Journal of Artificial Intelligence Research 70 (2021) 1281-1307 Submitted 07/2020; published 03/2021

Weighted First-Order Model Counting in the Two-Variable Fragment With Counting Quantifiers

Ondřej Kuželka ONDREJ.KUZELKA@FEL.CVUT.CZ
Faculty of Electrical Engineering
Czech Technical University in Prague
Prague, Czech Republic

Abstract

It is known due to the work of Van den Broeck, Meert and Darwiche that weighted first-order model counting (WFOMC) in the two-variable fragment of first-order logic can be solved in time polynomial in the number of domain elements. In this paper we extend this result to the two-variable fragment with counting quantifiers.

1. Introduction

In this paper we study *weighted first-order model counting* (WFOMC), which is an important problem (not only) because it can be used for probabilistic inference in most statistical relational learning models (Van den Broeck et al., 2011; Getoor & Taskar, 2007). Probabilistic inference is in general intractable and the same holds for probabilistic inference in relational domains and therefore also for WFOMC. *Lifted inference* refers to a set of methods developed in the probabilistic inference literature which exploit structure and symmetries of the problems for making inference more tractable (e.g., Poole, 2003; de Salvo Braz, Amir, and Roth, 2005; Gogate and Domingos, 2011; Van den Broeck, 2011; Van den Broeck, Meert, and Darwiche, 2014; Kazemi et al., 2016). One of the most celebrated results on symmetric WFOMC comes from the works by Van den Broeck and his colleagues (Van den Broeck, 2011; Van den Broeck et al., 2014) which established that WFOMC can be solved in polynomial time for any fixed first-order logic sentence which contains at most two variables. In the lifted inference literature, problems which admit such polynomial-time algorithms are called *domain-liftable* (Van den Broeck, 2011).[1]

Kuusisto and Lutz (2018) recently extended the domain-liftability result for the two-variable fragment by allowing to add one *functionality* constraint. That is, one can specify that some binary relation should behave as a function. In their paper, they also mentioned (although without giving any details) that if they could further extend their result to multiple funcionality constraints, they could also establish domain liftability of the two-variable fragment of first-order logic with counting quantifiers $\exists_{=k}$, $\exists_{\leq k}$ and $\exists_{\geq k}$ (see, e.g., the work by Graedel, Otto, and Rosen, 1997), which stand for *exist exactly k*, *exist at most k* and *exist at least k*, respectively. Motivated by the work of Kuusisto and Lutz, in this paper, we first show a simpler method to add an arbitrary number of *functionality* constraints and *cardinality* constraints to sentences from the two-variable fragment while still guaranteeing

1. Like, among others, Van den Broeck (2011), Van den Broeck et al. (2014), Kazemi et al. (2016), Kuusisto and Lutz (2018), we also consider only the *symmetric* version of weighted first-order model counting. For details and differences with the asymmetric version, we refer to the work of Beame et al. (2015).

polynomial-time inference. We then use this result to prove that WFOMC is domain-liftable for sentences from the two-variable fragment of first-order logic with counting quantifiers.

The rest of the paper is structured as follows. Section 2 contains the background material needed for our technical results. In Sections 3-6, we work towards the proof of our main result which we give in Section 7. We discuss related work in Section 8 and conclude the paper in Section 9. The appendix located at the end of the paper then contains omitted proofs and additional examples as well as some additional technical material that is not needed for the main result but which further illustrates some of our techniques.

2. Background

In this section we describe the necessary technical background material.

2.1 Lagrange Interpolation

Lagrange interpolation (e.g., Seroul, 2000) is a classical method for finding the *unique* polynomial $p(x)$ of degree d that, for given $d+1$ points $(x_0, y_0), (x_1, y_1), \ldots, (x_d, y_d)$ satisfies $p(x_0) = y_0$, $p(x_1) = y_1$, $\ldots$, $p(x_d) = y_d$ (under the condition that $x_i \neq x_j$ for all $i \neq j$). There are various methods of finding the coefficients of the polynomial (e.g. based on special algorithms for solving systems of linear equations with Vandermonde matrices), in this paper it will be enough to consider the elementary method based on the explicit Lagrange formula:

$$L(x) = \sum_{i=0}^{d} y_i \cdot l_i(x)$$

where l_i is defined as

$$l_i(x) = \prod_{\substack{0 \leq j \leq d \\ i \neq j}} \frac{x - x_j}{x_i - x_j}.$$

The next proposition shows a useful property of "bit complexity" of the coefficients of such interpolating polynomials that will be useful later in this paper (the proof of this proposition is located in the appendix).

Proposition 1. *Let $L(x) = \sum_{j=0}^{d} a_j \cdot x^j$ be the interpolating polynomial (written in the standard form as a sum of monomials) of points (x_0, y_0), (x_1, y_1), $\ldots$, (x_d, y_d), where all x_j's are integers and all y_j's are rational numbers, represented as fractions of integers. Every a_j can be represented as a fraction $a_j = \frac{b_j}{c_j}$ and the number of bits needed to represent the integers b_j and c_j is polynomial in d and in the number of bits needed to represent the points (x_0, y_0), (x_1, y_1), $\ldots$, (x_d, y_d).*

2.2 First-Order Logic

We assume that the reader is familiar with first-order logic and we only cover it briefly in this section to set up notation used throughout the paper.

We work with function-free first-order logic languages $\mathcal{L}$, defined by a set of constants, called domain and usually denoted as Δ, a set of variables $\mathcal{V}$ and a set of predicates $\mathcal{R}$

(relations). When there is no risk of confusion, we assume such a language implicitly and do not specify its components $\mathcal{V}$, $\mathcal{R}$ (although we will usually specify the domain). We use *arity*(R) to denote the arity of a predicate R. An expression of the form $r(a_1, ..., a_k)$, with $a_1, ..., a_k \in \Delta \cup \mathcal{V}$ and $r \in \mathcal{R}$, is called an *atom* or *atomic formula*. For example, $sm(x)$, $sm(Alice)$ and $fr(Alice, y)$ are atoms. A variable which is not bound by any quantifier is called *free*. A first-order logic formula with no free variables is called a sentence. For instance, the formula $\forall x : \neg f(x, x)$ is a sentence, whereas the formula $f(x, x)$ is not a sentence as the variable x is free in it. A first-order logic formula in which none of the atoms contains any variables is called *ground*. A possible world ω is represented as a set of ground atoms that are true in ω. The satisfaction relation $\models$ is defined in the usual way: $\omega \models \alpha$ means that the formula α is true in ω. For instance, if $\omega = \{sm(Bob)\}$ is a possible world on the domain $\Delta = \{Alice, Bob\}$ then it holds $\omega \models (\exists x : sm(x))$ and $\omega \not\models (\forall x : sm(x))$.

The two-variable fragment of first-order logic (FO^2) is obtained by restricting the set of variables to $\mathcal{V} = \{x, y\}$. For example, the sentence $\forall x \forall y : a(x) \wedge e(x, y) \Rightarrow a(y)$ is in FO^2. The fragment of first-order logic FO^2 is interesting among others because (i) satisfiability is decidable for it (in particular it is NEXPTIME-complete) and (ii) weighted first-order model counting is polynomial-time (in the size of the domain) for any sentence from FO^2 (Van den Broeck, 2011; Van den Broeck et al., 2014).

2.2.1 First-Order Logic With Counting Quantifiers

An interesting extension of the 2-variable fragment of first order logic is obtained by adding *counting quantifiers* $\exists_{=k}$, $\exists_{\leq k}$ and $\exists_{\geq k}$ to it (Graedel et al., 1997). Satisfiability in this fragment of first-order logic is still decidable, although this fragment lacks the finite-model property that FO^2 enjoys.

The counting quantifiers can be introduced as follows. Let ω be a possible world defined on a domain Δ. The sentence $\exists_{\geq k} x : \psi(x)$ is true in ω if there are at least k distinct elements $t_1, \ldots, t_k \in \Delta$ such that $\omega \models \psi(t_i)$. The other two counting quantifiers can be defined using: $(\exists_{\leq k} x : \psi(x)) \Leftrightarrow \neg(\exists_{\geq k+1} \psi(x))$ and $(\exists_{=k} x : \psi(x)) \Leftrightarrow (\exists_{\leq k} : \psi(x) \wedge \exists_{\geq k} : \psi(x))$.

Example 1. To give an example of the expressive power of FO^2 with counting quantifiers, we can notice that it is easy to constrain binary relations to be functions using it. In all models of the sentence $\forall x \exists_{=1} y : f(x, y)$, f is a function from the domain to itself. Additionally, if we wanted to force f to be a bijection, we could use $(\forall x \exists_{=1} y : f(x, y)) \wedge (\forall y \exists_{=1} x : f(x, y))$ etc.

2.3 Weighted First-Order Model Counting

In this section we formally describe *weighted first-order model counting*. We start by defining an auxiliary concept, *cardinality of a relation*.

Definition 1 (Cardinality of Relation)**.** *Let ω be a possible world and R be a k-ary predicate. The cardinality of R in ω is defined as*

$$N(R, \omega) = |\{R(x_1, \ldots, x_k) \in \omega\}|,$$

i.e. $N(R, \omega)$ is the number of ground atoms of the predicate R that are true in ω.

Example 2. Let $\omega = \{fr(Alice, Bob), fr(Alice, Eve), sm(Alice)\}$. Then

$$N(fr, \omega) = |\{fr(Alice, Bob), fr(Alice, Eve)\}| = 2.$$

Next we define weighted first-order model counting.

Definition 2 (WFOMC, Van den Broeck, 2011)**.** *Let Ω be a set of possible worlds over a given domain Δ (Ω will often be the set of all possible worlds over Δ), $\mathcal{R}$ be the set of predicates in the language, $w(P)$ and $\overline{w}(P)$ be functions from predicates to complex[2] numbers (we call w and $\overline{w}$* weight functions*). Then for a given first-order logic sentence Γ, we define*

$$\mathrm{WFOMC}(\Gamma, w, \overline{w}, \Omega) = \sum_{\omega \in \Omega : \omega \models \Gamma} \prod_{R \in \mathcal{R}} w(R)^{N(R,\omega)} \cdot \overline{w}(R)^{|\Delta|^{arity(R)} - N(R,\omega)}.$$

We also define

$$\mathrm{WFOMC}(\Gamma, w, \overline{w}, \Delta) \stackrel{def}{=} \mathrm{WFOMC}(\Gamma, w, \overline{w}, \Omega_\Delta),$$

where Ω_Δ is the set of all possible worlds on the domain Δ (using the predicates from $\mathcal{R}$).

In this paper, when we do not explicitly define weights of some predicate R, we will assume that $w(R) = \overline{w}(R) = 1$.

Next we illustrate WFOMC on a small example. Additionally, we show how WFOMC can be used for inference in Markov logic networks in Section 2.4.1.

Example 3. Let $\Delta = \{A, B\}$, $\mathcal{R} = \{heads, tails\}$, $w(heads) = 2$, $w(tails) = \overline{w}(heads) = \overline{w}(tails) = 1$, and $\Gamma = \forall x : (heads(x) \vee tails(x)) \wedge (\neg heads(x) \vee \neg tails(x))$. There are four models of Γ on the domain Δ: $\omega_1 = \{heads(A), heads(B)\}$, $\omega_2 = \{heads(A), tails(B)\}$, $\omega_3 = \{tails(A), heads(A)\}$ and $\omega_4 = \{tails(A), tails(B)\}$. The resulting weighted model count is $\mathrm{WFOMC}(\Gamma, w, \overline{w}, \Delta) = 4 + 2 + 2 + 1 = 9$.

2.3.1 Two Useful Technical Results About WFOMC

We now describe two useful technical properties of WFOMC. The first of these is about bit complexity of WFOMC. Later in the paper, we will need to be able to bound the bit complexity of WFOMC and the next proposition does exactly that (its proof is located in the appendix).

