On the Completeness of Lifted Variable Elimination[*]

Nima Taghipour Daan Fierens Guy Van den Broeck Jesse Davis Hendrik Blockeel
Department of Computer Science, KU Leuven, Belgium

Abstract

Lifting aims at improving the efficiency of probabilistic inference by exploiting symmetries in the model. Various methods for lifted probabilistic inference have been proposed, but our understanding of these methods and the relationships between them is still limited, compared to their propositional counterparts. The only existing theoretical characterization of lifting is a completeness result for weighted first-order model counting. This paper addresses the question whether the same completeness result holds for other lifted inference algorithms. We answer this question positively for lifted variable elimination (LVE). Our proof relies on introducing a novel inference operator for LVE.

Introduction

Probabilistic logical models combine graphical models with elements of first-order logic to compactly model uncertainty in structured domains (De Raedt et al. 2008; Getoor and Taskar 2007). These domains can involve a large number of objects, making efficient inference a challenge. Lifted probabilistic inference methods address this problem by exploiting symmetries present in the structure of the model (Poole 2011; Kersting 2012). The basic principle is to identify “interchangeable” groups of objects and perform an inference operation once per group instead of once per individual in the group. Researchers have proposed “lifted” versions of many standard propositional inference algorithms, including variable elimination (Poole 2003; de Salvo Braz 2007; Milch et al. 2008), belief propagation (Singla and Domingos 2008; Kersting, Ahmadi, and Natarajan 2009), recursive conditioning (Poole, Bacchus, and Kisynski 2011), and weighted model counting (Gogate and Domingos 2011; Van den Broeck et al. 2011).

Despite the progress made, we have far less insight into lifted inference methods than into their propositional counterparts. Only recently has a definition been proposed for lifted inference. *Domain-lifted* inference requires the time-complexity of inference to be at most polynomial in the domain size (number of objects) of the model (Van den Broeck 2011). In contrast, standard propositional inference is typically exponential in the domain size in probabilistic logical models. Given this definition, it is possible to characterize, in the form of completeness results, which classes of models always permit lifted inference. *Weighted first-order model counting (WFOMC)* is the first lifted algorithm shown to be complete for a non-trivial model class: Van den Broeck (2011) showed that WFOMC is domain-lifted complete for 2-logvar models. These models can express many important regularities that commonly occur in real-world problems, such as (anti-)homophily and symmetry.

This raises the question whether WFOMC is fundamentally more powerful for lifted inference than other approaches, such as lifted variable elimination (LVE). In this paper, we address that question. We show that LVE is complete for the same class of models that WFOMC was shown to be complete for. This theoretical result advances our understanding about the relationship between these two approaches. The completeness theorem is proven by extending a state-of-the-art algorithm for LVE with a new operator, called group-inversion, and showing that the new algorithm has the completeness property (Taghipour et al. 2013b). In addition, we provide a brief high-level overview of the relation between various lifted operations of LVE and lifted search-based methods like WFOMC.

Representation

Like earlier work on LVE, we use a representation based on *parametrized random variables* and *parametric factors* (Poole 2003). This representation combines random variables and factors (as used in factor graphs) with concepts from first order logic. The goal is to compactly define complex probability distributions over large sets of variables.

We use the term ‘variable’ in both the logical and probabilistic sense. We use *logvar* for logical variables and *randvar* for random variables. We write variables in uppercase and values in lowercase.

Preliminaries. A *factor* $f = \phi_f(\mathcal{A}_f)$, where $\mathcal{A}_f = (A_1, \ldots, A_n)$ are randvars and ϕ_f is a *potential* function, maps each configuration of $\mathcal{A}_f$ to a real number. A *factor graph* is a set of factors F over randvars $\mathcal{A} = \bigcup_{f \in F} \mathcal{A}_f$ and defines a probability distribution $\mathcal{P}_F(\mathcal{A}) = \frac{1}{Z} \prod_{f \in F} \phi_f(\mathcal{A}_f)$, with Z a normalization constant.

[*]Most of the material in this work was published at the 16th International Conference on Artificial Intelligence and Statistics (AISTATS), 2013, Scottsdale, AZ, USA.

The vocabulary consists of *predicates* (representing properties and relations), *constants* (representing objects) and *logvars*. A *term* is a constant or a logvar. An *atom* is of the form $P(t_1, t_2, \ldots, t_n)$, where P is a predicate and each argument t_i is a term. An atom is *ground* if all its arguments are constants. Each logvar X has a finite *domain* $\mathcal{D}(X)$, which is a set of constants. A *constraint* C on a set of logvars $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a conjunction of inequalities of the form $X_i \neq t$ where t is a constant in $\mathcal{D}(X_i)$ or a logvar in $\mathbf{X}$. A *substitution* $\theta = \{X_1 \to t_1, \ldots, X_n \to t_n\}$ maps logvars to terms. When all t_i's are constants, θ is called a *grounding substitution*. Given a constraint C, we use $gr(\mathbf{X}|C)$ to denote the set of grounding substitutions to $\mathbf{X}$ that are consistent with C.

