and let us show that J' has a homomorphism to I' depicted in Figure 5. See Figure 7(b) for an example of such a possible world, and Figure 7(a) for the corresponding possible world of G.

We use the fact that, as the possible world ω of G has no path from s to t, there is an *s,t-cut of ω*, i.e., a function ϕ mapping each vertex of G to either L or R such that s is mapped to L, t is mapped to R, and for every edge $\{x, y\}$ such that $\phi(x) \neq \phi(y)$ then the edge was not kept in G'. See Figure 7(a) for an illustration.

We map u in J' to u in I' and v_s to v, which maps the copy of e between u and v_s in J' to a copy of e in I'. Now observe that we can map to v' in I' all the nodes v_w such that $\phi(w) = \text{L}$, including v_s. The edges between these nodes in J', whether they were kept in ω or not, are mapped by going back-and-forth on the edge (u, v') in I'. In the same way we can map to v in I' all the nodes v_w such that $\phi(w) = \text{R}$, including v_t and all edges between these nodes, going back-and-forth on edge (u', v) in I'.

We must still map the edges of the cut, i.e., edges $c = \{x, y\}$ such that $\phi(x) = \text{L}$ and $\phi(y) = \text{R}$. In J', these edges give rise to two edges (u_c, v_x) and (u_c, v_y), one of which is a copy of e and the other one is a copy of e with the fact F_m missing – which one is which depends on the arbitrary orientation choice that we made when defining π. Depending on the case, we map u_c either to u or to u' so that the two incident edges to u_c are mapped in I' either to (u, v') (a copy of e) and (u, v) (a copy of e minus F_m), or to (u', v') (a copy of e minus F_m) and (u', v) (a copy of e). Thus, we have explained how we map the copies of u and v, the copies of e (including the ones without F_m), and the two facts F_l and F_r.

As usual we have not discussed the facts that do not involve a copy of u or v in J', or the facts that involve one of them and are not facts of e, F_l, or F_r, but these are found in I' in the same way that they occur in J' (noting that we have only mapped copies of u to copies of u, and copies of v to copies of v). This concludes the definition of the homomorphism and concludes the proof. ◀

In a nutshell, Proposition 6.5 establishes via (1) a connection between possible worlds of G and the possible worlds of $\mathcal{I}$: if there exists a path of a certain length in a possible world of G, then the corresponding possible world of $\mathcal{I}$ admits a homomorphism from $I^{n+1}_{e,N}$. Thus, analogously to Section 4, we can immediately relate (1) to query satisfaction, because we know that the edge e is iterable. We may hope to have a dual correspondence via (2), but notice that Proposition 6.5 asserts only the existence of an homomorphism to the result of the fine dissociation process, and we do not yet know the status of fine dissociation relative to the query. The following lemma achieves this.

▶ **Lemma 6.6.** *Let Q be a query and (I, e) be a minimal tight pattern for Q. Let F_l, F_r be an arbitrary neighbor choice for e, and F_m be an arbitrary fact of e. Then, the result of the fine dissociation of e in I relative to F_l, F_r, F_m does not satisfy the query Q.*

This proof is somewhat technical. The intuition is that we rely on the minimality of the edge e to argue that any edge with a smaller weight, or with the same weight and a smaller side weight, cannot be tight, so can be dissociated without changing the status of the query. Specifically, we first dissociate the edges with F_m missing, as their weight is less (the dashed double edges in Figure 5), and we then get rid of the dissociated copies by mapping them into the other copies of e. Then, the resulting copies of e have a smaller side weight, and can be dissociated also, so that we reach something having a homomorphism to the (non-fine) dissociation of e. We can then conclude, because e is tight. The process of the proof is illustrated as Figure 8.

