Skolemization for Weighted First-Order Model Counting

Guy Van den Broeck
Computer Science Department
University of California, Los Angeles
guyvdb@cs.ucla.edu

Wannes Meert
Computer Science Department
KU Leuven
wannes.meert@cs.kuleuven.be

Adnan Darwiche
Computer Science Department
University of California, Los Angeles
darwiche@cs.ucla.edu

Abstract

First-order model counting emerged recently as a novel reasoning task, at the core of efficient algorithms for probabilistic logics. We present a Skolemization algorithm for model counting problems that eliminates existential quantifiers from a first-order logic theory without changing its weighted model count. For certain subsets of first-order logic, lifted model counters were shown to run in time polynomial in the number of objects in the domain of discourse, where propositional model counters require exponential time. However, these guarantees apply only to Skolem normal form theories (i.e., no existential quantifiers) as the presence of existential quantifiers reduces lifted model counters to propositional ones. Since textbook Skolemization is not sound for model counting, these restrictions precluded efficient model counting for directed models, such as probabilistic logic programs, which rely on existential quantification. Our Skolemization procedure extends the applicability of first-order model counters to these representations. Moreover, it simplifies the design of lifted model counting algorithms.

1 Introduction

Weighted model counting (WMC) is a generalization of model counting (Gomes, Sabharwal, and Selman 2009). In model counting, also known as #SAT, one counts the number of satisfying assignments of a propositional sentence. In WMC, each assignment has an associated weight and the task is to compute the *sum of the weights* of all satisfying assignments. One application of WMC is to probabilistic graphical models. For example, exact inference algorithms for Bayesian networks encode probabilistic inference as a WMC task, which can then be solved by knowledge compilation (Darwiche 2002) or exhaustive DPLL search (Sang, Beame, and Kautz 2005).

WMC also plays an important role in *first-order probabilistic* representations. These became popular in recent years, in statistical relational learning (Getoor and Taskar 2007) and probabilistic logic learning (De Raedt et al. 2008), which are concerned with modeling and learning complex logical and probabilistic interactions between large numbers of objects. Efficient algorithms again reduce exact probabilistic inference to a WMC problem on a propositional knowledge base (Chavira, Darwiche, and Jaeger 2006;

Fierens et al. 2011; 2013). Encoding first-order probabilistic models into propositional logic retains a key advantage of the Bayesian network algorithms: WMC naturally exploits determinism and local structure in the probabilistic model (Boutilier et al. 1996; Chavira and Darwiche 2005). A disadvantage is that the high-level first-order structure is lost. Poole (2003) observed that knowing the *symmetries* that are abundant in first-order structure can speed up probabilistic inference. Lifted inference algorithms reason about groups of objects as a whole, similar to the high-level reasoning of first-order resolution. This has lead Van den Broeck et al. (2011) and Gogate and Domingos (2011) to propose *weighted first-order model counting* (WFOMC) as the core reasoning task underlying lifted inference algorithms. WFOMC assigns a weight to interpretations in finite-domain, function-free first-order logic, and computes the sum of the weights of all models.

Counting models at the first-order level has computational advantages. For certain classes of theories, knowing the first-order structure gives exponential speedups (Van den Broeck 2011). For example, counting the models of a first-order universally quantified CNF with up to two logical variables per clause can always be done in time polynomial in the size of the domain of discourse. In contrast, a propositionalization of these CNFs will often have a treewidth polynomial in the domains size, and propositional model counting runs in exponential time.

One major limitation of first-order model counters, however, is that they require input in *Skolem normal form* (i.e., without existential quantifiers). This is a common requirement for first-order automated reasoning algorithms, such as theorem provers. It is usually dealt with by Skolemization, which introduces Skolem constants and functions. However, the introduction of functions is problematic for first-order model counters as they expect a function-free input.

The main contribution of this paper is a *Skolemization procedure* that is specific for weighted first-order model counting. The procedure maps a logical input theory to an output theory that is devoid of existential quantifiers and functions, yet has an identical weighted first-order model count. The procedure is modular, in that it remains sound when extending the input and output theories with a new sentence. Furthermore, it is purely first-order as it is independent of the domain of discourse.

The proposed Skolemization algorithm has a range of implications. First, it opens up new possibilities for lifted inference algorithms. For example, on Markov Logic Networks with quantifiers (Richardson and Domingos 2006), and various forms of *Probabilistic Logic Programs* (e.g., De Raedt, Kimmig, and Toivonen (2007)), lifted algorithms currently provide little or no benefit over propositional ones. The main reason is that the WFOMC form of these representations generally contain existential quantifiers. The proposed Skolemization algorithm allows us, for the first time, to perform lifted inference on these representations. Second, there are *liftability theorems* that define classes of theories for which WFOMC is domain-lifted, meaning that it runs in time polynomial in the domain size (Jaeger and Van den Broeck 2012). These theorems had to assume Skolem normal form for the mentioned reason, but now apply more generally. Finally, the Skolemization algorithm averts the need for special inference rules that deal with existential quantifiers, *simplifying the design* of future WFOMC algorithms.

2 Weighted First-Order Model Counting

We start by formally defining the weighted first-order model counting task. We also compare it to propositional weighted model counting and discuss existing algorithms.

Background

Throughout this paper, we will work with the *function-free finite-domain* fragment of first-order logic (FOL), which we now briefly review. An *atom* $P(t_1, \ldots, t_n)$ consists of predicate P/n of arity n followed by n arguments, which are either *constants* from a finite domain $\mathbf{D} = \{A, B, \ldots\}$ or *logical variables* $\{x, y, \ldots\}$. We use $\mathbf{y}$ to denote a sequence of logical variables. A *literal* is an atom or its negation. A *formula* combines atoms with logical connectives and quantifiers $\exists$ and $\forall$. A logical variable x is *quantified* if it is enclosed by a $\forall x$ or $\exists x$. A *free variable* is one that is not quantified. A *sentence* is a formula without free variables. A formula is *ground* if it contains no logical variables. A *clause* is a disjunction of literals and a *CNF* is a conjunction of clauses. The *groundings* of a quantifier-free formula is the set of formulas obtained by instantiating the free variables with any possible combination of constants from $\mathbf{D}$. The grounding of $\forall x, \phi$ and $\exists x, \phi$ is the conjunction resp. disjunction of all groundings of ϕ.

We will make use of *Herbrand semantics* (Hinrichs and Genesereth 2006), as is customary in statistical relational learning and probabilistic logic learning. The *Herbrand base* of sentence Δ for domain $\mathbf{D}$ is the set of all ground atoms that can be constructed from predicates and constants in $\mathbf{D}$. A *Herbrand interpretation* is a truth-value assignment to all atoms in the Herbrand base. We will find it convenient to represent interpretations as sets of literals. A Herbrand model of Δ is a Herbrand interpretation ω that satisfies Δ, denoted by $\omega \models_{\mathbf{D}} \Delta$.

Definitions

We first review propositional weighted model counting.

Definition 1 (WMC)**.** Given
– a *sentence* Δ in propositional logic over literals $\mathcal{L}$, and
– a *weight function* $w : \mathcal{L} \to \mathbb{R}^{\geq 0}$,

the *weighted model count* (WMC) is

$$\text{WMC}(\Delta, w) = \sum_{\omega \models \Delta} \prod_{l \in \omega} w(l).$$

WFOMC lifts WMC to the first-order level as follows.

Definition 2 (WFOMC[1])**.** Given
– a *sentence* Δ in FOL containing predicates $\mathcal{P}$,
– a set of constants $\mathbf{D}$, including the constants in Δ, and
– a pair of *weight functions* $w, \bar{w} : \mathcal{P} \to \mathbb{R}$,

the *weighted first-order model count* (WFOMC) is

$$\text{WFOMC}(\Delta, \mathbf{D}, w, \bar{w})$$
$$= \sum_{\omega \models_{\mathbf{D}} \Delta} \prod_{l \in \omega_0} \bar{w}(\text{pred}(l)) \prod_{l \in \omega_1} w(\text{pred}(l)),$$

where ω_0 and ω_1 consists of the true, respectively false, literals in ω, and pred maps literals to their predicate.