Parametrized randvars. The representation associates atoms with randvars. To this end, every predicate is assigned a *range*, i.e., a set of possible values, e.g., $range(BloodType) = \{a, b, ab, o\}$. A ground atom then represents a randvar, e.g., $BloodType(joe)$.

To compactly encode distributions over many randvars, the concept of a *parametrized randvar (PRV)* was introduced (Poole 2003). A PRV is of the form $P(\mathbf{X})|C$, where $P(\mathbf{X})$ is an atom and C is a constraint on $\mathbf{X}$. A PRV represents a set of randvars. Concretely, the set of randvars represented by a PRV $\mathcal{V} = P(\mathbf{X})|C$ is denoted $RV(\mathcal{V})$ and is defined as $\{P(\mathbf{X})\theta | \theta \in gr(\mathbf{X}|C)\}$.

Example 1. Suppose $\mathcal{D}(X) = \mathcal{D}(Y) = \{a, b, c, d\}$, where a stands for the person ann, b for bob, etc. The PRV $Friends(X, Y)|X \neq Y$ represents a set of 12 randvars, namely $\{Friends(a, b), Friends(a, c), \ldots, Friends(d, c)\}$. Similarly, the (unconstrained) PRVs $Smokes(X)$ and $Drinks(X)$ each represent a set of 4 randvars. □

Parametric factors (parfactors). Like PRVs compactly encode sets of randvars, *parfactors* compactly encode sets of factors. A parfactor is of the form $\forall \mathbf{L} : \phi(\mathcal{A})|C$, with $\mathbf{L}$ a set of logvars, C a constraint on $\mathbf{L}$, $\mathcal{A} = (A_i)_{i=1}^n$ a sequence of atoms parametrized with $\mathbf{L}$, and ϕ a potential function on $\mathcal{A}$. The set of logvars occurring in $\mathcal{A}$ is denoted $logvar(\mathcal{A})$, and we have $logvar(\mathcal{A}) \subseteq \mathbf{L}$. When $logvar(\mathcal{A}) = \mathbf{L}$, we omit $\mathbf{L}$ and write the parfactor as $\phi(\mathcal{A})|C$. A factor $\phi(\mathcal{A}')$ is called a *grounding* of a parfactor $\phi(\mathcal{A})|C$ if $\mathcal{A}'$ can be obtained by instantiating $\mathbf{L}$ according to a grounding substitution $\theta \in gr(\mathbf{L}|C)$. The set of all groundings of a parfactor g is denoted $gr(g)$.

Example 2. We use the following as our running example. Below we abbreviate $Drinks$ to D, $Friends$ to F and $Smokes$ to S. The parfactor

$$g_1 = \phi_1(S(X), F(X, Y), D(Y))|X \neq Y \quad (1)$$

represents a set of 12 factors, namely $gr(g_1) = \{\phi_1(S(a), F(a, b), D(b)), \ldots, \phi_1(S(d), F(d, c), D(c))\}$. If we choose the entries in the potential ϕ_1 appropriately, we can use this parfactor to encode, for instance, that if X is a smoker and is friends with Y, then Y is likely to be a drinker. The parfactor

$$g_2 = \phi_2(F(X, Y), F(Y, X))|X \neq Y \quad (2)$$

also represents 12 factors, and can be used to encode, for instance, that friendship is likely to be symmetric. □

Parfactor models. When talking about a *model* below, we mean a set of parfactors. In essence, a set of parfactors G is a compact way of defining a set of factors $F = \{f | f \in gr(g) \wedge g \in G\}$. The corresponding probability distribution is $\mathcal{P}_G(\mathcal{A}) = \frac{1}{Z} \prod_{f \in F} \phi_f(\mathcal{A}_f)$.

(Lifted) Variable Elimination

This section reviews the lifted variable elimination algorithm that we build on, namely C-FOVE (Milch et al. 2008).

Variable elimination calculates a marginal distribution by *eliminating* randvars in a specific order from the model until reaching the desired marginal (Poole and Zhang 2003). To eliminate a single randvar V, it first *multiplies* all factors containing V into a single factor and then *sums out* V from that single factor. Lifted variable elimination (LVE) does this on a lifted level by eliminating parametrized randvars (i.e., whole *sets* of randvars) from parfactors (i.e., *sets* of factors). The outer loop of LVE is as follows.

```
Inputs: $G$: a model; $Q$: the query randvar.
while $G$ contains other randvars than $Q$:
  if a PRV $\mathcal{V}$ can be eliminated by lifted sum-out
    $G \leftarrow$ eliminate $\mathcal{V}$ in $G$ by lifted sum-out
  else apply an enabling operator on parfactors in $G$
end while
return $G$
```

As this shows, LVE works by applying a set of lifted *operators*. We now discuss the most basic operators. Beside these, LVE has *counting* operators, which we discuss later.

