Exploiting Logical Structure in Lifted Probabilistic Inference

Vibhav Gogate and **Pedro Domingos**
Computer Science & Engineering
University of Washington
Seattle, WA 98195, U.S.A.
{*vgogate,pedrod*}@*cs.washington.edu*

Abstract

Representations that combine first-order logic and probability have been the focus of much recent research. Lifted inference algorithms for them avoid grounding out the domain, bringing benefits analogous to those of resolution theorem proving in first-order logic. However, all lifted probabilistic inference algorithms to date treat potentials as black boxes, and do not take advantage of their logical structure. As a result, inference with them is needlessly inefficient compared to the logical case. We overcome this by proposing the first lifted probabilistic inference algorithm that exploits determinism and context specific independence. In particular, we show that AND/OR search can be lifted by introducing POWER nodes in addition to the standard AND and OR nodes. Experimental tests show the benefits of our approach.

Introduction

Recent years have seen much interest in using first-order logic to compactly specify large probabilistic models (Getoor and Taskar 2007; Domingos and Lowd 2009). For example, Markov logic uses a set of weighted first-order formulas to define a Markov network, with a feature for each possible grounding of each formula with objects in the domain (Domingos and Lowd 2009). Initially, only the representation of a model benefited from the compactness of first-order logic; inference was still carried out by grounding out the entire network and applying standard techniques. However, since the size of the network is exponential in the number of objects, this seriously limited scalability.

More recently, a number of techniques for *exact lifted inference* in first-order probabilistic models have been proposed (Poole 2003; Milch et al. 2005; Braz, Amir, and Roth 2005; 2006). Lifted inference treats sets of objects that are indistinguishable given the evidence as single units, and can provide very large speedups, analogous to those obtainable by resolution in first-order logic. The only approach to date is first-order variable elimination (FOVE) (Poole 2003; Braz, Amir, and Roth 2005; 2006). FOVE, like propositional variable elimination, has exponential space complexity, which limits its applicability to realistic domains. Also, it does not take advantage of the structure of the logical formulas to define potentials, treating them as black boxes. Since large cliques are very common in first-order probabilistic models, this can make FOVE impractical even in cases where efficient inference is otherwise possible. In particular, one can take advantage of the local structure in the models specified using logical formulas such as context-specific independence (Boutilier 1996) and determinism.

The main contribution of this paper is the specification of a lifted inference scheme that uses the mechanics of search, Boolean constraint propagation and pruning, to better exploit the local structure of the formulas. In particular, we define the *AND/OR/POWER search space* for lifted graphical models, which adapts and extends the recently introduced AND/OR search spaces for propositional graphical models (Dechter and Mateescu 2007). The AND/OR search space contains two types of nodes: AND nodes which correspond to problem decomposition and OR nodes which correspond to conditioning. The AND/OR/POWER search space has two additional node types, which capture properties of lifted inference: POWER-AND nodes which aggregate decomposition and POWER-OR nodes which aggregate conditioning, over groups of random variables.

We begin by briefly reviewing background on Markov logic and AND/OR search spaces. We then describe our new framework of AND/OR/POWER search spaces for performing lifted inference in Markov logic. Finally, we present our experimental results and conclude.

Background

We denote first order logical variables by small letters x and y. Constants are denoted by capital letters, e.g., X, Y, A. A predicate or an atom defined over constants or variables, or both is denoted by letters R, S and T, e.g., $R(x, A)$, $S(x, y, z)$ and $T(B)$. Each variable x can take values from some domain Δ_x of constant symbols. Sets are represented by bold letters e.g. $\mathbf{x} = \{x_1, \ldots, x_n\}$ is a set of variables and $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a set of constants.

We will consider first order Knowledge Bases (KBs) which are in conjunctive normal form (CNF). Each formula in a CNF is a clause in which all variables are universally quantified, where a clause is a disjunction over predicates, constants or their negation. A predicate or a formula is grounded by replacing all its variables by constants. Propositionalization or grounding is the process of replacing a first-order KB by an equivalent propositional formula. A

Clauses	Weight	Domains
$R(x,y) \vee \neg S(x)$	1.4	$\Delta_x = \{A,B\}$
$S(x) \vee T(x,z)$	1.1	$\Delta_y = \{A,B\}$
		$\Delta_z = \{A,B\}$

Figure 1: Example Markov Logic network.

