Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

A Tractable First-Order Probabilistic Logic

Pedro Domingos and **W. Austin Webb**
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350, U.S.A.
{*pedrod, webb*}*@cs.washington.edu*

Abstract

Tractable subsets of first-order logic are a central topic in AI research. Several of these formalisms have been used as the basis for first-order probabilistic languages. However, these are intractable, losing the original motivation. Here we propose the first non-trivially tractable first-order probabilistic language. It is a subset of Markov logic, and uses probabilistic class and part hierarchies to control complexity. We call it TML (Tractable Markov Logic). We show that TML knowledge bases allow for efficient inference even when the corresponding graphical models have very high treewidth. We also show how probabilistic inheritance, default reasoning, and other inference patterns can be carried out in TML. TML opens up the prospect of efficient large-scale first-order probabilistic inference.

Introduction

The tradeoff between expressiveness and tractability is a central problem in AI. First-order logic can express most knowledge compactly, but is intractable. As a result, much research has focused on developing tractable subsets of it. First-order logic and its subsets ignore that most knowledge is uncertain, severely limiting their applicability. Graphical models like Bayesian and Markov networks can represent many probability distributions compactly, but their expressiveness is only at the level of propositional logic, and inference in them is also intractable.

Many languages have been developed to handle uncertain first-order knowledge. Most of these use a tractable subset of first-order logic, like function-free Horn clauses (e.g., (Wellman, Breese, and Goldman 1992; Poole 1993; Muggleton 1996; De Raedt, Kimmig, and Toivonen 2007)) or description logics (e.g., (Jaeger 1994; Koller, Levy, and Pfeffer 1997; d'Amato, Fanizzi, and Lukasiewicz 2008; Niepert, Noessner, and Stuckenschmidt 2011)). Unfortunately, the probabilistic extensions of these languages are intractable, losing the main advantage of restricting the representation. Other languages, like Markov logic (Domingos and Lowd 2009), allow a full range of first-order constructs. Despite much progress in the last decade, the applicability of these representations continues to be significantly limited by the difficulty of inference.

The ideal language for most AI applications would be a tractable first-order probabilistic logic, but the prospects for such a language that is not too restricted to be useful seem dim. Cooper (1990) showed that inference in propositional graphical models is already NP-hard, and Dagum and Luby (1993) showed that this is true even for approximate inference. Probabilistic inference can be reduced to model counting, and Roth (1996) showed that even approximate counting is intractable, even for very restricted propositional languages, like CNFs with no negation and at most two literals per clause, or Horn CNFs with at most two literals per clause and three occurrences of any variable.

We can trivially ensure that a first-order probabilistic knowledge base is tractable by requiring that, when grounded, it reduce to a low-treewidth graphical model. However, this is a syntactic condition that is not very meaningful or perspicuous in terms of the domain, and experience has shown that few real domains can be well modeled by low-treewidth graphs. This is particularly true of first-order domains, where even very simple formulas can lead to unbounded treewidth.

Although probabilistic inference is commonly assumed to be exponential in treewidth, a line of research that includes arithmetic circuits (Darwiche 2003), AND-OR graphs (Dechter and Mateescu 2007) and sum-product networks (Poon and Domingos 2011) shows that this need not be the case. However, the consequences of this for first-order probabilistic languages seem to have so far gone unnoticed. In addition, recent advances in lifted probabilistic inference (e.g., (Gogate and Domingos 2011)) make tractable inference possible even in cases where the most efficient propositional structures do not. In this paper, we take advantage of these to show that it is possible to define a first-order probabilistic logic that is tractable, and yet expressive enough to encompass many cases of interest, including junction trees, non-recursive PCFGs, networks with cluster structure, and inheritance hierarchies.

We call this language Tractable Markov Logic, or TML for short. In TML, the domain is decomposed into parts, each part is drawn probabilistically from a class hierarchy, the part is further decomposed into subparts according to its class, and so forth. Relations of any arity are allowed, provided that they are between subparts of some part (e.g., workers in a company, or companies in a market), and that they depend on each other only through the part's class. Inference in TML is reducible to computing partition functions, as in all probabilistic languages. It is tractable because there is a correspondence between classes in the hierarchy and sums in the computation of the partition function, and between parts and products.

We begin with some necessary background. We then define TML, prove its tractability, and illustrate its expressiveness. The paper concludes with extensions and discussion.

Background

TML is a subset of Markov logic (Domingos and Lowd 2009). A *Markov logic network (MLN)* is a set of weighted first-order clauses. It defines a Markov network over all the ground atoms in its Herbrand base, with a feature corresponding to each ground clause in the Herbrand base. (We assume Herbrand interpretations throughout this paper.) The weight of each feature is the weight of the corresponding first-order clause. If $\mathbf{x}$ is a possible world (assignment of truth values to all ground atoms), then $P(\mathbf{x}) = \frac{1}{Z}\exp\left(\sum_i w_i n_i(\mathbf{x})\right)$, where w_i is the weight of the ith clause, $n_i(\mathbf{x})$ is its number of true groundings in $\mathbf{x}$, and $Z = \sum_{\mathbf{x}} \exp\left(\sum_i w_i n_i(\mathbf{x})\right)$ is the *partition function*. For example, if an MLN consists of the formula $\forall \mathtt{x} \forall \mathtt{y}\ \mathtt{Smokes(x)} \wedge \mathtt{Friends(x,y)} \Rightarrow \mathtt{Smokes(y)}$ with a positive weight and a set of facts about smoking and friendship (e.g., $\mathtt{Friends(Anna, Bob)}$), the probability of a world decreases exponentially with the number of pairs of friends that do not have similar smoking habits.

Inference in TML is carried out using probabilistic theorem proving (PTP) (Gogate and Domingos 2011). PTP is an inference procedure that unifies theorem proving and probabilistic inference. PTP inputs a *probabilistic knowledge base (PKB)* K and a query formula Q and outputs $P(Q|K)$. A PKB is a set of first-order formulas F_i and associated *potential values* $\phi_i \geq 0$. It defines a factored distribution over possible worlds with one factor or potential per grounding of each formula, with the value of the potential being 1 if the ground formula is true and ϕ_i if it is false. Thus $P(\mathbf{x}) = \frac{1}{Z}\prod_i \phi_i^{n_i(\mathbf{x})}$, where $n_i(\mathbf{x})$ is the number of false groundings of F_i in $\mathbf{x}$. *Hard* formulas have $\phi_i = 0$ and *soft* formulas have $\phi_i > 0$. An MLN is a PKB where the potential value corresponding to formula F_i with weight w_i is $\phi_i = e^{-w_i}$.

