Lignages efficaces sur les instances quasi-arborescentes: Limites et extensions

Antoine Amarilli
Télécom ParisTech,
Université Paris-Saclay, France
antoine.amarilli@telecom-paristech.fr

Pierre Bourhis
CRIStAL, UMR 9189, CNRS,
Université Lille 1, France
pierre.bourhis@univ-lille1.fr

Pierre Senellart
Télécom ParisTech,
Université Paris-Saclay, France
& IPAL, CNRS, NUS, Singapore
pierre.senellart@telecom-paristech.fr

ABSTRACT

Il est généralement infaisable (#P-difficile) d'évaluer des requêtes sur les bases de données probabilistes. Des résultats de dichotomie ont permis d'identifier [20, 19, 24] quelles requêtes (dites *safe*) peuvent être évaluées efficacement, en rattachant cela à des représentations du lignage [35]. Nous avons précédemment montré [2], à l'aide de techniques différentes, que l'évaluation de requêtes arbitraires en logique monadique du second ordre est faisable en temps linéaire sur les bases de données probabilistes, à condition de borner la *largeur d'arbre* des instances.

Dans ce travail, nous étudions les limites et les extensions possibles de ce résultat. Nous montrons d'abord, pour l'évaluation probabiliste de requêtes, qu'il est *nécessaire* de borner la largeur d'arbre pour assurer la faisabilité de MSO : en effet, il y a même des requêtes FO dont l'évaluation probabiliste est infaisable sur n'importe quelle classe de graphes de largeur d'arbre non bornée qui soit efficacement constructible. Cette dichotomie s'appuie sur des bornes polynomiales récentes pour l'extraction de graphes planaires comme mineurs [11] ; elle implique des bornes inférieures pour des problèmes non-probabilistes analogues comme l'évaluation de requêtes et de comptage d'assignements sur des familles closes par sous-instances. Nous montrons ensuite comment notre résultat de faisabilité peut s'expliquer en termes de lignage : on peut représenter le lignage d'une requête MSO sur une instance quasi-arborescente comme un circuit quasi-arborescent, un OBDD de taille polynomiale, ou une d-DNNF de taille linéaire. En revanche, nous pouvons étendre notre premier résultat de nécessité aux représentations de lignage, et exhiber une UCQ avec inégalités telles que, pour n'importe quelle famille de graphes de largeur d'arbre non bornée, le lignage ne peut pas être représenté par un OBDD de taille polynomiale ; nous pouvons même caractériser les requêtes qui ont cette propriété. Nous montrons enfin comment notre approche sur les instances quasi-arborescentes permet d'expliquer la faisabilité de l'évaluation pour les requêtes safe *sans inversion* : leurs instances d'entrée peuvent être récrites pour borner leur largeur d'arbre.

Cet article est une version légèrement modifiée d'un article publié à la conférence PODS'2016 [4].

1. INTRODUCTION

Many applications must deal with data which may be erroneous. This makes it necessary to extend relational database instances, to allow for uncertain facts. One of the simplest such formalisms [48] is that of *tuple-independent databases* (TID): each tuple in the database is annotated with an independent probability of being present. The semantics of a TID instance is to see it as a concise representation of a probability distribution on standard non-probabilistic instances.

An important challenge when dealing with probabilistic data is that data management tasks become intractable. The main one is *query evaluation*: given an input database query q, for instance a conjunctive query, and given a relational instance I, determine the answers to q on I. When q is Boolean, we just ask whether I satisfies q. The corresponding problem in the TID setting asks for the *probability* that $I \models q$, that is, the total probability weight of the possible subsets of the TID instance I that satisfy q. The query q is usually assumed to be fixed, and we look at the complexity of this problem as a function of the input instance (or TID) I, that is, the *data complexity*. Sadly, while this task is highly tractable and parallelizable (in the complexity class AC^0) in the non-probabilistic context, exact computation is generally intractable (#P-hard) on TID instances, even for the simple conjunctive query $\exists xy\ R(x) \wedge S(x, y) \wedge T(y)$. See [18].

Faced with this intractability, two natural directions are possible. The first is to restrict the language of *queries* to focus on queries that are tractable on *all* instances, called *safe*. This has proven a very fruitful direction [20], culminating in the dichotomy result of Dalvi and Suciu [19]: the data complexity of a given union of conjunctive queries (UCQ) is either in PTIME or #P-hard. More recently, the safe non-repeated CQs *with negation* were characterized in [24].

The second approach is to restrict the *instances*, to focus on instance families that are tractable for *all* queries in highly expressive languages. In a recent work [2], going through the setting of semiring provenance [30], and using a celebrated result by Courcelle [14], we have started to explore this direction. We showed that, for queries in MSO (*monadic second-order*, a highly expressive language capturing UCQs), data complexity is linear-time on *treelike* instances, i.e., instances of *treewidth* bounded by a constant. Of course, this result says nothing of non-treelike instances, but covers use cases previously studied in their own right, such as probabilistic XML [12] (without data values).

This new direction raises several important questions:

- First, is this the best that one can hope for? For probability evaluation, could the tractability on bounded-treewidth instances be generalized, e.g., to bounded *clique-width* instances [15], as for MSO in the non-probabilistic case? More ambitiously, could we separate tractable and intractable instances with a dichotomy theorem?
- Second, can our bounded-treewidth tractability result

be explained in terms of *lineage*? The *lineage* of a query intuitively represents how it can be satisfied on the instance, and can be used to compute its probability: for many fragments of safe queries [35], tractability can be shown via a tractable representation of their lineage. In [2], we build a bounded-treewidth circuit representation of Boolean provenance. How does this compare to the usual lineage classes of OBDDs and d-DNNFs in knowledge compilation?

- Third, can we link the query-based tractability approach to our instance-based one? Can we explain the tractability of some safe queries by reducing them to query evaluation on treelike instances?

This paper answers all of these questions.

Contributions. Our *first main result* (in Section 4) is that bounded treewidth characterizes the tractable families of graphs for MSO queries in the probabilistic context. More precisely, we construct a query q_{h} for which probability evaluation is intractable on *any* unbounded-treewidth family of graphs satisfying mild constructibility requirements; query evaluation is precisely $\mathrm{FP}^{\#\mathrm{P}}$-complete under randomized polynomial-time (RP) reductions. Thus, tractability on bounded-treewidth instances is really the best we can get, on arity-2 signatures. Surprisingly, we show that q_{h} can be taken to be a (non-monotone) FO query; this is in stark contrast with non-probabilistic query evaluation [37, 27] where FO queries are fixed-parameter tractable under much milder conditions than bounded treewidth [36]. This provides the lower bound of a dichotomy, the upper bound being our result in [2].

In Section 5, we explain how this dichotomy result can be adapted to non-probabilistic MSO query evaluation and match counting on subgraph-closed graph families. While the necessity of bounded-treewidth for non-probabilistic query evaluation was studied before [37, 27], our use of a recent polynomial bound on grid minors [11] allows us to obtain stronger results in this context, which we review. Our work thus answers the conjecture of [31] (Conjecture 8.3) for MSO, which [37] answered for MSO_2, under similar complexity-theoretic assumptions.

In Section 6, we move from probability evaluation to the computation of tractable *lineages*. Our tractability result in [2] computes a bounded-treewidth lineage of linear size for MSO queries on bounded-treewidth instances. We revisit this upper bound and show that we can compute an OBDD lineage of polynomial size (by results in [34]) and a d-DNNF lineage of linear size (a new result). We show that on bounded-*pathwidth* instances (a notion more restrictive than that of bounded-treewidth), we obtain a bounded-pathwidth lineage, and hence a constant-width OBDD (by [34]). Further, all these representations can be efficiently constructed.

We then move in Section 7 to our *second main result*, which applies to tractable *OBDD lineages* rather than tractable query evaluation. It shows a dichotomy on arity-2 signatures, for the weaker query language of *UCQs with disequalities*: while bounded-treewidth instances admit efficient OBDDs for such queries, any constructible unbounded-treewidth instance family must have superpolynomial OBDDs for some query (depending only on the signature).

Last, in Section 8, we connect our approach to query-based tractability conditions [20, 19]. We show that, for safe UCQs that admit a concise OBDD representation (that is, precisely inversion-free UCQs from [35]), one can rewrite any instance to a bounded-treewidth instance (actually, to a bounded-pathwidth one), such that the query lineage, and hence the query probability, remain the same. Thus, in this sense, safe queries are tractable because their input instances may as well be bounded-pathwidth.

Related work. Bounded-treewidth has been shown to be a sufficient condition for tractability of query evaluation (this is by Courcelle's theorem [14], generalized to arbitrary relational structures in [25]), counting of query matches [5], and probabilistic query evaluation [2].

For MSO query evaluation on non-probabilistic instances, bounded-treewidth is known not to be necessary, e.g., query evaluation is tractable assuming bounded *clique-width* [15]. FO query evaluation is tractable assuming milder conditions [36]. Two separate lines of work investigated the necessity of bounding the treewidth of instances to ensure the tractability of other data management tasks.

First, in [42, 43], Marx shows that treewidth-based algorithms for binary constraint-satisfaction problems (CSP) are, assuming the exponential-time hypothesis, *almost optimal*: they can only be improved by a logarithmic factor. These works do not rely on the graph minor theorem [47] as we do, as they preceded the results of [11] that provide polynomial bounds on the size of grid minors: see the discussion in the Introduction of [43]. Instead, they characterize high treewidth via embeddings of low *depth*. The results of [42, 43] were further applied to inference in undirected [10] and directed [38] *graphical models*. All these works are specific to the setting and problem that they study, namely CSP and inference.

Second, another line of work [41, 37, 27] has shown necessity of bounded treewidth when a class of graphs is closed under some operations: extracting topological minors in [41], extracting subgraphs in [37], and extracting subgraphs and vertex relabeling in [27]. This requires that there are sufficiently many instances of high treewidth, through notions of *strong unboundedness* [37] and *dense unboundedness* [27]. We strengthen the results of [27] in Section 5.2 of this paper, using our techniques. None of these works consider probabilistic evaluation or match counting, which we do here.

Other related work is discussed throughout the paper, where relevant; in particular works related to lineages in Sections 6 and 7 and to safe queries in Section 8.

The next section (Section 2) presents preliminaries, and Section 3 gives our formal problem statement. We then move to our new results in Section 4 onwards.

For space reasons, we omit all proofs. For Sections 4, 5, and 7, the proofs can be found in Chapter 6 of [1]. Full proofs of the other sections can be found in the appendix.

2. PRELIMINARIES

Instances. A *relational signature* σ is a set of relations $R, S, T, \ldots$, each having an *arity* denoted $\mathrm{arity}(R) \in \mathbb{N}_{>0}$. The signature σ is *arity-k* if k is the maximum arity of a relation in σ.

A *relational instance* (or simply σ-instance or instance) I is a finite set of ground *facts* on the signature σ, and a *class* or *family* of instances $\mathcal{I}$ is just a (possibly infinite) set of instances. A *subinstance* of I is a subset of its facts. We follow the *active domain semantics*, where the *domain*

$\mathrm{dom}(I)$ of I is the finite set of elements that occur in facts. Hence, for $I' \subseteq I$, $\mathrm{dom}(I')$ is the (possibly strict) subset of $\mathrm{dom}(I)$ formed of the elements that occur in facts of I'. The *size* of I, denoted $|I|$, is its number of facts.

A *homomorphism* from a σ-instance I to a σ-instance I' is a function $h : \mathrm{dom}(I) \to \mathrm{dom}(I')$ such that, for all $R(a_1, \ldots, a_k) \in I$, we have $R(h(a_1), \ldots, h(a_k)) \in I'$. A homomorphism is an *isomorphism* if it is bijective and its inverse is also a homomorphism.