Proof of Lemma 6.6. Fix the query Q, the minimal tight pattern (I, e), and the choice of F_l, F_m, and F_r. Assume by way of contradiction that the result I_1 of the process satisfies the query Q. Consider now the edges $e''_1 = \{u, v\}$ and $e'_1 = \{u', v'\}$: their weight in I_1, by construction, is one less than the weight of e. Hence, as (I, e) is minimal, by Definition 5.4 we know that each of these edges cannot be tight: if one of these edges were, say e'_1, then (I_1, e'_1) would be a tight pattern with e'_1 having a strictly smaller weight, which is impossible. Thus, as we assumed that I satisfies the query, it must mean that we can dissociate e'_1, then e''_1 using the dissociation process of Definition 4.3. The resulting instance I_2 (see Figure 8) still satisfies the query: if it did not, it would mean that one of the dissociations has make the query false, which would imply that we have found a tight pattern of strictly smaller weight. (Strictly speaking, if we dissociate e_1 and then e'_1 in the result, and the query is false, then either e_1 was tight in I_1, or e'_1 was tight in the intermediate result, and anyway this is a contradiction because the weight of e'_1 does not change in the intermediate step.)

Now let us consider the structure of I_2: say we have first dissociated $e''_1 = \{u, v\}$ to remove this edge, renamed u and v to u_1 and v_1, created u_2 and v_2, and add back copies of

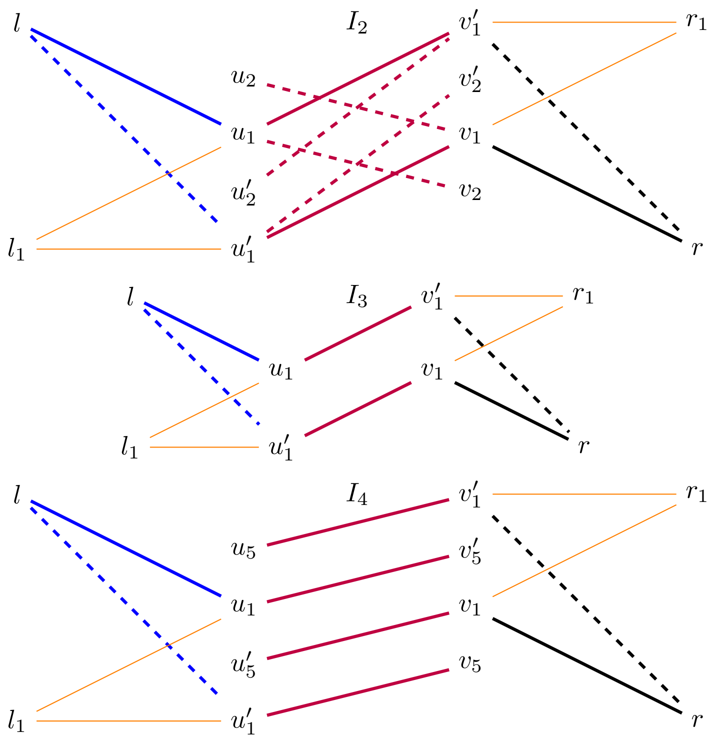


Figure 8 Illustration of the proof of Lemma 6.6, with I and I' from Figure 5, and I_5 being on Figure 4

the edge from u_1 to v_2 and from u_2 to v_1. Next, we have dissociated $e'_1 = \{u', v'\}$ (note that these are different vertices), removed e'_1, renamed u' and v' to u'_1 and v'_1, created u'_2 and v'_2, and created copies of e'_1 from u'_1 to v'_2 and from u'_2 to v'_1. Note that u_2, v_2, u'_2, v'_2 are leaf vertices only occurring on the copies of the dissociated edges (the edges with the same facts as e except F_m). We have copies of the edge e (from the fine dissociation) from u_1 to v'_1 and from u'_1 to v_1.

Observe now that we can map the leaves to other vertices to define a homomorphism:

- we map u_2 to u'_1 and map the edge $\{u_2, v_1\}$ to the edge $\{u'_1, v_1\}$ whose facts are those of e, so a superset of the facts;
- we map v_2 to v'_1 and map the edge $\{u_1, v_2\}$ to $\{u_1, v'_1\}$;
- we map u'_2 to u_1 and map the edge $\{u'_2, v'_1\}$ to the edge $\{u_1, v'_1\}$;
- we map v'_2 to v_1 and map the edge $\{u'_1, v'_2\}$ to the edge $\{u'_1, v_1\}$.

