Tractable Probabilistic Knowledge Bases with Existence Uncertainty

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

Abstract

A central goal of AI is to reason efficiently in domains that are both complex and uncertain. Most attempts toward this end add probability to a tractable subset of first-order logic, but this results in intractable inference. To address this, Domingos and Webb (2012) introduced tractable Markov logic (TML), the first tractable first-order probabilistic representation. Despite its surprising expressiveness, TML has a number of significant limitations. Chief among these is that it does not explicitly handle existence uncertainty, meaning that all possible worlds contain the same objects and relations. This leads to a number of conceptual problems, such as models that must contain meaningless combinations of attributes (e.g., horses with wheels). Here we propose a new formalism, tractable probabilistic knowledge bases (TPKBs), that overcomes this problem. Like TML, TPKBs use probabilistic class and part hierarchies to ensure tractability, but TPKBs have a much cleaner and user-friendly object-oriented syntax and a well-founded semantics for existence uncertainty. TML is greatly complicated by the use of probabilistic theorem proving, an inference procedure that is much more powerful than necessary. In contrast, we introduce an inference procedure specifically designed for TPKBs, which makes them far more transparent and amenable to analysis and implementation. TPKBs subsume TML and therefore essentially all tractable models, including many high-treewidth ones.

Introduction

Most interesting tasks in AI require both the representation of complex structure and the ability to handle uncertainty. There has historically been a divide among researchers as to the relative importance of these two requirements, resulting in one camp that emphasizes logical methods mostly derived from first-order logic and another that emphasizes statistical approaches such as Bayesian and Markov networks. There have been many attempts to connect these two using various combinations of logic and probability, usually employing some tractable subset of first-order logic like function-free horn clauses (Wellman, Breese, and Goldman 1992; Poole 1993; Muggleton 1996; De Raedt, Kimmig, and Toivonen 2007) or description logics (Jaeger 1994; Koller, Levy, and Pfeffer 1997; d'Amato, Fanizzi, and Lukasiewicz 2008; Niepert, Noessner, and Stuckenschmidt 2011). However, the tractability of the underlying logic is lost when made probabilistic, losing the advantages of the restricted representation. Arguably the most general formalism is Markov logic (Domingos and Lowd 2009), which includes both first-order logic and probabilistic graphical models as special cases. However, these approaches have seen only limited applicability, primarily because of the intractability of inference, which seems difficult to overcome since even in propositional graphical models, both exact and approximate inference are NP-hard (Cooper 1990; Dagum and Luby 1993). Additionally, probabilistic inference can be reduced to model counting, which is intractable even for heavily restricted propositional languages (Roth 1996).

Probabilistic inference is often assumed to be exponential in the treewidth of the model, but a closely related family of formalisms including arithmetic circuits (Darwiche 2003), AND/OR graphs (Dechter and Mateescu 2007), and sum-product networks (Poon and Domingos 2011) take advantage of context-specific independence to achieve efficient inference even in high treewidth models. Advances in lifted probabilistic inference, specifically Probabilistic Theorem Proving (PTP) (Gogate and Domingos 2011), allow for tractable inference in cases when it would not be possible even for these more efficient propositional structures. PTP was the primary inspiration (and inference algorithm) for TML, the first tractable first-order probabilistic logic (Domingos and Webb 2012). TML achieves tractability by imposing a hierarchical structure on its domain, in which all objects are subparts or sub-...subparts of a single top object, the possible attributes (including subparts) of an object are dependent only on its class, and relations, though they may have any arity, are allowed only between objects that are subparts of the same part.

An important issue in statistical relational AI is existence uncertainty, that is, dealing with objects and attributes that exist in some possible worlds but not in others. For instance, if a family may have either no children or one child, and children have red hair as a predicate (which is true if the child has red hair and false otherwise), then it makes no sense to have to consider the child's red hair in a family that has no children. There are representations that take existence uncertainty into account (e.g. Milch et al. 2005), but these lack tractable inference, and are quite complex. TML does not handle existence uncertainty; instead, it marginalizes over all relations involving non-existent objects, which makes for

rather odd semantics. The goal of this paper is to combine the tractability of TML with the ability to handle existence uncertainty. We accomplish this by defining a possible world as a set of objects and the truth values of the atoms involving them, rather than just a set of truth values over a universal set of objects. We show how this semantics suffices to properly handle existence uncertainty, and how to preserve tractability when it is used.