Graphs. Throughout the paper, a *graph* will always be undirected, simple, and unlabeled, unless otherwise specified. Formally, we see a graph G as an instance of the *graph signature* with a single predicate E of arity 2 such that: (i) $\forall x\, E(x,x) \notin G$; and (ii) $\forall xy\, E(x,y) \in G \Rightarrow E(y,x) \in G$. As we follow the active domain semantics, this implies that we disallow *isolated vertices* in graphs. The facts of G are called *edges*. The set of *vertices* (or *nodes*) of a graph G, denoted $V(G)$, is its domain. Two vertices x and y of a graph G are *adjacent* if $E(x,y) \in G$, x and y are then called the *endpoints* of the edge, and the edge is *incident* to them; two edges are *incident* if they share a vertex.

The *degree* of a vertex x is the number of its adjacent vertices. For $k \in \mathbb{N}$, a graph is *k-regular* if all vertices have degree k. More generally, it is *K-regular*, where K is a finite set of integers, if every vertex has degree k for some $k \in K$. Finally, a graph is *degree-k* if k is the maximum of the degree of all its vertices, i.e., if it is $\{1, \ldots, k\}$-regular. A graph is *planar* if it can be drawn on the plane without edge crossings, in the standard sense [23].

A *path* of length $n \in \mathbb{N}_{>0}$ in a graph G is a set of edges $\{E(x_0, x_1), E(x_1, x_2), \ldots, E(x_{n-1}, x_n)\}$ that are all in G; the path is *simple* if all x_i's are distinct. A *cycle* is a path of length $n \geqslant 3$ where all vertices are distinct except that $x_0 = x_n$; a graph is *cyclic* if it has a cycle. A graph is *connected* if there is a path from every vertex to every other vertex. A *subdivision* of a graph G is a graph obtained by replacing each edge by an arbitrary non-empty simple path (every node on this path being fresh except the endpoints of the original edge).

Treewidth and pathwidth. A *tree* T is an acyclic connected graph (remember that graphs are undirected). A *tree decomposition* of a graph G is a tree with a labeling function λ from its nodes (called *bags*) to sets of vertices of G, ensuring: (i) for every edge $E(u,v) \in G$, there is a bag $n \in V(T)$ such that $\lambda(n)$ contains both u and v; (ii) for every node u of G, the subtree of T formed of all bags whose λ-image contains u must be connected. The *width* of T is $\max_{n \in V(T)} |\lambda(n)| - 1$. The *treewidth* of a graph G, denoted $\mathsf{tw}(G)$, is the minimum width of any tree decomposition of G.

The *treewidth* of a relational instance I, denoted $\mathsf{tw}(I)$, is defined as usual as the treewidth of its *Gaifman graph*, namely, the graph on the *domain* $\mathrm{dom}(I)$ of I that connects any two elements that co-occur in a fact. When the signature is arity-2, we can see an instance I as a labeled graph, and the treewidth of I is then exactly the treewidth of this graph.

A *path decomposition* is a tree decomposition where T is also a path. The *pathwidth* of a graph G is the minimum width of any path decomposition of the graph. The *pathwidth* of a relational instance is the pathwidth of its Gaifman graph.

Queries. A *query* on the signature σ is a formula in second-order logic over predicates of σ, with its standard semantics. All queries that we consider have no constants; unless otherwise specified, they are *Boolean*, i.e., they have no free variable. We write $I \models q$ whenever an instance I satisfies the query q. We will be especially interested in the language FO of first-order logical sentences (where second-order quantifications are disallowed) and the language MSO of monadic second-order logical sentences (where the only second-order quantifications are over unary predicates).

We will also consider the language CQ of *conjunctive queries*, i.e., existentially quantified conjunctions of atoms over the signature; the language $\mathrm{CQ}^{\neq}$ of conjunctive queries where additional atoms of the form $x \neq y$ (called *disequality* atoms) are allowed, where x and y are variables appearing in some regular atom; the language UCQ of *union of conjunctive queries*, namely, disjunctions of CQs; the language $\mathrm{UCQ}^{\neq}$ of disjunctions of $\mathrm{CQ}^{\neq}$ queries. The size $|q|$ of a $\mathrm{UCQ}^{\neq}$ query q is its total number of atoms, i.e., the sum of the number of atoms in each $\mathrm{CQ}^{\neq}$.

A *homomorphism* from a CQ q to an instance I is a mapping h from the variables of q to $\mathrm{dom}(I)$ such that for each atom $R(x_1, \ldots, x_k)$ of q we have $R(h(x_1), \ldots, h(x_k)) \in I$. For $\mathrm{CQ}^{\neq}$ queries, we require that $h(x) \neq h(y)$ whenever q contains the disequality atom $x \neq y$. A *homomorphism* from a $\mathrm{UCQ}^{\neq}$ q to I is a homomorphism from some disjunct of q to I: it witnesses that $I \models q$. A *match* of a $\mathrm{UCQ}^{\neq}$ q on an instance I is a subset of I which is the image of a homomorphism from q to I; a *minimal match* is a match that is minimal for inclusion.

A query is *monotone* if $I \models q$ and $I \subseteq I'$ imply $I' \models q$ for any two instances I, I'. A query is *closed under homomorphisms* if we have $I' \models q$ whenever $I \models q$ and I has a homomorphism to I', for any I and I'. UCQ is an example of query class that is both monotone and closed under homomorphisms, while $\mathrm{UCQ}^{\neq}$ is monotone but not closed under homomorphisms.

3. PROBLEM STATEMENT

We study the problem of *probability evaluation*:

DEFINITION 3.1. *Given an instance I, a* probability valuation *is a function π that maps each fact of I to a value*[1] *in $[0,1]$. A probability valuation defines a* probability distribution *on subinstances of I, which we also write π by a slight abuse of notation. The distribution π is intuitively obtained by seeing each fact F as kept with probability $\pi(F)$ and removed with probability $1 - \pi(F)$, all such choices being independent. Formally, the probability of $I' \subseteq I$ in this distribution is:*

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

The probability evaluation problem *for a query q on a class $\mathcal{I}$ of relational instances asks, given an instance $I \in \mathcal{I}$ and a probability valuation π on I, what is the probability that q holds according to the probability distribution, i.e., it is the problem of computing $\pi(q, I) := \sum_{I' \subseteq I \text{ such that } I' \models q} \pi(I')$.*

In other words, probability evaluation asks for the probability of q over a TID instance defined by I and π. Note that we only consider classes $\mathcal{I}$ of instances with no associated

[1]All non-integer numbers are rational numbers represented as the pair of their numerator and denominator.

probabilities, and the probability valuation π is given as an additional input — it is not indicated in $\mathcal{I}$. The complexity of the probability evaluation problem will always be studied in *data complexity*: the query q and class $\mathcal{I}$ is fixed, and the input is the instance $I \in \mathcal{I}$ and the probability valuation.

We also explore the problem of computing *tractable lineages* (or *provenance*), defined and studied in Section 6 onwards.

We rely on results of [2] that show the tractability in data complexity of provenance computation and probability evaluation on treelike (i.e., bounded-treewidth) instances. This holds for *guarded second-order* queries, but as such queries collapse to MSO under bounded treewidth [29], we always use MSO queries here. First, [2] shows that we can construct Boolean circuits that represent the provenance of MSO queries on treelike instances; we can also construct monotone circuits for monotone queries. The results also apply to other semirings, but this will not be our focus here. Second, [2] shows that probability evaluation is then tractable, namely, *ra-linear*: in linear time up to the (polynomial) cost of arithmetic operations.

Our goal is thus to investigate to what extent we can generalize the following tractability result from [2]:

THEOREM 3.2 [2]. *For any signature σ, for any (monotone) MSO query q, for any $k \in \mathbb{N}$, there is an algorithm which, given an input instance I of treewidth $\leqslant k$:*

- *Computes a (monotone) Boolean provenance circuit of q on I, in linear time in I;*
- *Given a probability valuation of I, computes the probability of q on I, in ra-linear time.*

We first focus on the second point (probability evaluation) in Section 4, followed by a digression about non-probabilistic evaluation in Section 5. We then study the first point (lineages) in Sections 6–7. We close with a connection to safe queries in Section 8.

4. PROBABILITY EVALUATION

This section studies whether we can extend the above tractability result by lifting the bounded-treewidth requirement. We answer in the negative by a *dichotomy result* on arity-two signatures: there are queries for which probabilistic evaluation is tractable on bounded-treewidth families but is intractable on *any* efficiently constructible unbounded-treewidth family. A first technical issue is to formalize what we mean by *efficiently* constructible. We use the following notion:

DEFINITION 4.1. *We call $\mathcal{I}$* treewidth-constructible *if for all $k \in \mathbb{N}$, if $\mathcal{I}$ contains instances of treewidth $\geqslant k$, we can construct one in polynomial time given k written in unary*[2].

In particular, this implies that $\mathcal{I}$ must contain a subfamily of unbounded-treewidth instances that are small, i.e., have size polynomial in their treewidth. We discuss the impact of this choice of definition, and alternate definitions of *efficiently* constructible instances, in Section 5.

A second technical issue is that we need to restrict to signatures of arity 2. We will then show our dichotomy for *any* such signature. This suffices to show that our Theorem 3.2 cannot be generalized: its generalization should apply to any signature, in particular arity-2 ones. Yet, we do not know whether the dichotomy applies to signatures of arity > 2.

Our *main result* on probability evaluation is as follows. In this result, $\mathrm{FP}^{\#\mathrm{P}}$ is the class of function problems which can be solved in PTIME with a deterministic Turing machine having access to a #P-oracle, i.e., an oracle for counting problems that can be expressed as the number of accepting paths for a nondeterministic PTIME Turing machine.

THEOREM 4.2. *Let σ be an arbitrary arity-2 signature. Let $\mathcal{I}$ be a treewidth-constructible class of σ-instances. Then the following dichotomy holds:*

- *If there is $k \in \mathbb{N}$ such that $\mathsf{tw}(I) \leqslant k$ for every $I \in \mathcal{I}$, then for every MSO query q, the probability evaluation problem for q on instances of $\mathcal{I}$ is solvable in ra-linear time.*
- *Otherwise, there is an FO query q_{h} (depending on σ but not on $\mathcal{I}$) such that the probability evaluation problem for q_{h} on $\mathcal{I}$ is $\mathrm{FP}^{\#\mathrm{P}}$-complete under randomized polynomial time (RP) reductions.*

The first part of this result is precisely the second point of Theorem 3.2. We thus sketch the proof of the hardness result of the second part. Pay close attention to the statement: while some FO queries (in particular, unsafe CQs [19]) may have $\mathrm{FP}^{\#\mathrm{P}}$-hard probability evaluation when *all* input instances are allowed, our goal here is to build a query that is hard even when input instances are restricted to *arbitrary families* satisfying our conditions, a much harder claim.

We reduce from the problem of counting graph *matchings*, namely, the number of edge subsets of a graph that have no pair of incident edges. This problem is known to be #P-hard on 3-regular planar graphs [50]. We define a FO query q_{h} that tests for matchings on such graphs (encoded in a certain way), and we rely on the connection between probability evaluation and model counting so that the probability of q_{h} on (an encoding of) a graph G reflects its number of matchings.

