Solving a Special Case of the Intensional vs Extensional Conjecture in Probabilistic Databases

Mikaël Monet
DCC, University of Chile & IMFD Chile
Santiago, Chile
mikael.monet@imfd.cl

ABSTRACT

We consider the problem of exact probabilistic inference for Union of Conjunctive Queries (UCQs) on tuple-independent databases. For this problem, two approaches currently coexist. In the *extensional method*, query evaluation is performed by exploiting the structure of the query, and relies heavily on the use of the inclusion–exclusion principle. In the *intensional method*, one first builds a representation of the *lineage* of the query in a tractable formalism of knowledge compilation. The chosen formalism should then ensure that the probability can be efficiently computed using simple disjointness and independence assumptions, without the need of performing inclusion–exclusion. The extensional approach has long been thought to be strictly more powerful than the intensional approach, the reason being that for some queries, the use of inclusion–exclusion seemed unavoidable. In this paper we introduce a new technique to construct lineage representations as *deterministic decomposable circuits* in polynomial time. We prove that this technique applies to a class of UCQs that had been conjectured to separate the complexity of the two approaches. In essence, we show that relying on the inclusion–exclusion formula can be avoided by using negation. This result brings back hope to prove that the intensional approach can handle all tractable UCQs.

CCS CONCEPTS

• **Theory of computation → Data provenance**; **Incomplete, inconsistent, and uncertain databases**.

KEYWORDS

Tuple-independent databases, knowledge compilation, deterministic decomposable Boolean circuits, inclusion–exclusion principle

ACM Reference Format:
Mikaël Monet. 2018. Solving a Special Case of the Intensional vs Extensional Conjecture in Probabilistic Databases. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/1122445.1122456

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Woodstock '18, June 03–05, 2018, Woodstock, NY
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Probabilistic databases [36] have been introduced in answer to the need to capture data uncertainty and reason about it. In their simplest and most common form, probabilistic databases consist of a relational database in which each tuple is annotated with an independent probability value. This value is supposed to represent how confident we are about having the tuple in the database. This is known as the *tuple-independent database* (TID) model [17, 39]. While a traditional database can only *satisfy* or *violate* a Boolean query, a probabilistic database has a certain probability of satisfying it. Given a Boolean query Q, the *probabilistic query evaluation problem for* Q (PQE(Q)) then asks for the probability that the query holds on an input probabilistic database. Here, the complexity of PQE(Q) is measured as a function of the input database, hence considering that the Boolean query Q is fixed. This is known as *data complexity* [37], and is motivated by the fact that the queries are usually much smaller than the data.

When Q is a union of conjunctive queries, a dichotomy result is provided by the seminal work of Dalvi and Suciu [12]: either Q is *safe* and PQE(Q) is in polynomial time (PTIME), or Q is not safe and PQE(Q) is #P-hard. The algorithm to compute the probability of a safe UCQ exploits the first-order structure of the query to find a so called *safe extensional query plan*, using extended relational operators that can manipulate probabilities. An unusual feature of this algorithm is the use of the inclusion–exclusion principle (more precisely, of the *Möbius inversion formula*), which allows for distinct hard sub-queries to cancel each other. This approach is referred to as *extensional query evaluation*, or *lifted inference* [19, 30].

A second approach to PQE for safe UCQs is *intensional query evaluation* or *grounded inference*, and consists of two steps. First, compute a representation of the *lineage* [18] (also called *provenance*) of the query Q on the database D, which is a Boolean function intuitively representing which tuples of D suffice to satisfy Q. Second, perform weighted model counting on the lineage to obtain the probability. This is, for instance, the approach taken by the recent probabilistic database management system ProvSQL [33]. To ensure that model counting is tractable, we use the structure of the query to represent the lineage in tractable formalisms from the field of knowledge compilation. This includes read-once Boolean formulas [20], free or ordered binary decision diagrams (FBDDs [1], OBDDs [10, 38]), deterministic decomposable normal forms (d-DNNFs [13]), decision decomposable normal forms (dec-DNNFs [21, 22]), decomposable logic decision diagrams (DLDDs [6]), deterministic decomposable circuits (d-Ds [13, 26, 36][1]), structured versions thereof [29], etc. These can all be seen as restricted classes of Boolean circuits,

[1]The notation d-D was introduced in [26].

that *by definition* allow for efficient (in fact, linear) probability computation: the $\wedge$-gates are *decomposable* (their inputs depend on disjoint sets of variables, hence represent independent probabilistic events) and the $\vee$-gates are *deterministic* (their inputs represent disjoint probabilistic events). There are many advantages of this approach compared to lifted inference. First, and this is also true for non-probabilistic evaluation, the lineage can help explain the query answer [11]. Second, having the lineage in a good knowledge compilation formalism can be useful for various other applications: we could for instance update the tuples' probabilities and compute the new result easily, or compute the most probable state of the data that satisfies the query [14, 34], or enumerate satisfying states with constant delay [2], or produce random samples of satisfying states [34], etc. This ability to reuse the lineage for multiple tasks is precisely one of the main motivations of the field of knowledge compilation.

A natural question is then to ask if the intensional approach is as powerful as the extensional one. To answer it, a whole line of research started, whose goal is to determine exactly which queries can be handled by which formalisms of knowledge compilation. It is known, for example, that the UCQs whose lineages have polynomial-sized read-once formulas representations are exactly the *hierarchical-read-once* UCQs [24, 28], and that those having polynomial-sized OBDDs are exactly the *inversion-free* UCQs [24]. (This later result on OBDDs was extended to UCQs with disequalities atoms [23] and to self-join–free CQs with negations [16].) However, as far as we know, the characterization is open for all the other formalisms of knowledge compilation. What we call the *intensional–extensional conjecture*, formulated in [12, 24, 36], states that, for safe queries, extensional query evaluation is strictly more powerful than the knowledge compilation approach. Or, in other words, that there exists a UCQ which is safe (i.e., that can be handled by the extensional approach) whose lineages on arbitrary databases cannot be computed in PTIME in a tractable knowledge compilation formalism (i.e., cannot be handled by the intensional approach). As we have already mentioned, the conjecture depends on which tractable formalism we consider. However, generally speaking, the idea would be that knowledge compilation formalisms cannot simulate efficiently the Möbius inversion formula used in the algorithm for safe queries.

The conjecture has recently been shown in [6] to hold for the formalism of DLDDs (including OBDDs, FBDDs and dec-DNNFs), which captures the execution of modern model counting algorithms by restricting the power of determinism. Another independent result [9] shows that the conjecture also holds when we consider the class of *structured* d-DNNFs (d-SDNNFs, also including OBDDs), which are d-DNNFs that follow the structure of a *v-tree* [29]. However the question is still widely open for the most expressive formalisms, namely, d-DNNFs and d-Ds. Indeed, it could be the case that the conjecture does not hold for such expressive formalisms, which would imply that we could explain the tractability of all safe UCQs via knowledge compilation, and thus enjoy all of its advantages.

In this paper we focus on a class of Boolean queries that have been intensively studied already [6, 9, 12, 24, 26, 36], and that we name here the *$\mathcal{H}$-queries*. An $\mathcal{H}$-query can be defined as a Boolean combination of very simple CQs. To ease notation, we can always denote an $\mathcal{H}$-query as Q_φ, where φ is a Boolean function whose variables correspond to simple CQs (see Section 3.1 for the exact definition). Hence, when the Boolean function φ is monotone, Q_φ is a UCQ, and we write $\mathcal{H}^+$ the set of $\mathcal{H}$-queries that are UCQs. The safe $\mathcal{H}^+$-queries were conjectured in [12, 24, 36] to not have tractable lineage representations in any good knowledge compilation formalism. In fact, the $\mathcal{H}^+$-queries are precisely the ones that were used to prove the intensional–extensional conjecture for both DLDDs [6] and d-SDNNFs [9], leaving little hope for the remaining formalisms. In a sense, these queries are the simplest UCQs that illustrate the need of the Möbius inversion formula in Dalvi and Suciu's algorithm. Hence, they are also the first serious obstacle to disproving the conjecture for the most expressive formalisms of knowledge compilation.

Contributions, short version. Building on preliminary investigations [26], we develop a new technique to construct d-Ds in polynomial time for some of the $\mathcal{H}$-queries, and we prove that our technique applies to *all the safe $\mathcal{H}^+$-queries*. What is more, we also show that this technique applies to some $\mathcal{H}$-queries that are not UCQs (i.e., in $\mathcal{H} \setminus \mathcal{H}^+$), and we show #P-hardness for a subset of those for which it does not work. A picture of the situation can be found in Figure 1. Not illustrated in Figure 1, we also provide a detailed analysis of when lineages for an $\mathcal{H}$-query Q_φ can be transformed into lineages for another $\mathcal{H}$-query $Q_{\varphi'}$, with only a polynomial increase in size.

Contributions, long version. We now explain these results and our methodology more exhaustively. The dichotomy theorem of Dalvi and Suciu implies that an $\mathcal{H}^+$-query Q_φ is safe if and only if the Möbius value of the *CNF lattice* of φ is zero (and for $Q_\varphi \in \mathcal{H} \setminus \mathcal{H}^+$ we do not know, because these are not UCQs). Our starting point is to reformulate that criterion by showing that this value is equal to the *Euler characteristic* of φ. This connection seems to have been unnoticed so far in the probabilistic database literature. Hence, we obtain that $Q_\varphi \in \mathcal{H}^+$ is safe if and only if the Euler characteristic of φ is zero. While it is not clear how to define the CNF lattice for the $\mathcal{H}$-queries that are not UCQs, one advantage of our characterization is that the Euler characteristic is defined for any Boolean function φ.

We then use this new characterization to show that all $\mathcal{H}$-queries Q_φ for which φ has zero Euler characteristic have d-D representation of their lineages constructible in polynomial time. This can be considered as the main result of this article. Its proof contains three ingredients:

- The first one is a result on constructing OBDDs for restricted fragments of UCQs with negations [16], that we use as a black box (but we explain the relevant parts in Appendix, for completeness).
- The second one is a notion of what we call a *fragmentable Boolean function*, intuitively designed to ensure that Q_φ has d-Ds constructible in PTIME whenever φ is fragmentable, and that relies on the result of [16].
- The third component is a certain transformation between Boolean functions, where φ can be transformed into φ' by iteratively (1) removing from φ two connected (i.e., that differ

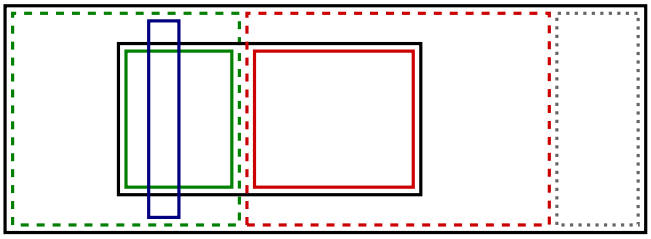

Figure 1: Picture of the situation. Outmost black rectangle: all $\mathcal{H}$-queries. Interior black horizontal rectangle: all $\mathcal{H}^+$-queries (UCQs). Solid green rectangle: all safe $\mathcal{H}^+$-queries according to the dichotomy of [12], i.e., whose CNF lattice have zero Möbius value. Solid red rectangle: all unsafe $\mathcal{H}^+$-queries. Blue vertical rectangle: all the $\mathcal{H}$-queries having OBDDs constructible in PTIME (upper bound [16], lower bound [6]). These are the inversion-free $\mathcal{H}$-queries and correspond to what we call degenerate Boolean functions. Dashed green rectangle (left): all the $\mathcal{H}$-queries to which our technique can apply, implying that they have d-Ds constructible in PTIME. These correspond to fragmentable Boolean functions, or equivalently, to those having zero Euler characteristic. Dashed red rectangle (right): the $\mathcal{H}$-queries corresponding to Boolean functions having non-zero Euler characteristic and for which we could show #P-hardness. Dotted gray rectangle: all the remaining $\mathcal{H}$-queries corresponding to Boolean functions having non-zero Euler characteristic. We conjecture these to be #P-hard as well. Note that each individual region contains an infinite number of queries.

in only one variable) satisfying valuations; or (2) adding to φ two connected valuations that did not satisfy φ.

This transformation defines an equivalence relation $\simeq$ between all Boolean functions, and we can show that if $\varphi \simeq \bot$ (the Boolean function that is always false), then φ is fragmentable. We then show that when φ has zero Euler characteristic it is equivalent to $\bot$, thus completing the proof. We also justify the definition of our transformation by showing that using only (1) or only (2) is not enough.

Our next main result is to show that if φ and φ' have the same Euler characteristic, then (a) PQE(Q_φ) and PQE($Q_{\varphi'}$) are reducible to each other (under PTIME Turing reductions); (b) we can compute in polynomial time d-D representations of the lineage of Q_φ on arbitrary databases iff we can do the same for $Q_{\varphi'}$; and (c) the lineages of Q_φ can be represented as d-Ds of polynomial size iff those of $Q_{\varphi'}$ can. To show these equivalences, we prove that the equivalences classes of $\simeq$ correspond exactly to the different values of the Euler characteristic. We think that this last combinatorial result can be of independent interest. We then obtain the equivalences (a-b-c) by observing that our transformation always preserves the corresponding properties. As a bonus, the first equivalence (a) also allows us to show #P-hardness of some $\mathcal{H}$-queries that are not UCQs, thus slightly extending the dichotomy of [12] for the $\mathcal{H}$-queries.