The weight functions assign a weight to each predicate. The weight of a positive (negative) literal is the weight of its predicate in w ($\bar{w}$). The weight of a model is the product of its literal weights. Finally, the total count is the sum of the weights of all the Herbrand models of Δ.

Our WFOMC definition deviates from WMC in two ways. First, WMC directly assigns weights to *literals*. WFOMC instead assigns weights to *predicates*, and defines literal weights in terms of predicate weights. This distinguishes WFOMC from probabilistic databases (see Section 6). If for modeling reasons, certain literals need to be assigned unique weights, this can always be achieved by introducing new predicates.

Second, our definition permits predicate weights to be *negative* numbers. Negative weights will turn out to be crucial for our Skolemization algorithm. Historically, the WMC weight function has mostly been used to represent probabilities. This led to the (sometimes implicit) assumption that weights are between zero and one, or at least nonnegative. Nevertheless, all exact weighted model counters we are aware of can handle negative weights.[2] It appears that the positive weight assumption is more intrinsic to approximate weighted model counters (Wei and Selman 2005; Gogate and Dechter 2011). Section 6 discusses negative weights in more detail.

Motivation

A WFOMC problem can always be propositionalized into a WMC problem. We can ground Δ for $\mathbf{D}$, turn every atom

---

[1]This definition is based on Van den Broeck et al. (2011). WFOMC is called *Lifted WMC* in Gogate and Domingos (2011).

[2]In fact, the only underlying requirement of exact model counting approaches is that literal weights are elements from a *commutative semiring* (Kimmig, Van den Broeck, and De Raedt 2012).

in the Herbrand base into a propositional atom, and associate with every propositional literal the weight of its original predicate. One may wonder why we define this task at the first-order level.

Our motivation is computational. Similar to how a single step of first-order resolution can perform a large number of propositional resolution steps, a WFOMC solver can often provide exponential speedups over WMC solvers. First-order quantifiers make statements about groups of symmetric objects, which we can reason about jointly.

Without going into algorithmic details, we will now illustrate this principle on concrete examples. For the sake of simplicity, the examples are non-weighted model counting problems, corresponding to WFOMC problems where $w(P) = \bar{w}(P) = 1$ for all predicates P. Consider Δ to be

$$\texttt{Stress(A)} \Rightarrow \texttt{Smokes(A)}. \tag{1}$$

Assuming that $\mathbf{D} = \{A\}$, every interpretation of $\texttt{Stress(A)}$ and $\texttt{Smokes(A)}$ satisfies Δ, except when $\texttt{Stress(A)}$ is true and $\texttt{Smokes(A)}$ is false. Therefore, the model count is 3. Now let Δ be

$$\forall x, \texttt{Stress}(x) \Rightarrow \texttt{Smokes}(x). \tag{2}$$

Without changing $\mathbf{D}$, the model count is still 3. When we expand $\mathbf{D}$ to contain n constants, we get n independent copies of Formula 1. For each person x, atoms $\texttt{Stress}(x)$ and $\texttt{Smokes}(x)$ can jointly take 3 values, and the total model count becomes 3^n.

This example already demonstrates the benefits of first-order counting. A *propositional* model counter on the groundings of Formula 2 would detect that all n clauses are independent, recompute for every clause that it has 3 models, and multiply these counts n times. Propositional model counters have no elementary operation for exponentiation. A *first-order* model counter reads from the first-order structure that it suffices to compute the model count of a single ground clause, and then knows to exponentiate. It never actually grounds the formula, and given the size of $\mathbf{D}$, it runs in logarithmic time. This gives an exponential speedup over propositional counting, which runs in linear time.

These first-order counting techniques can interplay with propositional ones. Take for example Δ to be

$$\forall y, \texttt{ParentOf}(y) \land \texttt{Female} \Rightarrow \texttt{MotherOf}(y). \tag{3}$$

This sentence is about a specific individual who may be a female, depending on whether the proposition $\texttt{Female}$ is true. We can separately count the models in which $\texttt{Female}$ is true, and those in which it is false (i.e., a Shannon decomposition). When $\texttt{Female}$ is false, Δ is satisfied, and the $\texttt{ParentOf}$ and $\texttt{MotherOf}$ atoms can take on any value. This gives 4^n models. When $\texttt{Female}$ is true, Δ is structurally identical to Formula 2, and has 3^n models. The total model count is then $3^n + 4^n$.

These concepts can be applied recursively to count more complicated formulas. Take for example

$$\forall x, \forall y, \texttt{ParentOf}(x, y) \land \texttt{Female}(x) \Rightarrow \texttt{MotherOf}(x, y).$$

There is now a partition of the ground clauses into n independent sets of n clauses. The sets correspond to values of

x, and the individual clauses to values of y. The formula for each specific x, that is, each set of clauses, is structurally identical to Formula 3 and has count of $3^n + 4^n$. The total model count is then $(3^n + 4^n)^n$.

The most impressive improvements are attained when propositional model counters run in time *exponential* in n, yet first-order model counters run in *polynomial* time. To consider an example where this comes up, let Δ be

$$\forall x, \forall y, \texttt{Smokes}(x) \land \texttt{Friends}(x, y) \Rightarrow \texttt{Smokes}(y). \tag{4}$$

This time, the clauses in the grounding of Δ are no longer independent, and it would be wrong to simply exponentiate their counts. Let us first assume that we know a partial interpretation of the $\texttt{Smokes}$ atoms with k positive literals (i.e., k people smoke). The question is now: how many models extend this partial interpretation? Formula 4 encodes that a smoker cannot be friends with a non smoker. Hence, out of n^2 $\texttt{Friends}$ atoms, $k(n-k)$ have to be false, and the others can take either truth value. Thus, the number of models is $2^{n^2-k(n-k)}$. Second, we know that there are $\binom{n}{k}$ partial interpretations with k smokers, and k can range from 0 to n. This results in the total model count of

$$\sum_{k=0}^{n} \binom{n}{k} 2^{n^2 - k(n-k)}.$$

In fact, the systems discussed in the next section can automatically construct this formula and compute the model count of Formula 4 in time polynomial in n. On the other hand, existing propositional WMC algorithms require time that is exponential in n on this problem. We note here that the *treewidth* of the grounding of Δ is linear in n.

There are space considerations that motivate first-order model counting as well. When converting a WFOMC problem to WMC, the grounding of Δ has size polynomial in the size of $\mathbf{D}$, but the degree of this polynomial can be high. When the grounding does not fit into memory, even approximate WMC becomes a problem.

Algorithms

Several algorithms exist for solving propositional WMC. Exact solvers are based on either exhaustive DPLL search (Sang, Beame, and Kautz 2005), or knowledge compilation to a circuit language that supports efficient model counting, such as d-DNNF (Darwiche 2002; Chavira and Darwiche 2008) or SDD (Choi, Kisa, and Darwiche 2013). Approximate WMC algorithms use local search (Wei and Selman 2005) or sampling (Gogate and Dechter 2011).

More recently, algorithms were introduced that directly solve the WFOMC task. They take a WFOMC problem and automatically generate and evaluate the types of expressions shown in the previous section. Their elementary operations include exponentiation, summation and binomial coefficients. They are called *lifted inference* algorithms. In particular, two lifted algorithms were proposed for *exact* WFOMC, one based on first-order knowledge compilation (Van den Broeck et al. 2011; Van den Broeck 2011; Van den Broeck 2013), and the other based on first-order DPLL search (Gogate and Domingos 2011). Approximate

algorithms were also proposed, including lifted importance sampling (Gogate and Domingos 2011; Gogate, Jha, and Venugopal 2012). More generally, there is a large literature on exact and approximate *lifted probabilistic inference* in statistical relational models, which can be adapted to solve certain WFOMC tasks. See Kersting (2012) for an overview.

Normal Forms

It is common for logical reasoning algorithms to operate on *normal form* representations instead of arbitrary sentences. For example, propositional SAT solvers and weighted model counters often expect CNF inputs. We distinguish the following first-order normal forms.