The main idea is that 3-regular planar graphs can be *extracted* from our family $\mathcal{I}$, using the following notion:

DEFINITION 4.3. *An* embedding *of a graph H in a graph G is an injective mapping f from the vertices of H to the vertices of G and a mapping g that maps the edges (u, v) of H to paths in G from $f(u)$ to $f(v)$, with all paths being vertex-disjoint. A graph H is a* topological minor *of a graph G if there is an embedding of H in G.*

We then use the following lemma, that rephrases the recent polynomial bound [11] on Robertson and Seymour's grid minor theorem [47] to the realm of topological minors; in so doing, we use the folklore observation that a degree-3 minor of a graph is always a topological minor:

LEMMA 4.4. *There is $c \in \mathbb{N}$ such that for any degree-3 planar graph H, for any graph G of treewidth $\geqslant |V(H)|^c$, H is a topological minor of G and an embedding of H in G can be computed in randomized polynomial time in $|G|$.*

Hence, intuitively, given an input 3-regular planar graph G (the input to the hard problem), we can extract it in randomized polynomial-time (RP) as a topological minor of (the Gaifman graph of) an instance I of our family $\mathcal{I}$ that we obtain using treewidth-constructibility. Once it is extracted, we show that, by choosing the right probability valuation for I, the probability of q_{h} on I allows us to reconstruct the answer to the original hard problem, namely, the number of

[2]The requirement that k be given in unary rather than in binary means that *more* instance families are treewidth-constructible, so treewidth-constructibility in this sense is a weaker assumption than if the input k could be written in binary.

matchings of G. The minor extraction step is what complicates the design of q_{h}, as q_{h} must then test for matchings in a way which is *invariant under subdivisions*: this is especially tricky in FO as we can only make local tests.

Choice of hard query. Not only is our query q_{h} independent from the class of instances $\mathcal{I}$, but it is also an FO query, so, in the *non-probabilistic* setting, its data complexity on any instance is in AC^0. In fact, our choice of q_{h} has also *linear-time* data complexity: one can determine in linear time in an input instance I whether $I \models q_{\text{h}}$. This contrasts sharply with the $\text{FP}^{\#\text{P}}$-completeness (under RP reductions) of *probability evaluation* for q_{h} on *any* unbounded-treewidth instance class (if it is treewidth-constructible).

The query q_{h}, however, is not monotone. We can alternatively show Theorem 4.2 for a MSO query which is *monotone*, but not in FO: more specifically, we use a query in $\text{C2RPQ}^{\neq}$, the class of *conjunctive two-way regular path queries* [8, 9] where we additionally allow disequalities between variables.

We will show an analogue of Theorem 4.2 in the setting of *tractable lineages* in Section 7, which applies to $\text{UCQ}^{\neq}$, an even weaker language. We do not know whether Theorem 4.2 itself can be shown with such queries, or with a *monotone* FO query. However, we know that Theorem 4.2 could not be shown with a query closed under homomorphism; this is implied by Proposition 7.9.

Providing valuations with the instances. When we fix the instance family $\mathcal{I}$, the probability valuation is not prescribed as part of the family, but can be freely chosen. If the instances of $\mathcal{I}$ were provided with their probability valuations, or if probability valuations were forced to be 1/2, then it is unlikely that an equivalent to Theorem 4.2 would hold.

Indeed, fix *any* query q such that, given any instance I, it is in #P to count how many subinstances of I satisfy q; e.g., let q be a CQ. Consider a family $\mathcal{I}$ of instances *with valuations* such that there is only one instance in $\mathcal{I}$ per encoding length: e.g., take the class of R-grids with probability 1/2 on each edge, for some binary relation R. Consider the problem, given the *length* of the encoding of an instance I (written in unary), of computing how many subinstances of I satisfy q. This problem is in the class $\#\text{P}_1$ [49]. Hence, the probability computation problem for q on $\mathcal{I}$ is in $\text{FP}^{\#\text{P}_1}$: rewrite the encoding of the input instance I to a word of the same length in a unary alphabet, use the $\#\text{P}_1$-oracle to compute the number of subinstances, and normalize the result by dividing by the number of possible worlds of I.

It thus seems unlikely that probabilistic evaluation of q on $\mathcal{I}$ with its valuations is #P-hard, so that our dichotomy result probably does not adapt if input instance families are provided with their valuations.

5. NON-PROBABILISTIC EVALUATION

Theorem 4.2 in Section 4 uses the recent technology of [11] that shows polynomial bounds for the grid minor theorem of [47]. These improved bounds also yield new results in the non-probabilistic setting. We accordingly study in this section the problem of *non-probabilistic* query evaluation, again defined in terms of data complexity:

Definition 5.1. *The* evaluation problem *(or* model-checking problem*), for a fixed query q on an instance family $\mathcal{I}$, is as follows: given an instance $I \in \mathcal{I}$, decide whether $I \models q$.*

Observe that the probability evaluation problem in Section 4 allowed the valuation to set edges to have probability 0. We could thus restrict to any subinstance of an instance in the class $\mathcal{I}$. In other words, the freedom to choose valuations in probability evaluation gave us at least the possibility of choosing subinstances for non-probabilistic query evaluation. This is why we will study in this section the non-probabilistic query evaluation problem on instance classes $\mathcal{I}$ which are *closed under taking subinstances* (or *subinstance-closed*), namely, for any $I \in \mathcal{I}$ and $I' \subseteq I$, we have $I' \in \mathcal{I}$.

As before, we will prove dichotomy results for this problem on unbounded-treewidth instance families, though we will use an MSO query rather than an FO query. We give two phrasings of our results. The first one, in Section 5.1, still requires treewidth-constructibility, and shows hardness for every level of the polynomial hierarchy, again under RP reductions. The second phrasing, in Section 5.2, is inspired by the results of [27], which it generalizes: it relies on complexity assumptions (namely, the non-uniform exponential time hypothesis) but works with a weaker notion of constructibility, namely, it requires treewidth to be strongly unbounded poly-logarithmically.

Last, we study in Section 5.3 the problem of *match counting* in the non-probabilistic setting, for which no analogous results seemed to exist.

As in Section 4, we restrict to signatures of arity 2.

5.1 Hardness Formulation

Our first dichotomy result for non-probabilistic MSO query evaluation is as follows; it is phrased using the notion of treewidth-constructibility. In this result, Σ_i^{P} denotes the complexity class at the i-th existential level of the polynomial hierarchy.

Theorem 5.2. *Let σ be an arbitrary arity-2 signature. Let $\mathcal{I}$ be a class of σ-instances which is treewidth-constructible and subinstance-closed. The following dichotomy holds:*

- *If there exists $k \in \mathbb{N}$ such that $\mathsf{tw}(I) \leqslant k$ for every $I \in \mathcal{I}$, then for every MSO query q, the evaluation problem for q on $\mathcal{I}$ is solvable in linear time.*
- *Otherwise, for each $i \in \mathbb{N}$, there is an MSO query q_{h}^i (depending only on σ, not on $\mathcal{I}$) such that the evaluation problem for q_{h}^i on $\mathcal{I}$ is Σ_i^{P}-hard under RP reductions.*

The upper bound is by Courcelle's results [14, 25], so our contribution is the hardness part, which we now sketch.

The main thing to change relative to the proof of Theorem 4.2 is the hard problems from which we reduce. We use hard problems on planar $\{1,3\}$-regular graphs, which we obtain from the *alternating coloring problem* as [27, 26], restricted to such graphs using techniques shown there, plus an additional construction to remove vertex labellings. Here is our formal claim about the existence of such hard problems:

Lemma 5.3. *For any $i \in \mathbb{N}$, there exists an MSO formula ψ_i on the signature of graphs such that the evaluation of ψ_i on planar $\{1,3\}$-regular graphs is Σ_i^{P}-hard. Moreover, for any such graph G, we have $G \models \psi_i$ iff $G' \models \psi_i$ for any subdivision G' of G.*

The rest of the proof of Theorem 5.2 proceeds similarly as that of Theorem 4.2.

Hypotheses. Theorem 5.2 relies crucially on the class $\mathcal{I}$ being *subinstance-closed*. Otherwise, considering the class $\mathcal{I}$ of cliques of a single binary relation E, this class is clearly

Table 1: Summary of results for non-probabilistic query evaluation: if a graph class has unbounded treewidth in *some* sense, is closed under *some* operations, and has (in data complexity) tractable model checking in *some* sense for *some* logic, then *some* complexity assumption is violated

Logic	Unboundedness	Closure	Tractability	Consequence	Source
MSO	unbounded	subgraph	PTIME	no violation: holds for some classes	[41] Prop. 32
MSO_2	unbounded	subgraph	PTIME	no violation: holds for some classes	[37] (remark)
∃MSO	unbounded	topolog. minors	PTIME	P = NP	[41] Thm. 11
MSO_2	strongly unb. polylog.	subgraph	PTIME	PH ⊆ DTIME($2^{o(n)}$)	[37] Thm. 1.2
MSO	densely unb. polylog.	subgr., vert. lab.	quasi-poly	PH ⊆ DTIME($2^{o(n)}$)/subEXP	[27] Thm. 5.5
MSO	unb., treewidth-constr.	subgraph	PTIME	PH ⊆ RP	**here Thm. 5.2**
MSO	densely unb. polylog.	subgraph	quasi-poly	PH ⊆ DTIME($2^{o(n)}$)/subEXP	**here Thm. 5.5**

unbounded-treewidth and treewidth-constructible, yet it has bounded clique-width so MSO query evaluation has linear data complexity on this class [16].

Further, the hypothesis of *treewidth-constructibility* is also crucial. Without this assumption, Proposition 32 of [41] shows the existence of graph families of unbounded treewidth which are subinstance-closed yet for which MSO query evaluation is in PTIME.

5.2 Alternate Formulation

We now give an alternative phrasing of Theorem 5.2 which connects it to the existing results of [37, 27]. Table 1 tersely summarizes their results in comparison to our own results and other related results. As [37, 27] are phrased in terms of graphs, and not arbitrary arity-2 relational instances, we do so as well in this subsection. Before stating our result, we summarize these earlier works to explain how our work relates to them.

[37, 27] show the intractability of MSO on any subgraph-closed unbounded-treewidth families of graphs, under finer notions than our *treewidth-constructibility*. Kreutzer and Tazari [37] proposed the notion of families of graphs with treewidth *strongly unbounded poly-logarithmically* and showed that MSO_2 (MSO with quantifications over both vertex- and edge-sets) over any such graph families is not fixed-parameter tractable in a strong sense (it is not in XP), unless the exponential-time hypothesis (ETH) fails. Ganian et al. [27] proved a related result, introducing the weaker notion of *densely unbounded poly-logarithmically* but requiring graph families to be closed under *vertex relabeling*; in such a setting, Theorem 4.1 of [27] shows that MSO (with vertex labels) cannot be fixed-parameter quasi-polynomial unless the *non-uniform* exponential-time hypothesis fails.

These two results of [37] and [27] are incomparable: [37] requires a stronger unboundedness notion (strongly unbounded vs densely unbounded) and a stronger query language (MSO_2 vs MSO), but it does not require vertex relabeling, and makes a weaker complexity theory assumption (ETH vs non-uniform ETH). See the Introduction of [27] for a detailed comparison.

Our Theorem 5.2 uses MSO and no vertex labeling, but it requires *treewidth-constructibility*, which is stronger than densely/strongly poly-logarithmic unboundedness: strongly unboundedness only requires constructibility in $o(2^n)$ and densely unboundedness does not require constructibility at all. The advantage of treewidth-constructibility is that we were able to show *hardness* of our problem (under RP reductions), without making *any* complexity assumptions. However, if we make the same complexity-theoretic hypotheses as [27], we now show that we can phrase our results in a similar way to theirs, and thus strengthen them.

We accordingly recall the notion of densely poly-logarithmic unboundedness, i.e., Definition 3.3 of [27]:

Definition 5.4. *A graph class $\mathcal{G}$ has treewidth* densely unbounded poly-logarithmically *if for all $c > 1$, for all $m \in \mathbb{N}$, there exists a graph $G \in \mathcal{G}$ such that $\mathsf{tw}(G) \geqslant m$ and $|V(G)| < O(2^{m^{1/c}})$.*

We now state our intractability result on densely unbounded poly-logarithmically graph classes. It is identical to Theorem 5.5 of [27] but applies to arbitrary MSO formulae, without a need for vertex relabeling: in the result, PH denotes the polynomial hierarchy. This result answers Conjecture 8.3 of [31] (as we pointed out in the Introduction).

