Understanding the Complexity of Lifted Inference and Asymmetric Weighted Model Counting *

Eric Gribkoff
University of Washington
eagribko@cs.uw.edu

Guy Van den Broeck
UCLA, KU Leuven
guyvdb@cs.ucla.edu

Dan Suciu
University of Washington
suciu@cs.uw.edu

Abstract

We highlight our work on lifted inference for the asymmetric Weighted First-Order Model Counting problem (WFOMC), which counts the assignments that satisfy a given sentence in first-order logic. This work is at the intersection of probabilistic databases and statistical relational learning. First, we discuss how adding negation can lower the query complexity, and describe the essential element (resolution) necessary to extend a previous algorithm for positive queries to handle queries with negation. Second, we describe our novel dichotomy result for a non-trivial fragment of first-order CNF sentences with negation. Finally, we discuss directions for future work.

Overview

Our work is concerned with weighted first-order model counting (WFOMC), where we sum the weights of assignments that satisfy a sentence (or query) in finite-domain first-order logic. This reasoning task underlies efficient algorithms for probabilistic reasoning in AI, with statistical relational representations such as Markov logic (Van den Broeck et al. 2011; Gogate and Domingos 2011) and probabilistic programs (Van den Broeck, Meert, and Darwiche 2014). Moreover, WFOMC uncovers a deep connection between AI and database research, where query evaluation in *probabilistic databases* (PDBs) (Suciu et al. 2011) essentially considers the same task. A PDB defines a probability, or weight, for every possible world, and each database query is a sentence encoding a set of worlds, whose combined probability we want to compute. We refer to our setting as *asymmetric* WFOMC, because it allows the weights of each atom to be distinct. This subsumes the symmetric setting considered in AI, where probabilities are set per relation.

The WFOMC task captures query answering in probabilistic database. Take for example the database:

$\texttt{Prof}(\mathsf{Anne}) : 0.9$	$\texttt{Prof}(\mathsf{Charlie}) : 0.1$
$\texttt{Student}(\mathsf{Bob}) : 0.5$	$\texttt{Student}(\mathsf{Charlie}) : 0.8$
$\texttt{Advises}(\mathsf{Anne}, \mathsf{Bob}) : 0.7$	$\texttt{Advises}(\mathsf{Bob}, \mathsf{Charlie}) : 0.1$

and the monotone DNF query:

$$Q \;=\; \exists x, \exists y,\, \texttt{Prof}(x) \wedge \texttt{Advises}(x, y) \wedge \texttt{Student}(y).$$

If we set $\Delta = Q$ and weight function w to map each literal to its probability in the database, then our query answer is:

$$\Pr(Q) = \text{WFOMC}(\Delta, w) = 0.9 \cdot 0.7 \cdot 0.5 = 0.315.$$

We refer to the general case above as *asymmetric* WFOMC, because it allows $w(\texttt{Prof}(\mathsf{Anne}))$ to be different from $w(\texttt{Prof}(\mathsf{Charlie}))$. We use *symmetric* WFOMC to refer to the special case where w simplifies into two weight functions $w^\star, \bar{w}^\star$ that map *predicates* to weights, instead of literals, that is:

$$w(\ell) = \begin{cases} w^\star(P) & \text{when } \ell \text{ is of the form } P(c) \\ \bar{w}^\star(P) & \text{when } \ell \text{ is of the form } \neg P(c) \end{cases}$$

Symmetric WFOMC no longer directly captures PDBs. Yet it can still encode many SRL models, including parfactor graphs (Poole 2003), Markov logic networks (MLNs) (Richardson and Domingos 2006) and probabilistic logic programs (De Raedt et al. 2008). We refer to (Van den Broeck, Meert, and Darwiche 2014) for the details.