The difference with [26]. In [26], we studied with Dan Olteanu the compilation of the $\mathcal{H}$-queries into d-DNNFs. This preliminary work already contains the idea of covering the satisfying valuations of φ by removing two adjacent satisfying valuations, through the notion of what we called a *nice Boolean function* (of which fragmentability is a generalization). This article contained an experimental study that the proposed approach might work for the safe $\mathcal{H}^+$-queries, but had no proof. We still do not know if this approach works for all safe $\mathcal{H}^+$-queries, but we do know that it cannot work for all $\mathcal{H}$-queries with zero Euler characteristic (more on that in Section 7). What was missing to get to a proof for the $\mathcal{H}$-queries is the reformulation with the Euler characteristic (which allows one to understand the combinatorial meaning of having a zero Möbius value of the CNF lattice), and the fact that we might also need to *add* adjacent non-satisfying valuations.

Paper structure. To help the reader, we have depicted the structure of the paper and of the proofs as a DAG in Appendix A. We start in Section 2 with short preliminaries. In Section 3 we define the $\mathcal{H}$-queries, review what is known about them and reformulate the safety criterion for the $\mathcal{H}^+$-queries in terms of the Euler characteristic. We continue in Section 4 by introducing our notion of fragmentable Boolean function and we prove that if φ is fragmentable then Q_φ has d-Ds. In Section 5 we present our transformation and prove that if φ has zero Euler characteristic then φ is fragmentable, which implies our main result. We analyze in Section 6 what happens in the case of a non-zero Euler characteristic. Last, we expose in Section 7 open questions, justify the definition of our transformation, and discuss the applicability of our technique to a broader class of queries than the $\mathcal{H}$-queries. We conclude in Section 8.

2 PRELIMINARIES

Boolean functions. A (Boolean) *valuation* of a set V is a subset of V. We write 2^V the set of all valuations of V. For a Boolean valuation ν and variable l of V, we write $\nu^{(l)} \stackrel{\text{def}}{=} \begin{cases} \nu \cup \{l\} & \text{if } l \notin \nu \\ \nu \setminus \{l\} & \text{if } l \in \nu \end{cases}$, i.e., $\nu^{(l)}$ is ν except that the membership of l has been flipped. A *Boolean function* φ on V is a mapping $\varphi : 2^V \to \{\mathsf{False}, \mathsf{True}\}$ that associates to each valuation ν a value $\varphi(\nu)$ in $\{\mathsf{False}, \mathsf{True}\}$. A valuation ν is *satisfying* when $\varphi(\nu) = \mathsf{True}$, also written $\nu \models \varphi$. We write $\mathrm{SAT}(\varphi)$ the set of satisfying valuations of φ, and $\#\varphi$ the number of satisfying valuations of φ. We write $\bot$ (resp., $\top$) the Boolean function that maps every valuation to False (resp., True). A Boolean function φ is *monotone* if $\varphi(\nu) \implies \varphi(\nu')$ for every valuations ν, ν' such that $\nu \subseteq \nu'$. A monotone Boolean function can always be represented as a DNF of the form $C_0 \vee \ldots \vee C_n$, where we see each clause simply as the set of variables that it contains. Moreover, there is a unique *minimized* DNF representing φ, where no clause is a subset of another; we denote it by φ_{DNF}. We will similarly consider the unique minimized CNF representation φ_{CNF} of a monotone Boolean function φ.[2] Two Boolean functions φ, φ' are *disjoint* when $\varphi \wedge \varphi' = \bot$, or in other words, when $\mathrm{SAT}(\varphi \wedge \varphi') = \emptyset$.

Definition 2.1. Let φ be a Boolean function on V. We say that φ *depends on variable* $l \in V$ when there exists a valuation $\nu \subseteq V$ such that $\varphi(\nu) \neq \varphi(\nu^{(l)})$. We write $\mathrm{DEP}(\varphi) \subseteq V$ for the set of variables

[2] Note that the uniqueness of φ_{CNF} is less obvious than for φ_{DNF}, see for instance [31].

on which φ depends. We call φ *nondegenerate* if $\mathsf{DEP}(\varphi) = V$, and *degenerate* otherwise.

Definition 2.2 (See [32, 35]). Let φ be a Boolean function. The *Euler characteristic of* φ, denoted $\mathsf{e}(\varphi)$, is $\sum_{\nu \models \varphi} (-1)^{|\nu|}$.

Tuple-independent databases. A *tuple-independent (TID) database* is a pair (D, π) consisting of a relational instance D and a function π mapping each tuple $t \in D$ to a rational probability $\pi(t) \in [0; 1]$. A TID instance (D, π) defines a probability distribution Pr on $D' \subseteq D$, where $\Pr(D') \stackrel{\text{def}}{=} \prod_{t \in D'} \pi(t) \times \prod_{t \in D \setminus D'} (1 - \pi(t))$. Given a Boolean query Q, the *probabilistic query evaluation problem for* Q ($\mathsf{PQE}(Q)$) asks, given as input a TID instance (D, π), the probability that Q is satisfied in the distribution Pr. That is, formally, $\Pr(Q, (D, \pi)) \stackrel{\text{def}}{=} \sum_{D' \subseteq D \text{ s.t. } D' \models Q} \Pr(D')$. Dalvi and Suciu [12] have shown a dichotomy result on UCQs for PQE: either Q is *safe* and $\mathsf{PQE}(Q)$ is PTIME, or Q is not safe and $\mathsf{PQE}(Q)$ is #P-hard. Due to space constraints, we must refer to [12] for a presentation of their algorithm to compute the probability of a safe query, though it is not necessary to understand the current paper: we will reproduce the relevant parts in the next section. We write $\mathsf{PQE}(Q) \leqslant_{\mathrm{T}} \mathsf{PQE}(Q')$ when $\mathsf{PQE}(Q)$ reduces in PTIME (under Turing reductions) to $\mathsf{PQE}(Q')$. We write $\mathsf{PQE}(Q) \equiv_{\mathrm{T}} \mathsf{PQE}(Q')$ when we have both $\mathsf{PQE}(Q) \leqslant_{\mathrm{T}} \mathsf{PQE}(Q')$ and $\mathsf{PQE}(Q') \leqslant_{\mathrm{T}} \mathsf{PQE}(Q)$.

Lineages. The *lineage* of a Boolean query Q over D is a Boolean function $\mathsf{Lin}(Q, D)$ on the tuples of D mapping each sub-database $D' \subseteq D$ of D to True or False depending on whether D' satisfies Q or not. It is clear that, by definition, for any probability mapping $\pi : D \to [0; 1]$ we have $\Pr(Q, (D, \pi)) = \Pr(\mathsf{Lin}(Q, D), \pi)$, where the later is defined just as expected (see Definition B.2 in Appendix B.2 for the formal definition). This connection was first observed in [18].

Knowledge compilation formalisms. The lineage, being a Boolean function, can be represented with any formalism that represents Boolean functions (Boolean formulas, BDDs, Boolean circuits, etc). In the intensional query evaluation context, the crucial idea is to use a formalism that allows tractable probability computation. In this work we will specifically focus on *deterministic decomposable circuits* (d-Ds) and *deterministic decomposable normal forms* [13] (d-DNNFs). Let C be a Boolean circuit (featuring $\wedge$, $\vee$, $\neg$, and variable gates). An $\wedge$-gate g of C is *decomposable* if for every two input gates $g_1 \neq g_2$ of g we have $\mathsf{Vars}(g_1) \cap \mathsf{Vars}(g_2) = \emptyset$, where $\mathsf{Vars}(g)$ denotes the set of variable gates that have a directed path to g in C. We call C *decomposable* if each $\wedge$-gate is. An $\vee$-gate g of C is *deterministic* if for every pair $g_1 \neq g_2$ of input gates of g, the Boolean functions captured by g_1 and g_2 are disjoint. We call C *deterministic* if each $\vee$-gate is. A *negation normal form* (NNF) is a circuit in which the inputs of $\neg$-gates are always variable gates. Probability computation is in linear time for d-Ds (hence, for d-DNNFs): to compute the probability of a d-D, compute by a bottom-up pass the probability of each gate, where $\wedge$ gates are evaluated using $\times$, $\vee$ gates using $+$, and $\neg$ gates using $1 - x$.

For a formalism $\mathcal{C}$ of representation of Boolean functions, we denote by $\mathcal{C}(\mathrm{PTIME})$ the class of all Boolean queries Q such that, given as input an arbitrary database D, we can compute in polynomial time (in data complexity) an element of $\mathcal{C}$ representing $\mathsf{Lin}(Q, D)$. Similarly, we write $\mathcal{C}(\mathrm{PSIZE})$ for the class of Boolean queries Q such that, for any database D, there exists an element of $\mathcal{C}$ representing $\mathsf{Lin}(Q, D)$ whose size is polynomial in that of D (but this element is not necessarily computable in PTIME).

The Möbius function. The characterization of the safe $\mathcal{H}^+$-queries by [12] uses the Möbius function of a poset, which we define here. Let $P = (A, \leqslant)$ be a finite poset. The Möbius function [35] $\mu_P : A \times A \to \mathbb{Z}$ of P is defined on pairs (u, v) with $u \leqslant v$ by $\mu_P(u, u) \stackrel{\text{def}}{=} 1$ and $\mu_P(u, v) \stackrel{\text{def}}{=} -\sum_{u < w \leqslant v} \mu_P(w, v)$ for $u < v$. It is used to express the *Möbius inversion formula*, which is a generalization for posets of the inclusion–exclusion principle (see Proposition B.1 in Appendix B.2).

3 THE $\mathcal{H}$-QUERIES

In this section we define the $\mathcal{H}$-queries and review what is known about them, before reformulating the safety criterion for $\mathcal{H}^+$-queries.

3.1 Reviewing what is known

The building blocks of these queries are the conjunctive queries $h_{k,i}$, which were first defined in the work of Dalvi and Suciu [12] to show the hardness of UCQs that are not safe:

Definition 3.1. Let $k \in \mathbb{N}_{\geqslant 1}$, and let $R, S_1, \ldots, S_k, T$ be pairwise distinct relational predicates, with R and T being unary, and S_i for $0 \leqslant i \leqslant k$ being binary. The queries $h_{k,i}$ for $0 \leqslant i \leqslant k$ are defined by:

- $h_{k,0} = \exists x \exists y\, R(x) \wedge S_1(x, y)$;
- $h_{k,i} = \exists x \exists y\, S_i(x, y) \wedge S_{i+1}(x, y)$ for $1 \leqslant i < k$;
- $h_{k,k} = \exists x \exists y\, S_k(x, y) \wedge T(y)$.

As in [6], we define the $\mathcal{H}_k$-queries to be Boolean combinations of the queries $h_{k,i}$:

Definition 3.2. Let $k \in \mathbb{N}_{\geqslant 1}$ and φ be a Boolean function on variables $V = \{0, \ldots, k\}$. We define the query Q_φ to be the query represented by the first order formula $\varphi[0 \mapsto h_{k,0}, \ldots, k \mapsto h_{k,k}]$, i.e., φ where we substituted each variable $i \in V$ by the formula $h_{k,i}$.

The query class $\mathcal{H}_k$ (resp., $\mathcal{H}_k^+$) is then the set of queries Q_φ when φ ranges over all Boolean functions (resp., monotone Boolean functions) on variables $\{0, \ldots, k\}$. We finally define $\mathcal{H}$ (resp., $\mathcal{H}^+$) to be $\bigcup_{k=1}^{\infty} \mathcal{H}_k$ (resp., $\bigcup_{k=1}^{\infty} \mathcal{H}_k^+$). Observe that the queries in $\mathcal{H}^+$ are in particular (equivalent to some) UCQs.

Example 3.3. *Let $k = 3$, and φ_9 be the monotone Boolean function $(2 \vee 3) \wedge (0 \vee 3) \wedge (1 \vee 3) \wedge (0 \vee 1 \vee 2)$. Then Q_{φ_9} represents the query $(h_{32} \vee h_{33}) \wedge (h_{30} \vee h_{33}) \wedge (h_{31} \vee h_{33}) \wedge (h_{30} \vee h_{31} \vee h_{32}) \in \mathcal{H}_3^+$. This query was introduced in [12], where it is called q_9. It is the simplest safe $\mathcal{H}^+$-query for which Dalvi and Suciu's algorithm requires the Möbius inversion formula.*

At this point, it is important to warn the reader not to confuse the function φ, whose purpose is to define Q_φ, and the lineage of Q_φ on a database. While they are both Boolean functions, the similarity stops here: we use φ simply to alleviate the notations. Also, from now on and until the end, we fix $k \in \mathbb{N}_{\geqslant 1}$ and $V = \{0, \ldots, k\}$, and we will never use the symbols k and V for anything else.

To study the $\mathcal{H}_k$-queries, let us first listen to what the dichotomy of [12] has to say about them. We shall temporarily restrict our attention to monotone functions, i.e., to queries in $\mathcal{H}_k^+$, because the dichotomy theorem applies only to UCQs. We need to define the *CNF lattice* of φ:

Definition 3.4. Let φ be a monotone Boolean function represented by its (unique) minimized CNF $\varphi_{\text{CNF}} = C_0 \wedge \ldots \wedge C_n$ (remember that we see each clause simply as the set of variables that it contains). For $\mathbf{s} \subseteq \{0, \ldots, n\}$, we define $d_{\mathbf{s}} \stackrel{\text{def}}{=} \bigcup_{i \in \mathbf{s}} C_i$. Note that $d_\emptyset$ is $\emptyset$, and that it is possible to have $d_{\mathbf{s}} = d_{\mathbf{s}'}$ for $\mathbf{s} \neq \mathbf{s}'$. We define the poset $L^\varphi_{\text{CNF}} = (A, \leqslant)$, where A is $\{d_{\mathbf{s}} \mid \mathbf{s} \subseteq \{0, \ldots, n\}\}$, and where $\leqslant$ is reversed set inclusion[3]. One can check that L^φ_{CNF} is a (finite) lattice, but this fact is not important for this paper. In particular, L^φ_{CNF} has a greatest element $\hat{1}$, which is $\emptyset$, and has a least element $\hat{0}$, which is $\text{DEP}(\varphi)$.

