Lifted Inference for Probabilistic Logic Programs*

Wannes Meert[1], Guy Van den Broeck[1,2], and Adnan Darwiche[2]

[1] Computer Science Department,
KU Leuven, Belgium
[2] Computer Science Department
University of California, Los Angeles

Abstract. First-order model counting emerged recently as a novel reasoning task, at the core of efficient algorithms for probabilistic logics such as MLNs. 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). 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. Recently, we presented a novel Skolemization algorithm for model counting problems that eliminates existential quantifiers from a first-order logic theory without changing its weighted model count. Our Skolemization procedure extends the applicability of first-order model counters to probabilistic logic programming. For the first time, this enables lifted inference with these representations.

Keywords: Lifted probabilistic inference, Probabilistic logic programs, Skolemization

1 Introduction

A number of inference algorithms for probabilistic logic programs are based on weighted model counting (WMC). In model counting 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. For example, exact inference algorithms for ProbLog [10] encode probabilistic inference as a WMC task, which can then be solved by knowledge compilation [6] or exhaustive DPLL search [19].

WMC plays an important role in inference for *first-order probabilistic* representations in general. These became popular in recent years, in statistical relational learning [12] and probabilistic logic learning [7], which are concerned with

* This is a technical summary of Van den Broeck, Meert, and Darwiche [22], which was published at the 14th International Conference on Principles of Knowledge Representation and Reasoning, extended with an experimental evaluation.

modeling and learning complex logical and probabilistic interactions between large numbers of objects. Efficient algorithms reduce exact probabilistic inference to a WMC problem on a propositional knowledge base [3, 11, 10]. 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. A disadvantage is that the high-level first-order structure is lost. Poole [18] 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. [2] and Gogate and Domingos [13] 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 [21]. 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, however, is that lifted model counters require input in *Skolem normal form* (i.e., without existential quantifiers). This makes it inefficient to apply them to probabilistic logic programs.

In this paper we show how a recently introduced *Skolemization procedure* [22] can be used to apply WFOMC to probabilistic logic programs and thereby perform lifted probabilistic inference for probabilistic logic programs.

2 Background

In this section, we briefly review first-order logic and model counting.

2.1 First-order logic

Throughout this paper, we will work with the *function-free finite-domain* fragment of first-order logic (FOL). An *atom* $\mathtt{P}(t_1, \ldots, t_n)$ consists of predicate $\mathtt{P}/n$ of arity n followed by n arguments, which are either *constants* from a finite domain $\mathbf{D} = \{\mathsf{A}, \mathsf{B}, \ldots\}$ 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.

2.2 Weighted First-order Model Counting

We will now introduce the Weighted First-order Model Counting (WFOMC) task [2]. Given (i) a *sentence* Δ in FOL containing predicates $\mathcal{P}$, (ii) a set of constants $\mathbf{D}$, including the constants in Δ, and (iii) a pair of *weight functions* $\mathrm{w}, \bar{\mathrm{w}} : \mathcal{P} \to \mathbb{R}$, the *weighted first-order model count* (WFOMC) is

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

where ω_0 and ω_1 consist of the true, respectively false, literals in the model 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 ($\bar{\mathrm{w}}$). The weight of a model is the product of its literal weights. Finally, the total count is the sum of the weights of all Herbrand models of Δ. Note that we permit predicate weights to be *negative* numbers, which is crucial for the Skolemization algorithm.

3 Skolemization for WFOMC

A sentence without existential quantifiers is typically obtained using *Skolemization*, which eliminates existential quantifiers from a sentence 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). Because it introduces functions, WFOMC cannot be applied to the resulting sentence.

In this section we repeat the novel Skolemization technique for WFOMC introduced in [22]. It takes as input a triple $(\Delta, \mathrm{w}, \bar{\mathrm{w}})$ whose Δ is an arbitrary sentence and returns a triple $(\Delta', \mathrm{w}', \bar{\mathrm{w}}')$ with the same weighted model count and 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 and passed on to WFOMC. The proposed technique does not introduce functions.