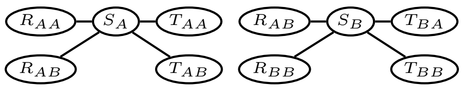

Figure 2: Primal Graph of a PropMRF obtained by grounding the MLN shown in Figure 1.

first-order KB is said to be satisfiable if its equivalent propositional KB is satisfiable. A first-order KB represents hard constraints over the domain. Any truth assignment to the grounded propositional variables that does not satisfy one or more clauses is invalid or has zero probability. The basic idea in Markov logic networks is to soften these constraints by associating weights with the formulas.

Before defining Markov Logic networks (MLNs), we introduce propositional MRFs or PropMRF in short. A PropMRF, denoted by M, is a set of pairs (F_i, w_i) where F_i is a propositional clause and w_i is a real number. Let $\bar{X}$ be an assignment to all variables in M, then the probability distribution represented by M is given by: $\Pr(\bar{X}) = \frac{1}{Z} \exp\left(\sum_i w_i F_i(\bar{X}_{V(F_i)})\right)$ where $Z = \sum_{\bar{X}} \exp\left(\sum_i w_i F_i(\bar{X}_{V(F_i)})\right)$ is called the **partition function**, $\bar{X}_{V(F_i)}$ is the projection of the full assignment $\bar{X}$ to the variables $V(F_i)$ of F_i and $F_i(\bar{X}_{V(F_i)})$ is a function which is 1 if $\bar{X}_{V(F_i)}$ evaluates F_i to True and 0 otherwise. The **primal graph** of a PropMRF has variables as its vertices and an edge between any two nodes that are involved in the same clause. It is known that any Bayesian or Markov network can be encoded as a PropMRF, such that the probability distributions represented by the two are the same (Chavira and Darwiche 2008).

DEFINITION **1** (**Markov Logic networks**). *A Markov logic network (MLN) L is a set of pairs (F_i, w_i) where F_i is a clause in first order logic and w_i is a real number. Given a finite set of constants $\mathbf{X} = \{X_1, \ldots, X_m\}$, it defines a PropMRF $M_{L,\mathbf{X}}$ as follows. $M_{L,\mathbf{X}}$ contains a propositional variable for each possible grounding of each predicate appearing in L. $M_{L,\mathbf{X}}$ contains a weighted propositional clause, with weight w_i, for each possible grounding of each weighted formula F_i in L.*

EXAMPLE **1.** *Figure 1 shows a MLN defined by three predicates R, S and T and two weighted first order logic clauses. The primal graph of the ground MLN is shown in Figure 2.*

Important queries over a MLN L are computing the partition function of $M_{L,\mathbf{X}}$ and computing the probability of a formula given evidence, where evidence is an instantiated subset of variables in $M_{L,\mathbf{X}}$. Because the latter query type is a special case of the former, in the sequel, we define inference algorithms for computing the partition function only.

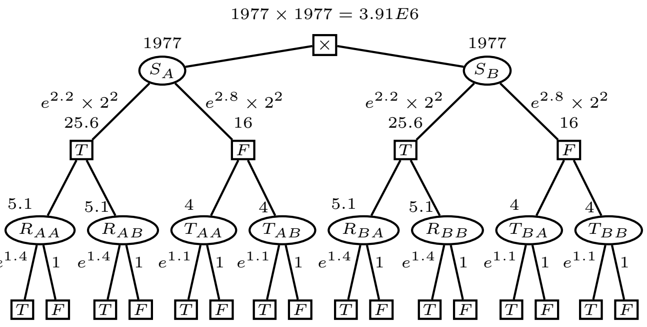

Figure 3: AND/OR search space for the MLN shown in Figure 2. For brevity, variable $S(A)$ is abbreviated as S_A, similarly variable $T(A, B)$ is abbreviated as T_{AB} etc.

AND/OR search spaces for Ground MLNs

The partition function of a MLN can be computed by applying any known probabilistic inference scheme such as variable elimination (Zhang and Poole 1994) or recursive conditioning (Darwiche 2001) over its ground PropMRF. These algorithms are instances of performing a depth-first search with or without caching over the AND/OR search space for graphical models (Dechter and Mateescu 2007). Since the PropMRF is specified using logical formulas, it lends itself naturally to various *Boolean constraint propagation and pruning techniques* available in the literature (Sang, Beame, and Kautz 2005; Chavira and Darwiche 2008). We will therefore deviate slightly from the original description of AND/OR search spaces in (Dechter and Mateescu 2007), to account for Boolean constraint propagation and pruning.