The dichotomy theorem then tells us the following:

Proposition 3.5 ([12], see also [6, Section 3.3]). *Let φ be monotone. If φ is degenerate then $\text{PQE}(Q_\varphi)$ is PTIME. If φ is nondegenerate, let L^φ_{CNF} be the CNF lattice of φ, and μ_{CNF} be the Möbius function on L^φ_{CNF}. Then, if $\mu_{\text{CNF}}(\hat{0}, \hat{1}) = 0$ then $\text{PQE}(Q_\varphi)$ is PTIME, otherwise it is #P-hard.*

Example 3.6. *Consider φ_9 from Example 3.3. Clearly φ_9 is nondegenerate. The Hasse diagram of its CNF lattice is shown in Figure 2, with the value $\mu_{\text{CNF}}(n, \hat{1})$ for each node n. We have $\mu_{\text{CNF}}(\hat{0}, \hat{1}) = \mu_{\text{CNF}}(\{0, 1, 2, 3\}, \emptyset) = 0$, hence $\text{PQE}(Q_{\varphi_9})$ is PTIME.*

The solid green and red rectangles in Figure 1 represent the tractable and #P-hard $\mathcal{H}^+$-queries.

Having reviewed the tractability of the $\mathcal{H}^+$-queries, we now do the same concerning the compilation of $\mathcal{H}$-queries to knowledge compilation formalisms. First, when φ is degenerate, then $Q_\varphi \in$ OBDD(PTIME) (hence $Q_\varphi \in$ d-D(PTIME)):

Proposition 3.7 (Implied by Lemma 3.8 of [16]). *If φ (not necessarily monotone) is degenerate, then $Q_\varphi \in$ OBDD(PTIME).*

In this work we will only need the fact that $Q_\varphi \in$ d-D(PTIME) when φ is degenerate. Since our results depend on Proposition 3.7 and in order to be self-contained, we reproduce its proof in Appendix B.1, but it can be safely skipped: the reader can see it as a black box.

Proposition 3.7 is in sharp contrast to when φ is nondegenerate. Indeed in that case the authors of [6] show an exponential lower bound on the size of what they call *Decomposable Logic Decision Diagrams* (DLDDs) for Q_φ. A DLDD is a d-D in which the determinism of $\vee$-gates is restricted to simply choosing the value of a variable. That is, each $\vee$-gate is of the form $(v \wedge g) \vee (\neg v \wedge g')$ for some variable v. An OBDD being in particular a DLDD, we have represented in Figure 1 *all* the $\mathcal{H}$-queries in OBDD(PTIME) by the vertical blue rectangle (and we even know that those outside this rectangle are not even in OBDD(PSIZE), thanks to this lower bound).

[3] We could have defined it with $\leqslant$ being set inclusion, this is just a matter of taste. We chose to use reversed set inclusion to fit with related work.

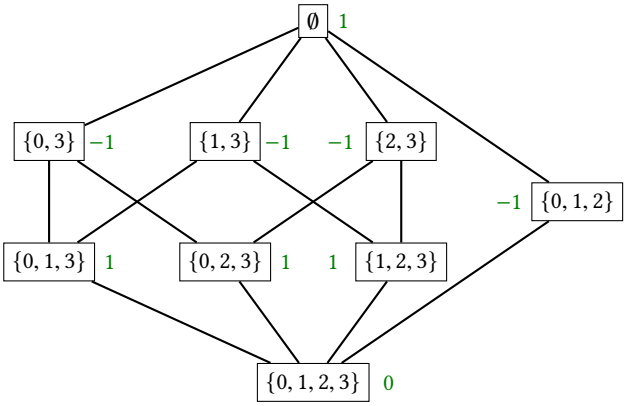

Figure 2: Hasse diagram of $L^{\varphi_9}_{\text{CNF}}$. The green value besides each node n is $\mu_{\text{CNF}}(n, \emptyset)$, and can be computed top-down following the definition.

When φ is monotone and nondegenerate, another independent exponential lower bound [9] tells us than we cannot impose *structuredness* either (i.e., use d-SDNNFs). These two lower bounds mean that to build tractable lineages for the $\mathcal{H}$-queries, one cannot restrict the expressivity of determinism too much, nor restrict the structure in which the variables appear. One question is then: do the safe nondegenerate $\mathcal{H}^+$-queries have polynomial sized (and computable in PTIME?) d-DNNFs (or d-Ds)?

3.2 Reformulation of safety for the $\mathcal{H}^+$-queries

With this section starts the presentation of our technical contributions. Here, we reformulate the safety criterion of [12] in terms of the Euler characteristic (recall Definition 2.2). As a by-product, we also show that for the $\mathcal{H}^+$-queries, using the CNF lattice or the *DNF lattice*[4] is equivalent (that question was left open in [26]). We show the following (remember that $k \in \mathbb{N}_{\geqslant 1}$ and $V = \{0, \ldots, k\}$ are fixed):

Lemma 3.8. *Let φ be a nondegenerate monotone Boolean function on V. Then we have $\text{e}(\varphi) = \mu_{\text{CNF}}(\hat{0}, \hat{1}) = (-1)^k \mu_{\text{DNF}}(\hat{0}, \hat{1})$, where μ_{CNF} (resp., μ_{DNF}) is the Möbius function of L^φ_{CNF} (resp., L^φ_{DNF}).*

Proof sketch. The idea is to use Möbius's inversion formula on the CNF and DNF lattices to obtain three different expressions of a variant of the characteristic polynomial of φ [27], and then to observe that the leading coefficients are equal to the targeted terms. The full proof can be found in Appendix B.2. □

A result that looks very similar to Lemma 3.8 is *Philip Hall's theorem* (see [35, Proposition 3.8.6] and text above), relating the Möbius value $\mu_P(\hat{0}, \hat{1})$ of a poset P with the Euler characteristic of a certain *abstract simplicial complex*[5] defined from P. However, we did not see a direct relation between this result and our lemma.

Combining Lemma 3.8 with Proposition 3.5, and with the observation that any degenerate function φ has $\text{e}(\varphi) = 0$, we obtain the following simple characterization of the safe $\mathcal{H}^+$-queries:

[4] Defined similarly as the CNF lattice but by starting from the DNF minimized expression of φ instead of the CNF expression.

[5] A synonym for the negation of a monotone Boolean function.

Corollary 3.9. *Let φ be monotone. If $\mathrm{e}(\varphi) = 0$ then $\mathrm{PQE}(Q_\varphi)$ is PTIME, otherwise it is #P-hard.*

Using the Euler characteristic instead of the Möbius function to characterize the safe $\mathcal{H}^+$-queries has two advantages. First, $\mathrm{e}(\varphi)$ is conceptually much simpler to grasp than $\mu_{\mathrm{CNF}}(\hat{0}, \hat{1})$.[6] The second one is that $\mathrm{e}(\varphi)$ is always defined, whereas it is not entirely clear how to define the CNF lattice when φ is not monotone.

While the proof of Lemma 3.8 is not very challenging, the connection does not seem to have been made in the literature so far.

4 FRAGMENTABLE BOOLEAN FUNCTIONS

We now introduce our notion of fragmentable Boolean function and show that whenever φ is fragmentable, then $Q_\varphi \in$ d-D(PTIME). We also show that if $\mathrm{e}(\varphi) \neq 0$ then φ cannot be fragmentable.

4.1 Definition

We first need to define what we call an *¬-∨-template*.

Definition 4.1. An *¬-∨-template* is a Boolean circuit whose internal nodes (i.e., those that are not a leaf) are either ¬- or ∨-gates. We call *holes* the variables (i.e., the leaves) of T. Let $l_0, \ldots, l_n$ be the holes of T, and let $\varphi_0, \ldots, \varphi_n$ be Boolean functions. Then $T[\varphi_0, \ldots, \varphi_n]$ represents the Boolean function obtained by substituing each hole l_i by φ_i, and then seeing the result as a "hybrid" Boolean circuit[7]. We say that $T[\varphi_0, \ldots, \varphi_n]$ is *deterministic* when every ∨-gate *in the template* is deterministic.

Observe that it can be the case that $T[\varphi_0, \ldots, \varphi_n]$ is deterministic while T itself is not. For a simple example, take T to be the template with two holes $l_0 \vee l_1$, and take $\varphi_0 \stackrel{\text{def}}{=} x$ and $\varphi_1 \stackrel{\text{def}}{=} \neg x$. Then $T[\varphi_0, \varphi_1]$ is deterministic but T is not. Proposition 3.7 then suggests the following definition:

Definition 4.2. We say that a Boolean function φ is *fragmentable* if there exist a ¬-∨-template T and degenerate Boolean functions $\varphi_0, \ldots, \varphi_n$ (with $n+1$ equals the number of holes of T) such that $T[\varphi_0, \ldots, \varphi_n]$ is deterministic and is equivalent to φ.

Note that in Definition 4.1 we allowed ¬-∨-templates to consist of a single leaf, which is then also the root. In particular, this implies that any degenerate Boolean function is fragmentable.

Example 4.3. *Consider φ_9 from Example 3.3. Let T be the ¬-∨-template $T \stackrel{\text{def}}{=} l_0 \vee l_1 \vee l_2 \vee l_3$, and let $\varphi_0 \stackrel{\text{def}}{=} 0 \wedge \neg 2 \wedge 3$; $\varphi_1 \stackrel{\text{def}}{=} \neg 1 \wedge 2 \wedge 3$; $\varphi_2 \stackrel{\text{def}}{=} \neg 0 \wedge 1 \wedge 3$; and $\varphi_3 \stackrel{\text{def}}{=} 0 \wedge 1 \wedge 2$, which are all degenerate. One can easily see that $T[\varphi_0, \varphi_1, \varphi_2, \varphi_3]$ is deterministic (because any two φ_i, φ_j with $i \neq j$ are disjoint) and (less easily) that $T[\varphi_0, \varphi_1, \varphi_2, \varphi_3]$, i.e., $\varphi_0 \vee \varphi_1 \vee \varphi_2 \vee \varphi_3$, is equivalent to φ_9. Therefore, φ_9 is fragmentable. Observe the we did not use ¬-gates in the template.*

4.2 First properties

The main property of fragmentability is that it implies having d-Ds constructible in polynomial time. This is actually its sole purpose.

Proposition 4.4. *If φ is fragmentable then $Q_\varphi \in$ d-D(PTIME).*

Proof. Let T be a ¬-∨-template and $\varphi_0, \ldots, \varphi_n$ be degenerate Boolean functions such that $T[\varphi_0, \ldots, \varphi_n]$ if deterministic and equivalent to φ, and let D be an arbitrary database. Since $\varphi_0, \ldots \varphi_n$ are degenerate, by Proposition 3.7 we can construct in PTIME deterministic decomposable Boolean circuits $C_0, \ldots, C_n$ capturing the lineages $\mathsf{Lin}(Q_{\varphi_i}, D)$ of Q_{φ_i} on D, for $0 \leqslant i \leqslant n$. Let C_φ be the Boolean circuit obtained by plugging the circuits C_i at the holes of T. Then one can check that C_φ is a d-D and that it captures $\mathsf{Lin}(Q_\varphi, D)$. □

Remember that we are working with data complexity here, so we can assume that we know T and $\varphi_0, \ldots, \varphi_n$ already. Although we do not know the exact complexity of finding these given φ, the results of the next section will imply (Corollary 5.12) that this task is computable.

Example 4.5. *Continuing Example 4.3, since φ_9 is fragmentable we obtain that $Q_{\varphi_9} \in$ d-D(PTIME)*[8].

Another property of fragmentability is that it implies having zero Euler characteristic.

Proposition 4.6. *If φ is fragmentable then $\mathrm{e}(\varphi) = 0$.*

Proof. Easily proved by bottom-up induction on $T[\varphi_0, \ldots, \varphi_n]$ and by using the facts that (1) $\mathrm{e}(\varphi) = 0$ when φ is degenerate; (2) $\mathrm{e}(\neg\varphi) = -\mathrm{e}(\varphi)$; and (3) $\mathrm{e}(\varphi \vee \varphi') = \mathrm{e}(\varphi) + \mathrm{e}(\varphi')$ when φ and φ' are disjoint. □

At first glance, the definition of fragmentability seems completely ad-hoc. Indeed, it could very well be the case that some $\mathcal{H}$-query, say even $\mathcal{H}^+$-query, is in d-D(PTIME) by other means than Proposition 4.4. Yet, we will show in the next section the surprising fact that the converse of Proposition 4.6 is also true, implying that fragmentability is the right notion to consider.

5 UPPER BOUND

We dedicate this section to showing that having a zero Euler characteristic implies being fragmentable. Formally:

Proposition 5.1. *If $\mathrm{e}(\varphi) = 0$ then φ is fragmentable.*

Before embarking on its proof, we spell out its consequences. The first one is that all the $\mathcal{H}$-queries Q_φ with $\mathrm{e}(\varphi) = 0$ are in d-D(PTIME), thanks to Proposition 4.4:

Theorem 5.2. *If $\mathrm{e}(\varphi) = 0$ then $Q_\varphi \in$ d-D(PTIME).*

This is, in a sense, the main result of this article. We have represented in Figure 1 all the $\mathcal{H}$-queries with zero Euler characteristic by the dashed green rectangle. Theorem 5.2 applies in particular to all the safe $\mathcal{H}^+$-queries, thanks to Corollary 3.9. So we get:

Corollary 5.3. *All safe $\mathcal{H}^+$-queries are in d-D(PTIME).*

Remember that the intensional–extensional conjecture was thought to hold for the safe $\mathcal{H}^+$-queries, because the use of inclusion–exclusion seemed unavoidable. This surprising result shows that it *is*

[6] For instance, a satisfactory consequence is that we can easily express the number of (not necessarily monotone) functions with $\mathrm{e}(\varphi) = 0$: this is $\sum_{j=0}^{2^k} \binom{2^k}{j}^2$.

