Lifted Reasoning Meets Weighted Model Integration

Jonathan Feldstein[1] **Vaishak Belle**[1]

[1]University of Edinburgh, United Kingdom

Abstract

Exact inference in probabilistic graphical models is particularly challenging in the presence of relational and other deterministic constraints. For discrete domains, weighted model counting has emerged as an effective and general approach in a variety of formalisms. Weighted first-order model counting, which allows relational atoms and function-free first order logic has pushed the envelope further, by exploiting symmetry properties over indistinguishable groups of objects, and by extension avoids the need to perform inference on the exponential ground theory. Given the limitation to discrete domains, the formulation of weighted model integration was proposed as an extension to weighted model counting for mixed discrete-continuous domains over both symbolic and numeric weight functions. While that formulation has enjoyed considerable attention in recent years, there is very little understanding on whether the task can be solved at a lifted level, that is, whether we can reason with relational models by avoiding grounding. In this paper, we consider this question. We show how to generalize algorithmic ideas known in the circuit compilation for function-free lifted inference to functions with a continuous range.

1 INTRODUCTION

Unifying logic and probability is a long-standing challenge in AI [Russell, 2015]. In that regard, statistical relational learning and probabilistic relational models [Getoor and Taskar, 2007] are promising candidates [Raedt et al., 2016]. The logical notions capture objects, properties and relations, whereas the underlying probability theory addresses uncertainty and noisy knowledge acquisition. Unfortunately, although probabilistic inference is already known to be computationally intractable [Valiant, 1979, Bacchus et al., 2009], the enabling of logical representations makes the construction of general-purpose algorithms harder, especially in the presence of relational and other deterministic constraints. In that regard, weighted model counting (WMC) has emerged as an effective and general approach in a variety of formalisms, including Bayesian networks [Chavira and Darwiche, 2008], their relational extensions [Chavira et al., 2006], factor graphs [Choi et al., 2013], probabilistic programs [Fierens et al., 2013], and probabilistic databases [Suciu et al., 2011]. They rely on SAT technology — WMC is simply the weighted version of model counting or #SAT [Bacchus et al., 2009] — which makes inference possible in the presence of logical constraints. Exact WMC solvers are based on knowledge compilation or exhaustive DPLL search [Sang et al., 2005]. Knowledge compilation is a paradigm which aims to construct a circuit [Darwiche and Marquis, 2002, Muise et al., 2012], and is particularly attractive in the presence of repeated query computations [Van den Broeck et al., 2010]. Approximate WMC algorithms use local search [Wei and Selman, 2005] or sampling [Chakraborty et al., 2014].

However, WMC has major limitations, the first is that it is only capable of reasoning with a propositonal language. In the presence of first-order logic (FOL), which allows relational representations, several approaches have been introduced [de Salvo Braz et al., 2005, Van den Broeck et al., 2011, Gogate and Domingos, 2011]. The benefit of using relational representations is that symmetry properties over indistinguishable groups of objects can be exploited, thereby avoiding the need to perform inference on the ground theory, which in the worst case is exponential in size. This technique is known as *lifted reasoning*.

The second limitation of WMC, which also applies to the lifted reasoning approaches named above, is that they all assume a discrete domain. Generalizing the assembly language for those representations to continuous and mixed-discrete domains was not well-understood, until recently

[Belle et al., 2015]. The notion of weighted model integration (WMI) has been shown to upgrade WMC to such hybrid settings. Logically speaking, WMC and lifted WMC focus on propositional and function-free relational logic, whereas WMI allows us to additionally handle nullary function symbols (i.e., nullary terms) over continuous domains, and practically resorts to satisfiability modulo theory (SMT) technology [Barrett et al., 2009, Sebastiani and Tomasi, 2012]. However, WMI remains limited to atomic variables, as it focuses on nullary function symbols, and it does not handle relational atoms. In this work, we take steps towards lifting the paradigm of WMI and present four results.

Results First, we propose a definition of weighted first-order model integration (WFOMI). Then, we discuss how we can handle k-ary function symbols, with finitely many arguments, but continuous ranges and present as the second result an algorithm to efficiently compute the WFOMI. We will need to consider how to generalize algorithmic ideas known in the circuit construction for function-free lifted inference to function symbols with a continuous range. Thirdly, we identify settings in which WFOMI can be reduced to WFOMC. Finally, we highlight the efficiency of our algorithm with empirical results.

In terms of organisation, first in section 2 we introduce some preliminaries. Then, in section 3, using a simple relational mixed discrete-continuous example, we study the issues therein in more detail. Next, we show, in section 4, how a hybrid (in the sense of discrete-continuous) FOL circuit can be obtained and the problems posed by integration in the presence of non-trivial but realistic weight functions. Then, in section 5, we discuss settings where WFOMI can be reduced to WFOMC. In section 6, we run comparisons of our algorithms to state-of-the-art hybrid solvers. At the end, in section 7, we discuss related work, and conclude the work in section 8.

2 PRELIMINARIES

In propositional logic a *formula* ϕ is defined over atoms and logical connectives ($\wedge$, $\vee$, $\implies$, $\neg$). An *atom* in propositional logic is a Boolean variable. A *literal* is either an atom or its negation. A *clause* is the disjunction of literals. A formula in *conjunctive normal form* (CNF) is a conjunction of clauses. A *model* M assigns atoms to $\{true, false\}$, and the notion of satisfaction is defined inductively as usual for formulae. Given a propositional formula ϕ over a set $\mathbf{B}$ of Boolean variables and logical connectives, model counting (#SAT) is the problem of counting the possible assignments to the variables in $\mathbf{B}$, such that the formula is satisfied.

In (finite-domain) first-order logic (FOL), *terms* are either logical variables or constants from a finite set – the domain. Atoms are made of *predicates* with arity n and terms. An atom is *ground* if it does not contain variables. A *substitution* $\theta = [X_1/t_1, \dots, X_n/t_n]$ is a mapping from logical variables to logical terms. Applying the substitution θ to a formula ϕ is then denoted by $\phi\theta$. In addition, in FOL, we add universal and existential quantifiers ($\forall, \exists$). A constraint set, denoted cs, is a conjunction of constraints applied to the domain. The set of substitutions to a set of logical variables $\mathbf{X}$ for which cs is satisfied is denoted by solutions($cs, \mathbf{X}$).

