Lifted Inference with Linear Order Axiom

Jan Tóth, Ondřej Kuželka

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

Abstract

We consider the task of weighted first-order model counting (WFOMC) used for probabilistic inference in the area of statistical relational learning. Given a formula ϕ, domain size n and a pair of weight functions, what is the weighted sum of all models of ϕ over a domain of size n? It was shown that computing WFOMC of any logical sentence with at most two logical variables can be done in time polynomial in n. However, it was also shown that the task is $\#\mathsf{P}_1$-complete once we add the third variable, which inspired the search for extensions of the two-variable fragment that would still permit a running time polynomial in n. One of such extension is the two-variable fragment with counting quantifiers. In this paper, we prove that adding a linear order axiom (which forces one of the predicates in ϕ to introduce a linear ordering of the domain elements in each model of ϕ) on top of the counting quantifiers still permits a computation time polynomial in the domain size. We present a new dynamic programming-based algorithm which can compute WFOMC with linear order in time polynomial in n, thus proving our primary claim.

Introduction

The task of probabilistic inference is at the core of many statistical machine learning problems and much effort has been invested into performing inference faster. One of the techniques, aimed mostly at problems from the area of statistical relational learning (Getoor and Taskar 2007), is lifted inference (Van den Broeck et al. 2021). A very popular way to perform lifted inference is to encode the particular problem as an instance of the weighted first-order model counting (WFOMC) problem. It is worth noting that applications of WFOMC range much wider, making it an interesting research subject in its own right. For instance, it was used to aid in conjecturing recursive formulas in enumerative combinatorics (Barvínek et al. 2021).

Computing WFOMC in the two-variable fragment of first-order logic (denoted as $\mathbf{FO}^2$) can be done in time polynomial in the domain size, which is also referred to as $\mathbf{FO}^2$ being *domain-liftable* (Van den Broeck 2011). Unfortunately, it was also shown that the same does not hold in $\mathbf{FO}^3$ where the problem turns out to be $\#\mathsf{P}_1$-complete in general (Beame et al. 2015). That has inspired a search for extensions of $\mathbf{FO}^2$ that would still be domain-liftable. Several new classes have been identified since then.

Kazemi et al. (2016) introduced $\mathbf{S}^2\mathbf{FO}^2$ and $\mathbf{S}^2\mathbf{RU}$. Kuusisto and Lutz (2018) extended the two-variable fragment with one functionality axiom and showed such language to still be domain-liftable. That result was later generalized to the two-variable fragment with counting quantifiers, denoted $\mathbf{C}^2$(Kuželka 2021). Moreover, van Bremen and Kuželka (2021b) proved that $\mathbf{C}^2$ extended by the tree axiom is still domain-liftable as well.[1]

Another extension of $\mathbf{C}^2$ can be obtained by adding a linear order axiom. Linear order axiom (Libkin 2004) enforces some relation in the language to introduce a linear (total) ordering on the domain elements. Such a constraint is inexpressible using only two variables, requiring special treatment. This logic fragment has also received some attention from logicians (Charatonik and Witkowski 2015).

In this paper, we show that extending $\mathbf{C}^2$ with a linear order axiom yields another domain-liftable language. We present a new dynamic programming-based algorithm for computing WFOMC in $\mathbf{C}^2$ with linear order. The algorithm's running time is polynomial in the domain size meaning that $\mathbf{C}^2$ with linear order is domain-liftable.

Even though our result is mostly of theoretical interest, we still provide some interesting applications and experiments. Among others, we perform exact inference in a Markov Logic Network (Richardson and Domingos 2006) on a random graph model similar to the one of Watts and Strogatz (Watts and Strogatz 1998).[2]

Background

Let us now review necessary concepts, definitions and assumptions as well as notation.

We use boldface letters such as $\mathbf{k}$ to differentiate vectors from scalar values such as n. If we do not name individual vector components such as $\mathbf{k} = (k_1, k_2, \ldots, k_d)$, then the i-th element of $\mathbf{k}$ is denoted by $(\mathbf{k})_i$. Since our vectors only have non-negative entries, the sum of vector elements, i.e.,

[1] Other recent works in lifted inference not directly related to our work presented here are works of van Bremen and Kuželka (2021a), Malhotra and Serafini (2022) and Wang et al. (2022).

[2] This paper is accompanied by a technical report available at https://arxiv.org/abs/2211.01164

$\sum_{i=1}^{d}(\mathbf{k})_i$, always coincides with the L^1-norm. Hence, we use $|\mathbf{k}|$ as a shorthand for the sum. We also introduce special name $\boldsymbol{\delta}_j$ for a vector such that

$$(\boldsymbol{\delta}_j)_i = \begin{cases} 1 \text{ if } i = j, \\ 0 \text{ otherwise.} \end{cases}$$

For a vector $\mathbf{k} = (k_1, k_2, \ldots, k_d)$ with $|\mathbf{k}| = n$,

$$\binom{|\mathbf{k}|}{\mathbf{k}} = \binom{n}{k_1, k_2, \ldots, k_d}$$

denotes the multinomial coefficient. We make use of one non-trivial identity of multinomial coefficients (Berge 1971), namely

$$\sum_{j=1}^{d} \binom{n-1}{\mathbf{k} - \boldsymbol{\delta}_j} = \binom{n}{\mathbf{k}}.$$

We also assume the set of natural numbers $\mathbb{N}$ to contain zero and that $0^0 = 1$. We use $[n]$ to denote the set $\{ 1, 2, \ldots, n \}$.

First-Order Logic

We work with a function-free subset of first-order logic. The language is defined by a finite set of *constants* Δ, a finite set of *variables* $\mathcal{V}$ and a finite set of *predicates* $\mathcal{P}$. If the arity of a predicate $P \in \mathcal{P}$ is k, we also write P/k. An *atom* has the form $P(t_1, t_2, \ldots, t_k)$ where $P/k \in \mathcal{P}$ and $t_i \in \Delta \cup \mathcal{V}$. A *literal* is an atom or its negation. A *formula* is an atom and a literal. More complex formulas may be formed from existing formulas by logical 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. A formula is called a $sentence$ if it contains no free variables. A formula is called *ground* if it contains no variables.

As is customary in computer science, we adopt the *Herbrand semantics* (Hinrichs and Genesereth 2006) with a finite domain. Since we have a finite domain with a one-to-one correspondence to the constant symbols, we denote the domain also with Δ. We denote the *Herbrand base* by HB. We use ω to denote a *possible world*, i.e., any subset of HB. When we wish to restrict a possible world ω to only atoms with a particular predicate P, we write $\omega[P]$.

We work with logical sentences containing at most two variables (the language of $\mathbf{FO}^2$). We assume our $\mathbf{FO}^2$ sentences to be constant-free. Dealing with constants in lifted inference is a challenge in its own right. Treatment of conditioning on evidence as well as using constants in sentences is available in other literature (Van Den Broeck and Davis 2012; Van Haaren et al. 2016).

Weighted Model Counting and Lifted Formulation

Throughout this paper, we study the *weighted first-order model counting*. We will also make use of its propositional variant, the *weighted model counting*. Let us formally define both these tasks.