Proposition 2. *Let Γ be a first-order logic sentence, $\mathcal{R} = \{R_1, R_2, \ldots, R_m\}$ be the set of predicates from a given first-order language, Δ be a domain and Ω_Δ be the set of all possible worlds on the domain Δ using the predicates from $\mathcal{R}$. Let w and $\overline{w}$ be weight functions that assign to each predicate $R \in \mathcal{R}$ a rational number $w(R) = w'(R)/w''(R)$ and $\overline{w}(R) = \overline{w}'(R)/\overline{w}''(R)$, where $w'(R)$, $w''(R)$, $\overline{w}'(R)$ and $\overline{w}''(R)$ are integers. Let us further define $M = \max_R \max\{|w'(R)|, |w''(R)|, |\overline{w}'(R)|, |\overline{w}''(R)|\}$. Then $\mathrm{WFOMC}(\Gamma, w, \overline{w}, \Delta)$ can be represented as a rational number a/b and the number of bits needed to encode the integers a and b is bounded by a polynomial in $|\Delta|$ and $\log M$.*

2. Normally, in the literature, the weights of predicates are real numbers. However, we will also use complex-valued weights in this paper, therefore we define the WFOMC problem accordingly using complex-valued weights.

At some point in the paper, we will also need to replace certain subformulas by their negations, without actually using negation. This is possible using a technique described by Beame et al. (2015), stated in Appendix A.2 of their paper, which we restate in the proposition below.[3]

Proposition 3. *Let $\neg\psi(x_1, \ldots, x_k)$ be a subformula of a first-order logic sentence Φ with k free variables $x_1, \ldots, x_k$. Let C, D be two new predicates of arity k. Let Φ' denote the sentence obtained from Φ by replacing the subformula $\neg\psi(x_1, \ldots, x_k)$ with $C(x_1, \ldots, x_k)$. Let*

$$\begin{aligned}\Upsilon = \forall x_1 \forall x_2 \ldots \forall x_k : ((\psi(x_1, \ldots, x_k) \vee C(x_1, \ldots, x_k)) \\ \wedge (C(x_1, \ldots, x_k) \vee D(x_1, \ldots, x_k)) \wedge (\psi(x_1, \ldots, x_k) \vee D(x_1, \ldots, x_k)))\end{aligned}$$

and extend the given weight functions w and $\overline{w}$ by defining $w(C) = \overline{w}(C) = w(D) = 1$ and $\overline{w}(D) = -1$. Then it holds

$$\mathrm{WFOMC}(\Phi, w, \overline{w}, \Omega) = \mathrm{WFOMC}(\Phi' \wedge \Upsilon, w, \overline{w}, \Omega_{ext})$$

where Ω is the set of all possible worlds on the domain Δ w.r.t. a given first-order logic language $\mathcal{L}$ and Ω_{ext} is the set of all possible worlds on the domain Δ w.r.t. $\mathcal{L}$ extended by predicates C and D.

2.4 Domain-Lifted Inference

Importantly, there are classes of first-order logic sentences for which weighted model counting can be solved in polynomial-time. In particular, let Ω be the set of all possible worlds over a given domain Δ and a given set of relations $\mathcal{R}$. As shown by Van den Broeck et al. (2014), when the theory Γ consists only of first-order logic sentences, each of which contains at most two logic variables and uses the classical quantifiers $\exists$ and $\forall$ (i.e. the FO^2 fragment), the weighted model count can be computed in time polynomial in the size of the domain Δ. This is not the case in general when the number of variables in the formulas is greater than two unless $\mathrm{P} = \#\mathrm{P}_1$ (Beame et al., 2015).[4] Within statistical relational learning, the term used for problems that have such polynomial-time algorithms is *domain liftability*.

Definition 3 (Domain liftability)**.** *An algorithm for computing WFOMC with rational weights is said to be domain-liftable if it runs in time polynomial in the size of the domain and the number of bits needed to represent the weights.*

The definition of domain liftability presented here differs slightly from the original definition by Van den Broeck (2011) in that it also requires lifted algorithms to depend polynomially on the size of the representation of the formulas' weights. A justification for this definition follows from the work of Jaeger (2015), Section 4.2. In particular, as pointed out by Jaeger, all existing domain-lifted exact-inference algorithms are also domain-lifted according to the definition that we use here.

3. The same transformation also appears in the literature under the name "relaxed Tseitin transform" (Meert, Vlasselaer, & Van den Broeck, 2016).
4. $\#\mathrm{P}_1$ is the set of #P problems over a unary alphabet.

2.4.1 An Application of WFOMC: Inference in Markov Logic Networks

A Markov logic network (MLN, Richardson and Domingos, 2006) is a set of weighted first-order logic formulas (α, w), where $w \in \mathbb{R}$ and α is a function-free first-order logic formula. The semantics are defined w.r.t. the groundings of the first-order logic formulas, relative to some finite set of constants Δ, called *domain*. An MLN Φ induces the probability distribution on possible worlds $\omega \in \Omega$ over a given domain:

$$P_\Phi(\omega) = \frac{1}{Z} \exp\left(\sum_{(\alpha,w)\in\Phi} w \cdot n(\alpha,\omega)\right), \tag{1}$$

where $n(\alpha,\omega)$ is the number of groundings of α satisfied in ω (when α does not contain any variables, we define $n(\alpha,\omega) = \mathbb{1}(\omega \models \alpha)$), and Z, called *partition function*, is a normalization constant to ensure that P_Φ is a probability distribution. We also allow infinite weights. A weighted formula of the form $(\alpha, +\infty)$ is understood as a hard constraint imposing that all worlds ω in which $n(\alpha,\omega)$ is not maximal have zero probability (this can also be deduced by taking the limit $w \to +\infty$). If all formulas in an MLN have at most k variables, we call such an MLN *k-variable*.

Computation of the partition function Z of an MLN can be converted to WFOMC. To compute the partition function Z using weighted model counting, we proceed as Van den Broeck et al. (2011). Let an MLN $\Phi = \{(\alpha_1, w_1), \ldots, (\alpha_m, w_m)\}$ over a set of possible worlds Ω be given. For every $(\alpha_j, w_j) \in \Phi$, where the free variables in α_j are exactly $x_1, \ldots, x_k$ and where $w \neq +\infty$, we create a new formula $\forall x_1, \ldots, x_k : \xi_j(x_1, \ldots, x_k) \Leftrightarrow \alpha_j(x_1, \ldots, x_k)$ where ξ_j is a new fresh predicate. When $w = +\infty$, we instead create a new formula $\forall x_1, \ldots, x_k : \alpha_j(x_1, \ldots, x_k)$. We denote the resulting set of new formulas Γ. Then we set $w(\xi_j) = \exp(w_j)$ and $\overline{w}(\xi_j) = 1$ and for all other predicates we set both w and $\overline{w}$ equal to 1. It is easy to check that then $\mathrm{WFOMC}(\Gamma, w, \overline{w}, \Omega) = Z$, which is what we needed to compute. To compute the marginal probability of a given first-order logic sentence γ, we have $P_\Phi[X \models q] = \frac{\mathrm{WFOMC}(\Gamma\cup\{q\}, w, \overline{w}, \Omega)}{\mathrm{WFOMC}(\Gamma, w, \overline{w}, \Omega)}$ where X is sampled from the MLN.

For more examples of applications of weighted first-order model counting to statistical relational learning problems, we refer to the work of Van den Broeck (2013).

3. Weighted Model-Counting Functions

Before getting to the *weighted model-counting functions*, we need to define notation for vectors of "relation-cardinalities". For a given possible world ω and a given list of predicates $\Psi = (R_1, R_2, ..., R_m)$, we define the respective vector of relation-cardinalities as

$$\mathbf{N}(\Psi, \omega) \stackrel{def}{=} (n_1, \ldots, n_m),$$

where $n_i = N(R_i, \omega)$ is the cardinality of the relation R_i in ω, i.e. the number of ground atoms of the form $R_i(c_1, \ldots, c_{\mathrm{arity}(R_i)})$ that are true in ω.

Example 4. Let $\omega = \{sm(Alice), sm(Bob), fr(Alice, Bob)\}$ and $\Psi = (sm, fr)$. Then the vector of relation-cardinalities is $\mathbf{N}(\Psi, \omega) = (2, 1)$.

Next we define *model-counting function* (which we will also call *MC-function*).

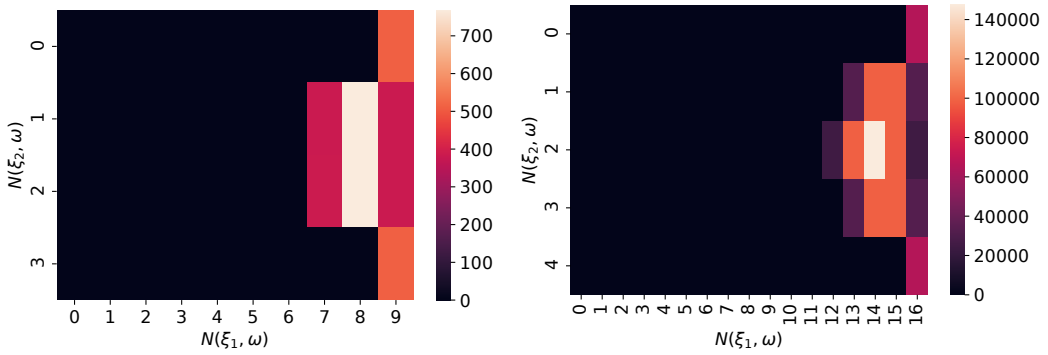

Figure 1: Model-counting functions from Example 6.

Definition 4 (Model-Counting Function)**.** *Let Ω be a set of possible worlds and let $\Psi = (R_1, R_2, \ldots, R_m)$ be a list of predicates. We define the model counting function as*

$$\mathrm{MC}_{\Psi,\Omega}(\mathbf{n}) = |\{\omega \in \Omega | \mathbf{N}(\Psi, \omega) = \mathbf{n}\}|,$$

where $\mathbf{n}$ is a vector of integers. Given first-order logic sentence Γ and a domain Δ we also define

$$\mathrm{MC}_{\Psi,\Gamma,\Delta}(\mathbf{n}) \stackrel{def}{=} \mathrm{MC}_{\Psi,\Omega_{\Gamma,\Delta}}(\mathbf{n}),$$

where $\Omega_{\Gamma,\Delta}$ is the set of models of the sentence Γ on the domain Δ (assuming some given first-order language that also specifies the predicates).

Intuitively, for any $\mathbf{n} \in \mathbb{N}^m$, the model counting function gives us the number of possible worlds (from the given set Ω) that satisfy $\mathbf{N}(\Psi, \omega) = \mathbf{n}$.

Example 5. Let us consider the domain $\Delta = \{A, B, C, D\}$, the sentence $\Gamma = \forall x : (\mathit{heads}(x) \vee \mathit{tails}(x)) \wedge (\neg \mathit{heads}(x) \vee \neg \mathit{tails}(x))$ and the list $\Psi = (\mathit{heads})$. We assume that the first-order language over which the possible worlds are defined contains only the predicates *heads* and *tails*. Then we have, for instance, $\mathbf{MC}_{\Psi,\Gamma,\Delta}(0) = 1$ (since there is only one model of Γ with 0 *heads* and that is $\{\mathit{tails}(A), \mathit{tails}(B), \mathit{tails}(C), \mathit{tails}(D)\}$) and $\mathbf{MC}_{\Psi,\Gamma,\Delta}(1) = 4$ and so on.

Example 6. In Figure 1, we show examples of two MC-functions, $\mathrm{MC}_{\Psi,\Gamma,\Delta}(\mathbf{n})$, where $\Psi = (\xi_1, \xi_2)$, $\Gamma = (\forall x \forall y : \xi_1(x, y) \Leftrightarrow (\mathit{sm}(x) \wedge \mathit{fr}(x, y) \Rightarrow \mathit{sm}(y))) \wedge (\forall x : \xi_2(x) \Leftrightarrow \mathit{sm}(x))$, and Δ is a domain of size 3 and 4, respectively. Note that the form of the sentence Γ corresponds to the encoding of an MLN with two formulas $\alpha = \mathit{sm}(x) \wedge \mathit{fr}(x, y) \Rightarrow \mathit{sm}(y)$ and $\beta = \mathit{sm}(x)$ (cf. Section 2.4.1).

The concept of model-counting function can be straightforwardly generalized to weighted model-counting functions that we define next. Weighted model-counting functions are the main "work-horses" that we use in the rest of the paper.

