Discrete-Continuous Mixtures in Probabilistic Programming: Generalized Semantics and Inference Algorithms

Yi Wu [1] **Siddharth Srivastava** [2] **Nicholas Hay** [3] **Simon S. Du** [4] **Stuart Russell** [1]

Abstract

Despite the recent successes of probabilistic programming languages (PPLs) in AI applications, PPLs offer only limited support for random variables whose distributions combine discrete and continuous elements. We develop the notion of *measure-theoretic Bayesian networks (MTBNs)* and use it to provide more general semantics for PPLs with arbitrarily many random variables defined over arbitrary measure spaces. We develop two new general sampling algorithms that are provably correct under the MTBN framework: the lexicographic likelihood weighting (LLW) for general MTBNs and the lexicographic particle filter (LPF), a specialized algorithm for state-space models. We further integrate MTBNs into a widely used PPL system, BLOG, and verify the effectiveness of the new inference algorithms through representative examples.

1. Introduction

As originally defined by Pearl (1988), Bayesian networks express joint distributions over finite sets of random variables as products of conditional distributions. Probabilistic programming languages (PPLs) (Koller et al., 1997; Milch et al., 2005a; Goodman et al., 2008; Wood et al., 2014b) apply the same idea to potentially infinite sets of variables with general dependency structures. Thanks to their expressive power, PPLs have been used to solve many real-world applications, including Captcha (Le et al., 2017), seismic monitoring (Arora et al., 2013), 3D pose estimation (Kulkarni et al., 2015), generating design suggestions (Ritchie et al., 2015), concept learning (Lake et al., 2015), and cognitive science applications (Stuhlmüller & Goodman, 2014).

In practical applications, we often have to deal with a mixture of continuous and discrete random variables. Existing PPLs support both discrete and continuous random variables, but not discrete-continuous mixtures, i.e., variables whose distributions combine discrete and continuous elements. Such variables are fairly common in practical applications: sensors that have thresholded limits, e.g. thermometers, weighing scales, speedometers, pressure gauges; or a hybrid sensor that can report a either real value or an error condition. The occurrence of such variables has been noted in many other applications from a wide range of scientific domains (Kharchenko et al., 2014; Pierson & Yau, 2015; Gao et al., 2017).

Many PPLs have a restricted syntax that forces the expressed random variables to be either discrete or continuous, including WebPPL (Goodman & Stuhlmüller, 2014), Edward (Tran et al., 2016), Figaro (Pfeffer, 2009) and Stan (Carpenter et al., 2016). Even for PPLs whose *syntax* allows for mixtures of discrete and continuous variables, such as BLOG (Milch et al., 2005a), Church (Goodman, 2013), Venture (Mansinghka et al., 2014) and Anglican (Wood et al., 2014a), the underlying *semantics* of these PPLs implicitly assumes the random variables are not mixtures. Moreover, the inference algorithms associated with the semantics inherit the same assumption and can produce incorrect results when discrete-continuous mixtures are used.

Consider the following GPA example: a two-variable Bayes net $\mathit{Nationality} \rightarrow \mathit{GPA}$ where the nationality follows a binary distribution

$$P(\mathit{Nationality} = \mathit{USA}) = P(\mathit{Nationality} = \mathit{India}) = 0.5$$

and the conditional probabilities are discrete-continuous mixtures

$$\begin{aligned} &\mathit{GPA}|\mathit{Nationality} = \mathit{USA} \\ \sim &0.01 \cdot \mathbf{1}\{\mathit{GPA} = 4\} + 0.99 \cdot \mathrm{Unif}(0, 4), \\ &\mathit{GPA}|\mathit{Nationality} = \mathit{India} \\ \sim &0.01 \cdot \mathbf{1}\{\mathit{GPA} = 10\} + 0.99 \cdot \mathrm{Unif}(0, 10). \end{aligned}$$

This is a typical scenario in practice because many top students have perfect GPAs. Now suppose we observe a student with a GPA of 4.0. Where do they come from? If the student is Indian, the probability of any singleton set $\{g\}$

[1]University of California, Berkeley [2]Arizona State University [3]Vicarious Inc. [4]Carnegie Mellon University. Correspondence to: Yi Wu <jxwuyi@gmail.com>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

where $0 < g < 10$ is zero, as this range has a probability *density*. On the other hand if the student is American, the set $\{4\}$ has the probability 0.01. Thus, by Bayes theorem, $P(\textit{Nationality} = \textit{USA} | \textit{GPA} = 4) = 1$, which means the student *must* be from the USA.

However, if we run the default Bayesian inference algorithm for this problem in PPLs, e.g., the standard importance sampling algorithm (Milch et al., 2005b), a sample that picks India receives a density weight of $0.99/10.0 = 0.099$, whereas one that picks USA receives a discrete-mass weight of 0.01. Since the algorithm does not distinguish probability density and mass, it will conclude that the student is very probably from India, which is far from the truth.

We can fix the GPA example by considering a density weight infinitely smaller than a discrete-mass weight (Nitti et al., 2016; Tolpin et al., 2016). However, the situation becomes more complicated when involving more than one evidence variable, e.g., GPAs over multiple semesters for students who may study in both countries. Vector-valued variables also cause problems—does a point mass in three dimensions count more or less than a point mass in two dimensions? These practical issues motivate the following two tasks:

- Inherit all the existing properties of PPL semantics and extend it to handle random variables with mixed discrete and continuous distributions;
- Design provably correct inference algorithms for the extended semantics.

In this paper, we carry out all these two tasks and implement the extended semantics as well as the new algorithms in a widely used PPL, Bayesian Logic (BLOG) (Milch et al., 2005a).

1.1. Main Contributions

Measure-Theoretical Bayesian Nets (MTBNs) Measure theory can be applied to handle discrete-continuous mixtures or even more abstract measures. In this paper, we define a generalization of Bayesian networks called *measure-theoretic Bayesian networks (MTBNs)* and prove that every MTBN represents a unique measure on the input space. We then show how MTBNs can provide a more general semantic foundation for PPLs.

More concretely, MTBNs support (1) random variables with infinitely (even uncountably) many parents, (2) random variables valued in *arbitrary measure spaces* (with $\mathbb{R}^N$ as one case) distributed according to *any measure* (including discrete, continuous and mixed), (3) establishment of conditional independencies implied by an infinite graph, and (4) open-universe semantics in terms of the possible worlds in the vocabulary of the model.

