Logical Methods in Computer Science
Volume 18, Issue 1, 2022, pp. 2:1–2:31
https://lmcs.episciences.org/

Submitted Jan. 08, 2021
Published Jan. 07, 2022

THE DICHOTOMY OF EVALUATING HOMOMORPHISM-CLOSED QUERIES ON PROBABILISTIC GRAPHS

ANTOINE AMARILLI [a] AND İSMAİL İLKAN CEYLAN [b]

[a] LTCI, Télécom Paris, Institut Polytechnique de Paris, France
e-mail address: antoine.amarilli@telecom-paris.fr

[b] Department of Computer Science, University of Oxford, United Kingdom
e-mail address: ismail.ceylan@cs.ox.ac.uk

Abstract. We study the problem of *query evaluation on probabilistic graphs*, namely, tuple-independent probabilistic databases over signatures of arity two. We focus on the class of queries closed under homomorphisms, or, equivalently, the *infinite* unions of conjunctive queries. Our main result states that the probabilistic query evaluation problem is #P-hard for all *unbounded* queries from this class. As *bounded* queries from this class are equivalent to a union of conjunctive queries, they are already classified by the dichotomy of Dalvi and Suciu (2012). Hence, our result and theirs imply a complete data complexity dichotomy, between polynomial time and #P-hardness, on evaluating homomorphism-closed queries over probabilistic graphs. This dichotomy covers in particular all fragments of infinite unions of conjunctive queries over arity-two signatures, such as *negation-free (disjunctive) Datalog*, *regular path queries*, and a large class of *ontology-mediated queries*. The dichotomy also applies to a restricted case of probabilistic query evaluation called *generalized model counting*, where fact probabilities must be 0, 0.5, or 1. We show the main result by reducing from the problem of counting the valuations of positive partitioned 2-DNF formulae, or from the source-to-target reliability problem in an undirected graph, depending on properties of minimal models for the query.

1. Introduction

The management of *uncertain and probabilistic data* is an important problem in many applications, e.g., automated knowledge base construction [DGH$^+$14, HSBW13, MCH$^+$15], data integration from diverse sources, predictive and stochastic modeling, applications based on (error-prone) sensor readings, etc. To represent probabilistic data, the most basic model is that of tuple-independent *probabilistic databases (TIDs)* [SORK11]. 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

Key words and phrases: Probabilistic graphs, probabilistic databases, probabilistic query evaluation, infinite unions of conjunctive queries, data complexity dichotomy, #P-hardness.

[*] This paper has appeared as a conference paper at ICDT 2020 with the title: "A Dichotomy for Homomorphism-Closed Queries on Probabilistic Graphs" [AC20].

LOGICAL METHODS IN COMPUTER SCIENCE DOI:10.46298/LMCS-18(1:2)2022

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. We write PQE(Q) to denote the problem of PQE relative to a *fixed* query Q.

Dalvi and Suciu [DS12] obtained a dichotomy for evaluating *unions of conjunctive queries (UCQs)* on tuple-independent probabilistic databases. Their dichotomy is measured in *data complexity*, i.e., as a function of the input TID and with the query being fixed. More specifically, they have shown that, for a given UCQ Q, PQE(Q) is either in polynomial time, or it is #P-hard. In the terminology of Dalvi and Suciu, a UCQ Q is called *safe* if PQE(Q) can be computed in polynomial time, and it is called *unsafe* otherwise. This dichotomy result laid the foundation for many other studies on the complexity of probabilistic query evaluation [ABS16, CDV21, FO16, JL12, OH08, OH09, RS09].

Despite this extensive research on TIDs, there is little known about probabilistic query evaluation for monotone query languages beyond UCQs. In particular, only few results are known for languages featuring *recursion*, which is an essential ingredient in many applications. For instance, it is unknown whether PQE admits a dichotomy for Datalog queries, for regular path queries, or for ontology-mediated queries [Cey17]. The main motivation of this paper is thus to obtain a fine-grained classification for the complexity of probabilistic query evaluation relative to these query languages. Our focus is on a large class of queries beyond first-order: we study the queries that are *closed under homomorphisms*. We denote the class of such queries by UCQ$^\infty$ as they are equivalent to *infinite* unions of conjunctive queries. We distinguish between *bounded* UCQ$^\infty$ queries, which are logically equivalent to a UCQ, and *unbounded* UCQ$^\infty$ queries, which cannot be expressed as a UCQ. Notably, UCQ$^\infty$ captures (negation-free) disjunctive Datalog, regular path queries, and a large class of ontology-mediated queries.

Our focus in this work is on *probabilistic graphs*, i.e., probabilistic databases where all relations have at most *arity two*. Data models based on binary relations are quite common in knowledge representation. Knowledge graphs such as NELL [MCH$^+$15], Yago [HSBW13] and Knowledge Vault [DGH$^+$14] are solely based on binary relations, and are widely used for tasks such as information and relation extraction [MBSJ09], rule mining [DRDT$^+$15], and knowledge graph completion [BUGD$^+$13]. To encode more sophisticated domain knowledge *ontologies* are employed. Ontologies are prominently formulated in description logics [BCM$^+$07], which is a family of languages, defined over unary relations (i.e., concepts) and binary relations (i.e., roles). In these and similar contexts, we want to evaluate UCQ$^\infty$ queries on (graph-structured) data, while taking into account the uncertainty of the data. Therefore, we study the complexity of probabilistic query evaluation on probabilistic graphs, and ask whether evaluating UCQ$^\infty$ queries admits a data complexity dichotomy.

The main result of this paper is that PQE(Q) is #P-hard for *any unbounded* UCQ$^\infty$ *query* on probabilistic graphs. Our result thus implies a dichotomy on PQE for UCQ$^\infty$ over such graphs: as *bounded* UCQ$^\infty$ queries are equivalent to UCQs, they are already classified by Dalvi and Suciu, and we show that all other UCQ$^\infty$ queries are unsafe, i.e., the PQE problem is #P-hard for them. Of course, it is not surprising that *some* unbounded queries in UCQ$^\infty$ are unsafe for similar reasons as unsafe UCQs, but the challenge is to show hardness for *every* unbounded UCQ$^\infty$ query: we do this by leveraging model-theoretic properties of this query class.

The proof consists of two main parts. First, we study UCQ$^\infty$ queries with a model featuring a so-called *non-iterable edge*. For all such queries, we show #P-hardness by reducing from the problem of counting the valuations of positive partitioned 2-DNF formulae (#PP2DNF).

Second, we focus on all other unbounded queries in UCQ$^\infty$, i.e., UCQ$^\infty$ queries with *no* model featuring such a *non-iterable edge*. For these queries, we give a reduction from the source-to-target connectivity problem in an undirected graph (#U-ST-CON). This second reduction is considerably harder and relies on a careful study of minimal models.

This paper is organized as follows. We start by discussing closely related work for probabilistic query evaluation with a particular focus on existing classification results in Section 2. We introduce preliminaries in Section 3, and formally state our result in Section 4. We prove the result in Sections 5–7. We first deal in Section 5 with the case of queries having a model with a non-iterable edge (reducing from #PP2DNF), then argue in Section 6 that unbounded queries must have a model with a minimal tight edge, before explaining in Section 7 how to use this (when the edge is iterable) to reduce from #U-ST-CON. We then present two generalizations of our main result in Section 8. We conclude in Section 9.

2. Related Work

Research on probabilistic databases is a well-established field; see e.g. [SORK11]. The first dichotomy for queries on such databases was shown by Dalvi and Suciu [DS07]: a self-join-free conjunctive query is safe if it is *hierarchical*, and #P-hard otherwise. They then extended this result to a dichotomy for all UCQs [DS12]. Beyond UCQs, partial dichotomy results are known for some queries with negation [FO16], with disequality ($\neq$) joins in the queries [OH08], or with inequality ($<$) joins [OH09]. Some results are known for extended models, e.g., the dichotomy of Dalvi and Suciu has been lifted from TIDs to open-world probabilistic databases [CDV21]. However, we are not aware of dichotomies in the probabilistic database literature that apply to Boolean queries beyond first-order logic, or to queries with fixpoints. Query evaluation on probabilistic databases has also been studied in restricted contexts, e.g., when probabilistic tuples are only allowed to have probability 0.5. This is for instance the focus of the recent paper of Kenig and Suciu [KS21], which we discuss in Section 8.

Query evaluation on probabilistic graphs has also been studied in the context of *ontology-mediated queries* (OMQs) [JL12, BCL17, BCL19]. An OMQ is a composite query that typically consists of a UCQ and an *ontology*. The only classification result on PQE for OMQs beyond first-order-rewritable languages is given for the description logic $\mathcal{ELI}$ [JL12]. This result applies to a class of queries that go beyond first-order logic. Our work generalizes this result (Theorem 6 of [JL12]) by showing hardness for any unbounded UCQ$^\infty$, not just the ones expressible as OMQs based on $\mathcal{ELI}$. Part of our techniques (Section 5) are related to theirs, but the bulk of our proof (Sections 6 and 7) uses new techniques, the need for which had in fact been overlooked in [Jun14, JL12]. Specifically, we identified a gap in the proofs of Theorem 6 of [JL12] and Theorem 5.31 of [Jun14] concerning a subtle issue of "back-and-forth" matches related to the use of inverse roles of $\mathcal{ELI}$. We have communicated this with the authors of [Jun14, JL12], which they kindly acknowledged [JL20]. Our proof thus completes the proof of Theorem 6 in [JL12], and generalizes it to all unbounded UCQ$^\infty$.

3. Preliminaries

In this section, we introduce all technical preliminaries relevant to our study. In particular, we introduce the query languages studied in this paper, and the tuple-independent probabilistic database model. We also discuss briefly the complexity classes relevant to our study, as well as two canonical #P-hard problems which are used later in the reductions.

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., it consists *only* of predicates of arity two. Our results can easily be extended to signatures with relations having predicates of arity one and two, as we show in Section 8.

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

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

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

Every σ-instance I can be seen as a $\sigma^{\leftrightarrow}$-instance consisting of all the σ-facts in I, and all the facts $R^-(b, a)$ for each fact $R(a, b)$ of I. Thus, for a σ-instance I, and for an element $a \in \mathrm{dom}(I)$, we define the set of all $\sigma^{\leftrightarrow}$-facts of the form $F = R(a, b)$ in I as:

$$\begin{aligned} &\{S(a, a) \mid S \in \sigma, S(a, a) \in I\} \\ \cup\ &\{S(a, b) \mid S \in \sigma, b \in \mathrm{dom}(I), S(a, b) \in I\} \\ \cup\ &\{S^-(a, b) \mid S \in \sigma, b \in \mathrm{dom}(I), S(b, a) \in I\}. \end{aligned}$$

If we say that we create a fact $R(a, b)$ for $R \in \sigma^{\leftrightarrow}$, we mean that we create $S(a, b)$ if $R = S$ for some $S \in \sigma$, and $S(b, a)$ if $R = S^-$ for some $S \in \sigma$.