Definition 5 (Weighted Model-Counting Function)**.** *Let Ω be a set of possible worlds and let $\Psi = (R_1, R_2, \ldots, R_m)$ be a list of predicates. We define the model counting function as:*

$$\mathrm{WMC}_{\Psi,\Omega}(\mathbf{n}, w, \overline{w}) = \mathrm{WFOMC}(\top, w, \overline{w}, \{\omega \in \Omega | \mathbf{N}(\Psi, \omega) = \mathbf{n}\}),$$

where $\top$ (tautology) is the trivial sentence which is always true. Given a first-order logic sentence Γ and a domain Δ, we also define

$$\mathrm{WMC}_{\Psi,\Gamma,\Delta}(\mathbf{n}, w, \overline{w}) \stackrel{def}{=} \mathrm{WMC}_{\Psi,\Omega_{\Gamma,\Delta}}(\mathbf{n}, w, \overline{w}),$$

where $\Omega_{\Gamma,\Delta}$ is the set of models of the sentence Γ on the domain Δ (assuming some given first-order language that specifies the predicates).

Clearly, model-counting functions are a special case of weighted model counting functions for $w \equiv 1$ and $\overline{w} \equiv 1$.

In the next subsection we explain how to compute weighted model-counting functions using a WFOMC oracle.

3.1 Computing Weighted Model-Counting Functions

At first it may not be obvious how to compute weighted model-counting functions efficiently. In our previous work (Kuzelka, 2020), we described a method based on discrete Fourier transform that can be used for computing weighted model-counting functions (we describe this method in the appendix). A downside of this method is that it requires computing WFOMC over complex numbers. Even though existing lifted inference algorithms, which normally only count over real-valued weights, can be straightforwardly modified to also allow complex numbers, it would be nicer if we could do the same without modifying them. In this section, we show that it is possible to compute weighted model-counting functions efficiently also without WFOMC over complex numbers. In particular, we prove the following proposition.

Proposition 4. *Let Δ be a set of domain elements, Γ be a first-order logic sentence and $\Psi = (R_1, \ldots, R_m)$ be a list of relations. If $\mathrm{WFOMC}(\Gamma, w, \overline{w}, \Delta)$ can be computed in time polynomial in $|\Delta|$ and in the number of bits needed to encode w and $\overline{w}$ then the corresponding WMC-function $\mathrm{WMC}_{\Psi,\Gamma,\Delta}(\mathbf{n}, w, \overline{w})$ can also be computed in time polynomial in $|\Delta|$ and in the number of bits needed to encode w and $\overline{w}$.*

Proof. We prove this proposition by showing how to compute weighted model-counting functions using Lagrange interpolation when we have access to a WFOMC oracle.

Let w^* and $\overline{w}^*$ be weight functions defined by: $w^*(R) = \overline{w}^*(R) = 1$ for all $R \in \Psi$ and $w^*(R) = w(R)$ and $\overline{w}^*(R) = \overline{w}(R)$ for all the other predicates $R \notin \Psi$. Then we can write:

$$\mathrm{WFOMC}(\Gamma, w, \overline{w}^*, \Delta) = \sum_{\mathbf{n} \in \mathcal{D}} \mathrm{WMC}_{\Psi,\Gamma,\Delta}(\mathbf{n}, w^*, \overline{w}^*) \cdot \prod_{i=1}^{m} w(R_i)^{n_i}$$

where n_i denotes the i-th component of $\mathbf{n}$,

$$\mathcal{D} = \{0, 1, \ldots, M_1\} \times \{0, 1, \ldots, M_2\} \times \cdots \times \{0, 1, \ldots, M_m\},$$

and $M_1 = |\Delta|^{arity(R_1)}$, $M_2 = |\Delta|^{arity(R_2)}$, $\ldots$, $M_m = |\Delta|^{arity(R_m)}$.

It follows that, when we fix all weights of all predicates $R \notin \Psi$ (i.e. if we keep them constant), $\mathrm{WFOMC}(\Gamma, w, \overline{w}^*, \Delta)$ becomes a polynomial in the weights $w(R_1), \ldots, w(R_m)$. Let us denote this polynomial as $W(w_1, \ldots, w_m)$. Importantly, $\mathrm{WMC}_{\Psi,\Gamma,\Delta}(\mathbf{n}, w^*, \overline{w}^*)$ is the

coefficient of $\prod_{i=1}^{m} w(R_i)^{n_i}$ in this polynomial, where n_i denotes the i-th component of $\mathbf{n}$. As we will show shortly, we can efficiently compute the coefficients of the monomials using a WFOMC oracle, which means we will be able to efficiently compute WMC from a WFOMC oracle. For that we first introduce another, univariate, polynomial:

$$W_0(t) \stackrel{def}{=} W\left(t, t^{M_1+1}, t^{(M_1+1)(M_2+1)}, \ldots, t^{(\ldots(M_1+1)(M_2+1))\ldots)(M_{m-1}+1)}\right).$$

The polynomial W_0 has the convenient property that the coefficient of

$$t^{n_1+(M_1+1)n_2+\cdots+(\ldots(M_1+1)(M_2+1))\ldots)n_m}$$

is equal to the coefficient of $w_1^{n_1} w_2^{n_2} \ldots w_m^{n_m}$ in $W(w_1, w_2, \ldots, w_m)$. Let us denote this coefficient by $A_{n_1,\ldots,n_m}$. It follows that $A_{n_1,\ldots,n_m}$ is also equal to $\mathrm{WMC}_{\Psi,\Gamma,\Delta}(\mathbf{n}, w^*, \overline{w}^*)$, from which we can then obtain

$$\begin{aligned}\mathrm{WMC}_{\Psi,\Gamma,\Delta}(\mathbf{n}, w, \overline{w}) &= \mathrm{WMC}_{\Psi,\Gamma,\Delta}(\mathbf{n}, w^*, \overline{w}^*) \cdot \prod_{i=1}^{m} w(R_i)^{n_i} \cdot \overline{w}(R)^{|\Delta|^{arity(R_i)}-n_i} \\ &= A_{n_1,\ldots,n_m} \cdot \prod_{i=1}^{m} w(R_i)^{n_i} \cdot \overline{w}(R)^{|\Delta|^{arity(R_i)}-n_i}.\end{aligned}$$

The only missing part is to show that we can extract the coefficient $A_{n_1,\ldots,n_m}$ efficiently using a WFOMC oracle. This can be done using Lagrange interpolation (cf. Section 2.1). First, we define $|\mathcal{D}|+1$ points (x_i, y_i) for the polynomial interpolation problem: $(1, W_0(1))$, $(2, W_0(2))$, …, $(|\mathcal{D}|+1, W_0(|\mathcal{D}|+1))$. These points are computed using a WFOMC oracle. Using Proposition 2, we have that the number of bits needed to encode each of $W_0(1)$, $W_0(2)$, …, $W_0(|\mathcal{D}|+1)$ is polynomial in $|\mathcal{D}|$. Combining that with Proposition 1, we then have that the number of bits needed to encode the coefficients of the polynomial interpolating the points $(1, W_0(1))$, $(2, W_0(2))$, …, $(|\mathcal{D}|+1, W_0(|\mathcal{D}|+1))$ grows only polynomially with $|\mathcal{D}|$. Since $|\mathcal{D}|$ is itself bounded by a polynomial in the size of the domain $|\Delta|$, it follows that we can extract the coefficients and consequently the WMC-function that we want to compute in time polynomial in $|\Delta|$ and the number of bits needed to encode w and $\overline{w}$.

□

Remark 1. *We could do a bit better in terms of practical efficiency than the construction from the above proof if we replaced the univariate Lagrange interpolation by its multivariate version. We could use, e.g., Lemma 5 from the work of Koiran and Perifel (2011). Then we would only need to evaluate WFOMC on weights from the set* $\{0, 1, 2, \ldots, |\mathcal{D}|\}$. *For simplicity, in the proof of the above proposition we opted for the more elementary approach, which is enough for our purposes.*

4. WFOMC with Cardinality Constraints

In this paper, a *simple cardinality constraint* is an expression of the form $|R| \in \mathcal{A}$ where R is a predicate and $\mathcal{A} \subseteq \mathbb{N}$. A possible world ω satisfies a given cardinality constraint $|R| \in \mathcal{A}$ if $N(R, \omega) \in \mathcal{A}$, i.e. if the number of ground atoms $R_i(t_1, \ldots, t_n)$ that are true in ω is in $\mathcal{A}$.

We write $\omega \models (|R| \in \mathcal{A})$ when the cardinality constraint $|R| \in \mathcal{A}$ is satisfied in ω. We will also use the notation $|R| \bowtie k$, where $\bowtie \in \{=, \leq, \geq, <, >\}$ and $k \in \mathbb{N}$. So, e.g., $|R| \leq k$ is a short for $|R| \in \{0, 1, \ldots, k\}$. Finally, we allow cardinality constraints as atomic formulas in first-order logic formulas. For instance, $(|f| = 2) \wedge (\forall x \forall y : f(x, y) \Rightarrow f(y, x))$ is a valid formula (its models can be interpreted as undirected graphs with exactly one edge) and the satisfaction relation $\models$ is extended naturally.

Computing WFOMC with cardinality constraints can be done using WMC-functions. Moreover as the next proposition shows, domain-liftability is preserved when we add cardinality constraints to a sentence which is domain-liftable.

Proposition 5. *Let Γ be a first-order logic sentence, $\psi(x_1, \ldots, x_m)$ be a propositional logic formula and let $\Upsilon = \psi(|R_{i_1}| \bowtie k_1, \ldots, |R_{i_m}| \bowtie k_m)$, where ψ is a Boolean formula and $\bowtie \in \{=, \leq, \geq, <, >\}$. If computing the WFOMC of Γ is domain-liftable then computing the WFOMC of $\Gamma \wedge \Upsilon$ is also domain-liftable.*

Proof. Let $\Psi = (R_{i_1}, \ldots, R_{i_m})$. Since computing $\mathrm{WFOMC}(\Gamma, w, \overline{w}, \Delta)$ is domain-liftable, so is computing $\mathrm{WMC}_{\Psi,\Gamma,\Delta}(\mathbf{n})$, which follows from Proposition 4. In addition we only need to evaluate the WMC-function on a set of polynomially-many (in $|\Delta|$) points, specifically on the set $\mathcal{D} = \{0, 1, \ldots, M_1\} \times \{0, 1, \ldots, M_2\} \times \cdots \times \{0, 1, \ldots, M_m\}$ where $M_1 = |\Delta|^{arity(R_{i_1})}$, $M_2 = |\Delta|^{arity(R_{i_2})}$, $\ldots$, $M_m = |\Delta|^{arity(R_{i_m})}$. Finally, we can compute $\mathrm{WFOMC}(\Gamma \wedge \Upsilon, w, \overline{w}, \Delta)$ as:

$$\mathrm{WFOMC}(\Gamma, w, \overline{w}, \Delta) = \sum_{\mathbf{n} \in \mathcal{D}} \mathbb{1}\left(\psi(|R_{i_1}| \bowtie k_1, \ldots, |R_{i_m}| \bowtie k_m)\right) \cdot \mathrm{WMC}_{\Psi,\Gamma,\Delta}(\mathbf{n})$$

where n_i denotes the i-th component of the vector $\mathbf{n}$. Hence, $\mathrm{WFOMC}(\Gamma \wedge \Upsilon, w, \overline{w}, \Delta)$ can be computed in time polynomial in the size of the domain which finishes the proof. □

Remark 2. *The techniques from this section do not apply only to encoding of cardinality constraints. It is easy to replace the Boolean formula ψ by a function to rational numbers and show that one can efficiently compute weighted first-order model counts with "non-multiplicative" weight functions, i.e. with weight functions that only depend on $\mathbf{N}(\Psi, \omega)$. We will not need this more general setting in this paper.*

5. WFOMC with Functionality Constraints[5]

A *functionality constraint* is a constraint expressed by a first-order-logic sentence of the form

$$\forall x \exists_{=1} y : \psi(x, y),$$

which asserts that for every x there is exactly one y such that $\psi(x, y)$ is true. In this section we show how to compute WFOMC of a 2-variable first-order logic sentence with an

5. In the next section, we generalize the results presented here to allow constraints of the form $\forall x \exists_{=k} y : \psi(x, y)$, of which $\forall x \exists_{=1} y : \psi(x, y)$ is a special case. However, the case of functionality constraints, besides being easier to understand, is important on its own and has been studied in the literature (Kuusisto & Lutz, 2018). We note that whereas Kuusisto and Lutz (2018) only allowed one functionality constraint, here we already allow multiple functionality constraints.

arbitrary number of functionality constraints while still guaranteeing runtime polynomial in the domain size $|\Delta|$.

