Symmetric Weighted First-Order Model Counting

Paul Beame
University of Washington
beame@cs.washington.edu

Guy Van den Broeck
KU Leuven
guy.vandenbroeck@cs.kuleuven.be

Eric Gribkoff
University of Washington
eagribko@cs.washington.edu

Dan Suciu
University of Washington
suciu@cs.washington.edu

ABSTRACT

The FO Model Counting problem (FOMC) is the following: given a sentence Φ in FO and a number n, compute the number of models of Φ over a domain of size n; the Weighted variant (WFOMC) generalizes the problem by associating a weight to each tuple and defining the weight of a model to be the product of weights of its tuples. In this paper we study the complexity of the symmetric WFOMC, where all tuples of a given relation have the same weight. Our motivation comes from an important application, inference in Knowledge Bases with soft constraints, like Markov Logic Networks, but the problem is also of independent theoretical interest. We study both the data complexity, and the combined complexity of FOMC and WFOMC. For the data complexity we prove the existence of an FO^3 formula for which FOMC is $\#\text{P}_1$-complete, and the existence of a Conjunctive Query for which WFOMC is $\#\text{P}_1$-complete. We also prove that all γ-acyclic queries have polynomial time data complexity. For the combined complexity, we prove that, for every fragment FO^k, $k \geq 2$, the combined complexity of FOMC (or WFOMC) is #P-complete.

1. INTRODUCTION

Probabilistic inference is becoming a central data management problem. Large knowledge bases, such as Yago [19], Nell [2], DeepDive [6], Reverb [11], Microsoft's Probase [43] or Google's Knowledge Vault [8], have millions to billions of uncertain tuples. These systems scan large corpora of text, such as the Web or complete collections of journal articles, and extract automatically billions of structured facts, representing large collections of knowledge. For an illustration, Google's Knowledge Vault [8] contains 1.6B triples of the form (subject, predicate, object), for example, `</m/02mjmr, /people/person/place_of_birth /m/02hrh0_>` where `/m/02mjmr` is the Freebase id for Barack Obama, and `/m/02hrh0_` is the id for Honolulu [8]. The triples are extracted automatically from the Web, and each triple is annotated with a probability p representing the confidence in the extraction.

A central and difficult problem in such systems is probabilistic inference, or, equivalently weighted model counting. The classical FO Model Counting problem (FOMC) is: given a sentence Φ in First-Order Logic (FO) and a number n, compute the number of structures over a domain of size n that satisfy the sentence Φ; in this paper we consider only *labeled structures*, i.e. isomorphic structures are counted as distinct. We denote the number of models by $\text{FOMC}(\Phi, n)$, for example $\text{FOMC}(\forall x \exists y R(x, y), n) = (2^n - 1)^n$.[1] In the Weighted FO Model Counting (WFOMC) variant, one further associates a real number $w(t)$ called *weight* to each tuple t over the domain of size n, and defines the weight of a structure as the product of the weights of all tuples in that structure. The Weighted Model Count $\text{WFOMC}(\Phi, n, \mathbf{w})$ is defined as the sum of the weights of all structures over a domain of size n that satisfy the sentence Φ. Weights map immediately to probabilities, in the following way: if each tuple t is included in the database independently with probability $w(t)/(1 + w(t))$, then the probability that a formula Φ is true is $\Pr(\Phi) = \text{WFOMC}(\Phi, n, \mathbf{w}) / \text{WFOMC}(\text{true}, n, \mathbf{w})$, where $\text{WFOMC}(\text{true}, n, \mathbf{w}) = \prod_t (1 + w(t))$ is the sum of weights of all structures.

In this paper we study the *symmetric* WFMOC problem, where all tuples from the same relation have the same weight, which we denote w_i. For example, a random graph $G(n, p)$ is a symmetric structure, since every edge is present with the same probability p (equivalently: has weight $p/(1-p)$), and FOMC is another special case where all weights are set to 1. The symmetric WFMOC problem occurs naturally in Knowledge Bases with soft constraints, as we illustrate next.

Example 1.1. *A Markov Logic Network (MLN) [7] is a finite set of soft or hard constraints. Each constraint is a pair (w, φ), where φ is a formula, possibly with free variables $\mathbf{x}$, and $w \in [0, \infty]$ is a weight*[2]*. For example,*

PODS'15, May 31–June 4, 2015, Melbourne, Victoria, Australia.
Copyright © 2015 ACM 978-1-4503-2757-2/15/05 ...$15.00.
http://dx.doi.org/10.1145/2745754.2745760.

[1] For a fixed x, there are 2^n assignments to $R(x, y)$, which all satisfy $\exists y R(x, y)$, except the one where all atoms are false. Moreover, the models for the n values of x can be counted independently and multiplied.

[2] In typical MLN systems, users specify the log of the weight rather than the weight. The pair $(1.098, \varphi)$ means that the weight of φ is $w = \exp(1.098) \approx 3$. Using logs simplifies the learning task. We do not address learning and will omit logs; (w, φ) means that φ has weight w.

the soft constraint

$$(3, \mathit{Spouse}(x,y) \wedge \mathit{Female}(x) \Rightarrow \mathit{Male}(y)) \qquad (1)$$

specifies that, typically, a female's spouse is male, and associates the weight $w = 3$ to this constraint. If $w = \infty$ then we call (w, φ) a hard constraint.

The semantics of MLNs naturally extend the Weighted Model Counting setting. Given a finite domain (set of constants), an MLN defines a probability distribution over all structures for that domain (also called possible worlds*). Every structure D has a weight*

$$W(D) = \prod_{(w,\varphi(\mathbf{x}))\in MLN, \mathbf{a}\in D^{|\mathbf{x}|} : w<\infty \wedge D\models\varphi[\mathbf{a}/\mathbf{x}]} w$$

In other words, for each soft constraint (W, φ), and for every tuple of constants $\mathbf{a}$ such that $\varphi(\mathbf{a})$ holds in D, we multiply D's weight by w. For example, given the MLN that consists only of the soft constraint (1), the weight of a world D is 3^N, where N is the number of pairs of constants a, b for which $\mathit{Spouse}(a,b), \mathit{Female}(a) \Rightarrow \mathit{Male}(b)$ holds in D. The weight $W(\Phi)$ of a sentence Φ is defined as the sum of weights of all worlds D that satisfy both Φ and all hard constraints in the MLN; its probability is obtained by normalizing $\Pr_{MLN}(\Phi) = W(\Phi)/W(\mathit{true})$. Notice that the symmetric WFOMC problem corresponds to the special case of an MLN consisting of one soft constraint $(w_i, R_i(\mathbf{x}_i))$ for each relation R_i, where $|\mathbf{x}_i| = \mathit{arity}(R_i)$.

Today's MLN systems (Alchemy [26], Tuffy [30, 44]) use an MCMC algorithm called MC-SAT [31] for probabilistic inference. The theoretical convergence guarantees of MC-SAT require access to a uniform sampler over satisfying assignments to a set of constraints. In practice, MC-SAT implementations rely on SampleSAT [42], which provides no guarantees on the uniformity of solutions. Several complex examples are known in the literature where model counting based on SampleSAT leads to highly inaccurate estimates [16].

A totally different approach to computing $\Pr_{MLN}(\Phi)$ is to reduce it to a symmetric WFOMC [39, 15, 37, 22], and this motivates our current paper. We review here briefly one such reduction, adapting from [22, 37].

Example 1.2. *Given an MLN, replace every soft constraint $(w, \varphi(\mathbf{x}))$ by two new constraints: $(\infty, \forall\mathbf{x}(R(\mathbf{x}) \vee \varphi(\mathbf{x})))$ and $(1/(w-1), R(\mathbf{x}))$. Here R is a new relational symbol with the same arity as the number of free variables in φ, and the constraint $(1/(w-1), R(\mathbf{x}))$ defines R as a relation where all tuples have weight $1/(w-1)$. Therefore, the probability of a formula Φ in the MLN can be computed as a conditional probability over a symmetric, tuple-independent database: $\Pr_{MLN}(\Phi) = \Pr(\Phi|\Gamma)$, where Γ is the conjunction of all hard constraints*[3]*. Note that this reduction to WFOMC is independent of the finite domain under consideration.*

For example, the soft constraint in (1) is translated into the hard constraint:

$$\forall x, y(R(x,y) \vee \neg\mathit{Spouse}(x,y) \vee \neg\mathit{Female}(x) \vee \mathit{Male}(y))$$

and a tuple-independent probabilistic relation R where all tuples have weight $1/(3-1) = 1/2$, or, equivalently, have probability $(1/2)/(1 + (1/2)) = 1/3$.

Thus, our main motivation for studying the symmetric WFOMC is very practical, as symmetric models have been extensively researched in the AI community recently, for inference in MLNs and beyond [24, 39, 29, 41]. Some tasks on MLNs, such as parameter learning [38], naturally exhibit symmetries. For others, such as computing conditional probabilities given a large "evidence" database, the symmetric WFOMC model is applicable when the database has bounded Boolean rank [36]. Moreover, the problem is of independent theoretical interest as we explain below. We study both the data complexity, and the combined complexity. In both settings we assume that the vocabulary $\sigma = (R_1, \ldots, R_m)$ is fixed, and so are the weights $\mathbf{w} = (w_1, \ldots, w_m)$ associated with the relations. In *data complexity*, the formula Φ is fixed, and the only input is the number n representing the size of the domain. In this case WFOMC is a counting problem over a unary alphabet: *given an input 1^n, compute* WFOMC$(\Phi, n, \mathbf{w})$. It is immediate that this problem belongs to the class $\#\mathrm{P}_1$, which is the set of #P problems over a unary input alphabet [34]. In the *combined complexity*, both n and the formula Φ are input.

In this paper we present results on the data complexity and the combined complexity of the FOMC and WFOMC problem, and also some results on the associated decision problem.

Results on Data Complexity

In a surprising result [37] has proven that for FO^2 the data complexity of symmetric WFOMC is in PTIME (reviewed in Appendix C).[4] This is surprising because FO^2 (the class of FO formulas restricted to two logical variables) contains many formulas for which the asymmetric problem was known to be #P-hard. An example is $\Phi = \exists x\exists y(R(x) \wedge S(x,y) \wedge T(y))$, which is #P-hard over asymmetric structures, but the number of models is[5] $2^{2n+n^2} - \sum_{k,m}\binom{n}{k}\binom{n}{m}2^{n^2-km}$, which is a number computable in time polynomial in n.[6] More generally, the symmetric WFOMC problem for Φ is in PTIME.

This begs the question: could it be the case that *every* FO formula is in PTIME? The answer was shown to be negative by Jaeger and Van den Broeck [21, 20], using the following argument. Recall that the *spectrum*, $\mathtt{Spec}(\Phi)$, of a formula Φ is the set of numbers n for which Φ has a model over a domain of size n [9]. Jaeger and Van den Broeck observed that the spectrum membership problem, "*is $n \in \mathtt{Spec}(\Phi)$?*", can be reduced to WFOMC, by checking whether $\mathrm{FOMC}(\Phi, n) > 0$.

[3] The reason why this works is the following: in original MLN, each tuple $\mathbf{a}$ contributes to $W(D)$ a factor of 1 or w, depending on whether $\varphi(\mathbf{a})$ is false or true in D; after the rewriting, the contribution of $\mathbf{a}$ is $1/(w-1)$ when $\varphi(\mathbf{a})$ is false, because in that case $R(\mathbf{a})$ must be true, or $1 + 1/(w-1) = w/(w-1)$ when $\varphi(\mathbf{a})$ is true, because $R(\mathbf{a})$ can be either false or true. The ratio is the same $1 : w = [1/(w-1)] : [w/(w-1)]$.

[4] PTIME data complexity for symmetric WFOMC is called *domain-liftability* in the AI and lifted inference literature [35].

[5] Fix the relations R, T, and let their cardinalities be $|R| = k$ and $|T| = m$. Then the structure does *not* satisfy Φ iff S contains none of the km tuples in $R \times T$, proving the formula.

[6] Tractability of Φ was noted before in, for example [32, 35].

Then, using a result in [23], if $\texttt{ETIME} \neq \texttt{NETIME}$, then there exists a formula Φ for which computing WFOMC is not in polynomial time[7]. However, no hardness results for the symmetric WFOMC were known to date.

What makes the data complexity of the symmetric WFOMC difficult to analyze is the fact that the input is a single number n. Valiant already observed in [34] that such problems are probably not candidates for being #P-complete. Instead, he defined the complexity class $\#P_1$, to be the set of counting problems for NP computations over a single-letter input alphabet. Very few hardness results are known for this class: we are aware only of a graph matching problem that was proven by Valiant, and of a language-theoretic problem by Bertoni and Goldwurm [1].

Our data complexity results are the following. First, we establish the existence of an FO sentence Θ_1 for which the data complexity of the FOMC problem is $\#P_1$-hard; and we also establish the existence of a conjunctive query Υ_1 for which the data complexity of the WFOMC problem is $\#P_1$-hard. Second, we prove that every γ-acyclic conjunctive query without self-joins is in polynomial time, extending the result in [37] from FO^2 to γ-acyclic conjunctive queries. We give now more details about our results, and explain their significance.

The tractability for FO^2 [37] raises a natural question: do other restrictions of FO, like FO^k for $k \geq 3$, also have polynomial data complexity? By carefully analyzing the details of the construction of Θ_1 we prove that it is actually in FO^3. This implies a sharp boundary in the FO^k hierarchy where symmetric WFOMC transitions from tractable to intractable: for k between 2 and 3. The tractability of γ-acyclic queries raises another question: could all conjunctive queries be tractable for symmetric WFOMC? We answer this also in the negative: we prove that there exists a conjunctive query Υ_1 for which the symmetric WFOMC problem is $\#P_1$-hard. It is interesting to note that the decision problem associated to WFOMC, namely *given n, does $n \in \texttt{Spec}(\Phi)$?* is trivial for conjunctive queries, since every conjunctive query has a model over any domain of size $n \geq 1$. Therefore, our $\#P_1$-hardness result for Υ_1 is an instance where the decision problem is easy while the corresponding weighted counting problem is hard. We note that, unlike WFOMC, we do not know the exact complexity of the unweighted, FOMC problem for conjunctive queries.