Definition 1. *(Weighted Model Counting) Let ϕ be a logical formula over some propositional language $\mathcal{L}$. Let* HB *denote the Hebrand base of $\mathcal{L}$ (i.e., the set of all propositional variables). Let $w : \mathsf{HB} \mapsto \mathbb{R}$ and $\overline{w} : \mathsf{HB} \mapsto \mathbb{R}$ be a pair of weightings assigning a positive and a negative weight to each variable in $\mathcal{L}$. We define*

$$\mathsf{WMC}(\phi, w, \overline{w}) = \sum_{\omega \subseteq \mathsf{HB} : \omega \models \phi} \prod_{l \in \omega} w(l) \prod_{l \in \mathsf{HB} \setminus \omega} \overline{w}(l).$$

Definition 2. *(Weighted First-Order Model Counting) Let ϕ be a logical formula over some relational language $\mathcal{L}$. Let n be the domain size. Let* HB *denote the Hebrand base of $\mathcal{L}$ over the domain $\Delta = \{ 1, 2, \ldots, n \}$. Let $\mathcal{P}$ be the set of the predicates of the language $\mathcal{L}$ and let $\mathsf{pred} : \mathsf{HB} \mapsto \mathcal{P}$ map each atom to its corresponding predicate symbol. Let $w : \mathcal{P} \mapsto \mathbb{R}$ and $\overline{w} : \mathcal{P} \mapsto \mathbb{R}$ be a pair of weightings assigning a positive and a negative weight to each predicate in $\mathcal{L}$. We define* $\mathsf{WFOMC}(\phi, n, w, \overline{w}) =$

$$\sum_{\omega \subseteq \mathsf{HB} : \omega \models \phi} \prod_{l \in \omega} w(\mathsf{pred}(l)) \prod_{l \in \mathsf{HB} \setminus \omega} \overline{w}(\mathsf{pred}(l)).$$

Remark 1. *Since for any domain Δ of size n, we can define a bijective mapping π such that $\pi(\Delta) = \{ 1, 2, \ldots, n \}$,* WFOMC *is defined for an arbitrary domain of size n.*

Cells and Domain-Liftability of $\mathbf{FO}^2$

We will not build on the original proof of domain-liftability of $\mathbf{FO}^2$ (Van den Broeck 2011; Van den Broeck, Meert, and Darwiche 2014), but rather on the more recent one (Beame et al. 2015). Let us review some parts of that proof as we make use of them later in the paper.

An important concept is the one of a *cell*.

Definition 3. *A cell of a first-order formula ϕ is a maximally consistent conjunction of literals formed from atoms in ϕ using only a single variable.*

We will denote cells as $C_1(x), C_2(x), \ldots, C_p(x)$ and assume that they are ordered (indexed). Note, however, that the ordering is purely arbitrary.

Example 1. *Consider $\phi = Sm(x) \land Fr(x, y) \Rightarrow Sm(y)$. Then there are four cells:*

$$\begin{aligned} C_1(x) &= Sm(x) \land Fr(x, x), \\ C_2(x) &= \neg Sm(x) \land Fr(x, x), \\ C_3(x) &= \neg Sm(x) \land \neg Fr(x, x), \\ C_4(x) &= Sm(x) \land \neg Fr(x, x). \end{aligned}$$

It turns out, that if we fix a particular assignment of domain elements to the cells and if we then condition on such evidence, the WFOMC computation decomposes into mutually independent and symmetric parts, simplifying the computation significantly.

When we say assignment of domain elements to cells, we mean a domain partitioning allowing empty partitions, that is ordered with respect to a chosen cell ordering. Each partition S_j then holds the constants assigned to the cell C_j. Such partitioning can be captured by a vector. We call such a vector a *partitioning vector* and often shorten the term to a *p-vector*.

Definition 4. *Let $C_1, C_2, \dots, C_p$ be cells of some logical formula. Let n be the number of elements in a domain. A partitioning vector (or a p-vector) of order n is any vector $\boldsymbol{k} \in \mathbb{N}^p$ such that $|\boldsymbol{k}| = n$.*

Moreover, conditioning on some cells may immediately lead to an unsatisfiable formula. To avoid unnecessary computation with such cells, we only work with *valid cells* (van Bremen and Kuželka 2021a).

Definition 5. *A valid cell of a first-order formula $\phi(x, y)$ is a cell of $\phi(x, y)$ and is also a model of $\phi(x, x)$.*

Example 2. *Consider $\phi = F(x, y) \wedge (G(x) \vee H(x))$.*

Cells setting both $G(x)$ and $H(x)$ to false are not valid cells of ϕ.

Let us now introduce some notation for conditioning on particular (valid) cells. Denote

$$\psi_{ij}(x, y) = \psi(x, y) \wedge \psi(y, x) \wedge C_i(x) \wedge C_j(y),$$
$$\psi_k(x) = \psi(x, x) \wedge C_k(x),$$

and define

$$r_{ij} = \mathsf{WMC}(\psi_{ij}(A, B), w', \overline{w}'), \tag{1}$$
$$w_k = \mathsf{WMC}(\psi_k(A), w, \overline{w}), \tag{2}$$

where $A, B \in \Delta$ and the weights w', $\overline{w}'$ are the same as w, $\overline{w}$ except for the atoms appearing in the cells conditioned on. Those weights are set to one, since the weights of the unary and binary reflexive atoms are already accounted for in the w_k terms.

Finally, we can write $\mathsf{WFOMC}(\phi, n, w, \overline{w}) =$

$$\sum_{\mathbf{k} \in \mathbb{N}^p : |\mathbf{k}| = n} \binom{n}{\mathbf{k}} \prod_{i,j \in [p] : i < j} r_{ij}^{(\mathbf{k})_i (\mathbf{k})_j} \prod_{i \in [p]} r_{ii}^{\binom{(\mathbf{k})_i}{2}} w_i^{(\mathbf{k})_i}, \tag{3}$$

which implies that universally quantified $\mathbf{FO}^2$ is domain-liftable since Equation 3 may be evaluated in time polynomial in n. Using a specialized *Skolemization* procedure for WFOMC (Van den Broeck, Meert, and Darwiche 2014), we can easily extend the result to the entire $\mathbf{FO}^2$ fragment.

Cardinality Constraints and Counting Quantifiers

WFOMC can be further generalized to WFOMC *under cardinality constraints* (Kuželka 2021). For a predicate $P \in \mathcal{P}$, we may extend the input formula by one or more cardinality constraints of the type $(|P| \bowtie k)$, where $\bowtie \in \{ \leq, =, \geq \}$ and $k \in \mathbb{N}$. Intuitively, a cardinality constraint $(|P| = k)$ is satisfied in ω if there are exactly k ground atoms with predicate P in ω. Similarly for the inequality signs.

Counting quantifiers are a generalization of the traditional existential quantifier. For a variable $x \in \mathcal{V}$, we allow usage of a quantifier of the form $\exists^{\bowtie k} x$, where $\bowtie \in \{ \leq, =, \geq \}$ and $k \in \mathbb{N}$. Satisfaction of formulas with counting quantifiers is defined naturally, in a similar manner 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 $\forall i \in [k] : \omega \models \psi(A_i)$.

Kuželka (2021) showed $\mathbf{C}^2$ to be a domain-liftable language. That was done by reducing WFOMC in $\mathbf{C}^2$ to WFOMC in $\mathbf{FO}^2$ under cardinality constraints and showing that the two-variable fragment with cardinality constraints is also domain-liftable.

Linear Order Axiom

Assuming logic with equality, we can encode that the predicate R enforces a linear ordering on the domain using the following logical sentences (Libkin 2004):

1. $\forall x : R(x, x)$,
2. $\forall x \forall y : R(x, y) \vee R(y, x)$,
3. $\forall x \forall y : R(x, y) \wedge R(y, x) \Rightarrow (x = y)$,
4. $\forall x \forall y \forall z : R(x, y) \wedge R(y, z) \Rightarrow R(x, z)$.