Inference Algorithms We propose a provably correct inference algorithm, lexicographic likelihood weighting (LLW), for general MTBNs with discrete-continuous mixtures. In addition, we propose LPF, a particle-filtering variant of LLW for sequential Monte Carlo (SMC) inference on state-space models.

Incorporating MTBNs into an existing PPL We incorporate MTBNs into BLOG with simple modifications and then define the generalized BLOG language, *measure-theoretic BLOG*, which formally supports arbitrary distributions, including discrete-continuous mixtures. We prove that every generalized BLOG model corresponds to a unique MTBN. Thus, all the desired theoretical properties of MTBNs can be carried to measure-theoretic BLOG. We also implement the LLW and LPF algorithms in the backend of measure-theoretic BLOG and use three representative examples to show their effectiveness.

1.2. Organization

This paper is organized as follows. We first discuss related work in Section 2. In Section 3, we formally define *measure-theoretic Bayesian nets* and study their theoretical properties. Section 4 describes the LLW and LPF inference algorithms for MTBNs with discrete-continuous mixtures and establishes their correctness. In Section 5, we introduce the measure-theoretic extension of BLOG and study its theoretical foundations for defining probabilistic models. In Section 6, we empirically validate the generalized BLOG system and the new inference algorithms on three representative examples.

2. Related Work

The motivating GPA example has been also discussed as a special case under some other PPL systems (Tolpin et al., 2016; Nitti et al., 2016). Tolpin et al. (2016) and Nitti et al. (2016) proposed different solutions specific to this example but did not address the general problems of representation and inference with random variables with mixtures of discrete and continuous distributions. In contrast, we present a general formulation with provably correct inference algorithms.

Our approach builds upon the foundations of the BLOG probabilistic programming language (Milch, 2006). We use a measure theoretic formulation to generalize the syntax and semantics of BLOG to random variables that may have infinitely many parents and mixed continuous and discrete distributions. The BLP framework Kersting & De Raedt (2007) unifies logic programming with probability models, but requires each random variable to be influenced by a finite set of random variables in order to define the semantics. This amounts to requiring only finitely many ances-

tors of each random variable. Choi et al. (2010) present an algorithm for carrying out lifted inference over models with purely continuous random variables. They also require parfactors to be functions over finitely many random variables, thus limiting the set of influencing variables for each node to be finite. Gutmann et al. (2011a) also define densities over finite dimensional vectors. In a relatively more general formulation (Gutmann et al., 2011b) define the distribution of each random variable using a definite clause, which corresponds to the limitation that each random variable (either discrete or continuous) has finitely many parents. Frameworks building on Markov networks also have similar restrictions. Wang & Domingos (2008) only consider networks of finitely many random variables, which can have either discrete or continuous distributions. Singla & Domingos (2007) extend Markov logic to infinite (non-hybrid) domains, provided that each random variable has only finitely many influencing random variables.

In contrast, our approach not only allows models with arbitrarily many random variables with mixed discrete and continuous distributions, but each random variable can also have arbitrarily many parents as long as all ancestor chains are finite (but unbounded). The presented work constitutes a rigorous framework for expressing probability models with the broadest range of cardinalities (uncountably infinite parent sets) and nature of random variables (discrete, mixed, and even arbitrary measure spaces), with clear semantics in terms of first-order possible worlds and the generalization of conditional independences on such models.

Lastly, there are also other works using measure-theoretic approaches to analyze the semantics properties of probabilistic programs but with different emphases, such as the commutativity (Staton, 2017), design choices for monad structures (Ramsey, 2016) and computing a disintegration (Shan & Ramsey, 2017).

3. Measure-Theoretic Bayesian Networks

In this section, we introduce *measure-theoretic Bayesian networks (MTBNs)* and prove that an MTBN represents a unique measure with desired theoretical properties. We assume familiarity with measure-theoretic approaches to probability theory. Some background is included in Appx. A.

We begin with some necessary definitions of graph theory.

Definition 3.1. *A **digraph** G is a pair $G = (V, E)$ of a set of vertices V, of any cardinality, and a set of directed edges $E \subseteq V \times V$. The notation $u \to v$ denotes $(u, v) \in E$, and $u \mapsto v$ denotes the existence of a path from u to v in G.*

Definition 3.2. *A vertex $v \in V$ is a **root vertex** if there are no incoming edges to it, i.e., there is no $u \in V$ such that $u \to v$. Let $\mathrm{pa}(v) = \{u \in V : u \to v\}$ denote the set of parents of a vertex $v \in V$, and $\mathrm{nd}(v) = \{u \in V : \text{not } v \mapsto u\}$ denote its set of non-descendants.*

Definition 3.3. *A **well-founded digraph** (V, E) is one with no countably infinite ancestor chain $v_0 \leftarrow v_1 \leftarrow v_2 \leftarrow \ldots$.*

This is the natural generalization of a finite directed acyclic graph to the infinite case. Now we are ready to give the key definition of this paper.

Definition 3.4. *A **measure-theoretic Bayesian network** $M = (V, E, \{\mathcal{X}_v\}_{v\in V}, \{K_v\}_{v\in V})$ consists of (a) a well-founded digraph (V, E) of any cardinality, (b) an arbitrary measurable space $\mathcal{X}_v$ for each $v \in V$, and (c) a probability kernel K_v from $\prod_{u\in \mathrm{pa}(v)} \mathcal{X}_u$ to $\mathcal{X}_v$ for each $v \in V$.*

By definition, MTBNs allow us to define very general and abstract models with the following two major benefits:

1. We can define random variables with infinitely (even uncountably) many parents because MTBN is defined on a well-founded digraph.

2. We can define random variables in arbitrary measure spaces (with $\mathbb{R}^N$ as one case) distributed according to any measure (including discrete, continuous and mixed).

Next, we related MTBN to a probability measure. Fix an MTBN $M = (V, E, \{\mathcal{X}_v\}_{v\in V}, \{K_v\}_{v\in V})$. For $U \subseteq V$ let $\mathcal{X}_U = \prod_{u\in U} \mathcal{X}_u$ be the product measurable space over variables $u \in U$. With this notation, K_v is a kernel from $\mathcal{X}_{\mathrm{pa}(v)}$ to $\mathcal{X}_v$. Whenever $W \subseteq U$ let $\pi^U_W \colon \mathcal{X}_U \to \mathcal{X}_W$ denote the projection map. Let $\mathcal{X}_V$ be our base measurable space upon which we will consider different probability measures μ. Let X_v for $v \in V$ denote both the underlying set of $\mathcal{X}_v$ and the random variable given by the projection $\pi^V_{\{v\}}$, and X_U for $U \subseteq V$ the underlying space of X_U and the random variable given by the projection π^V_U.