Thus, the resulting instance I_3 (see Figure 8) still satisfies the query. Relative to I_1, it is the result of replacing u with copies u_1, u'_1, and v with copies v_1, v'_1, and having one copy of e from u'_1 to v_1 and from u_1 to v'_1, with all facts incident to u and v replicated on u_1, u'_1 and v_1, v'_1, except F_l and F_r which only involve u_1 and v_1. In other words, the instance I_3 is isomorphic to the result I_1 of the fine dissociation (Figure 5), except that we have not created copies of e without F_m between u_1 and v_1 and between u'_1 and v'_1. We have justified, from our assumption that I_1 satisfies the query, that I_3 also does.

Let us now use the second minimality criterion on I_3 on the edges $e_4 = \{u_1, v'_1\}$ and $e'_4 = \{u'_1, v_1\}$ to simplify the instance further. The weight of these edges is the same as that of e, but their side weight is smaller: indeed, u_1 has exactly as many incident facts as u did, and v_1' has the same number as v except that F_r is missing, so the side weight of e_4 is indeed smaller. The same holds for e_4' because v_1 has exactly the same incident facts as v and u_1' has the same as u except F_l. This means that these edges are not tight, as otherwise it would contradict the second criterion in Definition 5.4. Thus, we can dissociate one and then the other, and the query will still be satisfied. Say we first dissociate e_4 and then e_4', and call I_4 the result. We create u_5 and v_5' and replace e_4 by copies from u_1 to v_5' and from u_5 to v_1', with v_5' and u_5 being leaves; and we create u_5' and v_5 and replace e_4' by copies from u_1' to v_5 and from u_5' to v_1, with v_5 and u_5' being leaves. The resulting instance I_4 (see Figure 8) still satisfies the query.

Now, we can merge back vertices to reach an instance I_5 isomorphic to the dissociation of e in I, which will yield our contradiction. Let us map u_1' to u_1 and v_5 to v_5': this defines a homomorphism because the edge $\{u_1', v_5\}$ can be mapped to $\{u_1, v_5'\}$, this was the only edge involving v_5, and all other facts involving u_1' have a copy involving u_1 by definition of the fine dissociation. Let us also map v_1' to v_1 and v_5 to v_5' in the same fashion, which is correct for exactly the same reason. The resulting instance I_5 still satisfies the query. Now observe that I_5 is isomorphic to the result of the (non-fine) dissociation of e in I (Figure 4): we have added two leaves u_5' and v_5', the vertices u_1 and v_1 indeed correspond to u and v, we have removed the edge from u to v and replaced it by copies from u_1 to v_5' and from u_5' to v_1.

Thus we have deduced that dissociating e in I yields an instance that satisfies the query. But as (I, e) was a tight pattern, this is impossible, so we have reached a contradiction and the proof is finished. ◀

Given Proposition 6.5, and Lemma 6.6, we can now prove Theorem 6.2.

Proof of Theorem 6.2. Fix the query Q and the minimal tight pattern (I, e). By definition, e is then a non-leaf edge: pick an arbitrary neighborhood choice N and fact F_m of e. We show #P-hardness of PQE(Q) by reducing from U-ST-CON (Definition 2.1). Given an undirected graph G, we use Proposition 6.5 to compute in PTIME a probabilistic instance $\mathcal{I}$. As in the proof of Theorem 4.6, what matters is to show that (1.) in the forward case the query holds on $\phi(G')$, and (2.) in the backward case the query does not hold in $\phi(G')$.

For (1.), the result follows from the fact that the query Q is closed under homomorphisms, and the edge e was assumed to be iterable relative to N (Definition 4.3), so the iterates satisfy Q and $\phi(G')$ also does. For (2.), we know by Lemma 6.6 that the result of the fine dissociation does not satisfy the query, so $\phi(G')$ does not satisfy it either. ◀

This concludes the proof of Theorem 6.2, and together with Theorem 4.6, our main theorem (Theorem 3.3) is established.

