Logical Methods in Computer Science
Volume 21, Issue 4, 2025, pp. 30:1–30:31
https://lmcs.episciences.org/

Submitted Nov. 08, 2024
Published Dec. 15, 2025

APPROXIMATING QUERIES ON PROBABILISTIC GRAPHS *

ANTOINE AMARILLI [a], TIMOTHY VAN BREMEN [b], OCTAVE GASPARD [c], AND KULDEEP S. MEEL [d]

[a] Univ. Lille, Inria Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France; LTCI, Télécom Paris, Institut polytechnique de Paris
e-mail address: antoine.a.amarilli@inria.fr

[b] Nanyang Technological University
e-mail address: timothy.vanbremen@ntu.edu.sg

[c] École Polytechnique
e-mail address: octave.gaspard@polytechnique.edu

[d] University of Toronto
e-mail address: meel@cs.toronto.edu

Abstract. Query evaluation over probabilistic databases is notoriously intractable—not only in combined complexity, but often in data complexity as well. This motivates the study of approximation algorithms, and particularly of *combined FPRASes*, with runtime polynomial in both the query and instance size. In this paper, we focus on tuple-independent probabilistic databases over binary signatures, i.e., *probabilistic graphs*, and study when we can devise combined FPRASes for probabilistic query evaluation.

We settle the complexity of this problem for a variety of query and instance classes, by proving both approximability results and (conditional) inapproximability results together with (unconditional) DNNF provenance circuit size lower bounds. This allows us to deduce many corollaries of possible independent interest. For example, we show how the results of Arenas et al. [ACJR21a] on counting fixed-length strings accepted by an NFA imply the existence of an FPRAS for the two-terminal network reliability problem on directed acyclic graphs, a question asked by Zenklusen and Laumanns [ZL11]. We also show that one cannot extend a recent result of van Bremen and Meel [vBM23] giving a combined FPRAS for self-join-free conjunctive queries of bounded hypertree width on probabilistic databases: neither the bounded-hypertree-width condition nor the self-join-freeness hypothesis can be relaxed. We last show how our methods can give insights on the evaluation and approximability of *regular path queries* (RPQs) on probabilistic graphs in the data complexity perspective, showing in particular that some of them are (conditionally) inapproximable.

* This is an extended version of an ICDT'24 conference paper [AvBM24]. It fixes the proof of Proposition 4.1 and other minor errors, and includes new material on RPQs from Gaspard's bachelor thesis [Gas24].

This project was supported in part by the National Research Foundation Singapore under its NRF Fellowship programme [NRF-NRFFAI1-2019-0004] and Campus for Research Excellence and Technological Enterprise (CREATE) programme, as well as the Ministry of Education Singapore Tier 1 and 2 grants R-252-000-B59-114 and MOE-T2EP20121-0011. Amarilli was partially supported by the ANR project EQUUS ANR-19-CE48-0019, by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 431183758, and by the ANR project ANR-18-CE23-0003-02 ("CQFD"). This work was done in part while Amarilli was visiting the Simons Institute for the Theory of Computing.

LOGICAL METHODS IN COMPUTER SCIENCE
DOI:10.46298/LMCS-21(4:30)2025

1. Introduction

Tuple-independent probabilistic databases (TID) are a simple and principled formalism to model uncertainty and noise in relational data [DS04, SORK11, DS12]. In the TID model, each tuple of a relational database is annotated with an independent probability of existence; all tuples are assumed to be independent. In the *probabilistic query evaluation* (PQE) problem, given a Boolean query Q and a TID instance I, we must compute the probability that Q holds in a subinstance sampled from I according to the resulting distribution. The PQE problem has been studied in database theory both in terms of *combined complexity*, where the query and instance are part of the input, and in *data complexity*, where the query is fixed and only the instance is given as input [Var82]. Unfortunately, many of the results so far [SORK11] show that the PQE problem is highly intractable, even in data complexity for many natural queries (e.g., a path query of length three), and hence also in combined complexity.

Faced with this intractability, a natural approach is to study *approximate PQE*: we relax the requirement of computing the exact probability that the query holds, and settle for an approximate answer. This approach has been studied in data complexity [SORK11]: for any fixed union of conjunctive queries (UCQ), we can always tractably approximate the answer to PQE, additively (simply by Monte Carlo sampling), or multiplicatively (using the Karp-Luby approximation algorithm on a disjunctive-normal-form representation of the query provenance). However, these approaches are not tractable in combined complexity, and moreover the latter approach exhibits a "slicewise polynomial" runtime of the form $O(|I|^{|Q|})$—rather than, say, $O(2^{|Q|}\mathsf{poly}(|I|))$—which seriously limits its practical utility. Thus, our goal is to obtain a *combined FPRAS* for PQE: by this we mean a fully polynomial-time randomized approximation scheme, giving a multiplicative approximation of the probability, whose runtime is polynomial in the query and TID (and in the desired precision). This approach has been recently proposed by van Bremen and Meel [vBM23], who show a combined FPRAS for CQs when assuming that the query is self-join-free and has bounded hypertree width; their work leaves open the question of which other cases admit combined FPRASes.

Main Results. In this paper, following the work of Amarilli, Monet and Senellart [AMS17] for exact PQE, we investigate the combined complexity of *approximate* PQE in the setting of *probabilistic graphs*. In other words, we study *probabilistic graph homomorphism*, which is the equivalent analogue of CQ evaluation: given a (deterministic) query graph G, and given a instance graph H with edges annotated with independent probabilities (like a TID), we wish to approximate the probability that a randomly selected subgraph $H' \subseteq H$ admits a homomorphism from G. This setting is incomparable to that of [vBM23], because it allows for self-joins and for queries of unbounded width, but assumes that relations are binary.

Of course, the graph homomorphism problem is intractable in combined complexity if the input graphs are arbitrary (even without probabilities). Hence, we study the problem when the query graph and instance graph are required to fall in restricted graph classes, chosen to ensure tractability in the non-probabilistic setting. We use similar classes as those from [AMS17]: *path graphs* which may be *one-way* (1WP: all edges are oriented from left to right) or *two-way* (2WP: edge orientations are arbitrary); *tree graphs* which may be *downward* (DWT: all edges are oriented from the root to the leaves) or *polytrees* (PT: edge orientations are arbitrary); and, for the instance graph, *directed acyclic graphs* (DAG), or *arbitrary graphs* (All).

For all combinations of these classes, we show either (i) the existence of a combined FPRAS, or (ii) the non-existence of such an FPRAS, subject to standard complexity-theoretic assumptions. We summarize our results in Table 1, respectively for graphs that are *labelled* (i.e., the signature features several binary relations), or *unlabelled* (i.e., only one binary relation).

Result 1.1 (Sections 3 and 4)**.** *The results in Table 1, described in terms of the graph classes outlined above, hold.*

In summary, for the classes that we consider, our results mostly show that the general intractability of combined PQE carries over to the approximate PQE problem. The important exception is Proposition 3.1: the PQE problem for one-way path queries on *directed acyclic graphs* (DAGs) admits a combined FPRAS. We discuss more in detail below how this result is proved and some of its consequences. Another case is left open: in the unlabelled setting, we do not settle the approximability of combined PQE for one-way path queries (or equivalently downward tree queries) on arbitrary graphs. For all other cases, either exact combined PQE was already shown to be tractable in the exact setting [AMS17], or we strengthen the #P-hardness of exact PQE from [AMS17] by showing that combined FPRASes conditionally do not exist. We stress that our results always concern *multiplicative approximations*: as non-probabilistic graph homomorphism is tractable for the classes that we consider, we can always obtain additive approximations for PQE simply by Monte Carlo sampling. Further note that our intractability results are always shown in *combined complexity*—in data complexity, for the queries that we consider, PQE is always multiplicatively approximable via the Karp-Luby algorithm [SORK11].

As an important consequence, our techniques yield connections between approximate PQE and *intensional* approaches to the PQE problem. Recall that the intensional approach was introduced by Jha and Suciu [JS13] in the setting of exact evaluation, and when measuring data complexity. They show that many tractable queries for PQE also admit tractable provenance representations. More precisely, for these queries Q, there is a polynomial-time algorithm that takes as input any database instance and computes a representation of the Boolean provenance of Q in a form which admits tractable model counting (e.g., OBDD, d-DNNF, etc.). This intensional approach contrasts with *extensional* approaches (like [DS12]) which exploit the structure of the query directly: comparing the power of intensional and extensional approaches is still an open problem [Mon20].

In line with this intensional approach, we complement our conditional hardness results on approximate PQE with *unconditional* lower bounds on the *combined* size of tractable representations of query provenance. Namely, we show a moderately exponential lower bound on DNNF provenance representations which applies to all our non-approximable query-instance class pairs:

Result 1.2 (Section 5, informal)**.** *Let $\langle \mathcal{G}, \mathcal{H} \rangle$ be a conditionally non-approximable query-instance class pair studied in this paper. For any $\epsilon > 0$, there is an infinite family $G_1, G_2, \ldots$ of $\mathcal{G}$ queries and an infinite family $H_1, H_2, \ldots$ of $\mathcal{H}$ instances such that, for every $i > 0$, every DNNF circuit representing the provenance $\mathsf{Prov}_{H_i}^{G_i}$ has size at least $2^{\Omega\left((||G_i||+||H_i||)^{1-\epsilon}\right)}$.*

The class of DNNF circuits is arguably the most succinct circuit class in knowledge compilation that still has desirable properties [Dar01, DM02]. Such circuits subsume in particular the class of *structured DNNFs*, for which tractable approximation algorithms were

recently proposed [ACJR21b]. Thus, these bounds help to better understand the limitations of intensional approaches.

Moreover, since we also show *strongly* exponential lower bounds for the treewidth-1 query class 1WP in particular, our results give an interesting example of a CQ class for which (non-probabilistic) query evaluation is in linear-time combined complexity [Yan81], but the size of every DNNF representation of query provenance is exponential:

Result 1.3 (Proposition 5.1)**.** *There exists an infinite family $G_1, G_2, \ldots$ of treewidth-1 CQs, and an infinite family $H_1, H_2, \ldots$ of instances on a fixed binary signature such that, for every $i > 0$, every DNNF circuit representing the provenance $\mathsf{Prov}_{H_i}^{G_i}$ has size at least $2^{\Omega(||G_i||+||H_i||)}$.*

Note that this result stands in contrast to the data complexity perspective, where the provenance of *every* fixed CQ—no matter its treewidth—admits a polynomially-sized DNNF circuit representation, more precisely as a provenance formula in disjunctive normal form (DNF).

Consequences. Our results and techniques have several interesting consequences of potential independent interest. First, they imply that we cannot relax the hypotheses of the result of van Bremen and Meel mentioned earlier [vBM23]. They show the following result on combined FPRASes for PQE in the more general context of probabilistic databases:

Theorem 1.4 (Theorem 1 of [vBM23])**.** *Let Q be a self-join-free conjunctive query of bounded hypertree width, and H a tuple-independent database instance. Then there exists a combined FPRAS for computing the probability of Q on H, i.e., an FPRAS whose runtime is $\mathsf{poly}(|Q|, ||H||, \epsilon^{-1})$, where ϵ is the multiplicative error.*

It was left open in [vBM23] whether intractability held without these assumptions on the query. Hardness is immediate if we do not bound the width of queries and allow arbitrary self-join-free CQs, as combined query evaluation is then NP-hard already in the non-probabilistic setting. However, it is less clear whether the self-join-freeness condition can be lifted. Our results give a negative answer, already in a severely restricted setting:

Result 1.5 (Corollaries 6.1 and 6.2)**.** *Assuming $\mathsf{RP} \neq \mathsf{NP}$, neither the bounded hypertree width nor self-join-free condition in Theorem 1.4 can be relaxed: even on a fixed signature consisting of a single binary relation, there is no FPRAS to approximate the probability of an input treewidth-1 CQ on an input treewidth-1 TID instance.*

A second consequence implied by our techniques concerns the *two-terminal network reliability problem* (ST-CON) on directed acyclic graphs (DAGs). Roughly speaking, given a directed graph $G = (V, E)$ with independent edge reliability probabilities $\pi : E \to [0, 1]$, and two distinguished vertices $s, t \in V$, the ST-CON problem asks for the probability that there is a path from s to t. The problem is known to be #P-hard even on DAGs [PB83, Table 2]. The existence of an FPRAS for ST-CON is a long-standing open question [Kan94], and the case of DAGs was explicitly left open by Zenklusen and Laumanns [ZL11]. Our results allow us to answer in the affirmative:

Result 1.6 (Theorem 6.3)**.** *There exists an FPRAS for the ST-CON problem over DAGs.*

This result and our approximability results follow from the observation that path queries on directed acyclic graphs admit a compact representation of their Boolean provenance as

non-deterministic ordered binary decision diagrams (nOBDDs). We are then able to use a recent result by Arenas et al. [ACJR21a, Corollary 4.5] giving an FPRAS for counting the satisfying assignments of an nOBDD, adapted to the weighted setting.

We last explore a third consequence of our work by studying the PQE problem for *regular path queries* (RPQs), i.e., queries asking for the existence of a walk in the graph (of arbitrary length) that forms a word belonging to a regular expression. Unlike the other queries studied in this work, RPQs are generally not expressible as conjunctive queries, and so PQE is not necessarily approximable even in the data complexity perspective. Thus, specifically for RPQs, we study PQE in data complexity rather than combined complexity. Of course, some RPQs are equivalent to UCQs, for instance those with regular expressions describing a finite language, and so are approximable. We show that, for all other RPQs (so-called *unbounded* RPQs), the PQE problem is at least as hard as ST-CON in data complexity. For some unbounded RPQs, we also show (conditional) inapproximability. This result does not directly follow from our combined complexity results, but uses very similar techniques.

Paper Structure. In Section 2, we review some of the technical background. We then present our main results on approximability, divided into the labelled and unlabelled case, in Sections 3 and 4 respectively. Next, in Section 5, we show lower bounds on DNNF provenance circuit sizes. In Section 6, we show some consequences for previous work [vBM23], as well as for the two-terminal network reliability problem. We show consequences for the data complexity of PQE for RPQs in Section 7. We conclude in Section 8.

2. Preliminaries

We provide some technical background below, much of which closely follows that in [ACMS20] and [AMS17].

