Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

Conditioning in First-Order Knowledge Compilation and Lifted Probabilistic Inference

Guy Van den Broeck and **Jesse Davis**
Department of Computer Science, KU Leuven
Celestijnenlaan 200A, B-3001 Heverlee, Belgium
guy.vandenbroeck@cs.kuleuven.be

Abstract

Knowledge compilation is a powerful technique for compactly representing and efficiently reasoning about logical knowledge bases. It has been successfully applied to numerous problems in artificial intelligence, such as probabilistic inference and conformant planning. Conditioning, which updates a knowledge base with observed truth values for some propositions, is one of the fundamental operations employed for reasoning. In the propositional setting, conditioning can be efficiently applied in all cases. Recently, people have explored compilation for first-order knowledge bases. The majority of this work has centered around using first-order d-DNNF circuits as the target compilation language. However, conditioning has not been studied in this setting. This paper explores how to condition a first-order d-DNNF circuit. We show that it is possible to efficiently condition these circuits on unary relations. However, we prove that conditioning on higher arity relations is #P-hard. We study the implications of these findings on the application of performing lifted inference for first-order probabilistic models. This leads to a better understanding of which types of queries lifted inference can address.

1 Introduction

Knowledge compilation is a powerful technique for reasoning about logical theories or knowledge bases. Knowledge compilation transforms or compiles a logical theory into a circuit language where certain inferences (e.g., model counting or consistency checking) can be done in polynomial time in the size of the circuit. With the circuit, it is possible to efficiently answer a large number of queries. While the compilation step may be computationally expensive, it is a one time cost that can be amortized over all subsequent queries.

Circuit reuse is one of the main computational advantages of compilation. Transformations modify the compiled circuit, enabling it to answer additional queries. Applying an efficient transformation is much cheaper than modifying the original theory and recompiling. Conditioning, which is a important operation in logic, is such a transformation. It updates a knowledge base to incorporate information about the truth values of a set of literals in the theory. All circuits in the propositional knowledge compilation map (Darwiche and Marquis 2002) support polynomial time conditioning.

Traditionally, knowledge compilation primarily focused on propositional logic. Recently, we have proposed a *first-order* circuit language, called first-order deterministic decomposable negation normal form (FO d-DNNF) circuits, and a compilation algorithm for first-order theories (Van den Broeck et al. 2011; Van den Broeck 2011). In contrast to its wide support in propositional knowledge compilation, it is currently unknown whether conditioning in *first-order* knowledge compilation is possible. This paper makes two key contribution about the feasibility and complexity of conditioning FO d-DNNF circuits. The first is an algorithm to efficiently condition a FO d-DNNF on both propositions and unary relations. The second is proving that conditioning on higher-arity relations is #P-hard.

Probabilistic inference is a well-known application of propositional knowledge compilation. By compiling a probabilistic model, such as a Bayesian network, inference can be solved by weighted model counting on the compiled circuit (Chavira, Darwiche, and Jaeger 2006; Chavira and Darwiche 2008; Fierens et al. 2011). Conditioning the circuit permits computing both marginal and conditional probabilities for each variable in the domain.

First-order knowledge compilation can be used to perform lifted inference for first-order probabilistic models (Getoor and Taskar 2007; De Raedt et al. 2008). Lifted inference (Poole 2003) algorithms are more efficient than propositional algorithms because they exploit symmetries in these models. However, the inability to condition first-order circuits requires current approaches to compile a new circuit for each query and/or evidence set. This makes computing conditional probabilities exponential in the size of evidence term, partially defeating the purpose of lifted inference (Van den Broeck 2011). The results from this paper lead to three insights into lifted probabilistic inference. First, it is possible to compile a single circuit for a first-order probabilistic model that can answer all queries on single argument atoms. Second, a compiled circuit can allow for probabilistic inference that is polynomial in the size of the evidence, provided it contains only propositions and unary relations. Empirically, this results in greatly improved performance on two benchmark domains in lifted inference. Third, in general, it is #P-hard to compute conditional probabilities if there is evidence on relations of arity two or higher.

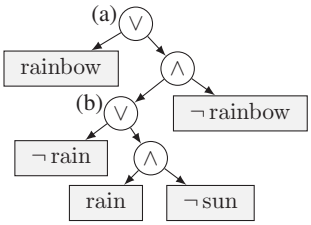


Figure 1: Propositional d-DNNF Circuit for Example 1

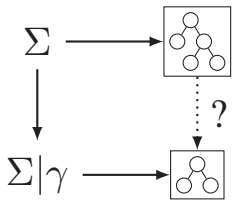


Figure 2: Conditioning a Logical Circuit

2 Propositional Conditioning

This section briefly reviews the basics of knowledge compilation and conditioning in the propositional setting. It then mentions several applications of conditioning, with a particular emphasis on probabilistic inference.