0-1 Laws. Our data complexity hardness result sheds some interesting light on 0-1 laws. Recall that, if C is a class of finite structures and P is a property over these structures, then $\mu_n(P)$ denotes the fraction of labeled[8] structures in C over a domain of size n that satisfy the property P [27]. A logic has a 0-1 law over the class of structures C, if for any property P expressible in that logic, $\mu(P) \stackrel{\text{def}}{=} \lim_{n\to\infty} \mu_n(P)$ is either 0 or 1. Fagin [13] proved a 0-1 law for First-Order logic and all structures, by using an elegant transfer theorem: there exists a unique, countable structure $\mathbf{R}$, which is characterized by an infinite set of *extension axioms*, τ. He proved that, for every extension axiom, $\lim \mu_n(\tau) = 1$, and this implies $\lim \mu_n(\Phi) = 1$ if Φ is true in $\mathbf{R}$, and $\lim \mu_n(\Phi) = 0$ if Φ is false in $\mathbf{R}$. Compton [3] proved 0-1 laws for several classes of structures C. A natural question to ask is the following: does there exists an elementary proof of the 0-1 laws, by computing a closed formula $\text{FOMC}(\Phi, n)$ for every Φ, then using elementary calculus to prove that that $\mu_n(\Phi)$ converges to 0 or 1? For example, if $\Phi = \forall x \exists y R(x, y)$, then $\text{FOMC}(\Phi, n) = (2^n - 1)^n$ and $\mu_n(\Phi) = (2^n - 1)^n / 2^{n^2} \to 0$; can we repeat this argument for every Φ? On a historical note, Fagin confirms in personal communication that he originally tried to prove the 0-1 law by trying to find such a closed formula, which failed as an approach. Our $\#P_1$-result for FO proves that no such elementary proof is possible, because no closed formula for $\text{FOMC}(\Phi, n)$ can be computed in general (unless $\#P_1$ is in PTIME).

Results on the Combined Complexity

Our main result on the combined complexity is the following. We show that, for any $k \geq 2$, the combined complexity of FOMC for FO^k is #P-complete; membership is a standard application of Scott's reduction, while hardness is by reduction from the model counting problem for Boolean formulas. Recall that the vocabulary σ is always assumed to be fixed: if it were allowed to be part of the input, then every Boolean formula is a special case of an FO^0 formula, by creating a new relational symbol of arity zero for each Boolean variable, and all hardness results for Boolean formulas carry over immediately to FO^0.

The Associated Decision Problem

We also discuss and present some new results on the decision problem associated with (W)FOMC: "*given Φ, n, does Φ have a model over a domain of size n?*". The data complexity variant is, of course, the spectrum membership problem, which has been completely solved by Jones and Selman [23], by proving that the class of spectra coincides with $\texttt{NETIME}$, that is, $\{\texttt{Spec}(\Phi) \mid \Phi \in \mathit{FO}\} = \texttt{NETIME}$. Their result assumes that the input n is represented in binary, thus the input size is $\log n$. In this paper we are interested in the unary representation of n, as 1^n, which is also called the *tally notation*, in which case case $\texttt{NETIME}$ naturally identifies with NP_1. Fagin proved that, in the tally notation, $\{\texttt{Spec}(\Phi) \mid \Phi \in \mathit{FO}\} = NP_1$ [12, Theorem 6, Part 2].

For the decision problem, our result is for the combined complexity: given both Φ, n, does $n \in \texttt{Spec}(\Phi)$? We prove that this problem is NP-complete for FO^2, and PSPACE-complete for FO. The first of these results has an interesting connection to the finite satisfiability problem for FO^2, which we discuss here. Recall the classical satisfiability problem in finite model theory: "*given a formula Φ does it have a finite model?*", which is equivalent to checking $\texttt{Spec}(\Phi) \neq \emptyset$. Grädel, Kolaitis and Vardi [17] have proven the following two

[7] Recall that $\texttt{ETIME} = \bigcup_{c\geq 1} \texttt{DTIME}(2^{cn})$ and $\texttt{NETIME} = \bigcup_{c\geq 1} \texttt{NTIME}(2^{cn})$, and are not to be confused with the more familiar classes EXPTIME and NEXPTIME, which are $\bigcup_{c\geq 1} \texttt{DTIME}(2^{n^c})$ and $\bigcup_{c\geq 1} \texttt{NTIME}(2^{n^c})$ respectively.

[8] The attribute *labeled* means that isomorphic structures are counted as distinct; 0-1 laws for unlabeled structures also exist. In this paper, we discuss labeled structures only.

results for FO^2: if a formula Φ is satisfiable then it has a finite model of size at most exponential in the size of the sentence Φ, and deciding whether Φ is satisfiable is NEXPTIME-complete in the size of Φ. These two results already prove that the combined complexity for deciding $n \in \texttt{Spec}(\Phi)$ cannot be in polynomial time: otherwise, we could check satisfiability in EXPTIME by iterating n from 1 to exponential in the size of Φ, and checking $n \in \texttt{Spec}(\Phi)$. Our result settles the combined complexity, proving that it is NP-complete.

The paper is organized as follows: we introduce the basic definitions in Section 2, present our results for the data complexity of the FOMC and WFOMC problems in Section 3, present all results on the combined complexity in Section 4, then conclude in Section 5.

2. BACKGROUND

We review here briefly the main concepts, some already introduced in Section 1.

Weighted Model Counting (WMC). The *Model Counting* problem is: given a Boolean formula F, compute the number of satisfying assignments $\#F$. In *Weighted Model Counting* we are given two real functions $w, \bar{w} : \text{Vars}(F) \rightarrow \mathbf{R}$ associating two weights $w(X), \bar{w}(X)$ to each variable in $\text{Vars}(F) = \{X_1, \ldots, X_n\}$. The weighted model count $\text{WMC}(F, w, \bar{w})$ is defined as:

$$\text{WMC}(F, w, \bar{w}) \overset{\text{def}}{=} \sum_{\theta:\theta(F)=1} W(\theta) \tag{2}$$

where, $\forall\theta : \text{Vars}(F) \rightarrow \{0, 1\}$:

$$W(\theta) \overset{\text{def}}{=} \prod_{i:\theta(X_i)=0} \bar{w}(X_i) \times \prod_{i:\theta(X_i)=1} w(X_i) \tag{3}$$

The model count is a special case $\#F = \text{WMC}(F, 1, 1)$.

The standard definition of WMC in the literature does not mention $\bar{w}$, instead sets $\bar{w} = 1$; as we will see, our extension is non-essential. When $\bar{w} = 1$, then we simply drop $\bar{w}$ from the notation, and write $\text{WMC}(F, w)$ instead of $\text{WMC}(F, w, 1)$. In the *probability computation problem*, each variable X_i is set to true with some known probability $p(X_i) \in [0, 1]$, and we want to compute $\Pr(F, p) \overset{\text{def}}{=} \text{WMC}(F, p, 1 - p)$, the probability that F is true. All these variations are equivalent, because of the following identities:

$$\text{WMC}(F, w, \bar{w}) = \text{WMC}(F, w/\bar{w}, 1) \times \prod_i \bar{w}(X_i) \tag{4}$$

$$\text{WMC}(F, w, \bar{w}) = \Pr(F, w/(w + \bar{w})) \times \prod_i (w(X_i) + \bar{w}(X_i))$$

Throughout the paper we write 1 for the constant function with value 1, and $w_1 + w_2$, and w_1/w_2 for functions $X \mapsto w_1(X) + w_2(X)$ and $X \mapsto w_1(X)/w_2(X)$ resp.

Weighted First-Order Model Counting (WFOMC). Consider FO formulas over a fixed relational vocabulary $\sigma = (R_1, \ldots, R_m)$ and equality $=$. Given a domain size n, denote $\text{Tup}(n)$ the set of ground tuples (i.e., ground atoms without equality) over the domain, thus $|\text{Tup}(n)| = \sum_i n^{\text{arity}(R_i)}$. The *lineage* of an FO sentence Φ refers to a Boolean function $F_{\Phi,n}$ over $\text{Tup}(n)$ (a ground FO sentence), as well as the corresponding Boolean function over propositional variables referring to ground tuples (a propositional sentence). It is defined inductively by $F_{t,n} = t$ for ground tuples t, $F_{\neg\Phi,n} = \neg F_{\Phi,n}$, $F_{(\Phi_1 \text{ op } \Phi_2),n} = F_{\Phi_1,n} \text{ op } F_{\Phi_2,n}$ for $\text{op} \in \{\wedge, \vee\}$, $F_{a=b,n} = \texttt{false}$, $F_{a=a,n} = \texttt{true}$ and $F_{\exists x\Phi,n} = \bigvee_{a\in[n]} F_{\Phi[a/x],n}$, $F_{\forall x\Phi,n} = \bigwedge_{a\in[n]} F_{\Phi[a/x],n}$. For any fixed sentence Φ, the size of its lineage is polynomial in n. Given a domain size n and weight functions $w, \bar{w} : \text{Tup}(n) \rightarrow \mathbf{R}$, the Weighted First-Order Model Count of Φ is $\text{WFOMC}(\Phi, n, w, \bar{w}) \overset{\text{def}}{=} \text{WMC}(F_{\Phi,n}, w, \bar{w})$.

Symmetric WFOMC. In the *symmetric* WFOMC, the weight of a tuple depends only on the relation name and not on the domain constants. We call a *weighted vocabulary* a triple $(\sigma, \mathbf{w}, \bar{\mathbf{w}})$ where $\sigma = (R_1, \ldots, R_m)$ is a relational vocabulary and $\mathbf{w} = (w_1, \ldots, w_m)$, $\bar{\mathbf{w}} = (\bar{w}_1, \ldots, \bar{w}_m)$ represent the weights (real numbers) for the relational symbols. For any domain size n, we extend these weights to $\text{Tup}(n)$ by setting $w'(R_i(a_1, \ldots, a_k)) = w_i$ and $\bar{w}'(R_i(a_1, \ldots, a_k)) = \bar{w}_i$, and we define $\text{WFOMC}(\Phi, n, \mathbf{w}, \bar{\mathbf{w}}) \overset{\text{def}}{=} \text{WFOMC}(\Phi, n, w', \bar{w}')$. Throughout this paper we assume that WFOMC refers to the symmetric variant, unless otherwise stated.

For a simple illustration, consider the sentence $\varphi = \exists y S(y)$. Then $\text{WFOMC}(\varphi, n, w_S, \bar{w}_S) = (\bar{w}_S + w_S)^n - (\bar{w}_S)^n$, because the sum of the weights of all possible worlds is $(\bar{w}_S + w_S)^n$, and we have to subtract the weight of the world where $S = \emptyset$. For another example, consider $\Phi = \forall x \exists y R(x, y)$. The reader may check that $\text{WFOMC}(\Phi, n, w_R, \bar{w}_R) = ((w_R + \bar{w}_R)^n - \bar{w}_R^n)^n$. In particular, over a domain of size n, the formula Φ has $(2^n - 1)^n$ models (by setting $w_R = \bar{w}_R = 1$).

Data Complexity and Combined Complexity. We consider the weighted vocabulary $(\sigma, \mathbf{w}, \bar{\mathbf{w}})$ fixed. In the *data complexity*, we fix Φ and study the complexity of the problem: *given n, compute* $\text{WFOMC}(\Phi, n, \mathbf{w}, \bar{\mathbf{w}})$. In the combined complexity, we study the complexity of the problem: *given Φ, n, compute* $\text{WFOMC}(\Phi, n, \mathbf{w}, \bar{\mathbf{w}})$. All our upper bounds continue to hold if the weights $\mathbf{w}, \bar{\mathbf{w}}$ are part of the input. We also consider the data- and combined-complexity of the associated decision problem (where we ignore the weights) *given n, does Φ have a model over a domain of size n?*

Weights and Probabilities. While in practical applications the weights are positive real numbers, and the probabilities are numbers in $[0, 1]$, in this paper we impose no restrictions on the values of the weights and probabilities. The definition (2) of $\text{WMC}(F, w)$ applies equally well to negative weights, and, in fact, to any semiring structure for the weights [25]. There is, in fact, at least one application of negative probabilities [22], namely the particular reduction from MLNs to WFOMC described in Example 1.2: a newly introduced relation has weight $1/(w - 1)$, which is negative when

Problem	Weights for R, S, and T tuples	Solution for $\Phi = \forall x \forall y(R(x) \vee S(x,y) \vee T(y))$
Symmetric FOMC	$w = \bar{w} = 1$	$\text{FOMC}(\Phi, n) = \sum_{k,m=0,n} \binom{n}{k}\binom{n}{m} 2^{n^2-km}$
Symmetric WFOMC	w_R, w_S, w_T, $\bar{w}_R, \bar{w}_S, \bar{w}_T$	$\text{WFOMC}(\Phi, n, \mathbf{w}, \bar{\mathbf{w}}) = \sum_{k,m=0,n} \binom{n}{k}\binom{n}{m} W_{k,m}$ where $W_{k,m} = w_R^{n-k} \cdot \bar{w}_R^k \cdot w_S^{km} \cdot (w_S + \bar{w}_S)^{n^2-km} \cdot w_T^{n-m} \cdot \bar{w}_T^m$
Asymmetric WFOMC	$w(R(i), w(S(i,j)), w(T(j))$ $\bar{w}(R(i)), \bar{w}(S(i,j)), \bar{w}(T(j))$ depend on i, j	#P-hard for Φ [4]

Table 1: Three variants of WFOMC, of increasing generality, illustrated on the sentence $\Phi = \forall x \forall y(R(x) \vee S(x,y) \vee T(y))$. This paper discusses the symmetric cases only.

$w < 1$. Then, the associated probability $p = w/(1+w)$ belongs to $(-\infty, 0) \cup (1, \infty)$.

