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 of the recent successes of probabilistic programming languages (PPLs) in AI applications, PPLs offer only limited support for discrete-continuous mixture random variables. 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 which are provably correct under the MTBN framework: lexicographic likelihood weighting (LLW) for general MTBNs and lexicographic particle filter (LPF), a specialized algorithm for state space models. We further integrate MTBN into a widely used PPL system, BLOG, and verify the effectiveness of our 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 its expressiveness power, PPLs have been used to solve many real-world applications, including Captcha (Le et al., 2017), seismic monitoring (Moore & Russell, 2017), 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).

A major drawback of existing PPLs is that they can only support discrete and continuous random variables but not their mixtures. In practical applications, we often have to deal with a mixture of continuous and discrete random variables. Combination of discrete and continuous distributions are ubiquitous 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. This kind of random variables has also been studied 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 which forces the expressed random variables to be either discrete or continuous (Goodman & Stuhlmüller, 2014; Tran et al., 2016; Pfeffer, 2009; Carpenter et al., 2016). Even for those PPLs which support mixtures of discrete and continuous variables by its *syntax* (Milch et al., 2005a; Goodman, 2013; Mansinghka et al., 2014; Wood et al., 2014a), the underlying *semantics* of these PPLs implicitly assume the random variables are not mixtures. Moreover, the inference algorithm associated with the semantics inherit the same assumption: if one defines a random variable which is a discrete-continuous mixture in such a PPL and directly adopts its default inference algorithm, the produced inference results can be wrong.

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

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

and the conditional probabilities are discrete-continuous mixtures

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

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 $\{g\}$ where $0 < g < 10$ is zero, as this range has a probability *density*. On the other hand if the student is American, the

[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.

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 variables, 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 require PPL researchers to accomplish the following two tasks.

- Inherit all the properties of existing PPL semantics[1] and extend it to handle *general* discrete-continuous mixtures;
- Design provably correct inference algorithms for the generalized semantics.

In this paper, we provide a general framework for the generalized PPL semantics and propose new algorithms based on this framework to handle mixture of discrete and continuous variables. Notably, our generalized framework can be also applied to other general measure spaces beyond discrete-continuous mixtures.

1.1. Main Contributions

Measure Theoretical Bayesian Nets Measure theory can be applied to handle the discrete-continuous mixtures or even more abstract measures. In this paper, we bring the measure-theoretical machinery to Bayesian nets and propose measure theoretical Bayesian nets (MTBNs). We give formal definitions of MTBNs and more importantly, we show theoretically, every MTBN represents a unique measure on the input space. Therefore, we have a sound theoretical foundation and can then use MTBN to provide generalized semantics for any PPL.

Inference Algorithms Unlike discrete or continuous random variables, the existence of discrete-continuous mixtures often requires a user to incorporate specialized tricks for different models on top of classical PPL inference algorithms. Based on MTBN, we propose a general and provably correct inference algorithm, lexicographic likelihood weighting (LLW). In addition, we further adapt LLW into the sequential Monte Carlo (SMC) framework for state space models.

Incorporating MTBN into an existing PPL We incorporate MTBNs to a widely used PPL, Bayesian Logic (BLOG) (Milch et al., 2005a). With simple modifications, we 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 MTBN can be carried to measure-theoretic BLOG. We also implement 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 its theoretical properties. In Section 4, we propose a general inference algorithm, lexicographic likelihood weighting (LLW) for general MTBN and its variant, lexicographic particle filter (LPF), which is specialized for state space models. 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. We conclude in Section 7 and defer most technical details to appendix.

2. Related Work

The motivating GPA example has been also discussed as a special case in many other PPL systems, such as (Tolpin et al., 2016) and (Nitti et al., 2016). These systems proposed different specialized tricks to derive the correct answer particularly for this example but do not provide any general correctness guarantees. On the other hand, our MTBN framework is much more general and LLW inference algorithm is provably correct.

The closest related work to our framework is by Milch (2006), who utilize a supportive numbering of random variables, implying that each random variable has finitely many consistent parents. In addition, they only handle random variables with countably infinite ranges. The BLP framework presented by Kersting & De Raedt (2007) unifies logic programming with probability models, but requires each random variable to be influenced by a finite set of random

[1] (1) Random variables with infinitely (even uncountably) many parents; (2) Establishment of conditional independencies implied by an infinite graph; and (3) Open-universe semantics in terms of the possible worlds in the vocabulary of the model.

variables in order to define the semantics. This amounts to requiring only finitely many ancestors of each node. 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 analyzing the semantics properties of PPLs from measure theoretical perspectives (Shan & Ramsey, 2017; Staton, 2017; Ramsey, 2016).

3. Measure-Theoretic Bayesian Networks

In this section, we introduce *measure-theoretic Bayesian nets (MTBNs)* and prove that a 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 $E \subseteq V \times V$. Write $u \to v$ if $(u, v) \in E$, and $u \mapsto v$ if there is 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., 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 $v_0 \leftarrow v_1 \leftarrow v_2 \leftarrow \dots$.*

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 $\pi^U_W : \mathcal{X}_U \to \mathcal{X}_W$ 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. *A 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 Nets – conditional independence and conditional distributions defined by parent-child relationships. Here we implicit 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 our definition of MTBN is proper.

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

Theorem 3.6 lays out the foundation of MTBN. Its proof requires a series of intermediate results. We first define 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. The full proof is in Appx. B.

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 into the SMC framework.

