BLOG: Relational Modeling with Unknown Objects

Brian Milch MILCH@CS.BERKELEY.EDU
Bhaskara Marthi MARTHI@CS.BERKELEY.EDU
Stuart Russell RUSSELL@CS.BERKELEY.EDU
Computer Science Division, University of California, Berkeley, CA 94720-1776 USA

Abstract

In many real-world probabilistic reasoning problems, one of the questions we want to answer is: how many objects are out there? Examples of such problems range from multi-target tracking to extracting information from text documents. However, most probabilistic modeling formalisms — even first-order ones — assume a fixed, known set of objects. We introduce a language called BLOG for specifying probability distributions over relational structures that include varying sets of objects. In this paper we present BLOG informally, by means of example models for multi-target tracking and citation matching. We discuss some attractive features of BLOG models and some avenues of future work.

1. Introduction

In many probabilistic reasoning problems, from multi-target tracking to extracting information from text documents, our task is to infer facts about the real-world objects that generated our data. The set of real-world objects involved is seldom known in advance. Thus, we need a probabilistic modeling formalism that allows for uncertainty about what objects exist.

Existing formalisms that combine probability with logic programming (Kersting & De Raedt, 2001; Sato & Kameya, 2001) make both the *unique names assumption* — that each term in the logical language refers to a distinct object — and the *domain closure assumption* — that the only objects are those denoted by the terms of the language. Thus, these formalisms only allow a fixed, known set of objects. Probabilistic relational models (PRMs) have been extended in several ways to allow unknown objects. PRMs may include *number variables* which specify the number of objects that stand in a given relation to an existing object (Koller & Pfeffer, 1998). A PRM may also include *existence variables*, which specify, for instance, whether a role exists for a given actor in a given movie (Getoor et al., 2002). Finally, PRMs have been extended to allow uncertainty about the total number of objects of a given type (Pasula et al., 2003). But there has been no unified syntax for describing all these kinds of uncertainty.

In this paper, we present a language called BLOG (Bayesian Logic) which is designed for reasoning about unknown objects. A BLOG model defines a probability distribution over model structures of a first-order logical language, which may include varying sets of objects. In (Milch et al., 2004), we discuss BLOG formally and prove that if a BLOG model satisfies certain acyclicity conditions, it defines a unique probability distribution. In this paper, we take a more informal approach, introducing the language by example. We also discuss how to assert evidence about unknown objects, and highlight some attractive features of BLOG models for information extraction tasks.

2. The aircraft domain

In this section, we describe one domain that we will use as a running example. Consider the problem of tracking an unknown number of aircraft, over an area that contains an unknown number of air bases. At each time step, an aircraft is either on the ground at some base, or flying with some position and velocity. Aircraft that are on the ground have some probability of beginning a flight to some destination at each step.

Suppose we observe the world through radar. For each time step t we receive a set of blips, each of which has an (x, y, z) location. A blip either is generated by some aircraft, in which case the location depends noisily on the aircraft's position, or is a false detection (resulting from clouds, etc.) We do not observe the true set of aircraft or airbases that exist, nor the identity of the aircraft generating a particular blip.

Here are some questions that we might be interested

Table 1. Language $\mathcal{L}_{\text{air}}$ for the aircraft domain.

Functor symbol	Type signature	Return type
Location	(AirBase)	R2Vector
HomeBase	(Aircraft)	AirBase
CurBase	(Aircraft, NaturalNum)	AirBase
State	(Aircraft, NaturalNum)	R6Vector
Dest	(Aircraft, NaturalNum)	AirBase
TakesOff	(Aircraft, NaturalNum)	Boolean
Lands	(Aircraft, NaturalNum)	Boolean
BlipTime	(RadarBlip)	NaturalNum
BlipSource	(RadarBlip)	Aircraft
ApparentPos	(RadarBlip)	R3Vector

in, given a set of observations:

- Did the blip at position $(3.3, 6.2, 9)$ at time 5 come from an aircraft, and if so, what is its destination?
- How many airbases exist?
- Are the blip at $(2, 4, 1)$ at time 3 and the blip at $(6, -1, 4)$ at time 9 from the same source?

