Available online at www.sciencedirect.com
ScienceDirect
Artificial Intelligence 00 (2023) 1–17
Artificial Intelligence

Lifted Inference with Tree Axioms

Timothy van Bremen[a], Ondřej Kuželka[b]

[a]School of Computing, National University of Singapore
[b]Faculty of Electrical Engineering, Czech Technical University in Prague

Abstract

We consider the problem of weighted first-order model counting (WFOMC): given a first-order sentence ϕ and domain size $n \in \mathbb{N}$, determine the weighted sum of models of ϕ over the domain $\{1, \ldots, n\}$. Past work has shown that any sentence using at most two logical variables admits an algorithm for WFOMC that runs in time polynomial in the given domain size [1, 2]. The same property was later also shown to hold for $\mathbf{C}^2$, the two-variable fragment with counting quantifiers [3]. In this paper, we further expand this result to any $\mathbf{C}^2$ sentence ϕ with the addition of a *tree axiom*, stating that some distinguished binary relation in ϕ forms a tree in the graph-theoretic sense.

Keywords:

1. Introduction

Given a first-order sentence ϕ and domain size $n \in \mathbb{N}$, the *first-order model counting* (FOMC) task is to determine the number of models of ϕ over the domain $\{1, \ldots, n\}$. Its weighted variant, *weighted first-order model counting* (WFOMC), assigns each predicate in the signature of ϕ to real-valued positive and negative weights, and asks for the weighted sum of all models of ϕ.

As a natural generalization of its propositional counterpart *weighted model counting* (WMC), inference and learning problems in statistical-relational models such as Markov logic networks [4] and probabilistic logic programs [5] are reducible to WFOMC [2]. In addition, several problems in enumerative combinatorics enjoy a straightforward reduction to FOMC, as long as the characterizing properties of the structure in question can be described by a first-order logic sentence. For example, setting $\phi = \forall x \exists y(E(x, y) \vee E(y, x)) \wedge \forall x \neg E(x, x)$ models directed graphs with no isolated vertices, and the FOMC of ϕ on n gives the number of such labelled graphs with n nodes. A common thread in all of these applications is that one typically has some fixed sentence ϕ, and wants to understand how the complexity of computing the (weighted) first-order model count grows with the domain size n, specified as a unary input. This corresponds broadly to the notion of *data complexity* from the databases and finite model theory literature [6].

In a seminal result, [1] and [2] proved that the data complexity of WFOMC for any sentence in the two-variable fragment of first-order logic, $\mathbf{FO}^2$, lies in the complexity class FP (the functional analogue of P). This proof, initially split across the two papers, was later consolidated in [7]. The authors of the latter paper also exhibited a three-variable sentence for which the data complexity of WFOMC is $\#\mathsf{P}_1$-complete[1], thereby giving a complete characterization of

[1] $\#\mathsf{P}_1$, as defined by [8], is the complexity class of counting problems for NP over a unary input alphabet.

the data complexity for WFOMC in terms of the number of variables in the sentence. Later work [3] also showed the tractability of counting for $\mathbf{C}^2$, the superfragment of $\mathbf{FO}^2$ comprising two-variable sentences with the addition of *counting quantifiers* of the form $\exists^{\bowtie n} x$ for $\bowtie \in \{<, >, =, \geq, \leq\}$.

In this paper, we extend the FP upper bound result beyond $\mathbf{C}^2$ to allow for a feature that is inexpressible in first-order logic [9, Chapter 3.6]: a *tree axiom*, stating that some distinguished binary relation in the sentence forms a tree in the corresponding graph of any model. To do this, we use a tool from the world of graph theory: the weighted version of Kirchhoff's matrix tree theorem [10], which, roughly speaking, states that the weighted sum of all spanning trees of a graph can be quickly computed as the determinant of any minor of the graph's Laplacian matrix.

This extension to tree axioms allows one to model several notable problems in enumerative combinatorics on trees: for example, we may now efficiently answer questions like "how many labelled rooted trees on n nodes have exactly (resp. at most, at least) k leaves"? We show some experimental results of our approach on these types of questions near the end of this paper.

There are also interesting implications for probabilistic inference. Inference algorithms that run efficiently in the input domain size underpin the research field of *lifted inference*, which aims to exploit high-level structure for faster inference in graphical models. Using the results presented here, we are now able to efficiently deal with expressions that state, for example, that possible worlds that are "tree-like" are more likely to occur in the context of a Markov logic network. We later explore some preliminary applications of our results to Markov logic networks in this spirit. We also note that many real-world constructs are naturally modelled as trees: for example, *paths* are trees where every node has at most one child. It is not difficult to imagine Markov logic networks in which we are interested in computing the probabilities of paths along some route, subject to certain constraints modelled by the network.

2. Related Work

This work follows a long line of research in lifted inference on statistical-relational models. Many such papers have focused on operating on first-order versions of graphical models directly [11, 12, 13, 14]. On the other hand, a separate line of work has introduced WFOMC as a natural extension of its propositional counterpart WMC, and has shown how inference in the first-order models mentioned above can be reduced to WFOMC [15]. Within this framework, an effort has been undertaken to understand which fragments of first-order logic allow for tractable inference. The most interesting results for us in the context of this paper are the tractability of two-variable logic [1, 2] along with its extension to functionality constraints [16] and more recently counting quantifiers [3], as well as the fragments $\mathbf{S}^2\mathbf{FO}^2$ and $\mathbf{S}^2\mathbf{RU}$ [17].

Notably, all existing work we are aware of in the lifted inference literature has focused on identifying tractable *fragments* of first-order logic in the strict sense, rather than considering extensions that allow for representing properties beyond what is expressible in first-order logic. In this paper, we take an important first step past this barrier by allowing for the expression of tree axioms.

However, we do note that languages going "beyond" the expressiveness of first-order logic have been investigated in other contexts. Since two-variable logic over finite trees is equivalent in expressive power to navigational XPath, a popular query language for XML documents, the satisfiability problem on this language has received attention in the logic literature [18, 19]. Similarly, in the world of probabilistic databases (closely related to *asymmetric* WFOMC, briefly discussed in the following section), results have been shown for more expressive languages. For instance, query evaluation for the class of *homomorphism-closed queries* on probabilistic graphs has been shown to admit a dichotomy between polytime and #P-hardness depending on the query in question [20, 21]. In particular, this class encompasses negation-free disjunctive Datalog as well as regular path queries. Extending existing (either lower or upper bound) results for WFOMC to richer languages like these remains an open problem.

3. Background

In this section, we give some background on first-order logic and the WFOMC problem, as well as a brief review of the graph theory needed for the paper.

3.1. First-order Logic

3.1.1. Preliminaries

We deal with the function-free, finite domain fragment of first-order logic. An *atom* of arity k takes the form $P(x_1, \ldots, x_k)$, where P/k comes from a vocabulary of *predicates* (also called *relations*), and each argument x_i is a logical variable from a vocabulary of variables. A *literal* is an atom or its negation. A *formula* is formed by connecting one or more literals together using conjunction or disjunction. A formula may optionally be surrounded by one or more quantifiers of the form $\exists x$ or $\forall x$, where x is a logical variable. A logical variable in a formula is said to be *free* if it is not bound by any quantifier. A formula with no free variables is called a *sentence*. A structure $\mathcal{A}$ interprets each predicate in a formula over a finite domain Δ; we denote the restriction of $\mathcal{A}$ to a predicate R_i with $\mathcal{A}_{R_i}$. We follow the usual semantics of first-order logic for deciding whether a structure satisfies (is a model of) a sentence. Given a sentence ϕ and finite domain Δ, we refer to the *grounding* of ϕ over Δ as the propositional formula obtained by replacing universal quantifiers with conjunctions and existential quantifiers with disjunctions, over substitutions of the bound variables with domain elements in Δ.

3.1.2. WFOMC

We are now ready to formally define first-order model counting along with its weighted counterpart.

Definition 1. *The first-order model count (FOMC) of a sentence ϕ over a domain of size n is defined as:*

$$\mathsf{FOMC}(\phi, n) = |\mathsf{models}_n(\phi)|$$

where $\mathsf{models}_n(\phi)$ denotes the set of all models of ϕ over the domain $\Delta = [n] = \{1, \ldots, n\}$.

The *weighted* first-order model count of a sentence is defined in terms of weightings on the predicates. Note that since these weightings are defined on the predicate level, all groundings of the same predicate get the same weight. This corresponds to *symmetric* WFOMC. The case in which every possible ground atom in an interpretation is instead annotated with an individual weight corresponds to *asymmetric* WFOMC and is commonly associated with probabilistic databases, which is beyond the scope of this paper. We instead refer the interested reader to [22] for an overview of that case.