The last sentence, expressing transitivity of the relation R, is the problematic one as it requires three logical variables. Hence, we will not simply append this axiomatic definition to the input formula but rather make use of a specialized algorithm. However, we must keep the axioms in mind, when constructing cells. Substituting x for both y and z into the axioms above leaves us with (after simplification) a single sentence enforcing reflexivity, i.e., $\forall x : R(x, x)$. Only cells adhering to this constraint can be valid.

Throughout this paper, we denote the constraint that a predicate R introduces a linear order on the domain as $Linear(R)$. For easier readability, we also make use of the traditional symbol $\leq$ for the linear order predicate whenever possible. We also prefer the infix notation rather than the prefix one as it is more commonly used together with $\leq$ sign. We also use $(A < B)$ as a shorthand for $(A \leq B) \wedge \neg(B \leq A)$.

We often write $\phi = \psi \wedge Linear(\leq)$, where we assume ψ to be some logical sentence in $\mathbf{FO}^2$ or $\mathbf{C}^2$ and $\leq$ one of the predicates of the language of ψ. Let us formalize the model of such a sentence.

Definition 6. *Let ψ be a logical sentence possibly containing binary predicate $\leq$. A possible world ω is a model of $\phi = \psi \wedge Linear(\leq)$ if and only if ω is a model of ψ, and $\omega[\leq]$ satisfies the linear order axioms.*

Our usual goal will be to compute WFOMC of ϕ over some domain. In such cases, part of the input will be weightings $(w, \overline{w})$. Since we are treating $\leq$ as a special predicate that is only supposed to enforce an ordering of domain elements in the models of ϕ, we will always assume $w(\leq) = \overline{w}(\leq) = 1$.

One more consideration should be given to our assumption of having equality in the language. That is not a hard requirement since encoding equality in $\mathbf{C}^2$ (or $\mathbf{FO}^2$ with cardinality constraints) is relatively simple, compared to full first-order logic. For example, we may use the axioms:

1. $\forall x : (x = x)$,
2. $\forall x \exists^{=1} y : (x = y)$.

Example 3. *As a simple example of what the linear order axiom allows us to express, consider the sentence $\phi = \forall x \forall y : \psi(x, y) \wedge Linear(\leq)$, where*

$$\psi(x, y) = T(x) \wedge (x \leq y) \Rightarrow T(y).$$

How can we interpret models of ϕ? Due to $Linear(\leq)$, the $\leq$ predicate will define a total ordering on the domain, e.g., $1 \leq 2 \leq \ldots \leq n$. Thus, we can think of the domain as a sequence.

Algorithm 1 IncrementalWFOMC

Input: An $\mathbf{FO}^2$ sentence ϕ, $n \in \mathbb{N}$, weightings $(w, \overline{w})$
Output: $\mathsf{WFOMC}(\phi, n, w, \overline{w})$
Require: $\forall i \in [n] \, \forall \mathbf{k} \in \mathbb{N}^p, |\mathbf{k}| = i : T_i[\mathbf{k}] = 0$

```
1:  for each cell C_j do
2:      T_1[δ_j] = w_j
3:  end for
4:  for i = 2 to n do
5:      for each cell C_j do
6:          for each (k_old, W_old) ∈ T_{i-1} do
7:              W_new ← W_old · w_j · ∏_{l=1}^{p} r_{jl}^{(k_old)_l}
8:              k_new ← k_old + δ_j
9:              T_i[k_new] ← T_i[k_new] + W_new
10:         end for
11:     end for
12: end for
13: return ∑_{k ∈ N^p : |k| = n} T_n[k]
```

The formula $\psi(x, y)$ then seeks to split that sequence into its beginning (head of the sequence) and its end (tail of the sequence). The predicate $T/1$ denotes the tail of the sequence. Whenever there is a constant, for which $T/1$ is set to true in a model (it is part of the tail), then all constants greater also have $T/1$ set to true. Constants, for which $T/1$ is set to false, then belong to the sequence head.

Approach

To prove our main result, we proceed as follows. First, we present a new algorithm based on dynamic programming that computes WFOMC of a universally quantified $\mathbf{FO}^2$ sentence in an incremental manner, and it does so in time polynomial in the domain size. Note, that the assumption of universal quantification is not a limiting one, since we can apply the Skolemization for WFOMC to our input sentence before running the algorithm. Second, we show how to adapt the algorithm to compute WFOMC of a formula $\phi = \psi \wedge Linear(\leq)$, where ψ is a universally quantified $\mathbf{FO}^2$ sentence. And third, we use the algorithm as a new WFOMC oracle in the reductions of WFOMC in $\mathbf{C}^2$ to WFOMC in $\mathbf{FO}^2$, thus proving $\mathbf{C}^2$ extended by a linear order axiom to be domain-liftable.

New Algorithm

Our algorithm for computing $\mathsf{WFOMC}(\phi, n, w, \overline{w})$ for an $\mathbf{FO}^2$ sentence ϕ works in an incremental manner. The domain size is inductively enlarged in a similar way as in the *domain recursion rule* (Van den Broeck 2011; Kazemi et al. 2016). For each domain size i, the WFOMC for each possible p-vector is computed. The results are tracked in a table T_i which maps possible p-vectors to real numbers (the weighted counts). The results are then reused to compute entries in the table T_{i+1}. See Algorithm 1 for details.

To compute an entry $T_{i+1}[\mathbf{u}]$ for a p-vector $\mathbf{u}$, we must find all entries $T_i[\mathbf{k}]$ such that $\mathbf{k} + \boldsymbol{\delta}_j = \mathbf{u}$ and C_j is one of the cells. Intuitively speaking, we will assign the new domain element $(i+1)$ to the cell C_j, which will extend the existing models with new ground atoms containing the new domain element. The models will be extended by atoms corresponding to the subformula $\psi_j(i+1)$ (which, if we are only working with valid cells, are simply the positive literals from C_j) and by atoms corresponding to the subformula $\psi_{jk}(i+1, i')$ for each cell C_k and each domain element already processed (i.e., $1 \leq i' < i+1$). As we can construct the new models by extending the old, we can also compute the new model weight from the old. The weight update can be seen on Line 7 of Algorithm 1.

To prove correctness of Algorithm 1, we prove that its result is the same as is specified in Equation 3. For better readability, we split the proof into an auxiliary lemma, which proves a particular property of table entries at the end of each iteration i, and the actual statement of the algorithm's correctness.

Lemma 1. *At the end of iteration i of the for-loop on lines $4-12$, it holds that*

$$T_i[\boldsymbol{k}] = \binom{i}{\boldsymbol{k}} \prod_{i,j \in [p] : i < j} r_{ij}^{(\boldsymbol{k})_i (\boldsymbol{k})_j} \cdot \prod_{i=1}^{p} r_{ii}^{\binom{(\boldsymbol{k})_i}{2}} w_i^{(\boldsymbol{k})_i},$$

for any $i \geq 2$ and any p-vector $\boldsymbol{k}$ such that $|\boldsymbol{k}| = i$.

Proof. Let us prove Lemma 1 by induction on the iteration number.

First, consider $i = 2$. When entering the loop for the first time, we have $T_1[\boldsymbol{\delta}_j] = w_j$ for each cell C_j. Then, for a particular cell C_j selected on Line 5, there are two cases to consider.