4.1. Lexicographic likelihood weighting

Suppose we have a MTBN with finitely many random variables $X_1, \ldots, X_N$, and that, without loss of generality, we observe real-valued random variables $X_1, \ldots, X_M$ for $M < N$ as evidence. Suppose the distribution of X_i given its parents $X_{\text{pa}(i)}$ is a mixture between a density $f_i(x_i|x_{\text{pa}(i)})$ with respect to Lebesgue and a discrete distribution $F_i(x_i|x_{\text{pa}(i)})$, i.e., for any $\epsilon > 0$, we have $P(X_i \in [x_i - \epsilon, x_i]|X_{\text{pa}(i)}) = \sum_{x \in [x_i - \epsilon, x_i]} F_i(x_i|X_{\text{pa}(i)}) + \int_{x_i - \epsilon}^{x_i} f_i(x|X_{\text{pa}(i)})\, dx$. Note this implies $F_i(x_i|x_{\text{pa}(i)})$ 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:

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

Finally, having K samples $x^{(1)}, \ldots, x^{(K)}$ 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* $\mathbb{E}[f(X)|X_{1:M}]$.

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.

Algorithm 1 Lexicographic Likelihood Weighting

Require: densities f, masses F, evidences E, and K.
  for $i = 1 \ldots K$ **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 $\left(\sum_{i: d^{(i)} = d^\star} w^{(i)} f(x^{(i)})\right) / \left(\sum_{i: d^{(i)} = d^\star} w^{(i)}\right)$

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 which to weight 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 $i = 1, 2, \ldots$ 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 $\alpha_t(y), \alpha_{t+1}(y), \ldots$ 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 $(\alpha_1(Y), \ldots)$, 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 only prove where the evidence variables 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

Likelihood weighting based algorithms suffer from curse of dimensionality. One important class of models with high dimensionality are state space 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 $0 < t \le T$.

Sequential Monte Carlo (SMC) (Doucet et al., 2001), i.e., particle filer, is a widely adopted 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.

In the MTBN setting, the distribution of Y_t 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 particle filter should be accordingly modified: following the idea

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
    $d^\star \leftarrow \min_{k:w^{(k)}\ne 0} d^{(k)}$
    $\forall k : d^{(k)} > d^\star$, $w^{(k)} \leftarrow 0$
    Output $\left(w^{(k)} f(x_t^{(k)})\right) / \left(\sum_k w^{(k)}\right)$
    resample particles according to $w^{(k)}$
  end for

from LLW, when computing the weight of a particle, we enumerate all the observations $y_{t,i}$[2] at time step t and again update a tuple (d, w), initially (0,1), by

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

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 classical proof from importance sampling to particle filter.

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. Here our focus is 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[3].

We begin with a brief description of the core syntax of BLOG, with particular emphasis on (1) the most important existing syntax for open-universe semantics, i.e., the number

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

[3] 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.

statement[4], and (2) a newly introduced syntax to accommodate MTBN, 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. Line 3 to 5 defines 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. Line 7 and 13 define two random variables by keyword `random`. Line 7 to 12 states that the GPA of an applicant is distributed as a mixture of weighted discrete and continuous distributions. For US applicants, the range of values $0 < GPA < 4$ follows a truncated Gaussian with bounds 0 and 4 (line 9). The probability mass outside the range is attributed to the corresponding bounds: $P(GPA = 0) = P(GPA = 4) = 10^{-4}$ (lines 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 $\bar{x} = x_1, \dots, x_k$; φ_j are first-order formulas with free variables $\bar{y}_j$; $\bar{e}_j$ are tuples of expressions over a subset of $x_1, \dots, x_k$; 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.

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

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:

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

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 the 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:

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

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 the measure-theoretical BLOG, we only allow `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 study 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* $\{\tau_1, \dots, \tau_k\}$ *is* $\bigcup_{j \in \mathbb{N}} \{\mathcal{U}_j\}$, *where*

- $\mathcal{U}_0 = \langle U_1^0, \dots, U_k^0 \rangle$, $U_j^0 = \{c_j : c_j$ *is a* `distinct` τ_i *constant in* $\mathcal{M}\}$
- $\mathcal{U}_{i+1} = \langle U_1^{i+1}, \dots, U_k^{i+1} \rangle$, *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* $\mathcal{U}^i$, $m \in \mathbb{N}\}$

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:*

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

- *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 $\kappa(\mathcal{M}) = \{K_{pa(V_s[\bar{u}])} : V_s[\bar{u}] \in BRV(\mathcal{M})\}$.

Definition 5.5. *The network 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:

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

This theorem provides the theoretical foundation of the generalized BLOG language.

6. Experiment Results

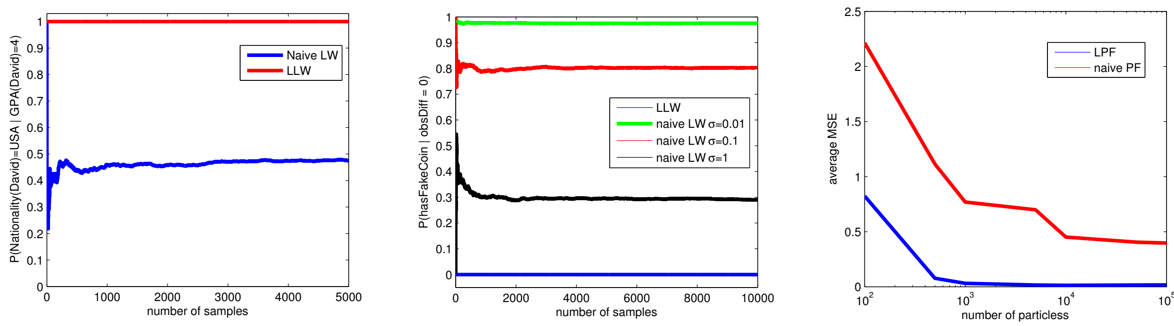
We implement the measure-theoretic extension of BLOG and evaluate our inference algorithms on three models where naive algorithms fail: (1) the GPA model (GPA); (2) the noisy scale model (Scale); and (3) a SSM, the aircraft tracking model (Aircraft-Tracking). The implementation is based on the BLOG's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