Definition 2. *Denote the set of predicates appearing in a sentence ϕ by P_ϕ. A weighting on ϕ is a pair of mappings $w : P_\phi \to \mathbb{R}$ and $\bar{w} : P_\phi \to \mathbb{R}$.*

Definition 3. *Let $(w, \bar{w})$ be a weighting on a sentence ϕ. The weighted first-order model count (WFOMC) of ϕ over a domain of size n under $(w, \bar{w})$ is:*

$$\mathsf{WFOMC}(\phi, n, w, \bar{w}) = \sum_{\mu \in \mathsf{models}_n(\phi)} \prod_{L \in \mu_T} w(\mathsf{pred}(L)) \prod_{L \in \mu_F} \bar{w}(\mathsf{pred}(L))$$

where μ_T denotes the set of true ground atoms in the model μ, and μ_F the false ground atoms. The notation $\mathsf{pred}(L)$ maps an atom L to its corresponding predicate name.

3.1.3. Cardinality Constraints

Recent work has further generalized the concept of WFOMC to *WFOMC under cardinality constraints* [3]. Although this extended notion of WFOMC is not strictly needed for our core results on trees in the context of two-variable logic, it will be helpful when modelling more complex properties which often require the more expressive fragment $\mathbf{C}^2$, whose details are provided later in the paper. In particular, existing lifted inference methods for $\mathbf{C}^2$ work by reducing WFOMC on a $\mathbf{C}^2$ sentence to WFOMC on an $\mathbf{FO}^2$ sentence under cardinality constraints.

Definition 4. *Let $(w, \bar{w})$ be a weighting on a sentence ϕ, and let C denote a set of constraints of the form $\{|R_1| \bowtie c_1, \ldots, |R_k| \bowtie c_k\}$ for predicates R_i in ϕ, $\bowtie \in \{<, >, =, \geq, \leq\}$, and $c_i \in \mathbb{N}$. The WFOMC of ϕ over a domain of size n under $(w, \bar{w})$ and constraints C is:*

$$\mathsf{WFOMC}(\phi, n, w, \bar{w}, C) = \sum_{\mu \in \mathsf{models}_n(\phi), \mathsf{sat}(\mu, C)} \prod_{L \in \mu_T} w(\mathsf{pred}(L)) \prod_{L \in \mu_F} \bar{w}(\mathsf{pred}(L))$$

where $\mathsf{sat}(\mu, C)$ is true if and only if the number of true atoms in the model μ for each predicate in ϕ satisfies the relevant constraints in C, and other notation is as before.

3.1.4. Modularity

In this paper, we will also be performing reductions between WFOMC problems. We will often want such reductions to be *modular*, a term whose definition we follow from [2].

Property 1 (Modularity)**.** *A reduction is said to be modular if, for every tuple* $(\phi, w, \bar{w})$ *of sentence and weights passed to the reduction with output* $(\phi', w', \bar{w}')$*, we have:*

$$\mathsf{WFOMC}(\phi \wedge \gamma, n, w, \bar{w}) = \mathsf{WFOMC}(\phi' \wedge \gamma, n, w', \bar{w}')$$

for every domain size n and sentence γ*.*

Intuitively, this means that any reduction on ϕ is not invalidated by replacing γ with a different sentence.

3.1.5. Expressiveness

Finally, we conclude with a short remark on the expressiveness of first-order logic that is useful to bear in mind when considering the results presented later in the paper.

Remark 1. *Throughout this paper, we deal with finite domains; this means that any property we can think of can be expressed by grounding it out. This includes tree axioms: we can enumerate all of the possible trees that can be formed on elements of the domain. However, this induces a blow-up in the formula length by a factor exponential in the domain size, thus rendering it of limited practical utility.*

3.2. Data Complexity of WFOMC

Given a fragment $\mathcal{F}$ of first-order logic, we may consider its *data complexity* for WFOMC: fixing the input sentence as some $\phi \in \mathcal{F}$ and weights $(w, \bar{w})$, what is the complexity of computing $\mathsf{WFOMC}(\phi, n, w, \bar{w})$ when treating the domain size n as a unary input?

3.2.1. Data Complexity for $\mathbf{FO}^2$

In Appendix C of [7], the authors show that the data complexity of WFOMC for any sentence in the syntactic fragment of first-order logic limited to two variables is in FP. We reproduce a sketch of this proof below. The term used in the statistical-relational learning literature for logical fragments with FP data complexity is *domain-liftability* [1]. We will also use this term throughout the paper, and occasionally slightly abuse notation by referring to individual sentences (rather than entire fragments) as domain-liftable as well.

Theorem 1 ([7])**.** *The fragment of first-order logic limited to two variables,* $\mathbf{FO}^2$*, is domain-liftable.*

Proof sketch. Suppose that we wish to compute $\mathsf{WFOMC}(\phi, n, w, \bar{w})$ for some input sentence $\phi \in \mathbf{FO}^2$, domain size $n \in \mathbb{N}$, and weights $(w, \bar{w})$. Begin by applying the reduction in [23] and eliminating existential quantifiers as shown in [2] to get a universally quantified sentence $\phi = \forall x \forall y : \psi(x, y)$ such that all atoms in ψ are either unary or binary.

Take the 2^u cells (also called *1-types*) formed as maximal consistent conjunctions of literals over the u predicates in ϕ containing only the variable x, and denote them $C_1, \ldots, C_{2^u}$. Now, consider the possible partitions of $[n]$ into 2^u disjoint sets. Each of these partitions can be thought of as representing a series of assignments of subsets of the n domain elements to each of the cells. Then the models of ϕ over the domain $\Delta = [n]$ are precisely the models of the following sentence:

$$\eta = \bigwedge_{i,j \in [2^u], i<j} \forall x : S_i \forall y : S_j \ (\psi(x, y) \wedge \psi(y, x)) \wedge \bigwedge_{k \in [2^u]} \forall x : S_k \forall y : S_k \ \psi(x, y)$$

where S_i denotes the elements of $[n]$ assigned to the cell C_i, and the notation $\forall x : S_i$ denotes universal quantification limited to the set S_i. Since we know the truth values of the unary and reflexive binary atoms given by each cell assignment C_i, we may simplify the body of each conjunct by replacing every unary and reflexive binary atom with true or false as appropriate. Write $\psi_i(x, y)$ for the simplified version of $\psi(x, y)$ when both x and y are restricted to

taking values from S_i, and $\psi_{ij}(x, y)$ for the simplified version of $\psi(x, y) \wedge \psi(y, x)$ when x and y are restricted to taking values from S_i and S_j respectively. We then have:

$$\eta = \bigwedge_{\substack{i,j\in[2^u]\\ i<j}} \forall x : S_i \forall y : S_j\ \psi_{ij}(x, y) \wedge \bigwedge_{k\in[2^u]} \forall x : S_k \forall y : S_k\ \psi_k(x, y)$$

Observe that each conjunct in the formula above is independent (that is, they do not share any propositional variables when grounded out). Denote $r_{ij} = \mathsf{WMC}(\psi_{ij}(a, b), w, \bar{w})$, $s_k = \mathsf{WMC}(\psi_k(a, b) \wedge \psi_k(b, a), w, \bar{w})$, and $w_k = \mathsf{WMC}(\psi_k(c, c), w, \bar{w})$. Summing across the different possible configurations of cell cardinalities, and multiplying by a multinomial coefficient to account for the different possible selections of domain elements for a given configuration, we get:

$$\mathsf{WFOMC}(\phi, n, w, \bar{w}) = \sum_{n_1+\cdots+n_{2^u}=n} \binom{n}{n_1, \ldots, n_{2^u}} \prod_{\substack{i,j\in[2^u]\\ i<j}} r_{ij}^{n_i n_j} \prod_{i\in[2^u]} s_i^{n_i(n_i-1)/2} w_i^{n_i} \tag{1}$$

Clearly, evaluating this equation can be done in time polynomial in the domain size, and so we have that $\mathbf{FO}^2$ is domain-liftable. □

3.2.2. Data Complexity for $\mathbf{C}^2$

Unfortunately, in many applications, the expressiveness of $\mathbf{FO}^2$ as a modelling language is too limited. Recent work has shown that two-variable logic extended with *counting quantifiers* of the form $\exists^{\bowtie n} x$ for $\bowtie \in \{<, >, =, \geq, \leq\}$ (called $\mathbf{C}^2$), is also domain-liftable [3]. Note that this fragment is strictly more expressive than $\mathbf{FO}^2$: counting quantifiers allow one to express concepts like "each vertex has at most two outgoing edges", whereas without them we could only express that each vertex has at least one outgoing edge, or no outgoing edges at all. To show the domain-liftability of $\mathbf{C}^2$, we first need to understand how to efficiently deal with cardinality constraints when computing the WFOMC.