Given a PropMRF M and its primal graph G, the associated AND/OR search tree, has alternating levels of AND and OR nodes, starting with a dummy AND node $\times$ (see Figure 3 for an example in which the AND nodes are squares and OR nodes are ovals). Each node is associated with a current PropMRF. The PropMRF associated with the root AND node $\times$ is M. The OR nodes correspond to the variables. Each OR node has two or more child AND nodes corresponding to the value assignments from the domains of the variables. Each AND node has one or more child OR nodes (except leaf AND nodes which have no children), where each child OR node corresponds to a variable from a unique connected component of the primal graph of the PropMRF associated with its parent. Semantically, the OR nodes represent *conditioning* whereas the AND nodes represent *problem decomposition*.

The PropMRFs associated with child OR nodes $\{n_1, \ldots, n_k\}$ of a parent AND node n are obtained as follows. Let $M_{n,1}, \ldots, M_{n,k}$ be the (disjoint) components of the PropMRF M_n (corresponding to the dis-connected components of the primal graph of M_n) associated with n, then $M_{n,i}$ is the PropMRF associated with n_i. The PropMRF at an AND node n is obtained by adding its corresponding value assignment to the PropMRF M_p at its parent OR node p and simplifying M_p using Boolean constraint propagation and pruning. We illustrate it via the following example.

EXAMPLE **2.** *Consider the primal graph of the ground MLN* *shown in Figure 2. A possible AND/OR search space for this problem is shown in Figure 3. The root AND node $\times$ has two child OR nodes $S(A)$ and $S(B)$, corresponding to a variable from each component of the primal graph shown in Figure 2. After we condition on $S(A) = True$, we can remove the clauses containing $T(A,A)$ and $T(A,B)$, because they evaluate to True. Also, we can simplify the clauses containing $R(A,A)$ and $R(A,B)$ by removing the literal $S(A)$ from them. Thus, the `PropMRF` associated with the AND node corresponding to $S(A) = True$ contains only two weighted clauses $(R(A,A), 1.4)$ and $(R(A,B), 1.4)$, which are disjoint. We can decompose this `PropMRF` into two components: $R(A,A)$ and $R(A,B)$, and so on, yielding the AND/OR search tree given in Figure 3.*

Given an AND/OR search space for a `PropMRF` M, it was pointed out in (Dechter and Mateescu 2007) that the partition function can be computed by labeling the AND/OR tree appropriately and recursively computing the value of all nodes from the leaves to the root, defined next.

DEFINITION **2** (**Labels and Values in an AND/OR tree**). *Each arc emanating from an OR node n to an AND node m is associated with a* **label**, *denoted by $l(n,m)$. Let M_n and M_m be the `PropMRF` associated with n and m respectively. Let $(F_1, w_1), \ldots, (F_k, w_k)$ be the clauses that evaluate to True in M_n because of the assignment corresponding to m and let L be the number of variables that are present in M_n but not in M_m and which are not assigned to either True or False (either because of the current assignment or because of Boolean constraint propagation). Then, $l(n,m) = \exp\left(\sum_{i=1}^{k} w_i\right) \times 2^L$. The* ***value*** *of a node n, denoted by $v(n)$, is defined recursively from the leaves to the root as follows. Let $chi(n)$ be the set of child nodes of n. If n is a leaf AND node then $v(n) = 1$. If n is an internal AND node, then $v(n) = \prod_{i \in chi(n)} v(i)$. If n is an OR node, then $v(n) = \sum_{m \in chi(n)} v(m) \times l(n,m)$. The value of the root AND node is equal to the partition function.*

EXAMPLE **3**. *Figure 3 shows an AND/OR search tree in which each OR to AND arc is annotated with labels derived from the clauses shown in Figure 1. Each node is annotated with its value. The partition function is the value of the root node:* 3.9×10^6.

AND/OR/POWER Search for MLNs

Just as in lifted variable elimination, the main idea in AND/OR/POWER search is to exploit the symmetry present in the ground `PropMRF`, and to do so without actually constructing it. We consider two such symmetry properties defined previously: inversion (Poole 2003; Braz, Amir, and Roth 2006) and the counting argument (Braz, Amir, and Roth 2005), and define them relative to the AND/OR search space. We begin with an example that illustrates inversion.