Tractable Probabilistic Knowledge Bases (TPKBs) offer all of the expressive power of TML with a straightforward handling of existence uncertainty, a cleaner, object-oriented syntax, and a dedicated inference algorithm that is easier to implement and analyze.

Syntax

A TPKB is a set of *class declarations* and *object declarations* that obey certain restrictions.

Class declarations

A class declaration for class C specifies the subparts, subclasses, and relations for C, as well as weights for the subclasses and relations and has the form

```
class C {
   subclasses S1 w1 , ... , Si wi;
   subparts C1 P1[n1], ... , Ck Pj[nj];
   relations R1(...) w1, ... , Rk(...) wk;
}
```

Each $R_l(...)$ has the form $R_l(P_a, ..., P_z)$ with each of P_a,...,P_z being one of the subparts P_1,...,P_j of C. Weights play roughly the same role as in Markov Logic, determining the probability that a given object belongs to a given subclass or that a given relation is true. For relations, weights are optional, and if a relation weight does not appear it means that the relation is *hard*, i.e. it holds for every instance of the class, otherwise, we call it *soft* in which case it may hold or not, with probability determined by its weight.

The *subparts* of C are parts which every object of class C must have and are specified by a part name P_i, the class membership of that part C_i, and the number n_i of parts of that type where n_i is optional and 1 by default. For instance an object of class TraditionalFamily might have a single Pet subpart of class Animal and two Adult subparts, each of class Person, so if the Smiths are a TraditionalFamily then we can refer to their pet as Smiths.Pet and the adults as Smiths.Adult[1] and Smiths.Adult[2]. We refer to the subparts of class C as $P(C)$.

When we specify that a class C has *subclasses* $S_1, ..., S_j$, it means that any object of class C must belong to exactly one of the subclasses $S_1, ..., S_j$, with the probability distribution over subclass memberships determined by the weights $w_1, ..., w_j$. For instance, the class Family may have subclasses TwoParentFamily, which has two Adult parts, and OneParentFamily, with one, with respective weights 1.2 and 0.3. We refer to the subclasses of class C as $S(C)$.

The *relations* of C specify what relationships may hold among the subparts of any object of class C, and with what weight. For instance, the class TraditionalFamily may have the relation Married(Adult[1], Adult[2]). Relations may apply to the object as a whole, rather than to its subparts, in which case we can omit parentheses and arguments, for example the relation Mortgage may apply to a family, and be true if that family holds a mortgage. The reason we use weights rather than probabilities is compactness - the weight of a relation rule in a class need only represent the change in the relation's log probability from whatever ancestor class it was introduced at. For instance, if TraditionalFamily and and OneParentFamily are both subclasses of class Family, and traditional and one-parent families are equally likely to hold a mortgage, then the Mortgage relation need only appear at the Family class level. However, if traditional families are more likely to hold a mortgage, then Mortgage would appear again under TraditionalFamily with positive weight.

As an example, consider the Family class.

```
class Family {
   subclasses
    TraditionalFamily 1.2,
    OnParentFamily 0.3;
   subparts
    Animal Pet,
    ...;
   relations
    Mortgage 1.7,
    ...;
}
```

Each declared subclass has its own description.

```
class TraditionalFamily {
   subclasses
    ...;
   subparts
    Person Adult[2],
    ...;
   relations
    Married(Adult[1], Adult[2]) 2.3,
    ...;
}
```

Object declarations

An object declaration creates an instance of a class. The class of an object determines its subparts and relations, so the declaration of that object specifies the names/identities of its subparts and information about its subclass memberships and relations in the form of subclass and relation facts.

A given object may have a number of possible classes, so we use the object-class pair (O, C) to refer to an object O of class C. The subclasses and relations of O determined by its class C and are referred to as the subclasses and relations of O at class C.