3. Possible worlds

Our modeling approach is to specify a probability distribution over a set of possible worlds. A possible world for the aircraft domain consists of:

- a set of air bases, each with a location in $\mathbb{R}^2$;
- a set of aircraft, each with a home base;
- a function that maps each (a, t) pair to the air base where aircraft a is located at time t, or a null value if a is in the air at time t;
- a function that maps each (a, t) pair to aircraft a's state vector (position and velocity) in $\mathbb{R}^6$ at time t;
- a function that maps each (a, t) pair to the base that is aircraft a's current destination at time t;
- for each time $t \in \{1, \dots, T\}$, a set of radar blips observed at that time, each with an apparent source position;
- a function that maps each radar blip to its source aircraft, or a null value if it is a false detection.

We use first-order logic to describe such sets of possible worlds in a formal way. A typed first-order language (Enderton, 2001) $\mathcal{L}$ consists of a set of type symbols and a set of functor symbols (i.e., function and predicate symbols). Each functor symbol has a *type signature* $(s_1, \dots, s_k)$, where each s_i is a type symbol. Such a functor is known as a k-ary functor symbol; if $k = 0$, it is also known as a constant symbol. Each functor symbol also has a *return type* s, which is Boolean for predicate symbols. We also allow functors to take on the special value null. Table 1 shows the functors for the aircraft domain.

As *model structure* ω of a typed first-order language includes an *extension* $[s]^\omega$ for each type, which is the set of objects of that type in ω. Also, for each functor symbol f with signature $(s_1, \dots, s_k)$ and return type s, it includes an *interpretation* $[f]^\omega$, which is a function from $[s_1]^\omega \times \cdots \times [s_k]^\omega$ to $[s]^\omega \cup \{\mathsf{null}\}$. We will use model structures of a language to represent possible worlds formally.[1] For example, model structures of $\mathcal{L}_{\text{air}}$ correspond to possible worlds in the aircraft domain. The syntax and semantics of terms and formulas of a typed first-order language are as in standard first-order logic, except that terms have types and the arguments to a functor must be of the appropriate types.

4. The probabilistic model

To describe a probability distribution over possible worlds, it is often useful to imagine a *generative process* that samples a possible world. For the aircraft domain, we generate a possible world ω as follows. First, create some number (chosen according to a prior) of AirBase objects, and sample a location for each one. For each base b, create some number of aircraft having b as home base. For each time step t (starting at 0), and each aircraft a, sample values for a's attributes at time t. Specifically, if a is in flight at time $t-1$, decide whether a lands at time t. If so, the base that used to be a's destination becomes its current base. Otherwise, sample a's state vector at time t conditioning on its state at time $t-1$ and the location of a's destination. If a is on the ground at time $t-1$, decide whether it takes off at time t, and if so, sample a new destination and initial state for a. Finally, for each a in flight at time t, choose whether to create a radar blip corresponding to a. Also, create some number of false detections at time t.

This generative process induces a probability distribution over possible worlds. We now present BLOG, a formal language for specifying such distributions. A BLOG model begins by specifying a typed first-order language $\mathcal{L}$. A type can either be user-defined or be chosen from a library of standard types such as NaturalNum. The functor symbols of $\mathcal{L}$ are divided into two sets : *random* and *nonrandom*. The model then specifies, for each type s, a set of *guaranteed objects* $G_{\mathcal{M}}(s)$ which exist in every possible world, and the value of each nonrandom functor f on each tuple of guaranteed objects. In the aircraft domain, for ex-

