Towards Verified Stochastic Variational Inference for Probabilistic Programs

WONYEOL LEE, School of Computing, KAIST, South Korea
HANGYEOL YU, School of Computing, KAIST, South Korea
XAVIER RIVAL, INRIA Paris, Département d’Informatique of ENS, and CNRS/PSL University, France
HONGSEOK YANG, School of Computing, KAIST, South Korea

Probabilistic programming is the idea of writing models from statistics and machine learning using program notations and reasoning about these models using generic inference engines. Recently its combination with deep learning has been explored intensely, which led to the development of so called deep probabilistic programming languages, such as Pyro, Edward and ProbTorch. At the core of this development lie inference engines based on stochastic variational inference algorithms. When asked to find information about the posterior distribution of a model written in such a language, these algorithms convert this posterior-inference query into an optimisation problem and solve it approximately by a form of gradient ascent or descent. In this paper, we analyse one of the most fundamental and versatile variational inference algorithms, called score estimator or REINFORCE, using tools from denotational semantics and program analysis. We formally express what this algorithm does on models denoted by programs, and expose implicit assumptions made by the algorithm on the models. The violation of these assumptions may lead to an undefined optimisation objective or the loss of convergence guarantee of the optimisation process. We then describe rules for proving these assumptions, which can be automated by static program analyses. Some of our rules use nontrivial facts from continuous mathematics, and let us replace requirements about integrals in the assumptions, such as integrability of functions defined in terms of programs’ denotations, by conditions involving differentiation or boundedness, which are much easier to prove automatically (and manually). Following our general methodology, we have developed a static program analysis for the Pyro programming language that aims at discharging the assumption about what we call model-guide support match. Our analysis is applied to the eight representative model-guide pairs from the Pyro webpage, which include sophisticated neural network models such as AIR. It finds a bug in one of these cases, reveals a non-standard use of an inference engine in another, and shows that the assumptions are met in the remaining six cases.

CCS Concepts: • Mathematics of computing → Bayesian computation; Variational methods; • Theory of computation → Probabilistic computation; Denotational semantics; • Software and its engineering → Correctness; Automated static analysis; • Computing methodologies → Machine learning.

Additional Key Words and Phrases: Probabilistic programming, static analysis, semantics, correctness

ACM Reference Format:
1 INTRODUCTION

Probabilistic programming refers to the idea of writing models from statistics and machine learning using program notations and reasoning about these models using generic inference engines. It has been the subject of active research in machine learning and programming languages, because of its potential for enabling scientists and engineers to design and explore sophisticated models easily; when using these languages, they no longer have to worry about developing custom inference engines for their models, a highly-nontrivial task requiring expertise in statistics and machine learning. Several practical probabilistic programming languages now exist, and are used for a wide range of applications [Carpenter et al. 2017; Gehr et al. 2016; Goodman et al. 2008; Gordon et al. 2014; Mansinghka et al. 2014; Minka et al. 2014; Narayanan et al. 2016; Wood et al. 2014].

We consider inference engines that lie at the core of so called deep probabilistic programming languages, such as Pyro [Bingham et al. 2019], Edward [Tran et al. 2018, 2016] and ProbTorch [Siddharth et al. 2017]. These languages let users freely mix deep neural networks with constructs from probabilistic programming, in particular, those for writing Bayesian probabilistic models. In so doing, they facilitate the development of probabilistic deep-network models that may address the problem of measuring the uncertainty in current non-Bayesian deep-network models; a non-Bayesian model may predict that the price of energy goes up and that of a house goes down, but it cannot express, for instance, that the model is confident with the first prediction but not the second.

The primary inference engines for these deep probabilistic programming languages implement stochastic (or black-box) variational inference algorithms. Converting inference problems into optimisation problems is the high-level idea of these algorithms. When asked to find information about the posterior distribution of a model written in such a language, these algorithms convert this question to an optimisation problem and solve the problem approximately by performing a gradient descent or ascent on the optimisation objective. The algorithms work smoothly with gradient-based parameter-learning algorithms for deep neural networks, which is why they form the backbone for deep probabilistic programming languages.

In this paper, we analyse one of the most fundamental and versatile variational inference algorithms, called score estimator or REINFORCE [Paisley et al. 2012; Ranganath et al. 2014; Williams 1992; Wingate and Weber 2013], using tools from denotational semantics and program analysis [Cousot and Cousot 1977, 1979, 1992]. We formally express what this algorithm does on models denoted by probabilistic programs, and expose implicit assumptions made by the algorithm on the models. The violation of these assumptions can lead to undefined optimisation objective or the loss of convergence guarantee of the optimisation process. We then describe rules for proving these assumptions, which can be automated by static program analyses. Some of our rules use nontrivial facts from continuous mathematics, and let us replace requirements about integrals in the assumptions, such as integrability of functions defined in terms of programs’ denotations, by the conditions involving differentiation or boundedness, which are much easier to prove automatically (and manually) than the original requirements.

Following our general methodology, we have developed a static program analysis for the Pyro programming language that can discharge one assumption of the inference algorithm about so
called model-guide pairs. In Pyro and other deep probabilistic programming languages, a program denoting a model typically comes with a companion program, called guide, decoder, or inference network. This companion, which we call guide, helps the inference algorithm to find a good approximation to what the model ultimately denotes under a given dataset (i.e., the posterior distribution of the model under the dataset); the algorithm uses the guide to fix the search space of approximations, and solves an optimisation problem defined on that space. A model and a guide should satisfy an important correspondence property, which says that they should use the same sets of random variables, and for any such random variable, if the probability of the variable having a particular value is zero in the model, it should also be zero in the guide. If the property is violated, the inference algorithm may attempt to solve an optimisation problem with undefined optimisation objective and return parameter values that do not make any sense. Our static analysis checks this correspondence property for Pyro programs. When applied to eight representative model-guide pairs from the Pyro webpage, which include sophisticated neural network models such as Attend-Infer-Repeat (AIR), the analysis found a bug in one of these cases, revealed a non-standard use of the inference algorithm in another, and proved that the property holds in the remaining six cases.

Another motivation for this paper is to demonstrate an opportunity for programming languages and verification research to have an impact on the advances of machine learning and AI technologies. One popular question is: what properties should we verify on machine-learning programs? Multiple answers have been proposed, which led to excellent research results, such as those on robustness of neural networks [Mirman et al. 2018]. But most of the existing research focuses on the final outcome of machine learning algorithms, not the process of applying these algorithms. One of our main objectives is to show that the process often relies on multiple assumptions on models and finding automatic ways for discharging these assumptions can be another way of making PL and verification techniques contribute. While our suggested solutions are not complete, they are intended to show the richness of this type of problems in terms of theory and practice.

We summarise the contributions of the paper:

- We formally express the behaviour of the most fundamental variational inference algorithm on probabilistic programs using denotational semantics, and identify requirements on program denotations that are needed for this algorithm to work correctly.
- We describe conditions that imply the identified requirements but are easier to prove. The sufficiency of the conditions relies on nontrivial results from continuous mathematics. We sketch a recipe for building program analyses for checking these conditions automatically.
- We present a static analysis for the Pyro language that checks the correspondence requirement of model-guide pairs. The analysis is based on our recipe, but extends it significantly to address challenges for dealing with features of the real-world language. Our analysis has successfully verified 6 representative Pyro model-guide examples, and found a bug in one example.

The extended version of the paper can be found in [Lee et al. 2019].

2 VARIATIONAL INFERENCE AND VERIFICATION CHALLENGES BY EXAMPLES

We start by explaining informally the idea of stochastic variational inference (in short SVI), one fundamental SVI algorithm, and the verification challenges that arise when we use this algorithm.

2.1 Stochastic Variational Inference

In a probabilistic programming language, we specify a model by a program. The program `model()` in Figure 1(a) is an example. It describes a joint probability density \( p(v, obs) \) on two real-valued random variables \( v \) and \( obs \). The value of the former is not observed, while the latter is observed to have the value 0. Finding out the value of \( v \) is the objective of writing this model. The joint
1 # define model and guide
2 def model():
3     v = pyro.sample("v", Normal(0., 5.))
4     if (v > 0):
5         pyro.sample("obs", Normal(1., 1.), obs=0.)
6     else:
7         pyro.sample("obs", Normal(-2., 1.), obs=0.)
8 def guide():
9     theta = pyro.param("theta", 3.)
10     theta = pyro.param("theta").item()
11     v = pyro.sample("v", Normal(theta, 1.))
12 # perform stochastic variational inference
13 svi = SVI(model, guide, Adam({"lr": 1.0e-2}),
14     loss=Trace_ELBO())
15 for step in range(2000):
16     svi.step()
17     loss=Trace_ELBO())
18 print("trained theta =",
19     pyro.param("theta").item())

(a) Example model-guide pair for stochastic variational inference in Pyro.

(b) Probability densities of the model and the guide as a function of \( v \in \mathbb{R} \).

(c) KL divergence from the guide to the model (plus \( \log p(\text{obs}=0) \)) as a function of \( \theta \in \mathbb{R} \).

Fig. 1. Example of performing stochastic variational inference.

density \( p(v, \text{obs}) \) is expressed in terms of prior \( p(v) \) and likelihood \( p(\text{obs}|v) \) in the program. The prior \( p(v) \) of \( v \) is the normal distribution with mean 0 and standard deviation 5, and it expresses the belief about the possible value of \( v \) before any observation. The likelihood \( p(\text{obs}|v) \) is a normal distribution whose mean and standard deviation are either (1, 1) or (−2, 1) depending on the sign of the value of \( v \). The purpose of most inference algorithms is to compute exactly or approximately the posterior density given a prior and a likelihood. In our example, the posterior \( p(v|\text{obs}=0) \) is:

\[
p(v|\text{obs}=0) = \frac{p(v, \text{obs}=0)}{\int dv \ p(v, \text{obs}=0)} = \frac{p(v) \cdot p(\text{obs}=0|v)}{p(\text{obs}=0)}.
\]

Intuitively, the posterior expresses an updated belief on \( v \) upon observing \( \text{obs} = 0 \). The dashed blue and solid orange lines in Figure 1(b) show the prior and posterior densities, respectively. Note that the density of a positive \( v \) in the prior went up in the posterior. This is because when \( v > 0 \), the mean of \( p(\text{obs}|v) \) is 1, a value closer to the observed value 0 than the alternative −2 for the mean.

SVI algorithms approach the posterior inference problem from the optimisation angle. They consider a collection of approximating distributions to a target posterior, formulate the problem of finding a good approximation in the collection as an optimisation problem, and solve the optimisation problem. The solution becomes the result of those algorithms. In Pyro, the users specify such a collection by a single parameterised program called guide; the collection can be generated by instantiating the parameters with different values. The program guide() in Figure 1(a) is an example. It has a real-valued parameter \( \theta \) (written as theta in the program), and states that the probability density \( q_\theta(v) \) of \( v \) is the normal distribution with unknown mean \( \theta \) and standard deviation 1. The lines 13–17 in the figure show how to apply a standard SVI engine of Pyro (called Trace_ELBO) to find a good \( \theta \). They instruct the engine to solve the following optimisation problem:

\[
\arg\min_\theta \text{KL}(q_\theta(v)||p(v|\text{obs}=0)), \quad \text{where} \quad \text{KL}(q_\theta(v)||p(v|\text{obs}=0)) \triangleq \mathbb{E}_{q_\theta(v)} \left[ \log \frac{q_\theta(v)}{p(v|\text{obs}=0)} \right].
\]
The optimisation objective \( \nabla_\theta \text{KL}(q_\theta(v)||p(v|\text{obs}=0)) \) is the \( \text{KL divergence} \) from \( q_\theta(v) \) to \( p(v|\text{obs}=0) \), and measures the similarity between the two densities, having a small value when the densities are similar. The KL divergence is drawn in Figure 1(c) as a function of \( \theta \), and the dotted green line in Figure 1(b) draws the density \( q_\theta \) at the optimum \( \theta \). Note that the mean of this distribution is biased toward the positive side, which reflects the fact that the property \( v > 0 \) has a higher probability than its negation \( v \leq 0 \) in the posterior distribution.

One of the most fundamental and versatile algorithms for SVI is score estimator (also called REINFORCE). It repeatedly improves \( \theta \) in two steps. First, it estimates the gradient of the optimisation objective with samples from the current \( q_{\theta_n} \):

\[
\nabla_\theta \text{KL}(q_\theta(v)||p(v|\text{obs}=0)) \bigg|_{\theta=\theta_n} \approx \frac{1}{N} \sum_{i=1}^{N} (\nabla_\theta \log q_{\theta_n}(v_i)) \cdot \log \frac{q_{\theta_n}(v_i)}{p(v_i, \text{obs}=0)}
\]

where \( v_1, \ldots, v_N \) are independent samples from the distribution \( q_{\theta_n} \). Then, the algorithm updates \( \theta \) with the estimated gradient (the specific learning rate 0.01 is chosen to improve readability):

\[
\theta_{n+1} \leftarrow \theta_n - 0.01 \times \frac{1}{N} \sum_{i=1}^{N} (\nabla_\theta \log q_{\theta_n}(v_i)) \cdot \log \frac{q_{\theta_n}(v_i)}{p(v_i, \text{obs}=0)}.
\]

When the learning rate 0.01 is adjusted according to a known scheme, the algorithm is guaranteed to converge to a local optimum (in many cases) because its gradient estimate satisfies the following unbiasedness property (in those cases):

\[
\nabla_\theta \text{KL}(q_\theta(v)||p(v|\text{obs}=0)) \bigg|_{\theta=\theta_n} = \mathbb{E} \left[ \frac{1}{N} \sum_{i=1}^{N} (\nabla_\theta \log q_{\theta_n}(v_i)) \cdot \log \frac{q_{\theta_n}(v_i)}{p(v_i, \text{obs}=0)} \right]
\]  

(1)

where the expectation is taken over the independent samples \( v_1, \ldots, v_N \) from \( q_{\theta_n} \).

### 2.2 Verification Challenges

We now give two example model-guide pairs that illustrate verification challenges related to SVI.

The first example appears in Figure 2(a). It is the Bayesian regression example from the Pyro webpage (this example is among the benchmarks used in §8), which solves the problem of finding a line that interpolates a given set of points in \( \mathbb{R}^2 \).

The problem with this example is that the KL divergence of its model-guide pair, the main optimisation objective in SVI, is undefined. The model and guide in the figure use the random variable \( \sigma \), but they use different non-zero-probability regions, called supports, for it. In the model, the support is \([0, 10]\), while that in the guide is \( \mathbb{R} \). But the KL divergence from a guide to a model is defined only if for every random variable, its support in the guide is included in that in the model. We point out that this support mismatch was found by our static analyser explained in §8.

Figures 2(b) and 2(c) show two attempts to resolve the undefined-KL issue. To fix the issue, we change the distribution of \( \sigma \) in the model in (b), and in the guide in (c). These revisions remove the problem about the support of \( \sigma \), but do not eliminate that of the undefined KL. In both (b) and (c), the KL divergence is \( \infty \). This happens mainly because \( \sigma \) can be arbitrarily close to 0 in the guide in both cases, which makes integrand in the definition of the KL divergence diverge to \( \infty \).