All conditional probabilities can be computed as ratios of partition functions: $P(Q|K) = Z(K \cup \{(Q,0)\})/Z(K)$, where $Z(K)$ is the partition function of K and $Z(K \cup \{(Q,0)\})$ is the partition function of K with Q added as a hard formula. PTP computes partition functions by lifted weighted model counting. The *model count* of a CNF is the number of worlds that satisfy it. In weighted model counting each literal has a weight, and a CNF's count is the sum over satisfying worlds of the product of the weights of the literals that are true in that world. PTP uses a lifted, weighted extension of the Relsat model counter, which is in turn an extension of the DPLL satisfiability solver. PTP first converts the PKB into a CNF L and set of literal weights W. In this process, each soft formula F_i is replaced by a hard formula $F_i \Leftrightarrow A_i$, where A_i is a new atom with the same arguments as F_i and the weight of $\neg A_i$ is ϕ_i, all other weights being 1.

Algorithm 1 LWMC(CNF L, substs. S, weights W)

// Base case
if all clauses in L are satisfied **then**
  return $\prod_{A \in M(L)} (W_A + W_{\neg A})^{n_A(S)}$
if L has an empty unsatisfied clause **then return** 0
// Decomposition step
if there exists a lifted decomposition $\{L_{1,1}, \ldots, L_{1,m_1}, \ldots, L_{k,1}, \ldots, L_{k,m_k}\}$ of L under S **then**
  return $\prod_{i=1}^{k} [\text{LWMC}(L_{i,1}, S, W)]^{m_i}$
// Splitting step
Choose an atom A
Let $\{\Sigma_{A,S}^{(1)}, \ldots, \Sigma_{A,S}^{(l)}\}$ be a lifted split of A for L under S
return $\sum_{i=1}^{l} n_i W_A^{t_i} W_{\neg A}^{f_i} \text{LWMC}(L|\sigma_i;\ S_i, W)$

Algorithm 1 shows pseudo-code for LWMC, the core routine in PTP. $M(L)$ is the set of atoms appearing in L, and $n_A(S)$ is the number of groundings of A consistent with the substitution constraints in S. A *lifted decomposition* is a partition of a first-order CNF into a set of CNFs with no ground atoms in common. A *lifted split* of an atom A for CNF L is a partition of the possible truth assignments to groundings of A such that, in each part, (1) all truth assignments have the same number of true atoms and (2) the CNFs obtained by applying these truth assignments to L are identical. For the ith part, n_i is its size, t_i/f_i is the number of true/false atoms in it, σ_i is some truth assignment in it, $L|\sigma_i$ is the result of simplifying L with σ_i, and S_i is S augmented with the substitution constraints added in the process. In a nutshell, LWMC works by finding lifted splits that lead to lifted decompositions, progressively simplifying the CNF until the base case is reached.

Tractable Markov Logic

A *TML knowledge base (KB)* is a set of rules. These can take one of three forms, summarized in Table 1. $\mathtt{F : w}$ is the syntax for formula $\mathtt{F}$ with weight $\mathtt{w}$. Rules without weights are hard. Throughout this paper we use lowercase strings to represent variables and capitalized strings to represent constants.

Subclass rules are statements of the form "$\mathtt{C_1}$ is a subclass of $\mathtt{C_2}$," i.e., $\mathtt{C_1} \subseteq \mathtt{C_2}$. For example, $\mathtt{Is(Family, SocialUnit)}$. For each TML subclass rule, the corresponding MLN contains the two formulas shown in Table 1. In addition, for each pair of classes $(\mathtt{C_1}, \mathtt{C_2})$ appearing in a TML KB K such that, for some $\mathtt{D}$, $\mathtt{Is(C_1, D)}$ and $\mathtt{Is(C_2, D)}$ also appear in K, the MLN corresponding to K contains the formula $\forall \mathtt{x}\ \neg\mathtt{Is(x, C_1)} \vee \neg\mathtt{Is(x, C_2)}$. In other words, subclasses of the same class are mutually exclusive.

Using weights instead of probabilities as parameters in subclass rules allows the subclasses of a class to not be ex-

	Name	TML Syntax	Markov Logic Syntax	Comments
Rules	Subclass	$\texttt{Is}(\texttt{C}_1, \texttt{C}_2) : \texttt{w}$	$\forall \texttt{x}\ \texttt{Is}(\texttt{x}, \texttt{C}_1) \Rightarrow \texttt{Is}(\texttt{x}, \texttt{C}_2)$, $\forall \texttt{x}\ \texttt{Is}(\texttt{x}, \texttt{C}_2) \Rightarrow \texttt{Is}(\texttt{x}, \texttt{C}_1) : \texttt{w}$	
	Subpart	$\texttt{Has}(\texttt{C}_1, \texttt{C}_2, \texttt{P}, \texttt{n})$	$\forall \texttt{x} \forall \texttt{i}\ \texttt{Is}(\texttt{x}, \texttt{C}_1) \Rightarrow \texttt{Is}(\texttt{P}_\texttt{i}(\texttt{x}), \texttt{C}_2)$	$1 \leq \texttt{i} \leq \texttt{n}$; $\texttt{P}, \texttt{n}$ optional
	Relation	$\texttt{R}(\texttt{C}, \texttt{P}_1, \ldots, \texttt{P}_\texttt{n}) : \texttt{w}$	$\forall \texttt{x} \forall \texttt{i}_1 \ldots \forall \texttt{i}_\texttt{n}\ \texttt{Is}(\texttt{x}, \texttt{C}) \Rightarrow \texttt{R}(\texttt{P}_{\texttt{i}_1}(\texttt{x}), \ldots, \texttt{P}_{\texttt{i}_\texttt{n}}(\texttt{x})) : \texttt{w}$	$\neg \texttt{R}(\ldots)$ also allowed
Facts	Instance	$\texttt{Is}(\texttt{X}, \texttt{C})$	$\texttt{Is}(\texttt{X}, \texttt{C})$	$\neg \texttt{Is}(\texttt{X}, \texttt{C})$ also allowed
	Subpart	$\texttt{Has}(\texttt{X}_1, \texttt{X}_2, \texttt{P})$	$\texttt{X}_2 = \texttt{P}(\texttt{X}_1)$	
	Relation	$\texttt{R}(\texttt{X}, \texttt{P}_1, \ldots, \texttt{P}_\texttt{n})$	$\texttt{R}(\texttt{P}_1(\texttt{X}), \ldots, \texttt{P}_\texttt{n}(\texttt{X}))$	$\neg \texttt{R}(\ldots)$ also allowed

Table 1: The TML language.

haustive. This in turn allows classes to have default subclasses that are left implicit, and inherit all the class's parts, relations, weights, etc., without change. A default subclass has weight 0; any subclass can be made the default by subtracting its weight from the others'.