Definition 3.5. *An MTBN M **represents** a measure μ on $\mathcal{X}_V$, if for all $v \in V$:*

- *X_v is conditionally independent of its non-descendants $X_{\mathrm{nd}(v)}$ given its parents $X_{\mathrm{pa}(v)}$.*
- *$K_v(X_{\mathrm{pa}(v)}, A) = \mathbb{P}_\mu[X_v \in A | X_{\mathrm{pa}(v)}]$ holds almost surely for any $A \in \mathcal{X}_v$, i.e., K_v is a version of the conditional distribution of X_v given its parents.*

Def. 3.5 captures the generalization of the local properties of Bayes networks – conditional independence and conditional distributions defined by parent-child relationships. Here we assume the conditional probability exists and is unique. This is a mild condition because this holds as long as the probability space is regular (Kallenberg, 2002).

The next theorem shows that MTBNs are well-defined.

Theorem 3.6. *An MTBN M represents a unique measure μ on $\mathcal{X}_V$.*

The proof of theorem 3.6 requires several intermediate results and is presented in Appx. B. The proof proceeds by first defining a projective family of measures. This gives a way to recursively construct our measure μ. We then define a notion of consistency such that every consistent projective family constructs a measure that M represents. Lastly, we give an explicit characterization of the unique consistent projective family, and thus of the unique measure M represents.

4. Generalized Inference Algorithms

We introduce the lexicographic likelihood weighting (LLW) algorithm for provably correct inference on MTBNs. We also present lexicographic particle filter (LPF) for state-space models by adapting LLW for the sequential Monte Carlo (SMC) framework.

4.1. Lexicographic likelihood weighting

Suppose we have an MTBN with finitely many random variables $X_1, \dots, X_N$, and that, without loss of generality, we observe real-valued random variables $X_1, \dots, X_M$ for $M < N$ as evidence. Suppose the distribution of X_i given its parents $X_{\mathrm{pa}(i)}$ is a mixture between a density $f_i(x_i|x_{\mathrm{pa}(i)})$ with respect to the Lebesgue measure and a discrete distribution $F_i(x_i|x_{\mathrm{pa}(i)})$, i.e., for any $\epsilon > 0$, we have $P(X_i \in [x_i - \epsilon, x_i]|X_{\mathrm{pa}(i)}) = \sum_{x\in[x_i-\epsilon,x_i]} F_i(x_i|X_{\mathrm{pa}(i)}) + \int_{x_i-\epsilon}^{x_i} f_i(x|X_{\mathrm{pa}(i)})\,dx$. This implies that $F_i(x_i|x_{\mathrm{pa}(i)})$ is nonzero for at most countably many values x_i. If F_i is nonzero for finitely many points, it can be represented by a list of those points and their values.

Lexicographic Likelihood Weighting (LLW) extends the classical likelihood weighting (Milch et al., 2005b) to this setting. It visits each node of the graph in topological order, sampling those variables that are not observed, and accumulating a weight for those that are observed. In particular, at an evidence variable X_i we update a tuple (d, w) of the number of densities and a weight, initially $(0, 1)$, by:

$$(d, w) \leftarrow \begin{cases} (d, wF_i(x_i|x_{\mathrm{pa}(i)})) & F_i(x_i|x_{\mathrm{pa}(i)}) > 0, \\ (d+1, wf_i(x_i|x_{\mathrm{pa}(i)})) & \text{otherwise.} \end{cases} \tag{1}$$

Finally, having K samples $x^{(1)}, \dots, x^{(K)}$ by this process and accordingly a tuple $(d^{(i)}, w^{(i)})$ for each sample $x^{(i)}$, let $d^* = \min_{i:w^{(i)}\neq 0} d^{(i)}$ and estimate $E[f(X)|X_{1:M}]$ by

$$\frac{\sum_{\{i:d^{(i)}=d^*\}} w^{(i)}\, f(x^{(i)})}{\sum_{\{i:d^{(i)}=d^*\}} w^{(i)}}. \tag{2}$$

The algorithm is summarised in Alg. 1 The next theorem shows this procedure is consistent.

Theorem 4.1. *LLW is consistent:* (2) *converges almost surely to* $\mathbb{E}[f(X)|X_{1:M}]$.

Algorithm 1 Lexicographic Likelihood Weighting

Require: densities f, masses F, evidences E, and K.
  for $i = 1 \dots K$ **do**
    sample all the ancestors of E from prior
    compute $(d^{(i)}, w^{(i)})$ by Eq. (1)
  end for
  $d^\star \leftarrow \min_{i:w^{(i)}\neq 0} d^{(i)}$
  Return $\left(\sum_{i:d^{(i)}=d^\star} w^{(i)} f(x^{(i)})\right) / \left(\sum_{i:d^{(i)}=d^\star} w^{(i)}\right)$

In order to prove Theorem 4.1, the main technique we adopt is to use a more restricted algorithm, the Iterative Refinement Likelihood Weighting (IRLW) as a reference.

4.1.1. Iterative refinement likelihood weighting

Suppose we want to approximate the posterior distribution of an $\mathcal{X}$-valued random variable X conditional on a $\mathcal{Y}$-valued random variable Y, for arbitrary measure spaces $\mathcal{X}$ and $\mathcal{Y}$. In general, there is no notion of a probability density of Y given X for weighing samples. If, however, we could make a discrete approximation Y_t of Y then we could weight samples by the probability $P[Y_t = y_t|X]$. If we increase the accuracy of the approximation with the number of samples, this should converge in the limit. We show this is possible, if we are careful about how we approximate:

Definition 4.2. *An **approximation scheme** for a measurable space $\mathcal{Y}$ consists of a measurable space $\mathcal{A}$ and measurable approximation functions $\alpha_i \colon \mathcal{Y} \to \mathcal{A}$ for $i = 1, 2, \dots$ and $\alpha_i^j \colon \mathcal{A} \to \mathcal{A}$ for $i < j$ such that $\alpha_j \circ \alpha_i^j = \alpha_i$ and y can be measurably recovered from the subsequence $\alpha_t(y), \alpha_{t+1}(y), \dots$ for any $t > 0$.*

When Y is a real-valued variable we will use the approximation scheme $\alpha_n(y) = 2^{-n}\lceil 2^n y\rceil$ where $\lceil r\rceil$ denotes the ceiling of r, i.e., the smallest integer no smaller than it. Observe in this case that $P(\alpha_n(Y) = \alpha_n(y)) = P(\alpha_n(y) - 2^{-n} < Y \leq \alpha_n(y))$ which we can compute from the CDF of Y.