An SVI-specific verification challenge related to this example is how to prove the well-definedness of the KL divergence and more generally the optimisation objective of an SVI algorithm. In §6.2, we provide a partial answer to the question. We give a condition for ensuring the well-definedness of the KL divergence. Our condition is more automation-friendly than the definition of KL, because it does not impose the difficult-to-check integrability requirement present in the definition of KL.
def model(...):
    ...  # same as model in (a)
    sigma = pyro.sample("sigma",
        Normal(5., 5.))
    ...  # same as guide in (a)
    v = pyro.sample("v",
        Uniform(theta-1., theta+1.))

def guide(...):
    theta = pyro.param("theta", 3.)
    v = pyro.sample("v",
        Uniform(theta-1., theta+1.))
Towards Verified Stochastic Variational Inference for Probabilistic Programs 16:7

To sum up, the second example shows that even if the KL divergence is defined, its gradient is sometimes undefined, and also that even if both the KL divergence and its gradient are defined, the sample-based estimate of the gradient in a standard SVI algorithm may be biased—this means that the equation similar to (1) does not hold and an SVI algorithm is no longer guaranteed to converge to a local optimum. Proving that these failure cases do not arise is another SVI-specific verification challenge. In §6.3, we give another example of similar flavour, and provide an automation-friendly condition that ensures the existence of the KL divergence and its gradient as well as the unbiasedness of the gradient estimate of the score estimator.

We conclude the section by emphasising that the aforementioned issues could have a large impact on the results of SVI. For instance, running the Bayesian regression example in Figure 2(a) in Pyro after a small change (0.4 instead of 0.05 in line 14) sometimes results in a crash. Also, running its revised version in Figure 2(c) leads to the complete failure of an optimiser and produces meaningless results. These observations strengthen the importance of resolving the verification challenges presented above.4

3 REVIEW OF MEASURE THEORY AND NOTATIONS

A σ-algebra Σ on a set X is a collection of subsets of X such that (i) X ∈ Σ; (ii) A0 ∪ A1 ∈ Σ and X \ A0 ∈ Σ for all A0, A1 ∈ Σ; (iii) ∪n A_n ∈ Σ when all subsets A_n are in Σ. An equivalent but easier-to-remember characterisation is that Σ is closed under boolean operations and countable union and intersection. We call the pair (X, Σ) of a set and a σ-algebra measurable space, and subsets in Σ measurable. A function f from a measurable space (X, Σ) to another measurable space (X’, Σ’) is measurable if f⁻¹(B) ∈ Σ for all B ∈ Σ’.

An example of measurable space is the n-dimensional Euclidean space R^n with the Borel σ-algebra B = σ(\{(−∞, r] \times \cdots \times (−∞, r_n) \mid r \in \mathbb{R}^n\}), where σ is the closure operator that converts a collection of subsets of R^n into the smallest σ-algebra containing the collection. Subsets X of R^n, such as [0, ∞)^n, form measurable spaces with the σ-algebra \{X \cap A \mid A \in B\}. Another example is a set X with the so called discrete σ-algebra on X that consists of all subsets of X.

A measure µ on a measurable space (X, Σ) is a function from Σ to [0, ∞] such that µ(∅) = 0 and µ satisfies the countable additivity condition: µ(∪n B_n) = ∑n µ(B_n) for any countable family of disjoint measurable subsets B_n. A well-known example is the Lebesgue measure λ^n on R^n which maps each measurable subset of R^n to its volume in the usual sense.5 When µ(X) ≤ 1, we call µ subprobability measure. If µ(X) = 1, we may drop “sub”, and call µ probability measure.

The Lebesgue integral ∫ is a partial operator that maps a measure µ on (X, Σ) and a real-valued measurable function on the same space (X, Σ) to a real number. It is denoted by ∫ f(x) dx. To follow the paper, it is enough to know that this integral generalises the usual Riemann integral from calculus.6 For a measure ν on (X, Σ), if ν(A) = ∫ f(x) dx · 1_{x \in A} for non-negative f, we say that f is the density of ν with respect to µ and call µ reference measure.

In the paper, we use a few well-known methods for building measurable spaces.

The first method applies when we are given a set X and a collection of functions \{f_i : X → X_i \mid i \in I\} to measurable spaces (X_i, Σ_i). The method is to equip X with the smallest σ-algebra Σ making all f_i’s measurable: Σ = σ(\{f_i⁻¹(B) \mid i \in I, B ∈ Σ_i\}).

4The aforementioned issues do not always cause problems in inference results (e.g., Figures 2(a) and 2(c) mostly give reasonable results), because the random seeds and the initial values for SVI could be set well so that the probability of SVI going wrong becomes low. We emphasise, however, that the probability is non-zero.

5The Lebesgue measure λ^n is the unique measure on R^n that sets the volume of the unit cube (0, 1)^n to 1 and is translation invariant: for all measurable subsets A and r ∈ R^n, λ^n(A) = λ^n((r’−r \mid r’ ∈ A)).

6Another useful fact is that when f is non-negative, ∫ f(x) dx = sup Σ_i(∫ f(x) · 1_{x \in A_i} dx) + µ(A_i) where the supremum is taken with respect to all finite partitions \{A_i\}_i of X into measurable subsets.
The second relies on two constructions for sets, i.e., product and disjoint union. Suppose that we are given measurable spaces \((X_i, \Sigma_i)\) for all \(i \in I\). We define a product measurable space that has \(\prod_{i \in I} X_i\) as its underlying set and the following product \(\sigma\)-algebra \(\bigotimes_{i \in I} \Sigma_i\) as its \(\sigma\)-algebra:
\[
\bigotimes_{i \in I} \Sigma_i \triangleq \sigma\left(\left\{ \left. \prod_i A_i \right| \text{there is a finite } I_0 \subseteq I \text{ such that } (\forall j \in I \setminus I_0, A_j = X_j) \land (\forall i \in I_0, A_i \in \Sigma_i) \right\}\right).
\]

The construction of the product \(\sigma\)-algebra can be viewed as a special case of the first method where we consider the smallest \(\sigma\)-algebra on \(\prod_{i \in I} X_i\) that makes every projection map to \(X_i\) measurable.

Throughout the paper, we take the convention that the set \(\mathbb{N}\) of natural numbers includes 0.

The third method builds a measurable space out of measures or a certain type of measures, such as subprobability measures. For a measurable space \((X, \Sigma)\), we form a measurable space with measures. The underlying set \(\text{Mea}(X)\) and \(\sigma\)-algebra \(\Sigma_{\text{Mea}(X)}\) of the space are defined by
\[
\text{Mea}(X) \triangleq \{ \mu \mid \mu \text{ is a measure on } (X, \Sigma) \}, \quad \Sigma_{\text{Mea}(X)} \triangleq \sigma\left(\left\{ \{ \mu \mid \mu(A) \leq r \} \mid A \in \Sigma_1, r \in \mathbb{R} \right\}\right).
\]

The difficult part to grasp is \(\Sigma_{\text{Mea}(X)}\). Once again, a good approach for understanding it is to realise that \(\Sigma_{\text{Mea}(X)}\) is the smallest \(\sigma\)-algebra that makes the function \(\mu \mapsto \mu(A)\) from \(\text{Mea}(X)\) to \(\mathbb{R}\) measurable for all measurable subsets \(A \in \Sigma\). This measurable space gives rise to a variety of measurable spaces, each having a subset \(M\) of \(\text{Mea}(X)\) as its underlying set and the induced \(\sigma\)-algebra \(\Sigma_M = \{ A \cap M \mid A \in \Sigma_{\text{Mea}(X)} \}\).

A measurable function \(f\) from \((X, \Sigma)\) to \((\text{Mea}(Y), \Sigma_{\text{Mea}(Y)})\) is called kernel. If \(f(x)\) is a subprobability measure (i.e., \(f(x) \in \text{Sp}(Y)\)) for all \(x\), we say that \(f\) is a subprobability kernel. In addition, if \(f(x)\) is a probability measure (i.e., \(f(x) \in \text{Pr}(Y)\)) for all \(x\), we call \(f\) probability kernel. A good heuristic is to view a probability kernel as a random function and a subprobability kernel as a random partial function. We use well-known facts that a function \(f : X \to \text{Mea}(Y)\) is a subprobability kernel if and only if it is a measurable map from \((X, \Sigma)\) to \((\text{Sp}(Y), \Sigma_{\text{Sp}(Y)})\), and that similarly a function \(f\) is a probability kernel if and only if it is a measurable function from \((X, \Sigma)\) to \((\text{Pr}(Y), \Sigma_{\text{Pr}(Y)})\).

We use a few popular operators for constructing measures throughout the paper. We say that a measure \(\mu\) on a measurable space \((X, \Sigma)\) is finite if \(\mu(X) < \infty\), and \(\sigma\)-finite if there is a countable partition of \(X\) into measurable subsets \(X_n\) such that \(\mu(X_n) < \infty\) for every \(n\). Given a finite or countable family of \(\sigma\)-finite measures \(\{\mu_i\}_{i \in I}\) on measurable spaces \((X_i, \Sigma_i)\)’s, the product measure of \(\mu_i\)’s, denoted \(\bigotimes_{i \in I} \mu_i\), is the unique measure on \((\prod_{i \in I} X_i, \bigotimes_{i \in I} \Sigma_i)\) such that \((\bigotimes_{i \in I} \mu_i)(\prod_{i \in I} A_i) = \prod_{i \in I} \mu_i(A_i)\) for all measurable subsets \(A_i\) of \(X_i\). Given a finite or countable family of measures \(\{\mu_i\}_{i \in I}\) on disjoint measurable spaces \((X_i, \Sigma_i)\)’s, the sum measure of \(\mu_i\)’s, denoted \(\bigoplus_{i \in I} \mu_i\), is the unique measure on \((\bigoplus_{i \in I} X_i, \bigoplus_{i \in I} \Sigma_i)\) such that \((\bigoplus_{i \in I} \mu_i)(\bigcup_{i \in I} A_i) = \bigcup_{i \in I} \mu_i(A_i)\).

Throughout the paper, we take the convention that the set \(\mathbb{N}\) of natural numbers includes 0. For all positive integers \(n\), we write \([n]\) to mean the set \(\{1, 2, \ldots, n\}\).

### 4 SIMPLE PROBABILISTIC PROGRAMMING LANGUAGE

In this section, we describe the syntax and semantics of a simple probabilistic programming language, which we use to present the theoretical results of the paper. The measure semantics in
The program specifies a model with one random variable \( \text{score} \). The next score statement refines this belief with the normal distribution with mean \( E_1 \) and standard deviation \( E_2 \) and names the sample with the string \( S \). The next score statement refines this belief with the normal distribution with mean \( E_1 \) and standard deviation \( E_2 \) and the value of this sample is observed to be \( E_0 \). It lets the programmers express information about observed data inside programs. Operationally, this construct can be understood as an instruction for updating a global variable that stores the so-called importance score of the execution. The importance score quantitatively records how well the random choices in the current execution match the observations, and the score statement updates this score by multiplying it with the density at \( E_0 \) of the appropriate normal distribution.

Consider the following program:

\[
x := \text{sample}_\text{norm}(\text{"a"}, 0.0, 5.0); \text{score}_\text{norm}(3.0, x, 1.0).
\]

The program specifies a model with one random variable "a". Using a relatively flat normal distribution, the program specifies a prior belief that the value of the random variable "a" is likely to be close to 0.0 and lie between \(-2 \times 5.0\) and \(2 \times 5.0\). The next score statement refines this belief with one data point 3.0, which is a noisy observation of the value of "a" (bound to \( x \)). The parameters to the normal density in the statement express that the noise is relatively small, between \(-2 \times 1.0\) and \(2 \times 1.0\). Getting the refined belief, called posterior distribution, is the reason that a data scientist writes a model like this program. It is done by an inference algorithm of the language.

Permitting only the normal distribution does not limit the type of models expressible in the language. Every distribution can be obtained by transforming the standard normal distribution.\(^7\)

\(^7\)Here we consider only Borel spaces. Although using only the normal distribution does not affect the expressiveness, it has an impact on stochastic variational inference to be discussed later, because it requires a guide to use only normal distributions. But the impact is not significant, because most well-known approaches for creating guides from the machine-learning literature (such as extensions of variational autoencoder) use normal distributions only, or can be made to do so easily.
4.2 Measure Semantics

The denotational semantics of the language just presented is mostly standard, but employs some twists to address the features for probabilistic programming [Staton et al. 2016].

Here is a short high-level overview of the measure semantics. Our semantics defines multiple measurable spaces, such as Store and State, that hold mathematical counterparts to the usual actors in computation, such as program stores (i.e., mappings from variables to values) and states (which consist of a store and further components). Then, the semantics interprets expressions $E, B, S$ and commands $C$ as measurable functions of the following types:

$$
\|E\| : \text{Store} \rightarrow \mathbb{R}, \quad \|B\| : \text{Store} \rightarrow \mathcal{B}, \quad \|S\| : \text{Store} \rightarrow \text{Str}, \quad \|C\| : \text{State} \rightarrow \text{Sp(State } \times [0, \infty)).
$$

Here $\mathbb{R}$ is the measurable space of reals with the Borel $\sigma$-algebra, and $\mathcal{B} \text{ and } \text{Str}$ are discrete measurable spaces of booleans and strings. Store and State are measurable spaces for stores (i.e., maps from variables to values) and states which consist of a store and a part for recording information about sampled random variables. Note that the target measurable space of commands is built by first taking the product of measurable spaces State and $[0, \infty)$ and then forming a space out of subprobability measures on State $\times [0, \infty)$. This construction indicates that commands denote probabilistic computations, and the result of each such computation consists of an output state and a score which expresses how well the computation matches observations expressed with the score statements in the command $C$. Some of the possible outcomes of the computation may lead to non-termination or an error, and these abnormal outcomes are not accounted for by the semantics, which is why $\|C\|(\sigma)$ for a state $\sigma \in \Sigma$ is a subprobability distribution. The semantics of expressions is much simpler. It just says that expressions do not involve any probabilistic computations, so that they denote deterministic measurable functions.