Subpart rules are statements of the form "Objects of class $\texttt{C}_1$ have $\texttt{n}$ subparts $\texttt{P}$ of class $\texttt{C}_2$." $\texttt{n} = 1$ by default. For example, $\texttt{Has}(\texttt{Platoon}, \texttt{Lieutenant}, \texttt{Commander})$ means that platoons have lieutenants as commanders. If $\texttt{P}$ is omitted, $\texttt{P}_\texttt{i}$ is $\texttt{C}_2$ with the index $\texttt{i}$ appended. For example, $\texttt{Has}(\texttt{Family}, \texttt{Child}, 2)$, meaning that a typical family has two children, translates to $\forall \texttt{x}\ \texttt{Is}(\texttt{x}, \texttt{Family}) \Rightarrow \texttt{Is}(\texttt{Child}_1(\texttt{x}), \texttt{Child}) \wedge \texttt{Is}(\texttt{Child}_2(\texttt{x}), \texttt{Child})$. $\texttt{P}_\texttt{i}(\texttt{x})$ is syntactic sugar for $\texttt{P}(\texttt{i}, \texttt{x})$.

Subpart rules are always hard because uncertainty over part decompositions can be represented in the subclass rules. For example, the class `Family` may have two subclasses, `TraditionalFamily`, with two parents, and `OneParentFamily`, with one. The probability that a family is of one or the other type is encoded in the weights of the corresponding subclass rules. This can become cumbersome when objects can have many different numbers of parts, but syntactic sugar to simplify it is straightforward to define. For example, we can define parameterized classes like $\texttt{C}(\texttt{C}', \texttt{n}, \texttt{p})$ as a shorthand for a class $\texttt{C}$ and $\texttt{n}$ subclasses $\texttt{C}_\texttt{i}$, where $\texttt{C}_\texttt{i}$ is $\texttt{C}$ with $\texttt{i}$ parts of class $\texttt{C}'$, and $P(\texttt{C}_\texttt{i}|\texttt{C})$ is the binomial probability of $\texttt{i}$ successes in $\texttt{n}$ trials with success probability $\texttt{p}$. In this notation, $\texttt{Family}(\texttt{Child}, 5, \frac{1}{2})$ represents families with up to five children, with the probability of i children being $\frac{1}{32}\binom{5}{i}$. However, for simplicity in this paper we keep the number of constructs to a minimum.

`Is` and `Has` are reserved symbols used to represent subclass and subpart relationships, respectively. `R` in Table 1 is used to represent arbitrary user-defined predicates, which we call *domain predicates*.

Relation rules are statements of the form "Relation `R` holds between subparts $\texttt{P}_1$, $\texttt{P}_2$, etc., of objects of class `C`." (Or, if $\neg\texttt{R}(\ldots)$ is used, `R` does not hold.) For example, $\texttt{Parent}(\texttt{Family}, \texttt{Adult}, \texttt{Child})$ says that in a family every adult is a parent of every child. The relation rules for a class and its complementary classes allow representing arbitrary correlations between domain predicates and classes. Relation rules for a class must refer only to parts defined by subpart rules for that class. The rule $\texttt{R}(\texttt{C})$ with no part arguments represents $\forall \texttt{x}\ \texttt{Is}(\texttt{x}, \texttt{C}) \Rightarrow \texttt{R}(\texttt{x})$. Distributions over attributes of objects can be represented by rules of this form, with an `R` for each value of the attribute, together with `Has` rules defining the attributes (e.g., $\texttt{Has}(\texttt{Hat}, \texttt{Color})$, $\texttt{Red}(\texttt{Color})$). In propositional domains, all relation rules (and facts) are of this form.

Using weights instead of probabilities in relation rules allows TML KBs to be much more compact than otherwise. This is because the weight of a relation rule in a class need only represent the *change* in that relation's log probability from the superclass to the class. Thus a relation can be omitted in all classes where its probability does not change from the superclass. For example, if the probability that a child is adopted is the same in traditional and one-parent families, then only the class `Family` need contain a rule for $\texttt{Parent}(\texttt{Family}, \texttt{Adult}, \texttt{Child})$. But if it is higher in one-parent families, then this can be represented by a rule for $\texttt{Parent}(\ldots)$ with positive weight in `OneParentFamily`.

For each rule form there is a corresponding fact form involving objects instead of classes (see lower half of Table 1): `X` is an instance of `C`, $\texttt{X}_2$ is subpart `P` of $\texttt{X}_1$, and relation `R` holds between subparts $\texttt{P}_1$, $\texttt{P}_2$, etc., of `X`. For example, a TML KB might contain the facts $\texttt{Is}(\texttt{Smiths}, \texttt{Family})$, $\texttt{Has}(\texttt{Smiths}, \texttt{Anna}, \texttt{Adult1})$ and $\texttt{Married}(\texttt{Smiths}, \texttt{Anna}, \texttt{Bob})$. Subpart facts must be consistent with subpart rules, and relation facts must be consistent with relation rules. In other words, a TML KB may only contain a fact of the form $\texttt{Has}(\texttt{X}_1, \texttt{X}_2, \texttt{P})$ if it contains a rule of the form $\texttt{Has}(\texttt{C}_1, \texttt{C}_2, \texttt{P}, \texttt{n})$ with the same `P`, and it may only contain a fact of the form $\texttt{R}(\texttt{X}, \texttt{P}_1, \ldots, \texttt{P}_\texttt{n})$ if it contains a rule of the form $\texttt{R}(\texttt{C}, \texttt{P}_1, \ldots, \texttt{P}_\texttt{n})$ with the same $\texttt{P}_1, \ldots, \texttt{P}_\texttt{n}$. If $\texttt{Family}(\texttt{Child}, 5, \frac{1}{2})$ is a parameterized class as defined above, the fact that the Smiths have three children can be represented by $\texttt{Has}(\texttt{Smiths}, \texttt{Child}, 3)$.