EXAMPLE **4**. *Consider the AND/OR search tree shown in Figure 3. The value of the OR node $S(A)$ is same as the value of the OR node $S(B)$. Generalizing, if the domain of x was $\{X_1, \ldots, X_k\}$ instead of $\{A, B\}$, then the root node in Figure 3 would have k children, all of which would have*

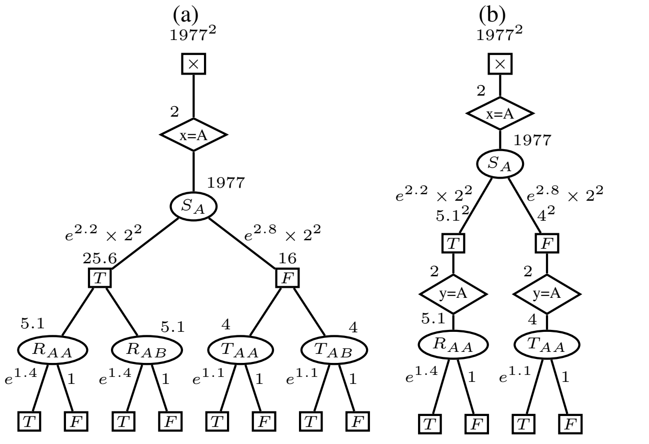

Figure 4: Augmenting the AND/OR search space with POWER-AND nodes.

the same value. Thus, instead of computing each of them separately, we could just compute one of them and raise it to an appropriate power. Figure 4(a) shows how this can be achieved by introducing a POWER-AND node (x=A), denoted by a diamond. The POWER-AND node takes the value that it receives from its child OR node, raises it to its associated power-count of 2 and then passes this new value to its parent AND node. Notice that $R(A,B)$ and $R(A,A)$ (similarly $T(A,A)$ and $T(A,B)$) have the same value as well. Thus, we could introduce POWER-AND nodes there too, yielding the search space shown in Figure 4(b).

Next, we illustrate how to apply the counting argument (Braz, Amir, and Roth 2005) over an AND/OR search tree.

EXAMPLE **5**. *Consider a MLN having two clauses $S(x)$ and $\neg S(x)$ with weights w_1 and w_2 respectively. Let $\Delta_x = \{A, B, C\}$. The OR tree (an AND/OR tree in which each AND node has at most one child OR node) corresponding to the ground MLN is shown in Figure 5(a) (note that the example is for illustration purposes only. We could also apply inversion here, which will be more efficient). We can transform this OR tree into an equivalent OR tree having just 8 AND nodes by clustering the variables together as shown in Figure 5(b). Each AND node in Figure 5(b) corresponds to one of the 8 leaf AND nodes in Figure 5(a). The arc-label of each arc on the new OR tree in Figure 5(b) is simply the product of the arc-labels along the path from the root to the corresponding leaf node in Figure 5(a).*

By definition, the value of the OR node denoted by n in the OR tree of Figure 5(b) is given by: $v(n) = \sum_{j \in chi(n)} v(j) \times l(n,j)$. The AND nodes marked in double (and triple) boxes in Figure 5(b) are identical in the sense that they have the same arc-labels. In general, if x has N constants in its domain, then the arc-label would equal w_1^K and w_2^{N-K} where K is the number of groundings of $S(x)$ that are True in the corresponding assignment. If all the AND nodes that have the same arc-label also have the same value, then we can group them together as we show next. Let $J_n(K) = \{j \in chi(n) | l(n,j) = e^{K.w_1} \times e^{(K-m).w_2}\}$ for $K = 0$ to N. It is easy to see that $|J_n(K)| = \binom{N}{K}$. If $v(i) = v(j)$ for all