First, we can notice that we can replace any functional constraint of the form $\forall x \exists_{=1} y : \psi(x, y)$, where $\psi(x, y)$ is a formula, with free variables exactly x and y, by $(\forall x \forall y : \xi(x, y) \Leftrightarrow \psi(x, y)) \wedge (\forall x \exists_{=1} y : \xi(x, y))$, where ξ is a fresh predicate not occurring anywhere else. Therefore we will assume without loss of generality that the only functional constraints are of the form $\forall x \exists_{=1} y : R(x, y)$ where R is a predicate. The main result of this section is then the following theorem.

Theorem 1. *Let Γ be an FO^2 sentence and $\Upsilon = (|R_{i_1}| \bowtie k_1) \wedge \cdots \wedge (|R_{i_m}| \bowtie k_m) \wedge (\forall x \exists_{=1} y : R_{i_1}(x, y)) \wedge \cdots \wedge (\forall x \exists_{=1} y : R_{i_{m'}}(x, y))$ be a conjunction of cardinality and functionality constraints where $\bowtie \in \{=, \leq, \geq, <, >\}$. Computing the WFOMC of $\Gamma \wedge \Upsilon$ is domain-liftable.*

Next we prove a simple lemma that will allow us to reduce WFOMC with functionality (and possibly also cardinality) constraints to WFOMC involving only cardinality constraints and no functionality constraints.

Lemma 1. *Let Ω be the set of all possible worlds on a domain Δ. Let Γ be a first-order logic sentence. Let $\Phi = (\forall x \exists_{=1} y : R_{i_1}(x, y)) \wedge \cdots \wedge (\forall x \exists_{=1} y : R_{i_h}(x, y))$ and $\Phi' = (\forall x \exists y : R_{i_1}(x, y)) \wedge (|R_{i_1}| = |\Delta|) \wedge \cdots \wedge (\forall x \exists y : R_{i_h}(x, y)) \wedge (|R_{i_h}| = |\Delta|)$. Then for all $\omega \in \Omega$: $(\omega \models \Gamma \wedge \Phi) \Leftrightarrow (\omega \models \Gamma \wedge \Phi')$.*

Proof. Let R be any of the relations $R_{i_1}, \ldots, R_{i_h}$. The constraint $\forall x \exists_{=1} y : R(x, y)$ can be rewritten as: (i) $\forall x \exists y : R(x, y)$ and (ii) $\forall x, y, z : R(x, y) \wedge R(x, z) \Rightarrow y = z$. ($\Rightarrow$) It follows from (i) that $|R| \geq |\Delta|$. If $|R| > |\Delta|$ then by the pigeon-hole principle, there must be at least one $s \in \Delta$ such that $R(s, t)$ and $R(s, t')$ for some $t \neq t' \in \Delta$ which contradicts (ii). Hence, $\forall x \exists_{=1} y : R(x, y)$ implies $|R| = |\Delta|$ and $\forall x \exists y : R(x, y)$. ($\Leftarrow$) What we need to show is that if $(\forall x \exists y : R(x, y)) \wedge (|R| = |\Delta|)$ holds then (i) and (ii) must hold as well. Clearly, (i) must hold. So let us suppose, for contradiction, that $(\forall x \exists y : R(x, y)) \wedge (|R| = |\Delta|)$ holds but there is some $s \in \Delta$ such that $R(s, t)$ and $R(s, t')$ for some $t \neq t' \in \Delta$. We have $|\{(x, y) \in \Delta^2 | R(x, y) \wedge x \neq s\}| \geq |\Delta| - 1$ (from $\forall x \exists y : R(x, y)$). Therefore it is easy to see that $|R| \geq |\{(x, y) \in \Delta^2 | R(x, y) \wedge x \neq s\}| + 2 > |\Delta|$, which is a contradiction. □

Note that the constraints $|R_{i_1}| = |\Delta|, \ldots, |R_{i_h}| = |\Delta|$ are cardinality constraints which we already know how to deal with.

We are now ready to prove Theorem 1.

Proof of Theorem 1. Using Lemma 1, we can reduce the problem of computing the WFOMC of $\Gamma \wedge \Upsilon$ to the problem of computing the WFOMC of $\Gamma \wedge \Upsilon' \wedge \Upsilon''$ where Υ' contains only cardinality constraints and Υ'' has the form $(\forall x \exists y : \psi_1(x, y)) \wedge \cdots \wedge (\forall x \exists y : \psi_{m'}(x, y))$. Since $\Gamma \wedge \Upsilon''$ is an FO^2 sentence (hence domain-liftable) and Υ' is a conjunction of cardinality constraints, we can use Proposition 5 to finish the proof. □

Next we provide an illustration of the techniques derived so far.

Example 7. How many fixed points does a uniformly sampled function from $\{1, 2, \ldots, n\}$ to itself have? We can answer this question using WFOMC with functionality constraints. We define

$$\Gamma = (\forall x \exists y : f(x, y)) \wedge (\forall x : \xi(x) \Leftrightarrow f(x, x)).$$

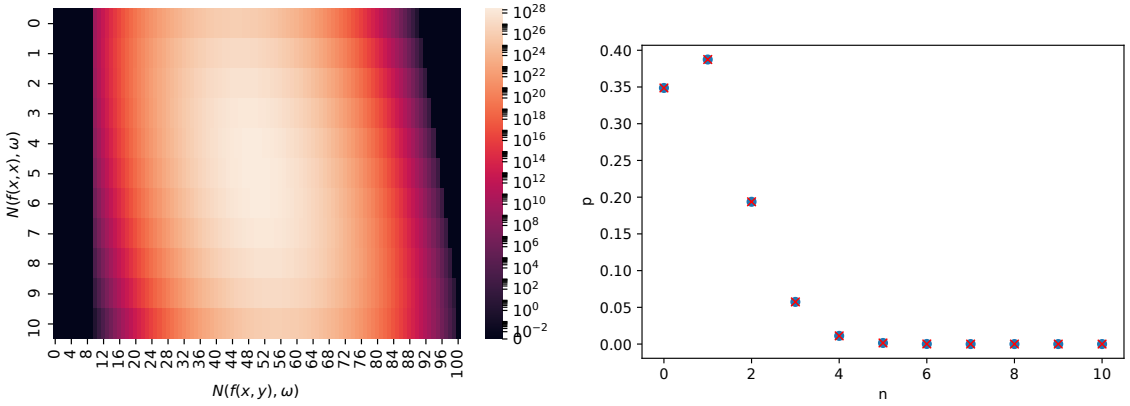

Figure 2: **Left:** The MC-function from Example 7. **Right:** The distribution of the number of fixed points (see Example 7).

Here, we introduced a fresh new predicate ξ such that, for all $t \in \Delta$, $\xi(t)$ is true if and only if t is a fixed point of f. Next we define

$$\Psi = \{f, \xi\}.$$

Then, using the techniques described in Section 3, we compute the MC-function $\mathrm{MC}_{\Psi,\Gamma,\Delta}$, which is shown for $n = 10$ in the left panel of Figure 2. Now we can extract the distribution that we wanted to compute from this MC-function by ignoring all points except those where $|f| = |\Delta|$. That is, the probability p_k that a uniformly sampled function has k fixed points is equal to:

$$p_k = \frac{\mathrm{MC}_{\Psi,\Gamma,\Delta}(|\Delta|, k)}{\sum_{j=1}^{|\Delta|} \mathrm{MC}_{\Psi,\Gamma,\Delta}(|\Delta|, j)}.$$

We show the computed distribution in the right panel of Figure 2 (blue circles). As a sanity check, we also computed the distribution analytically using the formula $\binom{n}{k}(n-1)^{n-k}/n^n$ and displayed it in the same plot (red crosses). As expected, the values computed using the two approaches are the same.

We give a slightly more complex example, computing the number of anti-involutive functions, in Appendix C.

6. The $\exists_{=k}$-Quantifier

In this section we show how the techniques used for WFOMC with functionality constraints, described in the previous section, can be further generalized to more complex settings. In particular, we show how to use them to compute WFOMC with the counting quantifier $\exists_{=k}$.

6.1 Two Types of Constraints: $\forall x \exists_{=k} y$ and $\exists_{=k} x \forall y$

We start by showing how to extend domain-liftability to FO^2 with constraints of the form $\forall x \exists_{=k} y : R(x, y)$ and $\exists_{=k} x \forall y : R(x, y)$, where $\exists_{=k}$ is the counting quantifier "exists exactly k".

Theorem 2. *Let Γ be an FO^2 sentence and*

$$\begin{aligned}\Upsilon = (|R_{i_1}| \bowtie c_1) \wedge \cdots \wedge (|R_{i_m}| \bowtie c_m) \\ \wedge (\forall x \exists_{=k_1} y : \psi_1(x,y)) \wedge \cdots \wedge (\forall x \exists_{=k_{m'}} y : \psi_{m'}(x,y)) \\ \wedge (\exists_{=k'_1} x \forall y : \varphi_1(x,y)) \wedge \cdots \wedge (\exists_{=k'_{m''}} x \forall y : \varphi_{m''}(x,y)),\end{aligned}$$

where $\bowtie \in \{=, \leq, \geq, <, >\}$. Then computing the WFOMC of $\Gamma \wedge \Upsilon$ is domain-liftable.

To prove Theorem 2, we will need the following two lemmas.

Lemma 2. *Let Ω be the set of all possible worlds on a domain Δ. Let Γ be a first-order logic sentence. Let Φ be a first-order logic sentence with cardinality constraints, defined as follows:*

$$\begin{aligned}\Phi = (|R| = k \cdot |\Delta|) \wedge (\forall x, y : R(x,y) \Leftrightarrow (f_1^R(x,y) \vee f_2^R(x,y) \vee \cdots \vee f_k^R(x,y))) \\ \wedge \bigwedge_{i=1}^{k} (\forall x \exists y : f_i^R(x,y)) \wedge \bigwedge_{i,j=1, i \neq j}^{k} (\forall x, y : \neg f_i^R(x,y) \vee \neg f_j^R(x,y)).\end{aligned}$$

Then it holds:

$$\mathrm{WFOMC}(\Gamma \wedge \forall x \exists_{=k} y : R(x,y), w, \overline{w}, \Omega) = \frac{1}{(k!)^{|\Delta|}} \mathrm{WFOMC}(\Gamma \wedge \Phi, w, \overline{w}, \Omega_{ext}),$$

where Ω is the set of all possible worlds on the domain Δ w.r.t. a given first-order logic language $\mathcal{L}$ and Ω_{ext} is the set of all possible worlds on the domain Δ w.r.t. $\mathcal{L}$ extended by predicates $f_1^R, \ldots, f_k^R$ and where $w(f_1^R) = \overline{w}(f_1^R) = \cdots = w(f_k^R) = \overline{w}(f_k^R)$.

Proof. First we show that, for all $\omega \in \Omega_{ext}$, it holds: if $\omega \models \Gamma \wedge \Phi$ then $\omega \models \Gamma \wedge \forall x \exists_{=k} y : R(x,y)$. The sentence Φ implies that for every $t_1, t_2 \in \Delta$ such that $R(t_1, t_2)$ is true, there is exactly one $i \in \{1, 2, \ldots, k\}$ such that $f_i^R(t_1, t_2)$ is true. It follows that $|R| = |f_1^R| + \cdots + |f_k^R|$. Now, using similar reasoning as in the proof of Lemma 1, we can see that $|f_1^R| + \cdots + |f_k^R| = |R| = k \cdot |\Delta|$ together with $\bigwedge_{i=1}^{k} (\forall x \exists y : f_i^R(x,y))$ and $\bigwedge_{i,j=1,i\neq j}^{k} (\forall x, y : \neg f_i^R(x,y) \vee \neg f_j^R(x,y))$ also implies that all of $f_1^R(x,y), \ldots, f_k^R(x,y)$ must be functions. It follows that $\forall x \exists_{=k} y : R(x,y)$ must be true in any possible world $\omega \in \Omega$ that satisfies Φ.