2.1 Propositional Knowledge Compilation

In propositional knowledge compilation, a propositional theory or knowledge base is compiled into a target circuit language. While the compilation step may be computationally expensive, it only needs to be done once per theory. The payoff comes when the compiled circuit admits efficient (i.e., polynomial time) inference operations such as weighted model counting and consistency checking (i.e., satisfiability). The compiled circuit can answer a wide variety of queries very quickly, and the compilation cost can be spread over all subsequent queries.

All circuit languages in the knowledge compilation map are subsets of the negation normal form (NNF) language (Darwiche and Marquis 2002). A NNF sentence is a directed, acyclic graph where the leaves are labeled with either a literal (e.g., x or $\neg$ x) or a truth value. The inner nodes represent formulae in propositional logic, being either a conjunction or a disjunction. A circuit language is a set of circuits that have a shared set of properties. A language of particular interest for this paper are deterministic decomposable negation normal form (d-DNNF) circuits (Darwiche 2001). A d-DNNF circuit restricts a NNF circuit such that all the conjunctions are decomposable and all the disjunctions are deterministic. In a *decomposable conjunction*, each pair of child nodes must be independent (i.e., they cannot share any variables). In a *deterministic disjunction*, only one of disjunction's children can be true at the same time (i.e., their conjunction is unsatisfiable).

Example 1. Figure 1 shows a d-DNNF circuit. The node labeled (a) represents the theory $\mathrm{sun} \wedge \mathrm{rain} \Rightarrow \mathrm{rainbow}$.

2.2 Conditioning Propositional Circuits

Conditioning is one of the most basic transformations that can be applied to a circuit. Intuitively, conditioning updates a knowledge base to reflect information about the truth values of specific propositions. More formally, conditioning a logical theory Σ on a term γ is denoted as $\Sigma|\gamma$, where $\gamma = l_1 \wedge l_2 \wedge \cdots \wedge l_n$ and each l_i is a literal. For each positive (negative) l_i, conditioning replaces all positive (negative) occurrences of l_i in Σ with $true$ and all negative (positive) occurrences by $false$.

Example 2. Conditioning the theory of Example 1 on $\neg\,\mathrm{rainbow}$ results in the theory $\mathrm{sun} \wedge \mathrm{rain} \Rightarrow false$, or equivalently $\neg\,\mathrm{sun} \vee \neg\,\mathrm{rain}$.

Figure 2 illustrates the difference between conditioning a knowledge base Σ and conditioning a circuit on the term γ. The vertical arrows represent conditioning on γ. Conditioning both logical theories and circuits is an efficient (polytime) operation. The horizontal arrows represent compilation of a logical theory into a circuit. Since compilation is computationally demanding, conditioning the circuit, as opposed to the Σ, permits reuse of the previously compiled circuit. Thus with a single expensive compilation step followed by conditioning, the circuit can perform efficient inference over all theories $\Sigma|\gamma$, facilitating circuit reuse.

In propositional knowledge compilation, Darwiche and Marquis (2002) have proven that every language in the knowledge compilation map can be conditioned on any term in polynomial time. In other words, the dotted line in Figure 2 exists for each type of propositional circuit. Furthermore, conditioning any circuit preserves its properties. That is, the conditioned circuit is in exactly the same circuit family as the original (i.e., unconditioned) circuit. Conditioning a NNF circuit on γ is achieved by replacing all terminal nodes that occur as variables in γ by *true* of *false* terminals.

Example 3. Conditioning the circuit of Figure 1 on $\neg\,\mathrm{rainbow}$ replaces the terminal $\mathrm{rainbow}$ with *false* and the terminal $\neg\mathrm{rainbow}$ with $true$. After some simplifications, this is equivalent to the circuit rooted at the node labeled (b).

2.3 Applications of Conditioning

Knowledge compilation and conditioning have been used to solve tasks in many different fields, such as fault diagnosis (Elliott and Williams 2006), conformant planning (Palacios et al. 2005) and databases (Dalvi, Schnaitter, and Suciu 2010). One application of knowledge compilation we explore in this paper is for probabilistic inference (e.g., (Chavira, Darwiche, and Jaeger 2006; Chavira and Darwiche 2008; Fierens et al. 2011)). Given a probabilistic model, M, the main inference task is to compute the conditional probability of some query term q given the values of evidence variables e by summing out the remaining variables from the probability distribution. Knowledge compilation solves this task by posing it as a weighted model counting problem. M assigns a probability (or weight) to each possible configuration of values the variables in M can take on. Intuitively, calculating $p(q|e, M)$ requires computing the weight of the configurations where e and q are both true and dividing it by the weight of the configurations where e is true. Formally, this can be written as follows:

$$\mathrm{P}(q|e, M) = \frac{\mathrm{WMC}(q \wedge e \wedge M)}{\mathrm{WMC}(e \wedge M)} \quad (1)$$

$$= \frac{\mathrm{WMC}(M|e|q) \cdot \prod_{l \in q} \mathrm{w}(l) \cdot \prod_{l \in e} \mathrm{w}(l)}{\mathrm{WMC}(M|e) \cdot \prod_{l \in e} \mathrm{w}(l)} \quad (2)$$

where WMC stands for the weighted model count. Note that conditioning is used to transform Equation 1 to Equation 2.