We now explain how this high-level idea gets implemented in our semantics. Let $\text{Var}$ be a countably infinite set of variables. Our semantics uses the following sets:

$$
\text{Stores} \quad s \in \text{Store} \triangleq [\text{Var} \rightarrow \mathbb{R}] \quad \text{(which is isomorphic to } \prod_{x \in \text{Var}} \mathbb{R})
$$

$$
\text{Random databases} \quad r \in \text{RDB} \triangleq \bigcup_{K \subseteq \text{fin Str}} [K \rightarrow \mathbb{R}] \quad \text{(which is isomorphic to } \bigcup_{K \subseteq \text{fin Str}} \prod_{\alpha \in K} \mathbb{R})
$$

$$
\text{States} \quad \sigma \in \text{State} \triangleq \text{Store} \times \text{RDB},
$$

where $[X \rightarrow Y]$ denotes the set of all functions from $X$ to $Y$. A state $\sigma$ consists of a store $s$ and a random database $r$. The former fixes the values of variables, and the latter records the name (given as a string) and the value of each sampled random variable. The domain of $r$ is the names of all the sampled random variables. By insisting that $r$ should be a map, the semantics asserts that no two random variables have the same name. For each state $\sigma$, we write $\sigma_s$ and $\sigma_\alpha$ for its store and random database components, respectively. Also, for a variable $x$, a string $\alpha$ and a value $v$, we write $\sigma[x \mapsto v]$ and $\sigma[\alpha \mapsto v]$ to mean $(\sigma_s[x \mapsto v], \sigma_\alpha)$ and $(\sigma_s, \sigma_\alpha[\alpha \mapsto v])$.

We equip all of these sets with $\sigma$-algebras and turn them to measurable spaces in a standard way. Note that we constructed the sets from $\mathbb{R}$ by repeatedly applying the product and disjoint-union operators. We equip $\mathbb{R}$ with the usual Borel $\sigma$-algebra. Then, we parallel each usage of the product and the disjoint-union operators on sets with that of the corresponding operators on $\sigma$-algebras. This gives the $\sigma$-algebras for all the sets defined above. Although absent from the above definition, the measurable spaces $\mathcal{B}$ and $\text{Str}$ equipped with discrete $\sigma$-algebras are also used in our semantics.

We interpret expressions $E, B, S$ as measurable functions $\|E\| : \text{Store} \rightarrow \mathbb{R}, \|B\| : \text{Store} \rightarrow \mathcal{B}$, and $\|S\| : \text{Store} \rightarrow \text{Str}$, under the assumption that the semantics of primitive real-valued $f$ of
We write a probabilistic program to answer queries about the model and data that it describes. Among such queries, posterior inference is one of the most important and popular. Let \( \sigma_1 = (s_1, []) \) be the initial state that consists of some fixed store and the empty random database. In our setting, posterior inference amounts to finding information about the following probability measure with mean \( \mu \) and standard deviation \( \sigma \).

\[
\text{Posterior Inference and Density Semantics}
\]

**Theorem 4.3.** For every command \( C \), their semantics \( [C] \in \mathcal{K} \) of commands \( C \). Here \( N(\nu; \mu, \sigma) \) is the density of the normal distribution with mean \( \mu \) and standard deviation \( \sigma \).

\[
\text{arity } n \text{ and string-valued } g \text{ of arity } (m, l) \text{ are given by measurable functions } [f] : \mathbb{R}^n \rightarrow \mathbb{R} \text{ and } [g] : \text{Str}^m \times \mathbb{R}^l \rightarrow \text{Str}. \text{ It is standard, and we describe it only for some sample cases of } E \text{ and } S:\n\]

\[
\begin{align*}
[x]s & \triangleq s(x), & [f(E_0, \ldots, E_{n-1})]s & \triangleq [f(E_0)s, \ldots, E_{n-1}])s, \\
\alpha s & \triangleq \alpha, & [g(S_0, \ldots, S_{m-1}, E_0, \ldots, E_{l-1})]s & \triangleq [g(S_0)s, \ldots, S_{m-1}s, E_0s, \ldots, E_{l-1}])s.
\end{align*}
\]

**Lemma 4.1.** For all expressions \( E, B, \) and \( S \), their semantics \([E], [B]\) and \([S]\) are measurable functions from \( \text{Store} \rightarrow \mathbb{R}, \mathbb{R} \) and \( \text{Str} \) respectively.

We interpret commands \( C \) as measurable functions from \( \text{State} \rightarrow \text{Sp} \text{(State } \times [0, \infty)) \), i.e., subprobability kernels from \( \text{State} \rightarrow \text{State} \times [0, \infty) \). Let \( \mathcal{K} \) be the set of subprobability kernels from \( \text{State} \rightarrow \text{State} \times [0, \infty) \), and \( \Sigma_{\text{State} \times [0, \infty]} \) be the \( \sigma \)-algebra of the product space \( \text{State} \times [0, \infty) \). We equip \( \mathcal{K} \) with the partial order \( \sqsubseteq \); for all \( \kappa, \kappa' \in \mathcal{K} \), \( \kappa \sqsubseteq \kappa' \) if and only if \( \kappa(\sigma)(A) \leq \kappa'(\sigma)(A) \) for all \( \sigma \in \text{State} \) and \( A \in \Sigma_{\text{State} \times [0, \infty)} \). The next lemma is a minor adaptation of a known result [Kozen 1981].

**Lemma 4.2.** \( (\mathcal{K}, \sqsubseteq) \) is an \( \omega \)-complete partial order with the least element \( \bot \triangleq (\lambda \sigma. \lambda A. 0) \).

The measure semantics of a command \( C \) is defined in Figure 5. The interpretation of the loop is the least fixed point of the function \( F \) on the \( \omega \)-complete partial order \( \mathcal{K} \). The function \( F \) is continuous, so the least fixed point is obtained by the \( \omega \)-limit of the sequence \( \{F^n(\bot)\}_n \). In the definition of \( F \), the argument \( \kappa \) is a subprobability kernel, and it represents the computation after the first iteration of the loop. The semantics of the sample statement uses an indicator function to exclude erroneous executions where the argument \( S \) denotes a name already used by some previous random variable, or the standard deviation \( E_2\sigma \) is not positive. When this check passes, it distills \( A \) to a property on the value of \( x \) and computes the probability of the property using the normal distribution with mean \( E_1\sigma \) and standard deviation \( E_2\sigma \).

**Theorem 4.3.** For every command \( C \), its interpretation \([C]\) is well-defined and belongs to \( \mathcal{K} \).

### 4.3 Posterior Inference and Density Semantics

We write a probabilistic program to answer queries about the model and data that it describes. Among such queries, posterior inference is one of the most important and popular. Let \( \sigma_1 = (s_1, []) \) be the initial state that consists of some fixed store and the empty random database. In our setting, posterior inference amounts to finding information about the following probability measure \( \text{Pr}(C, \cdot) \) for a command \( C \). For a measurable \( A \subseteq \text{RDB} \),

\[
\begin{align*}
\text{Mea}(C, A) & \triangleq \int [C](\sigma)(d(\sigma, w))(w \cdot 1_{[\sigma \in \text{StoreX} \cup A]}), & Z_C & \triangleq \text{Mea}(C, \text{RDB}), & \text{Pr}(C, A) & \triangleq \frac{\text{Mea}(C, A)}{Z_C}.
\end{align*}
\]
The probability measure $\Pr(C, \cdot)$ is called the posterior distribution of $C$, and $\Mea(C, \cdot)$ the unnormalised posterior distribution of $C$ (in $\Mea(C, \cdot)$ and $\Pr(C, \cdot)$, we elide the dependency on $s_I$ to avoid clutter). Finding information about the former is the goal of most inference engines of existing probabilistic programming languages. Of course, $\Pr(C, \cdot)$ is not defined when the normalising constant $Z_C$ is infinite or zero. The inference engines regard such a case as an error that a programmer should avoid, and consider only $C$ without those errors.

Most algorithms for posterior inference use the density semantics of commands. They implicitly pick measures on some measurable spaces used in the semantics. These measures are called reference measures, and constructed out of Lebesgue and counting measures [Bhat et al. 2012, 2013; Hur et al. 2015]. Then, the algorithms interpret commands as density functions with respect to these measures. One outcome of this density semantics is that the unnormalised posterior distribution $\Mea(C, \cdot)$ of a command $C$ has a measurable function $f : RDB \rightarrow [0, \infty)$ such that $\Mea(C, A) = \int \rho(dr) (1_{[r \in A]} \cdot f(r))$, where $\rho$ is a reference measure on $RDB$. Function $f$ is called density of $\Mea(C, \cdot)$ with respect to $\rho$.

In the rest of this subsection, we define the meanings of commands using density functions. To do this, we need to set up some preliminary definitions.

First, we look at a predicate and an operator for random databases, which are about the possibility and the very act of merging two databases. For $r, r' \in RDB$, define the predicate $r \# r'$ by:

$$r \# r' \iff \dom(r) \cap \dom(r') = \emptyset.$$  

When $r \# r'$, let $r \uplus r'$ be the random database obtained by merging $r$ and $r'$:

$$\dom(r \uplus r') \triangleq \dom(r) \cup \dom(r'); \quad (r \uplus r')(\alpha) \triangleq \text{if } \alpha \in \dom(r) \text{ then } r(\alpha) \text{ else } r'(\alpha).$$

**Lemma 4.4.** For every measurable $h : RDB \times RDB \times RDB \rightarrow \mathbb{R}$, the function $(r, r') \mapsto 1_{[r \# r']} \times h(r, r', r \uplus r')$ from $RDB \times RDB$ to $\mathbb{R}$ is measurable.

Second, we define a reference measure $\rho$ on $RDB$:

$$\rho(R) \triangleq \sum_{K \subseteq \text{str}} \left( \bigotimes_{\alpha \in K} \lambda \right)(R \cap [K \rightarrow \mathbb{R}]),$$

where $\lambda$ is the Lebesgue measure on $\mathbb{R}$. As explained in the preliminary section, the symbol $\otimes$ here represents the operator for constructing a product measure. In particular, $\bigotimes_{\alpha \in K} \lambda$ refers to the product of the $|K|$ copies of the Lebesgue measure $\lambda$ on $\mathbb{R}$. In the above definition, we view functions in $[K \rightarrow \mathbb{R}]$ as tuples with $|K|$ real components and measure sets of such functions using the product measure $\bigotimes_{\alpha \in K} \lambda$. When $K$ is the empty set, $\bigotimes_{\alpha \in K} \lambda$ is the nullary-product measure on $\{\emptyset\}$, which assigns $1$ to $\{\emptyset\}$ and $0$ to the empty set.

The measure $\rho$ computes the size of each measurable subset $R \subseteq RDB$ in three steps. It splits a given $R$ into groups based on the domains of elements in $R$. Then, it computes the size of each group separately, using the product of the Lebesgue measure. Finally, it adds the computed sizes. The measure $\rho$ is not finite, but it satisfies the $\sigma$-finiteness condition,\(^\star\) the next best property.

Third, we define a partially-ordered set $\mathcal{D}$ with certain measurable functions. We say that a function $g : \text{Store} \times RDB \rightarrow \{\bot\} \cup (\text{Store} \times RDB \times [0, \infty) \times [0, \infty))$ uses random databases locally or is local if for all $s, s' \in \text{Store}, r, r' \in RDB$, and $w', p' \in [0, \infty),$

$$g(s, r) = (s', r', w', p') \implies (\exists r'' : r'' \# r'') \wedge r = r' \uplus r'' \wedge g(s, r'') = (s', [\emptyset], w', p')) \wedge (\forall r''' : r \# r''') \implies g(s, r \uplus r''') = (s', r', r''', w', p').$$

\(^\star\)The condition lets us use Fubini theorem when showing the well-formedness of the density semantics in this subsection and relating this semantics with the measure semantics in the previous subsection.

The condition describes the way that $g$ uses a given random database $r$, which plays the role of a bank of random seeds (that $g$ may partially consume as it needs random values). Some part $r''$ of $r$ may be consumed by $g$, but the unconsumed part $r'$ of $r$ does not change and is returned in the output. Also, the behaviour of $g$ does not depend on the unconsumed $r'$. We define the set $\mathcal{D}$ by

$$\mathcal{D} = \{ g : \text{Store} \times RDB \rightarrow \{ \bot \} \cup (\text{Store} \times RDB \times [0, \infty] \times [0, \infty]) \mid g \text{ is measurable and local} \}. \quad (3)$$

Here we view $\{ \bot \}$ and $[0, \infty]$ as measurable spaces equipped with discrete and Borel $\sigma$-algebras. Also, we regard $\text{Store} \times RDB$ and $\{ \bot \} \cup (\text{Store} \times RDB \times [0, \infty] \times [0, \infty])$ as measurable spaces constructed by the product and disjoint-union operators on measurable spaces, as explained in §3.

The locality in the definition of $\mathcal{D}$ formalises expected behaviours of commands. In fact, as we will show shortly, it is satisfied by all commands in our density semantics. This property plays an important role when we establish the connection between the density semantics in this subsection and the standard measure semantics in §4.2.

The functions in $\mathcal{D}$ are ordered pointwise: for all $g, g' \in \mathcal{D}$,

$$g \sqsubseteq g' \iff \forall (s, r) \in \text{Store} \times RDB. (g(s, r) = \bot \vee g(s, r) = g'(s, r)).$$

**Lemma 4.5.** $(\mathcal{D}, \sqsubseteq)$ is an $\omega$-complete partial order and has the least element $\bot \mapsto \bot$. Thus, every continuous function $G$ on $\mathcal{D}$ has a least fixed point (and this least fixed point is unique).

For each $g \in \mathcal{D}$, let $g^\dagger$ be the following lifting to a function on $\{ \bot \} \cup (\text{Store} \times RDB \times [0, \infty] \times [0, \infty])$:

$$g^\dagger(\bot) = \bot, \quad g^\dagger(s, r, w, p) = \begin{cases} \bot & \text{if } g(s, r) = \bot, \\ \langle s', r', w \times w', p \times p' \rangle & \text{if } g(s, r) = \langle s', r', w', p' \rangle. \end{cases}$$

This lifting lets us compose two functions in $\mathcal{D}$.

Using these preliminary definitions, we define a density semantics in Figure 6, where a command $C$ means a function $[C]_d \in \mathcal{D}$. The notation $r \setminus v$ in the figure means the removal of the entry $v$ from the finite map $r$ if $v \in \text{dom}(r)$; otherwise, $r \setminus v$ is just $r$. The set membership $[C]_d \in \mathcal{D}$ says that $[C]_d$ is a local measurable function from $\text{Store} \times RDB$ to $\{ \bot \} \cup \text{Store} \times RDB \times [0, \infty] \times [0, \infty]$. Thus, the function $[C]_d$ takes a store $s$ and a random database $r$ as inputs, where the former fixes the values of variables at the start of $C$ and the latter specifies random seeds some of which $C$ may consume to sample random variables. Given such inputs, the function outputs an updated store $s'$, the part $r'$ of $r$ not consumed by $C$, the total score $w'$ expressing how well the execution of $C$ matches observations, and the probability density $p'$ of $C$ at the consumed part of $r$. If $r$ does not contain enough random seeds, or the execution of $C$ encounters some runtime error, or it falls into an infinite loop, then the function returns $\bot$.

