The Workshops of the Thirtieth AAAI Conference on Artificial Intelligence
Beyond NP: Technical Report WS-16-05

Toward Caching Symmetrical Subtheories for Weighted Model Counting

Tim Kopp
University of Rochester
Rochester, NY

Parag Singla
Indian Institute of Technology
Hauz Khas, New Delhi

Henry Kautz
University of Rochester
Rochester, NY

Abstract

Model counting and weighted model counting are key problems in artificial intelligence. Marginal inference can be reduced to model counting in many statistical-relational systems, such as Markov Logic. One common approach used by model counters is splitting a theory into disjoint subtheories, performing model counting on the subtheories, and then caching the result. If an identical subtheory is encountered again in the search, the cached result is used, greatly reducing runtime. In this work we introduce a way to cache symmetric subtheories compactly, which could potentially decrease required cache size, increase cache hits, and decrease runtime of solving.

Introduction

Given a clausal propositional theory, the problem of model counting is the problem of determining the number of assignments to the variables of the theory that satisfy it. In the weighted variety of the problem, weights are assigned to the the clauses (or predicates) of the theory, the weight of a possible world is the sum of the weights of the satisfied clauses (or weights of the literals assigned to true), and the weighted model counting problem is to find the sum of the weights of the satisfying possible worlds. Model counting and its variants are #P-complete problems, as it is the counting version of the SAT problem.

Model counting and its weighted variants are important problems in artificial intelligence, as many important problems we want to solve can be reduced to model counting. In probabilistic graphical models, such as Markov networks, the problem of exact marginal inference can be reduced to weighted model counting. Model counting can therefore also be used to solve marginal inference for other representations of graphical models, such as statistical-relational learning systems like Markov Logic (Richardson and Domingos 2006).

It is these systems in particular from which we draw inspiration. Although model counting, as stated, is a problem over propositional theories, it is often the case that the problems people typically want to solve are more easily expressed in a higher level language, such as first order logic. The input to the model counter is created through the process of propositionalizing (or grounding) the first order representation of the problem.

In propositional theories, it is possible to detect and leverage symmetries between variables to reduce solving and counting time. However, the number of propositional symmetries may be exponential, so using them effectively typically involves heuristically choosing which symmetries to account for. In first order theories, however, it is possible to compactly represent many symmetries and exploit them. This is the common thread that runs through many lifted inference algorithms for both marginal and MPE inference, such as e.g. (Mittal et al. 2014), (Singla and Domingos 2008) and many more.

In this paper, we will focus on symmetries over terms, that is, constants in the theory. Specifically, we will exploit the class of term equivalent symmetries, which were introduced in various forms in earlier works such as (Poole 2003; Bui, Huynh, and Riedel 2013). Two terms are term equivalent if they may be swapped without altering the theory. These symmetries can be efficiently detected in evidence (Kopp, Singla, and Kautz 2015), a quality that makes them especially well-suited to the type of problems typically formulated in systems like Markov Logic.

The main contributions of this paper are threefold: first, we introduce a canonical form, a way of representing many symmetric theories compactly, second we give a procedure for putting a theory in canonical form, and lastly, we describe a method for using the canonical form to exploit symmetries in model counting.

Related Work

Our work has connections to research in in both the machine learning and constraint-satisfaction research communities. Developments include lifted versions of variable elimination (Poole 2003; de Salvo Braz, Amir, and Roth 2005), belief propagation (Singla and Domingos 2008; Singla, Nath, and Domingos 2014), and DPLL (Gogate and Domingos 2011). The approach of defining symmetries using group theory and detecting them by graph isomorphism is shared by Bui *et al.*'s work on lifted variational inference (Bui, Huynh, and Riedel 2013). Niepert gives a group-theoretic formalism of symmetries at the level of constants (Niepert 2012; 2013), applying them to MCMC methods. Kopp et. al. define the class of symmetries we exploit in this work (Kopp, Singla, and Kautz 2015). Bui notes that symmetry groups can be defined on the basis

of *unobserved* constants in the domain, while the symmetries we exploit can be explicitly found in the evidence. Two lines of work in SRL make use of problem transformations. First-order knowledge compilation (Van den Broeck 2013; den Broeck, Meert, and Darwiche 2014) transforms a relational problem into a form for which MPE, marginal, and MAP inference is tractable. Recent work on MAP inference in Markov Logic has identified special cases where a relational formula can be transformed by replacing a quantified formula with a *single* grounding of the formula (Mittal et al. 2014).