Literal Arity	Complexity of Conditioning
0	Polynomial
1	Polynomial if supported by compilation
≥ 2	#P-hard

Figure 3: Complexity of Conditioning a FO d-DNNF

After compiling M into a circuit that supports weighted model counting, conditioning this circuit permits calculating the marginal probability of each proposition in polynomial time. Furthermore, conditioning allows the circuit to compute the conditional probability of any query given any term (i.e., conjunction of literals) in polynomial time. Darwiche (2009) gives a more detailed overview.

3 First-Order Conditioning

The following subsections introduce knowledge compilation for first-order theories. We analyze which types of literals can be efficiently conditioned on in a compiled circuit that supports polytime model counting. We consider different cases based on the arity of the conditioned literal. The discussion focuses on the first-order deterministic decomposable negation normal form (FO d-DNNF) circuit language. The results for FO d-DNNFs are summarized in Figure 3

Function Free First-Order Logic An *atom* $\mathrm{p}(t_1, \ldots, t_n)$ in function free first-order logic (FFFOL) consists of a predicate p/n of arity n followed by n arguments, which are either (lowercase) *constants* or (uppercase) *logical variables*. An atom is *ground* if it does not contain any variables. A *literal* is an atom a or its negation $\neg a$. A *clause* is a disjunction over a finite set of literals. A theory in *conjunctive normal form* (CNF) is a conjunction of clauses. A *substitution* $\Sigma\{X/t\}$ replaces the variable X in the formula Σ by the term t. We will assume that all logical variables are universally quantified. When conditioning a theory in FFFOL on the term γ, we assume that γ consists of ground literals l_i.

3.1 First-Order Knowledge Compilation

First-order knowledge compilation compiles a first-order knowledge base into a target circuit language. This paper considers the FO d-DNNF language (Van den Broeck et al. 2011), which represents theories in FFFOL with domain constraints, where all the formulae are universally quantified. *Domain constraints* are constraints that define a finite domain for each logical variable. They can take the form of $X = t$, where X is a logical variable and t is a constant or variable; $X \in D$, where D is a domain; or the negation ($\neq, \notin$) of these constraints.

A FO d-DNNF circuit is a directed, acyclic graph, where the leaves represent literals and the inner nodes represent formulae in FFFOL with domain constraints. In addition to the proposition inner node types, a FO d-DNNF includes the following three node types:

- Inclusion-exclusion $\mathrm{IE}(\phi, \psi, \chi)$, representing the formula $\phi \vee \psi$ with the additional property that $\chi \equiv \phi \wedge \psi$.

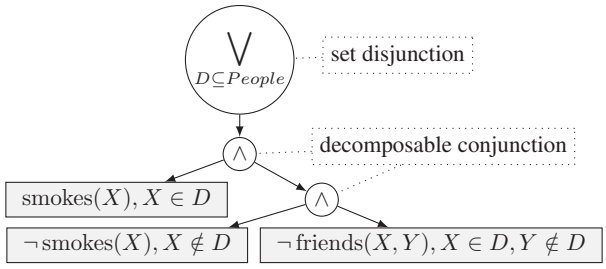


Figure 4: FO d-DNNF Circuit for Formula 3

- Decomposable set conjunction $\bigwedge_{x \in D} \Sigma\{X/x\}$ with one child circuit Σ. It represents a decomposable conjunction over all instances of Σ where one variable X is substituted by a constant x from the domain D.
- Deterministic set disjunction $\bigvee_{D' \subseteq D} \Sigma\{F/D'\}$ with one child circuit Σ. It represents a deterministic disjunction over all instances of Σ where the domain variable F is substituted by a subset of the domain D.

Example 4. Figure 4 illustrates a FO d-DNNF for the theory

$$\mathrm{smokes}(X) \wedge \mathrm{friends}(X, Y) \Rightarrow \mathrm{smokes}(Y). \qquad (3)$$

The theory states that smokers are only friends with other smokers. The circuit introduces a new domain D, which is a subset of $People$. It states that there exists such a D for which (i) all people in D are smokers (ii) no other people are smokers and (iii) smokers are not friends with non smokers.