Graphs and Graph Homomorphisms. Let σ be a non-empty finite set of labels called the *signature*. When $|\sigma| = 1$, we say that we are in the *unlabelled setting*; otherwise, we say we are in the *labelled setting*. In this paper, we study only *directed* graphs with edge labels from σ. A graph G over σ is a tuple (V, E, λ) with finite non-empty vertex set V, edge set $E \subseteq V^2$, and $\lambda\colon E \to \sigma$ a labelling function mapping each edge to a single label (we may omit λ in the unlabelled setting). The *size* $||G||$ of G is its number of edges. We write $x \xrightarrow{R} y$ for an edge $e = (x, y) \in E$ with label $\lambda(e) = R$, and $x \to y$ for $(x, y) \in E$ (no matter the edge label): we say that x is the *source* of e and that y is the *target* of e. We sometimes use a simple regular-expression-like syntax (omitting the vertex names where irrelevant) to represent path graphs: for example, we write $\to\to$ to represent an unlabelled path of length two, and the notation $\to^k$ to denote an unlabelled path of length k. All of this syntax extends to labelled graphs in the obvious way. A graph $H = (V', E', \lambda')$ is a *subgraph* of G, written $H \subseteq G$, if $V = V'$, $E' \subseteq E$, and λ' is the restriction of λ to E'.

A *graph homomorphism* h from a graph $G = (V_G, E_G, \lambda_G)$ to a graph $H = (V_H, E_H, \lambda_H)$ is a function $h : V_G \to V_H$ such that, for all $(u, v) \in E_G$, we have $(h(u), h(v)) \in E_H$ and $\lambda_H((h(u), h(v))) = \lambda_G((u, v))$. We write $G \rightsquigarrow H$ to say that such a homomorphism exists, and sometimes refer to a homomorphism from G to H as a *match* of G in H.

Probabilistic Graphs and Probabilistic Graph Homomorphisms. A *probabilistic graph* is a pair (H, π), where H is a graph with edge labels from σ, and $\pi : E \to [0, 1]$ is a probability labelling on the edges. Note that edges e in H are annotated both by their probability value $\pi(e)$ and their σ-label $\lambda(e)$. Intuitively, π gives us a succinct specification of a probability distribution over the $2^{||H||}$ possible subgraphs of H, by independently including each edge e with probability $\pi(e)$. Formally, the distribution induced by π on the subgraphs $H' \subseteq H$ is defined by $\Pr_\pi(H') = \prod_{e \in E'} \pi(e) \prod_{e \in E \setminus E'} (1 - \pi(e))$.

In this paper, we study the *probabilistic graph homomorphism* problem PHom: given a graph G called the *query graph* and a probabilistic graph (H, π) called the *instance graph*, with G and H carrying labels from the same signature σ, we must compute the probability $\Pr_\pi(G \rightsquigarrow H)$ that a subgraph of H, sampled according to the distribution induced by π, admits a homomorphism from G. That is, we must compute $\Pr_\pi(G \rightsquigarrow H) := \sum_{H' \subseteq H \text{ s.t. } G \rightsquigarrow H'} \Pr_\pi(H')$.

We study PHom in *combined complexity*, i.e., when the query graph G, instance graph (H, π), and signature σ are all given as input. Further, we study PHom when we restrict G and H to be taken from specific *graph classes*, i.e., infinite families of (non-probabilistic) graphs, denoted respectively $\mathcal{G}$ and $\mathcal{H}$. (Note that $\mathcal{H}$ does not restrict the probability labelling π.) To distinguish the *labelled* and *unlabelled* setting, we denote by $\mathsf{PHom}_{\mathsf{L}}(\mathcal{G}, \mathcal{H})$ the problem of computing $\Pr_\pi(G \rightsquigarrow H)$ for $G \in \mathcal{G}$ and (H, π) with $H \in \mathcal{H}$ when no restriction is placed on the input signature σ for graphs in $\mathcal{G}$ and $\mathcal{H}$. On the other hand, we write $\mathsf{PHom}_{\not\mathsf{L}}(\mathcal{G}, \mathcal{H})$ when $\mathcal{G}$ and $\mathcal{H}$ are restricted to be classes of unlabelled graphs, i.e., $|\sigma| = 1$. We focus on *approximation algorithms*: fixing classes $\mathcal{G}$ and $\mathcal{H}$, a *fully polynomial-time randomized approximation scheme* (FPRAS) for $\mathsf{PHom}_{\mathsf{L}}(\mathcal{G}, \mathcal{H})$ (in the labelled setting) or $\mathsf{PHom}_{\not\mathsf{L}}(\mathcal{G}, \mathcal{H})$ (in the unlabelled setting) is a randomized algorithm that runs in time $\mathsf{poly}(||G||, ||H||, \epsilon^{-1})$ on inputs $G \in \mathcal{G}$, (H, π) for $H \in \mathcal{H}$, and $\epsilon > 0$. The algorithm must return, with probability at least $3/4$, a *multiplicative approximation* of the probability $\Pr_\pi(G \rightsquigarrow H)$, i.e., a value between $(1 - \epsilon) \Pr_\pi(G \rightsquigarrow H)$ and $(1 + \epsilon) \Pr_\pi(G \rightsquigarrow H)$.

Graph Classes. We study PHom on the following graph classes, which are defined on a graph G with edge labels from some signature σ, and thus can either be labelled or unlabelled depending on σ:

- G is a *one-way path* (1WP) if it is of the form $a_1 \xrightarrow{R_1} \dots \xrightarrow{R_{m-1}} a_m$ for some m, with all $a_1, \dots, a_m$ being pairwise distinct, and with $R_i \in \sigma$ for $1 \leq i < m$.
- G is a *two-way path* (2WP) if it is of the form $a_1 - \dots - a_m$ for some m, with pairwise distinct $a_1, \dots, a_m$, and each $-$ being $\xrightarrow{R_i}$ or $\xleftarrow{R_i}$ (but not both) for some label $R_i \in \sigma$.
- G is a *downward tree* (DWT) if it is a rooted unranked tree (each node can have an arbitrary number of children), with all edges pointing from parent to child in the tree.
- G is a *polytree* (PT) if its underlying undirected graph is a rooted unranked tree, without restrictions on the edge directions.
- G is a *DAG* (DAG) if it is a (directed) acyclic graph.

These refine the classes of connected queries considered in [AMS17], by adding the DAG class. We denote by All the class of all graphs. Note that both 2WP and DWT generalize 1WP and are incomparable; PT generalizes both 2WP and DWT; DAG generalizes PT; All generalizes DAG (see Figure 2 of [AMS17]).

Note that our notion of labelled graphs, and the classes of graphs defined above, are different from the notion of database instances, where we would typically allow two vertices

to be connected by multiple edges with different labels. Formally, for a non-empty finite signature σ, an *arity-two database* over σ consists of an *active domain* V and a set E of labelled edges of the form (u, a, v) with $u, v \in V$ and $a \in \sigma$. The notion of a probabilistic arity-two database is defined in the expected way by giving a probability to every edge of E and assuming independence across all edges (in particular all edges having the same endpoints are also independent). A graph over σ can be seen as an arity-two database where for every pair $(u, v) \in V \times V$ there exists at most one edge in E of the form (u, a, v) for some $a \in \sigma$. Note that in the unlabelled setting with $|\sigma| = 1$ there is no difference between both notions. For simplicity, all our results will be phrased in terms of graphs and not of arity-two databases; in particular, all our lower bounds will apply in the restricted setting of graphs that we study. However, all our upper bound results will in fact also hold when allowing arity-two databases as input—we mention this explicitly when stating these results.

Boolean Provenance. We use the notion of *Boolean provenance*, or simply *provenance* [IJ84, ABS15, Sen19]. In the context of databases, provenance intuitively represents which subsets of the instance satisfy the query: it is used in the intensional approach to probabilistic query evaluation [JS13]. In this paper, we use provenance to show both upper and lower bounds.

Formally, let $G = (V_G, E_G, \lambda_G)$ and $H = (V_H, E_H, \lambda_H)$ be graphs. Seeing E_H as a set of Boolean variables, a *valuation* ν of E_H is a function $\nu \colon E_H \to \{0, 1\}$ that maps each edge of H to 0 or 1. Such a valuation ν defines a subgraph H_ν of H where we only keep the edges mapped to 1, formally $H_\nu = (V_H, \{e \in E_H \mid \nu(e) = 1\}, \lambda_H)$. The *provenance* of G on H is then the Boolean function Prov_H^G having as variables the edges E_H of H and mapping every valuation ν of E_H to 1 (true) or 0 (false) depending on whether $G \rightsquigarrow H_\nu$ or not. Generalizing this definition, for any integer n, for any choice of $a_1, \ldots, a_n \in V_G$ and $b_1, \ldots, b_n \in V_H$, we write $\mathsf{Prov}_H^G[a_1 := b_1, \ldots, a_n := b_n]$ to denote the Boolean function that maps valuations ν of E_H to 1 or 0 depending on whether or not there is a homomorphism $h \colon G \to H_\nu$ which additionally satisfies $h(a_i) = b_i$ for all $1 \leq i \leq n$.

For our lower bounds, we will often seek to represent Boolean formulas as the provenance of queries on graphs:

Definition 2.1. Given two graphs G and H, and a Boolean formula ϕ whose variables $\{e_1, \ldots, e_n\} \subseteq E_H$ are edges of H, we say that Prov_H^G represents ϕ on $(e_1, ..., e_n)$ if for every valuation $\nu : E_H \to \{0, 1\}$ that maps edges not in $\{e_1, ..., e_n\}$ to 1, we have $\nu \models \phi$ if and only if $\mathsf{Prov}_H^G(\nu) = 1$.

Circuits and Knowledge Compilation. We consider representations of Boolean functions in terms of *non-deterministic (ordered) binary decision diagrams*, as well as *decomposable circuits*, which we define below.

A *non-deterministic binary decision diagram* (nBDD) on a set of variables $V = \{v_1, \ldots, v_n\}$ is a rooted DAG D whose nodes carry a label in $V \sqcup \{0, 1, \vee\}$ (using $\sqcup$ to denote disjoint union), and whose edges can carry an optional label in $\{0, 1\}$ subject to the following requirements:

(1) there are exactly two leaves (called *sinks*), one labelled by 1 (the 1-*sink*), and the other by 0 (the 0-*sink*);
(2) internal nodes are labelled either by $\vee$ (called an $\vee$-*node*) or by a variable of V (called a *decision node*); and

(3) each decision node has exactly two outgoing edges, labelled 0 and 1; the outgoing edges of $\vee$-nodes are unlabelled.

The size $||D||$ of D is its number of edges. Let ν be a valuation of V, and let π be a path in D going from the root to one of the sinks. We say that π is *compatible* with ν if for every decision node n of the path, letting $v \in V$ be the variable labelling n, then π passes through the outgoing edge of n labelled with $\nu(v)$. In particular, no constraints are imposed at $\vee$-nodes; thus, we may have that multiple paths are compatible with a single valuation. The nBDD D *represents* a Boolean function, also written D by abuse of notation, which is defined as follows: for each valuation ν of V, we set $D(\nu) := 1$ if there exists a path π from the root to the 1-sink of D that is compatible with ν, and set $D(\nu) := 0$ otherwise. Given an nBDD D over variables V, we denote by $\mathsf{Mods}(D)$ the set of satisfying valuations ν of D such that $D(\nu) = 1$, and by $\mathsf{MC}(D)$ the number $|\mathsf{Mods}(D)|$ of such valuations. Further, given a rational probability function $w : V \to [0, 1]$ on the variables of V, define $\mathsf{WMC}(D, w)$ to be the probability that a random valuation ν satisfies D, that is, $\mathsf{WMC}(D, w) = \sum_{\nu \in \mathsf{Mods}(D)} \prod_{x \in V \text{ s.t. } \nu(x)=1} w(x) \prod_{x \in V \text{ s.t. } \nu(x)=0} (1 - w(x))$.

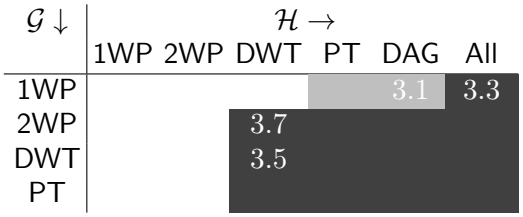
In this paper, we primarily focus on a subclass of nBDDs called *non-deterministic ordered binary decision diagrams* (nOBDDs). An nOBDD D is an nBDD for which there exists a strict total order $\prec$ on the variables V such that, for any two decision nodes $n \neq n'$ such that there is a path from n to n', then, letting v and v' be the variables that respectively label n and n', we have $v \prec v'$. This implies that, along any path going from the root to a sink, the sequence of variables will be ordered according to V, with each variable occurring at most once. We use nOBDDs because they admit tractable approximate counting of their satisfying assignments, as we discuss later.

We also show lower bounds on a class of *circuits*, called *decomposable negation normal form* (DNNF) circuits. A *circuit* on a set of variables V is a directed acyclic graph $C = (G, W)$, where G is a set of *gates*, where $W \subseteq G \times G$ is a set of edges called *wires*, and where we distinguish an *output gate* $g_0 \in G$. The *inputs* of a gate $g \in G$ are the gates g' such that there is a wire (g', g) in W. The gates can be labelled with variables of V (called a *variable gate*), or with the Boolean operators $\vee$, $\wedge$, and $\neg$. We require that gates labelled with variables have no inputs, and that gates labelled with $\neg$ have exactly one input. A circuit C defines a Boolean function on V, also written C by abuse of notation. Formally, given a valuation ν of V, we define inductively the *evaluation* ν' of the gates of C by setting $\nu'(g) := \nu(v)$ for a variable-gate g labelled with variable v, and setting $\nu'(g)$ for other gates to be the result of applying the Boolean operators of g to $\nu'(g_1), \ldots, \nu'(g_n)$ for the inputs $g_1, \ldots, g_n$ of g. We then define $C(\nu)$ to be $\nu'(g_0)$ where g_0 is the output gate of C.

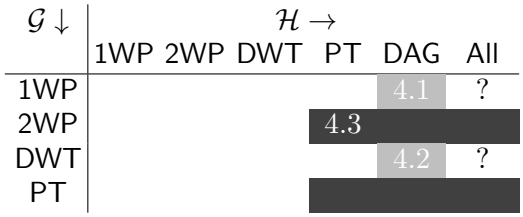
The circuit is in *negation normal form* if negations are only applied to variables, i.e., for every $\neg$-gate, its input is a variable gate. The circuit is *decomposable* if the $\wedge$-gates always apply to inputs that depend on disjoint variables: formally, there is no $\wedge$-gate g with two distinct inputs g_1 and g_2, such that some variable v labels two variable gates g_1' and g_2' with g_1' having a directed path to g_1 and g_2' having a directed path to g_2. A *DNNF* is a circuit which is both decomposable and in negation normal form. Note that we can translate nOBDDs in linear time to DNNFs, more specifically to *structured DNNFs* [ACMS20, Proposition 3.8].