The literature surrounding the use of symmetries in constraint satisfaction and model counting is quite extensive. Here we give just a few examples of key developments in the field. Symmetry detection has been based either on graph isomorphism on propositional theories as in the original work by by Crawford *et. al* (Crawford et al. 1996); by interchangeability of row and/or columns in CSPs specified in matrix form (Meseguer and Torras 2001); by checking for other special cases of geometric symmetries (Sellmann and Hentenryck 2005), or by determining that domain elements for a variable are exchangeable (Audemard, Benhamou, and Henocque 2006). Researchers have suggested symmetry-aware modifications to backtracking CSP solvers for variable selection, branch pruning, and no-good learning (Meseguer and Torras 2001; Flener et al. 2009). A recent survey of symmetry breaking for CSP (Walsh 2012) described alternatives to the lex-leader formulation of SBPs, including one based on Gray codes.

Symmetries over terms

In this section we provide the background necessary to understand the types of symmetries we will exploit in the subsequent sections. Symmetry-breaking for satisfiability testing, introduced by Crawford et. al.(Crawford et al. 1996), is based on concepts from group theory. A *permutation* σ is a mapping from a set L to itself. A permutation group is a set of permutations that is closed under composition and contains the identity and a unique inverse for every element. A literal is an atom or its negation. A clause is a disjunction over literals. A CNF theory $\mathcal{T}$ is a set (conjunction) of clauses. Let L be the set of literals of $\mathcal{T}$. We consider only permutations that respect negation, that is $\sigma(\neg l) = \neg\sigma(l)$ ($l \in L$). The *action* of a permutation on a theory, written $\sigma(\mathcal{T})$, is the CNF formula created by applying σ to each literal in $\mathcal{T}$. We say σ is a *symmetry* of $\mathcal{T}$ if it results in the same theory i.e. $\sigma(\mathcal{T}) = \mathcal{T}$.

A model M is a truth assignment to the atoms of a theory. The action of σ on M, written $\sigma(M)$, is the model where $\sigma(M)(P) = M(\sigma(P))$. The key property of σ being a symmetry of $\mathcal{T}$ is that $M \models \mathcal{T}$ iff $\sigma(M) \models \mathcal{T}$. The *orbit* of a model M under a symmetry group Σ is the set of models that can be obtained by applying any of the symmetries in Σ. A symmetry group divides the space of models into disjoint sets, where the models in an orbit either all satisfy or all do not satisfy the theory.

In this work, we will consider symmetries over the terms (constants) of a first order relational theory. A *relational theory* is a tuple $\mathcal{T} = (F, \mathcal{E})$, where F is a set of first-order formulas and $\mathcal{E}$ is a set of evidence. We restrict the formulas in F to be built from predicates, variables, quantifiers, and logical connectives, but no constants or function symbols. $\mathcal{E}$ is a set of ground literals; that is, literals built from predicates and constant symbols. Universal and existential quantification is over the set of the theory's constants $\mathcal{D}$ (i.e. the constants that appear in its evidence). Optionally, the formulas or predicates may be weighted.

In (Kopp, Singla, and Kautz 2015), two classes of symmetries over terms of relational theories were formalized, and methods to exploit the symmetries for MPE inference were given. The first was the set of *term symmetries*. A term symmetry is a permutation σ of the terms in the theory such that $\sigma(\mathcal{T}) = \mathcal{T}$. Note that a permutation of terms over a ground theory induces a permutation of atoms of the theory. The second class of symmetries over terms is a special case of the former called *term equivalent symmetries*. A term equivalent symmetry is a partitioning of the terms in the theory such that if two terms C_1 and C_2 appear in the same term equivalent class, they can be permuted without changing the theory. In other words, they define equivalence classes of terms. Both classes of symmetries can be detected over the evidence of a theory.

Consider a relational theory that models allocation of computational resources to distributed computing tasks. The domain might have a type for CPUs in a cluster, cores on a CPU, and computational tasks. It may have predicates that describe properties of the computational resources, as well as the requirements of the tasks. We might want to determine the best (MPE) allocation of resources to tasks, or we might use the theory to describe a stochastic scheduling algorithm, and be interested in the (marginal) probability of a task being mapped to a particular core. In this domain, all of the cores that belong to the same CPU belong to the same term equivalent symmetry group. Furthermore, if two CPUs have the same number and types of cores, then there are term symmetries that permute the CPUs while also permuting their respective cores.