Satisfiability modulo theory (SMT), is an extension of SAT, where we consider real variables, in addition to Boolean variables. An SMT atom over n real variables $\mathbf{x}$ is of the form $f(\mathbf{x}) \bowtie c$, where $c \in \mathbb{R}$, f is a real arithmetic function, and $\bowtie \in \{=, \neq, <, >, \leq, \geq\}$. An SMT formula is made from SMT and Boolean atom and composed over logical connectives such as $\{\wedge, \vee, \neg, \implies\}$. The definition of literals and clauses is as before. The semantics are defined inductively as usual, see Barrett et al. [2009].

2.1 WEIGHTED MODEL COUNTING

Weighted model counting (WMC) is an extension to #SAT, where we introduce a function $w : \mathcal{L} \to \mathbb{R}^{\geq 0}$, which maps literals to non-negative weights. These can be seen as an unnormalized likelihood. The notation $l \in M$ and $M \models \triangle$ means that the literal l is true in the model M and M satisfies a theory $\triangle$, respectively. We can now give the definition of WMC.

Definition 1 (WMC)**.** *Let $\triangle$ be a propositional formula, and $w : \mathcal{L} \to \mathbb{R}^{\geq 0}$ a non-negative function over the literals in the formula. Then we define*

$$\text{WMC}(\triangle, w) = \sum_{M \models \triangle} \prod_{l \in M} w(l)$$

Example 1 (WMC)**.** *Let us assume the simple propositional formula $p \vee q$, with weights $w(p) = 0.1$, $w(\neg p) = 0.9$, $w(q) = 1$, $w(\neg q) = 2$. It has three models $M_1 = \{p, \neg q\}, M_2 = \{\neg p, q\}, M_3 = \{p, q\}$, and the weighted model count is*

$$\text{WMC}(\triangle, w) = 0.1 \cdot 2 + 0.9 \cdot 1 + 0.1 \cdot 1 = 1.2$$

We can then compute the probability of p using

$$Pr(p) = \frac{\text{WMC}(p \wedge \triangle, w)}{\text{WMC}(\triangle, w)} = \frac{0.1 \cdot 2 + 0.1 \cdot 1}{1.2} = 0.25$$

2.2 WEIGHTED FIRST-ORDER MODEL COUNTING

The first limitation of WMC is that it does not exploit relational structures and only works on the propositional level. Weighted first-order model counting (WFOMC), in contrast, exploits symmetries using (function-free) FOL. For FOL, the weight function is simplified to be applicable directly to predicates.

Definition 2 (WFOMC)**.** *Let $\triangle$ be a FOL formula. Let $w : \mathcal{P} \rightarrow \mathbb{R}^{\geq 0}$ be a non-negative function over the predicates in the formula. For a literal l let p(l) denote the predicate of the literal. Then we define:*

$$\text{WFOMC}(\triangle, w) = \sum_{M \models \triangle} \prod_{l \in M} w(p(l)) \ .$$

Universal ($\forall$) and existential ($\exists$) quantifiers allow to reason over sets of symmetric objects. The process of abstraction is called *lifted reasoning*. Lifted reasoning reduces the computational cost significantly. For example, consider the formula $p \vee q$ which has 3 models, whereas the formula $\forall x(P(x) \vee Q(x))$ has 3^n models (out of the 4^n possible assignments for a domain of size n leading to $2n$ propositions in the ground theory). Lifted reasoning checks, whether such a quantity could be obtained algebraically by a syntactic procedure, without having to apply a standard propositional model counter to the ground theory $(P(1) \vee Q(1)) \wedge \cdots \wedge (P(n) \vee Q(n))$.

2.3 WEIGHTED MODEL INTEGRATION

The second limitation of WMC, is that it assumes a discrete domain, in the sense that the underlying variables are typically Boolean, or at best range over a finite set. This limitation extends to WFOMC. Weighted Model Integration (WMI) [Belle et al., 2015] extends WMC to hybrid domains by allowing SMT formulae instead of pure propositional logic.

For our purposes, we consider propositional interpretations that map SMT literals to $\{true, false\}$[Belle et al., 2015]. The notations for propositional interpretation are understood as before. Let $l^+ \in M$ be the domain of the continuous variables determined by the constraints specified by the propositional interpretation M.

Definition 3 (Weighted Model Integration [Belle et al., 2015, Morettin et al., 2017])**.** *Let $\triangle$ be an SMT formula over a set of Boolean variables* $\mathbf{B}$ *and SMT atoms over a set of real variables $X \in \mathbb{R}$. Let $w : \mathcal{L} \mapsto f(\mathbf{X})$ be a function mapping the literals in the formula (Boolean and SMT) to functions over the continuous variables in $\triangle$. The weighted model integral is defined as*

$$\text{WMI}(\triangle, w) = \sum_{M \models \triangle} \int_{l^+ \in M} \prod_{l \in M} w(l).$$

A hybrid example of WMI, following Example 1, is provided in the appendix. Table 1 summarizes the different existing approaches and WFOMI, which we introduce below.

Table 1: Comparison of WMC, WFOMC, WMI, and WFOMI

	Logic	weight functions
WMC	Propositional	$w : \mathcal{L} \mapsto \mathbb{R}^{\geq 0}$
WFOMC	function-free FOL	$w : \mathcal{P} \mapsto \mathbb{R}^{\geq 0}$
WMI	SMT	$w : \mathcal{L} \mapsto f(\mathbf{X})$
WFOMI	SMT FOL	$w : \mathcal{P} \mapsto f(\mathcal{F})$

3 MOTIVATING EXAMPLE

WFOMC, as discussed before, does not allow to reason about continuous variables. WMI, on the other hand, allows continuous variables but is limited to non-relational atoms. To motivate the need for relations in mixed discrete-continuous domains we consider an example from medical data [Lee et al., 2008, Yoon et al., 2006]. (The example is simplified for presentation purposes.)

Let us consider a set $People$ and two properties, s.t. $\forall x \in People$: BMI$(x) \sim \mathcal{N}(\mu = 27, \sigma^2 = 36)$, and diabetes$(x) \in \{true, false\}$. (For simplicity, the range of the BMI is truncated to BMI$(x) \in [10; 45]$.) For the remainder of the paper we will use $b(x)$ and $d(x)$ for BMI(x) and diabetes(x) in equations, respectively, for presentation purposes.

