Complexity of Weighted First-Order Model Counting in the Two-Variable Fragment with Counting Quantifiers: A Bound to Beat

Jan Tóth , Ondřej Kuželka
Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic
{tothjan2, ondrej.kuzelka}@fel.cvut.cz

Abstract

We study the time complexity of weighted first-order model counting (WFOMC) over the logical language with two variables and counting quantifiers. The problem is known to be solvable in time polynomial in the domain size. However, the degree of the polynomial, which turns out to be relatively high for most practical applications, has never been properly addressed. First, we formulate a time complexity bound for the existing techniques for solving WFOMC with counting quantifiers. The bound is already known to be a polynomial with its degree depending on the number of cells of the input formula. We observe that the number of cells depends, in turn, exponentially on the parameters of the counting quantifiers appearing in the formula. Second, we propose a new approach to dealing with counting quantifiers, reducing the exponential dependency to a quadratic one, therefore obtaining a tighter upper bound. It remains an open question whether the dependency of the polynomial degree on the counting quantifiers can be reduced further, thus making our new bound a *bound to beat*.

1 Introduction

The weighted first-order model counting (WFOMC) problem was originally proposed in the area of lifted inference as a method to perform probabilistic inference over statistical relational learning (SRL) models on the lifted level (Van den Broeck et al., 2011). It allowed, among other things, fast learning of various SRL models (Van Haaren et al., 2015). However, its applications have ranged beyond (symbolic) probabilistic reasoning since then, including conjecturing recursive formulas in enumerative combinatorics (Barvínek et al., 2021) and discovering combinatorial integer sequences (Svatoš et al., 2023).

Regardless of the particular application context, WFOMC is also used to define a class of tractable (referred to as *domain-liftable*) modeling languages, i.e., languages which permit WFOMC computation in time polynomial in the domain size. The logical fragment with two variables was the first to be identified as such (Van den Broeck, 2011; Van den Broeck, Meert, and Darwiche, 2014). Negative result proving that logic with three variables contains $\#P_1$-complete counting problems followed (Beame et al., 2015), spawning many attempts to recover at least some of the expressive power provided by three and more variables, yet retaining the domain-liftability property.

Kazemi et al. (2016) introduced two new liftable classes, namely $\mathbf{S}^2\mathbf{FO}^2$ and $\mathbf{S}^2\mathbf{RU}$. Kuusisto and Lutz (2018) extended the two-variable fragment with one function constraint and showed such language to be domain-liftable, too. That result was later generalized to the two-variable fragment with counting quantifiers, denoted by $\mathbf{C}^2$ (Kuželka, 2021). Moreover, several axioms can be added on top of the counting quantifiers, still retaining domain-liftability as well (van Bremen and Kuželka, 2021b; Tóth and Kuželka, 2023; Malhotra and Serafini, 2023; Malhotra, Bizzaro, and Serafini, 2023).

It follows from the domain-liftability of $\mathbf{C}^2$ that WFOMC computation time over formulas from $\mathbf{C}^2$ can be upper-bounded by a polynomial in the domain size. However, it turns out that the polynomial's degree depends exponentially on the particular counting quantifiers appearing in the formula. In this paper, we propose a new approach to dealing with counting quantifiers, when computing WFOMC over $\mathbf{C}^2$, which decreases the degree's dependency on the counting parameters from an exponential to a quadratic one, leading to a super-exponential speedup overall.[1]

2 Background

Notation-wise, we adhere to the standard way of writing both algebraic and logical formulas. For readability purposes, we sometimes use $\cdot$ to denote multiplication, and other times, as is also common, we drop the operation sign. We use boldface letters such as $\mathbf{x}$ to denote vectors and for any $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, 2, \ldots, n\}$.

2.1 First-Order Logic

We work with a function-free subset of first-order logic (FOL). A particular language is defined by a finite set of variables $\mathcal{V}$, a finite set of constants (also called the domain) Δ and a finite set of predicates $\mathcal{P}$. Assuming a predicate $P \in \mathcal{P}$ with arity k, we also write $P/k \in \mathcal{P}$. An *atom* has the form $P(t_1, t_2, \ldots, t_k)$ where $P/k \in \mathcal{P}$ and $t_i \in \Delta \cup \mathcal{V}$ are called *terms*. A *literal* is an atom or its negation. A set of *formulas* can be then defined inductively. Both atoms and literals are formulas. Given some formulas, more complex formulas may be formed by combining them using logical

[1]This paper is accompanied by a technical report available at https://arxiv.org/abs/2404.12905.

connectives or by surrounding them with a universal ($\forall x$) or an existential ($\exists x$) quantifier where $x \in \mathcal{V}$.

A variable x in a formula is called *free* if the formula contains no quantification over x; otherwise, x is called *bound*. A formula is called a *sentence* if it contains no free variables. A formula is called *ground* if it contains no variables.

We use the definition of truth (i.e., semantics) from *Herbrand Logic* (Hinrichs and Genesereth, 2006). A language's *Herbrand Base* (HB) is the set of all ground atoms that can be constructed given the sets $\mathcal{P}$ and Δ. A possible world, usually denoted by ω, is any subset of HB. Atoms contained in a possible world ω are considered to be true, the rest, i.e., those contained in HB $\setminus \omega$, are considered false. The truth value of a more complex formula in a possible world is defined naturally. A possible world ω is a *model* of a formula φ (denoted by $\omega \models \varphi$) if φ is satisfied in ω.

FOL Fragments We often do not work with the entire language of FOL but rather some of its fragments. The simplest fragment we work with is the logic with at most two variables, denoted as $\mathbf{FO}^2$. As the name suggests, any formula from $\mathbf{FO}^2$ contains at most two logical variables.

The second fragment that we consider is the two-variable fragment with cardinality constraints. We keep the restriction of at most two variables, but we recover some of the expressive power of logics with more variables by introducing an additional syntactic construct, namely a *cardinality constraint* (CC). We denote such language as $\mathbf{FO}^2 + \mathbf{CC}$. CCs have the form $(|P| = k)$, where $P \in \mathcal{P}$ and $k \in \mathbb{N}$. Intuitively, such a cardinality constraint is satisfied in ω if there are exactly k ground atoms with predicate P therein.

Finally, the fragment that we pay most of our attention to is the two-variable logic with counting quantifiers, denoted by $\mathbf{C}^2$. A *counting quantifier* is a generalization of the traditional existential quantifier. For a variable $x \in \mathcal{V}$, a quantifier of the form $(\exists^{=k} x)$, where $k \in \mathbb{N}$, is allowed by our syntax. Satisfaction of formulas with counting quantifiers is defined naturally, similar to the satisfaction of cardinality constraints. For example, $\exists^{=k} x\ \psi(x)$ is satisfied in ω if there are exactly k constants $\{A_1, A_2, \dots, A_k\} \subseteq \Delta$ such that $\forall i \in [k] : \omega \models \psi(A_i)$.[2]

Note the distinction between cardinality constraints and counting quantifiers. While the counting formula $\exists^{=k} x\ R(x)$ can be equivalently written using a single cardinality constraint $(|R| = k)$, the formula $\forall x \exists^{=k} y\ R(x, y)$ no longer permits such a simple transformation.

2.2 Weighted First-Order Model Counting

Let us start by formally defining the task that we study.

Definition 1. *(Weighted First-Order Model Counting) Let φ be a formula over a fixed logical language $\mathcal{L}$. Let $(w, \overline{w})$ be a pair of weight functions assigning a weight to each predicate in $\mathcal{L}$. Let n be a natural number. Denote $\mathcal{MOD}(\varphi, n)$ the set of all models of φ on a domain on size n. We define* $\mathrm{WFOMC}(\varphi, n, w, \overline{w}) =$

$$\sum_{\omega \in \mathcal{MOD}(\varphi, n)} \prod_{l \in \omega} w(\mathsf{pred}(l)) \prod_{l \in \mathrm{HB} \setminus \omega} \overline{w}(\mathsf{pred}(l)),$$

where the function pred *maps each literal to its predicate.*

In general, WFOMC is a difficult problem. There exists a sentence with three logical variables, for which the computation is $\#P_1$-complete with respect to n (Beame et al., 2015). However, for some logical languages, WFOMC can be computed in time polynomial in the domain size, which is also referred to as the language being *domain-liftable*.