Theorem 2 ([3], slightly reformulated)**.** *Let ϕ be a first-order logic sentence and C be a set of cardinality constraints (as in Definition 4). If there is a domain-lifted oracle for $\mathsf{WFOMC}(\phi, n, w, \bar{w})$ then $\mathsf{WFOMC}(\phi, w, n, \bar{w}, C)$ can be computed in polynomial time using a polynomial number of queries to this oracle.*

Proof sketch. We observe that the WFOMC of ϕ can be written as a polynomial in the positive and negative weights of the predicates, given respectively by the weightings w, $\bar{w}$. Specifically, suppose that we have predicates P_1/k_1, P_2/k_2, …, P_m/k_m. Then we can write

$$\mathsf{WFOMC}(\phi, n, w, \bar{w}) = \sum_{\mathbf{n}\in\mathcal{D}} A(\mathbf{n}) \mathbf{w}^{\mathbf{n}} \bar{\mathbf{w}}^{\bar{\mathbf{n}}}$$

where we used the notation:

$$\begin{aligned}
\mathbf{n} &\triangleq [n_1, \ldots, n_m],\\
\bar{\mathbf{n}} &\triangleq [n^{k_1} - n_1, n^{k_2} - n_2, \ldots, n^{k_m} - n_m],\\
\mathbf{w}^{\mathbf{n}} \triangleq \prod_{i=1}^{m} w(P_i)^{n_i},\ \mathbf{w}^{\bar{\mathbf{n}}} &\triangleq \prod_{i=1}^{m} \bar{w}(P_i)^{n^{k_i}-n_i},\\
\mathcal{D} &\triangleq \{0, 1, 2, \ldots, n^{k_1}\} \times \cdots \times \{0, 1, 2, \ldots, n^{k_m}\}.
\end{aligned}$$

Here, $A(\mathbf{n})$ then must be the number of possible worlds in which the predicate P_1 has cardinality exactly n_1, the predicate P_2 has cardinality exactly n_2, etc. It follows that if we are given access to the oracle for $\mathsf{WFOMC}(\phi, n, w, \bar{w})$ then we can use Lagrange interpolation to extract $A(\mathbf{n})$ for any $\mathbf{n}$, which then allows us to compute the WFOMC of ϕ with the cardinality constraints C in a straightforward way. For details, we refer to Proposition 5 in [3]. □

Importantly, the theorem above does not require the sentence ϕ to be from $\mathbf{FO}^2$. It can be applied to any domain-liftable sentence. Finally, combining Theorem 1 and Theorem 2 yields the following result.

Theorem 3 ([3])**.** *The fragment of first-order logic limited to two variables with counting quantifiers,* $\mathbf{C}^2$*, is domain-liftable.*

Proof sketch. The principal observation behind this result is that one can transform $\phi \in \mathbf{C}^2$ into a new sentence $\phi' \in \mathbf{FO}^2$, weights $(w', \bar{w}')$, and a set of cardinality constraints C on the predicates in ϕ' such that:

$$\mathsf{WFOMC}(\phi, n, w, \bar{w}) = K_{\phi,n} \cdot \mathsf{WFOMC}(\phi', n, w', \bar{w}', C)$$

for some constant $K_{\phi,n} \in \mathbb{R}$, whose value can be obtained in time polynomial in the domain size. In addition, this transformation has the helpful property that it leaves conjuncts of ϕ containing no counting quantifiers untouched. Since we know that $\mathbf{FO}^2$ is domain-liftable by Theorem 1, and we can handle cardinality constraints by Theorem 2, we are done. We again refer the reader to [3] for the details. □

3.3. Graph Theory

We will also need some tools from graph theory.

3.3.1. Preliminaries

Throughout this section, we assume that graphs are undirected and contain no self-loops. A *tree* is a connected acyclic graph. A *spanning tree* of a graph is a subgraph that is a tree and contains all vertices of the original graph. In the context of this paper, we will uniquely characterise a (weighted) graph by its symmetric *weighted adjacency matrix* W, whose element at position (a, b) denotes the weight on the edge $\{a, b\}$. Unconnected edges are given zero weight. Denote $\mathsf{ver}(W)$ as the set of vertices in W, and $\mathsf{edg}(W)$ as the set of edges with non-zero weight (i.e., $\mathsf{edg}(W) = \{\{i, j\} \mid W_{(i,j)} \neq 0\}$). We associate with W a *weighted degree matrix* $D(W)$:

$$(D(W))_{(a,b)} = \begin{cases} \sum_{k=1}^{|\mathsf{ver}(W)|} (W)_{(a,k)} & \text{if } a = b \\ 0 & \text{otherwise} \end{cases}$$

3.3.2. Kirchhoff's Theorem

Using the definitions above, we can define the *Laplacian matrix* of a graph.

Definition 5 (Laplacian matrix)**.** *The Laplacian* $L(W)$ *of a weighted graph* W *is defined as:*

$$L(W) = D(W) - W$$

The Laplacian matrix is critical to the statement of Kirchhoff's theorem below.

Theorem 4 (Kirchhoff's matrix tree theorem)**.** *Let* W *be a weighted graph, and let* $L(W)$ *denote the Laplacian of* W*. Further, let* $[L(W)]_{i,i}$ *denote the minor of* $L(W)$ *obtained by deleting row* i *and column* i*, for some* $i \leq |\mathsf{ver}(W)|$*. Then the weighted sum of the spanning trees of* W *is the determinant of any of the minors of its Laplacian* $L(W)$*; that is:*

$$\sum_{P \in S(W)} \prod_{\{a,b\} \in P} (W)_{(a,b)} = \det([L(W)]_{i,i})$$

for all $i \leq |\mathsf{ver}(W)|$*, where* $S(W)$ *denotes the set of (weighted) spanning trees of* W*.*

Kirchhoff's theorem gives us a way to quickly sum the weighted spanning trees of a graph in time polynomial in the graph size. However, later in the paper we will also need a way to sum the weighted spanning trees subject to the constraint that certain edges must be present in those trees.

Definition 6. *Let*

$$\mathsf{TreeSum}(W, \mathcal{F}) = \sum_{\substack{T \in S(W) \\ \mathcal{F} \subseteq T}} \prod_{\{a,b\} \in T} W_{(a,b)}$$

denote the weighted sum of spanning trees of W *that contain all edges in* $\mathcal{F} \subseteq \mathsf{edg}(W)$*.*

How can we compute $\mathsf{TreeSum}(W, \mathcal{F})$ in a manner whose runtime is still polynomial in the graph size? The opposite case—that is, imposing the constraint that certain edges are *not* present—is easy: simply set the weight of these edges to zero in W and apply Theorem 4. On the other hand, for the "must-be-present" case, we can contract the forced edges of the graph while keeping track of their weights. Suppose we are given the set of edges $\mathcal{F} \subseteq \mathsf{edg}(W)$ whose presence we wish to force in the spanning trees of the graph W. We may assume without loss of generality that $\mathcal{F}$ is acyclic; otherwise we know immediately $\mathsf{TreeSum}(W, \mathcal{F}) = 0$. With this assumption in mind, pick an edge $e_1 = \{i, j\} \in \mathcal{F}$. Then we may update W by deleting the edge $\{i, j\}$ and the vertex j, and redirecting any edge that originally connected j to a third vertex k to i. If i and k are already connected, we distinguish between two possible cases: if $\{i, k\} \notin \mathcal{F}$ and $\{j, k\} \notin \mathcal{F}$, we simply add the weight of the edge $\{j, k\}$ to the existing edge $\{i, k\}$.[2] Otherwise, exactly one of $\{i, k\}$ or $\{j, k\}$ must lie in $\mathcal{F}$ (they cannot both lie in $\mathcal{F}$, since if they did $\mathcal{F}$ would contain a cycle, contradicting the assumption above). In this case, we simply delete whichever edge is not contained in $\mathcal{F}$, and delete $\{i, j\}$ and the vertex j, since the edge not in $\mathcal{F}$ cannot form part of any spanning tree, as the other two edges in the triangle are forced to be included which would induce a cycle.

Repeating this process for every edge $e_i \in \mathcal{F}$, we eventually end up with a new graph W' in which all of the forced edges have been contracted. We may then compute $\mathsf{TreeSum}(W, \mathcal{F})$ as:

$$\mathsf{TreeSum}(W, \mathcal{F}) = w(e_1) \dots w(e_n) \det([L(W')]_{1,1})$$

where $w(e_i)$ denotes the original weight of the edge e_i in W.