– A theory in *prenex normal form* consists of formulas $Q_1 x_1, \ldots, Q_n x_n, \phi$, where each Q_i is either a universal or existential quantifier, and ϕ is quantifier-free.

– A theory in *prenex clausal form* is a theory in prenex normal form where ϕ is a clause.

– A theory in *Skolem normal form* is a theory in prenex normal form where all Q_i are universal quantifiers.

– A *first-order CNF* is a theory in Skolem and prenex clausal form. Thus, all sentences take the form $\forall x_1, \ldots, \forall x_n, l_1 \vee \cdots \vee l_m$.

Existing WFOMC algorithms require a theory to be in *first-order CNF*. The same requirement is often posed by automated theorem provers, such as first-order resolution.

3 Skolemization for WFOMC

It is well known that one can take any arbitrary formula and convert it to prenex clausal form. This involves pushing negations inside, pushing quantifiers to the front, and distributing disjunctions over conjunctions. The situation for Skolem normal form is different.

Motivation

Not every formula can be transformed into an equivalent Skolem normal form. This problem is typically dealt with by *Skolemization*, which eliminates existential quantifiers from a prenex normal form. This is done by replacing existentially quantified variables by Skolem constants and functions. The result is not logically equivalent to the original formula, but only *equisatisfiable* (i.e., satisfiable precisely when the original formula is satisfiable).

The standard Skolemization algorithm is specific to the satisfiability task and may be unsuitable for other tasks. It is particularly unsuitable for WFOMC as it may produce a result with functions, which are not permitted in the WFOMC task. For example, standard Skolemization would transform the formula

$$\forall x, \exists y, \texttt{WorksFor}(x, y) \vee \texttt{Boss}(x) \tag{5}$$

into the following formula with the Skolem function $\texttt{Sk}()$.

$$\forall x, \texttt{WorksFor}(x, \texttt{Sk}(x)) \vee \texttt{Boss}(x).$$

As soon as we allow functions, the Herbrand base becomes infinite, which makes the model counting task ill-defined, therefore, ruling out standard Skolemization for WFOMC.[3]

Algorithm

This section introduces a Skolemization technique for WFOMC. It takes as input a triple $(\Delta, \mathrm{w}, \bar{\mathrm{w}})$ whose Δ is an arbitrary sentence and returns a triple $(\Delta', \mathrm{w}', \bar{\mathrm{w}}')$ whose Δ' is in Skolem normal form (i.e., no existential quantifiers). Such a Δ' can then be turned into first-order CNF using standard transformations. The proposed technique does not introduce functions. It satisfies two properties, one is essential and the other expands the applications of the technique.

The essential property is soundness.

Property 1 (Soundness)**.** *Skolemization of* $(\Delta, \mathrm{w}, \bar{\mathrm{w}})$ *to* $(\Delta', \mathrm{w}', \bar{\mathrm{w}}')$ *is sound iff for any* **D**, *we have that*

$$\mathrm{WFOMC}(\Delta, \mathbf{D}, \mathrm{w}, \bar{\mathrm{w}}) = \mathrm{WFOMC}(\Delta', \mathbf{D}, \mathrm{w}', \bar{\mathrm{w}}').$$

To motivate the second property, we note that one may be interested in queries of the form $\mathrm{WFOMC}(\Delta \wedge \phi, \mathbf{D}, \mathrm{w}, \bar{\mathrm{w}})$, where Δ, w and $\bar{\mathrm{w}}$ are fixed, but where ϕ is changing. For example, we will see in Section 4 that probabilistic inference can be reduced to these types of queries. Therefore, we want to achieve a stronger form of soundness.

Property 2 (Modularity)**.** *Skolemization of* $(\Delta, \mathrm{w}, \bar{\mathrm{w}})$ *to* $(\Delta', \mathrm{w}', \bar{\mathrm{w}}')$ *is modular iff for any* **D** *and any sentence* ϕ,

$$\mathrm{WFOMC}(\Delta \wedge \phi, \mathbf{D}, \mathrm{w}, \bar{\mathrm{w}}) = \mathrm{WFOMC}(\Delta' \wedge \phi, \mathbf{D}, \mathrm{w}', \bar{\mathrm{w}}').$$

That is, by replacing ϕ, one does not invalidate the Skolemization obtained under a different ϕ.

The proposed Skolemization algorithm eliminates existential quantifiers one by one. Its basic building block is the following transformation.

Definition 3. Suppose that Δ contains a subexpression of the form $\exists x, \phi(x, \mathbf{y})$, where $\phi(x, \mathbf{y})$ is an arbitrary sentence containing the free logical variables x and $\mathbf{y}$. Let n be the number of variables in $\mathbf{y}$. First, we introduce two new predicates: the *Tseitin predicate* $\texttt{Z}/n$ and the *Skolem predicate* $\texttt{S}/n$. Second, we replace the expression $\exists x, \phi(x, \mathbf{y})$ in Δ by the atom $\texttt{Z}(\mathbf{y})$, and append the formulas

$$\forall \mathbf{y}, \forall x, \ \texttt{Z}(\mathbf{y}) \vee \neg \phi(x, \mathbf{y})$$
$$\forall \mathbf{y}, \ \texttt{S}(\mathbf{y}) \vee \ \texttt{Z}(\mathbf{y})$$
$$\forall \mathbf{y}, \forall x, \ \texttt{S}(\mathbf{y}) \vee \neg \phi(x, \mathbf{y}).$$

The functions w' and $\bar{\mathrm{w}}'$ are equal to w and $\bar{\mathrm{w}}$, except that $\mathrm{w}'(\texttt{Z}) = \bar{\mathrm{w}}'(\texttt{Z}) = \mathrm{w}'(\texttt{S}) = 1$ and $\bar{\mathrm{w}}'(\texttt{S}) = -1$.

In the resulting theory Δ', a single existential quantifier is now eliminated. This building block can eliminate single universal quantifiers as well. When Δ contains a subexpression $\forall x, \phi(x, \mathbf{y})$, we replace it by $\neg \exists x, \neg \phi(x, \mathbf{y})$, whose existential quantifier can be eliminated with Definition 3.

We can now show the following.

---

[3]One could obtain a Skolem normal form by grounding existential quantifiers, replacing them by large, but finite disjunctions. While this may still permit limited runtime improvements on vacuous formulas, it is for all practical purposes equivalent to reducing the WFOMC problem to a WMC problem. Moreover, that transformation is dependent on the domain and leads to large formulas whose conversion to CNF blows up (e.g., when grounding $\exists x \forall y$).

Theorem 3 (Modularity). *Repeated application of Definition 3 comprises a modular Skolemization algorithm.*

The detailed proof can be found in the appendix.

Intuition

Our Skolemization algorithm implicitly tries to enforce an equivalence between the eliminated subexpression and the Tseitin predicate[4], which is explicitly written as

$$\forall \mathbf{y}, \ \mathtt{Z}(\mathbf{y}) \Leftrightarrow [\exists x, \phi(x, \mathbf{y})].$$

This equivalence contains an existential quantifier so it cannot be represented explicitly. Instead, the algorithms enforces a relaxed equivalence, represented by the three formulas in Definition 3. The intuition is that by relaxing the equivalence we introduce additional models to the theory, but for every additional model with weight W, there is exactly one additional model with weight $-W$.[5] The WFOMC therefore stays the same.

The interaction between the three relaxed formulas, the intended equivalence, and the model weights becomes more apparent after a case analysis on $\mathtt{Z}(\mathbf{y})$:

1. When $\mathtt{Z}(\mathbf{y})$ is false, it implies that $\exists x, \phi(x, \mathbf{y})$ is false, which is intended. It also implies that $\mathtt{S}(\mathbf{y})$ is true, which does not change the model count, since we multiply by 1.