**Lemma 4.6.** For every command $C$, its semantics $[C]_d$ is well-defined and belongs to $\mathcal{D}$.
The density semantics $\llbracket C \rrbracket_d$ is closely related to the measure semantics $\llbracket C \rrbracket$ defined in §4.2. Both interpretations of $C$ describe the computation of $C$ but from slightly different perspectives. To state this relationship formally, we need a few notations. For $g \in D$ and $s \in Store$, define

$$dens(g, s) : RDB \rightarrow [0, \infty), \quad dens(g, s)(r) \triangleq \begin{cases} w' \times p' & \text{if } \exists s', w', p'. \ (g(s, r) = (s', [], w', p')), \\ 0 & \text{otherwise}, \end{cases}$$

$$get(g, s) : RDB \rightarrow Store \cup \{\bot\}, \quad get(g, s)(r) \triangleq \begin{cases} s' & \text{if } \exists s', w', p'. \ (g(s, r) = (s', [], w', p')), \\ \bot & \text{otherwise}. \end{cases}$$

Both $dens(g, s)$ and $get(g, s)$ are concerned with random databases $r$ that precisely describe the randomness needed by the execution of $g$ from $s$. This is formalised by the use of $[]$ in the definitions. The function $dens(g, s)$ assigns a score to such an $r$, and in so doing, it defines a probability density on $RDB$ with respect to the reference measure $\rho$. The function $get(g, s)$ computes a store that the computation of $g$ would result in when started with such an $r$. We often write $dens(C, s)$ and $get(C, s)$ to mean $dens(\llbracket C \rrbracket_d, s)$ and $get(\llbracket C \rrbracket_d, s)$, respectively.

**Lemma 4.7.** For all $g \in D$, the following functions from $Store \times RDB$ to $\mathbb{R}$ and $\{\bot\} \cup Store$ are measurable: $(s, r) \mapsto dens(g, s)(r)$ and $(s, r) \mapsto get(g, s)(r)$.

The next lemma is the main reason that we considered the locality property. It plays a crucial role in proving Theorem 4.9, the key result of this subsection.

**Lemma 4.8.** For all non-negative bounded measurable functions $h : (\{\bot\} \cup Store) \times RDB \rightarrow \mathbb{R}$, stores $s$, and functions $g_1, g_2 \in D$, we have that

$$\int \rho(dr) \left( dens(g_2^R \circ g_1, s)(r) \cdot h\left( get(g_2^R \circ g_1, s)(r), r \right) \right)$$

$$= \int \rho(dr_1) \left( dens(g_1, s)(r_1) \cdot 1_{[get(g_1, s)(r_1) \neq \bot]} \right) \cdot \int \rho(dr_2) \left( dens(g_2, get(g_1, s)(r_1))(r_2) \cdot 1_{[r_1 \neq r_2]} \cdot h\left( get(g_2, get(g_1, s)(r_1))(r_2), r_1 \cup r_2 \right) \right).$$

Assume that $(\{\bot\} \cup Store) \times RDB$ is ordered as follows: for all $(a, r), (a', r') \in (\{\bot\} \cup Store) \times RDB$, $(a, r) \sqsubseteq (a', r') \iff (a = \bot \lor a = a') \land r = r'$.

**Theorem 4.9.** For all non-negative bounded measurable monotone functions $h : (\{\bot\} \cup Store) \times RDB \rightarrow \mathbb{R}$ and states $\sigma$, $\int \llbracket C \rrbracket (\sigma)(d(\sigma', w')) (w' \cdot h(\sigma'_s, \sigma'_t)) = \int \rho(dr') \left( dens(C, \sigma_s)(r') \cdot 1_{[r' \neq \sigma_t]} \cdot h\left( get(C, \sigma_s)(r'), r' \cup \sigma_t \right) \right)$.

**Corollary 4.10.** $Mea(C, A) = \int \rho(dr) \left( 1_{[r \in A]} \cdot dens(C, s_I)(r) \right)$ for all $C$ and all measurable $A$.

This corollary says that $dens(C, s_I)$ is the density of the measure $Mea(C, \cdot)$ with respect to $\rho$, and supports our informal claim that $\llbracket C \rrbracket_d$ computes the density of the measure $\llbracket C \rrbracket$.

### 5 STOCHASTIC VARIATIONAL INFERENCE

In this section, we explain stochastic variational inference (SVI) algorithms using the semantics that we have developed so far. In particular, we describe the requirements made implicitly by one fundamental SVI algorithm, which is regarded most permissive by the ML researchers because the algorithm does not require the differentiability of the density of a given probabilistic model.

We call a command $C$ model if it has a finite nonzero normalising constant:

$$Z_C = Mea(C, RDB) = \left( \int \llbracket C \rrbracket (\sigma_I)(d(\sigma, w)) w \right) \in (0, \infty),$$
model $C \equiv \{ s := \text{sample}_{\text{norm}}("slope", 0.0, 5.0); i := \text{sample}_{\text{norm}}("intercept", 0.0, 5.0); x_1 := 1.0; y_1 := 2.3; x_2 := 2.0; y_2 := 4.2; x_3 := 3.0; y_3 := 6.9; \text{score}_{\text{norm}}(y_1, s \cdot x_1 + i, 1.0); \text{score}_{\text{norm}}(y_2, s \cdot x_2 + i, 1.0); \text{score}_{\text{norm}}(y_3, s \cdot x_3 + i, 1.0) \}$
guide $D_{\theta} \equiv \{ s := \text{sample}_{\text{norm}}("slope", \theta_1, \exp(\theta_2)); i := \text{sample}_{\text{norm}}("intercept", \theta_3, \exp(\theta_4)) \}$

where $\sigma_I$ is the initial state. Given a model $C$, the SVI algorithms attempt to infer a good approximation of $C$’s posterior distribution $\Pr(C, \cdot)$ defined in (2). They tackle this posterior-inference problem in two steps.

First, the SVI algorithms fix a collection of approximate distributions. They usually do so by asking the developer of $C$ to provide a command $D_{\theta}$ parameterised by $\theta \in \mathbb{R}^p$, which can serve as a template for approximation distributions. The command $D_{\theta}$ typically has a control-flow structure similar to that of $C$, but it is simpler than $C$: it does not use any score statements, and may replace complex computation steps of $C$ by simpler ones. In fact, $D_{\theta}$ should satisfy two formal requirements, which enforce this simplicity. The first is

$$\text{Mea}(D_{\theta}, \text{RDB}) = 1 \quad \text{for all } \theta \in \mathbb{R}^p,$$

which means that the normalising constant of $D_{\theta}$ is 1. The second is that $D_{\theta}$ should keep the score (i.e., the $w$ component) to be 1, i.e.,

$$\mathbb{D}[D_{\theta}](\sigma_I)(\text{State} \times ([0, \infty) \setminus \{1\})) = 0.$$

Meeting these requirements is often not too difficult. A common technique is to ensure that $D_{\theta}$ does not use the score statement and always terminates. Figure 7 gives an example of $(C, D_{\theta})$ for simple Bayesian linear regression with three data points. Note that in this case, $D_{\theta}$ is obtained from $C$ by deleting the score statements and replacing the parameter $0.0$ and $5.0$ of normal distributions by parameter $\theta = (\theta_1, \theta_2, \theta_3, \theta_4)$. Following the terminology of Pyro, we call a parameterised command $D_{\theta}$ guide if it satisfies the two requirements just mentioned.

Second, the SVI algorithms search for a good parameter $\theta$ that makes the distribution described by $D_{\theta}$ close to the posterior of $C$. Concretely, they formulate an optimisation problem where the optimisation objective expresses that some form of distance from $D_{\theta}$’s posterior distribution to $C$’s posterior should be minimised. Then, they solve the problem by a version of gradient descent.

The KL divergence is a standard choice for distance. Let $\mu, \mu'$ be measures on $\text{RDB}$ that have densities $g$ and $g'$ with respect to the measure $\rho$. The KL divergence from $g$ to $g'$ is defined by

$$\text{KL}(g||g') \equiv \int \rho(\text{dr}) \left( g(r) \cdot \log \frac{g(r)}{g'(r)} \right).$$

In words, it is the ratio of densities $g$ and $g'$ averaged according to $g$. If $g = g'$, the ratio is always 1, so that the KL becomes 0. The KL divergence is defined only if the following conditions are met:

- **Absolute continuity**: $g'(r) = 0 \implies g(r) = 0$ for all $r \in \text{RDB}$, which ensures that the integrand in (4) is well-defined even when the denominator $g'(r)$ in (4) takes the value 0;
- **Integrability**: the integral in (4) has a finite value.

Using our semantics, we can express the KL objective as follows:

$$\arg\min_{\theta \in \mathbb{R}^p} \text{KL} \left( \frac{\text{dens}(D_{\theta}, s_I)}{\text{dens}(C, s_I)} \right) / Z_C.$$  

\(^{9}\text{This condition can be relaxed in a more general formulation of the KL divergence stated in terms of the so called Radon-Nikodym derivative. We do not use the relaxed condition to reduce the amount of materials on measure theory in the paper.\)}}
Recall that \( \text{dens}(D_\theta, s_I) \) and \( \text{dens}(C, s_I)/Z_C \) are densities of the probability measures of the command \( D_\theta \) and the posterior of \( C \) (Corollary 4.10), and they are defined by means of our density semantics in §4.3. Most SVI engines solve this optimisation problem by a version of gradient descent.

In the paper, we consider one of the most fundamental and versatile SVI algorithms. The algorithm is called score estimator or REINFORCE, and it works by estimating the gradient of the objective in (5) using samples and performing the gradient descent with this estimated gradient. More concretely, the algorithm starts by initialising \( \theta \) with some value (usually chosen randomly) and updating it repeatedly by the following procedure:

(i) Sample \( r_1, \ldots, r_N \) independently from \( \text{dens}(D_\theta, s_I) \)

(ii) \( \theta \leftarrow \theta - \eta \times \left( \frac{1}{N} \sum_{i=1}^{N} (\nabla_\theta \log \text{dens}(D_\theta, s_I)(r_i)) \cdot \log \frac{\text{dens}(D_\theta, s_I)(r_i)}{\text{dens}(C, s_I)(r_i)} \right) \)

Here \( N \) and \( \eta \) are hyperparameters to this algorithm, the former determining the number of samples used to estimate the gradient and the latter, called learning rate, deciding how much the algorithm should follow the direction of the estimated gradient. Although we do not explain here, sampling \( r_1, \ldots, r_N \) and computing all of \( \text{dens}(D_\theta, s_I)(r_i) \), \( \text{dens}(C, s_I)(r_i) \) and \( \nabla_\theta (\log \text{dens}(D_\theta, s_I)(r_i)) \) can be done by executing \( D_\theta \) and \( C \) multiple times under slightly unusual operational semantics [Yang 2019]. The SVI engines of Pyro and Anglican implement such operational semantics.

The average over the \( N \) terms in the \( \theta \)-update step is the core of the algorithm. It approximates the gradient of the optimisation objective in (5):

\[
\nabla_\theta \text{KL}\left( \frac{\text{dens}(D_\theta, s_I)}{Z_C} \right) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \nabla_\theta \log \text{dens}(D_\theta, s_I)(r_i) \right) \cdot \log \frac{\text{dens}(D_\theta, s_I)(r_i)}{\text{dens}(C, s_I)(r_i)}.
\]

The average satisfies an important property called unbiasedness, summarised by Theorem 5.1.

**Theorem 5.1.** Let \( C \) be a model, \( D_\theta \) be a guide, and \( N \neq 0 \in \mathbb{N} \). Define \( \text{KL}_{(-)} : \mathbb{R}^P \rightarrow \mathbb{R}_{\geq 0} \) as \( \text{KL}_\theta \triangleq \text{KL}(\text{dens}(D_\theta, s_I)||\text{dens}(C, s_I)/Z_C) \). Then, \( \text{KL}_{(-)} \) is well-defined and continuously differentiable with

\[
\nabla_\theta \text{KL}_\theta = \mathbb{E}_{\Pi_i \text{dens}(D_\theta, s_I)(r_i)} \left[ \frac{1}{N} \sum_{i=1}^{N} \left( \nabla_\theta \log \text{dens}(D_\theta, s_I)(r_i) \right) \cdot \log \frac{\text{dens}(D_\theta, s_I)(r_i)}{\text{dens}(C, s_I)(r_i)} \right]
\]

if

(R1) \( \text{dens}(C, s_I)(r) = 0 \implies \text{dens}(D_\theta, s_I)(r) = 0 \), for all \( r \in \text{RDB} \) and \( \theta \in \mathbb{R}^P \);

(R2) for all \( (r, \theta, j) \in \text{RDB} \times \mathbb{R}^P \times [p] \), the function \( \nu \mapsto \text{dens}(D_{\theta[j:v]}, s_I)(r) \) on \( \mathbb{R} \) is differentiable;

(R3) for all \( \theta \in \mathbb{R}^P \),

\[
\int \rho(\text{dr}) \left( \text{dens}(D_{\theta}, s_I)(r) \cdot \log \frac{\text{dens}(D_{\theta}, s_I)(r)}{\text{dens}(C, s_I)(r)} \right) < \infty;
\]

(R4) for all \( (\theta, j) \in \mathbb{R}^P \times [p] \), the function

\[
\nu \mapsto \int \rho(\text{dr}) \left( \text{dens}(D_{\theta[j:v]}, s_I)(r) \cdot \log \frac{\text{dens}(D_{\theta[j:v]}, s_I)(r)}{\text{dens}(C, s_I)(r)} \right)
\]

on \( \mathbb{R} \) is continuously differentiable;

(R5) for all \( \theta \in \mathbb{R}^P \),

\[
\nabla_\theta \int \rho(\text{dr}) \left( \text{dens}(D_{\theta}, s_I)(r) \cdot \log \frac{\text{dens}(D_{\theta}, s_I)(r)}{\text{dens}(C, s_I)(r)} \right) = \int \rho(\text{dr}) \nabla_\theta \left( \text{dens}(D_{\theta}, s_I)(r) \cdot \log \frac{\text{dens}(D_{\theta}, s_I)(r)}{\text{dens}(C, s_I)(r)} \right);
\]

(R6) for all \( \theta \in \mathbb{R}^P \),

\[
\int \rho(\text{dr}) \nabla_\theta \text{dens}(D_{\theta}, s_I)(r) = \nabla_\theta \int \rho(\text{dr}) \text{dens}(D_{\theta}, s_I)(r).
\]

Here \( \theta[j:v] \) denotes a vector in \( \mathbb{R}^P \) that is the same as \( \theta \) except that its \( j \)-th component is \( v \).
The conclusion of this theorem (Equation (6)) and its proof are well-known [Ranganath et al. 2014], but the requirements in the theorem (and the continuous differentiability of $\KL_{\theta}$ in the conclusion) are rarely stated explicitly in the literature.

The correctness of the algorithm crucially relies on the unbiasedness property in Theorem 5.1. The property ensures that the algorithm converges to a local minimum with probability 1. Thus, it is important that the requirements in the theorem are met. In fact, some of the requirements there are needed even to state the optimisation objective in (5), because without them, the objective does not exist. In the next section, we describe conditions that imply those requirements and can serve as target properties of program analysis for probabilistic programs. The latter point is worked out in detail in §7 and §8 where we discuss program analysis for probabilistic programs and SVI.