Approximate Weighted Counting for nOBDDs. Recently, Arenas et al. [ACJR21a] showed the following result on approximate counting of satisfying assignments of an nOBDD.

Theorem 2.2 (Corollary 4.5 of [ACJR21a])**.** *Let D be an nOBDD. Then there exists an FPRAS for computing $\mathsf{MC}(D)$.*



(A) Complexity of $\mathsf{PHom_L}(\mathcal{G},\mathcal{H})$.



(B) Complexity of $\mathsf{PHom_{\not L}}(\mathcal{G},\mathcal{H})$.

TABLE 1. Results on approximation proved in this paper. Key: white ( ) means that the problem is in PTIME; light grey (■) means that it is #P-hard but admits an FPRAS; dark grey (■) means #P-hardness and non-existence of an FPRAS, assuming RP $\neq$ NP. All cells without a reference to a corresponding proposition are either implied by one of the other results in this paper, or pertain to exact complexity and were already settled in [AMS17].

For our upper bounds, we need a slight strengthening of this result to apply to *weighted model counting* (WMC) in order to handle probabilities. This can be achieved by translating the approach used in [vBM23, Section 5.1] to the nOBDD setting. We thus show (see Appendix A):

Theorem 2.3. *Let D be an nOBDD on variables V, and let $w : V \to [0,1]$ be a rational probability function defined on V. Then there exists an FPRAS for computing $\mathsf{WMC}(D,w)$, running in time polynomial in $||D||$ and w.*

3. Results in the Labelled Setting

We now move on to the presentation of our results. We start with the *labelled* setting of probabilistic graph homomorphism, in which no restriction is placed on the signature σ of the query and instance graphs. Our results are summarized in Table 1a. Note that the *exact* tractability and hardness results shown in [AMS17] (and repeated in Table 1a) actually pertain to a slightly more restricted setting in which the signature σ is fixed, different to the setting studied here in which σ forms part of the problem input. Fortunately, it is easy to check that all results still hold: for the #P-hardness results, this is immediate, and for the PTIME tractability results, a careful inspection of the relevant claims ([AMS17, Proposition 4.10] and [AMS17, Proposition 4.11]) shows that tractability safely carries over to this more general setting. We can therefore focus on proving (in)tractability of approximation results in the remainder of this paper.

1WP on DAG. We start by showing the tractability of approximation for $\mathsf{PHom_L}(\mathsf{1WP},\mathsf{DAG})$, which also implies tractability of approximation for $\mathsf{PHom_L}(\mathsf{1WP},\mathsf{PT})$, since $\mathsf{PT} \subseteq \mathsf{DAG}$.

Proposition 3.1. $\mathsf{PHom_L}(\mathsf{1WP},\mathsf{DAG})$ *is #P-hard already in data complexity, but it admits an FPRAS. This holds even if the input instance is a probabilistic arity-two database.*

For #P-hardness, the result already holds in the unlabelled setting, so it will be shown in Section 4 (see Proposition 4.1). Hence, we focus on the upper bound. We rely on the notion of a *topological ordering* of the edges of a directed acyclic graph $H = (V,E)$: it is

simply a strict total order $(E, \prec)$ with the property that for every consecutive pair of edges $e_1 = (a_1, a_2)$ and $e_2 = (a_2, a_3)$, we have that $e_1 \prec e_2$. Let us fix such an ordering.

Proof of Proposition 3.1. We will show that every 1WP query on a DAG instance (possibly an arity-two database, i.e., with multiple edges having the same endpoints) admits an nOBDD representation of its provenance, which we can compute in combined polynomial time. We can then apply Theorem 2.3, from which the result follows. Let $G = a_1 \xrightarrow{R_1} \dots \xrightarrow{R_m} a_{m+1}$ be the input 1WP query, and $H = (V, E)$ the instance arity-two database. We make the following claim:

Claim 3.2. For every $v \in V$, we can compute in time $O(||G|| \times ||H||)$ an nOBDD representing $\mathsf{Prov}_H^G[a_1 := v]$ which is ordered by the topological ordering $\prec$ fixed above.

Proof. We build an nBDD D consisting of the two sinks and of the following nodes:

- $|V| \times ||G||$ $\vee$-nodes written $n_{u,i}$ for $u \in V$ and $1 \leq i \leq m$; and
- $|E| \times ||G||$ decision nodes written $d_{e,i}$ for $e \in E$ and $1 \leq i \leq m$ which test the edge e.

Each $\vee$-node $n_{u,i}$ for $u \in V$ and $1 \leq i \leq m$ has outgoing edges to each $d_{e,i}$ for every edge e emanating from u which is labelled R_i. For each decision node $d_{e,i}$, letting w be the target of edge e, then $d_{e,i}$ has an outgoing 0-edge to the 0-sink and an outgoing 1-edge to either $n_{w,i+1}$ if $i < m$ or to the 1-sink if $i = m$. The root of the nBDD is the node $n_{v,1}$.

This construction clearly respects the time bound. To check correctness of the resulting nBDD, it is immediate to observe that, for any path from the root to a sink, the sequence of decision nodes traversed is of the form $d_{e_1,1}, \dots, d_{e_k,k}$ where the $e_1, \dots, e_k$ form a path of consecutive edges starting at v and successively labelled $R_1, \dots, R_k$. This implies that the nBDD is in fact an nOBDD ordered by $\prec$. Further, such a path reaches the 1-sink iff $k = m$ and all decisions are positive, which implies that whenever the nOBDD accepts a subgraph H' of H then indeed there is a match of G in H' that maps a_1 to v. For the converse direction, we observe that, for any subgraph H' of H such that there is a match of G in H' mapping a_1 to v, then, letting $e_1, \dots, e_m$ be the images of the successive edges of G in H', there is a path from the root of D to the 1-sink which tests these edges in order. This establishes correctness and concludes the proof of the claim. □

Now observe that $\mathsf{Prov}_H^G = \mathsf{Prov}_H^G[a_1 := v_1] \vee \dots \vee \mathsf{Prov}_H^G[a_1 := v_n]$, where $v_1, \dots, v_n$ are precisely the vertices of H. Thus, it suffices to simply take the disjunction of each nOBDD obtained using the process above across every vertex in H, which yields in linear time the desired nOBDD. From here we can apply Theorem 2.3, concluding the proof. □

1WP on Arbitrary Graphs. We show, however, that tractability of approximation does *not* continue to hold when relaxing the instance class from DAG to arbitrary graphs. This also implies that more expressive classes of query graphs—such as 2WP, DWT, and PT also cannot be tractable to approximate on instances in the class All.

Proposition 3.3. $\mathsf{PHom_L}(\mathsf{1WP}, \mathsf{All})$ *does not admit an FPRAS unless* $\mathsf{RP} = \mathsf{NP}$. *This holds even for a fixed signature consisting of two labels.*

Proof. Our result hinges on the following claim:

Claim 3.4. Let $d > 1$ be a constant and let σ be a fixed signature with at least two labels. Given a monotone 2-CNF formula ϕ on n variables where each variable occurs in at most

d clauses, we can build in time $O(|\phi|)$ a 1WP G_ϕ and graph H_ϕ in the class All and over signature σ containing edges $(e_1, \ldots, e_n)$ such that $\mathsf{Prov}_{H_\phi}^{G_\phi}$ represents ϕ on $(e_1, \ldots, e_n)$.

Proof. Let $\phi = \bigwedge_{1 \leq i \leq m}(X_{f_1(i)} \vee X_{f_2(i)})$ be the input CNF instance over the variables $\{X_1, \ldots, X_n\}$, where $m > 0$ is the number of clauses. As we are in the labelled setting, let U and R be two distinct labels from the signature. Define the 1WP query graph G_ϕ to be $\xrightarrow{U}\xrightarrow{U}\left(\xrightarrow{R}{}^{d+2}\xrightarrow{U}\right)^m\xrightarrow{U}$. The instance graph H_ϕ in the class All is defined in the following way:

- For all $1 \leq i \leq n$, add an edge $a_i \xrightarrow{R} b_i$.
- Add an edge $c_0 \xrightarrow{U} d_0$ and for each clause $1 \leq j \leq m$, an edge $c_j \xrightarrow{U} d_j$.
- Add two edges $c'_0 \xrightarrow{U} c_0$ and $d_m \xrightarrow{U} d'_m$
- For each clause $1 \leq j \leq m$ and variable X_i occurring in that clause, let p be the number of this occurrence of X_i in the formula (i.e., the occurrence of X_i in the j-th clause is the p-th occurrence of X_i), with $1 \leq p \leq d$ by assumption on ϕ. Then add a path of length p of R-edges from d_{j-1} to a_i and a path of length $(d+1) - p$ of R-edges from b_i to c_j.

The construction of G_ϕ and H_ϕ is in $O(|\phi|)$. Furthermore, notice the following ($\star$). For any $1 \leq i \leq n$, the edge $e = a_i \xrightarrow{R} b_i$ has at most d incoming R-paths and d outgoing R-paths; the outgoing paths have pairwise distinct *length* (i.e., the number of edges until the next edge is a U-edge), and likewise for the incoming paths. What is more, each incoming R-path of length p corresponds to an outgoing path of length $(d+1) - p$ and together they connect some d_{j-1} to some c_j via the edge e, where the j-th clause contains variable X_i.

Moreover, notice the following ($\star\star$): the only two pairs of contiguous U-edges are $c'_0 \xrightarrow{U} c_0 \xrightarrow{U} d_0$ and $c_m \xrightarrow{U} d_m \xrightarrow{U} d'_m$.

Now, define $(e_1, \ldots, e_n)$ to be precisely the edges of the form $a_i \xrightarrow{R} b_i$ for every $1 \leq i \leq n$. Intuitively, the presence or absence of each of these edges corresponds to the valuation of each variable in ϕ. We claim that $\mathsf{Prov}_{H_\phi}^{G_\phi}$ represents ϕ on $(e_1, \ldots, e_n)$. Call a subgraph of H_ϕ a *possible world* if it contains all the edges not in $(e_1, \ldots, e_n)$ (as these are fixed to 1). It will suffice to show that there is a bijection between the satisfying valuations of ϕ and the possible worlds of H_ϕ that admit a homomorphism from G_ϕ.

Indeed, consider the bijection defined in the obvious way: keep the edge $a_i \xrightarrow{R} b_i$ iff X_i is assigned to true in the valuation. First suppose that some valuation of $\{X_1, \ldots, X_n\}$ satisfies ϕ. Then, for each clause $1 \leq j \leq m$, there is a variable in the clause which evaluates to true. We build a match of G_ϕ on the corresponding possible world of H_ϕ as follows:

- map the leftmost U-edge to $c'_0 \xrightarrow{U} c_0$,
- map the rightmost U-edge to $d_m \xrightarrow{U} d'_m$,
- for the other U-edges, map the j-th of these U-edges to $c_j \xrightarrow{U} d_j$ for all $0 \leq j \leq m$,
- map the R-paths for each $1 \leq j \leq m$ by picking a variable X_i witnessing that the clause is satisfied and going via the path of length $1 + (p) + ((d+1) - p) = d + 2$ that uses the edge $a_i \xrightarrow{R} b_i$, which is present by assumption.

Conversely, assume that we have a match of G_ϕ on a possible world of H_ϕ. We show that the corresponding valuation satisfies ϕ. It is an easy consequence of ($\star\star$) that the first U-edge must be mapped to $c'_0 \xrightarrow{U} c_0$, so the second U-edge must be mapped to $c_0 \xrightarrow{U} d_0$.

Let us show by finite induction on $0 \leq j \leq m$ that the $(j+2)$-th U-edge must be mapped to $c_j \xrightarrow{U} d_j$ and that if $j \geq 1$ the j-th clause is satisfied. The base case of $j = 0$ is clear because the second U-edge is mapped to $c_0 \xrightarrow{U} d_0$.

Let us take $j \geq 1$ and show the induction step. By induction hypothesis, the $(j+1)$-th U-edge is mapped to $c_{j-1} \xrightarrow{U} d_{j-1}$. Now, the R-path that follows must be mapped to a path from d_{j-1} to some a_i, and then take the edge $a_i \xrightarrow{R} b_i$, whose presence witnesses that the corresponding variable X_i is true. But importantly, in order for the path to have length precisely $d+2$ before reaching another U-edge, it must be the case that the length of the path before and after the edge $a_i \xrightarrow{R} b_i$ sums up to $d+1$. As a result of $(\star)$, this is only possible by taking a path that leads to $c_j \xrightarrow{U} d_j$. Thus we know that variable X_i occurs in the j-th clause so that this clause is satisfied, and we know that the $(j+2)$-th U-edge is mapped to $c_j \xrightarrow{U} d_j$. This establishes the induction step. Thus, the inductive proof establishes that all clauses are satisfied. This establishes the converse direction of the correctness claim, and concludes the proof. □

We can now conclude the proof of Proposition 3.3. By [Sly10, Theorem 2], counting the independent sets of a graph of maximal degree 6 admits an FPRAS only if $\mathsf{RP} = \mathsf{NP}$. It is not hard to see that this problem is equivalent to counting satisfying assignments of a monotone 2-CNF formula in which a variable can appear in up to 6 clauses (see, for example, [LL15, Proposition 1.1]). Thus, given a monotone 2-CNF formula ϕ obeying this restriction, we can apply Claim 3.4 above for the class of formulas in which $d = 6$ to obtain (deterministic) graphs G_ϕ and H_ϕ on a fixed signature σ with two labels, and then build a probabilistic graph H'_ϕ identical to H_ϕ, in which the edges $(e_1, \ldots, e_n)$ are assigned probability 0.5 and all other edges probability 1. Now, any FPRAS for the probabilistic graph homomorphism problem on G_ϕ and H'_ϕ would give an approximation of $N/2^n$, where N is the number of satisfying assignments of ϕ and n is the number of variables of ϕ. Simply multiplying the result of the FPRAS by 2^n would give us an FPRAS to approximate N, and so would imply $\mathsf{RP} = \mathsf{NP}$, concluding the proof. □

DWT on DWT. Having classified the cases of one-way path queries (1WP) on all instance classes considered, we turn to more expressive queries. The next two query classes to consider are two-way path queries (2WP) and downward trees queries (DWT). For these query classes, exact computation on 2WP instances is tractable by [AMS17], so the first case to classify is that of DWT instances. Exact computation is intractable in this case by [AMS17], and we show here that, unfortunately, approximation is intractable as well, so that the border for exact tractability coincides with that for approximate tractability. We first focus on DWT queries:

Proposition 3.5. $\mathsf{PHom_L}(\mathsf{DWT}, \mathsf{DWT})$ *does not admit an FPRAS unless* $\mathsf{RP} = \mathsf{NP}$. *This holds even for a fixed signature consisting of two labels.*

Proof. Our result hinges on the following, whose proof adapts [Mon18, Proposition 2.4.3]:

Claim 3.6. Let σ be a fixed signature with at least two labels. Given a monotone 2-CNF formula ϕ on n variables, we can build in time $O(|\phi| \log |\phi|)$ DWT graphs G_ϕ and H_ϕ over ϕ, with the latter containing edges $(e_1, \ldots, e_n)$ such that $\mathsf{Prov}_{H_\phi}^{G_\phi}$ represents ϕ on $(e_1, \ldots, e_n)$.