The *class hierarchy* $H(K)$ of a TML KB K is a directed graph with a node for each class and instance in K, and an edge from node(`A`) to node(`B`) iff $\texttt{Is}(\texttt{B}, \texttt{A}) \in K$. The class hierarchy of a TML KB must be a forest. In other words, it is either a tree, with classes as internal nodes and instances as leaves, or a set of such trees.

A TML KB must have a single *top class* that is not a subclass or subpart of any other class, i.e., a class $\texttt{C}_0$ such that the KB contains no rule of the form $\texttt{Is}(\texttt{C}_0, \texttt{C})$ or $\texttt{Has}(\texttt{C}, \texttt{C}_0, \texttt{P}, \texttt{n})$. The top class must have a single instance, called the *top object*. For example, a KB describing American society might have `Society` as top class and `America` as top object, related by $\texttt{Is}(\texttt{America}, \texttt{Society})$. American society might in turn be composed of 100 million families, one of which might be the Smiths: $\texttt{Has}(\texttt{America}, \texttt{Family}, 10^8)$,

$\texttt{Has}(\texttt{America}, \texttt{Family1}, \texttt{Smiths})$.

The semantics of TML is the same as that of infinite Markov logic (Singla and Domingos 2007), except for the handling of functions. In TML we want to allow limited use of functions while keeping domains finite. We do this by replacing the Herbrand universe and Herbrand base with the following.

The *universe* $U(K)$ of a TML KB K is the top object $\texttt{X}_0$ and all its possible subparts, i.e., all terms constructible by applying to $\texttt{X}_0$ the functions along each rooted path in K's part decomposition. If $\texttt{Has}(\texttt{C}_0, \texttt{C}_1, \texttt{P}, \texttt{n}) \in K$, then $\{\texttt{P}_1(\texttt{C}_0), \ldots, \texttt{P}_\texttt{n}(\texttt{C}_0)\} \subset U(K)$; if in addition $\texttt{Has}(\texttt{C}_1, \texttt{C}_2, \texttt{Q}, \texttt{m}) \in K$, then $\{\texttt{Q}_1(\texttt{P}_1(\texttt{C}_0)), \ldots, \texttt{Q}_\texttt{m}(\texttt{P}_\texttt{n}(\texttt{C}_0))\} \subset U(K)$; etc. Constants other than $\texttt{X}_0$ are merely syntactic sugar for parts of $\texttt{X}_0$, and subpart facts serve solely to introduce these constants. In the example above, Anna is syntactic sugar for $\texttt{Adult1}(\texttt{Family1}(\texttt{America}))$. In addition, $U(K)$ includes all class symbols appearing in K.

The *base* $B(K)$ of a TML KB K consists of: an $\texttt{Is}(T, \texttt{C})$ ground atom for each term T in $U(K)$ and each possible class of T; an $\texttt{R}(T, \ldots)$ ground atom for each relation rule of each possible class of T; and the ground clauses corresponding to these atoms. C is a possible class of T iff it is T's class in the part rule that introduces it or an ancestor or descendant of that class in H(K); or, if T is the top object, the top class or its descendants. A TML KB K defines a Markov network over the atoms in $B(K)$ with a feature corresponding to each clause in $B(K)$, etc.

The *part decomposition* $D(K)$ of a TML KB K is a directed graph with a node for each object in $U(K)$ and a leaf for each domain atom in $B(K)$ and its negation. It contains an edge from node(A) to non-leaf node(B) iff $\texttt{Has}(\texttt{C}, \texttt{D}, \texttt{P}, \texttt{n}) \in K$, where C is a possible class of A and D is a possible class of B. It contains an edge from node(B) to leaf node(G), where G is a ground literal, iff B is the first argument of G in a hard rule of a possible class of B, or of G or $\neg G$ in a soft rule of such a class.

A TML KB must be *consistent*: for every pair $(T, \texttt{C})$, where T is a term in $U(K)$ and C is a possible class of T, if the ground literal G ($\texttt{Is}(\ldots)$ or $\texttt{R}(\ldots)$, negated or not) is a descendant of one of T's subparts in $D(K)$ when T is of class C, then $\neg G$ may not be a descendant of any other subpart of T in $D(K)$ when T is of class C or one of its subclasses; and C and its subclasses may not have a soft relation rule for G's predicate with the same subparts, or hard rule with the opposite polarity of G.

A TML KB is *non-contradictory* iff its hard rules and facts contain no contradictions. TML KBs are assumed to be non-contradictory; the partition function of a contradictory KB is zero.

Figure 1 shows a fragment of a TML KB representing a social domain.

Tractability

We now show that TML is tractable. Since all conditional probabilities can be computed as ratios of partition functions, and all entailment queries can be reduced to computing conditional probabilities (Gogate and Domingos 2011), to prove that a language is tractable it suffices to prove that computing the partition function of every KB in that language is tractable.

We begin by defining an algorithm for inference in TML that is an extension of PTP. We call it PTP_T. PTP_T inputs a KB K and outputs its partition function $Z(K)$. It proceeds as follows:

1. *Cleanup.* Delete from K all $\texttt{Has}(\texttt{C}_1, \texttt{C}_2, \texttt{P}, \texttt{n})$ rules for which $\texttt{Has}(\texttt{C}_3, \texttt{C}_2, \texttt{P}, \texttt{n}) \in K$, with $\texttt{C}_3$ being an ancestor of $\texttt{C}_1$ in $H(K)$, since these rules are redundant and do not affect the value of Z.
2. *Translation.* Translate K into MLN syntax, according to Table 1, and the MLN into PKB form (i.e., rules and potential values; notice that this rescales Z, but it it easy to compute the rescaling factor and divide Z by it before returning).
3. *Demodulation.* For each subpart fact $\texttt{X} = \texttt{P}(T)$, where T is a term, replace all appearances of X in K by $\texttt{P}(T)$, starting with the top object of K as T and proceeding down $D(K)$. (E.g., replace Smiths by $\texttt{Family1}(\texttt{America})$ and Anna by $\texttt{Adult1}(\texttt{Family1}(\texttt{America}))$). Delete all subpart facts from K.
4. *Normalization.* Convert K into a CNF and set of literal weights, as in PTP.
5. *Model counting.* Call LWMC (Algorithm 1) on K, choosing lifted decompositions and lifted splits as follows in calls that do not satisfy the base case:
   - (i) Let L be the CNF in the current call of LWMC. If L contains no class symbols, it consists solely of unit clauses over domain atoms, which form a lifted decomposition. Execute it. If L consists of a single domain atom, split on it.
   - (ii) If L contains sub-CNFs with different class arguments and/or disjoint substitution constraints for the object arguments, these form a lifted decomposition. Execute it.
   - (iii) Otherwise, choose a class C with no superclasses and no superparts in L. If $\texttt{Is}(T, \texttt{C}) \in L$ for some term T, split on this atom and then split on the $\texttt{A}(T, \ldots)$ atoms generated by this split from the $\texttt{A}(\texttt{x}, \ldots)$ atoms introduced for rules on C in Step 4.

Notice that handling functions in PTP_T requires no extensions of PTP beyond redefining the Herbrand universe/base as in the previous section, performing Step 3 above, and treating ground terms as constants.

The size $|K|$ of a KB K is the number of rules and facts in it. We can now show that TML is tractable by showing that $\text{PTP}_\text{T}(K)$ always runs in time and space polynomial in $|K|$.

Theorem 1 *The partition function of every TML knowledge base can be computed in time and space polynomial in the size of the knowledge base.*

Proof. It is easy to see that Steps 1-4 of PTP_T run in polynomial time and space. Let n_c, n_p and n_r be respectively the number of subclass, subpart and relation rules in the KB K. Notice that $|U(K)|$, the number of objects in the

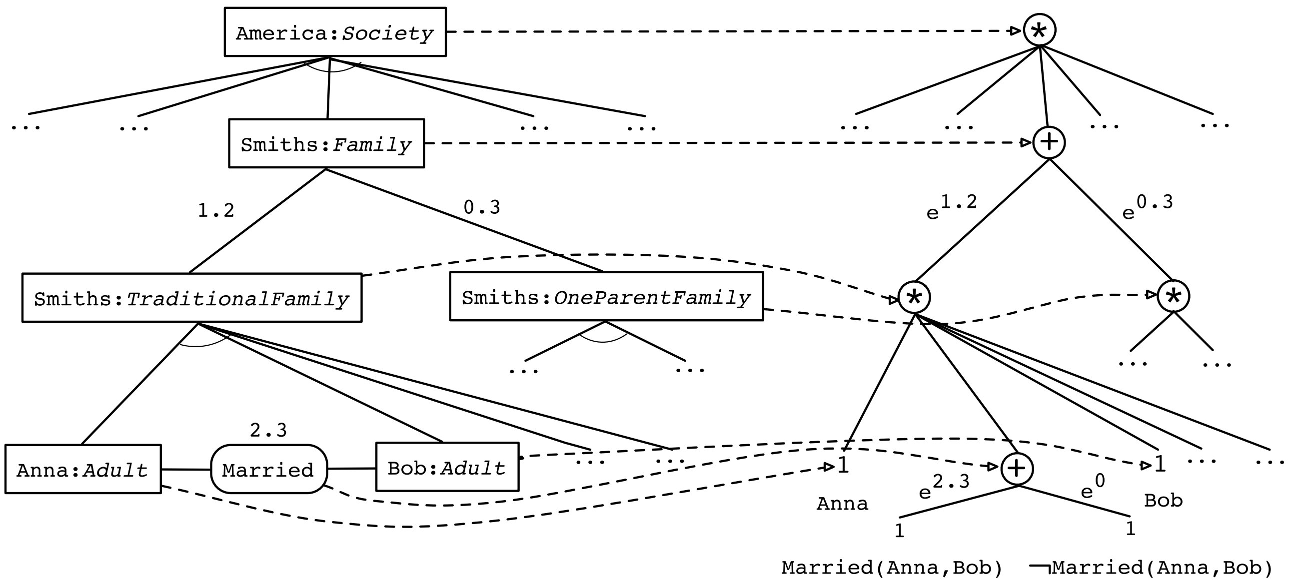

Figure 1: Example of a TML knowledge base and its partition function Z. On the left, rectangles represent objects and their classes, rounded rectangles represent relations, arcs joined by an angle symbol represent subpart rules, other arcs represent subclass or relation rules, and labels represent rule weights. On the right, circles represent operations in the computation of Z (sums or products), and arc labels represent coefficients in a weighted sum. Dashed arrows indicate correspondences between nodes in the KB and operations in Z. If Anna and Bob on the left had internal structure, the leaves labeled "Anna" and "Bob" on the right would be replaced with further computations.

domain (terms in the universe of K), is $O(n_p)$: $|U(K)| = \sum_i \prod_j \mathtt{n}_{ij}$, where i ranges over rooted paths in $D(K)$, of which there are $O(n_p)$, and j ranges over the $\mathtt{n}$'s in the subpart rules on a path, which are constants.

In general PTP, finding lifted decompositions and lifted splits in LWMC could take exponential time. However, in $\mathrm{PTP_T}(K)$ (Step 5), this can be done in $O(1)$ time by maintaining a stack of atoms for splitting. The stack is initialized with $\mathtt{Is}(\mathtt{X}_0, \mathtt{C}_0)$, and after splitting on an $\mathtt{Is}(T, \mathtt{C})$ atom, for some term T and class $\mathtt{C}$ (and on the $\mathtt{A}(\ldots)$ atoms corresponding to $\mathtt{C}$'s rules, introduced in Step 4), the resulting decomposition is executed and the atoms in the consequents of $\mathtt{C}$'s rules are added (see below). Thus each call to LWMC is polynomial, and the key step in the proof is bounding the number of calls to LWMC.