An object declaration has the form

C O {P_a D_a,...; S^*_b,...; $R^*_c(...)$,...}

C is the class and O is the object name. Each P_m D_m statement is a *naming fact* that assigns part P_m of O the name D_m. If P_m was introduced as C_m $P_m[n_m]$ with $n_m > 1$, the part being named must be specified using notation $P_m[l]$ $D_{m,l}$, denoting that we are naming the lth P_m of O. Note that names for subparts are not required.

$\mathtt{S}^*_\mathtt{b}, ...$ is a list of *subclass facts* and is either an empty list, a single $\mathtt{S}_\mathtt{m}$ where m is in the range $1, ..., i$ or a list of $\neg\mathtt{S}_\mathtt{m}$ statements where each m is drawn from the range $1, ..., i$ (though the list may not contain every subclass). The first means that nothing is known about object O's subclass membership, the second that O belongs to subclass $\mathtt{S}_\mathtt{m}$, the third that O does not belong to any of the subclasses $\mathtt{S}_\mathtt{b}, ...$

Similarly, each $\mathtt{R}^*_\mathtt{m}(...)$ is a *relation fact* and is either $\mathtt{R}_\mathtt{m}(...)$, denoting that the relation is known to be true, or $\neg\mathtt{R}_\mathtt{m}(...)$, denoting that the relation is known to be false. If no $\mathtt{R}^*_\mathtt{m}(...)$ appears it means that nothing is known about relation $\mathtt{R}_\mathtt{m}(...)$'s truth value.

As an example, we show the declaration for the Smiths, which is a TraditionalFamily whose two adults are married but do not have a mortgage.

TraditionalFamily Smiths {
    ...;
    Adult[1] Anna,
    Adult[2] Bob;
    ¬Mortgage,
    Married(Anna, Bob);
    ...;
}

Facts pertaining to relations at a higher class level may be stated at a lower level, but not the other way around.

Restrictions

There are a number of structural restrictions on class and object declarations necessary to ensure that inference is tractable. To describe these we must first define a notion of ancestor/descendent in the TPKB. Recall from earlier that it is useful to think of an object O at a particular class level C, referred to as $(\mathtt{O}, \mathtt{C})$. We extend this notation to include the ground atom $\mathtt{R}(\mathtt{O}, ...)$, which refers to the relation $\mathtt{R}(...)$ of class C grounded with the relevant subparts of object O.

Definition 1 *The descendants of the pair* $(\mathtt{O}, \mathtt{C})$ *in TPKB K are*

1. $(\mathtt{P_i}(\mathtt{O}), \mathtt{C_i})$ *for each* $\mathtt{P_i}$ *(with class* $\mathtt{C_i}$*) a subpart of* O *according to* C
2. $(\mathtt{O}, \mathtt{S})$ *for each* S *a subclass of* C
3. $\mathtt{R}(\mathtt{O}, ...)$ *for each relation* R *of* C
4. *The descendants of each of the above.*

There are no other descendants of $(\mathtt{O}, \mathtt{C})$*, and the first three types of descendant are referred to as children of* $(\mathtt{O}, \mathtt{C})$.

With this, the full definition of a TPKB is straightforward. The overarching purpose of the following definition is to allow the distribution defined by the TPKB to have an efficient, recursive decomposition. The first restriction gives the decomposition a starting point. The second ensures that the subparts of an object are independent, not mentioning anything they can disagree on, so that the distribution factors as a product over them. The third prevents contradictory relation truth values. The fourth prevents an object from having itself as a descendant, which is nonsensical, or from having infinitely many descendants.