GPA model: Fig. 1 contains the BLOG code for the GPA example as explained in Sec. 5. Since the GPA of David is exactly 4, the Bayes rule tells that David must be from USA. We evaluate LLW and the naive LW on this model in Fig 4(a), where the naive LW produces a completely wrong answer.

```
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

Scale model: In the noisy scale (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 no fake coin, the scale always balances (Line 7). 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



(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.

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 while for naive LW, the result is incorrect and highly depends on the σ parameter: as σ increases, the output approaches the true probability.

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 noisily measure the distance from the aircraft to its own location (Line 13); if the aircraft is out of range, the radar with almost surely just outputs its radius (Line 10 to 11). Now we observe the measurements from all the radars points for T time steps and we want to infer the location of the aircraft. With the measure-theoretic extension, the generalized BLOG program provides more expressiveness power for modelling 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 PPL. To illustrate this case, we manually generate a synthesis dataset of $T = 8$ time steps[5] and evaluate LPF against the naive particle filter with different number 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 predicted the true locations while naive PF converges to the wrong results.

[5]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

In this paper we proposed measure-theoretic Bayesian network, a general framework to generalize existing PPL semantics to support random variables over arbitrary measure spaces, and provably correct inference algorithms in this framework to handle discrete-continuous mixtures. We also incorporate MTBN into a widely used PPL, BLOG, by a simple syntax extension and implement the algorithms into the generalized BLOG PPL, which makes the PPL system practical for a much larger domain of applications.

We also 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

Carpenter, Bob, Gelman, Andrew, Hoffman, Matt, Lee, Daniel, Goodrich, Ben, Betancourt, Michael, Brubaker, Michael A, Guo, Jiqiang, Li, Peter, Riddell, Allen, et al. Stan: A probabilistic programming language. *Journal of Statistical Software*, 20(2): 1–37, 2016.

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

Doucet, Arnaud, De Freitas, Nando, and Gordon, Neil. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pp. 3–14. Springer, 2001.

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

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

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

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

Goodman, Noah D, Mansinghka, Vikash K, Roy, Daniel M, Bonawitz, Keith, and Tenenbaum, Joshua B. Church: A language for generative models. In *UAI-08*, 2008.

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

Gutmann, Bernd, Thon, Ingo, Kimmig, Angelika, Bruynooghe, Maurice, and De Raedt, Luc. The magic of logical inference in probabilistic programming. *Theory and Practice of Logic Programming*, 11(4-5):663–680, 2011b.

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

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

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

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

Koller, Daphne, McAllester, David, and Pfeffer, Avi. Effective bayesian inference for stochastic programs. In *AAAI-97*, 1997.

Kulkarni, Tejas D, Kohli, Pushmeet, Tenenbaum, Joshua B, and Mansinghka, Vikash. 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, Brenden M, Salakhutdinov, Ruslan, and Tenenbaum, Joshua B. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Le, Tuan Anh, Baydin, Atilim Gunes, and Wood, Frank. Inference compilation and universal probabilistic programming. In *Artificial Intelligence and Statistics*, pp. 1338–1348, 2017.

Li, Lei and Russell, Stuart J. The blog language reference. Technical report, Technical Report UCB/EECS-2013-51, EECS Department, University of California, Berkeley, 2013.

Mansinghka, Vikash, Selsam, Daniel, and Perov, Yura. Venture: a higher-order probabilistic programming platform with programmable inference. *arXiv preprint arXiv:1404.0099*, 2014.

McAllester, David, Milch, Brian, and Goodman, Noah D. Random-world semantics and syntactic independence for expressive languages. Technical report, 2008.

Milch, Brian, Marthi, Bhaskara, Russell, Stuart J., Sontag, David, Ong, Daniel L., and Kolobov, Andrey. BLOG: Probabilistic models with unknown objects. In *Proc. of IJCAI*, pp. 1352–1359, 2005a.