[7] "Hybrid" because the leaves of the circuit are Boolean functions, i.e., we do not care how these are represented concretely.

[8] We do not claim credit for this fact, as it has been known for at least 4 years by Guy Van den Broeck and Dan Olteanu (with whom we initially worked on the problem).

avoidable, and that we can get away by using only decomposability and determinism.

Less importantly, by combining Proposition 5.1 and Proposition 4.6, we obtain the following:

Corollary 5.4. *φ is fragmentable if and only if $\mathsf{e}(\varphi) = 0$.*

In the rest of this section, we prove Proposition 5.1. The idea is the following. In Section 5.1, we define a notion of transformation between Boolean functions. This transformation defines an equivalence relation (denoted $\simeq$) between all Boolean functions, and we show that for a Boolean function φ, if $\varphi \simeq \bot$ then φ is fragmentable. Then, in Section 5.2 we show that $\varphi \simeq \bot$ when $\mathsf{e}(\varphi) = 0$.

5.1 The transformation

(We remind once again the reader that the set $V = \{0, \ldots, k\}$ of variables is fixed, and that we only consider Boolean functions with variables V.) We start with the definition of our notion of transformation between Boolean functions:

Definition 5.5. Let φ, φ' be Boolean functions, $\nu \subseteq V$ a valuation and $l \in V$ a variable. Then we write $\varphi \overset{+(\nu,\, l)}{\rightsquigarrow} \varphi'$ whenever $\nu \not\models \varphi$ and $\nu^{(l)} \not\models \varphi$ and $\mathsf{SAT}(\varphi') = \mathsf{SAT}(\varphi) \cup \{\nu, \nu^{(l)}\}$. Similarly, we write $\varphi \overset{-(\nu,\, l)}{\rightsquigarrow} \varphi'$ whenever $\varphi' \overset{+(\nu,\, l)}{\rightsquigarrow} \varphi$, i.e., when $\nu \models \varphi$ and $\nu^{(l)} \models \varphi$ and $\mathsf{SAT}(\varphi') = \mathsf{SAT}(\varphi) \setminus \{\nu, \nu^{(l)}\}$. We will also use the following auxiliary relations:

- we write $\varphi \overset{\pm(\nu,\, l)}{\rightsquigarrow} \varphi'$ when $\varphi \overset{+(\nu,\, l)}{\rightsquigarrow} \varphi'$ or $\varphi \overset{-(\nu,\, l)}{\rightsquigarrow} \varphi'$;
- we write $\varphi \overset{+}{\rightsquigarrow} \varphi'$ when $\varphi \overset{+(\nu,\, l)}{\rightsquigarrow} \varphi'$ for some ν, l, and similarly for $\varphi \overset{-}{\rightsquigarrow} \varphi'$ and $\varphi \overset{\pm}{\rightsquigarrow} \varphi'$;
- we write, $\overset{+}{\rightsquigarrow}{}^*$, $\overset{-}{\rightsquigarrow}{}^*$ and $\overset{\pm}{\rightsquigarrow}{}^*$ for the reflexive transitive closures thereof.

It is clear from the definitions that $\overset{\pm}{\rightsquigarrow}{}^*$ is symmetric, so let us write $\simeq$ the induced equivalence relation. Next, we explain how to visually understand this transformation.

Definition 5.6. Let $\mathbf{G}_V$ be the undirected graph with node set 2^V and edge set $\{\{\nu, \nu^{(l)}\} \mid \nu \subseteq V, l \in V\}$. Let φ be a Boolean function. We define the colored graph $\mathbf{G}_V[\varphi]$ to be the graph $\mathbf{G}_V$ where we have colored every satisfying valuation of φ.

Example 5.7. *Consider again the function φ_9 from Example 3.3 (here $V = \{0, 1, 2, 3\}$) The graph $\mathbf{G}_V[\varphi_9]$ is shown in Figure 3 (ignore for now the fact that some edges are dashed and green).*

Then, we have $\varphi \overset{\pm}{\rightsquigarrow}{}^* \varphi'$ whenever we can go from $\mathbf{G}_V[\varphi]$ to $\mathbf{G}_V[\varphi']$ by iteratively (1) coloring two uncolored adjacent nodes; or (2) uncoloring two adjacent colored nodes. In particular, observe that this transformation never changes the Euler characteristic. We illustrate the transformation in Figure 4 (ignore for now the last sentence in the description).

We can now show that if $\varphi \simeq \bot$, then φ is fragmentable:

Proposition 5.8. *If $\varphi \simeq \bot$ then φ is fragmentable.*

Proof. For some $n \in \mathbb{N}$, we have $\bot = \varphi_0 \overset{\pm(\nu_1,\, l_1)}{\rightsquigarrow} \cdots \overset{\pm(\nu_n,\, l_n)}{\rightsquigarrow} \varphi_n = \varphi$ for some valuations and variables ν_i, l_i. We show by induction on $0 \leqslant i \leqslant n$ that φ_i is fragmentable by constructing a $\neg$-$\vee$-template T_i having $i + 1$ holes $l_0, \ldots, l_i$ and degenerate Boolean functions ψ_j for $0 \leqslant j \leqslant i$ such that $T_i[\psi_0, \ldots, \psi_i]$ is deterministic and represents φ_i. The base case of $i = 0$ is trivial since $\bot$ is degenerate, hence fragmentable. For the inductive case, supose $i > 0$ with φ_{i-1} fragmentable, and let T_{i-1} and $\psi_0, \ldots, \psi_{i-1}$ be a template and degenerate functions witnessing that φ_{i-1} is fragmentable. Define ψ_i by $\mathsf{SAT}(\psi_i) \overset{\text{def}}{=} \{\nu_i, \nu_i^{(l_i)}\}$. Surely ψ_i is degenerate (it does not depend on variable l_i). We then distinguish the two possible cases:

- we have $\varphi_{i-1} \overset{+(\nu_i,\, l_i)}{\rightsquigarrow} \varphi_i$. Then we can write φ_i as the deterministic disjunction $\varphi_{i-1} \vee \psi_i$. But then we can define T_i to be the template $T_i \overset{\text{def}}{=} T_{i-1} \vee l_i$, and one can easily check that $T_i[\psi_0, \ldots, \psi_i]$ is deterministic and represents φ_i. Therefore, φ_i is indeed fragmentable.
- we have $\varphi_{i-1} \overset{-(\nu_i,\, l_i)}{\rightsquigarrow} \varphi_i$. Then we can write φ_i as $\neg(\neg\varphi_{i-1} \vee \psi_i)$, with the disjunction being deterministic. But then we can define T_i to be the template $T_i \overset{\text{def}}{=} \neg(\neg T_{i-1} \vee l_i)$, and once again it is direct that $T_i[\psi_0, \ldots, \psi_i]$ is deterministic and represents φ_i. Hence φ_i is fragmentable. □

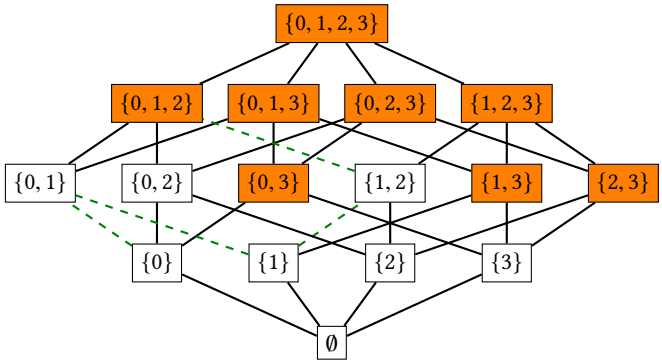

Figure 3: The colored graph $\mathbf{G}_V[\varphi_9]$. Note that it is different from $L^{\varphi_9}_{\text{CNF}}$ (Figure 2). In particular, the semantics of the nodes is not the same.

5.2 If $\mathsf{e}(\varphi) = 0$ then $\varphi \simeq \bot$

The missing piece to show Proposition 5.1 is to prove that whenever $\mathsf{e}(\varphi) = 0$ then φ is equivalent to $\bot$. This is what we do in this section. Formally:

Proposition 5.9. *If $\mathsf{e}(\varphi) = 0$ then $\varphi \simeq \bot$.*

In order to do that, we need two simple lemmas, which we will also reuse in the next section. The first is what we call the *chaining lemma*:

Lemma 5.10 (Chaining lemma). *Let φ be a Boolean function, and $\nu \neq \nu'$ be two valuations such that there is a simple path $\nu = \nu_0 - \cdots - \nu_{n+1} = \nu'$ from ν to ν' in $\mathbf{G}_V$ with $n \geqslant 0$ and $\nu_i \notin \mathsf{SAT}(\varphi)$ for $1 \leqslant i \leqslant n$. Then we have the following:*

Chainkilling. *If $(-1)^{|\nu|} \neq (-1)^{|\nu'|}$ (i.e., n is even) and $\{\nu, \nu'\} \subseteq \mathsf{SAT}(\varphi)$ then, defining φ' by $\mathsf{SAT}(\varphi') \overset{\text{def}}{=} \mathsf{SAT}(\varphi) \setminus \{\nu, \nu'\}$, we have $\varphi \overset{\pm}{\rightsquigarrow}{}^* \varphi'$. In other words, we can uncolor both ν and ν'. We say that* we have chainkilled ν and ν'.

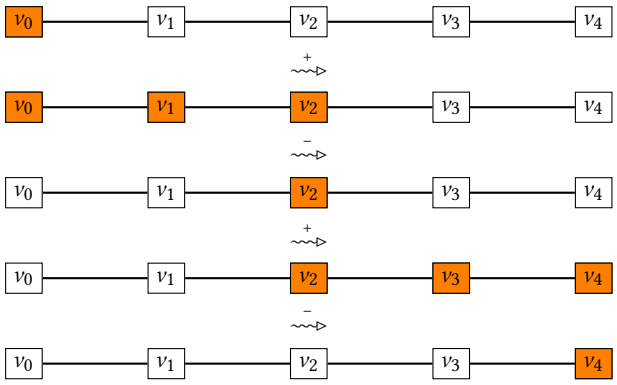

Figure 4: Imagine that the path at the top is a subgraph of the colored graph $G_V[\varphi]$ for some function φ (for instance, the dashed green path in Figure 3). The consecutive subgraphs are obtained by single steps of our transformation. The total transformation also illustrates what we call a chainswap (Definition 5.10).

Chainswapping. *If $(-1)^{|v|} = (-1)^{|v'|}$ (i.e., n is odd) and $v \in \mathrm{SAT}(\varphi)$ and $v' \notin \mathrm{SAT}(\varphi)$ then, defining φ' by $\mathrm{SAT}(\varphi') \stackrel{\text{def}}{=} (\mathrm{SAT}(\varphi)\setminus\{v\})\cup\{v'\}$, we have $\varphi \overset{\pm}{\rightsquigarrow} \overset{*}{\triangleright}\, \varphi'$. In other words, we can uncolor v and color v'. We say that* we have chainswapped v to v'.

Proof. We only explain the chainkilling part, as chainswapping works similarly. Let $n = 2i$. For $0 \leqslant j < i$, do the following: color the nodes v_{2j+1} and v_{2j+2} and then uncolor the nodes v_{2j} and v_{2j+1}. Finally, uncolor the nodes v_{2i} and v_{2i+1}. (Or, alternatively, first color all the nodes on the path and then uncolor everything.) □

Figure 4 illustrates a chainswap. In this section we will only need the chainkilling part of the lemma, but we will use chainswapping in the next section. The second lemma that we need is the following (in this section we only need its instantiation when $\mathrm{e}(\varphi) = 0$), which we call the *fetching lemma*:

Lemma 5.11 (Fetching lemma). *Let φ be such that $\#\varphi \neq |\mathrm{e}(\varphi)|$. Then there exist satisfying valuations v, v' of φ with $(-1)^{|v|} \neq (-1)^{|v'|}$ and a simple path $v = v_0 - \cdots - v_{n+1} = v'$ from v to v' in G_V (hence with n even) such that $v_i \notin \mathrm{SAT}(\varphi)$ for $1 \leqslant i \leqslant n$.*

Proof. Since $\#\varphi \neq |\mathrm{e}(\varphi)|$, there exist satisfying valuations v'', v''' of φ with $(-1)^{|v''|} \neq (-1)^{|v'''|}$. Let $v'' = v''_0 - \cdots - v''_{m+1} = v''''$ be an arbitrary simple path from v'' to v''' in G_V (such a path clearly exists because G_V is a connected graph). Now, let $i \stackrel{\text{def}}{=} \max(0 \leqslant j \leqslant m \mid (-1)^{|v''_j|} = (-1)^{|v''|}$ and $v''_j \models \varphi)$, and then let $i' \stackrel{\text{def}}{=} \min(i < j \leqslant m+1 \mid (-1)^{|v''_j|} = (-1)^{|v'''|}$ and $v''_j \models \varphi)$. Then we can take v to be v''_i and v' to be $v''_{i'}$, which satisfy the desired property. □

Given the right circumstances ($\#\varphi \neq |\mathrm{e}(\varphi)|$), this lemma fetches two valuations v, v' for us to chainkill them. With these tools in place, it is now easy to prove Proposition 5.9:

Proof of Proposition 5.9. Let φ be such that $\mathrm{e}(\varphi) = 0$. We prove the claim by induction on $\#\varphi$. When $\#\varphi = 0$ then we have $\varphi = \bot$, hence a fortiori $\varphi \simeq \bot$. Assume now that $\#\varphi > 0$. Lemma 5.11 fetches two satisfying valuations v, v' of φ, which we then chainkill. Let φ' be the function obtained. We again have $\mathrm{e}(\varphi') = 0$, because the transformation $\overset{\pm}{\rightsquigarrow}$ does not change the Euler characteristic. Moreover, we have $\#\varphi' = \#\varphi - 2$, so that $\varphi' \simeq \bot$ by induction hypothesis, and therefore $\varphi \simeq \bot$ as well. □

Notice how the same valuation could be colored and uncolored multiple times, in case it appears on the paths of different pairs of valuations that are chainkilled. Also, and although this is of absolutely no use to us, we note here the amusing fact that an analogue of Proposition 5.9 works for any connected bipartite colored graph. Moreover, by inspection of the proofs, we also obtain the following:

Corollary 5.12. *It is computable, given a fragmentable Boolean function φ, to find a $\neg$-$\vee$-template T and degenerate Boolean functions $\varphi_0, \ldots, \varphi_n$ such that $T[\varphi_0, \ldots, \varphi_n]$ if deterministic and equivalent to φ.*

This concludes the upper-bound results of this paper. In the next section, we reuse the tools that we have developed so far to carry a detailed analysis of the remaining cases ($\mathrm{e}(\varphi) \neq 0$).