7 Outlook and Conclusions

We have shown that, on arity-two signatures, for any unbounded UCQ$^\infty$, the probabilistic query evaluation problem (PQE) is #P-hard. This leads to a dichotomy on PQE for all UCQ$^\infty$ queries: either they are unbounded and PQE is #P-hard, or they are bounded and the dichotomy by Dalvi and Suciu applies. This result thus classifies the complexity of PQE for all query languages that are in UCQ$^\infty$, i.e., are closed under homomorphism.

Our result captures many natural query languages, in particular disjunctive Datalog over binary signatures and important fragments such as regular path queries. Similarly, we conclude a set of classification results for PQE on a rich class of ontology-mediated queries, namely, those that are definable as UCQ$^\infty$ (which in particular disallows negation). In particular, our result implies a dichotomy of OMQs that use the negation-free fragment of $\mathcal{ALCHI}$, because any such OMQ can be expressed in monadic disjunctive Datalog over binary signatures (by Theorem 6 of [6]). The same thus holds about subclasses of this logic, e.g., $\mathcal{ELHI}$, and $\mathcal{ELI}$ as in [28].

There are two natural directions in which to extend our result. The first would be to study queries that are *not* homomorphism-closed, e.g., queries with disequalities, or even with negation (so non-monotone queries). We believe that this would require significantly different techniques, because already for UCQs we are not aware of a full dichotomy results beyond the partial results achieved in [21].

The second natural direction is to show a corresponding result on general signatures, without the arity restriction. Our conjecture would be that the corresponding result is also true, i.e., that PQE is #P-hard for any unbounded UCQ^∞ even on non-binary signatures. We believe that much of the proof material can be adapted, but what we do not know how to extend is the definitions of the operations (dissociation, fine dissociation, iteration). Indeed, in the binary case, when we modify an edge, all incident facts only touch one element of the edge, in particular their intersections with the edge cannot overlap. In the general case, this is not true, and complicated intersection patterns may prevent us from creating copies of edges freely. The main roadblock in this sense would be to propose a notion of dissociation and minimality for which the analogue of the results in Section 5 would hold (in particular, rewriting with it should terminate). We do not yet see how this could be done, and leave this to future work.

Other than that, an intriguing question is whether our hardness result could be shown for the *unweighted* case of the probabilistic query evaluation problem, where all edges must carry probability 0.5. However, the complexity of this unweighted problem is poorly understood already in the case of UCQs.

Finally, an ambitious question is whether our dichotomy on unbounded queries could be extended to the complexity of other problems than PQE, e.g., the problem of non-probabilistic query evaluation, counting the number of matches, or approximating the answers to PQE. This is challenging, however, as "simpler" problems such as non-probabilistic query evaluation make it more difficult to show "hardness", and we believe that the complexity picture there could be very different than our result showing hardness of all unbounded queries. In particular, in our case, it is comparatively easy to handle queries with a non-iterable edge as we can then show #P-hardness directly (Section 4), but the analogue of this would not hold for non-probabilistic problems. A related question is to understand if our results relate to dichotomies for the CSP problem and its counting variants #CSP [9], but CSP formulations typically do not ask about counting subinstances: the closest analogue that we know is #SUB [15] which asks about counting the number of subgraphs of an input graph that are *isomorphic* (not *homomorphic*) to a query graph.

References

1 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of databases*. 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 Franz Baader, Diego Calvanese, Deborah L McGuinness, Daniele Nardi, and Peter F Patel-Schneider, editors. *The Description Logic handbook*. 2007.

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

5 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.

6 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.

7 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.

8 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.

9 Andrei A. Bulatov. The complexity of the counting constraint satisfaction problem. *J. ACM*, 60(5):34:1–34:41, 2013.

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

11 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.

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

13 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.

14 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.

15 Radu Curticapean and Dániel Marx. Complexity of counting subgraphs: Only the boundedness of the vertex-cover number counts. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 130–139. IEEE, 2014.