Lifted sum-out. This operator sums-out a PRV, and hence all the randvars represented by that PRV, from the model. Lifted sum-out is applicable only under a precondition (each randvar represented by the PRV appears in exactly one grounding of exactly one parfactor in the model). The goal of all other operators is to manipulate the parfactors into a form that satisfies this precondition. In this sense, all operators except lifted sum-out are *enabling operators*.

Lifted multiplication. This operator performs the equivalent of many factor multiplications in a single lifted operation. It prepares the model for sum-out by replacing all the parfactors that share a particular PRV by a single equivalent product parfactor.

Splitting and shattering. These operators rewrite the model into a *normal* form in which, e.g., each pair of PRVs represent either identical or disjoint randvars.

Completeness of LVE

Lifting can yield significant speedups over standard inference. This has been demonstrated empirically in large models where a lifted algorithm can conclude inference without grounding. There, a central feature of lifted inference is scalability w.r.t. the *domain size* (the number of objects in the model). This is formally captured in the definition of *domain-lifted* inference (Van den Broeck 2011):

Definition 1 (Domain-lifted algorithm) *A probabilistic inference algorithm is* domain-lifted *for a model G, query $\mathcal{Q}$ and evidence $\mathcal{E}$ iff it runs in polynomial time in* $|\mathcal{D}_1|$, ..., $|\mathcal{D}_k|$, *with $\mathcal{D}_i$ the domain of logvar $X_i \in logvar(G, \mathcal{Q}, \mathcal{E})$.*

Note that this definition of 'lifting' requires time *polynomial* in the domain size. This is to contrast with standard propositional inference, which is often exponential in the domain size for common probabilistic logical models. This definition allows us to evaluate lifted algorithms not only by empirical evaluation on specific models, but by theoretically characterizing their completeness w.r.t. useful model classes (Van den Broeck 2011):

Definition 2 (Completeness) *An algorithm is* complete *for a class $\mathcal{M}$ of models, if it is domain-lifted for all models $G \in \mathcal{M}$ and all ground queries $\mathcal{Q}$ and evidence $\mathcal{E}$.*

Intuitively, this means that we can analyze a model syntactically and know a priori whether lifting is possible. Among all the lifted inference algorithms, the only existing completeness results belongs to WFOMC, which was shown to be complete for 2-logvar models (Van den Broeck 2011). This refers to any model where each parfactor contains at most 2 logvars. While C-FOVE has a lifted solution for some 2-logvar models, it is not complete w.r.t. this class. Consider the 2-logvar model from Example 2. C-FOVE can handle the model consisting only of the first parfactor g_1 in a lifted way (i.e., without grounding). However, including the second parfactor g_2 forces C-FOVE to ground the model and run inference with exponential complexity in the domain size. This raises the question whether this is due to an inherent limitation of LVE.

In this paper, we answer this question negatively by presenting a lifted inference solution for LVE for all 2-logvar models. For this, we introduce a novel lifted operator in C-FOVE, resulting in the C-FOVE$^+$ algorithm. This allows us to present the first completeness results for LVE, showing that C-FOVE$^+$ is complete in the same sense as WFOMC (for the proof, see Taghipour et al., 2013b).

Theorem 1 *C-FOVE$^+$ is a complete domain-lifted algorithm for 2-logvar models.*

Importance of the result. Our completeness result furthers our understanding of the relation between LVE and search-based methods. Van den Broeck (2011) showed that WFOMC is complete for the class of 2-WFOMC models. Any such model can be represented as a 2-logvar model, and vice versa. Our completeness result for LVE is thus equally strong as that of WFOMC.

The class of 2-logvar models includes many useful and often employed models in statistical relational learning. It can model multiple kinds of relations, including: *homophily* between linked entities, e.g., $\phi(Property(X), Related(X,Y), Property(Y))$; *symmetry*, e.g., $\phi(Friend(X,Y), Friend(Y,X))$; *anti-symmetry*, e.g., $\phi(Smaller(X,Y), Smaller(Y,X))$; and *reflexivity*, e.g., $\phi(Knows(X,X))$. Theorem 1 guarantees that for these models, LVE can perform inference in time polynomial in the domain size.