6 EQUIVALENCES AND HARDNESS

The results of the last section imply that, for any two Boolean functions φ, φ' with $\mathrm{e}(\varphi) = \mathrm{e}(\varphi') = 0$, we have $\varphi \simeq \varphi'$. Indeed, by Proposition 5.9 we have $\varphi \simeq \bot$ and $\varphi' \simeq \bot$, implying $\varphi \simeq \varphi'$. In fact, we can show a more general statement about our transformation. We claim:

Proposition 6.1. *We have $\mathrm{e}(\varphi) = \mathrm{e}(\varphi')$ if and only if $\varphi \simeq \varphi'$.*

The proof is similar to that of Proposition 5.9, but is more involved. We first discuss the consequences of Proposition 6.1 in terms of tractability and compilation of the $\mathcal{H}$-queries in Section 6.1, and then prove the proposition in Section 6.2.

6.1 Consequences

Proposition 6.1 allows us to prove:

Theorem 6.2. *Let φ, φ' such that $\mathrm{e}(\varphi) = \mathrm{e}(\varphi')$. Then the following hold:*

- **(a)** *we have $\mathrm{PQE}(Q_\varphi) \equiv_{\mathrm{T}} \mathrm{PQE}(Q_{\varphi'})$;*
- **(b)** *we have $Q_\varphi \in$ d-D(PTIME) iff $Q_{\varphi'} \in$ d-D(PTIME);*
- **(c)** *we have $Q_\varphi \in$ d-D(PSIZE) iff $Q_{\varphi'} \in$ d-D(PSIZE).*

Proof. The proof is similar to that of Proposition 5.8. We sketch the only-if directions of (b) and (c), but (a) works similarly. By Proposition 6.1, we have $\varphi = \varphi_0 \overset{\pm(v_1,\, l_1)}{\rightsquigarrow\triangleright} \cdots \overset{\pm(v_n,\, l_n)}{\rightsquigarrow\triangleright} \varphi_n = \varphi'$ for some $n \in \mathbb{N}$ and valuations and variables v_i, l_i. We show by induction on $0 \leqslant i \leqslant n$ that $Q_{\varphi_i} \in$ d-D(PTIME). The case $i = 0$ holds by assumption. For the inductive case we proceed just as in Proposition 5.8, but this time we use the degeneracy of ψ_i to obtain $Q_{\psi_i} \in$ d-D(PTIME) by Proposition 3.7. The whole construction can clearly be carried out in PTIME data complexity. □

Observe how each item says something different: as far as we can tell, and except when $\mathrm{e}(\varphi) = \mathrm{e}(\varphi') = 0$, in which case all three

items are equivalent since PQE(Q_φ) ∈ d-D(PTIME) by Section 5, there is no obvious implication between any two of them.

Also observe that Theorem 6.2, together with the observations that $\mathrm{e}(\bot) = 0$ and $Q_\bot \in$ d-D(PTIME), imply our main result of Theorem 5.2. We decided to present them separately to make the proofs more modular, so that someone only interested into the upper bound results of the paper can understand the relevant parts more easily (since, as we said, Proposition 6.1 requires more work; see also Figure 6).

We must now discuss a result of [6] concerning FBDDs for the $\mathcal{H}$-queries which, on the surface, seems closely related to the last two items of Theorem 6.2. There, the authors show the following (we paraphrase to fit our notation):

Theorem 6.3 [6, Theorem 3.9]. *Let φ be nondegenerate, and φ' be an arbitrary Boolean function. Then any FBDD F representing the lineage of Q_φ on a database D can be transformed into an FBDD F' representing the lineage of $Q_{\varphi'}$ on the same database, with only a polynomial increase in size.*

First, we note that the proof techniques are seemingly very different: while we did not need to work at the tuple-level, the proof of Theorem 3.9 proceeds by chirugically examining the FBDD F in order to transform it into F'. In fact, one could even argue that the two results are inherently incomparable, for the following reasons:

(1) Theorem 6.2, items (b) and (c), do not hold when applied to FBDDs. Indeed, take $\varphi \stackrel{\text{def}}{=} \bot$ and φ' to be φ_9 from Example 3.3. We have $\mathrm{e}(\bot) = \mathrm{e}(\varphi_9) = 0$. Yet, $Q_\bot \in$ FBDD(PTIME) (clearly), but $Q_{\varphi_9} \notin$ FBDD(PSIZE) by the lower bounds of [6].
(2) Less convincingly, if we could show a version of Theorem 6.3 with d-Ds instead of FBDDs, then *all* $\mathcal{H}$-queries (for $k \geqslant 2$) would be in d-D(PSIZE), including those that are #P-hard by [12]. Indeed, for all $k \in \mathbb{N}_{\geqslant 2}$, there clearly exists a query $Q_\varphi \in \mathcal{H}_k$ such that φ is nondegenerate and $\mathrm{e}(\varphi) = 0$. Then by Theorem 5.2 we have $Q_\varphi \in$ d-D(PTIME), hence any other $\mathcal{H}_k$-query would also be in d-D(PSIZE). This would not be in contradiction with anything, as far as we know, but it would still be quite surprising.

Second, by combining Theorem 6.3 with an exponential lower bound on FBDD representations of one particular nondegenerate $\mathcal{H}$-query $Q_{\varphi_{\text{big-FBDDs}}}$, the authors of [6] obtain an exponential lower bound on FBDDs for all nondegenerate $\mathcal{H}$-queries. In our case we observe a similar phenomenon with Theorem 6.2, where an exponential lower bound on d-D representations for a query $Q_{\varphi_{\text{big-d-Ds}}}$ would also apply to all queries with same Euler characteristic. However, the existence of such a query $Q_{\varphi_{\text{big-d-Ds}}}$ would answer a long-standing open problem in knowledge compilation: that of separating d-Ds (or even d-DNNFs) with DNFs [4, 8, 15]. This is because the lineage of any UCQ on a database can always be computed in PTIME as a DNF.

Nevertheless, we can still use Theorem 6.2 to show a hardness result extending that of [12] for $\mathcal{H}^+$-queries (though this was not the primary goal of this paper). Specifically, we show:

Proposition 6.4. *Let φ be a Boolean function (not necessarily monotone) with $\mathrm{e}(\varphi) \neq 0$ and such that $\min(\mathrm{e}(\varphi) : \varphi$ is monotone$) \leqslant \mathrm{e}(\varphi) \leqslant \max(\mathrm{e}(\varphi) : \varphi$ is monotone$)$. Then PQE(Q_φ) is #P-hard.*

Proof. Let φ be such a Boolean function. We show in Appendix C that there exists a monotone Boolean function φ_{mon} such that $\mathrm{e}(\varphi_{\text{mon}}) = \mathrm{e}(\varphi)$. But then we have $\mathrm{e}(\varphi_{\text{mon}}) \neq 0$, hence by Corollary 3.9 we have that PQE($Q_{\varphi_{\text{mon}}}$) is #P-hard, and by Theorem 6.2, item (a), it holds that PQE(Q_φ) is #P-hard as well. □

We represented by the dashed red rectangle in Figure 1 all the $\mathcal{H}$-queries to which this result applies. This unfortunately does not apply to all the $\mathcal{H}$-queries Q_φ with $\mathrm{e}(\varphi) \neq 0$: for instance, consider the function $\varphi_{\text{max-Euler}}$ whose satisfying valuations are exactly all the valuations of even size. Then we have $\mathrm{e}(\varphi_{\text{max-Euler}}) = 2^k$, and this value is not reachable by a monotone function.

6.2 Proof of Proposition 6.1

As we have already observed, the transformation does not change the Euler characteristic, so we only need to show the "only if" part of Proposition 6.1, i.e., that if $\mathrm{e}(\varphi) = \mathrm{e}(\varphi')$ then $\varphi \simeq \varphi$. Furthermore, since $\mathrm{e}(\varphi) = -\mathrm{e}(\neg\varphi)$, it is enough to show it when $\mathrm{e}(\varphi) \geqslant 0$. We proceed in three steps. The first step is to transform the functions so that they have only satisfying valuations of even size. Formally:

Lemma 6.5. *Let φ a Boolean function with $\mathrm{e}(\varphi) \geqslant 0$. Then there exists $\varphi_{\min} \simeq \varphi$ having only satisfying valuations of even size.*

Proof. We proceed as in the proof of Proposition 5.9: until φ has satisfying valuations of odd size, we use Lemma 5.11 to fetch two valuations ν, ν', and then we chainkill them with Lemma 5.10. Indeed, this always decreases the number of valuations of odd size by one. □

By applying Lemma 6.5 to φ and φ', we are left with two functions $\varphi_{\min}$ and $\varphi'_{\min}$, both having only valuations of even size, and with $\#\varphi_{\min} = \#\varphi'_{\min}$. The second step is then to transform these into what we call a *canonical form*.

Definition 6.6. We say that a Boolean function φ is in canonical form when (1) φ only has satisfying valuations of even size; and (2) there does not exist two valuations ν, ν' of even size with $|\nu'| < |\nu|$ and $\nu \in \mathrm{SAT}(\varphi)$ but $\nu' \notin \mathrm{SAT}(\varphi)$ (in other words, the satisfying valuations of φ are the smallest possible). Let us call such a pair of valuations a *bad pair*.

We then show:

Lemma 6.7. *Let φ be a Boolean function having only satisfying valuations of even size. Then there exists a Boolean function $\breve{\varphi}$ in canonical form such that $\breve{\varphi} \simeq \varphi$.*

Proof. We prove the claim by induction on the number B_φ of bad pairs. If $B_\varphi = 0$ then we are done, so assume $B_\varphi > 0$, and let (ν, ν') be a bad pair of φ. We distinguish two cases:

- we have $\nu' \subseteq \nu$. Then there exists a descending path p from ν to ν' in $\mathbf{G}_V$. By "descending" we mean that the sizes of the valuations along the path are strictly decreasing (by 1, of course). Let ν_i be a valuation on that path that satisfies φ and such that no intermediary valuation on p from ν_i to ν' satisfies φ. Such a ν_i clearly exists, since in the worst case we can take $\nu_i \stackrel{\text{def}}{=} \nu$. Observe that (ν_i, ν') is a bad pair. We now chainswap ν_i to ν'. Let φ' be the function obtained. It is clear that φ' only has satisfying valuations of even size, and

that $B_{\varphi'} = B_\varphi - 1$ (since we have eliminated the bad pair (ν_i, ν') without introducing others), so that the claim holds by induction hypothesis.

- we have $\nu' \not\subseteq \nu$. Then consider another valuation ν'' with $|\nu'| = |\nu''|$ and such that there is a descending path from ν to ν'' in $\mathbf{G}_V$ (i.e., such that $\nu'' \subseteq \nu$: there clearly exist one such ν''). If ν'' does not satisfy φ, simply use the preceding item with ν'' instead of ν', and we are done. If ν'' satisfies φ, then consider a path $\nu'' = \nu_0^= - \nu_1^+ - \nu_1^= - \cdots - \nu_j^+ - \nu_j^= - \cdots \nu_n^+ - \nu_n^= = \nu'$ from ν'' to ν' that alternates between valuations $\nu_j^=$ of size $|\nu''| = |\nu'|$ and valuations ν_j^+ of size $|\nu''| + 1$. Again, such a path clearly exists. For instance if $\nu = \{0,1,2,3\}$ and $\nu' = \{4,5\}$ and $\nu'' = \{2,3\}$, we have the path $\nu'' = \{2,3\} - \{2,3,4\} - \{3,4\} - \{3,4,5\} - \{4,5\} = \nu'$. Now, consider the smallest j such that $\nu_j^= \notin \mathrm{SAT}(\varphi)$ (which is defined since $\nu' = \nu_n^= \notin \mathrm{SAT}(\varphi)$). We now chainswap ν'' to $\nu_j^=$, which does not introduce any bad pair, and preserves the fact that all satisfying valuations are of even size. Then, we can simply use the preceding item with ν'' instead of ν'. □

We now apply Lemma 6.7 to $\varphi_{\min}$ and $\varphi'_{\min}$, thus obtaining $\breve{\varphi}_{\min}$ and $\breve{\varphi}'_{\min}$, both in canonical form, and with $\#\breve{\varphi}_{\min} = \#\breve{\varphi}'_{\min}$ again. The third step is to show that $\breve{\varphi}_{\min} \simeq \breve{\varphi}'_{\min}$. Letting $M \stackrel{\text{def}}{=} \max(|\nu| : \nu \models \breve{\varphi}_{\min})$ and $M' \stackrel{\text{def}}{=} \max(|\nu| : \nu \models \breve{\varphi}'_{\min})$, we necessarily have $M = M'$ because the functions are in canonical form and have the same number of satisfying valuations. If $M = |V|$ then clearly $\breve{\varphi}_{\min} = \breve{\varphi}'_{\min}$, because then $\mathrm{SAT}(\breve{\varphi}_{\min})$ and $\mathrm{SAT}(\breve{\varphi}'_{\min})$ both contain exactly *all* the valuations of even size. Otherwise, we can use valuations of size $M+1$ to transform, say, $\breve{\varphi}_{\min}$ into $\breve{\varphi}'_{\min}$, by iteratively doing the following:

(1) pick a valuation $\nu \in \mathrm{SAT}(\breve{\varphi}_{\min}) \setminus \mathrm{SAT}(\breve{\varphi}'_{\min})$ and a valuation $\nu' \in \mathrm{SAT}(\breve{\varphi}'_{\min}) \setminus \mathrm{SAT}(\breve{\varphi}_{\min})$; these exist and are necessarily of size M;
(2) let $\nu = \nu_0^= - \nu_1^+ - \nu_1^= - \cdots - \nu_j^+ - \nu_j^= - \cdots \nu_n^+ - \nu_n^= = \nu'$ be a simple path from ν to ν' that alternates between valuations $\nu_j^=$ of size $|\nu| = |\nu'| = M$ and valuations ν_j^+ of size $M+1$. Let $0 = j_0 < \ldots < j_m < n$ be all the indices such that $\nu_{j_l}^= \models \breve{\varphi}_{\min}$. We then chainswap $\nu_{j_m}^=$ to ν', and then for $0 \leqslant p < m$ we chainswap $\nu_{j_p}^=$ to $\nu_{j_{p+1}}^=$. The total transformation then simply amounts to uncoloring ν and coloring ν'.