To finish the proof, let $[\omega]_{\mathcal{L}}$ denote the "projection" of ω on the language $\mathcal{L}$ which is the possible world obtained from ω by removing all atoms whose predicates are not contained in $\mathcal{L}$ (i.e. $f_1^R, \ldots, f_k^R$). One can show easily that, for every model $\omega \in \Omega$ of the sentence $\Gamma \wedge \forall x \exists_{=k} y : R(x,y)$, there are exactly $(k!)^{|\Delta|}$ models $\omega' \in \Omega_{ext}$ such that $\omega = [\omega']_{\mathcal{L}}$, which follows from the following: (i) if, for any $t \in \Delta$, we permute t_1, t_2, $\ldots$, t_k in $f_1^R(t, t_1)$, $f_2^R(t, t_2)$ $\ldots$, $f_k^R(t, t_k)$ in the model ω', we get another model of $\Gamma \wedge \Phi$, (ii) up to these permutations, the predicates f_i^k in ω' are determined uniquely by ω. Finally, the weights of all these ω's are the same as those of ω. □

Lemma 3. *Let Ω be the set of all possible worlds on a domain Δ. Let Γ be a first-order logic sentence and U^R and R be predicates. Then it holds:*

$$\begin{aligned}&\mathrm{WFOMC}(\Gamma \wedge \exists_{=k} x \forall y : R(x,y), w, \overline{w}, \Omega) \\ &\quad = \mathrm{WFOMC}(\Gamma \wedge (\exists_{=k} x : U^R(x)) \wedge (\forall x : U^R(x) \Leftrightarrow (\forall y : R(x,y))), w, \overline{w}, \Omega_{ext}).\end{aligned}$$

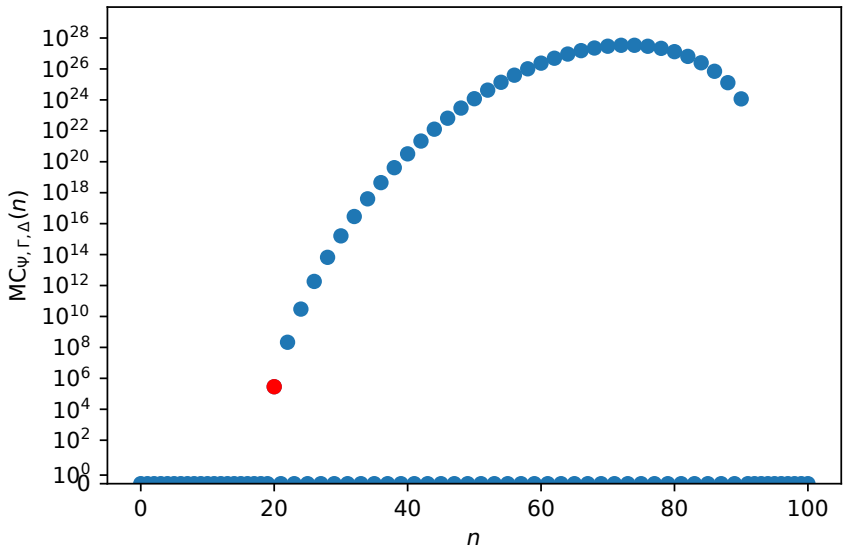

Figure 3: The function $\mathrm{MC}_{\Psi,\Gamma,\Delta}(n)/2^{|\Delta|}$, where Ψ and Γ are described in the main text in Section 6.1.1 and $|\Delta| = 10$. The y-coordinate of the red dot is equal to the number of 2-regular graphs on 10 vertices.

where Ω is the set of all possible worlds on the domain Δ w.r.t. a given first-order logic language $\mathcal{L}$ and Ω_{ext} is the set of all possible worlds on the domain Δ w.r.t. $\mathcal{L}$ extended by the predicate U^R (in particular, we assume w.l.o.g. that $\mathcal{L}$ did not originally contain this predicate).

Proof. The proof is straightforward. □

We are now ready to prove Theorem 2.

Proof of Theorem 2. First, we can replace any constraint of the form $\forall x \exists_{=k} y : \psi(x,y)$, where $\psi(x,y)$ is a formula, with free variables exactly x and y, by $(\forall x \forall y : \xi(x,y) \Leftrightarrow \psi(x,y)) \wedge (\forall x \exists_{=k} y : \xi(x,y))$, where ξ is a fresh predicate not occurring anywhere else. Therefore we will assume without loss of generality that the constraints are of the form $\forall x \exists_{=k} y : R(x,y)$ where R is a predicate. Then, repeatedly using Lemma 3, we can get rid of all constraints of the form $\exists_{=k} x \forall y : R(x,y)$. Besides new first-order logic sentences, this also produces new constraints of the form $\exists_{=k} x : U^R(x)$ which can be easily encoded using cardinality constraints $|U^R| = k$. Finally, we can use Lemma 2 repeatedly to get rid of the constraints of the form $\forall x \exists_{=k} y : R(x,y)$. Since the resulting sentence contains only two variables and cardinality constraints, it follows from Proposition 5 that we can compute its WFOMC in time polynomial in the size of the domain Δ. □

6.1.1 An Illustration: Counting K-Regular Graphs

We now illustrate the techniques developed in this section on the problem of computing the number of 2-regular graphs on n vertices. An undirected graph is called *k-regular* if all its vertices have degree k. Note that we do not count non-isomorphic graphs here.

We start by writing down the axioms defining 2-regular graphs:

$$\forall x : \neg e(x, x), \tag{2}$$
$$\forall x, y : e(x, y) \Rightarrow e(y, x), \tag{3}$$
$$\forall x \exists_{=2} y : e(x, y). \tag{4}$$

Here (2) forbids self-loops, (3) requires e to be a symmetric relation (to model undirected graphs) and (4) requires every vertex to have two out-going edges. Since, by (3), edges are guaranteed to be symmetric, (4) is enough to guarantee that every vertex will have degree 2.

Since every sentence in the above theory has at most 2 variables and contains only quantifiers $\forall$ and $\exists_{=k}$, we can apply the techniques developed in this section to compute the number of 2-regular graphs using WFOMC. We now provide details. First, we can rewrite (4) using only $\forall$, $\exists$ and cardinality constraints as in Lemma 2:

$$\forall x \forall y : \xi(x, y) \Leftrightarrow e(x, y), \tag{5}$$
$$|\xi| = 2|\Delta|, \tag{6}$$
$$\forall x \exists y : f_1(x, y), \tag{7}$$
$$\forall x \exists y : f_2(x, y), \tag{8}$$
$$\forall x \forall y : \xi(x, y) \Leftrightarrow (f_1(x, y) \vee f_2(x, y)), \tag{9}$$
$$\forall x \forall y : \neg f_1(x, y) \vee \neg f_2(x, y). \tag{10}$$

We set the weights of all the predicates to 1 except for ξ for which we set $w(\xi) = \lambda$ and $\overline{w} = 1$. Let Γ be the conjunction of (2), (3), (5), (7), (8), (9), (10). Since Γ is in FO^2, we can use, e.g., the algorithm from the work (Beame et al., 2015) to compute WFOMC for any weight λ in time polynomial in the size of the domain and number of bits needed to encode λ. Thus we can use the techniques described in Section 3 to compute the MC-function $\mathrm{MC}_{\Psi,\Gamma,\Delta}$, where we set $\Psi = \{\xi\}$. Finally, we still need to divide the MC-function by $2!^{|\Delta|} = 2^{|\Delta|}$ to account for the over-counting caused by f_1 and f_2. The number of 2-regular graphs is then equal to $\mathrm{MC}_{\Psi,\Gamma,\Delta}(\mathbf{n})/2^{|\Delta|}$ where $\mathbf{n} = 2|\Delta|$. For 3, 4, 5, 6, 7, 8, 9, 10 vertices this method yields the following numbers of 2-regular graphs: 1, 3, 12, 70, 465, 3507, 30016, 286884. One can check that these numbers are exactly the same as the numbers of 2-regular graphs listed in the *On-Line Encyclopedia of Integer Sequences* as sequence A001205.[6] We show one example of the MC-function divided by 2^{10} in Figure 3 for $|\Delta| = 10$. The x-coordinate of the red point shown there is $n = 2|\Delta| = 20$ and its y-coordinate $\mathrm{MC}_{\Psi,\Gamma,\Delta}/2^{10} = 286884$ is the number of 2-regular graphs on 10 vertices.

One can easily adapt the above example for counting the number of k-regular graphs for a general k (although the complexity of the encoding grows with k). We note that counting k-regular graphs is, in fact, an interesting, not completely trivial, problem; for instance, it has been studied for $k = 2, 3, 4, 5$ by Goulden and Jackson (1986) and for the general case of arbitrary k by Gessel (1990).

6. http://oeis.org/A001205

6.2 The General Case

In this section we use the results from the previous sections to show that computing WFOMC of arbitrary two-variable first-order logic sentences with the quantifiers $\exists_{=k}$ is domain-liftable. In particular, we now consider the case when the sentences can contain quantifiers $\exists_{=k}$ at arbitrary places.

Theorem 3. *Let Γ be a sentence in the two-variable fragment of first-order logic, possibly containing a finite number of counting quantifiers $\exists_{=k_1}, \exists_{=k_2}, \ldots, \exists_{=k_m}$. Then computing* $\mathrm{WFOMC}(\Gamma, w, \overline{w}, \Delta)$ *is domain-liftable.*

We already know from the previous sections how to do inference with special types of constraints, in particular with cardinality constraints and constraints of the form $\forall x \exists_{=k} y$ and $\exists_{=k} x \forall y$. As we will see later, the main difficulty in extending these results is to be able to encode sentences of the form $\forall x : A(x) \Leftrightarrow (\exists_{=k} y : R(x,y))$, which will allow us to transform a given sentence to a form manageable by Theorem 2. We show how to do it in two steps. In Lemma 4, we show how to encode sentences of the form $\forall x : A(x) \vee (\exists_{=k} y : R(x,y))$ using the form of constraints that we already know how to handle. We then use Lemma 4, together with Proposition 3 (Appendix A.2, Beame et al., 2015), to encode the sentence $\forall x : A(x) \Leftrightarrow (\exists_{=k} y : R(x,y))$ when we prove Theorem 3.

Lemma 4. *Let R be a relation, Γ be a first-order logic sentence and Υ be a conjunction of $\forall x \exists_{=k} y$ and $\exists_{=k} x \forall y$-constraints and cardinality constraints. Let us define $\Phi = \Phi_1 \wedge \Phi_2 \wedge \Phi_3 \wedge \Phi_4$ where:*

$$\begin{aligned}
\Phi_1 =& \forall x \exists_{=k} y : B^R(x,y), \\
\Phi_2 =& (|U^R| = k), \\
\Phi_3 =& \forall x \forall y : (A(x) \wedge B^R(x,y)) \Rightarrow U^R(y), \\
\Phi_4 =& \forall x \forall y : \neg A(x) \Rightarrow (R(x,y) \Leftrightarrow B^R(x,y)).
\end{aligned}$$

Then the following holds for WFOMC:

$$\mathrm{WFOMC}(\Gamma \wedge \Upsilon \wedge (\forall x : A(x) \vee (\exists_{=k} y : R(x,y))), w, \overline{w}, \Delta) = \frac{1}{\binom{|\Delta|}{k}} \mathrm{WFOMC}(\Gamma \wedge \Upsilon \wedge \Phi, w, \overline{w}, \Omega_{ext}),$$

where Ω is the set of all possible worlds on the domain Δ w.r.t. a given first-order logic language $\mathcal{L}$ and Ω_{ext} is the set of all possible worlds on the domain Δ w.r.t. $\mathcal{L}$ extended by predicates U^R and B^R (in particular, we assume w.l.o.g. that $\mathcal{L}$ did not originally contain these predicates).