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 Erich Grädel, Yuri Gurevich, and Colin Hirsch. The complexity of query reliability. In *Proceedings of the 17th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS-98)*, pages 227–234. ACM, 1998.

25 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.

26 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.

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

28 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.

29 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.

30 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 *Lecture Notes in Computer Science*, pages 326–340, 2008.

31 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.

32 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.

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

34 Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. *Probabilistic Databases*, volume 3. 2011.

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

A Extension to signatures with unary and binary predicates

In Section 2, we claim that our results extend to the case of a signature featuring unary and binary predicates. We now justify this claim formally by showing the analogue of Theorem 3.3 for signatures with relations of arity 1 and 2.

▶ **Theorem A.1.** *Let Q be an unbounded* UCQ^∞ *on a signature with relations of arity 1 and 2. Then,* $\mathrm{PQE}(Q)$ *is #P-hard.*

Proof. Fix the signature σ and query Q. Let σ' be the arity-two signature constructed from σ by replacing each relation R of arity 1 by a relation R' of arity 2. Let ϕ be a function mapping any instance I of σ' to the instance $\phi(I)$ obtained by replacing each fact $R'(a, b)$ by the fact $R(a)$. (Note that we can create the same fact $R(a)$ because of multiple facts of the form $R'(a, b)$.) Let us define Q' to be the query on σ' which is satisfied precisely on the instances I such that $\phi(I)$ satisfies Q.

We first claim that Q' is closed under homomorphisms. Indeed, letting I and I' be two σ'-instances such that I has a homomorphism to I', it is clear that a restriction of this function defines a homomorphism from $\phi(I)$ to $\phi(I')$, so that if I satisfies Q' then $\phi(I)$ satisfies Q, thus $\phi(I')$ satisfies Q because Q is closed under homomorphisms, so I' satisfies Q'.

We next claim that Q' is unbounded. Indeed, assuming by way of contradiction that Q' is equivalent to a UCQ, let Q'' be the UCQ on σ obtained by rewriting each disjunct of Q' to replace each atom of the form $R'(x, y)$ by $R(x)$. We claim that, for any σ-instance I, there is a σ'-instance I' such that $\phi(I') = I$, and I satisfies Q'' iff I' satisfies Q. Indeed, take I and define I' by replacing each unary fact $R(a)$ of I by the facts $R'(a, b)$ for each possible b of the domain of I. It is clear that $\phi(I') = I$, it is clear that if I' satisfies Q' then the projection of a match to I satisfies Q'', and conversely if I satisfies Q'' then there is a match of a disjunct Q'' in I, which we can expand to a match of the same disjunct of Q'' in I'. We have thus shown that Q and Q'' are equivalent, because whenever a σ-instance I satisfies Q then the instance I' is such that $\phi(I') = I$, so I' satisfies Q', thus I satisfies Q''; and conversely if I satisfies Q'' then by definition I' satisfies Q', hence $\phi(I') = I$ satisfies Q. Now, Q is then equivalent to Q'' which is a UCQ, and this contradicts the unboundedness of Q.

By Theorem 3.3, we know that PQE(Q') is #P-hard, It only remains to show how to reduce in PTIME from PQE(Q') to PQE(Q), concluding the proof. Consider a TID $\mathcal{I}' = (I', \pi')$ with I' on σ, and let us define in PTIME the TID $\mathcal{I} = (I, \pi)$ with $I := \phi(I')$, and π giving to each σ-fact of arity two in I the same probability as in I', and giving to each σ-fact $R(a)$ of arity one in I the probability $1 - \prod_{F=R'(a,b)\in I'}(1 - \pi'(F))$. We can compute this quantity in polynomial time. Now, it is clear that ϕ defines a bijection between the possible worlds of I' and the possible worlds of I' where each unary fact F is repeated with a multiplicity equal to the number of facts of I' that project down to F; and that this bijection is probability-preserving. This establishes that the reduction is correct, and concludes the proof. ◀