Lemma 4.3. *If X, Y are real-valued random variables with $\mathbb{E}\,|X| < \infty$, then $\lim_{i\to\infty} \mathbb{E}[X|\alpha_i(Y)] = \mathbb{E}[X|Y]$.*

Proof. Let $\mathcal{F}_i = \sigma(\alpha_i(Y))$ be the sigma algebra generated by $\alpha_i(Y)$. Whenever $i \leq j$ we have $\alpha_i(Y) = (\alpha_j \circ \alpha_i^j)(Y)$ and so $\mathcal{F}_i \subseteq \mathcal{F}_j$. This means $\mathbb{E}[X|\alpha_i(Y)] = \mathbb{E}[X|\mathcal{F}_i]$ is a martingale, so we can use martingale convergence results. In particular, since $\mathbb{E}\,|X| < \infty$

$$\mathbb{E}[X|\mathcal{F}_i] \to \mathbb{E}[X|\mathcal{F}_\infty] \quad \text{a.s. and in } L^1,$$

where $\mathcal{F}_\infty = \bigcup_i \mathcal{F}_i$ is the sigma-algebra generated by $\{\alpha_i(Y) : i \in \mathbb{N}\}$ (see Theorem 7.23 in (Kallenberg, 2002)).

Y is a measurable function of the sequence $(\alpha_1(Y), \dots)$, as $\lim_{i\to\infty} \alpha_i(Y) = Y$, and so $\sigma(Y) \subseteq \mathcal{F}_\infty$. By definition

the sequence is a measurable function of Y, and so $\mathcal{F}_\infty \subseteq \sigma(Y)$, and so $\mathbb{E}[X|\mathcal{F}_\infty] = \mathbb{E}[X|Y]$ giving our result. □

Iterative refinement likelihood weighting (IRLW) samples $x^{(1)}, \ldots, x^{(K)}$ from the prior and evaluates:

$$\frac{\sum_{i=1}^{K} P(\alpha_n(Y)|X = x^{(i)}) f(x^{(i)})}{\sum_{i=1}^{K} P(\alpha_n(Y)|X = x^{(i)})} \tag{3}$$

Using Lemma 4.3, G.12, and G.13, we can show IRLW is consistent.

Theorem 4.4. *IRLW is consistent:* (3) *converges almost surely to* $\mathbb{E}[f(X)|Y]$.

4.1.2. Proof of Theorem 4.1

Now we are ready to prove Theorem 4.1.

Proof of Theorem 4.1. We prove the theorem for evidence variables that are leaves It is straightforward to extend the proof when the evidence variables are non-leaf nodes. Let x be a sample produced by the algorithm with number of densities and weight (d, w). With $I_n = \prod_{i=1\ldots M}(\alpha_n(x_i) - 2^{-n}, \alpha_n(x_i)]$ a 2^{-n}-cube around $x_{1:M}$ we have

$$\lim_{n\to\infty} \frac{P(X_{1:M} \in I_n | X_{M+1:N} = x_{M+1:N})}{w\, 2^{-dn}} = 1.$$

Using I_n as an approximation scheme by Def. 4.2, the numerator in the above limit is the weight used by IRLW. But given the above limit, using $w\, 2^{-dn}$ as the weight will give the same result in the limit. Then if we have K samples, in the limit of $n \to \infty$ only those samples $x^{(i)}$ with minimal $d^{(i)}$ will contribute to the estimation, and up to normalization they will contribute weight $w^{(i)}$ to the estimation. □

4.2. Lexicographic particle filter

We now consider inference in a special class of high-dimensional models known as state-space models, and show how LLW can be adapted to avoid the curse of dimensionality when used with such models. A state-space model (SSM) consists of latent states $\{X_t\}_{0\le t\le T}$ and the observations $\{Y_t\}_{0\le t\le T}$ with a special dependency structure where $\mathrm{pa}(Y_t) = X_t$ and $\mathrm{pa}(X_t) = X_{t-1}$ for $0 < t \le T$.

SMC methods (Doucet et al., 2001), also knowns as particle filters, are a widely used class of methods for inference on SSMs. Given the observed variables $\{Y_t\}_{0\le t\le T}$, the posterior distribution $P(X_t|Y_{0:t})$ is approximated by a set of K particles where each particle $x_t^{(k)}$ represents a sample of $\{X_i\}_{0\le i\le t}$. Particles are propagated forward through the transition model $P(X_t|X_{t-1})$ and resampled at each time step t according to the weight of each particle, which is defined by the likelihood of observation Y_t.

Algorithm 2 Lexicographic Particle Filter (LPF)

Require: densities f, masses F, evidences Y, and K
  for $t = 0, \ldots, T$ **do**
    for $k = 0, \ldots, K$ **do**
      $x_t^{(k)} \leftarrow$ sample from transition
      compute $(d^{(k)}, w^{(k)})$ by Eq. 4
    end for
    $d^\star \leftarrow \min_{k:w^{(k)}\neq 0} d^{(k)}$
    $\forall k : d^{(k)} > d^\star, w^{(k)} \leftarrow 0$
    Output $\left(w^{(k)} f(x_t^{(k)})\right) / \left(\sum_k w^{(k)}\right)$
    resample particles according to $w^{(k)}$
  end for

In the MTBN setting, the distribution of Y_t[1] given its parent X_t can be a mixture of density $f_t(y_t|x_t)$ and a discrete distribution $F_t(y_t|x_t)$. Hence, the resampling step in a particle filter should be accordingly modified: following the idea from LLW, when computing the weight of a particle, we enumerate all the observations $y_{t,i}$ at time step t and again update a tuple (d, w), initially (0,1), by

$$(d, w) \leftarrow \begin{cases} (d, w F_t(y_{t,i}|x_t)) & F_t(y_{t,i}|x_t) > 0, \\ (d+1, w f_t(y_{t,i}|x_t)) & \text{otherwise.} \end{cases} \tag{4}$$

We discard all those particles with a non-minimum d value and then perform the normal resampling step. We call this algorithm lexicographical particle filter (LPF), which is summarized in Alg. 2.

The following theorem guarantees the correctness of LPF. Its Proof easily follows the analysis for LLW and the classical proof of particle filtering based on importance sampling.

Theorem 4.5. *LPF is consistent: the outputs of Alg. 2 converges almost surely to* $\{E[f(X_t)|Y_{0:t}]\}_{0\le t\le T}$.