6 CONDITIONS FOR STOCHASTIC VARIATIONAL INFERENCE

Ideally we want to have program analysers that discharge the six requirements R1-R6 in Theorem 5.1. However, except R1 and R2, the requirements are not ready for serving as the targets of static analysis algorithms. Automatically discharging them based on the first principles (such as the definition of integrability with respect to a measure) may be possible, but seems less immediate than doing so using powerful theorems from continuous mathematics.

In this section, we explain conditions that imply the requirements R3-R6 and are more friendly to program analysis than the requirements themselves. The conditions are given in two boxes (9) and (10). Throughout the section, we fix a model $C$ and a guide $D_{\theta}$.

6.1 Assumption

Throughout the section, we assume that the densities of $D_{\theta}$ and $C$ have the following form:

$$
\text{dens}(D_{\theta}, s_j)(r) = \sum_{i=1}^{M} \mathbb{1}_{[r \in A_i]} \prod_{\alpha \in K_i} \mathcal{N}(r(\alpha); \mu_{(i,a)}(\theta), \sigma_{(i,a)}(\theta)),
$$

and

$$
\text{dens}(C, s_j)(r) = \sum_{i=1}^{M} \mathbb{1}_{[r \in A_i]} \left( \prod_{\alpha \in K_i} \mathcal{N}(r(\alpha); \mu'_{(i,a)}(r), \sigma'_{(i,a)}(r)) \right) \left( \prod_{j[N_i]} \mathcal{N}(c_{(i,j)}; \mu''_{(i,j)}(r), \sigma''_{(i,j)}(r)) \right),
$$

where

- $M, N_i \in \mathbb{N} \setminus \{0\}$;
- $A_1, \ldots, A_M$ are disjoint measurable subsets of $RDB$;
- $K_i$’s are finite sets of strings such that $\text{dom}(r) = K_i$ for all $r \in A_i$;
- $\mu_{(i,a)}$ and $\sigma_{(i,a)}$ are measurable functions from $\mathbb{R}^P$ to $\mathbb{R}$ and $(0, \infty)$, respectively;
- $\mu'_{(i,a)}$ and $\mu''_{(i,j)}$ are measurable functions from $[K_i \to \mathbb{R}]$ to $\mathbb{R}$;
- $\sigma'_{(i,a)}$ and $\sigma''_{(i,j)}$ are measurable functions from $[K_i \to \mathbb{R}]$ to $(0, \infty)$;
- $c_{(i,j)}$ is a real number.

In programming terms, our assumption first implies that both $C$ and $D_{\theta}$ use at most a fixed number of random variables. That is, the number of random variables they generate must be finite not only within a single execution but also across all possible executions since the names of all random variables are found in a finite set $\bigcup_{i=1}^{M} K_i$. This property is met if the number of steps in every execution of $C$ and $D_{\theta}$ from $\sigma_I$ is bounded by some $T$ and the executions over the same program path use the same set of random variables. Note that the bound may depend on $s_I$. Such a bound exists for most probabilistic programs.10 Also, the assumption says that the parameters of normal distributions in sample statements in $D_{\theta}$ may depend only on $\theta$, but not on other sampled random variables using probabilistic grammars or those from Bayesian nonparametrics.

---

10Notable exceptions are models using probabilistic grammars or those from Bayesian nonparametrics.
variables. This is closely related to a common approach for designing approximate distributions in variational inference, called mean-field approximation, where the approximate distribution consists of independent normal random variables.

We use the term “assumption” here, instead of “condition” in the following subsections because the assumed properties are rather conventional and they are not directly related to the requirements in Theorem 5.1, at least not as much as the conditions that we will describe next.

6.2 Condition for Requirement R3

Note that the integral in the requirement R3 can be written as the sum of two expectations:

$$\int \rho(dr) \left( \frac{\text{dens}(D_\theta, s_1)(r)}{\text{dens}(C, s_1)(r)} \cdot \log \frac{\text{dens}(D_\theta, s_1)(r)}{\text{dens}(C, s_1)(r)} \right) = \mathbb{E}_{\text{dens}(D_\theta, s_1)}[\log \text{dens}(D_\theta, s_1)(r)] - \mathbb{E}_{\text{dens}(D_\theta, s_1)}[\log \text{dens}(C, s_1)(r)].$$

(8)

The minus of the first term (i.e., $$-\mathbb{E}_{\text{dens}(D_\theta, s_1)}[\log \text{dens}(D_\theta, s_1)(r)]$$) is called the differential entropy of the density $$\text{dens}(D_\theta, s_1)$$. Intuitively, it is large when the density on $$\mathbb{R}^n$$ is close to the Lebesgue measure, which is regarded to represent the absence of information. The differential entropy is sometimes undefined [Ghourchian et al. 2017]. Fortunately, a large class of probability densities (containing many commonly used probability distributions) have well-defined entropies [Ghourchian et al. 2017; Nair et al. 2006]. Our $$\text{dens}(D_\theta, s_1)$$ is one of such fortunate cases.

**Theorem 6.1.** $$\mathbb{E}_{\text{dens}(D_\theta, s_1)}[\log \text{dens}(D_\theta, s_1)(r)] < \infty$$ under our assumption in §6.1.

We remark that a violation of the assumption (7) for $$D_\theta$$ can result in an undefined entropy, as illustrated by the following examples.

**Example 6.2.** Consider guides $$D_{(i, \theta)}$$ defined as follows ($$i = 1, 2$$):

$$D_{(i, \theta)} \equiv (x_1 := \text{sample}_{\text{norm}}("a_1", \theta, 1); \ x_2 := \text{sample}_{\text{norm}}("a_2", \theta, E_i[x_1]))$$

where for some $$n \geq 1$$ and $$c \neq 0 \in \mathbb{R}, \tag{9}$$

$$E_1[x_1] \equiv \text{if } (x_1 = 0) \text{ then } 1 \text{ else } \exp(-1/|x_1|), \quad E_2[x_1] \equiv \exp(c \cdot x_1^2).$$

None of $$\text{dens}(D_{(i, \theta)}, s_1)'$$’s satisfies the assumption (7) because the standard deviation of the normal distribution for $$x_2$$ depends on the value of $$x_1$$. The entropies of $$\text{dens}(D_{(i, \theta)}, s_1)'$$’s are all undefined:

$$\mathbb{E}_{\text{dens}(D_{(i, \theta)}, s_1)}[\log \text{dens}(D_{(i, \theta)}, s_1)(r)] = \infty \text{ for all } i = 1, 2.$$  

Since the first term of (8) is always finite by Theorem 6.1, it is enough to ensure the finiteness of the second term of (8). For $$i \in [M]$$, define the set of (absolute) affine functions on $$[K_i \rightarrow \mathbb{R}]$$ as:

$$\mathcal{A}_i \triangleq \left\{ f \in [K_i \rightarrow \mathbb{R}] \rightarrow \mathbb{R} \mid f(r) = c + \sum_{a \in K_i} (c_a \cdot |r(a)|) \text{ for some } c, c_a \in \mathbb{R} \right\}.$$  

Our condition for ensuring the finiteness of the second term is as follows:

\begin{equation}
\begin{align*}
\text{For all } i \in [M], \text{ there are } f', f'', f', l', u', l', u' & \in \mathcal{A}_i \text{ such that} \\
|\mu_{(i,a)}'(r)| & \leq \exp(f'(r)), \quad \exp(l'(r)) \leq \sigma_{(i,a)}'(r) \leq \exp(u'(r)), \\
|\mu_{(i,j)}'(r)| & \leq \exp(f''(r)), \quad \exp(l''(r)) \leq \sigma_{(i,j)}''(r) \leq \exp(u''(r)),
\end{align*}
\end{equation}

(9)

**Theorem 6.3.** The condition (9) implies $$\mathbb{E}_{\text{dens}(D_\theta, s_1)}[\log \text{dens}(C, s_1)(r)] < \infty$$ under our assumption in §6.1. Thus, in that case, it entails the requirement R3 (i.e., the objective in (5) is well-defined).

\[\text{Formally, we should implement } E_i[x_1] \text{ as an application of a primitive function } f_1 \text{ to } x_1 \text{ that has the semantics described by the if-then-else statement here.}\]
Our condition in (9) is sufficient but not necessary for the objective to be defined. However, its violation is a good warning, as illustrated by our next examples.

**Example 6.4.** Consider models $C_1, \ldots, C_4$ and a guide $D_\theta$ defined as follows:

\begin{align*}
C_i & \equiv (x_1 := \text{sample}_{\text{norm}}(a_1, 0, 1); \ x_2 := \text{sample}_{\text{norm}}(a_2, E_i(x_1), 1)) \quad \text{for } i = 1, 2 \\
C_i & \equiv (x_1 := \text{sample}_{\text{norm}}(a_1, 0, 1); \ x_2 := \text{sample}_{\text{norm}}(a_2, 0, E_i(x_1))) \quad \text{for } i = 3, 4 \\
D_\theta & \equiv (x_1 := \text{sample}_{\text{norm}}(a_1, \theta_1, 1); \ x_2 := \text{sample}_{\text{norm}}(a_2, \theta_2, 1))
\end{align*}

where for some $n \geq 1$ and $c \neq 0 \in \mathbb{R}$,

\[ E_1(x_1) \equiv \text{if } (x_1=0) \text{ then } 0 \text{ else } \frac{1}{x_1^n}, \quad E_2(x_1) \equiv E_4(x_1) \equiv \exp(c \cdot x_1^3), \quad E_3(x_1) \equiv \text{if } (x_1=0) \text{ then } 1 \text{ else } |x_1|^n. \]

Let $A = \{\{a_1^*, a_2^*\} \to \mathbb{R}\}$. For $r \in A$, define

\[ \mu'_i(r) \triangleq \text{if } r(a_1^*) = 0 \text{ then } 0 \text{ else } 1/r(a_1^*)^{n_i}, \quad \mu'_i(r) \triangleq \exp(c \cdot r(a_1^*)^3), \]

\[ \sigma'_j(r) \triangleq \text{if } r(a_1^*) = 0 \text{ then } 0 \text{ else } |r(a_1^*)|^{n_j}, \quad \sigma'_j(r) \triangleq \exp(c \cdot r(a_1^*)^3). \]

Then, we have that

\begin{align*}
dens(C_i, s_j)(r) & = \mathbb{I}_{\{r \in A\}} \cdot \mathcal{N}(r(a_1^*); 0, 1) \cdot \mathcal{N}(r(a_2^*); \mu'_i(r), 1) \quad \text{for } i = 1, 2 \\
dens(C_i, s_j)(r) & = \mathbb{I}_{\{r \in A\}} \cdot \mathcal{N}(r(a_1^*); 0, 1) \cdot \mathcal{N}(r(a_2^*); 0, \sigma'_j(r)) \quad \text{for } i = 3, 4 \\
dens(D_\theta, s_j)(r) & = \mathbb{I}_{\{r \in A\}} \cdot \mathcal{N}(r(a_1^*); \theta_1, 1) \cdot \mathcal{N}(r(a_2^*); \theta_2, 1).
\end{align*}

None of $\mu'_1$, $\mu'_2$, $\sigma'_3$, and $\sigma'_4$ satisfies the condition in (9). The function $\mu'_i$ is not bounded in $\{r \in A \mid -1 \leq r(a_1^*) \leq 1 \land -1 \leq r(a_2^*) \leq 1\}$, but every $\mu'_i$ satisfying the condition in (9) should be bounded. Also, the cubic exponential growth of $\mu'_i$ cannot be bounded by any linear exponential function on $r$. The violation of the condition by $\sigma'_3$ and $\sigma'_4$ can be shown similarly.

In fact, the objective in (5) is not defined for all of the four cases. This is because for all $i = 1, \ldots, 4$, 

\[ \mathbb{E}_{\text{dens}(D_\theta, s_j)(r)} \left[ |\log \text{dens}(C_i, s_j)(r)| \right] = \infty. \]

We now show that the condition in (9) is satisfied by a large class of functions in neural network learning applications, including functions using neural networks. We call a function $\text{nn} : \mathbb{R}^n \to \mathbb{R}$ an affine-bounded neural network if there exist functions $f_j : \mathbb{R}^{n_j} \to \mathbb{R}^{n_{j+1}}$ and affine functions $l_j : \mathbb{R}^{n_j} \to \mathbb{R}$ for all $1 \leq j \leq d$ such that (i) $n_1 = n$ and $n_{d+1} = 1$; (ii) $\text{nn} = f_d \circ \cdots \circ f_1$, and (iii) $\|f_j(v)\|_1 \leq l_j(|v_1|, \ldots, |v_{n_j}|)$ for all $1 \leq j \leq d$ and $v \in \mathbb{R}^{n_j}$, where $\| \cdot \|_1$ denotes the $\ell_1$-norm. Note that each component of $f_j$ can be, for instance, an affine function, one of commonly used activation functions (e.g., relu, tanh, sigmoid, softplus), or one of min/max functions (min and max).

Therefore, most of neural networks used in machine learning applications are indeed affine-bounded. Lemma 6.5 indicates that a wide range of functions satisfy the condition (9).

**Lemma 6.5.** Pick $i \in [M]$ and $\alpha \in K_i$. Let $(\alpha_1, \ldots, \alpha_J)$ be an enumeration of the elements in $K_i$, and $\bar{\alpha} \triangleq (r(\alpha_1), \ldots, r(\alpha_J)) \in \mathbb{R}^J$ be an enumeration of the values of $r \in A_i$. Consider any affine-bounded neural network $\text{nn} : \mathbb{R}^J \to \mathbb{R}$, polynomial $\text{poly} : \mathbb{R}^J \to \mathbb{R}$, and $c \in (0, \infty)$. Then, the below list of functions $\mu'_i(\alpha, \alpha)$ and $\sigma'_i(\alpha, \alpha)$ on $A_i$ satisfy the condition in (9):

\begin{align*}
\mu'_i(\alpha, \alpha)(r) & = \text{nn}(\bar{\alpha}), \quad \mu'_i(\alpha, \alpha)(r) = \text{poly}(\bar{\alpha}), \quad \mu'_i(\alpha, \alpha)(r) = \exp(\text{nn}(\bar{\alpha})), \\
\sigma'_i(\alpha, \alpha)(r) & = |\text{nn}(\bar{\alpha})| + c, \quad \sigma'_i(\alpha, \alpha)(r) = |\text{poly}(\bar{\alpha})| + c, \quad \sigma'_i(\alpha, \alpha)(r) = \exp(\text{nn}(\bar{\alpha})), \\
\sigma'_i(\alpha, \alpha)(r) & = (|\text{nn}(\bar{\alpha})| + c)^{-1}, \quad \sigma'_i(\alpha, \alpha)(r) = (|\text{poly}(\bar{\alpha})| + c)^{-1}, \quad \sigma'_i(\alpha, \alpha)(r) = \text{softplus}(\text{nn}(\bar{\alpha})),
\end{align*}

where softplus$(v) \triangleq \log(1 + \exp(v))$. Moreover, the same holds for $\mu''_i(\alpha, \alpha)$ and $\sigma''_i(\alpha, \alpha)$ as well.
6.3 Condition for Requirements R4-R6

Assume that the model $C$ and the guide $D_\theta$ satisfy our assumption and condition in the previous two subsections. Our condition for the requirements R4-R6 is given below:

For all $i \in [M], \alpha \in K_i$, and $(\theta, j) \in \mathbb{R}^p \times [p],$

\[
\begin{align*}
\text{the function } v \in \mathbb{R} \mapsto \mu_{(i,\alpha)}(\theta[j : v]) & \text{ is continuously differentiable; } \\
\text{the function } v \in \mathbb{R} \mapsto \sigma_{(i,\alpha)}(\theta[j : v]) & \text{ is continuously differentiable.}
\end{align*}
\]

Theorem 6.6. If both our assumption in §6.1 and the condition (9) hold, then the condition (10) implies the requirements R4-R6.