LWMC starts by splitting on $\mathtt{Is}(\mathtt{X}_0, \mathtt{C}_0)$, where $\mathtt{X}_0$ is the top object and $\mathtt{C}_0$ the top class, because by definition $\mathtt{C}_0$ is the only class without superclasses or superparts in K and $\mathtt{X}_0$ is its only instance, and no decomposition is initially possible because $\mathtt{X}_0$ is connected by function applications to all clauses in K. After splitting on $\mathtt{Is}(T, \mathtt{C})$ and all $\mathtt{A}(T, \mathtt{C}, \ldots)$ for some term T and class $\mathtt{C}$: (1) for the true groundings of $\mathtt{Is}(T, \mathtt{C})$ all rules involving $\mathtt{C}$ become unit clauses not involving $\mathtt{C}$; (2) for the false groundings of $\mathtt{Is}(T, \mathtt{C})$, all rules involving $\mathtt{C}$ become satisfied, and can henceforth be ignored. Consider case (1). Suppose L now contains k sets of unit clauses originating from relation rules and facts involving $\mathtt{Is}(T, \mathtt{C})$, at most one set per predicate symbol, and l unit clauses originating from subpart rules involving $\mathtt{Is}(T, \mathtt{C})$. Let L_i be the sub-CNF involving the ith subpart of T according to $\mathtt{C}$ and all its sub-subparts according to $\mathtt{C}$'s descendants in $D(K)$, let R_j be the set of unit clauses with predicate symbol $\mathtt{R_j}$, and let L' be the rest of the CNF. Then $\{R_1, \ldots, R_j, \ldots, R_k, L_1, \ldots, L_i, \ldots, L_l, L'\}$ is a lifted decomposition of L, for the following reasons: (1) $R_1, \ldots, R_k$ do not share any ground atoms because they have different predicate symbols; (2) since K is consistent, either $L_1, \ldots, L_l$ do not share any ground atoms with each other, $R_1, \ldots, R_k$ and L', or they share only pure literals, which can be immediately eliminated; (3) if $R_1, \ldots, R_k$ share any non-pure literals with L', they can be incorporated into L'.

Performing this decomposition leads to a set of recursive calls to LWMC. Overall, with caching (Gogate and Domingos 2011), there is on the order of one call to LWMC per $(T, \mathtt{C})$ pair, where T is a term in $U(K)$ and $\mathtt{C}$ is a possible class of T, and one call per (T, R) pair, where T is a term and R is a relation rule of a possible class of T. The time complexity of LWMC is therefore of the order of the number of such pairs, as is the space complexity (cache size). Since the number of terms in $U(K)$ is $O(n_p)$, the total number of calls to LWMC is $O(n_p(n_r + n_c)) = O(|K|^2)$. □

The constants involved in this bound could be quite high, on the order of the product of the $\mathtt{n}$'s in the subpart rules on a path in $D(K)$. However, lifted inference will generally make this much lower by handling many of the objects allowed by such sequences of rules as one.

Notice that computing the partition function of a TML

KB is not simply a matter of traversing its class hierarchy or its part decomposition. It involves, for each set of objects with equivalent evidence, computing a sum over their possible classes, a product over their subparts in each class, a sum over the possible classes of each subpart, products over their sub-subparts, etc. Thus there is a correspondence between the structure of the KB and the structure of its partition function, which is what makes computing the latter tractable. This is illustrated in Figure 1.

Expressiveness

Given its extremely simple syntax, TML may at first appear to be a very restricted language. In this section we show that TML is in fact surprisingly powerful. For example, it can represent compactly many distributions that graphical models cannot, and it can represent probabilistic inheritance hierarchies over relational domains.

We begin by showing that all junction trees can be represented compactly in TML. A *junction tree J* over a set of discrete variables $R = \{R_k\}$ is a tree where each node represents a subset Q_i of R (called a *clique*), and which satisfies the *running intersection property*: if R_k is in cliques Q_i and Q_j, then it is in every clique on the path from Q_i to Q_j (Pearl 1988). The *separator* S_{ij} corresponding to a pair of cliques (Q_i, Q_j) with an edge between them in J is $Q_i \cap Q_j$. A distribution $P(R)$ over R is representable using junction tree J if instantiating all the variables in a separator renders the variables on different sides of it independent. In this case $P(R) = \prod_{Q_i \in \text{nodes}(J)} P(Q_i) / \prod_{S_{ij} \in \text{edges}(J)} P(S_{ij})$. $P(Q_i)$ is the *potential* over Q_i, and its size is exponential in $|Q_i|$. The size of a junction tree is the sum of the sizes of the potentials over its cliques and separators. Junction trees are the standard data structure for inference in graphical models. The *treewidth* of a graphical model is the size of the largest clique in the corresponding junction tree (obtained by triangulating the graph, etc.), minus one. The cost of inference on a junction tree is exponential in its treewidth.

Theorem 2 *A distribution representable by a junction tree of size n is representable by a TML KB of size $O(n)$.*

Proof. To represent a junction tree J, include in the corresponding TML KB K a class for each state of each clique and a part for each state of each separator in J as follows. Choose a root clique Q_0, and for each of its states q_0^t add to K the rule $\mathtt{Is}(\mathtt{C_t}, \mathtt{C_0})$, where $\mathtt{C_0}$ is the top class, with weight $\log P(q_0^t)$. If the non-separator variable R_k is true in q_0^t, add the hard rule $\mathtt{R_k}(\mathtt{C_t})$, otherwise add $\neg\mathtt{R_k}(\mathtt{C_t})$. (For simplicity, we consider only Boolean variables.) For each q_0^t and each state $s_{0,j}^u$ of each separator $S_{0,j}$ involving Q_0 compatible with q_0^t, add $\mathtt{Has}(\mathtt{C_t}, \mathtt{C_u})$. For each $s_{0,j}^u$, add $\mathtt{Is}(\mathtt{C_u}, \mathtt{C_{0,j}})$ with weight $-\log P(s_{0,j}^u)$. If the separator variable R_l is true in $S_{0,j}$, add the hard rule $\mathtt{R_l}(\mathtt{C_u})$, otherwise add $\neg\mathtt{R_l}(\mathtt{C_u})$. For each state q_j^v of the clique Q_j connected to Q_0 by $S_{0,j}$, add $\mathtt{Is}(\mathtt{C_v}, \mathtt{C_u})$ with weight $\log P(q_j^v)$, etc. The unnormalized probability of a truth assignment a to the $\mathtt{R}$ atoms, obtained by summing out the $\mathtt{Is}$ atoms, is $\prod_{Q_i} P(Q_i = q_i^a) / \prod_{S_{ij}} P(S_{ij} = s_{ij}^a)$, where q_i^a (s_{ij}^a) is the state of Q_i (S_{ij}) consistent with a. The partition function, obtained by then summing out the $\mathtt{R}$ atoms, is 1. Therefore K represents the same distribution as J. □

It follows from Theorem 2 that any low-treewidth graphical model can be represented by a compact TML KB. Importantly, however, many high-treewidth graphical models can also be represented by compact TML KBs. In particular, TML subsumes sum-product networks, the most general class of tractable probabilistic models found to date (Poon and Domingos 2011), as shown below.