Secondary result. Beside our main result (Theorem 1), we also present a second result (Theorem 2), which is in line with a known result for lifted recursive conditioning (Poole, Bacchus, and Kisynski 2011). This theorem applies to models that restrict the number of logvars per *atom*, whereas Theorem 1 restricts the number of logvars per *parfactor*. Let us call a parfactor model *monadic* if each atom in the model contains at most 1 logvar.

Theorem 2 *C-FOVE$^+$ is a complete domain-lifted algorithm for monadic models.*

Summary. The state of the art in LVE is the result of various complementary efforts (Poole 2003; de Salvo Braz 2007; Milch et al. 2008; Apsel and Brafman 2011; Taghipour et al. 2012; Taghipour and Davis 2012; Taghipour et al. 2013b). Table 1 summarizes the resulting LVE algorithms and shows their completeness w.r.t. the two classes of monadic and 2-logvar models. The next sections briefly introduce the machinery that allows us to establish the completeness results; for the proofs of theorems and further details, we refer to Taghipour et al. (2013b).

A New Operator: Group Inversion

We now introduce a new lifted operator called *group inversion*. This operator is required to make LVE complete for important classes of models, as argued above. Group inversion generalizes the existing inversion operator of LVE (de Salvo Braz, Amir, and Roth 2005; Poole 2003) and is inspired by the concept of disconnected groundings in lifted recursive conditioning (Poole, Bacchus, and Kisynski 2011). We first review inversion, and then define group inversion.

Inversion

Lifted sum-out eliminates a PRV, i.e., a whole set of randvars, in a single operation. An important principle that it relies on is *inversion*, which consists of turning a sum of products into a product of sums (de Salvo Braz, Amir, and Roth 2005; Poole 2003). Consider the sum of products $\sum_i \sum_j i \cdot j$. If the range of j does not depend on i, it can be rewritten as $(\sum_j j)(\sum_i i)$, which is a product of sums. More generally, given n variables $x_1, \ldots, x_n$, with independent ranges, we have

$$\sum_{x_1}\sum_{x_2}\cdots\sum_{x_n}\prod_{i=1}^{n} f(x_i) = \prod_{i=1}^{n}\sum_{x_i} f(x_i).$$

Furthermore, if all x_i have the same range, this equals $(\sum_{x_1} f(x_1))^n$. That is, the summation can be performed for *only one* representative x_1 and the result used for all x_i.

Exactly the same principle can be applied in lifted inference for summing out randvars. Suppose we need to sum out $F(X,Y)$ from the parfactor $g_1 = \phi_1(S(X), F(X,Y), D(Y))$ from Example 2. For each instantiation (x,y) of (X,Y), $F(x,y)$ has the same range, hence applying inversion yields

$$\sum_{F(a,a)}\sum_{F(a,b)}\cdots\sum_{F(d,d)}\prod_{\theta\in\Theta} g_1\theta = \prod_{\theta\in\Theta}\Big(\sum_{F(X,Y)\theta} g_1\theta\Big) \qquad (3)$$

with $\Theta = gr(X,Y)$. This shows that we can perform the sum-out operations independently for each $F(X,Y)\theta = F(x,y)$, and multiply the results. Furthermore, since all the factors $g_1\theta$ are groundings of the same parfactor and have

Algorithm	Description	Completeness w.r.t.	
		Monadic	2-logvar
FOVE	Operators: MULTIPLY , SUM-OUT (Poole 2003; de Salvo Braz 2007)	×	×
C-FOVE	+ *counting formulas* + COUNT-CONVERT (Milch et al. 2008)	×	×
GC-FOVE	+ *extensionally complete constraint languages* + ABSORPTION (Taghipour et al. 2013a)	×	×
C-FOVE$^{\#}$	+ *generalized counting* (Taghipour and Davis 2012) $\approx$ + *joint formulas* (Apsel and Brafman 2011)	✓	×
C-FOVE^{+}	C-FOVE$^{\#}$ + *group inversion* (Taghipour et al. 2013b)	✓	✓

Table 1: Summary of completeness results for the family of LVE algorithms (✓: Complete, ×: Not complete).

the same potential ϕ_1, the result of summing out their second argument $F(X,Y)\theta$ is also the same potential, denoted ϕ'_1. It thus suffices to only perform one instance of these sum-out operations and rewrite Expression 3 as

$$\prod_{(x,y)} \Big(\sum_{F(x,y)} \phi_1(S(x), F(x,y), D(y)) \Big)$$
$$= \prod_{(x,y)} \Big(\phi'_1(S(x), D(y)) \Big) = gr(g'_1)$$

where $g'_1 = \phi'_1(S(X), D(Y))$. This is what lifted sum-out by inversion does: it directly computes the parfactor g'_1 from $g_1 = \phi_1(S(X), F(X,Y), D(Y))$ by summing out $F(X,Y)$ from g_1 in a single operation. This single lifted operation replaces $|\Theta|$ sum-out operations on the ground level.