The proof of the theorem uses the following nontrivial result [Klenke 2014, Theorem 6.28] about exchanging differentiation and integration, a consequence of the dominated convergence theorem.

Theorem 6.7. Let $V \subset \mathbb{R}$ be an open interval, and $(X, \Sigma, \mu)$ be a measure space. Suppose that a measurable function $f : V \times X \to \mathbb{R}$ satisfies the following conditions: (a) for all $v \in V$, the integral $\int \mu(dx) f_v(x)$ is well-defined; (b) for almost all $x \in X$ (w.r.t. $\mu$) and all $v \in V$, the partial derivative $\nabla_v f_v(x)$ with respect to $v$ is well-defined;\footnote{A more popular notation is $(\partial f_v(x)) / (\partial v)$, but we opt for $\nabla_v f_v(x)$ to avoid clutter.} (c) there is a measurable function $h : X \to \mathbb{R}$ such that $\int \mu(dx) h(x)$ is well-defined and $|\nabla_v f_v(x)| \leq h(x)$ for all $v \in V$ and almost all $x \in X$ (w.r.t. $\mu$). Then, for all $v \in V$, both sides of the below equation are well-defined, and the equality holds:

$$
\nabla_v \int \mu(dx) f_v(x) = \int \mu(dx) \nabla_v f_v(x).
$$

Note that the theorem ensures not only the validity of interchanging differentiation and integration, but also the differentiability of $\int \mu(dx) f_v(x)$ (w.r.t. $v$) and the integrability of $\nabla_v f_v(x)$ over $x \in X$.

Our condition in (10) is sufficient but not necessary for the requirements R4-R6 to hold, in particular for the objective in (5) to have well-defined partial derivatives in $\theta$. However, its violation is a good indication of a potential problem. The following example illustrates this point.

Example 6.8. Consider a model $C$ and a guide $D_\theta$ defined as follows:

$$
C \equiv x := \text{sample}_{\text{norm}}("a", 0, 1) \quad D_\theta \equiv x := \text{sample}_{\text{norm}}("a", 0, E[\theta])
$$

where $E[\theta] \equiv \text{relu}(\theta) + 2$ and $\text{relu}(v) \triangleq \mathbf{1}_{[v \geq 0]} \cdot v$. Such $E[\theta]$ can definitely appear in machine learning applications, once a guide starts to use neural networks with parameters $\theta$. Let $A = \{"a" \rightarrow \mathbb{R}\}$ and $\sigma(\theta) \triangleq \text{relu}(\theta) + 2$. Then, $\text{dens}(C, s_j)(r) = \mathbf{1}_{[r \in A]} \cdot \mathcal{N}(\text{r("a")}; 0, 1)$ and $\text{dens}(D_\theta, s_j)(r) = \mathbf{1}_{[r \in A]} \cdot \mathcal{N}(\text{r("a")}; 0, \sigma(\theta))$. Note (10) is violated: $\sigma$ is non-differentiable at $\theta = 0$. A simple calculation shows:

$$
\nabla_\theta \int \rho(dr) \frac{\text{dens}(D_\theta, s_j)(r)}{\text{dens}(C, s_j)(r)} \log \frac{\text{dens}(D_\theta, s_j)(r)}{\text{dens}(C, s_j)(r)} = \begin{cases} 
0 & \text{if } \theta \in (-\infty, 0) \\
((2 + \theta)^2 - 1) / (2 + \theta) & \text{if } \theta \in (0, \infty) \\
\text{undefined} & \text{if } \theta = 0.
\end{cases}
$$

Hence, the objective in (5) does not have a well-defined partial derivative in $\theta$ at $\theta = 0$. □

7 ANALYSIS

In this section, we describe a recipe for building a static analysis that automatically discharges some of the assumptions and conditions given in §6. The recipe ensures that the constructed static analyses are sound with respect to the density semantics in §4.3. We illustrate it by describing four static analyses for verifying the model-guide support match (the requirement R1), the guide-parameter differentiability (the requirement R2), the condition (9), and the condition (10). The analysis for the model-guide support match has been developed significantly more for the Pyro
programming language, and applied to analyse realistic examples of the language. This fully-blown analysis and our experiments will be described in §8.

Throughout this section, we assume that the parameters \( \theta \) of a guide \( D_0 \) are included in \( \text{Var} \), and are only read by \( D_0 \) and not accessed by a model \( C \). What we used to call \( s_I \) will be the part of the store for variables in \( \text{Var} \setminus \theta \), and what we used to write \( \theta \) will correspond to the other part for \( \theta \).

### 7.1 Generic Program Analysis Framework

Our recipe is for building a static analysis that infers information about the state transformation of a given command. It is similar to the conventional methodology for building a so called relational static analysis, which also attempts to find information about the relationship between input and output states of a given command. However, our recipe diverges from the convention in one important point: while the abstract states of conventional relational analyses represent relations on states, we let abstract states directly express sets of concrete state transformers. This departure from the convention is due to the difficulty of using relations for expressing properties of state transformers that we desire. For instance, we could not express a set of functions with a certain type of differentiability using relations.

Recall the domain \( D \) in (3), and the notion of admissible subset from domain theory: \( D_0 \subseteq D \) is admissible if it contains \( \bot \) and is closed under taking the limits of \( \omega \)-chains in \( D_0 \).

Our recipe assumes an abstraction instance defined by the following items:

- An abstract domain, i.e., a set \( \mathcal{T}^\# \) with a designated element \( \bot^\# \).
- A concretisation function, i.e., a function \( \gamma : \mathcal{T}^\# \rightarrow \mathcal{P}(D) \) such that for every \( t \in \mathcal{T}^\# \), \( \gamma(t) \) is an admissible subset of \( D \). Note that the concretisation interprets each abstract element \( t \) as a set of concrete transformers in \( D \). The admissibility is imposed to enable the sound analysis of loops.
- A widening operator \( \text{widen} : \mathcal{T}^\# \times \mathcal{T}^\# \rightarrow \mathcal{T}^\# \), such that for all \( t_1, t_2 \in \mathcal{T}^\# \) and \( i \in [2] \), \( \gamma(t_i) \subseteq \gamma(\text{widen}(t_1, t_2)) \) and for every sequence \( \{t_n\}_{n \geq 1} \) in \( \mathcal{T}^\# \), its widened sequence \( \{t'_n\}_{n \geq 1} \), defined by \( t'_1 \triangleq t_1 \) and \( t'_{n+1} \triangleq \text{widen}(t'_n, t_{n+1}) \) for \( n \geq 1 \), has an index \( m \) such that \( t'_m = t'_{m+1} \).
- An abstract conditional operator for every expression \( E \), that is, a function \( \text{cond}(E)^\# : \mathcal{T}^\# \times \mathcal{T}^\# \rightarrow \mathcal{T}^\# \) such that for all \( t_1, t_2 \in \mathcal{T}^\# \) and \( g_1, g_2 \in D \), if \( g_1 \in \gamma(t_1) \) and \( g_2 \in \gamma(t_2) \), then \( (\lambda(s,r). (if (\langle E \rangle s = true) then g_1(s,r) else g_2(s,r))) \in \gamma(\text{cond}(E)^\#(t_1, t_2)) \).
- An abstract composition operator \( \circ^\# : \mathcal{T}^\# \times \mathcal{T}^\# \rightarrow \mathcal{T}^\# \) such that for all \( t_1, t_2 \in \mathcal{T}^\# \) and \( g_1, g_2 \in D \), if \( g_1 \in \gamma(t_1) \) and \( g_2 \in \gamma(t_2) \), then \( g_2^\circ g_1 \in \gamma(t_2 \circ^\# t_1) \).
- For all expressions \( E_0, E_1, E_2 \) and for all variables \( x \), the abstract elements \( \text{skip}^\# \), \( \text{update}(x, E_0)^\# \), \( \text{sample}(x, S, E_1, E_2)^\# \), and \( \text{score}(E_0, E_1, E_2)^\# \) in \( \mathcal{T}^\# \) such that

\[
\begin{align*}
\llbracket \text{skip} \rrbracket_d^\# & \in \gamma(\text{skip}^\#), & \llbracket x := \text{sample}(S, E_1, E_2) \rrbracket_d^\# & \in \gamma(\text{sample}(x, S, E_1, E_2)^\#), \\
\llbracket x := E_0 \rrbracket_d^\# & \in \gamma(\text{update}(x, E_0)^\#), & \llbracket \text{score}(E_0, E_1, E_2) \rrbracket_d^\# & \in \gamma(\text{score}(E_0, E_1, E_2)^\#).
\end{align*}
\]

Given these data, we define the static analysis \( \llbracket C \rrbracket^\# \in \mathcal{T}^\# \) of a command \( C \) in Figure 8. Here the \( \text{wfix} T \) is the usual widened fixed point of \( T \), which is defined as the first element \( t_m \) with \( t_m = t_{m+1} \) in the widened sequence \( \{t_n\}_{n \geq 1} \) where \( t_1 = \bot^\# \) and \( t_{n+1} \triangleq \text{widen}(t_n, T(t_n)) \).

![Fig. 8. Abstract semantics \( \llbracket C \rrbracket^\# \in \mathcal{T}^\# \) of commands C](image-url)
Theorem 7.1 (Soundness). For all commands $C$, we have $\llbracket C \rrbracket_d \in \gamma(\llbracket C \rrbracket^\#)$.

In the rest of this section, we instantiate this framework into four static analysis instances. In each case, we describe the abstract domain, the abstract bottom element, and the concretisation function. Moreover, in the first two cases, we detail the transfer functions. In the following, for a tuple $(s', r', w', p') \in \text{Store} \times \text{RDB} \times [0, \infty) \times [0, \infty)$, we use the subscripts $-s$, $-r$, $-w$, and $-p$ to denote its components. For instance, $(s', r', w', p')_s = s'$ and $(s', r', w', p')_r = r'$.

7.2 Analysis for Model-Guide Match

The first instance analysis finds information about the names of sampled random variables. Such information can be used for discharging the requirement $R1$, the correspondence between the support of a model and that of a guide. The analysis is based on the below abstraction:

$$
\mathcal{T}^\# \triangleq \{ \bot^\#, \top^\# \} \cup \mathcal{P}(\text{Str}), \quad \bot^\# \triangleq \bot, \quad \top^\# \triangleq \{ \lambda(s, r). \bot \}, \\
\gamma(\top^\#) \triangleq \mathcal{D}, \quad \gamma(\bot^\#) \triangleq \{ g \in \mathcal{D} \mid \forall s, r. \ g(s, r) \neq \bot \land (g(s, r))_r = [] \implies \text{dom}(r) = K \},
$$

where $[ ]$ denotes the empty random database. A typical abstract element in $\mathcal{T}^\#$ is a set of names $K$, which represents concrete commands sampling random variables in $K$. The domain $\mathcal{T}^\#$ contains $\bot^\#$ and $\top^\#$ to express two extreme cases, the set containing only one command that always returns $\bot$, and the set of all commands.

Sound abstract operations can be derived from the density semantics and from the abstraction following the abstract interpretation methodology [Cousot and Cousot 1977]:

$$
\text{widen}(\bot^\#, S) = \text{widen}(S, \bot^\#) = S; \quad \text{cond}(E)^\# = \text{widen}; \quad \text{skip}^\# = \text{update}(x, E_0)^\# = \emptyset; \\
\text{widen}(\top^\#, S) = \text{widen}(S, \top^\#) = \top^\#; \quad \circ^\# = \text{widen}; \quad \text{sample}(x, S, E_1, E_2)^\# = \{ x \}; \\
\text{widen}(S_0, S_1) = S_0 \text{ if } S_0 = S_1, \top^\# \text{ otherwise for } S_0, S_1 \in \mathcal{P}(\text{Str}); \quad \text{score}(E_0, E_1, E_2)^\# = \emptyset.
$$

We can use the resulting abstraction to discharge the requirement $R1. We just need to run it on both $C$ and $D_\theta$, and check whether $\llbracket C \rrbracket^\# = \llbracket D_\theta \rrbracket^\# = K$ for some $K \in \mathcal{P}(\text{Str})$. The positive answer implies the requirement $R1$, because all the random variables are drawn from the normal distribution. Our extension of this analysis for Pyro (§8) does not rely on this exclusive use of the normal distribution, and tracks information about the type of distribution of each random variable and state properties, so as to prove the model-guide support match for realistic Pyro programs.

7.3 Analysis for Guide Parameter Differentiability

The second instance analysis aims at proving the differentiability of the density of a guide $D_\theta$ with respect to its parameters $\theta$. It infers the continuous partial differentiability of multiple functions with respect to variables in the input state. The analysis is defined by the below abstraction:

$$
\mathcal{T}^\# \triangleq \mathcal{P}(\text{Var}) \times \mathcal{P}(\text{Var}) \times \mathcal{P}(\text{Var} \times \text{Var}), \quad \bot^\# \triangleq (\text{Var}, \text{Var}, \text{Var} \times \text{Var}), \\
\gamma(X, Y, R) \triangleq \{ g \in \mathcal{D} \mid \forall x \in X. \forall y \in Y. \ g(s, r) \neq \bot \implies s(x) = (g(s, r))_x(x) \}; \\
\wedge(y \in Y. \forall s, r. \ g(s, r) \neq \bot \implies \lambda u \in \mathbb{R}. \ \text{dens}(g, s[y \mapsto v])_r(r) \in C^1) \\
\wedge(\forall (z, u) \in R. \forall s, r. \ g(s, r) \neq \bot \implies \lambda v \in \mathbb{R}. \ g(s[z \mapsto v])_r(u) \in \mathbb{R}\text{-valued and } C^1) \}.
$$

By “$C^1$”, we mean that the relevant function is continuously differentiable. Being a $\mathbb{R}$-valued function in the last part requires that $g(s[z \mapsto v])_r$ be never $\bot$. An $(X, Y, R)$ in $\mathcal{T}^\#$ expresses a property of a transformer $g \in \mathcal{D}$ (which can be viewed as semantic command) that $g$ does not change variables in $X$, its density is $C^1$ with respect to each variable in $Y$ in the input state, and for each $(z, u) \in R$, it assigns a real value to $u$ in the output state in a $C^1$ manner with respect to $z$ in the input state.