The first case is $\mathbf{k}_{old} = \boldsymbol{\delta}_j$. Then $W_{old} = w_j$ and

$$W_{new} = w_j w_j \left(\prod_{i \in [p] : i \neq j} r_{ji}^0 \right) r_{jj} = 1 \cdot r_{jj} w_j^2.$$

Moreover, $\mathbf{k}_{new} = 2\boldsymbol{\delta}_j$. Since this is the only scenario where we obtain such $\mathbf{k}_{new}$ and since $\binom{2}{2\boldsymbol{\delta}_j} = 1$, we have

$$T_2[2\boldsymbol{\delta}_j] = \binom{2}{2\boldsymbol{\delta}_j} r_{jj} w_j^2.$$

The second possibility is that $\mathbf{k}_{old} = \boldsymbol{\delta}_{j'}$, where $j' \neq j$. Then $W_{old} = w_{j'}$ and $W_{new} = w_j' w_j r_{jj'}$. The new p-vector $\mathbf{k}_{new} = \boldsymbol{\delta}_j + \boldsymbol{\delta}_{j'}$ will also be obtained when the selected cell is $C_{j'}$ and $\mathbf{k}_{old} = \boldsymbol{\delta}_j$. The resulting W_{new} will be the same as above. Those values will be summed together (Line 9) and produce

$$T_2[\boldsymbol{\delta}_j + \boldsymbol{\delta}_{j'}] = 2 \cdot r_{jj'} w_j w_{j'} = \binom{2}{\boldsymbol{\delta}_j + \boldsymbol{\delta}_{j'}} r_{jj'} w_j w_{j'}.$$

Hence, the lemma holds at the end of the first iteration.

Second, assume the claim holds at the end of iteration i. Let us investigate the entry $T_{i+1}[\mathbf{k}]$. For now, consider $\mathbf{k}$ without any zero entries. Then there are p cases that will produce a particular p-vector $\mathbf{k} = (k_1, k_2, \ldots, k_p)$, namely

$$\begin{gathered} \mathbf{k}_{old} = (k_1 - 1, k_2, \ldots, k_p) \text{ and cell } C_1 \\ \mathbf{k}_{old} = (k_1, k_2 - 1, \ldots, k_p) \text{ and cell } C_2 \\ \vdots \\ \mathbf{k}_{old} = (k_1, k_2, \ldots, k_p - 1) \text{ and cell } C_p. \end{gathered}$$

For a particular cell C_j and $\mathbf{k}_{old} = \mathbf{k} - \boldsymbol{\delta}_j$, we have by induction hypothesis:

$$W_{old} = \binom{i}{\mathbf{k} - \boldsymbol{\delta}_j} r_{jj}^{\binom{(\mathbf{k})_j - 1}{2}} w_j^{(\mathbf{k})_j - 1} \prod_{i,l \in [p]: i<l, i \neq j \neq l} r_{il}^{(\mathbf{k})_i (\mathbf{k})_l} \prod_{i \in [p]: i \neq j} r_{ii}^{\binom{(\mathbf{k})_i}{2}} w_i^{(\mathbf{k})_i} \prod_{i \in [p]: i \neq j} r_{ji}^{((\mathbf{k})_j - 1)(\mathbf{k})_i}.$$

Following the weight update on Line 7 and simplifying afterwards, the value will become

$$W_{new} = \binom{i}{\mathbf{k} - \boldsymbol{\delta}_j} \prod_{i,l \in [p]: i<l} r_{il}^{(\mathbf{k})_i (\mathbf{k})_l} \prod_{i \in [p]} r_{ii}^{\binom{(\mathbf{k})_i}{2}} w_i^{(\mathbf{k})_i}.$$

Observe that the product after the multinomial coefficient will be the same for any of the p cases outlined above. Hence, the final new table entry is given by

$$\begin{aligned} T_{i+1}[\mathbf{k}] &= \sum_{j=1}^{p} \binom{i}{\mathbf{k} - \boldsymbol{\delta}_j} \prod_{i,l \in [p]: i<l} r_{il}^{(\mathbf{k})_i (\mathbf{k})_l} \prod_{i \in [p]} r_{ii}^{\binom{(\mathbf{k})_i}{2}} w_i^{(\mathbf{k})_i} \\ &= \prod_{i,l \in [p]: i<l} r_{il}^{(\mathbf{k})_i (\mathbf{k})_l} \prod_{i \in [p]} r_{ii}^{\binom{(\mathbf{k})_i}{2}} w_i^{(\mathbf{k})_i} \sum_{j=1}^{p} \binom{i}{\mathbf{k} - \boldsymbol{\delta}_j} \\ &= \binom{i+1}{\mathbf{k}} \prod_{i,l \in [p]: i<l} r_{il}^{(\mathbf{k})_i (\mathbf{k})_l} \prod_{i \in [p]} r_{ii}^{\binom{(\mathbf{k})_i}{2}} w_i^{(\mathbf{k})_i}, \end{aligned}$$

which is consistent with the claim.

The last thing to consider is if there are some zero entries in $\mathbf{k}$. Suppose there are z of them and w.l.o.g. assume they are on the positions $(p - z + 1), (p - z + 2), \dots p$. Then we obtain a result such that

$$T_{i+1}[\mathbf{k}] = \prod_{i,j \in [p]: i<j} r_{ij}^{(\mathbf{k})_i (\mathbf{k})_j} \prod_{i=1}^{p} r_{ii}^{\binom{(\mathbf{k})_i}{2}} w_i^{(\mathbf{k})_i} \sum_{j=1}^{p-z} \binom{i}{\mathbf{k} - \boldsymbol{\delta}_j}$$

Denote $\mathbf{u}$ the first $p - z$ components of the vector $\mathbf{k} - \boldsymbol{\delta}_j$. Note that $\binom{i}{\mathbf{k} - \boldsymbol{\delta}_j} = \binom{i}{\mathbf{u}}$, since the last z entries are all zeros. Hence, even now it holds that

$$T_{i+1}[\mathbf{k}] = \binom{i+1}{\mathbf{k}} \prod_{i,j \in [p]: i<j} r_{ij}^{(\mathbf{k})_i (\mathbf{k})_j} \prod_{i=1}^{p} r_{ii}^{\binom{(\mathbf{k})_i}{2}} w_i^{(\mathbf{k})_i}.$$

□

Theorem 1. *Algorithm 1 computes* $\mathsf{WFOMC}(\phi, n, w, \overline{w})$ *of a universally quantified* $\boldsymbol{FO}^2$ *sentence* ϕ *in prenex normal form. Moreover, it does so in time polynomial in the domain size* n.

Proof. By Lemma 1, we have

$$T_n[\mathbf{k}] = \binom{n}{\mathbf{k}} \prod_{i,j \in [p]: i<j} r_{ij}^{(\mathbf{k})_i (\mathbf{k})_j} \prod_{i=1}^{p} r_{ii}^{\binom{(\mathbf{k})_i}{2}} w_i^{(\mathbf{k})_i}.$$

On Line 13, all those entries are summed together which produces a formula identical to the one in Equation 3.