As a final comment on negative weights, we note that the complexity of the symmetric WFOMC problem is the same for arbitrary weights as for positive weights. Indeed, the expression $\text{WFOMC}(\Phi, n, \mathbf{w})$ is a multivariate polynomial in m variables $w_1, \ldots, w_m$, where each variable has degree n. The polynomial has $(n+1)^m = n^{O(1)}$ real coefficients. Given access to an oracle computing this polynomial for arbitrary positive values for w, we can compute in polynomial time all $n^{O(1)}$ coefficients with as many calls to the oracle; once we know the coefficients we can compute the polynomial at any values $w_1, \ldots, w_m$, positive or negative.

For all upper bounds in this paper we assume that the weights $w, \bar{w}$, or probabilities p, are given as rational numbers represented as fractions of two integers of n bits each. We assume w.l.o.g. that all fractions have the same denominator: this can be enforced by replacing the denominators by their least common multiplier, at the cost of increasing the number of bits of all integers to at most n^2. It follows that the weight of a world $W(\theta)$ (Eq.(3)) and $\text{WMC}(F, w, \bar{w})$ can be represented as ratios of two integers, each with $n^{O(1)}$ bits.

Summary. Table 1 summarizes the taxonomy and illustrates the various weighted model counting problems considered in this paper. Throughout the rest of the paper, FOMC and WFOMC refer to the symmetric variant, unless otherwise mentioned.

3. DATA COMPLEXITY

Recall that the language FO^k consists of FO formulas with at most k distinct logical variables.

3.1 Lower Bounds

Our first lower bound is for an FO^3 sentence:

Theorem 3.1. *There exists an FO^3 sentence, denoted Θ_1, s.t. the FOMC problem for Θ_1 is $\#P_1$-complete.*

Van den Broeck et al. [37] have shown that the Symmetric WFOMC problem for every FO^2 formula has polynomial time data complexity (the proof is reviewed in Appendix C); Theorem 3.1 shows that, unless $\#\text{P}_1$ is in PTIME, the result cannot extend to FO^k for $k > 2$.

Our second lower bound is for a conjunctive query, or, dually, a positive clause without equality. Recall that a *clause* is a universally quantified disjunction of literals, for example $\forall x \forall y(R(x) \vee \neg S(x,y))$. A *positive clause* is a clause where all relational atoms are positive. A *conjunctive query* (CQ) is an existentially quantified conjunction of positive literals, e.g. $\exists x \exists y(R(x) \wedge S(x,y))$. Positive clauses without the equality predicate are the duals of CQs, and therefore the WFOMC problem is essentially the same for positive clauses without equality as for CQs. Note that the dual of a clause with the equality predicate is a CQ with $\neq$, *e.g.* the dual of $\forall x \forall y(R(x,y) \vee x = y)$ is $\exists x \exists y(R(x,y) \wedge x \neq y)$.

Corollary 3.2. *There exists a positive clause Ξ_1 without equality s.t. the Symmetric WFOMC problem for Ξ_1 is $\#P_1$-hard. Dually, there exists a CQ Υ_1 s.t. the Symmetric WFOMC problem for Υ_1 is $\#P_1$-hard.*

Corollary 3.2 shows that the tractability result for γ-acyclic conjunctive queries (discussed below in Theorem 3.6) cannot be extended to all CQs. The proof of the Corollary follows easily from three lemmas, which are of independent interest, and which we present here; the proofs of the lemmas are in the appendix. We say that a vocabulary σ' *extends* σ if $\sigma \subseteq \sigma'$, and that a weighted vocabulary $(\sigma', \mathbf{w}', \bar{\mathbf{w}}')$ *extends* $(\sigma, \mathbf{w}, \bar{\mathbf{w}})$ if $\sigma \subseteq \sigma'$ and the tuples $\mathbf{w}', \bar{\mathbf{w}}'$ extend $\mathbf{w}, \bar{\mathbf{w}}$.

Lemma 3.3. *Let $(\sigma, \mathbf{w}, \bar{\mathbf{w}})$ be a weighted vocabulary and Φ an FO sentence over σ. There exists an extended weighted vocabulary $(\sigma', \mathbf{w}', \bar{\mathbf{w}}')$ and sentence Φ' over σ', such that Φ' is in prenex-normal form with a quantifier prefix $\forall^*$, and* $\text{WFOMC}(\Phi, n, \mathbf{w}, \bar{\mathbf{w}}) = \text{WFOMC}(\Phi', n, \mathbf{w}', \bar{\mathbf{w}}')$ *for all n.*

This lemma was proven by [37], and says that all existential quantifiers can be eliminated. The main idea is to replace a sentence of the form $\forall \mathbf{x} \exists y \, \psi(\mathbf{x}, y)$ by $\forall \mathbf{x} \forall y \, (\neg\psi(\mathbf{x}, y) \vee A(\mathbf{x}))$, where A is a new relational symbol of arity $|\mathbf{x}|$ and with weights $w_A = 1, \bar{w}_A = -1$. For every value $\mathbf{x} = \mathbf{v}$, in a world where $\exists y \, \psi(\mathbf{v}, y)$ holds, $A(\mathbf{v})$ holds too and the new symbol contributes a factor $+1$ to the weight; in a world where $\exists y \, \psi(\mathbf{v}, y)$ does not hold, then $A(\mathbf{v})$ may be true or false, and the weights of the two worlds cancel each other out.

Note that the lemma tells us nothing about the model count of Φ and Φ', since in Φ' we are forced to set some negative weights. If we had $\text{FOMC}(\Phi, n) = \text{FOMC}(\Phi', n)$, then we could reduce the satisfiability problem for an arbitrary FO sentence Φ to that for a sentence with a $\forall^*$ quantifier prefix, which is impossible, since the former is undecidable while the latter is decidable.

The next lemma, also following the proof in [37], says that all negations can be eliminated.

Lemma 3.4. *Let $(\sigma, \mathbf{w}, \bar{\mathbf{w}})$ be a weighted vocabulary and Φ a sentence over σ in prenex-normal form with quantifier prefix $\forall^*$. Then there exists an extended weighted vocabulary $(\sigma', \mathbf{w}', \bar{\mathbf{w}}')$ and a positive FO sentence Φ' over σ', also in prenex-normal form with quantifier prefix $\forall^*$, s.t.* $\mathrm{WFOMC}(\Phi, n, \mathbf{w}, \bar{\mathbf{w}}) = \mathrm{WFOMC}(\Phi', n, \mathbf{w}', \bar{\mathbf{w}}')$ *for all n.*

The idea is to create two new relational symbols A, B for every negated subformula $\neg\psi(\mathbf{x})$, replace the formula by $A(\mathbf{x})$, and add the sentence $\forall\mathbf{x}(\psi(\mathbf{x}) \vee A(\mathbf{x})) \wedge (A(\mathbf{x}) \vee B(\mathbf{x})) \wedge (\psi(\mathbf{x}) \vee B(\mathbf{x}))$. By setting the weights $w_A = \bar{w}_A = w_B = 1$, $\bar{w}_B = -1$ we ensure that, for every constant $\mathbf{x} = \mathbf{v}$, either $\neg\psi(\mathbf{v}) \equiv A(\mathbf{v})$, in which case $B(\mathbf{v})$ is forced to be true and the two new symbols contribute a factor $+1$ to the weight, or $\psi(\mathbf{v}) \equiv A(\mathbf{v}) \equiv \mathtt{true}$, in which case $B(\mathbf{v})$ can be either true or false, and the weights cancel out.

Finally, we remove the $=$ predicate.

Lemma 3.5. *Let $(\sigma, \mathbf{w}, \bar{\mathbf{w}})$ be a weighted vocabulary and Φ a sentence over σ. Then there exists an extended vocabulary σ' and sentence Φ' without the equality predicate $=$, such that, for all n,* $\mathrm{WFOMC}(\Phi, n, \mathbf{w}, \bar{\mathbf{w}})$ *can be computed in polynomial time using $n + 1$ calls to an oracle for* $\mathrm{WFOMC}(\Phi', n, \mathbf{w}', \bar{\mathbf{w}}')$*, where $(\sigma', \mathbf{w}', \bar{\mathbf{w}}')$ is an extension of $(\sigma, \mathbf{w}, \bar{\mathbf{w}})$.*

The idea is to introduce a new relational symbol E, replace every atom $x = y$ with $E(x, y)$, and add the sentence $\forall x\ E(x, x)$. Let $\mathbf{w}', \bar{\mathbf{w}}'$ be the extension of $\mathbf{w}, \bar{\mathbf{w}}$ with $w'_E = z$, $\bar{w}'_E = 1$. Then $\mathrm{WFOMC}(\Phi', n, \mathbf{w}', \bar{\mathbf{w}}')$ is a polynomial of degree n^2 in z where each monomial has degree $\geq n$ in z, because the hard constraint $\forall x\ E(x, x)$ forces $|E| \geq n$. Moreover, the coefficient of z^n is precisely $\mathrm{WFOMC}(\Phi, n, \mathbf{w}, \bar{\mathbf{w}})$, because that corresponds to the worlds where $|E| = n$, hence it coincides with $=$. We compute this coefficient using $n + 1$ calls to an oracle for $\mathrm{WFOMC}(\Phi', n, \mathbf{w}', \bar{\mathbf{w}}')$.

Now we give the proof of Corollary 3.2. Starting with the $\#\mathrm{P}_1$-complete sentence Θ_1, we apply the three lemmas and obtain a positive sentence Φ, with quantifier prefix $\forall^*$ and without the equality predicate, that is $\#\mathrm{P}_1$-hard. We write it as a conjunction of clauses, $\Phi = C_1 \wedge C_2 \wedge \cdots \wedge C_k$ (recall that a clause is universally quantified), and then apply the inclusion-exclusion formula: $\Pr(\Phi) = \sum_{s \subseteq [k], s \neq \emptyset} (-1)^{|s|+1} \Pr(\bigvee_{i \in s} C_i)$. Since any disjunction of clauses is equivalent to a single clause, we have reduced the computation problem $\Pr(\Phi)$ to computing the probabilities of $2^k - 1$ clauses. By duality, this reduces to computing the probabilities of $2^k - 1$ conjunctive queries, $\Pr(Q_1), \Pr(Q_2), \ldots, \Pr(Q_{2^k-1})$. We can reduce this problem to that of computing the probability of a single conjunctive query Υ_1, by the following argument. Create $2^k - 1$ copies of the relational symbols in the FO vocabulary, and take the conjunction of all queries, where each query uses a fresh copy of the vocabulary. Then $\Pr(Q_1 \wedge \cdots \wedge Q_{2^k-1}) = \Pr(Q_1) \cdots \Pr(Q_{2^k-1})$, because now every two distinct queries Q_i, Q_j have distinct relational symbols. Using an oracle to compute the probability of $\Upsilon_1 \overset{\text{def}}{=} \bigwedge_i Q_i$, we can compute any $\Pr(Q_i)$ by setting to 1 the probabilities of all relations occurring in Q_j, for $j \neq i$: in other words, the only possible world for a relation R in Q_j is one where R is the cartesian product of the domain; assuming $n \geq 1$, Q_j is true, $\Pr(Q_j) = 1$, and hence $\Pr(\Upsilon_1) = \Pr(Q_i)$. We repeat this for every i and compute $\Pr(Q_1), \ldots, \Pr(Q_{2^k-1})$. This proves that the CQ Υ_1 is $\#\mathrm{P}_1$-hard. Its dual, Ξ_1, is a $\#\mathrm{P}_1$-hard positive clause without equality. This proves Corollary 3.2.

3.2 Upper Bounds

A CQ is *without self-joins* if all atoms refer to distinct relational symbols. It is standard to associate a hypergraph with CQs, where the variables are nodes, and the atoms are hyper-edges. We define a γ-acyclic conjunctive query to be a conjunctive query w/o self-joins whose associated hypergraph is γ-acyclic. We prove:

Theorem 3.6. *The data complexity of Symmetric WFOMC for γ-acyclic CQs is in PTIME.*

Fagin's definition of γ-acyclic hypergraphs [14] is reviewed in the proof of Theorem 3.6.

An open problem is to characterize the conjunctive queries without self-joins that are in polynomial time. While no such query has yet been proven to be hard (Υ_1 in Corollary 3.2 has self-joins), it is widely believed that, for any $k \geq 3$, the symmetric WFOMC problem for a *typed cycle* of length k, $C_k = \exists x_1 \cdots x_k(R_1(x_1, x_2), R_2(x_2, x_3), \ldots, R_k(x_k, x_1))$, is hard. We discuss here several insights into finding the tractability border for conjunctive queries, summarized in Figure 1.

This boundary does not lie at γ-acyclicity: the query $c_\gamma = R(x, z), S(x, y, z), T(y, z)$ is γ-cyclic (with cycle $RxSyTzR$; see Fagin [14]), yet it still has PTIME data complexity. The key observation is that γ-cycles allow the last variable z to appear in all predicates, turning it into a *separator variable* [5], hence $\Pr(Q) = \prod_{a \in [n]} \Pr(Q[a/z])$, which is $[\Pr(Q[a/z])]^n$ by symmetry; $Q[a/z]$ is isomorphic to the query in Table 1 and can be computed in polynomial time. A weaker notion of acyclicity, called *jtdb* (for join tree with disjoint branches), can be found in [10]. It also does not characterize the tractability boundary: *jtdb* contains the γ-cyclic query above, but it does not contain the PTIME query $c_{jtdb} = R(x, y, z, u), S(x, y), T(x, z), V(x, u)$.

Fagin [14] defines two increasingly weaker notions of acyclicity: β- and α-acyclic. α-Acyclic queries are as hard as any conjunctive query without self-joins. Indeed, if $Q = \exists\mathbf{x}\varphi(\mathbf{x})$ is a conjunctive query w/o self-joins, then the query $Q' = \exists\mathbf{x}(A(\mathbf{x}) \wedge \varphi(\mathbf{x}))$ is α-acyclic, where A is a new relational symbol, containing all variables of Q. By setting the probability of A to 1, we have $\Pr(Q) = \Pr(Q')$. Thus, if all α-acyclic queries have PTIME data complexity, then all conjunctive queries w/o self-joins have PTIME data complexity.