Proof. Let $\phi = \bigwedge_{1 \leq i \leq m}(X_{f_1(i)} \vee X_{f_2(i)})$ be the input CNF instance over the variables $\{X_1, \ldots, X_n\}$. We let $L = \lceil \log_2 m \rceil$ be the number of bits needed to write clause numbers in binary. As we are in the labelled setting, let 0 and 1 be two distinct labels from the signature. Construct the query graph G_ϕ as having a root z and child nodes $c_1, \ldots, c_m$ corresponding to the clauses and having a child path of length L labelled by the clause number. Formally:

- For all $1 \leq i \leq m$, add an edge $z \xrightarrow{0} c_i$.
- For each $1 \leq i \leq m$, letting $b_1 \cdots b_L$ be the clause number i written in binary, add a path of L edges $c_i \xrightarrow{b_1} d_{i,1} \xrightarrow{b_2} \ldots \xrightarrow{b_{L-1}} d_{i,L-1} \xrightarrow{b_L} d_{i,L}$.

Now, construct the DWT instance H_ϕ as having a root z' and child nodes $x_1, \ldots, x_n$ corresponding to the variables, each variable node x_i having child paths of length L labelled by the numbers of the clauses made true when setting x to true. Formally:

- For all $1 \leq i \leq n$, add the edges $z' \xrightarrow{0} x_i$.
- For all $1 \leq i \leq n$ and $1 \leq j \leq m$ such that X_i occurs in the j-th clause of ϕ (i.e., $i = f_1(j)$ or $i = f_2(j)$), letting $b_1 \cdots b_L$ be the clause number j written in binary, add a path of L edges $x_i \xrightarrow{b_1} y_{i,j,1} \xrightarrow{b_2} \ldots \xrightarrow{b_{L-1}} y_{i,j,L-1} \xrightarrow{b_L} y_{i,j,L}$.

It is clear that $G_\phi \in \mathsf{DWT}$, $H_\phi \in \mathsf{DWT}$, and that both graphs can be built in time $O(|\phi| \log |\phi|)$. Now, define $(e_1, \ldots, e_n)$ to be the edges of the form $z' \xrightarrow{0} x_i$ for every $1 \leq i \leq n$.

We claim that $\mathsf{Prov}_{H_\phi}^{G_\phi}$ represents ϕ on $(e_1, \ldots, e_n)$. Calling again a *possible world* a subgraph of H_ϕ that contains all the edges not in $(e_1, \ldots, e_n)$ (as these are fixed to 1), it suffices to show that there is a bijection between the satisfying valuations ν of ϕ and the possible worlds of H_ϕ that admit a homomorphism from G_ϕ. Indeed, consider the bijection defined in the obvious way: keep the edge $z' \xrightarrow{T} x_i$ iff X_i is assigned to true in the valuation. First, if there is a homomorphism from G_ϕ to a possible world H_ϕ, then the root z of the query must be mapped to a (since this is the only element with outgoing paths of length $L+1$ as prescribed by the query), and then it is clear that the image of any such homomorphism must take the form of a DWT instance that contains, for each clause number $1 \leq i \leq m$, a path of length L representing this clause number. This witnesses that the valuation ν makes a variable true which satisfies clause i. Hence, ν is a satisfying assignment of ϕ. Conversely, for every satisfying assignment ν, considering the corresponding possible world of H_ϕ, we can construct a homomorphism mapping the edges of G_ϕ to the edges of H_ϕ, by mapping the path of every clause to a path connected to a variable that witnesses that this clause is satisfied by ν. □

The result then follows by an argument analogous to the one in Proposition 3.3. □

2WP on DWT. We then move to $\mathsf{2WP}$ queries:

Proposition 3.7. $\mathsf{PHom_L}(\mathsf{2WP}, \mathsf{DWT})$ *does not admit an FPRAS unless* $\mathsf{RP} = \mathsf{NP}$. *This holds even for a fixed signature consisting of two labels.*

This result follows from a general reduction technique from DWT queries on DWT instances to $\mathsf{2WP}$ queries on DWT instances, which allows us to conclude using the result already shown on DWT queries (Proposition 3.5). We note that this technique could also have been used to simplify the proofs of hardness of exact computation in [AMS17] and [ABMS17]. We claim:

Lemma 3.8. *For any* DWT *query G, we can compute in time $O(||G||)$ a* 2WP *query G' which is equivalent to G on* DWT *instances: for any* DWT *H, there is a homomorphism from G to H iff there is a homomorphism from G' to H.*

Proof. Let G be a DWT query. We build G' following a tree traversal of G. More precisely, we define the translation inductively as follows. If G is the trivial query with no edges, then we let the translation of G be the trivial query with no edges (seen as a query consisting of a single vertex and no edges). Otherwise, let x be the root of G, let $x \xrightarrow{R_1} y_1, \ldots, x \xrightarrow{R_n} y_n$ be the successive children, and call $G_1, \ldots, G_n$ the DWT subqueries of G respectively rooted at $y_1, \ldots, y_n$. Let $G'_1, \ldots, G'_n$ be the respective translations of $G_1, \ldots, G_n$ as 2WP queries obtained inductively. For each $1 \leq i \leq n$, let G''_i be $\xrightarrow{R_i} G'_i \xleftarrow{R_i}$, i.e., the 2WP obtained by extending the 2WP G'_i by adding an edge $\xrightarrow{R_i}$ to the left (connected to the left endpoint) and adding an edge $\xleftarrow{R_i}$ to the right (connected to the right endpoint). In particular, if G'_i is the trivial query with no edges then G''_i is $\xrightarrow{R_i}\xleftarrow{R_i}$. We then define the translation of G to be the 2WP obtained as the concatenation of the queries $G''_1, \ldots, G''_n$, merging the right endpoint of G''_i and the left endpoint of G''_{i+1} for each $1 \leq i < n$. This translation is in linear time, and the translated query has twice as many edges as the original query. Note that we can also inductively define a homomorphism from G' to G mapping the first and last elements of G' to the root of G: this is immediate in the base case, and in the inductive claim we obtain suitable homomorphisms from each G'_i to each G_i by induction and combine them in the expected way.

We claim that, on any DWT instance H, for any vertex v of H, there is a match of G mapping the root of G to v iff there is a match of G' mapping both the first variable and last variable of the path to v. One direction is clear: from the homomorphism presented earlier that maps G' to G, we know that any match of G in H implies that there is a match of G' in H mapping the first and last elements as prescribed. Let us show the converse, and let us actually show by structural induction on G a stronger claim: for any vertex v of H, if there is a match of G' mapping the first variable of G' to a vertex v, then the last variable is also mapped to v and there is a match of G mapping the root variable to v. If G is the trivial query, then this is immediate: a match of the trivial query G' mapping the first variable to v must also map the last variable to v (it is the same variable), and we conclude. Otherwise, let us write x the root of G and $x \xrightarrow{R_1} y_1, \ldots, x \xrightarrow{R_n} y_n$ be the children and $G_1, \ldots, G_n$ the subqueries as above. We know that the match of G' maps the first variable to a vertex v, and as H is a DWT instance it maps x to a child v_1 of v. Considering G_1 and its translation G'_1, we notice that we have a match of G'_1 where the first variable is mapped to v_1. Hence, by induction, the last variable is also mapped to v_1, and we have a match of G_1 where the root variable is mapped to v_1. Now, as H is a DWT instance, the next edge $\xleftarrow{R_1}$ must be mapped to the edge connecting v and v_1, so that the next variable in G' is mapped to v. Repeating this argument for the successive child edges and child queries in G, we conclude that the last variable of G' is mapped to v, and we obtain matches of $G_1, \ldots, G_n$ that can be combined to a match of G. □

Lemma 3.8, when taken together with Proposition 3.5, allows us to prove Proposition 3.7: it reduces in linear time (in combined complexity) the evaluation of a DWT query on a DWT probabilistic instance to the evaluation of an equivalent 2WP query on the same instance and with the same signature. This establishes that any approximation algorithm for 2WP

queries on DWT instances would give an approximation for DWT queries on DWT instances, which by Proposition 3.5 is conditionally impossible.

These results complete Table 1, concluding the classification of the complexity of PHom in the labelled setting: all cases that were intractable for exact computation are also hard to approximate, with the notable exception of 1WP queries on DAG instances.

4. Results in the Unlabelled Setting

We now turn to the *unlabelled* setting of probabilistic graph homomorphism, where the signature σ is restricted to contain only one label ($|\sigma| = 1$). Our results are summarized in Table 1b: we settle all cases except $\mathsf{PHom}_{\not\mathsf{L}}(\mathsf{1WP}, \mathsf{All})$ and $\mathsf{PHom}_{\not\mathsf{L}}(\mathsf{DWT}, \mathsf{All})$, for which we do not give an FPRAS or hardness of approximation result. Note that both problems are #P-hard for exact computation [AMS17]. Further, they are in fact equivalent, because DWT queries are equivalent to 1WP queries in the unlabelled setting (stated in [AMS17] and reproved as Proposition 4.2 below).

1WP on DAG. We start with 1WP queries, and state the following:

Proposition 4.1. $\mathsf{PHom}_{\not\mathsf{L}}(\mathsf{1WP}, \mathsf{DAG})$ *is* #P*-hard already in data complexity, but it admits an FPRAS.*

The positive result directly follows from the existence of an FPRAS in the labelled setting, which we have shown in the previous section (Proposition 3.1). By contrast, the #P-hardness does not immediately follow from previous work, as DAG instances were not studied in [AMS17]. We can nevertheless obtain it by inspecting the usual #P-hardness proof of PQE for the CQ $\exists x\, y\ R(x), S(x, y), T(y)$ on TID instances [SORK11]. We give a proof in Appendix B.

DWT on DAG. We can easily generalize the above result from 1WP queries to DWT queries, given that they are known to be equivalent in the unlabelled setting:

Proposition 4.2 [AMS17]**.** $\mathsf{PHom}_{\not\mathsf{L}}(\mathsf{DWT}, \mathsf{DAG})$ *is* #P*-hard already in data complexity, but admits an FPRAS.*

This is implicit in [AMS17, Proposition 5.5]: we give a self-contained proof in Appendix C.

2WP on PT. In contrast to 1WP queries, which are exactly tractable on PT instances and admit an FPRAS on DAG instances, 2WP queries have no FPRAS already on PT instances:

Proposition 4.3. $\mathsf{PHom}_{\not\mathsf{L}}(\mathsf{2WP}, \mathsf{PT})$ *does not admit an FPRAS unless* RP = NP*.*

Proof. It suffices to prove the claim below, which is the analogue to the unlabelled setting of Claim 3.6 after having transformed the query to 2WP via Lemma 3.8:

Claim 4.4. Given a monotone 2-CNF formula ϕ on n variables, we can build in time $O(|\phi| \log |\phi|)$ an unlabelled 2WP graph G_ϕ and unlabelled PT graph H_ϕ, with the latter containing edges $(e_1, \ldots, e_n)$ such that $\mathsf{Prov}_{H_\phi}^{G_\phi}$ represents ϕ on $(e_1, \ldots, e_n)$.

We show this claim via a general-purpose reduction from the labelled setting to the unlabelled setting, which works in fact when queries and instances are in the class All. This reduction codes labels via specific unlabelled paths; a similar but ad-hoc technique was used to prove [AMS17, Proposition 5.6]:

Lemma 4.5. *For any constant $k \geq 2$, given a query G in the class* All *and instance graph H in the class* All *on a labelled signature with labels $\{1, \ldots, k\}$, we can construct in linear time an unlabelled query G' in the class* All *and instance graph H' in the class* All *such that there is a (labelled) homomorphism from G to H iff there is an (unlabelled) homomorphism from G' to H'. Further, if G is a* 2WP *then G' is also a* 2WP*, and if H is a* PT *then H' is also a* PT*.*

Proof. We construct G' from G and H' from H by replacing every edge by a fixed path that depends on the label of the edge. Specifically, we consider every edge $x \xrightarrow{i} y$ of the query, where x is the source, y is the target, and $1 \leq i \leq k$ is the label. We code such an edge in G' by a path defined as follows: $x \rightarrow^{k+3} \leftarrow \rightarrow^{i+1} \leftarrow^{k+2} y$, where exponents denote repeated edges and where intermediate vertices are omitted. We code the instance H to H' in the same way. This process is clearly linear-time, and it is clear that if G is a 2WP then G' is also a 2WP, and that if H is a PT then H' is also a PT. Further, to establish correctness of the reduction, one direction of the equivalence is trivial: a homomorphism h from G to H clearly defines a homomorphism from G' to H' by mapping the coding in G' of every edge e of G to the coding of the image of e by h in H'

What is interesting is the converse direction of the equivalence. We show it via a claim on homomorphic images of the coding of individual edges: for any $1 \leq i \leq k$, letting e' be the coding of an edge $e = x \xrightarrow{i} y$, for any homomorphism h' from e' to H', there must exist an edge $f = a \xrightarrow{i} b$ in H such that h' maps x to a and y to b. This claim implies the converse direction of the equivalence: if there is a homomorphism h' from G' to H', then applying the claim to the restrictions of h' to the coding of each edge of G, we see that h' defines a function h that maps the vertices of G to vertices of H, and that h is a homomorphism. Hence, all that remains is to prove the claim, which we do in the rest of the proof.