5. Generalized Probabilistic Programming Languages

In Section 3 and Section 4 we provided the theoretical foundation of MTBN and general inference algorithms. This section describes how to incorporate MTBN into a practical PPL. We focus on a widely used open-universe PPL, BLOG (Milch, 2006). We define the generalized BLOG language, the *measure-theoretic BLOG*, and prove that every well-formed measure-theoretic BLOG model corresponds to a unique MTBN. Note that our approach also applies to other PPLs[2].

[1] There can be multiple variables observed. Here the notation Y_t denotes $\{Y_{t,i}\}_i$ for conciseness.

[2] It has been shown that BLOG has equivalent semantics to other PPLs (Wu et al., 2014; McAllester et al., 2008).

```
1 Type Applicant, Country;
2 distinct Country NewZealand, India, USA;
3 #Applicant(Nationality = c) ~
4  if (c==USA) then Poisson(50)
5  else Poisson(5);
6 origin Country Nationality(Applicant);
7 random Real GPA(Applicant s) ~
8  if Nationality(s) == USA then
9      Mix({ TruncatedGauss(3, 1, 0, 4) -> 0.9998,
10           4 -> 0.0001, 0 -> 0.0001})
11 else Mix({ TruncatedGauss(5, 4, 0, 10) -> 0.989,
12           10 -> 0.009, 0 -> 0.002});
13 random Applicant David ~
14     UniformChoice({a for Applicant a});
15 obs GPA(David) = 4;
16 query Nationality(David) = USA;
```

Figure 1. A BLOG code for the GPA example.

We begin with a brief description of the core syntax of BLOG, with particular emphasis on (1) number statements, which are critical for expressing open-universe models[3], and (2) new syntax for expressing MTBNs, i.e., the `Mix` distribution. Further description of BLOG's syntax can be found in Li & Russell (2013).

5.1. Syntax of measure-theoretic BLOG

Fig. 1 shows a BLOG model with measure-theoretic extensions for a multi-student GPA example. Line 1 declares two *types*, *Applicant* and *Country*. Line 2 defines 3 distinct countries with keyword `distinct`, New Zealand, India and USA. Lines 3 to 5 define a *number statement*, which states that the number of US applicants follows a Poisson distribution with a higher mean than those from New Zealand or India. Line 6 defines an *origin function*, which maps the object being generated to the arguments that were used in the number statement that was responsible for generating it. Here *Nationality* maps applicants to their nationalities. Lines 7 and 13 define two random variables by keyword `random`. Lines 7 to 12 state that the GPA of an applicant is distributed as a mixture of weighted discrete and continuous distributions. For US applicants, the range of values $0 < \mathit{GPA} < 4$ follows a truncated Gaussian with bounds 0 and 4 (line 9). The probability mass outside the range is attributed to the corresponding bounds: $P(\mathit{GPA} = 0) = P(\mathit{GPA} = 4) = 10^{-4}$ (line 10). GPA distributions for other countries are specified similarly. Line 13 defines a random applicant David. Line 15 states that the David's GPA is observed to be 4 and we query in line 16 whether David is from USA.

Number Statement (line 3 to 5) Fig. 2 shows the syntax of a number statement for Type_i. In this specification, g_j are origin functions (discussed below); $\bar{y}_j$ are tuples of arguments drawn from $\bar{x} = x_1, \ldots, x_k$; φ_j are first-order formulas with free variables $\bar{y}_j$; $\bar{e}_j$ are tuples of expressions over a subset of $x_1, \ldots, x_k$; and $c_j(\bar{e}_j)$ specify kernels $\kappa_j : \Pi_{\{\mathcal{X}_{\tau_e} : e \in \bar{e}_j\}} \mathcal{X}_e \to \mathbb{N}$ where τ_e is the type of the expression e.

$$
\begin{aligned}
&\#\text{Type}_i(g_1 = x_1, \ldots, g_k = x_k) \sim \\
&\quad \text{if } \varphi_1(\bar{y}_1) \text{ then } c_1(\bar{e}_1) \\
&\quad \text{else if } \varphi_2(\bar{y}_2) \text{ then } c_2(\bar{e}_2) \\
&\quad \ldots \\
&\quad \text{else } c_m(\bar{e}_m);
\end{aligned}
$$

Figure 2. Syntax of number statements

The arguments $\bar{x}$ provided in a number statement allow one to utilize information about the rest of the model (and possibly other generated objects) while describing the number of objects that should be generated for each type. These assignments can be recovered using the origin functions g_j, each of which is declared as:

$$\texttt{origin}\ \text{Type}_j\ \ g_j(\text{Type}_i),$$

where $Type_j$ is the type of the argument x_j in the number statement of $Type_i$ where g_j was used. The value of the j^{th} variable used in the number statement that generated u, an element of the universe, is given by $g_j(u)$. Line 6 in Fig. 1 is an example of origin function.

Mixture Distribution (line 9 to 12) In measure-theoretic BLOG, we introduce a new distribution, the mixture distribution (e.g., lines 9-10 in Fig. 1). A mixture distribution is specified as:

$$\texttt{Mix}(\{c_1(\bar{e}_1) \to w_1(\bar{e}'), \ldots, c_k(\bar{e}_k) \to w_k(\bar{e}')\}),$$

where c_i are arbitrary distributions, and w_i's are arbitrary real valued functions that sum to 1 for every possible assignment to their arguments: $\forall \bar{e}' \sum_i w_i(\bar{e}') = 1$. Note that in our implementation of measure-theoretical BLOG, we only allow a `Mix` distribution to express a mixture of densities and masses for simplifying the system design, although it still possible to express the same semantics without `Mix`.

5.2. Semantics of measure-theoretic BLOG

In this section we present the semantics of measure-theoretic BLOG and its theoretical properties. Every BLOG model implicitly defines a first-order vocabulary consisting of the set of functions and types mentioned in the model. BLOG's semantics are based on the standard, open-universe semantics of first-order logic. We first define the set of all possible elements that may be generated for a BLOG model.

Definition 5.1. *The set of* possible elements $\mathcal{U}_\mathcal{M}$ *for a BLOG model* $\mathcal{M}$ *with types* $\{\tau_1, \ldots, \tau_k\}$ *is* $\bigcup_{j \in \mathbb{N}} \{\mathcal{U}_j\}$, *where*

- $\mathcal{U}_0 = \langle U_1^0, \ldots, U_k^0 \rangle$, $U_j^0 = \{c_j : c_j$ *is a* `distinct` τ_i *constant in* $\mathcal{M}\}$
- $\mathcal{U}_{i+1} = \langle U_1^{i+1}, \ldots, U_k^{i+1} \rangle$, *where* $U_m^{i+1} = U_m^i \cup \{u_{\nu,\bar{u},m} : \nu(\bar{x})$ *is a number statement of type* τ_m, $\bar{u}$ *is a tuple of elements of the type of* $\bar{x}$ *from* $\mathcal{U}^i$, $m \in \mathbb{N}\}$

[3] The specialized syntax in BLOG to express models with infinite number of variables.