Therefore, we indeed have $\breve{\varphi}_{\min} \simeq \breve{\varphi}'_{\min}$, which implies $\varphi_{\min} \simeq \varphi'_{\min}$ and then $\varphi \simeq \varphi'$, concluding the proof of Proposition 6.1.

7 OPEN PROBLEMS AND FUTURE WORK

In this section we mention some of the numerous questions that this work must leave behind. Some of them are directly related to other open problems in knowledge compilation, while others seem specific to probabilistic databases. We also justify the definition of our transformation.

Hardness and lower bounds. We start with the question of showing hardness for the queries in the dotted gray rectangle of Figure 1:

Open problem 1. *Show #P-hardness of all $\mathcal{H}$-queries Q_φ with* $\mathrm{e}(\varphi) \neq 0$.

This would require adapting the proofs of [12] to UCQs with negations, which appears to be challenging. Note that, thanks to Theorem 6.2 and Lemmas 6.5 and 6.7, it would be enough to show the hardness of the queries in canonical form. In an orthogonal direction, and as we have mentioned already, Theorems 5.2 and 6.2 are begging for a counterpart lower bound:

Open problem 2. *Show superpolynomial lower bounds for d-D representations of the lineages of all $\mathcal{H}$-queries Q_φ with* $\mathrm{e}(\varphi) \neq 0$.

However, a lower bound on d-Ds seems far out of reach with current techniques. Instead, a framework to show lower bounds on d-DNNFs has recently been introduced in [8], using tools from communication complexity. Hence, maybe showing a lower bound on d-DNNFs representations could be an easier target (though this would still answer an important open problem in knowledge compilation).

Using fewer negations. A brief inspection of our proofs indicates that we have spent a generous number of negations to construct the d-Ds. One can readily wonder if that was really necessary. In this regard, an easy observation is that if $\varphi \overset{-}{\rightsquigarrow}{}^* \bot$, then $Q_\varphi \in$ d-DNNF(PTIME), and that if $\varphi \overset{+}{\rightsquigarrow}{}^* \top$ then $\neg Q_\varphi \in$ d-DNNF(PTIME). This was actually the approach taken in [26]. The facts $\varphi \overset{-}{\rightsquigarrow}{}^* \bot$ and $\varphi \overset{+}{\rightsquigarrow}{}^* \top$ can be reformulated using more standard notions of graph theory: we have $\varphi \overset{-}{\rightsquigarrow}{}^* \bot$ iff the subgraph of $\mathbf{G}_V[\varphi]$ induced by the colored nodes has a perfect matching, and $\varphi \overset{+}{\rightsquigarrow}{}^* \top$ iff that induced by the non-colored nodes has a perfect matching. We then conjecture the following:

Conjecture 1 (See also [25, 26]). *If φ is monotone and* $\mathrm{e}(\varphi) = 0$, *then the subgraph of $\mathbf{G}_V[\varphi]$ induced by the colored nodes,* or *that induced by the non-colored nodes, has a perfect matching.*

First, we note that the claim does not hold if we do not impose φ to be monotone. Indeed, consider the function $\varphi_{\text{no-PM}}$ for $k = 4$, whose graph $\mathbf{G}_V[\varphi_{\text{no-PM}}]$ we have depicted in Figure 5. This function has zero Euler characteristic, yet it is easy to see that the subgraph induced by the colored (resp., non-colored) nodes has no perfect matching: the colored node $\{3,4\}$ (resp., non-colored node $\{0,3,4\}$) is isolated. In a sense, this Boolean function justifies the definition of our transformation, and shows that the approach of [26] was doomed to fail for the $\mathcal{H}$-queries that are not UCQs. Second, and much harder to see, the "or" in Conjecture 1 is necessary. Indeed, there exists a monotone function $\varphi_{\text{one-neg}}$ (with $k = 5$) such that the colored nodes have no perfect matching but the non-colored nodes do (and vice versa, by symmetry). We depicted $\mathbf{G}_V[\varphi_{\text{one-neg}}]$ in Appendix D (this is actually the smallest such function). Third, we have checked in [26], using the SAT solver Glucose [5], that this conjecture holds for all monotone Boolean functions with $k \leqslant 5$, amounting to about 20 million non-isomorphic (under permutation of the variables) nondegenerate[9] functions. This conjecture seems also closely related to the open problem in knowledge compilation of determining whether d-DNNFs are closed under negation[10].

[9] Since for degenerate functions the conjecture clearly holds.
[10] Formally: given a d-DNNF D, is there a d-DNNF D' of polynomial size representing $\neg D$?

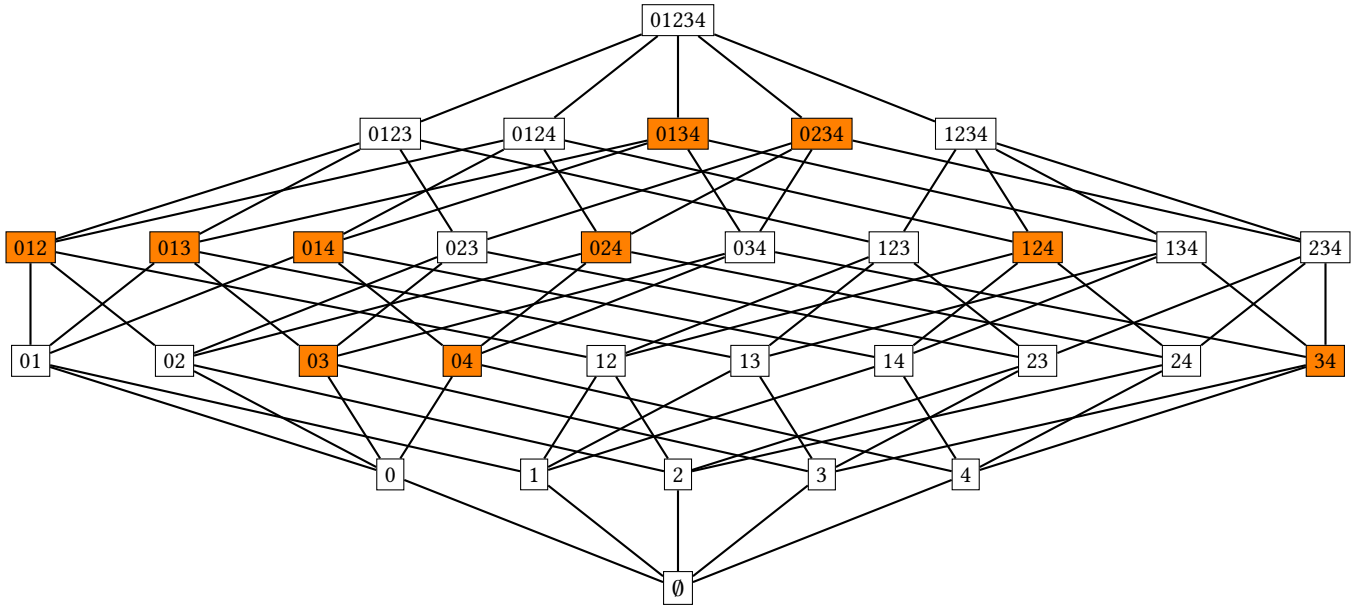

Figure 5: The colored graph $G_V[\varphi_{\text{no-PM}}]$, illustrating that Conjecture 1 must be restricted to monotone functions. For space reasons we write, e.g., 024 instead of $\{0, 2, 4\}$.

Generalize our technique to all UCQs. We now discuss the most important question: extending the techniques we have developed to capture a larger class of queries than the $\mathcal{H}$-queries. To do this, recall that the algorithm of [12] for UCQs recursively alternates between two steps. The first step is to find what is called a *separator variable*, intuitively ensuring that the subqueries obtained are independent. This step can clearly be simulated with d-Ds. The second step is to perform inclusion–exclusion using Möbius's inversion formula. Here the algorithm recurses on the subqueries of the CNF lattice whose Möbius coefficient is not zero, potentially allowing for #P-hard queries to be ignored during the computation. It is this step that seems hard to simulate using a knowledge compilation approach. With this article, we have shown that we can simulate it, in the special case where the bottom term in the CNF lattice is the only hard subquery (and has a zero Möbius coefficient), and where we can recursively construct d-Ds for subqueries that are disjunctions of two connected terms in the poset of the original query (i.e., the poset of Definition 5.6). Although we do not have a concrete example at hand[11], this already seems to define a larger class of queries than the $\mathcal{H}$-queries. However, it is far from obvious how to extend our technique to avoid *other* #P-hard queries than the bottom one. We leave this task for future work.

Open problem 3. *Generalize our techniques to all UCQs, i.e., show that all safe UCQs are in d-D*(PTIME).

8 CONCLUSION

To the best of our knowledge, we provide the first result formally proving that the inclusion–exclusion principle can be simulated using decomposability and determinism only. We see this as yet another indication that knowledge compilation is an effective way to treat query answering on probabilistic databases [3, 4, 24]. Although this new technique applies only to a restricted class of UCQs, the queries considered here seem to already contain the core difficulty of the intensional–extensional conjecture. We think that solving this problem for all UCQs will require solving it for UCQs with negations (more precisely, for Boolean combinations of CQs). This is reminiscent of the algorithm of [12] for UCQs, which, even when applied to a CQ, can introduce UCQs in the computation.

[11] We do not think that such an example would be particularly enlightening.

ACKNOWLEDGMENTS

I acknowledge Dan Olteanu for our initial work on the problem [26]. In particular, this joint work already contains the idea of covering the satisfying valuations of φ with mutually exclusive terms, which we reuse intensively here (see the paragraph "The difference with [26]" in the introduction for more details). I am grateful to Pierre Senellart for careful proofreading of the paper. I thank the Millennium Institute for Foundational Research on Data (IMFD) for funding this research.

REFERENCES

[1] Sheldon B. Akers. 1978. Binary decision diagrams. *IEEE Transactions on computers* 6 (1978), 509–516.
[2] Antoine Amarilli, Pierre Bourhis, Louis Jachiet, and Stefan Mengel. 2017. A circuit-based approach to efficient enumeration. In *ICALP*.
[3] Antoine Amarilli, Pierre Bourhis, and Pierre Senellart. 2015. Provenance circuits for trees and treelike instances. In *International Colloquium on Automata, Languages, and Programming*. Springer, 56–68.
[4] Antoine Amarilli, Florent Capelli, Mikaël Monet, and Pierre Senellart. 2019. Connecting knowledge compilation classes and width parameters. *Theory of Computing Systems* (Jun 2019).
[5] Gilles Audemard and Laurent Simon. 2009. Predicting learnt clauses quality in modern SAT solvers. In *IJCAI*.
[6] Paul Beame, Jerry Li, Sudeepa Roy, and Dan Suciu. 2017. Exact model counting of query expressions: limitations of propositional methods. *ACM Transactions on Database Systems* 42, 1 (2017), 1.

[7] Anders Björner and Gil Kalai. 1988. An extended Euler-Poincaré theorem. *Acta Mathematica* 161, 1 (1988), 279–303.

[8] Simone Bova, Florent Capelli, Stefan Mengel, and Friedrich Slivovsky. 2016. Knowledge compilation meets communication complexity. In *IJCAI*, Vol. 16. 1008–1014.

[9] Simone Bova and Stefan Szeider. 2017. Circuit treewidth, sentential decision, and query compilation. In *PODS*. ACM, 233–246.

[10] Bryant. 1986. Graph-based algorithms for Boolean function manipulation. *IEEE Trans. Comput.* C-35, 8 (1986), 677–691.

[11] Peter Buneman, Sanjeev Khanna, and Tan Wang-Chiew. 2001. Why and where: a characterization of data provenance. In *ICDT*. Springer, 316–330.

[12] Nilesh N. Dalvi and Dan Suciu. 2012. The dichotomy of probabilistic inference for unions of conjunctive queries. *J. ACM* 59, 6 (2012), 30.

