First-Order Probabilistic Languages: Into the Unknown

Brian Milch[1] and Stuart Russell[2]

[1] Computer Science and AI Laboratory
Massachusetts Institute of Technology
32 Vassar St. Room 32-G480
Cambridge, MA 02139, USA
milch@csail.mit.edu

[2] Computer Science Division
University of California at Berkeley
Berkeley, CA 94720-1776, USA
russell@cs.berkeley.edu

Abstract. This paper surveys *first-order probabilistic languages* (FOPLs), which combine the expressive power of first-order logic with a probabilistic treatment of uncertainty. We provide a taxonomy that helps make sense of the profusion of FOPLs that have been proposed over the past fifteen years. We also emphasize the importance of representing uncertainty not just about the attributes and relations of a fixed set of objects, but also about what objects exist. This leads us to *Bayesian logic*, or *BLOG*, a language for defining probabilistic models with unknown objects. We give a brief overview of BLOG syntax and semantics, and emphasize some of the design decisions that distinguish it from other languages. Finally, we consider the challenge of constructing FOPL models automatically from data.

1 Introduction

Many real-world tasks, from identifying objects in images to extracting facts about people from text documents, require probabilistic reasoning about many related objects. These tasks often require weighing competing pieces of evidence, so some form of probabilistic reasoning is necessary. However, the number of random variables needed to describe such a scenario grows with the number of objects. Thus, *propositional* probabilistic languages such as Bayesian networks (BNs) — which describe a fixed set of random variables, and specify dependencies and probability distributions for each variable individually — are insufficient.

To represent probabilistic models for such tasks, we need *first-order probabilistic languages* (FOPLs): probabilistic modeling languages that can model large families of random variables compactly by abstracting over objects. A significant number of FOPLs have been proposed over the last fifteen years or so. In Section 2, we organize many of the proposed languages into a taxonomy, attempting to clarify the major ways in which they differ from one another. An important desideratum for FOPLs is the ability to represent uncertainty about the number of objects that exist and the correspondence between observations and underlying objects. In Section 3, we focus on a FOPL that we developed with this goal in mind: *Bayesian logic*, or BLOG [13]. In addition to discussing its syntax and semantics, we highlight some of BLOG's distinctive design features.

Section 4 turns to the question of learning FOPL models. Parameter estimation for FOPL models is well-understood, and there has been considerable work on learning the dependency structure of such models. However, an even more challenging problem remains open: how to automatically hypothesize new functions or predicates, or even new types of objects, to explain the data.

2 A Taxonomy of FOPLs

2.1 Outcome spaces

The most basic way in which certain FOPLs differ from others is in their outcome spaces: that is, the sets of outcomes to which they assign probabilities. In most FOPLs, the outcome space is a set of *relational structures*, which specify a set of objects and some relations (or functions) on these objects. To make this idea more concrete, consider the following pedagogical example:

Example 1. Suppose we are given a list of papers that have been submitted to a conference over several years. Each paper is either accepted or not accepted. We are also given a list of researchers, which includes the primary author of each paper. Suppose that each researcher can be classified as brilliant or not brilliant, and the probability that a paper is accepted depends on whether its primary author is brilliant or not. Given the authorship and acceptance status of certain papers, we would like to predict which other papers will be accepted.

A relational structure for Example 1 specifies a set of papers, a set of researchers, a unary predicate Accepted that applies to papers, a unary predicate Brilliant that applies to researchers, and a function PrimaryAuthor that maps papers to researchers. Depending on the what aspects of the scenario are known in advance, the outcomes may share some *relational skeleton* [3]: for instance, they may all have the same sets of objects and the same PrimaryAuthor function.

One reason for the diversity of FOPLs is that different communities talk about relational structures in different ways. In logic, a relational structure is a logical model structure: a domain of discourse plus an interpretation of a logical language over that domain. Exanples of FOPLs that define distributions over logical model structures include Halpern's logic of probability on possible worlds [5], relational Bayesian networks (RBNs) [7], PRISM [34], Markov logic [33] and BLOG [13]. Relational structures can also be thought of as instances of a relational database schema. This view has led to a distinct set of FOPLs, including probabilistic relational models (PRMs) [10, 3] and relational Markov networks (RMNs) [36].

The statistics community thinks of possible outcomes in yet another way: as instantiations of a set of random variables. The statistical analogue of a unary