[1] Strictly speaking, we only allow structures over a particular universe of discourse. See (Milch et al., 2004) for details.

```
#{AirBase}:
    ~ NumBasesDistrib()
Location(b):
    ~ UniformLocation()
#{Aircraft : HomeBase ↦ b}:
    ~ NumAircraftDistrib()
TakesOff(a, t):
    if Greater(t, 0) ∧ ¬InFlight(a, Pred(t))
        then ~ TakeoffBernoulli()
Lands(a, t):
    if Greater(t, 0) ∧ InFlight(a, Pred(t))
        then ~ LandingDistrib(State(a, Pred(t)),
                              Location(Dest(a, Pred(t))))
CurBase(a, t):
    if t = 0 then = HomeBase(a)
    elseif TakesOff(a, t) then = null
    elseif Lands(a, t) then = Dest(a, Pred(t))
    else  = CurBase(a, Pred(t))
InFlight(a, t):
    = (CurBase(a, t) = null)
State(a, t):
    if TakesOff(a, t)
        then ~ InitState(Location(CurBase(a, Pred(t))))
    elseif InFlight(a, t)
        then ~ StateTransition(State(a, Pred(t)),
                               Location(Dest(a, t)))
Dest(a, t):
    if TakesOff(a, t)
        then ~ UniformChoice({AirBase b})
    elseif InFlight(a, t)
        then = Dest(a, Pred(t))
#{RadarBlip : BlipSource ↦ a, BlipTime ↦ t}:
    if InFlight(a, t) then
        ~ NumDetectionsDistrib()
#{RadarBlip : BlipSource ↦ null, BlipTime ↦ t}:
    ~ NumFalseAlarmsDistrib()
ApparentPos(r):
    if BlipSource(r) = null
        then ~ FalseDetectionDistrib()
    else  ~ ObsDistrib(State(BlipSource(r), BlipTime(r)))
```

Figure 1. BLOG model for the aircraft domain.

ample, there are guaranteed objects for the natural numbers, and the nonrandom functor $\mathsf{Pred}(x)$ denotes the standard predecessor function.

The main part of the BLOG model consists of statements specifying conditional probability distributions. The generative process described above includes two types of sampling operations. The first type involves sampling the value of some functor applied to some objects. The second type involves creating a set of objects having a certain property (e.g. aircraft with a given home base), where the number of newly created objects is sampled from some distribution. BLOG has a statement type for each of these operations.

4.1. Dependency statements

A BLOG model includes a *dependency statement* for each random functor, that describes how to sample the value of that functor applied to each tuple of objects. Consider, for example, the dependency statement for State in Figure 1. Suppose we are in the process of sampling a world ω and are about to sample a value for $\mathsf{State}(a,t)$ for some particular objects a and t. We first check whether the condition after the if statement — $\mathsf{TakesOff}(a,t)$ — holds in ω (thus, our sampling process needs to have already chosen the value of $\mathsf{TakesOff}(a,t)$ at this point). Suppose it is false. We then check the condition $\mathsf{InFlight}(a,t)$ in ω. Suppose this condition does hold. We then compute $\mathsf{State}(a,\mathsf{Pred}(t))$ and $\mathsf{Location}(\mathsf{Dest}(a,t))$ and pass them to the *conditional probability distribution* (CPD) StateTransition, which we assume is defined elsewhere by the user using a language such as Java. This function samples a value for $\mathsf{State}(a,t)$ from the return type of State.

In general, the lefthand side of a dependency statement specifies the functor symbol being sampled, and provides variable names that will be used to refer to the arguments of the functor. The righthand side consists of an if-then-else statement. The sampling process checks the clauses of this if-then-else statement until it finds one that is true. It then instantiates the arguments for the CPD and calls it. The CPD is either defined by the user or part of a library of standard distributions, such as Gaussian. If none of the clauses of the if statement are satisfied, then the value is null by default.

In the dependency statement for State, the arguments to the CPD were just the values of certain functors. However, BLOG also allows passing a *set* of values into a CPD. There are two situations where this is necessary. First, we might want to select one element of the set. For example, the first clause for the Dest functor in Figure 1 results in a uniform choice over the set of all air bases. The second situation where we need to pass in a set is when the values have to be aggregated in some way by the CPD. For example, when reasoning about movies, the success of a movie might depend on the sum of the Fame variables of each actor in the movie.