The methods for exploiting these symmetries involved adding clauses to the theory that restricted the search space while guaranteeing that at least one assignment in each orbit was left in the space. For model counting, this technique cannot be applied straightforwardly, since reducing the search space will reduce the number of models. In this work, we leverage the class of term equivalent symmetries in a way that does not directly modify the search space, and is therefore suitable for model counting.

A canonical form for symmetric theories

We introduce the notion of a canonical form for ground first-order clausal theories under a set of symmetries, and define a particular canonical form for such theories under term equivalent symmetries.

Let $\mathcal{T}$ be a ground, first-order clausal theory, and let Σ be a set of symmetries over terms, literals, or clauses in $\mathcal{T}$. Let there be an ordering on all of the possible theories. We say that another theory $\mathcal{T}'$ is symmetric to $\mathcal{T}$ under term partition $\mathcal{Z}$ if there is a permutation σ that respects Σ such that $\sigma(\mathcal{T}) = \mathcal{T}'$. A theory $\mathcal{T}$ is in canonical form if for every

$\mathcal{T}'$ that is symmetric to $\mathcal{T}$ under Σ, $\mathcal{T}$ comes before $\mathcal{T}'$ in the ordering.

First we will precisely define an ordering on ground theories. There are many possible orderings, here we give just one.

Put an ordering on the constants of the theory: $C_1, \ldots, C_n$. Put an ordering on the predicates of the theory and their negations: $P_1, \neg P_1, \ldots, P_k, \neg P_k$. This induces an ordering on the literals of the theory: a literal L comes before L' in the ordering if the predicate of L comes before that of L' in the ordering of predicates. If they are the same, then L comes before L' if, when considering the arguments in order, the first differing argument of L comes before that of L' in the ordering of constants. When considering the literals of a clause in turn, we always do so according to this order.

Now we define an ordering on clauses, which is actually a hierarchy of orderings. A level of the hierarchy is only considered if the comparison criteria of the above levels are equal for the clauses to be compared. The highest ordering is the ordering of weights: A clause C comes before C' in the ordering if the weight of C is less than the weight of C' i.e. $w(C) < w(C')$. If the theory does not have weighted clauses, this level is omitted. The next ordering is the ordering on clause lengths. A clause C comes before C' in the ordering if $w(C) = w(C')$ and C has fewer literals than C', i.e. $l(C) < l(C')$. The final ordering is the ordering on literals. A clause C comes before C' in the ordering if $w(C) = w(C')$, $l(C) = l(C')$, and the first differing literal of C comes before that of C' in the ordering of literals.

Finally we define the ordering on theories. Given two theories, we consider each clause of the theories in the order defined above. A theory $\mathcal{T}$ comes before $\mathcal{T}'$ in the ordering if the first differing clause of $\mathcal{T}$ comes before that of $\mathcal{T}'$ in the ordering of clauses.

Now that we have an ordering on the theories, we define a canonical form for term equivalent symmetries. Let $\mathcal{T}$ be a ground, first-order clausal theory, and $\mathcal{Z}$ be a term equivalent partition of the constants in $\mathcal{T}$. We say that $\mathcal{T}$ is in canonical form if for every $\mathcal{T}'$ that is symmetric to $\mathcal{T}$ under $\mathcal{Z}$, $\mathcal{T}$ comes before $\mathcal{T}'$ in the ordering of theories.

A procedure for canonicalization

Now that we have a precisely defined canonical form for theories under term equivalent partitions, we give a procedure for converting a theory $\mathcal{T}$ to canonical form under term equivalent partition $\mathcal{Z}$. First, for each clause C in $\mathcal{T}$, we reorder the literals of C so that they appear in the ordering of literals defined in the previous section. If the length of the clause is $l(C)$, then this takes time $O(l(C)\log(l(C)))$. Let l be the length of the longest clause, and $|T|$ be the number of clauses in the theory. To complete this step for each clause in the theory, it takes time $O(|T| \cdot l\log(l))$. Next, we reorder the clauses of $\mathcal{T}$ according to the ordering of clauses defined in the previous section. This takes time $O(|T|\log(|T|))$.