For all we know, β-acyclic queries could well coincide with the class of tractable conjunctive queries w/o self joins. We present here some evidence that all β-cyclic queries are hard, by reduction from typed cycles, C_k. For that, we need to consider a slight generalization of WFOMC for conjunctive queries w/o self-joins, were each existential variable x_i ranges over a distinct domain, of size n_i: the standard semantics corresponds to the special case where all domains sizes n_i are equal. We prove that for any β-cyclic query Q, there exists k such that $\mathrm{WFOMC}(C_k, \mathbf{n}, \mathbf{w}, \bar{\mathbf{w}})$ can be reduced to

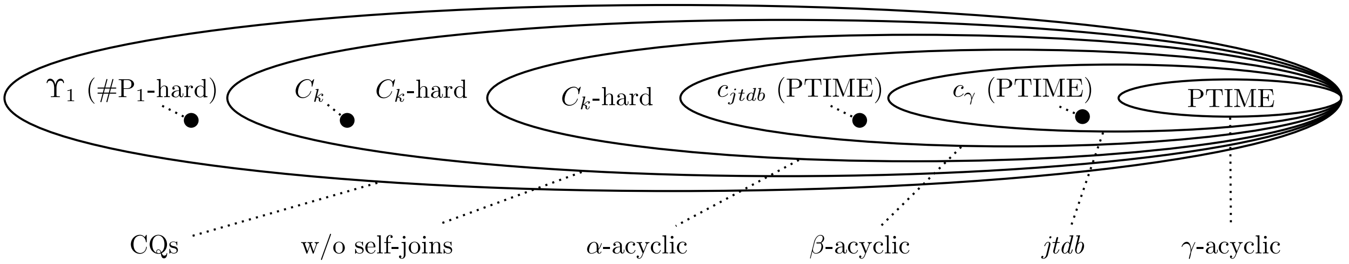

Figure 1: A summary of data complexity results for conjunctive queries (or positive clauses). C_k-hardness is an informal concept described in the main text.

WFOMC($Q, \mathbf{n}', \mathbf{w}', \bar{\mathbf{w}}'$). Hence, the existence of a β-cyclic query with PTIME data complexity would imply PTIME data complexity for at least one C_k (informally called C_k-hardness in Fig. 1). The reduction is as follows. By definition, a β-cyclic query Q contains a weak β-cycle [14] of the form $R_1 x_1 R_2 x_2 \ldots x_{k-1} R_k x_k R_{k+1}$, where $k \geq 3$, all x_i and R_i are distinct, each x_i occurs in both R_i and R_{i+1}, but in no other R_j, and $R_{k+1} = R_1$. Then, we reduce the WFOMC for C_k to that of Q. First, for each relational symbol R_j in Q, if R_j appears in the cycle then we define $w'_j = w_j$ and $\bar{w}'_j = \bar{w}_j$, otherwise $w'_j = \bar{w}'_j = 1$. Second, for all variables x_i that appear in the cycle we set their domain size n_i to be the same as that of the corresponding variable in C_k, otherwise we set $n_i = 1$. Then Q and C_k have the same WFOMC.

Finally, we discuss a peculiar sentence, whose complexity we left open in [18]:

Theorem 3.7. *The data complexity of the symmetric WFOMC problem is in PTIME for the query*

$$\begin{aligned} Q_{S4} =& \forall x_1 \forall x_2 \forall y_1 \forall y_2 (S(x_1, y_1) \vee \\ & \neg S(x_2, y_1) \vee S(x_2, y_2) \vee \neg S(x_1, y_2)) \end{aligned}$$

In [18] we showed that Q_{S4} is in PTIME under the modified semantics, where S is a bipartite graph. This implies that the range of the variables x_1, x_2 is disjoint from the range of the variables y_1, y_2. Now we extended the proof to the standard semantics used in this paper. What makes this query interesting is that the algorithm used to compute it requires a subtle use of dynamic programming, and none of the existing lifted inference rules in the literature are sufficient to compute this query. This suggests that we do not yet have a candidate for a complete set of lifted inference rules for the symmetric WFOMC.

3.3 Proofs

Proof of Theorem 3.1. We briefly recall the basic notions from Valiant's original papers [33, 34]. A *counting Turing machine* is a nondeterministic TM with a read-only input tape and a work tape, that (magically) prints in binary, on a special output tape, the number of its accepting computations. The class #P_1 consists of all functions computed by some counting TM with polynomial (non-deterministic) running time and a unary input alphabet. A function f is #P_1-hard if, for any function g in #P_1 there exists a polynomial time, deterministic TM T_{det} with access to an oracle for f that computes g. Notice that T_{det}'s input alphabet is unary. As usual, f is called #P_1-complete if it is both hard, and in #P_1.

Our proof of Theorem 3.1 consists of two steps. First we construct a #P_1-complete function f, which is computable by a linear time counting TM U_1, which we call a *universal* #P_1 machine; in fact, we will define f by describing U_1. A similar construction in [34] is sketched too briefly to see how the particular pairing function can work; we use a different pairing function and give full details. To prove FO^3 membership, we also need to ensure U_1 runs in (nondeterministic) linear time, which requires some care given that the input is given in unary. Once we have defined U_1, the second step of the proof is a standard construction of an FO formula to simulate U_1: we follow Libkin [28, p. 167], but make several changes to ensure that the formula is in FO^3. The two steps are:

Lemma 3.8. *There exists a counting TM, U_1, with a unary input alphabet, such that (i) U_1 runs in linear time, and (ii) the function f that it computes is #P_1-hard.*

It follows immediately that f is #P_1-complete.

Lemma 3.9. *Let T be any counting TM with a unary input alphabet computing some function f. Suppose T runs in time $O(n^a)$. Then there exists an FO^k formula Φ over some relational vocabulary σ, s.t. $f(n) = \text{FOMC}(\Phi, n)/(n!)$, where $k = 3a$ for $a \geq 1$.*

Theorem 3.1 follows by applying this lemma to U_1, hence $a = 1$ and the formula is in FO^3. By allowing runtimes $O(n^a)$ with $a > 1$, the lemma implies: #$P_1 = \{f \mid \exists \Phi \in FO, \forall n : f(n) = \lfloor \text{FOMC}(\Phi, n)/n! \rfloor\}$; this is an extension of the classic result by Jones and Selman [23], which, restated for the tally notation says $\text{NP}_1 = \{\texttt{Spec}(\Phi) \mid \Phi \in FO\}$ (see [12], [9, Sec.5]). By considering FOMC over unlabeled structures, denoted UFOMC, the correspondence becomes even stronger. In UFOMC, all models that are identical up to a permutation of the constants are counted once, and #$P_1 = \{\text{UFOMC}(\Phi, n) \mid \Phi \in FO\}$.

Proof of Lemma 3.8. The idea for U_1 is simple: its input n is represented in unary and encodes two numbers i, j: $n = e(i, j)$, for some encoding function e to be defined below. U_1 first computes i, j from n, then simulates the ith #P_1 counting TM on input j. The difficult part is to ensure that U_1 runs in linear time:

every TM i that it simulates runs in time $O(j^{k_i})$ for some exponent k_i that depends on i, and thus if we construct U_1 naively to simply simulate machine i on input j, then its runtime is no longer polynomial.

We start by describing an enumeration of counting TMs in $\#\mathrm{P}_1$, $M_1, M_2, \ldots, M_i, \ldots$, with the property that M_i runs in time $\leq (i \cdot j^i + i)^2$ on an input j. We start by listing all counting TMs over a unary input alphabet in standard order $M'_1, M'_2, \ldots$. Then we dovetail pairs of the form $M_i = (M'_r, s)$ where r is an index in the standard TM order and s is a number. M_i represents the counting TM that simulates M'_r on input j with a timer for $s \cdot j^s + s$ steps. The machine M_i can be constructed with at most quadratic slowdown over M'_r (due to the need to increment the counter). We further ensure that dovetailing $M_i = (M'_r, s)$ is done such that $i \geq s$; for that, it suffices to advance r in such a way that i advances at least as fast as s, that is, $M_1 = (M'_1, 1), M_2 = (M'_2, 1), M_3 = (M'_1, 2), M_4 = (M'_2, 2), M_5 = (M'_1, 3), \ldots$. It follows that, for every i, the runtime of M_i on input j is $\leq (i \cdot j^i + i)^2$. It remains to show that the list $M_1, M_2, \ldots, M_i, \ldots$ enumerates precisely all $\#\mathrm{P}_1$ functions. Indeed, each function in this list is in $\#\mathrm{P}_1$, because the runtime of M_i is polynomial in the input j. Conversely, every function in $\#\mathrm{P}_1$ is computed by some M_i in our list, because it is computed by some M'_r whose runtime on input j is $\leq a_r \cdot j^{k_r} + b_r$ and this is $\leq s \cdot j^s + s$ if we choose $s \overset{\text{def}}{=} \max(a_r, b_r, k_r)$. This completes the construction of the enumeration $M_1, M_2, \ldots$.

We describe now the counting machine U_1. Its input is a number n in unary, which represents an encoding $n = e(i, j)$ of two integers i, j. We will choose the encoding function e below such that it satisfies three properties: (a) U_1 can compute i, j from $n = e(i, j)$ in linear time (with auxiliary tapes), (b) $e(i, j) \geq (i \cdot j^i + i)^2$, and (c) for every fixed i, the function $j \mapsto e(i, j)$ can be computed in PTIME. We first prove the lemma, assuming that e satisfies these three properties.

The counting machine U_1 starts by computing a binary representation of its unary input n on its work tape: this step takes linear time in n. Next, it extracts i, j in linear time in n (by property (a)), then it simulates M_i on input j. The runtime of the last step is $\leq (i \cdot j^i + i)^2 \leq e(i, j)$ (by property (b)), hence U_1 runs in linear time in the input $n = e(i, j)$. It remains to prove that the function f computed by U_1 is $\#\mathrm{P}_1$-hard. Consider any function g in $\#\mathrm{P}_1$: we will describe a polynomial-time, deterministic Turing machine T_{det} with an oracle for f that computes g. Since g is in $\#\mathrm{P}_1$ there exists i such that g is computed by M_i. On input j, T_{det} computes $n = e(i, j)$ in PTIME (by property (c)), stores it on the oracle tape, then invokes U_1 and obtains the result $g(j) = f(n)$.

It remains to describe the encoding function e. We take $e(i, j) = 2^i 3^{4i \cdot \lceil \log_3 j \rceil}(6j + 1)$. To prove (a), note that i is obtained by counting the trailing zeroes in the binary representation of n, j is obtained by first computing a ternary representation of $3^{4i \cdot \lceil \log_3 j \rceil}(6j+1)$, ignoring trailing zeros and deriving j from $6j + 1$. (b) $2^i 3^{4i \cdot \lceil \log_3 j \rceil}(6j + 1) \geq (i \cdot j^i + i)^2$ follows through direct calculations, using the fact that $3^{4i \cdot \lceil \log_3 j \rceil} \geq j^{4i}$. (c) is straightforward. □

Proof of Lemma 3.9. We describe here the most important steps of the proof, and delegate the details to Appendix B. We will consider only the case $k = 1$, i.e. the counting TM runs in linear time: the case when $k > 1$ is handled using a standard technique that encodes n^k time stamps using a relation of arity k. We briefly review Trakhtenbrot's proof from Libkin [28, p. 167]: for every deterministic TM, there is a procedure that generates a formula Θ_1 such that TM has an accepting computation starting with an empty input tape iff Θ_1 is satisfiable. The signature for Θ_1 is (this is a minor variation over Libkin's):

$$\sigma = \{<, Min, T_0, T_1, H, (S_q)_{q \in \text{States}(T)}\}$$

Then Θ_1 states that (1) $x < y$ is a total order on the domain and $Min(x)$ is its minimum element, (2) $T_0(t, p)$ (or $T_1(t, p)$) is true iff at time t the tape has a 0 (or a 1) on position p, (3) $H(t, p)$ is true iff at time t the head is on position p, and $S_q(t)$ is true iff at time t the machine is in state q. Libkin [28] describes the sentence Θ_1 that states that all these constraints are satisfied.

We adapt this to a more general construction that is sufficient to prove Lemma 3.9. We address five changes: (1) Our TM is non-deterministic, (2) has k tapes instead of 1, (3) its runtime is $c \cdot n$ instead of n, for some $c > 1$, (4) the input tape initially contains n symbols 1, and (5) Θ_1 needs to be in FO^3.

Support for non-deterministic transitions requires only a slight modification to the sentences. It is also easy to represent multiple tapes, by using k different relations $T_{0\tau_i}, T_{1\tau_i}$, and similarly k head relations H_{τ_i}, for $i = 1, k$. To encode transitions in FO^3, we will assume that the multi-tape TM always reads or writes only one tape at each time. This is without loss of generality: a state that reads and writes all tapes can be converted into a sequence of $2k$ states that first read one by one each tape and "remember" their symbols, then write one by one each tape and move their heads.

Next, we show how to encode running times (and space) up to $c \cdot n$ for some integer constant $c > 1$, with only a domain of size n available. The standard way is to increase the arity of the relations, e.g. with arity a we can represent n^a time steps, but this is not possible within FO^3. Instead, we partition the computation into c *epochs*, each having exactly n time steps, and similarly we partition the tapes into c regions, each with n cells. We denote $T_{0\tau er}(t, p)$, $T_{1\tau er}(t, p)$ the relations T_0, T_1 specialized to tape τ, epoch e, and region r, and similarly define $H_{\tau er}$ and S_{qe}. Furthermore, we modify the sentences that encode the TM transition relation to move the heads across epochs and regions, using only 3 variables. The fourth item is easy: we write a formula stating that initially (at time 1 of epoch 1), region 1 of (input) tape τ_1 is full of 1's, and all other regions and tapes are full of 0's. Moreover, Appendix B shows that Θ_1 can be written in FO^3.