The *Gaifman graph* of an instance I is the undirected graph having $\mathrm{dom}(I)$ as vertex set, and having an edge $\{u, v\}$ between any two $u \neq v$ in $\mathrm{dom}(I)$ that co-occur in some fact of I. An instance is *connected* if its Gaifman graph is connected. We call $\{u, v\}$ an (undirected) *edge* of I, and the facts of I that it *covers* are the σ-facts of I whose domain is a subset of $\{u, v\}$. Note that a fact of the form $R(u, u)$ is covered by all edges involving u. Slightly abusing notation, we say that an *ordered* pair $e = (u, v)$ is a (directed) *edge* of I if $\{u, v\}$ is an edge of the Gaifman graph, and say that it *covers* the following $\sigma^{\leftrightarrow}$-facts of I:

$$\begin{aligned} &\{S(u, u) \mid S \in \sigma, S(u, u) \in I\} \\ \cup\ &\{S(v, v) \mid S \in \sigma, S(v, v) \in I\} \\ \cup\ &\{S(u, v) \mid S \in \sigma, S(u, v) \in I\} \\ \cup\ &\{S^-(u, v) \mid S \in \sigma, S(v, u) \in I\}. \end{aligned}$$

Note that the directed edge (v, u) covers the same facts as (u, v), except that in non-unary facts the relations $S \in \sigma$ and the reverse relations S^- are swapped.

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

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

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

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

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

A natural query language captured by UCQ^∞ is *Datalog*, again without negation or inequalities. A Datalog program defines a signature of *intensional predicates*, including a 0-ary predicate Goal(), and consists of a set of *rules* which explain how intensional facts can be *derived* from other intensional facts and from the facts of the instance (called *extensional*). The interpretation of the intensional predicates is defined by taking the (unique) least fixpoint of applying the rules, and the query holds if and only if the Goal() predicate can be derived. For formal definitions of this semantics, we refer the reader to the standard literature [AHV95]. Datalog can in particular be used to express *regular path queries* (RPQs) and *conjunctions of regular path queries with inverses* (C2RPQs) [Bar13].

As Datalog queries are homomorphism-closed, we can see each Datalog program as a UCQ^∞, with the disjuncts intuitively corresponding to *derivation trees* for the program.

Example 3.1. Consider the following Datalog program with one monadic intensional predicate U over extensional signature R, S, T:

$$\begin{aligned} R(x,y) &\to U(x), \\ U(x), S(x,y) &\to U(y), \\ U(x), T(x,y) &\to \mathrm{Goal}(). \end{aligned}$$

This program tests if the instance contains a path of facts $R(a_0, a_1), S(a_1, a_2), \ldots, S(a_{n-1}, a_n), T(a_n, a_{n+1})$ for some $n > 0$, intuitively corresponding to the regular path query RS^*T. This is an unbounded UCQ$^\infty$.

However, note that the class UCQ$^\infty$ is a larger class than Datalog, because there are homomorphism-closed queries that are not expressible in Datalog [DK08].

Ontology-mediated queries, or OMQs [BCLW14], are another subclass of UCQ$^\infty$. An OMQ is a pair $(Q, \mathcal{T})$, where Q is (typically) a UCQ, and $\mathcal{T}$ is an ontology. A database instance I *satisfies* an OMQ $(Q, \mathcal{T})$ if the instance I and the logical theory $\mathcal{T}$ entail the query Q in the standard sense – see, e.g., [BCLW14], for details. There are ontological languages for OMQs based on *description logics* [BCM$^+$07] and on *existential rules*, also known as *tuple-generating dependencies (TGDs)* [CGK13, CGL12]. It is well known that every OMQ $(Q, \mathcal{T}) \in (\mathrm{UCQ}, \mathrm{TGD})$ is closed under homomorphisms. Thus, the dichotomy result of the paper applies to every OMQ from $(\mathrm{UCQ}, \mathrm{TGD})$ over unary and binary predicates, which, in turn, covers several OMQ languages based on description logics. There are also many OMQs that can be equivalently expressed as a query in Datalog or in disjunctive Datalog over an arity-two signature [BCLW14, EOŠ$^+$12, GS12], thus falling in the class UCQ$^\infty$. In particular, this is the case of any OMQ involving negation-free $\mathcal{ALCHI}$ (Theorem 6 of [BCLW14]), and of fragments of $\mathcal{ALCHI}$, e.g., $\mathcal{ELHI}$, and $\mathcal{ELI}$ as in [JL12].

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

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

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

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

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

Complexity background. FP is the class of functions $f : \{0,1\}^* \mapsto \{0,1\}^*$ computable by a polynomial-time deterministic Turing machine. The class #P, introduced by Valiant in [Val79], contains the computation problems that can be expressed as the number of accepting paths of a nondeterministic polynomial-time Turing machine. Equivalently, a function $f : \{0,1\}^* \mapsto \mathbb{N}$ is in #P if there exists a polynomial $p : \mathbb{N} \mapsto \mathbb{N}$ and a polynomial-time deterministic Turing machine M such that for every $x \in \{0,1\}^*$, it holds that:

$$f(x) = \left| \left\{ y \in \{0,1\}^{p(|x|)} \mid M \textit{ answers 1 on the input } (x, y) \right\} \right|.$$

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

To show #P-hardness, we use *polynomial-time Turing reductions* [Coo71]. A function f is #P-complete under polynomial time Turing reductions if it is in #P and every $g \in \#\mathrm{P}$ is in FP^f. Polynomial-time Turing reductions are the most common reductions for the class #P and they are the reductions used to show #P-hardness in the dichotomy of Dalvi and Suciu [DS12], so we use them throughout this work.

Problems. We will show hardness by reducing from two well-known #P-hard problems. For some queries, we reduce from #PP2DNF [PB83], which is a standard tool to show hardness of unsafe UCQs. The original problem uses Boolean formulae; here, we give an equivalent rephrasing in terms of bipartite graphs.

Definition 3.2. Given a bipartite graph $H = (A, B, C)$ with edges $C \subseteq A \times B$, a *possible world* of H is a pair $\omega = (A', B')$ with $A' \subseteq A$ and $B' \subseteq B$. We call the possible world *good* if it is not an independent set, i.e., if one vertex of A' and one vertex of B' are adjacent in C; and call it *bad* otherwise. The *positive partitioned 2DNF problem (*#PP2DNF*)* is the following: given a bipartite graph, compute how many of its possible worlds are good.

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

For other queries, we reduce from a different problem, known as the *undirected st-connectivity problem* (#U-ST-CON) [PB83]:

Definition 3.3. An *st-graph* is an undirected graph $G = (W, C)$ with two distinguished vertices $s \in W$ and $t \in W$. A *possible world* of G is a subgraph $\omega = (W, C')$ with $C' \subseteq C$. We call the possible world *good* if C' contains a path connecting s and t, and *bad* otherwise. The *source-to-target undirected reachability problem (*#U-ST-CON*)* is the following: given an st-graph, compute how many of its possible worlds are good.

4. Result Statement

The goal of this paper is to extend the dichotomy of Dalvi and Suciu [DS12] on PQE for UCQs. Their result states:

Theorem 4.1 [DS12]**.** *Let Q be a UCQ. Then,* PQE(Q) *is either in* FP *or it is* #P*-hard.*

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

Theorem 4.2 (Dichotomy)**.** *Let Q be a* UCQ^∞ *over an arity-two signature. Then, either Q is equivalent to a safe UCQ and* $\mathrm{PQE}(Q)$ *is in* FP*, or it is not and* $\mathrm{PQE}(Q)$ *is* #P*-hard.*

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

Theorem 4.3. *Let Q be an unbounded* UCQ^∞ *query over an arity-two signature. Then,* $\mathrm{PQE}(Q)$ *is* #P*-hard.*

This result applies to the very general class of unbounded UCQ^∞. It implies in particular that the PQE problem is #P-hard for all Datalog queries that are not equivalent to a UCQ, as in Example 3.1: this is the case of all Datalog queries except the ones that are nonrecursive or where recursion is *bounded* [HKMV95].

Effectiveness and uniformity. We do not study whether our dichotomy result in Theorem 4.2 is effective, i.e., we do not study the problem of determining, given a query, whether it is safe or unsafe. The dichotomy of Theorem 4.1 for UCQs is effective via the algorithm of [DS12]: this algorithm has a super-exponential bound (in the query), with the precise complexity being open. Our dichotomy concerns the very general query language UCQ^∞, and its effectiveness depends on how the input is represented: to discuss this question, we need to restrict queries to some syntactically defined fragment. If we restrict to Datalog queries, it is not clear whether our dichotomy is effective, because it is undecidable, given an arbitrary Datalog program as input, to determine whether it is bounded [GMSV93]. This means that there is little hope for our dichotomy to be decidable over arbitrary Datalog queries, but on its own it does not imply undecidability, so the question remains open. However, our dichotomy is effective for query languages for which boundedness is decidable, e.g., monadic Datalog, its generalization GN-Datalog [BTCCB15], C2RPQs [BFR19], or ontology-mediated query answering with guarded existential rules [BBLP18].

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

Proof outline. Theorem 4.3 is proven in Sections 5–7. There are two cases, depending on the query. We study the first case in Section 5, which covers queries for which we can find a model with a so-called *non-iterable edge*. Intuitively, this is a model where we can make the query false by replacing the edge by a back-and-forth path of some length between two neighboring facts that it connects. For such queries, we can show hardness by a reduction from #PP2DNF, essentially like the hardness proof for the query $Q_0 : R(w,x), S(x,y), T(y,z)$ which is the arity-two variant of the unsafe query of [DS07, Theorem 5.1]. This hardness proof covers some bounded queries (including Q_0) and some unbounded ones.

In Section 6, we present a new ingredient, to be used in the second case, i.e., when there is no model with a non-iterable edge. We show that any unbounded query must always have a model with an edge that is *tight*, i.e., the query no longer holds if we replace that edge with two copies, one copy connected only to the first element and another copy connected only to the second element. What is more, we can find a model with a tight edge which is *minimal* in some sense, which we call a *minimal tight pattern*.

In Section 7, we use minimal tight patterns for the second case, covering unbounded queries that have a minimal tight pattern where the tight edge of the pattern is iterable. This applies to all queries to which Section 5 did not apply (and also to some queries to which it did). Here, we reduce from the #U-ST-CON problem: intuitively, we use the iterable edge for a kind of reachability test, and we use the minimality and tightness of the pattern to show the soundness and completeness of the reduction.

Generalizations. In Section 8, we give two generalizations of our result. First, we observe that our reductions only use tuple probabilities from $\{0, 0.5, 1\}$. This means that all #P-hardness results hold even when restricting the probabilistic query evaluation problem to the so-called *generalized model counting problem* studied for instance in [KS21], so we can also state our dichotomy in this context. Second, we show that all our results also apply when we consider signatures featuring predicates with arity 1 and 2.

5. Hardness with Non-Iterable Edges

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

Definition 5.1. Let I be an instance. We say that an element $u \in \mathrm{dom}(I)$ of I is a *leaf* if it occurs in only one undirected edge. We say that an edge (directed or undirected) is a *leaf edge* if one of its elements (possibly both) is a leaf; otherwise, it is a *non-leaf edge*.

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

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

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

Example 5.2. Given an instance $I = \{R(a, b), T(b, b), S(c, b), R(d, c)\}$, the edge (b, c) is non-leaf and the only possible incident pair for it is $(R(a, b), R^{-}(c, d))$.