The algorithm eliminates existential quantifiers one by one[3]. Its basic building block is the following transformation.

Definition 1. *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:*

[3] The Skolemization algorithm is implemented in the WFOMC system: `http://dtai.cs.kuleuven.be/wfomc`

the Tseitin predicate $\mathtt{Z}/n$ *and the* Skolem predicate $\mathtt{S}/n$*. Second, we replace the expression* $\exists x, \phi(x, \mathbf{y})$ *in* Δ *by the atom* $\mathtt{Z}(\mathbf{y})$*, and append the formulas*

$$\begin{aligned} \forall \mathbf{y}, \forall x,\ & \mathtt{Z}(\mathbf{y}) \vee \neg\phi(x, \mathbf{y}) \\ \forall \mathbf{y},\ & \mathtt{S}(\mathbf{y}) \vee \ \mathtt{Z}(\mathbf{y}) \\ \forall \mathbf{y}, \forall x,\ & \mathtt{S}(\mathbf{y}) \vee \neg\phi(x, \mathbf{y}). \end{aligned}$$

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

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 1.

The repeated application of Definition 1 comprises a modular Skolemization algorithm. It will terminate with a sentence in Skolem normal form and, moreover, this can be achieved in time polynomial in the size of Δ.

4 Skolem Normal Form Encoding of Probabilistic Logic

We will ilustrate in this section how the proposed Skolemization technique can extend the scope of first-order model counters to probabilistic logic programs. For this paper, we explain lifted inference for the *ProbLog* language [8, 10]. The results, however, generalize to other probabilistic logic languages, such as PRISM [20] or Primula [14].

Consider a probabilistic logic program that induces the distribution $\Pr_{\mathbf{D}}(.)$ 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

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

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, the program

$$\begin{aligned} & 0.1 :: \mathtt{Attends}(x). \\ & 0.3 :: \mathtt{ToSeries}(x). \\ & \mathtt{Series} \texttt{ :- } \mathtt{Attends}(x), \mathtt{ToSeries}(x). \end{aligned}$$

expresses that if more people attend a workshop, it more likely turns into a series of workshops. The first two lines are probabilistic facts F, and the last line is the logic program L.

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 $\mathbf{D}$.[4] 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. 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 $\mathbf{D} = \{A, B\}$ (two people), the above first-order ProbLog program represents the following grounding:

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

This ground ProbLog program contains 4 probabilistic facts which corresponds to 2^4 possible worlds. The weight of, for example, the world in which $\texttt{Attends}(A)$ and $\texttt{ToSeries}(A)$ are true would be $0.1 \cdot (1 - 0.1) \cdot 0.3 \cdot (1 - 0.3) = 0.0189$ and the model would be $\{\texttt{Attends}(A), \texttt{ToSeries}(A), \texttt{Series}\}$.

Encoding a ProbLog Program The transformation from a ProbLog program to a first-order logic theory is based on Clark's completion [4]. This is a transformation from logic programs to first-order logic. For certain classes of programs, called *tight* logic programs, 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 $\texttt{P}$, the completion contains a single sentence encoding all its rules. These rules have the form $\texttt{P}(\mathbf{x}) \texttt{:-} b_i(\mathbf{x}, \mathbf{y}_i)$, 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 $\forall \mathbf{x},\ \texttt{P}(\mathbf{x}) \Leftrightarrow \bigvee_i \exists \mathbf{y}_i,\ b_i(\mathbf{x}, \mathbf{y}_i)$. If the program contains cyclic rules, the completion is not sound, and it is necessary to first apply a conversion to remove positive loops [16].

Definition 2. *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*[5] $p :: a$ *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

[4] Our treatment assumes a function-free and finite-domain fragment of ProbLog. Starting from classical ProbLog semantics, one can obtain a 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.