Finally, $\mathrm{FOMC}(\Theta_1, n)$ is precisely the number of accepting computations of the TM on input n, times $n!$, coming from the $n!$ ways of ordering the domain. □

Proof of Theorem 3.6. We show how to compute $\Pr(Q)$ rather than $\text{WFOMC}(Q, n)$: we have seen in Sec. 2 that these two are equivalent. We actually prove the theorem for a more general form of query, where each variable x_i range over a domain of size n_i, thus, $Q = \exists x_1 \in [n_1], \ldots, \exists x_m \in [n_m]\varphi$, where φ is quantifier-free. The probability of a query under the standard semantics (when all variables range over the same domain $[n]$) is obtained by simply setting $n_1 = \cdots = n_m = n$.

To prove the theorem, we use an equivalent definition of γ-acyclicity given by Fagin [14], which we give here together with our algorithm for computing $\Pr(Q)$. The graph is γ-acyclic if it can be reduced to an empty graph by applying the following rules, in any order.

(a) If a node x is isolated (i.e., it belongs to precisely one edge, say $R(x, y, z)$), then delete x. In this case we replace the relation $R(x, y, z)$ by a new relation $R'(y, z)$, where each tuple has probability $1 - (1 - p)^{n_x}$, where p is the probability of tuples in R.

(b) If an edge $R(x)$ is a singleton (i.e., if it contains exactly one node), then delete that edge (but do not delete the node from other edges that might contain it). Here, we condition on the size $k = |R|$. For each k, let p_k be the probability of the *residual query* obtained by removing $R(x)$ and restricting the range of x to $[k]$. By symmetry, this probability depends only on $k = |R|$, and does not depend on the choice of the k elements in the domain. Then $\Pr(Q) = \sum_k \binom{n_x}{k} p_R^k (1 - p_R)^{n_x - k} p_k$, where p_R denotes the probability of a tuple $\Pr(R(i))$, and is the same for all constants i (by symmetry).

(c) If an edge is empty, $R()$, then delete it. We multiply the probability of the residual query by p_R.

(d) If two edges (say $R(x, y, z)$, $S(x, y, z)$) contain precisely the same nodes, then delete one of these edges. Here we replace the two atoms by a new atom $R'(x, y, z)$ whose probability is $p_R \cdot p_S$.

(e) If two nodes x, y are edge-equivalent, then delete one of them from every edge that contains it. (Recall that two nodes are edge-equivalent if they are in precisely the same edges.) Here we replace the two variables x, y by a new variable z, whose range has size $n_z \overset{\text{def}}{=} n_x \cdot n_y$.

Each operation above is in polynomial time in the size of the binary representation of the inputs, and there are only polynomially many operations. Therefore the entire computation is in polynomial time, because each intermediate result can be represented using polynomially many bits. This follows from the fact that the number of models is $2^{O(n^a)}$, where a is the maximum arity of any relation in Q, hence the number of models can be represented using $O(n^a) = n^{O(1)}$ bits.

Example 3.10. *Consider the following linear chain query:*

$$Q = \exists x_0 \exists x_1 \cdots \exists x_m R_1(x_0, x_1) \wedge \cdots R_m(x_{m-1}, x_m)$$

where the probabilities of the m relations are $p_1, \ldots, p_m$. Denote $P_{n_0,\ldots,n_m}$ the probability of Q when the domains of $x_0, x_1, \ldots, x_m$ are sets of sizes $n_0, n_1, \ldots, n_m$ (thus, initially $n_0 = n_1 = \cdots = n_m = n$). Then the variable x_m is isolated (item a), hence we can eliminate it and update the probability of R_m to $1-(1-p_m)^{n_m}$. Now R_m is a singleton relation, hence we can remove it (item b), and restrict the domain of x_{m-1} to have size k_{m-1}, for $k_{m-1} = 1, n_{m-1}$. Therefore:

$$\begin{aligned} P_{n_0,\ldots,n_{m-1},n_m} = \sum_{k_{m-1}=1,n_{m-1}} & P_{n_0,\ldots,n_{m-2},k_{m-1}} \cdot \binom{n_m}{k_m} \\ & \cdot [1 - (1 - p_m)^{k_m}]^{k_{m-1}} \\ & \cdot [(1 - p_m)^{k_m}]^{n_{m-1} - k_{m-1}} \end{aligned}$$

Repeating this process we arrive at an expression that is computable in polynomial time in n (for a fixed m). Notice that this formula does not appear to be computable in polynomial time in both n and m. We leave open the combined complexity of acyclic queries.

Proof of Theorem 3.7. First note that, by using resolution, the query implies the following statement, for every $k \geq 2$:

$$\begin{aligned} \forall x_1, y_1, \ldots, x_k, y_k (S(x_1, y_1) \vee \neg S(x_2, y_1) \\ \vee S(x_2, y_2) \vee \neg S(x_2, y_3) \\ \vee \ldots \vee \neg S(x_1, y_k)) \end{aligned} \quad (5)$$

For any two numbers n_1, n_2, denote $Q_{n_1 n_2} = \forall x_1 \in [n_1], \forall x_2 \in [n_1], \forall y_1 \in [n_2], \forall y_2 \in [n_2]$, $(S(x_1, y_1) \vee \neg S(x_2, y_1) \vee S(x_2, y_2) \vee \neg S(x_1, y_2))$, in other words we restrict the range of the variables to some domains $[n_1], [n_2]$. These domains are not required to be disjoint, instead we use the standard assumption $n_1 \leq n_2$ implies $[n_1] \subseteq [n_2]$. When $n_1 = n_2 = n$ then $Q_{n_1 n_2}$ is equivalent to Q_{S4}. We claim the following. If D is a model of $Q_{n_1 n_2}$, then either property P_a or P_b holds in D:

$$\begin{aligned} P_a \equiv & \exists x \in [n_1], \forall y \in [n_2], S(x, y) \\ P_b \equiv & \exists y \in [n_2], \forall x \in [n_1], \neg S(x, y) \end{aligned}$$

Suppose not. Consider any model of $Q_{n_1 n_2}$ that does not satisfy P_a, P_b. Pick any element $x_1 \in [n_1]$. As P_a does not hold, $\exists y_1 \in [n_2] \neg S(x_1, y_1)$. As P_b does not hold, $\exists x_2 \in [n_1], S(x_2, y_1)$. Continuing, $\exists y_2 \in [n_2]$, $\neg S(x_2, y_2)$, $\exists x_3 \in [n_1], S(x_3, y_2)$ and $\exists y_3 \in [n_2]$, $\neg S(x_3, y_3)$. Continuing, we obtain an arbitrarily long sequence of values $x_1, y_1, x_2, \ldots$ such that: $\neg S(x_1, y_1)$, $S(x_2, y_1)$, $\neg S(x_2, y_2)$, $\ldots$, $S(x_{n_1}, y_{n_1 - 1})$, $\neg S(x_{n_1}, y_{n_1})$. Note that we can never have $x_i = x_j$ or $y_i = y_j$, for $i \neq j$, because that would violate Eq.(5) for $k = j - i$. Since the domain is finite, this is a contradiction.

Therefore, either P_a or P_b holds. Clearly, both statements cannot hold, as they are exclusive events. Denote f and g the weighted model count for $Q_{n_1 n_2}$ in these two cases:

$$\begin{aligned} f(n_1, n_2) = & \text{WFOMC}(Q_{n_1 n_2} \wedge P_a, n, w, \bar{w}) \\ g(n_1, n_2) = & \text{WFOMC}(Q_{n_1 n_2} \wedge P_b, n, w, \bar{w}) \end{aligned}$$

Then we have $\text{WFOMC}(Q_{n_1 n_2}, n, w, \bar{w}) = f(n_1, n_2) + g(n_1, n_2)$. It remains to show how to compute f, g.

Consider a model that satisfies P_a, hence the set $X = \{x \mid \forall y \in [n_2], S(x, y)\}$ is non-empty, hence $k = |X| \geq 1$.

Remove the elements X from the domain $[n_1]$ (and rename the elements such that $[n_1] - X = [n_1 - k]$) and call D' the resulting substructure. Then D' still satisfies the query $Q_{(n_1-k),n_2}$, and, by the removal of all elements X, cannot satisfy P_a, hence it must satisfy P_b. This justifies the following recurrence, completing the proof of Theorem 3.7:

$$f(n_1, 0) = 1 \quad f(n_1, n_2) = \sum_{k=1}^{n_1} \binom{n_1}{k} w^{kn_2} g(n_1 - k, n_2)$$

$$g(0, n_2) = 1 \quad g(n_1, n_2) = \sum_{\ell=1}^{n_2} \binom{n_2}{\ell} \bar{w}^{n_1\ell} f(n_1, n_2 - \ell)$$

4. COMBINED COMPLEXITY

In the combined complexity we consider a fixed vocabulary $\sigma = (R_1, \ldots, R_m)$, and assume that both Φ and n are given as part of the input. As before, n is given in unary (tally) notation. We consider both the FOMC problem, "*compute* FOMC(Φ, n)", and the associated decision problem "*is* $n \in \mathtt{Spec}(\Phi)$*?*". Our upper bound for FOMC also holds for WFOMC. Recall that the spectrum $\mathtt{Spec}(\Phi)$ of a formula Φ is the set of numbers n for which Φ has a model over a domain of size n.

Vardi [40] proved that the model checking problem, "*given* Φ *and a structure* D*, is* Φ *true in* D*?*" is PSPACE-complete. This implies that the above decision problem is also in PSPACE: to check $n \in \mathtt{Spec}(\Phi)$ enumerate over all structures D of a domain of size n, and check if Φ is true in D. By the same argument, FOMC is also in PSPACE. We prove:

Theorem 4.1. *(1) For every* $k \geq 2$*, the combined complexity for FOMC for* FO^k *is #P-complete. (2) The combined complexity for the decision problem* $n \in \mathtt{Spec}(\Phi)$ *is NP-complete for* FO^2*, and is PSPACE-complete for FO.*

The #P-membership in (1) also holds for the WFOMC problem. Recall that the vocabulary σ is fixed. If σ were allowed to be part of the input, then the lower bound in (1) follows immediately from the #P-hardness result for #SAT, because any Boolean formula is trivially encoded as an FO^0 formula, by introducing a new, zero-ary relational symbol for every Boolean variable.

Proof of Theorem 4.1. We start by proving item (1) of Theorem 4.1. To prove membership in #P, it suffices to show that the lineage of a sentence φ of size s over a domain of size n is polynomial in s and n, then use the fact that WMC for Boolean functions is in #P. However, FO^k formulas have, in general, exponentially large lineage, e.g. the formula checking for the existence of a path of length n, $\exists x \exists y (R(x, y) \wedge \exists x (R(y, x) \wedge \exists y (R(y, x) \wedge \ldots)))$, over a domain of size n has lineage of size $\Omega(n^n)$. Instead, we first transform the formula by removing all nested variables. For that, we apply Scott's reduction, which we give below, following the presentation by Grädel, Kolaitis, and Vardi [17, Prop.3.1]; while Scott's reduction was described for FO^2, it carries over unchanged to FO^k. More precisely, the reduction converts a sentence φ of size s into a new sentence φ^* over an extended vocabulary, satisfying the following properties:

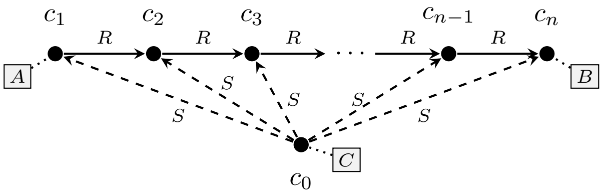

Figure 2: All models of φ_F represent graphs of the depicted form. There is a linear graph of R-edges between the distinguished nodes A and B. The S-edges go from a distinguished node C to all others. They are optional, and are in one-to-one correspondence with the variables X_i in F.

1. The finite models of φ and φ^* are in one-to-one correspondence, and the corresponding models have the same weight.
2. φ^* has size $O(s)$.
3. φ^* is a conjunction of sentences in prenex normal form, i.e. $Q_1 x_1 Q_2 x_2 \cdots Q_k x_k \psi$ where each Q_i is either $\forall$ or $\exists$, and ψ is quantifier-free.

The new formula has a lineage of size $O(n^k s)$, because its quantifier depth is bounded by $k = O(1)$, which implies WFOMC is in #P. It remains to describe Scott's reduction, which we review here briefly, for completeness. Introduce a new relational symbol S_ψ for every subformula ψ of φ, where the arity of S_ψ equals the number of free variables in ψ, and define the sentence $\theta_\psi \equiv \forall x_1 \cdots \forall x_\ell (S_\psi(x_1, \ldots, x_\ell) \Leftrightarrow \theta'_\psi)$, where θ'_ψ depends on the structure of ψ as follows: if ψ is an atomic formula, then $\theta'_\psi = \psi$, if $\psi = \psi_1 \wedge \psi_2$ then $\theta'_\psi = S_{\psi_1} \wedge S_{\psi_2}$, if $\psi = \neg\psi_1$ then $\theta'_\psi = \neg S_{\psi_1}$ and if $\psi = \forall x \psi_1$ then $\theta'_\psi = \forall x S_{\psi_1}$. The new formula φ^* is defined as $S_\varphi \wedge \bigwedge_\psi \theta_\psi$. By setting $w(S_\psi) = \bar{w}(S_\psi) = 1$ for all new symbols, we ensure that the models of φ and φ^* are not just in one-to-one correspondence; they have the same weights.

Next we prove #P-hardness for FO^2 (this implies hardness for FO^k for every $k \geq 2$). We use reduction from #SAT: given a Boolean formula F over n variables $X_1, \ldots, X_n$, compute $\#F$. This problem is #P-hard [33].