We introduce a new WFOMC algorithm **Lift$^{\mathbf{R}}$** that extends the algorithm of (Dalvi and Suciu 2012). This latter algorithm only supports queries without negation, but comes with strong theoretical guarantees in the form of a dichotomy theorem. **Lift$^{\mathbf{R}}$** applies to general CNF queries, with arbitrary negation, by introducing a resolution operation to discover new prime implicates. As we show below, this step is crucial to proving completeness of the algorithm for a subclass of CNF sentences with negation. Our algorithm performs *lifted inference*, meaning that it exploits the relational structure of the query. Moreover, it performs domain-lifted inference, because it runs in time polynomial in the database size (Van den Broeck 2013). **Lift$^{\mathbf{R}}$** is outlined in figure 1; see (Gribkoff, Van den Broeck, and Suciu 2014) for a full description

Negation Can Lower the Complexity

The presence of negations can lower a query's complexity, and **Lift$^{\mathbf{R}}$** exploits this. To see this, consider the query

$$Q \;= \forall x \forall y\; (\texttt{Tweets}(x) \vee \neg\texttt{Follows}(x, y)) \\ \wedge \forall x \forall y\; (\texttt{Follows}(x, y) \vee \neg\texttt{Leader}(y)).$$

*Extended Abstract of (Gribkoff, Van den Broeck, and Suciu 2014)

Algorithm **Lift$^{\mathbf{R}}$**
Input: Ranked and shattered query Q
Probabilistic DB with domain D
Output: $\Pr(Q)$

1 Step 0: If Q is a single ground literal ℓ, return its probability $\Pr(\ell)$ in PDB
2 Step 1: Write Q as a union−CNF: $Q = Q_1 \vee Q_2 \vee \cdots \vee Q_m$
3 Step 2: If $m > 1$ and Q can be partitioned into two sets $Q = Q' \vee Q''$ with disjoint relation symbols, return $1 - (1 - \Pr(Q_1)) \cdot (1 - \Pr(Q_2))$
4 /* **Decomposable Disjunction** */
5 Step 3: If Q cannot be partitioned, return $\sum_{s \subseteq [m]} \Pr(\bigwedge_{i \in s} Q_i)$
6 /* **Inclusion/Exclusion** − perform cancellations before recursion */
7 Step 4: Write Q in CNF: $Q = C_1 \wedge C_2 \wedge \cdots \wedge C_k$
8 Step 5: If $k > 1$, and Q can be partitioned into two sets $Q = Q' \wedge Q''$ with disjoint relation symbols, return $\Pr(Q_1) \cdot \Pr(Q_2)$
9 /* **Decomposable Conjunction** */
10 Step 6: If Q has a separator variable, return $\prod_{a \in D} \Pr(C_1[a/x_1] \wedge \cdots \wedge C_k[a/x_k])$
11 /* **Decomposable Universal Quantifier** */
12 Otherwise **FAIL**

Figure 1: **Algorithm for Computing** $\Pr(Q)$

The query says that if x follows anyone then x tweets, and that everybody follows the leader[1].

Our goal is to compute the probability $\Pr(Q)$, knowing the probabilities of all (ground) atoms in the domain. We note that the two clauses are dependent (since both refer to the relation `Follows`), hence we cannot simply multiply their probabilities; in fact, we will see that if we remove all negations, then the resulting query is #P-hard; the algorithm described by (Dalvi and Suciu 2012) would immediately get stuck on this query. Instead, **Lift$^{\mathbf{R}}$** takes advantage of the negation, by first computing the prime implicate

$$\forall x\ \texttt{Tweets}(x) \vee \forall y\ \neg\texttt{Leader}(y),$$

which is a disconnected clause (the two literals use disjoint logical variables, x and y respectively). After applying distributivity we obtain:

$$\begin{aligned} Q &= (Q \wedge (\forall x\ \texttt{Tweets}(x))) \vee (Q \wedge (\forall y\ \neg\texttt{Leader}(y))) \\ &= Q_1 \vee Q_2 \end{aligned}$$

and **Lift$^{\mathbf{R}}$** applies the inclusion-exclusion formula:

$$\Pr(Q) = \Pr(Q_1) + \Pr(Q_2) - \Pr(Q_1 \wedge Q_2)$$

[1]To see this, rewrite the query as $(\texttt{Follows}(x,y) \Rightarrow \texttt{Tweets}(x)) \wedge (\texttt{Leader}(y) \Rightarrow \texttt{Follows}(x,y))$.