We can now define the *iteration process* on an instance $I_{e,\Pi}$, which intuitively replaces the edge e by a path of copies of e, keeping the facts of Π at the beginning and end of the path, and copying all other incident facts. Note that, while the instances that we work with are over the signature σ, we will see them as $\sigma^{\leftrightarrow}$ instances in the definition of this process,

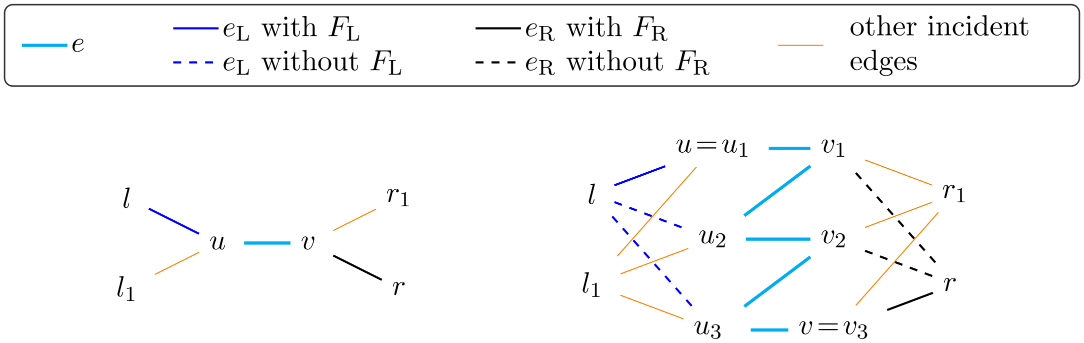

e
e_{L} with F_{L}
e_{L} without F_{L}
e_{R} with F_{R}
e_{R} without F_{R}
other incident edges
l
l_1
u
v
r_1
r
$u = u_1$
v_1
u_2
v_2
u_3
$v = v_3$

Figure 1. Example of iteration from an instance $I_{e,\Pi}$ (left) to $I^3_{e,\Pi}$ (middle). We write $\Pi = (F_{\mathrm{L}}, F_{\mathrm{R}})$ and call e_{L} and e_{R} the edges of F_{L} and F_{R}. Each line represents an edge covering in general multiple $\sigma^{\leftrightarrow}$-facts. A key is given at the top.

e.g., when creating copies of facts, in particular of the $\sigma^{\leftrightarrow}$-facts F_{L} and F_{R}; but the facts that we actually create are σ-facts, and the resulting instance is a σ-instance. The iteration process is represented in Figure 1, and defined formally below:

Definition 5.3. Let $I_{e,\Pi}$ be a σ-instance where $e = (u, v)$, $\Pi = (F_{\mathrm{L}}, F_{\mathrm{R}})$, $F_{\mathrm{L}} = R_{\mathrm{L}}(l, u)$, $F_{\mathrm{R}} = R_{\mathrm{R}}(v, r)$, and let $n \geq 1$. The *n-th iterate* of e in I relative to Π, denoted $I^n_{e,\Pi}$, is a σ-instance with domain $\mathrm{dom}(I^n_{e,\Pi}) := \mathrm{dom}(I) \cup \{u_2, \ldots, u_n\} \cup \{v_1, \ldots, v_{n-1}\}$, where the new elements are fresh, and where we use u_1 to refer to u and v_n to refer to v for convenience. The facts of $I^n_{e,\Pi}$ are defined by applying the following steps:

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

Note that, for $n = 1$, we obtain exactly the original instance. Intuitively, we replace e by a path going back-and-forth between copies of u and v (and traversing e alternatively in one direction and another). The intermediate vertices have the same incident facts as the original endpoints except that we have not copied the left-incident fact and the right-incident fact of the incident pair.

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

Observation 5.4. For any instance I, for any non-leaf edge e of I, for any incident pair Π for e, and for any $1 \leq i \leq j$, it holds that $I^j_{e,\Pi}$ has a homomorphism to $I^i_{e,\Pi}$.

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

Hence, choosing an instance I that satisfies Q, a non-leaf edge e of I, and an incident pair Π, there are two possibilities. Either all iterates $I^n_{e,\Pi}$ satisfy Q, or there is some iterate

$I^{n_0}_{e,\Pi}$ with $n_0 > 1$ that violates Q (and, by Observation 5.4, all subsequent iterates also do). We call e *iterable* relative to Π in the first case, and *non-iterable* in the second case:

Definition 5.5. A non-leaf edge e of a model I of a query Q is *iterable relative to an incident pair* Π if $I^{n}_{e,\Pi}$ satisfies Q for each $n \geq 1$; otherwise, it is *non-iterable relative to* Π. We call e *iterable* if it is iterable relative to some incident pair, and *non-iterable* otherwise.

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

Theorem 5.6. *For every* UCQ$^\infty$ *Q, if Q has a model I with a non-leaf edge e that is non-iterable, then* PQE(Q) *is #P-hard.*

Let us illustrate on an example how to apply this result:

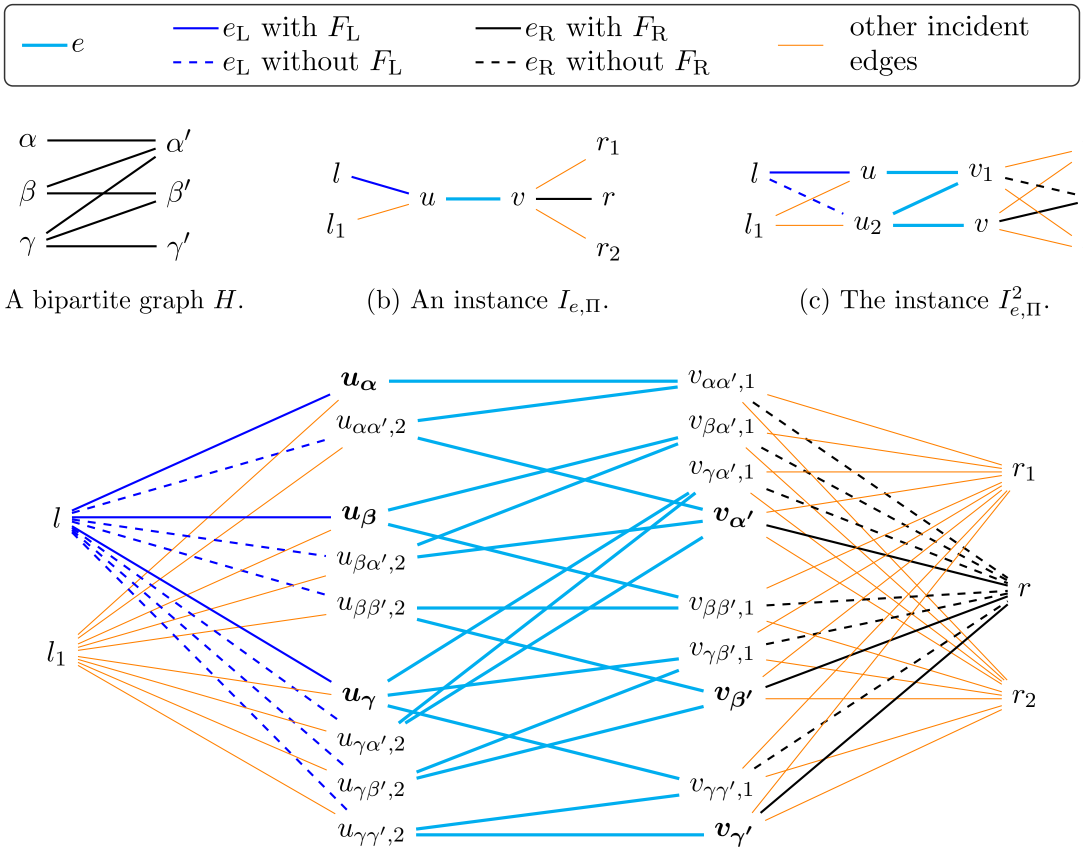
Example 5.7. Consider the RPQ RS^*T. This query has a model $\{R(a,b), S(b,c), T(c,d)\}$ with an edge (b,c) that is non-leaf and non-iterable. Indeed its iterate with $n = 2$ relative to the only possible incident pair yields $\{R(a,b), S(b,c'), S(b',c'), S(b',c), T(c,d)\}$ which does not satisfy the query. Hence, Theorem 5.6 shows that PQE is #P-hard for this RPQ. Importantly, the choice of the model matters, as this query also has models where all non-leaf edges are iterable, for instance $\{R(a,b), S(b,c), T(c,d), R(a',b'), S(b',c'), T(c',d')\}$, or $\{R(a,b), T(b,c)\}$ which has no non-leaf edge at all.

Note that Theorem 5.6 does not assume that the query is unbounded, and also applies to some bounded queries. For instance, the unsafe CQ $Q_0 : R(w,x), S(x,y), T(y,z)$ can be shown to be unsafe using this result, with the model $\{R(a,b), S(b,c), T(c,d)\}$ and edge (b,c). However, Theorem 5.6 is too coarse to show #P-hardness for all unsafe UCQs; for instance, it does not cover $Q_0' : R(x,x), S(x,y), T(y,y)$, or $Q_1 : (R(w,x), S(x,y)) \lor (S(x,y), T(y,z))$. It will nevertheless be sufficient for our purposes when studying *unbounded queries*, as we will see in the next sections.

Hence, in the rest of this section, we prove Theorem 5.6. Let $I_{e,\Pi}$ be the instance with the non-iterable, non-leaf edge, and let us take the smallest $n_0 > 1$ such that $I^{n_0}_{e,\Pi}$ violates the query. The idea is to use $I^{n_0}_{e,\Pi}$ to show hardness of PQE by reducing from #PP2DNF (Definition 3.2). Thus, let us explain how we can use $I_{e,\Pi}$ to code a bipartite graph H in polynomial time into a TID $\mathcal{I}$. The definition of this coding does not depend on the query Q, but we will use the properties of $I_{e,\Pi}$ and n_0 to argue that it defines a reduction between #PP2DNF and PQE, i.e., there is a correspondence between the possible worlds of H and the possible worlds of $\mathcal{I}$, such that good possible worlds of H are mapped to possible worlds of $\mathcal{I}$ which satisfy Q. Let us first define the coding, which we also illustrate on an example in Figure 2:

Definition 5.8. Let $I_{e,\Pi}$ be a σ-instance where $e = (u,v)$, $\Pi = (F_{\mathrm{L}}, F_{\mathrm{R}})$, $F_{\mathrm{L}} = R_{\mathrm{L}}(l,u)$, $F_{\mathrm{R}} = R_{\mathrm{R}}(v,r)$, and let $n \geq 1$. Let $H = (A,B,C)$ be a connected bipartite graph. The *coding* of H relative to $I_{e,\Pi}$ and n is a TID $\mathcal{I} = (J,\pi)$ with domain $\mathrm{dom}(J) := (\mathrm{dom}(I) \setminus \{u,v\}) \cup \{u_a \mid a \in A\} \cup \{v_b \mid b \in B\} \cup \{u_{c,2}, \ldots, u_{c,n} \mid c \in C\} \cup \{v_{c,1}, \ldots, v_{c,n-1} \mid c \in C\}$, where the new elements are fresh. The facts of the σ-instance J and the probability mapping π are defined as follows:

- *Copy non-incident facts:* Initialize J as the induced subinstance of I on $\mathrm{dom}(I) \setminus \{u,v\}$.
- *Copy incident facts F_{L} and F_{R}:* Add to J the $\sigma^{\leftrightarrow}$-fact $R_{\mathrm{L}}(l,u_a)$ for each $a \in A$, and add to J the $\sigma^{\leftrightarrow}$-fact $R_{\mathrm{R}}(v_b,r)$ for each $b \in B$.