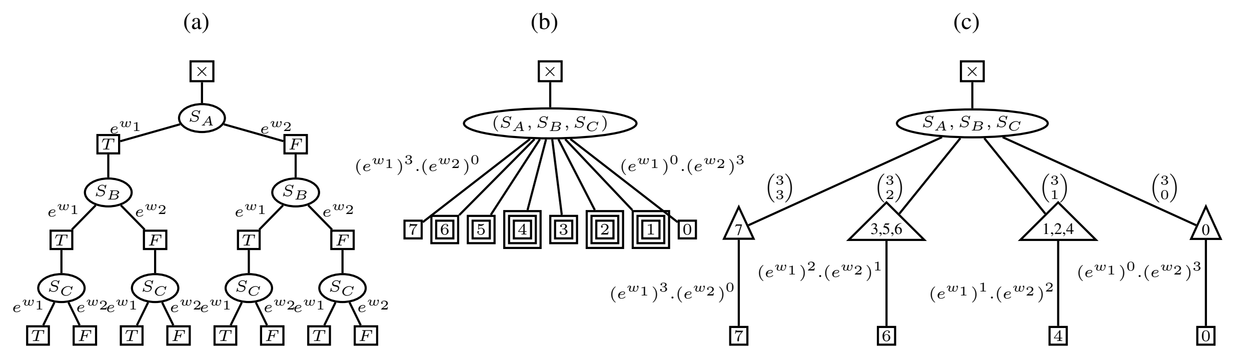

Figure 5: (a) OR search tree for a ground MLN with two clauses $S(x)$ and $\neg S(x)$ having weights w_1 and w_2 respectively. The domain of x is $\{A, B, C\}$. (b) Equivalent OR search tree to the tree shown in (a). Some arc-labels are not shown for brevity. Note that the leaf node "0" corresponds to the assignment (F, F, F) to $S(A), S(B), S(C)$, "1" corresponds to (F, F, T) and so on. (c) Tree obtained by aggregating similar OR nodes in the tree shown in (b) using POWER-OR nodes (triangles).

$i, j \in J_n(K)$, then we can rewrite $v(n)$ as:

$$v(n) \;=\; \sum_{K=0}^{N} \binom{N}{K} \times v(j \in J_n(K)) \times e^{K.w_1+(N-K).w_2}$$

Graphically, we can represent the above counting argument using $N + 1$ POWER-OR nodes, where the K^{th} node aggregates the information over all the child AND nodes contained in the partition $J_n(K)$. Figure 5(c) shows how POWER-OR nodes, denoted by a triangle can be used to compactly represent the OR tree given in Figure 5(b).

Next, we will formalize the two symmetries over AND/OR search trees yielding AND/OR/POWER search trees. We start with a required definition.

DEFINITION 3 (**Domain and Range of a Predicate given a set of constraints**). *The domain of a predicate R, denoted by Δ_R, is a subset of the Cartesian product of the domains of its logical variables that satisfy the given set of constraints. The range of a predicate R, denoted by Γ_R is the set of truth assignments over all elements in its domain.*

Note that the constraints over the domain of a predicate can be introduced because of evidence or because of conditioning while performing inference. We explain this issue in the next subsection (handling evidence and shattering).

EXAMPLE 6. *Given a predicate $R(x, y, z)$, $\Delta_x = \Delta_y = \Delta_z = \{A, B, C\}$ and a constraint $x \neq y \neq z$, the domain of $R(x, y, z)$ is $\{(A, B, C), (A, C, B), (B, A, C), (B, C, A), (C, A, B), (C, B, A)\}$ while its range is all possible truth assignments to all variables in its domain. Thus, the range of $R(x, y, z)$ has $2^6 = 64$ elements.*

The AND/OR/POWER search space consists of alternating levels of AND, POWER-AND, OR and POWER-OR nodes, starting with a dummy AND node $\times$. Each node n maintains a current MLN L_n obtained by applying its corresponding operator to the MLN L_p at its parent p. Each predicate R in the current MLN L_n is associated with a set of constraints C_R on its domain. The operators for each node type are defined as follows.

Let n be an AND node and R be a predicate in L_n. Let $\Delta_{R,1}, \ldots, \Delta_{R,K}$ $(K \geq 1)$ be a partition of the domain Δ_R of R subject to the constraints C_R such that $\forall i, j\ i \neq j$, given $\vec{X}_{R,i} \in \Delta_{R,i}$ and $\vec{X}_{R,j} \in \Delta_{R,j}$, the propositional variables $R(\vec{X}_{R,i})$ and $R(\vec{X}_{R,j})$ lie in different components of the primal graph of the ground Markov network of L_n. Let $\Delta_{R,m,1}, \ldots, \Delta_{R,m,L}$ be a partition of $\Delta_{R,m}$ such that $\forall i, j\ i \neq j$, given $\vec{X}_{R,m,i} \in \Delta_{R,m,i}$ and $\vec{X}_{R,m,j} \in \Delta_{R,m,j}$, the propositional variables $R(\vec{X}_{R,m,i})$ and $R(\vec{X}_{R,m,j})$ lie in different components of the primal graph of the ground Markov network of L_n and the MLN $\theta_{R,m,i}L_n$ is equivalent (isomorphic) to the MLN $\theta_{R,m,j}L_n$ subject to a renaming of the variables and constants, where $\theta_{R,m,i}$ is a substitution that summarizes the partition $\Delta_{R,m,i}$ and $\theta_{R,m,i}L_n$ is the MLN obtained by applying $\theta_{R,m,i}$ to L_n. $\theta_{R,m,i}L_n$ can be obtained from L_n by **unifying** $\theta_{R,m,i}$ with all predicates that share a formula with R and updating their domains accordingly. Then, n has K child POWER-AND nodes $n_1, \ldots, n_K$, where each node n_m is associated with the MLN $\theta_{R,m,i}L_n$ and the substitution $\theta_{R,m,i}$.