As for the second part of the claim. The first loop on lines $1-3$ runs in time $\mathcal{O}(1)$ with respect to n. The large loop on lines $4-12$ runs in $\mathcal{O}(n)$. The first nested loop (lines $5-11$) is again independent of n, and the second (lines $6-10$) runs in $\mathcal{O}(n^p)$. The final sum on Line 13 also runs in $\mathcal{O}(n^p)$. Overall, we can upper bound the algorithm's time complexity by $\mathcal{O}(n^{p+1})$. Hence, the running time is polynomial in the domain size n. □

Enforcing a Linear Order

When adding the linear order axiom to the input sentence ψ, each model of ψ will be with respect to some domain ordering. Assume we find the set Ω of all models for one fixed ordering. Having a domain permutation π,

$$\Omega' = \bigcup_{\omega \in \Omega} \{ \pi(\omega) \}$$

will be the set of all models with respect to the new domain ordering defined by π. Hence, the situation is symmetric for any particular ordering of the domain.

Theorem 2. *Let ϕ be a formula of the form $\phi = \psi(x, y) \wedge Linear(\leq)$, where $\psi(x, y)$ is a universally quantified* $\boldsymbol{FO}^2$ *sentence and $\leq$ is one of its predicates. Let Δ be a domain over which we want to compute* WFOMC.

If $\omega \models \phi$ and π is a permutation of Δ, such that $\pi(\Delta) \neq \Delta$, then $\pi(\omega) \models \phi$, where application of π to a possible world is defined by appropriate substitution of the domain elements in ground atoms. Moreover, $\omega \neq \pi(\omega)$.

Proof. If ω is a model of ϕ, we can partition ω into two disjoint sets: $\omega[\leq]$ holding only atoms with the predicate $\leq$ and $\omega_\psi = \omega \setminus \omega[\leq]$. $\omega[\leq]$ defines an ordering of Δ and ω_ψ is then a model of $\forall x \forall y : \psi(x, y)$ respecting the ordering defined by $\omega[\leq]$. Applying the permutation π to $\omega[\leq]$ will define a different domain ordering.

Since there are no constants in ϕ, $\pi(\omega_\psi)$ will still be a model of $\forall x \forall y : \psi(x, y)$ (we simply apply a different substitution to the variables in ψ). Moreover, since ω_ψ respected the ordering defined by $\omega[\leq]$,$\pi(\omega_\psi)$ will respect the new ordering defined by $\pi(\omega[\leq])$.

Hence $\pi(\omega) = \pi(\omega[\leq]) \cup \pi(\omega_\psi)$ is another model of ϕ and it must be different from ω, because $\pi(\omega[\leq])$ defines a different ordering than $\omega[\leq]$. □

Corollary 1. *To compute* $\mathsf{WFOMC}(\phi, n, w, \overline{w})$*, where $\phi = \psi(x, y) \wedge Linear(\leq)$, we can compute* WFOMC *for one ordered domain of size n and then multiply the result by the factorial of n, since there are $n!$ different permutations of the domain.*

Let us now show that we can compute WFOMC of a formula $\phi = \psi \wedge Linear(\leq)$ for a fixed domain ordering using only slightly modified Algorithm 1. The modified algorithm will take advantage of the fact that when we are processing the i-th domain element, it holds that $i' < i$ for all already processed domain elements i'. Hence, when extending the domain by the constant i (and consequently, extending the models by atoms containing i), the only difference will be in the models of the subformulas $\psi_{ij}(A, B)$, where

$A, B \in \Delta$. The one constant must be "greater" than the other in the sense of the enforced domain ordering. Thus, we only need to redefine r_{ij} to reflect this. Then, we may prove that $\mathbf{FO}^2$ with a linear order axiom is domain-liftable in a similar manner to how we proved correctness of Algorithm 1 for $\mathbf{FO}^2$ alone.

Let us redefine $r_{ij} =$

$$\mathsf{WMC}(\psi_{ij}(A, B) \wedge (B \leq A) \wedge \neg(A \leq B), w', \overline{w}') \quad (4)$$

Theorem 3. $\mathsf{IncrementalWFOMC}$ *with* r_{ij} *values from Equation 4 computes* $\mathsf{WFOMC}(\phi, n, w, \overline{w})$ *of a universally quantified* $\boldsymbol{FO}^2$ *sentence* ϕ *in prenex normal form on the ordered domain* $\Delta = \{ 1 \leq 2 \leq \ldots \leq n \}$. *Moreover, it does so in time polynomial in the domain size* n.

Proof. Let us prove the claim by induction on size of the domain.

The base step is analogical to the one in proof of Lemma 1. More generally speaking, for a domain of a constant size K ($K = 1$ in Algorithm 1), we may simply ground the problem and compute its WMC without any lifting. Since K is a constant with respect to n, we won't exceed the polynomial running time.

The inductive step differs from the one for Lemma 1, but still builds on the same intuition. Now, assume that our algorithm computes WFOMC with linear order for a domain of size i, where the result is stored as the table entries $T_i[\mathbf{k}]$ for all p-vectors $\mathbf{k}$ such that $|\mathbf{k}| = i$ (the final result would be obtained by summing those entries together). Consider processing of the element $(i + 1)$. For a particular cell C_j and a p-vector $\mathbf{k}$, adding the new element will again extend the existing models with new atoms. First, atoms corresponding to the subformula $\psi_j(i+1)$ will be added, hence the old weight must be multiplied by w_j. Second, atoms corresponding to the subformulas $\psi_{jk}(i+1, i')$ for each cell C_k and each processed element $i'(1 \leq i' < i + 1)$. However, only possible worlds satisfying $i' < i + 1$ on top of that, will be models of the input sentence with respect to the fixed domain ordering. That is precisely captured by r_{ij} from Equation 4. Other possible worlds will be assigned zero weight. Hence,

$$W_{new} = W_{old} \cdot w_j \cdot \prod_{l=1}^{p} r_{jl}^{(\mathbf{k})_l}.$$

There are more possible p-vectors $\mathbf{u}$ and cells C_m such that $\mathbf{u} + \boldsymbol{\delta}_m = \mathbf{k} + \boldsymbol{\delta}_j = \mathbf{k}_{new}$. Those all correspond to different, mutually independent models whose weights can be added together. Since we are processing all possible p-vectors, those also correspond to the only existing models.

Therefore, at the end of the final iteration, we will have summed up weights of all existing models of size n. And since we only substituted one value in the original Algorithm 1, the computation still runs in time polynomial in the domain size. $\square$

Theorem 4. *The language of* $\boldsymbol{FO}^2$ *extended by a linear order axiom is domain-liftable.*

Proof. For an input sentence $\phi = \psi \wedge Linear(\leq)$, where ψ is an $\mathbf{FO}^2$ sentence, start with converting ψ to a prenex normal form with each predicate having arity at most 2 (Grädel, Kolaitis, and Vardi 1997). Then apply the Skolemization for WFOMC (Van den Broeck, Meert, and Darwiche 2014) to obtain a sentence of the form $\phi = \forall x \forall y : \psi(x, y) \wedge Linear(\leq)$, where ψ is a quantifier-free formula.

By Theorem 3, we know that Algorithm 1 computes $\mathsf{WFOMC}(\phi, n, w, \overline{w})$ for one fixed ordering of the domain in time polynomial with respect to the domain size. Once we have that value, we may multiply it by $n!$ to obtain the overall WFOMC, as is stated in Corollary 1. The entire computation thus runs in time polynomial in the domain size. $\square$

A Worked Example of IncrementalWFOMC

Let us now use another example of splitting a sequence to demonstrate the work of Algorithm 1. Consider the sentence $\phi = \forall x \forall y : \psi(x, y) \wedge Linear(\leq)$, where ψ is the conjunction of

$$\begin{aligned} &\neg H(x) \vee \neg T(x), \\ &H(y) \wedge (x \leq y) \Rightarrow H(x), \\ &T(x) \wedge (x \leq y) \Rightarrow T(y). \end{aligned}$$

This time, we model a three-way split of a sequence, differentiating its head, tail and middle. We have already seen the third formula, which defines a property of the sequence tail. The second formula does the same for the head. We also require that for each element, at least one of $H/1, T/1$ is set to false. If both were set to true, then one element should be part of both the head and the tail, which is obviously something, we do not want. If they are both set to false, then the element is part of the sequence middle.