Further, a person's body mass index (BMI) strongly increases the likelihood to develop type 2 diabetes [Lee et al., 2008]. Let us for simplicity consider a linear dependency:

$$w(d(x)) = \frac{b(x)}{10} - 1 \text{ and } w(\neg d(x)) = 8 - \frac{b(x)}{10} \quad (1)$$

However, as long as the person is underweight or normal weight, that is BMI$(x) \leq 35$, the influence of its BMI on the likelihood of having diabetes will be less significant. This can be encoded into the model by adding the following relation to the knowledge base (KB)

$$\text{a}_1(x) \equiv [d(x) \implies b(x) \geq 35] \ , \quad (2)$$

where a_1 is an auxiliary predicate abstracting the formula. For the auxiliary predicate, we assume $w(\text{a}_1(x)) = 10, w(\neg\text{a}_1(x)) = 1$, that is, a model that satisfies the relation is ten times more likely then a model which does not, i.e. $\neg$(BMI$(x) \geq 35$) $\wedge$ diabetes(x). This KB has four models, which are summarized in Table 2.

Table 2: The four models of the KB of the diabetes diagnostic example

	M_1	M_2	M_3	M_4
BMI$(x) \geq 35$	$true$	$true$	$false$	$false$
diabetes(x)	$true$	$false$	$true$	$false$
$\text{a}_1(x)$	$true$	$true$	$false$	$true$

Dependencies, as in Eq. 1, where weights are functions of atoms cannot be captured by function free relational models

[Van den Broeck, 2013], which assume constant weights for each polarity of an atom. Thus, in this paper we will attempt to solve two generalizations from WFOMC; first, we allow SMT atoms, secondly, similarly to WMI [Zuidberg Dos Martires et al., 2019, Morettin et al., 2019], we allow weight functions of the function symbols appearing in the theory, e.g. $BMI(x)$, in contrast to constant weights.

The important assumption here, like in WFOMC is that the distributions, and thus the weight functions, of all the properties and relations are the same for all instances of a given type. This is precisely the assumption that implies symmetries between the objects of the set, which allows one to perform lifted inference. If we would use standard WMI to solve this problem we would have to reason about 4^n different models, where n is the number of objects in the $People$ set. Using the lifted approach we only have to compute the weight of four models, as we show below.

4 LIFTED WMI

To solve problems as described in the previous section, but without an exponential grounding, we need to upgrade the grammar of the language and the model count algorithm proposed in Van den Broeck [2013].

The main goal of this paper is to allow relational formulae over expressions with continuous ranges. This is why we need in addition to the standard k-ary predicates, such as family(x, y), SMT atoms and predicates defined over k-ary function symbols over the standard first-order connectives. Function symbols map terms to continuous ranges $f : \{t_1, \ldots, t_k\} \to \mathbb{R}$. We denote by $\mathcal{F}$ the set of all function symbols appearing in a theory. We define the syntax and semantics inductively, as usual; for example, see Barrett et al. [2009]. However, note that in our setting, the only predicates are binary inequality operators, i.e. $\bowtie \in \{=, \neq, \geq, \leq, <, >\}$ and atoms are assumed to be of the form $f(t_1, \ldots, t_k) \bowtie c$, where c is a numeric constant. For example, BMI(Alice) = 36, and BMI$(x) \geq 35$ are atoms, the former a ground atom. Expressions over atoms with arithmetic expressions, e.g. $f(x) + g(x) \leq c$, which WMI solvers can solve, are left for future research. In addition to SMT atoms, we are interested in weight functions that are not constant, such as those mentioned in the motivational example, e.g. Eq. 1. These weight functions can also be defined for relational atoms. For example, in Eq. 1 $w(d(x))$ is a function of BMI(x). We are now prepared to provide a definition of a first-order WMI, closely following the definition of WFOMC.

Definition 4 (WFOMI). *The weighted first-order model integral for a theory $\triangle$ is defined as*

$$\text{WFOMI}(\triangle, w) = \sum_{M \models \triangle} \int_{l^+ \in M} \prod_{l \in M} w(p(l)) \;, \qquad (3)$$

where l are the literals in the model M, $p(l)$ returns the predicate that the literal is an instance of, l^+ are the boundaries defined by the SMT atoms in M, and $w : \mathcal{P} \mapsto f(\mathcal{F})$ is a function mapping the predicates to functions over the set of function symbols $\mathcal{F}$ in M.

In order to perform inference efficiently, we follow the two step approach used in Van den Broeck [2013] for WFOMC. First, we compile the KB into a circuit, which we discuss next. Second, we perform the weight computation on the circuit, which is discussed in the second section, where we will study settings for which we still can achieve polytime inference.

4.1 KNOWLEDGE COMPILATION

Knowledge compilation yields a circuit that allows efficient repeated query computations [Van den Broeck et al., 2010, Kolb et al., 2019]. A FOL theory is complied into sda-DNNF (smooth, automorphic, deterministic, decomposable negation normal form) circuits [Van den Broeck, 2013]. These circuits are composed of leaf nodes, labelled with literals of the formula, and internal nodes labelled with logical operators. For formulae ϕ and ψ, the internal node operators are *decomposable conjunction*: $\phi \owedge \psi$ with $\phi \perp\!\!\!\perp \psi$, meaning ϕ and ψ are independent (i.e. $atoms(gr(\phi)) \cap atoms(gr(\psi)) = \emptyset$), *deterministic disjunction*: $\phi \ovee \psi$ with $\phi \wedge \psi \models \bot$, meaning ϕ and ψ are contradictory, and *smooth intensional disjunction*: $\exists \mathsf{D}, cs : \phi$ which is smooth iff $\phi\theta_1 \vee \cdots \vee \phi\theta_n$ is smooth, where $\{\theta_1 \ldots \theta_n\} = solutions(cs, \mathsf{D})$, cs is a set of constraints and D is a subset of the domain. $\phi\theta_1 \vee \cdots \vee \phi\theta_n$ is smooth iff for all i and j the groundings $gr(\phi\theta_i)$ and $gr(\phi\theta_j)$ contain the same atoms. These circuits allow polynomial time query computations in the size of the circuit. In the two-variable fragment setting, which is the FOL fragment with formulae containing at most two variables, the algorithm is guaranteed to produce a circuit in polynomial time [Van den Broeck et al., 2014]. Details on the compilation process and derivation of internal nodes operations are provided in the appendix.

We follow the knowledge compilation process in Van den Broeck [2013] closely. The compilation process involves auxiliary operations with preconditions and post conditions, such as SPLIT, each helping to achieve one of the features (smooth, automorphic, deterministic, decomposable) of the circuit. For example, the goal of SPLITTING a clause γ w.r.t a constrained atom a (e.g. $\forall x, x \in Birds : flies(x)$) is to divide a clause into an equivalent set of clauses s.t. for each atom a_γ in γ, either the atom is independent from a or it is subsumed by it. This independence leads to the *decomposability* of the circuit. Details on SPLIT and an example are described in the appendix.