2. When $\mathtt{Z}(\mathbf{y})$ is true, it implies that only three states of $\mathtt{S}(\mathbf{y})$ and $\exists x, \phi(x, \mathbf{y})$ are allowed:

 (a) $\exists x, \phi(x, \mathbf{y})$ is true and $\mathtt{S}(\mathbf{y})$ is true. This is again intended, because $\mathtt{Z}(\mathbf{y})$ and $\exists x, \phi(x, \mathbf{y})$ are equivalent.

 (b) $\exists x, \phi(x, \mathbf{y})$ is false and $\mathtt{S}(\mathbf{y})$ is true. This is an unintended state with a positive weight W.

 (c) $\exists x, \phi(x, \mathbf{y})$ is true and $\mathtt{S}(\mathbf{y})$ is false. This is an unintended state with a weight $-W$. The negative weight comes from the fact that $\bar{\mathrm{w}}(\mathtt{S}) = -1$.

The weights of the unintended models cancel each other out.

Examples

We will now illustrate our Skolemization algorithm on concrete examples. Suppose that Δ is Formula 5, that is,

$$\forall x, \exists y, \ \mathtt{WorksFor}(x, y) \vee \mathtt{Boss}(x).$$

We can apply Definition 3 to the subexpression $\exists y, \mathtt{WorksFor}(x, y) \vee \mathtt{Boss}(x)$, resulting in a Δ' equal to

$$\forall x, \ \mathtt{Z}(x)$$
$$\forall x, \forall y, \ \mathtt{Z}(x) \vee \neg [\mathtt{WorksFor}(x, y) \vee \mathtt{Boss}(x)]$$
$$\forall x, \ \mathtt{Z}(x) \vee \mathtt{S}(x)$$
$$\forall x, \forall y, \ \mathtt{S}(x) \vee \neg [\mathtt{WorksFor}(x, y) \vee \mathtt{Boss}(x)].$$

The first formulas is the original formula with the subexpression substituted by $\mathtt{Z}(x)$.

To get a better insight into the result, we will simplify it using first-order unit propagation (Van den Broeck et al.

---

[4]This equivalence represents a set of propositional Tseitin encodings, in which each $\mathtt{Z}$ atom is a Tseitin variable (Tseitin 1983).

[5]This is not dissimilar to the inclusion-exclusion principle.

2011) while noting that the first formula is a unit clause. The simplified theory is

$$\forall x, \forall y, \ \mathtt{S}(x) \vee \neg \mathtt{WorksFor}(x, y)$$
$$\forall x, \ \mathtt{S}(x) \vee \neg \mathtt{Boss}(x).$$

We verify the correctness of this Skolemization as follows.

– When $\mathtt{Boss}(x)$ is true, the formula is satisfied for x, and the models of Δ' are intended, that is, they correspond to models of Δ. Indeed, $\mathtt{S}(x)$ is entailed to be true and the model weights are multiplied by one.

– When $\mathtt{Boss}(x)$ is false and $\mathtt{WorksFor}(x, y)$ is true for at least one y, then $\mathtt{S}(x)$ is entailed to be true. Again these models are intended, because $\exists y, \mathtt{WorksFor}(x, y) \vee \mathtt{Boss}(x)$ is now satisfied for x. The model weights are multiplied by one.

– When $\mathtt{Boss}(x)$ is false and $\mathtt{WorksFor}(x, y)$ is false for all y then $\mathtt{S}(x)$ can be either true or false. This is where unintended models appear, once with $\mathtt{S}(x)$ true and once with $\mathtt{S}(x)$ false. Because they have opposing weights, the contributions of these unintended models cancel out.

As a second example, consider Δ to be

$$\forall x, \exists y, \exists \mathtt{z}, \ \mathtt{Parents}(x, y, \mathtt{z}) \vee \mathtt{Adam}(x).$$

Skolemization of the inner existential quantifier results in

$$\forall x, \exists y, \ \mathtt{Z}_1(x, y)$$
$$\forall x, \forall y, \forall \mathtt{z}, \ \mathtt{Z}_1(x, y) \vee \neg [\mathtt{Parents}(x, y, \mathtt{z}) \vee \mathtt{Adam}(x)]$$
$$\forall x, \forall y, \ \mathtt{Z}_1(x, y) \vee \mathtt{S}_1(x, y)$$
$$\forall x, \forall y, \forall \mathtt{z}, \ \mathtt{S}_1(x, y) \vee \neg [\mathtt{Parents}(x, y, \mathtt{z}) \vee \mathtt{Adam}(x)]$$

This example shows the need for a Tseitin predicate $\mathtt{Z}_1$. The first sentence still contains an existential quantifier. One more elimination and unit propagation step replaces that sentence by $\forall y, \forall x, \ \mathtt{S}_2(\mathbf{y}) \vee \neg \mathtt{Z}_1(x, y)$ and the result is in Skolem normal form.

Properties

Theorem 3 suggests the repeated application of Definition 3, as long as the sentence contains an existential quantifier, or a universal quantifier not in prenex form. This approach has one caveat: eliminating $\exists x, \phi(x, \mathbf{y})$ adds the expression $\neg \phi(x, \mathbf{y})$ to Δ'. When we eliminate quantifiers from left to right, $\phi(x, \mathbf{y})$ itself can contain quantifiers. This operation will introduce new quantifiers in $\neg \phi(x, \mathbf{y})$ and cause a blow up due to the duplication in the newly added formulas. This can be avoided by eliminating from right to left, that is, from innermost to outermost. We can show the following theorem, whose proof is in the appendix.

Theorem 4 (Termination and Complexity). *Repeated application of Definition 3 will terminate with a sentence in Skolem normal form. Moreover, this can be achieved in time polynomial in the size of Δ.*

The resulting Skolem normal form sentence can subsequently be transformed into first-order CNF. When using Tseitin's transformation (Tseitin 1983), this can even be done in polynomial time.

In our first example, the Tseitin predicate Z could be removed from Δ' by unit propagation. The following proposition generalizes that observation.

Proposition 5. *Suppose that we are eliminating a subexpression* $\exists x, \phi(x, \mathbf{y})$ *from a sentence* $\forall \mathbf{y}, \exists \mathbf{x}, \phi(x, \mathbf{y})$ *using the procedure of Definition 3. That is, the existential quantifier in this subexpression is only preceded by universal quantifiers. Then, we can avoid adding Tseitin predicate Z and instead define* Δ' *to be*

$$\forall \mathbf{y}, \forall x, \ \mathtt{S}(\mathbf{y}) \vee \neg \phi(x, \mathbf{y}).$$

This simplifies the transformation when applicable, in particular when Δ is already in prenex normal form.

4 WFOMC Encodings

We will show in this section how the proposed Skolemization technique can extend the scope of first-order model counters to new situations. We will consider in particular one undirected first-order probabilistic language (Markov Logic) and one directed language (Probabilistic Logic Programs). First-order model counters currently apply to a subset of the first representation, and not to the second representation. With Skolemization, these model counters can now be applied to both. Our treatment is based on providing WFOMC encodings of these representations, to which our Skolemization technique is then applied.[6]

Consider a first-order probabilistic model that induces the distribution $\mathrm{Pr}_{\mathbf{D}}(.)$ for domain $\mathbf{D}$. A *WFOMC encoding* of this model is a triple $(\Delta, \mathrm{w}, \bar{\mathrm{w}})$ which guarantees that for any sentence ϕ (usually a conjunction of literals) and domain $\mathbf{D}$, we have that

$$\mathrm{Pr}_{\mathbf{D}}(\phi) = \frac{\mathrm{WFOMC}(\Delta \wedge \phi, \mathbf{D}, \mathrm{w}, \bar{\mathrm{w}})}{\mathrm{WFOMC}(\Delta, \mathbf{D}, \mathrm{w}, \bar{\mathrm{w}})}.$$

Markov Logic Networks

We will now introduce a WFOMC encoding for *Markov logic networks* (MLN) (Richardson and Domingos 2006).

Representation An MLN is a set of tuples (w, ψ), where w is a real number representing a weight and ψ is a formula in first-order logic. When w is infinite, ψ represents a first-order logic constraint, also called a *hard formula*.