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

in the head of that rule. This is not the case for most Prolog programs. For example, if we apply Definition 2 to the example above, an existential quantifier appears in the sentence:

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

Furthermore, w maps `Attends` to 0.1 and `ToSeries` to 0.3, and $\bar{\text{w}}$ maps `Attends` to 0.9 and `ToSeries` to 0.7. Both w and $\bar{\text{w}}$ 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. Therefore, current algorithms resort to grounding the quantifier which removes part of the first-order structure and makes the WFOMC encoding specific to the domain $\mathbf{D}$.

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

$$\begin{aligned}
&\texttt{Series} \vee \neg \texttt{Z} \\
&\neg\texttt{Series} \vee \texttt{Z} \\
\forall x,\ &\texttt{Z} \vee \neg\texttt{Attends}(x) \vee \neg\texttt{ToSeries}(x) \\
&\texttt{Z} \vee \texttt{S} \\
\forall x,\ &\texttt{S} \vee \neg\texttt{Attends}(x) \vee \neg\texttt{ToSeries}(x)
\end{aligned}$$

This sentence is in Skolem normal form and can now be processed by WFOMC algorithms.

A simple ProbLog program as the one above is identical to a noisy-or structure [5], popular in Bayesian networks. Negative parameters have also come up for optimizing calculations for the noisy-or structure [9] and this particular structure has been lifted to the first-order case by Kisynski and Poole [17]. The approach followed by Kisynski and Poole [17] can be considered a special case of the Skolemization algorithm applied to a noisy-or structure.

Inference Given the resulting CNF, we computed the probability of `Series` using both propositional WMC and lifted WFOMC. The results are depicted in Fig. 1a and show a significant speedup for the lifted approach with respect to propositional inference.

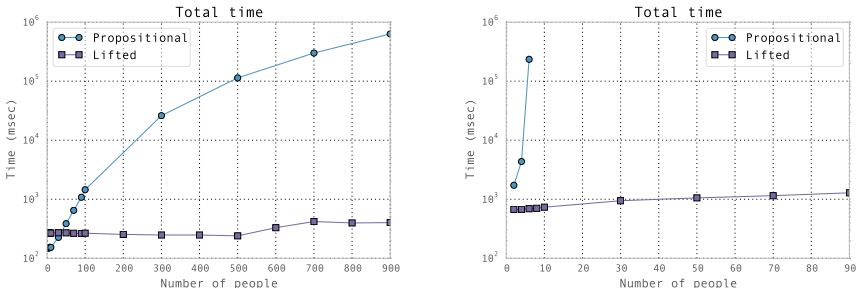
If we extend the example to express that we are mainly interested in attendees that have joint publications, we obtain:

$$\begin{aligned}
&0.1 :: \texttt{Attends}(x). \\
&0.3 :: \texttt{ToSeries}(x). \\
&\texttt{Series} \texttt{ :- } \texttt{Attends}(x), \texttt{Coauthor}(x, y), \texttt{Attends}(y), \texttt{ToSeries}(x, y).
\end{aligned}$$

which after Skolemization and conversion to CNF becomes:

$$\begin{aligned}
&\texttt{Series} \vee \neg \texttt{Z} \\
&\neg\texttt{Series} \vee \texttt{Z} \\
\forall x,\ &\texttt{Z} \vee \neg\texttt{Attends}(x) \vee \neg\texttt{Coauthor}(x,y) \vee \neg\texttt{Attends}(y) \vee \neg\texttt{ToSeries}(x,y) \\
&\texttt{Z} \vee \texttt{S} \\
\forall x,\ &\texttt{S} \vee \neg\texttt{Attends}(x) \vee \neg\texttt{Coauthor}(x,y) \vee \neg\texttt{Attends}(y) \vee \neg\texttt{ToSeries}(x,y)
\end{aligned}$$

Because of the extra logic variable, propositional inference becomes exponential in the size of the domain (see Fig. 1b). Lifted inference on the other hand is polynomial in the size of the domain resulting in an exponential speedup.



(a) For the workshop example. (b) For the extended workshop example.

Fig. 1: Computing the probability of `Series`.