Def. 5.1 allows us to define the set of random variables corresponding to a BLOG model.

Definition 5.2. *The set of* basic random variables *for a BLOG model $\mathcal{M}$, $BRV(\mathcal{M})$, consists of:*

- *for each number statement $\nu(\bar{x})$, a number variable $V_\nu[\bar{u}]$ over the standard measurable space $\mathbb{N}$, where $\bar{u}$ is of the type of $\bar{x}$.*
- *for each function $f(\bar{x})$ and tuple $\bar{u}$ from $\mathcal{U}_\mathcal{M}$ of the type of $\bar{x}$, a* function application variable *$V_f[\bar{u}]$ with the measurable space $\mathcal{X}_{V_f[\bar{u}]} = \mathcal{X}_{\tau_f}$, where $\mathcal{X}_{\tau_f}$ is the measurable space corresponding to τ_f, the return type of f.*

We now define the space of consistent assignments to random variables.

Definition 5.3. *An instantiation σ of the basic RVs defined by a BLOG model $\mathcal{M}$ is* consistent *if and only if:*

- *For every element $u_{\nu,\bar{v},i}$ used in an assignment of the form $\sigma(V_f[\bar{u}]) = w$ or $\sigma(V_\nu[\bar{u}]) = m > 0$, $\sigma(V_\nu[\bar{v}]) \geq i$;*
- *For every fixed function symbol f with the interpretation $\tilde{f}$, $\sigma(V_f[\bar{u}]) = \tilde{f}(\bar{u})$; and*
- *For every element $u_{\nu,\bar{u}=\langle u_1,\ldots,u_m\rangle,i}$, generated by the number statement ν, with origin functions $g_1,\ldots,g_m$, for every $g_j \in \{g_1,\ldots,g_m\}$, $\sigma(V_{g_j}[u_{\nu,\bar{u},i}]) = u_j$. That is, origin functions give correct inverse maps.*

Lemma 5.4. *Every consistent assignment σ to the basic RVs for $\mathcal{M}$ defines a unique possible world in the vocabulary of $\mathcal{M}$.*

The proof of Lemma 5.4 is in Appx. F. In the following definition, we use the notation $e[\bar{u}/\bar{x}]$ to denote a substitution of every occurrence of the variable x_i with u_i in the expression e. For any BLOG model $\mathcal{M}$, let $V(\mathcal{M}) = BRV(\mathcal{M})$; for each $v \in V$, $\mathcal{X}_v$ is the measurable space corresponding to v. Let $E(\mathcal{M})$ consist of the following edges for every number statement or function application statement of the form $s(\bar{x})$:

- The edge $(V_g[\bar{w}], V_s[\bar{u}])$ if g is a function symbol in $\mathcal{M}$ such that $g(\bar{y})$ appears in $s(\bar{x})$, and either $g(\bar{w}) = g(\bar{y})[\bar{u}/\bar{x}]$ or an occurrence of $g(\bar{y})$ in $s(\bar{x})$ uses quantified variables $z_1,\ldots,z_n$, $\bar{u}'$ is a tuple of elements of the type of $\bar{z}$ and $g(\bar{w}) = g(\bar{y})[\bar{u}/\bar{x}][\bar{u}'/\bar{z}]$.
- The edge $(V_\nu[\bar{v}], V_s[\bar{u}])$, for element $u_{\nu,\bar{v},i} \in \bar{u}$.

Note that the first set of edges defined in $E(\mathcal{M})$ above may include infinitely many parents for $V_s[\bar{u}]$. Let the dependency statement in the BLOG model $\mathcal{M}$ corresponding to a number or function variable $V_s[\bar{f}]$ be s. Let $expr(s)$ be the set of expressions used in s. Each such statement then defines in a straightforward manner, a kernel $K_{s(\bar{u})} : \mathcal{X}_{expr(s(\bar{u}))} \to \mathcal{X}_{V_s[\bar{u}]}$. In order ensure consistent assignments, we include a special value *null* $\in \mathcal{X}_\tau$ for each τ in $\mathcal{M}$, and require that $K_{s(\bar{u})}(\sigma(pa(V_s[\bar{u}])), \{null\}^c) = 0$ whenever σ violates the first condition of consistent assignments (Def. 5.3). In other words, all the local kernels ensure are *locally consistent*: variables involving an object $u_{\nu,\bar{u},i}$ get a non-*null* assignment only if the assignment to its number statement represents the generation of at least i objects ($\sigma(V_\nu(\bar{u})) \geq i$). Each kernel of the form $K_{s(\bar{u})}$ can be transformed into a kernel $K_{pa(V_s[\bar{u}])}$ from its parent vertices (representing basic random variables) by composing the kernels determining the truth value of each expression $e \in expr(v)$ in terms of the basic random variables, with the kernel $K^e_{V_s[\bar{u}]}$. Let $\kappa(\mathcal{M}) = \{K_{pa(V_s[\bar{u}])} : V_s[\bar{u}] \in BRV(\mathcal{M})\}$.

```
1 fixed Real sigma = 1.0; // stddev of observation
2 random Real FakeCoinDiff ~
3  TruncatedGaussian(0.5, 1, 0.1, 1);
4 random Bool hasFakeCoin ~ BooleanDistrib(0.5);
5 random Real obsDiff ~ if hasFakeCoin
6   then Gaussian(FakeCoinDiff, sigma*sigma)
7   else Mix({ 0 -> 1.0 });
8 obs obsDiff = 0;
9 query hasFakeCoin;
```

Figure 3. BLOG code for the Scale example

Definition 5.5. *The MTBN M for a BLOG model $\mathcal{M}$ is defined using $V = V(\mathcal{M}), E = E(\mathcal{M})$, the set of measurable spaces $\{\mathcal{X}_v : v \in BRV(\mathcal{M})\}$ and the kernels for each vertex given by $\kappa(\mathcal{M})$.*

By Thm. 3.6, we have the main result of this section, which provides the theoretical foundation for the generalized BLOG language:

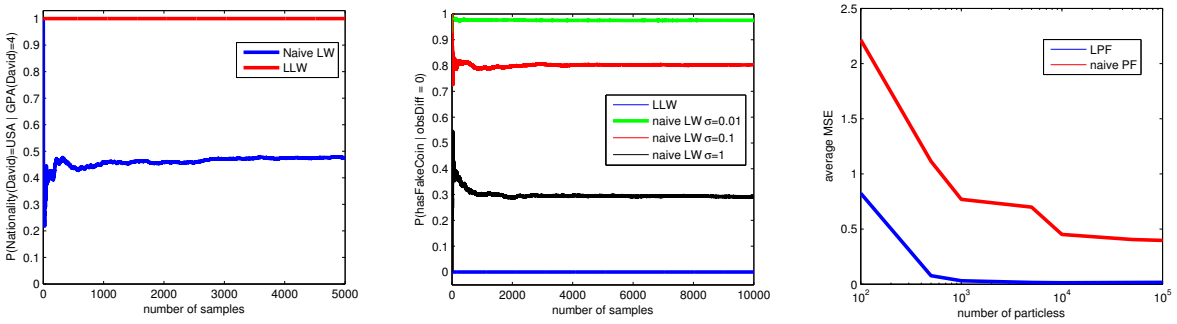
Theorem 5.6. *If the MTBN $\mathcal{M}$ for a BLOG model is a well-founded digraph, then $\mathcal{M}$ represents a unique measure μ on $\mathcal{X}_{BRV(\mathcal{M})}$.*

6. Experiment Results

We implemented the measure-theoretic extension of BLOG and evaluated our inference algorithms on three models where naive algorithms fail: (1) the GPA model (GPA); (2) the noisy scale model (Scale); and (3) a SSM, the aircraft tracking model (Aircraft-Tracking). The implementation is based on BLOG's C++ compiler (Wu et al., 2016).

GPA model: Fig. 1 presents the BLOG code for the GPA example as explained in Sec. 5. Since the GPA of David is exactly 4, Bayes rule implies that David must be from USA. We evaluate LLW and the naive LW on this model in Fig 4(a), where the naive LW converges to an incorrect posterior.

Scale model: In the noisy scale example (Fig. 3), we have an even number of coins and there might be a fake coin among them (Line 4). The fake coin will be slightly heavier than a normal coin (Line 2-3). We divide the coins into two halves and place them onto a noisy scale. When there is no fake coin, the scale always balances (Line 7).



(a) GPA model (b) Scale model (c) Aircraft-Tracking model

Figure 4. Experiment results on (a) the GPA model, (b) the noisy scale model and (c) the aicraft-tracking model.

When there is a fake coin, the scale will noisily reflect the weight difference with standard deviation σ (`sigma` in Line 6). Now we observe that the scale is balanced (Line 8) and we would like to infer whether a fake coin exists. We again compare LLW against the naive LW with different choices of the σ parameter in Fig. 4(b). Since the scale is precisely balanced, there must not be a fake coin. LLW always produces the correct answer but naive LW converges to different incorrect posteriors for different values of σ; as σ increases, naive LWs result approaches the true posterior.

Aircraft-Tracking model: Fig. 5 shows a simplified BLOG model for the aircraft tracking example. In this state-space model, we have $N = 6$ radar points (Line 1) and a single aircraft to track. Both the radars and the aircraft are considered as points on a 2D plane. The prior of the aircraft movement is a Gaussian process (Line 3 to 6). Each radar r has an effective range `radius(r)`: if the aircraft is within the range, the radar will noisily measure the distance from the aircraft to its own location (Line 13); if the aircraft is out of range, the radar will almost surely just output its radius (Line 10 to 11). Now we observe the measurements from all the radar points for T time steps and we want to infer the location of the aircraft. With the measure-theoretic extension, a generalized BLOG program is more expressive for modeling truncated sensors: if a radar outputs exactly its radius, we can surely infer that the aircraft must be out of the effective range of this radar. However, this information cannot be captured by the original BLOG language. To illustrate this case, we manually generated a synthesis dataset of $T = 8$ time steps[4] and evaluated LPF against the naive particle filter with different numbers of particles in Fig. 4(c). We take the mean of the samples from all the particles as the predicted aircraft location. Since we know the ground truth, we measure the average mean square error between the true location and the prediction. LPF accurately predicts the true locations while the naive PF converges to the incorrect results.

[4]The full BLOG programs with complete data are available at https://goo.gl/f7qLwy.

```
1  type t_radar; distinct t_radar R[6];
2  // model aircraft movement
3  random Real X(Timestep t) ~ if t == @0
4    then Gaussian(2, 1) else Gaussian(X(prev(t)), 4);
5  random Real Y(Timestep t) ~ if t == @0
6    then Gaussian(-1, 1) else Gaussian(Y(prev(t)), 4);
7  // observation model of radars
8  random Real obs_dist(Timestep t, t_radar r) ~
9    if dist(X(t),Y(t),r) > radius(r) then
10     mixed({radius(r)->0.999,
11     TruncatedGauss(radius(r),0.01,0,radius(r))->0.001})
12   else
13     TruncatedGauss(dist(X(t),Y(t),r),0.01,0,radius(r));
14 // observation and query
15 obs obs_dist(@0, R[0]) = ...;
16 ... // evidence numbers omitted
17 query X(t) for Timestep t;
18 query Y(t) for Timestep t;
```

Figure 5. BLOG code for the Aircraft-Tracking example

7. Conclusion

We presented a new formalization, measure-theoretic Bayesian networks, for generalizing the semantics of PPLs to include random variables with mixtures of discrete and continuous distributions. We developed provably correct inference algorithms for such random variables and incorporated MTBNs into a widely used PPL, BLOG. We believe that together with the foundational inference algorithms, our proposed rigorous framework will facilitate the development of powerful techniques for probabilistic reasoning in practical applications from a much wider range of scientific areas.

Acknowledgment

This work is supported by the DARPA PPAML program, contract FA8750-14-C-0011. Simon S. Du is funded by NSF grant IIS1563887, AFRL grant FA8750-17-2-0212 and DARPA D17AP00001.

References

Arora, N. S., Russell, S., and Sudderth, E. NET-VISA: Network processing vertically integrated seismic analysis. *Bulletin of the Seismological Society of America*, 103(2A):709–729, 2013.

Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M. A., Guo, J., Li, P., Riddell, A., et al. Stan: A probabilistic programming language. *Journal of Statistical Software*, 20(2):1–37, 2016.