Theorem 5.5. *Unless* PH ⊆ DTIME($2^{o(n)}$)/SubEXP*, there is no graph class $\mathcal{G}$ satisfying all three properties:*

- *a) $\mathcal{G}$ is closed under taking subgraphs;*
- *b) the treewidth of $\mathcal{G}$ is densely unbounded poly-logarithmically;*
- *c) the evaluation problem for any MSO query q on $\mathcal{G}$ is quasi-polynomial, i.e., in time $O(n^{\log^d n \times f(|q|)})$ for $n = |V(G)|$, an arbitrary constant $d \geqslant 1$, and some computable function f.*

The proof technique is essentially the same as in [27] up to using the newer results of [11]. It is immediate that an analogous result holds for probability query evaluation, as standard query evaluation obviously reduces to it (take the probability valuation giving probability 1 to each fact).

5.3 Match Counting

We conclude this section by moving to the problem of *match counting*, i.e., counting how many assignments satisfy a *non-Boolean MSO formula*. Match counting should not be confused with *model counting* (counting how many subinstances satisfy a Boolean formula) which is closely related[3] to probability evaluation.

To our knowledge, no dichotomy-like result on match counting for MSO queries was known. This section shows such a result; as in Section 5.1, we assume treewidth-constructibility, closure under subinstances, and arity-2 signatures.

We define the match counting problem as follows:

[3]The number of models of a query q in an instance I of size n is 2^n multiplied by the probability of q under the probability valuation of I that gives probability 1/2 to each fact of I.

Table 2: Upper bounds for lineage representations (including intractable ones) from Section 6. Note that computation bounds imply size bounds

Instance	Queries	Representation	Note	Time	Source	
bounded-pw	MSO	OBDD	$O(1)$ width	$O(n)$	**here**	**Thm. 6.7**
bounded-pw	(monotone) MSO	(monotone) circuit	bounded-pw	$O(n)$	**here**	**Prop. 6.8**
bounded-tw	MSO	OBDD		$O(\text{Poly}(n))$	**here**	**Thm. 6.5**
bounded-tw	(monotone) MSO	(monotone) circuit	bounded-tw	$O(n)$	[2]	Thm. 4.2
bounded-tw	MSO	d-DNNF		$O(n)$	**here**	**Thm. 6.11**
any	inversion-free UCQ	OBDD	$O(1)$ width	$O(\text{Poly}(n))$	[35]	Prop. 5
any	positive relational algebra	monotone formula		$O(\text{Poly}(n))$	[33]	Thm. 7.1
any	Datalog	monotone circuit		$O(\text{Poly}(n))$	[22]	Thm. 2

Definition 5.6. *The* counting problem *for an MSO formula $q(\mathbf{X})$ (with free second-order variables) on an instance family $\mathcal{I}$ is the problem, given an instance $I \in \mathcal{I}$, of counting how many vectors $\mathbf{A}$ of domain subsets are such that I satisfies $q(\mathbf{A})$.*

The restriction to free second-order variables is without loss of generality, as free first-order variables can be rewritten to free second-order ones, asserting in the formula that they must be interpreted as singletons.

We show the following dichotomy result:

Theorem 5.7. *Let σ be an arbitrary arity-2 signature. Let $\mathcal{I}$ be a subinstance-closed and treewidth-constructible class of σ-instances. The following dichotomy holds:*

- *If there exists $k \in \mathbb{N}$ such that $\mathsf{tw}(I) \leqslant k$ for every $I \in \mathcal{I}$, then for every MSO query $q(\mathbf{X})$ with free second-order variables, the counting problem for q on $\mathcal{I}$ is solvable in ra-linear time.*
- *Otherwise, there is an MSO query $q'_{\mathrm{h}}(X)$ (depending only on σ, not on $\mathcal{I}$) with one free second order variable such that the counting problem for q'_{h} on $\mathcal{I}$ is $FP^{\#\mathrm{P}}$-complete under RP reductions.*

The first claim is shown in [5]. The proof of the second claim proceeds as for Theorem 4.2. We reduce from the problem of counting Hamiltonian cycles in planar 3-regular graphs G, which is #P-hard [40], and which we express in MSO on the incidence graph of G.

Unlike in Theorem 4.2, the query q'_{h} does not have tractable model checking (as opposed to probability evaluation). We do not know whether we can show a similar result with such a tractable query.

6. LINEAGE UPPER BOUNDS

From *probability evaluation* in Section 4 (and its non-probabilistic variants in Section 5), we now turn to our second problem: the study of *tractable lineage representations*.

Indeed, a common way to achieve tractable probability evaluation is to represent the *lineage* of queries on input instances in a *tractable* formalism [35]. This section shows how the tractability of MSO probability evaluation on bounded-treewidth instances can be explained via lineages: Table 2 (upper part) summarizes the upper bounds that we prove.

Intuitively, the *lineage* of a query on an instance describes how the query depends on the facts of the instance. Formally:

Definition 6.1. *The* lineage *of a query q on an instance I is a Boolean function φ whose variables are the facts of I, such that, for any $I' \subseteq I$, $I' \models q$ iff the corresponding valuation makes φ true. If q is monotone, then φ is a monotone Boolean function, in which case it can equivalently be called the* PosBool$[X]$*-provenance [30] of q on I.*

Lineages are related to probability evaluation, because evaluating the probability of query q under a probability valuation π of instance I amounts to evaluating the probability of the lineage φ, under the corresponding probability valuation on variables. Thus, if we can represent φ in a formalism that enjoys tractable probability computation, then we can tractably evaluate the probability $\pi(q, I)$ of q on I.

In this section, we show that MSO queries on bounded-treewidth instances admit tractable lineage representations in many common formalisms: they have linear-size bounded-treewidth Boolean circuits (as shown in [2]), but also have polynomial-size OBDDs [7, 46] (with a stronger claim for *bounded-pathwidth*), as well as linear-size d-DNNFs [21]. Further, as we show, all these lineage representations can be efficiently computed. Note that in all these results, as in [2], tractability only refers to *data complexity*, with large constant factors in the query and instance width: we leave to future work the study of query and instance classes for which lineage computation enjoys a lower *combined complexity*.

After our results on tractable lineage computations in this section, we will study in the next section in which sense bounded-treewidth is *necessary* to obtain tractable lineages.

This section applies to signatures of arbitrary arity.

Bounded-treewidth circuits.

We first recall our results from [2] and introduce a first representation of lineages: *Boolean circuits*, called *provenance circuits* in [2] and *expression DAGs* in [34]:

Definition 6.2. *A* lineage circuit *for query q and instance I is a Boolean circuit with input gates and with NOT, OR, and AND internal gates, whose inputs are the facts of the database, and which computes the lineage of q on I. A* monotone *lineage circuit has no NOT gate. The* treewidth *and* pathwidth *of a lineage circuit are that of the circuit's graph.*

As recalled in Theorem 3.2, [2] showed that we can compute (monotone) lineage circuits for (monotone) MSO queries on bounded-treewidth instances in linear time. Further, these circuits themselves have *bounded-treewidth*, which is why probability evaluation is tractable on them, using message passing algorithms [39]. Hence:

Theorem 6.3 ([2], Theorems 4.4 and 5.3). *For any fixed MSO query q and constant $k \in \mathbb{N}$, given an input instance I of treewidth $\leqslant k$, we can compute in linear time a bounded-treewidth lineage circuit C of q on I.*

If q is monotone then we can take C to be monotone.

We study how to adapt this to other tractable lineage representations.

OBDDs. We start by defining *OBDDs*, a common tractable representation of Boolean functions [7, 46]:

DEFINITION 6.4. *An* ordered binary decision diagram *(or* OBDD*) is a rooted directed acyclic graph (DAG) whose leaves are labeled* 0 *or* 1*, and whose non-leaf nodes are labeled with a variable and have two outgoing edges labeled* 0 *and* 1*. We require that there exists a total order* Π *on the variables such that, for every path from the root to a leaf, no variable occurs in two different internal nodes on the path, and the order in which the variables occur is compatible with* Π*.*

An OBDD defines a Boolean function on its variables: each valuation is mapped to the value of the leaf reached from the root by following the path given by the valuation.

The size *of an OBDD is its number of nodes, and its* width *is the maximum number of nodes at every* level*, where a level is the set of nodes reachable by enumerating all possible values of variables in a prefix of* Π*.*

Probability evaluation for OBDDs is tractable [46]. Our result is that we can compute polynomial-size OBDDs for MSO queries on bounded-treewidth instances in PTIME:

THEOREM 6.5. *For any fixed MSO query* q *and constant* $k \in \mathbb{N}$*, there is* $c \in \mathbb{N}$ *such that, given an input instance* I *of treewidth* $\leqslant k$*, one can compute in time* $O(|I|^c)$ *an OBDD (of size* $O(|I|^c)$*) for the lineage of* q *on* I*.*

We show this using Corollary 2.14 of [34]: any bounded-treewidth Boolean circuit can be represented by an equivalent OBDD of polynomial width. We complete this result and show that the OBDD can also be computed in polynomial time, which clearly implies Theorem 6.5 (using Theorem 6.3):

LEMMA 6.6. *For any* $k \in \mathbb{N}$*, there is* $c \in \mathbb{N}$ *such that, given a Boolean circuit* C *of treewidth* k*, we can compute an equivalent OBDD in time* $O(|C|^c)$*.*

Bounded-pathwidth. We have explained the tractability of MSO probability evaluation on bounded-treewidth instances, showing that we could compute bounded-treewidth lineage circuits for them. We strengthen these results in the case of *bounded-pathwidth* instances, showing that we can compute *constant-width* OBDDs:

THEOREM 6.7. *For any fixed MSO query* q *and constant* $k \in \mathbb{N}$*, given an input instance* I *of pathwidth* $\leqslant k$*, one can compute in polynomial time an OBDD of constant width for the lineage of* q *on* I*.*

To prove the result, we first observe (adapting [2]) that we can compute *bounded-pathwidth* lineage circuits in linear time on bounded-pathwidth instances:

PROPOSITION 6.8. *For any fixed* $k \in \mathbb{N}$ *and (monotone) MSO query* q*, for any* σ*-instance* I *of* pathwidth $\leqslant k$*, we can construct a (monotone) lineage circuit* C *of* q *on* I *in time* $O(|I|)$*. The* pathwidth *of* C *only depends on* k *and* q *(not on* I*).*

By Corollary 2.13 of [34], this implies the existence of a constant-width OBDD representation, which we again show to be computable, proving Theorem 6.7.

LEMMA 6.9. *For any* $k \in \mathbb{N}$*, for any Boolean circuit* C *of pathwidth* $\leqslant k$*, we can compute in polynomial time in* C *an OBDD equivalent to* C *whose width depends only on* k*.*

d-DNNFs. We now turn to the more expressive tractable lineage formalism of *d-DNNFs*, introduced in [21]; we follow the definitions of [35]:

DEFINITION 6.10. *A* deterministic, decomposable negation normal form *(*d-DNNF*) is a Boolean circuit* C *that satisfies the following conditions:*

1. *Negation is only applied to input gates: the input of any NOT gate must always be an input gate.*
2. *The inputs of AND-gates depend on disjoint sets of input gates. Formally, for any AND-gate* g*, for any two gates* $g_1 \neq g_2$ *which are inputs of* g*, there is no input gate* g' *which is reachable (as a possibly indirect input) from both* g_1 *and* g_2*.*
3. *The inputs of OR-gates are mutually exclusive. Formally, for any OR-gate* g*, for any two gates* $g_1 \neq g_2$ *which are inputs of* g*, there is no valuation of the inputs of* C *under which* g_1 *and* g_2 *both evaluate to true.*

It is tractable to evaluate the probability of a d-DNNF [21], and d-DNNFs capture the tractability of probability evaluation for many safe queries (see [35]). We show that it also explains the ra-linearity of MSO probability evaluation on bounded-treewidth instances, as we can construct *linear* d-DNNFs for them:

THEOREM 6.11. *For any fixed MSO query* q *and constant* $k \in \mathbb{N}$*, given an input instance* I *of treewidth* $\leqslant k$*, one can compute in time* $O(|I|)$ *a d-DNNF representation of the lineage of* q *on* I*.*

7. OBDD SIZE BOUNDS

We have shown in the previous section that MSO queries on bounded-treewidth instances have tractable lineage representations as circuits and OBDDs. This section focuses on OBDDs and shows our *second main dichotomy result*: bounded-treewidth is necessary for MSO query lineages to have polynomial OBDDs.