Definition 2. *(Domain-Liftability) Consider a logical language $\mathcal{L}$. The language is said to be* domain-liftable *if and only if for any fixed $\varphi \in \mathcal{L}$ and any $n \in \mathbb{N}$, it holds that WFOMC($\varphi, n, w, \overline{w}$) can be computed in time polynomial in n.*

Thus, when we study domain-liftable languages, we focus on the time complexity with respect to the domain size, which is assumed to be the only varying input. That is also called *data complexity* of WFOMC in other literature (Beame et al., 2015).

In the remainder of this text, it is often the case that we reduce one WFOMC computation (instance) to another over a different (larger) formula, possibly with fresh (not appearing in the original vocabulary) predicates. Even then, the assumed input language remains fixed in the context of the new WFOMC instance. Moreover, if we do not specify some of the weights for the new predicates, they are assumed to be equal to 1.

The first language proved to be domain-liftable was the language of $\mathbf{UFO}^2$, i.e., universally quantified $\mathbf{FO}^2$ (Van den Broeck, 2011), which was later generalized to the entire $\mathbf{FO}^2$ fragment (Van den Broeck, Meert, and Darwiche, 2014). The original proof, making use of first-order knowledge compilation (Van den Broeck et al., 2011), was later reformulated using *1-types* (which we call *cells*) from logic literature (Beame et al., 2015).

Definition 3. *(Valid Cell) A cell of a first-order formula φ is a maximal consistent conjunction of literals formed from atoms in φ using only a single variable. Moreover, a cell $C(x)$ of a first-order formula $\varphi(x, y)$ is valid if and only if $\varphi(t, t) \wedge C(t)$ is satisfiable for any constant $t \in \Delta$.*

Example 1. *Consider $\varphi = G(x) \vee H(x, y)$.*
Then φ has four cells:

$$C_1(x) = \neg G(x) \wedge \neg H(x, x),$$
$$C_2(x) = \neg G(x) \wedge H(x, x),$$
$$C_3(x) = G(x) \wedge \neg H(x, x),$$
$$C_4(x) = G(x) \wedge H(x, x).$$

However, only cells C_2, C_3 and C_4 are valid.

Another domain-liftable language is $\mathbf{FO}^2 + \mathbf{CC}$. WFOMC over $\mathbf{FO}^2 + \mathbf{CC}$ is solved by repeated calls to an oracle solving WFOMC over $\mathbf{FO}^2$. The number of required calls depends on the arities of predicates that appear in cardinality

[2] Both CCs and counting quantifiers can also be defined to allow inequality operators. However, in the techniques that we study, inequalities are handled by transforming them to equalities (Kuželka, 2021). After such transformation, one must repeatedly solve the case with equalities only. Hence, for brevity, we present only that one case.

constraints. Consider $\Upsilon = \bigwedge_{i=1}^{m} (|R_i| = k_i)$ and let us denote $\alpha(\Upsilon) = \sum_{i=1}^{m} (arity(R_i) + 1)$. For an $\mathbf{FO}^2 + \mathbf{CC}$ formula $\Gamma = \Phi \wedge \Upsilon$ such that $\Phi \in \mathbf{FO}^2$ and Υ same as above, we will require n^α calls to the oracle (Kuželka, 2021).

Having defined WFOMC, valid cells, and function α, we can state known upper bounds for computing WFOMC over $\mathbf{FO}^2$ and $\mathbf{FO}^2 + \mathbf{CC}$. We concentrate those results into a single theorem.

Theorem 1. *Let Γ be an $\mathbf{FO}^2$ sentence with p valid cells. Let $\Upsilon = \bigwedge_{i=1}^{m} (|R_i| = k_i)$ be m cardinality constraints, where $R_1, R_2, \ldots, R_m$ are some predicates from the language of Γ and each $k_i \in \mathbb{N}$. For any $n \in \mathbb{N}$ and any fixed weights $(w, \overline{w})$, WFOMC($\Gamma, n, w, \overline{w}$) can be computed in time $\mathcal{O}(n^{p+1})$, and WFOMC($\Gamma \wedge \Upsilon, n, w, \overline{w}$) can be computed in time $\mathcal{O}(n^{\alpha(\Upsilon)} \cdot n^{p+1})$. Since both the bounds are polynomials in n, both languages are domain-liftable.*

The first bound follows from the cell-based domain-liftability proof (Beame et al., 2015).[3] The second bound follows from Propositions 4 and 5 in Kuželka (2021).

It is important to note that Theorem 1 assumes all mathematical operations to take constant time. Hence, the theorem omits factors relating to *bit complexity*, which Kuželka (2021) also addresses. However, those factors remain the same in all transformations that we consider. Therefore, we omit them for improved readability. For a more detailed discussion on bit complexity, see the accompanying technical report.

2.3 Solving WFOMC with Counting Quantifiers

Yet another domain-liftable language is the language of $\mathbf{C}^2$. WFOMC over $\mathbf{C}^2$ is solved by a reduction to WFOMC over $\mathbf{FO}^2 + \mathbf{CC}$ (Kuželka, 2021). The reduction consists of several steps that we review in the lemmas below. The lemmas concentrate results from other publications. Hence, we make appropriate references to each of them.

First, we review a specialized *Skolemization* procedure for WFOMC (Van den Broeck, Meert, and Darwiche, 2014), which turns an arbitrary $\mathbf{FO}^2$ sentence into a $\mathbf{UFO}^2$ sentence. Since all algorithms for solving WFOMC over $\mathbf{FO}^2$ expect a universally quantified sentence as an input, this is a paramount procedure. Compared to the source publication, we present a slightly modified Skolemization procedure. The modification is due to Beame et al. (2015).

Lemma 1. *Let $\Gamma = Q_1 x_1 Q_2 x_2 \ldots Q_k x_k \Phi(x_1, \ldots, x_k)$ be a first-order sentence in prenex normal form with each quantifier Q_i being either $\forall$ or $\exists$ and Φ being a quantifier-free formula. Denote by j the first position of $\exists$. Let $\boldsymbol{x} = (x_1, \ldots, x_{j-1})$ and $\varphi(\boldsymbol{x}, x_j) = Q_{j+1} x_{j+1} \ldots Q_k x_k \Phi$. Set*

$$\Gamma' = \forall \boldsymbol{x} \left((\exists x_j \varphi(\boldsymbol{x}, x_j)) \Rightarrow A(\boldsymbol{x}) \right),$$

where A is a fresh predicate. Then, for any $n \in \mathbb{N}$ and any weights $(w, \overline{w})$ with $w(A) = 1$ and $\overline{w}(A) = -1$, it holds that

$$\text{WFOMC}(\Gamma, n, w, \overline{w}) = \text{WFOMC}(\Gamma', n, w, \overline{w}).$$

Lemma 1 suggests how to eliminate one existential quantifier. By transforming the implication inside Γ' into a disjunction, we obtain a universally quantified sentence. Repeating the procedure for each sentence in the input formula will eventually lead to one universally quantified sentence.

Next, we present a technique to eliminate negation of a subformula without distributing it inside. The procedure builds on ideas from the Skolemization procedure, and it was presented as Lemma 3.4 in Beame et al. (2015). It was also described as a *relaxed Tseitin transform* in Meert, Vlasselaer, and Van den Broeck (2016).

Lemma 2. *Let $\neg\psi(\boldsymbol{x})$ be a subformula of a first-order logic sentence Γ with k free variables $\boldsymbol{x} = (x_1, \ldots, x_k)$. Let C/k and D/k be two fresh predicates with $w(C) = \overline{w}(C) = w(D) = 1$ and $\overline{w}(D) = -1$. Denote Γ' the formula obtained from Γ by replacing the subformula $\neg\psi(\boldsymbol{x})$ with $C(\boldsymbol{x})$. Let $\Upsilon = (\forall \boldsymbol{x}\ C(\boldsymbol{x}) \vee D(\boldsymbol{x})) \wedge (\forall \boldsymbol{x}\ C(\boldsymbol{x}) \vee \psi(\boldsymbol{x})) \wedge (\forall \boldsymbol{x}\ D(\boldsymbol{x}) \vee \psi(\boldsymbol{x}))$. Then, it holds that*

$$\text{WFOMC}(\Gamma, n, w, \overline{w}) = \text{WFOMC}(\Gamma' \wedge \Upsilon, n, w, \overline{w}).$$

Finally, we move to dealing with counting quantifiers.[4] We start with a single counting quantifier. The approach follows from Lemma 3 in Kuželka (2021).