Transforming a CNF in FFFOL into a FO d-DNNF is done by a compilation algorithm, which applies a sequence of operations that simplify the logical theory. Unlike the proposition setting, the compilation algorithm must consider formulae that apply to sets of objects, which are captured by the domain constraints. Compilation often requires modifying the domain constraints. The compilation operators are complete for any theory in FFFOL with universally quantified formulae of up to two logical variables. Theories in this class of problems, called 2-WFOMC, can always be expressed by and compiled into FO d-DNNFs. In many cases, other theories can still be compiled, but there are no guarantees. See Van den Broeck et al. (2011) and Van den Broeck (2011) for an overview of the compilation algorithm.

3.2 Conditioning on Propositions

Conditioning on a proposition works the same way as in conditioning of propositional NNF circuits. All that it requires is replacing terminal nodes which represent that literal by a *true* or *false* terminal.

Theorem 1. *A FO d-DNNF circuit can be conditioned on any proposition (or literal with arity zero) in polynomial time and its result is also a FO d-DNNF.*

Proof outline. Following from the results in the propositional setting (Darwiche and Marquis 2002), conditioning preserves the properties of the propositional inner nodes. For the deterministic set disjunction and decomposable set

conjunction, since their operands are all isomorphic, conditioning affects them all the same way, preserving their respective properties. The inclusion-exclusion node property is also preserved, because $\chi|\gamma \equiv \phi|\gamma \wedge \psi|\gamma$. □

3.3 Conditioning on Unary Relations

Unlike the propositional setting, where each leaf node represents a single proposition, a FO d-DNNF circuit terminal node can be a non-ground literal, which represents an entire *set* of ground literals. For example, the $\mathrm{smokes}(X), X \in D$ terminal in Figure 4 represents a set of people that smoke. The conditioning term does not necessarily provide the truth value for each grounding of a non-ground terminal. Even if the evidence provides the truth value for each grounding, they could be different. Thus, conditioning a circuit on a subset of the groundings requires partitioning this set of literals (i.e., into those that are true, false and unknown). Each of these partitions needs to be treated separately. Therefore, given an arbitrary FO d-DNNF, it is unclear if it can be conditioned on any unary literal.

However, by compiling extra clauses into the circuit, it is possible to efficiently condition on any unary relation term. The applicability of this transformation is independent of the domains the theory is evaluated on and the specific terms on which it will be conditioned. In order to support conditioning for a unary predicate $\mathrm{p}/1$, each clause it appears in must be split into three clauses, representing the cases when a grounding of $\mathrm{p}/1$ is true, false or unknown.

Proposition 2. *Any theory in FFFOL with domain constraints can be transformed into an equivalent representation that allows for conditioning on a unary relation* $\mathrm{p}(X)$*. First, each clause containing* $\mathrm{p}(X)$ *is split into three copies with additional domain constraints: (i)* $X \in D_\top$ *(ii)* $X \in D_\bot$ *and (iii)* $X \notin D_\top$, $X \notin D_\bot$*. For clauses with a positive* $\mathrm{p}(X)$ *literal, clause (i) is removed and* $\neg\,\mathrm{p}(X)$ *is removed from clause (ii). Conversely, for clauses with a negative* $\mathrm{p}(X)$ *literal, clause (ii) is removed and* $\mathrm{p}(X)$ *is removed from clause (i).*

Example 5. To illustrate the procedure, consider the following clause, which omits constraints on Y for readability:

$$\mathrm{p}(X) \vee \mathrm{q}(X,Y), X \in D. \tag{4}$$

We now introduce the subdomain $D_\top \subseteq D$ of constants for which we condition on $\mathrm{p}(X)$ being true, and the subdomain $D_\bot \subseteq D$ of constants for which we condition on $\mathrm{p}(X)$ being false. Knowing that $D_\top \cap D_\bot = \emptyset$, this divides the clause into

$$\begin{aligned}
&\mathrm{p}(X) \vee \mathrm{q}(X,Y),\ X \in D_\top\\
&\mathrm{p}(X) \vee \mathrm{q}(X,Y),\ X \in D_\bot\\
&\mathrm{p}(X) \vee \mathrm{q}(X,Y),\ X \in D,\ X \notin D_\top,\ X \notin D_\bot.
\end{aligned}$$

Since $\mathrm{p}(X)$ appears as a positive literal in the initial formula, and we know that $\mathrm{p}(X),\ X \in D_\top$ is true and $\mathrm{p}(X),\ X \in D_\bot$ is false, these can be simplified to

$$\mathrm{q}(X,Y),\ X \in D_\bot \tag{5}$$

$$\mathrm{p}(X) \vee \mathrm{q}(X,Y),\ X \in D,\ X \notin D_\top,\ X \notin D_\bot. \tag{6}$$