Building further on the example given before, consider the following MLN

$$1.3 \quad \exists y, \ \mathtt{WorksFor}(x, y) \vee \mathtt{Boss}(x). \tag{6}$$

This statement softens the logical sentence we saw earlier. Instead of saying that every person either has a boss, or is a boss, it states that worlds with many employed people are more likely. That is, it is now possible to have a world with unemployed people, but the more unemployed people there are, the lower the probability of that world.

The semantics of a first-order MLN Φ is defined in terms of its *grounding* for a given domain of constants $\mathbf{D}$. The grounding of Φ is the MLN obtained by first grounding all

---

[6]These encodings are implemented in the WFOMC system: `http://dtai.cs.kuleuven.be/wfomc`

its quantifiers and then replacing each formula in Φ with all its groundings (using the same weight). With the domain $\mathbf{D} = \{\mathsf{A}, \mathsf{B}\}$ (e.g., two people, Alice and Bob), the above first-order MLN represents the following grounding.

$$1.3 \quad \mathtt{WorksFor}(\mathsf{A}, \mathsf{A}) \vee \mathtt{WorksFor}(\mathsf{A}, \mathsf{B}) \vee \mathtt{Boss}(\mathsf{A})$$
$$1.3 \quad \mathtt{WorksFor}(\mathsf{B}, \mathsf{A}) \vee \mathtt{WorksFor}(\mathsf{B}, \mathsf{B}) \vee \mathtt{Boss}(\mathsf{B})$$

This ground MLN contains six different random variables, which correspond to all groundings of atoms $\mathtt{WorksFor}(x, y)$ and $\mathtt{Boss}(x)$. This leads to a distribution over 2^6 possible worlds (i.e., interpretation). The weight of each world is simply the product of all weights e^w, where (w, γ) is a ground MLN formula and γ is satisfied by the world. The weights of worlds that do not satisfy a hard formula are set to zero. The probabilities of worlds are obtained by normalizing their weights.

Encoding a Markov Logic Network

Definition 4. The WFOMC encoding $(\Delta, \mathrm{w}, \bar{\mathrm{w}})$ of an MLN is constructed as follows. For each MLN formula $(w_i, \phi_i(\mathbf{x}_i))$, where $\mathbf{x}_i$ denotes the free logical variables in ϕ_i, we introduce a parameter predicate $\mathtt{P_i}/|\mathbf{x}_i|$. For each MLN formula, Δ contains the sentence $\forall \mathbf{x}_i, \ \mathtt{P_i}(\mathbf{x}_i) \Leftrightarrow \phi_i(\mathbf{x}_i)$. The weight function sets $\mathrm{w}(\mathtt{P_i}) = e^{w_i}, \bar{\mathrm{w}}(\mathtt{P_i}) = 1$, and $\mathrm{w}(\mathtt{Q}) = \bar{\mathrm{w}}(\mathtt{Q}) = 1$ for all other predicates $\mathtt{Q}$.

Each $\mathtt{P_i}$ captures the truth value of ϕ_i and carries its weight. Hard formulas can directly be encoded as constraints.

The encoding of Formula 6 has Δ equal to

$$\forall x, \ \mathtt{P}(x) \Leftrightarrow \exists y, \ \mathtt{WorksFor}(x, y) \vee \mathtt{Boss}(x).$$

Its w maps P to $e^{1.3}$ and all other predicates to 1. Its $\bar{\mathrm{w}}$ maps all predicates to 1.

As discussed in Section 2, WFOMC algorithms require first-order CNF input. Definition 4 will only yield a Δ in Skolem normal form (and thus rewritable into CNF) if the MLN formulas are quantifier-free. Then, the only quantifiers in Δ are the universal ones introduced by the encoding itself. Therefore, Van den Broeck et al. (2011) and Gogate and Domingos (2011) resort to *grounding all quantifiers* in the MLN formulas so as to obtain a CNF. This makes the WFOMC encoding specific to the domain $\mathbf{D}$, and partly removes first-order structure from the problem.

Our discussion is based on Van den Broeck et al. (2011). It is similar to the encoding of Gogate and Domingos (2011), whose parameter predicates have more arguments. While these encodings are specific to MLNs, it is straightforward to generalize them to other undirected languages, such as *parfactor graphs* (Poole 2003).

Applying Skolemization We can now perform WFOMC inference in MLNs with quantifiers. Skolemization and CNF

conversion for the example above results in a Δ' equal to

$$\forall x, \ \mathtt{P}(x) \vee \neg\mathtt{Z}(x)$$
$$\forall x, \ \neg\mathtt{P}(x) \vee \mathtt{Z}(x)$$
$$\forall x, \forall y, \ \mathtt{S}(x) \vee \neg\mathtt{WorksFor}(y, x)$$
$$\forall x, \ \mathtt{S}(x) \vee \neg\mathtt{Boss}(x)$$
$$\forall x, \ \mathtt{S}(x) \vee \mathtt{Z}(x)$$
$$\forall x, \forall y, \ \mathtt{Z}(x) \vee \neg\mathtt{WorksFor}(x, y)$$
$$\forall x, \forall y, \ \mathtt{Z}(x) \vee \neg\mathtt{Boss}(x)$$

This theory can be used for WFOMC inference.

Probabilistic Logic Programs

We now show a WFOMC encoding for a directed first-order probabilistic language. The encoding is explained for the *ProbLog* language (De Raedt, Kimmig, and Toivonen 2007; Fierens et al. 2013).

ProbLog Representation ProbLog extends logic programs with facts that are annotated with probabilities. A ProbLog program Φ is a set of probabilistic facts F and a regular logic program L. A probabilistic fact $p :: a$ consists of a probability p and an atom a. A logic program is a set of rules, with the form Head:-Body, where the head is an atom and the body is a conjunction of literals. For example,

$$0.1 :: \mathtt{Attends}(x).$$
$$0.3 :: \mathtt{ToSeries}(x).$$
$$\mathtt{Series}\text{:-}\mathtt{Attends}(x), \mathtt{ToSeries}(x).$$

This program expresses that if more people attend a workshop, it more likely turns into a series of workshops.

The semantics of a ProbLog program Φ are defined by a distribution over the *groundings* of the probabilistic facts for a given domain of constants **D** (Sato 1995).[7] The probabilistic facts $p_i :: a_i$ induce a set of possible worlds, one for each possible partition of a_i in positive and negative literals. The set of true a_i literals with the logic program L define a well-founded model (Van Gelder, Ross, and Schlipf 1991). The probability of such a model is the product of p_i for all true a_i literals and $1 - p_i$ for all false a_i literals.

For the domain **D** $= \{A, B\}$ (two people), the above first-order ProbLog program represents the following grounding:

$$0.1 :: \mathtt{Attends}(A).$$
$$0.1 :: \mathtt{Attends}(B).$$
$$0.3 :: \mathtt{ToSeries}(A).$$
$$0.3 :: \mathtt{ToSeries}(B).$$
$$\mathtt{Series}\text{:-}\mathtt{Attends}(A), \mathtt{ToSeries}(A).$$
$$\mathtt{Series}\text{:-}\mathtt{Attends}(B), \mathtt{ToSeries}(B).$$

This ground ProbLog program contains 4 probabilistic facts which corresponds to 2^4 possible worlds. The

---

[7]Our treatment assumes a function-free and finite-domain fragment of ProbLog. Starting from classical ProbLog semantics, one can obtain the finite function-free domain for a given query by exhaustively executing the Prolog program and keeping track of the goals that are called during resolution.

weight of, for example, the world in which Attends(A) and ToSeries(A) are true would be $0.1 \cdot (1 - 0.1) \cdot 0.3 \cdot (1 - 0.3) = 0.0189$ and the model would be $\{\mathtt{Attends}(A), \mathtt{ToSeries}(A), \mathtt{Series}\}$.