We first state this result in Section 7.1. Its upper bound is Theorem 6.5, and its lower bound applies to a specific UCQ$^{\neq}$ q_{p} (which only depends on the signature). We show that q_{p} has no polynomial-width OBDDs on *any* arity-2 instance family with treewidth densely unbounded polylogarithmically. This second dichotomy result thus shows that bounded-treewidth is necessary for some UCQ$^{\neq}$ queries to have tractable OBDDs; it applies to a more restricted class than the FO query of our first main dichotomy result (Theorem 4.2), but applies to a different task (the computation of OBDD lineages, rather than probability evaluation).

We then study in Section 7.2 the language of *connected* UCQ$^{\neq}$. For this language, we show that queries can be classified in a *meta-dichotomy* result: we characterize the *intricate* queries, such as q_{p}, which have no polynomial OBDDs on any unbounded treewidth family in the sense above; and we show that non-intricate queries actually have *constant-width* OBDDs on some well-chosen unbounded-treewidth instance family. Hence, if a connected UCQ$^{\neq}$ has polynomial OBDDs on some unbounded-treewidth instance family, then it must have constant-width OBDDs on some other such family.

Finally, we investigate in Section 7.3 whether our second dichotomy result holds for more restricted fragments than UCQ$^{\neq}$. First, we show that connected CQ$^{\neq}$ queries are never intricate, so we cannot show our dichotomy result with such queries. Second, we show the same for connected UCQ; in

fact, we show that no query *closed under homomorphisms* could be used. We last show that our meta-dichotomy fails for disconnected queries.

As in Sections 4 and 5, we limit ourselves to arity-2 signatures in this section.

7.1 A Dichotomy on OBDD Size

This section shows that our Theorem 6.5 on the existence of tractable lineage representations as OBDDs is unlikely to extend to milder conditions than bounded-treewidth. Indeed, there are even UCQ$^{\neq}$ queries that have no polynomial-width OBDDs on any unbounded-treewidth input instance with treewidth densely unbounded poly-logarithmically, again on arity-two signatures. Here is our *second main dichotomy result* which shows this:

THEOREM 7.1. *There exists a constant $d \in \mathbb{N}$ such that the following holds. Let σ be an arbitrary arity-2 signature and $\mathcal{I}$ be a class of σ-instances. Assume there is a function $f(k) = O\big(2^{k^{1/d}}\big)$ such that, for all $k \in \mathbb{N}$, if $\mathcal{I}$ contains instances of treewidth $\geqslant k$, one of them has size $\leqslant f(k)$. We have the following dichotomy:*

- *If there is $k \in \mathbb{N}$ such that $\mathsf{tw}(I) \leqslant k$ for every $I \in \mathcal{I}$, then for every MSO query q, an OBDD of q on I can be computed in time polynomial in $|I|$.*
- *Otherwise, there is a* UCQ$^{\neq}$ *query q_{p} (depending on σ but not on $\mathcal{I}$) such that the width of any OBDD of q_{p} on $I \in \mathcal{I}$ cannot be bounded by any polynomial in $|I|$.*

This does *not* require treewidth-constructibility, and imposes instead a slight weakening[4] of densely unbounded poly-logarithmic treewidth. It does not require $\mathcal{I}$ to be subinstance-closed either, unlike in Section 5.

The first part of the theorem is by Theorem 6.5, so we sketch the proof of the second part. Our choice of UCQ$^{\neq}$ q_{p} intuitively tests the existence of a path of length 2 in the Gaifman graph of the instance, i.e., a violation of the fact that the possible world is a matching of the original instance. Again, while we know that probability evaluation for q_{p} is FP$^{\#\mathrm{P}}$-hard if we allow *arbitrary* input instances (as counting matchings reduces to it), our task is to show that q_{p} has no polynomial-width OBDDs when restricting to *any* instance family that satisfies the conditions, a much harder task.

To show this, we draw a link between treewidth and OBDD width for q_{p} on *individual* instances, with the following result (which is specific to q_{p}):

LEMMA 7.2. *Let σ be an arity-2 signature. There exist constants $d', k_0 \in \mathbb{N}$ such that for any instance I on σ of treewidth $\geqslant k_0$, the width of an OBDD for q_{p} on I is $\geqslant 2^{(\mathsf{tw}(I))^{1/d'}}$.*

7.2 A Meta-Dichotomy for UCQ$^{\neq}$

For which queries does Theorem 7.1 adapt? It does not extend to all *unsafe* [19] queries, as a query may be unsafe and still be tractable on some unbounded-treewidth instance family: for instance, the standard unsafe query $R(x) \wedge S(x,y) \wedge T(y)$ from [18] has trivial OBDDs on the family of S-grids without unary relations.

We answer this question, again on arity-2 signatures, by introducing a notion of *intricate* queries. We show that it precisely characterizes the *connected* UCQ$^{\neq}$ queries for which the dichotomy of Theorem 7.1 applies. Let us first recall the definition of *connected* UCQ$^{\neq}$ queries:

DEFINITION 7.3. *A* CQ$^{\neq}$ *is* connected *if, building the graph G on its atoms that connects those that share a variable (ignoring $\neq$-atoms), G is connected (in particular it has no isolated vertices, unless it consists of a single isolated vertex). A* UCQ$^{\neq}$ *is* connected *if all its* CQ$^{\neq}$ *disjuncts are* connected.

We now give our definition of *intricate* queries. We characterize them by looking at *line instances*:

DEFINITION 7.4. *A* line instance *is an instance I of the following form: a domain $a_1, \ldots, a_n$, and, for $1 \leqslant i < n$, one single binary fact between a_i and a_{i+1}: either $R(a_i, a_{i+1})$ for some $R \in \sigma$ or $R(a_{i+1}, a_i)$ for some binary $R \in \sigma$. (Recall that, as σ is* arity-two*, its maximal arity is two, so it must include at least one binary relation.)*

The intuition is that a query is intricate if, on any sufficiently long line instance, it must have a minimal match that contains the two middle facts (i.e., the ones that are incident to the middle element). Here is the formal definition of *intricate* queries:

DEFINITION 7.5. *A* UCQ$^{\neq}$ *q is* n-intricate *for $n \in \mathbb{N}$ if, for every line instance I with $|I| = 2n+2$, letting F and F' be the two facts of I incident to the middle element a_{n+2}, there is a* minimal *match of q on I that includes both F and F'.*

We call q intricate *if it is $|q|$-intricate.*

Observe that queries q with $|q| < 2$ clearly cannot be intricate. Further, if a query has no matches that include only binary facts, then it cannot be intricate; in other words, any disjunct that contains an atom for a unary relation can be ignored when determining whether a query is intricate. By contrast, our query q_{p} of Theorem 7.1 was designed to be intricate, in fact q_{p} is 0-intricate. Also note that an n-intricate query is always m-intricate for any $m > n$: consider the restriction of any line instance of size $2m+2$ to a line instance of size $2n+2$, and find a match in the restriction.

We note that we can decide whether UCQ$^{\neq}$ queries are intricate or not, by enumerating line instances. We do not know the precise complexity of this task:

LEMMA 7.6. *Given a connected* UCQ$^{\neq}$ *q, we can decide in PSPACE whether q is intricate.*

We can now state our *meta-dichotomy*: a dichotomy such as Theorem 7.1 holds for a connected UCQ$^{\neq}$ q if and only if it is intricate. Further, *non-intricate* queries must actually have *constant-width* OBDD on some counterexample unbounded-treewidth family:

THEOREM 7.7. *For any connected* UCQ$^{\neq}$ *q on an arity-2 signature:*

- *If q is not intricate, there is a treewidth-constructible and unbounded-treewidth family $\mathcal{I}$ of instances such that q has* constant-width *OBDDs on $\mathcal{I}$; the OBDDs can be computed in PTIME from the input instance.*
- *If q is intricate, then Theorem 7.1 applies to q: in particular, for any unbounded-treewidth family $\mathcal{I}$ of instances satisfying the hypotheses, q does not have polynomial-width OBDDs on $\mathcal{I}$.*

7.3 Other Query Classes

We finish by investigating the status of other query classes relative to our meta-dichotomy, to see whether Theorem 7.1 could be shown for queries in an even less expressive class than UCQ$^{\neq}$, such as CQ$^{\neq}$ or UCQ.

[4]The condition is weaker because we require the subexponentiality to work for some fixed d, not an arbitrary c.

Connected $\mathrm{CQ}^{\neq}$ *queries.* We classify the connected $\mathrm{CQ}^{\neq}$ queries relative to Theorem 7.7, by showing that a connected $\mathrm{CQ}^{\neq}$ can never be intricate. This explains why, for instance, the query $R(x) \wedge S(x,y) \wedge T(y)$ is not intricate, as is witnessed by the family of S-grids.

PROPOSITION 7.8. *A connected* $\mathrm{CQ}^{\neq}$ *is never intricate.*

By Theorem 7.7, this implies that any $\mathrm{CQ}^{\neq}$ query q has an unbounded-treewidth, treewidth-constructible family of instances $\mathcal{I}$ such that q has constant-width OBDDs on $\mathcal{I}$ (that can be computed in PTIME); and it also implies that we could not have proven Theorem 7.1 with a connected $\mathrm{CQ}^{\neq}$ query.

Homomorphism-closed. Second, we investigate the status in our meta-dichotomy of queries without inequalities, i.e., connected UCQs. We can in fact show a result for all queries that are *closed under homomorphisms*, no matter whether they are connected or not. Further, we can even choose a *single* class of instances which is easy for *all* query closed under homomorphisms. (Remember that our queries are always constant-free.)

PROPOSITION 7.9. *For any arity-2 signature, there is a treewidth-constructible instance family $\mathcal{I}$ with unbounded treewidth and $w \in \mathbb{N}$ such that any query q closed under homomorphisms has OBDDs of width w on $\mathcal{I}$ that can be computed in PTIME in the input instance.*

Hence, a connected UCQ is never intricate, so we could not have shown Theorem 7.1 with a UCQ query rather than a $\mathrm{UCQ}^{\neq}$ query.

Again, this result should not be confused with those of [34, 35]. Of course, not all homomorphism-closed queries, or even UCQs, have constant-width OBDDs on arbitrary instances. We are merely claiming the *existence* of high-treewidth instance classes for which we have constant-width OBDDs whatever the query.

Beyond connected queries. We consider last whether our dichotomy in Theorem 7.1 could extend to *disconnected* $\mathrm{CQ}^{\neq}$, which are not covered by Proposition 7.8 or by the meta-dichotomy of Theorem 7.7.

If the signature has more than one binary relation, this is hopeless: the easy argument used in the proof of Proposition 7.8 in this case can also apply to disconnected $\mathrm{CQ}^{\neq}$.