(a) A bipartite graph H.

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

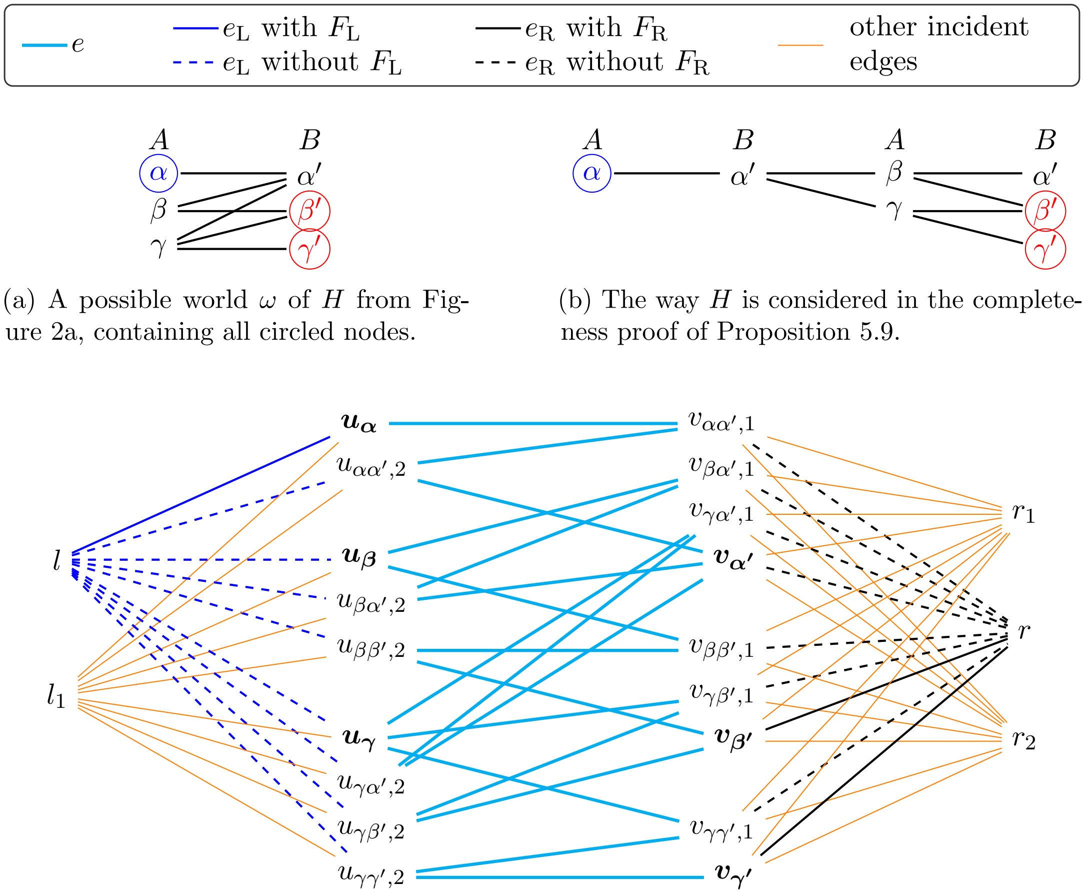
(c) The instance $I^2_{e,\Pi}$.

(d) Coding of the bipartite graph H relative to $I^2_{e,\Pi}$. Bold elements correspond to vertices of H. For brevity, we write the edge (x, ξ) simply as $x\xi$.

Figure 2. Example of the coding of a bipartite graph H shown in Figure 2a. We encode H relative to an instance $I_{e,\Pi}$ (Figure 2b), with a non-leaf edge e and an incident pair Π. The result $I^2_{e,\Pi}$ of iterating e in I with $n = 2$ (Definition 5.3) is shown in Figure 2c. The coding of H relative to $I_{e,\Pi}$ and $n = 2$ (Definition 5.8) is shown in Figure 2d, with the probabilistic facts being the copies of F_{L} and F_{R} in the edges in solid blue and black.

- *Copy other left-incident facts:* For each $\sigma^{\leftrightarrow}$-fact $F'_{\mathrm{L}} = R'_{\mathrm{L}}(l', u)$ of I that is left-incident to e (i.e., $l' \notin \{u, v\}$) and where $F'_{\mathrm{L}} \neq F_{\mathrm{L}}$, add to J the facts $R'_{\mathrm{L}}(l', u_a)$ for each $a \in A$, and add to J the facts $R'_{\mathrm{L}}(l', u_{c,j})$ for each $2 \leq j \leq n$ and $c \in C$.
- *Copy other right-incident facts:* For each $\sigma^{\leftrightarrow}$-fact $F'_{\mathrm{R}} = R'_{\mathrm{R}}(v, r')$ of I that is right-incident to e (i.e., $r' \notin \{u, v\}$) and where $F'_{\mathrm{R}} \neq F_{\mathrm{R}}$, add to J the facts $R'_{\mathrm{R}}(v_b, r')$ for each $b \in B$ and add to J the facts $R'_{\mathrm{R}}(v_{c,j}, r')$ for each $1 \leq j \leq n - 1$ and $c \in C$.
- *Create copies of e:* For each $c \in C$ with $c = (a, b)$, copy e on the following pairs: $(u_{c,i}, v_{c,i})$ for $1 \leq i \leq n$, and $(u_{c,i+1}, v_{c,i})$ for $1 \leq i \leq n - 1$, where we use $u_{c,1}$ to refer to u_a and $v_{c,n}$ to refer to v_b.

Finally, we define the function π such that it maps all the facts created in the step "Copy incident facts F_{L} and F_{R}" to 0.5, and all other facts to 1.



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

(b) The way H is considered in the completeness proof of Proposition 5.9.

(c) The possible world $\phi(\omega)$ of the coding (Figure 2d) for ω. The edges (l, u_β), (l, u_γ), and $(v_{\alpha'}, r)$ are changed to dashed lines, as they correspond to vertices of H that are not kept in ω.

FIGURE 3. Example for the completeness direction of the proof of Proposition 5.9. Figure 3a shows a bad possible world ω of the bipartite graph. The corresponding possible world of the coding of Figure 2d (using the instance $I^2_{e,\Pi}$ of Figure 2b) is given in Figure 3c. In the proof, we explore H as depicted in Figure 3b to argue that Figure 3c has a homomorphism to $I^5_{e,\Pi}$.

Observe how this definition relates to the definition of iteration (Definition 5.3): we intuitively code each edge of the bipartite graph as a copy of the path of copies of e in the definition of the n-th iterate of (u, v). Note also that there are exactly $|A| + |B|$ uncertain facts, by construction. It is clear that, for any choice of $I_{e,\Pi}$ and n, this coding is in polynomial time in H.

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

Proposition 5.9. *Let the TID $\mathcal{I} = (J, \pi)$ be the coding of a connected bipartite graph $H = (A, B, C)$ relative to an instance $I_{e,\Pi}$ and to $n \geq 1$ as described in Definition 5.8, and let ϕ be the bijective function defined above from the possible worlds of H to those of $\mathcal{I}$. Then:*

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

Proof. Observe that (1) corresponds to the soundness of the reduction, and (2) to the completeness. Intuitively, (1) holds because $\phi(\omega)$ then contains a subinstance isomorphic to $I^n_{e,\Pi}$. To show (2), we need a more involved argument, see e.g. Figure 3: when ω is bad, we can show how to "fold back" $\phi(\omega)$, going from the copies of F_L to the copies of F_R, into the iterate $I^{3n-1}_{e,\Pi}$. This uses the fact that ω is bad, so the copies of F_L and F_R must be sufficiently far from one another.

(1) Let us assume that $\phi(\omega) = J'$. We more specifically claim that J' has a subinstance which is isomorphic to $I^n_{e,\Pi}$. 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 $I^{3n-1}_{e,\Pi}$. To do this, first map all facts of J' that do not involve a copy of u or v to the corresponding facts of $I^{3n-1}_{e,\Pi}$ using the identity mapping. We will now explain how the copies of u and v in J' are mapped to copies of u and v in $I^{3n-1}_{e,\Pi}$: this is clearly correct for the facts in J' that use these copies of u and v and that were created as copies of left-incident or right-incident facts to e in I except F_L and F_R. Thus, we must simply ensure that this mapping respects the facts in J' that were created as copies of F_L, of F_R, or of the edge e, as we have argued that all other facts of J' are correctly mapped to $I^{3n-1}_{e,\Pi}$.

Our way to do this is illustrated in Figure 3. The first step is to take all copies of F_L in J', which correspond to vertices in $a \in A$ that were kept, and to map them all to the element u in $I^{3n-1}_{e,\Pi}$, which is possible as it has the incident fact F_L. In Figure 3c, this is only the copy of F_L on (l, u_α). Now, we start at the elements of the form u_a, and we follow the paths of $2n-1$ copies of e back-and-forth from these elements until we reach elements of the form v_b: we map these paths to the first $2n-1$ edges of the path of copies of e from u to v in $I^{3n-1}_{e,\Pi}$. In Figure 3c, we reach $v_{\alpha'}$. From our assumption about the possible world J', none of the v_b reached at that stage have an incident copy of F_R, 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 ω of H, which is impossible as ω is bad.

The second step is to go *back* in J' on the copies of e incident to these elements that were not yet visited, and we follow a path of copies of e that were not yet mapped. We map these to the next $2n-1$ copies of e, going *forward* in the path from u to v in $I^{3n-1}_{e,\Pi}$. We then reach elements of the form u_a, and they do not have any incident copies of F_L because all such edges and their outgoing paths were visited in the first step. In Figure 3c, we reach u_β and u_γ.

The third step is to go *forward* in J' on the copies of e incident to these elements that were not yet visited, and follow a path of copies of e that goes to elements of the form v_b, mapping this to the last $2n-1$ edges of the path from u to v in $I_{e,\Pi}^{3n-1}$. Some of these v_b may now be incident to copies of F_{R}, but the same is true of v in $I_{e,\Pi}^{3n-1}$, and we have just reached v. Indeed, note that we have followed $(2n-1) \times 3$ copies of e in J' in total (going forward each time), and this is equal to $2 \times (3n-1) - 1$, the number of copies of e in the path from u to v in $I_{e,\Pi}^{3n-1}$. Thus, we can map the copies of F_{R} correctly. In Figure 3c, we reach $v_{\beta'}$ and $v_{\gamma'}$.

In Figure 3c, we have visited everything after the third step. However, in general, there may be some elements of J' that we have not yet visited, and for which we still need to define a homomorphic image. Thus, we perform more steps until all elements are visited. Specifically, in even steps, we go *back* on copies of e in J' from the elements reached in the previous step to reach elements that were not yet visited, going *back* on the path from u to v in $I_{e,\Pi}^{3n-1}$, reaching elements of the form u_a (which cannot be incident to any copy of F_{L} for the same reason as in the second step). In odd steps, we go *forward* in J' on copies of e, going *forward* on the path from u to v in $I_{e,\Pi}^{3n-1}$, reaching elements of the form v_b in J' that we map to b in $I_{e,\Pi}^{3n-1}$, including the F_{R}-fact that may be incident to them.