Our goal is to compute $\mathsf{WFOMC}(\phi, n, w, \overline{w})$, where $(w, \overline{w})$ are some weight functions. For more clarity in the computations below, we leave the weights as parameters (except for the $\leq$ predicate, whose weights are fixed to one). We will substitute concrete numbers at the end of our example.

First, we construct valid cells of ψ. There are 3 in total:

$$\begin{aligned} C_1(x) &= H(x) \wedge \neg T(x) \wedge (x \leq x) \\ C_2(x) &= \neg H(x) \wedge T(x) \wedge (x \leq x) \\ C_3(x) &= \neg H(x) \wedge \neg T(x) \wedge (x \leq x) \end{aligned}$$

Having valid cells, we need to compute the values r_{ij} and w_k. Since we left the input weight functions as parameters, those cannot be specified numerically. Instead, we use their respective symbols.

Finally, we can start with the pseudocode. Following the loop on Lines 1–3, we obtain the table T_1 as follows:

$$\begin{aligned} T_1[(1, 0, 0)] &= w_1 \\ T_1[(0, 1, 0)] &= w_2 \\ T_1[(0, 0, 1)] &= w_3 \end{aligned}$$

For the main loop on Lines 4–12, we have $i = [2, 3]$ and $j = [1, 2, 3]$.

- Set $i = 2$.
  - Set $j = 1$. Now we iterate over entries in T_1.
    First, we have $\mathbf{k}_{old} = (1, 0, 0)$ and $W_{old} = w_1$.

We compute the new weight as

$$W_{new} \leftarrow W_{old} \cdot w_1 \cdot r_{11}^1 \cdot r_{12}^0 \cdot r_{13}^0 = w_1^2 r_{11}.$$

The new p-vector will be $\mathbf{k}_{new} \leftarrow (2,0,0)$.
The old value $T_2[(2,0,0)] = 0$.
Hence, we will set

$$T_2[(2,0,0)] \leftarrow 0 + w_1^2 r_{11}.$$

Analogically with other key-value pairs, we arrive at

$$T_2[(1,1,0)] \leftarrow 0 + w_1 w_2 r_{12}$$
$$T_2[(1,0,1)] \leftarrow 0 + w_1 w_3 r_{13}$$

– Set $j = 2$. Again, iterate over entries in T_1.
First, we have $\mathbf{k}_{old} = (1,0,0)$ and $W_{old} = w_1$.
We compute the new weight as

$$W_{new} \leftarrow W_{old} \cdot w_2 \cdot r_{21}^1 \cdot r_{22}^0 \cdot r_{23}^0 = w_1 w_2 r_{21}.$$

The new p-vector $\mathbf{k}_{new} \leftarrow (1,1,0)$ already has non-zero value set in T_2, i.e.,

$$T_2[(1,1,0)] = w_1 w_2 r_{12}.$$

Hence, we will now assign

$$T_2[(1,1,0)] \leftarrow w_1 w_2 (r_{12} + r_{21}).$$

Again, analogically for other values:

$$T_2[(0,2,0)] \leftarrow 0 + w_2^2 r_{22}$$
$$T_2[(0,1,1)] \leftarrow 0 + w_2 w_3 r_{23}$$

– After repeating the steps for $j = 3$, we arrive at the complete table T_2 with entries:

$$\begin{aligned}
T_2[(2,0,0)] &= w_1^2 r_{11} \\
T_2[(1,1,0)] &= w_1 w_2 (r_{12} + r_{21}) \\
T_2[(1,0,1)] &= w_1 w_3 (r_{13} + r_{31}) \\
T_2[(0,2,0)] &= w_2^2 r_{22} \\
T_2[(0,1,1)] &= w_2 w_3 (r_{23} + r_{32}) \\
T_2[(0,0,2)] &= w_3^2 r_{33}
\end{aligned}$$

- When performing the computation for $i = 3$, we now iterate over entries in T_2. Hence, for each j, there will now be six p-vector keys and their respective values to process.

  Eventually, we arrive at T_3 such that

$$\begin{aligned}
T_3[(3,0,0)] &= w_1^3 r_{11}^3 \\
T_3[(2,1,0)] &= w_1^2 w_2 r_{11} [r_{12}(r_{12} + r_{21}) + r_{21}^2] \\
T_3[(2,0,1)] &= w_1^2 w_3 r_{11} [r_{13}(r_{13} + r_{31}) + r_{31}^2] \\
T_3[(1,2,0)] &= w_1 w_2^2 r_{22} [r_{21}(r_{21} + r_{12}) + r_{12}^2] \\
T_3[(1,1,1)] &= w_1 w_2 w_3 [r_{12} r_{13} (r_{23} + r_{32}) \\
&\quad + r_{21} r_{23} (r_{13} + r_{31}) + r_{31} r_{32} (r_{12} + r_{21})] \\
T_3[(1,0,2)] &= w_1 w_3^2 r_{33} [r_{31}(r_{31} + r_{13}) + r_{13}^2] \\
T_3[(0,3,0)] &= w_2^3 r_{22}^3 \\
T_3[(0,2,1)] &= w_2^2 w_3 r_{22} [r_{23}(r_{23} + r_{32}) + r_{32}^2] \\
T_3[(0,1,2)] &= w_2 w_3^2 r_{33} [r_{32}(r_{32} + r_{23}) + r_{23}^2] \\
T_3[(0,0,3)] &= w_3^3 r_{33}^3
\end{aligned}$$

Per Line 13, the final result is obtained by summing all the values in T_3 that are written above.

To find the number of three-way sequence splits, we set all weights to one. For unitary weights, we obtain

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \qquad \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}.$$

Plugging those values into T_3 and summing produces

$$\sum_{\mathbf{k} \in \mathbb{N}^3 : |\mathbf{k}| = 3} T_3[\mathbf{k}] = 10,$$

which can be checked to be the correct value, e.g., by using the popular *stars and bars* method.

Domain-Liftability of $\mathbf{C}^2$ with Linear Order

WFOMC in $\mathbf{C}^2$ may be reduced to WFOMC in $\mathbf{FO}^2$ under cardinality constraints. WFOMC under cardinality constraints may then be solved by repeated calls to a WFOMC oracle. As there will only be a polynomial number of such calls in the domain size, it follows that $\mathbf{FO}^2$ with cardinality constraints and also $\mathbf{C}^2$ are domain-liftable (Kuželka 2021).

Since the $\mathbf{C}^2$ domain-liftability proof only relies on a domain-lifted WFOMC oracle, we may use our new algorithm for computing WFOMC with linear order as that oracle, leading to our final result.

Theorem 5. *The language of $\mathbf{C}^2$ extended by a linear order axiom is domain-liftable.*

We omit the proof as it would consist of almost word by word restating of the already available proof on domain-liftability of $\mathbf{C}^2$ (Kuželka 2021) with only cosmetic changes.