We illustrate the process above with the following simple example: consider the triangle graph K_3 with weights 2, 3, and 4 on the edges, and suppose we wish to compute the weighted sum of spanning trees while forcing inclusion of the edge with weight 2. Clearly there are two such spanning trees, so the answer we should get is $(2 \cdot 3) + (2 \cdot 4) = 14$.

Now consider applying the procedure described above: we contract the edge with weight 2, and obtain a graph with a single edge with weight $3 + 4 = 7$. This obviously has the single spanning tree consisting of the single edge of weight 7. Multiplying in the edge we contracted of weight 2 as described above, we get $2 \cdot 7 = 14$ as desired.

4. Approach

We now come to the primary results of this paper. In this section, we will show how to efficiently compute the WFOMC of a $\mathbf{C}^2$ sentence containing either of the following axioms:

- $\mathsf{Tree}(R, \mathit{Leaf})$, expressing that the (symmetric, antireflexive) binary relation R forms a tree with leaves defined by the unary relation Leaf.
- $\mathsf{DirectedRootedTree}(\mathit{Root}, E, \mathit{Leaf})$, expressing that the binary relation E represents directed edges of a rooted tree, with root node given by the unary relation Root, and leaves given by the unary relation Leaf. The edges are directed from the leaves towards the root of the tree.
- $\mathsf{BinaryOrderedTree}(\mathit{Root}, \mathit{Left}, \mathit{Right}, \mathit{Leaf})$, expressing that the binary relations Left and Right represent directed edges of a full binary rooted ordered tree, with root node given by the unary relation Root, and leaves given by the unary relation Leaf.

As we shall see, the second axiom expressible by conjoining the first axiom (Tree) with some appropriately-chosen sentences in $\mathbf{C}^2$, and the third axiom in turn follows by building on the second axiom. Thus, only the first axiom will require a full algorithmic treatment. The next two will then follow as a consequence of Theorem 3.

[2] The reason why this is sound is as follows. No tree can contain the edges $\{i, j\}, \{i, k\}, \{j, k\}$ at the same time or else it would contain a cycle. Moreover, there is a natural bijection between the trees which contain the edge $\{i, k\}$ and those that contain the edge $\{j, k\}$, which maps trees with weight $W \cdot w(\{i, k\})$ to trees with weight $W \cdot w(\{j, k\})$ (here W is the product of all other edges in the tree except for $\{i, k\}$ or $\{j, k\}$). The bijection simply takes a tree containing $\{i, k\}$ and replaces it by $\{j, k\}$. It is then not difficult to see the correctness of the procedure described here.

4.1. Trees

In this section, we prove that the addition of a tree axiom $\mathsf{Tree}(R, \mathit{Leaf})$ to a $\mathbf{C}^2$ sentence preserves domain-liftability. Our approach will be as follows: first, we will show that the addition of a "limited" version of the tree axiom that does not allow for quantification of the leaves (i.e. $\mathsf{Tree}(R)$ rather than $\mathsf{Tree}(R, \mathit{Leaf})$) can be added to an $\mathbf{FO}^2$ sentence while preserving domain-liftability (Lemma 1). We will then build off of this lemma to show the same for $\mathbf{C}^2$ (Lemma 2). Finally, we will leverage this result to prove our final theorem, showing that we can add the "full" tree axiom ($\mathsf{Tree}(R, \mathit{Leaf})$) to any $\mathbf{C}^2$ sentence and still compute the WFOMC in time polynomial in the domain size (Theorem 5). We begin by formalizing the semantics of the two tree axioms—both the limited version $\mathsf{Tree}(R)$ as well as the full axiom $\mathsf{Tree}(R, \mathit{Leaf})$.

Definition 7. *Let ϕ be an arbitrary first-order sentence, possibly containing some binary relation R. Then a structure $\mathcal{A}$ is a model of the sentence $\xi = \phi \wedge \mathsf{Tree}(R)$ if and only if:*

1. *$\mathcal{A}$ is a model of ϕ, and*
2. *the relation defined by $\mathcal{A}_R$ is antireflexive and symmetric, and*
3. *the relation R forms an R-tree in $\mathcal{A}$: that is, $\mathcal{A}_R$ is a tree when interpreted as an undirected graph*

The concept is illustrated with the following simple example.

Example 1. *Let $\xi = \mathsf{Tree}(R)$. Then, by Cayley's formula, $\mathsf{FOMC}(\xi, n) = n^{n-2}$, the number of trees on n nodes [24].*

We can now extend this definition to the full axiom that allows for expression of properties on the leaves.

Definition 8. *Let ϕ be an arbitrary first-order sentence, possibly containing some binary relation R and unary relation Leaf. Then a structure $\mathcal{A}$ is a model of the sentence $\xi = \phi \wedge \mathsf{Tree}(R, \mathit{Leaf})$ if and only if:*

1. *$\mathcal{A}$ is a model of $\phi \wedge \mathsf{Tree}(R)$, and*
2. *the leaves of the tree defined by $\mathcal{A}_R$ are precisely the elements in $\mathcal{A}_{\mathit{Leaf}}$*

With these definitions in mind, we are now ready to prove our domain-liftability results, starting with the lemma for the limited tree axiom on two-variable sentences without counting quantifiers.

Lemma 1. *Let η be a sentence in $\mathbf{FO}^2$, and let $\xi = \eta \wedge \mathsf{Tree}(R)$. Then ξ is domain-liftable.*

Proof. We will follow the notation and general approach from Theorem 1. Fix our domain $\Delta = \{1, \ldots, n\}$. First, enforce the symmetry and antireflexiveness of R by defining:

$$\begin{aligned}\phi = \eta \wedge \\ &\forall x : \neg R(x, x) \wedge \\ &\forall x \forall y : R(x, y) \rightarrow R(y, x)\end{aligned}$$

After transforming ϕ to the form $\phi = \forall x \forall y : \psi(x, y)$, recall from Equation (1) that we can write the WFOMC of ϕ as:

$$\mathsf{WFOMC}(\phi, n, w, \bar{w}) = \sum_{n_1 + \cdots + n_{2^u} = n} \binom{n}{n_1, \ldots, n_{2^u}} \prod_{\substack{i,j \in [2^u] \\ i<j}} r_{ij}^{n_i n_j} \prod_{i \in [2^u]} s_i^{n_i(n_i-1)/2} w_i^{n_i}$$

for suitably defined r_{ij}, s_i, and w_i terms.

We will now need to adapt this approach so that we account only for the models of ϕ that satisfy the tree property on R. Suppose we fix some cell assignment $C : [n] \rightarrow [2^u]$ of the domain elements to one of the 2^u possible cells. All of the notation below will assume this fixed cell assignment. This assignment also induces a unique cardinality configuration $(n_1, \ldots, n_{2^u})$, where each n_i is the cardinality $|C^{-1}(i)|$ of the preimage of i in C. For $a, b \in \Delta$, denote:

$$\varphi(a, b) = \begin{cases} \psi_{ij}(a, b) & \text{if } i = C(a),\ j = C(b) \text{ and } i \neq j \\ \psi_i(a, b) \wedge \psi_i(b, a) & \text{if } i = C(a) = C(b) \end{cases}$$

Define $\mathcal{F}$ to be the set of all domain element pairs $\{a, b\} \in \Delta^2$ for which $\mathsf{WMC}(\varphi(a,b) \wedge \neg R(a,b), w, \bar{w}) = 0$. Intuitively, $\mathcal{F}$ can be viewed as the pairs of vertices between which R-edges are forced to exist in any model of ϕ. Note that $\mathcal{F}$ is itself a symmetric and antireflexive relation, as a result of it satisfying the corresponding axioms in ϕ, and is dependent only on C.