We repeat these additional backward-and-forward steps until everything reachable has been visited. At the end of the process, from our assumption that H is a connected bipartite graph, we have visited all the elements of J' for which we had not defined a homomorphic image yet, and we have argued that the way we have mapped them is indeed a homomorphism. This concludes the construction of the homomorphism, and concludes the proof. □

Thanks to Proposition 5.9, we can now prove the main result of this section:

Proof of Theorem 5.6. Fix the query Q, the instance I, the non-leaf edge e of I which is non-iterable, the incident pair Π relative to which it is not iterable, and let us take the smallest $n_0 > 1$ such that $I_{e,\Pi}^{n_0}$ does not satisfy the query, but $I_{e,\Pi}^{n_0-1}$ does.

We show the #P-hardness of PQE(Q) by reducing from #PP2DNF (Definition 3.2). Let $H = (A, B, C)$ be an input connected bipartite graph. We apply the coding of Definition 5.8 with $n_0 - 1$ and obtain a TID $\mathcal{I}$. This coding can be done in polynomial time.

Now let us use Proposition 5.9. We know that $I_{e,\Pi}^{n_0-1}$ satisfies Q, but $I_{e,\Pi}^{3(n_0-1)-1}$ does not, because $n_0 > 1$ so $3(n_0-1)-1 = 3n_0 - 4 \geq n_0$, and as we know that $I_{e,\Pi}^{n_0}$ violates Q, then so does $I_{e,\Pi}^{3(n_0-1)-1}$ thanks to Observation 5.4. Thus, Proposition 5.9 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. □

6. 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, non-leaf edge. This leaves open the case of unbounded queries for which all non-leaf edges in all models can be iterated. We first note that this case is not hypothetical, i.e., there actually exist some unbounded queries for which, in all models, all non-leaf edges can be iterated:

Example 6.1. Consider the following Datalog program:

$$\begin{aligned} R(x,y) &\to A(y), \\ A(x), S(x,y) &\to B(y), \\ B(x), S(y,x) &\to A(y), \\ B(x), T(x,y) &\to \mathrm{Goal}(). \end{aligned}$$

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

Importantly, if we tried to reduce from #PP2DNF for this query as in the previous section, then the reduction would fail because the edge is iterable: in possible worlds of the bipartite graph, where we have not retained two adjacent vertices, we would still have matches of the query in the corresponding possible world of the probabilistic instance, where we go from a chosen vertex to another by going *back-and-forth* on the copies of e that code the edges of the bipartite graph. These are the "back-and-forth matches" which were missed in [Jun14, JL12] and are discussed in [JL20].

In light of this, we handle the case of such queries in the next two sections. In this section, we prove a general result for unbounded queries (independent from the previous section): all unbounded queries must have a model with a *tight edge*, which is additionally *minimal* in some sense. Tight edges and iterable edges will then be used in Section 7 to show hardness for unbounded queries which are not covered by the previous section.

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

Definition 6.2. The *dissociation* of a non-leaf edge $e = (u, v)$ in I is the instance I' where:

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

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

Example 6.3. Consider the following instance:

$$I = \{R(a,b), S(b,a), T(b,a), R(a,c), S(c,b), S(d,b), U(a,a), U(b,b)\}.$$

The edge (a, b) is non-leaf, as witnessed by the edges $\{a, c\}$ and $\{b, c\}$. The result of the dissociation is then:

$$\begin{aligned} I' = \{&R(a,b'), S(b',a), T(b',a), R(a',b), S(b,a'), T(b,a'), \\ &R(a,c), S(c,b), S(d,b), U(a,a), U(a',a'), U(b,b), U(b',b')\} \end{aligned}$$

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

Definition 6.4. Let Q be a query and I be a model of Q. An edge e of I is *tight* if it is non-leaf, and the result of the dissociation of e in I does not satisfy Q. A *tight pattern* for the query Q is a pair (I, e) of a model I of Q and of an edge e of I that is tight.

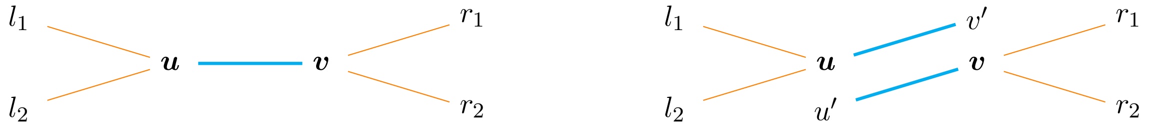


FIGURE 4. An instance (left) with a non-leaf edge (u, v), and the result (right) of dissociating (u, v).

Intuitively, a tight pattern is a model of a query containing at least three edges $\{u, a\}$, $\{a, b\}$, $\{b, v\}$ (possibly $u = v$) such that performing a dissociation makes the query false. For instance, for the unsafe CQ $Q_0 : R(w, x), S(x, y), T(y, z)$ from [DS07], a tight pattern would be $\{R(a, b), S(b, c), T(c, d)\}$ with the edge (b, c). Again, not all unsafe CQs have a tight pattern, e.g., $Q_0' : R(x, x), S(x, y), T(y, y)$, and $Q_1 : (R(w, x), S(x, y)) \lor (S(x, y), T(y, z))$ from Section 5 do not.

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

Definition 6.5. Given an instance I with a non-leaf edge $e = (a, b)$, the *weight* of e is the number of facts covered by e in I (including unary facts). The *side weight* of e is the number of $\sigma^{\leftrightarrow}$-facts in I that are left-incident to e, plus the number of $\sigma^{\leftrightarrow}$-facts in I that are right-incident[1] to e. Given a query Q, we say that a tight pattern (I, e) is *minimal* if:

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

We can now state the main result of this section:

Theorem 6.6. *Every unbounded* UCQ$^\infty$ Q *has a model* I *with a non-leaf edge* e *such that* (I, e) *is a minimal tight pattern.*

The idea of how to find tight patterns is as follows. We first note that the only instances without non-leaf edges are intuitively disjoint unions of star-shaped subinstances. Now, if a query is unbounded, then its validity cannot be determined simply by looking at such subinstances (unlike Q_0' or Q_1 from Section 5), so there must be a model of the query with an edge that we cannot dissociate without breaking the query, i.e., a tight pattern. Once we know that there is a tight pattern, then it is simple to argue that we can find a model with a tight edge that is minimal in the sense that we require.

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

Observation 6.7. For any instance I, any 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.

[1] Recall that left-incident and right-incident facts do not include unary facts.

Let us write $e = (a, b)$. 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. □

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

Proposition 6.8. *For every signature σ, there exists a bound $k_\sigma > 0$, ensuring the following: for every instance I on σ having no non-leaf edge, there exists a subinstance $I' \subseteq I$ such that I has a homomorphism to I' and such that we have $|I'| < k_\sigma$.*

Proof. We start by outlining the main idea behind of the proof. Connected instances having no non-leaf edges can have at most one non-leaf element, with all edges using this element and a leaf. Now, each edge can be described by the set of facts that it covers, for which there are finitely many possibilities (exponentially many in the signature size). We can thus collapse together the edges that have the same set of facts and obtain the subinstance I'. Now, disconnected instances having no non-leaf edges are unions of the connected instances of the form above, so the number of possibilities up to homomorphic equivalence is again finite (exponential in the number of possible connected instances). We can then conclude by collapsing together connected components that are isomorphic.

Let us now formally prove the result, first for connected instances I. In this case, we define the constant $k'_\sigma := 2^{4 \times |\sigma|}$. 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 (a, b) and consists of at most $4\,|\sigma|$ facts: there are $|\sigma|$ possible facts of the form $R(a, b)$, plus $|\sigma|$ possible facts of the form $R(b, a)$, plus $|\sigma|$ possible facts of the form $R(a, a)$, plus $|\sigma|$ possible facts of the form $R(b, b)$. Thus, 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. 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, contradicting our assumption that I has no non-leaf edge. 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.

Second, we formally show the result for instances I that are not necessarily connected. Letting I be such an instance, we consider its connected components $I_1, \ldots, I_m$, i.e., the disjoint subinstances induced by the connected components of the Gaifman graph of I. Each of these is connected and has no non-leaf edges, so, by the proof of the previous paragraph, there are subsets $I'_1 \subseteq I_1, \ldots, I'_m \subseteq I_m$ with at most k'_σ facts each and a homomorphism of each I_i to its I'_i. Now, there are only constantly many instances with at most k'_σ 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 its I'_i. This concludes the proof. □

We can now prove Theorem 6.6 by appealing to the unboundedness of the query. To do this, we will rephrase unboundedness in terms of *minimal models*:

Definition 6.9. A *minimal model* of a query Q is an instance I that satisfies Q and such that every proper subinstance of I violates Q.

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

Observation 6.10. A UCQ$^\infty$ query Q is unbounded iff it has a minimal model I with more than k facts for any $k \in \mathbb{N}$.

We are now ready to conclude the proof of Theorem 6.6:

Proof of Theorem 6.6. We start by showing the first part of the claim: any unbounded query has a tight pattern. Let k_σ be the bound from Proposition 6.8. By Observation 6.10, let I_0 be a minimal model with more than k_σ 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 $I := I'$.

Observation 6.7 implies that the dissociation process must terminate after at most n_0 steps, where n_0 is the number of non-leaf edges of I_0. Let I_n be the result of this process, with $n \leq n_0$. 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 6.8 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 in I_0 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 more than k_σ facts, I'_0 is a strict subset of I_0 that satisfies Q. This contradicts the minimality of I_0. Thus, we 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. □

7. Hardness with Tight Iterable Edges

In this section, we conclude the proof of Theorem 4.3 by showing that a minimal tight pattern can be used to show hardness when it is iterable. Formally:

Theorem 7.1. *For every* UCQ$^\infty$ Q*, if* Q *has a model* I *with a non-leaf edge* e *that is iterable then* PQE(Q) *is #P-hard.*

This covers all the queries to which Section 5 did not apply. We note however that it does not subsume the result of that section, i.e., there are some unbounded queries to which it does not apply.

Example 7.2. Consider again the RPQ RS^*T from Example 5.7. Recall that we can find some models with iterable edges (e.g., $\{R(a,b), S(b,c), T(c,d), R(a',b'), S(b',c'), T(c',d')\}$), but this query has no models with an iterable edge which is tight. Thus, hardness for this query cannot be shown with the result in this section, and we really need Theorem 5.6 to cover it. Of course, there are also some unbounded queries for which hardness can be shown with either of the two results, e.g., a disjunction of the RPQ RS^*T and of the query of Example 6.1 on a disjoint signature.