The simple case in WFOMI is when the SMT atoms in the KB are disjunct, e.g. in our example we only consider BMI$(x) \geq 35$ and BMI$(x) < 35$. In this case, as the domains can be decomposed into disjoint sets, the steps to compile the circuit with the auxiliary operations discussed in Van den Broeck [2013] can be applied directly to the hybrid setting and the proofs remain the same.

However, we can run into problems, when we have "overlapping" SMT atoms. In the discrete setting, predicates are either identical or distinct, which is key to prove the correctness of the compilation process. With SMT, there is more nuance. Consider the example of two SMT atoms, (1) $18 \leq \text{age}(X) < 99$ and (2) $65 \leq \text{age}(X) < 99$. Clearly if (1) is false then (2) cannot be true, since the allowed values of (2) are a subset of those of (1).

We can solve this problem by preprocessing our formulae, restoring this identical-distinct separation. Given a functional symbol $f(X)$, as defined above, and two SMT atoms of the form $l_1 \leq f(X) \leq u_1$ and $l_2 \leq f(X) \leq u_2$ with $l_1, u_1, l_2, u_2 \in \mathbb{R}$, there are three cases to consider:

1. $[l_1, u_1] \cap [l_2, u_2] = \varnothing$. Then we can keep the SMT atoms unchanged as they are independent.
2. $l_1 \leq l_2 \leq u_1 \leq u_2$. Then we introduce new SMT atoms $l_1 \leq f(X) \leq l_2, l_2 < f(X) \leq u_1, u_1 < f(X) \leq u_2$ and we can rewrite $l_1 \leq f(X) \leq u_1 \equiv l_1 \leq f(X) \leq l_2 \veebar l_2 < f(X) \leq u_1$, and $l_2 \leq f(X) \leq u_2 \equiv l_2 < f(X) \leq u_1 \veebar u_1 < f(X) \leq u_2$.
3. $l_1 \leq l_2 \leq u_2 \leq u_1$. Then we introduce new SMT atoms $l_1 \leq f(X) \leq l_2, u_2 < f(X) \leq u_1$ and we can rewrite $l_1 \leq f(X) \leq u_1 \equiv l_1 \leq f(X) \leq l_2 \veebar l_2 \leq f(X) \leq u_2 \veebar u_2 < f(X) \leq u_1$,

where $\veebar$ is the *exclusive or*: it is true if one or the other operand is true, but not both. After this preprocessing, any standard logical sentence solver can transform the new sentence into a CNF and the normal compilation rules apply.

Theorem 1. *A hybrid KB with non-overlapping SMT atoms can be compiled into an sda-DNNF circuit using the auxiliary operations presented in Van den Broeck [2013].*

The proofs of correctness remain the same as in Van den Broeck [2013].

An important consequence of Theorem 1 is that the overall skeleton of the final circuit is analogous to the discrete setting, and that we can guarantee the compilation of a hybrid theory into a circuit for the two variable fragment. For the motivational example Forclift [Van den Broeck et al., 2011] yields the sda-DNNF circuit shown in Fig. 1. However, the nature of the weight functions in WFOMI means that we cannot compute the result immediately from such a circuit with the same operations as for WFOMC, and Theorem 1 alone does not guarantee polytime inference. In the next section, we discuss the different internal nodes separately and how the WFOMI can be computed from an sda-DNNF.

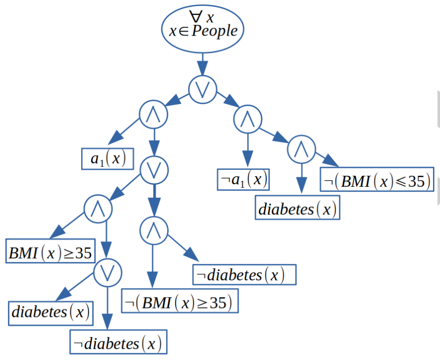

Figure 1: sda-DNNF circuit of the motivational example

4.2 COMPUTATION OF WFOMI

To compute the WFOMI, we traverse the circuit bottom-up and we apply an algebraic operation at each node depending on the logical operator. We will closely follow the definitions for WFOMC from Van den Broeck [2013].

Lemma 1. *For an extensional operator $\owedge$ or $\ovee$ in an sda-DNNF circuit with children ϕ and ψ, the following holds:*

$$\text{WFOMI}(\phi \owedge \psi, w) = \text{WFOMI}(\phi, w) \times \text{WFOMI}(\psi, w) \tag{4}$$

$$\text{WFOMI}(\phi \ovee \psi, w) = \text{WFOMI}(\phi, w) + \text{WFOMC}(\psi, w) \tag{5}$$

Lemma 2. *For an intensional operator of the form $\exists \mathsf{D}, cs$, where D is a subset of the domain, the following holds:*

$$\text{WFOMI}(\exists \mathsf{D}, cs : \phi, w) = \sum_{i=1}^{n} |\Theta_i|_{\text{WFOMI}}(\phi\theta_i, w) \tag{6}$$

where ϕ is the child of the operator, Θ_i is the set of all substitutions for D that satisfy the constraints in cs, such that $i = |\mathsf{D}|$ and $\theta_i \in \Theta_i$, and n is the full domain size.

Lemma 3. *For an intensional operator of the form $\forall X \in \boldsymbol{X}$ with child node ϕ, where for each literal $l \in \phi$ all function symbols occurring in $w(p(l))$ share exactly the same variables, the following holds:*

$$\text{WFOMI}(\forall X \in \boldsymbol{X} : \phi) = \text{WFOMI}(\phi\theta)^{|\boldsymbol{X}|} \tag{7}$$

The restrictions for the weight functions are important, as otherwise, applying these operations can lead to incorrect results, as we show in the example below. All proofs in this paper can be found in the appendix.