Definition 2 *A set K of class and object declarations is a TPKB iff it satisfies the following conditions:*

1. *There is a single object,* $\mathtt{X_0}$*, called the* top object*, which is the sole instance of class* $\mathtt{C_0}$*, called the* top class*, such that* $(\mathtt{X_0}, \mathtt{C_0})$ *has every other* $(\mathtt{O}, \mathtt{C})$ *in K as a descendant.*
2. *For any two subparts* $\mathtt{P_i}(\mathtt{O})$ *and* $\mathtt{P_j}(\mathtt{O})$ *(introduced with classes* $\mathtt{C_i}$ *and* $\mathtt{C_j}$*) of object* O*,* $(\mathtt{P_i}(\mathtt{O}), \mathtt{C_i})$ *and* $(\mathtt{P_j}(\mathtt{O}), \mathtt{C_j})$ *share as descendants only objects* $(\mathtt{O}', \mathtt{C}')$ *such that* $\mathtt{C}'$ *has no superclasses, subclasses, or soft relations. Note that this holds even if the two subparts are introduced in different possible classes of* O.
3. *If* $(\mathtt{O}, \mathtt{C})$ *has a hard relation* $\mathtt{R}(\mathtt{O}, ...)$ *as a child, then it may have no descendant* $\mathtt{R}(\mathtt{O}, ...)$ *that is soft or* $\neg\mathtt{R}(\mathtt{O}, ...)$ *that is hard.*
4. *No object* $(\mathtt{O}, \mathtt{C})$ *may have a descendant of form* $(\mathtt{O}', \mathtt{C})$ *or* $(\mathtt{O}, \mathtt{C}')$ *with* $\mathtt{C} \neq \mathtt{C}'$ *and* $\mathtt{O} \neq \mathtt{O}'$*, which is to say that no object may have itself as a descendent, or have an descendent with the same class as itself.*

Semantics

A TPKB defines a distribution over possible worlds. However, TPKBs employ a much richer kind of possible world, which consists of a set of objects and the truth values of the atoms that involve them. Different possible worlds may contain different objects, atoms, and truth values. A key feature that makes TPKBs easy to reason about is that atoms appear in a possible world only when their arguments also appear in that world. More concretely, applying this to the earlier example with children and hair color, the hair color predicate applied to the child will only be present in worlds where the child actually exists, that is, worlds in which the family in question have a child. These semantics allow us to make probabilistic queries that assume the existence of a given object or to ask for the probability that a given object exists.

Definition 3 *The* possible objects *of a TPKB are the top object and every subpart of a possible object, under every possible class of that object. No other objects are possible.*

We define the class membership predicate Is, where $\mathtt{Is}(\mathtt{O}, \mathtt{C})$ is true if object O is of class C and false otherwise. A predicate (e.g., Is) applied to a set of arguments (e.g., O and C) is an *atom* (e.g., $\mathtt{Is}(\mathtt{O}, \mathtt{C})$). Likewise, $\mathtt{R}(\mathtt{O}, ...)$ from earlier can be thought of as a relation atom which is true if the relation holds on O (or its subparts) and false otherwise. We refer to class membership and relation atoms and their negations as *literals*.

Definition 4 *A* possible world *W of a TPKB K is an ordered pair (X,L) where X is a set of objects and L is a set of literals that satisfies the following requirements:*

1. $\mathtt{X_0} \in X$ *and* $\mathtt{Is}(\mathtt{X_0}, \mathtt{C_0}) \in L$.
2. *If* $\mathtt{O} \in X$ *and* $\mathtt{Is}(\mathtt{O}, \mathtt{C}) \in L$*, then*
   (a) *if* S *is a subclass of* C *then either* $\mathtt{Is}(\mathtt{O}, \mathtt{S}) \in L$ *or* $\neg\mathtt{Is}(\mathtt{O}, \mathtt{S}) \in L$*,*
   (b) *exactly one of the positive subclass literals* $\mathtt{Is}(\mathtt{O}, \mathtt{S})$ *is in L,*
   (c) *if* R *is a relation belonging to* C *then either* $\mathtt{R}(\mathtt{O}, ...) \in L$ *or* $\neg\mathtt{R}(\mathtt{O}, ...) \in L$*, and*
   (d) O*'s parts according to* C *are all in X, and* $\mathtt{Is}(\mathtt{O}', \mathtt{C}') \in L$ *for each part* $\mathtt{O}'$ *of* O *and corresponding class* $\mathtt{C}'$ *according to* C.

3. *No other objects are members of X and no other literals are members of L.*

A closely related notion is that of the *possible subworlds* of an object $\mathtt{O}$ at class $\mathtt{C}$, which is just the definition above except with $\mathtt{O}$ in place of $\mathtt{X}_0$ and $\mathtt{C}$ in place of $\mathtt{C}_0$.