Define the vocabulary σ consisting of 3 unary symbols A, B, C, and 2 binary symbols R, S. Given a Boolean formula F, we construct an FO^2 sentence φ_F such that, over a domain of size $n+1$, the number of models of φ_F is FOMC$(\varphi_F, n+1) = (n+1)! \cdot \#F$. The sentence φ_F enforces a particular graph structure, as illustrated in Figure 2, by asserting the following:

- There exists three unique, distinct elements x, y, z such that $A(x), B(y), C(z)$ are true:
  $\exists x A(x) \wedge \forall x, \forall y (A(x) \wedge A(y)) \Rightarrow x = y$, and similarly for B and C;
  $\neg\exists x (A(x) \wedge B(x))$ and similarly for A, C, and B, C.
- There exist n elements $x_1, \ldots, x_n$ such that the following holds:
  $A(x_1), R(x_1, x_2), R(x_2, x_3), \ldots, R(x_{n-1}, x_n), B(x_n)$.
  This is expressible in FO^2, by reusing variables.

Untyped triangles	$\exists x, y, z(R(x,y), R(y,z), R(z,x))$
Typed triangles (3-cycle)	$\exists x, y, z(R(x,y), S(y,z), T(z,x))$
k-cycle, for $k \geq 3$	$\exists x_1, \ldots, x_k(R_1(x_1,x_2), R_2(x_2,x_3), \ldots, R_k(x_k,x_1)$
Transitivity	$\forall x, y, z(E(x,y) \wedge E(y,z) \Rightarrow E(x,z))$
Homophily	$\forall x, y, z(R(x,y) \wedge S(x,z) \Rightarrow R(z,y))$
Extension Axiom (Simplified)	$\forall x_1, x_2, x_3(x_1 \neq x_2 \wedge x_1 \neq x_3 \wedge x_2 \neq x_3 \;\Rightarrow\; \exists y E(x_1,y) \wedge E(x_2,y) \wedge E(x_3,y))$

Table 2: A list of open problems: for each formula it is conjectured that FOMC is hard.

- For every number $m \in [2n]-\{n\}$, it is not the case that there exists m elements $x_1, \ldots, x_m$ such that: $A(x_1), R(x_1,x_2), R(x_2,x_3), \ldots, R(x_{m-1},x_m), B(x_m)$.
- For all x, y, if $R(x,y)$ then neither $C(x)$ nor $C(y)$.
- For all x, y, if $S(x,y)$ then $C(x)$.
- Finally, φ_F contains a statement obtained from F by replacing each Boolean variable X_i by the sentence $\gamma_i \stackrel{\text{def}}{=} \exists x, \exists z(S(z,x) \wedge \alpha_i(x))$, where $\alpha_i(x)$ is the following formula with free variable x: there exists a path $A(x_1), R(x_1,x_2), \ldots, R(x_{i-1},x)$ (if $i = 1$ then $\alpha_1(x) \equiv A(x)$).

The reader may check that, for any database instance D over a domain of size $n+1$ that satisfies φ_F there exists a unique permutation $c_0, c_1, \ldots, c_n$ over its domain such that the relations A, B, C and R contain precisely the following tuples: $C(c_0), A(c_1), B(c_n)$, $R(c_1,c_2), \ldots$, $R(c_{n-1},c_n)$. Indeed, if it contained some $R(c_i,c_j)$ with $j \neq i+1$, then $c_1, \ldots, c_i, c_j, c_{j+1}, \ldots, c_n$ forms a path from A to B of some length $m \leq 2n$ and $m \neq n$, which contradicts the sentence φ_F; notice also that $\alpha_i(x)$ is true iff $x = c_i$. Therefore the only relation that is left unspecified in D is S, which may contain an arbitrary number of tuples of the form $S(c_0,c_i)$. These tuples are in one-to-one correspondence with the Boolean variables X_i, proving our claim.

Now we prove item (2) of Theorem 4.1. The claim for FO^2 follows immediately from the proof above. It remains to prove that the combined complexity for FO is PSPACE, for which we use a reduction from the Quantified Boolean Formula (QBF) problem, which is known to be PSPACE complete. A *Quantified Boolean Formula* is a formula of the form $Q_1X_1Q_2X_2\ldots Q_nX_nF$ where each Q_i is a quantifier $\forall$ or $\exists$, and F is a Boolean formula over the variables $X_1, \ldots, X_n$. We make the following change to the construction above. Recall that a Boolean variable X_i in F was represented by $S(c_0,c_i)$. Now we extend S to a ternary relation $S(x,y,u)$, restrict u to two constants (we choose c_1 and c_n arbitrarily) and represent X_i by $S(c_0,c_i,c_1)$ and $\neg X_i$ by $S(c_0,c_i,c_n)$. Then, we replace the quantifiers $\forall X_i$ or $\exists X_i$ with $\forall u$ or $\exists u$. More precisely, the new formula φ_F contains the following statements:

- If $S(x,y,u)$ is true, then u is either the distinguished A or the distinguished B element: $\forall x, y, u(S(x,y,u) \Rightarrow A(u) \vee B(u))$.
- If u, v are the distinguished A and B elements, then $S(x,y,u)$ is the negation of $S(x,y,v)$: $\forall u, v, x, y(A(u) \wedge B(v) \Rightarrow (S(x,y,u) \texttt{ xor } S(x,y,v)))$.

Finally, we rewrite a QBF $\forall X_i(\ldots)$ into $\forall u(A(u) \vee B(u) \Rightarrow \ldots)$ and a QBF $\exists X_i(\ldots)$ into $\exists u((A(u) \vee B(u)) \wedge \ldots)$. We omit the straightforward details.

5. CONCLUSIONS

In this paper we discuss the symmetric Weighted FO Model Counting Problem. Our motivation comes from probabilistic inference in Markov Logic Networks, with applications to modern, large knowledge bases, but the problem is also of independent theoretical interest. We studied both the data complexity, and the combined complexity. For the data complexity we established for the first time the existence of an FO sentence for which the Symmetric Model Counting problem is $\#\mathrm{P}_1$-hard, and also the existence of a Conjunctive Query for which the Symmetric Weighted Model Counting problem is $\#\mathrm{P}_1$-hard. We also showed that for all γ-acyclic conjunctive queries WFOMC can be computed in polynomial time. For the combined complexity, we proved a tight bound of #P-completeness for FO^2. We also discussed the associate decisions problem.

We end this paper with a list of open problems, listed in Table 2: for each query in the table, the complexity of the FOMC or the WFOMC problem is open.

Acknowledgments. We thank Ronald Fagin, Phokion Kolaitis and Lidia Tendera for discussions on topics related to this paper. This work was partially supported by NSF IIS-1115188, IIS-0911036, CCF-1217099, and the Research Foundation-Flanders (FWO-Vlaanderen).

6. REFERENCES

[1] Alberto Bertoni and Massimiliano Goldwurm. On ranking 1-way finitely ambiguous NL languages and $\#\mathrm{P}_1$-complete census functions. *ITA*, 27(2):135–148, 1993.

[2] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.

[3] Kevin J. Compton. The computational complexity of asymptotic problems i: Partial orders. *Inf. Comput.*, 78(2):108–123, 1988.

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

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

[6] http://deepdive.stanford.edu/.

[7] Pedro Domingos and Daniel Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence.* Morgan & Claypool Publishers, 2009.

[8] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *KDD*, 2014.

[9] Arnaud Durand, Neil D. Jones, Johann A. Makowsky, and Malika More. Fifty years of the spectrum problem: survey and new results. *Bulletin of Symbolic Logic*, 18(4):505–553, 2012.

[10] David Duris. Some characterizations of γ and β-acyclicity of hypergraphs. *Information Processing Letters*, 112(16):617–620, 2012.

[11] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *EMNLP*, pages 1535–1545, 2011.

[12] Ronald Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In *SIAM-AMS Proceedings 7*, pages 43–73, 1974.

[13] Ronald Fagin. Probabilities on finite models. *J. Symb. Log.*, 41(1):50–58, 1976.

[14] Ronald Fagin. Degrees of acyclicity for hypergraphs and relational database schemes. *J. ACM*, 30(3):514–550, 1983.

[15] Vibhav Gogate and Pedro Domingos. Probabilistic theorem proving. In *UAI*, pages 256–265, 2011.

[16] Carla P Gomes, Joerg Hoffmann, Ashish Sabharwal, and Bart Selman. From sampling to model counting. In *IJCAI*, pages 2293–2299, 2007.

[17] Erich Grädel, Phokion G. Kolaitis, and Moshe Y. Vardi. On the decision problem for two-variable first-order logic. *Bulletin of Symbolic Logic*, 3(1):53–69, 1997.

[18] Eric Gribkoff, Guy Van den Broeck, and Dan Suciu. Understanding the complexity of lifted inference and asymmetric weighted model counting. In *UAI*, pages 280–289, 2014.

[19] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *AIJ*, 194:28–61, 2013.

[20] Manfred Jaeger. Lower complexity bounds for lifted inference. *TPLP*, 2012.

[21] Manfred Jaeger and Guy Van den Broeck. Liftability of probabilistic inference: Upper and lower bounds. In *StarAI*, 2012.

[22] Abhay Kumar Jha and Dan Suciu. Probabilistic databases with MarkoViews. *VLDB*, 5(11):1160–1171, 2012.

[23] Neil Jones and Alan Selman. Turing machines and the spectra of first-order formulas with equality. In *STOC*, pages 157–167, 1972.

[24] Kristian Kersting, Babak Ahmadi, and Sriraam Natarajan. Counting belief propagation. In *UAI*, pages 277–284, 2009.

[25] Angelika Kimmig, Guy Van den Broeck, and Luc De Raedt. Algebraic model counting. *arXiv preprint arXiv:1211.4475*, 2012.

[26] S. Kok, P. Singla, M. Richardson, and P. Domingos. The alchemy system for statistical relational AI, 2005.

[27] Phokion G. Kolaitis and Moshe Y. Vardi. 0-1 laws for fragments of existential second-order logic: A survey. In *MFCS*, pages 84–98, 2000.

[28] Leonid Libkin. *Elements of Finite Model Theory*. Springer, 2004.

[29] Mathias Niepert. Symmetry-aware marginal density estimation. In *AAAI*, 2013.

[30] Feng Niu, Christopher Ré, AnHai Doan, and Jude W. Shavlik. Tuffy: Scaling up statistical inference in Markov logic networks using an RDBMS. *VLDB*, 4(6):373–384, 2011.

[31] Hoifung Poon and Pedro Domingos. Sound and efficient inference with probabilistic and deterministic dependencies. In *AAAI*, pages 458–463, 2006.

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

[33] Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.

[34] Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 1979.

[35] Guy Van den Broeck. On the completeness of first-order knowledge compilation for lifted probabilistic inference. In *NIPS*, pages 1386–1394, 2011.

[36] Guy Van den Broeck and Adnan Darwiche. On the complexity and approximation of binary evidence in lifted inference. In *NIPS*, 2013.

[37] Guy Van den Broeck, Wannes Meert, and Adnan Darwiche. Skolemization for weighted first-order model counting. In *KR*, 2014.

[38] Guy Van den Broeck, Wannes Meert, and Jesse Davis. Lifted generative parameter learning. In *StaRAI*, 2013.

[39] Guy Van den Broeck, Nima Taghipour, Wannes Meert, Jesse Davis, and Luc De Raedt. Lifted probabilistic inference by first-order knowledge compilation. In *IJCAI*, pages 2178–2185. AAAI Press, 2011.

[40] Moshe Y. Vardi. The complexity of relational query languages (extended abstract). In *STOC*, pages 137–146, 1982.

[41] Deepak Venugopal and Vibhav Gogate. Scaling-up importance sampling for markov logic networks. In *NIPS*, 2014.

[42] Wei Wei, Jordan Erenrich, and Bart Selman. Towards efficient sampling: Exploiting random walk strategies. In *AAAI*, pages 670–676, 2004.

[43] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Qili Zhu. Probase: a probabilistic taxonomy for text understanding. In *SIGMOD*, pages 481–492, 2012.

[44] Ce Zhang and Christopher Ré. Towards high-throughput gibbs sampling at scale: a study across storage managers. In *SIGMOD*, pages 397–408, 2013.

APPENDIX

A. THE THREE LEMMAS

A.1 Proof of Lemma 3.3: Removing Exists

Following the proof of [37], we show how to eliminate existential quantifiers from a WFOMC problem (a form of Skolemization). Assume that Φ is in prenex normal form: $\Phi = Q_1 x_1 Q_2 x_2 \ldots Q_k x_k \Psi$, where each Q_i is either $\forall$ or $\exists$, and Ψ is quantifier-free. Let i be the first position of an $\exists$, and denote $\varphi(\mathbf{x}, x_i) = Q_{i+1} x_{i+1} \ldots Q_k x_k \Psi$; note that φ is a formula with free variables $\mathbf{x} = (x_1, \ldots, x_{i-1})$. We have:

$$\Phi = \forall \mathbf{x} \exists x_i \varphi(\mathbf{x}, x_i)$$

Let A be a fresh relational symbol of arity i. The new formula Φ' is:

$$\Phi' = \forall \mathbf{x}((\exists x_i \varphi(\mathbf{x}, x_i)) \Rightarrow A(\mathbf{x})) \tag{6}$$

Let $w', \bar{w}'$ denote the weights of Φ's vocabulary extended with $w(A) = 1$ and $\bar{w}(A) = -1$. We claim that $\text{WFOMC}(\Phi, n, w, \bar{w}) = \text{WFOMC}(\Phi', n, w', \bar{w}')$. Consider a possible world $D \subseteq \text{Tup}(n)$ that satisfies Φ'. Call D "good" if it also satisfies Φ. In a good world D, for any constants $\mathbf{x} = \mathbf{a}$, the sentence $\exists x_i \varphi(\mathbf{a}, x_i)$ is true, hence $A(\mathbf{a})$ is also true (because Φ' is true), and therefore the weight of D is the same as the weight of $D - \{A\}$, the world obtained from D by removing all tuples referring to the relational symbol A: $W(D, w', \bar{w}') = W(D - \{A\}, w, \bar{w})$. Thus, the sum of the weights of the good worlds (see Eq.(3)) is precisely $\text{WFOMC}(\Phi, n, w, \bar{w})$. We prove that the sum of the weights of the bad worlds is zero. Let D be a bad world: it satisfies Φ' but not Φ. Thus, there exists some constants $\mathbf{a}$ s.t. the sentence $\exists x_i \varphi(\mathbf{a}, x_i)$ is false; choose $\mathbf{a}$ to be the first such constants, in some lexicographic order. Let D' be the world obtained from D by flipping the status of $A(\mathbf{a})$: thus D, D' are identical, but one sets $A(\mathbf{a})$ to true and the other to false. Both satisfy Φ', and $W(D, w', \bar{w}') = -W(D', w', \bar{w}')$, therefore they cancel out in the sum of Eq.(3). This proves that $\text{WFOMC}(\Phi, n, w, \bar{w}) = \text{WFOMC}(\Phi', n, w', \bar{w}')$.