Lemma 3. *Let Γ be a first-order logic sentence. Let Ψ be a $\boldsymbol{C}^2$ sentence such that $\Psi = \exists^{=k} x\ R(x)$. Let $\Psi' = (|R| = k)$ be a cardinality constraint. Then, it holds that*

$$\text{WFOMC}(\Gamma \wedge \Psi, n, w, \overline{w}) = \text{WFOMC}(\Gamma \wedge \Psi', n, w, \overline{w}).$$

Next, we deal with a specific case of a formula quantified as $\forall\exists^{=k}$. The following lemma was Lemma 2 in Kuželka (2021)

Lemma 4. *Let Γ be a first-order logic sentence. Let Ψ be a $\boldsymbol{C}^2$ sentence such that $\Psi = \forall x \exists^{=k} y\ R(x, y)$. Let Υ be an $\mathbf{FO}^2 + \mathbf{CC}$ sentence defined as*

$$\Upsilon = (|R| = k \cdot n) \wedge (\forall x \forall y\ R(x,y) \Leftrightarrow \bigvee_{i=1}^{k} f_i(x,y))$$
$$\wedge \bigwedge_{1 \le i < j \le k} (\forall x \forall y\ f_i(x,y) \Rightarrow \neg f_j(x,y))$$
$$\wedge \bigwedge_{i=1}^{k} (\forall x \exists y\ f_i(x,y)),$$

where $f_i/2$ are fresh predicates not appearing anywhere else. Then, it holds that

$$\text{WFOMC}(\Gamma \wedge \Psi, n, w, \overline{w}) = \frac{\text{WFOMC}(\Gamma \wedge \Upsilon, n, w, \overline{w})}{(k!)^n}.$$

[3] The state-of-the-art algorithm for computing WFOMC over $\mathbf{FO}^2$, i.e., FastWFOMC, improves the bound considerably in some cases (van Bremen and Kuželka, 2021a). However, as the improvements are not guaranteed in the general case, we work with this bound as an effective worst case. Moreover, as we demonstrate in the experimental section, our new encoding described further in the text improves the FastWFOMC runtime for $\mathbf{C}^2$ sentences reduced to $\mathbf{FO}^2 + \mathbf{CC}$ as well.

[4] For brevity, the counting subformula in Lemmas 3, 4 and 5 contains only a single atom on a predicate R. That does not impede generality as the atom may represent a general subformula φ equated to the atom using an additional universally quantified sentence, i.e., $\forall x\ R(x) \Leftrightarrow \varphi(x)$ or $\forall x \forall y\ R(x,y) \Leftrightarrow \varphi(x,y)$.

Algorithm 1 Converts $\mathbf{C}^2$ formulas into $\mathbf{UFO}^2 + \mathbf{CC}$

Input: Sentence $\Gamma \in \mathbf{C}^2$
Output: Sentence $\Gamma^* \in \mathbf{UFO}^2 + \mathbf{CC}$

```
for all sentence ∃=k x ψ(x) in Γ do
    Apply Lemma 3
end for
for all sentence ∀x∃=k y ψ(x, y) in Γ do
    Apply Lemma 4
end for
for all subformula φ(x) = ∃=k y ψ(x, y) in Γ do
    Create new predicates R/2 and A/1
    Let μ ← ∀x∀y R(x, y) ⇔ ψ(x, y)
    Let ν ← ∀x A(x) ⇔ (∃=k y R(x, y))
    Apply Lemmas 2, 5, and 4 to ν
    Replace φ(x) by A(x)
    Append μ ∧ ν to Γ
end for
for all sentence with an existential quantifier in Γ do
    Apply Lemma 1
end for
return Γ
```

Finally, we present a case that helps deal with an arbitrary counting formula. It was originally presented as Lemma 4 in Kuželka (2021).

Lemma 5. *Let Γ be a first-order logic sentence. Let Ψ be a $\mathbf{C}^2$ sentence such that $\Psi = \forall x \; A(x) \lor (\exists^{=k} y \; R(x,y))$. Define $\Upsilon = \Upsilon_1 \land \Upsilon_2 \land \Upsilon_3 \land \Upsilon_4$ such that*

$$\Upsilon_1 = \forall x \forall y \; \neg A(x) \Rightarrow (R(x,y) \Leftrightarrow B^R(x,y))$$
$$\Upsilon_2 = \forall x \forall y \; (A(x) \land B^R(x,y)) \Rightarrow U^R(y))$$
$$\Upsilon_3 = (|U^R| = k)$$
$$\Upsilon_4 = \forall x \exists^{=k} y \; B^R(x,y),$$

where $U^R/1$ and $B^R/2$ are fresh predicates not appearing anywhere else. Then, it holds that

$$\text{WFOMC}(\Gamma \land \Psi, n, w, \overline{w}) = \frac{\text{WFOMC}(\Gamma \land \Upsilon, n, w, \overline{w})}{\binom{n}{k}}.$$

Algorithm 1 shows how to combine the lemmas above to reduce WFOMC over $\mathbf{C}^2$ to WFOMC over $\mathbf{UFO}^2 + \mathbf{CC}$.

3 An Upper Bound for Existing Techniques

As we have already mentioned above, WFOMC$(\varphi, n, w, \overline{w})$ for $\varphi \in \mathbf{FO}^2 + \mathbf{CC}$ can be computed in time $\mathcal{O}(n^{p+1+\alpha})$, where p is the number of valid cells of φ (Beame et al., 2015; Kuželka, 2021) and $\alpha = \sum_{i=1}^{m} (arity(R_i) + 1)$ with $R_1, R_2, \ldots, R_m$ being all the predicates appearing in cardinality constraints. Let us see how the bound increases when computing WFOMC over $\mathbf{C}^2$.

3.1 A Worked Example on Removing Counting

To be able to compute WFOMC of a particular $\mathbf{C}^2$ sentence, we must first encode the sentence in $\mathbf{UFO}^2 + \mathbf{CC}$. Let us start with an example of applying Algorithm 1 to do just that.

Consider computing WFOMC$(\varphi, n, w, \overline{w})$ for the sentence

$$\varphi = \exists^{=k} x \exists^{=l} y \; \psi(x,y), \tag{1}$$

where ψ is a quantifier-free formula from the two-variable fragment and $k, l \in \mathbb{N}$.

First, let us introduce two new fresh predicates, namely $R/2$ and $P/1$. The predicate R will *replace* the formula ψ and P will do the same for the counting subformula $\exists^{=l} y \; \psi(x,y)$. Specifically, we obtain

$$\varphi^{(1)} = \left(\exists^{=k} x \; P(x)\right) \tag{2}$$
$$\land \left(\forall x \; P(x) \Leftrightarrow \left(\exists^{=l} y \; R(x,y)\right)\right). \tag{3}$$
$$\land \left(\forall x \forall y \; R(x,y) \Leftrightarrow \psi(x,y)\right) \tag{4}$$

While Sentence 2 is already easily encoded using a single cardinality constraint as Lemma 3 suggests, Sentence 3 requires more work. Let us split the sentence into two implications:

$$\varphi^{(2)} = (|P| = k) \land (\forall x \forall y \; R(x,y) \Leftrightarrow \psi(x,y)) \tag{5}$$
$$\land \left(\forall x \; P(x) \Rightarrow \left(\exists^{=l} y \; R(x,y)\right)\right) \tag{6}$$
$$\land \left(\forall x \; P(x) \Leftarrow \left(\exists^{=l} y \; R(x,y)\right)\right). \tag{7}$$