We now define the abstract operations induced by this abstraction. Given an expression $E$, we let $\mathcal{V}(E)$ denote the set of variables that occur in $E$, and we write $C^1(E)$ for the set of variables with
respect to which \(|E|\) is \(C^1\) (based on classical differentiability rules). The definitions below follow from general principles such as the multivariate chain rule and account for discontinuities induced by conditions which break differentiability.

\[\text{widen(X₀, Y₀, R₀), (X₁, Y₁, R₁)} = (X₀ ∩ X₁, Y₀ ∩ Y₁, R₀ ∩ R₁)\]

\[\text{cond(E)} = (X₀ ∩ X₁, (Y₀ ∩ Y₁) \setminus \mathcal{V}(E), \{(z, u) ∈ R₀ \cap R₁ | z \notin \mathcal{V}(E) \lor u \in X₀ \cap X₁\})\]

\[\circ(\mathcal{E}(X₀, Y₀, R₀), (X₁, Y₁, R₁)) = (X₀ ∩ X₁, \{x ∈ Y₁ | \forall y \in \text{Var.} (x, y) ∈ R₁ \land y ∈ Y₀\}, \{(z, v) | \forall u \in \text{Var.} (z, u) ∈ R₁ \land (u, v) ∈ R₀\})\]

\[\text{skip} = (\text{Var}, \text{Var} \times \text{Var})\]

\[\text{update}(x, E) = (\text{Var} \setminus \{x\}, \text{Var}, \text{Var} \times (\text{Var} \setminus \{x\}) \cup \{(y, x) | y ∈ C^1(E)\})\]

\[\text{sample}(x, S, E₁, E₂) = (\text{Var} \setminus \{x\}, C^1(E₁) \setminus (\mathcal{V}(S) \cup \mathcal{V}(E₂)), (\text{Var} \times (\mathcal{V}(S) \cup \mathcal{V}(E₂)) \times \text{Var})\]

\[\text{score}(E₀, E₁, E₂) = (\text{Var}, C^1(E₀) \cap C^1(E₁)) \setminus \mathcal{V}(E₂), (\text{Var} \setminus \mathcal{V}(E₂)) \times \text{Var}\]

To discharge the differentiability requirement \(R₂\), we need to run this analysis on a guide \(D_θ\). If the \(Y\) component of the analysis result contains all the parameters \(θ\) (i.e., there exists \((X, Y, R)\) such that \(|D_θ| = (X, Y, R)\) and \(θ ⊆ Y\), then the requirement \(R₂\) is met.

### 7.4 Analysis for Condition (10)

The third analysis extends the second by tracking and checking more properties. Its aim is to prove the condition (10). Just like the second analysis, it infers information about the continuous partial differentiability of multiple functions involving the output state and the density. Also, it checks whether the density of a given command \(C_0\) has the form

\[d(e)(X₀, Y₀, R₀), (X₁, Y₁, R₁)) = (X₀ ∩ X₁, Y₀ ∩ Y₁, R₀ ∩ R₁)\]

\[\text{cond}(E) = (X₀ ∩ X₁, (Y₀ ∩ Y₁) \setminus \mathcal{V}(E), \{(z, u) ∈ R₀ \cap R₁ | z \notin \mathcal{V}(E) \lor u \in X₀ \cap X₁\})\]

\[\circ((X₀, Y₀, R₀), (X₁, Y₁, R₁)) = (X₀ ∩ X₁, \{x ∈ Y₁ | \forall y \in \text{Var.} (x, y) ∈ R₁ \land y ∈ Y₀\}, \{(z, v) | \forall u \in \text{Var.} (z, u) ∈ R₁ \land (u, v) ∈ R₀\})\]

\[\text{skip} = (\text{Var}, \text{Var} \times \text{Var})\]

\[\text{update}(x, E) = (\text{Var} \setminus \{x\}, \text{Var}, \text{Var} \times (\text{Var} \setminus \{x\}) \cup \{(y, x) | y ∈ C^1(E)\})\]

\[\text{sample}(x, S, E₁, E₂) = (\text{Var} \setminus \{x\}, C^1(E₁) \setminus (\mathcal{V}(S) \cup \mathcal{V}(E₂)), (\text{Var} \times (\mathcal{V}(S) \cup \mathcal{V}(E₂)) \times \text{Var})\]

\[\text{score}(E₀, E₁, E₂) = (\text{Var}, C^1(E₀) \cap C^1(E₁)) \setminus \mathcal{V}(E₂), (\text{Var} \setminus \mathcal{V}(E₂)) \times \text{Var}\]

To discharge the differentiability requirement \(R₂\), we need to run this analysis on a guide \(D_θ\). If the \(Y\) component of the analysis result contains all the parameters \(θ\) (i.e., there exists \((X, Y, R)\) such that \(|D_θ| = (X, Y, R)\) and \(θ ⊆ Y\), then the requirement \(R₂\) is met.

### 7.5 Analysis for Condition (9)

The last instance is a static analysis that aims at proving the condition (9). The analysis checks whether a given command \(C_0\) has a density of the following form:

\[\text{dens}(\mathcal{C}_0, s)(r) = \text{dens}(g, s)(r) = \sum_{i=1}^{M} \mathbb{1}_{r∈A_i} \left( \prod_{α∈K_i} \mathcal{N}(r(α); μ_i(α)(s, r), σ_i(α)(s, r)) \right) \left( \prod_{j∈[N_i]} \mathcal{N}(μ_i(α)(s, r), σ_i(α)(s, r)) \right)\]
We present a static analysis that can verify the support correspondence for Pyro model-guide pairs.

with probabilistic programming, which leads to more interesting examples.

Example 8.1 (Use of indexed random-variable names). Example 8.1 (Use of indexed random-variable names). The code excerpt (of a model or guide program in Pyro) below samples $N \times M$ instances of independent random variables, and names them with "$x_{1_1}$", "$x_{1_2}$", ",", "$x_{N_M}$".

```
for i in range(1,N+1):
    for j in range(1,M+1):
        val = pyro.sample("x_{}.{}".format(i,j), Normal(m,d))
```

Since Pyro is based on PyTorch and is adapted to implement data-science applications, Pyro programs heavily use multidimensional arrays, called tensors, and operations over them from PyTorch, in particular, element-wise operations and broadcasting. More precisely, when a binary operation (such as addition) is applied to two tensors of identical size, it outputs a new tensor of the same size where each output element is computed separately from the corresponding input elements. This principle often makes it possible to parallelise computations. Broadcasting occurs when a binary operation is applied to two tensors of different dimensions that can somehow be unified. Intuitively, it introduces and duplicates dimensions to produce tensors of identical dimensions, so that binary operations can be performed element-wise.
Towards Verified Stochastic Variational Inference for Probabilistic Programs

Tensors are also heavily used for sampling, which has several consequences. First, it means that an analysis targeted at Pyro programs should be able to track information about at least the dimensions of sampled tensors. Second, the dimensions of sampled tensors are grouped such that each of these groups has a different property with respect to probabilistic independence of tensor components. Pyro inference engines exploit this property for optimisation, for instance, via subsampling, but for this optimisation to be valid, the grouping of dimensions in a model should match that of a guide. Our analysis tracks information about dimension grouping of sampled tensors and the use of the Pyro construct called plate, which enables the optimisation just mentioned.

8.2 Abstract Domain

We extend our analysis from §7 so that it tracks not just how each part of a given program transforms with index sequences. If

where

Z

is our custom domain for zones, which express higher-dimensional rectangles in

. To emphasise this union interpretation, we write

for the subset. The following table summarises the definitions of our abstract domain, based on

:

An element

means a set of concrete random databases

for each concrete store

. The domain of

consists of names that are obtained by concatenating baseline strings

in

with index sequences. If

maps a baseline string

to

, it does not give any information about index sequence and a probability distribution used for the random variable. Otherwise, the second component of

specifies the information about the distribution, and the first component a region of

that contains the index sequences used in the names. The region is described by a finite subset

of

, which means the disjoint union of the rectangles represented by the

. To emphasise this union interpretation, we write

for the subset. The following table summarises the definitions of our abstract domain, based on

:

A higher-dimensional rectangular zone

is described by a finite sequence of intervals

, each of which is made of two bounds. A bound

may be defined as one or two constraints which express that this bound is equal to a constant, or to a variable plus a constant, or both. This intuitive denotation defines a concretisation function

:

Example 8.2. Consider the code of Example 8.1. After

iterations in the main loop and

iterations in the last execution of the inner loop, we expect the analysis to come up with the invariant, ["\(x\)" \(\mapsto ([1, i-1] \times [1, M] \cup [i, i] \times [1, j], \text{normal}(m, d))\)]. In turn, at the exit of the main loop, we expect the analysis to infer the invariant, ["\(x\)" \(\mapsto ([1, i=N] \times [1, j=M], \text{normal}(m, d))\)].

In addition to these constraints over random databases, our analyser also uses an abstraction that maintains typing information and numerical constraints over variables. We do not fully formalise these constraints as the overall structure of the domain relies on a classical reduced product.

Finally, we describe the combination of the above state abstraction with the abstraction of §7.2. More precisely, we start with the abstract domain exposed in §7.2 (which we denote by

) and...
build a novel abstract domain \( \mathcal{T}^\#_s \) that also satisfies the requirements of §7.1. We let the set of abstract elements be \( \mathcal{T}^\#_s \triangleq [\mathcal{RDB}^\# \rightarrow \mathcal{T}^\# \times \mathcal{RDB}^\#] \). Intuitively, such an element maps an input abstract random database into a set of functions together with an over-approximation of their output, when applied to this abstract input. The concretisation below formalises this: for all \( t_s \in \mathcal{T}^\#_s \),
\[
y_s(t_s) \triangleq \{ g \in \mathcal{D} \mid \forall r^\#_i \in \mathcal{RDB}^\#, t_s(r^\#_i) = (t, r^\#_o) \implies \exists g' \in y(t). \forall (s, r) \in y_r(r^\#_i). g(s, r) = g'(s, r) \land (g(s, r) = \bot \lor (g(s, r)_s, g(s, r)_r) \in y_r(r^\#_o)) \}.
\]

8.3 Computation of Loop Invariants
Although our static analysis for Pyro requires a state abstraction, its principles and structure are similar to those of the general analysis shown in §7. In the following, we first describe the integration of the state abstraction in the analysis of §7.2.

The abstract operations in \( \mathcal{T}^\#_s \) can all be derived by lifting those in \( \mathcal{T}^\# \) into functions. We illustrate this for the abstract composition \( \circ^\# \) for \( \mathcal{T}^\#_s \). Recall the operator \( \circ^\# \) for \( \mathcal{T}^\# \) in §7.2. Given \( t_s, t'_s \in \mathcal{T}^\#_s \),
\[
t_s \circ^\# t'_s \triangleq \lambda r^\#_0. \langle t \circ^\# t', r^\#_0 \rangle \text{ where } (t', r^\#_0) = t'_s(r^\#_0) \text{ and } (t, r^\#_2) = t_s(r^\#_1).
\]
The other abstract operators over \( \mathcal{T}^\#_s \) are defined in a similar manner.

As in most static analysis problems, the computation of precise loop invariants requires a carefully designed widening operator. In the case of zones, the analysis needs to generalise information about the bounds. We assume \( r^\#_0, r^\#_1 \in \mathcal{RDB}^\# \) are abstract random databases, and present the main steps in the computation of the widening \( \text{widen}(r^\#_0, r^\#_1) \).

(i) For each \( r^\#_i \), fuse zones \( Z^\#_0 \) and \( Z^\#_1 \) in \( r^\#_i \) into a single zone where \( Z^\#_0 \) and \( Z^\#_1 \) are the same except for one component and they represent adjacent high-dimensional rectangles. The following rewriting illustrates this step.
\[
\begin{align*}
\text{widen}(\{ x \mapsto [1, i-1] \times [1, j=M] \cup [i, i] \times [1, j=M], \text{ normal}(m, d) \}) \\
\rightarrow \{ x \mapsto [1, i] \times [1, j=M], \text{ normal}(m, d) \}
\end{align*}
\]

(ii) We generalise the intervals of corresponding zones in \( r^\#_0 \) and \( r^\#_1 \) using a weak version of unification on their bounds. The bounds of corresponding intervals in two such zones survive this step in a weakened form if they share at least one syntactically equal expression. Otherwise, the bounds are dropped, which causes the introduction of \( \bot \) into abstract random databases. The following rewriting instances illustrate this bound generalisation.
\[
\begin{align*}
\text{widen}(\{ x \mapsto [1, i] \times [1, j-1=M], \text{ normal}(m, d) \}) \\
\rightarrow \{ x \mapsto [1, i] \times [1, M], \text{ normal}(m, d) \}
\end{align*}
\]
\[
\begin{align*}
\text{widen}(\{ x \mapsto [1, i] \times [1, j-1], \text{ normal}(m, d) \}) \\
\rightarrow \{ x \mapsto [1, i] \times [1, j=M], \text{ normal}(m, d) \}
\end{align*}
\]
\[
\begin{align*}
\text{widen}(\{ x \mapsto [1, i] \times [1, j-1], \text{ normal}(m, d) \}) \\
\rightarrow \{ x \mapsto [1, i] \times [1, j=M], \text{ normal}(m, d) \}
\end{align*}
\]

The analysis also applies standard widening techniques to typing information and numerical constraints mentioned in §8.2. Finally, a model-guide pair can be verified if and only if their analyses return equivalent abstract random databases, without any name mapped into \( \bot \).

8.4 Experimental Evaluation
We have implemented a prototype analyser and carried out experiments so as to assess the effectiveness of our analysis to verify the support correspondence for Pyro model-guide pairs.\(^{13}\)

\(^{13}\)Code is available at https://github.com/wonyeol/static-analysis-for-support-match.
(i) Can our analysis discover incorrect model-guide pairs in realistic probabilistic programs?
(ii) Can the analysis verify correct model-guide pairs despite the complexity of the definition?
(iii) Is the analysis efficient enough so that it can cope with realistic probabilistic programs?

**Benchmark programs.** We took example programs from the Pyro webpage that can be handled by standard SVI engines or can be naturally rewritten to be so. Those engines do not use Pyro’s recent vectorised enumeration-based optimisation (enabled by the option `TraceEnum_ELBO`), and are run with the option `Trace_ELBO`. We made this choice because the optimisation is newly introduced and whether it is used or not changes the set of random variables used in a Pyro program.

We have applied our analysis to two groups of Pyro programs. The first is the *Pyro regression test suite* [Uber AI Labs 2019b], which comprises 66 examples exercising the basic programming patterns expected in Pyro programs. While each of these is small, they cover many standard patterns of both correct and incorrect model-guide pairs. Among these, 33 test cases use only `TraceEnum_ELBO` and fall outside the scope of our analysis, and 6 of the remaining 33 test cases come with two subsampling options. Hence, we can consider 39 experiments based on this regression suite.