Predecessor Relation

Having a domain ordering, an important relation is the one of the immediate predecessor. Denoting $Pred(x,y)$ the predecessor relation, i.e., x is the immediate predecessor of y under the order enforced by $\leq$, we may encode the predecessor relation using the sentences

1. $\forall x : \neg Perm(x,x)$,
2. $\forall x \exists^{=1} y : Perm(x,y)$,
3. $\forall y \exists^{=1} x : Perm(x,y)$,
4. $\forall x \forall y : Pred(x,y) \Rightarrow Perm(x,y)$,
5. $\forall x \forall y : Pred(x,y) \Rightarrow (x \leq y)$,
6. $|Pred| = n - 1$.

We use an auxiliary relation $Perm/2$ for the encoding. $Perm/2$ is assumed to be a fresh predicate symbol and it captures a specific permutation of elements. Each domain element is mapped to its immediate successor in the ordering, except for the last one (as it has no successors). The last element in the ordering is mapped by $Perm/2$ to the very first one, which is the only transition in our permutation from a *greater* element to a *smaller* one. Finally, with the permutation defined, we "copy" all its *smaller-to-greater* transitions over to the predecessor relation. See the online technical report for details as well as generalization of the predecessor relation.

Experiments

To check our results empirically, as well as to assess how our approach scales, we implemented the proposed algorithm in the Julia programming language (Bezanson et al. 2017). The implementation follows the algorithmic approach presented in the paper, with one notable exception. Counting quantifiers and cardinality constraints are not handled by repeated calls to a WFOMC oracle and subsequent polynomial interpolation (Kuželka 2021). Instead, they are processed by introducing a symbolic variable[3] for each cardinality constraint and computing the polynomial (that would be interpolated) explicitly in a single run of the algorithm. We made use of the `Nemo.jl` package (Fieker et al. 2017) for polynomial representation and manipulation.

Inference in Markov Logic Networks

Using IncrementalWFOMC, we can perform exact lifted probabilistic inference over Markov Logic Networks that use the language of $\mathbf{C}^2$ with the linear order axiom. We propose one such network over a random graph model similar to the one of Watts and Strogatz. Then, we present inference results for that network obtained by our algorithm.

First, we review necessary background. Then, we describe our graph model. Finally, we present the computed results.

Markov Logic Networks Markov Logic Networks, often abbreviated as MLNs (Richardson and Domingos 2006), are a popular model from the area of statistical relational learning. An MLN Φ is a set of weighted first-order logic formulas (possibly with free variables) with weights taking on values from the real domain or infinity:

$$\Phi = \{\, (w_1, \alpha_1), (w_2, \alpha_2), \dots, (w_k, \alpha_k) \,\}$$

Given a domain Δ, the MLN defines a probability distribution over possible worlds such as

$$Pr_{\Phi,\Delta}(\omega) = \frac{[\![\omega \models \Phi_\infty]\!]}{Z} \exp\left(\sum_{(w_i,\alpha_i)\in\Phi_{\mathbb{R}}} w_i \cdot N(\alpha_i, \omega) \right)$$

where $\Phi_{\mathbb{R}}$ denotes formulas with real-valued weights (soft constraints), Φ_∞ denotes formulas with infinity-valued weights (hard constraints), $[\![\cdot]\!]$ is the indicator function, Z is the normalization constant ensuring valid probability values and $N(\alpha_i, \omega)$ is the number of substitutions to free variables of α_i that produce a grounding of those free variables that is satisfied in ω. The distribution formula is equivalent to the one of a Markov Random Field (Koller and Friedman 2009). Hence, an MLN along with a domain define a probabilistic graphical model and inference in the MLN is thus inference over that model.

Inference (and also learning) in MLNs is reducible to WFOMC (Van den Broeck, Meert, and Darwiche 2014). For each $(w_i, \alpha_i(\mathbf{x}_i)) \in \Phi_{\mathbb{R}}$, introduce a new formula $\forall \mathbf{x}_i : \xi_i(\mathbf{x}_i) \Leftrightarrow \alpha_i(\mathbf{x}_i)$, where ξ_i is a fresh predicate, and set $w(\xi_i) = \exp(w_i), \overline{w}(\xi_i) = 1$ and $w(Q) = \overline{w}(Q) = 1$ for all other predicates Q. Formulas in Φ_∞ are added to the theory as additional constraints. Denoting the new theory by Γ and a query by ϕ, we can compute the inference as

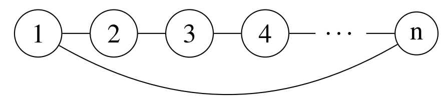
$$Pr_{\Phi,\Delta}(\phi) = \frac{\mathsf{WFOMC}(\Gamma \wedge \phi, |\Delta|, w, \overline{w})}{\mathsf{WFOMC}(\Gamma, |\Delta|, w, \overline{w})}.$$

Watts-Strogatz Model The model of Watts and Strogatz (Watts and Strogatz 1998) is a procedure for generating a random graph of specific properties.

First, having n ordered nodes, each node is connected to K (assumed to be an even integer) of its closest neighbors by undirected edges (discarding parallel edges). If the sequence end or beginning are reached, we wrap to the other end.

Second, each edge (i, j) for each node i is *rewired* with probability β. Rewiring of (i, j) means that node k is chosen at random and the edge is changed to (i, k).

Our Model We start constructing our graph model in the same manner as Watts and Strogatz, with $K = 2$. Ergo, we obtain one cyclic chain going over all our domain elements:



However, we do not perform the rewiring. Instead, we simply add m additional edges at random. Hence, all nodes will be connected by the chain and, moreover, there will be various *shortcuts* as well.

Finally, we add a weighted formula saying that *friends* (friendship is represented by the edges) of smokers also smoke. Intuitively, for large enough weight, our model should prefer those possible worlds where either nobody smokes or everybody does.

Let us now formally state the MLN that we work with:

$$\begin{aligned}
\Phi = \{&(\infty, \forall x : \neg Perm(x,x)), && (5)\\
&(\infty, \forall x \exists y : Perm(x,y)), && (6)\\
&(\infty, \forall y \exists x : Perm(x,y)), && (7)\\
&(\infty, \forall x \forall y : Pred(x,y) \Rightarrow Perm(x,y)), && (8)\\
&(\infty, \forall x \forall y : Pred(x,y) \Rightarrow (x \le y)), && (9)\\
&(\infty, |Perm| = n), && (10)\\
&(\infty, |Pred| = n-1), && (11)\\
&(\infty, \forall x \forall y : Perm(x,y) \Rightarrow E(x,y)), && (12)\\
&(\infty, \forall x \forall y : E(x,y) \Rightarrow E(y,x)), && (13)\\
&(\infty, \forall x : \neg E(x,y)), && (14)\\
&(\infty, |E| = 2n + 2m), && (15)\\
&(\ln w, Sm(x) \wedge E(x,y) \Rightarrow Sm(y))\} && (16)
\end{aligned}$$

Senteces 5 through 11 come from the predecessor definition. They define the basic cyclic chain, albeit a directed one. We reduced the counting quantifiers to ordinary existential quantifiers by adding the cardinality constraint (the sentence) 10 (Kuželka 2021).

Formula 12 copies all $Perm/2$ transitions to $E/2$ and formula 13 makes the edges undirected. Moreover, sentence 14 prohibits loops. Sentence 15 then requires that there are exactly $n + m$ undirected edges in the graph. As all these

[3]Symbolic weights have also been recently used in probabilistic generating circuits (Zhang, Juba, and Van den Broeck 2021) in a similar way to ours.

are hard constraints, every model must define our predefined graph model.