We note that Φ' can be written equivalently as:

$$\Phi' = \forall \mathbf{x} \forall x_i (\neg \varphi(\mathbf{x}, x_i) \vee A(\mathbf{x}))$$

In other words, we have replaced the first existential quantifier in Φ by a universal quantifier (and may have increased the number of $\exists$ on positions $i+1, i+2, \ldots$). Lemma 3.3 follows by applying this procedure inductively.

A.2 Proof of Lemma 3.4: Removing Negation

Let $\neg\psi(\mathbf{x})$ be a negated subformula of Φ, with k free variables $\mathbf{x}$. Let A, B be two new relational symbols of arity k. Let Φ_p denote the sentence obtained from Φ by replacing the subformula $\neg\psi(\mathbf{x})$ with $A(\mathbf{x})$. Denote:

$$\Delta = \forall \mathbf{x}[(\psi(\mathbf{x}) \vee A(\mathbf{x})) \wedge (A(\mathbf{x}) \vee B(\mathbf{x})) \wedge (\psi(\mathbf{x}) \vee B(\mathbf{x}))] \tag{7}$$

Extend the weight functions $w, \bar{w}$ to $w', \bar{w}'$ by setting $w(A) = \bar{w}(A) = w(B) = 1$, $\bar{w}(B) = -1$. Define $\Phi' = \Phi_p \wedge \Delta$. We claim that $\text{WFOMC}(\Phi, n, w, \bar{w}) = \text{WFOMC}(\Phi', n, w', \bar{w}')$. To prove this, consider a world $D \subseteq \text{Tup}(n)$ over the vocabulary of Φ', and assume that D satisfies Φ'. Call D "good", if the statement $\forall \mathbf{x}(\psi(\mathbf{x}) \text{ xor } A(\mathbf{x}))$ holds. It is easy to see that in any good world, $\forall \mathbf{x} B(\mathbf{x})$ holds too, hence the good world has the same weight as the world obtained by stripping it of the additional relations A, B, and, furthermore, their contributions to $\text{WFOMC}(\Phi', n, w', \bar{w}')$ is precisely $\text{WFOMC}(\Phi, n, w, \bar{w})$. Consider a bad world: it satisfies Φ', but there exists $\mathbf{a}$ such both $\psi(\mathbf{a})$ and $A(\mathbf{a})$ are true. In that case $B(\mathbf{a})$ can be set arbitrarily to true or false and still satisfy the formula Φ', hence the contributions of these two pairing worlds cancel out. This proves that $\text{WFOMC}(\Phi, n, w, \bar{w}) = \text{WFOMC}(\Phi', n, w', \bar{w}')$.

Lemma 3.4 follows by apply this process repeatedly.

A.3 Proof of Lemma 3.5: Removing Equality

Let $E(x, y)$ be a new predicate symbol, with weights $w(E) = z$ and $\bar{w}(E) = 1$, where z is a real value to be determined below. Define Φ_E to be obtained from Φ by replacing every equality predicate $x = y$ with $E(x, y)$, and define:

$$\Phi' = \Phi_E \wedge \forall x E(x, x)$$

Consider the count $f(z) = \text{WFOMC}(\Phi', n, w', \bar{w}')$ as a function of z, where $w', \bar{w}'$ extend $w, \bar{w}$ with $w(E) = z, \bar{w}(E) = 1$. This is a polynomial of degree n^2 in z. Since Φ' asserts $\forall x E(x, x)$, all monomials in f have a degree $\geq n$. Let $c \cdot z^n$ be the monomial of degree n. Then we claim that its coefficient $c = \text{WFOMC}(\Phi, n, w, \bar{w})$. Indeed, every world D where E has exactly n tuples is a world where E is interpreted as the equality predicate. We can compute c using $n + 1$ calls to an oracle for $f(z)$ as follows. Fix $\delta > 0$, and denote $\Delta^0 f = f$, $\Delta^{k+1} f(z) = (\Delta^k f)(z + \delta) - (\Delta^k f)(z)$. Then $\Delta^n f(0) = c \cdot n! = \binom{n}{0} f(0) - \binom{n}{1} f(\delta) + \binom{n}{2} f(2\delta) - \cdots (-1)^n \binom{n}{n} f(n\delta)$.

B. A #P$_1$-HARD SENTENCE Θ_1

We prove here Lemma 3.9: shows how to reduce a linear-time, multi-tape counting TM with a unary input alphabet (such as the #P$_1$-complete TM) to the FOMC problem on a first-order sentence. The sentence that encodes the #P$_1$-complete TM is referred to as Θ_1. This proof is based on the standard encoding of a deterministic Turing machine into first-order logic, as used to prove Trakhtenbrot's theorem [28, p. 167]. We extend this construction in several ways: (1) towards non-deterministic counting Turing machines, (2) with multiple tapes, (3) with a run time of $c \cdot n$ for some fixed c, instead of n, (4) to have n symbols 1 on the input tape, followed by symbols 0, and finally (5) to obtain a sentence in FO^3.

As discussed in Section 3.3, we need to encode run times and space with lengths up to $c \cdot n$, yet we only have a domain size of exactly n available. This is solved by partitioning the run time into c *epochs* of n steps, and the space into c *regions* of n cells. Moreover, we assume w.l.o.g. that there are two symbols: $\{0, 1\}$.

B.1 Signature

The signature of Θ_1 consists of the following predicates P/a, where a is the arity of P:

- $</2$, denoting a *strict linear order* on the domain,

- *Succ*/2, denoting the *successor* relation w.r.t. the order on the domain,
- *Min*/1 and *Max*/1, denoting the *smallest* and *largest* domain element
- *state* predicates $S_{qe}/1$, where $S_{qe}(t)$ is true precisely when the machine is in state q at time t in epoch e,
- *head* predicates $H_{\tau er}/2$, where $H_{\tau er}(t,p)$ is true precisely when at time t in epoch e, the head for tape τ is at position p in region r, and
- *tape* predicates $T_{s\tau er}/2$, where $T_{s\tau er}(t,p)$ is true precisely when at time t in epoch e, tape τ contains symbol $s \in \{0,1\}$ at position p in region r,
- *movement* predicates $\mathit{Left}_{\tau er}/2$ and $\mathit{Right}_{\tau er}/2$, where $\mathit{Left}_{\tau er}(t,p)$ is true precisely when the head on tape τ at time t in epoch e is to the left of p in region r (or when p, r is the first cell on its tape and the head is there), and *Right* is defined similarly, and
- *frame* predicate $\mathit{Unchanged}_{\tau er}/2$, where we have that $\mathit{Unchanged}_{\tau er}(t,p)$ is true precisely when position p in region r of tape τ did not change going from time t in epoch e to the next time step.

B.2 Sentences

To encode the Turing machine, we let Θ_1 consist of the following sentences.

1. $<$ is an arbitrary strict linear order (total, antisymmetric, irreflexive, and transitive):

$$\begin{aligned} &\forall x, \forall y,\ \neg(x = y) \Rightarrow (x < y) \vee (y < x) \\ &\forall x, \forall y,\ \neg(x < y) \vee \neg(y < x) \\ &\forall x, \forall y, \forall z,\ (x < y) \wedge (y < z) \Rightarrow (x < z) \end{aligned}$$

2. *Min* is the smallest element, and *Max* is the largest element:

$$\begin{aligned} &\forall x,\ \mathit{Min}(x) \Leftrightarrow \neg\exists y, (y < x) \\ &\forall x,\ \mathit{Max}(x) \Leftrightarrow \neg\exists y, (x < y) \end{aligned}$$

3. *Succ* is the successor relation:

$$\forall x, \forall y,\ \mathit{Succ}(x,y) \Leftrightarrow (x < y) \wedge \neg\exists z, (x < z) \wedge (z < y)$$

4. At any time, the machine is in exactly one state:

$$\bigwedge_{q,q',e:q\neq q'} \forall t,\ \neg S_{qe}(t) \vee \neg S_{q'e}(t)$$

$$\bigwedge_{e} \forall t,\ \bigvee_{q} S_{qe}(t)$$

5. At any time, the head is in exactly one position per tape:

   (a) The head is in at least one position:

$$\bigwedge_{\tau,e} \forall t, \exists p, \bigvee_{r} H_{\tau er}(t,p)$$

   (b) The head is in at most one region:

$$\bigwedge_{\tau,e,r} \forall t, \forall p,\ H_{\tau er}(t,p) \Rightarrow \bigwedge_{r':r'\neq r} \forall p', \neg H_{\tau er'}(t,p')$$

   (c) The head is in at most one position per region:

$$\bigwedge_{\tau,e,r} \forall t, \forall p,\ H_{\tau er}(t,p) \Rightarrow \neg\exists p', \neg(p = p') \wedge H_{\tau er}(t,p')$$

6. At any time, each tape position has exactly one symbol:

$$\bigwedge_{\tau,e,r} \forall t, \forall p,\ T_{0\tau er}(t,p) \Leftrightarrow \neg T_{1\tau er}(t,p)$$

7. In the initial configuration of the TM (first time step),

   (a) it is in state q_1, and its heads are in the first position:

$$\forall x, \mathit{Min}(x) \Rightarrow S_{q_1 e_1}(x) \wedge \bigwedge_{\tau} H_{\tau e_1 r_1}(x,x)$$

   (b) the first (input) tape τ_1 contains n symbols 1 in the first region, followed by symbol 0 in all other regions (starting with cell $n+1$), and all other tapes τ_i contain symbol 0:

$$\forall t, \mathit{Min}(t) \Rightarrow \forall p, T_{1\tau_1 e_1 r_1}(t,p) \wedge \bigwedge_{i:i>1} T_{0\tau_1 e_1 r_i}(t,p) \wedge \bigwedge_{i,r:i>1} T_{0\tau_i e_1 r}(t,p)$$

8. An encoding of the transition relation δ. For example, that state q_a operates on tape τ_a, and that $\delta(q_a, 0) = \{(q_b, 1, L), (q_c, 0, R)\}$ is encoded into the following sentences.

   (a) What changed when t is before the end of an epoch (i.e., has a successor in the epoch):

$$\bigwedge_{e,r} \forall t, t', \forall p, \begin{bmatrix} S_{q_a e}(t) \\ \wedge H_{\tau_a er}(t,p) \\ \wedge T_{0\tau_a er}(t,p) \\ \wedge \mathit{Succ}(t,t') \end{bmatrix} \Rightarrow \begin{bmatrix} S_{q_b e}(t') \\ \wedge \mathit{Left}_{\tau_a er}(t',p) \\ \wedge T_{1\tau_a er}(t',p) \end{bmatrix} \vee \begin{bmatrix} S_{q_c e}(t') \\ \wedge \mathit{Right}_{\tau_a er}(t',p) \\ \wedge T_{0\tau_a er}(t',p) \end{bmatrix}$$

   (b) What changed when t is at the end of an epoch:

$$\bigwedge_{i,r:1\leq i<c} \forall t, t', \forall p, \begin{bmatrix} S_{q_a e_i}(t) \\ \wedge H_{\tau_a e_i r}(t,p) \\ \wedge T_{0\tau_a e_i r}(t,p) \\ \wedge \mathit{Max}(t) \\ \wedge \mathit{Min}(t') \end{bmatrix} \Rightarrow \begin{bmatrix} S_{q_b e_{i+1}}(t') \\ \wedge \mathit{Left}_{\tau_a e_{i+1} r}(t',p) \\ \wedge T_{1\tau_a e_{i+1} r}(t',p) \end{bmatrix} \vee \begin{bmatrix} S_{q_c e_{i+1}}(t') \\ \wedge \mathit{Right}_{\tau_a e_{i+1} r}(t',p) \\ \wedge T_{0\tau_a e_{i+1} r}(t',p) \end{bmatrix}$$

   (c) What does not change on the tapes: other cells in the region of τ_a where the head is, regions with no head, and tapes other than τ_a.

$$\bigwedge_{e,r} \forall t, \forall p, \left[\begin{array}{r} S_{q_a e}(t) \\ \wedge H_{\tau_a er}(t,p) \end{array} \right]$$
$$\Rightarrow \forall p', (p = p') \vee \mathit{Unchanged}_{\tau_a er}(t,p')$$

$$\bigwedge_{\tau,e,r} \forall t, \forall p, H_{\tau er}(t,p)$$
$$\Rightarrow \bigwedge_{r':r' \neq r} \forall p, \mathit{Unchanged}_{\tau er'}(t,p)$$

$$\bigwedge_{e} \forall t, S_{q_a e}(t)$$
$$\Rightarrow \bigwedge_{\tau,r:\tau \neq \tau_a} \forall p, \mathit{Unchanged}_{\tau er}(t,p)$$

(d) The positions of the heads on tapes other than τ_a do not change:

$$\bigwedge_{\tau,e,r:\tau \neq \tau_a} \forall t, t', \forall p, \left[\begin{array}{r} S_{q_a e}(t) \\ \wedge H_{\tau er}(t,p) \\ \wedge \mathit{Succ}(t,t') \end{array} \right]$$
$$\Rightarrow H_{\tau er}(t',p)$$

$$\bigwedge_{\tau,i,r:\tau \neq \tau_a, 1 \leq i < c} \forall t, t', \forall p, \left[\begin{array}{r} S_{q_a e_i}(t) \\ \wedge H_{\tau e_i r}(t,p) \\ \wedge \mathit{Max}(t) \\ \wedge \mathit{Min}(t') \end{array} \right]$$
$$\Rightarrow H_{\tau e_{i+1} r}(t',p)$$