Sentence 6 can easily be rewritten into a form processable by Lemma 5, whereas Sentence 7 will first need to be transformed using Lemma 2, since the sentence can be rewritten as

$$\forall x \; P(x) \lor \neg \left(\exists^{=l} y \; R(x,y)\right).$$

After applying Lemma 2, we obtain

$$\varphi^{(3)} = (|P| = k) \land (\forall x \forall y \; R(x,y) \Leftrightarrow \psi(x,y)) \tag{8}$$
$$\land \left(\forall x \; \overline{P}(x) \lor \left(\exists^{=l} y \; R(x,y)\right)\right) \tag{9}$$
$$\land \left(\forall x \; C(x) \lor \left(\exists^{=l} y \; R(x,y)\right)\right) \tag{10}$$
$$\land \left(\forall x \; D(x) \lor \left(\exists^{=l} y \; R(x,y)\right)\right) \tag{11}$$
$$\land \left(C(x) \lor D(x)\right) \land \left(\forall x \; \overline{P}(x) \Leftrightarrow \neg P(x)\right), \tag{12}$$

with $\overline{w}(D) = -1$. Apart from applying Lemma 2, we also introduced another fresh predicate $\overline{P}/1$, which wraps the negation of P for brevity further down the line.

Next, we need to apply Lemma 5 three times to Sentences 9 through 11. The repetitions can, luckily, be avoided. By the distributive property of conjunctions and disjunctions, we can *factor* $\overline{P}/1, C/1$ and $D/1$ out from the sentences, thus obtaining

$$\varphi^{(4)} = (|P| = k) \land (\forall x \forall y \; R(x,y) \Leftrightarrow \psi(x,y)) \tag{13}$$
$$\land \left(\forall x \; F(x) \Leftrightarrow \left(C(x) \land D(x) \land \overline{P}(x)\right)\right) \tag{14}$$
$$\land \left(\forall x \; F(x) \lor \left(\exists^{=l} y \; R(x,y)\right)\right) \tag{15}$$
$$\land \left(C(x) \lor D(x)\right) \land \left(\forall x \; \overline{P}(x) \Leftrightarrow \neg P(x)\right), \tag{16}$$

Let us denote $\mathcal{T}_l(K, L)$ the transformation result when applying Lemma 5 to a sentence

$$\left(\forall x \; K(x) \lor \left(\exists^{=l} y \; L(x,y)\right)\right),$$

which is now the form of Sentence 15. After that application, our original sentence φ becomes

$$\begin{aligned}\varphi^{(5)} = &(|P| = k) \wedge (\forall x \forall y\ R(x,y) \Leftrightarrow \psi(x,y))\\ &\wedge \mathcal{T}_l(F,R) \wedge \big(\forall x\ F(x) \Leftrightarrow \big(C(x) \wedge D(x) \wedge \overline{P}(x)\big)\big)\\ &\wedge (C(x) \vee D(x)) \wedge \big(\forall x\ \overline{P}(x) \Leftrightarrow \neg P(x)\big).\end{aligned}$$

We are still not done. The result of Lemma 5 still contains another counting quantifier. Specifically, the sentence $\mathcal{T}_l(K,L)$ contains the subformula

$$\forall x \exists^{=l} y\ B^L(x,y),$$

where $B^L/2$ is a fresh predicate. We need Lemma 4 to eliminate this counting construct. Using said lemma will turn $\mathcal{T}_l(K,L)$ into

$$\mathcal{T}_l'(K,L) = \forall x \forall y\ \neg K(x) \Rightarrow \big(L(x,y) \Leftrightarrow B^L(x,y)\big) \tag{17}$$

$$\wedge\ \forall x \forall y\ (K(x) \wedge B^L(x,y)) \Rightarrow U^L(y)) \tag{18}$$

$$\wedge\ \big(|U^L| = l\big) \wedge \big(|B^L| = n \cdot l\big) \tag{19}$$

$$\wedge\ \forall x \forall y\ B^L(x,y) \Leftrightarrow \bigvee_{i=1}^{l} f_i(x,y) \tag{20}$$

$$\wedge \bigwedge_{1 \le i < j \le l} (\forall x \forall y\ f_i(x,y) \Rightarrow \neg f_j(x,y)) \tag{21}$$

$$\wedge \bigwedge_{i=1}^{l} (\forall x \exists y\ f_i(x,y)). \tag{22}$$

We have obtained an $\mathbf{FO}^2 + \mathbf{CC}$ sentence. Unfortunately, we are still unable to directly compute WFOMC for such a formula either. The problem lies in the l sentences making up Formula 22, each containing an existential quantifier. Following Lemma 1, we can replace each $\forall x \exists y\ \ f_i(x,y)$ with $\forall x \forall y\ \neg f_i(x,y) \vee A_i(x)$, where $A_i/1$ is a fresh predicate with $\overline{w}(A_i) = -1$ for each i. Denote $\mathcal{T}_l''(K,L)$ the result of applying such change to $\mathcal{T}_l'(K,L)$. Finally, we obtain a $\mathbf{UFO}^2 + \mathbf{CC}$ sentence

$$\begin{aligned}\varphi^{*} = &(|P| = k) \wedge (\forall x \forall y\ R(x,y) \Leftrightarrow \psi(x,y))\\ &\wedge \mathcal{T}_l''(F,R) \wedge \big(\forall x\ F(x) \Leftrightarrow \big(C(x) \wedge D(x) \wedge \overline{P}(x)\big)\big)\\ &\wedge (C(x) \vee D(x)) \wedge \big(\forall x\ \overline{P}(x) \Leftrightarrow \neg P(x)\big).\end{aligned}$$

3.2 Deriving the Upper Bound

As one can observe, the formula φ^* in the example above has grown considerably compared to its original form. The question is whether the formula growth can influence the asymptotic bound from Theorem 1. At first glance, the answer may seem negative. That is due to the fact that the transformation only extends the vocabulary (which is assumed to be fixed), adds cardinality constraints (which are concentrated in the function α), and increases the length of the input formula (which is also constant with respect to n). However, there is one caveat to be aware of. When extending the vocabulary, we may introduce new valid cells. The vocabulary is fixed once WFOMC computation starts, but if we have formula $\Gamma \wedge \Phi$ such that $\Gamma \in \mathbf{FO}^2 + \mathbf{CC}$ and $\Phi \in \mathbf{C}^2$, then this formula already has p valid cells. Once we construct $\Gamma \wedge \Phi^*$, where $\Phi^* \in \mathbf{FO}^2 + \mathbf{CC}$ is obtained from Φ using Algorithm 1, the new formula will have p^* valid cells and, possibly, $p \le p^*$. If we wish to express a complexity bound for $\mathbf{C}^2$, we should inspect the possible increase in p to obtain p^*.

To deal with an arbitrary $\mathbf{C}^2$ formula, we need to be able to deal with subformulas such as the one in Sentence 3. As one can observe from both the example above and Algorithm 1, encoding such a sentence in $\mathbf{UFO}^2 + \mathbf{CC}$ requires, in order of appearance, Lemmas 2, 5, 4 and 1. Applying Lemmas 2 and 5 introduces only a constant number of fresh predicates. Hence, the increase in p can be expressed by multiplying with a constant β. See the accompanying technical report for the derivation of a value for β.

Although the constant β may increase our polynomial degree considerably, there is another, much more substantial, influence. An application of Lemma 4, which additionally requires Lemma 1 to deal with *unskolemized* formulas such as in Formula 22 will introduce $2k$ new predicates. Although k is a parameter of the counting quantifiers, i.e., part of the language, and the language is assumed to be fixed, the size of the encoding of $\mathbf{C}^2$ in $\mathbf{UFO}^2 + \mathbf{CC}$ obviously depends on k. Hence, the number of cells may also increase with respect to k, and, as we formally state below, it, in fact, does.