Next, we iterate over the arguments of the predicates of the theory in order, building a permutation σ. For each argument C_i we encounter we do the following: If C_i has not been encountered before, we find the first constant C_i' in the ordering that is term equivalent to C_i under $\mathcal{Z}$, add $C_i \to C_i'$ to σ, and delete C_i' from $\mathcal{Z}$. If C_i was encountered before (or equivalently, if C_i appears on the left hand side of σ), it is skipped. Note that because we delete C_i' from $\mathcal{Z}$, no constant will appear on the right hand side of σ more than once. Likewise, because we skip constants that have already been considered, no constant will appear on the left hand side of σ more than once. The procedure ends when either all of the arguments have been iterated over, or when every constant in the theory appears on both sides of σ.

If we store the term equivalent partitions in sorted arrays, we can use a simple pointer to make the lookup and deletion of C_i' a constant time operation. Thus, if the number of arguments appearing in the theory is a, then this step takes time $O(a)$.

Finally, to return the canonical form, we simply apply the permutation we built to the input theory, i.e. $\mathcal{T}' = \sigma(\mathcal{T})$. This takes time proportional to the number of terms in the theory, so $O(a)$. Applying the permutation can actually be done in-place while computing it.

Theorem 1. *The theory $\mathcal{T}'$ found by the above procedure is the canonical form of $\mathcal{T}$ under $\mathcal{Z}$.*

Proof. This proof takes two parts. First we will prove that $\mathcal{T}'$ is equivalent to $\mathcal{T}$ under $\mathcal{Z}$, then we will prove that $\mathcal{T}'$ is the first such equivalent theory. By construction, the permutation σ produced by the above procedure only permutes a term to a term that appears in the same term equivalent partition defined by $\mathcal{Z}$. Furthermore, σ is one-to-one and onto. Therefore applying σ yields a term equivalent theory under $\mathcal{Z}$.

Next we show that $\mathcal{T}'$ is the first such theory in the ordering. Suppose not. Suppose there was a theory $\mathcal{T}^0$ that was term equivalent with $\mathcal{T}$ and appeared before $\mathcal{T}'$ in the ordering. By symmetry, $\mathcal{T}^0$ is equivalent to $\mathcal{T}'$ under $\mathcal{Z}$. Furthermore, by symmetry, the clauses of $\mathcal{T}'$ and $\mathcal{T}^0$ have the same weights and predicate signatures. By the definition of the ordering, the first pair of terms that differ between $\mathcal{T}^0$ and $\mathcal{T}'$, C^0 and C' respectively, must be such that C^0 comes before C' in the ordering. However, if that were true, then we must have chosen a term from a partition of $\mathcal{Z}$ that was not least in the ordering when building $\mathcal{T}'$. This is not how the procedure is defined. Contradiction. Therefore $\mathcal{T}'$ is the first theory in the ordering that is equivalent to $\mathcal{T}$ under $\mathcal{Z}$.

Therefore $\mathcal{T}'$ is the canonical form of $\mathcal{T}$. □

Next we give a worked example. We apply the given procedure to two symmetric theories, and find that they have the same canonical form.

Example 1 Suppose that after detecting term equivalent symmetries from evidence, we found the term equivalent partition to be $\mathcal{Z} = \{\{A, B, C\},\ \{X, Y, Z\}\}$.

$P(A) \vee Q(Y)$	$P(B) \vee Q(X)$
$Q(Z)$	$Q(Y)$
$P(C) \vee Q(X) \vee P(Z)$	$P(A) \vee Q(Z) \vee P(Y)$

Let the ordering on terms and predicates be alphabetical. We

first rewrite the theory according to the induced ordering.

$$\begin{array}{ll} Q(Z) & Q(Y) \\ P(A) \vee Q(Y) & P(B) \vee Q(X) \\ P(C) \vee P(Z) \vee Q(X) & P(A) \vee P(Y) \vee Q(Z) \end{array}$$

We will perform the permutation in place. We iterate over the terms in the theory. On the left, we have $Z \to X$, and on the right, $Y \to X$. We apply this partial symmetry to the first term.

$$\begin{array}{ll} Q(X) & Q(X) \\ P(A) \vee Q(Y) & P(B) \vee Q(X) \\ P(C) \vee P(Z) \vee Q(X) & P(A) \vee P(Y) \vee Q(Z) \end{array}$$

We continue to the next term. On the left, it is already the least term, so we have $A \to A$. On the right, $B \to A$.

$$\begin{array}{ll} Q(X) & Q(X) \\ P(A) \vee Q(Y) & P(A) \vee Q(X) \\ P(C) \vee P(Z) \vee Q(X) & P(A) \vee P(Y) \vee Q(Z) \end{array}$$

On the left, Y is already the least term, so we have $Y \to Y$. On the right, $X \to Y$.