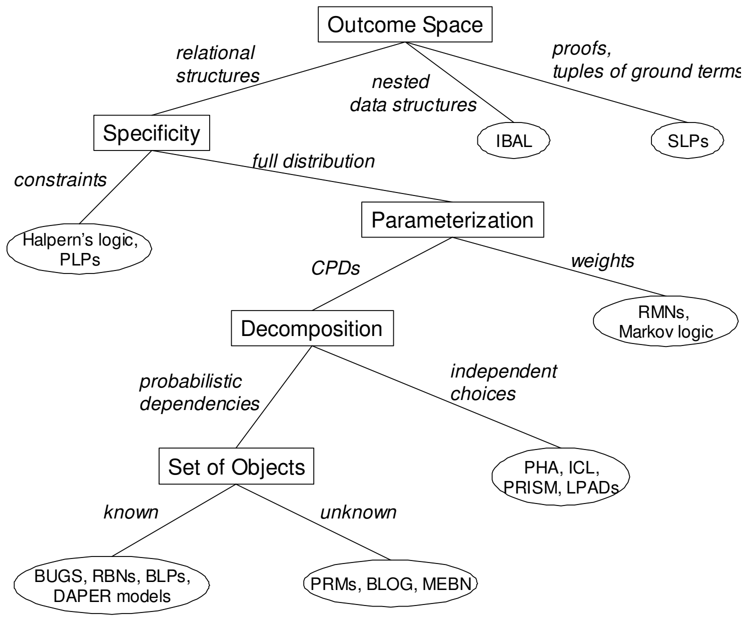

Fig. 1. A taxonomy of first-order probabilistic languages.

predicate Accepted is a family of binary-valued random variables A_i, indexed by natural numbers i that represent papers. Similarly, the function PrimaryAuthor can be represented as an indexed family of random variables P_i, whose values are natural numbers representing researchers. Thus, instantiations of a set of random variables can represent relational structures. Indexed families of random variables are a basic modeling element in the BUGS system [37], where they are represented graphically using "plates" that contain co-indexed nodes.

There are two well-known FOPLs whose possible outcomes are not relational structures in the sense we have defined. One is stochastic logic programs (SLPs) [17]. An SLP defines a distribution over proofs from a given logic program. If a particular goal predicate R is specified, then an SLP also defines a distribution over tuples of logical terms: the probability of a tuple $(t_1, \ldots, t_k)$ is the sum of the probabilities of proofs of $R(t_1, \ldots, t_k)$. SLPs are useful for defining distributions over objects that can be encoded as terms, such as strings or trees; they can also emulate more standard FOPLs [31]. The other prominent FOPL with a unique outcome space is IBAL [26], a programming language that allows stochastic choices. An IBAL program defines a distribution over *environments* that map symbols to values. These values may be individual symbols, like the values of variables in a BN; but they may also be other environments, or even functions.

This analysis defines the top level of the taxonomy shown in Figure 1. In the rest of the paper, we will focus on languages that define probability distributions over relational structures.

2.2 Specificity

Among the FOPLs that define distributions over relational structures, the first distinction we can draw is between languages that fully define a distribution, and those that only impose constraints on a distribution. As an example of the latter type, Halpern's logic of probability on possible worlds [5] allows statements such as $\forall x\, P(\mathsf{Brilliant}(x)) = 0.3$. Such statements just specify particular marginal probabilities: in general, they do not fully define a distribution. Probabilistic logic programs (PLPs) [21] are essentially a version of Halpern's language restricted to Horn clauses, although one can obtain a full distribution from a PLP by finding the maximum entropy distribution consistent with the PLP's constraints [12]. The FOPLs that we will discuss from here on all define probability distributions completely, just as BNs and Markov networks do.

2.3 Conditional probabilities versus weights

In the propositional realm, Bayesian networks are directed models that specify a conditional probability distribution (CPD) for each variable given some parent variables, whereas *Markov networks* are undirected models that use weights to define the relative probabilities of instantiations. This distinction carries over to the first-order case. The CPD-based or directed FOPLs include BUGS [37], PRISM [34], PRMs [10], Bayesian logic programs (BLPs) [8], and BLOG [13]. The principal weight-based or undirected formalisms are relational Markov networks [36] and Markov logic [33].

To understand the trade-offs between directed and undirected representations, consider a directed FOPL model for Example 1 with the following CPDs:

$\mathsf{Brilliant}(r) \sim$

True	False
0.2	0.8

, $\mathsf{Accepted}(p) \sim$

	Accepted(p)	
Brilliant(PrimaryAuthor(p))	True	False
True	0.8	0.2
False	0.3	0.7

If the relational skeleton contains just one paper $\mathsf{Pub1}$ and just one researcher $\mathsf{Res1}$, with $\mathsf{PrimaryAuthor}(\mathsf{Pub1}) = \mathsf{Res1}$, then this model defines the BN in Figure 2(a). If there are two papers by $\mathsf{Res1}$, we get the BN in Figure 2(b).

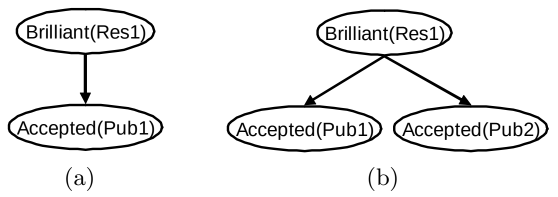

Fig. 2. BNs defined by a directed FOPL model whose relational skeleton includes (a) one paper, or (b) two papers.

This directed model has several attractive properties. First, the parameters have clear interpretations as prior and conditional probabilities, and can be estimated from fully observed data using elementary formulas. Even more importantly, the parameters are *modular*: they reflect causal processes that apply regardless of the relational skeleton. Thus, if we estimate the parameters using only examples with one paper per researcher, we will get the same CPDs that we would get from examples with two papers per researcher. We can also exploit a related modularity property when performing inference: rather than doing inference on the whole BN defined by the FOPL model, it suffices to use the subgraph consisting of the query and evidence nodes and their ancestors [22].