Encoding a ProbLog Program The transformation from a ProbLog program to a first-order logic theory is based on Clark's completion (Clark 1978). This is a transformation from logic programs to first-order logic. For certain classes of programs, called *tight* logic programs (Fages 1994), it is correct, in the sense that every model of the logic program is a model of the completion, and vice versa. Intuitively, for each predicate P, the completion contains a single sentence encoding all its rules. These rules have the form $\mathtt{P}(\mathbf{x})\text{:-}b_i(\mathbf{x}, \mathbf{y}_i)$, where b_i is a body and $\mathbf{y}_i$ are the variables that appear in the body b_i but not in the head. The sentence encoding these rules in the completion is $\forall\mathbf{x}, \ \mathtt{P}(\mathbf{x}) \Leftrightarrow \bigvee_i \exists\mathbf{y}_i, \ b_i(\mathbf{x}, \mathbf{y}_i)$. If the program contains cyclic rules, the completion is not sound, and, it is necessary to first apply a conversion to remove positive loops (Janhunen 2004).

Definition 5. The WFOMC encoding $(\Delta, \mathrm{w}, \bar{\mathrm{w}})$ of a tight ProbLog program has Δ equal to Clark's completion of L. For each probabilistic fact[8] $p :: a$ we set the weight function to $\mathrm{w}(\mathrm{pred}(a)) = p$ and $\bar{\mathrm{w}}(\mathrm{pred}(a)) = 1 - p$.

Again, a Skolem normal form is required to use WFOMC. However, we get this form only when the variables that appear in the body of a rule also appear in the head of a rule. This is not the case for most Prolog programs though. For example, if we apply Definition 5 to the example above, an existential quantifier appears in the sentence:

$$\mathtt{Series} \Leftrightarrow \exists x, \mathtt{Attends}(x) \wedge \mathtt{ToSeries}(x).$$

Furthermore, w maps Attends to 0.1 and ToSeries to 0.3, and $\bar{\mathrm{w}}$ maps Attends to 0.9 and ToSeries to 0.7. Both w and $\bar{\mathrm{w}}$ are 1 for all other predicates. This example is not in Skolem normal form and requires Skolemization before it can be processed by WFOMC algorithms.

Applying Skolemization Skolemization followed by CNF conversion gives a Δ' equal to

$$\mathtt{Series} \vee \neg\mathtt{Z}$$
$$\neg\mathtt{Series} \vee \mathtt{Z}$$
$$\forall x, \ \mathtt{Z} \vee \neg\mathtt{Attends}(x) \vee \neg\mathtt{ToSeries}(x)$$
$$\mathtt{Z} \vee \mathtt{S}$$
$$\forall x, \ \mathtt{S} \vee \neg\mathtt{Attends}(x) \vee \neg\mathtt{ToSeries}(x)$$

Sentence Δ' is in Skolem normal form and is now processable by WFOMC algorithms.

A simple ProbLog program as the one above is identical to a noisy-or structure (Cozman 2004), popular in Bayesian network modeling. Skolemization thus offers a fundamental method to lift first-order, directed structures, such as the noisy-or, in a generic manner (see also Section 6).

---

[8]If multiple probabilistic facts are defined for the same predicate, auxiliary predicates need to be introduced.

5 Liftability Implications

In our motivation for introducing first-order model counting, we touched upon the runtime and complexity improvements that can be attained by first-order counting. These complexity improvements have inspired a particular notion of lifted inference, called *domain-lifted* inference, which says that a WFOMC algorithm is lifted when it runs in time polynomial in the size of **D** (Van den Broeck 2011).

While this notion of lifted inference may not capture everyone's perception of lifting, it does provide a clear formal framework. In particular, we can now talk about classes of sentences Δ for which an algorithm is domain-lifted. We say that the algorithm is *complete* for those classes. We can also talk about classes of sentences Δ for which there exists, or cannot exist a domain-lifted algorithm. We call the former classes *liftable* (Jaeger and Van den Broeck 2012).

All existing completeness and liftability theorems require that Δ is in first-order CNF. This requirement carries over from the existing WFOMC algorithms. Given our Skolemization algorithm, we can now restate these theorems to apply more generally. For example, the positive liftability result of Van den Broeck (2011) becomes the following

Corollary 6. *Suppose that Δ is a theory of sentences with up to two logical variables, and otherwise arbitrary structure. The complexity of computing the WFOMC of Δ is polynomial in the size of **D**. That is, this class is domain-liftable.*

Other notions of liftability also include queries ϕ in the complexity analysis, since they are important for lifted probabilistic inference. Based on Van den Broeck and Davis (2012), and Van den Broeck and Darwiche (2013), we can now claim the following.

Corollary 7. *Suppose that Δ is a theory of sentences with up to two logical variables, and otherwise arbitrary structure. The complexity of computing the WFOMC of $\Delta \wedge \phi$ is polynomial in the size of **D** and ϕ, provided that ϕ is a conjunction of only unary literals, and binary literals of bounded Boolean rank.*

These WFOMC liftability theorems have direct implications for all languages with a WFOMC encoding. For example, we can now say that MLNs with up to two logical variables per formula are domain-liftable, regardless of the quantifiers used. Previously, this was only true for quantifier-free MLNs. We can now also show that ProbLog programs with up to two logical variables per clause are guaranteed to be liftable. This is the first such liftability result for probabilistic logic programs.

6 Related Work

In the encodings for MLNs and probabilistic logics, the weight functions (indirectly) represent probabilities and are therefore always positive. Our Skolemization algorithm introduces negative weights. This might appear odd when interpreting the weights as negative probabilities. This issue has been discussed before. For example, Feynman (1987) writes *"Negative probabilities allow an abstract calculation which permits freedom to do mathematical calculations in any order simplifying the analysis enormously"*.

The potential of negative probabilities was already observed by Jha and Suciu (2012) for answering queries in probabilistic databases and served as inspiration for our approach. Probabilistic databases (Suciu et al. 2011) are fundamentally a type of first-order probabilistic model. It can be viewed as a special type of weighted model counting problem (Δ, w), where the weight function encodes the probability $w(t)$ with which a tuple t can be found in the database. A query on such a database is typically a union of conjunctive queries (UCQ), which corresponds to a monotone DNF sentence Δ. A noticeable difference with most WMC solvers (and WFOMC) is that the solvers for probabilistic databases expect the theory Δ to be in DNF instead of CNF. Different from WFOMC is that although the query (i.e., Δ) is first-order, the weight function is defined on the propositional level like in WMC. Weights are thus assigned to ground literals (the tuples) whereas for WFOMC weights are assigned to predicates (the tables). This allows WFOMC to exploit more types of symmetries.

Jha and Suciu (2012) propose to extend probabilistic databases with MarkoViews, a representation similar to MLNs, in which each weighted formula is again a UCQ query, that is, a monotone DNF. To compute the probability of a query, they introduce negative tuple probabilities.

The use of negative probabilities has also come up for optimizing calculations for specific structures in probabilistic graphical models like noisy-or (Díez and Galán 2003). This particular case has been translated to the first-order case by Kisynski and Poole (2011) and resulted in an approach to lift noisy-or structures. In Section 4 we showed how the application of Skolemization leads to lifting noisy-or and both methods turn out to output a similar encoding for this particular case. Therefore, the approach followed by Kisynski and Poole (2011) can be considered a special case of the Skolemization algorithm applied to a noisy-or model.

Jaeger (2012) shows a negative liftability proof that uses *relational Skolemization*. Similar to our approach, subexpressions containing an existential quantifier are transformed and relaxed to eliminate the quantifier. Relational Skolemization, however, does not guarantee a correct model count. It rather guarantees that if the weight of a model is non-zero it will also be non-zero in the Skolemized version.

7 Conclusions

In this paper, we introduced a Skolemization procedure that is sound for weighted first-order model counting. It extends the applicability of first-order model counters to encodings which require an existential quantifier such as Markov logic models with quantifiers and probabilistic logic programs. It also extends the class of first-order sentences whose models we can count efficiently.

Acknowledgments

Appendix

A Proof of Theorem 3

We will now prove the sequence of steps that leads to the removal of an existential quantifier in Δ to obtain Δ', w' and $\bar{w}'$ while maintaining modularity. To replace the expression $\exists x, \phi(x, \mathbf{y})$ we perform the following steps.