Consider an edge $e = x \xrightarrow{i} y$ as in the claim statement, and let e' be its coding and h' the homomorphism mapping e' to H'. Observe that, in H', the only directed paths of length $k + 3$ are the first $k + 3$ edges of the coding of edges of H. (This hinges on the fact that the paths of length $k + 3$ defined in the coding of edges of H are never adjacent in H' to another edge that goes in the same direction, even across multiple edges, and no matter the directions of edges in H.) This means that, considering the directed path $\rightarrow^{k+3}$ at the beginning of e', there must be an edge $f = a \xrightarrow{j} b$ of H, with coding f' in H', such that the source x of e is mapped to the source a of f, and the first $k + 3$ edges of e' are mapped to the first $k + 3$ edges of f'. What remains to be shown is that $i = j$ and that y is mapped to b.

To this end, we continue studying what can be the image of e' into f'. After the directed path $\rightarrow^{k+3}$, the next edge $\leftarrow$ of e' must have been mapped forward to the next edge $\leftarrow$ of f': indeed, it cannot be mapped backwards on the last edge of the preceding path $\rightarrow^{k+3}$ because $k + 3 > 1$ and $i + 1 > 1$ so the next edges $\rightarrow^{i+1}$ would then have no image. Then the next directed path $\rightarrow^{i+1}$ of e' is mapped in f', necessarily forward because we fail if we map the first edge backwards: this implies that there at least as many edges going in that direction in f' as there are in e', i.e., $i \leq j$. Now, the last path $\leftarrow^{k+2}$ of e' cannot be mapped backwards because $k + 2 > i + 1$, so we must map it forwards in f': for this to be possible, we must have reached the end of the directed path $\rightarrow^{j+1}$ in f', so that we have $j = i$. We are now done reading e' and f', so we have indeed mapped y to b. This, along with $i = j$, establishes that the claim is true, and concludes the proof. □

We can thus prove Claim 4.4, starting from Claim 3.6 and translating the labelled DWT query first to a labelled 2WP query via Lemma 3.8, and then the labelled 2WP query and PT instance (with precisely two labels) to an *unlabelled* 2WP query and PT instance via Lemma 4.5. Using the same argument as in Proposition 3.3, we conclude the proof of Proposition 4.3. □

5. DNNF Lower Bounds

In this section, we investigate how to represent the provenance of the query-instance pairs that we consider. More specifically, we study whether there exist polynomially-sized representations in tractable circuit classes of Boolean provenance functions Prov_H^G, for $G \in \mathcal{G}$ and $H \in \mathcal{H}$ in the graph classes studied in this paper. Certainly, for every graph class $\mathcal{G}$ and $\mathcal{H}$, the (conditional) non-existence of an FPRAS for $\mathsf{PHom}(\mathcal{G}, \mathcal{H})$ implies that, conditionally, we cannot compute nOBDD representations of provenance in polynomial time combined complexity—as otherwise we could obtain an FPRAS via Theorem 2.3. In fact, beyond nOBDDs, it follows from [ACJR21b, Theorem 6.3] that, conditionally, we cannot tractably compute provenance representations even in the more general class of *structured DNNFs*. Indeed, as for nOBDDs, fixed edges in the reductions can be handled by conditioning [PD08, Proposition 4].

However, even in settings where there is conditionally no combined FPRAS, it could be the case that there are polynomial-*sized* tractable circuits that are difficult to compute, or that we can tractably compute circuits in a more general formalism such as *unstructured* DNNF circuits. The goal of this section is to give a negative answer to these two questions, for all of the non-approximable query-instance class pairs studied in Sections 3 and 4.

Specifically, we show lower bounds on the size of DNNF circuits for infinite families of graphs taken from these classes. Remember that DNNF is arguably the most general knowledge compilation circuit class that still enjoys some tractable properties [DM02]. Hence, these lower bounds imply that no tractable provenance representation exists in other tractable subclasses of DNNFs, e.g., structured DNNFs [PD08], or Decision-DNNFs [BLRS17]. We also emphasize that, unlike the inapproximability results of Sections 3 and 4 which assumed RP $\neq$ NP, all of the DNNF lower bounds given here are unconditional.

We first show a *strongly* exponential lower bound for labelled 1WP queries on instances in the class All:

Proposition 5.1. *Let σ be any fixed signature containing at least two labels. There is an infinite family $G_1, G_2, \ldots$ of labelled* 1WP *queries and an infinite family $H_1, H_2, \ldots$ of labelled instances in the class* All *and on signature σ such that, for any $i > 0$, any DNNF circuit representing the Boolean function $\mathsf{Prov}_{H_i}^{G_i}$ has size $2^{\Omega(||G_i||+||H_i||)}$.*

Proof. By *treewidth* of a monotone 2-CNF formula, we mean the treewidth of the graph on the variables whose edges correspond to clauses in the expected way; and by *degree* we mean the maximal number of clauses in which any variable occurs. Let us consider an infinite family $\phi_1, \phi_2, \ldots$ of monotone 2-CNF formulas of constant degree $d = 3$ whose treewidth is linear in their size: this exists by [GM09, Proposition 1, Theorem 5]. We accordingly know by [ACMS20, Corollary 8.5] that any DNNF computing ϕ_i must have size $2^{\Omega(|\phi_i|)}$ for all $i > 1$. Using Claim 3.4, we obtain infinite families $G_1, G_2, \ldots$ of 1WP and $H_1, H_2, \ldots$ of graphs in the class All and on signature σ such that $\mathsf{Prov}_{H_i}^{G_i}$ represents ϕ_i on some choice of

edges, and we have $||G_i|| + ||H_i|| = O(|\phi_i|)$ for all $i > 0$ (from the running time bound). Now, any representation of $\mathsf{Prov}_{H_i}^{G_i}$ as a DNNF can be translated in linear time to a representation of ϕ_i as a DNNF of the same size, simply by renaming the edges $(e_1, \ldots, e_n)$ to the right variables, and replacing all other variables by the constant 1. This means that the lower bound on the size of DNNFs computing ϕ_i also applies to DNNFs representing $\mathsf{Prov}_{H_i}^{G_i}$, i.e., they must have size at least $2^{\Omega(|\phi_i|)}$, hence $2^{\Omega(||G_i||+||H_i||)}$ as we claimed. □

We now present lower bounds for the remaining non-approximable query-instance class pairs, which are not strongly exponential but rather *moderately* exponential. This is because our encoding of CNFs into these classes (specifically, Claim 3.6, and its images by Lemma 3.8 and Lemma 4.5) give a bound which is not linear but linearithmic (i.e., in $O(|\phi| \log |\phi|)$). We leave to future work the question of proving strongly exponential lower bounds for these classes, like we did in Proposition 5.1.

Proposition 5.2. *Let σ be any fixed signature containing at least two labels. For any $\epsilon > 0$, there is an infinite family $G_1, G_2, \ldots$ of labelled* DWT *queries and an infinite family $H_1, H_2, \ldots$ of labelled* DWT *instances on signature σ such that, for any $i > 0$, any DNNF circuit representing the Boolean function $\mathsf{Prov}_{H_i}^{G_i}$ has size at least $2^{\Omega\left((||G_i||+||H_i||)^{1-\epsilon}\right)}$.*

Proof. The proof is identical to that of Proposition 5.1, except that we apply Claim 3.6: for all $i > 0$, $||G_i|| + ||H_i|| = O(|\phi_i| \log |\phi_i|)$. We perform a change of variables: if we write $y = |\phi_i| \log |\phi_i|$, then we can show that $|\phi_i| = e^{W(y)}$, where W denotes the Lambert W function [CGH+96]; equivalently $|\phi_i| = y/W(y)$ as the W function satisfies $W(z)e^{W(z)} = z$ for all $z > 0$. Thus, the lower bound of $2^{\Omega(|\phi_i|)}$ on DNNF representations of ϕ_i implies that any DNNF for $\mathsf{Prov}_{H_j}^{G_j}$ has size at least $2^{\Omega\left(\frac{||G_i||+||H_i||}{W(||G_i||+||H_i||)}\right)}$. In particular, as W grows more slowly than n^ϵ for any $\epsilon > 0$, this gives a bound of $2^{\Omega\left((||G_i||+||H_i||)^{1-\epsilon}\right)}$ for sufficiently large ϕ_j. □

The proof for the following two claims are analogous to that of Proposition 5.2, but using Lemma 3.8 (for the first result) and Claim 4.4 (for the second result):

Proposition 5.3. *Let σ be any fixed signature containing at least two labels. For any $\epsilon > 0$, there is an infinite family $G_1, G_2, \ldots$ of labelled* 2WP *queries and an infinite family $H_1, H_2, \ldots$ of labelled* DWT *instances on signature σ such that, for any $i > 0$, any DNNF circuit representing the Boolean function $\mathsf{Prov}_{H_i}^{G_i}$ has size at least $2^{\Omega\left((||G_i||+||H_i||)^{1-\epsilon}\right)}$.*

Proposition 5.4. *Let σ be any fixed signature containing at least two labels. For any $\epsilon > 0$, there is an infinite family $G_1, G_2, \ldots$ of unlabelled* 2WP *queries and an infinite family $H_1, H_2, \ldots$ of unlabelled* PT *instances on signature σ such that, for any $i > 0$, any DNNF circuit representing the Boolean function $\mathsf{Prov}_{H_i}^{G_i}$ has size at least $2^{\Omega\left((||G_i||+||H_i||)^{1-\epsilon}\right)}$.*

We finish by remarking that all of the lower bounds above apply to acyclic query classes (i.e., queries of treewidth 1), for which non-probabilistic query evaluation is well-known to be linear in combined complexity [Yan81]. Thus, these results give an interesting example of query classes for which query evaluation is in linear-time combined complexity, but computing even a DNNF representation of query provenance is exponential (as we presented in Result 1.3).

6. Consequences

In this section, we consider some corollaries and extensions to the results above.

Optimality of a Previous Result. Recall from the introduction that, as was shown in [vBM23], PQE for self-join-free conjunctive queries of bounded hypertree width admits a combined FPRAS (in the general setting of probabilistic databases, rather than probabilistic graphs):

Theorem 1.4 (Theorem 1 of [vBM23])**.** *Let Q be a self-join-free conjunctive query of bounded hypertree width, and H a tuple-independent database instance. Then there exists a combined FPRAS for computing the probability of Q on H, i.e., an FPRAS whose runtime is $\mathsf{poly}(|Q|, ||H||, \epsilon^{-1})$, where ϵ is the multiplicative error.*

Can a stronger result be achieved? Our Proposition 4.3 immediately implies the following:

Corollary 6.1. *Assuming $\mathsf{RP} \neq \mathsf{NP}$, even on a fixed signature consisting of a single binary relation there is no FPRAS to approximate the probability of an input treewidth-1 CQ on an input treewidth-1 TID instance.*

Hence, tractability no longer holds with self-joins. So, as unbounded hypertree width queries are intractable in combined complexity even for *deterministic* query evaluation, we have:

Corollary 6.2. *Assuming $\mathsf{RP} \neq \mathsf{NP}$, the result in Theorem 1.4 is optimal in the following sense: relaxing either the self-join-free or bounded-hypertree-width condition on the query implies the non-existence of a combined FPRAS.*

Network Reliability. The *two-terminal network reliability problem*, dubbed *ST-CON* for brevity, intuitively asks the following: given a directed graph with probabilistic edges and with source and target vertices s and t, compute the probability that s and t remain connected, assuming independence across edges. Formally, working in the unlabelled setting of a signature σ with $|\sigma| = 1$, we are given a probabilistic graph (H, π) on signature σ together with two vertices s and t. We must compute the probability $\sum_{H' \subseteq H \text{ s.t. there is an } st-\text{path in } H'} \Pr_\pi(H')$, where by *st-path* we mean a directed path from s to t. Valiant showed that ST-CON is #P-complete [Val79, Theorem 1], and Provan and Ball showed that this holds already on directed acyclic graphs [PB83, Table 2]. Hardness also holds for the related problem of *all-terminal reliability* [PB83, Table 1], which asks for the probability that the probabilistic graph remains connected as a whole. Given the inherent #P-hardness of these problems, subsequent research has focused on developing tractable approximations.

Although significant progress has been made on FPRASes for the related problems of *all-terminal (un)reliability* [GJ19, Kar01], designing an FPRAS for ST-CON has remained open. This question was even open for the restricted case of directed acyclic graphs; indeed, it was explicitly posed as an open problem by Zenklusen and Laumanns [ZL11]. We now point out that the nOBDD construction of Proposition 3.1 implies an FPRAS for ST-CON on DAGs, again by leveraging the approximate counting result of Arenas et al. [ACJR21a]:

Theorem 6.3. *There exists an FPRAS for the ST-CON problem over directed acyclic graphs.*

Proof. Given as input an unlabelled probabilistic DAG instance $H = (V, E)$ and two distinguished source and target vertices s and $t \in V$, construct as follows the labelled DAG

instance $H' = (V, E, \lambda)$, whose set of labels is $\{R, R_s, R_t, R'\}$. All vertices and edges are identical to that of H, but every edge of the form (s, x) emanating from s is assigned label $\lambda((s, x)) = R_s$, every edge (x, t) directed towards t is assigned label $\lambda((x, t)) = R_t$, and every other edge (x, y) is assigned the label $\lambda((x, y)) = R$. In the case that $(s, t) \in E$, then assign $\lambda((s, t)) = R'$.

Now, by the result in Proposition 3.1, we can construct an nOBDD for each of the following $|E|$ different labelled 1WP queries: $\xrightarrow{R'}$, $\xrightarrow{R_s}\xrightarrow{R_t}$, $\xrightarrow{R_s}\xrightarrow{R}\xrightarrow{R_t}$, $\ldots$, $\xrightarrow{R_s}\left(\xrightarrow{R}\right)^{|E|-2}\xrightarrow{R_t}$. All of the nOBDDs have the same ordering (given by some topological ordering of the edges of H'), so we may take their disjunction to obtain a (complete) nOBDD D in linear time, whose accepting paths are in bijection with the (s, t)-connected valuations of the edges in H. From here we conclude by applying Theorem 2.3. □

We remark that an improved running time bound for Theorem 6.3 was obtained independently by Feng and Guo [FG24]. It may also be possible to improve the running time bounds obtained for our approach by leveraging recent faster algorithms for nOBDD approximate counting such as [MCM24], which give improved bounds for Theorem 2.2.