Group Inversion: Principle

Inversion only works when the summations are independent. Our first contribution is based on the following observation. When we cannot apply inversion because of dependencies between factors, we can still partition the factors (and the summations) into *groups* such that dependencies exist only among factors within a group, but not between groups. Furthermore, we can do this at the lifted level: we can compute the result for one group and use it for all groups, provided that these groups are *isomorphic*, i.e., that there exists a one-to-one-mapping of one group's randvars to the others'.

Consider the problematic parfactor from Example 2, $g_2 = \phi_2(F(X,Y), F(Y,X))|X \neq Y$. Figure 1 depicts this model graphically. Consider summing out $F(X,Y)$ from this model. The sum related to one particular instantiation (a,b) looks as follows.

$$\ldots \sum_{F(a,b)} \sum_{F(b,a)} \phi_2(F(a,b), F(b,a)) \cdot \phi_2(F(b,a), F(a,b))$$

The product contains two factors over the considered pair of randvars $F(a,b)$ and $F(b,a)$. Inversion is not applicable here, since the product of these two cannot be moved out of the summation over either randvar. Still, the two summations are independent of all other factors, and can be isolated from the rest of the computation. The same can be done for each pair of instantiations $\{(x,y),(y,x)\}$ of (X,Y), corresponding to each pair of factors *grouped* in the same box in

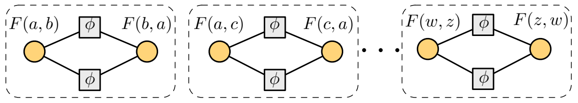

Figure 1: Group inversion on pairs of randvars. Circles represent randvars, squares represent factors. Dashed boxes indicate the partitioning into groups.

Figure 1. This means that summing out $F(X,Y)$ from g_2 can be done using *group inversion*:

$$\sum_{F(a,b)} \sum_{F(a,c)} \cdots \sum_{F(d,c)} \Big(\prod_{\theta_{xy} \in \Theta} g_2\,\theta_{xy} \Big)$$
$$= \prod_{\{\theta_{xy},\theta_{yx}\} \in \Theta} \Big(\sum_{F(X,Y)\theta_{xy}} \sum_{F(X,Y)\theta_{yx}} g_2\,\theta_{xy} \cdot g_2\,\theta_{yx} \Big)$$

where θ_{xy} is a grounding substitution $\{X \to x, Y \to y\}$ in $\Theta = gr(X,Y|X \neq Y)$. Lifting is now possible again since the groups are isomorphic (see Figure 1): for all distinct pairs of substitutions $(\theta_{xy}, \theta_{yx})$ in Θ, the pairs of factors $(g_2\theta_{xy}, g_2\theta_{yx})$ share the same potential ϕ_2. Thus, the multiplicands of each pair also have the same potential ϕ'_2, and summing out their arguments results in the same potential ϕ''_2. Hence, it suffices to perform only one (lifted) instance of these operations as follows

$$\prod_{\{(x,y),(y,x)\}} \Big(\sum_{F(x,y)} \sum_{F(y,x)} \phi'_2(F(x,y), F(y,x)) \Big)$$
$$= \prod_{\{(x,y),(y,x)\}} \phi''_2() = \prod_{(x,y)} \phi''_2()^{1/2} = gr(g'_2),$$

where g'_2 is the parfactor $\forall X, Y : \phi''_2()|\ X \neq Y$ with ϕ''_2 a potential function with no arguments, i.e., a constant, because both arguments have been summed-out.

Group inversion partitions the set of factors (and randvars) into independent and isomorphic groups. An important question is what such a partitioning looks like. Figure 1 shows this for the above example. In general, let us call two factors *directly linked* if they share a randvar, and let *linked* be the transitive closure of this relation. Factors that are linked end up in the same group. Sometimes this yields useful partitionings, sometimes not. As a 'negative' example, consider a parfactor $\phi(P(X), P(Y))$. Any two factors $\phi(P(a), P(b))$ and $\phi(P(c), P(d))$ are linked, since both are directly linked to $\phi(P(b), P(c))$. Hence the only option is the trivial partition in which all factors are in a single, large group, which is not practically useful. As a 'positive' example, consider the case where each atom uses all the logvars in the parfactor, as in the earlier example $\phi(F(X,Y), F(Y,X))|X \neq Y$. In such cases, we can always partition the randvars into groups whose size is independent of the domain size. The reason is that in such cases, the arguments of the atoms in a linked group are necessarily *permutations* of each other. Hence the size of a linked group can be no larger than the number of possible permutations, which is independent of the domain size. We use this property in our group inversion operator; for a formal operator definition, please see Taghipour et al. (2013b).