Isolate the Quantifier Introduce a new *Tseitin* predicate Z/n, whose arity n is the number of $\mathbf{y}$ variables. Set $w'(Z) = \bar{w}'(Z) = 1$ and for all other predicates P, set $w'(P) = w(P)$ and $\bar{w}'(P) = \bar{w}(P)$. Construct Δ' by replacing the expression $\exists x, \phi(x, \mathbf{y})$ in Δ by the atom $Z(\mathbf{y})$, and appending the equivalence .

In any grounding of Δ, this step performs a Tseitin encoding of all groundings of $\exists x, \phi(x, \mathbf{y})$. The groundings of $Z(\mathbf{y})$ play the role of Tseitin variables. This step therefore satisfies Property 2.

Split the Equivalence Rewrite equivalence $\forall \mathbf{y}, \ Z(\mathbf{y}) \Leftrightarrow \exists x, \phi(x, \mathbf{y})$ as two implications, $\forall \mathbf{y}, \ Z(\mathbf{y}) \Rightarrow \exists x, \phi(x, \mathbf{y})$ and $\forall \mathbf{y}, \ Z(\mathbf{y}) \Leftarrow \exists x, \phi(x, \mathbf{y})$. In clausal form, these become

$$\forall \mathbf{y}, \exists x, \ \neg Z(\mathbf{y}) \vee \ \phi(x, \mathbf{y})$$
$$\forall \mathbf{y}, \forall x, \ \ Z(\mathbf{y}) \vee \neg \phi(x, \mathbf{y}).$$

This step satisfies Property 2 because it is a logical equivalence.

Convert to a Feature Introduce a new *Skolem predicate* predicate S/n. Set $w(S) = 1$ and $\bar{w}(S) = 0$ and replace the sentence $\forall \mathbf{y}, \exists x, \ \neg Z(\mathbf{y}) \vee \phi(x, \mathbf{y})$ by

$$\forall \mathbf{y}, \ S(\mathbf{y}) \Leftrightarrow \exists x, \neg Z(\mathbf{y}) \vee \phi(x, \mathbf{y}).$$

In all models of the resulting theory where $\forall \mathbf{y}, \exists x, \ \neg Z(\mathbf{y}) \vee \phi(x, \mathbf{y})$ is not satisfied, there will exist a $\mathbf{y}$ for which $\exists x, \neg Z(\mathbf{y}) \vee \phi(x, \mathbf{y})$ is not satisfied. This will cause at least one $S(\mathbf{y})$ atom to be false in those models, which means that the weight of those models is multiplied by 0. The weight of all other models remains the same. This step therefore satisfies Property 2.

Convert to an Implication Set $\bar{w}(S) = -1$ and turn the equivalence $\forall \mathbf{y}, \ S(\mathbf{y}) \Leftrightarrow \exists x, \neg Z(\mathbf{y}) \vee \phi(x, \mathbf{y})$ into an implication $\forall \mathbf{y}, \ S(\mathbf{y}) \Leftarrow \exists x, \neg Z(\mathbf{y}) \vee \phi(x, \mathbf{y})$, which in clausal form becomes

$$\forall \mathbf{y}, \ S(\mathbf{y}) \vee \ Z(\mathbf{y})$$
$$\forall \mathbf{y}, \forall x, \ S(\mathbf{y}) \vee \neg \phi(x, \mathbf{y}).$$

Replacing the equivalence by an implication and changing $\bar{w}(S)$ to -1 is correct for the following reason. Let $S(\mathbf{y}) \Leftrightarrow \Sigma(\mathbf{y})$ be the above equivalence which is in Δ, and let Γ represent all other sentences in Δ (i.e., $\Delta \equiv (\Sigma(\mathbf{y}) \Leftrightarrow S(\mathbf{y})) \wedge \Gamma$). Our goal is now to construct a triple $(\Delta', w', \bar{w}')$, where $\Delta' \equiv (\Sigma(\mathbf{y}) \Rightarrow S(\mathbf{y})) \wedge \Gamma$, such that $\mathrm{WFOMC}(\Delta \wedge \phi, \mathbf{D}, w, \bar{w}) = \mathrm{WFOMC}(\Delta' \wedge \phi, \mathbf{D}, w', \bar{w}')$ for all domains $\mathbf{D}$ and all sentences ϕ.

Let $\Sigma(A)$ and $S(A)$ be any arbitrary grounding of $\Sigma(\mathbf{y})$ and $S(\mathbf{y})$. A case analysis on the values of $\Sigma(A)$ and $S(A)$ shows that $\mathrm{WFOMC}(\Delta \wedge \phi, \mathbf{D}, w, \bar{w})$ and $\mathrm{WFOMC}(\Delta' \wedge \phi, \mathbf{D}, w', \bar{w}')$ consist of the following terms (for compactness we drop $\mathbf{D}$ from the notation since it doesn't change).

$\Sigma(\mathbf{A})$	$S(\mathbf{A})$	$\mathrm{WFOMC}(\Delta \wedge \phi, w, \bar{w})$
1	1	$w(S) \cdot \mathrm{WFOMC}(\Gamma \wedge \Sigma(\mathbf{A}) \wedge \phi, w, \bar{w})$
1	0	0
0	1	0
0	0	$\bar{w}(S) \cdot \mathrm{WFOMC}(\Gamma \wedge \neg\Sigma(\mathbf{A}) \wedge \phi, w, \bar{w})$

$\Sigma(\mathbf{A})$	$S(\mathbf{A})$	$\mathrm{WFOMC}(\Delta' \wedge \phi, w', \bar{w}')$
1	1	$w'(S) \cdot \mathrm{WFOMC}(\Gamma \wedge \Sigma(\mathbf{A}) \wedge \phi, w', \bar{w}')$
1	0	0
0	1	$w'(S) \cdot \mathrm{WFOMC}(\Gamma \wedge \neg\Sigma(\mathbf{A}) \wedge \phi, w', \bar{w}')$
0	0	$\bar{w}'(S) \cdot \mathrm{WFOMC}(\Gamma \wedge \neg\Sigma(\mathbf{A}) \wedge \phi, w', \bar{w}')$

Note that $\bar{w}(S) = 0$ in the encoding of Δ, and that thus

$$\mathrm{WFOMC}(\Delta \wedge \phi, w, \bar{w}) =$$
$$w(S) \cdot \mathrm{WFOMC}(\Gamma \wedge \Sigma(\mathbf{A}) \wedge \phi, w, \bar{w})$$
$$+ 0 \cdot \mathrm{WFOMC}(\Gamma \wedge \neg\Sigma(\mathbf{A}) \wedge \phi, w, \bar{w})$$
$$\mathrm{WFOMC}(\Delta' \wedge \phi, w', \bar{w}') =$$
$$w'(S) \cdot \mathrm{WFOMC}(\Gamma \wedge \Sigma(\mathbf{A}) \wedge \phi, w', \bar{w}')$$
$$+ [w'(S) + \bar{w}'(S)]$$
$$\cdot \mathrm{WFOMC}(\Gamma \wedge \neg\Sigma(\mathbf{A}) \wedge \phi, w', \bar{w}')$$

Setting $w'(P) = w(P)$ for all predicates P except for S ensures that $\mathrm{WFOMC}(\Gamma \wedge \Sigma(\mathbf{A}) \wedge \phi, w, \bar{w}) = \mathrm{WFOMC}(\Gamma \wedge \Sigma(\mathbf{A}) \wedge \phi, w', \bar{w}')$, that $\mathrm{WFOMC}(\Gamma \wedge \neg\Sigma(\mathbf{A}) \wedge \phi, w, \bar{w}) = \mathrm{WFOMC}(\Gamma \wedge \neg\Sigma(\mathbf{A}) \wedge \phi, w', \bar{w}')$. Furthermore, set $w(S) = w'(S)$. What remains for $\mathrm{WFOMC}(\Delta \wedge \phi, w, \bar{w})$ to equal $\mathrm{WFOMC}(\Delta' \wedge \phi, w', \bar{w}')$ is that $w'(S) + \bar{w}'(S) = w(S) + \bar{w}(S) = 0$, which is achieved by setting $\bar{w}'(S) = -w(S) = -1$.