Before proving this lemma, we give some intuition about the transformation used in it. First, we introduce an auxiliary binary relation B^R and we require it to satisfy $\forall x \exists_{=k} y : B^R(x,y)$ (that is what Φ_1 does). Then we introduce an auxiliary unary relation U^R and we require it to have cardinality equal to k (this is done by Φ_2). The combination of Φ_1 and Φ_2 and the constraint expressed by Φ_3 then has the effect that $R^B(t,y)$ is uniquely determined, up to the choice of U^R, for any $t \in \Delta$ for which $A(t)$ is true. This will be important to guarantee

correct counting. Finally, Φ_4 forces $B^R(t,y)$ to be equivalent to $R(t,y)$ for any $t \in \Delta$ for which $A(t)$ is false, which among others means that $B^R(t,y)$ is also uniquely determined in this case. Moreover, Φ_4 is only satisfied when $\exists_{=k} y : R(t,y)$ is true for all $t \in \Delta$ for which $A(t)$ is false, which is what we need to encode. The choice of U^R then corresponds to the $\binom{|\Delta|}{k}$ factor in the statement of the lemma. The proof gives more details.

Proof of Lemma 4. First, we show that for every possible world $\omega \in \Omega_{ext}$ it holds: if $\omega \models \Phi$ then $\omega \models (\forall x : A(x) \vee (\exists_{=k} y : R(x,y)))$. For contradiction, let us assume that $\omega^* \in \Omega_{ext}$ is a possible world such that $\omega^* \models \Phi$ and $\omega^* \not\models (\forall x : A(x) \vee (\exists_{=k} y : R(x,y)))$. Then there must be a $t \in \Delta$ such that $\omega^* \models \neg A(t) \wedge \neg(\exists_{=k} y : R(t,y))$. At the same time, it must be the case that $\omega^* \models \exists_{=k} y : B^R(t,y)$ (from Φ_1) and $\omega^* \models \forall y : R(t,y) \Leftrightarrow B^R(t,y)$ (from Φ_4). However, these cannot be all true at the same time, thus, we have arrived at a contradiction.

To finish the proof, let $[\omega]_{\mathcal{L}}$ denote the "projection" of ω on the language $\mathcal{L}$ which is the possible world obtained from ω by removing all atoms whose predicates are not contained in $\mathcal{L}$ (i.e. U^R and B^R). We show that for every $\omega \in \Omega$ such that $\omega \models (\forall x : A(x) \vee (\exists_{=k} y : R(x,y)))$ there are exactly $\binom{|\Delta|}{k}$ possible worlds $\omega' \in \Omega_{ext}$ such that $\omega' \models \Phi$ and $\omega = [\omega']_{\mathcal{L}}$. Let $\omega \in \Omega$ be any such possible world. Due to Φ_2, to extend ω, we have to select a set $\{t_1, t_2, \ldots, t_k\}$ of elements of the domain Δ and make $U^R(t_1)$, $U^R(t_2)$, $\ldots$, $U^R(t_k)$ true (and no other). This can be done in $\binom{|\Delta|}{k}$ different ways. Once, we set the U^R predicate in this way, there is only one way to extend the possible world ω: For every $t \in \Delta$ such that $\omega' \models A(t)$, it must be true $\omega' \models B(t,t_1) \wedge \cdots \wedge B^R(t,t_k)$. This is because, due to Φ_1, for every $t \in \Delta$ there must be exactly k domain elements t'_1, $\ldots$, t'_k such that $\omega' \models B^R(t,t'_1) \wedge \cdots \wedge B^R(t,t'_k)$. Moreover, due to Φ_3, if $\omega' \models A(t) \wedge B^R(t,t'_i)$ are true then $\omega' \models U^R(t'_i)$ must be true as well. However, there are only k such domain elements t'_i (due to Φ_2). This means that $B^R(t,t'_i)$ is uniquely determined. Moreover, for all the other $t \in \Delta$ such that $\omega' \models \neg A(t)$, B^R must coincide with R and hence is uniquely determined as well. This is what we needed to finish the proof. $\square$

Now we are ready to prove Theorem 3.

Proof of Theorem 3. We start by showing how to deal with sentences of the form $\Gamma \wedge \Upsilon$ where

$$\Upsilon = \forall x : A(x) \Leftrightarrow (\exists_{=k} y : R(x,y)).$$

First we rewrite the sentence

$$\Upsilon_1 = \forall x : A(x) \Rightarrow (\exists_{=k} y : R(x,y))$$

into a form that we already know how to handle. For that we introduce a new unary predicate B and define

$$\Upsilon'_1 = (\forall x : A(x) \Leftrightarrow \neg B(x)) \wedge (\forall x : B(x) \vee (\exists_{=k} y : R(x,y))).$$

Sentences in this form can already be handled by Lemma 4. Let Ω'_{ext} be the set of all possible worlds on domain Δ over the given language extended by the predicate B. Then for every possible world $\omega \in \Omega$ which is a model of $\Gamma \wedge \Upsilon_1$ there is exactly one possible world in Ω'_{ext} which is a model of $\Gamma \wedge \Upsilon'_1$, and vice versa.

Next we need to show how to handle the sentence $\Upsilon_2 = \forall x : A(x) \Leftarrow (\exists_{=k} y : R(x,y))$. At first this may seem difficult but Proposition 3 comes to the rescue here. Specifically, if we equivalently write

$$\Upsilon_2 = \forall x : A(x) \vee \neg(\exists_{=k} y : R(x,y)),$$

we can get rid of the negation in front of $(\exists_{=k} y : R(x,y))$ using Proposition 3 as follows. We create two new unary predicates C and D and define

$$\Upsilon_2' = \forall x : ((\exists_{=k} y : R(x,y)) \vee C(x)) \wedge (C(x) \vee D(x)) \wedge ((\exists_{=k} y : R(x,y)) \vee D(x))).$$

It follows from Proposition 3 that if we set $w(C) = \overline{w}(C) = w(D) = 1$ and $\overline{w}(D) = -1$ then for any sentence Θ it will hold

$$\text{WFOMC}(\Theta \wedge \Upsilon_2, w, \overline{w}, \Omega) = \text{WFOMC}(\Theta \wedge (\forall x : A(x) \vee C(x)) \wedge \Upsilon_2', w, \overline{w}, \Omega_{ext}''),$$

where Ω is the set of all possible worlds on the domain Δ w.r.t. the given first-order logic language $\mathcal{L}$ and Ω_{ext}'' is the set of all possible worlds on the domain Δ w.r.t. $\mathcal{L}$ extended by the predicates C and D.

Next we can apply consecutively the two steps described above which gives us

$$\text{WFOMC}(\Gamma \wedge \Upsilon, w, \overline{w}, \Omega) = \text{WFOMC}(\Gamma \wedge (\forall x : A(x) \vee C(x)) \wedge \Upsilon_1' \wedge \Upsilon_2', w, \overline{w}, \Omega_{ext}'''),$$

where Ω is the set of all possible worlds on the domain Δ w.r.t. a given first-order logic language $\mathcal{L}$ and Ω_{ext}''' is the set of all possible worlds on the domain Δ w.r.t. $\mathcal{L}$ extended by the predicates B, C and D. From Lemma 4, we already know how to handle the sentences Υ_1' and Υ_2' when computing WFOMC.

Finally, we use the above to compute WFOMC of arbitrary two-variable sentences which may contain quantifiers $\exists_{=k}$. This is relatively straightforward. Let Γ be such a sentence. We just need to repeatedly replace every subformula of the form $\exists_{=k} y : \psi(x,y)$ by $A_\psi(x)$, where A_ψ is a fresh unary predicate, and add a "definition" of this predicate in the form $(\forall x \forall y : B_\psi(x,y) \Leftrightarrow \psi(x,y)) \wedge (A_\psi(x) \Leftrightarrow (\exists_{=k} y : B_\psi(x,y)))$. It is not difficult to see that the resulting sentence after being rewritten according to the rules described in this proof (i) can be handled by Theorem 2, (ii) that it contains only two variables and (iii) that its size is independent of the size of the domain. Additionally, the only way in which the resulting sentence will depend on the domain is through the cardinality constraints of the form $|R| = k|\Delta|$ and, in any such a constraint, $k|\Delta|$ will grow only polynomially with the size of the domain (this is no problem for establishing domain liftability). The statement of the theorem then follows from the above and from Theorem 2. □

7. The Two-Variable Fragment with Counting

In this section we show that the results from the previous sections imply domain liftability for the two-variable fragment of first-order logic with counting quantifiers.

Theorem 4. *Weighted first-order model counting is domain-liftable for the two-variable fragment of first-order logic with counting quantifiers.*

Proof. We can prove this theorem by reducing it to the case with just the $\exists_{=k}$-quantifier, whose domain-liftability is established by Theorem 3.

First, any sub-formula of the form $\exists_{\leq k} y : \psi(x, y)$ can be replaced by:

$$(\forall y : \neg\psi(x, y)) \vee (\exists_{=1} y : \psi(x, y)) \vee (\exists_{=2} y : \psi(x, y)) \vee \cdots \vee (\exists_{=k} y : \psi(x, y)).$$

Obviously, the size of the result of the above transformation is independent of the domain size and if the original formula was in FO^2, the new one will be in FO^2 as well.

Next we need to get rid of the sub-formulas of the form $\exists_{\geq k} y : \psi(x, y)$. Note that we cannot blindly apply the same method we used for the sub-formulas with the quantifier $\exists_{\leq k}$ because, in that case, the number of disjuncts in the resulting formula would grow with the size of the domain. Instead, we proceed as follows. We equivalently rewrite the sub-formula as:

$$\exists_{\geq k} y : \psi(x, y) = \neg\neg(\exists_{\geq k} y : \psi(x, y)) = \neg(\exists_{\leq k-1} y : \psi(x, y)).$$

We already know how to handle sub-formulas of this form. So we are done. □

8. Related Work

The work presented in this paper builds on a long stream of research in lifted inference (Poole, 2003; de Salvo Braz et al., 2005; Van den Broeck, 2011; Gogate & Domingos, 2011; Van den Broeck, 2013; Van den Broeck et al., 2014; Beame et al., 2015; Kazemi et al., 2016; Kuusisto & Lutz, 2018). On the technical level, in the present paper, we directly exploit results that established domain-liftability of the two-variable fragment of first-order logic (Van den Broeck, 2011; Van den Broeck et al., 2014), which we extended by allowing counting quantifiers. The result on domain-liftability of the two-variable fragment was relatively recently extended in two somewhat related directions that we describe below.

First, Kazemi et al. (2016) showed that so-called *domain-recursion rule*, which had been previously proposed by Van den Broeck, allows to enlarge the class of domain-liftable theories. In particular, they identified two new domain-liftable fragments of first-order logic which they call $\mathrm{S}^2\mathrm{FO}^2$ and $\mathrm{S}^2\mathrm{RU}$. These two classes contain among others certain theories with functionality axioms but, as also pointed out by Kuusisto and Lutz (2018), not all FO^2 theories with functionality axioms are contained in them. The domain-liftable class identified in our work, i.e. FO^2 with counting quantifiers, and the classes studied by Kazemi et al. (2016) are incomparable. However, we believe that our techniques and theirs could be combined in future work.

More recently, Kuusisto and Lutz (2018) showed that WFOMC for FO^2 with at most one functionality constraint is domain-liftable using a rather complex argument. They also mentioned in the same work (although without giving any details) that if one could extend this result to multiple functionality constraints, domain-liftability of FO^2 with counting quantifiers would follow. Specifically, in the concluding section of their paper, they say: *It can be shown that WFOMC for formulae of two-variable logic with counting* C^2 *can be reduced to WFOMC for* FO^2 *with several functionality axioms.* Thus, in principle, our results from Section 6.1, where we established domain-liftability of FO^2 with an arbitrary number of functionality constraints, combined with their remark would also be sufficient to establish our main result. However, since there are no details and no proof, we had to provide these

ourselves. In fact, we did not use reductions directly based on functionality constraints in our proofs (since that seemed to be rather wasteful), therefore we suspect that our reductions might also be more efficient.

Finally, the methods that we used in the present paper resemble techniques used in enumerative combinatorics (Stanley, 1986), in particular generating functions. We plan to investigate these connections more closely in future work.