The drawback of directed models is that they must not have any cycles. This requirement is especially burdensome in FOPLs, because we must ensure that the probability model is acyclic for every relational skeleton in some class. Also, certain properties of relations are difficult to describe without creating cycles: for instance, it is not easy to specify that for all people a and b, if $\mathsf{Likes}(a, b)$ is true than $\mathsf{Likes}(b, a)$ is probably true as well.

Undirected models, on the other hand, have no acyclicity constraints. An undirected model is defined by *potential functions* that assign weights to instantiations based on some subsets of the random variables. The weight of an instantiation is the product of the weights assigned by all the potentials; these weights are then normalized to yield a probability distribution. In the first-order case, a model specifies *potential function templates* that apply to all sets of variables that satisfy certain conditions. For instance, in Example 1, we can include a potential template that applies to $\mathsf{Brilliant}(r)$ for every researcher r, and another template that applies to $\{\mathsf{Brilliant}(r), \mathsf{Accepted}(p)\}$ for all pairs (r, p) such that $\mathsf{PrimaryAuthor}(p) = r$. Figure 3 shows the Markov networks that result when these templates are applied to relational skeletons with one or two papers.

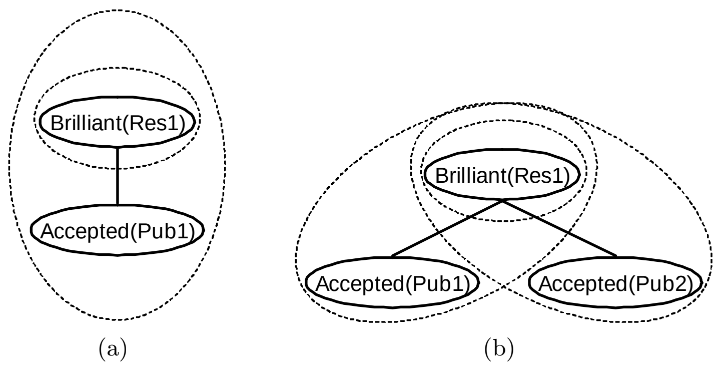

Fig. 3. Markov networks defined by an undirected FOPL model whose relational skeleton includes (a) one paper, or (b) two papers. Dotted ovals indicate sets of variables that are in the domain of the same potential function.

This undirected FOPL model can reproduce the distributions defined by our directed model above: we can simply set the potential on $\mathsf{Brilliant}(r)$ equal to the CPD for $\mathsf{Brilliant}(r)$, and the potential on $\{\mathsf{Brilliant}(r), \mathsf{Accepted}(p)\}$ to the CPD for $\mathsf{Accepted}(p)$. However, suppose we estimate our parameters solely on examples with one paper per researcher (recall that this caused no problems in the directed case). Our learning algorithm may arrive at the following parameterization for the network in Figure 3(a), defining the same joint distribution as the CPD-like parameterization:

$\forall\, r$:

Brilliant(r)	
True	False
1	1

$\forall\, (r, p)$ s.t. $\mathsf{PrimaryAuthor}(p) = r$:

	Accepted(p)	
Brilliant(r)	True	False
True	0.16	0.04
False	0.24	0.56

The meanings of the parameters in these potential templates are no longer so obvious. The potential on $\mathsf{Brilliant}(r)$ is all 1's, but the marginal distribution on $\mathsf{Brilliant}(\mathsf{Res1})$ in Figure 3(a) still ends up being $(0.2, 0.8)$. This is because the event $\mathsf{Brilliant}(\mathsf{Res1}) = \mathsf{True}$ receives a total weight of $0.16 + 0.04 = 0.2$ in the potential over $\{\mathsf{Brilliant}(\mathsf{Res1}), \mathsf{Accepted}(\mathsf{Pub1})\}$. This coupling between potentials means that maximum-likelihood parameters for Markov networks cannot be found with simple formulas: one must use a gradient-based optimization algorithm [33].

Now consider what happens if we apply the undirected probability model above to the two-paper network in Figure 3(b). Then the template for pairs (r, p) such that $\mathsf{PrimaryAuthor}(p) = r$ applies twice, and the marginal distribution on $\mathsf{Brilliant}(\mathsf{Res1})$ ends up being proportional to $(0.2^2, 0.8^2)$, which normalizes to about $(0.06, 0.94)$. If the actual probability that a researcher is brilliant is 0.2, then these parameters are sub-optimal: we would not learn them if we had instances with two papers in our training set.[3] Thus, unlike in the directed case, we need to ensure that the relational skeletons in our training set reflect the diversity of relational skeletons that we may encounter in test data.