Counting

A central concept in LVE (and in our completeness proofs) that we have not discussed yet is *counting*, which is essentially a tool for allowing more lifting to take place. This requires an extension of the parfactor representation with *counting formulas* (Milch et al. 2008). In this section, we review counting formulas and the operators for handling them in LVE (Milch et al. 2008; Apsel and Brafman 2011; Taghipour and Davis 2012).

Representation. A *counting formula* is of the form $\#_{X:C}[P(\mathbf{X})]$, with $X \in \mathbf{X}$ and C a constraint on X. We call X the *counted logvar*. A *ground counting formula* is a counting formula in which all arguments except the counted logvar are constants. Such a formula represents a *counting randvar* (CRV). The value of a CRV is a histogram of the form $\{(v_i, n_i)\}_{i=1}^{|range(P)|}$, showing for each value $v_i \in range(P)$ the number n_i of covered randvars whose state is v_i.

Example 3. $\#_{Y:Y\neq X}[F(X,Y)]$ is a counting formula. Assume $\mathcal{D}(X) = \mathcal{D}(Y) = \{a,b,c,d\}$, then $\#_{Y:Y\neq a}[F(a,Y)]$ is a ground counting formula. It represents a CRV that counts how many people are (and are not) friends with a. The value of this CRV depends on the value of the three randvars $\{F(a,b), F(a,c), F(a,d)\}$. For instance, if $F(a,b) = true$, $F(a,c) = false$ and $F(a,d) = true$, the value of the CRV is the histogram $\{(true, 2), (false, 1)\}$, meaning that a has two friends and one 'non-friend'. Note that while there are $2^{|\mathcal{D}(X)|-1}$ different joint values for the covered randvars, there are only $|\mathcal{D}(X)|$ different histograms for the CRV. □

Counting formulas can be introduced in the model by the following two operators.

(Just-different) counting conversion. This operator replaces an atom (in a particular factor) by a counting formula, e.g., replace $F(X,Y)$ by $\#_Y[F(X,Y)]$. This is applicable on a set of logvars that only appear in a single atom or in *just-different* atoms (Apsel and Brafman 2011), i.e., pairs of atoms $P(X_1, \mathbf{X}), P(X_2, \mathbf{X})$ with a constraint $X_1 \neq X_2$.

Joint conversion. This auxiliary operator works on a pair of atoms $A(X), B(X)$ and replaces any occurrence of $A(.)$ or $B(.)$ with a *joint* atom $J_{AB}(.)$, whose range is the Cartesian product of the range of A and B. This is useful because it can enable counting conversion, namely when the result of the joint conversion is a model with just-different atoms.

Example 4. Consider the parfactor $\phi(S(X), D(X), S(Y), D(Y)) \ |X \neq Y$. Joint conversion on atoms $S(.)$ and $D(.)$ rewrites this parfactor as $\phi'(J_{SD}(X), J_{SD}(X),\ J_{SD}(Y), J_{SD}(Y))|X \neq Y$, which can be simplified to $\phi''(J_{SD}(X), J_{SD}(Y))|X \neq Y$. Note that $J_{SD}(X)$ and $J_{SD}(Y)$ are now just-different atoms. Next, just-different counting conversion rewrites this parfactor as $\phi'''(\#_X[J_{SD}(X)])$. This parfactor is now ready for application of lifted sum-out. □

Group inversion for counting formulas. Counting formulas, like atoms, can be eliminated by lifted sum-out. This can be done by group inversion with a small modification of the operator (Taghipour et al. 2013b): evaluate the target parfactor for each possible histogram, and compute a *weighted sum* of the resulting terms. The coefficient of each term is the *multiplicity* of its associated histograms, i.e., the number of possible assignments to the randvars that yield that histogram (Milch et al. 2008).

Relation to Lifted Search-Based Inference

Theorem 1 shows that LVE is complete for the same known class of models as WFOMC. We now show that this is not a coincidence by discussing the connection between the lifted operations of LVE and WFOMC (we use WFOMC as a representative of lifted search-based methods).

Lifted inference methods use two main tools for achieving efficiency, i.e., for *lifting*: (1) *Decompose* the problem into isomorphic and independent subproblems and *solve one* representative instance; (2) *Count* the number of isomorphic configurations for a group of *interchangeable* objects instead of enumerating all possible configurations. Below we show how LVE and WFOMC benefit from these tools.

WFOMC. Given a model as a weighted Boolean formula, WFOMC performs inference by compiling the model into a FO d-DNNF circuit, and evaluating this circuit. See Figure 2 for an example of a FO d-DNNF circuit. WFOMC can also be used for inference on parfactor models, after converting the parfactor model to a weighted formula (Van den Broeck et al. 2011). The algorithm consists of several compilation rules. Below we relate them to the operations of LVE.