Lemma 6. *For any $m \in \mathbb{N}$, there exists a sentence $\Gamma = \varphi \wedge \bigwedge_{i=1}^{m} \big(\forall x \exists^{=k_i} y\ \psi_i(x,y)\big)$ such that the $\mathbf{UFO}^2 + \mathbf{CC}$ encoding of Γ obtained using Lemma 4 has $\mathcal{O}(p \cdot \prod_{i=1}^{m} \gamma(k_i))$ valid cells, where p is the number of valid cells of φ and $\gamma(k) = (k+2) \cdot 2^{k-1}$.*

Proof. Consider the sentence

$$\Gamma = \bigwedge_{i=1}^{m} (\forall x \forall y\ E_i(x,y) \Rightarrow E_i(y,x)) \wedge \tag{23}$$

$$\bigwedge_{i=1}^{m} \big(\forall x \exists^{=k_i} y\ E_i(x,y)\big). \tag{24}$$

In this setting, φ is Formula 23 and it has $p = 2^m$ valid cells. We need to apply Lemmas 4 and 1 m times to encode sentences in Formula 24 into $\mathbf{UFO}^2 + \mathbf{CC}$. Let us investigate one such application.

First, consider valid cells of φ, that contain $E_i(x,x)$ negatively. Then all $f_{ij}(x,x)$ must also be negative (the index j now refers to the predicates introduced in a single application of Lemma 4), which will immediately satisfy all *skolemization clauses* obtained by application of Lemma 1. Hence, the atoms $A_{ij}(x)$ will be allowed to be present either positively or negatively for all j. Thus, the number of such cells will increase 2^{k_i} times.

Second, consider valid cells of φ, that contain $E_i(x,x)$ positively. Then exactly one of $f_{ij}(x,x)$ can be satisfied in each cell. That will cause the number of cells to be multiplied by k_i. Next, for a particular cell, denote t the index such that $f_{it}(x,x)$ is positive in that cell. Then all $A_{ij}(x)$

such that $j \neq t$ will again be free to assume either a positive or a negative form. Only $A_{it}(x)$ will be fixed to being positive. Hence, the number of such cells will further be multiplied by 2^{k_i-1}.

Overall, for a single application of Lemma 4, the number of cells will be $\mathcal{O}(p \cdot 2^{k_i} + p \cdot k_i \cdot 2^{k_i-1}) \in \mathcal{O}(p \cdot (k_i+2) \cdot 2^{k_i-1})$, where we upper bound both partitions of the valid cells of φ by their total number. After repeated application of Lemmas 4 and 1, the bound above directly leads to the bound we sought to prove.[5] □

One last consideration for dealing with all of $\mathbf{C}^2$ is Lemma 3. The lemma adds one new cardinality constraint to the formula, which does not increase the number of cells in any way. It will, however, require more calls to an oracle for WFOMC over $\mathbf{FO}^2$. That influence on the overall bound can still be concentrated in the function α. To distinguish the values from before encoding $\mathbf{C}^2$ and after, let us denote the new value α'.

Finally, we are ready to state the overall time complexity bound for computing WFOMC over $\mathbf{C}^2$.

Theorem 2. *Consider an arbitrary $\mathbf{C}^2$ sentence rewritten as $\varphi = \Gamma \wedge \bigwedge_{i=1}^{m} \left(\forall x\, P_i(x) \Leftrightarrow \left(\exists^{=k_i} y\, R_i(x,y)\right)\right)$, where $\Gamma \in \mathbf{FO}^2 + \mathbf{CC}$. For any $n \in \mathbb{N}$ and any fixed weights $(w, \overline{w})$, $\mathrm{WFOMC}(\varphi, n, w, \overline{w})$ can be computed in time $\mathcal{O}(n^{\alpha'} \cdot n^{1+p\cdot\prod_{i=1}^{m}\beta\cdot\gamma(k_i)})$, where p is the number of valid cells of Γ, α' and β are constants with respect to both n and the counting parameters k_i and $\gamma(k) = (k+2)\cdot 2^{k-1}$.*

Proof. The proof mostly follows from Lemma 6 and the discussion above.

The last thing to show is that the bound on the number of cells derived in Lemma 6 is as general as possible (i.e., there is no other sentence that would invalidate the upper bound). That is straightforward. The sentence Γ shown in the proof of Lemma 6 affords the atoms with the predicates f_i and A_i the highest possible number of degrees of freedom. The truth values of the atoms are determined only by sentences added through the application of Lemmas 4 and 1. Therefore, there cannot be more valid cells, and the bound is as loose as possible, even though the claim in Lemma 6 is only existential.

For a complete proof, see the accompanying technical report. □

4 Improving the Upper Bound

Let us now inspect the bound from Theorem 2. Although it is polynomial in n, meaning that $\mathbf{C}^2$ is, in fact, domain-liftable (Kuželka, 2021), we can see that the number of valid cells (a part of the polynomial's degree) grows exponentially with respect to the counting parameters k_i. In this section, we propose an improved encoding to the one from Lemma 4 which reduces said growth to a quadratic one.

The new encoding does not build on entirely new principles, instead, it takes the existing transformation and makes it more efficient. As in Lemma 4, we will describe the situation for dealing with one $\forall\exists^{=k}$-quantified subformula. The procedure could easily be generalized to having $m \in \mathbb{N}$ such subformulas by repeating the process for each of them independently.

The most significant issue with the current encoding are the *Skolemization* predicates $A_i/1$, which increase the number of valid cells exponentially with respect to k. The new encoding will seek to constrain those predicates so that the increase is reduced. Let us start with a formula

$$\Gamma = \varphi \wedge \forall x \exists^{=k} y\ \psi(x,y), \tag{25}$$

where $\varphi \in \mathbf{FO}^2 + \mathbf{CC}$ and ψ is quantifier-free. Let us also consider the encoding of Γ in $\mathbf{UFO}^2 + \mathbf{CC}$, i.e.,

$$\begin{aligned}\Gamma^* = {} & \varphi \wedge (\forall x \forall y\ R(x,y) \Leftrightarrow \psi(x,y)) \wedge (|R| = n \cdot k) \\ & \wedge \left(\forall x \forall y\ R(x,y) \Leftrightarrow \bigvee_{i=1}^{k} f_i(x,y)\right) \\ & \wedge \bigwedge_{1 \le i < j \le k} (\forall x \forall y\ f_i(x,y) \Rightarrow \neg f_j(x,y)) \\ & \wedge \bigwedge_{i=1}^{k} (\forall x \forall y\ \neg f_i(x,y) \vee A_i(x)),\end{aligned}$$

with fresh predicates $R/2, f_i/2$ and $A_i/1$ and weights $\overline{w}(A_i) = -1$ for all $i \in [k]$.

4.1 Canonical Models

The new encoding will leverage a concept that we call a *canonical model*, which we gradually build in this subsection.

Let ω be a model of Γ^* and $t \in \Delta$ be an arbitrary domain element. Denote $\mathcal{A}^t \subseteq [k]$ the set of indices such that

$$\omega \models \bigwedge_{j \in \mathcal{A}^t} A_j(t) \wedge \bigwedge_{j \in [k] \setminus \mathcal{A}^t} \neg A_j(t).$$

Now, let us transform ω into ω_t, which will be another model of Γ^*.

First, we separate all atoms in ω (atoms true in ω) without the predicates $f_i/2$ and $A_i/1$ into the set $\mathcal{R}_0$, atoms on $A_i/1$ not containing the constant t into $\mathcal{R}_t^A$ and atoms on $f_i/2$ not containing the constant t on the first position into $\mathcal{R}_t^f$.