Milch, Brian, Marthi, Bhaskara, Sontag, David, Russell, Stuart, Ong, Daniel L., and Kolobov, Andrey. 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, Brian Christopher. *Probabilistic models with unknown objects*. PhD thesis, University of California at Berkeley, Berkeley, CA, USA, 2006.

Moore, David and Russell, Stuart. Signal-based bayesian seismic monitoring. In *Artificial Intelligence and Statistics*, pp. 1293–1301, 2017.

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

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

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

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

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

Ritchie, Daniel, Lin, Sharon, Goodman, Noah D, and Hanrahan, Pat. 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, Chung-chieh and Ramsey, Norman. 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, Parag and Domingos, Pedro. Markov logic in infinite domains. In *In Proc. UAI-07*, 2007.

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

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

Tolpin, David, van de Meent, Jan Willem, Yang, Hongseok, and Wood, Frank. 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, Dustin, Kucukelbir, Alp, Dieng, Adji B., Rudolph, Maja, Liang, Dawen, and Blei, David M. Edward: A library for probabilistic modeling, inference, and criticism. *arXiv preprint arXiv:1610.09787*, 2016.

Wang, Jue and Domingos, Pedro. Hybrid markov logic networks. In *AAAI*, volume 8, pp. 1106–1111, 2008.

Wood, Frank, Meent, Jan Willem, and Mansinghka, Vikash. A new approach to probabilistic programming inference. In *Artificial Intelligence and Statistics*, pp. 1024–1032, 2014a.

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

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

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

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

A. Background on Measure-theoretical Probability Theory

We assume familiarity with measure-theoretic approaches to probability theory, but provide the fundamental definitions. The standard Borel σ-algebra is assumed in all the discussion. See (Durrett, 2013) and (Kallenberg, 2002) for introduction and further details.

A **measurable space** $(X, \mathcal{X})$ (space, for short) is an underlying set X paired with a σ-algebra $\mathcal{X} \subseteq 2^X$ of measurable subsets of X, i.e., a family of subsets containing the underlying set X which is closed under complements and countable unions. We'll denote the measurable space simply by $\mathcal{X}$ where no ambiguity results. A function $f\colon \mathcal{X} \to \mathcal{Y}$ between measurable spaces is measurable if measurable sets pullback to measurable sets: $f^{-1}(B) \in \mathcal{X}$ for all $B \in \mathcal{Y}$. A **measure** μ on a measurable space $\mathcal{X}$ is a function $\mu\colon \mathcal{X} \to [0, \infty]$ which satisfies countable additivity: for any countable sequence $A_1, A_2, \dots \in \mathcal{X}$ of disjoint measurable sets $\mu(\cup_i A_i) = \sum_i \mu(A_i)$. $\mathbb{P}_\mu[S]$ denotes the probability of a statement S under the base measure μ, and similarly for conditional probabilities. A probability kernel is the measure-theoretic generalization of a conditional distribution. It is commonly used to construct measures over a product space, analogously to how conditional distributions are used to define joint distributions in the chain rule.

Definition A.1. *A **probability kernel** K from one measurable space $\mathcal{X}$ to another $\mathcal{Y}$ is a function $K\colon X \times \mathcal{Y} \to [0, 1]$ such that (a) for every $x \in X$, $K(x, \cdot)$ is a probability measure over $\mathcal{Y}$, and (b) for every $B \in \mathcal{Y}$, $K(\cdot, B)$ is a measurable function from $\mathcal{X}$ to $[0, 1]$.*

Given an arbitrary index set T and spaces $\mathcal{X}_t$ for each index $t \in T$, the **product space** $\mathcal{X} = \prod_{t \in T} \mathcal{X}_t$ is the space with underlying set $X = \prod_{t \in T} X_t$ the Cartesian product of the underlying sets, adorned with the smallest σ-algebra such that the projection functions $\pi_t\colon \mathcal{X} \to \mathcal{X}_t$ are measurable.

B. MTBNs Represent Unique Measures

We prove here Theorem 3.6. Its proof requires a series of intermediate results. We first define a projective family of measures. This gives a way to recursively construct our measure μ. We define a notion of consistency such that every consistent projective family constructs a measure that M represents. We end by giving an explicit characterization of the unique consistent projective family, and thus of the unique measure M represents. The appendix contains additional technical material required in the proofs.