Choi, J., Amir, E., and Hill, D. J. Lifted inference for relational continuous models. In *UAI*, volume 10, pp. 126–134, 2010.

Doucet, A., De Freitas, N., and Gordon, N. An introduction to sequential Monte Carlo methods. In *Sequential Monte Carlo methods in practice*, pp. 3–14. Springer, 2001.

Durrett, R. *Probability: Theory and Examples*. Cambridge University Press, 2013.

Gao, W., Kannan, S., Oh, S., and Viswanath, P. Estimating mutual information for discrete-continuous mixtures. In *Advances in Neural Information Processing Systems*, pp. 5988–5999, 2017.

Goodman, N. D. The principles and practice of probabilistic programming. In *ACM SIGPLAN Notices*, volume 48, pp. 399–402. ACM, 2013.

Goodman, N. D. and Stuhlmüller, A. The Design and Implementation of Probabilistic Programming Languages. http://dippl.org, 2014. Accessed: 2018-6-5.

Goodman, N. D., Mansinghka, V. K., Roy, D. M., Bonawitz, K., and Tenenbaum, J. B. Church: A language for generative models. In *UAI-08*, 2008.

Gutmann, B., Jaeger, M., and De Raedt, L. Extending problog with continuous distributions. In *Inductive Logic Programming*, pp. 76–91. Springer, 2011a.

Gutmann, B., Thon, I., Kimmig, A., Bruynooghe, M., and De Raedt, L. The magic of logical inference in probabilistic programming. *Theory and Practice of Logic Programming*, 11(4-5):663–680, 2011b.

Jech, T. *Set theory*. Springer, 2003.

Kallenberg, O. *Foundations of Modern Probability*. Springer, 2002. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0387953132.

Kersting, K. and De Raedt, L. Bayesian logic programming: Theory and tool. *Statistical Relational Learning*, pp. 291, 2007.

Kharchenko, P. V., Silberstein, L., and Scadden, D. T. Bayesian approach to single-cell differential expression analysis. *Nature methods*, 11(7):740, 2014.

Koller, D., McAllester, D., and Pfeffer, A. Effective Bayesian inference for stochastic programs. In *AAAI-97*, 1997.

Kulkarni, T. D., Kohli, P., Tenenbaum, J. B., and Mansinghka, V. Picture: A probabilistic programming language for scene perception. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pp. 4390–4399, 2015.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Le, T. A., Baydin, A. G., and Wood, F. Inference compilation and universal probabilistic programming. In *Artificial Intelligence and Statistics*, pp. 1338–1348, 2017.

Li, L. and Russell, S. J. The BLOG language reference. Technical report, Technical Report UCB/EECS-2013-51, EECS Department, University of California, Berkeley, 2013.

Mansinghka, V., Selsam, D., and Perov, Y. Venture: a higher-order probabilistic programming platform with programmable inference. *arXiv preprint arXiv:1404.0099*, 2014.

McAllester, D., Milch, B., and Goodman, N. D. Random-world semantics and syntactic independence for expressive languages. Technical report, 2008.

Milch, B., Marthi, B., Russell, S. J., Sontag, D., Ong, D. L., and Kolobov, A. BLOG: Probabilistic models with unknown objects. In *Proc. of IJCAI*, pp. 1352–1359, 2005a.

Milch, B., Marthi, B., Sontag, D., Russell, S., Ong, D. L., and Kolobov, A. Approximate inference for infinite contingent Bayesian networks. In *Tenth International Workshop on Artificial Intelligence and Statistics, Barbados*, 2005b. URL http://www.gatsby.ucl.ac.uk/aistats/AIabst.htm.

Milch, B. C. *Probabilistic models with unknown objects*. PhD thesis, University of California at Berkeley, Berkeley, CA, USA, 2006.

Nitti, D., De Laet, T., and De Raedt, L. Probabilistic logic programming for hybrid relational domains. *Machine Learning*, 103(3):407–449, 2016.

Pearl, J. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

Pfeffer, A. Figaro: An object-oriented probabilistic programming language. *Charles River Analytics Technical Report*, 137:96, 2009.

Pierson, E. and Yau, C. Zifa: Dimensionality reduction for zero-inflated single-cell gene expression analysis. *Genome biology*, 16(1):241, 2015.

Ramsey, N. All you need is the monad.. what monad was that again. In *PPS Workshop*, 2016.

Ritchie, D., Lin, S., Goodman, N. D., and Hanrahan, P. Generating design suggestions under tight constraints with gradient-based probabilistic programming. In *Computer Graphics Forum*, volume 34, pp. 515–526. Wiley Online Library, 2015.

Shan, C.-c. and Ramsey, N. Exact Bayesian inference by symbolic disintegration. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*, pp. 130–144. ACM, 2017.

Singla, P. and Domingos, P. Markov logic in infinite domains. In *In Proc. UAI-07*, 2007.

Staton, S. Commutative semantics for probabilistic programming. In *European Symposium on Programming*, pp. 855–879. Springer, 2017.

Stuhlmüller, A. and Goodman, N. D. Reasoning about reasoning by nested conditioning: Modeling theory of mind with probabilistic programs. *Cognitive Systems Research*, 28:80–99, 2014.

Tolpin, D., van de Meent, J. W., Yang, H., and Wood, F. Design and implementation of probabilistic programming language anglican. *arXiv preprint arXiv:1608.05263*, 2016. URL https://github.com/probprog/anglican-examples/blob/master/worksheets/indian-gpa.clj.

Tran, D., Kucukelbir, A., Dieng, A. B., Rudolph, M., Liang, D., and Blei, D. M. Edward: A library for probabilistic modeling, inference, and criticism. *arXiv preprint arXiv:1610.09787*, 2016.

Wang, J. and Domingos, P. Hybrid Markov logic networks. In *AAAI*, volume 8, pp. 1106–1111, 2008.

Wood, F., Meent, J. W., and Mansinghka, V. A new approach to probabilistic programming inference. In *Artificial Intelligence and Statistics*, pp. 1024–1032, 2014a.

Wood, F., van de Meent, J. W., and Mansinghka, V. A new approach to probabilistic programming inference. In *Proceedings of the 17th International conference on Artificial Intelligence and Statistics*, pp. 1024–1032, 2014b.

Wu, Y., Li, L., and Russell, S. BFiT: From possible-world semantics to random-evaluation semantics in open universe. *3rd NIPS Workshop on Probabilistic Programming*, 2014.

Wu, Y., Li, L., Russell, S., and Bodik, R. Swift: Compiled inference for probabilistic programming languages. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, 2016.