Theorem 3 *A distribution representable by a valid sum-product network of size n is representable by a TML KB of size $O(n)$.*

Proof. Add to the TML KB a class for each non-leaf node in the SPN. For each arc from a sum node S to one of its children C with weight w, add the rule $\mathtt{Is}(\mathtt{C}, \mathtt{S})$ with weight $\log w$. For each arc from a product node P to one of its children C, add the rule $\mathtt{Has}(\mathtt{P}, \mathtt{C})$. For each arc from a sum node S to a positive/negative indicator $R/\overline{R}$ with weight w, add the rule $\mathtt{R}(\mathtt{S})/\neg\mathtt{R}(\mathtt{S})$ with weight $\log w$. For each arc from a product node P to a positive/negative indicator R, add the hard rule $\mathtt{R}(\mathtt{S})/\neg\mathtt{R}(\mathtt{S})$. This gives the same probability as the SPN for each possible state of the indicators, and the number of rules in the KB is the same as the number of arcs in the SPN. □

Moreover, this type of tractable high-treewidth distribution is very common in the real world, occurring whenever two subclasses of the same class have different subparts. For example, consider images of objects which may be animals, vehicles, etc. Animals have a head, torso, legs, etc. Vehicles have a cabin, hood, trunk, wheels, and so on. However, all of these ultimately have pixel values as atomic properties. This results in a very high treewidth graphical model, but a tractable TML PKB.

It is also straightforward to show that non-recursive probabilistic context-free grammars (Chi 1999), stochastic image grammars (Zhu and Mumford 2007), hierarchical mixture models (Zhang 2004) and other languages can be compactly encoded in TML. As an example, we give a proof sketch of this for PCFGs below.

Theorem 4 *A non-recursive probabilistic context-free grammar of size n is representable by a TML KB of size polynomial in n.*

Proof. Assume without loss of generality that the grammar is in Chomsky normal form. Let m be the maximum sentence length allowed by the grammar. For each nonterminal C and possible start/end point i/j for it according to the grammar, $0 \leq i < j \leq m$, add a class $\mathtt{C_{ij}}$ to the TML KB K. For the hth production $C \rightarrow DE$ with probability p and start/end/split points $i/j/k$, add the subclass rule $\mathtt{Is}(\mathtt{C^h_{ij}}, \mathtt{C_{ij}})$ with weight $\log p$ and the subpart rules $\mathtt{Has}(\mathtt{C^h_{ij}}, \mathtt{D_{ik}})$ and $\mathtt{Has}(\mathtt{C^h_{ij}}, \mathtt{E_{kj}})$. For each production of the form $C \rightarrow t$, where t is a terminal, add a relation rule of the form $\mathtt{R_t}(\mathtt{C_{i,i+1}})$ per position i. This assigns to each (sentence, parse tree) pair the same probability as the PCFG. Since there are n productions and each generates $O(m^3)$ rules, the size of K is $O(nm^3)$. □

A sentence to parse is expressed as a set of facts of the form $\mathtt{R_t(i)}$, one per token. The traceback of $\mathrm{PTP_T}$ applied to the KB comprising the grammar and these facts, with sums replaced by maxes, is the MAP parse of the sentence.

Most interestingly, perhaps, TMLs can represent probabilistic inheritance hierarchies, and perform a type of default reasoning over them (Etherington and Reiter 1983). In an inheritance hierarchy, a property of an object is represented at the highest class that has it. For example, in TML notation, $\mathtt{Flies(Bird)}$, but not $\mathtt{Flies(Animal)}$. If $\mathtt{Is(Opus, Bird)}$, then $\mathtt{Flies(Opus)}$. However, rules have exceptions, e.g., $\mathtt{Is(Penguin, Bird)}$ and $\neg\mathtt{Flies(Penguin)}$. In standard logic, this would lead to a contradiction, but in default reasoning the more specific rule is allowed to defeat the more general one. TML implements a probabilistic generalization of this, where the probability of a predicate can increase or decrease from a class to a subclass depending on the weights of the corresponding relation rules, and does not change if there is no rule for the predicate in the subclass. More formally:

Proposition 1 *Let* $(\mathtt{C_0}, \ldots, \mathtt{C_i}, \ldots \mathtt{C_n})$ *be a set of classes in a TML KB* K*, with* $\mathtt{C_0}$ *being a root of* $H(K)$ *and* $\mathtt{C_n}$ *having no subclasses, and let* $\mathtt{Is(C_{i+1}, C_i)} \in K$ *for* $0 \leq i < n$*. Let* w_i *be the weight of* $\mathtt{R(C_i)}$ *for some domain predicate* $\mathtt{R}$*, and let* K *contain no other rules for* $\mathtt{R}$*. If* X *is an object for which* $\mathtt{Is(X, C_n)} \in K$*, then the unnormalized probability of* $\mathtt{R(X)}$ *is* $\exp(\sum_{i=0}^{n} w_i)$.

For example, if we know that Opus is a penguin, decreasing the weight of $\mathtt{Flies(Penguin)}$ decreases the likelihood that he flies. A generalization of Prop. 1 applies to higher-arity predicates, summing weights over all tuples of classes of the arguments.

Further, inference can be sped up by ignoring rules for classes of depth greater than k in $H(K)$. As k is increased, inference time increases, and inference approximation error decreases. If high- and low-probability atoms are pruned at each stage, this becomes a form of coarse-to-fine probabilistic inference (Kiddon and Domingos 2011), with the corresponding approximation guarantees, and the advantage that no further approximation (like loopy belief propagation in Kiddon and Domingos (2011)) is required. It can also be viewed as a type of probabilistic default reasoning.