However, quite surprisingly, on signatures with a *single* binary relation (and arbitrarily many unary ones) we can show a weakening of Theorem 7.1 for a disconnected $\mathrm{CQ}^{\neq}$. The first part adapts (it holds for all MSO), so only the lower bound is interesting, which we can rephrase as before to a lower bound on OBDD width on individual input instances:

PROPOSITION 7.10. *Let σ be an arity-2 signature with only one binary relation. There exists a disconnected* $\mathrm{CQ}^{\neq}$ *query q_{d}, a constant $d' > 1$ and integer $n_0 \in \mathbb{N}$ such that: for any instance I on σ of size $\geqslant n_0$, letting k be the treewidth of I, the width of any OBDD for q_{d} is $\Omega(k^{1/d'})$.*

This implies that q_{d} does not satisfy the first part of the meta-dichotomy of Theorem 7.7. Surprisingly, however, we can show that q_{d} has OBDDs of width $O(k)$ on some unbounded-treewidth and treewidth-constructible instance class. Hence, q_{d} does not satisfy the second part of the meta-dichotomy either, so q_{d} witnesses that there are *disconnected* $\mathrm{CQ}^{\neq}$ that do not follow our meta-dichotomy at all! We leave to future work a more precise study of disconnected queries.

8. CONNECTION TO SAFE QUERIES

We conclude this paper by connecting our results to *query-based* tractability conditions. More specifically, we focus on UCQs with polynomial OBDD representations of their lineage: by the results of [35], those are the *inversion-free UCQs*. We will show that the tractability of such queries can be explained by our *data-based* tractability conditions: more precisely, for any inversion-free UCQ, there is a lineage-preserving rewriting of input instances to instances that have constant *tree-depth* [45], and hence (by Lemma 11 of [6]) have constant pathwidth and treewidth. Let us define *tree-depth*:

DEFINITION 8.1. *An* elimination forest *for an (undirected) graph G is a forest F on the vertices of G such that, for any edge $\{x, y\}$ of G, one of x and y is a descendant of the other in F. The* tree-depth *of G is the minimal height of an elimination forest of G. The* tree-depth *of an instance I is that of its Gaifman graph.*

This section applies to signatures of arbitrary arity.

Unfoldings. Our results are based on instance rewritings of a general kind, possibly of independent interest. We let I denote an arbitrary instance in this paragraph, and let q denote a query closed under homomorphisms.

DEFINITION 8.2. *An* unfolding *of instance I is an instance I' with a homomorphism h to I which is* bijective on facts*: for any fact $F(\mathbf{a})$ of I, there is exactly one fact $F(\mathbf{a}')$ in I such that $h(a'_i) = a_i$ for all i.*

The bijection defined by the homomorphism allows us to see the lineage of q on an unfolding I' of I as a Boolean function on the same variables as the lineage of q on I.

We use unfoldings as a tool to show lineage-preserving instance rewritings. Indeed, we can see from the homomorphism h from I' to I that any match of q in I' is preserved in I through h. In other words, the following is immediate:

LEMMA 8.3. *If I' is an unfolding of I and φ and φ' are the lineages of q on I and I', then for any valuation ν of the facts of I, if $\nu(\varphi') = 1$ then $\nu(\varphi) = 1$.*

The converse generally fails, but a sufficient condition is:

DEFINITION 8.4. *An unfolding I' of I* respects *q if, for any match $M \subseteq I$ of q on I, letting M' be its preimage in I', we have $M' \models q$.*

Intuitively, the unfolding does not "break" the matches of q. This ensures that the lineage is preserved exactly:

LEMMA 8.5. *If I' is an unfolding of I that respects q, then q has the same lineage on I and I'.*

Inversion-free UCQs. We use unfoldings to study Boolean constant-free *inversion-free* UCQ queries. We do not restate their formal definition here, and refer the reader to Section 2 of [35]. The following is known:

THEOREM 8.6 (Proposition 5 of [35]). *For any inversion-free UCQ q, for any input instance I, the lineage of q on I has an OBDD of constant width (i.e., the width only depends on q).*

When studying inversion-free UCQs, it is convenient to assume that the *ranking* transformation was applied to the query and instance [17, 19]. A UCQ is *ranked* if, defining a binary relation on its variables by setting $x < y$ when

x occurs before y in some atom, then $<$ has no cycle. In particular, in a ranked query, no variable occurs twice in an atom. An instance is *ranked* if there is a total order $<$ on its domain such that for any fact $R(\mathbf{a})$ and $1 \leqslant i < j \leqslant \text{arity}(R)$, we have $a_i < a_j$. In particular, no element occurs twice in a fact. Up to changing the signature, we can always rewrite a UCQ q to a ranked UCQ q', and rewrite separately any instance I to a ranked instance I', so that the lineage of q on I is the same as that of q' on I'; see [17, 19] for details.

We will thus assume that the ranking transformation has been applied to the query, and to the instance. Note that this can be performed in linear time in the instance, and does not change its treewidth, pathwidth, or tree-depth, as the Gaifman graph is unchanged by this operation.

Once this ranking transformation has been performed, we can show the following:

THEOREM 8.7. *For any ranked inversion-free UCQ q, for any ranked instance I, there is an unfolding I' of I that respects q and has tree-depth $\leqslant \text{arity}(\sigma)$.*

Hence, in particular, q has the same lineage on I' and on I, as shown by Lemma 8.5. As pathwidth is less than tree-depth [6], by Theorem 6.7, this implies the result of Theorem 8.6, and (via Proposition 6.8) generalizes it slightly: it shows that the lineage can even be represented by a bounded-pathwidth circuit.

Theorem 8.7 thus suggests that the tractability of probability evaluation for inversion-free UCQs can be understood in terms of bounded-tree-depth and bounded-pathwidth tractability: what inversion-free UCQs "see" in an instance is a bounded tree-depth structure.

9. CONCLUSION

The main result of this work justifies that bounded treewidth is the right condition on instances to make probability evaluation tractable, with a dichotomy between *ra-linear* evaluation for *MSO* queries assuming bounded treewidth, and *$FP^{\#P}$-hardness* under RP reductions for *FO queries* otherwise. Our second main result extends this to $\text{UCQ}^{\neq}$ for tractable OBDD representations, namely, to the *intricate* queries that we identify: they have no polynomial OBDDs on *any* sufficiently dense unbounded-treewidth instance class.

We do not know if our results extend beyond arity-2, e.g., using techniques from CSPs [44]. Another question is whether the first dichotomy result extends to more restricted query classes, or even to the $\text{UCQ}^{\neq}$ q_{p} of Theorem 7.1: indeed, probability evaluation of q_{p} is #P-hard but we were unable to show this *under arbitrary subdivisions*. Another extension would be to use PTIME rather than RP reductions, which would follow from a derandomization of [11]. In terms of lineage, we do not know either if our OBDD results extend to other lineage classes, e.g., FBDDs or d-DNNFs.

Our main hope, though, concerns the *unfolding* technique of Section 8. We showed that, for inversion-free UCQs, unfolding *always* reduces an instance to a bounded-treewidth one while preserving lineage. Could there be a *query-dependent*, lineage-preserving unfolding of instances, lowering the treewidth by undoing joins that the query does not "see"? Such a technique could yield a tractability condition on the instance *and* query, covering and extending *both* bounded-treewidth and safe queries. It could also be practically useful to approximate query probabilities, maybe in conjunction with the query-based *dissociation* technique [28].

Acknowledgements. We thank Mikaël Monet for careful proofreading, and Chandra Chekuri and Dan Suciu for technical clarifications on [11] and [35], respectively. We also thank the anonymous reviewers of PODS 2016 for their valuable comments, in particular for strengthening Theorem 8.7 to tree-depth. This work was partly funded by the Télécom ParisTech Research Chair on Big Data and Market Insights.

10. REFERENCES

[1] A. Amarilli. *Leveraging the Structure of Uncertain Data*. PhD thesis, Télécom ParisTech, 2016. 2016-ENST-0021. https://tel.archives-ouvertes.fr/tel-01345836.

[2] A. Amarilli, P. Bourhis, and P. Senellart. Provenance circuits for trees and treelike instances. In *ICALP*, 2015.

[3] A. Amarilli, P. Bourhis, and P. Senellart. Provenance circuits for trees and treelike instances (extended version). https://arxiv.org/abs/1511.08723, 2015.

[4] A. Amarilli, P. Bourhis, and P. Senellart. Tractable lineages on treelike instances: Limits and extensions. In *PODS*, 2016. https://arxiv.org/abs/1604.02761.

[5] S. Arnborg, J. Lagergren, and D. Seese. Easy problems for tree-decomposable graphs. *J. Algorithms*, 12(2), 1991. http://www.sciencedirect.com/science/article/pii/019667749190006K.

[6] H. L. Bodlaender, J. R. Gilbert, H. Hafsteinsson, and T. Kloks. Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *J. Algorithms*, 18(2), 1995. http://www.sciencedirect.com/science/article/pii/S0196677485710097.

[7] R. E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Comput. Surv.*, 24(3), 1992. http://doai.io/10.1145/136035.136043.

[8] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Containment of conjunctive regular path queries with inverse. In *KR*, 2000. https://www.inf.unibz.it/~calvanese/papers-html/KR-2000.html.

[9] D. Calvanese, G. De Giacomo, and M. Y. Vardi. Decidable containment of recursive queries. *Theor. Comput. Sci.*, 336(1), 2005. http://www.sciencedirect.com/science/article/pii/S0304397504007236.

[10] V. Chandrasekaran, N. Srebro, and P. Harsha. Complexity of inference in graphical models. In *UAI*, 2008. https://arxiv.org/abs/1206.3240.

[11] C. Chekuri and J. Chuzhoy. Polynomial bounds for the grid-minor theorem. In *STOC*, 2014. https://arxiv.org/abs/1305.6577.

[12] S. Cohen, B. Kimelfeld, and Y. Sagiv. Running tree automata on probabilistic XML. In *PODS*, 2009. http://www.cs.huji.ac.il/~sara/papers/running-tree-automata.pdf.

[13] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata: Techniques and applications, 2007. http://tata.gforge.inria.fr/.

[14] B. Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inf. Comput.*, 85(1), 1990. http://www.sciencedirect.com/science/article/pii/089054019090043H.

[15] B. Courcelle, J. Engelfriet, and G. Rozenberg. Handle-rewriting hypergraph grammars. *JCSS*, 46(2), 1993. http://www.sciencedirect.com/science/article/pii/002200009390004G.

[16] B. Courcelle, J. A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theor. Comput. Sci.*, 33(2), 2000. https://perso.ens-lyon.fr/eric.thierry/Graphes2007/cmr00.pdf.

[17] N. Dalvi, K. Schnaitter, and D. Suciu. Computing query probability with incidence algebras. In *PODS*, 2010. http://doai.io/10.1145/1807085.1807113.

[18] N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *VLDBJ*, 16(4), 2007. https://homes.cs.washington.edu/~suciu/vldbj-probdb.pdf.

[19] N. Dalvi and D. Suciu. The dichotomy of probabilistic inference for unions of conjunctive queries. *JACM*, 59(6), 2012. https://homes.cs.washington.edu/~suciu/jacm-dichotomy.pdf.

[20] N. N. Dalvi and D. Suciu. The dichotomy of conjunctive queries on probabilistic structures. In *PODS*, 2007. https://arxiv.org/abs/cs/0612102.

[21] A. Darwiche. On the tractable counting of theory models and its application to truth maintenance and belief revision. *J. Applied Non-Classical Logics*, 11(1-2), 2001. https://arxiv.org/abs/cs/0003044.

[22] D. Deutch, T. Milo, S. Roy, and V. Tannen. Circuits for Datalog provenance. In *ICDT*, 2014. https://openproceedings.org/2014/conf/icdt/DeutchMRT14.pdf.

[23] R. Diestel. *Graph Theory*. Springer, 2005. https://www.emis.de/monographs/Diestel/en/.

