Understanding the Complexity of Lifted Inference and Asymmetric Weighted Model Counting

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

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

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

Abstract

In this paper we study lifted inference for the Weighted First-Order Model Counting problem (WFOMC), which counts the assignments that satisfy a given sentence in first-order logic (FOL); it has applications in Statistical Relational Learning (SRL) and Probabilistic Databases (PDB). We present several results. First, we describe a lifted inference algorithm that generalizes prior approaches in SRL and PDB. Second, we provide a novel dichotomy result for a non-trivial fragment of FO CNF sentences, showing that for each sentence the WFOMC problem is either in PTIME or #P-hard in the size of the input domain; we prove that, in the first case our algorithm solves the WFOMC problem in PTIME, and in the second case it fails. Third, we present several properties of the algorithm. Finally, we discuss limitations of lifted inference for symmetric probabilistic databases (where the weights of ground literals depend only on the relation name, and not on the constants of the domain), and prove the impossibility of a dichotomy result for the complexity of probabilistic inference for the entire language FOL.

1 INTRODUCTION

Weighted model counting (WMC) is a problem at the core of many reasoning tasks. It is based on the model counting or #SAT task (Gomes et al., 2009), where the goal is to count assignments that satisfy a given logical sentence. WMC generalizes model counting by assigning a weight to each assignment, and computing the *sum of their weights*. WMC has many applications in AI and its importance is increasing. Most notably, it underlies state-of-the-art probabilistic inference algorithms for Bayesian networks (Darwiche, 2002; Sang et al., 2005; Chavira and Darwiche, 2008), relational Bayesian networks (Chavira et al., 2006) and probabilistic programs (Fierens et al., 2011).

This paper is concerned with weighted *first-order* model counting (WFOMC), where we sum the weights of assignments that satisfy a sentence in finite-domain first-order logic. Again, this reasoning task underlies efficient algorithms for probabilistic reasoning, this time for popular representations in statistical relational learning (SRL) (Getoor and Taskar, 2007), such as Markov logic networks (Van den Broeck et al., 2011; Gogate and Domingos, 2011) and probabilistic logic programs (Van den Broeck et al., 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.

Early on, the disconnect between compact relational representations of uncertainty, and the intractability of inference at the ground, propositional level was noted, and efforts were made to exploit the relational structure for inference, using so-called *lifted inference* algorithms (Poole, 2003; Kersting, 2012). SRL and PDB algorithms for WFOMC all fall into this category. Despite these commonalities, there are also important differences. SRL has so far considered *symmetric* WFOMC problems, where relations of the same type are assumed to contribute equally to the probability of a world. This assumption holds for certain queries on SRL models, such as single marginals and partition functions, but fails for more complex conditional probability queries. These break lifted algorithms based on symmetric WFOMC (Van den Broeck and Darwiche, 2013). PDBs, on the other hand, have considered the *asymmetric* WFOMC setting from the start. While there are many semantics for PDBs, the most common models are tuple-independent PDBs, which assign each tuple a distinct probability, many tuples have probability zero, and no symmetries can be expected. However, current asymmetric WFOMC algorithms (Dalvi and Suciu, 2012) suffer from a major limitation of

their own, in that they can only count models of sentences in monotone disjunctive normal form (MDNF) (i.e., DNF without negation). Such sentences represent unions of conjunctive database queries (UCQ). WFOMC encodings of SRL models almost always fall outside this class.

The present work seeks to upgrade a well-known PDB algorithm for asymmetric WFOMC (Dalvi and Suciu, 2012) to the SRL setting, by enabling it to count models of arbitrary sentences in conjunctive normal form (CNF). This permits its use for lifted SRL inference with arbitrary soft or hard evidence, or equivalently, probabilistic database queries with negation. Our first contribution is this algorithm, which we call **Lift$^{\mathbf{R}}$**, and is presented in Section 3.

Although **Lift$^{\mathbf{R}}$** has clear practical merits, we are in fact motivated by fundamental theoretical questions. In the PDB setting, our algorithm is known to come with a sharp complexity guarantee, called the *dichotomy theorem* (Dalvi and Suciu, 2012). By only looking at the structure of the first-order sentence (i.e., the database query), the algorithm reports failure when the problem is #P-hard (in terms of data complexity), and otherwise guarantees to solve it in time polynomial in the domain (i.e., database) size. It can thus precisely classify MDNF sentences as being tractable or intractable for asymmetric WFOMC. Whereas several complexity results for symmetric WFOMC exist (Van den Broeck, 2011; Jaeger and Van den Broeck, 2012), the complexity of asymmetric WFOMC for SRL queries with evidence is still poorly understood. Our second and main contribution, presented in Section 4, is a *novel dichotomy result* over a small but non-trivial fragment of CNFs. We completely classify this class of problems as either computable in polynomial time or #P-hard. This represents a first step towards proving the following conjecture: **Lift$^{\mathbf{R}}$** provides a dichotomy for asymmetric WFOMC on arbitrary CNF sentences, and therefore perfectly classifies all related SRL models as tractable or intractable for conditional queries.

As our third contribution, presented in Section 5, we illustrate the algorithm with examples that show its application to common probabilistic models. We discuss the capabilities of **Lift$^{\mathbf{R}}$** that are not present in other lifted inference techniques.

As our fourth and final contribution, in Section 6, we discuss extensions of our algorithm to symmetric WFOMC, but also show the impossibility of a dichotomy result for arbitrary first-order logic sentences.

2 BACKGROUND

We begin by introducing the necessary background on relational logic and weighted model counting.