TML cannot compactly represent arbitrary networks with dependencies between related objects, since these are intractable, by reduction from computing the partition function of a 3-D Ising model (Istrail 2000). However, the size of a TML KB remains polynomial in the size of the network if the latter has hierarchical cluster structure, with a bound on the number of links between clusters (e.g., cities, high schools, groups of friends).

Extensions and Discussion

If we set a limit on the number of (direct) subparts a part can have, then TML can be extended to allow arbitrary dependencies within each part while preserving tractability. Notice that this is a much less strict requirement than low treewidth (cf. Theorem 3). TML can also be extended to allow some types of multiple inheritance (i.e., to allow class hierarchies that are not forests). In particular, an object may belong to any number of classes, as long as each of its predicates depends on only one path in the hierarchy. Tractability is also preserved if the class hierarchy has low treewidth (again, this does not imply low treewidth of the KB).

As a logical language, TML is similar to description logics (Baader et al. 2003) in many ways, but it makes different restrictions. For example, TML allows relations of arbitrary arity, but only among subparts of a part. Description logics have been criticized as too restricted to be useful (Doyle and Patil 1991). In contrast, essentially all known tractable probabilistic models over finite domains can be compactly represented in TML, including many high treewidth ones, making it widely applicable.

Tractability results for some classes of first-order probabilistic models have appeared in the AI and database literatures (e.g., (Jha et al. 2010)). However, to our knowledge no language with the flexibility of TML has previously been proposed.

Conclusion

TML is the first non-trivially tractable first-order probabilistic logic. Despite its tractability, TML encompasses many widely-used languages, and in particular allows for rich probabilistic inheritance hierarchies and some high-treewidth relational models. TML is arguably the most expressive tractable language proposed to date. With TML, large-scale first-order probabilistic inference, as required by the Semantic Web and many other applications, potentially becomes feasible.

Future research directions include further extensions of TML (e.g., to handle entity resolution and ontology alignment), learning TML KBs, applications, further study of the relations of TML to other languages, and general tractability results for first-order probabilistic logics.

An open-source implementation of TML is available at http://alchemy.cs.washington.edu/lite.

Acknowledgements

This research was partly funded by ARO grant W911NF-08-1-0242, AFRL contract FA8750-09-C-0181, NSF grant 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, AFRL, NSF, ONR, or the United States Government.

References

Baader, F.; Calvanese, D.; McGuinness, D.; and Patel-Scheider, P. 2003. *The Description Logic Handbook*. Cambridge, UK: Cambridge University Press.

Chi, Z. 1999. Statistical properties of probabilistic context-free grammars. *Computational Linguistics* 25:131–160.

Cooper, G. 1990. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence* 42:393–405.

Dagum, P., and Luby, M. 1993. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence* 60:141–153.

d'Amato, C.; Fanizzi, N.; and Lukasiewicz, T. 2008. Tractable reasoning with Bayesian description logics. In Greco, S., and Lukasiewicz, T., eds., *Scalable Uncertainty Management*. Berlin, Germany: Springer. 146–159.

Darwiche, A. 2003. A differential approach to inference in Bayesian networks. *Journal of the ACM* 50:280–305.

De Raedt, L.; Kimmig, A.; and Toivonen, H. 2007. Problog: A probabilistic Prolog and its application in link discovery. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, 2462–2467. Hyderabad, India: AAAI Press.

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

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

Doyle, J., and Patil, R. 1991. Two theses of knowledge representation: Language restrictions, taxonomic classification, and the utility of representation services. *Artificial Intelligence* 48:261–297.

Etherington, D., and Reiter, R. 1983. On inheritance hierarchies with exceptions. In *Proceedings of the Third National Conference on Artificial Intelligence*, 104–108. Washington, DC: AAAI Press.

Gogate, V., and Domingos, P. 2011. Probabilistic theorem proving. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, 256–265. Barcelona, Spain: AUAI Press.

Istrail, S. 2000. Statistical mechanics, three-dimensionality and NP-completeness: I. Universality of intractability for the partition function of the Ising model across non-planar surfaces. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, 87–96. Portland, OR: ACM Press.

Jaeger, M. 1994. Probabilistic reasoning in terminological logics. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning*, 305–316. Cambridge, MA: Morgan Kaufmann.

Jha, A.; Gogate, V.; Meliou, A.; and Suciu, D. 2010. Lifted inference from the other side: The tractable features. In Lafferty, J.; Williams, C.; Shawe-Taylor, J.; Zemel, R.; and Culotta, A., eds., *Advances in Neural Information Processing Systems 24*. Vancouver, Canada: NIPS Foundation. 973–981.

Kiddon, C., and Domingos, P. 2011. Coarse-to-fine inference and learning for first-order probabilistic models. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 1049–1056. San Francisco, CA: AAAI Press.

Koller, D.; Levy, A.; and Pfeffer, A. 1997. P-Classic: A tractable probabilistic description logic. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, 390–397. Providence, RI: AAAI Press.

Muggleton, S. 1996. Stochastic logic programs. In De Raedt, L., ed., *Advances in Inductive Logic Programming*. Amsterdam, Netherlands: IOS Press. 254–264.

Niepert, M.; Noessner, J.; and Stuckenschmidt, H. 2011. Log-linear description logics. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, 2153–2158. Barcelona, Spain: IJCAII.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA: Morgan Kaufmann.

Poole, D. 1993. Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence* 64:81–129.

Poon, H., and Domingos, P. 2011. Sum-product networks: A new deep architecture. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, 337–346. Barcelona, Spain: AUAI Press.

Roth, D. 1996. On the hardness of approximate reasoning. *Artificial Intelligence* 82:273–302.

Singla, P., and Domingos, P. 2007. Markov logic in infinite domains. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, 368–375. Vancouver, Canada: AUAI Press.

Wellman, M.; Breese, J. S.; and Goldman, R. P. 1992. From knowledge bases to decision models. *Knowledge Engineering Review* 7:35–53.

Zhang, N. 2004. Hierarchical latent class models for cluster analysis. *Journal of Machine Learning Research* 5:697–723.

Zhu, S., and Mumford, D. 2007. A stochastic grammar of images. *Foundations and Trends in Computer Graphics and Vision* 2:259–362.