Second, we define an auxiliary injective function $g_t : \mathcal{A}^t \mapsto [k]$ mapping elements of $\mathcal{A}^t$ to the first $|\mathcal{A}^t|$ positive integers, i.e.,

$$g_t(j) = |\{j' \in \mathcal{A}^t \mid j' \le j\}| \tag{26}$$

Third, we define a set of atoms $\mathcal{A}_t^{new}$ such that

$$\mathcal{A}_t^{new} = \{A_{g_t(j)}(t) \mid \omega \models A_j(t)\},$$

i.e., we accumulate the Skolemization atoms with the constant t that are satisfied in ω and we change their indices to

[5] In the proof, we opted for as simple formula as possible. It would be easy to handle computing WFOMC for Γ by decomposing the problem into m identical and independent problems. It is, however, also easy to envision a case where such decomposition is not as trivial. Consider adding constraints such that for each x and y, there is only one i such that $E_i(x,y)$ is satisfied.

the first $|\mathcal{A}^t|$ positive integers. Next, we do a similar thing for atoms with f_i's and the constant t at the first position. Note that we use the same function g_t that was defined (with respect to $\mathcal{A}^t$) in Equation 26. Hence, we construct a set

$$\mathcal{F}_t^{new} = \{f_{g_t(j)}(t,t') \mid \omega \models f_j(t,t'), t' \in \Delta\}.$$

Finally, we are ready to define the new model of Γ^* as

$$\omega_t = \mathcal{R}_0 \cup \mathcal{A}_t^{new} \cup \mathcal{F}_t^{new} \cup \bigcup_{t' \in \Delta \setminus \{t\}} \left(\mathcal{R}_{t'}^A \cup \mathcal{R}_{t'}^f\right).$$

Lemma 7. *For any $\omega \models \Gamma^*$ and any $t \in \Delta$, ω_t constructed as described above is another model of Γ^*.*

Proof. To prove the claim, it is sufficient to note that we can permute the indices i, and the sentence Γ^* will remain the same. The transformation permutes the indices so that the atoms $A_i(t)$ that are satisfied have the lowest possible indices. □

Now, suppose that we take some model of Γ^* and we repeatedly perform the transformation described above for all domain elements, i.e., for some ordering of the domain such as $\Delta = \{t_1, t_2, \ldots, t_n\}$, we construct ω_{t_1} from ω, then we construct ω_{t_2} from ω_{t_1} and so on, until we obtain $\omega_{t_n} = \omega^*$. Note that several models ω can lead to the same ω^*. Thus, ω^* effectively induces an equivalence class.

Definition 4. *A model ω^* constructed in the way described above is a* canonical model *of Γ^*. Moreover, all models of Γ^* that lead to the same canonical model are called* A-equivalent.[6]

A property of A-equivalent models will be useful in what follows. We formalize it as another lemma.

Lemma 8. *Let ω^* be a canonical model of Γ^*. There are*

$$\prod_{t \in \Delta} \binom{k}{|\mathcal{A}^t|} \tag{27}$$

models that are A-equivalent to ω^*. Moreover, any two A-equivalent models have the same weight.

Proof. It follows from the proof of Lemma 7 that the atoms of A_i's that are true in ω^* have the lowest possible indices. Hence, for a fixed $t \in \Delta$, the number of models that lead to ω^* depends only on the number of ways that we can split the indices between the satisfied and the unsatisfied atoms. There are k indices to choose from, and for a fixed assignment, the set $\mathcal{A}^t$ holds the indices of satisfied $A_j(t)$'s, which leads directly to Equation 27.

The second claim follows from the fact that the transformation of any ω into ω^* does not change the number of true atoms of any given predicate. □

[6]The letter "A" simply refers to the Skolemization predicates that we call A_i, although they could be called anything else.

4.2 The New Encoding

In this subsection, we use the concept of canonical models and observations from Lemma 8 to devise an encoding that counts only canonical models. By weighing them accordingly, we then recover the correct weighted model count of the original problem.

Let us start by introducing new fresh predicates $C_i/1$. We will want an atom $C_j(t)$ to be true if and only if $A_1(t), A_2(t), \ldots A_j(t)$ were the only Skolemization atoms satisfied in a model of Γ^* (the old encoding). Intuitively, the predicates C_i will constrain the models to only correspond to canonical models. Thus, we define C_i's as

$$\Gamma_C = \bigwedge_{j=0}^{k} \forall x\, C_j(x) \Leftrightarrow \left(\bigwedge_{h \in [j]} A_h(x) \wedge \bigwedge_{h \in [k]\setminus[j]} \neg A_h(x)\right).$$

The new encoding can then be described as

$$\Gamma^{new} = \Gamma^* \wedge \Gamma_C \wedge \left(\forall x \bigvee_{j=0}^{k} C_j(x)\right). \tag{28}$$

The final disjunction was added to make sure that we only count canonical models (at least one of C_i's is satisfied).

One more thing to consider is the weights for the predicates C_i. A particular atom $C_j(t)$ is satisfied in a model ω if the model is a canonical model corresponding to $C_j(t)$ according to the sentence Γ_C. Following Lemma 8, such a model represents an entire set of A-equivalent models, each with the same weight. Hence, if $C_j(t)$ is satisfied, we should count the model weight of ω as many times as how many A-equivalent models to ω there are. If $C_j(t)$ is, on the other hand, unsatisfied, we want to keep the weight the same. Therefore, we set $w(C_i) = \binom{k}{i}$ and $\overline{w}(C_i) = 1$ for all $i \in [k]$.

Lemma 9. *For any sentence $\Gamma \in$ **C**^{2} such as the one in Equation 25 and $\Gamma^{new} \in$ **UFO**^{2} + **CC** obtained from Γ as in Equation 28, for any $n \in \mathbb{N}$ and any weights $(w, \overline{w})$ extended for predicates C_i as above, it holds that*

$$\text{WFOMC}(\Gamma, n, w, \overline{w}) = \frac{\text{WFOMC}(\Gamma^{new}, n, w, \overline{w})}{(k!)^n}.$$

Proof. The factor comes from Lemma 4. The rest of the proof follows from the discussion above. For more formal proof, see the accompanying technical report. □

4.3 The Improved Upper Bound

With the new encoding, we can decrease the number of valid cells of a sentence obtained after applying Lemma 4 to a **C**^{2} sentence. Hence, we can improve the upper bound on time complexity of computing WFOMC over **C**^{2}. Using the same notation as in Theorem 2, we can formulate Theorem 3.

Theorem 3. *Consider an arbitrary **C**^{2} sentence rewritten as $\varphi = \Gamma \wedge \Phi$, where $\Gamma \in$ **FO**^{2} + **CC** and $\Phi = \bigwedge_{i=1}^{m} \left(\forall x\, P_i(x) \Leftrightarrow \left(\exists^{=k_i} y\, R_i(x,y)\right)\right)$. For any $n \in \mathbb{N}$ and any fixed weights $(w, \overline{w})$, WFOMC$(\varphi, n, w, \overline{w})$ can be computed in time $\mathcal{O}(n^{\alpha'} \cdot n^{1+p \cdot \prod_{i=1}^{m} \beta \cdot \gamma'(k_i)})$, where p is the number of valid cells of Γ and $\gamma'(k) = \mathcal{O}(k^2 + 2k + 1)$.*

Proof. Let us derive γ'. All other values can be derived identically as in the proof of Theorem 2.

Let us return to the sentence from the proof of Lemma 6, i.e, $\Gamma = \bigwedge_{i=1}^{m} (\forall x \forall y\ E_i(x,y) \Rightarrow E_i(y,x)) \wedge \bigwedge_{i=1}^{m} \left(\forall x \exists^{=k_i} E_i(x,y)\right)$. As we already know, sentence Γ causes the largest increase in the number of valid cells.

With the old encoding, most of the truth values of the Skolemization atoms with predicates A_i were unconstrained, leading to an exponential blowup. Due to Equation 28, that is no longer the case. The sentence Γ_C forces $C_j(t)$ to be true if and only if $A_1(t), A_2(t), \ldots, A_j(t)$ are true and all other $A_i(t)$ with $j < i \leq k$ are false. Hence, if $C_{j'}(t)$ is true, then all other $C_j(t)$ with $j' \neq j$ are false. Moreover, at least one $C_i(t)$ must be true due to the final disjunction in Equation 28.

Therefore, we only have $(k+1)$ possibilities for assigning truth values to atoms with C_i predicates. The truth values of atoms with A_i's are then directly determined without any more degrees of freedom. Hence, for both cases considered in the proof of Lemma 6, we receive a factor $(k+1)$ instead of the exponential. Thus, we obtain $\gamma' = \mathcal{O}(p \cdot (k+1) + p \cdot k \cdot (k+1)) \in \mathcal{O}(p \cdot (k^2 + 2k + 1))$. □

5 Experiments

In this section, we support our theoretical findings by providing time measurements for various WFOMC computations using both the old and the new $\mathbf{C}^2$ encoding. We also provide tables comparing the number of valid cells p, which determines the polynomial degree, giving a more concrete idea of the speedup provided by the new encoding.