We say that a possible world $W = (X, L)$ does not contradict a literal l if $\neg l \notin$L (note that this does not imply that $l \in$L). Recall that facts are stated in object declarations, and function either to name a subpart of an object, specify a class that that object is in or a set of classes it isn't in, or specify the truth value of a relation. Facts are implemented as follows. If K contains the assertion that relation $\mathtt{R}$ of object $\mathtt{O}$ is true (false), then $\mathtt{R}(\mathtt{O}, ...)$ ($\neg\mathtt{R}(\mathtt{O}, ...)$) is included in K. A fact declaring that $\mathtt{O}$ is of class $\mathtt{S_i}$ means that both $\mathtt{Is}(\mathtt{O}, \mathtt{S_i})$ and $\neg\mathtt{Is}(\mathtt{O}, \mathtt{S_j})$ for every $j \neq i$ are in K. However, if the fact is just that $\mathtt{O}$ is not of class $\mathtt{S_i}$, this simply means that $\neg\mathtt{Is}(\mathtt{O}, \mathtt{S_i})$ is in K. To denote the set of possible worlds of $(\mathtt{O}, \mathtt{C})$ that do not contradict any of the facts in a TPKB K we write $\mathrm{W}_K(\mathtt{O}, \mathtt{C})$.

We are now ready to specify the distribution that a TPKB K defines over its possible worlds. This distribution is constructed recursively from distributions over possible subworlds of object-class pairs in K. The unnormalized distribution ϕ_K over possible subworlds $W = (X, L)$ of $(\mathtt{O}, \mathtt{C})$ can be written recursively as $\phi_K(\mathtt{O},\mathtt{C}, W) = 0$ if $\neg\mathtt{Is}(\mathtt{O}, \mathtt{C}) \in L$ and otherwise,

$$\phi_K(\mathtt{O}, \mathtt{C}, W) = \left(\prod_{P_i \in P(\mathtt{C})} \phi_K(P_i(\mathtt{O}), \mathtt{C_i}, W)^{n_i} \right) \times \left(\sum_{\mathtt{S_j} \in S(\mathtt{C})} e^{w_j} \phi_K(\mathtt{O}, \mathtt{S_j}, W) \right) \times \left(\prod_{\mathtt{R_k} \in R(\mathtt{C})} \phi_K^R(\mathtt{O}, \mathtt{R_k}, W) \right)$$

where $\phi_K^R(\mathtt{O}, \mathtt{R_k}, L) = e^{w_k}$ if $\mathtt{R_k}(\mathtt{O}, ...) \in L$, $\phi_K^R(\mathtt{O}, \mathtt{R_k}, L) = 1$ if $\neg\mathtt{R_k}(\mathtt{O}, ...) \in L$, and $\phi_K^R(\mathtt{O}, \mathtt{R_k}, L) = 1 + e^{w_k}$ if neither $\mathtt{R_k}(\mathtt{O}, ...) \in L$ nor $\neg\mathtt{R_k}(\mathtt{O}, ...) \in L$. If $\mathtt{R_k}$ is hard, then the fact $\neg\mathtt{R_k}(\mathtt{O}, ...)$ is interpreted as $\neg\mathtt{Is}(\mathtt{O}, \mathtt{C})$. If an object has no subparts, no subclasses, or no relations, the corresponding terms of the above equation are just 1.

The unnormalized distribution over possible worlds W of TPKB K is

$$\phi_K(\mathtt{X}_0, \mathtt{C}_0, W)$$

and its partition function is

$$Z(K) = \sum_{W \in W_K(\mathtt{X}_0, \mathtt{C}_0)} \phi_K(\mathtt{X}_0, \mathtt{C}_0, W).$$

Similarly, we denote by $Z(\mathtt{O}, \mathtt{C}, K)$ the partition function with respect to the possible subworlds of $(\mathtt{O}, \mathtt{C})$.