Independent partial grounding (IPG) is an instance of the first lifting tool, i.e., isomorphic decomposition. IPG is the direct counterpart of *inversion* (not group-inversion). Like inversion performs the computations for a representative constant (individual) in the domain of logvars, IPG replaces a logvar with one representative constant in its child circuit. Like inversion, it is applicable when the underlying problems are independent for *each individual*.

Domain recursion (DR) is a generalization of IPG. It works with a group of logvars, instead of one, and does the following: (i) separate an individual x_i from the domain D, (ii) partition the problem P_D into independent subproblems: P_{x_i} (involving x_i), and $P_{D\setminus\{x_i\}}$ (not involving x_i), (iii) repeat the same for $P_{D\setminus\{x_i\}}$. Since the problems are

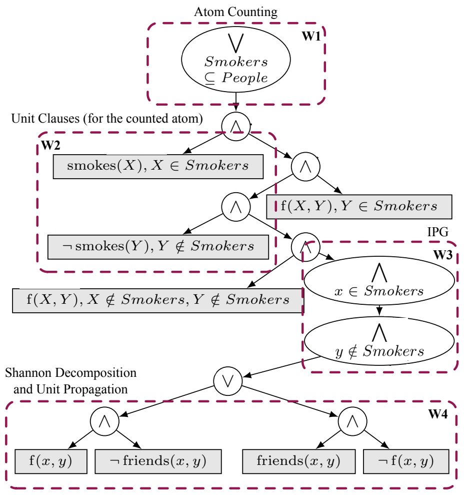
Atom Counting
W1
$\bigvee$ $Smokers \subseteq People$
Unit Clauses (for the counted atom)
W2
smokes$(X), X \in Smokers$
f$(X,Y), Y \in Smokers$
$\neg$ smokes$(Y), Y \notin Smokers$
IPG
W3
f$(X,Y), X \notin Smokers, Y \notin Smokers$
$\bigwedge$ $x \in Smokers$
$\bigwedge$ $y \notin Smokers$
Shannon Decomposition and Unit Propagation
W4
f(x,y)
$\neg$ friends(x,y)
friends(x,y)
$\neg$ f(x,y)

Figure 2: An example FO d-DNNF (taken from Van den Broeck, 2011).

isomorphic up to the domain size, it solves one instance of the problems for each size. DR tackles the same problem as *group inversion*, but with a different approach: group inversion solves an instance of the problem for a representative group of individuals, independent of the domain size; DR iterates over the possible domain sizes. This difference is not due to an intrinsic distinction between LVE and WFOMC. An operator similar to group inversion is conceivable for WFOMC (see Van den Broeck, 2013), and so is DR for LVE.

Atom counting (AC) is an example of the second lifting tool, counting, which exploits interchangeability among randvars. Instead of branching on all possible joint values for the randvars, AC parametrizes the circuit w.r.t. the number of randvars with each value: it assumes a (sub-)domain for the $true$ atoms, adds a *unit clause* for them, and iterates over the domain size. It is the direct counterpart of *counting conversion* in LVE. AC is followed by a *unit propagation* (of the produced unit clause), just as counting conversion is followed by a *sum-out* (of the produced counting formula).

Shannon decomposition introduces a branch in the circuit for each value of a ground atom, with a unit clause corresponding to that value. It is useful when followed by *unit propagation*. The combination of these two rules corresponds to the sum-out operation of LVE.

Example 5. Consider the *Friends and Smokers* theory consisting of the weighted clause $S(X) \wedge Fr(X,Y) \Rightarrow S(Y)$ (a parfactor model consisting of $\phi(S(X), Fr(X,Y), S(Y))$). The task is to compute the partition function. WFOMC solves this by evaluating the FO d-DNNF circuit shown in Figure 2. Reading the circuit top-down we see the following operations: (W1) Atom counting on $S(X)$, (W2) Eliminate $S(X)$ with unit propagation, (W3) IPG on the logvars X, Y of $Fr(X,Y)$, (W4) Eliminate $Fr(X,Y)$ with Shannon decomposition and unit propagation. LVE solves the problem as follows: (V1,V2) Eliminate $Fr(X,Y)$ using lifted sum-out by *inversion*, (V3) Counting conversion on $S(X)$, (V4) Eliminate $S(X)$ by summing-out $\#_X[S(X)]$. Note the correspondence between the operations performed by the two methods. First, they both eliminate $Fr(X,Y)$ by the first lifting tool, WFOMC with IPG and LVE with inversion. Second, they both eliminate $S(X)$ from the model by the second tool, namely *counting*. As such, both methods use the same tools for the corresponding inference subproblems. They thus achieve the same efficiency gains compared to their ground counterparts, respectively. □

Order of operations. The order in which the operations are performed in LVE is the reverse order of the corresponding operations (in a top-down traversal) of the circuit compiled by WFOMC. This relation between the two lifted methods resonates with a similar relation between their propositional counter-parts, namely search-based and VE-based methods (Darwiche 2001; Dechter and Mateescu 2007).