Now, consider some R-tree T over the vertices defined by Δ such that $\mathcal{F} \subseteq T$: as discussed above, trees satisfying this property are the only ones that can occur as part of a model of ϕ under the cell assignment C. We will compute the weighted sum w_T of every model $\mathcal{A}$ of ϕ under the assignment C, that satisfies $T = \mathcal{A}_R$ (in other words, the WFOMC of ϕ under C limited to models that contain exactly the tree T). We can write:

$$\begin{aligned} w_T &= \prod_{i \in [2^u]} w_i^{n_i} \prod_{\{a,b\} \in T} \mathsf{WMC}(\varphi(a,b) \wedge R(a,b), w, \bar{w}) \prod_{\{a,b\} \notin T} \mathsf{WMC}(\varphi(a,b) \wedge \neg R(a,b), w, \bar{w}) \\ &= \prod_{i \in [2^u]} w_i^{n_i} \prod_{\{a,b\} \in \mathcal{F}} \mathsf{WMC}(\varphi(a,b) \wedge R(a,b), w, \bar{w}) \prod_{a<b, \{a,b\} \in T \setminus \mathcal{F}} \frac{\mathsf{WMC}(\varphi(a,b) \wedge R(a,b), w, \bar{w})}{\mathsf{WMC}(\varphi(a,b) \wedge \neg R(a,b), w, \bar{w})} \cdot \\ &\quad \prod_{a<b, \{a,b\} \notin \mathcal{F}} \mathsf{WMC}(\varphi(a,b) \wedge \neg R(a,b), w, \bar{w}) \end{aligned}$$

Now, summing both sides of the equation across all trees containing $\mathcal{F}$:

$$\begin{aligned} \sum_{T: \mathcal{F} \subseteq T} w_T &= \sum_{T: \mathcal{F} \subseteq T} \prod_{i \in [2^u]} w_i^{n_i} \prod_{\{a,b\} \in \mathcal{F}} \mathsf{WMC}(\varphi(a,b) \wedge R(a,b), w, \bar{w}) \prod_{a<b, \{a,b\} \in T \setminus \mathcal{F}} \frac{\mathsf{WMC}(\varphi(a,b) \wedge R(a,b), w, \bar{w})}{\mathsf{WMC}(\varphi(a,b) \wedge \neg R(a,b), w, \bar{w})} \cdot \\ &\quad \prod_{a<b, \{a,b\} \notin \mathcal{F}} \mathsf{WMC}(\varphi(a,b) \wedge \neg R(a,b), w, \bar{w}) \\ &= \prod_{i \in [2^u]} w_i^{n_i} \prod_{a<b, \{a,b\} \notin \mathcal{F}} \mathsf{WMC}(\varphi(a,b) \wedge \neg R(a,b), w, \bar{w}) \sum_{T: \mathcal{F} \subseteq T} \prod_{\{a,b\} \in \mathcal{F}} \mathsf{WMC}(\varphi(a,b) \wedge R(a,b), w, \bar{w}) \cdot \\ &\quad \prod_{a<b, \{a,b\} \in T \setminus \mathcal{F}} \frac{\mathsf{WMC}(\varphi(a,b) \wedge R(a,b), w, \bar{w})}{\mathsf{WMC}(\varphi(a,b) \wedge \neg R(a,b), w, \bar{w})} \\ &= \mathsf{TreeSum}(\mathbb{W}, \mathcal{F}) \prod_{i \in [2^u]} w_i^{n_i} \prod_{a<b, \{a,b\} \notin \mathcal{F}} \mathsf{WMC}(\varphi(a,b) \wedge \neg R(a,b), w, \bar{w}) \end{aligned}$$

where $\mathbb{W}$ is the graph with weighted adjacency matrix:

$$\mathbb{W}_{(a,b)} = \begin{cases} \mathsf{WMC}(\varphi(a,b) \wedge R(a,b), w, \bar{w}) & \text{if } \{a,b\} \in \mathcal{F} \\ \frac{\mathsf{WMC}(\varphi(a,b) \wedge R(a,b), w, \bar{w})}{\mathsf{WMC}(\varphi(a,b) \wedge \neg R(a,b), w, \bar{w})} & \text{otherwise} \end{cases}$$

Recall that we saw how to compute $\mathsf{TreeSum}(\mathbb{W}, \mathcal{F})$ in time polynomial in the size of $\mathbb{W}$ earlier in the paper. Denote $Z(n_1, \ldots, n_{2^u}) = \sum_{T: \mathcal{F} \subseteq T} w_T$. That is, $Z(n_1, \ldots, n_{2^u})$ is the weighted sum of the models satisfying the tree axiom that correspond to one assignment with the cell cardinality configuration $(n_1, \ldots, n_{2^u})$; all such assignments are isomorphic and have the same weight. Clearly, each $Z(n_1, \ldots, n_{2^u})$ can be computed in time polynomial in the domain size n using the expression derived above. Summing across all possible cell configurations we get:

$$\mathsf{WFOMC}(\xi, n, w, \bar{w}) = \sum_{n_1 + \cdots + n_{2^u} = n} \binom{n}{n_1, \ldots, n_{2^u}} Z(n_1, \ldots, n_{2^u})$$

which again can be evaluated in time polynomial in the domain size.

Finally, note that this result also holds in the presence of cardinality constraints on ξ. Recall that Theorem 2 asserts that if we can efficiently compute the WFOMC of a sentence, then we can also efficiently compute the WFOMC of that same sentence extended with some cardinality constraints, where "efficiently" means in time polynomial in the size of the domain. We can also apply this theorem to the sentence ξ containing a tree axiom. □

Next, we will use Lemma 1 to show that we can add the limited tree axiom to a two-variable sentence with counting quantifiers.

Lemma 2. *Let η denote some sentence in $\mathbf{C}^2$, and let $\xi = \eta \wedge \mathsf{Tree}(R)$. Then ξ is domain-liftable.*

Proof. From Lemma 1, we know that we can compute the WFOMC of any two-variable sentence with a single tree axiom and cardinality constraints in time polynomial in the size of the domain. In addition, recall that from Theorem 3, we can take an arbitrary $\mathbf{C}^2$ sentence η and produce an $\mathbf{FO}^2$ sentence η' along with cardinality constraints C and weights $(w', \bar{w}')$ which have the property that $\mathsf{WFOMC}(\eta, n, w, \bar{w}) = K_{\eta,n} \cdot \mathsf{WFOMC}(\eta', n, w', \bar{w}', C)$. Importantly, the reduction also gives us the constant $K_{\eta,n}$ so that we can compute the WFOMC of η efficiently.

Now, one problem when applying this reduction in our case could be that $\mathsf{Tree}(R)$ is not expressible in $\mathbf{C}^2$. That is true in general, but for a fixed domain we can express it in $\mathbf{C}^2$ as a ground formula (as described in Remark 1). Obviously, the size of such an encoding of the tree axiom would be very large (exponential in the domain size). Fortunately, we do not need to actually construct it. By inspecting the reduction in [3], one can verify that the reduction is modular (Property 1) and leaves the encoding of the tree axiom untouched, so we could replace it in the end by $\mathsf{Tree}(R)$, which also means that we did not have to replace $\mathsf{Tree}(R)$ by its ground encoding in the first place. Hence we see that the transformation from [3] works also in the presence of tree axioms. □

Finally, putting Lemma 1 and 2 together gives us our final result for the full axiom $\mathsf{Tree}(R, \mathit{Leaf})$.

Theorem 5. *Let $\eta \in \mathbf{C}^2$ denote some sentence in $\mathbf{C}^2$, and let $\xi = \eta \wedge \mathsf{Tree}(R, \mathit{Leaf})$. Then ξ is domain-liftable.*

Proof. Let

$$\gamma = \forall x : \mathit{Leaf}(x) \leftrightarrow (\exists^{=1} y : R(x, y))$$

Clearly, $\mathit{Leaf}(x)$ defines precisely the leaves in the tree defined by R, and thus:

$$\mathsf{WFOMC}(\xi, n, w, \bar{w}) = \mathsf{WFOMC}(\eta \wedge \gamma \wedge \mathsf{Tree}(R), n, w, \bar{w})$$

Since the right-hand of this expression is computable in time polynomial in the domain size n by Lemma 2, we have that ξ is domain-liftable. □

4.2. Directed Rooted Trees

Of course, in practice one often wishes to model something more interesting than just the plain Tree axiom like we saw in Example 1. We will now show how to model *directed rooted trees* as described at the beginning of the section. Recall that this axiom expresses that the binary relation E represents directed edges of a tree, with root node given by the unary relation *Root*, and leaves given by the unary relation *Leaf*. Note here that the root node is never considered a leaf, even if it has degree 1.

Proposition 1. *Let $\phi \in \mathbf{C}^2$ be an arbitrary sentence, and let ψ denote the sentence below:*

$$\begin{aligned}
\psi = &\exists^{=1} x : \mathit{Root}(x) \wedge \\
&\forall x : \mathit{Root}(x) \rightarrow D(x, x) \wedge \\
&\forall x \exists^{=1} y : D(x, y) \wedge \\
&\forall x \forall y : R(x, y) \rightarrow (D(x, y) \vee D(y, x)) \wedge \\
&\forall x \forall y : E(x, y) \leftrightarrow (D(x, y) \wedge (\neg \mathit{Root}(x) \vee \neg \mathit{Root}(y)) \wedge \\
&\mathsf{Tree}(R) \ \wedge \forall x : \mathit{Leaf}(x) \leftrightarrow (\forall y : \neg D(y, x))
\end{aligned}$$

where R and D are fresh predicate symbols not occurring in ϕ. Then:

$$\mathsf{WFOMC}(\phi \wedge \mathsf{DirectedRootedTree}(\mathit{Root}, E, \mathit{Leaf}), w, \bar{w}, n, C) = \mathsf{WFOMC}(\phi \wedge \psi, w', \bar{w}', n, C)$$

where $(w', \bar{w}')$ is obtained by extending $(w, \bar{w})$ with neutral weights on all new predicates in ψ.