For all of our experiments, we used *FastWFOMC.jl*,[7] an open source Julia implementation of the FastWFOMC algorithm (van Bremen and Kuželka, 2021a), which is arguably the state-of-the-art, reported by its authors to outperform the first approach to computing WFOMC in a lifted manner, i.e, ForcLIFT,[8] which is based on knowledge compilation (Van den Broeck et al., 2011). Apart from time measurements for $\mathbf{C}^2$ sentences, we also inspect sentences from one of the domain-liftable $\mathbf{C}^2$ extensions, namely $\mathbf{C}^2$ with the linear order axiom (Tóth and Kuželka, 2023).

Most of our experiments were performed in a single thread on a computer with an AMD Ryzen 5 7500F CPU running at 3.4GHz and having 32 GB RAM. Problems containing the linear order axiom, which have considerably higher memory requirements, were solved using a machine with AMD EPYC 7742 CPU running at 2.25GHz and having 512 GB of RAM.

5.1 Performance Measurements

First, consider a $\mathbf{C}^2$ sentence encoding k-regular undirected graphs without loops, i.e.,

$$\Gamma_1 = (\forall x\ \neg E(x,x)) \wedge (\forall x \forall y\ E(x,y) \Rightarrow E(y,x)) \\ \wedge \left(\forall x \exists^{=k} y\ E(x,y)\right).$$

[7] https://github.com/jan-toth/FastWFOMC.jl
[8] https://dtaid.cs.kuleuven.be/wfomc

k	3	4	5
p_{old}	8	16	32
p_{new}	4	5	6

Table 1: Number of valid cells for k-regular graphs

In our experiments, we simply count the number of such graphs on n vertices, i.e., we set $w(E) = \overline{w}(E) = 1$, and we compute WFOMC$(\Gamma_1, n, w, \overline{w})$ for gradually increasing domain sizes n. Since our new encoding is aimed at improving the runtime with respect to the counting parameter k, we perform the computation for several different parameters k as well.

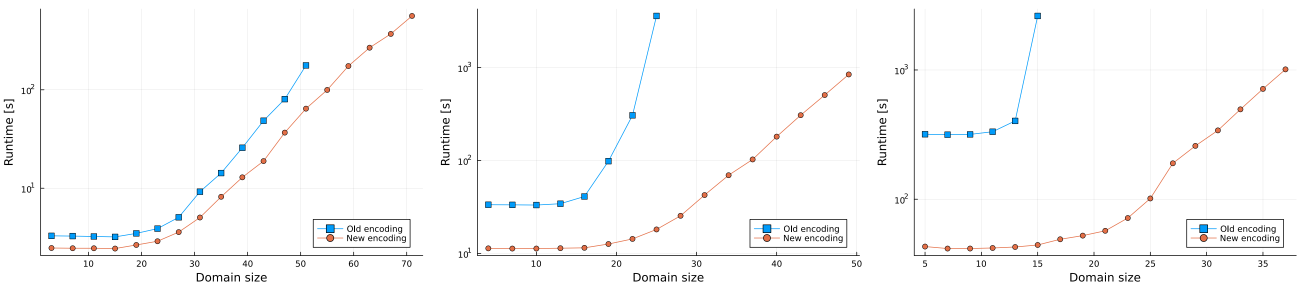
Figure 1 shows the measured runtimes for $k \in \{3, 4, 5\}$. As one can observe, the new encoding surpasses the old in each case. The difference may not seem as distinct for $k = 3$ compared to $k = 4$ or $k = 5$, but it is still substantial. For one, runtime for $n = 51$ already exceeded runtime of 1000 seconds in the case of the old encoding, whereas the new encoding did not reach that value even for $n = 70$. Additionally to the figure, Table 1 shows the number of valid cells p_{old} produced by the old encoding of Γ_1 into $\mathbf{UFO}^2 + \mathbf{CC}$ and p_{new} produced by the new one. As one can observe, e.g., for $k = 5$, the new encoding reduces the runtime from $\mathcal{O}(n^{33})$ to $\mathcal{O}(n^7)$.

See the accompanying technical report for time measurements on additional and more complex $\mathbf{C}^2$ formulas.

Next, let us consider a sentence from the language of $\mathbf{C}^2$ extended with the linear order axiom. Sentence Γ_2 encodes a graph similar to the Model A of the Barabási-Albert model (Albert and Barabási, 2002), an algorithm for generating random networks:

$$\begin{aligned}\Gamma_2 = &(\forall x\ Eq(x,x)) \wedge (|Eq| = n) \\ &\wedge \exists^{=k+1} x\ K(x) \\ &\wedge \forall x\ \neg R(x,x) \\ &\wedge \forall x \forall y\ K(x) \wedge K(y) \wedge \neg Eq(x,y) \Rightarrow R(x,y) \\ &\wedge \forall x \exists^{=k} y\ R(x,y) \\ &\wedge \forall x \forall y\ R(x,y) \wedge \neg (K(x) \wedge K(y)) \Rightarrow y \leq x \\ &\wedge \forall x \forall y\ K(x) \wedge \neg K(y) \Rightarrow x \leq y \\ &\wedge Linear(\leq)\end{aligned}$$

In a sense, the graph encoded by Γ_2 on n vertices is sequentially *grown*. We start by ordering the vertices using the linear order axiom. Then, a complete graph K_{k+1} is formed on the first $k+1$ vertices. Afterward, we start growing the graph by *appending* remaining vertices $i \in \{k+2, k+3, \ldots, n\}$ one at a time. When appending a vertex i, we introduce k outgoing edges that can only connect to the vertices $\{1, 2, \ldots, i-1\}$, i.e., all the new edges have a form (i, j) where $j \in \{1, 2, \ldots, i-1\}$. Ultimately, when counting the number of such graphs, we may not be interested in the same solutions differing by vertex ordering only, so we can divide the final number by $n!$. The ordering through the linear order axiom is, however, a very useful modeling construct—without it, modeling graphs such as



(a) 3-regular (b) 4-regular (c) 5-regular

Figure 1: Runtime for counting k-regular graphs

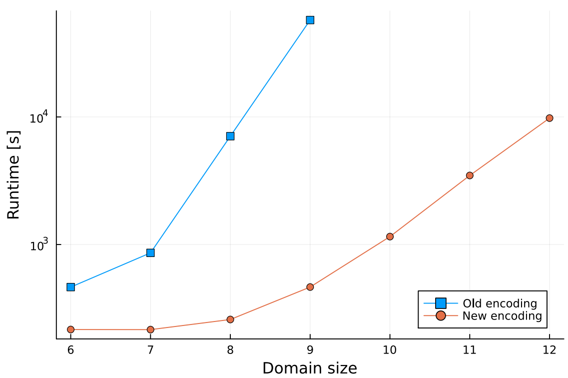


Figure 2: Runtime for counting BA(3) graphs

the one above would likely not be possible in a domain-lifted way.

From now on, let us refer to graphs defined by the sentence Γ_2 as $\text{BA}(k)$. Figure 2 depicts the runtime for counting the graphs BA(3) on n vertices using the old and the new encoding. The problems lead to 16 and 8 valid cells, respectively.

Furthermore, we can easily define a Markov Logic Network (Richardson and Domingos, 2006) on the $\text{BA}(k)$ graphs and, using WFOMC, perform exact lifted inference therein. See the accompanying technical report for such an experiment.

5.2 Remarks on Performance

The experiments confirm that the new encoding is indeed more efficient, outperforming the old encoding on all tested instances (including the additional test scenarios presented in the accompanying technical report). While that should not come as a surprise, since we have derived a bound provably better than the old one, it is also not completely obvious because FastWFOMC uses many algorithmic tricks that partially make up for the inefficiencies of the old encoding. However, not even our new bound allowed us to compute WFOMC within reasonable time for large domain sizes on all the tested problems.