Let n be a POWER-AND node and R be a predicate in L_n. The child node of n is an OR node m corresponding to R and L_m is same as L_n.

Let n be an OR node corresponding to a predicate R. Let $\Gamma_{R,1}, \ldots, \Gamma_{R_K}$ be a partition of the range Γ_R of R subject to the constraints C_R such that given $\bar{X}_{R,i,a} \in \Gamma_{R,i}$, $\bar{X}_{R,i,b} \in \Gamma_{R,i}$, $\bar{X}_{R,i,a} \neq \bar{X}_{R,i,b}$, the MLN $L_{n,\bar{X}_{R,i,a}}$ is equivalent (isomorphic) to $L_{n,\bar{X}_{R,i,b}}$, where $L_{n,\bar{X}_{R,i,a}}$ is a MLN obtained by adding the assignments corresponding to $\bar{X}_{R,i,a}$ to L_n. Then, n has K child POWER-OR nodes $n_1, \ldots, n_K$ where each node n_m is associated with the MLN L_n and the partition $\Gamma_{R,m}$.

Let n be a POWER-OR node and $\bar{X}$ be an assignment from the partition Γ_n of the range of R associated with n. The child node of n is an AND node m corresponding to $\bar{X}$. The MLN L_m is obtained by adding the assignments corresponding to $\bar{X}$ to L_n and simplifying, using for example, *Boolean constraint propagation and pruning*.

The partition function of the MLN can be computed by defining arc and node labels over the AND/OR/POWER search space and then performing value computations.

DEFINITION 4 (**Arc label, node label and value of a node**). *The AND/OR/POWER search space has two types of arc-labels, two types of node labels and a value associated with each node. The label of a node n is denoted by $s(n)$ and its value is denoted by $v(n)$. The arc-label of an arc from a node n to a node m is denoted by $l(n, m)$.*

Each arc from an OR node n to its child POWER-OR node m is labeled with the number of ground OR arcs that it corresponds to. These can be derived using multi-nomial coefficients based on the size of the range of the predicate associated with n, see for example (Braz, Amir, and Roth 2005; Ng, Lloyd, and Uther 2008). Each arc from the POWER-OR node n to the AND node m is labeled with a pair $\langle p_t, p_f\rangle$ where p_t and p_f are the exponentiated sum of the weights of first-order clauses satisfied by assigning the predicate associated with n to True and False respectively.

Each POWER-AND node is labeled with the number of solutions of the constraints on the substitution that it represents, which in turn equals the number of ground AND arcs that it infers across. Each POWER-OR node n is labeled with a pair $\langle n_t, n_f\rangle$, where n_t and n_f are the number of propositional variables that are assigned to True and False respectively, where each propositional variable corresponds to a possible grounding (namely an element of the domain) of the predicate associated with n.

The value of a node is defined recursively from the leaves to the root as follows. The value of all leaf AND nodes is initialized to 1. The value of an AND node n is given by: $v(n) = \prod_{i\in chi(n)} v(i)$. The value of a POWER-AND node i is the value of its child OR node j raised by its labeled power-count, namely $v(i) = v(j)^{s(i)}$. The value of a POWER-OR node i labeled by $s(i) = \langle n_t, n_f\rangle$ having a child AND node j, such that $l(i, j) = \langle p_t, p_f\rangle$ is given by $v(i) = v(j) \times p_t^{n_t} \times p_f^{n_f}$. The value of a OR node i is given by $v(i) = \sum_{j\in chi(i)} v(j) \times l(i, j)$.

We can prove that the value of the root AND node $\times$ equals the partition function of the MLN. The proof follows from the formal properties of the inversion and counting arguments given in (Braz, Amir, and Roth 2005; Ng, Lloyd, and Uther 2008), and the correctness of the AND/OR search space transformations. In summary,

THEOREM 1. *Given an AND/OR/POWER search space for a MLN L, with its nodes and arcs labeled according to Definition 4, the value of the root AND node computed using Definition 4 is equal to the partition function of L.*

The partition function can be computed by using only linear space, by traversing the AND/OR/POWER search space in depth-first manner.