2.1 RELATIONAL LOGIC

Throughout this paper, we will work with the relational 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* or *logical variables* $\{x, y, \ldots\}$. A *literal* is an atom or its negation. A *formula* combines atoms with logical connectives and quantifiers $\exists$ and $\forall$. A *substitution* $[a/x]$ replaces all occurrences of x by a. Its application to formula F is denoted $F[a/x]$. A formula is a sentence if each logical variable x is enclosed by a $\forall x$ or $\exists x$. A formula is *ground* if it contains no logical variables. A *clause* is a universally quantified disjunction of literals. A *term* is an existentially quantified conjunction of literals. A *CNF* is a conjunction of clauses, and a *DNF* is a disjunction of terms. A *monotone* CNF or DNF contains no negation symbols. As usual, we drop the universal quantifiers from the CNF syntax.

The semantics of sentences are defined in the usual way (Hinrichs and Genesereth, 2006). An interpretation, or world, I that satisfies sentence Δ is denoted by $I \vDash \Delta$, and represented as a set of literals. Our algorithm checks properties of sentences that are undecidable in general FOL, but decidable, with the following complexity, in the CNF fragment we investigate.

Theorem 2.1. *(Sagiv and Yannakakis, 1980) (Farré et al., 2006) Checking whether logical implication $Q \Rightarrow Q'$ or equivalence $Q \equiv Q'$ holds between two CNF sentences is Π_2^p-complete.*

2.2 WEIGHTED MODEL COUNTING

Weighted model counting was introduced as a propositional reasoning problem.

Definition 2.2 (WMC)**.** *Given a propositional* sentence Δ *over literals $\mathcal{L}$, and a* weight function $w : \mathcal{L} \to \mathbb{R}^{\geq 0}$*, the* weighted model count *(WMC) is*

$$\mathrm{WMC}(\Delta, w) = \sum_{I \vDash \Delta} \prod_{\ell \in I} w(\ell).$$

We will consider its generalization to *weighted first-order model counting* (WFOMC), where Δ is now a sentence in relational logic, and $\mathcal{L}$ consists of all ground first-order literals for a given domain of constants.

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

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

and the UCQ (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 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 $w(\texttt{Prof(Anne)})$ to be different from $w(\texttt{Prof(Charlie)})$. 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 et al., 2014) for the details, and show here the following example MLN.

$$2 \quad \texttt{Prof}(x) \wedge \texttt{Advises}(x, y) \Rightarrow \texttt{Student}(y)$$

It states that the probability of a world increases by a factor e^2 with every pair of people x, y for which the formula holds. Its WFOMC encoding has Δ equal to

$$\forall x, \forall y, \texttt{F}(x, y) \Leftrightarrow \\ [\texttt{Prof}(x) \wedge \texttt{Advises}(x, y) \Rightarrow \texttt{Student}(y)]$$

and weight functions $w^\star, \bar{w}^\star$ such that $w^\star(\texttt{F}) = e^2$ and all other predicates map to 1.

Answering an SRL query Q given evidence E, that is, $\Pr(Q \mid E)$, using a symmetric WFOMC encoding, generally requires solving two WFOMC tasks:

$$\Pr(Q \mid E) = \frac{\text{WFOMC}(Q \wedge E \wedge \Delta, w)}{\text{WFOMC}(E \wedge \Delta, w)}$$

Symmetric WFOMC problems are strictly more tractable than asymmetric ones. We postpone the discussion of this observation to Section 5, but already note that all theories Δ with up to two logical variables per formula support *domain-lifted* inference (Van den Broeck, 2011), which means that any WFOMC query runs in time polynomial in the domain size (i.e, number of constants). For conditional probability queries, even though fixed-parameter complexity bounds exist that use symmetric WFOMC (Van den Broeck and Darwiche, 2013), the actual underlying reasoning task is asymmetric WFOMC, whose complexity we investigate for the first time.

Finally, we make three simplifying observations. First, SRL query Q and evidence E typically assign values to random variables. This means that the query and evidence can be absorbed into the asymmetric weight function, by setting the weight of literals disagreeing with Q or E to zero. We hence compute:

$$\Pr(Q \mid E) = \frac{\text{WFOMC}(\Delta, w_{Q \wedge E})}{\text{WFOMC}(\Delta, w_E)}$$

This means that our complexity analysis for a given encoding Δ applies to both numerator and denominator for arbitrary Q and E, and that polytime WFOMC for Δ implies polytime $\Pr(Q \mid E)$ computation. The converse is not true, since it is possible that both WFOMC calls are #P-hard, but their ratio is in PTIME. Second, we will from now on assume that Δ is in CNF. The WFOMC encoding of many SRL formalisms is already in CNF, or can be reduced to it (Van den Broeck et al., 2014). For PDB queries that are in monotone DNF, we can simply compute $\Pr(Q) = 1 - \Pr(\neg Q)$, which reduces to WFOMC on a CNF. Moreover, by adjusting the probabilities in the PDB, this CNF can also be made monotone. Third, we will assume that $w(\ell) = 1 - w(\neg\ell)$, which can always be achieved by normalizing the weights.

Under these assumptions, we can simply refer to $\text{WFOMC}(Q, w)$ as $\Pr(Q)$, to Q as the CNF query, to $w(\ell)$ as the probability $\Pr(\ell)$, and to the entire weight function w as the PDB. This is in agreement with notation in the PDB literature.

3 ALGORITHM Lift$^{\mathbf{R}}$

We present here the lifted algorithm **Lift$^{\mathbf{R}}$** (pronounced *lift-ER*), which, given a CNF formula Q computes $\Pr(Q)$ in polynomial time in the size of the PDB, or fails. In the next section we provide some evidence for its completeness: under certain assumptions, if **Lift$^{\mathbf{R}}$** fails on formula Q, then computing $\Pr(Q)$ is #P-hard in the PDB size.