Proof. We can motivate this choice of ψ as follows. We pick a single vertex as a root, which we enforce[3] with $\exists^{=1} x : Root(x)$. We then define a new relation $D(x, y)$ that will represent auxiliary directed edges (the directed edges of the tree will be represented by another relation E). The edges are oriented in such a way that the edges go from vertices further from the root, to vertices closer to it. First, we add a self-loop for the root node $\forall x : Root(x) \rightarrow D(x, x)$. We also require that D satisfies the functionality constraint and that whenever there is an original undirected edge $R(x, y)$, then there must be either a directed edge $D(x, y)$ or $D(y, x)$. Since D must satisfy the functionality constraint and the root must have a self loop, any edge that connects the root with another vertex, must be directed towards the root (if any of these edges were directed away from the root then D would not satisfy the functionality constraint). We can use induction on the depth of the tree using essentially the same argument (the fact that any vertex must have out-degree 1 because of the functionality constraint) to show that D must be oriented in such a way that there is a connected path from any leaf to the root. Note that we cannot immediately use D as our directed edge relation, since D has a self-loop at the root node, which should not be present in a directed rooted tree. As a solution, we add a "wrapper" predicate E that has the same edges as D except for the self-loop at the root node. Last, we enforce with the axiom $\mathsf{Tree}(R)$ that the relation R forms an undirected tree. To encode the *Leaf* predicate, we can notice that leaves are precisely those vertices that have zero in-degree w.r.t. D (in particular, the root is never considered a leaf even if it has degree 1) and therefore we can encode leaves using $\forall x : Leaf(x) \leftrightarrow (\forall y : \neg D(y, x))$.

It is clear that the models of the aforementioned rules are precisely those satisfying the axiom $\mathsf{DirectedRootedTree}(Root, E, Leaf)$. It is also not difficult to show that, under the assumption that D is not used anywhere else (which we can assume w.l.o.g.), the reduction is also modular. □

Theorem 6. *Let:*

$$\xi = \phi \wedge \mathsf{DirectedRootedTree}(Root, E, Leaf)$$

for some $\phi \in \mathbf{C}^2$. Then ξ is domain-liftable.

Proof. Follows immediately from Proposition 1 and Theorem 5. □

4.3. Binary Rooted Ordered Trees

We now consider the axiom $\mathsf{BinaryOrderedTree}(Root, Left, Right, Leaf)$, expressing that the binary relations $Left$ and $Right$ represent directed edges of a full binary rooted ordered tree with root node given by the unary relation $Root$ and leaves given by the unary relation $Leaf$.

Proposition 2. *Let $\phi \in \mathbf{C}^2$ be an arbitrary sentence, and let ψ denote the sentence below:*

$$\begin{aligned}
\psi = &\forall x \exists^{=1} y : L(x, y) \wedge \\
&\forall x \exists^{=1} y : R(x, y) \wedge \\
&\forall x : Leaf(x) \leftrightarrow L(x, x) \wedge \\
&\forall x : Leaf(x) \leftrightarrow R(x, x) \wedge \\
&\forall x \forall y : E(x, y) \rightarrow (L(y, x) \vee R(y, x)) \wedge \\
&\forall x \forall y : L(x, y) \rightarrow (E(y, x) \vee Leaf(x)) \wedge \\
&\forall x \forall y : R(x, y) \rightarrow (E(y, x) \vee Leaf(x)) \wedge \\
&\forall x \forall y : \neg L(x, y) \vee \neg R(x, y) \vee Leaf(x) \wedge \\
&\forall x \forall y : Left(x, y) \leftrightarrow (L(x, y) \wedge \neg Leaf(x)) \wedge \\
&\forall x \forall y : Right(x, y) \leftrightarrow (R(x, y) \wedge \neg Leaf(x)) \wedge \\
&\mathsf{DirectedRootedTree}(Root, E, Leaf)
\end{aligned}$$

where $Left$ and $Right$ are fresh predicate symbols not occurring in ϕ. Then:

$$\mathsf{WFOMC}(\phi \wedge \mathsf{BinaryOrderedTree}(Root, Left, Right, Leaf), w, \bar{w}, n, C) = \mathsf{WFOMC}(\phi \wedge \psi, w', \bar{w}', n, C)$$

where $(w', \bar{w}')$ is obtained by extending $(w, \bar{w})$ with neutral weights on all new predicates in ψ.

[3] Equivalently, we may simply add the cardinality constraint $|Root| = 1$ instead.

Proof. Again like in Proposition 1, we motivate this choice of ψ as follows. We define relations $Left(x, y)$ and $Right(x, y)$, expressing that x is the left (resp. right) child node of y. Since the binary tree is full, every node that is not a leaf must have precisely one left and one right child. In order to encode the relations $Left(x, y)$ and $Right(x, y)$, we first define two auxiliary relations $L(x, y)$ and $R(x, y)$, which we call left and right pseudo-children, respectively, and which agree with $Left(x, y)$ and $R(x, y)$ when x is not a leaf. We require leaves to be their own pseudo-children, i.e., for a leaf x, we require $L(x, x)$ and $R(x, x)$ to hold. Any node must have both a left and a right pseudo-child, and if y is a left (resp. right) pseudo-child node of x we must either have that x is the parent of y, or they are the same node and hence x is a leaf. We also impose the constraint that if y is both a left and right pseudo-child of x, then x and y are the same node and hence a leaf. Then we define left and right children $Left(x, y)$ and $R(x, y)$ using the pseudo-children relations $L(x, y)$ and $R(x, y)$. Finally, in the last conjunct we impose the requisite DirectedRootedTree axiom to set the relations $Root$, E, and $Leaf$ in the appropriate way. □

Theorem 7. *Let:*

$$\xi = \phi \wedge \mathsf{BinaryOrderedTree}(Root, Left, Right, Leaf)$$

for some $\phi \in \mathbf{C}^2$. *Then* ξ *is domain-liftable.*

Proof. Follows immediately from Proposition 2 and Theorem 6. □

It would be straightforward to extend the approach used in this section to show that the above theorem holds also when we replace binary ordered trees by ordered k-ary trees.

5. Applications and Experiments

In this section, we will describe some applications of the tree axiom and examine the scalability of our approach in practice. To test our results experimentally, we implemented a weighted first-order model counter in Python that follows the algorithmic approach presented here with support for tree axioms. All experiments were performed on a computer with a six-core Intel i7 2.2GHz processor and 16 GB of RAM.

5.1. Combinatorics on Trees

We start by modelling some problems from enumerative combinatorics on trees.

5.1.1. k-coloured trees

We first consider the problem of counting k-coloured trees. Recall that a graph is said to be k-colourable if every vertex can be coloured with one of k colours such that no two vertices of the same colour are adjacent to one another. A k-coloured graph is a k-colourable graph together with a valid colour assignment. We first verified that our approach gave the correct expected sequence for 2-coloured trees: namely, we should have $\mathsf{FOMC}(\xi_2, n) = 2n^{n-2}$, because every one of the n^{n-2} trees on n nodes is 2-colourable, and any such tree has precisely two colourings (since colouring any one node uniquely determines the colour of the remainder). We then tested it on larger values of k. An indication of the runtime of our approach (with a 30 second timeout) on 2-, 3-, and 4-coloured trees is given in Figure 1.

5.1.2. Rooted trees with k leaves

We next consider the problem posed in the introduction of the paper: how many labelled rooted trees on n nodes have exactly k leaves? This can be modelled with the following sentence:

$$\xi = \mathsf{DirectedRootedTree}(Root, E, Leaf)$$

subject to the cardinality constraint $|Leaf| = k$. Note that, although this sentence appears relatively simple at first glance, computing its FOMC takes some work. First, the axiom is expanded into a $\mathbf{C}^2$ sentence as shown in Proposition 1. Next, the counting quantifiers introduced by the DirectedRootedTree axiom are reduced to cardinality constraints (Theorem 3), for which computing the FOMC is split across several different $\mathbf{FO}^2$ WFOMC oracle calls with different weights (Theorem 2). Finally, existential quantifiers in each of these calls must be eliminated as described in the proof of Theorem 1, which is a process that itself introduces new auxiliary Skolemization predicates (see [2] for the full details).