Handling Evidence and Shattering

In this subsection, we discuss the need for shattering (Braz, Amir, and Roth 2005; 2006) and handling evidence, where evidence is a truth assignment to a set of ground predicates. We demonstrate the ideas using the following example:

EXAMPLE 7. *Consider the MLN given in Figure 1. If we change the domains to $\Delta_x = \Delta_y = \Delta_z = \{A, B, C, D\}$, then the AND/OR/POWER search space shown in Figure 4(c) (with the power-counts changed to 4) can still be used to compute the partition function. However, given evidence $S(A) = True$, we cannot use it because the value received from the child OR node corresponding to $S(A)$ in the AND/OR search tree will be different from the values received from the other three child OR nodes. The solution in such cases is to* ***shatter*** *the MLN. The idea in shattering is to split the domain (or range) of $S(x)$ into multiple partitions, such that the inversion or the counting arguments can be applied to each partition. In our example, given $S(A) = True$, we can split the domain of $S(x)$ into two partitions: $\{A\}$ and $\{B, C, D\}$ and create POWER-AND nodes for each partition. Alternatively, we can represent this split by enforcing constraints on the domain of $S(x)$, namely $S(x)$ subject to $x = A$, and $S(x)$ subject to $x \neq A$.*

Note that shattering may be needed, even when no evidence is present. One could apply shattering over the full MLN in advance or just shatter the MLN w.r.t. the predicate that is chosen to be conditioned next. The latter approach, which is also called as *splitting as needed*, was shown to be more efficient (Kisynski and Poole 2009).

Evaluation

We compared the performance of AND/OR/POWER search with an implementation of FOVE in BLOG (available at http://people.csail.mit.edu/milch/blog/index.html) on the following problem, which is parameterized with five integer constants: n, m, s, e and c. We assume that we have just one typed logical variable $\{x\}$ that has c values in its domain. We have n predicates $R_i(x)$ for $i = 1$ to n. We generate m clauses by randomly selecting s predicates and negating each with equal probability. A clause over s predicates $\{R_1(x), \dots, R_s(x)\}$ has the following form: $R_1(x) \vee \dots \vee R_s(x)$. For each random problem, we choose e ground atoms as evidence, each of which is set to either True or False with equal probability.

For our experiments, we set $n = m$ and tried three values for them $\{30, 40, 50\}$. We varied the size of the clauses from 3 to 9 in increments of 2. Figure 6 shows the impact of increasing the clause size s on the average performance of FOVE and AND/OR/POWER search. The time is averaged over 10 problems generated using the random model with parameters: $e = 10$, $c = 50$, with s and $n = m$ varied. Each algorithm was given a time-bound of 20 minutes. From Figure 6, we can see that for small clause size $s = 3$, the time required by FOVE is much smaller than AND/OR/POWER search. However, as s increases beyond 5, AND/OR/POWER search is much faster than FOVE, which times out on all instances. This is because large clauses aid in Boolean constraint propagation and pruning, which FOVE does not take advantage of. For small clause sizes, however, Boolean constraint propagation and pruning does not help and FOVE is much faster.

Summary and Discussion

To summarize, we presented a framework called AND/OR/POWER search spaces, for performing lifted search in first order graphical models. The

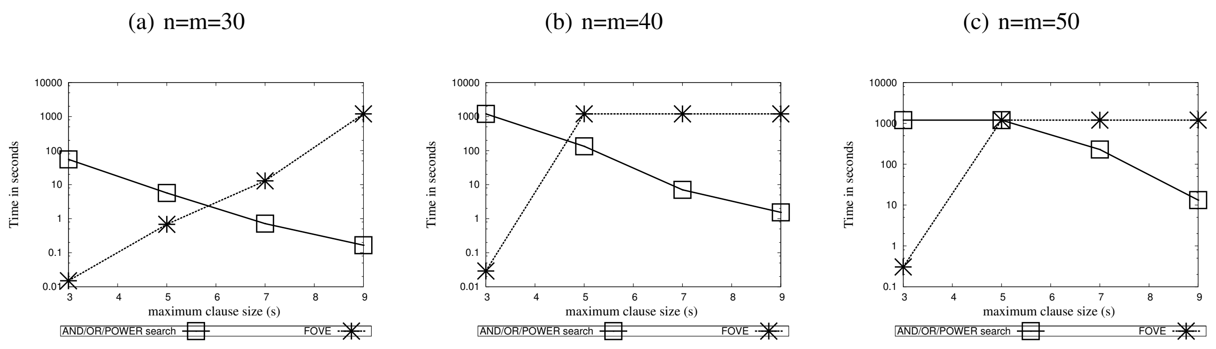

Figure 6: Figure showing the impact of increasing problem size and clause size on AND/OR/POWER search and FOVE on the problem generated using the random model with parameters: (n=m $\in \{30, 40, 50\}$, e=10,c=50, s is varied). Timeout used was 1200s.