3.1 DEFINITIONS

An *implicate* of Q is some clause C s.t. the logical implication $Q \Rightarrow C$ holds. C is a *prime implicate* if there is no other implicate C' s.t. $C' \Rightarrow C$.

A *connected component* of a clause C is a minimal subset of its atoms that have no logical variables in common with the rest of the clause. If some prime implicate C has more than one connected component, then we can write it as:

$$C = D_1 \vee D_2 \vee \cdots \vee D_m$$

where each D_i is a clause with distinct variables. Applying distributivity, we write Q in *union-CNF* form:

$$Q = Q_1 \vee Q_2 \vee \cdots \vee Q_m$$

where each Q_i is a CNF with distinct variables.

We check for disconnected prime implicates $D_1 \vee D_2$ where both D_1 and D_2 subsume some clause of Q. Intuitively, this means that when we apply inclusion/exclusion to the union-CNF, the resulting queries are simpler. The search

for D_1, D_2 can proceed using some standard inference algorithm, e.g. resolution. By Theorem 2.1, this problem is Π_2^p-complete in the size of the query Q, but independent of the PDB size.

A set of *separator variables* for a query $Q = \bigwedge_{i=1}^{k} C_i$ is a set of variables $x_i, i = 1, k$ such that, (a) for each clause C_i, x_i occurs in all atoms of C_i, and (b) any two atoms (not necessarily in the same clause) referring to the same relation R have their separator variable on the same position.

3.2 PREPROCESSING

We start by transforming Q (and PDB) such that:

1. No constants occur in Q.
2. If all the variables in Q are $x_1, x_2, \ldots, x_k$, then every relational atom in Q (positive or negated) is of the form $R(x_{i_1}, x_{i_2}, \ldots)$ such that $i_1 < i_2 < \ldots$

Condition (1) can be enforced by *shattering* Q w.r.t. its variables. Condition (2) can be enforced by modifying both the query Q and the database, in a process called *ranking* and described in the appendix. Here, we illustrate ranking on an example. Consider the query:

$$Q = (R(x,y) \vee S(x,y)) \wedge (\neg R(x,y) \vee \neg S(y,x))$$

Define $R_1(x,y) \equiv R(x,y) \wedge (x < y)$; $R_2(x) \equiv R(x,x)$; $R_3(y,x) \equiv R(x,y) \wedge (x > y)$. Define similarly S_1, S_2, S_3. Given a PDB with relations R, S, we define a new PDB$'$ over the six relations by setting $\Pr(R_1(a,b)) = \Pr(R(a,b))$ when $a < b$, $\Pr(R_1(a,b)) = 0$ when $a > b$, $\Pr(R_2(a)) = \Pr(R(a,a))$, etc. Then, the query Q over PDB is equivalent to the following query over PDB$'$:

$$(R_1(x,y) \vee S_1(x,y)) \wedge (\neg R_1(x,y) \vee \neg S_3(x,y)) \wedge$$
$$(R_2(x) \vee S_2(x)) \wedge (\neg R_2(x) \vee \neg S_2(x)) \wedge$$
$$(R_3(x,y) \vee S_3(x,y)) \wedge (\neg R_3(x,y) \vee \neg S_1(x,y))$$

3.3 ALGORITHM DESCRIPTION

Algorithm **Lift$^{\mathbf{R}}$**, given in Figure 1, proceeds recursively on the structure of the CNF query Q. When it reaches ground atoms, it simply looks up their probabilities in the PDB. Otherwise, it performs the following sequence of steps.

First, it tries to express Q as a union-CNF. If it succeeds, and if the union can be partitioned into two sets that do not share any relational symbols, $Q = Q_1 \vee Q_2$, then it applies a *Decomposable Disjunction*:

$$\Pr(Q) = 1 - (1 - \Pr(Q_1)(1 - \Pr(Q_2))$$

Otherwise, it applies the *Inclusion/Exclusion* formula:

$$\Pr(Q) = -\sum_{s \subseteq [m]} (-1)^{|s|} \Pr(\bigwedge_{i \in s} Q_i)$$

However, before computing the recursive probabilities, our algorithm first checks for equivalent expressions, i.e. it checks for terms s_1, s_2 in the inclusion/exclusion formula such that $\bigwedge_{i \in s_1} Q_i \equiv \bigwedge_{i \in s_2} Q_i$: in that case, these terms either cancel out, or add up (and need be computed only once). We show in Section 5.4 the critical role that the cancellation step plays for the completeness of the algorithm. To check cancellations, the algorithm needs to check for equivalent CNF expressions. This can be done using some standard inference algorithm (recall from Theorem 2.1 that this problem is Π_2^p-complete in the size of the CNF expression).