[24] R. Fink and D. Olteanu. A dichotomy for non-repeating queries with negation in probabilistic databases. In *PODS*, 2014. https://www.cs.ox.ac.uk/dan.olteanu/papers/fo-pods14.pdf.

[25] J. Flum, M. Frick, and M. Grohe. Query evaluation via tree-decompositions. *J. ACM*, 49(6), 2002. https://home.mathematik.uni-freiburg.de/flum/preprints/query.ps.

[26] R. Ganian, P. Hlinený, J. Kneis, D. Meister, J. Obdrzálek, P. Rossmanith, and S. Sikdar. Are there any good digraph width measures? In *IPEC*, 2010. https://arxiv.org/abs/1004.1485.

[27] R. Ganian, P. Hliněnỳ, A. Langer, J. Obdržálek, P. Rossmanith, and S. Sikdar. Lower bounds on the complexity of MSO1 model-checking. *JCSS*, 1(80), 2014. https://arxiv.org/abs/1109.5804.

[28] W. Gatterbauer and D. Suciu. Approximate lifted inference with probabilistic databases. *PVLDB*, 8(5), 2015. http://www.vldb.org/pvldb/vol8/p629-gatterbauer.pdf.

[29] E. Grädel, C. Hirsch, and M. Otto. Back and forth between guarded and modal logics. *TOCL*, 3(3), 2002. https://logic.rwth-aachen.de/pub/graedel/GrHiOt-tocl02.ps.

[30] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *PODS*, 2007. http://repository.upenn.edu/cgi/viewcontent.cgi?article=1022&context=db_research.

[31] M. Grohe. Logic, graphs, and algorithms. *Logic and Automata: History and Perspectives*, 2, 2008. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.113.9457&rep=rep1&type=pdf.

[32] C. Huang and A. Darwiche. Inference in belief networks: A procedural guide. *Int. J. Approximate Reasoning*, 1996. http://www.sciencedirect.com/science/article/pii/S0888613X96000692.

[33] T. Imieliński and W. Lipski, Jr. Incomplete information in relational databases. *J. ACM*, 31(4), 1984. http://docslide.us/documents/incomplete-information-in-relational-databases-imielinski-lipski.html.

[34] A. K. Jha and D. Suciu. On the tractability of query compilation and bounded treewidth. In *ICDT*, 2012. https://homes.cs.washington.edu/~suciu/file37_paper.pdf.

[35] A. K. Jha and D. Suciu. Knowledge compilation meets database theory: Compiling queries to decision diagrams. *Theory Comput. Syst.*, 52(3), 2013. https://homes.cs.washington.edu/~suciu/camera_ready.pdf.

[36] S. Kreutzer. Algorithmic meta-theorems. In *Parameterized and Exact Computation*. Springer, 2008. https://arxiv.org/abs/0902.3616.

[37] S. Kreutzer and S. Tazari. Lower bounds for the complexity of monadic second-order logic. In *LICS*, 2010. https://arxiv.org/abs/1001.5019.

[38] J. Kwisthout, H. L. Bodlaender, and L. C. van der Gaag. The necessity of bounded treewidth for efficient inference in bayesian networks. In *ECAI*, 2010. http://www.socsci.ru.nl/~johank/ECAI-623.pdf.

[39] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *J. Royal Statistical Society. Series B*, 1988. https://www.eecis.udel.edu/~shatkay/Course/papers/Lauritzen1988.pdf.

[40] M. Liskiewicz, M. Ogihara, and S. Toda. The complexity of counting self-avoiding walks in subgraphs of two-dimensional grids and hypercubes. *Theor. Comput. Sci.*, 1-3(304), 2003. http://www.sciencedirect.com/science/article/pii/S030439750300080X.

[41] J. A. Makowsky and J. Marino. Tree-width and the monadic quantifier hierarchy. *Theor. Comput. Sci.*, 303(1), 2003. http://www.sciencedirect.com/science/article/pii/S0304397502004498.

[42] D. Marx. Can you beat treewidth? In *FOCS*, 2007. http://doai.io/10.1109/FOCS.2007.27.

[43] D. Marx. Can you beat treewidth? *Theory of Computing*, 6(1), 2010. http://theoryofcomputing.org/articles/v006a005/v006a005.pdf.

[44] D. Marx. Tractable hypergraph properties for constraint satisfaction and conjunctive queries. *JACM*, 60(6), 2013. https://arxiv.org/abs/0911.0801.

[45] J. Nešetřil and P. O. Mendez. *Sparsity: Graphs, Structures, and Algorithms*, chapter Bounded Height Trees and Tree-Depth. 2012.

[46] D. Olteanu and J. Huang. Using OBDDs for efficient query evaluation on probabilistic databases. In *SUM*, 2008. http://citeseerx.ist.psu.edu/viewdoc/versions?doi=10.1.1.182.686.

[47] N. Robertson and P. D. Seymour. Graph minors. V. Excluding a planar graph. *J. Comb. Theory, Ser. B*, 41(1), 1986. http://www.sciencedirect.com/science/article/pii/0095895686900304.

[48] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases*. Morgan & Claypool, 2011.

[49] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3), 1979.

[50] M. Xia, P. Zhang, and W. Zhao. Computational complexity of counting problems on 3-regular planar graphs. *Theor. Comput. Sci.*, 384(1), 2007. http://www.sciencedirect.com/science/article/pii/S0304397507004653.

A. PROOFS FOR SECTION 6

LEMMA 6.6. *For any $k \in \mathbb{N}$, there is $c \in \mathbb{N}$ such that, given a Boolean circuit C of treewidth k, we can compute an equivalent OBDD in time $O(|C|^c)$.*

Proof. We rely on Corollary 2.14 of [34]: there is a doubly exponential function f such that, for any $k \in \mathbb{N}$, there is $c' := f(k) \in \mathbb{N}$ such that, for any tree decomposition T of width $\leqslant k$ of C, the OBDD O obtained for a certain variable order Π^R has width c'.

The order Π^R on variables is defined following an in-order traversal of T where children are ordered by the number of variables in this subtree; clearly this quantity can be computed over the entire tree in PTIME, so the order Π^R can be computed in PTIME. We show that we can construct the OBDD O in PTIME as well, in a level-wise manner inspired by [35].

Write $\Pi^R = X_1, \ldots, X_n$, and construct O level-by-level in the following way. Assuming that we have constructed O up to level $l-1$, create two children for each node at level $l-1$ (depending on the value of variable X_l), and then merge all such children n and n' that are *equivalent*. To define this, call *equivalent* two partial valuations ν and ν' of variables $X_1, \ldots, X_l$ if the Boolean function represented by C on the other variables $X_{l+1}, \ldots, X_n$ under ν is the same as under ν'. Now, call n and n' *equivalent* if, for any partial valuation ν leading to n (represented by a path from the root of O to n) and any partial valuation ν' leading to n', these two partial valuations of $X_1, \ldots, X_l$ are equivalent. As we will always ensure in the construction, any paths leading to the parent node of n are equivalent partial valuations of $X_1, \ldots, X_{l-1}$, so n and n' are equivalent iff, picking any two valuations ν for n and ν' for n' by following a path from the root to n and to n' respectively, ν and ν' are equivalent.

Hence, it suffices to show that there is a function g such that we can test in time $O(|C|^{g(k)})$ whether two partial valuations are equivalent. Indeed, we can then build O in the indicated time, because the maximal number of node pairs to test at any level of the OBDD is $\leqslant (2 \cdot |C|^{f(k)})^2$: we had at most $|C|^{f(k)}$ at the previous level, and each of them creates two children, before we merge the equivalent children. Hence, if we can test equivalence in the indicated time, then clearly we can construct O in time $O(|C|^c)$ for $c := 1 + 1 + 2 \cdot f(k) + g(k)$ (the first term accounts for the linear number of levels, and the second term accounts for the linear time required to find a partial valuation for a node).

We thus show that the equivalence of partial valuations can be tested in time $O(|C|^{g(k)})$ for some function g. Considering two partial valuations ν and ν' of the same set of variables $\mathcal{X}$, let C_ν and $C_{\nu'}$ be the two circuits obtained from C by substituting the input gates for $\mathcal{X}$ with constant gates according to ν and ν' respectively. Note that C_ν and $C_{\nu'}$ have the same set of input gates $\mathcal{X}'$, formed precisely of the variables not in $\mathcal{X}$. We rename the internal gates of $C_{\nu'}$ so that the only gates shared between C_ν and $C_{\nu'}$ are the input gates $\mathcal{X}'$. Now, C' be the circuit obtained by taking the union of C_ν and $C_{\nu'}$ (on the same set of variables), and adding an output gate and a constant number of gates such that the output gate is true iff the output gates of C_ν and $C_{\nu'}$ carry different values (this can be done with 5 additional gates in total). It is easy to see that there is a valuation of $\mathcal{X}'$ that makes the circuit C' evaluate to true iff the partial valuations ν and ν' are *not* equivalent. Now, observe that we can immediately construct from T a tree decomposition T'' of width $\leqslant 2k+5$ of C'. Indeed, it is obvious that T is a tree decomposition of C_ν, and we can rename gates to obtain from T a tree decomposition T' of $C_{\nu'}$, such that T and T' both have the same width k and the same skeleton. Now, construct T'' that has same skeleton as T and T' where each bag is the union of the corresponding bags of T and T', adding the 5 intermediate gates to each bag. The result T'' clearly has width $\leqslant 2k+5$ and it is immediate that it is a tree decomposition of C'.

We can then use message-passing techniques [39, 32] to determine in time exponential in $2k+5$ and polynomial in C' whether the bounded-treewidth circuit C' has a satisfying assignment, from which we deduce whether ν and ν' are equivalent. For details, see, e.g., Theorem D.2 of [2]. □

PROPOSITION 6.8. *For any fixed $k \in \mathbb{N}$ and (monotone) MSO query q, for any σ-instance I of* pathwidth $\leqslant k$*, we can construct a (monotone) lineage circuit C of q on I in time $O(|I|)$. The* pathwidth *of C only depends on k and q (not on I).*

Proof. Given a path decomposition of an instance I, which is a tree decomposition with a linear tree, the resulting tree encoding E of I (see [2, 3]) is clearly also a linear tree. From the proof of Theorem 4.4 of [3], we observe that the lineage circuit that we construct has a tree decomposition which can be made to be a path decomposition in this case, because it follows the structure of E. Hence, the circuit C has bounded pathwidth. □

LEMMA 6.9. *For any $k \in \mathbb{N}$, for any Boolean circuit C of pathwidth $\leqslant k$, we can compute in polynomial time in C an OBDD equivalent to C whose width depends only on k.*

Proof. As in the proof of Lemma 6.6, we can compute in PTIME the order Π^R on variables, and we can compute the OBDD under this order in the same way. This uses the fact that a path decomposition of circuit C is in particular a tree decomposition of C. □

THEOREM 6.11. *For any fixed MSO query q and constant $k \in \mathbb{N}$, given an input instance I of treewidth $\leqslant k$, one can compute in time $O(|I|)$ a d-DNNF representation of the lineage of q on I.*

Proof. We define a *bottom-up deterministic tree automaton* on alphabet Γ (or Γ-bDTA) in the standard manner. We start by adapting the proof of Proposition 3.1 of [3] to show the following result instead: a provenance d-DNNF of a *deterministic* $\overline{\Gamma}$-bDTA A on a $\overline{\Gamma}$-tree E can be constructed in time $O(|A| \cdot |E|)$. We construct the circuit exactly as in the proof of Proposition 3.1 of [3] and show that it is a d-DNNF.

First, observe that the only NOT gates that we use are the $g_n^{\neg \mathrm{i}}$, which are NOT gates of the g_n^{i}, which are input gates; so we only apply negation to leaf nodes.