[13] Adnan Darwiche. 2001. 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).

[14] Adnan Darwiche. 2009. *Modeling and reasoning with Bayesian networks*. Cambridge University Press.

[15] Adnan Darwiche and Pierre Marquis. 2002. A knowledge compilation map. *JAIR* 17 (2002), 229–264.

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

[17] Norbert Fuhr and Thomas Rölleke. 1997. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Transactions on Information Systems (TOIS)* 15, 1 (1997), 32–66.

[18] Todd J Green and Val Tannen. 2006. Models for incomplete and probabilistic information. In *International Conference on Extending Database Technology*. Springer, 278–296.

[19] Eric Gribkoff, Dan Suciu, and Guy Van den Broeck. 2014. Lifted probabilistic inference: a guide for the database researcher. *IEEE Data Eng. Bull.* 37, 3 (2014), 6–17.

[20] Vladimir Alexander Gurvich. 1977. Repetition-free Boolean functions. *Uspekhi Matematicheskikh Nauk* 32, 1 (1977), 183–184.

[21] Jinbo Huang and Adnan Darwiche. 2005. DPLL with a trace: From SAT to knowledge compilation. In *IJCAI*, Vol. 5. 156–162.

[22] Jinbo Huang and Adnan Darwiche. 2007. The language of search. *J. Artif. Int. Res.* 29, 1 (2007).

[23] Abhay Jha and Dan Suciu. 2012. On the tractability of query compilation and bounded treewidth. In *ICDT*. ACM, 249–261.

[24] Abhay Jha and Dan Suciu. 2013. Knowledge compilation meets database theory: compiling queries to decision diagrams. *Theory of Computing Systems* 52, 3 (2013), 403–440.

[25] Mikaël Monet. 2019. Perfect matching of monotone Boolean function with null Euler characteristic. https://cstheory.stackexchange.com/q/42626/38111

[26] Mikaël Monet and Dan Olteanu. 2018. Towards deterministic decomposable circuits for safe queries. In *AMW*.

[27] Noam Nisan and Mario Szegedy. 1994. On the degree of Boolean functions as real polynomials. *Computational complexity* 4, 4 (1994), 301–313.

[28] Dan Olteanu and Jiewen Huang. 2008. Using OBDDs for efficient query evaluation on probabilistic databases. In *SUM*. Springer, 326–340.

[29] Knot Pipatsrisawat and Adnan Darwiche. 2008. New compilation languages based on structured decomposability. In *AAAI*. AAAI Press, 517–522.

[30] David Poole. 2003. First-order probabilistic inference. In *IJCAI*, Vol. 3. 985–991.

[31] RDBury. 2018. Uniqueness of minimal positive CNFs. https://math.stackexchange.com/a/3039814/378365

[32] Bjarke Hammersholt Roune and Eduardo Sáenz-de Cabezón. 2013. Complexity and algorithms for Euler characteristic of simplicial complexes. *Journal of Symbolic Computation* 50 (2013), 170–196.

[33] Pierre Senellart, Louis Jachiet, Silviu Maniu, and Yann Ramusat. 2018. ProvSQL: Provenance and probability management in PostgreSQL. *Proceedings of the VLDB Endowment* 11, 12 (2018), 2034–2037.

[34] Shubham Sharma, Rahul Gupta, Subhajit Roy, and Kuldeep S Meel. 2018. Knowledge compilation meets uniform sampling. *EPiC Series in Computing* 57 (2018), 620–636.

[35] Richard P. Stanley. 2011. *Enumerative Combinatorics: Volume 1* (2nd ed.).

[36] Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. 2011. *Probabilistic Databases*. Morgan & Claypool.

[37] Moshe Y Vardi. 1982. The complexity of relational query languages. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*. ACM, 137–146.

[38] Ingo Wegener. 2004. BDDs—design, analysis, complexity, and applications. *Discrete Applied Mathematics* 138, 1-2 (2004), 229–251.

[39] Esteban Zimányi. 1997. Query evaluation in probabilistic relational databases. *Theoretical Computer Science* 171, 1-2 (1997), 179–219.

A OVERALL STRUCTURE

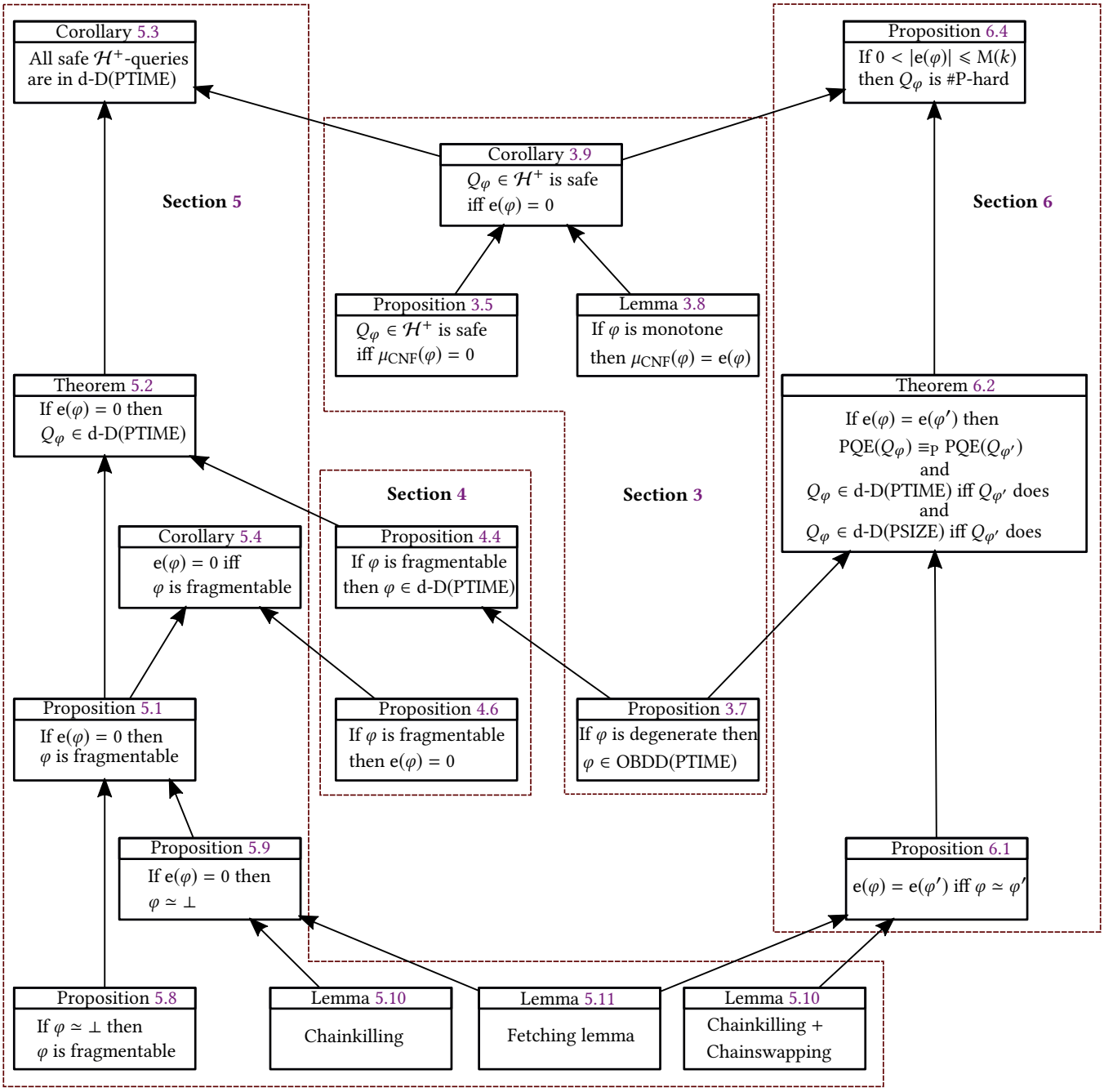

Figure 6: DAG representing the general structure of the proofs. Dashed regions indicate the section in which a result first appears. Remember that $k \in \mathbb{N}_{\geqslant 1}$ and the set of variables $V = \{0, \ldots, k\}$ are fixed.

B PROOFS FOR SECTION 3 (THE $\mathcal{H}$-QUERIES)

B.1 d-Ds for degenerate $\mathcal{H}$-queries

In this section we explain how to build d-Ds in polynomial time for $\mathcal{H}$-queries Q_φ such that φ is degenerate. Remember that we have:

Proposition 3.7 (Implied by Lemma 3.8 of [16]). *If φ (not necessarily monotone) is degenerate, then $Q_\varphi \in$ OBDD(PTIME).*

We only explain how it works for d-Ds, since this is what we need (but we will still use OBDDs in the proof, and we assume here that the reader is familiar with their definition). For a valuation $\nu \subseteq V$, we write φ_ν the Boolean function defined by $\mathrm{SAT}(\varphi_\nu) \stackrel{\text{def}}{=} \{\nu\}$. Since φ is degenerate, let $l \in V$ be a variable upon which it does not depend. We can then write φ as $\varphi = \bigvee_{\nu \models \varphi, l \notin \nu} (\varphi_\nu \vee \varphi_{\nu^{(l)}})$. The outermost $\vee$ being disjoint, we only need to explain how to build d-Ds for the queries $Q_{\varphi_\nu \vee \varphi_{\nu^{(l)}}}$. The query $Q_{\varphi_\nu \vee \varphi_{\nu^{(l)}}}$ can be written as $Q^L \wedge Q^R$, where Q^L is $\bigwedge_{i \in P_L} h_{k,i} \wedge \bigwedge_{i \in N_L} \neg h_{k,i}$ for some subsets $P_L, N_L \subseteq \{0, \ldots, l-1\}$ with $P_L \cap N_L = \emptyset$, and similarly Q^R is $\bigwedge_{i \in P_R} h_{k,i} \wedge \bigwedge_{i \in N_R} \neg h_{k,i}$ for some subsets $P_R, N_R \subseteq \{l+1, \ldots, k\}$ with $P_R \cap N_R = \emptyset$. Since the Boolean queries Q^L and Q^R do not share any relational predicate, by decomposability it is enough to show how to build d-Ds for these two, separately. We only show it for Q^L since Q^R works similarly. To do this, we will build OBDDs O_i for the queries $h_{k,i}$, $i \in \{0, \ldots, l-1\}$, under the same variable order Π_L (recall that the variables of the lineage are the tuples of the database). This will be enough, because we can then use standard techniques [38] to combine these OBDDs and obtain an OBDD for Q^L. Let $\{a_1, \ldots, a_n\}$ be the domain of the database D. The variable order Π_L that we consider is $\Pi_L \stackrel{\text{def}}{=} \Pi_L(1) \ldots \Pi_L(n)$, where $\Pi_L(i) \stackrel{\text{def}}{=} R(a_i), S_1(a_i, a_1), \ldots, S_{l-1}(a_i, a_1), S_1(a_i, a_2), \ldots, S_{l-1}(a_i, a_2), \ldots, S_1(a_i, a_n), \ldots, S_{l-1}(a_i, a_n)$. Under this variable order, it is easy to see that we can build in polynomial time OBDDs O_i for the queries $h_{k,i}$, $i \in \{0, \ldots, l-1\}$, concluding the proof.

B.2 Equivalence Möbius–Euler

In this section we prove Lemma 3.8. We start by recalling its statement (remember that $k \in \mathbb{N}_{\geqslant 1}$ and $V = \{0, \ldots, k\}$ are fixed):

Lemma 3.8. *Let φ be a nondegenerate monotone Boolean function on V. Then we have $\mathsf{e}(\varphi) = \mu_{\mathrm{CNF}}(\hat{0}, \hat{1}) = (-1)^k \mu_{\mathrm{DNF}}(\hat{0}, \hat{1})$, where μ_{CNF} (resp., μ_{DNF}) is the Möbius function of L^φ_{CNF} (resp., L^φ_{DNF}).*

The proof contains three ingredients. The first one is the Möbius inversion formula:

Proposition B.1 (See [35, Proposition 3.7.1]). *Let P be a finite poset, and let $f, g : P \to \mathbb{R}$. Then the following are equivalent:*

- *$g(x) = \sum_{u \leqslant x} f(u)$ for all $x \in P$;*
- *$f(x) = \sum_{u \leqslant x} \mu(u, x) g(u)$ for all $x \in P$.*

The second one is the notion of probability of a Boolean function, which we introduce formally here:

Definition B.2. A *probability assignment* π is a mapping from V to $[0; 1]$. Given a valuation $\nu \subseteq V$ and a probability assignment π, the *probability $\pi(\nu)$ of ν under π* is defined as

$$\pi(\nu) \stackrel{\text{def}}{=} \left(\prod_{x \in \nu} \pi(x) \right) \left(\prod_{x \in V \setminus \nu} (1 - \pi(x)) \right).$$

Given a Boolean function φ and a probability assignment π, *the probability* $\Pr(\varphi, \pi)$ *of φ under π* is then naturally defined as the total probability mass under π of the valuations that satisfy φ, that is $\Pr(\varphi, \pi) \stackrel{\text{def}}{=} \sum_{\nu \models \varphi} \pi(\nu)$.

We will be using specific probability assignments:

Definition B.3. For $t \in [0; 1]$, let π_t denote the probability assignment that maps every variable to t.