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(ℓ) in PDB
2  Step 1: Write Q as a union-CNF: Q = Q1 ∨ Q2 ∨ ··· ∨ Qm
3  Step 2: If m > 1 and Q can be partitioned into two
        sets Q = Q' ∨ Q'' with disjoint relation symbols,
        return 1 − (1 − Pr(Q1)) · (1 − Pr(Q2))
4       /* Decomposable Disjunction */
5  Step 3: If Q cannot be partitioned, return
        Σ_{s⊆[m]} (−1)^{|s|} Pr(∧_{i∈s} Qi)
6       /* Inclusion/Exclusion – perform cancellations
             before recursion */
7  Step 4: Write Q in CNF: Q = C1 ∧ C2 ∧ ··· ∧ Ck
8  Step 5: If k > 1, and Q can be partitioned into two
        sets Q = Q' ∧ Q'' with disjoint relation symbols,
        return Pr(Q1) · Pr(Q2)
9       /* Decomposable Conjunction */
10 Step 6: If Q has a separator variable, return
        Π_{a∈D} Pr(C1[a/x1] ∧ ··· ∧ Ck[a/xk])
11      /* Decomposable Universal Quantifier */
12 Otherwise FAIL
```

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

Finally, if neither of the above steps apply, then the algorithm checks if Q can be partitioned into two sets of clauses that do not share any common relation symbols. In that case, $Q = Q' \wedge Q''$, and its probability is computed using a *Decomposable Conjunction*:

$$\Pr(Q) = \Pr(Q') \cdot \Pr(Q'')$$

Finally, if none of the above cases apply to the CNF query $Q = C_1 \wedge C_2 \wedge \cdots \wedge C_k$, then the algorithm tries to find a set of separator variables $x_1, \ldots, x_k$ (one for each clause). If it finds them, then the probability is given by a *Decomposable Universal Quantifier*:

$$\Pr(Q) = \prod_{a \in \text{Domain}} \Pr(C_1[a/x_1] \wedge \cdots \wedge C_k[a/x_k])$$

We prove our first main result:

Theorem 3.1. *One of the following holds: (1) either **Lift^{R}** fails on Q, or (2) for any domain size n and a PDB consisting of probabilities for the ground tuples, **Lift^{R}** computes*

$\Pr(Q)$ *in polynomial time in* n.

Proof. (Sketch) The only step of the algorithm that depends on the domain size n is the decomposable universal quantifier step; this also reduces by 1 the arity of every relation symbol, since it substitutes it by the same constant a. Therefore, the algorithm runs in time $O(n^k)$, where k is the largest arity of any relation symbol. We note that the constant behind $O(\cdots)$ may be exponential in the size of the query Q. □

4 MAIN COMPLEXITY RESULT

In this section we describe our main technical result of the paper: that the algorithm is complete when restricted to a certain class of CNF queries.

We first review a prior result, to put ours in perspective. (Dalvi and Suciu, 2012) define an algorithm for Monotone DNF (called Unions Of Conjunctive Queries), which can be adapted to Monotone CNF; that adaptation is equivalent to **Lift$^{\mathbf{R}}$** restricted to Monotone CNF queries. (Dalvi and Suciu, 2012) prove:

Theorem 4.1. *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 of the paper extends Theorem 4.1 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 4.2. *For every Type-1 query Q, if algorithm **Lift^{R}** FAILS then computing* $\Pr(Q)$ *is #P-hard.*

The proof is a significant extension of the techniques used by (Dalvi and Suciu, 2012) to prove Theorem 4.1; we give a proof sketch in Section 7 and include the full proof in the appendix.

5 PROPERTIES OF Lift$^{\mathbf{R}}$

We now describe several properties of **Lift$^{\mathbf{R}}$**, and the relationship to other lifted inference formalisms.

5.1 NEGATIONS CAN LOWER THE COMPLEXITY

The presence of negations can lower a query's complexity, and our algorithm exploits this. To see this, consider the following query

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

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 atoms in the domain. We note that the two clauses are dependent (since both refer to the relation `Follow`), 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:

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

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

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

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

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

After simplifying the three queries, they become:

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

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 PDB.

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

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

(Dalvi and Suciu, 2012) proved that computing the probability 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 our algorithm takes advantage of this by applying resolution.

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

5.2 ASYMMETRIC WEIGHTS CAN INCREASE THE COMPLEXITY

(Van den Broeck, 2011) has proven that any query with at most two logical variables per clause is domain-liftable. Recall that this means that one can compute its probability in PTIME in the size of the domain, in the symmetric case, when all tuples in a relation have the same probability. However, queries with at most two logical variables per clause can become #P-hard when computed over asymmetric probabilities, as witnessed by the query h_1 above.

5.3 COMPARISON WITH PRIOR LIFTED FO-CIRCUITS