Intuitively, the main objective of this section is to show that an MTBN defines a unique measure that "factorizes" according to the network, as an extension to the corresponding result for Bayes Nets.

B.1. Consistent projective family of measures

Let K be a kernel from $\mathcal{X} \to \mathcal{Y}$ and L a kernel from $\mathcal{Y} \to \mathcal{Z}$. Their composition $K \circ L$ (note the ordering!) is a kernel from $\mathcal{X}$ to $\mathcal{Z}$ defined for $x \in X, C \in \mathcal{Z}$ by:

$$(K \circ L)(x, C) = \int K(x, dy) \int L(y, dz)\, 1_C(z). \tag{5}$$

To allow uniform notation, we will treat measurable functions and measures as special cases of kernels. A measurable function $f\colon X \to Y$ corresponds to the kernel K_f from $\mathcal{X}$ to $\mathcal{Y}$ given by $K_f(x, B) = 1(f(x) \in B)$ for $x \in X$ and $B \in \mathcal{Y}$. A measure μ on a space $\mathcal{X}$ is a kernel K_μ from 1, the one element measure space, to $\mathcal{X}$ given by $K_\mu(\cdot, A) = \mu(A)$ for $A \in \mathcal{X}$. Where this yields no confusion, we use f and μ in place of K_f and K_μ. (5) simplifies if the kernels are measures or functions. Let μ be a measure on $\mathcal{Y}_1$, K be a kernel from $\mathcal{X}_1$ to $\mathcal{Y}_1$, f be a measurable function from $\mathcal{X}_2$ to $\mathcal{X}_1$, and g be a measurable function from $\mathcal{Y}_1$ to $\mathcal{Y}_2$. Then $\mu \circ g$ is a measure on $\mathcal{Y}_2$ and $f \circ K \circ g$ is a kernel from $\mathcal{X}_2$ to $\mathcal{Y}_2$ with: $(\mu \circ g)(B) = \mu(g^{-1}(B))$, and $(f \circ K \circ g)(x, B) = K(f(x), g^{-1}(B))$.

Let Λ denote the class of upwardly closed sets: subsets of V containing all their elements' parents.

Definition B.1. *A **projective family** of measures is a family $\{\mu_U : U \in \Lambda\}$ consisting of a measure μ_U on $\mathcal{X}_U$ for every $U \in \Lambda$ such that whenever $W \subseteq U$ we have $\mu_W = \mu_U \circ \pi^U_W$, i.e., for all $A \in \mathcal{X}_W$, $\mu_W(A) = \mu_U((\pi^U_W)^{-1}(A))$.*

Def. B.1 captures the measure-theoretic version of the probability of a subset of variables being equal to the marginals obtained while "summing out" the probabilities of the other variables in a joint distribution.

Definition B.2. *Let μ be a measure on a measure space $\mathcal{X}$, and K a kernel from $\mathcal{X}$ to a measure space $\mathcal{Y}$. Then $\mu \otimes K$ is the measure on $\mathcal{X} \times \mathcal{Y}$ defined for $B \in \mathcal{X} \otimes \mathcal{Y}$ by: $(\mu \otimes K)(B) = \int \mu(dx) \int K(x, dy)\, 1_B(x, y)$.*

Def. B.2 defines the operation of composing a conditional probability with a prior on a parent, to obtain the corresponding joint distribution.

Definition B.3. *Let K_w for $w \in W$ be kernels from $\mathcal{X}_U$ to $\mathcal{X}_{\{w\}}$. Denote by $\prod_{w \in W} K_w$ the kernel from $\mathcal{X}_U$ to $\mathcal{X}_W$ defined for each $x_U \in \mathcal{X}_U$ by the infinite product of measures: $\left(\prod_{w \in W} K_w\right)(x_U, \cdot) = \otimes_{w \in W} K_w(x_U, \cdot)$.*

See (Kallenberg, 2002) 1.27 and 6.18 for definition and existence of infinite products of measures. Def. B.3 captures the kernel representation for taking the equivalent of products of conditional distributions of a set of variables with a common set U of parents.

Definition B.4. *A projective family $\{\mu_U : U \in \Lambda\}$ is* ***consistent with*** *M if for any $W, U \in \Lambda$ such that $W \subset U$ and $\mathrm{pa}(U) \subseteq W$, then: $\mu_U = \mu_W \otimes \prod_{u \in U \setminus W} (\pi^W_{\mathrm{pa}(u)} \circ K_u)$.*

Consistency in Def. B.4 captures the global condition that we would like to see in a generalization of a Bayes network. Namely, the distribution of any set of parent-closed random variables should "factorize" according to the network

A projective family $\{\mu_U : U \in \Lambda\}$ is consistent with M exactly when M represents μ_V:

Lemma B.5. *Let μ be a measure on $\mathcal{X}_V$, and define the projective family $\{\mu_U : U \in \Lambda\}$ by $\mu_U = \mu \circ \pi^V_U$. This projective family is consistent with M iff M represents μ.*

Proof. First we'll relate consistency (Def. 8) with conditional expectation and distribution properties of random variables. Take any $W, U \in \Lambda$ such that $W \subset U$ and $\mathrm{pa}(U) \subseteq W$ and observe that the following are equivalent:

- $\mu_U = \mu_W \otimes \prod_{u \in U \setminus W} (\pi^W_{\mathrm{pa}(u)} \circ K_u)$
- $\prod_{u \in U \setminus W} (\pi^W_{\mathrm{pa}(u)} \circ K_u)$ is a version of the conditional distribution of $X_{U \setminus W}$ given X_W,
- K_u is a version of the conditional distribution of X_u given $X_{\mathrm{pa}(u)}$ for all $u \in U \setminus W$, and $\{X_W, X_u : u \in U \setminus W\}$ are mutually independent conditional on $X_{\mathrm{pa}(U)}$.