Substituting $\{D_\top/\emptyset, D_\bot/\emptyset\}$ fills in empty domains in the constraint sets, which makes Clause 5 and the additional domain constraints on X in Clause 6 trivially satisfied, thereby recovering the original Clause 4. However, filling in non-empty domains for $D_\top$ or $D_\bot$, conditions the theory on a term of literals of $\mathrm{p}/1$. Note that the atoms $\mathrm{p}(X),\ X \in D_\top$ and $\mathrm{p}(X),\ X \in D_\bot$, which are conditioned on, are removed from the new theory entirely.

Conversely, if $\mathrm{p}/1$ had appeared as a negative literal in the initial formula, $\mathrm{p}/1$ would have be removed from the first formula and the second formula could be omitted.

The procedure outlined in Proposition 2 can be repeated to support conditioning on multiple unary relations. In summary, it is possible to condition FO d-DNNF circuits on any term of unary literals. However, this comes at the cost of a more complex compilation and a larger circuit.

3.4 Conditioning on Binary Relations

Next, we show that conditioning on binary relations is #P-hard by showing that #2SAT is reducible to it.

#2SAT A k-CNF formula is a CNF with k literals per clause. kSAT is the problem of deciding the satisfiability of a k-CNF formula. The #kSAT problem involves counting the number of satisfying assignments to a k-CNF formula. This task is also called model counting.

Example 6. The following formula is in 2-CNF:

$$(a \vee b) \wedge (a \vee \neg c) \wedge (\neg c \vee \neg d)$$

2SAT is decidable in polynomial time. However, #2SAT is #P-complete (Valiant 1979), which implies that it is not solvable in polynomial time unless $P = NP$.

Lemma 3. *Each propositional 2-CNF can be represented by conditioning the first-order logic theory*

$$\begin{aligned}
\mathrm{p}(X) \vee \quad \mathrm{p}(Y) \vee \neg\,\mathrm{c}_1(X,Y)\\
\mathrm{p}(X) \vee \neg\,\mathrm{p}(Y) \vee \neg\,\mathrm{c}_2(X,Y)\\
\neg\,\mathrm{p}(X) \vee \neg\,\mathrm{p}(Y) \vee \neg\,\mathrm{c}_3(X,Y),
\end{aligned} \tag{7}$$

where all variables are implicitly universally quantified.

The c_i-predicates encode which propositions appear together in the three possible types of clauses for 2-CNF. Conditioning on a positive c_i literal includes the clause of type i for the given propositions in the 2-CNF. For example, conditioning on $\mathrm{c}_1(a,b)$ adds $\mathrm{p}(a) \vee \mathrm{p}(b)$ to the theory. Conditioning on a negative c-literal omits the clause for the given propositions from the theory. For example, conditioning on $\neg\,\mathrm{c}_2(a,b)$ excludes $\mathrm{p}(a) \vee \neg\,\mathrm{p}(b)$ from the theory.

Example 7. Conditioning Theory 7 on the term

$$\begin{aligned}
&\mathrm{c}_1(a,b) \wedge \neg\,\mathrm{c}_1(a,a) \wedge \cdots \wedge \neg\,\mathrm{c}_1(d,d) \wedge\\
&\mathrm{c}_2(a,c) \wedge \neg\,\mathrm{c}_2(a,a) \wedge \cdots \wedge \neg\,\mathrm{c}_2(d,d) \wedge\\
&\mathrm{c}_3(c,d) \wedge \neg\,\mathrm{c}_3(a,a) \wedge \cdots \wedge \neg\,\mathrm{c}_3(d,d)
\end{aligned}$$

results in the theory

$$(\mathrm{p}(a) \vee \mathrm{p}(b)) \wedge (\mathrm{p}(a) \vee \neg\,\mathrm{p}(c)) \wedge (\neg\,\mathrm{p}(c) \vee \neg\,\mathrm{p}(d)),$$

which is isomorphic to the 2-CNF of Example 6.

Theorem 4. *In any formal system that can represent Theory 7, either conditioning on literals with arity ≥ 2 or model counting is #P-hard.*

Proof. Theory 7 can be conditioned on binary literals to represent any 2-CNF. A subsequent model counting step can solve any #2SAT problem. This shows that a #P-complete problem is reducible to conditioning and model counting on Theory 7, which means it must be at least as hard as any problem in #P, or #P-hard. □

Consequences for Knowledge Compilation Theorem 4 can be specialized for first-order knowledge compilation.