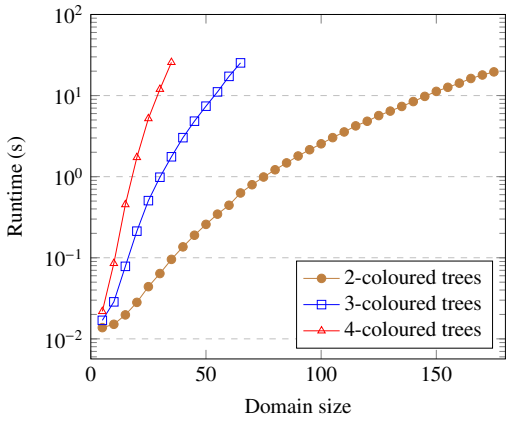

Figure 1. Runtime for counting 2-, 3-, and 4-coloured trees

	Number of leaves k		
Number of nodes n	1	2	3
2	2	–	–
3	6	3	–
4	24	36	4
5	120	360	140
6	720	3600	3000
7	5040	37800	54600
8	40320	423360	940800
9	362880	5080320	16087680

Table 1. Counting labelled rooted trees on n nodes with exactly k leaves

As a result of the steps above necessary to compute the FOMC, despite the fact that the runtime is still guaranteed to be polynomial in the domain size, our implementation struggled to scale to large values of n because the degree of the polynomial bounding the runtime was simply too high. We were able to scale to a domain size of $n = 9$ within a 10 minute timeout. However, we were able to check these results against *The On-Line Encyclopedia of Integer Sequences* (OEIS)[4] for values $k = 1$, 2 and 3 with sequences A000142, A055303, and A055304 respectively. In particular, note that the case $k = 1$ corresponds to counting paths of length n. The values we computed are given in Table 1.

5.2. Markov Logic Networks over Trees

We now turn to applications of our results to probabilistic inference in graphical models. Specifically, in this section we study the imposition of tree axioms on *Markov logic networks* (MLNs) [4], a popular statistical-relational formalism for which inference and learning is reducible to WFOMC.

5.2.1. Preliminaries

Formally, an MLN is a finite set of weighted first-order logic formulas $\{(w_1, \phi_1), \ldots, (w_n, \phi_n)\}$, where each w_i is either a real-valued weight or ∞, and ϕ_i is a quantifier-free first-order formula. MLNs can be viewed as a template for constructing Markov random fields. An MLN Φ paired with a domain Δ induces a probability distribution on possible worlds:

$$\Pr(\omega; \Phi, \Delta) = \begin{cases} \frac{1}{Z_{\Phi,\Delta}} \exp\left(\sum_{(w,\phi)\in\Phi_{\mathbb{R}}} w \cdot N(\phi, \omega)\right) & \text{if } \omega \models \Phi_\infty \\ 0 & \text{otherwise} \end{cases}$$

[4] https://oeis.org

where $\Phi_{\mathbb{R}}$ and Φ_{∞} denote the real-valued and ∞-valued formulas in Φ respectively, $N(\phi, \omega)$ denotes the number of groundings of ϕ satisfied in the world ω, and $Z_{\Phi,\Delta}$ is a normalization constant called the *partition function* of the MLN.

5.2.2. Reduction to WFOMC

In general, computing the partition function (and by extension, marginal inference) in a given MLN is #P-hard in the domain size. However, the aforementioned reduction to WFOMC has the helpful property that the number of variables in the sentence produced by the reduction is the same as the maximum number of variables appearing in any formula of the original MLN. In particular, this means that if the number of variables used in each of the formulas in the MLN is limited to two, the inference problem for such MLNs lies in FP. We present the reduction in question below.

Definition 9 ([1])**.** *Let $\Phi = \{(w_1, \phi_1), \ldots, (w_k, \phi_k), (\infty, \phi_{k+1}), \ldots, (\infty, \phi_n)\}$ be an MLN. Then the reduction of Φ to a weighted sentence is constructed as:*

$$\bigwedge_{i=1}^{k} \forall \mathbf{x} : A_i(\mathbf{x}) \leftrightarrow \phi_i(\mathbf{x}) \wedge \bigwedge_{i=k+1}^{n} \forall \mathbf{x} : \phi_i(\mathbf{x})$$

where each A_i is a fresh predicate not occurring in any of the ϕ_i, $\mathbf{x}$ denotes the collection of free variables occurring in each ϕ_i, and weights are set as follows: $w(A_i) = e^{w_i}$, $\bar{w}(A_i) = 1$, and $w(R_i) = \bar{w}(R_i) = 1$ for any other predicate R_i.

5.2.3. Experiments

Consider the classic "friends-and-smokers" MLN described below:

$$\begin{aligned} w_1 \quad & Smokes(x) \\ w_2 \quad & Friends(x, y) \wedge Smokes(x) \rightarrow Smokes(y) \\ \infty \quad & \neg Friends(x, x) \end{aligned}$$

Intuitively, this models that people who are friends with smokers are likely to smoke themselves. We consider the problem of computing the partition function of this MLN over different domain sizes, subject to either one of two constraints:

1. the hard constraint that the $Friends$ relation must form a tree, or
2. the soft constraint that models in which the $Friends$ relation forms a tree are to be preferred.

The former constraint can be dealt with by simply conjoining the axiom $\mathsf{Tree}(Friends)$ to the sentence obtained from applying the reduction in Definition 9. On the other hand, the latter constraint requires adding a new formula $(w_3, R(x, y) \leftrightarrow Friends(x, y))$ to the MLN for an appropriate weight value w_3, and conjoining the axiom $\mathsf{Tree}(R)$ to the reduced sentence. Experimental results for both the hard and soft constraints with fixed weight values are shown in Figure 2; the computation time of the partition function is virtually identical in both cases.

6. Discussion

An interesting question is what other axioms can be added to domain-liftable fragments using the techniques that we exploited in this paper, while still guaranteeing domain-liftability. For instance, can we add a *forest axiom*, stating that a certain relation corresponds to a forest, i.e. an undirected graph with no cycles (but not necessarily connected)?

To understand the limits of our techniques, it is useful to look again at how our method works. What we do in this paper can roughly be understood as follows: we first iterate over configurations of cells of the sentence, and for each such configuration we construct a weighted adjacency matrix (whose weights are calculated from the given sentence and the respective weighting functions). We then compute the weighted sum of the spanning trees of the weighted graph given by this adjacency matrix, using Kirchoff's matrix-tree theorem. Here, we ignore some details, but this picture should be enough for understanding the obstacles when trying to generalize this approach.

Computing the weighted sum of spanning trees of a graph can also be seen as evaluation of the Tutte polynomial [25] at the point $(1, 1)$. The Tutte polynomial is a graph polynomial of two variables—every graph G has a Tutte

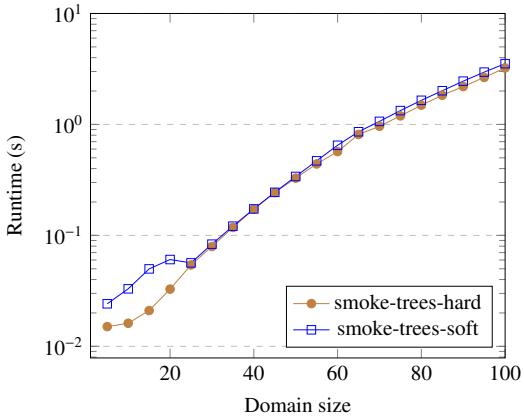

Figure 2. Runtime of our tool for computing the partition function of the "friends-and-smokers" MLN for various domain sizes, subject to soft and hard tree constraints

polynomial $T_G(x, y)$. Different points (x, y) correspond to different combinatorial properties of the graph. For instance, $T_G(1, 1)$ is the number of spanning trees of a connected graph G, and, when G is disconnected, the number of spanning forests with the same number of components as G. Similarly, the value $T_G(2, 1)$ is the number of subgraphs of G that are forests, the value $T_G(1, 2)$ counts the number of spanning subgraphs, etc.