2.4 Independent choices versus probabilistic dependencies

The category of CPD-based languages for defining complete distributions over relational structures is still quite large. However, one of the languages we have mentioned, namely PRISM [34], stands out from the rest in that it represents only deterministic dependencies and independent random choices. That is, each variable either has no parents, or has a deterministic CPD. Other FOPLs that

[3] The problem actually gets worse if we eliminate the apparently redundant potential template on $\mathsf{Brilliant}(r)$: then there is no parameterization that yields the desired distribution for all relational skeletons.

take this approach include probabilistic Horn abduction [27], independent choice logic [28] and logic programs with annotated disjunctions (LPADs) [39].[4]

It may not be immediately obvious how independent choices could suffice to represent all the randomness in a probabilistic model. First, consider the directed model that we defined in the previous section for Example 1. To sample a value for an $\mathsf{Accepted}(p)$ variable in that model, we flip a coin with a bias determined by the value of $\mathsf{Brilliant}(\mathsf{PrimaryAuthor}(p))$. The trick used in PRISM is, conceptually, to flip coins for all possible values of $\mathsf{Brilliant}(\mathsf{PrimaryAuthor}(p))$ ahead of time, and then choose which coin flip to use based on the actual value of $\mathsf{Brilliant}(\mathsf{PrimaryAuthor}(p))$. The initial coin flips can be represented by an auxiliary predicate $\mathsf{Accepted\_given\_Brilliant}(p, b)$, which represents the value that $\mathsf{Accepted}(p)$ would have if $\mathsf{Brilliant}(\mathsf{PrimaryAuthor}(p))$ were equal to b. The predicate $\mathsf{Accepted\_given\_Brilliant}$ has the following probability model:

$$\mathsf{Accepted\_given\_Brilliant}(p, \mathsf{True}) \sim \begin{array}{|cc|}\hline \mathsf{True} & \mathsf{False} \\ \hline 0.8 & 0.2 \\ \hline\end{array}$$

$$\mathsf{Accepted\_given\_Brilliant}(p, \mathsf{False}) \sim \begin{array}{|cc|}\hline \mathsf{True} & \mathsf{False} \\ \hline 0.3 & 0.7 \\ \hline\end{array}$$

Now the probability model for $\mathsf{Accepted}$ is deterministic (note that we are treating $\mathsf{Brilliant}$ here as a Boolean function, yielding values in $\{\mathsf{True}, \mathsf{False}\}$):

$$\mathsf{Accepted}(p) = \mathsf{Accepted\_given\_Brilliant}(p, \mathsf{Brilliant}(\mathsf{PrimaryAuthor}(p)))$$

The advantage of this technique is that it completely separates the logical and probabilistic parts of the language. This separation can be exploited to obtain efficient algorithms for certain tasks [35]. However, this decomposition often makes the representation considerably less intuitive.

2.5 Known versus unknown objects

The last distinction in our taxonomy regards whether a language requires the set of objects to be specified in the relational skeleton, or allows the set of objects to be unknown. To motivate our discussion of unknown objects, consider the following example, based on our earlier work on citation matching [24].

Example 2. Suppose we are given a set of citation strings extracted from the "References" sections of online papers. These citations use a variety of different formats; they use initials and abbreviations in different places; and they contain typographical errors. The task is to reconstruct a database of publications and researchers who are referred to in the citations. This database should contain just one record for each publication and each researcher, including all the true attributes of these entities that can be inferred from the citations.

[4] In LPADs, the independent choices do not set the values of ground atoms directly; instead, there is one choice for each ground disjunctive clause, and this choice determines which element of the clause's head will be entailed by the clause's body.

In this example, the sets of publications and researchers that underlie the citations are not known in advance. Furthermore, we do not know which citations refer to which publications, or which substrings of citations refer to which researchers. If Cit1 and Cit2 are two citations and PubCited is a function that maps citations to the publications they refer to, then the ground terms PubCited(Cit1) and PubCited(Cit2) may or may not denote the same object.

Most FOPLs assume that the objects are in one-to-one correspondence with a given set of constant symbols, or with the ground terms of the language. The CPD-based FOPLs that make such assumptions include BUGS [37] (where the objects correspond to specified sets of natural numbers), RBNs [7], BLPs [8], and directed acyclic probabilistic entity-relationship (DAPER) models [6]. One can model unknown objects to some extent in these languages by adding an Exists predicate, but one still has to specify all the objects that could exist, and craft the probability models so that objects for which Exists is false cannot serve as values for functions or have any probabilistic influence on other objects.

There are three prominent languages that make unknown objects a fundamental part of their semantics. One of these is PRMs, which allow uncertainty about the number of objects that stand in a given relation to an existing object (*e.g.*, papers written by a researcher) [10], about whether there exists an object that stands in certain relations to several other objects (*e.g.*, a role for a given actor in a given movie) [4], and about the total number of objects of a given type [24]. However, PRMs do not have a unified syntax that supports all these types of uncertainty. The language of multi-entity Bayesian networks (MEBN) [11] does have a consistent syntax, and incorporates Exists variables as part of its semantics. But MEBN still requires the modeler to list all objects that might exist. The third language that supports unknown objects is BLOG, which we discuss in the next section.

3 Bayesian Logic (BLOG)

In this section we give an informal overview of Bayesian logic (BLOG) [13], a language that facilitates defining probability distributions over relational structures with varying sets of objects. In fact, BLOG's design makes it an attractive choice even for scenarios that do not involve unknown objects.