9. Conclusions

In this work we showed that the two-variable fragment of first-order logic with counting quantifiers is domain-liftable. This significantly broadens the class of weighted first-order model counting problems that can be solved in polynomial time. There is still a lot one can do from here, especially for improving the practical efficiency of lifted inference algorithms on problems that result from our reductions.

Acknowledgements

This work was supported by Czech Science Foundation project "Generative Relational Models" (20-19104Y), the OP VVV project *CZ.02.1.01/0.0/0.0/16_019/0000765* "Research Center for Informatics" and a donation from X-Order Lab. The author is grateful to the anonymous reviewers for their helpful comments and Guy Van den Broeck for pointing out the problem of inference in the two-variable fragment of FOL with counting quantifiers several years back.

Appendix A. Computing Weighted Model-Counting Functions Using DFT

Here we describe an alternative approach to computing weighted model-counting functions based on discrete Fourier transform. This approach is a generalization of a method from our previous work (Kuzelka, 2020) where it was used in the context of Markov logic networks.

Notation We need a bit more notation here. We use i to denote the imaginary unit $i^2 = -1$ and $\langle v, w \rangle$ to denote the inner product of the vectors v and w (when v and w are real vectors, inner product coincides with scalar product).

A.1 Discrete Fourier Transform

Here we describe the basic properties of *multi-dimensional Fourier transform* (DFT) that will be needed in this paper. Let d be a positive integer and let $\mathbf{N} = [N_1, \ldots, N_d] \in (\mathbf{N} \setminus \{0\})^d$ be a vector of positive integers. Let us define $\mathcal{J} = \{0, 1, \ldots, N_1 - 1\} \times \{0, 1, \ldots, N_2 - 1\} \times \cdots \times \{0, 1, \ldots, N_d - 1\}$. Let $f : \mathcal{J} \to \mathbb{C}$ be a function defined on $\mathcal{J}$. Then the DFT of f is the function $g : \mathcal{J} \to \mathbb{C}$ defined as

$$g(\mathbf{k}) = \sum_{\mathbf{n} \in \mathcal{J}} f(\mathbf{n}) e^{-i2\pi \langle \mathbf{k}, \mathbf{n}/\mathbf{N} \rangle} \tag{11}$$

where $\mathbf{n}/\mathbf{N} \stackrel{def}{=} [[\mathbf{n}]_1/N_1, [\mathbf{n}]_2/N_2, \ldots, [\mathbf{n}]_d/N_d]$ (i.e. "/" denotes component-wise division). We use the notation $g = \mathcal{F}\{f\}$. The inverse transform is then given as

$$f(\mathbf{n}) = \frac{1}{\prod_{l=1}^{d} N_l} \sum_{\mathbf{k}\in\mathcal{J}} g(\mathbf{k}) e^{i2\pi\langle \mathbf{n},\mathbf{k}/\mathbf{N}\rangle}. \tag{12}$$

It holds that $f = \mathcal{F}^{-1}\{\mathcal{F}\{f\}\}$.

A.2 Computing Weighted Model-Counting Functions Using DFT

Let Ω be the set of all possible worlds on a given domain Δ and a given set of predicates $\mathcal{R}$. Here we show how to compute the WMC-function $\mathrm{WMC}_{\Psi,\Gamma,\Delta}(\mathbf{n})$ for given list of predicates $\Psi = (R_1, \ldots, R_m)$ and a given sentence Γ using DFT.

First, $\mathrm{WMC}_{\Psi,\Gamma,\Delta}(\mathbf{n})$ is a real-valued function of m-dimensional integer vectors. We can restrict the domain[7] of $\mathrm{WMC}_{\Psi,\Gamma,\Delta}(\mathbf{n})$ to the set

$$\mathcal{D} = \{0, 1, \ldots, M_1\} \times \{0, 1, \ldots, M_2\} \times \cdots \times \{0, 1, \ldots, M_m\}$$

where

$$M_1 = |\Delta|^{arity(R_1)},\ M_2 = |\Delta|^{arity(R_2)},\ \ldots,\ M_m = |\Delta|^{arity(R_m)}.$$

Second, from the definition of DFT we then have for the Fourier transform $g(\mathbf{k})$:

$$g(\mathbf{k}) = \sum_{\mathbf{n}\in\mathcal{D}} \mathrm{WMC}_{\Psi,\Gamma,\Delta}(\mathbf{n}, w, \overline{w}) \cdot e^{-i2\pi\langle \mathbf{k},\mathbf{n}/\mathbf{M}\rangle} \tag{13}$$

where $\mathbf{k} = (k_1, \ldots, k_m)$, $\mathbf{M} = (M_1 + 1, \ldots, M_m + 1)$ and the division in $\mathbf{n}/\mathbf{M}$ is again component-wise.

Third, let $\Omega^* = \{\omega \in \Omega | \omega \models \Gamma\}$. Let $\mathcal{R}$ be the set of all predicates R that have non-neutral weights (i.e. $w(R) \neq 1$ or $\overline{w}(R) \neq 1$). Using the definition of $\mathrm{WMC}_{\Psi,\Gamma,\Delta}(\mathbf{n}, w, \overline{w})$, we can write

$$\begin{aligned}
g(\mathbf{k}) &= \sum_{\mathbf{n}\in\mathcal{D}} \left(\sum_{\omega\in\Omega^*:\mathbf{N}(\Psi,\omega)=\mathbf{n}} \prod_{R\in\mathcal{R}} w(R)^{\mathbf{N}(R,\omega)} \cdot \overline{w}(R)^{|\Delta|^{arity(R)}-\mathbf{N}(R,\omega)} \right) e^{-i2\pi\langle \mathbf{k},\mathbf{n}/\mathbf{M}\rangle} \\
&= \sum_{\mathbf{n}\in\mathcal{D}} \sum_{\omega\in\Omega^*:\mathbf{N}(\Psi,\omega)=\mathbf{n}} \left(\prod_{R\in\mathcal{R}} w(R)^{\mathbf{N}(R,\omega)} \cdot \overline{w}(R)^{|\Delta|^{arity(R)}-\mathbf{N}(R,\omega)} \right) e^{-i2\pi\langle \mathbf{k}/\mathbf{M},\mathbf{N}(\Psi,\omega)\rangle} \\
&= \sum_{\omega\in\Omega^*} \left(\prod_{R\in\mathcal{R}} w(R)^{\mathbf{N}(R,\omega)} \cdot \overline{w}(R)^{|\Delta|^{arity(R)}-\mathbf{N}(R,\omega)} \right) e^{-i2\pi\langle \mathbf{k}/\mathbf{M},\mathbf{N}(\Psi,\omega)\rangle} \\
&= \mathrm{WFOMC}(\Gamma, w', \overline{w}', \Delta),
\end{aligned}$$

where we set $w'(R_i) = w(R_i) \cdot e^{-2i\pi k_i/M_i}$ for all $R_i \in \Psi$, $w'(R) = w(R)$ for all $R \in \mathcal{R} \setminus \Psi$ and $\overline{w}'(R) = 1$ for all $R \in \mathcal{R}$. Thus, we can compute the DFT of WMC-functions using a

7. Here, *domain* refers to the domain of a mathematical function, not to a *domain* as a set of domain elements Δ.

polynomial number (in $|\Delta|$) of queries to a WFOMC oracle. Once we have the DFT $g(\mathbf{k})$ of the MC-function, obtaining the MC-function from the DFT is trivial. We just compute the inverse DFT of $g(\mathbf{k})$. Importantly, to obtain this, we did not need to add explicit cardinality constraints, expressed as first-order logic sentences, to Γ or modify the formulas in it or in the set Ψ in any way.

A note on representation of complex numbers In our previous work (Kuzelka, 2020), we discuss the issues of representing complex numbers in the computations such as DFT in detail.

Appendix B. Omitted Proofs

In this section we give proofs that were omitted from the main text.

B.1 Proof of Proposition 1

First, we bound the coefficients of monomials of the polynomials $l_i(x)$. We write $l_i(x) = \sum_{j=0}^{d} \frac{e_{i,j}}{f_{i,j}} \cdot x^j$ where $e_{i,j}, f_{i,j} \in \mathbb{N}$. We have

$$\max_i \log |e_{i,j}| \leq \max_i \log \left(2^d \prod_{\substack{0 \leq j \leq d \\ i \neq j}}^{d} x_j \right) \leq \log \left(2^d \max_j |x_j|^d \right) = d \log \left(2 \max_j |x_j| \right), \tag{14}$$

$$\max_i \log |f_{i,j}| \leq \max_i \log \left(\prod_{\substack{0 \leq j \leq d \\ i \neq j}}^{d} |x_i - x_j| \right) \leq \max_i \log \left(\max_j |x_i - x_j|^d \right)$$
$$= d \log \left(2 \max_j |x_j| \right). \tag{15}$$

Let $y_i = \frac{y_i'}{y_i''}$ where both y_i' and y_i'' are integers. We then also have for the coefficient a_j of the monomial x^j in the interpolating polynomial:

$$a_j = \sum_{i=0}^{d} \frac{y_i'}{y_i''} \frac{e_{i,j}}{f_{i,j}} = \frac{\sum_{i=0}^{d} y_i' e_{i,j} \prod_{0 \leq k \leq d, k \neq j} y_k'' f_{k,j}}{\prod_{i=0}^{d} y_i'' f_{i,j}}.$$

Both the numerator and denominator of the last expression are integers and it holds

$$\log \left(\left| \sum_{i=0}^{d} y_i' e_{i,j} \cdot \prod_{0 \leq k \leq d, k \neq j} y_k'' f_{k,j} \right| \right) \leq \log \left(\left| d \max_i \left\{ y_i' e_{i,j} \right\} \cdot \prod_{0 \leq k \leq d, k \neq j} y_k'' f_{k,j} \right| \right)$$
$$\leq \log d + \log \max_i |y_i'| + \log \max_i |e_{i,j}| + \sum_{0 \leq k \leq d, k \neq j} \left(\log |y_k''| + \log |f_{k,j}| \right)$$
$$\leq \log d + \log \max_i |y_i'| + d \log \max_i |y_i''| + \left(d + d^2 \right) \log \left(2 \max_j |x_j| \right)$$

(here the last inequality follows from (14) and (15)). It also holds

$$\log\left(\left|\prod_{i=0}^{d} y_i'' f_{i,j}\right|\right) \leq d\log\max_i |y_i''| + d\max_i \log\left(|f_{i,j}|\right) \leq d\log\max_i |y_i''| + d^2\log\left(2\max_j |x_j|\right).$$

It follows that the number of bits needed to represent the coefficients of the interpolating polynomial grows only polynomially with the number of bits needed to encode the points (x_i, y_i), which is what we needed to show. □

B.2 Proof of Proposition 2

Let use define the following set of integer vectors

$$\mathcal{D} = \{0, 1, \ldots, M_1\} \times \{0, 1, \ldots, M_2\} \times \cdots \times \{0, 1, \ldots, M_m\}$$

where $M_1 = |\Delta|^{arity(R_1)}$, $M_2 = |\Delta|^{arity(R_2)}$, $\ldots$, $M_m = |\Delta|^{arity(R_m)}$. It is obvious that the weight of any possible world $\omega \in \Omega$ can be only one of the form $\prod_{i=1}^{m} w(R_i)^{n_i} \cdot \overline{w}(R_i)^{|\Delta|^{arity(R_i)}-n_i}$ for some $(n_1, n_2, \ldots, n_m) \in \mathcal{D}$. That means that there are only polynomially many, in Δ, different weights of possible worlds and the WFOMC is their weighted sum. Specifically, the WFOMC can be written as

$$\mathrm{WFOMC}(\Gamma, w, \overline{w}, \Delta) = \sum_{(n_1,\ldots,n_m)\in\mathcal{D}} C_{(n_1,\ldots,n_m)} \prod_{i=1}^{m} \left(\frac{w'(R_i)}{w''(R_i)}\right)^{n_i} \cdot \left(\frac{\overline{w}'(R_i)}{\overline{w}''(R_i)}\right)^{|\Delta|^{arity(R_i)}-n_i} \tag{16}$$

where $C_{(n_1,\ldots,n_m)} \in \mathbb{N}$. It is easy to see that $C_{(n_1,\ldots,n_m)} \leq 2^{m\cdot|\Delta|^A}$ and $n_i \leq |\Delta|^A$ where $A = \max_{R\in\mathcal{R}} arity(R)$. Next we define

$$D'_{(n_1,\ldots,n_m)} = C_{(n_1,\ldots,n_m)} \prod_{i=1}^{m} w'(R_i)^{n_i} \cdot \overline{w}'(R_i)^{|\Delta|^{arity(R_i)}-n_i}$$

$$D''_{(n_1,\ldots,n_m)} = \prod_{i=1}^{m} w''(R_i)^{n_i} \cdot \overline{w}''(R_i)^{|\Delta|^{arity(R_i)}-n_i}$$

We have

$$\log D'_{(n_1,\ldots,n_m)} = \log C_{(n_1,\ldots,n_m)} + \sum_{i=1}^{m} n_i \log w'(R_i) + \sum_{i=1}^{m} \left(|\Delta|^{arity(R_i)} - n_i\right) \log \overline{w}'(R_i)$$
$$\leq mA\log 2 + 2m|\Delta|^A \log M$$

and similarly also

$$\log D''_{(n_1,\ldots,n_m)} = \sum_{i=1}^{m} n_i \log w''(R_i) + \sum_{i=1}^{m} \left(|\Delta|^{arity(R_i)} - n_i\right) \log \overline{w}''(R_i) \leq 2m|\Delta|^A \log M.$$

It follows that each of the summands in (16) can be represented as a fraction where both the numerator and the denominator are represented using a polynomial number of bits in $|\Delta|$ and in $\log M$.