4.2. Number statements

The probability distributions governing the number of objects satisfying a particular property are specified using *number statements*. Consider the third statement in Figure 1. The lefthand side indicates that this statement determines the number of aircraft having a particular home base b. The righthand side has the same form as in dependency statements, except that it must sample from a distribution over natural numbers.

There can in general be multiple number statements for each type. The lefthand side of each number statement contains a set of conditions, which form a *potential object pattern*. In the example, this set consisted of the single condition $\mathsf{HomeBase}(a) = b$. Our semantics require that each nonguaranteed object in a possible world satisfy exactly one potential object pattern, and so when a value n is sampled for a given number statement, exactly n new objects are created.

5. Evidence and Queries

A BLOG model specifies a probability distribution over possible worlds of a language $\mathcal{L}$. Therefore, we can in principle introduce evidence simply by conditioning on a first-order sentence of $\mathcal{L}$. However, this approach runs into problems. Suppose, for example, that we have observed a blip at time 8 at position $(9.6, 1.2, 32.8)$, and condition on the sentence $\exists r\,(\mathsf{BlipTime}(r) = 8 \wedge \mathsf{ApparentPos}(r) = (9.6, 1.2, 32.8))$. We now want to query the destination air base of the aircraft that generated this blip. Unfortunately, we can't do this, because we don't have a way of referring to the blip outside the existential quantifier.

This problem is handled in logical reasoning systems using *Skolem constants*. Instead of asserting an existentially quantified sentence, one extends the language to include a new constant symbol, known as a Skolem constant. In the example, we might introduce the new constant symbol $\mathsf{R1}$ and condition on the sentence $\mathsf{BlipTime}(\mathsf{R1}) = 8 \wedge \mathsf{ApparentPos}(\mathsf{R1}) = (9.6, 1.2, 32.8)$.

However, our observation process is often such that we observe *all* objects generated by certain instances of number statements. For example, at time t we observe all the radar blips generated by the two number statements for $\mathsf{RadarBlip}$ in Figure 1 for that value of t. The fact that our observations are exhaustive can significantly affect our beliefs. For example, the probability that there are 10 aircraft in flight might be high given that there is a blip on the screen, but low given that there is *only* one blip on the screen.

We therefore use a syntax that allows us to state this exhaustiveness property. Suppose that, in addition to the previously mentioned blip, we observed only one other blip at time 8, at $(2, 1.6, 3)$. We first write $\{b : \mathsf{BlipTime}(b) = 8\} = \{\mathsf{B1}, \mathsf{B2}\}$. This asserts that the constant symbols $\mathsf{B1}$ and $\mathsf{B2}$ refer to all the blips b such that $\mathsf{BlipTime}(b) = 8$. We also make a local unique-names assumption, that $\mathsf{B1}$ and $\mathsf{B2}$ refer to different objects. The precise probabilistic semantics of such evidence statements is given in (Milch et al., 2004).

We then assert our observations as evidence, with the statements $\mathsf{ApparentPosition}(\mathsf{B1}) = (9.6, 1.2, 32.8)$ and $\mathsf{ApparentPosition}(\mathsf{B2}) = (2, 1.6, 3)$. We may now use the symbols $\mathsf{B1}$ and $\mathsf{B2}$ in our queries. For example, we might ask about the posterior distribution of $\mathsf{Dest}(\mathsf{BlipSource}(\mathsf{B1}))$.

6. Using BLOG Models for Information Extraction

In (Marthi et al., 2003), we argued that information extraction (IE) – the task of inferring facts from text documents – is a promising application for first-order probabilistic models that allow unknown objects. In this section, we give a BLOG model for the citation matching domain discussed in (Pasula et al., 2003; Marthi et al., 2003), and highlight some attractive features of BLOG models for IE.