3.1 Syntax

A BLOG model defines a probability distribution over model structures of a typed first-order language. To this end, the model defines a typed first-order language for a particular scenario; specifies certain nonrandom aspects of the scenario; and specifies a probability model for the remaining aspects. The probability model can be thought of as describing a generative process for constructing a possible world. This process has two kinds of steps: steps that set the value of a function[5] on some objects, and steps that add new objects to the world.

[5] We treat predicates as Boolean functions.

```
 1 type Res; type Pub; type Cit;

 2 guaranteed Cit Cit1, Cit2, Cit3, Cit4;

 3 #Res ~ NumResearchersPrior;
 4 random String Name(Res r) ~ NamePrior;

 5 #Pub ~ NumPublicationsPrior;
 6 random String Title(Pub p) ~ TitlePrior;
 7 random NaturalNum NumAuthors(Pub p) ~ NumAuthorsPrior;
 8 random Res NthAuthor(Pub p, NaturalNum n)
 9     if (n < NumAuthors(p)) then ~ Uniform({Res r});

10 random Pub PubCited(Cit c) ~ Uniform({Pub p});
11 random String Text(Cit c)
12     ~ FormatModel(Title(PubCited(c)),
13                   {n, Name(NthAuthor(PubCited(c), n)) for
14                    NaturalNum n : n < NumAuthors(PubCited(c))});
```

Fig. 4. A BLOG model for citation matching.

Figure 4 gives a complete BLOG model for Example 2. We will begin by walking through the generative process defined by this model; then we will discuss the syntax in more detail. Line 1 of Figure 4 says that there are three types of objects in this scenario; then line 2 asserts that four citations are guaranteed to exist. Line 3 begins the random part of the generative process: a random number of researchers are added to the world, with this number being sampled according to `NumResearchersPrior`. Then, for each researcher r, a name is sampled from `NamePrior`. Line 5 adds a random number of publications to the world. For each publication, the title and the number of authors are sampled from appropriate priors (lines 6–7). Then for each publication p and each number $n < \mathsf{NumAuthors}(p)$, a researcher is sampled uniformly at random to serve as the nth author of p. In line 10 we get to the model for citations: for each citation c, the publication cited is sampled uniformly from the set of publications. Finally, the text of each citation is sampled according to a format model that conditions on the title of the cited paper and the names of its authors.

The syntax in Figure 4 may seem complicated, but in fact it can be explained fairly simply. A BLOG model is a series of statements, each ending with a semicolon. The three statements in line 1 are *type declarations*; a BLOG model can also include *function declarations* that specify the type signatures of functions (these are necessary if we use a function before we define its probability model). Line 2 is a *guaranteed object statement* that asserts the existence of a set of distinct objects, and assigns a constant symbol to each one. Along with *nonrandom function definitions*, which do not appear in this model, guaranteed object statements define a relational skeleton. The probabilistic portion of the model consists of *number statements*, which describe steps where objects are added to the world, and *dependency statements*, which describe how values are assigned to functions. These six types of statements constitute the full syntax of BLOG.

Dependency statements and number statements have a rich syntax of their own. A BLOG model must contain exactly one dependency statement for each random function. If f is a function with return type τ_0 and argument types $\tau_1, \ldots, \tau_k$, then a dependency statement for f has the following general form:

```
random $\tau_0$ $f(\tau_1\ x_1, \ldots, \tau_k\ x_k)$
    if $cond_1$ then  $\sim$  $cpd_1(a_{1,1}, \ldots, a_{1,m_1})$
    elseif $cond_2$ then  $\sim$  $cpd_2(a_{2,1}, \ldots, a_{2,m_2})$
    ⋮
    else  $\sim$  $cpd_n(a_{n,1}, \ldots, a_{n,m_n})$;
```

The conditions $cond_1, \ldots, cond_{n-1}$ are arbitrary first-order formulas that can use the variables $x_1, \ldots, x_k$. The *elementary CPDs* $cpd_1, \ldots, cpd_n$ can be thought of as functions that take in a list of arguments $a_1, \ldots, a_m$ and return a probability distribution over objects of f's return type. More technically, they are the names of Java classes that implement a certain interface. The arguments a can be logical terms, such as $\mathsf{Title}(\mathsf{PubCited}(c))$; *set expressions*, such as $\{\mathsf{Pub}\ p\}$ or $\{\mathsf{Pub}\ p : \mathsf{Venue}(p) = \mathsf{ILP}\}$; or *tuple multiset expressions*, such as the one in lines 13–14, which defines a multiset of pairs consisting of an author number and a name.

Obviously, not all the dependency statements in Figure 4 have this full-fledged if-then-else form: we allow a number of abbreviations. The expression "`if` $cond_1$ `then`" can be omitted if $cond_1$ is simply True. Also, if a statement contains some non-trivial conditions but omits the `else` clause, then the function gets a default value of null when none of the conditions are satisfied. This default convention is exploited in line 9.

The number statements in Figure 4 are very simple, but in general, they can have the same syntax as dependency statements. The only difference is that the expression "`random` $\tau_0\ f(\tau_1\ x_1, \ldots, \tau_k\ x_k)$" is replaced with $\#\tau$, where τ is the type of object being generated.[6] Thus, the number of objects that exist can depend on other variables.