The probability of a possible world W is then

$$P(W|K) = \frac{\phi_K(\mathtt{X}_0, \mathtt{C}_0, W)}{Z(K)}.$$

The are several types of queries we can make of K. We can ask for $P(Q|K)$, where Q is a set of facts pertaining to objects in K, which we interpret as being restricted to those possible worlds in which all objects that appear in Q (as arguments to $\mathtt{Is}$ and $\mathtt{R}$ predicates) exist. We may also ask for the probability that a given object or set of objects exist, which is simply a matter of summing the probabilities of the possible worlds that contain all of the desired objects.

Inference

All queries mentioned in the previous section (i.e. all conditional marginals) can be computed as ratios of partition functions (Gogate and Domingos 2011), as

$$P(Q|K) = \frac{Z(K \cup Q)}{Z(K)}.$$

Computing partition functions is tractable in TPKBs by the recursive structure of ϕ_K. This structure is mirrored by Algorithm 1, which calculates the subpartition function of $(\mathtt{O}, \mathtt{C})$ by recursively taking a product over subpartition functions of subparts and relations and summing over subpartition functions of subclass children. The partition function of K is computed by applying Algorithm 1 to $(\mathtt{X}_0, \mathtt{C}_0)$.

Computing the partition function when certain objects are required to exist in all possible worlds, as is required to answer queries described above, is a more challenging problem, but it can be handled efficiently given that all objects mentioned in Q are subparts of the same object $\mathtt{O}$. Notice that $P(Q|K)$, under the interpretation given in the previous section, is the sum of unnormalized probabilities of possible worlds W that contain all objects referenced in Q and in which the query atoms are true divided by the sum of unnormalized probabilities of possible worlds W' that contain all objects referenced in Q. Further, notice that

$$Z(K, (\mathtt{O}, \mathtt{C})) = Z(K) - Z(K \cup \neg\mathtt{Is}(\mathtt{O}, \mathtt{C}))$$

contains only terms corresponding to possible worlds which contain the object $\mathtt{O}$ and it is of class $\mathtt{C}$. Thus, subject to that restriction that all objects in Q are subparts of the same object, the query $P(Q|K)$, interpreted as described, can be computed as

$$P(Q, (\mathtt{O}, \mathtt{C})|K) = \frac{Z(K \cup Q, (\mathtt{O}, \mathtt{C}))}{Z(K, (\mathtt{O}, \mathtt{C}))}$$

This quantity can be computed efficiently using the machinery used to calculate normal subpartition functions, described below. Probability of existence queries can be answered tractably for single objects as $Z(K, (\mathtt{O}, \mathtt{C}))/Z(K)$, but the tractability of such queries for more than one object is an open question.

For lifting, we also compile a table of evidence pertaining to each $(\mathtt{O}, \mathtt{C})$ pair, that is, the evidence that relates to $(\mathtt{O}, \mathtt{C})$ or any of its descendants. For any given instance of a class that it encounters, it checks for the relevant evidence, then checks to see if an instance with equivalent evidence has already been processed, if so, it returns the cached value, otherwise it runs the computation and caches the result before returning. The lifting is left implicit in the pseudocode of Algorithm 1. Note that we can perform MAP inference