The third ingredient is a univariate variant of a *characteristic polynomial* of φ [27], of which we will give three different expressions:

Definition B.4. Let φ be a nondegenerate monotone Boolean function, written as $\bigwedge_{0 \leqslant i \leqslant n} C_i$ in CNF and as $\bigvee_{0 \leqslant i \leqslant m} C'_i$ in DNF. Consider the CNF and DNF lattices of φ, L^φ_{CNF} and L^φ_{DNF}. For $\mathbf{s} \subseteq \{0, \ldots, n\}$ or $\mathbf{s} \subseteq \{0, \ldots, m\}$ we write $d_\mathbf{s} \stackrel{\text{def}}{=} \bigcup_{i \in \mathbf{s}} C_i$ and $d'_\mathbf{s} \stackrel{\text{def}}{=} \bigcup_{i \in \mathbf{s}} C'_i$, as well as $\alpha_\mathbf{s} \stackrel{\text{def}}{=} |d_\mathbf{s}|$ and $\alpha'_\mathbf{s} \stackrel{\text{def}}{=} |d'_\mathbf{s}|$. We define the polynomials P^φ, P^φ_{DNF}, and P^φ_{CNF} of $\mathbb{R}[t]$ as follows:

- $P^\varphi(t) \stackrel{\text{def}}{=} \Pr(\varphi, \pi_t)$;
- $P^\varphi_{\mathrm{CNF}}(t) \stackrel{\text{def}}{=} \displaystyle\sum_{x = d_\mathbf{s} \in L^\varphi_{\mathrm{CNF}}} \mu_{\mathrm{CNF}}(x, \hat{1})(1-t)^{\alpha_\mathbf{s}}$;
- $P^\varphi_{\mathrm{DNF}}(t) \stackrel{\text{def}}{=} 1 - \displaystyle\sum_{x = d'_\mathbf{s} \in L^\varphi_{\mathrm{DNF}}} \mu_{\mathrm{DNF}}(x, \hat{1}) t^{\alpha'_\mathbf{s}}$.

We will show that these polynomials are equal:

Lemma B.5. *Let φ be a nondegenerate monotone Boolean function. Then for all $t \in \mathbb{R}$, we have that $P^\varphi(t) = P^\varphi_{\text{CNF}}(t) = P^\varphi_{\text{DNF}}(t)$.*

This will imply Lemma 3.8: indeed, observe that the coefficient of t^{k+1} is $\sum_{\nu \models \varphi}(-1)^{k+1-|\nu|} = (-1)^{k+1}\sum_{\nu\models\varphi}(-1)^{|\nu|}$ in $P^\varphi(t)$, and is $(-1)^{k+1}\mu_{\text{CNF}}(\hat{0},\hat{1})$ in $P^\varphi_{\text{CNF}}(t)$, and is $-\mu_{\text{DNF}}(\hat{0},\hat{1})$ in $P^\varphi_{\text{DNF}}(t)$. Since $P^\varphi(t)$ and P^φ_{CNF} and P^φ_{DNF} are the same polynomials, these coefficients are equal. So, let us show Lemma B.5:

Proof of Lemma B.5. Clearly, it is enough to show that these polynomials are equal on $[0;1]$. We use Proposition B.1 on L^φ_{CNF} and on L^φ_{DNF} to compute $P^\varphi(t)$. We start with L^φ_{CNF}. Define the functions f and g, from L^φ_{CNF} to $\mathbb{R}$, as follows: let $d_{\mathbf{s}} \in L^\varphi_{\text{CNF}}$, then:

(1) $f(d_{\mathbf{s}}) \stackrel{\text{def}}{=} \Pr\left((\neg\bigvee_{i\in\mathbf{s}} C_i) \wedge \bigwedge_{i\in V\setminus\mathbf{s}} C_i, \pi_t\right)$; in other words the total probability mass of the valuations that do not satisfy any of the (disjunctive) clauses C_i for $i \in \mathbf{s}$ but satisfy all other clauses C_i.

(2) $g(d_{\mathbf{s}}) \stackrel{\text{def}}{=} \Pr(\neg\bigvee_{i\in\mathbf{s}} C_i, \pi_t)$; in other words the total probability mass of the valuations that do not satisfy any of the clauses C_i for $i \in \mathbf{s}$.

We clearly have $g(x) = \sum_{u\leqslant x} f(u)$ for all $x \in L^\varphi_{\text{CNF}}$, hence by Proposition B.1 we have $f(x) = \sum_{u\leqslant x}\mu_{\text{CNF}}(u,x)g(u)$. Moreover, for $x = d_{\mathbf{s}} \in L^\varphi_{\text{CNF}}$, we have that $g(x) = (1-t)^{\alpha_{\mathbf{s}}}$. Now, since $P^\varphi(t) = f(\hat{1})$, we indeed have that $P^\varphi(t) = P^\varphi_{\text{CNF}}(t)$.

Let us now have a look at L^φ_{DNF}. The reasoning is similar, but we include it for completeness. For $x = d'_{\mathbf{s}} \in L^\varphi_{\text{DNF}}$ define:

(1) $f(d'_{\mathbf{s}}) \stackrel{\text{def}}{=} \Pr\left((\bigwedge_{i\in\mathbf{s}} C'_i) \wedge \neg\bigvee_{i\in V\setminus\mathbf{s}} C'_i, \pi_t\right)$; in other words the total probability mass of the valuations that satisfy all the (conjunctive) clauses C'_i for $i \in \mathbf{s}$ but none of the other clauses C'_i.

(2) $g(d'_{\mathbf{s}}) \stackrel{\text{def}}{=} \Pr\left(\bigwedge_{i\in\mathbf{s}} C'_i, \pi_t\right)$; in other words the total probability mass of the valuations that satisfy all the clauses C'_i for $i \in \mathbf{s}$.

This time, we have $g(x) = \sum_{u\leqslant x} f(u)$ and $f(x) = \sum_{u\leqslant x}\mu_{\text{DNF}}(u,x)g(u)$ and $g(x) = t^{\alpha_{\mathbf{s}}}$ for all $x = d'_{\mathbf{s}} \in L^\varphi_{\text{DNF}}$. Combining with $P^\varphi(t) = 1 - f(\hat{1})$ we obtain that $P^\varphi(t) = P^\varphi_{\text{DNF}}(t)$. This finishes the proof. □

C PROOFS FOR SECTION 6 (EQUIVALENCES AND HARDNESS)

In this section we prove the following, which we used in the proof of Proposition 6.4 (remember, once again, that $k \in \mathbb{N}_{\geqslant 1}$ and the set of variables $V = \{0,\dots,k\}$ are fixed):

Lemma C.1. *Let φ be a Boolean function (not necessarily monotone) with $\mathrm{e}(\varphi) \neq 0$ and such that $\min(\mathrm{e}(\varphi) : \varphi$ is monotone$) \leqslant \mathrm{e}(\varphi) \leqslant \max(\mathrm{e}(\varphi) : \varphi$ is monotone$)$. Then there exists a monotone Boolean function φ_{mon} such that $\mathrm{e}(\varphi_{\text{mon}}) = \mathrm{e}(\varphi)$.*

Proof. We will assume wlog that $\mathrm{e}(\varphi) > 0$, as the case of $\mathrm{e}(\varphi) < 0$ is symmetric. The idea of the proof is to start from a monotone Boolean function φ_M that maximizes $\mathrm{e}(\varphi_M)$, and then to modify φ_M (specifically, we will remove satisfying valuations) to show that all the intermediate values between 0 and $\mathrm{e}(\varphi_M)$ for the Euler characteristic (hence, also the value $\mathrm{e}(\varphi)$) are achievable by a monotone Boolean function. More formally, consider the Boolean functions $\psi_0,\dots,\psi_{|\mathrm{SAT}(\varphi_M)|}$, obtained by starting with $\psi_0 \stackrel{\text{def}}{=} \varphi_M$ and iteratively removing exactly one satisfying assignment of maximal size. Then each ψ_i for $0 \leqslant i \leqslant |\mathrm{SAT}(\varphi_M)|$ is a monotone Boolean function, and we have that $\psi_{|\mathrm{SAT}(\varphi_M)|} = \bot$. Moreover, the Euler characteristic of ψ_i, ψ_{i+1} differ only by one. Since we have $\mathrm{e}(\bot) = 0$, and since we know that $0 < \mathrm{e}(\varphi) \leqslant \mathrm{e}(\psi_0)$, there exists a ψ_i such that $\mathrm{e}(\psi_i) = \mathrm{e}(\varphi)$, and we can then take ψ_i to be φ_{mon}. This concludes the proof. □

For completeness (but this is of no use to us), we note here that the monotone Boolean functions φ_M such that $|\mathrm{e}(\varphi_M)|$ is maximized are characterized precisely in [7]. Since [7] is all about simplicial complexes, we will paraphrase their result in terms of Boolean functions here. To do it correctly, we must keep in mind that (1) for our purposes, a "simplicial complex" is the same thing as "the negation of a monotone Boolean function", so we need to reverse the powerset; and (2) in simplicial complexes terminology, the *dimension* of a face $\nu \subseteq V$ is $|\nu| - 1$. We then have:

Theorem C.2 (See [7, Theorem 1.4]). *A monotone Boolean function φ_M such that $|\mathrm{e}(\varphi_M)| = \mathrm{M}(k)$ can only be obtained as follows:*

- *If k is even, then take $\mathrm{SAT}(\varphi_M) \stackrel{\text{def}}{=} \{\nu \subseteq V \mid |\nu| \geqslant n/2+1\}$;*
- *If k is odd, then take $\mathrm{SAT}(\varphi_M) \stackrel{\text{def}}{=} \{\nu \subseteq V \mid |\nu| \geqslant (n-1)/2+1\}$, or take $\mathrm{SAT}(\varphi_M) \stackrel{\text{def}}{=} \{\nu \subseteq V \mid |\nu| \geqslant (n+1)/2+2\}$.*

D PROOFS FOR SECTION 7 (OPEN PROBLEMS AND FUTURE WORK)

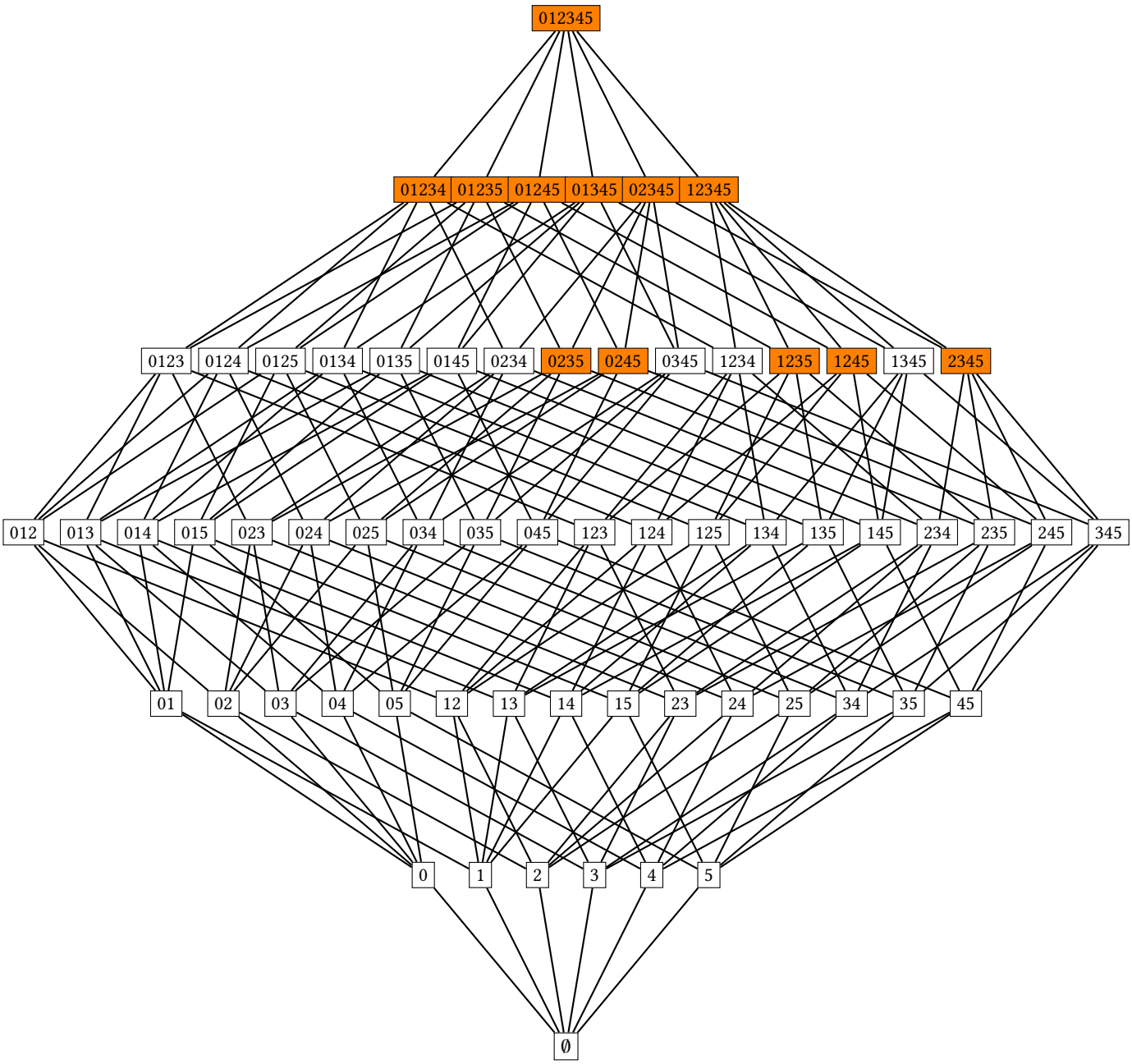

Figure 7: The colored graph $G_V[\varphi_{\text{one-neg}}]$, having $e(\varphi) = 0$, illustrating that in Conjecture 1, the "or" is necessary: the colored nodes have no perfect matching (because 012345 needs to be matched to both 01234 and 01345), but the non-colored ones do (checked with a SAT solver).