Still, one should keep in mind that the alternative, that is, solving WFOMC by propositionalization to WMC scales extremely poorly, as was repeatedly shown in the lifted inference literature (Meert, Van den Broeck, and Darwiche, 2014). Thus, we must rely on algorithms operating at the lifted level. Our work extends the domains that can be efficiently handled by lifted algorithms, but more work is needed to extend the reach of lifted inference algorithms further.

6 Related Work

This work builds on a long stream of results from the area of lifted inference (Poole, 2003; de Salvo Braz, Amir, and Roth, 2005; Jha et al., 2010; Gogate and Domingos, 2011; Taghipour et al., 2013; Braun and Möller, 2016; Dilkas and Belle, 2023). Particularly, we continue in the line of research into the task of weighted first-order model counting (Van den Broeck et al., 2011; Van den Broeck, 2011; Van den Broeck, Meert, and Darwiche, 2014; Beame et al., 2015; Kazemi et al., 2016; Kuusisto and Lutz, 2018; Kuželka, 2021; van Bremen and Kuželka, 2021b; Malhotra and Serafini, 2022; Tóth and Kuželka, 2023; Malhotra and Serafini, 2023; Malhotra, Bizzaro, and Serafini, 2023).

To the best of our knowledge, there is no other literature available on the exact complexity of computing WFOMC for $\mathbf{C}^2$ sentences. The closest resource is the one proving $\mathbf{C}^2$ to be domain-liftable (Kuželka, 2021), which we directly build upon and, in some sense, extend. Besides that, Malhotra and Serafini (2022) later proposed a slightly different approach to dealing with counting quantifiers, although they did not analyze the method's exact complexity either. However, as shown in the accompanying technical report, their techniques are also super-exponential in the counting parameters, not offering any speedup. Another relevant resource, concerned with designing an efficient algorithm for computing WFOMC over $\mathbf{FO}^2$, is van Bremen and Kuželka (2021a) whose FastWFOMC algorithm remains state-of-the-art and it can be used as a WFOMC oracle required to deal with cardinality constraints.

7 Conclusion

The best existing bound for the time complexity of computing WFOMC over $\mathbf{C}^2$ is polynomial in the domain size (Kuželka, 2021). However, as we point out, the polynomial's degree is exponential in the parameter k of the counting quantifiers. Using the new techniques presented in this paper, we reduce the dependency of the degree on k to a quadratic one, thus achieving a super-exponential speedup of the WFOMC runtime with respect to the counting parameter k.

The new encoding can potentially improve any applications of WFOMC over $\mathbf{C}^2$ or make some applications tractable in the practical sense. We support this statement further in the experimental section, where we provide runtime measurements for computing WFOMC of several $\mathbf{C}^2$ sentences and sentences from a domain-liftable $\mathbf{C}^2$ extension.

It remains an open question whether the complexity can be reduced even further. Thus, we only consider our new bound a bound to beat, and we certainly hope that someone will beat it in the future.

Acknowledgments

This work has received funding from the European Union's Horizon Europe Research and Innovation program under the grant agreement TUPLES No 101070149. JT's work was also supported by a CTU grant no. SGS23/184/OHK3/3T/13.

References

Albert, R., and Barabási, A.-L. 2002. Statistical mechanics of complex networks. *Reviews of modern physics* 74(1):47.

Barvínek, J.; van Bremen, T.; Wang, Y.; Železný, F.; and Kuželka, O. 2021. Automatic conjecturing of p-recursions using lifted inference. In *Inductive Logic Programming: 30th International Conference, ILP 2021, Virtual Event, October 25–27, 2021, Proceedings*, 17–25. Berlin, Heidelberg: Springer-Verlag.

Beame, P.; Van den Broeck, G.; Gribkoff, E.; and Suciu, D. 2015. Symmetric weighted first-order model counting. In *Proceedings of the 34th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '15, 313–328. New York, NY, USA: Association for Computing Machinery.

Braun, T., and Möller, R. 2016. Lifted junction tree algorithm. In *KI 2016: Advances in Artificial Intelligence: 39th Annual German Conference on AI, Klagenfurt, Austria, September 26-30, 2016, Proceedings 39*, 30–42. Springer.

de Salvo Braz, R.; Amir, E.; and Roth, D. 2005. Lifted first-order probabilistic inference. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, 1319–1325. Morgan Kaufmann San Francisco.

Dilkas, P., and Belle, V. 2023. Synthesising recursive functions for first-order model counting: Challenges, progress, and conjectures. In *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR 2023, Rhodes, Greece, September 2-8, 2023*, 198–207.

Gogate, V., and Domingos, P. 2011. Probabilistic theorem proving. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, UAI'11, 256–265. Arlington, Virginia, USA: AUAI Press.

Hinrichs, T., and Genesereth, M. 2006. Herbrand logic. Technical Report LG-2006-02, Stanford University, Stanford, CA. http://logic.stanford.edu/reports/LG-2006-02.pdf.

Jha, A.; Gogate, V.; Meliou, A.; and Suciu, D. 2010. Lifted inference seen from the other side: The tractable features. *Advances in Neural Information Processing Systems* 23.

Kazemi, S. M.; Kimmig, A.; Van den Broeck, G.; and Poole, D. 2016. New liftable classes for first-order probabilistic inference. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, 3125–3133. Red Hook, NY, USA: Curran Associates Inc.

Kuusisto, A., and 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*, 619–628.

Kuželka, O. 2021. Weighted first-order model counting in the two-variable fragment with counting quantifiers. *Journal of Artificial Intelligence Research* 70:1281–1307.

Malhotra, S., and Serafini, L. 2022. Weighted model counting in fo2 with cardinality constraints and counting quantifiers: A closed form formula. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36(5), 5817–5824.

Malhotra, S., and Serafini, L. 2023. Weighted first order model counting with directed acyclic graph axioms. *ArXiv* abs/2302.09830.

Malhotra, S.; Bizzaro, D.; and Serafini, L. 2023. Lifted inference beyond first-order logic. *ArXiv* abs/2308.11738.

Meert, W.; Van den Broeck, G.; and Darwiche, A. 2014. Lifted inference for probabilistic logic programs. Technical report, KU Leuven and University of California.

Meert, W.; Vlasselaer, J.; and Van den Broeck, G. 2016. A relaxed tseitin transformation for weighted model counting. In *International Workshop on Statistical Relational AI*.

Poole, D. 2003. First-order probabilistic inference. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, volume 3, 985–991.

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

Svatoš, M.; Jung, P.; Tóth, J.; Wang, Y.; and Kuželka, O. 2023. On discovering interesting combinatorial integer sequences. In Elkind, E., ed., *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, 3338–3346. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Taghipour, N.; Fierens, D.; Davis, J.; and Blockeel, H. 2013. Lifted variable elimination: Decoupling the operators from the constraint language. *Journal of Artificial Intelligence Research* 47:393–439.

Tóth, J., and Kuželka, O. 2023. Lifted inference with linear order axiom. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37(10), 12295–12304.

van Bremen, T., and Kuželka, O. 2021a. Faster lifting for two-variable logic using cell graphs. In de Campos, C., and Maathuis, M. H., eds., *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume

161 of *Proceedings of Machine Learning Research*, 1393–1402. PMLR.

van Bremen, T., and Kuželka, O. 2021b. Lifted inference with tree axioms. In *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning*, 599–608.

Van den Broeck, G.; Taghipour, N.; Meert, W.; Davis, J.; and De Raedt, L. 2011. Lifted probabilistic inference by first-order knowledge compilation. In Walsh, T., ed., *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, 2178–2185. AAAI Press/International Joint Conferences on Artificial Intelligence.

Van den Broeck, G.; Meert, W.; and Darwiche, A. 2014. Skolemization for weighted first-order model counting. In *Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning*, KR'14, 111–120. AAAI Press.

Van den Broeck, G. 2011. On the completeness of first-order knowledge compilation for lifted probabilistic inference. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, 1386–1394. Red Hook, NY, USA: Curran Associates Inc.

Van Haaren, J.; Van den Broeck, G.; Meert, W.; and Davis, J. 2015. Lifted generative learning of markov logic networks. *Machine Learning* 103(1):27–55.