3.2 Semantics

We have given an intuitive semantics for BLOG in terms of a random process that generates possible worlds. However, BLOG also has a more formal, declarative semantics [13]. A BLOG model defines a set of *basic random variables*: a *number variable* for each number statement, and a *function application variable* for each random function and each tuple of arguments that exist in any possible world. The distribution defined by a BLOG model can be represented as a *contingent Bayesian network* (CBN) [14] over these basic variables.

[6] In fact, BLOG supports more complex number statements to model scenarios where objects generate other objects [13].

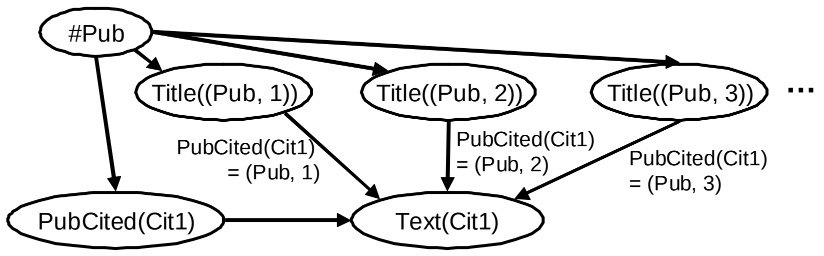

Fig. 5. A contingent Bayesian network for a simplified version of the BLOG model in Figure 4. This simplified model has just one citation and does not include researchers.

A CBN is a directed graphical model in which the edges are labeled with conditions that specify when they are active. For example, Figure 5 shows a CBN for a simplified version of the citation model from Figure 4. Note that the node $\mathsf{Text}(\mathsf{Cit1})$ has infinitely many parents, because it may depend on the title of any publication. Most treatments of Bayesian networks do not provide well-definedness results for networks that contain infinite parent sets. However, the edge labels in Figure 5 allow us to see that at most two edges into $\mathsf{Text}(\mathsf{Cit1})$ can be active in any single outcome: one edge from $\mathsf{PubCited}(\mathsf{Cit1})$, and one edge from $\mathsf{Title}(p)$ where $p = \mathsf{PubCited}(\mathsf{Cit1})$. It turns out that one can obtain stronger well-definedness results for CBNs than for standard BNs — well-defined CBNs can even contain cycles, as long as some edges on each cycle have mutually contradictory labels [14]. In [13], we build on these results to give conditions under which a BLOG model is guaranteed to define a unique probability distribution over possible worlds.

3.3 Design features

Distributions over function values. A dependency statement in BLOG can define a probability distribution for a function, such as NthAuthor or PubCited. By contrast, many FOPLs — including PRISM [34], relational Bayesian networks [7], DAPER models [6], and Markov logic [33] — only express uncertainty about the values of predicates. In a purely logical context, this limitation might be innocuous: one can simply write $\mathsf{PubCited}(c, p)$ rather than $\mathsf{PubCited}(c) = p$. However, using a predicate to represent a *random* functional relationship yields an unnecessarily complicated probability model. Instead of a single object-valued random variable $\mathsf{PubCited}(\mathsf{Cit1})$, one ends up with many binary random variables $\mathsf{PubCited}(\mathsf{Cit1}, p)$ — and all these binary variables are mutually dependent, because exactly one of them must have the value True.

Explicit aggregation. In BLOG, we allow our elementary CPDs to take multisets as arguments. This eliminates the need for separate "combination functions", as used, for example, in BLPs [8]: the burden of aggregation is now on the CPDs.

Contingent dependencies. Dependency statements make a clear distinction between the values that are passed into elementary CPDs, and the logical formulas in "if" statements and set expressions, which determine what CPD to apply and what values to pass into it. This contrasts with the situation in BLPs [8], where any logical atom that is included in a clause to govern when the clause applies is also passed into the CPD for the head variable. Also, unlike in the probabilistic knowledge bases of Ngo and Haddaway [22] or the logical BNs of Fierens *et al.* [2], the conditions that govern dependencies in a BLOG model do not have to be nonrandom.

The contingent dependency structure that a BLOG model makes explicit can be exploited in sampling-based algorithms for approximate inference [14, 15]. The basic insight is that algorithms such as likelihood weighting or Markov chain Monte Carlo only need to instantiate variables that are *context-specifically relevant* for the query: that is, variables that are known to be relevant given the other instantiated variables. Crucially, it is not necessary to instantiate all the variables that *might* be relevant for a query in some circumstances — this would be an infinite set if the query were about $\mathsf{Text}(\mathsf{Cit1})$ in Figure 5.

4 Learning in FOPLs

4.1 Parameters

Parameter estimation for FOPLs is well understood: the goal is to find parameters that maximize the likelihood of the data, or that have maximal *a posteriori* probability given the data and some Bayesian prior. As we noted in Section 2.3, parameter estimation tends to be computationally straightforward in directed models with complete data. For undirected models, and for directed models with unobserved variables, parameter estimation becomes computationally difficult as the number of random variables increases. However, this difficulty is common to all large probabilistic models, not just models defined by FOPLs.