The only soft constraint is sentence 16. By manipulating its weight, we may determine how important it is for the formula to be satisfied in an interpretation.

Inference We can use IncrementalWFOMC to run exact inference in the MLN described above. We may query the probability that a particular domain member (element) smokes. Obviously, the probability will be the same for any domain member. We will thus combine all of these together and query for the probability of there being exactly k smokers, instead.

Denote Γ the theory obtained when we reduce the MLN Φ to WFOMC. We may answer the query as

$$Pr(|Sm| = k) = \frac{\mathsf{WFOMC}(\Gamma \wedge (|Sm| = k), n, w, \overline{w})}{\mathsf{WFOMC}(\Gamma, n, w, \overline{w})}.$$

To relate our model to others which can be modelled without the linear order axiom, we compare the results to inference over a completely random undirected graph with the same number of edges.

Intuitively, completely random graph may form more disconnected components, thus not necessarily preferring the extremes, i.e., either nobody smokes or everybody does. We also keep the parameter m relatively small since, for large m, even the random graph would likely form just one connected component. The MLN over a random graph is defined as follows:

$$\begin{aligned} \Phi' = \{&(\infty, E(x,y) \Rightarrow E(y,x)), \\ &(\infty, \neg E(x,y)), \\ &(\infty, |E| = 2n + 2m), \\ &(\ln w, Sm(x) \wedge E(x,y) \Rightarrow Sm(y))\} \end{aligned}$$

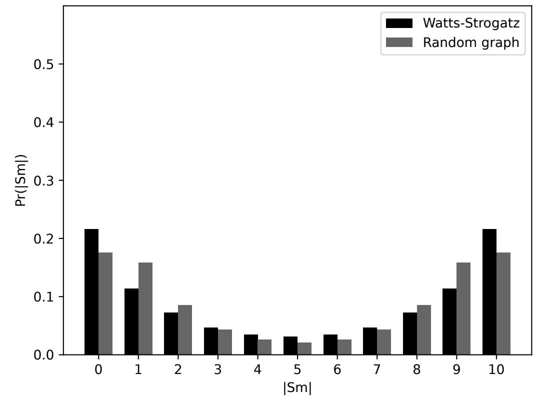
Figure 1 depicts the inference results for a domain size $n = 10$ and weight $w = 3$. The parameter m is set to $5, 8$ and 10, respectively. As one can observe, our model prefers the extreme values more, which is consistent with our intuition above.

Conclusion

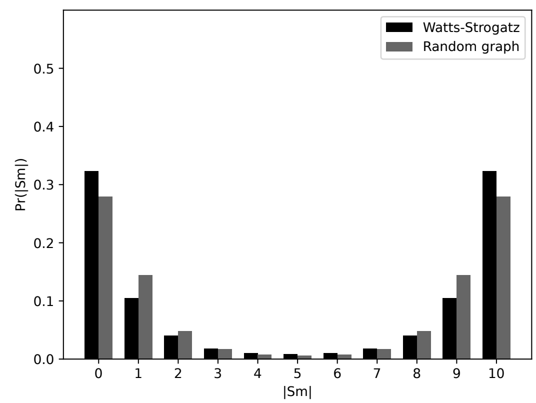
We showed how to compute WFOMC in $\mathbf{C}^2$ with linear order axiom in time polynomial in the domain size. Hence, we showed the language of $\mathbf{C}^2$ extended by a linear order to be domain-liftable. The computation can be performed using our new algorithm, IncrementalWFOMC.

Acknowledgements

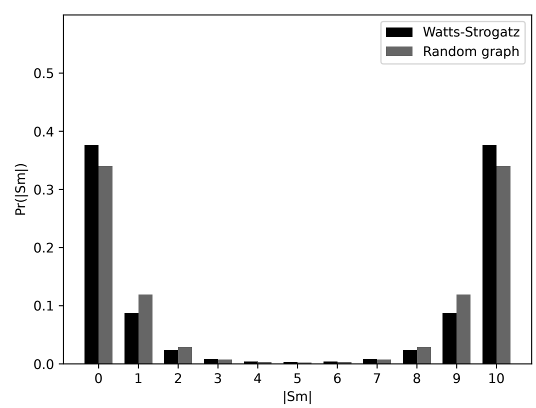
This work was supported by Czech Science Foundation project "Generative Relational Models" (20-19104Y) and partially by the OP VVV project *CZ.02.1.01/0.0/0.0/16_019/0000765* "Research Center for Informatics". JT's work was also supported by a donation from X-Order Lab.



(a) $m = 5$



(b) $m = 8$



(c) $m = 10$

Figure 1: Probability of n smokers for $w = 3$

References

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. ISBN 978-3-030-97453-4.

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. ISBN 9781450327572.

Berge, C. 1971. *Principles of Combinatorics*. ISSN. Elsevier Science. ISBN 9780080955810.

Bezanson, J.; Edelman, A.; Karpinski, S.; and Shah, V. B. 2017. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1): 65–98.

Charatonik, W.; and Witkowski, P. 2015. Two-variable Logic with Counting and a Linear Order. In Kreutzer, S., ed., *24th EACSL Annual Conference on Computer Science Logic (CSL 2015)*, volume 41 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 631–647. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-939897-90-3.

Fieker, C.; Hart, W.; Hofmann, T.; and Johansson, F. 2017. Nemo/Hecke: Computer Algebra and Number Theory Packages for the Julia Programming Language. In *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation*, ISSAC '17, 157–164. New York, NY, USA: ACM.

Getoor, L.; and Taskar, B. 2007. *Introduction to statistical relational learning*. The MIT Press.

Grädel, E.; Kolaitis, P. G.; and Vardi, M. Y. 1997. On the decision problem for two-variable first-order logic. *Bull. Symb. Log.*, 3(1): 53–69.

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.

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. ISBN 9781510838819.

Koller, D.; and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques*. Adaptive computation and machine learning. MIT Press. ISBN 9780262013192.

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.

Libkin, L. 2004. *Elements of Finite Model Theory*, chapter 1.2, 4. Springer. ISBN 3540212027.

Malhotra, S.; and Serafini, L. 2022. Weighted Model Counting in FO^2 with Cardinality Constraints and Counting Quantifiers: A Closed Form Formula. In *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence*, 5817–5824.

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

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. 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. ISBN 9781618395993.

Van Den Broeck, G.; and Davis, J. 2012. Conditioning in First-Order Knowledge Compilation and Lifted Probabilistic Inference. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, 1961–1967. AAAI Press.

Van den Broeck, G.; Kersting, K.; Natarajan, S.; and Poole, D. 2021. *An Introduction to Lifted Probabilistic Inference*. MIT Press.

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. ISBN 1577356578.

Van Haaren, J.; Van den Broeck, G.; Meert, W.; and Davis, J. 2016. Lifted generative learning of Markov logic networks. *Machine Learning*, 103: 27–55.

Wang, Y.; van Bremen, T.; Wang, Y.; and Kuželka, O. 2022. Domain-Lifted Sampling for Universal Two-Variable Logic and Extensions. In *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence*, 10070–10079.

Watts, D. J.; and Strogatz, S. H. 1998. Collective dynamics of 'small-world' networks. *Nature*, 393(6684): 440–442.

Zhang, H.; Juba, B.; and Van den Broeck, G. 2021. Probabilistic generating circuits. In *International Conference on Machine Learning*, 12447–12457. PMLR.