7. Regular Path Queries

In this section, we investigate the approximability of *regular path queries* (RPQs) on probabilistic graphs, measured in *data complexity*. Note that this differs from the rest of the paper in two respects. First, RPQs are distinct in expressiveness as a query class from conjunctive queries (CQs) studied earlier: they are generally not even expressible as *unions* of conjunctive queries (UCQs). Second, we focus on data complexity (i.e., treating the query as fixed, and the probabilistic graph as the sole input) in this section. This is in contrast to the combined complexity setting considered elsewhere in the paper; data complexity is relevant for RPQs because computing their probability on a probabilistic graph is not necessarily approximable, unlike CQs and UCQs which are always approximable in data complexity. It turns out that many of the techniques and results from earlier in the paper have direct applications for approximability in data complexity of RPQs, motivating the shift of focus here.

Preliminaries. We briefly review some background on languages, automata, and RPQs relevant to this section. An *alphabet* Σ is some finite set of symbols, which we typically denote with lowercase letters, i.e., $a, b, c, \ldots$ A *word* over Σ is a finite sequence of symbols from Σ. We denote by Σ^* the (infinite) set of all possible words over Σ, and call a subset $L \subseteq \Sigma^*$ of words a *language* over Σ.

We define a *deterministic finite automaton* (DFA) over an alphabet Σ as a tuple $A = (Q, \Sigma, q_0, F, \delta)$ where Q is a finite set of *states*, Σ is the alphabet, $q_0 \in Q$ is the *initial state*, $F \subseteq Q$ are the *final states*, and $\delta \colon Q \times \Sigma \to Q$ is the transition function. Let $w = a_1 a_2 \ldots a_n$ be a word over the alphabet Σ. We say that A *accepts* w if there exists some sequence of states $r_0, r_1, \ldots, r_n$ in Q such that: (1.) $r_0 = q_0$, (2.) $r_{i+1} = \delta(r_i, a_{i+1})$ for all $0 \leq i \leq n-1$, and (3.) $r_n \in F$. The set of strings accepted by A forms a language over Σ, which we call the language *recognized* by A. We call a language *regular* if it is accepted by some DFA. We often also use a regular expression syntax as shorthand for specifying regular languages: for instance, ab^*c^+ denotes the language of the words that start with a, continue with zero or more b's, and then finish by one or more c's.

We define a *regular path query* (RPQ) Q on a regular language L on alphabet Σ, denoted $Q = \mathsf{RPQ}(L)$, as a Boolean query whose semantics are given as follows: a labelled graph H with signature $\sigma := \Sigma$ *satisfies* $\mathsf{RPQ}(L)$ iff there exists some $\mathsf{1WP}$ query $G = \xrightarrow{a_1} \cdots \xrightarrow{a_n}$ such that $a_1 \dots a_n \in L$ and $G \rightsquigarrow H$.

Example 7.1. For singleton languages $L_w = \{w\}$ containing precisely one word w, then $\mathsf{RPQ}(L_w)$ is equivalent to the $\mathsf{1WP}$ query formed from w. As another example, when we take $L_0 = ab^*c$, then $\mathsf{RPQ}(L_0)$ holds on the labelled graphs such that there is a walk (i.e., a path which is not necessarily simple) which consists of b-edges and goes from an a-edge to a c-edge.

Clearly, the representation of Q is determined by the representation of the underlying regular language L, and so for concreteness we may assume throughout this section that L is specified as a DFA. However, as we use the data complexity perspective, it does not matter much: all results continue to hold if, for example, L is specified as a regular expression.

We study the *probabilistic query evaluation* problem for RPQs in *data complexity*, i.e., for each RPQ $Q = \mathsf{RPQ}(L)$ on alphabet Σ, we study a problem PQE_Q defined as follows. The input to PQE_Q is a probabilistic graph (H, π) on the set of labels $\sigma := \Sigma$, and PQE_Q asks for the total probability of the subgraphs $H' \subseteq H$ of (H, π) that satisfy Q according to the definition above. Note that fixing the query Q fixes in particular the signature σ over which input instances are expressed. In the statement of our upper bounds, we will also consider the variant of PQE_Q where the input is a probabilistic arity-two database instead of a probabilistic graph, and mention it explicitly when stating the upper bounds.

Some of our results on RPQ will use the notion of *Boolean provenance* for RPQs, which is defined in the same way as for the queries that we studied earlier in the paper. Recalling the notion of a *valuation* of a graph H, fixing an RPQ Q, the *provenance* of Q on H is the Boolean function Prov_H^Q having as variables the edges E of H and mapping every valuation ν of E to 1 (true) or 0 (false) depending on whether H_ν satisfies Q or not. Like Definition 2.1, given a Boolean formula ϕ whose variables $\{e_1, \dots, e_n\} \subseteq E$ are edges of H, we say that Prov_H^Q *represents* ϕ on $(e_1, ..., e_n)$ if for every valuation $\nu : E \to \{0, 1\}$ that maps edges not in $\{e_1, ..., e_n\}$ to 1, we have $\nu \models \phi$ if and only if $\mathsf{Prov}_H^Q(\nu) = 1$.

Infix-Free Languages. When evaluating RPQs on graphs, we have no fixed "endpoints" constraining where a query match must begin and end. Accordingly, distinct languages may be equivalent, in the sense that they give rise to the same RPQ. For example, we have that $\mathsf{RPQ}(a) = \mathsf{RPQ}(aa^*)$, since any labelled graph that contains a match of aa^* also contains a match of a, and vice versa. To formalize this idea and relate these languages, we assume that every language in this section is *infix-free*, in a sense that we define below.

Let L be a regular language over some alphabet Σ. Define a partial order $\preccurlyeq_L$ over words of L as follows: for words $u, v \in L$, we have $u \preccurlyeq_L v$ iff there exists words $s, t \in \Sigma^*$ such that $v = sut$. We call u an *infix* of v, and a *strict infix* of v if additionally $u \neq v$. The *infix-free sublanguage* of L, denoted $\mathsf{IF}(L)$, is the (possibly infinite) set of minimal elements of $\preccurlyeq_L$. We further say that a language L is *infix-free* if $L = \mathsf{IF}(L)$. Note that the infix-free sublanguage of an infinite language may be finite; recalling the example above, $\mathsf{IF}(aa^*) = a$. The following proposition establishes that the infix-free sublanguage of any regular language is itself regular: it immediately follows from the observation that $\mathsf{IF}(L) = L \setminus ((\Sigma^+ L \Sigma^*) \cup (\Sigma^* L \Sigma^+)) = L \cap ((\Sigma^+ L \Sigma^*) \cup (\Sigma^* L \Sigma^+))^C$, given that regular languages are closed under union, intersection, concatenation, and complementation.

Proposition 7.2 [PR10, Proposition 6]**.** *Let L be a regular language. Then $\mathsf{IF}(L)$ is regular.*

Moreover, a regular language gives rise to the same RPQ as that of its infix-free sublanguage.

Proposition 7.3. *Let L be a regular language. Then $\mathsf{RPQ}(L) = \mathsf{RPQ}(\mathsf{IF}(L))$.*

Proof. Consider an input graph G with signature Σ. We show that $\mathsf{RPQ}(L)$ evaluates to true iff $\mathsf{RPQ}(\mathsf{IF}(L))$ evaluates to true. We first show the forward direction; assume that $\mathsf{RPQ}(L)$ evaluates to true on G, and let $w \in L$ be a word labelling some corresponding walk π in G. By the construction of $\mathsf{IF}(L)$, there exists $v \in \mathsf{IF}(L)$ such that there exist (possibly empty) words $s, t \in \Sigma^*$ so that $w = svt$. Thus, we must also have a match from the word $v \in \mathsf{IF}(L)$ on G, and so $\mathsf{RPQ}(\mathsf{IF}(L))$ evaluates to true: just consider the appropriate subgraph of π witnessing a match of w. For the converse, suppose that $\mathsf{RPQ}(\mathsf{IF}(L))$ evaluates to true on G, and let $w \in \mathsf{IF}(L)$ be a word labelling some corresponding walk in G. Then since $\mathsf{IF}(L) \subseteq L$, we have that $w \in L$ and so $\mathsf{RPQ}(L)$ evaluates to true as well. □

Let L be an infix-free regular language. We call $\mathsf{RPQ}(L)$ *bounded* if L is finite, and *unbounded* otherwise.

Bounded RPQs. We observe that PQE can be tractably approximated for *every* bounded RPQ, because bounded RPQs are just a restricted kind of union of conjunctive queries (UCQs) on binary signatures, for which PQE is known to admit an FPRAS [SORK11]. This uses the Karp-Luby FPRAS (essentially a special case of the result in Theorem 2.3):

Proposition 7.4. *Let Q be a bounded RPQ. Then PQE_Q admits an FPRAS. This holds even if the input instance is a probabilistic arity-two database.*

Proof. By [BFR19, Proposition 5, Theorem 11], Q is equivalent to a UCQ which can be computed in time independent of the instance graph size, i.e., in constant time in data complexity. We can then compute the provenance of the fixed query Q on the input graph, and obtain it as a disjunctive normal form (DNF) formula, in polynomial-time data complexity. We may then apply the Karp-Luby FPRAS to compute the probability of this UCQ [SORK11, Section 5.3.2]. □

We further note that, for some bounded RPQs, we can even solve PQE_Q exactly in PTIME: for instance for the language ab. By contrast, there are other bounded RPQs, for example the language abc, for which the problem is $\#\mathsf{P}$-hard. In fact, there is a dichotomy for PQE_Q over the bounded RPQs Q between PTIME and $\#\mathsf{P}$: this follows from the more general Dalvi-Suciu dichotomy on PQE for UCQs [DS12].

Unbounded RPQs. For unbounded RPQs, we know that exact PQE is always $\#\mathsf{P}$-hard, by the results of [AC22]. Indeed, recall that a query is said to be *closed under homomorphisms* if the following holds: if a graph H satisfies the query and $H \rightsquigarrow H_0$, then H_0 also satisfies the query. The following is easy to observe:

Proposition 7.5. *Every RPQ is closed under homomorphisms.*

Proof. Let $Q = \mathsf{RPQ}(L)$ be an RPQ. Suppose H satisfies Q. Then by definition, there exists some $\mathsf{1WP}$ instance $G = \xrightarrow{a_1} \cdots \xrightarrow{a_n}$ such that $a_1 \dots a_n \in L$ and $G \rightsquigarrow H$. Now suppose $H \rightsquigarrow H_0$. Since the composition of two homomorphisms is itself a homomorphism, we have that $G \rightsquigarrow H_0$, and so H_0 satisfies Q as required. □

Hence, PQE for unbounded RPQs is #P-hard, because PQE is #P-hard in data complexity for every unbounded homomorphism-closed query on an arity-two signature [AC22, Theorem 3.3].

In terms of approximation, we now show that the PQE problem for unbounded RPQs is always at least as hard as the two-terminal network reliability problem (ST-CON) in directed graphs:

Proposition 7.6. *Let Q be any fixed unbounded RPQ on alphabet Σ. There is a polynomial-time reduction from ST-CON to PQE for Q. Formally, given an unlabelled graph (H, π) with probabilistic edges and given vertices s and t of H, we can build in polynomial time a labelled graph (H', π') with probabilistic edges on signature Σ and vertices s' and t' of H' such that the answer to ST-CON on (H, π) is the same as the answer to PQE_Q on (H', π').*

Proof. Let $Q = \mathsf{RPQ}(L)$ be the unbounded RPQ. We may assume without loss of generality that L is infix-free (for if not, we can simply make it infix-free, and the answer to PQE_Q is unchanged). We build our reduction from ST-CON on the unlabelled probabilistic graph (H, π), with distinguished vertices s and t.

By the pumping lemma, there is a word $xyz \in L$ comprising the concatenation of three subwords x, y, and z such that $xy^n z \in L$ for all $n \geq 0$, and $y \neq \epsilon$. We now show that $x \neq \epsilon$ and $z \neq \epsilon$. Indeed, suppose for a contradiction that $x = \epsilon$: then we have $xy^0 z = z \in L$ and so $yz \notin L$ (since L is infix-free), a contradiction. Similarly, if $z = \epsilon$, we have $x \in L$ and so $xy \notin L$, again a contradiction.

Now, we construct a labelled probabilistic graph (H', π') from the unlabelled ST-CON instance graph (H, π) as follows. H' is identical in structure to H, except for the addition of a new vertex x_i, connected to s via an x-labelled path whose edges all have probability 1, and similarly we add a new path connecting t to a new vertex x_e via a z-labelled path whose edges all have probability 1. Furthermore, the original unlabelled edges of H' are each replaced by an y-labelled path, with the first edge of the path carrying the same probability as the original corresponding edge in H, and each remaining edge of the path (if any) carrying probability 1.

It remains to show correctness, i.e., that the answer to ST-CON on (H, π) is the same as the answer to PQE_Q on (H', π'). Indeed, consider the probability-preserving bijection between subgraphs of (H, π) and (H', π') defined in the natural way: keep an edge in H iff its corresponding edge in H' is kept (preserving all the additional edges of probability 1 that arose from the construction of H'). It is clear that for every deterministic subgraph $H_d \subseteq H$ such that t is reachable from s, its counterpart $H'_d \subseteq H'$ contains a path labelled $xy^n z$ for some $n \geq 0$, and thus H'_d satisfies Q as desired. We now show that for any $H_d \subseteq H$ such that t is *not* reachable from s, its counterpart $H'_d \subseteq H'$ fails to satisfy Q. It is easy to see that the label for any path in H' forms an infix of $xy^n z$ for some $n \geq 0$. Thus, the label for any path in H'_d is a strict infix of $xy^n z$, since if the path was labelled $xy^n z$, there would be a path from x_i to x_e in H'_d, and thus from s to t in H_d. Finally, as L is infix-free, and $xy^n z \in L$, no strict infix of $xy^n z$ can belong to L. This implies that no path in H'_d is labelled with a word of L, and so H'_d does not satisfy Q. □

This result implies that finding an FPRAS for PQE_Q for *even one* unbounded RPQ would imply the existence of an FPRAS for ST-CON, solving a long-standing open problem. It is also easy to see that, for some specific unbounded RPQs Q, the problem PQE_Q is in fact polynomial-time equivalent to ST-CON.