Remark 1. $\text{WFOMI}(\forall X \in \boldsymbol{X}, \forall Y \in \boldsymbol{Y} : \phi)$ *can have exponentially many integration computations, without an obvious way to simplify it, if $\exists l \in \phi$ s.t. $w(l) = f(f_1(X), f_2(Y))$, where l is a literal in the theory ϕ, $w(l)$ is its weight function and f_1 and f_2 are function symbols which take as inputs different terms.*

This can be seen in the following example:

Example 2. *Consider a predicate $p(x, y)$, with $w(p(x, y)) = f(a(x), b(y))$, where $a(x), b(y)$ are function symbols over two logical variables x and y, with $a(x) \in [0, 1]$, $b(y) \in [0, 1]$, i.e. similar to BMI(x). Let $\phi \equiv [\forall x, y \text{ s.t. } x \in \{d_1, d_2\}, y \in \{d_3, d_4\} : p(x, y)]$. Let us simplify the notation, s.t. $a(d_i) \equiv a_i$ and $b(d_j) \equiv b_j$, and thus $f(a(d_i), b(d_j)) \equiv f(a_i, b_j)$ then*

$$\text{WFOMI}(\phi) =$$
$$w(p(d_1, d_3)) \times w(p(d_1, d_4)) \times w(p(d_2, d_3)) \times w(p(d_2, d_4)) =$$
$$\int_0^1 \cdots \int_0^1 f(a_1, b_3) f(a_1, b_4) f(a_2, b_3) f(a_2, b_4)\ da_1 da_2 db_3 db_4.$$

Even though half of the terms, in this example, can be factored out, the number of terms in this integral will still grow exponentially with growing set sizes. Thus, the computation of the WFOMI *can become intractable in this case.*

Consider also that applying the operations used for WFOMC *on sda-DNNF circuits would instead yield $\left(\int_0^1 \int_0^1 f(a_1, b_3) da_1 db_3\right)^{2\times 2}$ which is not the same as the integral above.*

Though this looks like a direct extension of WFOMC, Remark 1 shows that the result is more subtle than one would expect. However, the expressiveness of WFOMI is only slightly affected (see Lemma 3). For two variable predicates, we can still allow weight functions over functional symbols in $\mathcal{F}$ mapping two logical variable to a continuous range, e.g. $w(p(x, y)) = f_1(x, y) \cdot f_2(x, y)$.

Lastly, in order to be able to guarantee the same polytime inference as in WFOMC, we need to consider the problem of integration. Note that integration in general is a #P-complete operation[Kawamura, 2011]. Thus, the numerical computation of the WFOMI is not guaranteed to have polytime complexity. However, based on Gerhard [2004] Theorem 7.32, we can state the following:

Theorem 2. *If all weight functions of a given sda-DNNF are rational functions f/g, where $f, g \in \mathbb{Z} \setminus \{0\}$ are polynomials of degree at most $n \in \mathbb{N}$ with integer coefficients bounded, as defined in Gerhard [2004], and the constraints in Lemma 3 are satisfied, then* WFOMI *queries can be solved in polytime.*

Kolb et al. [2019] have considered the ideal stage for integration in the WMI setting, and they have discussed how to exploit factorizability to speed up the WMI computation. We need to consider the same problem for WFOMI, where the circuits are more complex. Due to the decomposability of the circuits all predicates dependent on the same logical variable are grouped under the respective $\forall$-nodes. The only predicates which are not grouped under the $\forall$-nodes are those that define the set over which the quantifier applies. For example $\exists x \in People : BMI(x) \geq 35$, splits the set into two subsets, s.t. one set $People^\top$ contains all the people with $BMI \geq 35$, and another subset $People^\perp$ with all other people, see Fig. 2 left. However, we can push those nodes below the $\forall$-nodes during parsing, see Fig. 2 right. This procedure allows to safely push the integration inwards down to the $\forall$-node.

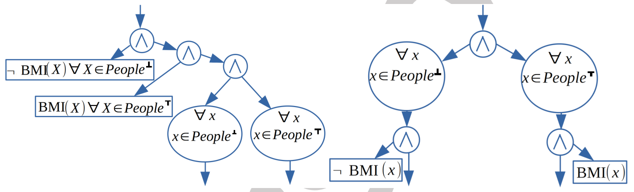

Figure 2: Left: Part of an sda-DNNF circuit, where predicates over logical variables are outside the $\forall$-node. Right: Modified circuit with all predicates over same logical variables under one $\forall$-node

4.3 ALGORITHM FOR WFOMI

In this section, we discuss how to implement the computation on the circuit as discussed above. In Algorithm 1, representing the main algorithm, we traverse the circuit top-down, and recursively call the WFOMI() function. Instead of passing weights through the circuit, we pass the weight functions of the literals *symbolically*, along with the boundaries given by inequalities of the literals, and we apply the four operations discussed above. If none of the node types fit, it means that the node is a leaf node, and the algorithm just returns its weight function and boundaries.

Algorithm 1: WFOMI(node, sets = (set, set$^\top$, set$^\perp$))

```
(child_0, child_1) = (node.child[0], node.child[1]);
if node.type = ∃ then
    return ComputeExistsNode(child_0, sets);
else if node.type = ∀ then
    return ComputeForAllNode(child_0, sets);
else if node.type = ∧ then
    return ComputeAndNode(child_0, child_1, sets);
else if node.type = ∨ then
    return ComputeOrNode(child_0, child_1, sets);
else
    return (node.wfs, node.bounds);
```

At $\exists$-nodes, Algorithm 2, the set is first split into two subsets, denoted $\top$ and $\perp$, as explained above. Then, as we reason about indistinguishable objects, and we do not care about which person specifically is in each set, but only how large the two subsets are we compute the WFOMI of the child node depending on the set size. However, we account for each possible variation of the set by multiplying the result with the binomial coefficient. Finally, we sum all the results for the variations of the set sizes.

Algorithm 2: ComputeExistsNode(child, sets)

```
wfs = 0 ;
forall $d \in$ range(set) do
    (wf, bounds) = WFOMI(child, (set, d, set-d));
    wfs += $\binom{|\text{set}|}{d}$ wf;
return (wfs, bounds = [{}]);
```

At $\forall$-nodes, Algorithm 3, we integrate each term of the weight function array of the child node, with their respective boundaries, and sum the result. As explained above, the $\exists$-nodes split the sets into two subsets. Depending on which set the $\forall$-node is over, e.g. $\forall x \in People^{\top}$, we take the power of that set size to account for each individual in that set.

Algorithm 3: ComputeForAllNode(child, sets)

```
terms = [];
forall wf, boundaries $\in$ WFOMI(child, sets) do
    terms.append(integrate(wf, boundaries));
wfs = (sum(terms))$^{|\text{node.set\_type(sets)}|}$;
return (wfs, bounds = [{}]);
```

At $\vee$-nodes, Algorithm 4, the terms of the child nodes are not directly summed up, but instead concatenated to an array. This helps the solver during integration steps, to determine over which boundaries to integrate each term.

Algorithm 4: ComputeOrNode(child$_0$, child$_1$, sets)

```
(wfs$_0$, bounds$_0$) = WFOMI(child$_0$, sets);
(wfs$_1$, bounds$_1$) = WFOMI(child$_1$, sets);
wfs = concat(wfs$_0$, wfs$_1$);
bounds = concat(bounds$_0$, bounds$_1$);
return (wfs, bounds);
```

At $\wedge$-nodes, Algorithm 5, we multiply the weight functions and find the intersection of the boundaries, if both terms of the child nodes have boundaries over the same variable. Otherwise, we keep the original boundaries. An example of Alg. 5 is provided in the appendix.