(Van den Broeck et al., 2011; Van den Broeck, 2013) introduce FO d-DNNF circuits, to compute symmetric WFOMC problems. An FO d-DNNF is a circuit whose nodes are one of the following: decomposable conjunction ($Q_1 \wedge Q_2$ where Q_1, Q_2 do not share any common predicate symbols), deterministic-disjunction ($Q_1 \vee Q_2$ where $Q_1 \wedge Q_2 \equiv \texttt{false}$), inclusion-exclusion, decomposable universal quantifier (a type of $\forall x, Q(x)$), and deterministic automorphic existential quantifier. The latter is an operation that is specific only to structures with symmetric weights, and therefore does not apply to our setting. We prove that our algorithm can compute all formulas that admit an FO d-DNNF circuit.

Fact 5.1. *If Q admits an FO d-DNNF without a deterministic automorphic existential quantifier, then* ***Lift$^{\mathbf{R}}$*** *computes $\Pr(Q)$ in PTIME in the size of the PDB.*

The proof is immediate by noting that all other node types in the FO d-DNNF have a corresponding step in **Lift$^{\mathbf{R}}$**, except for deterministic disjunction, which our algorithm computes using inclusion-exclusion: $\Pr(Q_1 \vee Q_2) = \Pr(Q_1) + \Pr(Q_2) - \Pr(Q_1 \wedge Q_2) = \Pr(Q_1) + \Pr(Q_2)$ because $Q_1 \wedge Q_2 \equiv \texttt{false}$.

5.4 CANCELLATIONS IN INCLUSION/EXCLUSION

We now look at a more complex query. First, let us denote four simple queries:

$$q_0 = (R(x_0) \vee S_1(x_0, y_0))$$
$$q_1 = (S_1(x_1, y_1) \vee S_2(x_1, y_1))$$
$$q_2 = (S_2(x_2, y_2) \vee S_3(x_2, y_2))$$
$$q_3 = (S_3(x_3, y_3) \vee T(y_3))$$

(Dalvi and Suciu, 2012) proved that their conjunction, i.e. the query $h_3 = q_0 \wedge q_1 \wedge q_2 \wedge q_3$, is #P-hard in data complexity. Instead of h_3, consider:

$$Q_W = (q_0 \vee q_1) \wedge (q_0 \vee q_3) \wedge (q_2 \vee q_3)$$

There are three clauses sharing relation symbols, hence we cannot apply a decomposable conjunction. However, each clause is disconnected, for example q_0 and q_1 do not share logical variables, and we can thus write Q_W as a disjunction. After removing redundant terms:

$$Q_W = (q_0 \wedge q_2) \vee (q_0 \wedge q_3) \vee (q_1 \wedge q_3)$$

Our algorithm applies the inclusion/exclusion formula:

$$\begin{aligned}\Pr(Q_W) = {} & \Pr(q_0 \wedge q_2) + \Pr(q_0 \wedge q_3) + \Pr(q_1 \wedge q_3) \\ & - \Pr(q_0 \wedge q_2 \wedge q_3) - \Pr(q_0 \wedge q_1 \wedge q_3) - \Pr(q_0 \wedge \cdots \wedge q_3) \\ & + \Pr(q_0 \wedge \cdots \wedge q_3)\end{aligned}$$

At this point our algorithm performs an important step: it cancels out the last two terms of the inclusion/exclusion formula. Without this key step, no algorithm could compute the query in PTIME, because the last two terms are precisely h_3, which is #P-hard. To perform the cancellation the algorithm needs to first check which FOL formulas are equivalent, which, as we have seen, is decidable for our language (Theorem 2.1). Once the equivalent formulas are detected, the resulting expressions can be organized in a lattice, as shown in Figure 2, and the coefficient of each term in the inclusion-exclusion formula is precisely the lattice's Möbius function (Stanley, 1997).

6 EXTENSIONS AND LIMITATIONS

We describe here an extension of **Lift$^{\mathbf{R}}$** to symmetric WFOMC, and also prove that a complete characterization of the complexity of all FOL queries is impossible.

6.1 SYMMETRIC WFOMC

Many applications of SRL require weighted model counting for FOL formulas over PDBs where the probabilities are associated to relations rather than individual tuples. That is, $\texttt{Friend}(a, b)$ has the same probability, independently of the constants a, b in the domain. In that symmetric WFOMC case, the model has a large number of symmetries (since the probabilities are invariant under permutations of constants), and lifted inference algorithms may further exploit these symmetries. (Van den Broeck, 2013) employ one operator that is specific to symmetric probabilities, called *atom counting*, which is applied to a unary predicate $R(x)$ and iterates over all possible worlds of that predicate. Although there are 2^n possible worlds for R, by conditioning on any world, the probability will depend only on the cardinality k of R, because of the symmetries. Therefore, the system iterates over $k = 0, n$, and adds the conditional probabilities multiplied by $\binom{n}{k}$. For example, consider the following query:

$$H = (\neg R(x) \vee S(x, y) \vee \neg T(y)) \quad (1)$$

Computing the probabilities of this query is #P-hard (Theorem 4.2). However, if all tuples $R(a)$ have the same probability $r \in [0, 1]$, and similarly tuples in S, T have proba-

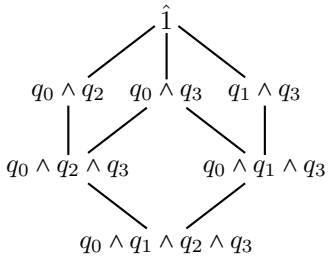

Figure 2: Lattice for Q_w. The bottom query is #P-hard, yet all terms in the inclusion/exclusion formula that contain this term cancel out, and $\Pr(Q_W)$ is computable in PTIME.

bilities s, t, then one can check that[2]

$$\Pr(H) = \sum_{k,l=0,n} r^k \cdot (1-r)^{n-k} \cdot t^l \cdot (1-t)^{n-l} \cdot (1-s^{kl})$$

Denote Sym-**Lift**$^{\mathbf{R}}$ the extension of **Lift**$^{\mathbf{R}}$ with a deterministic automorphic existential quantifier operator. The question is whether this algorithm is complete for computing the probabilities of queries over PDBs with symmetric probabilities. Folklore belief was that this existential quantifier operator was the only operator required to exploit the extra symmetries available in PDBs with symmetric probabilities. For example, all queries in (Van den Broeck et al., 2011) that can be computed in PTIME over symmetric PDBs have the property that, if one removes all unary predicates from the query, then the residual query can be computed in PTIME over asymmetric PDBs.

We answer this question in the negative. Consider the following query:

$$Q = (S(x_1, y_1) \vee \neg S(x_1, y_2) \vee \neg S(x_2, y_1) \vee S(x_2, y_2))$$

Here, we interpret $S(x, y)$ as a *typed relation*, where the values x and y are from two disjoint domains, of sizes n_1, n_2 respectively, in other words, $S \subseteq [n_1] \times [n_2]$.

Theorem 6.1. *We have that*

- $\Pr(Q)$ *can be computed in time polynomial in the size of a symmetric PDB with probability* p *as* $\Pr(Q) = f(n_1, n_2) + g(n_1, n_2)$ *where:*

$$f(0, n_2) = 1 \quad f(n_1, n_2) = \sum_{k=1}^{n_1} p^{kn_2} g(n_1 - k, n_2)$$

$$g(n_1, 0) = 1 \quad g(n_1, n_2) = \sum_{\ell=1}^{n_2} (1-p)^{n_1 \ell} f(n_1, n_2 - \ell)$$

- *Sym-**Lift**$^{\mathbf{R}}$ fails to compute* Q.

The theorem shows that new operators will be required for symmetric WFOMC. We note that it is currently open whether computing $\Pr(Q)$ is #P-hard in the case of asymmetric WFOMC.

[2]Conditioned on $|R| = k$ and $|T| = l$, the query is true if S contains at least one pair $(a, b) \in R \times T$.

Proof. Denote D_x, D_y the domains of the variables x and y. Fix a relation $S \subseteq D_1 \times D_2$. We will denote $a_1, a_2, \ldots \in D_1$ elements from the domain of the variable x, and $b_1, b_2, \ldots \in D_2$ elements from the domain of the variable y. For any a, b, define $a \prec b$ if $(a, b) \in S$, and $a \succ b$ if $(a, b) \notin S$; in the latter case we also write $b \prec a$. Then, (1) for any a, b, either $a \prec b$ or $b \prec a$, (2) $\prec$ is a partial order on the disjoint union of the domains D_1 and D_2 iff S satisfies the query Q. The first property is immediate. To prove the second property, notice that Q states that there is no cycle of length 4: $x_1 \prec y_2 \prec x_2 \prec y_1 \prec x_1$. By repeatedly applying resolution between Q with itself, we derive that there are no cycles of length 6, 8, 10, etc. Therefore, $\prec$ is transitive, hence a partial order. Any finite, partially ordered set has a minimal element, i.e. there exists z s.t. $\forall x$, $x \not\prec z$. Let Z be the set of all minimal elements, and denote $X = D_1 \cap Z$ and $Y = D_2 \cap Z$. Then exactly one of X or Y is non-empty, because if both were non-empty then, for $a \in X$ and $b \in Y$ we have either $a \prec b$ or $a \succ b$ contradicting their minimality. Assuming $X \neq \emptyset$, we have (a) for all $a \in X$ and $b \in D_2$, $(a, b) \in S$, and (b) Q is true on the relation $S' = (D_1 - X) \times D_2$. This justifies the recurrence formula for $\Pr(Q)$. □

6.2 THE COMPLEXITY OF ARBITRARY FOL QUERIES

We conjecture that, over asymmetric probabilities (asymmetric WFOMC), our algorithm is complete, in the sense that whenever it fails on a query, then the query is provably #P-hard. Notice that **Lift**$^{\mathbf{R}}$ applies only to a fragment of FOL, namely to CNF formulas without function symbols, and where all variables are universally quantified. We present here an impossibility result showing that a complete algorithm cannot exist for general FOL queries. We use for that a classic result by Trakhtenbrot (Libkin, 2004):

Theorem 6.2 (Finite satisfiability)**.** *The problem: "given a FOL sentence* Φ*, check whether there exists a finite model for* Φ*" is undecidable.*

From here we obtain:

Theorem 6.3. *There exists no algorithm that, given any FOL sentence* Q *checks whether* $\Pr(Q)$ *can be computed*

in PTIME in the asymmetric PDB size.

Proof. By reduction from the finite satisfiability problem. Fix the hard query H in Eq.(1), for which the counting problem is #P-hard. Recall that H uses the symbols R, S, T. Let Φ be any formula over a disjoint relational vocabulary (i.e. it doesn't use R, S, T). We will construct a formula Q, such that computing $\Pr(Q)$ is in PTIME iff Φ is unsatisfiable in the finite: this proves the theorem. To construct Q, first we modify Φ as follows. Let $P(x)$ be another fresh, unary relational symbol. Rewrite Φ into Φ' as follows: replacing every $(\exists x.\Gamma)$ with $(\exists x.P(x) \wedge \Gamma)$ and every $(\forall x.\Gamma)$ with $(\forall x.P(x) \Rightarrow \Gamma)$ (this is not equivalent to the guarded fragment of FOL); leave the rest of the formula unchanged. Intuitively, Φ' checks if Φ is true on the substructure defined by the domain elements that satisfy P. More precisely: for any database instance I, Φ' is true on I iff Φ is true on the substructure of I defined by the domain elements that satisfy $P(x)$. Define the query $Q = (H \wedge \Phi')$. We now prove the claim.

If Φ is unsatisfiable then so is Φ', and therefore $Pr(Q) = 0$ is trivially computable in PTIME.

If Φ is satisfiable, then fix any deterministic database instance I that satisfies Φ; notice that I is deterministic, and $I \models \Phi$. Let J be any probabilistic instance over the vocabulary for H over a domain disjoint from I. Define $P(x)$ as follows: $P(a)$ is true for all domain elements $a \in I$, and $P(b)$ is false for all domain elements $b \in J$. Consider now the probabilistic database $I \cup J$. (Thus, $P(x)$ is also deterministic, and selects the substructure I from $I \cup J$; therefore, Φ' is true in $I \cup J$.) We have $Pr(Q) = Pr(H \wedge \Phi') = Pr(H)$, because Φ' is true on $I \cup J$. Therefore, computing $Pr(Q)$ is #P-hard. Notice the role of P: while I satisfies Φ, it is not necessarily the case that $I \cup J$ satisfies Φ. However, by our construction we have ensured that $I \cup J$ satisfies Φ'. □

7 PROOF OF THEOREM 4.2