6.1. BLOG Model for Citation Matching

In the citation matching task, we are given some citations taken from the "works cited" lists of publications in a certain field. We wish to recover the true sets of researchers and publications in this field. For each publication, we want to infer the true title and author list; for each researcher, we want to infer a full name.

We use the following generative model for this domain. First, some number of researchers are generated, and a name is chosen for each one. Then a number of publications are generated, depending on the number of researchers (we do not assume the publications are generated by individual researchers). For each publication, we choose a number of authors, then choose the specific authors by sampling uniformly without replacement from the set of researchers. The title of the publication is chosen independently.

We choose not to model how the given list of citations is generated; instead, we just treat the citations as guaranteed objects. For each citation, the publication cited is chosen uniformly at random from the set of publications. The author names and title that appear in the citation are sampled according to some string corruption models (we assume the citation does not drop, add, or reorder authors). Finally, we construct the whole citation string, given the corrupted names and title.

A BLOG model for this domain is shown in Fig. 2. This model illustrates how BLOG allows us to define sets of objects that are passed as arguments to

```
#{Researcher r}:
    ~ NumResearchersDistrib()
Name(r):
    ~ NamePrior()
#{Publication p}:
    ~ NumPubsDistrib({Researcher r})
NumAuthors(p):
    ~ NumAuthorsDistrib()
Author(p, i):
    if Less(i, NumAuthors(p))
        then ~ SampleUnused({Reseacher r},
                            {Author(p, j) : Less(j, i)})
Title(p):
    ~ TitlePrior()
PubCited(c):
    ~ UniformChoice({Publication p})
NameAsCited(c, i):
    if Less(i, NumAuthors(PubCited(c)))
        then ~ NameObs(Name(Author(PubCited(c), i)))
TitleAsCited(c):
    ~ TitleObs(Title(PubCited(c)))
CitString(c):
    ~ CitDistrib({i, NameAsCited(c, i) :
                    Less(i, NumAuthors(PubCited(c)))},
                 TitleAsCited(c))
```

Figure 2. Blog model for citation matching.

a CPD, such as $\{\mathsf{Author}(p, j) : \mathsf{Less}(j, i)\}$. We can even pass sets of pairs, such as $\{i, \mathsf{NameAsCited}(c, i) : \mathsf{Less}(i, \mathsf{NumAuthors}(\mathsf{PubCited}(c)))\}$. An interesting issue is how to represent arbitrary-length sequences, such as a publication's author list. The model in Fig. 2 illustrates one representation, where we have an infinite sequence of variables $\mathsf{Author}(p, i)$ for each publication p, but $\mathsf{Author}(p, i) = \mathsf{null}$ for $i \geq \mathsf{NumAuthors}(p)$.

6.2. Attractive features of BLOG models

We will now discuss several attractive properties that follow naturally from the Blog modeling approach, but are lacking in many other approaches to IE.

6.2.1. Reasoning about objects not mentioned in the text

In (Marthi et al., 2003), we discussed how a citation matching system should handle a query such as, "Did Mike Jordan have a paper in UAI '97?" if it has not seen any citations to such a paper. We considered a system that may have access to documents, such as Mike Jordan's home page, which it can identify as *exhaustive* lists of the papers with a certain property. If the system has seen Mike Jordan's home page and noted that it does not contain a UAI '97 paper, then the probability that such a paper exists is very low. On the other hand, if it has not seen any exhaustive lists, it should return a higher probability.

We are not aware of any existing IE systems that support reasoning about unmentioned objects. However, the ability to do such reasoning arises almost unavoidably in Blog models. Even in the simple model of Fig. 2, which does not include any notion of an exhaustive list, our possible worlds can include uncited publications. Since we assume that the target of each citation is chosen uniformly at random, we can make inferences about the number of uncited publications: for instance, if every cited publication has been cited at least 10 times, then the probability that there is also an uncited publication out there is low. Of course, reasoning about unmentioned object may entail extra complexity in inference. The point is that in a Blog model, a query about unmentioned objects has well-defined semantics.