5 Conclusions

In this paper we show how a recently introduced *Skolemization procedure* [22] can be used to apply WFOMC to probabilistic logic programs and perform lifted inference, which is potentially exponentially faster. The liftability theorems that define classes of theories for which WFOMC is domain-liftable [15] and approaches for lifted learning [1] are now also applicable to probabilistic logic programs.

References

1. Van den Broeck, G., Meert, W., Davis, J.: Lifted generative parameter learning. In: Statistical Relational AI (StaRAI) workshop (2013)
2. Van den Broeck, G., Taghipour, N., Meert, W., Davis, J., De Raedt, L.: Lifted Probabilistic Inference by First-Order Knowledge Compilation. In: Proceedings of IJCAI. pp. 2178–2185 (2011)

3. Chavira, M., Darwiche, A., Jaeger, M.: Compiling relational Bayesian networks for exact inference. International Journal of Approximate Reasoning 42(1-2), 4–20 (May 2006)
4. Clark, K.: Negation as failure. In: Readings in nonmonotonic reasoning. pp. 311–325. Morgan Kaufmann Publishers (1978)
5. Cozman, F.G.: Axiomatizing noisy-or. In: Proceedings of European Conference on Artificial Intelligence (ECAI). pp. 979–980 (2004)
6. Darwiche, A.: A logical approach to factoring belief networks. Proceedings of KR pp. 409–420 (2002)
7. De Raedt, L., Frasconi, P., Kersting, K., Muggleton, S. (eds.): Probabilistic inductive logic programming: theory and applications. Springer-Verlag, Berlin, Heidelberg (2008)
8. De Raedt, L., Kimmig, A., Toivonen, H.: Problog: A probabilistic prolog and its application in link discovery. In: Proceedings of IJCAI. vol. 7, pp. 2462–2467 (2007)
9. Díez, F.J., Galán, S.F.: Efficient computation for the noisy max. International Journal of Intelligent Systems 18(2), 165–177 (2003)
10. Fierens, D., Van den Broeck, G., Renkens, J., Shterionov, D., Gutmann, B., Thon, I., Janssens, G., De Raedt, L.: Inference and learning in probabilistic logic programs using weighted Boolean formulas. Theory and Practice of Logic Programming (2013)
11. Fierens, D., Van den Broeck, G., Thon, I., Gutmann, B., De Raedt, L.: Inference in probabilistic logic programs using weighted CNF's. In: Proceedings of UAI. pp. 211–220 (Jul 2011)
12. Getoor, L., Taskar, B. (eds.): An Introduction to Statistical Relational Learning. MIT Press (2007)
13. Gogate, V., Domingos, P.: Probabilistic theorem proving. In: Proceedings of UAI. pp. 256–265 (2011)
14. Jaeger, M.: Relational Bayesian networks. In: Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI). pp. 266–273 (2002)
15. Jaeger, M., Van den Broeck, G.: Liftability of probabilistic inference: Upper and lower bounds. In: Proceedings of StarAI (2012)
16. Janhunen, T.: Representing normal programs with clauses. In: Proceedings of European Conference on Artificial Intelligence (ECAI). vol. 16, p. 358 (2004)
17. Kisynski, J., Poole, D.: Lifted aggregation in directed first-order probabilistic models. In: Proceedings of IJCAI. pp. 1922–1929 (2011)
18. Poole, D.: First-order probabilistic inference. In: Proceedings of IJCAI. pp. 985–991 (2003)
19. Sang, T., Beame, P., Kautz, H.: Solving Bayesian networks by weighted model counting. In: Proceedings of AAAI. vol. 1, pp. 475–482 (2005)
20. Sato, T.: A statistical learning method for logic programs with distribution semantics. In: Proceedings of the 12th International Conference on Logic Programming (ICLP). pp. 715–729 (1995)
21. Van den Broeck, G.: On the completeness of first-order knowledge compilation for lifted probabilistic inference. In: Advances in Neural Information Processing Systems 24 (NIPS). pp. 1386–1394 (2011)
22. Van den Broeck, G., Meert, W., Darwiche, A.: Skolemization for weighted first-order model counting. In: Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR) (2014)