Second, we show that the sets of leaves reachable from the children of any AND gate are pairwise disjoint. The AND gates that we create and that have multiple inputs are:

- The $g_n^{q_L, q_R}$, which are the AND of $g_{\mathrm{L}(n)}^{q_L}$ and $g_{\mathrm{R}(n)}^{q_L}$; now, $g_{\mathrm{L}(n)}^{q_L}$ only depends on the input gates $g_{n'}^{\mathrm{i}}$ for nodes n' of the subtree of E rooted at $\mathrm{L}(n)$, and likewise $g_{\mathrm{R}(n)}^{q_L}$ only depends on input gates in the right subtree;

- The $g_n^{q_L,q_R,\mathrm{i}}$, which are the AND of $g_n^{q_L,q_R}$ and g_n^{i}; now, the $g_n^{q_L,q_R}$ do not depend on g_n^{i}, only on input gates $g_{n'}^{\mathrm{i}}$ for n' a strict descendant of n in E;
- The $g_n^{q_L,q_R,\neg\mathrm{i}}$, which are the AND of $g_n^{q_L,q_R}$ and $g_n^{\neg\mathrm{i}}$; now, the $g_n^{q_L,q_R}$ do not depend on the sole input gate under $g_n^{\neg\mathrm{i}}$, i.e., g_n^{i}, but only on input gates $g_{n'}^{i}$ for n' a strict descendant of n in E.

Third, we show that the children of any OR gate are mutually exclusive. The OR gates that we create and that have multiple inputs are the following:

- The g_n^q when n is a leaf node of E, for which the claim is immediate, as the only two possible children are g_n^{i} and $g_n^{\neg\mathrm{i}}$ which are clearly mutually exclusive.
- The g_n^q when n is an internal node of E, which are the OR of gates of the form $g_n^{q_L,q_R,\mathrm{i}}$ or $g_n^{q_L,q_R,\neg\mathrm{i}}$ over several pairs q_L, q_R.

  To observe that these gates are mutually exclusive, remember that, for a valuation ν of the tree E, the gate $g_{n'}^q$ is true iff there is a run ρ of A on the subtree of $\nu(E)$ rooted at n' such that $\rho(n') = q$. However, as A is deterministic, for each n', there is at most one state q for which this is possible. Hence, for any valuation ν' of the circuit C, for our node n, there is at most one q'_L such that $g_{\mathrm{L}(n)}^{q'_L}$ is true under valuation ν', and only at most one q'_R such that $g_{\mathrm{R}(n)}^{q'_R}$ is true under ν'. Hence, by definition of the $g_n^{q_L,q_R}$, there is at most one of them which can be true under valuation ν', namely, $g_n^{q'_L,q'_R}$, which also means that only the gate $g_n^{q'_L,q'_R,\mathrm{i}}$ and the gate $g_n^{q'_L,q'_R,\neg\mathrm{i}}$ can be true under ν'. But these two gates are clearly mutually exclusive (only one can evaluate to true, depending on the value of $\nu(n)$), which proves the claim.
- The output gate g_0 which is the OR of gates of the form g_r^q for r the root node of E. Again, as A is deterministic, for any valuation ν' of C, letting ν be the corresponding valuation of the $\overline{\Gamma}$-tree E, there is only one state q' such that A has a run ρ on E with $\rho(r) = q'$, so at most one state q' such that $g_r^{q'}$ is true under ν.

Hence, the circuit constructed in the proof of Proposition 3.1 of [2] is a d-DNNF representation of the lineage of the automaton on the tree which has linear size.

We now adapt the proof of Theorem 4.2 of [3]. The theorem proceeds by constructing a bNTA for the query q [14] on the alphabet Γ_σ^k and modifying it to obtain a bNTA A' on $\overline{\Gamma_\sigma^k}$. We now additionally convert A' to a bDTA A'' on the same alphabet, which we can do using standard techniques [13]. All of this is performed independently of the instance.

Now, we conclude using the rest of the proof of Theorem 4.2 of [2]. The resulting circuit C' is the result of (bijectively) renaming the input gates, and replacing some input gates by constant gates, on the circuit C produced by Proposition 3.1 of [2]. However, by our previous observation, C is actually a d-DNNF circuit, so C' also is (up to evaluating negations of constant gates as constant gates). Hence, we have produced the desired d-DNNF, which by the statement of Theorem 4.2 of [2] is of linear size and is computed in linear time. □

B. PROOFS FOR SECTION 8

LEMMA 8.5. *If I' is an unfolding of I that respects q, then q has the same lineage on I and I'.*

Proof. By Lemma 8.3, it suffices to show that for any match M of q in I, the preimage M' of M by the bijection on facts is also a match of q; but this is precisely what is guaranteed by the fact that I respects q. □

THEOREM 8.7. *For any ranked inversion-free UCQ q, for any ranked instance I, there is an unfolding I' of I that respects q and has tree-depth $\leqslant \mathrm{arity}(\sigma)$.*

The roadmap of the proof is as follows. We use an *inversion-free expression* [35] for q to define an order on relation attributes which is compatible across relations. We then unfold each relation by distinguishing each element depending on the tuple of elements on the preceding positions; this is inspired by Proposition 5 of [35]. The result preserves the inversion-free expression and has a path decomposition that enumerates the facts lexicographically. To follow the roadmap, we first define inversion-free expressions as in [35]:

DEFINITION B.1. *A* hierarchical expression *[35] is a logical sentence built out of atoms, conjunction, disjunction, and existential quantification, where each variable is a* root variable*, i.e., occurs in all atoms in the scope of its existential quantifier.*

An inversion-free expression *is a hierarchical expression such that, for each relation symbol R, we can define a total order $<_R$ on its positions $\{R^1, \ldots, R^{\mathrm{arity}(R)}\}$, such that, in every R-atom $R(\mathbf{x})$, if $R^i <_R R^j$ then the quantifier $\exists x_j$ in the query is in the scope of the quantifier $\exists x_i$.*

By Proposition 2 of [35], a ranked UCQ is inversion-free iff it can be written as an inversion-free expression, so it suffices to show Theorem 8.7 for inversion-free expressions.

We first define our unfolding I' of an input instance I. For each fact $R(\mathbf{a})$ of I, we create the fact $R(\mathbf{b})$ defined as follows. Writing $R^{i_1} <_R \cdots <_R R^{i_n}$ the positions of R according to the total order $<_R$, we define b_{i_1} as the tuple (a_{i_1}), and define b_{i_j} as the tuple formed by concatenating $b_{i_{j-1}}$ and (a_{i_j}). We call f_R the operation thus defined, with $\mathbf{b} = f_R(\mathbf{a})$. Clearly the operation h mapping each tuple to its last element is a homomorphism from I' to I, and it is bijective on facts because it is the inverse of the operation that we described. Hence, I' is an unfolding of I. Note that this construction is similar to the one used in the proof of Proposition 5 in [35].

We must show that I' has bounded tree-depth. To do this, consider the elimination forest F defined on $\mathrm{dom}(I')$ by setting $\mathbf{b}$ to be the parent of $\mathbf{c}$ iff $\mathbf{b}$ is a longest strict prefix of $\mathbf{c}$. The forest F has one root per singleton element in $\mathrm{dom}(I')$: these elements correspond the (possibly strict) subset of $\mathrm{dom}(I)$ of the elements occurring at the first position R^{i_1} for the order $<_R$ for some relation R. It is clear that F is indeed an elimination forest for the Gaifman graph of I', as by construction any fact $R(\mathbf{b})$ of I' is such that, letting $n := \mathrm{arity}(R)$ and R^{i_n} be the last position of R in the order $<_R$, the elements of $\mathbf{b}$ are exactly the non-empty prefixes of b_{i_n}, so, for any pair b_i, b_j of elements of $\mathbf{b}$, one is a prefix of the other, so one is an ancestor of the other in F. We conclude by noticing that the elimination forest F has height $\mathrm{arity}(\sigma)$, so the tree-depth of I' is at most $\mathrm{arity}(\sigma)$.

The only thing left to show is that I' respects q. For this, let us consider the inversion-free expression Q of q. For any subexpression φ of Q with free variables $\mathbf{x}$, let us define the *ordered free variables* of φ, denoted $\mathrm{ofv}(\varphi)$, as follows. If

φ contains no atoms (i.e., it is the constant formula "true" or "false"), then $\mathbf{x}$ is empty and so is ofv(φ). Otherwise, as Q is inversion-free, it is in particular hierarchical, so all free variables of φ must occur in all atoms of φ: this is by definition, for any free variable x_i of φ, of the subexpression of Q that includes φ whose outermost operator is $\exists x_i$. Hence, consider any atom $A = R(\mathbf{x})$, and, remembering that no variable occurs twice in A (as Q is ranked), define ofv(φ) as the total order on $\mathbf{x}$ given by $x_i < x_j$ iff $R^i <_R R^j$.

It is clear that ofv(φ) is well-defined, i.e., that does not depend on our choice of atom in φ: this is because Q is an inversion-free expression, so the order of variables in atoms must reflect the order in which the variables are quantified.

We now show the claim that I' respects Q:

LEMMA B.2. *If I has a match M of Q, then, defining M' by mapping each fact $R(\mathbf{a})$ of M to the fact $R(f_R(\mathbf{a}))$ of I', M' is a match of Q in I'.*

Proof. Let f be defined on tuples of dom(I) by $f(\mathbf{a}) := (a_1, (a_1, a_2), \ldots, \mathbf{a})$. For any subformula φ with n free variables and any n-tuple $\mathbf{a}$ of dom(I), we write $I \models \varphi[\text{ofv}(\varphi) := \mathbf{a}]$ to mean the Boolean formula with constants obtained by substituting each variable in ofv(φ) by the corresponding element in $\mathbf{a}$ following the order of $\mathbf{a}$ and ofv(φ).

We proceed by induction on the subformulae of Q, showing that if a subformula φ and tuple $\mathbf{a} \in \text{dom}(M)$ is such that $M \models \varphi[\text{ofv}(\varphi) := \mathbf{a}]$, then $M' \models \varphi[\text{ofv}(\varphi) := f(\mathbf{a})]$.

- For atoms, this is by definition of ofv and of M'.
- For $\varphi \wedge \psi$, we observe that we have $\text{ofv}(\varphi) = \text{ofv}(\varphi \wedge \psi) = \text{ofv}(\psi)$: write $\mathbf{x}$ to refer to these ordered free variables. If $M \models (\varphi \wedge \psi)[\mathbf{x} := \mathbf{a}]$, then $M \models \varphi[\mathbf{x} := \mathbf{a}]$ and $M \models \psi[\mathbf{x} := \mathbf{a}]$, as by induction $M' \models \varphi[\mathbf{x} := f(\mathbf{a})]$ and $M' \models \psi[\mathbf{x} := f(\mathbf{a})]$, we deduce $M' \models (\varphi \wedge \psi)[\mathbf{x} := f(\mathbf{a})]$. For $\varphi \vee \psi$, the reasoning is the same.
- For $\varphi : \exists y\ \psi$, writing $\mathbf{x} := \text{ofv}(\varphi)$, by definition of ofv, y is the last variable of $\mathbf{x}' := \text{ofv}(\psi)$. As $M \models \varphi[\mathbf{x} := \mathbf{a}]$, by definition there is $c \in \text{dom}(M)$ such that, letting $\mathbf{a}'$ be the concatenation of $\mathbf{a}$ and c, $M \models \psi[\mathbf{x}' := \mathbf{a}']$. By induction hypothesis we have $M' \models \psi[\mathbf{x}' := f(\mathbf{a}')]$, and as removing the last element of $f(\mathbf{a}')$ yields $f(\mathbf{a})$, we deduce that $M' \models (\exists y\ \psi)[\mathbf{x} := f(\mathbf{a})]$.

The outcome of this induction is that $M \models Q$ implies $M' \models Q$, the desired claim. □