6.2.2. Representing attributes once per object

A Blog model like the one in Fig. 2 distinguishes an object's true attributes (e.g., $\mathsf{Title}(p)$) from the attributes associated with individual mentions (e.g., $\mathsf{TitleAsCited}(c)$). This means that we can reconstruct an object's true attributes using clues from many separate mentions. There is also a more subtle advantage: when we compute the probability of a possible world, probabilities for object attributes are multiplied in only once per object, not once per mention.

This point is worth noting because defining object attributes on a per-mention basis might seem like a good way to avoid reasoning about unknown objects. Indeed, that is the approach taken in (McCallum & Wellner, 2003), which defines three unified probabilistic models for IE. Model 1 uses the same variables that we would use in a Blog model: a set of attribute variables for each object, plus a variable for each mention that specifies the object it refers to. Model 2, on the other hand, avoids any explicit representation of objects, and associates a set of object attribute variables with each mention. The object attribute variables for co-referring mentions are constrained to be equal.

To see how this simplifying step can be harmful, consider the task of inferring attributes for people mentioned in documents. For example, suppose we have a document where the first name "Dana" occurs many times, and the probability that all these occurrences of "Dana" refer to the same person is close to 1. Now suppose we want to infer Dana's sex. According to U.S. Census data, the probability that a person named "Dana" is female is around 0.75. So Model 2 would include a potential giving weight 0.75 to $\mathsf{Sex}(x) = \mathsf{female}$

and weight 0.25 to $\mathsf{Sex}(x) = \mathsf{male}$ when $\mathsf{FirstName}(x) =$ "Dana". But this potential would be multiplied into the joint probability once per mention. So with n mentions, the posterior probability that the occurrences of "Dana" refer to a female would be approximately $(0.75)^n/((0.75)^n + (0.25)^n)$, which approaches 1 as n increases. In a document with a large number of mentions, this duplication of potentials might outweigh contextual clues about Dana's sex (such as pronouns), leading to incorrect results.

6.2.3. Reasoning about coreference: Beyond pairwise compatibilities

In the Blog model of Fig. 2, we can ask for the posterior probability that three citations are coreferent — that is, they have the same PubCited value. This probability depends on the chance that a single publication, with some attributes, would yield the three observed citation strings. Most existing methods for coreference resolution do not attempt to approximate this probability. Instead, they use *pairwise* compatibility scores representing the probability that two mentions corefer. This simplification is made, for instance, in the transition from Model 2 to Model 3 in (McCallum & Wellner, 2003). Model 3 does not include any attribute variables; it just includes a Boolean coreference variable for each pair of mentions (with constraints to enforce transitivity of the coreference relation).

This simplification can lead to incorrect coreference resolution, even if we are not interested in the attribute values for their own sake. For example, suppose we are given a news article with mentions of "Stuart", "Jones", and "Alice", and some grammatical cues suggest these mentions are all coreferent. If we only look at pairwise compatibilities, we will not realize that this three-way coreference is very unlikely (even if we enforce transitivity). There could easily be a person named "Stuart Jones", "Alice Jones", or "Alice Stuart". But it is very unlikely that a person would be referred to sometimes as "Stuart", sometimes as "Jones", and sometimes as "Alice", all in the same news article. This problem arises because there is ambiguity about which attribute of a person (first or last name) is being specified by the mention "Stuart".

7. Future Work

The obvious question at this point is how to do inference (and parameter estimation, which requires inference if our data are only partially observed) in Blog models. There are really two questions here. The first is under what conditions the inference problem is even decidable, given that a Blog model may permit an unbounded number of objects. We hope to define a Markov chain Monte Carlo (MCMC) algorithm that is guaranteed to converge to the correct probability for a Blog query under certain broad conditions.

The other question is: when can we do approximate inference efficiently in practice? It is worth noting that approximate inference is NP-hard even for standard Bayesian networks (Dagum & Luby, 1993), but this has not prevented Bayesian networks from being a popular representational formalism. Researchers have developed a toolbox of approximate inference algorithms that are accurate and efficient on some practical problems. We plan to build upon this work to develop approximate inference algorithms for Blog models.