Theorem 5. *Conditioning on literals with arity ≥ 2 is #P-hard in any circuit language that is expressive enough to represent Theory 7 and that allows for polytime model counting.*

Theory 7 is in 2-WFOMC and can therefore be compiled into a FO d-DNNF circuit. Because the FO d-DNNF language also supports polytime model counting, conditioning on binary terms is #P-hard in this language, and no polynomial algorithm for it exists, unless P=NP.

3-CNF and 3SAT This raises the question as to whether the inability to efficiently condition on binary relations is a property specific to the FO d-DNNF circuit language. We can make an analogous argument based on 3-CNF, which is represented by the first-order theory

$$\begin{aligned} \mathrm{p}(X) \vee \quad \mathrm{p}(Y) \vee \quad \mathrm{p}(Z) \vee \neg\, \mathrm{c}_1(X,Y,Z) \\ \mathrm{p}(X) \vee \quad \mathrm{p}(Y) \vee \neg\, \mathrm{p}(Z) \vee \neg\, \mathrm{c}_2(X,Y,Z) \\ \mathrm{p}(X) \vee \neg\, \mathrm{p}(Y) \vee \neg\, \mathrm{p}(Z) \vee \neg\, \mathrm{c}_3(X,Y,Z) \\ \neg\, \mathrm{p}(X) \vee \neg\, \mathrm{p}(Y) \vee \neg\, \mathrm{p}(Z) \vee \neg\, \mathrm{c}_4(X,Y,Z) \end{aligned} \tag{8}$$

3SAT *is* NP-complete and #3SAT is #P-complete.

Theorem 6. *Conditioning on literals with arity ≥ 3 is NP-hard in any circuit language that is expressive enough to represent Theory 8 and that allows for polytime consistency checking.*

Proof. Theory 8 can be conditioned on ternary literals to represent any 3-CNF. A subsequent consistency checking step can solve any 3SAT problem. □

Although Theory 8 is not in 2-WFOMC, it can be compiled into a FO d-DNNF circuit. This result does not provide further insight into FO d-DNNF circuits, as they support model counting and therefore conditioning is #P-hard. However, this result does provide insight into which transformations are efficient in other (yet to be defined) circuit languages that can perform consistency checking, but not model counting, in polynomial time. These correspond to the capabilities for propositional DNNF circuits (Darwiche and Marquis 2002).

4 Evidence in Lifted Probabilistic Inference

First-order knowledge compilation can be used to perform lifted inference (Poole 2003) in *first-order* probabilistic models (Getoor and Taskar 2007; De Raedt et al. 2008). Just like the propositional case, computing conditional probabilities in this setting requires calculating the weighted model counts in the numerator and denominator of Equation 1. Because of the lack of support for conditioning in first-order circuits, the current approach is to compile a separate circuit for the numerator and denominator. Furthermore, recompilation is needed whenever the query q or evidence e changes. However, by applying the transformation of Proposition 2, we can compile a single circuit that can compute the marginal or conditional probability of any proposition or unary atom, provided the evidence is only on other propositions or unary atoms.

This section deals with exact probabilistic inference. In the approximate lifted inference literature, Nath and Domingos (2010) and Hadiji, Ahmadi, and Kersting (2011) deal with a similar problem setting, where lifted belief propagation inference is performed with changing evidence.