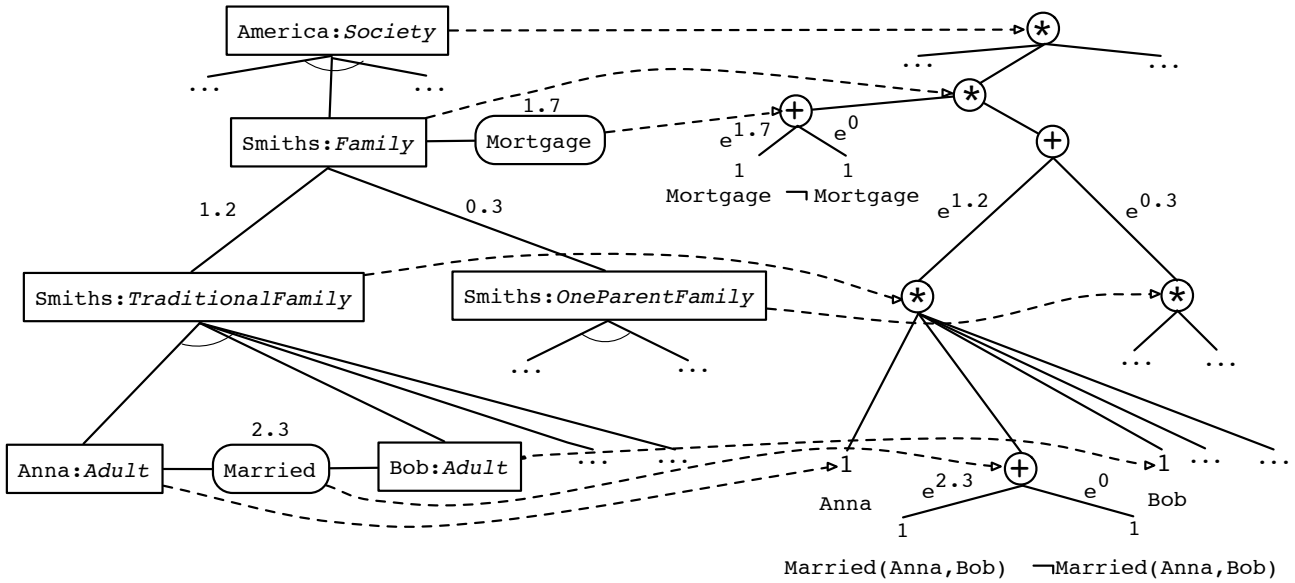

Figure 1: Example of a TPKB and its partition function Z, based on example in (Domingos and Webb 2012). On the left, rectangles represent objects and their classes, rounded rectangles represent relations, arcs joined by an angle symbol represent subpart rules, other arcs represent subclass or relation 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.

in a TPKB simply by replacing sums over subclasses with maxes and performing a traceback.

The structure of TPKBs ensures that SubZ is correct and efficient. Intuitively, restriction 2 ensures that once we know the class of an object, the subparts of that object and relations over them at that class are independent of each other and everything else in the TPKB, so we can compute their subpartition functions independently. The mutual exclusivity of subclasses functions similarly.

Theorem 1 *The partition function of TPKB K can be computed in time polynomial in the size of K.*

We omit the proof for brevity, but it is easily shown that the partition function of a TPKB can be computed in time and space that is $O(n_O(n_c + n_r))$, where n_O is the number of object declarations, n_c the number of class declarations, and n_r the number of relation rules.

Expressiveness

Despite their simple syntax and strong tractability guarantees, TPKBs are surprisingly expressive and subsume essentially all widely used tractable models and languages, including TML. This gives us many useful results automatically. For instance, it means we can represent junction trees with negligible overhead and probabilistic grammars (including PCFGs) with only polynomial overhead. Further, many high-treewidth models can also be represented as compact TPKBs. Specifically, TPKBs can compactly represent sum-product networks (SPNs), for which inference is linear in size and which are known to be able to represent many high-treewidth graphical models compactly.

This kind 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, limbs, etc. Vehicles have doors, headlights, windshields, 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 relatively compact TPKB.

One of the more interesting properties of TPKBs is that, like TML, they can represent probabilistic inheritance hierarchies and implement a kind 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, the class `Bird` has relation `Flies`, but the class `Animal` does not. If `Tux` is a `Bird`, then `Tux` may be able to fly. However, rules can have exceptions, e.g., if `Penguin` is a subclass of `Bird` and has hard relation $\neg$`Flies`. 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. TPKBs implement 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.

Proposition 1 *Let* $(\mathtt{C}_0, ..., \mathtt{C}_i, ...\mathtt{C}_n)$ *be a set of classes in a TPKB K where* $\mathtt{C}_0$ *has no superclasses and* $\mathtt{C}_n$ *has no subclasses and for* $0 \leq i < n$, $\mathtt{C}_{i+1}$ *is a subclass of* $\mathtt{C}_i$. *Let* w_i *be the weight of relation* R *in* $\mathtt{C}_i$ *and assume that no other*

Algorithm 1: SubZ(O,C, K)