B Proof of Theorem 4

Proof. We begin by proving termination. Let the *internal quantifier count* of a sentence be the number of quantifiers it contains, excluding the leading universal quantifiers. Suppose that a sentence has an internal quantifier count of m. We can select any subexpression that starts with a quantifier and apply Skolemization to it (potentially converting $\forall$ into $\exists$ first). This reduces the internal quantifier count of Δ to be at most $m - 1$ because at least one quantifier is removed. New sentences are added, however, containing the Tseitin and Skolem predicates, and expressions $\neg\phi(x, \mathbf{y})$. These sentences also have an internal quantifier count of at most $m-1$. Suppose that the sentences in Δ have an internal quantifier count of at most m_{max}. Applying Skolemization to one quantifier in each sentence reduces the maximal internal quantifier count to at most $m_{max} - 1$. Therefore, by repeating this procedure for a finite number of steps, we obtain a theory with an internal quantifier count of zero, which is in Skolem normal form.

Next, we prove polynomial complexity. We can remove the quantifiers in a sentence Δ one by one, starting from the innermost quantifier. The removed subexpression $\phi(x, \mathbf{y})$ does not contain any quantifiers, so the internal quantifier count of the added formulas is zero. They are in Skolem normal form. The innermost subexpression is replaced by a Tseitin predicate, reducing the internal quantifier count by

one. The number of required elimination steps before the entire sentence is in Skolem normal form is thus equal to the number of quantifiers in Δ. Moreover, the number of added formulas, and their size, is polynomial in the size of Δ. □

References

Boutilier, C.; Friedman, N.; Goldszmidt, M.; and Koller, D. 1996. Context-specific independence in Bayesian networks. In *Proceedings of UAI*, 115–123. Morgan Kaufmann Publishers Inc.

Chavira, M., and Darwiche, A. 2005. Compiling Bayesian networks with local structure. In *Proceedings of IJCAI*, volume 19, 1306.

Chavira, M., and Darwiche, A. 2008. On probabilistic inference by weighted model counting. *Artificial Intelligence* 172(6-7):772–799.

Chavira, M.; Darwiche, A.; and Jaeger, M. 2006. Compiling relational Bayesian networks for exact inference. *International Journal of Approximate Reasoning* 42(1-2):4–20.

Choi, A.; Kisa, D.; and Darwiche, A. 2013. Compiling probabilistic graphical models using sentential decision diagrams. In *Proceedings of the 12th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*.

Clark, K. 1978. Negation as failure. In *Readings in nonmonotonic reasoning*, 311–325. Morgan Kaufmann Publishers.

Cozman, F. G. 2004. Axiomatizing noisy-or. In *Proceedings of European Conference on Artificial Intelligence (ECAI)*, 979–980.

Darwiche, A. 2002. A logical approach to factoring belief networks. *Proceedings of KR* 409–420.

De Raedt, L.; Frasconi, P.; Kersting, K.; and Muggleton, S., eds. 2008. *Probabilistic inductive logic programming: theory and applications*. Berlin, Heidelberg: Springer-Verlag.

De Raedt, L.; Kimmig, A.; and Toivonen, H. 2007. Problog: A probabilistic prolog and its application in link discovery. In *Proceedings of IJCAI*, volume 7, 2462–2467.

Díez, F. J., and Galán, S. F. 2003. Efficient computation for the noisy max. *International Journal of Intelligent Systems* 18(2):165–177.

Fages, F. 1994. Consistency of Clark's completion and existence of stable models. *Journal of Methods of Logic in Computer Science* 1:51–60.

Feynman, R. P. 1987. Negative probability. *Quantum implications: essays in honour of David Bohm* 235–248.

Fierens, D.; Van den Broeck, G.; Thon, I.; Gutmann, B.; and De Raedt, L. 2011. Inference in probabilistic logic programs using weighted CNF's. In *Proceedings of UAI*, 211–220.

Fierens, D.; Van den Broeck, G.; Renkens, J.; Shterionov, D.; Gutmann, B.; Thon, I.; Janssens, G.; and De Raedt, L. 2013. Inference and learning in probabilistic logic programs using weighted Boolean formulas. *Theory and Practice of Logic Programming*.

Getoor, L., and Taskar, B., eds. 2007. *An Introduction to Statistical Relational Learning*. MIT Press.

Gogate, V., and Dechter, R. 2011. Samplesearch: Importance sampling in presence of determinism. *Artificial Intelligence* 175(2):694–729.

Gogate, V., and Domingos, P. 2011. Probabilistic theorem proving. In *Proceedings of UAI*, 256–265.

Gogate, V.; Jha, A. K.; and Venugopal, D. 2012. Advances in lifted importance sampling. In *Proceedings of AAAI*, 1910–1916.

Gomes, C. P.; Sabharwal, A.; and Selman, B. 2009. Model counting. *Handbook of Satisfiability* 185:633–654.

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.

Jaeger, M., and Van den Broeck, G. 2012. Liftability of probabilistic inference: Upper and lower bounds. In *Proceedings of StarAI*.

Jaeger, M. 2012. Lower complexity bounds for lifted inference. *arXiv preprint arXiv:1204.3255*.

Janhunen, T. 2004. Representing normal programs with clauses. In *Proceedings of European Conference on Artificial Intelligence (ECAI)*, volume 16, 358.

Jha, A., and Suciu, D. 2012. Probabilistic databases with MarkoViews. *Proceedings of the VLDB Endowment* 5(11):1160–1171.

Kersting, K. 2012. Lifted probabilistic inference. In *Proceedings of European Conference on Artificial Intelligence (ECAI)*.

Kimmig, A.; Van den Broeck, G.; and De Raedt, L. 2012. Algebraic model counting. *arXiv preprint arXiv:1211.4475*.

Kisynski, J., and Poole, D. 2011. Lifted aggregation in directed first-order probabilistic models. In *Proceedings of IJCAI*, 1922–1929.

Poole, D. 2003. First-order probabilistic inference. In *Proceedings of IJCAI*, 985–991.

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

Sang, T.; Beame, P.; and Kautz, H. 2005. Solving Bayesian networks by weighted model counting. In *Proceedings of AAAI*, volume 1, 475–482.

Sato, T. 1995. A statistical learning method for logic programs with distribution semantics. In *Proceedings of the 12th International Conference on Logic Programming (ICLP)*, 715–729.

Suciu, D.; Olteanu, D.; Ré, C.; and Koch, C. 2011. Probabilistic databases. *Synthesis Lectures on Data Management* 3(2):1–180.

Tseitin, G. S. 1983. On the complexity of derivation in propositional calculus. In *Automation of Reasoning*. Springer. 466–483.

Van den Broeck, G., and Darwiche, A. 2013. On the complexity and approximation of binary evidence in lifted inference. In *Advances in Neural Information Processing Systems 26 (NIPS)*.

Van den Broeck, G., and Davis, J. 2012. Conditioning in first-order knowledge compilation and lifted probabilistic inference. In *Proceedings of AAAI*.

Van den Broeck, G.; Taghipour, N.; Meert, W.; Davis, J.; and De Raedt, L. 2011. Lifted Probabilistic Inference by First-Order Knowledge Compilation. In *Proceedings of IJCAI*, 2178–2185.

Van den Broeck, G. 2011. On the completeness of first-order knowledge compilation for lifted probabilistic inference. In *Advances in Neural Information Processing Systems 24 (NIPS)*, 1386–1394.

Van den Broeck, G. 2013. *Lifted Inference and Learning in Statistical Relational Models*. Ph.D. Dissertation, KU Leuven.

Van Gelder, A.; Ross, K. A.; and Schlipf, J. S. 1991. The wellfounded semantics for general logic programs. *Journal of the ACM (JACM)* 38(3):619–649.

Wei, W., and Selman, B. 2005. A new approach to model counting. In *Theory and Applications of Satisfiability Testing*, 96–97. Springer.