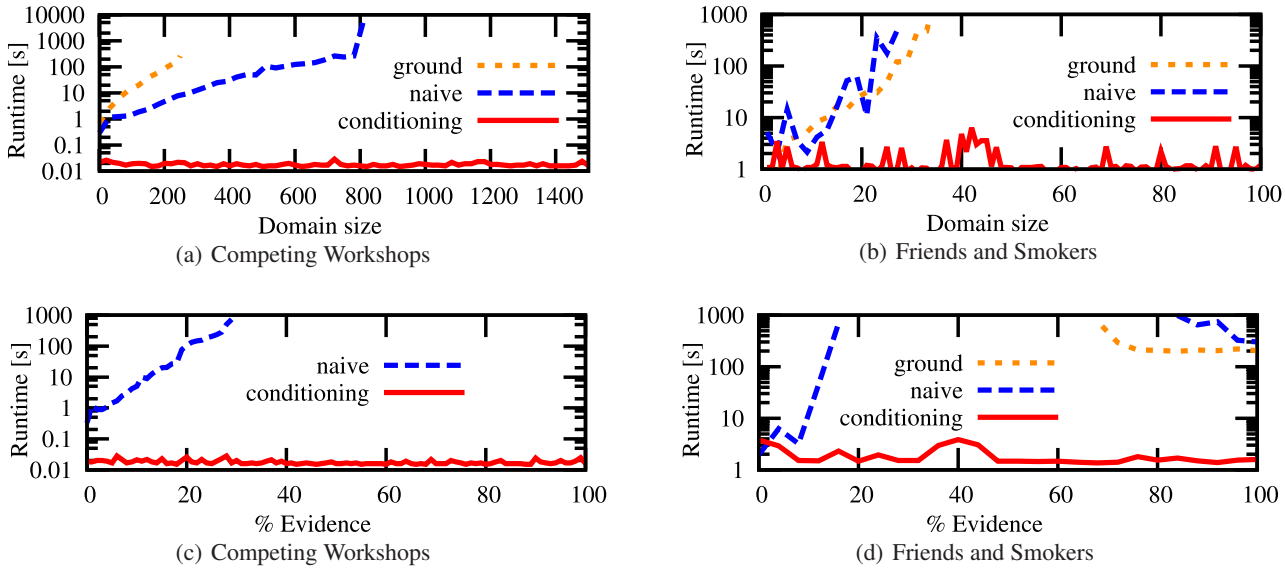
Positive Result One way to define lifted inference more formally is *domain lifted* inference (Van den Broeck 2011), which requires that the inference algorithm is polynomial in the domain sizes, but it can be exponential in the size of the model M, query q and evidence e. The results from Section 3.3 allow for a stronger definition: inference has to be polynomial in the domain sizes and the size of the evidence on propositions and unary relations. Lifted inference by first-order knowledge compilation is also domain-lifted according to this stronger definition, when applying the transformation of Proposition 2. This is a positive result for lifted inference, because many more queries about first-order probabilistic models can now be answered efficiently.

Negative Result In contrast, Theorem 5 is a strong negative result for lifted inference. It proves that first-order knowledge compilation for lifted inference will not be able to compute conditional probabilities with evidence on binary relations efficiently, unless P=NP. This raises the question whether the negative result holds only for this approach, or for any lifted inference algorithm on similar models.

Theorem 7. *For any probabilistic model that can express a uniform distribution over the models of $\neg\, \mathrm{q} \vee \phi$, where ϕ is Theory 7, computing conditional probabilities exactly is #P-hard.*

Proof outline. Querying for $\mathrm{P}(\mathrm{q}\,|e)$, where e assigns a truth value to every c_i atom, returns $\frac{c}{c+2^n}$, where c is the model count of the 2-CNF $\phi|e$ and 2^n the number of possible assignments to the n propositions in the 2-CNF. This allows us to solve arbitrary #2SAT problems by solving for c. □

This theorem applies to many first-order probabilistic languages, including parfactor graphs (de Salvo Braz, Amir, and Roth 2005), Markov logic networks (Richardson and Domingos 2006) and weighted first-order model counting (Van den Broeck 2011) and therefore to all known exact lifted inference algorithms which work on these representations, such as First-Order Variable Elimination (de Salvo Braz, Amir, and Roth 2005; Milch et al. 2008). It highlights an important limitation of all exact lifted inference methods: Computing probabilities with evidence on binary relations cannot be polynomial in the size of the evidence, unless P=NP.



(a) Competing Workshops (b) Friends and Smokers

(c) Competing Workshops (d) Friends and Smokers

Figure 5: Plots (a) and (b) show runtime with varying domain size and fixed percentage of evidence (50%). Plots (c) and (d) show runtime with varying percentage of evidence and fixed domain size ((c) 1000 people, (d) 50 people).

5 Experiments

To complement the theoretical results, we demonstrate the utility of conditioning in FO d-DNNF for the task of performing lifted probabilistic inference via weighted model counting. Our goal is to address the following question: Does conditioning on unary relations perform better than asserting evidence in lifted probabilistic inference?

Methodology To answer this question, we add support for conditioning to the FO d-DNNF compiler implementation[1] and compare the performance of these three systems:

Conditioning FO d-DNNF compiler with conditioning supported as described in Proposition 2.

Naive FO d-DNNF compiler without conditioning. For each evidence set, the evidence is conjoined to the theory (as in Equation 1), which is then compiled. This gives a separate circuit for each evidence set.

Ground Propositional knowledge compilation using the c2d compiler.[2] The unconditioned, grounded models are too large to compile, so the evidence must be used to simplify them, after which they are compiled.

We use two common benchmarks in the lifted inference literature. The *competing workshops* domain (Milch et al. 2008) models that (i) whether a person attends a workshop depends on the number of popular competing workshops and (ii) whether a workshop becomes a series depends on its attendance. The *friends and smoker*s domain (Singla and Domingos 2008) models that (i) smoking increases your risk of getting cancer and (ii) friends have similar smoking habits. In all experiments, the goal is to compute the partition function (i.e., the denominator of Equation 1).