4.4 REDUCING THE INTEGRATION PROBLEM FURTHER

As explained previously, the integration becomes the bottleneck of WFOMI. We propose to adapt the algorithm to significantly reduce the number of integrals needed to be solved. To see how this works, consider the weighted model integral of the motivational example $\text{WFOMI}(\triangle, w) = \int_{35}^{45} 10 \times w(b(x)) \times \left((\frac{b(x)}{10} - 1) + (8 - \frac{b(x)}{10}) \right) \mathrm{d}b(x) + \int_{10}^{35} 10 \times w(b(x)) \times (8 - \frac{b(x)}{10}) \mathrm{d}b(x) + \int_{10}^{35} w(b(x)) \times (\frac{b(x)}{10} - 1) \mathrm{d}b(x)$.

Algorithm 5: ComputeAndNode(child$_0$, child$_1$, sets)

```
(wfs$_0$, bounds$_0$) = WFOMI(child$_0$, sets);
(wfs$_1$, bounds$_1$) = WFOMI(child$_1$, sets);
wfs = flatten(wfs$_0^{\top}$ · wfs$_1$);
bounds = [];
forall bounds$_i$ $\in$ bounds$_0$ do
    forall bounds$_j$ $\in$ bounds$_1$ do
        new_bounds = bounds$_i$ ;
        forall $p \in$ pred(bound$_i$) $\cap$ pred(bound$_j$) do
            new_bounds[p] = bounds$_i$[p] $\cap$ bounds$_j$[p];
        forall $p \in$ pred(bounds$_j$) $\setminus$ pred(bounds$_j$) do
            new_bounds[p] = bounds$_j$[p];
        bounds.append(new_bounds);
return (wfs, bounds);
```

The first integral can be simplified, but leaving three integrals to solve. A better approach is to apply the distributive property and to realise that apart from the boundaries and constants, we only have two different integrals: $\int w(b(x)) \cdot (\frac{b(x)}{10} - 1)\ \mathrm{d}b(x)$ and $\int w(b(x)) \cdot (8 - \frac{b(x)}{10})\ \mathrm{d}b(x)$. The question then is how to avoid redundant computations of these integrals differing only in their range and constants.

We propose to keep the information related to the weight functions separated into a three parts data structure: a constant array, an array of boundaries, and a weight functions array. Each of those layers is treated separately at the different nodes. Through the use of a hashing algorithm, the array structure can be efficiently checked for duplicates before computing the numerical integrals, and redundant computations can be avoided. The new Algorithm 6 is shown in detail in the appendix.

Theorem 3. *By keeping the constants separately, and checking for duplicate terms, the number of integrals needed to be solved can be reduced by $\frac{1}{2^{(m+n)}}$, where m is the number of atoms that have only one weight function for both positive and negative instances, and n is the number of atoms with constant weights.*

In the example above, the same weight function for both $\text{BMI}(x) \geq 35$ and $\neg(\text{BMI}(x) \geq 35)$ are given by the same expression but differing in range. Thus, the number of integrals to be solved is halved.

5 WHEN IS A REDUCTION TO WFOMC POSSIBLE?

Our next result is to show that by means of relational abstraction, it is possible in some very limited cases to reduce WFOMI to WFOMC. We identify two cases based on syntactic form.

Theorem 4. WFOMI *can be reduced to* WFOMC *when only SMT literals have weight functions, and the weight functions are only functions of the function symbols in the predicates they represent.*

The first part of the theorem states that non-SMT atoms, in our example diabetes(x), have constant weights. An example of a weight function of an SMT atom which only depends on the function symbol of its predicate would be the weight function of (BMI$(x) \geq 35$).

Integrating over the leaf nodes, would lead to WFOMI$(\triangle, w) = \sum_{M \models \triangle} \prod_{l \in M} \int \cdots \int_{l^+ \in M} w(p(l))$, which is different from Definition 4. The new formula computes the model weight for the example case as $\int_{35}^{45} w(d(x)) \cdot \int_{35}^{45} w(b(x) \geq 35)$ instead of $\int_{35}^{45} w(d(x)) \cdot w(b(x) \geq 35)$. However, under the above mentioned assumptions, Theorem 4 states that the result will be equivalent, which can be easily checked. This implies that if no dependency between weight functions exists, we can integrate at the leaf node and use discrete solvers such as WFOMC.

The second case is more complicated. Consider, for argument sake, a new distribution for BMI(x), a step function, which amounts to assigning constant weights w(BMI$(x) \geq 35) = 0.2$ and $w(\neg$(BMI$(x) \geq 35)) = 0.8$. Notice that w(diabetes(x)) and $w(\neg$ diabetes$(x))$, are now again weight functions dependent on BMI(x).

The naive way would be to set the weights $w(d(x)) = w(\neg d(x)) = 1$ and to add auxiliary predicates similar to a$_1(x)$ for each pair of diabetes(x), $\neg$ diabetes(x), (BMI$(x) \geq 35$) and $\neg$(BMI$(x) \geq 35$). The new auxiliary predicates in addition to a$_1$, and their respective weights are

- $w(\mathrm{a}_2(x) \equiv [d(x) \wedge b(x) \geq 35]) = \int_{35}^{45} (\frac{b(x)}{10} - 1) \mathrm{d}b(x)$
- $w(\mathrm{a}_3(x) \equiv [\neg d(x) \wedge b(x) \geq 35]) = \int_{35}^{45} (8 - \frac{b(x)}{10}) \mathrm{d}b(x)$
- $w(\mathrm{a}_4(x) \equiv [d(x) \wedge b(x) < 35]) = \int_{10}^{35} (\frac{b(x)}{10} - 1) \mathrm{d}b(x)$
- $w(\mathrm{a}_5(x) \equiv [\neg d(x) \wedge b(x) < 35]) = \int_{10}^{35} (8 - \frac{b(x)}{10}) \mathrm{d}b(x)$
- $w(\neg \mathrm{a}_i(x)) = 1$, with $i \in \{2, \ldots, 5\}$

The new sda-DNNF is inherently larger. Due to its size and for readability purposes, the new circuit is in the appendix. The resulting WFOMI computation would also be WFOMI$(\triangle, w) = \sum_{M \models \triangle} \prod_{l \in M} \int \cdots \int_{l^+ \in M} w(p(l))$. However, again under the assumptions made above, the results would be the same, as the integrals over SMT atoms would reduce to constants, and constants can be factored out leading to the equivalent result as the WFOMI from Definition 4.

