PROBLEM:	Shapley(q)
INPUT:	A database $D = D_x \cup D_n$ and an endogenous fact $f \in D_n$.
OUTPUT:	The value Shapley(q, D_n, D_x, f).

- self join free boolean conjunctive queries: for every sjfbcq q
  - q is hierarchical: can be solved in polynomial time
  - q is not hierarchical: shapley is intractable FP hard
- probabilistic query evaluation(PQE) has the same tractability criterion as sjfbcq
- if pqe is tractable for q then so is the problem shapley)q

Probabilistic query evaluation. A *tuple-independent (TID) database* is a pair consisting of a database D and a function π mapping each fact $f \in D$ to a probability value $\pi(f) \in [0, 1]$. The TID (D, π) defines a probability distribution $\Pr_\pi$ on $D' \subseteq D$, where

$$\Pr_\pi(D') \stackrel{\text{def}}{=} \prod_{f \in D'} \pi(f) \times \prod_{f \in D \setminus D'} (1 - \pi(f)) .$$

Given a Boolean query q, *the probability that q is satisfied by* (D, π) is $\Pr(q, (D, \pi)) \stackrel{\text{def}}{=} \sum_{D' \subseteq D \text{ s.t. } q(D')=1} \Pr_\pi(D')$. The *probabilistic query evaluation problem for q*, PQE(q) for short, is then defined as follows.

PROBLEM:	PQE(q)
INPUT:	A tuple-independent database (D, π).
OUTPUT:	The value $\Pr(q, (D, \pi))$.

For two computational problems A and B, we write $A \leq_{\mathrm{T}}^{\mathrm{p}} B$ to assert the existence of a polynomial-time Turing reduction from A to B. We are ready to state the main result of this section.

Proposition 3.1. *For every Boolean query q, we have that* $\mathrm{Shapley}(q) \leq_{\mathrm{T}}^{\mathrm{p}} \mathrm{PQE}(q)$.

Sol1

4 THROUGH ATION

Search with Bing ···

Motivated by the connection to PQE that we have seen in Section 3, we now investigate whether an approach using knowledge compilation can be used for computing Shapley values. Indeed, a common method to compute the probability that a probabilistic database (D, π) satisfies a Boolean query q is to first compute the *lineage* of q on D in a formalism from knowledge compilation, and then to use the good properties of said formalism to compute $\Pr(q, (D, \pi))$ in linear time [18, 26, 33]. Recently, Arenas and

Boolean functions and query lineages. Let X be a finite set of variables. An *assignment ν of X* is a subset $\nu \subseteq X$ of X. We denote by 2^X the set of all assignments of X. A *Boolean function* φ over X is a function $\varphi : 2^X \to \{0, 1\}$. An assignment $\nu \subseteq X$ is *satisfying* if $\varphi(\nu) = 1$. We denote by $\mathsf{SAT}(\varphi) \subseteq 2^X$ the set of all satisfying assignments of φ, and by $\#\mathsf{SAT}(\varphi)$ the size of this set. For $k \in \mathbb{N}$, we define $\mathsf{SAT}_k(\varphi) \stackrel{\text{def}}{=} \mathsf{SAT}(\varphi) \cap \{\nu \subseteq X \mid |\nu| = k\}$, that is, the set of satisfying assignments of φ of Hamming weight k, and let $\#\mathsf{SAT}_k(\varphi)$ be the size of this set.

Let q be a Boolean query and D be a database. The *lineage* $\mathsf{Lin}(q, D)$ is the (unique) Boolean function whose variables are the facts of D, and that maps each sub-database $D' \subseteq D$ to $q(D')$. This definition extends straightforwardly to queries with free variables as follows: if $q(\bar{x})$ is a query with free variables $\bar{x}$ and $\bar{t}$ is a tuple of constants of the appropriate size, then $\mathsf{Lin}(q[\bar{x}/\bar{t}]), D)$ is the *lineage for the tuple $\bar{t}$*.

Proposition 4.4. *Given as input a deterministic and decomposable circuit C representing* $\mathsf{ELin}(q, D_\mathrm{n}, D_\mathrm{x})$ *for a database $D = D_\mathrm{x} \cup D_\mathrm{n}$ and Boolean query q, and an endogenous fact $f \in D_\mathrm{n}$, we can compute in polynomial time (in $|C|$) the value* $\mathsf{Shapley}(q, D_\mathrm{n}, D_\mathrm{x}, f)$.

exactly $D_n \setminus \{f\}$, and moreover that C_1 and C_2 are still deterministic and decomposable. By definition of the endogenous lineage, we can rewrite Equation (2) into the following.

$$\mathsf{Shapley}(q, D_n, D_x, f) = \sum_{k=0}^{|D_n|-1} \frac{k!(|D_n| - k - 1)}{|D_n|} \Big(\#\mathsf{SAT}_k(C_1) - \#\mathsf{SAT}_k(C_2) \Big). \tag{3}$$

In our case, we use ProvSQL as follows: we feed it the database D, query $q(\bar{x})$ and tuple $\bar{t}$, and ProvSQL computes $\mathsf{Lin}(q[\bar{x}/\bar{t}], D)$ as a Boolean circuit, called C in Figure 3. We note here that for SPJU queries, C can be computed in polynomial-time data complexity. We then set to 1 all the exogenous facts to obtain a Boolean circuit C' for $\mathsf{ELin}(q[\bar{x}/\bar{t}], D_x, D_n)$. Then, ideally, we would like to use the knowledge compiler to transform C' into an equivalent d-DNNF, in order to be able to apply Algorithm 1. Unfortunately, every knowledge compiler that we are aware of takes as input Boolean formulas in conjunctive normal form (CNF), and not arbitrary Boolean circuits. To circumvent it, we use the *Tseytin transformation* [35] to transform the circuit C' into a CNF $\varphi = \textsc{Tseytin}(C')$, whose size is linear in that of C'. This CNF φ has the following properties: (1) its variables are the variables of C' plus a set Z of additional variables;

[4]This additional NNF restriction is not important here, but, as far as we know, no knowledge compiler has the more general "deterministic and decomposable circuits" (without NNF) as a target. It is currently unknown whether d-DNNFs and deterministic and decomposable circuits are exponentially separated or not [11, Table 7].