With this goal in mind, there are two major approaches to explore. The first is to place some upper bound on the number of potential objects that exist, so we can represent the distribution over possible worlds with a large but finite Bayesian network. For some queries, we may not even need to impose an upper bound, because only a finite number of objects may be relevant. We can then apply an approximate inference algorithm such as loopy belief propagation (Murphy et al., 1999).

The other approach is to use a stochastic sampling technique such as MCMC, where we allow different states of our Markov chain to include different numbers of objects. MCMC algorithms of this type have been implemented for Bayesian mixture modeling (Neal, 2000) and for the citation matching task described in Sec. 6.1 (Pasula et al., 2003). However, we would like a general algorithm that applies to any probability distribution defined as a Blog model. In some of the MCMC algorithms mentioned above, the states of the Markov chain are not fully specified possible worlds, but rather partial descriptions that leave out (for example) the uncited publications. We believe that such query-specific simplifications of the MCMC state space can be applied to Blog models in general, and may lay the foundation for practical approximate inference.

We are also interested in extending Blog to represent undirected and conditional models, such as those used in (McCallum & Wellner, 2003). The Blog models we have described in this paper can be thought of as extending probabilistic relational models (Friedman et al., 1999) to handle unknown objects; we should also be able to extend their undirected analogues, relational Markov networks (Taskar et al., 2002). The main technical problem is ensuring that the normalization constant in an undirected model remains finite, even when we have an unbounded number of objects and hence an infinite set of possible outcomes.

Acknowledgments

This work was supported by ONR MURI N00014-00-1-0637 and the DARPA CALO program. The authors are also supported by an NSF Graduate Research Fellowship (Milch), and an NSERC PGS-B fellowship (Marthi). We thank Mark Paskin for useful discussions.

References

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

Enderton, H. B. (2001). *A mathematical introduction to logic*. Academic Press. 2nd edition.

Friedman, N., Getoor, L., Koller, D., & Pfeffer, A. (1999). Learning probabilistic relational models. *Proc. 16th IJCAI* (pp. 1300–1307).

Getoor, L., Friedman, N., Koller, D., & Taskar, B. (2002). Learning probabilistic models of link structure. *JMLR*, *3*, 679–707.

Kersting, K., & De Raedt, L. (2001). Adaptive Bayesian logic programs. *Proc. 11th Int'l Conf. on ILP*.

Koller, D., & Pfeffer, A. (1998). Probabilistic frame-based systems. *Proc. 15th AAAI* (pp. 580–587).

Marthi, B., Milch, B., & Russell, S. (2003). First-order probabilistic models for information extraction. *IJCAI Wksp on Learning Statistical Models from Relational Data*.

McCallum, A., & Wellner, B. (2003). Toward conditional models of identity uncertainty with application to proper noun coreference. *IJCAI Wksp on Information Integration on the Web*.

Milch, B., Marthi, B., & Russell, S. (2004). First-order probabilistic models with unknown objects. Unpublished manuscript.

Murphy, K. P., Weiss, Y., & Jordan, M. I. (1999). Loopy belief propagation for approximate inference: An empirical study. *Proc. 15th UAI* (pp. 467–475).

Neal, R. M. (2000). Markov chain sampling methods for Dirichlet process mixture models. *J. Computational and Graphical Statistics*, *9*, 249–265.

Pasula, H., Marthi, B., Milch, B., Russell, S., & Shpitser, I. (2003). Identity uncertainty and citation matching. In *NIPS 15*.

Sato, T., & Kameya, Y. (2001). Parameter learning of logic programs for symbolic-statistical modeling. *JAIR*, *15*, 391–454.

Taskar, B., Abbeel, P., & Koller, D. (2002). Discriminative probabilistic models for relational data. *Proc. 18th UAI* (pp. 485–492).