The forward direction is straightforward. For the converse we use the fact that conditional independence of families of random variables holds if it holds for all finite subsets, establishing that by chaining conditional independence (see (Kallenberg, 2002) p109 and 6.8). □

Lemma B.5 shows that Def. B.4 follows iff an MTBN represents the joint distribution – in other words, it follows iff the local Markov property holds.

B.2. There exists a unique consistent family

Each vertex $v \in V$ is assigned the unique minimal ordinal $d(v)$ such that $d(u) < d(v)$ whenever $(u, v) \in E$ (see (Jech, 2003) for an introduction to ordinals). For any $U \in \Lambda$ denote by $U^\alpha = \{u \in U : v(u) < \alpha\}$ the restriction of U to vertices of depth less than α. Defining $D = \sup_{v \in V}(d(v) + 1)$, the least strict upper bound on depth, we have that $U^D = U$ for all $U \in \Lambda$. In the following, fix a limit ordinal λ.

Definition B.6. *$\{\nu_\alpha : \alpha < \lambda\}$ is a* ***projective sequence of measures*** *on $\mathcal{X}_{U_\alpha}$ if whenever $\alpha < \beta < \lambda$ we have $\nu_\alpha = \nu_\beta \circ \pi^{U_\beta}_{U_\alpha}$.*

Def. B.6 generalizes the notion of subset relationships and the marginalization operations that hold between supersets and subsets to the case of infinite dependency chains

Definition B.7. *The limit $\lim_{\alpha<\lambda} \nu_\alpha$ of a projective sequence $\{\nu_\alpha : \alpha < \lambda\}$ of measures is the unique measure on $\mathcal{X}_U$ such that $\nu_\alpha = (\lim_{\alpha<\lambda} \nu_\alpha) \circ \pi^U_{U_\alpha}$ for all $\alpha < \beta$.*

Definition B.8. *Given any $U \in \Lambda$, inductively define a measure μ^α_U on $\mathcal{X}_{U^\alpha}$ by*

$$\mu^0_U = 1,$$
$$\mu^{\alpha+1}_U = \mu^\alpha_U \otimes \prod_{v \in U : d(v) = \alpha} (\pi^{U^\alpha}_{\mathrm{pa}(v)} \circ K_v),$$
$$\mu^\lambda_U = \lim_{\alpha<\lambda} \mu^\alpha_U \qquad \text{if } \lambda \text{ is a limit ordinal.}$$

μ^α_U stabilizes for $\alpha \geq D$ to define a measure on $\mathcal{X}_U$.

The above definition is coherent as μ^α_U can be inductively shown to be a projective sequence. Lemma B.9 and B.10 allow us to show in Theorem B.11 that $\{\mu^D_U : U \in \Lambda\}$ is the unique consistent projective family of measures.

Lemma B.9. *If $W \subseteq U$ for $W, U \in \Lambda$, then for all α: $\mu^\alpha_W = \mu^\alpha_U \circ \pi^{U^\alpha}_{W^\alpha}$.*

Proof is in Appx. C.

Lemma B.10. *If $W \subset U$ where $W, U \in \Lambda$, and if $\mathrm{pa}(U) \subseteq W$, then $W^\alpha \subset U^\alpha$, $\mathrm{pa}(U^\alpha) \subseteq W^\alpha$, and $\mu^\alpha_U = \mu^\alpha_W \otimes \prod_{u \in U^\alpha \setminus W^\alpha} (\pi^{W^\alpha}_{\mathrm{pa}(u)} \circ K_u)$.*

Proof is in Appx. D.

Using the above, the following shows MTBNs satisfy properties (1-3) from footnote 1 in introduction:

Theorem B.11. *$\{\mu^D_U : U \in \Lambda\}$ is the unique projective family of measures consistent with M.*

Proof is in Appx. E.

Intuitively, by Lemma B.9 and Lemma B.10, we assert that consistency holds for any ordinal-bounded (prefix in terms of parent ordering) sub-network. Then the main result, Thm. B.11, follows by setting this bound appropriately. Finally Lemma B.5 and Theorem B.11 lead to Theorem 3.6.

C. Proof for Lemma B.9

Proof. Proof by induction. Trivially true for $\alpha = 0$, so suppose this holds for α, and consider $\alpha + 1$. Then:

$$\begin{aligned}
\mu^{\alpha+1}_W &= \mu^\alpha_W \otimes \prod_{v \in W : d(v) = \alpha} (\pi^{W^\alpha}_{\mathrm{pa}(v)} \circ K_v) \\
&= \left(\mu^\alpha_U \circ \pi^{U^\alpha}_{W^\alpha}\right) \\
&\quad \otimes \left(\left(\prod_{v \in U : d(v) = \alpha} (\pi^{W^\alpha}_{\mathrm{pa}(v)} \circ K_v) \right) \circ \pi^{U^{\alpha+1} \setminus U^\alpha}_{W^{\alpha+1} \setminus W^\alpha} \right) \\
&= \left(\mu^\alpha_U \otimes \left(\pi^{U^\alpha}_{W^\alpha} \circ \prod_{v \in U : d(v) = \alpha} (\pi^{W^\alpha}_{\mathrm{pa}(v)} \circ K_v) \right) \right) \\
&\quad \circ (\pi^{U^\alpha}_{W^\alpha} \times \pi^{U^{\alpha+1} \setminus U^\alpha}_{W^{\alpha+1} \setminus W^\alpha}) \\
&= \left(\mu^\alpha_U \otimes \prod_{v \in U : d(v) = \alpha} (\pi^{U^\alpha}_{\mathrm{pa}(v)} \circ K_v) \right) \circ \pi^{U^{\alpha+1}}_{W^{\alpha+1}} \\
&= \mu^{\alpha+1}_U \circ \pi^{U^{\alpha+1}}_{W^{\alpha+1}}
\end{aligned}$$

The first step by Def. 12, the second by inductive hypothesis and Lemma G.11 as $\{v \in W : d(v) = \alpha\} = W^{\alpha+1} \setminus W^\alpha$ and $\{v \in$

$U : d(v) = \alpha\} = U^{\alpha+1} \setminus U^{\alpha}$, the third by Lemma G.6, the fourth by Lemma G.10 since $\pi^{U^\alpha}_{\mathrm{pa}(v)} = \pi^{W^\alpha}_{\mathrm{pa}(v)} \circ \pi^{U^\alpha}_{W^\alpha}$ and by elementary properties of projections, and the fifth by Definition B.8.