$$\begin{array}{ll} Q(X) & Q(X) \\ P(A) \vee Q(Y) & P(A) \vee Q(Y) \\ P(C) \vee P(Z) \vee Q(X) & P(A) \vee P(Y) \vee Q(Z) \end{array}$$

Next term, $C \to B$ on the left and $A \to B$ on the right.

$$\begin{array}{ll} Q(X) & Q(X) \\ P(A) \vee Q(Y) & P(A) \vee Q(Y) \\ P(B) \vee P(Z) \vee Q(X) & P(B) \vee P(Y) \vee Q(Z) \end{array}$$

Next term, we apply the $Z \to X$ from before on the left, and the $Y \to X$ from before on the right.

$$\begin{array}{ll} Q(X) & Q(X) \\ P(A) \vee Q(Y) & P(A) \vee Q(Y) \\ P(B) \vee P(X) \vee Q(X) & P(B) \vee P(X) \vee Q(Z) \end{array}$$

Last term, $X \to Z$ on the left, and $Z \to Z$ on the right.

$$\begin{array}{ll} Q(X) & Q(X) \\ P(A) \vee Q(Y) & P(A) \vee Q(Y) \\ P(B) \vee P(X) \vee Q(Z) & P(B) \vee P(X) \vee Q(Z) \end{array}$$

We see that the two different theories have the same canonical form. By inspection, we see that they are equivalent under the set of term symmetries.

Applying canonical form in model counting

Exact model counters such as RelSAT (Bayardo Jr. and Schrag 1997) and Cachet (Sang et al. 2004) leverage a caching procedure to count models more efficiently. Consider an arbitrary place in the search space, consisting of a partial assignment to the variables and a simplified version of theory. In a systematic search, the algorithm will count the number of models in this simplified theory, then backtrack to explore other parts of the search space. It is possible that, at a different point in the search space, the simplified theory will be identical to the simplified theory we previously counted the models of. Thus, these algorithms, upon counting the models in a simplified theory, will cache the simplified theory and the number (or weight) of models it has in a lookup table. Then, if it is encountered again, the value can be reused without performing the counting again.

This technique is even more effective when used in conjunction with theory decomposition. These systematic search algorithms will, at a given point in the search space, detect if the simplified theory can be split into two or more disjoint subtheories. A theory $\mathcal{T}$ can be partitioned in such a way if it contains two or more disjoint subsets of it's clauses $\mathcal{T}_1, \ldots, \mathcal{T}_k$ such that none of the $\mathcal{T}_i$ share any variables. Since they share no variables, their models can be counted independently. In a caching algorithm, it is the disjoint subtheories and their counts that are stored in the lookup table.

One problem with this method is knowing which subtheories to cache. If we cache every subtheory, our cache will take up too much memory. If too few, we may do redundant computation. Furthermore, algorithms that are not aware of symmetries will perform model counting on subtheories that are symmetrical, despite the fact that the result will be the same. We propose a method of leveraging canonical form to reduce the number of symmetrical subtheories that are recomputed. Furthermore, the technique will reduce the memory needed to cache many subtheories.

With the foundation provided in the previous section, the technique is actually very simple. We detect the term equivalent partition once before we begin the search. Each time a theory is split into disjoint subtheories, we convert the subtheories into canonical form with respect to the partition. If the canonical form of the subtheory does not appear in the cache we compute the number of models and cache the result with the canonical form as the key. If it does appear in the cache, that means the original subtheory is identical to *or term equivalent to* a subtheory that we saw previously. We save space in the cache because each cached theory is actually representative of every theory symmetric to it under the term equivalent partition. Furthermore, we reduce redundant computations because when we count the number of models in a subtheory, we reuse the value for every theory that is symmetric to it under the term equivalent partition, rather than just subtheories that are exactly identical to it. It is hypothesized that leveraging this technique will result in significant speedups in high-symmetry domains.

Conclusion and future work

We have provided a canonical form for ground first order CNF theories under a term equivalent partitioning, as well as an efficient procedure for converting a theory to canonical form. We then described a simple technique to increase the solving efficiency of exact model counters by leveraging this canonical form.

Short term future work includes implementing this system by modifying an existing exact model counter, and comparing the technique with the stock version, as well as other algorithms that leverage these types of symmetries. Longer term goals include creating a canonical form and efficient

procedure that canonicalizes theories under a set of term symmetries, a class of symmetries of which term equivalent symmetries are a special case.