Input: Object O, Class C, TPKB K
Output: Subpartition function of $(\mathtt{O}, \mathtt{C})$
if $\neg\ \mathtt{Is}(\mathtt{O}, \mathtt{C}) \in K$ **then**
  return *0*
else
  Z = 0
  for $\mathtt{S_i} \in$ *S(*C*)* **do**
    $\mathrm{Z} \leftarrow \mathrm{Z} + \mathrm{e}^{w_i}$ SubZ(O,$\mathtt{S_i}$,K)
  if *Z = 0* **then**
    $\mathrm{Z} \leftarrow 1$
  for $\mathtt{P_j} \in$ *P(*C*)* **do**
    $\mathrm{Z} \leftarrow \mathrm{Z} \times (\mathrm{SubZ}(\mathtt{P_j}(\mathtt{O}),\mathrm{C}_j,\mathrm{K}))^{n_j}$
  for $\mathtt{R_j} \in$ *R(*C*)* **do**
    if $\mathtt{R_j}(\mathtt{O}, ...) \in K$ **then**
      $\mathrm{Z} \leftarrow \mathrm{Z} \times e^{w_j}$
    else if $\neg\mathtt{R_j}(\mathtt{O}, ...) \in K$ **then**
      $\mathrm{Z} \leftarrow \mathrm{Z} \times 1$
    else
      $\mathrm{Z} \leftarrow \mathrm{Z} \times (1 + e^{w_j})$
  return Z

set of classes in K mention R. If O *is an object of class* $\mathtt{C_n}$*, then, the unnormalized probability of* $\mathtt{R}(\mathtt{O}, ...)$ *assuming that* O *exists is* $\exp(\sum_{i=0}^{n} w_i)$.

Extensions and Future Work

There are a number of easily realized extensions to TPKBs. For instance, they can be extended to allow some types of multiple inheritance, that is, classes with multiple superclasses, by allowing a class to have any number of superclasses so long as each of its relations appears in at most one superclass. Other topics for future work include extending TPKBs with more advanced notions of existence uncertainty (e.g. Poole 2007), numeric attributes, learning TPKBs, more advanced forms of inference that can work with and exploit existence uncertainty, applications, and general tractability results for first-order logics.

Conclusion

TPKBs are the first non-trivially tractable first-order probabilistic logic that also handles existence uncertainty. They inherit all the strengths of their predecessor, TML, but are much easier to use, reason about, and implement. TPKBs open up the prospect of large-scale first-order probabilistic inference that is both efficient and user-friendly. An open-source implementation of TPKBs is available at http://alchemy.cs.washington.edu/lite.

Acknowledgements: This research was partly funded by ARO grant W911NF-08-1-0242, AFRL contracts FA8750-09-C-0181 and FA8750-13-2-0019, 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

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*. 146–159. Springer.

Darwiche, A. 2003. A differential approach to inference in Bayesian networks. *JACM* 50:280–305.

De Raedt, L.; Kimmig, A.; and Toivonen, H. 2007. Problog: A probabilistic Prolog and its application in link discovery. *IJCAI*, 2462–2467.

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

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

Domingos, P., and Webb, W.A. 2012. A tractable first-order probabilistic logic. *AAAI* 2012.

Etherington, D., and Reiter, R. 1983. On inheritance hierarchies with exceptions. *NCAI*, 104–108.

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

Jaeger, M. 1994. Probabilistic reasoning in terminological logics. *ICPKRR*, 305–316.

Koller, D.; Levy, A.; and Pfeffer, A. 1997. P-Classic: A tractable probabilistic description logic. *NCAI*, 390–397.

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

Milch, B.; Marthi, B.; Russell, S.; Sontag, D.; Ong, D. L; and Kolobov, A. 2005. BLOG: Probabilistic Models with Unknown Objects. *IJCAI*, 1352-1359.

Niepert, M.; Noessner, J.; and Stuckenschmidt, H. 2011. Log-linear description logics. *IJCAI*, 2153–2158.

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

Poole, D. 2007. Logical generative models for probabilistic reasoning about existence, roles, and identity. *AAAI*.

Poon, H., and Domingos, P. 2011. Sum-product networks: A new deep architecture. *UAI*, 337–346.

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

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