Example 7.7. Consider the RPQ Q for the (infix-free) language ab^*c. Then Q is unbounded, so by Proposition 7.6 we know that ST-CON reduces in polynomial-time to PQE_Q. However, the converse is also true: given a probabilistic graph (H, π) on $\Sigma = \{a, b, c\}$, we can build an unlabelled probabilistic graph (H', π') in the following way. Initialize H' as the graph of the b-edges of H. Now, add to H' a source s and target t. Then connect s with an edge in H' to the target of each a-edge e in H with probability $\pi(e)$, and likewise connect the source of each c-edge e in H with an edge to t in H' with probability $\pi(e)$. It is now easy to see that the answer to ST-CON on (H', π') is the same as that of PQE_Q on (H, π).

Analogously to this example, we can show that PQE_Q is equivalent to ST-CON for any unbounded RPQ Q whose infix-free language is a so-called *local language*[1]. The class of *local languages*, which includes for instance ab^*c from the previous example, is a class of regular languages admitting several equivalent characterizations. In particular, a language is local iff it is recognized by a so-called *local automaton*. Recall the definition of a deterministic finite automaton (DFA) over an alphabet Σ as a tuple $A = (Q, \Sigma, q_0, F, \delta)$. A DFA is called *local* if all transitions labelled with the same letter lead to the same state; formally for each letter $a \in \Sigma$, there exists a unique state $q_a \in Q$ such that $\delta(q, a) = q_a$ for each $q \in Q$.

We claim:

Proposition 7.8. *Let Q be any fixed unbounded RPQ on alphabet Σ whose infix-free language L is local. Then there is a polynomial-time reduction from PQE_Q to ST-CON. This holds even if the input instance is a probabilistic arity-two database.*

Proof. Let A be a local DFA for L.

We build an unlabelled probabilistic graph (H', π') from the input probabilistic arity-two database (H, π) with alphabet Σ, and show that the answer to PQE_Q on (H, π) is the same as the answer of ST-CON on (H', π'). Let V be the active domain of H; the active domain of H' will be a subset of $(V \times \Sigma) \sqcup (V \times Q)$. For each letter $a \in \Sigma$ and state r of A such that r has an outgoing a-transition, for each $u \in V$, we create an edge of probability 1 in H' from (u, r) to (u, a). Further, for each edge e of H labelled a, writing $e = (u, v)$, we create the edge $e' = ((u, a), (v, r'))$ in H' with probability $\pi(e') := \pi(e)$, for r' the unique target state of all a-transitions in the local DFA A. Last, add a source s in H' with edges of probability 1 to (u, r_0) for each $u \in V$ with r_0 being the initial state of A, and a sink t in H' with edges of probability 1 from (u, r) for each $u \in V$ and each final state r of A.

The construction is in polynomial time, and the result (H', π') is indeed a probabilistic graph: note that we do not create the same edge twice. Further, there is a clear probability-preserving bijection between the subgraphs of H and the subgraphs of H' that keep the additional edges of probability 1, which we call the *possible worlds* of (H', π'). We claim that Q is satisfied in a possible world of (H, π) iff there is an st-path in the corresponding possible world of (H', π').

In the forward direction: if Q is true in a possible world of (H, π), then this is witnessed by a sequence of edges $e_1, \ldots, e_n$ forming a word $a_1 \cdots a_n$ that has an accepting run $r_0, \ldots, r_n$ in A in the DFA A. The corresponding unlabelled edges in the corresponding possible world of H' must be present, which together with the edges of probability 1 gives us an st-path. More precisely, writing $e_i = (u_i, v_i)$ for each $1 \leq i \leq n$ with $v_i = u_{i+1}$ for each $1 \leq i < n$, the path is $s, (u_1, r_0), (u_1, a_1), \ldots, (u_n, r_{n-1}), (u_n, a_n), (v_n, r_n), t$. This uses the fact that r_0

[1]This connection was realized in an independent collaboration with Gatterbauer, Makhija, and Monet [AGMM25], in which similar techniques are used to solve a different problem.

is the initial state of A, that each r_{i-1} has an outgoing a_i-transition for $1 \leq i \leq n$, that each r_i is the target state of a_i-transitions for $1 \leq i \leq n$, and that r_n is a final state of A.

In the backward direction: if there is an st-path in a possible world of H', then by construction of H' the path must be of the form $s, (u_1, r_0), (u_1, a_1), \ldots, (u_n, r_{n-1}), (u_n, a_n), (v_n, r_n), t$ for some choices of $u_1, \ldots, u_n, v_n \in V$ and $a_1, \ldots, a_n \in \Sigma$ and $r_0, \ldots, r_n$ states of A, with $r_0, \ldots, r_n$ an accepting run of A on the word $a_1 \ldots a_n$. Further, we must have kept in the corresponding possible world of (H, π) the edges $e_1, \ldots, e_n$ with $e_i \coloneqq (u_i, v_i)$ for each $1 \leq i \leq n$ and $v_i \coloneqq u_{i+1}$ for each $1 \leq i < n$, each e_i being labeled a_i for $1 \leq i \leq n$. Thus, this path witnesses that Q is satisfied in that possible world of (H, π). □

So, all unbounded RPQs are at least as hard as ST-CON, and there are some (namely, those whose infix-free language is local) that are polynomially equivalent to ST-CON. We do not know if the class of local languages is maximal with respect to this property. However, we can construct examples of unbounded RPQs Q whose language is not local, but for which PQE admits an FPRAS iff ST-CON admits an FPRAS. For instance:

Proposition 7.9. *Let L be an infix-free local language and let L' be an infix-free finite language such that L and L' are on disjoint alphabets. Let Q be the query corresponding to the disjunction of L and L', i.e., $\mathsf{RPQ}(L \sqcup L')$. Then PQE_Q admits a FPRAS iff ST-CON admits an FPRAS.*

Proof. For the forward direction, by Proposition 7.6, there is a PTIME reduction from ST-CON to PQE_Q: we can ensure that a match of L' is present by adding for instance a disjoint path of edges with probability 1 forming a word of L', and this does not affect the presence of a match of L because L' and L are on disjoint alphabets.

For the backward direction, we know that $\mathrm{PQE}_{\mathsf{RPQ}(Q')}$ admits an FPRAS by Proposition 7.4, and we know that $\mathrm{PQE}_{\mathsf{RPQ}(Q)}$ reduces to ST-CON by Proposition 7.8 which admits an FPRAS by hypothesis. Notice that the probability of L inducing a match on some deterministic subgraph H_d of the input probabilistic graph H is independent of the probability of L' inducing a match, since the alphabets of L and L' are disjoint. Let E_L denote the former event, and $E_{L'}$ the latter. Now, we wish to compute an (ϵ, δ)-approximation of $\Pr(E_L \cup E_{L'})$ (this is precisely the answer to $\mathrm{PQE}_{\mathsf{RPQ}(L \sqcup L')}$ on H). By the inclusion-exclusion rule and independence of events, we have $\Pr(E_L \cup E_{L'}) = \Pr(E_L) + \Pr(E_{L'}) - \Pr(E_L)\Pr(E_{L'})$. We may approximately evaluate this expression by calling the two FPRASes for $\Pr(E_L)$ and $\Pr(E_{L'})$ respectively, both with error parameter $\tau = \frac{\epsilon}{4}$ and confidence $\eta = \frac{\delta}{2}$. It is routine to verify that this gives an (ϵ, δ)-approximation of $\Pr(E_L \cup E_{L'})$ as desired. □

What about more expressive unbounded RPQs? Using techniques similar to those in Claim 3.4 earlier in the paper, we can show that some unbounded RPQs *do not* admit a FPRAS, conditionally to $\mathsf{RP} \neq \mathsf{NP}$.

Proposition 7.10. *Let Q be the unbounded RPQ defined from the infix-free language $aa(b^8a)^*a$. Then PQE_Q does not admit an FPRAS in data complexity unless $\mathsf{RP} = \mathsf{NP}$.*

The proof of this result hinges on the following, which is an immediate variant of Claim 3.4:

Claim 7.11. Let Σ be the alphabet $\{a, b\}$. Let $d > 1$ be a constant. Let Q_d be the unbounded RPQ defined by the infix-free regular language $aa(b^{d+2}a)^*a$. Given a monotone 2-CNF formula ϕ on n variables where each variable occurs in at most d clauses, we can build

in time $O(|\phi|)$ a graph H_ϕ in the class All containing edges $(e_1, \ldots, e_n)$ such that $\mathsf{Prov}^{Q_d}_{H_\phi}$ represents ϕ on $(e_1, \ldots, e_n)$.

Proof. We adapt the proof of Claim 3.4. Fix the constant $d > 1$. Define the RPQ Q_d as above: notice the similarity with the $\mathsf{1WP}$ query of the proof of Claim 3.4, except that we replace the exponent m with a Kleene star, and except that the labels U and R from the signature in Claim 3.4 respectively correspond to the letters a and b in the alphabet.

Given the monotone 2-CNF formula ϕ, we define the instance graph H_ϕ in the class All exactly like in the proof of Claim 3.4.

We now observe the following, which allows us to conclude from the proof of Claim 3.4: for any subgraph H'_ϕ of H_ϕ, we have that H'_ϕ satisfies the RPQ Q_d iff it satisfies the $\mathsf{1WP}$ query G_ϕ defined in the proof of Claim 3.4. One direction is immediate: if H'_ϕ satisfies the $\mathsf{1WP}$ query G_ϕ, then it satisfies Q_d, because G_ϕ corresponds to the word $aa(b^{d+2}a)^m a$ of Σ^* which is a word of L, so the image of a homomorphism of G_ϕ into H'_ϕ witnesses that Q_d is satisfied.

For the converse direction, assume that H'_ϕ satisfies the RPQ Q_d, and consider a $\mathsf{1WP}$ query G'_ϕ defining a word of L such that G'_ϕ has a homomorphism into $H_{\phi'}$. We claim that this word must be $aa(b^{d+2}a)^m a$, namely, the word that corresponds to G_ϕ, which suffices to conclude. Indeed, let us consider a match of such an $\mathsf{1WP}$ G'_ϕ in H'_ϕ. The property $(\star\star)$ of the proof of Claim 3.4 implies that the aa prefix of the match must be mapped to $c'_0 \xrightarrow{a} c_0 \xrightarrow{a} d_0$ and the aa suffix of the match must be mapped to $c_m \xrightarrow{a} d_m \xrightarrow{a} d'_m$. Now, it follows from the property $(\star)$ of the proof of Claim 3.4 that, in each factor of the form $ab^{d+2}a$, the initial a must be matched to an edge $c_i \xrightarrow{a} d_i$ and the final a must be matched to an edge $c_j \xrightarrow{a} d_j$ such that $j = i + 1$. It immediately follows that G'_ϕ must contain precisely $m + 3$ edges labelled a, which concludes the proof. □

Thanks to [Sly10, Theorem 2], we can now show Proposition 7.10 from Claim 7.11 using $d = 6$, in exactly the same way that Proposition 3.3 follows from Claim 3.4.

It is also easy to show an unconditional strongly exponential DNNF provenance lower bound on the same query as in Proposition 7.10, by adapting Proposition 5.1:

Proposition 7.12. *Let Q be the RPQ from Proposition 7.10. There is an infinite family $H_1, H_2, \ldots$ of labelled instances in the class All such that, for any $i > 0$, any DNNF circuit representing the Boolean function $\mathsf{Prov}^Q_{H_i}$ has size $2^{\Omega(||H_i||)}$.*

Proof. Like in the proof of Proposition 5.1, we consider an infinite family $\phi_1, \phi_2, \ldots$ of monotone 2-CNF formulas where each variable occurs in at most 3 clauses, such that any DNNF computing ϕ_i must have size $2^{\Omega(|\phi_i|)}$ for all $i > 1$. Claim 7.11 then gives us an infinite family $H_1, H_2, \ldots$ of graphs in the class All such that $\mathsf{Prov}^Q_{H_i}$ represents ϕ_i on some choice of edges, and we have $||H_i|| = O(|\phi_i|)$ for all $i > 0$ (from the running time bound). Now, again, any representation of $\mathsf{Prov}^Q_{H_i}$ as a DNNF can be translated in linear time to a representation of ϕ_i as a DNNF of the same size, simply by renaming variables and replacing variables by constants. This means that the lower bound on the size of DNNFs computing ϕ_i implies our desired lower bound. □

The results of Proposition 7.10 and Proposition 7.12 are shown for a specific choice of RPQ, but they do not apply to arbitrary RPQs. Indeed, by Proposition 7.9, there are some RPQs for which the existence of a FPRAS is equivalent to the existence of a FPRAS

for ST-CON, which is open. Hence, a natural question left open by the present work is to characterize the RPQs for which we can show similar inapproximability results or DNNF provenance lower bound results. In particular, establishing DNNF provenance lower bounds for ST-CON is another interesting question left open here.

8. Conclusions and Future Work

We studied the existence and non-existence of *combined approximation algorithms* for the PQE problem, as well as the existence of polynomially-sized tractable circuit representations of provenance, under the lens of combined complexity. We additionally considered the approximability of regular path queries in data complexity.

We see several potential directions for future work. First, it would be interesting to see if the results in Proposition 3.1 and Theorem 6.3 can be extended beyond DAG instances: graph classes of bounded *DAG-width* [BDH$^+$12] could be a possible candidate here. We also leave open the problem of filling in the two remaining gaps in Table 1. Namely, we would like to obtain either an FPRAS or hardness of approximation result for the equivalent problems $\mathsf{PHom}_{\not\mathsf{L}}(\mathsf{1WP}, \mathsf{All})$ and $\mathsf{PHom}_{\not\mathsf{L}}(\mathsf{DWT}, \mathsf{All})$. It is also natural to ask whether our results can be lifted from graph signatures to arbitrary relational signatures, or whether they apply in the *unweighted* setting where all edges are required to have the same probability [AK22, Ama23, KS21]. Another question is whether we can classify the combined complexity of approximate PQE for *disconnected* queries, as was done in [AMS17] in the case of exact computation, for queries that feature disjunction such as UCQs (already in the exact case [AMS17]), or for more general query classes, e.g., with recursion [AC22].

Lastly, for the data complexity of PQE for regular path queries, it remains open whether our results on specific query classes can be generalized to a full dichotomy characterizing which RPQs are approximable and which are (conditionally) not approximable; or which RPQs admit tractable provenance representations as DNNFs (or subclasses thereof) and which do not. However, such a result would require in particular to know whether ST-CON is approximable, and whether it admits tractable DNNF provenance representations.