9. The movement predicates are defined as

$$\bigwedge_{\tau,e,r} \forall t, \forall p, p', \left[\begin{array}{r} \mathit{Left}_{\tau er}(t,p) \\ \wedge \mathit{Succ}(p',p) \end{array} \right] \Leftrightarrow H_{\tau er}(t,p')$$

$$\bigwedge_{\tau,e,i:1 \leq i < c} \forall t, \forall p, p', \left[\begin{array}{r} \mathit{Left}_{\tau er_{i+1}}(t,p) \\ \wedge \mathit{Min}(p) \\ \wedge \mathit{Max}(p') \end{array} \right] \Leftrightarrow H_{\tau er_i}(t,p')$$

$$\bigwedge_{\tau,e} \forall t, \forall p, \left[\begin{array}{r} \mathit{Left}_{\tau er_1}(t,p) \\ \wedge \mathit{Min}(p) \end{array} \right] \Leftrightarrow H_{\tau er_1}(t,p)$$

$$\bigwedge_{\tau,e,r} \forall t, \forall p, p', \left[\begin{array}{r} \mathit{Right}_{\tau er}(t,p) \\ \wedge \mathit{Succ}(p,p') \end{array} \right] \Leftrightarrow H_{\tau er}(t,p')$$

$$\bigwedge_{\tau,e,i:1 \leq i < c} \forall t, \forall p, p', \left[\begin{array}{r} \mathit{Right}_{\tau er_i}(t,p) \\ \wedge \mathit{Max}(p) \\ \wedge \mathit{Min}(p') \end{array} \right] \Leftrightarrow H_{\tau er_{i+1}}(t,p')$$

$$\bigwedge_{\tau,e} \forall t, \forall p, \left[\begin{array}{r} \mathit{Right}_{\tau er_c}(t,p) \\ \wedge \mathit{Max}(p) \end{array} \right] \Leftrightarrow H_{\tau er_c}(t,p)$$

10. The frame predicates are defined as

$$\bigwedge_{s,\tau,e,r} \forall t, t', \forall p, \left[\begin{array}{r} T_{s\tau er}(t,p) \\ \wedge \mathit{Unchanged}_{\tau er}(t,p) \\ \wedge \mathit{Succ}(t,t') \end{array} \right]$$
$$\Leftrightarrow T_{s\tau er}(t',p)$$

$$\bigwedge_{s,\tau,i,r:1 \leq i < c} \forall t, t', \forall p, \left[\begin{array}{r} T_{s\tau e_i r}(t,p) \\ \wedge \mathit{Unchanged}_{\tau e_i r}(t,p) \\ \wedge \mathit{Max}(t) \\ \wedge \mathit{Min}(t') \end{array} \right]$$
$$\Leftrightarrow T_{s\tau e_{i+1} r}(t',p)$$

11. The machine terminates in an accepting state (e.g., q_1, q_5, q_{42}, etc.) :

$$\forall t, \mathit{Max}(t) \Rightarrow S_{q_1 e_c}(t) \vee S_{q_5 e_c}(t) \vee S_{q_{42} e_c}(t) \vee \ldots$$

It is easy to verify that Θ_1 uses at most three logical variables per sentence, and that Θ_1 is therefore in FO^3. Note that FO^3 permits variables to be reused within the same sentence.

For a fixed model of $</2$, that is, a fixed order on the domain, the models of Θ_1 for domain size n correspond one-to-one to the accepting computations of the TM on input n. Since there are exactly $(n!)$ models of $</2$, we can compute the number of accepting computations from the FOMC efficiently.

C. PTIME DATA COMPLEXITY FOR FO^2

The proof of the fact that the data complexity for FO^2 is in PTIME is spread over two references, [35] and [37]. We include here a brief proof, for completeness.

Given an FO^2 formula φ of size s, we start by applying the reduction in [17], which converts φ into a formula φ^* with the following properties:

- Every relational symbol occurring in φ^* has arity at most 2.
- Items 1 and 3 of Scott's reduction hold. (Item 2 becomes: φ^* has size $O(s \log s)$. In our case φ is fixed, so it suffices to note that the size of φ^* is $O(1)$.)

The reduction consists of Scott's reduction described above, plus the following transformation that ensures that all relational symbols have arity ≤ 2. Replace each relational atom of arity > 2 by a new unary or binary symbol, for example, replace the atoms $R(x,y,x)$, $R(y,y,y)$, $R(x,x,y)$ by $R_1(x,y)$, $R_2(x)$, $R_3(x,y)$. Then append to φ^* conjuncts asserting how the new relational symbols relate, for example $\forall x(R_1(x,x) \leftrightarrow R_2(x))$; we refer the reader to [17] for details.

We perform one more transformation: remove all existential quantifiers by using Lemma 3.3, thus transforming φ^* into a universally quantified sentences:

$$\varphi^* = \forall x \, \forall y \, \psi(x,y)$$

where $\psi(x,y)$ is a quantifier-free formula.

If φ^* contains any relational symbol R of arity zero then we perform a Shannon expansion and compute $P(\varphi^*) = \Pr(\varphi^*[R = \texttt{false}]) \cdot (1 - p(R)) + \Pr(\varphi^*[R = \texttt{true}]) \cdot p(R)$. Thus, we can assume w.l.o.g. that all relational symbols in φ^* have arity 1 or 2.

Assume first that all relational symbols in φ^* have arity 2. Then we write its lineage as:

$$F = \bigwedge_{a,b \in [n]:a<b} \psi(a,b) \wedge \bigwedge_{c \in [n]} \psi(c,c)$$

Since all atoms are binary, for any two distinct sets $\{a,b\} \neq \{a',b'\}$, the formulas $\psi(a,b)$ and $\psi(a',b')$ are independent probabilistic events, because they depend on disjoint sets of ground tuples: one depends on tuples of the form $R(a, b)$ or $R(b, a)$, the other on tuples of the form $R(a', b')$ or $R(b', a')$, and they are disjoint. (This would fail if ψ contained a unary atom, say $U(x)$, because we may have $a = a'$, $b \neq b'$, and in that case both formulas depend on the tuple $U(a)$.) Therefore:

$$\Pr(F) = \prod_{a,b \in [n]: a<b} \Pr(\psi(a, b)) \cdot \prod_{c \in [n]} \Pr(\psi(c, c))$$

Because the probabilities are symmetric, the quantity $p_1 = \Pr(\psi(a, b))$ is independent of a, b, while $p_2 = \Pr(\psi(c, c))$ is independent of c, and both can be computed in time $O(1)$. Therefore, $\Pr(\varphi) = \Pr(\varphi^*) = p(F) = p_1^{n(n-1)/2} p_2^n$. For a simple illustration, consider $\varphi^* = \forall x \forall y (R(x, y) \vee T(x, y)) \wedge (R(x, y) \vee T(y, x))$. Then:

$$\begin{aligned}F = \bigwedge_{a,b \in [n]: a<b} & (R(a, b) \vee T(a, b)) \wedge (R(a, b) \vee T(b, a)) \\ & \wedge (R(b, a) \vee T(b, a)) \wedge (R(b, a) \vee T(a, b)) \\ \wedge \bigwedge_{c \in [n]} & (R(c, c) \vee T(c, c))\end{aligned}$$

and the probability is given by $p_1^{n(n-1)/2} p_2^n$ where $p_1 = \Pr((R(a, b) \vee T(a, b)) \wedge (R(a, b) \vee T(b, a)) \wedge (R(b, a) \vee T(b, a)) \wedge (R(b, a) \vee T(a, b)))$ and $p_2 = \Pr(R(c, c) \vee T(c, c))$, both quantities that can be computed using brute force.

Next consider the case when φ^* has both unary and binary relational symbols. Let $R_1, \ldots, R_m$ be all unary symbols. Consider the 2^m cells defined by conjunctions of these atoms or their negation, denote them $C_1, \ldots, C_{2^m}$; that is $C_1(x) \equiv \neg R_1(x) \wedge \cdots \wedge \neg R_m(x)$, ..., $C_{2^m}(x) \equiv R_1(x) \wedge \cdots \wedge R_m(x)$. Let P denote any partition of $[n]$ into 2^m disjoint sets, i.e. $P = (S_1, \ldots, S_{2^m})$ such that $S_1 \cup \cdots \cup S_{2^m} = [n]$. Denote $(C_1, \ldots, C_{2^m}) = P$ the event that the 2^m cells define precisely the partition P. Then, summing over all partitions P gives us:

$$\Pr(\varphi^*) = \sum_P \Pr(\varphi^* \wedge (C_1, \cdots, C_{2^m}) = P) \tag{8}$$

Next, we split φ^* into a conjunction of several formulas, each x ranging over some cell S_i and y over some cell S_j:

$$\begin{aligned}\varphi^* = & \bigwedge_{i,j \in [2^m]: i<j} \forall x : S_i, \forall y : S_j, (\psi(x, y) \wedge \psi(y, x)) \\ & \wedge \bigwedge_{\ell \in [2^m]} \forall x : S_\ell, \forall y : S_\ell, \psi(x, y)\end{aligned}$$

When x ranges over S_i, then every unary predicate $R(x)$ containing the variable x is either true or false. Similarly, when y ranges over S_j, a predicate $R(y)$ is either true or false. Let $\psi_{ij}(x, y)$ (or $\psi_\ell(x, y)$) denote the formula $\psi(x, y) \wedge \psi(y, x)$ (or $\psi(x, y)$) where all the unary predicates have been replaced by true or false, according to the cells S_i, S_j (or S_ℓ respectively). Therefore:

$$\begin{aligned}\varphi^* = & \bigwedge_{i,j \in [2^m]: i<j} \forall x : S_i, \forall y : S_j, \psi_{ij}(x, y) \\ & \wedge \bigwedge_{\ell \in [2^m]} \forall x : S_\ell, \forall y : S_\ell, \psi_\ell(x, y)\end{aligned}$$

Notice that ψ_{ij} and ψ_ℓ have only binary predicates. All conjuncts in the expression above are independent probabilistic events: if $\{i_1, j_1\} \neq \{i_2, j_2\}$ then $\forall x : S_{i_1}, \forall y : S_{j_1}, \psi_{i_1 j_1}(x, y)$ and $\forall x : S_{i_2}, \forall y : S_{j_2}, \psi_{i_2 j_2}(x, y)$ are independent. Therefore, denoting $n_i = |S_i|$ for $i = 1, 2^m$, we have: $\Pr(\varphi^* \wedge (C_1, \cdots, C_{2^m}) = P) = \prod_{i,j \in [2^m]: i<j} q_{ij} \cdot \prod_{\ell \in [2^m} r_\ell$, where:

$$\begin{aligned}q_{ij} =& \Pr(\forall x : S_i, \forall y : S_j, \psi_{ij}(x, y)) \\ =& \prod_{a \in S_i, b \in S_j} \Pr(\psi_{ij}(a, b)) = r_{ij}^{n_i n_j} \\ r_\ell =& \Pr(\forall x : S_\ell, \forall y : S_\ell, \psi_\ell(x, y)) \\ =& \prod_{a,b \in S_\ell: a<b} \Pr(\psi_\ell(a, b) \wedge \psi_\ell(b, a)) \cdot \prod_{c \in S_\ell} \Pr(\psi_\ell(c, c)) \\ =& s_\ell^{n_\ell * (n_\ell - 1)/2} \cdot t_\ell^{n_\ell}\end{aligned}$$

where $r_{ij} = \Pr(\psi_{ij}(a, b))$, $s_\ell = \Pr(\psi_\ell(a, b) \wedge \psi_\ell(b, a))$, and $t_\ell = \Pr(\psi_\ell(c, c))$ are independent of the choices of a, b, c respectively, and can be computed by brute force in time $O(1)$. Finally, we use the fact that the probabilities are symmetric, which implies that the expression in Eq. (8) depends only on the cell cardinalities $n_1, \ldots, n_{2^m}$, and not on the actual cells $S_1, \ldots, S_{2^m}$. Therefore:

$$\begin{aligned}\Pr(\varphi^*) = & \sum_{n_1, \ldots, n_{2^m}: n_1 + \cdots + n_{2^m} = 1} \frac{n!}{n_1! \cdots n_{2^m}!} \\ & \cdot r_{ij}^{n_i n_j} s_\ell^{n_\ell * (n_\ell - 1)/2} \cdot t_\ell^{n_\ell}\end{aligned}$$

For a simple illustration, consider $\varphi^* = \forall x \forall y (R(x) \vee U(x, y) \vee T(y)) \wedge (\neg R(x) \vee \neg U(x, y) \vee \neg T(y))$. Denoting the four cells $\neg R \wedge \neg T$, $\neg R \wedge T$, $R \wedge \neg T$, $R \wedge T$ by $C_1, \ldots C_4$ respectively, we split φ into a conjunct of $6 + 4$ expressions, such that in each expression x and y are restricted to the domains C_i and C_j respectively, for $i \leq j$. Denoting $n_1, \ldots, n_4$ the sizes of these cells, we have:

$$\Pr(\varphi^*) = \sum_{n_1 + \cdots + n_4 = n} \frac{n!}{n_1! n_2! n_3! n_4!} r_{ij}^{n_i n_j} s_\ell^{n_\ell * (n_\ell - 1)/2} \cdot t_\ell^{n_\ell}$$

where $r_{12} = \Pr(U(a, b))$ (because $\forall x : S_1, \forall y : S_2, (R(x) \vee U(x, y) \vee T(y)) \wedge (\neg R(x) \vee \neg U(x, y) \vee \neg T(y)) \equiv \forall x : S_1, \forall y : S_2, T(x, y))$, and similarly for the others.