4.2 Dependency structure

Learning the dependency structure of FOPL models, on the other hand, raises issues that do not arise in the propositional case. In a Bayesian network, the dependency structure can be represented simply as a list of parents for each variable. But in a FOPL, we need a first-order representation of each variable's parent set. For instance, in Example 1, we need to learn that for all papers p, $\mathsf{Accepted}(p)$ depends on $\mathsf{Brilliant}(\mathsf{PrimaryAuthor}(p))$. Also, a variable often depends on a whole class of parents in a symmetrical way. In Example 1, if we take multiple authors into account by adding a predicate $\mathsf{HasAuthor}(p, r)$, then $\mathsf{Accepted}(p)$ might depend on some *aggregation function* of the variables $\{\mathsf{Brilliant}(r) : \mathsf{HasAuthor}(p, r)\}$, such as their average value, or the number that have the value True.

A well-known paper by Friedman *et al.* [3] introduces a method for learning the structure of a probabilistic relational model. In that work, a parent set is represented as a set of attribute chains, and parents that are reachable by the same attribute chain are aggregated together using one of a pre-defined library of aggregation functions. However, there are many other kinds of structures that we would like to be able to learn (and that are expressible in BLOG): for example, a variable may depend on different sets of parents in different contexts, or the parents may be selected using criteria other than single slot chains (*e.g.*, in Figure 4, $\mathsf{Text}(c)$ depends on those variables $\mathsf{Name}(\mathsf{NthAuthor}(\mathsf{PubCited}(c), n))$ for $n < \mathsf{NumAuthors}(\mathsf{PubCited}(c))$). Also, aggregation functions might be constructed from more primitive components rather than being chosen from a library.

There has been significant work on learning more complex selection and aggregation rules for estimating the conditional distribution of a single variable [30, 20, 25, 38]. However, there does not seem to be any work so far on using these sophisticated techniques to learn directed, acyclic FOPL models for multiple variables (although they have been used to learn cyclic directed models called *dependency networks* [19]). There has been other work on structure learning for stochastic logic programs [18] and for Markov logic [9], building on inductive logic programming techniques for searching over logical formulas. We are interested in developing structure learning algorithms for BLOG models; this line of work might begin with restricted versions of the BLOG dependency statement syntax.

4.3 Functions and types

Algorithms that learn the dependency structure of FOPL models typically assume that the functions, predicates, and object types are given. But as John McCarthy pointed out in his invited talk at ILP 2006, hypothesizing new objects and relations to explain observed data is a fundamental part of human learning. For instance, it would be useful to hypothesize a binary predicate on researchers, which might be called $\mathsf{Colleagues}(r_1, r_2)$, to explain how researchers co-occur in author lists. There has been considerable work in the inductive logic programming literature on *predicate invention* [16], but it is not yet clear how to generalize it to the probabilistic case. Inventing a new random function (or predicate) in a FOPL model corresponds to discovering a whole family of hidden variables. The task of discovering hidden variables in Bayesian networks has been investigated by Elidan and Friedman [1]; recently, Revoredo *et al.* [32] have taken some steps toward applying these ideas to BLPs.

It may also be possible to improve probabilistic models by automatically hypothesizing new types of objects. For example, to explain recurring substrings that come after the titles in citations, a system might hypothesize objects that could be called conferences. One simple form of type invention that has already been implemented involves clustering some observed objects, and treating the clusters as a new type of object [29]. In this case, the hypothesized type plays a predetermined role in the probabilistic model; in the general case, we would like a system to discover what roles need to be filled. Otero and Muggleton [23] sketch a learning algorithm for purely logical models that addresses this problem.

5 Conclusion

First-order probabilistic languages combine a principled treatment of uncertainty with the ability to describe large models formally and concisely. We hope the taxonomy of FOPLs given in this paper will make the wide landscape of proposed languages less daunting, and help researchers choose the most appropriate FOPL for a given application. This paper has highlighted two major areas of FOPL research: the development of languages such as BLOG, which support reasoning about the unknown objects that underlie a particular data set; and some preliminary work on discovering initially unknown predicates and object types that can be used to build more accurate and parsimonious models. In both of these areas, FOPL research is moving "into the unknown".

References

[1] G. Elidan and N. Friedman. Learning hidden variable networks: The information bottleneck approach. *JMLR*, 6:81–127, 2005.

[2] D. Fierens, H. Blockeel, M. Bruynooghe, and J. Ramon. Logical Bayesian networks and their relation to other probabilistic logical models. In *Proc. 15th Int'l Conf. on Inductive Logic Programming*, 2005.

[3] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proc. 16th IJCAI*, pages 1300–1307, 1999.

[4] L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of relational structure. In *Proc. 18th ICML*, pages 170–177, 2001.

[5] J. Y. Halpern. An analysis of first-order logics of probability. *Artificial Intelligence*, 46:311–350, 1990.

[6] D. Heckerman, C. Meek, and D. Koller. Probabilistic models for relational data. Technical Report MSR-TR-2004-30, Microsoft Research, 2004.