Finally, suppose λ is a limit ordinal. We need to show:

$$\lim_{\alpha<\lambda} \left(\mu^\alpha_U \circ \pi^{U^\alpha}_{W^\alpha}\right) = \left(\lim_{\alpha<\lambda} \mu^\alpha_U\right) \circ \pi^{U^\lambda}_{W^\lambda}.$$

This follows from Lemma G.2 because for all $\alpha < \lambda$ we have:

$$\begin{aligned}\left(\left(\lim_{\alpha<\lambda} \mu^\alpha_U\right) \circ \pi^{U^\lambda}_{W^\lambda}\right) \circ \pi^{W^\lambda}_{W^\alpha} &= \left(\left(\lim_{\alpha<\lambda} \mu^\alpha_U\right) \circ \pi^{U^\lambda}_{U^\alpha}\right) \circ \pi^{U^\alpha}_{W^\alpha} \\ &= \mu^\alpha_U \circ \pi^{U^\alpha}_{W^\alpha}\end{aligned}$$

The first by properties of projections, the second by Lemma G.2 characterizing limits. □

D. Proof for Lemma B.10

Proof. Trivial for $\alpha = 0$, so suppose this holds for α, and consider $\alpha + 1$. Then:

$$\begin{aligned}&\mu^{\alpha+1}_U \\ &= \mu^\alpha_U \otimes \prod_{v\in U:d(v)=\alpha} (\pi^{U^\alpha}_{\mathrm{pa}(v)} \circ K_v) \\ &= \mu^\alpha_W \otimes \prod_{u\in U^\alpha\setminus W^\alpha} (\pi^{W^\alpha}_{\mathrm{pa}(u)} \circ K_u) \otimes \prod_{v\in U:d(v)=\alpha} (\pi^{U^\alpha}_{\mathrm{pa}(v)} \circ K_v) \\ &= \mu^\alpha_W \otimes \prod_{u\in U^{\alpha+1}\setminus W^\alpha} (\pi^{W^\alpha}_{\mathrm{pa}(u)} \circ K_u) \\ &= \mu^\alpha_W \otimes \prod_{v\in W:d(v)=\alpha} (\pi^{W^\alpha}_{\mathrm{pa}(v)} \circ K_v) \\ &\qquad \otimes \prod_{u\in U^{\alpha+1}\setminus W^{\alpha+1}} (\pi^{W^{\alpha+1}}_{\mathrm{pa}(u)} \circ K_u) \\ &= \mu^{\alpha+1}_W \otimes \prod_{u\in U^{\alpha+1}\setminus W^{\alpha+1}} (\pi^{W^{\alpha+1}}_{\mathrm{pa}(u)} \circ K_u),\end{aligned}$$

The first step by Definition B.8, the second by inductive hypothesis. The third by Lemmas G.8 and G.9 since $U^{\alpha+1} \setminus W^\alpha = U^\alpha \setminus W^\alpha \cup \{v \in U : d(v) = \alpha\}$ where the union is disjoint, and as $\mathrm{pa}(v) \subseteq W^\alpha$ when $v \in U$ and $d(v) = \alpha$ implies that $\pi^{U^\alpha}_{\mathrm{pa}(v)} = \pi^{U^\alpha}_{W^\alpha} \circ \pi^{W^\alpha}_{\mathrm{pa}(v)}$. The fourth by Lemmas G.8 and G.9 since $U^{\alpha+1} \setminus W^\alpha = U^{\alpha+1} \setminus W^{\alpha+1} \cup \{v \in W : d(v) = \alpha\}$ where the union is disjoint, and as $\mathrm{pa}(u) \subseteq W^\alpha$ when $u \in U^{\alpha+1} \setminus W^{\alpha+1}$ implies that $\pi^{W^{\alpha+1}}_{\mathrm{pa}(v)} = \pi^{W^{\alpha+1}}_{W^\alpha} \circ \pi^{W^\alpha}_{\mathrm{pa}(v)}$. Finally, the fifth by Definition B.8.

Finally, suppose λ is a limit ordinal. The result will follow from the inductive hypothesis, Definition B.8, and as limits preserve products Lemma G.7 if we can show that

$$\lim_{\alpha<\lambda} \prod_{u\in U^\alpha\setminus W^\alpha} (\pi^{W^\alpha}_{\mathrm{pa}(u)} \circ K_u) = \prod_{u\in U^\lambda\setminus W^\lambda} (\pi^{W^\lambda}_{\mathrm{pa}(u)} \circ K_u).$$

First we must show the limit on the left is well-defined. Note that the kernel inside the limit maps from $\mathcal{X}_{W^\alpha}$ to $\mathcal{X}_{U^\alpha\setminus W^\alpha}$. As W^α and $U^\alpha \setminus W^\alpha$ are both increasing sets, we verify projective sequence property by taking any $\beta > \alpha$ and observing that

$$\begin{aligned}&\pi^{W^\beta}_{W^\alpha} \circ \prod_{u\in U^\alpha\setminus W^\alpha} (\pi^{W^\alpha}_{\mathrm{pa}(u)} \circ K_u) \\ &= \prod_{u\in U^\alpha\setminus W^\alpha} (\pi^{W^\beta}_{\mathrm{pa}(u)} \circ K_u) \\ &= \left(\prod_{u\in U^\beta\setminus W^\beta} (\pi^{W^\beta}_{\mathrm{pa}(u)} \circ K_u)\right) \circ \pi^{U^\beta\setminus W^\beta}_{U^\alpha\setminus W^\alpha}\end{aligned}$$

the first step from Lemma G.10 and properties of projections, and the second from Lemma G.11.

Finally, we must show the expression on the right satisfies the properties characterizing the limit. However, observe this follows from our demonstration of the projective sequence property above by simply replacing β with λ. □

E. Proof for Theorem B.11

Proof. That this is a consistent projective family follows from Lemmas B.9 and B.10 since $U^D = U$ for all $U \in \Lambda$.

For uniqueness, let $\{\hat{\mu}_U : U \in \Lambda\}$ be a consistent projective family of measures, any fix any $U \in \Lambda$. We'll show inductively that $\hat{\mu}_{U^\alpha} = \mu^\alpha_U$, and thus with $\alpha = D$ that $\hat{\mu}_U = \mu_U$, giving our result. This is trivial for $\alpha = 0$, so inductively suppose it holds for α. But then:

$$\begin{aligned}\hat{\mu}_{U^{\alpha+1}} &= \hat{\mu}_{U^\alpha} \otimes \prod_{u\in U^{\alpha+1}\setminus U^\alpha} (\pi^{U^\alpha}_{\mathrm{pa}(u)} \circ K_u) \\ &= \mu^\alpha_U \otimes \prod_{u\in U^{\alpha+1}\setminus U^\alpha} (\pi^{U^\alpha}_{\mathrm{pa}(u)} \circ K_u).\end{aligned}$$