[1] http://dtai.cs.kuleuven.be/wfomc/

[2] http://reasoning.cs.ucla.edu/c2d/

Results Figures 5(a) and 5(b) illustrate the results for the first experiment where we vary the number of objects in the domain while holding the proportion of observed unary literals at 50%. Both the naive and ground approaches are always slower than conditioning, whose curve stays relatively flat. When the domain size reaches 20 objects, conditioning is between one and two orders of magnitude faster than the baseline algorithms. It quickly becomes impossible for the baseline algorithms to compile the theories, whereas the conditioning approach has no such difficulty.

Figures 5(c) and 5(d) illustrate the results for the second experiment. Here, we hold the domain size constant (1000 and 50 people) and vary the proportion of observed unary literals. Conditioning works for any amount of evidence with relatively consistent runtimes. The naive approach works for low levels of evidence, which is the ideal case of lifted inference because most objects are indistinguishable and the problem is highly symmetric. However, once evidence reaches 10%, conditioning is one to three orders of magnitude faster than the naive approach. Ground compilation never runs for the competing workshops domain. In the friends and smokers domain, both the naive and ground strategies work for high evidence levels because the evidence sufficiently simplifies the theory. Still, both are at least two orders of magnitude slower than conditioning.

Both experiments clearly demonstrate that the benefit of supporting conditioning on unary relations far outweighs the cost of applying the transformation from Proposition 2 and compiling a more complex circuit. The answer to the experimental question is a clear and decisive yes.

6 Conclusions

Until now, it was unknown whether conditioning was possible for first-order knowledge compilation techniques. This paper makes two key contributions regarding conditioning in FO d-DNNF circuits. One, it provides an algorithm to efficiently condition on both propositions and unary relations. Two, it proves conditioning on higher arity relations is #P-hard. These two results have significant implications for exact lifted probabilistic inference techniques. The positive result is that more queries can be efficiently answered with a single circuit and inference can be polynomial in the size of the evidence on propositions and unary relations, whereas previously it was exponential. The negative results is that for evidence on binary relations, exact lifted inference algorithm can only efficiently compute conditional probabilities if P=NP. Still, many domains contain unary relations and these results improve the efficiency of lifted inference in practice, as is shown in the experiments.

Acknowledgements

Guy Van den Broeck is supported by the Research Foundation-Flanders (FWO-Vlaanderen). Jesse Davis is partially supported by the Onderzoeksfonds/Research Fund KU Leuven. The authors would like to thank Luc De Raedt for valuable feedback.

References

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.

Dalvi, N.; Schnaitter, K.; and Suciu, D. 2010. Computing query probability with incidence algebras. In *Proceedings of ACM SIGMOD/PODS Conference*, 203–214.

Darwiche, A., and Marquis, P. 2002. A Knowledge Compilation Map. *Journal of Artificial Intelligence Research* 17(1):229–264.

Darwiche, A. 2001. On the tractability of counting theory models and its application to belief revision and truth maintenance. *Journal of Applied Non-Classical Logics* 11(1-2):11–34.

Darwiche, A. 2009. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press.

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

de Salvo Braz, R.; Amir, E.; and Roth, D. 2005. Lifted First-Order Probabilistic Inference. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, 1319–1325.

Elliott, P., and Williams, B. 2006. DNNF-based belief state estimation. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*.

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 the 27th Conference on Uncertainty in Artificial Intelligence (UAI),*.

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

Hadiji, F.; Ahmadi, B.; and Kersting, K. 2011. Efficient sequential clamping for lifted message passing. *KI 2011: Advances in Artificial Intelligence* 122–133.

Milch, B.; Zettlemoyer, L. S.; Kersting, K.; Haimes, M.; and Kaelbling, L. P. 2008. Lifted Probabilistic Inference with Counting Formulas. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, 1062–1068.

Nath, A., and Domingos, P. 2010. Efficient lifting for online probabilistic inference. In *Proceedings of AAAI*.

Palacios, H.; Bonet, B.; Darwiche, A.; and Geffner, H. 2005. Pruning Conformant Plans by Counting Models on Compiled d-DNNF Representations. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling*, 141–150.

Poole, D. 2003. First-Order Probabilistic Inference. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 985–991.

Richardson, M., and Domingos, P. 2006. Markov Logic Networks. *Machine Learning* 62(1):107–136.

Singla, P., and Domingos, P. 2008. Lifted First-Order Belief Propagation. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, 1094–1099.

Valiant, L. G. 1979. The Complexity of Enumeration and Reliability Problems. *SIAM Journal on Computing* 8(3):410–421.

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 the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, 2178–2185.

Van den Broeck, G. 2011. On the Completeness of First-Order Knowledge Compilation for Lifted Probabilistic Inference. In *Annual Conference on Neural Information Processing Systems (NIPS)*.