Theorem 5. WFOMI *can be reduced to* WFOMC *if the weight functions only depend on other atoms with a step distribution, where the steps occur at the boundaries given by the atoms, or when predicates have constant weights.*

6 EMPIRICAL RESULTS

To the best of our knowledge, this is the first exact lifted solver for mixed discrete-continuous domains, so the most natural comparison is against state-of-the-art WMI solvers. The results reported below are the average of 100 runs on a Dell XPS13, Ubuntu, 16GB RAM with i7-10510U CPUs. The code can be found in the supplementary material.

Simple Diabetes Example We ran the motivational example, where we computed the probability of a specific person having diabetes, and compared our different algorithms to a state-of-the art exact (XADD) solver and an approximate (rejection) solver [Kolb et al., 2018]. The only difference to the example introduced above is that we used a crude polynomial approximation for BMI(x), as otherwise the XADD solver was not able to solve the problem.

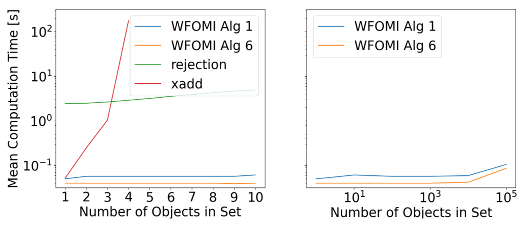

Figure 3: Comparison of query computation for different solvers on the simple diabetes example with different sizes of the *People* set.

Figure 3 shows that both WFOMI solvers' computation times stay constant, until a set size of 100,000 is reached. Still, even for such a large set size the results are computed correctly and the computation time is only 0.105s and 0.086 s for Algorithm 1 and Algorithm 6, respectively. The increase of computation time is due to the extremely large model weight (1.5e+61246). The exact XADD solver computation time increases, as expected exponentially, such that computations for a set size greater than four was not even possible. The approximate solver computation time only increases very slowly, as the majority of the computation time stems from integration step, which depends on the sample size, which was fixed as 1,000,000. However Figure 4 shows that the inaccuracy of the approximate solver increases exponentially, s.t. for larger set sizes we might get query results of over 100%.

Family Diabetes Example In the second experiment, we extend the KB with the predicate family$(x, y) \in \{true, false\}$. We further introduce an auxiliary predicate

$$\mathrm{a}_2(x, y) \equiv [(b(x) \geq 35) \wedge family(x, y) \implies (b(y) \geq 35)],$$

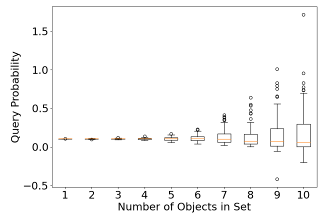

Figure 4: Boxplots showing the distribution of the results for the same query on the simple diabetes example on different set sizes.

which means that a family member of an obese person is more likely to be obese. The sda-DNNF is shown in the last section of the appendix in Fig.8. The rest of the assumptions and parameters remains the same as above.

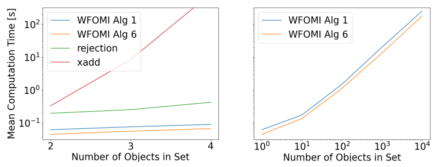

Figure 5: Comparison of query computation for different solvers on the family diabetes example with different sizes of the $People$ set.

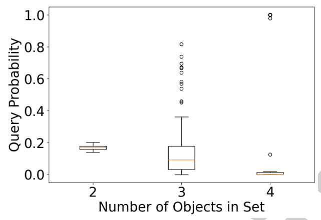

Figure 6: Boxplots showing the distribution of the results of the approximate solver for the same query on the family diabetes example on different set sizes.

Fig. 5 show that in contrast to the motivational example, the computation time does increase, however, only linearly. This result is expected, as the WFOMC computations on the friend-smokers example [Richardson and Domingos, 2006] also increased linearly. (These two examples have an analogue knowledge base.)

This is still far better than the exponential computation time increase of the current state-of-the-art solver. We were only able to run the example for a set size of 3-4, as with more objects in the set, both - the XADD solver and the approximate solver did not return any value. However, even for these few cases, we were able to show the significant impact of the new algorithms. The new solver had no difficulties running the example with 10^4 objects, as can be seen in Fig. 5 on the right. Fig. 6 shows the exponential inaccuracy of the approximate rejection solver, which we have already shown in the simpler example.

These results show the significance of the WFOMI solver on two simple example, as computation time remains very low and the solver returns an exact result. The results also show empirically the benefit of the improved Algorithm 6.

7 RELATED WORK

Since the formulation of WMI [Belle et al., 2015], numerous advances have been made [Belle et al., 2016, Morettin et al., 2017, Zeng and Van den Broeck, 2019]. Moreover, Kolb et al. [2018, 2019], Zuidberg Dos Martires et al. [2019] have even argued for a compilation target in service of repeated query computations. However, they remain limited to non-relational atoms. In similar spirit, de Salvo Braz et al. [2016] exploited properties over inequalities to make summation more effective but is still limited to nullary functions. Building on de Salvo Braz et al. [2016], de Salvo Braz and O'Reilly [2017] considers how inequalities can be exploited in relational models. A so-called inversion operator is investigated, which takes its name from the property that in *some cases* a summation of products can become a cheaper product of sums ($\sum\prod \rightarrow \prod\sum$). In contrast, as we discuss above, our goal was somewhat broader: we wanted to state precisely which first-order fragment over hybrid domains is liftable in service of constructing a first-order circuit and analyze the nuances in weight functions that affect this inquiry.

8 CONCLUSION

In this paper we presented an analysis on how to solve WMI with hybrid relational formulae and symbolic real-valued weight functions. We showed how a hybrid sda-DNNF circuit can be built and how the WMI can be computed efficiently. Many interesting directions remain for the future. One direction is, the learning of circuits, as this is very popular in discrete (propositional and relational) settings [Lowd and Domingos, 2007, 2008, Van den Broeck et al., 2012, Van den Broeck et al., 2013]. Learning the structure and parameters of the rules from data, would enable WFOMI to be tested and used on real world datasets allowing to assess the benefit of lifted reasoning in hybrid domains. Another interesting direction is to consider hybrid atoms involving arithmetic expressions over function symbols e.g. $\text{income}(x) + \text{income}(y) \leq 100,000$.

Acknowledgements

We would like to thank Efthymia Tsamoura, Marcin Korecki, and Sandor Bartha for useful discussions. The first author was supported by the Huawei Edinburgh Lab, and the second author was supported by a Royal Society University Research Fellowship.