AND/OR/POWER search space augments the usual OR (sum) and AND (product) nodes in the AND/OR search spaces for graphical models with the (lifted) POWER-AND and POWER-OR nodes that reason across a group of random variables. Performing a depth-first search traversal of this space yields the first lifted search scheme to date that can harness the power of logical techniques to better exploit the local structure of the formulas.

The specification of AND/OR/POWER search space is also important in the context of approximate inference because it lends itself immediately to *lifted importance sampling* (Gogate and Domingos 2010), a first lifted sampling approach to date. Lifted importance sampling can be understood as a partial exploration of the AND/OR/POWER search space, just as importance sampling can be understood as a partial exploration of the AND/OR search space (Gogate and Dechter 2008). Because the AND/OR/POWER search space is more compact, we can show that the estimates generated by lifted importance sampling will have smaller variance than the ones generated by first grounding the MLN, and then performing importance sampling over it.

Our work can be extended in several ways. For example, we could AND/OR/POWER space to perform propositional inference as well, by identifying and encoding symmetries in propositional graphical models, similar to the way in which lifted BP (Singla and Domingos 2008) was generalized over over propositional models (Kersting, Ahmadi, and Natarajan 2009). Finally, an interesting future work is to develop *lifted caching* approaches similar to those developed for propositional models (Sang, Beame, and Kautz 2005; Dechter and Mateescu 2007).

Acknowledgements

This research was partly funded by ARO grant W911NF-08-1-0242, AFRL contract FA8750-09-C-0181, DARPA contracts FA8750-05-2-0283, FA8750-07-D-0185, HR0011-06-C-0025, HR0011-07-C-0060 and NBCH-D030010, NSF grants IIS-0534881 and IIS-0803481, and ONR grant N00014-08-1-0670. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ARO, DARPA, NSF, ONR, or the United States Government.

References

Boutilier, C. 1996. Context-specific independence in Bayesian networks. In *UAI*, 115–123.

Braz, R. d. S.; Amir, E.; and Roth, D. 2005. Lifted first-order probabilistic inference. In *IJCAI*, 1319–1325.

Braz, R. d. S.; Amir, E.; and Roth, D. 2006. MPE and Partial Inversion in Lifted Probabilistic Variable Elimination. In *AAAI*, 1123–1130.

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

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

Dechter, R., and Mateescu, R. 2007. AND/OR search spaces for graphical models. *Artificial Intelligence* 171(2-3):73–106.

Domingos, P., and Lowd, D. 2009. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan and Claypool.

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

Gogate, V., and Dechter, R. 2008. AND/OR Importance Sampling. In *UAI*, 212–219.

Gogate, V., and Domingos, P. 2010. Exploiting logical structure in lifted probabilistic inference. Technical report, University of Washington, Seattle.

Kersting, K.; Ahmadi, B.; and Natarajan, S. 2009. Counting Belief Propagation. In *UAI*, 277–284.

Kisynski, J., and Poole, D. 2009. Constraint processing in lifted probabilistic inference. In *UAI*, 293–302.

Milch, B.; Marthi, B.; Russell, S. J.; Sontag, D.; Ong, D. L.; and Kolobov, A. 2005. Blog: Probabilistic models with unknown objects. In *IJCAI*, 1352–1359.

Ng, K. S.; Lloyd, J. W.; and Uther, W. T. 2008. Probabilistic modelling, inference and learning using logical theories. *Annals of Mathematics and Artificial Intelligence* 54(1-3):159–205.

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

Sang, T.; Beame, P.; and Kautz, H. 2005. Solving Bayesian networks by weighted model counting. In *AAAI*, 475–482.

Singla, P., and Domingos, P. 2008. Lifted first-order belief propagation. In *AAAI*, 1094–1099.

Zhang, N., and Poole, D. 1994. A simple approach to bayesian network computations. In *Proceedings of the Tenth Canadian Conference on Artificial Intelligence*, 171–178.