References

Audemard, G.; Benhamou, B.; and Henocque, L. 2006. Predicting and detecting symmetries in FOL finite model search. *J. Autom. Reasoning* 36(3):177–212.

Bayardo Jr., R. J., and Schrag, R. 1997. Using CSP look-back techniques to solve real-world SAT instances. In Kuipers, B., and Webber, B. L., eds., *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference, AAAI 97, IAAI 97, July 27-31, 1997, Providence, Rhode Island.*, 203–208. AAAI Press / The MIT Press.

Bui, H. H.; Huynh, T. N.; and Riedel, S. 2013. Automorphism groups of graphical models and lifted variational inference. In Nicholson, A., and Smyth, P., eds., *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence, UAI 2013, Bellevue, WA, USA, August 11-15, 2013*. AUAI Press.

Crawford, J. M.; Ginsberg, M. L.; Luks, E. M.; and Roy, A. 1996. Symmetry-breaking predicates for search problems. In Aiello, L. C.; Doyle, J.; and Shapiro, S. C., eds., *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR'96), Cambridge, Massachusetts, USA, November 5-8, 1996.*, 148–159. Morgan Kaufmann.

de Salvo Braz, R.; Amir, E.; and Roth, D. 2005. Lifted first-order probabilistic inference. In Kaelbling and Saffiotti (2005), 1319–1325.

den Broeck, G. V.; Meert, W.; and Darwiche, A. 2014. Skolemization for weighted first-order model counting. In Baral, C.; Giacomo, G. D.; and Eiter, T., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*. AAAI Press.

Flener, P.; Pearson, J.; Sellmann, M.; Hentenryck, P. V.; and Ågren, M. 2009. Dynamic structural symmetry breaking for constraint satisfaction problems. *Constraints* 14(4):506–538.

Gogate, V., and Domingos, P. M. 2011. Probabilistic theorem proving. In Cozman, F. G., and Pfeffer, A., eds., *UAI 2011, Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, Barcelona, Spain, July 14-17, 2011*, 256–265. AUAI Press.

Kaelbling, L. P., and Saffiotti, A., eds. 2005. *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005*. Professional Book Center.

Kopp, T.; Singla, P.; and Kautz, H. 2015. Lifted symmetry detection and breaking for map inference. In *Advances in Neural Information Processing Systems 28: 29th Annual Conference on Neural Information Processing Systems 2015. Proceedings of a meeting held 7-12 December 2015, Montreal, Quebec, Canada*.

Meseguer, P., and Torras, C. 2001. Exploiting symmetries within constraint satisfaction search. *Artif. Intell.* 129(1-2):133–163.

Mittal, H.; Goyal, P.; Gogate, V. G.; and Singla, P. 2014. New rules for domain independent lifted MAP inference. In Ghahramani, Z.; Welling, M.; Cortes, C.; Lawrence, N. D.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, 649–657.

Niepert, M. 2012. Lifted probabilistic inference: An mcmc perspective. In *Proc. of 2nd Intl. Workshop on Statistical Relational AI*.

Niepert, M. 2013. Symmetry-aware marginal density estimation. In desJardins, M., and Littman, M. L., eds., *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA*. AAAI Press.

Poole, D. 2003. First-order probabilistic inference. In Gottlob, G., and Walsh, T., eds., *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, 985–991. Morgan Kaufmann.

Richardson, M., and Domingos, P. M. 2006. Markov logic networks. *Machine Learning* 62(1-2):107–136.

Sang, T.; Bacchus, F.; Beame, P.; Kautz, H. A.; and Pitassi, T. 2004. Combining component caching and clause learning for effective model counting. In *SAT 2004 - The Seventh International Conference on Theory and Applications of Satisfiability Testing, 10-13 May 2004, Vancouver, BC, Canada, Online Proceedings*.

Sellmann, M., and Hentenryck, P. V. 2005. Structural symmetry breaking. In Kaelbling and Saffiotti (2005), 298–303.

Singla, P., and Domingos, P. M. 2008. Lifted first-order belief propagation. In Fox, D., and Gomes, C. P., eds., *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, 1094–1099. AAAI Press.

Singla, P.; Nath, A.; and Domingos, P. M. 2014. Approximate lifting techniques for belief propagation. In Brodley, C. E., and Stone, P., eds., *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, 2497–2504. AAAI Press.

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

Walsh, T. 2012. Symmetry breaking constraints: Recent results. In Hoffmann, J., and Selman, B., eds., *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.* AAAI Press.