References

F. Bacchus, S. Dalmao, and T. Pitassi. Solving #SAT and Bayesian inference with backtracking search. *J. Artif. Intell. Res. (JAIR)*, 34:391–442, 2009. doi: 10.1613/jair.2648.

C. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli. Satisfiability modulo theories. In *Handbook of Satisfiability*, chapter 26, pages 825–885. IOS Press, 2009.

V. Belle, A. Passerini, and G. Van den Broeck. Component caching in hybrid domains with piecewise polynomial densities. In *AAAI*, 2016.

Vaishak Belle, Andrea Passerini, and Guy Van den Broeck. Probabilistic inference in hybrid domains by weighted model integration. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

S. Chakraborty, D. J. Fremont, K. S. Meel, S. A. Seshia, and M. Y. Vardi. Distribution-aware sampling and weighted model counting for sat. *AAAI*, 2014.

M. Chavira, A. Darwiche, and M. Jaeger. Compiling relational Bayesian networks for exact inference. *International Journal of Approximate Reasoning*, 42(1-2):4–20, May 2006. ISSN 0888613X. doi: 10.1016/j.ijar.2005.10.001.

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

Arthur Choi, Doga Kisa, and Adnan Darwiche. Compiling probabilistic graphical models using sentential decision diagrams. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 121–132. Springer, 2013.

Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

Rodrigo de Salvo Braz and Ciaran O'Reilly. Exact inference for relational graphical models with interpreted functions: Lifted probabilistic inference modulo theories. In *UAI*, 2017.

Rodrigo de Salvo Braz, Eyal Amir, and Dan Roth. Lifted first-order probabilistic inference. In *Proc. IJCAI*, pages 1319–1325, 2005.

Rodrigo de Salvo Braz, Ciaran O'Reilly, Vibhav Gogate, and Rina Dechter. Probabilistic inference modulo theories. *arXiv preprint arXiv:1605.08367*, 2016.

Daan Fierens, Guy Van den Broeck, Joris Renkens, Dimitar Shterionov, Bernd Gutmann, Ingo Thon, Gerda Janssens, and Luc De Raedt. Inference and learning in probabilistic logic programs using weighted Boolean formulas. *Theory and Practice of Logic Programming*, 2013.

Jürgen Gerhard. *Modular algorithms in symbolic summation and symbolic integration*, volume 3218. Springer, 2004.

Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2007.

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

Akitoshi Kawamura. *Computational complexity in analysis and geometry*. University of Toronto, 2011.

Samuel Kolb, Martin Mladenov, Scott Sanner, Vaishak Belle, and Kristian Kersting. Efficient symbolic integration for probabilistic inference. In *IJCAI*, pages 5031–5037, 2018.

Samuel Kolb, Pedro Miguel Zuidberg Dos Martires, and Luc De Raedt. How to exploit structure while solving weighted model integration problems. *UAI 2019 Proceedings*, 2019.

Wei-Jei Lee, Weu Wang, Yi-Chih Lee, Ming-Te Huang, Kong-Han Ser, and Jung-Chien Chen. Effect of laparoscopic mini-gastric bypass for type 2 diabetes mellitus: comparison of bmi$>$ 35 and $<$ 35 kg/m^2. *Journal of Gastrointestinal Surgery*, 12(5):945–952, 2008.

Daniel Lowd and Pedro Domingos. Efficient weight learning for Markov logic networks. In *Proceedings of PKDD*, pages 200–211, 2007.

Daniel Lowd and Pedro Domingos. Learning arithmetic circuits. In *Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence (UAI)*, pages 383–392, 2008.

Paolo Morettin, Andrea Passerini, and Roberto Sebastiani. Efficient weighted model integration via smt-based predicate abstraction. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 720–728, 2017.

Paolo Morettin, Andrea Passerini, and Roberto Sebastiani. Advanced smt techniques for weighted model integration. *Artificial Intelligence*, 275:1–27, 2019.

C. Muise, S. McIlraith, J. Beck, and E. Hsu. D sharp: Fast d-DNNF compilation with sharpSAT. *Advances in Artificial Intelligence*, pages 356–361, 2012.

Luc De Raedt, Kristian Kersting, Sriraam Natarajan, and David Poole. Statistical relational artificial intelligence: Logic, probability, and computation. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 10(2):1–189, 2016.

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

Stuart Russell. Unifying logic and probability. *Communications of the ACM*, 58(7):88–97, 2015.

T. Sang, P. Beame, and H. A. Kautz. Performing bayesian inference by weighted model counting. In *AAAI*, pages 475–482, 2005.

R. Sebastiani and S. Tomasi. Optimization in SMT with $\mathcal{LA}\ \mathbb{Q}$ cost functions. In *Proc. IJCAR*, pages 484–498, 2012. doi: 10.1007/978-3-642-31365-3_38.

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

L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

G. Van den Broeck, Wannes Meert, and Jesse Davis. Lifted generative parameter learning. In *Statistical Relational Artificial Intelligence, AAAI Workshop*, 2013.

Guy Van den Broeck. *Lifted inference and learning in statistical relational models*. PhD thesis, Ph. D. Dissertation, KU Leuven, 2013.

Guy Van den Broeck, Ingo Thon, Martijn van Otterlo, and Luc De Raedt. DTProbLog: A decision-theoretic probabilistic Prolog. In Maria Fox and David Poole, editors, *Proceedings of the Twenty-fourth AAAI Conference on Artificial Intelligence,*, pages 1217–1222, 2010.

Guy Van den Broeck, Nima Taghipour, Wannes Meert, Jesse Davis, and Luc De Raedt. Lifted probabilistic inference by first-order knowledge compilation. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence*, pages 2178–2185, 2011.

Guy Van den Broeck, Wannes Meert, and Jesse Davis. Lifted parameter learning for Markov logic, 2012. (submitted).

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)*, pages 1–10, 2014.

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

Kun-Ho Yoon, Jin-Hee Lee, Ji-Won Kim, Jae Hyoung Cho, Yoon-Hee Choi, Seung-Hyun Ko, Paul Zimmet, and Ho-Young Son. Epidemic obesity and type 2 diabetes in asia. *The Lancet*, 368(9548):1681–1688, 2006.

Zhe Zeng and Guy Van den Broeck. Efficient search-based weighted model integration. *arXiv preprint arXiv:1903.05334*, 2019.

Pedro Miguel Zuidberg Dos Martires, Anton Dries, and Luc De Raedt. Exact and approximate weighted model integration withprobability density functions using knowledge compilation. In *Proceedings of the 30th Conference on Artificial Intelligence*. AAAI Press, 2019.