After simplifying the three queries, they become:

$$\begin{aligned} Q_1 &= \forall x \forall y\ (\texttt{Follows}(x,y) \vee \neg\texttt{Leader}(y)) \wedge \\ &\quad \forall x\ (\texttt{Tweets}(x)) \\ Q_2 &= \forall x \forall y\ (\texttt{Tweets}(x) \vee \neg\texttt{Follows}(x,y)) \wedge \\ &\quad \forall y\ (\neg\texttt{Leader}(y)) \\ Q_1 \wedge Q_2 &= \forall x\ (\texttt{Tweets}(x)) \wedge \forall y\ (\neg\texttt{Leader}(y)) \end{aligned}$$

The probability of Q_1 can now be obtained by multiplying the probabilities of its two clauses; same for the other two queries. As a consequence, our algorithm computes the probability $\Pr(Q)$ in polynomial time in the size of the domain and the probabilistic database.

If we remove all negations from Q and rename the predicates we get the following query:

$$h_1 \ = \ \forall x \forall y\ (R(x) \vee S(x,y)) \wedge (S(x,y) \vee T(y))$$

It was proven in (Dalvi and Suciu 2012) that computing the probability of the dual of h_1 is #P-hard in the size of the PDB. Thus, the query Q with negation is *easy*, while h_1 is hard, and **Lift$^{\mathbf{R}}$** takes advantage of this by applying resolution to find the disconnected prime implicate $\forall x \forall y\ \texttt{Tweets}(x) \vee \neg\texttt{Leader}(y)$.

A Dichotomy for Type-1 Queries

We prove a novel dichotomy for a subclass of CNF queries. First we review an earlier result, to put ours in perspective. In (Dalvi and Suciu 2012), an algorithm and dichotomy result for positive queries is shown. This result can be adapted to show that **Lift$^{\mathbf{R}}$** restricted to monotone (i.e., negation-free) queries admits a dichotomy, and the following theorem continues to hold.

Theorem 1 *If algorithm **Lift^{R}** FAILS on a Monotone CNF query Q, then computing* $\Pr(Q)$ *is #P-hard.*

However, the inclusion of negations in our query language increases significantly the difficulty of analyzing query complexities. Our major technical result extends Theorem 1 to a class of CNF queries with negation.

Define a *Type-1* query to be a CNF formula where each clause has at most two variables denoted x, y, and each atom is of one of the following three kinds:

- Unary symbols $R_1(x), R_2(x), R_3(x), \ldots$
- Binary symbols $S_1(x,y), S_2(x,y), \ldots$
- Unary symbols $T_1(y), T_2(y), \ldots$

or the negation of these symbols.

Our main result is:

Theorem 2 *For every Type-1 query Q, if algorithm **Lift^{R}** FAILS then computing* $\Pr(Q)$ *is #P-hard.*

The proof of this theorem required significant extension of the techniques used by (Dalvi and Suciu 2012) to prove Theorem 1. Details of our proof appear in (Gribkoff, Van den Broeck, and Suciu 2014).

Future Directions

Theorem 2 represents the first step towards proving the following conjecture: $\mathbf{Lift^R}$ provides a dichotomy for probabilistic queries on arbitrary universally-quantified CNF sentences where negation can be applied anywhere. Future work aims to prove this conjecture by expanding the existing dichotomy beyond Type-1 queries. We are also exploring new types of queries, and different assumptions about the probabilistic database, in particular symmetric probabilities, and how they affect our algorithm and dichotomy.

Acknowledgments

This work was partially supported by ONR grant #N00014-12-1-0423, NSF grants IIS-1115188 and IIS-1118122, and the Research Foundation-Flanders (FWO-Vlaanderen).

References

Dalvi, N., and Suciu, D. 2012. The dichotomy of probabilistic inference for unions of conjunctive queries. *Journal of the ACM (JACM)* 59(6):30.

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

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

Gribkoff, E.; Van den Broeck, G.; and Suciu, D. 2014. Understanding the Complexity of Lifted Inference and Asymmetric Weighted Model Counting. *ArXiv e-prints*. arXiv:1405.3250 [cs.AI].

Poole, D. 2003. First-order probabilistic inference. In *IJCAI*, volume 3, 985–991. Citeseer.

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

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

Van den Broeck, G.; Taghipour, N.; Meert, W.; Davis, J.; and De Raedt, L. 2011. Lifted probabilistic inference by first-order knowledge compilation. In *IJCAI*, 2178–2185.

Van den Broeck, G.; Meert, W.; and Darwiche, A. 2014. Skolemization for weighted first-order model counting. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR)*.

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