Connection to theorem proving. Gogate and Domingos (2011) established a connection between lifted WMC and theorem proving. They show that lifted WMC can be seen as a probabilistic generalization of the DPLL algorithm for theorem proving in logic. We point out that LVE, in turn, corresponds to a probabilistic generalization of theorem proving with a resolution-based theorem prover. This corresponds to the propositional case (Dechter 1999).

We finally note that although LVE and lifted search-based methods use the same tools for lifting, they are not necessarily equally efficient due to the characteristic differences between their propositional counterparts. For instance, search-based methods exploit the local structure of the model, such as determinism, while VE is oblivious to this structure. In the presence of such structure, search-based methods can be more efficient than LVE (for such a comparison, see Gogate and Domingos, 2011).

Conclusion

We showed how introducing a new inference operator, called group inversion, makes lifted variable elimination a complete domain-lifted algorithm for 2-logvar models. A corollary of the completeness result is that lifted variable elimination and WFOMC are currently known to be domain-lifted complete for the same subclass of models.

An interesting direction for future work is derivation of (positive or negative) completeness results for useful models that fall outside of the 2-logvar class. An example are 3-logvar models containing a transitive relation, e.g. $\phi(Like(X,Y), Like(Y,Z), Like(X,Z))$, for which no domain-lifted inference procedure is known. We further believe that future research on the relationships between the various lifted inference algorithms will yield valuable theoretical insights, similar to those about the propositional inference methods (Darwiche 2001; Dechter 1999; Dechter and Mateescu 2007).

Acknowledgements NT is supported by the research fund KU Leuven (GOA/08/008 and CREA/11/015). JD is partially supported by the research fund KU Leuven (CREA/11/015 and OT/11/051), and EU FP7 Marie Curie Career Integration Grant (#294068). DF and GVB are supported by FWO-Vlaanderen.

References

Apsel, U., and Brafman, R. I. 2011. Extended lifted inference with joint formulas. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI)*, 11–18.

Darwiche, A. 2001. Recursive conditioning. *Artif. Intell.* 126(1-2):5–41.

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.

de Salvo Braz, R. 2007. *Lifted First-order Probabilistic Inference*. Ph.D. Dissertation, Department of Computer Science, University of Illinois at Urbana-Champaign.

Dechter, R., and Mateescu, R. 2007. And/or search spaces for graphical models. *Artif. Intell.* 171(2-3):73–106.

Dechter, R. 1999. Bucket elimination: A unifying framework for reasoning. *Artif. Intell.* 113(1-2):41–85.

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

Gogate, V., and Domingos, P. 2011. Probabilistic theorem proving. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI)*, 256–265.

Kersting, K.; Ahmadi, B.; and Natarajan, S. 2009. Counting belief propagation. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*, 277–284.

Kersting, K. 2012. Lifted probabilistic inference. In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI)*, 27–31.

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 (AAAI)*, 1062–1608.

Poole, D., and Zhang, N. L. 2003. Exploiting contextual independence in probabilistic inference. *J. Artif. Intell. Res. (JAIR)* 18:263–313.

Poole, D.; Bacchus, F.; and Kisynski, J. 2011. Towards completely lifted search-based probabilistic inference. *CoRR* abs/1107.4035.

Poole, D. 2003. First-order probabilistic inference. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, 985–991.

Poole, D. 2011. Logic, probability and computation: Foundations and issues of statistical relational AI. In *Proceedings of the 11th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, 1–9.

Singla, P., and Domingos, P. 2008. Lifted first-order belief propagation. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI)*, 1094–1099.

Taghipour, N., and Davis, J. 2012. Generalized counting for lifted variable elimination. In *Proceedings of the 2nd International Workshop on Statistical Relational AI (StaRAI)*.

Taghipour, N.; Fierens, D.; Davis, J.; and Blockeel, H. 2012. Lifted variable elimination with arbitrary constraints. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 1194–1202.

Taghipour, N.; Fierens, D.; Davis, J.; and Blockeel, H. 2013a. Lifted variable elimination: Decoupling the operators from the constraint language. *Journal of Artificial Intelligence Research* 47.

Taghipour, N.; Fierens, D.; Van den Broeck, G.; Davis, J.; and Blockeel, H. 2013b. Completeness results for lifted variable elimination. In *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

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 *Proceedings of the 24th Annual Conference on Advances in Neural Information Processing Systems (NIPS)*, 1386–1394.

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