References

[ABMS17] Antoine Amarilli, Pierre Bourhis, Mikaël Monet, and Pierre Senellart. Combined tractability of query evaluation via tree automata and cycluits. In *ICDT*, 2017. doi:10.4230/LIPIcs.ICDT.2017.6.

[ABS15] Antoine Amarilli, Pierre Bourhis, and Pierre Senellart. Provenance circuits for trees and treelike instances. In *ICALP*, 2015. doi:10.1007/978-3-662-47666-6_5.

[AC22] Antoine Amarilli and İsmail İlkan Ceylan. The dichotomy of evaluating homomorphism-closed queries on probabilistic graphs. *LMCS*, 2022. doi:10.46298/lmcs-18(1:2)2022.

[ACJR21a] Marcelo Arenas, Luis Alberto Croquevielle, Rajesh Jayaram, and Cristian Riveros. #NFA admits an FPRAS: efficient enumeration, counting, and uniform generation for logspace classes. *J. ACM*, 68(6), 2021. Extended version available as arXiv preprint arXiv:1906.09226 [cs.DS]. doi:10.1145/3477045.

[ACJR21b] Marcelo Arenas, Luis Alberto Croquevielle, Rajesh Jayaram, and Cristian Riveros. When is approximate counting for conjunctive queries tractable? In *STOC*, 2021. Extended version available as arXiv preprint arXiv:2005.10029 [cs.DS]. doi:10.1145/3406325.3451014.

[ACMS20] Antoine Amarilli, Florent Capelli, Mikaël Monet, and Pierre Senellart. Connecting knowledge compilation classes and width parameters. *ToCS*, 2020. doi:10.1007/s00224-019-09930-2.

[AGMM25] Antoine Amarilli, Wolfgang Gatterbauer, Neha Makhija, and Mikaël Monet. Resilience for regular path queries: Towards a complexity classification. *PACMMOD*, 3(2), 2025. doi:10.1145/3725245.

[AK22] Antoine Amarilli and Benny Kimelfeld. Uniform reliability of self-join-free conjunctive queries. *LMCS*, 2022. doi:10.46298/lmcs-18(4:3)2022.

[Ama23] Antoine Amarilli. Uniform reliability for unbounded homomorphism-closed graph queries. In *ICDT*, 2023. doi:10.4230/LIPIcs.ICDT.2023.14.

[AMS17] Antoine Amarilli, Mikaël Monet, and Pierre Senellart. Conjunctive queries on probabilistic graphs: Combined complexity. In *PODS*, 2017. doi:10.1145/3034786.3056121.

[AvBM24] Antoine Amarilli, Timothy van Bremen, and Kuldeep S. Meel. Conjunctive queries on probabilistic graphs: The limits of approximability. In *ICDT*, volume 290, 2024. doi:10.4230/LIPICS.ICDT.2024.15.

[BDH+12] Dietmar Berwanger, Anuj Dawar, Paul Hunter, Stephan Kreutzer, and Jan Obdrzálek. The DAG-width of directed graphs. *J. Comb. Theory, Ser. B*, 102(4), 2012. doi:10.1016/j.jctb.2012.04.004.

[BFR19] Pablo Barceló, Diego Figueira, and Miguel Romero. Boundedness of conjunctive regular path queries. In *ICALP*, 2019. doi:10.4230/LIPIcs.ICALP.2019.104.

[BLRS17] Paul Beame, Jerry Li, Sudeepa Roy, and Dan Suciu. Exact model counting of query expressions: Limitations of propositional methods. *TODS*, 42(1), 2017. doi:10.1145/298463.

[CGH+96] Robert M. Corless, Gaston H. Gonnet, D. E. G. Hare, David J. Jeffrey, and Donald E. Knuth. On the Lambert *W* function. *Adv. Comput. Math.*, 5(1), 1996. doi:10.1007/BF02124750.

[Dar01] Adnan Darwiche. Decomposable negation normal form. *J. ACM*, 48(4), 2001. doi:10.1145/502090.502091.

[DM02] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *J. Artif. Intell. Res.*, 17, 2002. doi:10.1613/jair.989.

[DS04] Nilesh N. Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. In *VLDB*, 2004. doi:10.1016/B978-012088469-8.50076-0.

[DS12] Nilesh N. Dalvi and Dan Suciu. The dichotomy of probabilistic inference for unions of conjunctive queries. *J. ACM*, 59(6), 2012. doi:10.1145/2395116.2395119.

[FG24] Weiming Feng and Heng Guo. An FPRAS for two terminal reliability in directed acyclic graphs. In *ICALP*, volume 297, 2024. doi:10.4230/LIPIcs.ICALP.2024.62.

[Gas24] Octave Gaspard. Approximate evaluation of regular path queries on probabilistic graphs, 2024. Bachelor thesis.

[GJ19] Heng Guo and Mark Jerrum. A polynomial-time approximation algorithm for all-terminal network reliability. *SIAM J. Comput.*, 48(3), 2019. doi:10.1137/18M1201846.

[GM09] Martin Grohe and Dániel Marx. On tree width, bramble size, and expansion. *J. Comb. Theory, Ser. B*, 99(1), 2009. doi:10.1016/j.jctb.2008.06.004.

[IJ84] Tomasz Imielinski and Witold Lipski Jr. Incomplete information in relational databases. *J. ACM*, 31(4), 1984. doi:10.1145/1634.1886.

[JS13] Abhay Kumar Jha and Dan Suciu. Knowledge compilation meets database theory: Compiling queries to decision diagrams. *ToCS*, 52(3), 2013. doi:10.1007/s00224-012-9392-5.

[Kan94] Ravi Kannan. Markov chains and polynomial time algorithms. In *FOCS*. IEEE, 1994. doi:10.1109/SFCS.1994.365726.

[Kar01] David R. Karger. A randomized fully polynomial time approximation scheme for the all-terminal network reliability problem. *SIAM Rev.*, 43(3), 2001. doi:10.1137/S0036144501387141.

[KS21] Batya Kenig and Dan Suciu. A dichotomy for the generalized model counting problem for unions of conjunctive queries. In *PODS*, 2021. doi:10.1145/3452021.3458313.

[LL15] Jingcheng Liu and Pinyan Lu. FPTAS for counting monotone CNF. In *SODA*, 2015. doi:10.1137/1.9781611973730.101.

[MCM24] Kuldeep S Meel, Sourav Chakraborty, and Umang Mathur. A faster FPRAS for #NFA. *PACMMOD*, 2(2), 2024. doi:10.1145/3651613.

[Mon18] Mikaël Monet. *Combined complexity of probabilistic query evaluation. (Complexité combinée d'évaluation de requêtes sur des données probabilistes)*. PhD thesis, University of Paris-Saclay, France, 2018. https://pastel.archives-ouvertes.fr/tel-01980366.

[Mon20] Mikaël Monet. Solving a special case of the intensional vs extensional conjecture in probabilistic databases. In *PODS*, 2020. doi:10.1145/3375395.3387642.

[PB83] J. Scott Provan and Michael O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM J. Comput.*, 12(4), 1983. doi:10.1137/0212053.

[PD08] Knot Pipatsrisawat and Adnan Darwiche. New compilation languages based on structured decomposability. In *AAAI*, 2008.

[PR10] Elena V. Pribavkina and Emanuele Rodaro. State complexity of prefix, suffix, bifix and infix operators on regular languages. In *DLT*, 2010. doi:10.1007/978-3-642-14455-4_34.

[Sen19] Pierre Senellart. Provenance in databases: Principles and applications. In *Reasoning Web*, 2019. doi:10.1007/978-3-030-31423-1_3.

[Sly10] Allan Sly. Computational transition at the uniqueness threshold. In *FOCS*, 2010. doi:10.1109/FOCS.2010.34.

[SORK11] Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011. ISBN: 978-1608456802.

[Val79] Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3), 1979. doi:10.1137/0208032.

[Var82] Moshe Y. Vardi. The complexity of relational query languages (extended abstract). In *STOC*, 1982. doi:10.1145/800070.802186.

[vBM23] Timothy van Bremen and Kuldeep S. Meel. Probabilistic query evaluation: The combined FPRAS landscape. In *PODS*, 2023. doi:10.1145/3584372.3588677.

[Yan81] Mihalis Yannakakis. Algorithms for acyclic database schemes. In *VLDB*. IEEE Computer Society, 1981.

[ZL11] Rico Zenklusen and Marco Laumanns. High-confidence estimation of small s-t reliabilities in directed acyclic networks. *Networks*, 57(4), 2011. doi:10.1002/net.20412.

Appendix A. Proof of Theorem 2.3

Theorem 2.3. *Let D be an nOBDD on variables V, and let $w : V \to [0, 1]$ be a rational probability function defined on V. Then there exists an FPRAS for computing $\mathsf{WMC}(D, w)$, running in time polynomial in $||D||$ and w.*

Proof. We may assume without loss of generality that D contains no variable v such that $w(v) = 0$ or $w(v) = 1$, since any such variable can be dealt with in constant time by conditioning D accordingly. By [ACMS20, Lemma 3.16], we may assume that D is *complete*, that is, every path from the root to a sink of D tests every variable of V. We will use the fact that for any positive integer n and set of variables $S = \{x_1, \ldots, x_k\}$ such that $k \geq \lceil \log n \rceil$, we can construct in time $O(k)$ a complete OBDD $C_n(x_1, \ldots, x_k)$, implementing a "comparator" on the variables of S, that tests if the integer represented by the binary string $x_1 \ldots x_k$ is strictly less than n (hence, $C_n(x_1, \ldots, x_k)$ has precisely n satisfying assignments, for every sufficiently large value of k). In particular, when $k = 0$ then C_1 is the trivial OBDD comprising only a 1-sink.

As D is complete, there is a natural bijection between the models of the Boolean function captured by D, and the paths from the root to the 1-sink of D. Now, perform the following procedure for every variable label v_i with weight $w(v_i) = p_i/q_i$ appearing in D. Set $k = \lceil \log (d + 1) \rceil$, where $d = \max\{p_i, q_i - p_i\}$. Send the 1-edge emanating from every node $r \in D$ labelled with v_i to the OBDD $C_{p_i}(x_1, \ldots, x_k)$ (where $x_1, \ldots, x_k$ are fresh variables), redirecting edges to the 1-sink of $C_p(x_1, \ldots, x_k)$ to the original destination of the 1-edge from n. Do the same for 0-edge from r, but with the OBDD $C_{q_i - p_i}(x_1, \ldots, x_k)$. Observe that D remains a complete nOBDD. Moreover, it is not difficult to see that there are now exactly p_i paths from the root to the 1-sink of D that pass through the 1-edge emanating from r, and $q_i - p_i$ paths passing through the 0-edge.

After repeating this process for every variable in D, we may apply Theorem 2.2, before normalizing the result by the product of the weight denominators $\prod q_i$. □

Appendix B. Proof of Proposition 4.1

Proposition 4.1. $\mathsf{PHom}_{\not\mathsf{L}}(\mathsf{1WP}, \mathsf{DAG})$ *is* #P*-hard already in data complexity, but it admits an FPRAS.*

Proof. As mentioned in the main text, the positive result directly follows from the existence of an FPRAS in the labelled setting, which was shown in Proposition 3.1. It remains to show #P-hardness here. We will define a reduction from the #P-hard problem #PP2DNF from [SORK11, Theorem 3.1], which asks to count the satisfying valuations of a Boolean formula that is in 2-DNF (i.e., in disjunctive normal form with two variables per clause), that is positive (i.e., has no negative literals), and that is partitioned (i.e., variables are partitioned between two sets and each clause contains one variable from each set). Formally, #PP2DNF asks us to count the satisfying valuations of an input formula of the form $\phi = \bigvee_{(i,j)\in E} (X_i \wedge Y_j)$. We reduce to $\mathsf{PHom}_{\not\mathsf{L}}(G, \mathsf{DAG})$, where G is fixed to be the 1WP of length three, i.e., $\rightarrow\rightarrow\rightarrow$.

Construct a probabilistic graph $H = (\{s, t\} \sqcup X \sqcup Y, E_H)$ with probability labelling π, where s and t are fresh vertices, $X = \{X_1, \ldots, X_m\}$ and $Y = \{Y_1, \ldots, Y_n\}$ are vertices corresponding to the variables of ϕ, and the edge set E_H comprises:

- a directed edge $s \to X_i$ for every $X_i \in X$, with probability 0.5;
- a directed edge $X_i \to Y_j$ for every $(i, j) \in E$, with probability 1 (i.e., one directed edge per clause of ϕ);
- a directed edge $Y_j \to t$ for every $Y_j \in Y$, with probability 0.5.

It is clear that $H \in \mathsf{DAG}$. Moreover, we claim that $\Pr_\pi(G \rightsquigarrow H)$ is precisely the number of satisfying valuations of ϕ, divided by $2^{|X \sqcup Y|}$. Indeed, just consider the natural bijection between the subgraphs of H and the valuations of ϕ, where we keep the edge $s \to X_i$ iff X_i is assigned to true in a given valuation, and the edge $Y_j \to t$ iff Y_j is assigned to true. Then it is easy to check that a subgraph of H admits a homomorphism from G iff the corresponding valuation satisfies ϕ. □

Appendix C. Proof of Proposition 4.2

Proposition 4.2 [AMS17]**.** $\mathsf{PHom}_{\not\mathsf{L}}(\mathsf{DWT}, \mathsf{DAG})$ *is* #P*-hard already in data complexity, but admits an FPRAS.*

Proof. Hardness follows directly from Proposition 4.1, so we show the positive result here. Let G be a DWT query graph, and m its height, i.e., the length of the longest directed path it contains. Let G' be this 1WP of length m, computable in polynomial time from G. We claim the following.

Claim C.1. For any $H \in \mathsf{DAG}$, $\mathsf{PHom}_{\not\mathsf{L}}(G, H) = \mathsf{PHom}_{\not\mathsf{L}}(G', H)$.

Proof. Certainly, if $H' \subseteq H$ admits a homomorphism from G, then it admits one from G' too since $G' \subseteq G$. On the other hand, if H' admits a homomorphism from G', then it also admits one from G: just map all vertices of distance i from the root of G to the image of the i-th vertex of G'. □

Now the result follows from the FPRAS for $\mathsf{PHom}_{\not\subset}(G', \mathsf{DAG})$ given by Proposition 3.1. □