The first step by consistency of $\{\hat{\mu}_U\}$ (Definition B.4) since $U^\alpha \subseteq U^{\alpha+1}$ and $\mathrm{pa}(U^{\alpha+1}) \subseteq U^\alpha$, the second by inductive hypothesis, and the third by Definition B.8.

Let α be a limit ordinal. Since $\{\hat{\mu}_{U^\alpha}\}$ is a projective family and $U^\alpha = \bigcup_{\beta<\alpha} U^\beta$, by Lemma G.2 $\hat{\mu}_{U^\alpha} = \lim_{\beta<\alpha} \hat{\mu}_{U^\beta}$. By definition $\mu^\alpha_U = \lim_{\beta<\alpha} \mu^\beta_U$. Then since $\mu^\beta_U = \hat{\mu}_{U^\beta}$ for $\beta < \alpha$ inductively, $\mu^\alpha_U = \hat{\mu}_{U^\alpha}$ as the limit of this sequence is unique. □

F. Proof of Lemma 5.4

Proof. The possible world $\langle U^\sigma, I^\sigma\rangle$ is defined as follows. $U^\sigma = \langle U^\sigma_1, \ldots, U^\sigma_k\rangle$, where $U^\sigma_j = \{c_j : c_j$ is a distinct constant of type τ_j in $\mathcal{M}\} \cup \{u_{\nu,\bar{u},l} \in \mathcal{U}_\mathcal{M} : \nu$ is a number statement of type τ_j and $\sigma(V_\nu[\bar{u}]) \geq l\}$.

I^σ is defined as follows. For each function symbol $f(\bar{x})$ in $\mathcal{M}$, for each tuple $\bar{u}$ of the type of $\bar{x}$ constructed using elements of U^σ, $[f]^\sigma(\bar{u}) = \sigma(V_f[\bar{u}])$. The element $\sigma(V_f[\bar{u}])$ is a member of U^σ because of the last clause in the definition of consistent assignments (Def. 5.3) and the construction of U^σ. □

G. Additional Technical Details

For reasons of space, we present the following without their (straightforward) proofs.

Lemma G.1. *If μ is a measure on $\mathcal{X}$, and is K a kernel from $\mathcal{X}$ to $\mathcal{Y}$, then $(\mu \otimes K) \circ \pi^{\mathcal{X}\times\mathcal{Y}}_{\mathcal{X}} = \mu$.*

Lemma G.2. *A projective sequence of measures has a unique limit.*

Fix an ordinal λ, and suppose $\{U_\alpha \subseteq V : \alpha < \lambda\}$ is an increasing sequence of subsets of V, i.e., such that if $\alpha < \beta < \lambda$ then $U_\alpha \subseteq U_\beta$. Define $U = \bigcup_{\alpha<\lambda} U_\alpha$. Let $\{W_\alpha \subseteq V : \alpha < \lambda\}$ and W be another such sequence, supposing U and W are disjoint.

Definition G.3. $\{K_\alpha : \alpha < \lambda\}$ *is a **projective sequence of kernels** from $\mathcal{X}_{U_\alpha}$ to $\mathcal{X}_{W_\alpha}$ if whenever $\alpha < \beta < \lambda$ we have $\pi^{U_\beta}_{U_\alpha} \circ K_\alpha = K_\beta \circ \pi^{W_\beta}_{W_\alpha}$.*

Definition G.4. *The limit $\lim_{\alpha<\beta} K_\alpha$ of a projective sequence $\{K_\alpha : \alpha < \lambda\}$ of kernels is the unique kernel from $\mathcal{X}_U$ to $\mathcal{X}_W$ such that for all $\alpha < \lambda$ $\pi^U_{U_\alpha} \circ K_\alpha = (\lim_{\alpha<\beta} K_\alpha) \circ \pi^W_{W_\alpha}$.*

Lemma G.5. *A projective sequence of kernels has a unique limit.*

Lemma G.6. *Let $\mathcal{X}_1, \mathcal{Y}_1, \mathcal{X}_2, \mathcal{Y}_2$ be measurable spaces, μ be a measure on $\mathcal{X}_1$, K a kernel from $\mathcal{X}_2$ to $\mathcal{Y}_1$, $f\colon \mathcal{X}_1 \to \mathcal{X}_2$ a measurable function, and $g\colon \mathcal{Y}_1 \to \mathcal{Y}_2$ a measurable function. Then: $(\mu \otimes (f \circ K)) \circ (f \times g) = (\mu \circ f) \otimes (K \circ g)$ where $f \times g$ is the measurable function mapping (x, y) to $((f(x), g(y))$.*

Lemma G.7. *Let ν_α and K_α be as in Lemmas G.2 and G.5. Then $\lim_{\alpha<\lambda}(\nu_\alpha \otimes K_\alpha) = (\lim_{\alpha<\lambda} \nu_\alpha) \otimes (\lim_{\alpha<\lambda} K_\alpha)$.*

Lemma G.8. *μ measure on $\mathcal{X}$, K_1 a kernel from $\mathcal{X}$ to $\mathcal{Y}_1$, K_2 a kernel from $\mathcal{X}$ to $\mathcal{Y}_2$, $\mu \otimes K_1 \otimes (\pi^{\mathcal{X}\times\mathcal{Y}_1}_{\mathcal{X}} \circ K_2) = \mu \otimes \prod_{i=1,2} K_i$. where by abuse of notation $\pi^{\mathcal{X}\times\mathcal{Y}_1}_{\mathcal{X}}$ denotes the projection from $\mathcal{X} \times \mathcal{Y}_1$ to $\mathcal{X}$.*

Lemma G.9. *If $K_{i,j}$ are kernels from $\mathcal{X}$ to $\mathcal{Y}_{i,j}$ then $\prod_i \prod_j K_{i,j} = \prod_{i,j} K_{i,j}$.*

Lemma G.10. *If $f : \mathcal{X}' \to \mathcal{X}$ and K_i are kernels from $\mathcal{X}$ to $\mathcal{Y}_i$ then $f \circ \prod_i K_i = \prod_i f \circ K_i$.*

Lemma G.11. *If K_v for $v \in U$ are kernels from $\mathcal{X}$ to $\mathcal{X}_v$, and $W \subseteq U$ then $\left(\prod_{v\in U} K_v\right) \circ \pi^U_W = \prod_{v\in W} K_v$.*

Lemma G.12. *Let $(X, \mathcal{X})$ be a measurable space, $X, X_1, X_2, \ldots$ an iid random sequence on $\mathcal{X}$, and $w(x)$ be non-negative real-valued function of $(X, \mathcal{X})$. Then $\frac{\sum_{i=1}^n w(X_i) f(X_i)}{\sum_{i=1}^n w(X_i)} \xrightarrow{\text{a.s.}} \frac{\mathbb{E}\, w(X) f(X)}{\mathbb{E}\, w(X)}$.*

Lemma G.13. *For any measurable set E and measurable function $f(x)$: $\frac{\mathbb{E}\, P(E|X) f(X)}{\mathbb{E}\, P(E|X)} = \mathbb{E}[f(X)|E]$.*