Finally, $\mathrm{WFOMC}(\Gamma, w, \overline{w}, \Delta)$ is a sum of $|\mathcal{D}|$ such fractions and $|\mathcal{D}|$ is also polynomial in $|\Delta|$. The statement of the proposition follows from this. □

Appendix C. An Additional Example: Anti-Involutive Functions

Here we provide another example in which we illustrate how the techniques presented in Section 6.1, where we introduced $\forall\exists_{=1}$-constraints, can be used to efficiently encode more complex examples without much additional complexity Although the techniques that we developed in the later sections (Section 6.2 and Section 7) are more general but we may often pay for this generality by computational speed. For that we may sometimes need to slightly adjust these techniques. We illustrate it on the case of anti-involutive functions.

We look at functions from $M = \{1, 2, \ldots, m\}$ to $N = \{1, 2, \ldots, n\}$. We say that a function $f : M \to N$ is *anti-involutive* if $f(f(x)) \neq x$ for all $x \in M$. We are interested in the problem of counting all anti-involutive functions from M to N. For that we first define such functions in first-order logic with counting quantifiers and cardinality constraints (which could also be represented in this case using counting quantifiers):

$$|M| = m, \tag{17}$$
$$\forall x : M(x) \Rightarrow (\exists_{=1} y : f(x, y)), \tag{18}$$
$$\forall x, y : \neg M(x) \Rightarrow \neg f(x, y), \tag{19}$$
$$\forall x, y : \neg f(x, y) \vee \neg f(y, x). \tag{20}$$

We also assume that $\Delta = N$. The models of this theory on the domain Δ must be anti-involutive functions from a set of size m (rather than from the set M) to Δ. This means that to obtain the number of anti-involutive functions from M to N, we will need to divide the model count that we obtain by $\binom{n}{m}$.

First, we replace both (18) and (19) by:

$$|f| = m, \tag{21}$$
$$\forall x \exists y : M(x) \Rightarrow f(x, y). \tag{22}$$

The correctness of this transformation follows from similar reasoning as we used in the proof of Lemma 1. Now we have no more counting quantifiers, only first-order logic sentences and cardinality constraints. Hence, all we need is to compute the MC-function $\mathrm{MC}_{\Psi,\Gamma,\Delta}(\mathbf{n})$, where $\Psi = (f, M)$ and Γ is a conjunction of (20) and (22), and then use it to count only over the possible worlds that satisfy the cardinality constraints $|f| = |M| = m$.

We plotted the resulting MC-functions in the left panels of Figure 4 and Figure 5, for $n = 5$ and $n = 10$, respectively.[8] In the right panels of these two figures, we plotted the numbers of anti-involutive functions computed by our approach (blue circles). Note that each of these plots corresponds to the diagonal of the respective MC-function (i.e. $\mathrm{MC}_{\Psi,\Gamma,\Delta}(m, m)$) divided by $\binom{n}{m}$.

8. It is interesting to note that there is a good reason why the MC-function is zero in the roughly bottom half of the plots. Taking a closer look at the MC-function we can notice that it is zero for $|f| > \binom{n}{2}$. This is because of the constraint $\forall x \forall y : \neg f(x, y) \vee \neg f(y, x)$, which implies that the cardinality of the relation f cannot exceed the number of edges of the complete undirected graph on n vertices.

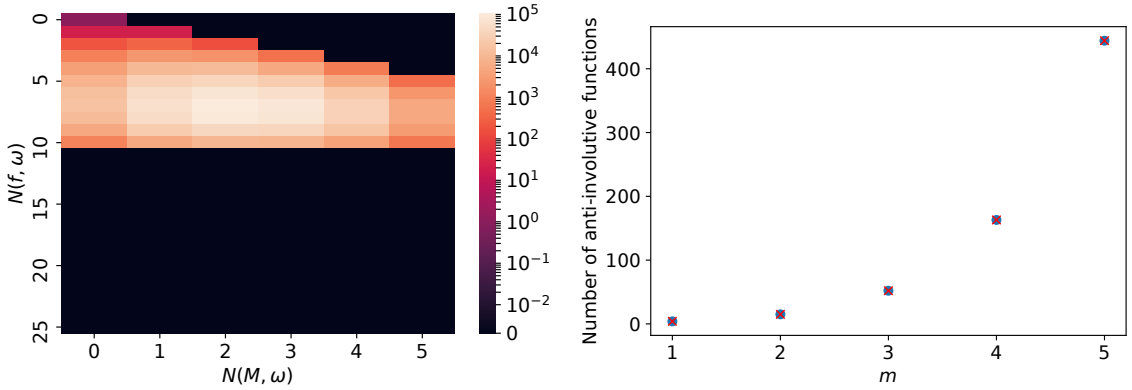

Figure 4: **Left:** The MC-function for computing the number of anti-involutive functions when $n = 5$. **Right:** The number of anti-involutive functions from $\{1, 2, \ldots, m\}$ to $\{1, 2, \ldots, 5\}$.

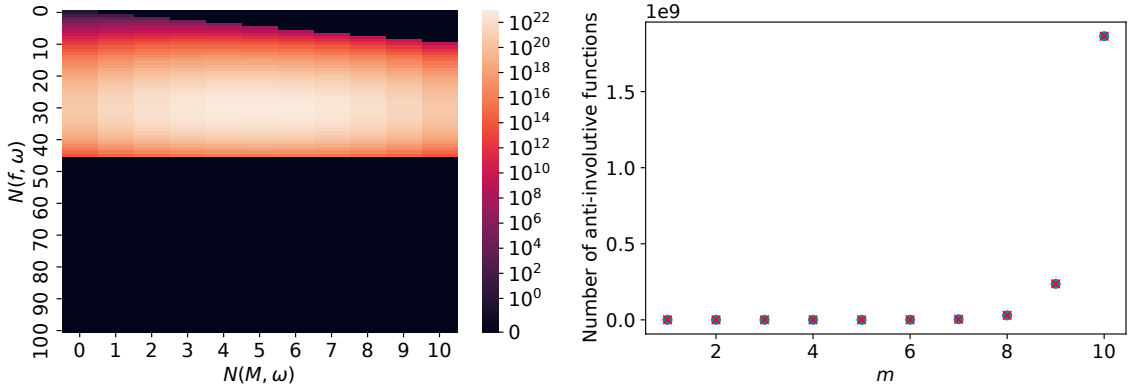

Figure 5: **Left:** The MC-function for computing the number of anti-involutive functions when $n = 10$. **Right:** The number of anti-involutive functions from $\{1, 2, \ldots, m\}$ to $\{1, 2, \ldots, 10\}$.

As a sanity check we compared our results with the numbers given by the explicit formula

$$F(m, n) = \sum_{i=0}^{\lfloor m/2 \rfloor} (-1)^i (n-1)^{m-2i} \binom{m}{2i} \frac{(2i)!}{2^i (i!)},$$

derived by Kuusisto and Lutz (2018). We plotted it as red crosses. As expected, both methods give the same results.

Alternatively, instead of replacing (18) and (19) by (21) and (22), we could have used the transformation from Lemma 4. However, that would actually lead to a more complex encoding. So, even though, we would still be able to solve the counting problem in time polynomial in the size of the domain (i.e. in n), the exponent of the polynomial might be higher. This illustrates the fact that there may often be more efficient transformations than those we used in our proofs. Arguably, there seems to be quite some potential in investigating more efficient transformations for certain cases.

References

Beame, P., Van den Broeck, G., Gribkoff, E., & Suciu, D. (2015). Symmetric weighted first-order model counting. In *Proceedings of the 34th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pp. 313–328. ACM.

de Salvo Braz, R., Amir, E., & Roth, D. (2005). Lifted first-order probabilistic inference. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pp. 1319–1325.

Gessel, I. M. (1990). Symmetric functions and p-recursiveness.. *J. Comb. Theory, Ser. A*, *53*(2), 257–285.

Getoor, L., & Taskar, B. (2007). *Introduction to statistical relational learning*, Vol. 1. MIT press Cambridge.

Gogate, V., & Domingos, P. M. (2011). Probabilistic theorem proving. In *UAI 2011, Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pp. 256–265. AUAI Press.

Goulden, I. P., & Jackson, D. M. (1986). Labelled graphs with small vertex degrees and p-recursiveness. *SIAM Journal on Algebraic Discrete Methods*, *7*(1), 60–66.

Graedel, E., Otto, M., & Rosen, E. (1997). Two-variable logic with counting is decidable. In *Proceedings of Twelfth Annual IEEE Symposium on Logic in Computer Science*, pp. 306–317. IEEE.

Jaeger, M. (2015). Lower complexity bounds for lifted inference. *TPLP*, *15*(2), 246–263.

Kazemi, S. M., Kimmig, A., Van den Broeck, G., & Poole, D. (2016). New liftable classes for first-order probabilistic inference. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems*, pp. 3117–3125.

Koiran, P., & Perifel, S. (2011). Interpolation in valiant's theory. *Computational Complexity*, *20*(1), 1–20.

Kuusisto, A., & Lutz, C. (2018). Weighted model counting beyond two-variable logic. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018*, pp. 619–628.

Kuzelka, O. (2020). Complex Markov logic networks: Expressivity and liftability. In *Proceedings of the Thirty-Sixth Conference on Uncertainty in Artificial Intelligence, UAI*.

Meert, W., Vlasselaer, J., & Van den Broeck, G. (2016). A relaxed tseitin transformation for weighted model counting. In *Proceedings of the Sixth International Workshop on Statistical Relational AI (StarAI)*, pp. 1–7.

Poole, D. (2003). First-order probabilistic inference. In *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pp. 985–991.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning*, *62*(1-2), 107–136.

Seroul, R. (2000). *Programming for mathematicians*. Springer Science & Business Media.

Stanley, R. P. (1986). What is enumerative combinatorics?. In *Enumerative combinatorics*, pp. 1–63. Springer.

Van den Broeck, G. (2011). On the completeness of first-order knowledge compilation for lifted probabilistic inference. In *Advances in Neural Information Processing Systems*, pp. 1386–1394.

Van den Broeck, G. (2013). *Lifted inference and learning in statistical relational models*. Ph.D. thesis, PhD thesis, KU Leuven.

Van den Broeck, G., Meert, W., & Darwiche, A. (2014). Skolemization for weighted first-order model counting. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pp. 1–10.

Van den Broeck, G., Taghipour, N., Meert, W., Davis, J., & De Raedt, L. (2011). Lifted probabilistic inference by first-order knowledge compilation. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence*, pp. 2178–2185. AAAI Press/International Joint Conferences on Artificial Intelligence.