The proof of Theorem 4.2 is based on a reduction from the #PP2-CNF problem, which is defined as follows. Given two disjoint sets of Boolean variables $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_n$ and a bipartite graph $E \subseteq [n] \times [n]$, count the number of satisfying truth assignments $\#\Phi$ to the formula: $\Phi = \bigwedge_{(i,j)\in E}(X_i \vee Y_j)$. (Provan and Ball, 1983) have shown that this problem is #P-hard.

More precisely, we prove the following: given any Type-1 query Q on which the algorithm **Lift$^{\mathbf{R}}$** fails, we can reduce the #PP2-CNF problem to computing $\Pr(Q)$ on a PDB with domain size n. The reduction consists of a combinatorial part (the construction of certain gadgets), and an algebraic part, which makes novel use of the concepts of algebraic independence (Yu, 1995) and annihilating polynomials (Kayal, 2009). We include the latter in the appendix, and only illustrate here the former on a particular query of Type-1.

We illustrate the combinatorial part of the proof on the following query Q:

$$(R(x) \vee \neg S(x,y) \vee T(y)) \wedge (\neg R(x) \vee S(x,y) \vee \neg T(y))$$

To reduce Φ to the problem of computing $\Pr(Q)$, we construct a structure with unary predicates R and T and binary predicate S, with active domain $[n]$.

We define the tuple probabilities as follows. Letting $x, y, a, b \in (0,1)$ be four numbers that will be specified later, we define:

$$\Pr(R(i)) = x$$
$$\Pr(T(j)) = y$$
$$\Pr(S(i,j)) = \begin{cases} a & \text{if } (i,j) \in E \\ b & \text{if } (i,j) \notin E \end{cases}$$

Note this PDB does not have symmetric probabilities: in fact, over structures with symmetric probabilities one can compute $\Pr(Q)$ in PTIME.

Let θ denote a valuation of the variables in Φ. Let E_θ denote the event $\forall i.(R(i) = \texttt{true iff } \theta(X_i) = \texttt{true})$ $\wedge\, \forall j.(T(j) = \texttt{true iff } \theta(Y_j) = \texttt{true})$.

E_θ completely fixes the unary predicates R and T and leaves S unspecified. Given E_θ, each Boolean variable corresponding to some $S(x,y)$ is now independent of every other $S(x',y')$. In general, given an assignment of $R(i)$ and $T(j)$, we examine the four formulas that define the probability that the query is true on (i,j): $F_1 = Q[R(i) = 0, T(j) = 0]$, $F_2 = Q[R(i) = 0, T(j) = 1]$, $F_3 = Q[R(i) = 1, T(j) = 0]$, $F_4 = Q[R(i) = 1, T(j) = 1]$.

For Q, F_1, F_2, F_3, F_4 are as follows:

$$F_1 = \neg S(i,j) \qquad F_2 = F_3 = \texttt{true} \qquad F_4 = S(i,j)$$

Denote f_1, f_2, f_3, f_4 the arithmetization of these Boolean formulas:

$$f_1 = \begin{cases} 1-a & \text{if } (i,j) \in E \\ 1-b & \text{if } (i,j) \notin E \end{cases}$$
$$f_4 = \begin{cases} a & \text{if } (i,j) \in E \\ b & \text{if } (i,j) \notin E \end{cases}$$

Note that $f_2 = f_3 = 1$ and do not change $\Pr(Q)$.

Define the parameters k, l, p, q of E_θ as k = number of i's s.t. $R(i) = \texttt{true}$, l = number of j's s.t. $T(j) = \texttt{true}$, p = number of $(i,j) \in E$ s.t. $R(i) = T(j) = \texttt{true}$, q = number of $(i,j) \in E$ s.t. $R(i) = T(j) = \texttt{false}$.

Let $N(k,l,p,q)$ = the number of θ's that have parameters k, l, p, q. If we knew all $(n+1)^2(m+1)^2$ values of $N(k,l,p,q)$, we could recover $\#\Phi$ by summing over $N(k,l,p,q)$ where $q = 0$. That is, $\#\Phi = \sum_{k,l,p} N(k,l,p,0)$.

We now describe how to solve for $N(k,l,p,q)$, completing the hardness proof for $\Pr(Q)$.

We have $\Pr(E_\theta) = x^k(1-x)^{n-k}y^l(1-y)^{n-l}$ and $\Pr(Q|E_\theta) = a^p(1-a)^q b^{kl-p}(1-b)^{(n-k)(n-l)-q}$. Combined, these give the following expression for $\Pr(Q)$:

$$\begin{aligned}\Pr(Q) &= \sum_\theta \Pr(Q|E_\theta)\Pr(E_\theta) \\ &= (1-b)^{n^2}(1-x)^n(1-y)^n \sum_{k,l,p,q} T \qquad (1)\end{aligned}$$

where:

$$\begin{aligned}T =& N(k,l,p,q) * (a/b)^p[(1-a)/(1-b)]^q \\ & [x/(1-b)^n(1-x)]^k[y/(1-b)^n \\ & (1-y)]^l[b(1-b)]^{kl} \\ =& N(k,l,p,q) * A^p B^q X^k Y^l C^{kl} \qquad (2)\end{aligned}$$

Equations (1) and (2) express $\Pr(Q)$ as a polynomial in X, Y, A, B, C with unknown coefficients $N(k,l,p,q)$. Our reduction is the following: we choose $(n+1)^2(m+1)^2$ values for the four parameters $x, y, a, b \in (0,1)$, consult an oracle for $\Pr(Q)$ for these settings of the parameters, then solve a linear system of $(n+1)^2(m+1)^2$ equations in the unknowns $N(k,l,p,q)$. The crux of the proof consists of showing that the matrix of the system is non-singular: this is far from trivial, in fact had we started from a PTIME query Q then the system *would* be singular. Our proof consists of two steps (1) prove that we can choose X, Y, A, B independently, in other words that the mapping $(x,y,a,b) \mapsto (X,Y,A,B)$ is locally invertible (has a non-zero Jacobian), and (2) prove that there exists a choice of $(n+1)^2(m+1)^2$ values for (X,Y,A,B) such that the matrix of the system is non-singular: then, by (1) it follows that we can find $(n+1)^2(m+1)^2$ values for (x,y,a,b) that make the matrix non-singular, completing the proof. For our particular example, Part (1) can be verified by direct computations (see Section A.3); for general queries this requires Section A.12. Part (2) for this query is almost as general as for any query and we show it in Section A.2.

8 RELATED WORK

The algorithm and complexity results of (Dalvi and Suciu, 2012), which apply to positive queries, served as the starting point for our investigation of asymmetric WFOMC with negation. See (Suciu et al., 2011) for more background on their work. The tuple-independence assumption of PDBs presents a natural method for modeling asymmetric WFOMC. Existing approaches for PDBs can express complicated correlations (Jha et al., 2010; Jha and Suciu, 2012) but only consider queries without negation.