If we wanted to generalize the techniques in this paper, say, to the forest axiom, we would need to be able to evaluate the Tutte polynomials, for graphs corresponding to adjacency matrices constructed for the different cell configurations in our algorithm, at the point $(2, 1)$. Unfortunately, evaluating the Tutte polynomial $T_G(2, 1)$ for an arbitrary graph G is #P-hard. In fact, evaluating Tutte polynomials is in polynomial time only for the points $(1, 1)$, $(-1, -1)$, $(0, -1)$, $(-1, 0)$, $(i, -i)$, $(-i, i)$, $(\exp(2\pi i/3), (\exp(4\pi i/3))$, $((\exp(4\pi i/3), \exp(2\pi i/3))$ and for the points satisfying $(x - 1)(y - 1) = 2$. For all other points (x, y), evaluating Tutte polynomials is #P-hard [26]. It is therefore unlikely that the approach that we took in this paper can be generalized easily to other types of axioms. On the other hand, it is not entirely impossible. The adjacency matrices that occur in our algorithm are not arbitrary—they have a certain block structure. It may be that evaluating Tutte polynomials for adjacency matrices with block structure is possible in polynomial time, but no such result is currently known, and it does not seem easy to obtain. In fact, such a result would likely be of sufficient interest on its own in combinatorics, as even simpler results, e.g. computing Tutte polynomials of complete graphs, are non-trivial [27].

Another direction for future work is improving the practical efficiency of the approach described here. Generally, the runtime of our approach is heavily dependent on the number of cell configurations we must consider in Equation 1. Since our approach for representing rooted trees, as well as the existing approach for modelling existential quantifiers, both rely on encodings into larger sentences, we often observe an impractically large impact on runtime. We see two possible approaches to tackle this issue: first, one can try to find more compact encodings (one such example is given in the appendix to this paper), or approaches for modelling these constructs without requiring encodings altogether. Alternatively, one can investigate approaches for more efficiently computing the WFOMC even when the number of cell configurations is high. The work described in [28] may provide some inspiration in this respect.

We conclude by remarking that approaches in other areas of logic that initially proved a theoretical—but intractable in practice—upper bound have seen later some success in adapting them for practical purposes: see, for example, such an approach for Courcelle's theorem [29].

7. Conclusion

We showed how to extend existing domain liftability results on $\mathbf{FO}^2$ and $\mathbf{C}^2$ to allow for the addition of tree axioms, and showed how to extend this core notion to model directed rooted trees and binary rooted ordered trees. We also presented some preliminary experimental results on our approach.

Acknowledgements

TvB was supported by the Research Foundation – Flanders (G095917N). OK 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".

Appendix A. Directed Trees: An Alternative Encoding

In this section we record a practically more efficient but less straightforward encoding for directed rooted trees:

$$\begin{aligned}\psi = &\mathsf{Tree}(R) \wedge \forall x : Root(x) \leftrightarrow D(x,x) \wedge \\ &\forall x \exists^{=1} y : D(x,y) \wedge \\ &\forall x \forall y : D(x,y) \rightarrow (R(x,y) \vee D(x,x)) \wedge \\ &\forall x \forall y : R(x,y) \rightarrow (D(x,y) \vee D(y,x)) \wedge \\ &\forall x \forall y : E(x,y) \leftrightarrow (D(x,y) \wedge (\neg Root(x) \vee \neg Root(y)) \wedge \\ &\forall x : Leaf(x) \leftrightarrow (\forall y : \neg D(y,x))\end{aligned}$$

This transformation has one less conjunct with a counting quantifier than the transformation described in the main text ($\exists^{=1} x : Root(x)$) and, instead of it, it contains the formula $\forall x \forall y : D(x,y) \rightarrow (R(x,y) \vee D(x,x))$ which together with the functionality constraint on D and the fact that the cardinality of R must be $n-1$ (and together with the rest of the formula) implies that there must be exactly one x such that $D(x,x)$, which must be the root (by $\forall x : Root(x) \leftrightarrow D(x,x)$).

References

[1] G. Van den Broeck, On the completeness of first-order knowledge compilation for lifted probabilistic inference, in: NIPS, 2011, pp. 1386–1394.
[2] G. Van den Broeck, W. Meert, A. Darwiche, Skolemization for weighted first-order model counting, in: KR, AAAI Press, 2014.
[3] O. Kuzelka, Weighted first-order model counting in the two-variable fragment with counting quantifiers, J. Artif. Intell. Res. 70 (2021) 1281–1307.
[4] M. Richardson, P. M. Domingos, Markov logic networks, Machine Learning 62 (1-2) (2006) 107–136.
[5] D. Fierens, G. Van den Broeck, J. Renkens, D. Shterionov, B. Gutmann, I. Thon, G. Janssens, L. De Raedt, Inference and learning in probabilistic logic programs using weighted boolean formulas, TPLP 15 (3) (2015) 358–401.
[6] M. Y. Vardi, The complexity of relational query languages (extended abstract), in: STOC, ACM, 1982, pp. 137–146.
[7] P. Beame, G. Van den Broeck, E. Gribkoff, D. Suciu, Symmetric weighted first-order model counting, in: PODS, ACM, 2015, pp. 313–328.
[8] L. G. Valiant, The complexity of enumeration and reliability problems, SIAM J. Comput. 8 (3) (1979) 410–421.
[9] L. Libkin, Elements of Finite Model Theory, Texts in Theoretical Computer Science. An EATCS Series, Springer, 2004.
[10] S. Chaiken, D. J. Kleitman, Matrix tree theorems, J. Comb. Theory Ser. A. 24 (3) (1978) 377–381.
[11] D. Poole, First-order probabilistic inference, in: IJCAI, Morgan Kaufmann, 2003, pp. 985–991.
[12] B. Milch, L. S. Zettlemoyer, K. Kersting, M. Haimes, L. P. Kaelbling, Lifted probabilistic inference with counting formulas, in: AAAI, AAAI Press, 2008, pp. 1062–1068.
[13] N. Taghipour, D. Fierens, J. Davis, H. Blockeel, Lifted variable elimination: Decoupling the operators from the constraint language, J. Artif. Intell. Res. 47 (2013) 393–439.
[14] T. Braun, R. Möller, Lifted junction tree algorithm, in: KI, Vol. 9904 of Lecture Notes in Computer Science, Springer, 2016, pp. 30–42.
[15] G. Van den Broeck, N. Taghipour, W. Meert, J. Davis, L. De Raedt, Lifted probabilistic inference by first-order knowledge compilation, in: IJCAI, IJCAI/AAAI, 2011, pp. 2178–2185.
[16] A. Kuusisto, C. Lutz, Weighted model counting beyond two-variable logic, in: LICS, ACM, 2018, pp. 619–628.
[17] S. M. Kazemi, A. Kimmig, G. Van den Broeck, D. Poole, New liftable classes for first-order probabilistic inference, in: NIPS, 2016, pp. 3117–3125.
[18] S. Benaim, M. Benedikt, W. Charatonik, E. Kieronski, R. Lenhardt, F. Mazowiecki, J. Worrell, Complexity of two-variable logic on finite trees, ACM Trans. Comput. Log. 17 (4) (2016) 32:1–32:38.
[19] W. Charatonik, P. Witkowski, Two-variable logic with counting and trees, in: LICS, IEEE Computer Society, 2013, pp. 73–82.
[20] A. Amarilli, İ. İ. Ceylan, The dichotomy of evaluating homomorphism-closed queries on probabilistic graphs, Log. Methods Comput. Sci. 18 (1) (2022).
[21] N. Dalvi, D. Suciu, The dichotomy of probabilistic inference for unions of conjunctive queries, J. ACM 59 (6) (2012) 30:1–30:87.
[22] E. Gribkoff, G. Van den Broeck, D. Suciu, Understanding the complexity of lifted inference and asymmetric weighted model counting, in: UAI, AUAI Press, 2014, pp. 280–289.

[23] E. Grädel, P. G. Kolaitis, M. Y. Vardi, On the decision problem for two-variable first-order logic, Bull. Symb. Log. 3 (1) (1997) 53–69.
[24] A. Cayley, A theorem on trees, Quart. J. Pure Appl. Math. 23 (1889) 376–378.
[25] E. W. Weisstein, Tutte polynomial. From MathWorld—A Wolfram Web Resource, last visited on 26/10/2022.
URL https://mathworld.wolfram.com/TuttePolynomial.html
[26] F. Jaeger, D. L. Vertigan, D. J. Welsh, On the computational complexity of the jones and tutte polynomials, in: Mathematical Proceedings of the Cambridge Philosophical Society, Vol. 108, Cambridge University Press, 1990, pp. 35–53.
[27] I. Pak, Computation of tutte polynomials of complete graphs (1993).
[28] T. van Bremen, O. Kuzelka, Faster lifting for two-variable logic using cell graphs, in: UAI, Vol. 161 of Proceedings of Machine Learning Research, AUAI Press, 2021, pp. 1393–1402.
[29] J. Kneis, A. Langer, A practical approach to courcelle's theorem, Electron. Notes Theor. Comput. Sci. 251 (2009) 65–81.