[7] M. Jaeger. Relational Bayesian networks. In *Proc. 13th UAI*, pages 266–273, 1997.

[8] K. Kersting and L. De Raedt. Adaptive Bayesian logic programs. In *Proc. 11th Int'l Conf. on Inductive Logic Programming*, 2001.

[9] S. Kok and P. Domingos. Learning the structure of Markov logic networks. In *Proc. 22nd ICML*, pages 441–448, 2005.

[10] D. Koller and A. Pfeffer. Probabilistic frame-based systems. In *Proc. 15th AAAI*, pages 580–587, 1998.

[11] K. B. Laskey and P. C. G. da Costa. Of starships and Klingons: First-order Bayesian logic for the 23rd century. In *Proc. 21st UAI*, pages 346–353, 2005.

[12] T. Lukasiewicz and G. Kern-Isberner. Probabilistic logic programming under maximum entropy. In *Proc. 5th ECSQARU*, pages 279–292, 1999.

[13] B. Milch, B. Marthi, S. Russell, D. Sontag, D. L. Ong, and A. Kolobov. BLOG: Probabilistic models with unknown objects. In *Proc. 19th IJCAI*, 2005.

[14] B. Milch, B. Marthi, D. Sontag, S. Russell, D. L. Ong, and A. Kolobov. Approximate inference for infinite contingent Bayesian networks. In *Proc. 10th AISTATS*, 2005.

[15] B. Milch and S. Russell. General-purpose MCMC inference over relational structures. In *Proc. 22nd UAI*, pages 349–358, 2006.

[16] S. Muggleton and W. Buntine. Machine invention of first-order predicates by inverting resolution. In *Proc. 5th ICML*, pages 339–352, 1988.

[17] S. H. Muggleton. Stochastic logic programs. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, pages 254–264. IOS Press, 1996.

[18] S.H. Muggleton. Learning structure and parameters of stochastic logic programs. *Electronic Trans. on AI*, 6, 2002.

[19] J. Neville and D. Jensen. Dependency networks for relational data. In *Proc. 4th IEEE Int'l Conf. on Data Mining*, 2004.

[20] J. Neville, D. Jensen, L. Friedland, and M. Hay. Learning relational probability trees. In *Proc. 9th KDD*, 2003.

[21] R. T. Ng and V. S. Subrahmanian. Probabilistic logic programming. *Information and Computation*, 101(2):150–201, 1992.

[22] L. Ngo and P. Haddawy. Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Comp. Sci.*, 171(1–2):147–177, 1997.

[23] R. Otero and S. Muggleton. On McCarthy's appearance and reality problem. In *ILP '06: Short Papers*, 2006.

[24] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. In *NIPS 15*. MIT Press, Cambridge, MA, 2003.

[25] C. Perlich and F. Provost. Aggregation-based feature invention and relational concept classes. In *Proc. 9th KDD*, 2003.

[26] A. Pfeffer. IBAL: A probabilistic rational programming language. In *Proc. 17th IJCAI*, 2001.

[27] D. Poole. Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence*, 64(1):81–129, 1993.

[28] D. Poole. The Independent Choice Logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, 94(1–2):5–56, 1997.

[29] A. Popescul and L. H. Ungar. Cluster-based concept invention for statistical relational learning. In *Proc. 10th KDD*, 2004.

[30] A. Popescul, L. H. Ungar, S. Lawrence, and D. M. Pennock. Statistical relational learning for document mining. In *Proc. 3rd IEEE Int'l Conf. on Data Mining*, pages 275–282, 2003.

[31] A. Puech and S. Muggleton. A comparison of stochastic logic programs and Bayesian logic programs. In *IJCAI Workshop on Learning Statistical Models from Relational Data*, 2003.

[32] K. Revoredo, A. Paes, G. Zaverucha, and V. Santos Costa. Combining predicate invention and revision of probabilistic FOL theories. In *ILP '06: Short Papers*, 2006.

[33] M. Richardson and P. Domingos. Markov logic networks. *MLJ*, 62:107–136, 2006.

[34] T. Sato and Y. Kameya. PRISM: A symbolic–statistical modeling language. In *Proc. 15th IJCAI*, pages 1330–1335, 1997.

[35] T. Sato and Y. Kameya. Parameter learning of logic programs for symbolic–statistical modeling. *JAIR*, 15:391–454, 2001.

[36] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proc. 18th UAI*, pages 485–492, 2002.

[37] A. Thomas, D. Spiegelhalter, and W. Gilks. BUGS: A program to perform Bayesian inference using Gibbs sampling. In J. Bernardo, J. Berger, A. Dawid, and A. Smith, editors, *Bayesian Statistics 4*. Oxford Univ. Press, 1992.

[38] A. Van Assche, C. Vens, H. Blockeel, and S. Džeroski. First order random forests: Learning relational classifiers with complex aggregates. *MLJ*, 64:149–182, 2006.

[39] J. Vennekens, S. Verbaeten, and M. Bruynooghe. Logic programs with annotated disjunctions. In *Proc. Int'l Conf. on Logic Programming*, pages 431–445, 2004.