From Theorem 7.1, it is easy to conclude the proof of Theorem 4.3:

Proof of Theorem 4.3. Let Q be an unbounded UCQ$^\infty$. If we have a model of Q with a non-iterable edge, then we conclude by Theorem 5.6 that PQE(Q) is #P-hard. Otherwise, by Theorem 6.6, we have a minimal tight pattern, and its edge is then iterable (otherwise the first case would have applied), so that we can apply Theorem 7.1. □

Thus, it only remains to show Theorem 7.1. The idea is to use the iterable edge e of the minimal tight pattern (I, e) for some incident pair Π to reduce from the undirected st-connectivity problem #U-ST-CON (Definition 3.3). Given an input st-graph G for #U-ST-CON, we will code it as a TID $\mathcal{I}$ built using $I_{e,\Pi}$, with one probabilistic fact per edge of G. To show a reduction, we will argue that good possible worlds of G correspond to possible worlds J' of $\mathcal{I}$ containing some iterate $I^n_{e,\Pi}$ of the instance (Definition 5.3), with n being the length of the path, and J' then satisfies Q because e is iterable. Conversely, we will argue that bad possible worlds of G correspond to possible worlds J' of $\mathcal{I}$ that have a homomorphism to a so-called *fine dissociation* of e in I, and we will argue that this violates the query Q thanks to our choice of (I, e) as a minimal tight pattern. The notion of *fine dissociation* will be defined for an edge relative to an incident pair, but also relative to a specific choice of fact covered by the edge, as we formally define below (and illustrate in Figure 5):

Definition 7.3. Let I be a σ-instance, let $e = (u, v)$ be a non-leaf edge in I, let $F_\mathrm{L} = R_\mathrm{L}(l, u)$ and $F_\mathrm{R} = R_\mathrm{R}(v, r)$ be an incident pair of e in I, and let F_M be a non-unary fact covered by the edge e. The result of performing the *fine dissociation* of e in I relative to $F_\mathrm{L}, F_\mathrm{R}$ and F_M is a σ-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:

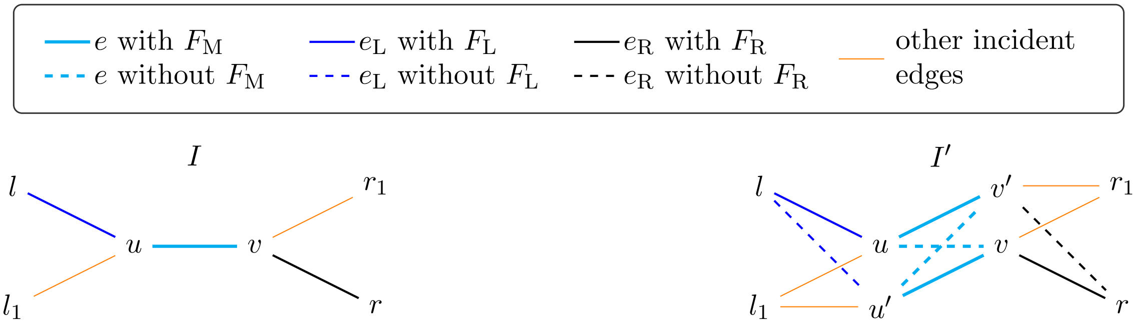

e with F_M
e without F_M
e_L with F_L
e_L without F_L
e_R with F_R
e_R without F_R
other incident edges
I
l
l_1
u
v
r_1
r
I'
l
l_1
u
u'
v'
v
r_1
r

FIGURE 5. Example of fine dissociation from an instance I (left) to I' (middle) for a choice of e, of $\Pi = (F_\mathrm{L}, F_\mathrm{R})$, and of F_M. We call e_L and e_R the edges of F_L and F_R.

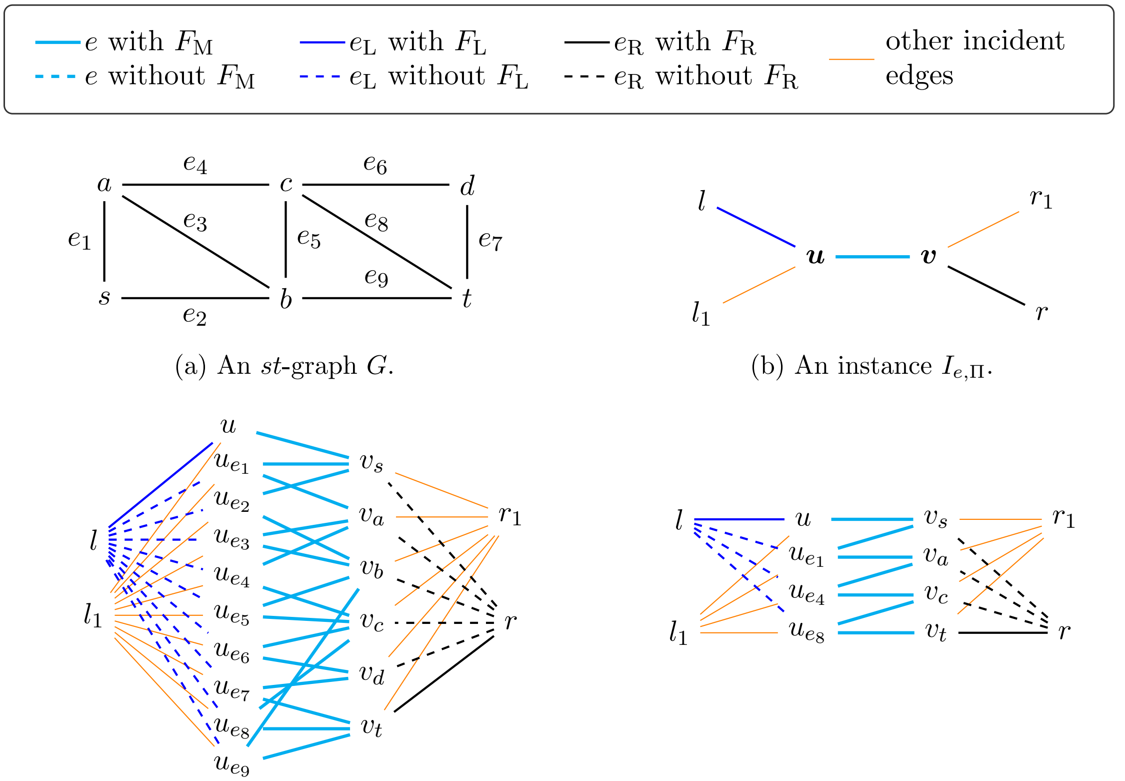
- *Copy non-incident facts:* Initialize I' as the induced subinstance of I on $\mathrm{dom}(I) \setminus \{u, v\}$.
- *Copy incident facts* F_L *and* F_R*:* Add the facts F_L and F_R to I'.
- *Copy other left-incident facts:* For every $\sigma^{\leftrightarrow}$-fact $F'_\mathrm{L} = R'_\mathrm{L}(l',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

Note that if the only non-unary fact covered by the edge e in I is F_M, then (u, v) and (u', v') are not edges in the result of the fine dissociation; otherwise, they are edges but with a smaller weight than e. Observe that fine dissociation is related both to dissociation (Section 6) and to iteration (Section 5). We will study later when fine dissociation can make the query false.

We can now start the proof of Theorem 7.1 by describing the coding. It depends on our choice of $I_{e,\Pi}$ and of a fact F_M, but like in Section 5 it does not depend on the query Q. Given an input st-graph G, we construct a TID $\mathcal{I}$ whose possible worlds will have a bijection to those of G. The process is illustrated on an example in Figure 6, and defined formally below:

Definition 7.4. Let $I_{e,\Pi}$ be a σ-instance where $e = (u, v)$, $\Pi = (F_\mathrm{L}, F_\mathrm{R})$, $F_\mathrm{L} = R_\mathrm{L}(l, u)$, $F_\mathrm{R} = R_\mathrm{R}(v, r)$ and let F_M be a non-unary fact of I covered by e. Let $G = (W, C)$ be an st-graph with source s and target t. The *coding* of G relative to $I_{e,\Pi}$ and F_M is a TID $\mathcal{I} = (J, \pi)$ with domain $\mathrm{dom}(J) := \mathrm{dom}(I) \cup \{u_c \mid c \in C\} \cup \{v_w \mid w \in W \setminus \{t\}\}$, where the new elements are fresh, and where we use v_t to refer to v for convenience. The facts of the σ-instance J and the probability mapping π are defined as follows:

- *Copy non-incident facts:* Initialize J as the induced subinstance of I on $\mathrm{dom}(I) \setminus \{u, v\}$.
- *Copy incident facts* F_L *and* F_R*:* Add the facts F_L and F_R to J.
- *Copy other left-incident facts:* For every $\sigma^{\leftrightarrow}$-fact $F'_\mathrm{L} = R'_\mathrm{L}(l', u)$ of I that is left-incident to e (i.e., $l' \notin \{u, v\}$) and where $F'_\mathrm{L} \neq F_\mathrm{L}$, add to J the facts $R'_\mathrm{L}(l', u_c)$ for each edge $c \in C$.


e with F_{M}
e without F_{M}
e_{L} with F_{L}
e_{L} without F_{L}
e_{R} with F_{R}
e_{R} without F_{R}
other incident edges
a e_4 c e_6 d
e_1 e_3 e_5 e_8 e_7
s e_2 b e_9 t
l l_1 u v r_1 r
u u_{e_1} u_{e_2} u_{e_3} u_{e_4} u_{e_5} u_{e_6} u_{e_7} u_{e_8} u_{e_9}
v_s v_a v_b v_c v_d v_t
l l_1 r_1 r
l l_1 u u_{e_1} u_{e_4} u_{e_8} v_s v_a v_c v_t r_1 r

(a) An st-graph G.

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

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

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

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

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

Finally, we define the function π as follows. For each edge c of C, we choose one arbitrary vertex $w \in c$, and set π to map the copy of the fact F_{M} in the edge (u_c, v_w) to 0.5, All other facts are mapped to 1 by π.