Close in spirit to the goals of our work are (Van den Broeck, 2011) and (Jaeger and Van den Broeck, 2012). They introduce a formal definition of lifted inference and describe a powerful knowledge compilation technique for WFOMC. Their completeness results for first-order knowledge compilation on a variety of query classes motivate our exploration of the complexity of lifted inference. (Cozman and Polastro, 2009) analyze the complexity of probabilistic description logics.

Other investigations of evidence in lifted inference include (Van den Broeck and Davis, 2012), who allow arbitrary hard evidence on unary relations, (Bui et al., 2012), who allow asymmetric soft evidence on a single unary relation, and (Van den Broeck and Darwiche, 2013), who allow evidence of bounded Boolean rank. Our model allows entirely asymmetric probabilities and evidence.

9 CONCLUSION

Our first contribution is the algorithm **Lift$^{\mathbf{R}}$** for counting models of arbitrary CNF sentences over asymmetric probabilistic structures. Second, we prove a novel dichotomy result that completely classifies a subclass of CNFs as either PTIME or #P-hard. Third, we describe capabilities of **Lift$^{\mathbf{R}}$** not present in prior lifted inference techniques. Our final contribution is an extension of our algorithm to symmetric WFOMC and a discussion of the impossibility of establishing a dichotomy for all first-order logic sentences.

Acknowledgements

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

Hung B Bui, Tuyen N Huynh, and Rodrigo de Salvo Braz. Exact lifted inference with distinct soft evidence on every object. In *AAAI*, 2012.

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

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

Fabio Gagliardi Cozman and Rodrigo Bellizia Polastro. Complexity analysis and variational inference for interpretation-based probabilistic description logics. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 117–125. AUAI Press, 2009.

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

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

Luc De Raedt, Paolo Frasconi, Kristian Kersting, and Stephen Muggleton, editors. *Probabilistic inductive logic programming: theory and applications*. Springer-Verlag, Berlin, Heidelberg, 2008. ISBN 3-540-78651-1, 978-3-540-78651-1.

Carles Farré, Werner Nutt, Ernest Teniente, and Toni Urpí. Containment of conjunctive queries over databases with null values. In *Database Theory–ICDT 2007*, pages 389–403. Springer, 2006.

Daan Fierens, Guy Van den Broeck, Ingo Thon, Bernd Gutmann, and Luc De Raedt. Inference in probabilistic logic programs using weighted CNF's. In *Proceedings of UAI*, pages 211–220, July 2011.

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

Vibhav Gogate and Pedro Domingos. Probabilistic theorem proving. In *Proceedings of UAI*, pages 256–265, 2011.

Carla P Gomes, Ashish Sabharwal, and Bart Selman. Model counting. *Handbook of Satisfiability*, 185:633–654, 2009.

Timothy Hinrichs and Michael Genesereth. Herbrand logic. Technical Report LG-2006-02, Stanford University, Stanford, CA, 2006.

Manfred Jaeger and Guy Van den Broeck. Liftability of probabilistic inference: Upper and lower bounds. In *Proceedings of the 2nd International Workshop on Statistical Relational AI*, 2012.

Abhay Jha and Dan Suciu. Probabilistic databases with markoviews. *Proceedings of the VLDB Endowment*, 5 (11):1160–1171, 2012.

Abhay Jha, Vibhav Gogate, Alexandra Meliou, and Dan Suciu. Lifted inference seen from the other side: The tractable features. In *Advances in Neural Information Processing Systems 23*, pages 973–981. 2010.

Neeraj Kayal. The complexity of the annihilating polynomial. In *Computational Complexity, 2009. CCC'09. 24th Annual IEEE Conference on*, pages 184–193. IEEE, 2009.

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

Leonid Libkin. *Elements of Finite Model Theory*. Springer, 2004. ISBN 3-540-21202-7.

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

J Scott Provan and Michael O Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, 12 (4):777–788, 1983.

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

Yehoshua Sagiv and Mihalis Yannakakis. Equivalences among relational expressions with the union and difference operators. *Journal of the ACM (JACM)*, 27(4):633–655, 1980.

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

Richard P. Stanley. *Enumerative Combinatorics*. Cambridge University Press, 1997.

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

Guy Van den Broeck. On the completeness of first-order knowledge compilation for lifted probabilistic inference. In *NIPS*, pages 1386–1394, 2011.

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

Guy Van den Broeck and Adnan Darwiche. On the complexity and approximation of binary evidence in lifted inference. In *Advances in Neural Information Processing Systems*, pages 2868–2876, 2013.

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

Guy Van den Broeck, Nima Taghipour, Wannes Meert, Jesse Davis, and Luc De Raedt. Lifted probabilistic inference by first-order knowledge compilation. In *IJCAI*, pages 2178–2185, 2011.

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

Jie-Tai Yu. On relations between jacobians and minimal polynomials. *Linear algebra and its applications*, 221: 19–29, 1995.