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

It is easy to see that the given coding is in polynomial time in the input G for every choice of $I_{e,\Pi}$ and F_{M}. Let us now define the bijection ϕ, mapping each possible world ω of G to a possible world of the TID $\mathcal{I}$ as follows. For each edge $c \in C$, we keep the probabilistic fact incident to u_c in the instance $\phi(\omega)$ if c is kept in the possible world ω, and we do not keep it otherwise. It is obvious that this correspondence is bijective and that all possible worlds have the same probability $0.5^{|C|}$. We can now explain why ϕ defines a reduction. Recall from Definition 3.3 that a possible world of G is *good* if it contains an s,t-path, and *bad* otherwise. Here is the formal statement:

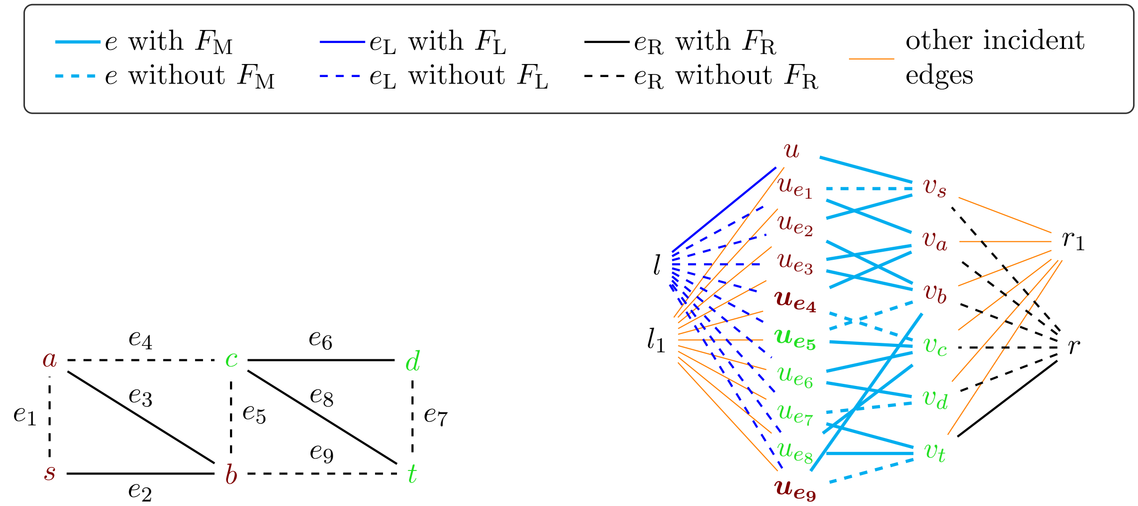
Proposition 7.5. *Let the TID $\mathcal{I} = (J, \pi)$ be the coding of an undirected st-graph G relative to an instance $I_{e,\Pi}$ and to F_{M} as described in Definition 7.4. Let ϕ be the bijective function defined above from the possible worlds of G to those of $\mathcal{I}$. Then:*

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

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

(1) Consider a witnessing path $s = w_1, \ldots, w_{n+1} = t$ in the possible world ω 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 the possible world $J' := \phi(\omega)$ actually has a subinstance isomorphic to $I_{e,\Pi}^{n+1}$. See Figure 6d for an example.

    To see why this is true, we take as usual the facts of J' that do not involve any copy of u or v and keep them as-is, because they occur in J' as they do in $I_{e,\Pi}^{n+1}$. Now, 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 ω, 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 $2n$ copies of e in alternating directions, we reach $v_t = v$, and finally we use the fact F_{R} which is incident to v. So, we have indeed found a subinstance of J' which is isomorphic to $I_{e,\Pi}^{n+1}$.
(2) Let us write $J' := \phi(\omega)$, and let us write $e = (u, v)$. Let us denote by I' the result of finely dissociating in I the edge e relative to the incident pair Π and the fact F_{M}: this is depicted in Figure 5. Let us show that J' has a homomorphism to I'. See Figure 7b for an example of a bad possible world J', and Figure 7a for the corresponding possible world ω of G.



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

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

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

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 ψ mapping each vertex of G to either L or R such that s is mapped to L, t is mapped to R, and every edge $\{x,y\}$ for which $\psi(x) \neq \psi(y)$ was not kept in ω. See Figure 7a for an illustration, where the red vertices are mapped to L and the green vertices are mapped to R.

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 $\psi(w) = \mathrm{L}$, 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 Figure 7b, this defines the image of v_a, v_b, and $u_{e_1}, u_{e_2}, u_{e_3}$ corresponding to the edges between them. In the same way we can map to v in I' all the nodes v_w such that $\psi(w) = \mathrm{R}$, including v_t and all edges between these nodes, going back-and-forth on edge (u', v) in I'. In Figure 7b, this defines the image of v_c, v_d, v_t, and $u_{e_6}, u_{e_7}, u_{e_8}$ corresponding to the edges between them.

We must still map the edges across the cut, i.e., edges $c = \{x,y\}$ such that $\psi(x) = \mathrm{L}$ and $\psi(y) = \mathrm{R}$. 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). In Figure 7b, this allows us to define the image of the bold vertices $(u_{e_4}, u_{e_5}, u_{e_9})$ corresponding to the edges across the cut. We follow the orientation choice when defining π, which can be seen by examining which edges are dashed, and we map u_{e_4} and u_{e_9} to u and map u_{e_5} to v.

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. □

Proposition 7.5 leads us to a proof of Theorem 7.1: good possible worlds of G give a possible world of $\mathcal{I}$ that satisfies Q thanks to the iterability of e, and bad possible worlds of G give a possible world of $\mathcal{I}$ having a homomorphism to the fine dissociation. The only missing piece is to argue that the fine dissociation does not satisfy the query. We can do this using the minimality and tightness of the pattern:

Lemma 7.6. *Let Q be a query, let (I, e) be a minimal tight pattern for Q, let Π be an arbitrary incident pair of e in I, and let F_{M} be an arbitrary non-unary fact covered by e in I. Then, the result of the fine dissociation of e in I relative to Π and F_{M} does not satisfy Q.*

Proof. We assume that the fine dissociation I_1 satisfies Q, and show a contradiction by rewriting it in several steps. The process of the proof is illustrated as Figure 8.

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 fine dissociation 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 6.5, 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 Q, it must mean that we can dissociate e_1', then e_1'' using the dissociation process of Definition 6.2 without violating Q. Formally, we first dissociate $e_1'' = (u, v)$ to remove this edge, rename u and v to u_1 and v_1, create u_2 and v_2, and add back copies of the edge from u_1 to v_2 and from u_2 to v_1. The dissociated edge is not tight as we argued, so Q is still satisfied in the result I_1'. Second, we dissociate $e_1' = (u', v')$, remove e_1', rename u' and v' to u_1' and v_1', create u_2' and v_2', and create copies of e_1' from u_1' to v_2' and from u_2' to v_1'. The dissociated edge e_1' has the same weight in I_1' as it did in I_1, so again it is not tight, and Q still holds in the result I_2. (See Figure 8.)

Note that u_2, v_2, u_2', v_2' are leaf vertices in I_2, which only occur 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 u_2, v_2, u_2' and v_2' 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');

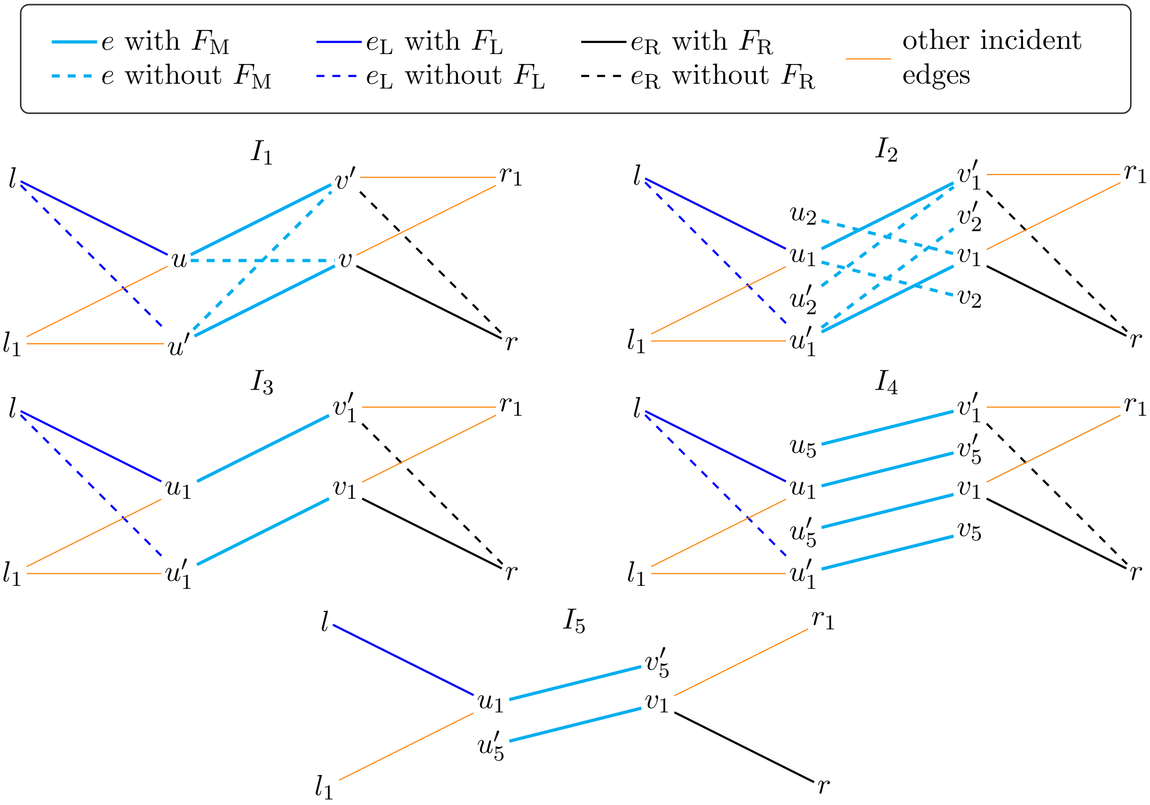


Figure 8. Illustration of the proof of Lemma 7.6, with I_1 being the fine dissociation I' of Figure 5, and I_5 being isomorphic to the dissociation on Figure 4.

- we map v'_2 to v_1 and map the edge (u'_1, v'_2) to the edge (u'_1, v_1).

The resulting instance I_3 (see Figure 8) is a homomorphic image of I_2, so it still satisfies Q. 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 Q, that I_3 also does.

Let us now modify I_3 using the second minimality criterion on e to dissociate the edges $e_4 = (u_1, v'_1)$ and $e'_4 = (u'_1, v_1)$, simplifying 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 in I_1, and v'_1 has the same number as v in I_1 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 6.5. Thus, we can dissociate one and then the other, and Q will still be satisfied. Say we first dissociate e_4: 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. Next, we dissociate e'_4, whose weight and side weight is unchanged relative to I_3: 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. As we argued, the minimality of e ensures that the edges that we dissociate are not tight, so the resulting instance I_4 (see Figure 8) still satisfies Q.

Now, we can finally merge back vertices to reach an instance I_5 isomorphic to the dissociation of e in I. This will yield our contradiction, because we assumed that e is tight. Specifically, 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 defines a homomorphism for the same reason. The resulting instance I_5 (see Figure 8) still satisfies Q. 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 Q. But as (I, e) was a tight pattern, this is impossible, so we have reached a contradiction and the proof is finished. □

We can now conclude the proof of the main result of the section, Theorem 7.1:

Proof of Theorem 7.1. Fix the query Q and the minimal tight pattern (I, e). By definition, e is then a non-leaf edge: pick an arbitrary incident pair Π and a non-unary fact F_{M} covered by e. We show the #P-hardness of PQE(Q) by reducing from U-ST-CON (Definition 3.3). Given an st-graph G, we apply the coding of Definition 7.4 and obtain a TID $\mathcal{I}$, which can be computed in polynomial time. As in the proof of Theorem 5.6, given a possible world ω of G, what matters is to show that (1.) if ω is good then $\phi(\omega)$ satisfies Q, and (2.) if ω is bad then $\phi(\omega)$ violates Q.

For this, we use Proposition 7.5. 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 (Definition 5.3), so it is iterable relative to any incident pair, in particular Π. Thus, the iterates satisfy Q, so $\phi(\omega)$ also does when ω is good. For (2.), we know by Lemma 7.6 that the result of the fine dissociation does not satisfy Q, so $\phi(\omega)$ does not satisfy it either when ω is bad. This establishes the correctness of the reduction and concludes the proof. □

We have thus established Theorem 4.3, and the main result of this paper.

8. Generalizations of the Dichotomy Result

This section presents two generalizations of our main result. We first show that the dichotomy also applies to a special case of probabilistic query evaluation, known as *generalized model counting*. Second, we strengthen the dichotomy result to the case where the signature can include unary predicates in addition to binary predicates.

A special case of probabilistic query evaluation. Recent work has studied the *generalized (first-order) model counting problem (GFOMC)* [KS21]: given a TID $\mathcal{I}$ where $\mathrm{P}_{\mathcal{I}}(t) \in \{0, 0.5, 1\}$ for every tuple $t \in \mathcal{I}$, GFOMC(Q) for a query Q is the problem of computing $\mathrm{P}_{\mathcal{I}}(Q)$. In other words, GFOMC($Q$) is a special case of PQE(Q), where each atom t in the TID can only have a probability $p \in \{0, 0.5, 1\}$.

To extend our result to this setting, we simply observe that all the reductions presented in this paper only use tuple probabilities from $\{0, 0.5, 1\}$. Thus, all our hardness results for PQE(Q) thus immediately apply to GFOMC(Q) and we obtain a corollary to our main hardness result (Theorem 4.3):

Corollary 8.1. *Let Q be an unbounded* UCQ^∞ *query over an arity-two signature. Then, the problem* GFOMC(Q) *is* #P*-hard.*

One can then ask if our dichotomy (Theorem 4.2) also generalizes to the GFOMC problem. Clearly, if PQE(Q) can be computed in polynomial time, then so can GFOMC(Q), and hence, for any safe UCQ Q, the GFOMC(Q) problem is immediately in FP by Dalvi and Suciu [DS12]. The other direction is more interesting, i.e., assuming a UCQ Q is unsafe for PQE, is it also unsafe for GFOMC? This was very recently shown to be true:

Theorem 8.2 (Theorem 2.2, [KS21])**.** *For any unsafe UCQ Q,* GFOMC(Q) *is* #P*-hard.*

In particular, this implies that all safe and unsafe queries coincide for UCQs, across the problems GFOMC and PQE. Then, combining this theorem with Corollary 8.1, we can state our dichotomy result also for GFOMC:

Theorem 8.3 (Dichotomy of GFOMC)**.** *Let Q be a* UCQ^∞ *over an arity-two signature. Then, either Q is equivalent to a safe UCQ and* GFOMC(Q) *is in* FP*, or it is not and* GFOMC(Q) *is* #P*-hard.*

Proof. Let Q be a safe UCQ. Then, since PQE(Q) is in FP, so is GFOMC(Q). If Q is not equivalent to a safe UCQ, then either it is equivalent to an unsafe UCQ and GFOMC(Q) is #P-hard by Theorem 2.2 of [KS21], or it is an unbounded query in UCQ^∞ and GFOMC(Q) is #P-hard by Corollary 8.1. □

Allowing unary predicates. We now turn to the question of extending our results to support unary predicates. Recall that we claimed in Section 3 that our results extend if the signature can feature unary and binary predicates. We now justify this claim formally by showing the analogue of Theorem 4.3 and Corollary 8.1 for signatures with relations of arity 1 and 2.

Theorem 8.4. *Let Q be an unbounded* UCQ^∞ *over a signature with relations of arity 1 and 2. Then,* GFOMC(Q)*, and hence* 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. Considering Q as an infinite union of CQs, we define Q' as a UCQ^∞ on σ' obtained by replacing every unary atom $R(x)$ in Q with the atom $R'(x, x)$. The resulting query Q' is unbounded. Indeed, assume to the contrary that Q' is equivalent to a UCQ Q''. As the truth of Q' by construction only depends on the presence or absence of facts of the form $R'(a, a)$, not $R'(a, b)$ with $a \neq b$, we can assume that Q'' only contains atoms of the form $R'(x, x)$ and not $R'(x, y)$. Now, replacing back each atom $R'(x, x)$ in Q'' with $R(x)$, we would obtain a UCQ that is equivalent to Q over the signature σ, contradicting the unboundedness of Q.

Thus, as Q' is an unbounded UCQ^∞, we know by Theorem 4.3 and Corollary 8.1 that GFOMC(Q') is #P-hard. Moreover, again by construction of Q', its satisfaction does not depend on the presence or absence of facts of the form $R'(a, b)$ with $a \neq b$. This implies that GFOMC(Q') is #P-hard even when assuming that the input TIDs contain no such facts.

Now, to show that GFOMC(Q) is #P-hard, we reduce from GFOMC(Q') where input TIDs are restricted to satisfy this additional assumption. Consider such a TID $\mathcal{I}' = (I', \pi')$. Consider the function ϕ that maps any instance I over σ' to the instance $\phi(I)$ obtained by replacing each fact $R'(a, a)$ by the fact $R(a)$. We build in polynomial time the TID $\mathcal{I} = (I, \pi)$ on σ, with $I := \phi(I')$, and with 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 1 in I the probability of the fact $R'(a, a)$ in I'. Then, ϕ defines a probability-preserving bijection between the possible worlds of $\mathcal{I}'$ and the possible worlds of $\mathcal{I}'$, and by construction ϕ guarantees that a possible world of $\mathcal{I}'$ satisfies Q' iff its ϕ image satisfies Q. This establishes that the reduction is correct, and concludes the proof. □

9. Conclusions

We have shown that PQE is #P-hard for any unbounded UCQ$^\infty$ over an arity-two signature, and hence proved 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. Our result captures many query languages; in particular disjunctive Datalog over binary signatures, regular path queries, and all ontology-mediated queries closed under homomorphisms.

There are three natural directions to extend our result. First, we could study queries that are *not* homomorphism-closed, e.g., with disequalities or negation. We believe that this would require different techniques as the problem is still open even when extending UCQs in this fashion (beyond the results of [FO16]). Second, we could lift the arity restriction and work over signatures of arbitrary arity: we conjecture that PQE is still #P-hard for any unbounded UCQ$^\infty$ in that case. Much of our proof techniques may adapt, but we do not know how to extend the definitions of dissociation, fine dissociation, and iteration. In particular, dissociation on a fact is difficult to adapt because incident facts over arbitrary arity signatures may intersect in complicated ways. We believe that the result could extend with a suitable dissociation notion and tight patterns with a more elaborate minimality criterion, but for now we leave the extension to arbitrary-arity signatures to future work. Third, a natural question for future work is whether our hardness result on unbounded homomorphism-closed queries also applies to the *(unweighted) model counting problem*, where all facts of the TID must have probability 0.5: the hardness of this problem has only been shown recently on the class of self-join free CQs [AK21] and on the so-called unsafe final type-I queries [KS21], but remains open as of this writing for unsafe UCQs in general.

Acknowledgments

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

References

[ABS16] 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)*. ACM, 2016.

[AC20] Antoine Amarilli and İsmail İlkan Ceylan. A dichotomy for homomorphism-closed queries on probabilistic graphs. In *Proceedings of the 23rd International Conference on Database Theory (ICDT-20)*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020.

[AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of databases*. Addison-Wesley, 1995.

[AK21] Antoine Amarilli and Benny Kimelfeld. Uniform reliability of self-join-free conjunctive queries. In *Proceedings of the 24th International Conference on Database Theory (ICDT-21)*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2021.

[Bar13] Pablo Barceló. Querying graph databases. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS-13)*, 2013.

[BBLP18] Pablo Barceló, Gerald Berger, Carsten Lutz, and Andreas Pieris. First-order rewritability of frontier-guarded ontology-mediated queries. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI-18)*. IJCAI, 2018.

[BCL17] 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)*. AAAI Press, 2017.

[BCL19] 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.

[BCLW14] 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.

[BCM$^+$07] Franz Baader, Diego Calvanese, Deborah L McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic handbook*. Cambridge University Press, 2007.

[BFR19] Pablo Barceló, Diego Figueira, and Miguel Romero. Boundedness of conjunctive regular path queries. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP-19)*, 2019. `doi:10.4230/LIPIcs.ICALP.2019.104`.

[BTCCB15] 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*. IEEE, 2015.

[BUGD$^+$13] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS-13)*. Curran Associates Inc., 2013.

[CDV21] İsmail İlkan Ceylan, Adnan Darwiche, and Guy Van den Broeck. Open-world probabilistic databases: Semantics, algorithms, complexity. *Artificial Intelligence*, 295, 2021.

[Cey17] İsmail İlkan Ceylan. *Query answering in probabilistic data and knowledge bases*. Doctoral thesis, TU Dresden, 2017.

[CGK13] Andrea Calì, Georg Gottlob, and Michael Kifer. Taming the infinite chase: Query answering under expressive relational constraints. *JAIR*, 48:115–174, 2013.

[CGL12] 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.

[Coo71] 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.

[DGH$^+$14] 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*. ACM, 2014.

[DK08] Anuj Dawar and Stephan Kreutzer. On Datalog vs. LFP. In *International Colloquium on Automata, Languages, and Programming (ICALP-08)*. Springer, 2008.

[DRDT$^+$15] Luc De Raedt, Anton Dries, Ingo Thon, Guy Van den Broeck, and Mathias Verbeke. Inducing probabilistic relational rules from probabilistic examples. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI-15)*. AAAI Press, 2015.

[DS07] Nilesh Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. *The VLDB Journal*, 16(4):523–544, 2007.

[DS12] Nilesh Dalvi and Dan Suciu. The dichotomy of probabilistic inference for unions of conjunctive queries. *J. ACM*, 59(6), 2012.

[EOŠ+12] Thomas Eiter, Magdalena Ortiz, Mantas Šimkus, Trung-Kien Tran, and Guohui Xiao. Query rewriting for Horn-$\mathcal{SHIQ}$ plus rules. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI-12)*, 2012.

[FO16] 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.

[GMSV93] 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.

[GS12] Georg Gottlob and Thomas Schwentick. Rewriting ontological queries into small nonrecursive Datalog programs. In *Proceedings of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR-12)*, 2012.

[HKMV95] 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.

[HSBW13] 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.

[JL12] 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 (ISWC-12)*, pages 182–197. Springer-Verlag, 2012.

[JL20] Jean Christoph Jung and Carsten Lutz. Erratum for 'Ontology-based access to probabilistic data with OWL-QL', 2020.

[Jun14] Jean Christoph Jung. *Reasoning in many dimensions: uncertainty and products of modal logics*. PhD thesis, University of Bremen, 2014.

[KS21] Batya Kenig and Dan Suciu. A dichotomy for the generalized model counting problem for unions of conjunctive queries. In *Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS-21)*, 2021.

[MBSJ09] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011. ACL, 2009.

[MCH+15] 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)*, 2015.

[OH08] Dan Olteanu and Jiewen Huang. Using OBDDs for efficient query evaluation on probabilistic databases. In *Proceedings of the 2nd International Conference on Scalable Uncertainty Management (SUM-08)*, volume 5291 of *LNCS*, 2008.

[OH09] 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.

[PB83] 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.

[RS09] Christopher Ré and Dan Suciu. The trichotomy of HAVING queries on a probabilistic database. *The VLDB Journal*, 18(5):1091–1116, 2009.

[SORK11] Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. *Probabilistic Databases*, volume 3. Morgan-Claypool, 2011.

[Val79] Leslie Gabriel Valiant. The complexity of computing the permanent. *TCS*, 8(2):189–201, 1979.