The second is a collection of examples from the Pyro webpage [Uber AI Labs 2019a]. The webpage features 18 *Pyro examples*, among which 9 involve model-guide pairs (other examples use automatic guide generation, or do not perform SVI at all). Out of these 9 examples, 5 use `Trace_ELBO`, and three can be converted naturally into `Trace_ELBO` versions although they do not use `Trace_ELBO`. Thus, 8 Pyro examples fall within the scope of our analysis. These examples correspond to advanced probabilistic models14 from the machine-learning literature, most of which use sophisticated neural networks and probabilistic modelling. Table 1 describes the structure of these 8 examples in detail.

**Prototype analyser and results.** Our analyser is implemented in OCaml, and supports the main data-structures and operations defined by Python, PyTorch, and Pyro. In particular, it precisely abstracts the shape of PyTorch tensor objects, the shape transformation information of PyTorch neural-network-related objects, the automatic broadcasting information of Pyro plate objects, and the shape of allocated indices for sample names, using the zone abstraction described above. It also supports common Pyro probability distributions, and can precisely cope with standard Pyro and PyTorch operations manipulating the Pyro distribution objects and PyTorch tensor objects. While our prototype supports a wide range of Python, PyTorch, and Pyro features, we point out that we did not implement a static analysis for the full Python (plus PyTorch and Pyro) language (e.g., no support for classes, dictionaries, named tuples, and user-defined functions).

The analysis results are summarised in Table 2 and are discussed in detail in the following. Run-times were measured on an Intel Core i7-7700 machine running Linux Ubuntu 16.04.

**Discovery of invalid model-guide pairs.** The analysis rejected two Pyro examples, `br` and `lda`, as incorrect due to an invalid model-guide pair. `br` is the Bayesian regression example discussed in §2.

For `br`, the analysis discovers that a random variable `sigma` is sampled from `Uniform(0., 10.)` in the model `C`, but from `Normal(., .)` in the guide `D_θ` (Figure 2(a)). Since the support of `sigma` in `D_θ` is not a subset of that in `C` (i.e., `R ⊈ [0, 10]`), the requirement `R1` is violated. Thus, the SVI objective, $KL(D_θ || C)$, is undefined, and `br` has an invalid model-guide pair.

For `lda`, the analysis discovers that a random variable `doc_topics` is sampled from `Dirichlet(.)` in the model `C`, but from `Delta(.)` in the guide `D_θ`. Since the reference measures of `doc_topics` in `C` and `D_θ` are different (the Lebesgue measure vs. the counting measure), $KL(D_θ || C)$ cannot be computed by (4). For this reason, `lda` has an invalid model-guide pair. Our analyser tracks the

14The models include variational autoencoder (VAE) [Kingma and Welling 2014], semi-supervised VAE [Kingma et al. 2014], deep exponential family [Ranganath et al. 2015], attend-infer-repeat [Eslami et al. 2016], deep Markov model [Krishnan et al. 2017], inference compilation [Le et al. 2017], and amortised latent Dirichlet allocation [Srivastava and Sutton 2017].
Table 1. Key features of the model-guide pairs from Pyro examples. LoC denotes the lines of code of model and guide. The columns “Total #” show the number of objects/commands of each type used in model and guide, and the columns “Total dimension” show the total dimension of tensors in model and guide, either sampled from sample or used inside score, as well as the dimension of $\theta$ in guide.

<table>
<thead>
<tr>
<th>Name</th>
<th>Corresponding probabilistic model</th>
<th>Total #</th>
<th>Total dimension</th>
<th>LoC for</th>
<th>plate</th>
<th>sample</th>
<th>score</th>
<th>sample</th>
<th>score</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>br</td>
<td>Bayesian regression</td>
<td>27</td>
<td>0</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>10</td>
<td>170</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>csis</td>
<td>Inference compilation</td>
<td>31</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>480</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lda</td>
<td>Amortised latent Dirichlet allocation</td>
<td>76</td>
<td>0</td>
<td>5</td>
<td>8</td>
<td>1</td>
<td>21008</td>
<td>64000</td>
<td>121400</td>
<td></td>
</tr>
<tr>
<td>vae</td>
<td>Variational autoencoder (VAE)</td>
<td>91</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>25600</td>
<td>200704</td>
<td>353600</td>
<td></td>
</tr>
<tr>
<td>sgdef</td>
<td>Deep exponential family</td>
<td>94</td>
<td>0</td>
<td>8</td>
<td>12</td>
<td>1</td>
<td>231280</td>
<td>1310720</td>
<td>231280</td>
<td></td>
</tr>
<tr>
<td>dmm</td>
<td>Deep Markov model</td>
<td>246</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>640000</td>
<td>281600</td>
<td>594000</td>
<td></td>
</tr>
<tr>
<td>ssvae</td>
<td>Semi-supervised VAE</td>
<td>349</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>24000</td>
<td>156800</td>
<td>844000</td>
<td></td>
</tr>
<tr>
<td>air</td>
<td>Attend-infer-repeat</td>
<td>410</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>20736</td>
<td>160000</td>
<td>6040859</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Analysis results on two benchmark sets: Pyro test suite (Left) and Pyro examples (Right). The model-guide pairs from Pyro test suite are grouped into 7 categories, based on which type of plate objects are used. #Same (or #Diff) denotes the number of model-guide pairs for which the output of our analyser, valid or invalid, is the same as (or different from) the documented output. #Crash denotes the number of pairs for which our analyser crashes. The column “Time” shows the analysis time in seconds; on Left, it is averaged over those model-guide pairs (in each category) for which our analyser does not crash. The column “Valid?” shows the output of our analysis, valid or invalid.

<table>
<thead>
<tr>
<th>Category</th>
<th>#Same</th>
<th>#Diff</th>
<th>#Crash</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>No plates</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0.001</td>
</tr>
<tr>
<td>Single for-plate</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>0.004</td>
</tr>
<tr>
<td>Nested for-plates</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0.026</td>
</tr>
<tr>
<td>Single with-plate</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0.001</td>
</tr>
<tr>
<td>Nested with-plates</td>
<td>7</td>
<td>2</td>
<td>0</td>
<td>0.002</td>
</tr>
<tr>
<td>Non-nested with-plates</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0.002</td>
</tr>
<tr>
<td>Nested for-plate &amp; with-plate</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>N/A</td>
</tr>
<tr>
<td>Total</td>
<td>29</td>
<td>2</td>
<td>8</td>
<td>0.003</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Valid?</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>br</td>
<td>x</td>
<td>0.006</td>
</tr>
<tr>
<td>csis</td>
<td>o</td>
<td>0.007</td>
</tr>
<tr>
<td>lda</td>
<td>x</td>
<td>0.014</td>
</tr>
<tr>
<td>vae</td>
<td>o</td>
<td>0.005</td>
</tr>
<tr>
<td>sgdef</td>
<td>o</td>
<td>0.070</td>
</tr>
<tr>
<td>dmm</td>
<td>o</td>
<td>0.536</td>
</tr>
<tr>
<td>ssvae</td>
<td>o</td>
<td>0.013</td>
</tr>
<tr>
<td>air</td>
<td>o</td>
<td>4.093</td>
</tr>
</tbody>
</table>

Verification of probabilistic programs relying on model-guide pairs. Among the Pyro test suite, the analysis successfully verifies 31 examples among 39. Interestingly, two of these 31 successful validations, highlighted in Table 2(a), correspond to cases that were flagged as “invalid model-guide pairs” in the Pyro git repository. Upon inspection, these two examples turn out to be correct.

15 We thank an anonymous reviewer and Eli Bingham for pointing this out and explaining it in detail.
Towards Verified Stochastic Variational Inference for Probabilistic Programs

On the other hand, 8 examples from the Pyro test suite could not be verified due to the crashes of the analyser. One of these failures is due to the need to reason more precisely about the content of a `for` loop (e.g., using some partitioning techniques), and seven are due to the use of plates with subsampling, as ranges for `for` loops. Therefore, these failures could be resolved using existing static analysis techniques and a more precise handling of the semantics of Python constructions.

Moreover, 6 Pyro examples (among the 8 that we considered) were verified successfully, which means all correct Pyro examples were verified. Finally, we corrected the two examples that were rejected due to invalid model-guide pairs, and these two examples were also successfully verified.

Analysis efficiency. The analysis returned within a second on each program in the Pyro test suite, and on most of the Pyro examples. The slowest analysis was observed on `air`, which was analysed within 5 seconds. Most of the Pyro examples sample from and score with distributions of very high dimension arranged in complex tensors, using nested `for` and `plate`'s. While they are not large, they present a high degree of logical complexity, that is representative of realistic probabilistic programs. The fact that such programs get analysed within seconds shows that the analysis and the underlying abstract domain to describe zones, sampled dimensions, and distributions can generalise predicates quickly so that precise loop invariants can be computed.

9 RELATED WORK, CONCLUSION AND LIMITATION

Related work. As far as we know, the idea of using SVI for probabilistic programs first appeared in [Wingate and Weber 2013]. When further insights into how to create generic (sometimes also called black-box) SVI engines were found [Kucukelbir et al. 2015, 2017; Ranganath et al. 2014], the idea was tried for realistic probabilistic programming languages, such as Stan [Kucukelbir et al. 2015] and Anglican [van de Meent et al. 2016], resulting in impressive case studies [Kucukelbir et al. 2015]. The major inference engines for deep probabilistic programming languages [Bingham et al. 2019; Siddharth et al. 2017; Tran et al. 2018, 2016] are all based on SVI nowadays. However, we do not know of any prior work that attempts to reveal implicit assumptions made by these SVI engines and to discharge these assumptions manually or automatically, as we did in this paper.

The correctness of a different type of inference engines based on Monte-Carlo methods has been the subject of study in the PL community. Such engines have clearly formulated correctness conditions from Markov chain theory [Geyer 2011; Green 1995; Hastings 1970; Metropolis et al. 1953], such as ergodicity and correct stationarity. Tools from formal semantics have been employed to show that the inference engines satisfy these conditions [Borgström et al. 2016; Hur et al. 2015; Kiselyov 2016; Scibior et al. 2018]. While looking at different algorithms, some of these works consider more expressive languages than what we used in the paper, in particular, those supporting higher-order functions. One interesting direction is to extend our results to such expressive languages using the ideas from these works, especially the operational technique in [Borgström et al. 2016] and the denotational monad-based technique in [Scibior et al. 2018].

The consequence of having random choice in a programming language has been actively investigated by the semantics researchers from the early days [Borgström et al. 2016; Ehrhard et al. 2014; Heunen et al. 2017; Jones and Plotkin 1989; Kozen 1981; Smolka et al. 2017; Staton 2017; Staton et al. 2016; Toronto et al. 2015; Vákár et al. 2019]. Our work uses the technique developed in this endeavour, such as Giry monad and denotational formulation of idealised importance sampling [Staton et al. 2016]. Also, just as we connected the measure semantics with the density semantics, [Kozen 1981] and [Borgström et al. 2016] related two semantics with similar flavours, although the considered languages (with or without continuous distribution and score statements) and the style of semantics (operational vs denotational) are different. Our density semantics uses a reference measure built out of Lebesgue measure as in [Bhat et al. 2012, 2013; Hur et al. 2015]. This choice
is found to cause an issue when the score statement is treated not as a scoring mechanism but in terms of conditional expectation [Wu et al. 2018]. How to resolve this matter is still open.

Static analyses for probabilistic programming languages or languages with probabilistic choice typically attempt to check probabilistic properties [Chakarov and Sankaranarayanan 2013; Cousot and Monerau 2012; Monniaux 2000; Wang et al. 2018], or perform posterior inference [Chaganty et al. 2013; Wang et al. 2018], or find information useful for posterior inference [Nori et al. 2014]. For instance, [Monniaux 2000] defines a probabilistic abstract interpretation framework, which is applied to estimate the failure probability [Monniaux 2001]. More recently, [Cousot and Monerau 2012] sets up a general framework to design probabilistic abstract interpretations, which lift conventional static analysis tools to a probabilistic setup, and [Wang et al. 2018] proposes a framework for analysing probabilistic programs based on hyper-graphs with probabilistic actions on edges. Our program analyses aim at a different type of verification tasks, namely, proving the safety requirements imposed by the SVI engines. Static analyses for checking the continuity properties of programs are proposed in [Chaudhuri et al. 2010]. Some techniques used in those analyses may help track the kind of smoothness properties considered in this paper precisely.

**Conclusion.** In this paper, we have set up a semantic model of probabilistic programming languages that allows to reason about the correctness of subtle optimisation algorithms such as SVI. In particular, we have identified a set of conditions that guarantee that these algorithms will produce meaningful results. These conditions simplify the verification of correctness, and open the way towards the automation of this verification. To evaluate the effectiveness of our framework, we have completed the design of a static analysis to validate the correspondence assumptions for the model-guide pairs. Our analysis could verify nontrivial implementations as well as uncover nontrivial bugs. Such bugs would lead to incorrect results, which would be very difficult to diagnose.

**Limitation.** We described four static analyses (§7.2-§7.5) for discharging the requirements R1-R6, and developed one of the four analyses further (§8.2-§8.4) to build a prototype analyser for actual Pyro programs. Here we clarify the underlying assumptions and limitations of these analyses.

The static analysis for model-guide support match (§7.2) was implemented into our static analyser for Pyro (§8.4) but with additional extensions (§8.2-§8.3), so that it does not make the assumption in §6.1; the assumption was introduced mainly to develop analyses for the requirements R3-R6. Hence, our static analyser handles both continuous and discrete random variables.

On the other hand, other analyses (§7.3-§7.5) are unimplemented and make the assumption in §6.1. We expect that the assumption can be relaxed, without much difficulty, to permit continuous or discrete distributions having finite entropy and moments of all degrees, because our proofs of theorems and lemmas bound various expectations mostly by entropies and moments. It would be more challenging, however, to relax the assumption to allow distributions not having finite entropy and moments of all degrees, or models having unboundedly many random variables (of any kind). In particular, addressing the later limitation might require techniques developed for proving that probabilistic systems has finite expected execution time.

This paper focuses on a particular optimisation objective (5) for SVI, but we point out that other objectives have been proposed for different inference or learning algorithms, such as variational EM (e.g., lda in §8.4) and importance weighted autoencoder [Burda et al. 2016]. One interesting research direction is to develop techniques for verifying the well-definedness of these objectives.

**ACKNOWLEDGMENTS**

We thank Fritz Obermeyer and Eli Bingham for explaining the subtleties of Pyro and suggesting us to try the Pyro regression test suite. Sam Staton, Ohad Kammar and Matthijs Vákár helped us to understand the semantics of recursion in the probabilistic programming languages better. Lee, Yang
and Yu were supported by the Engineering Research Center Program through the National Research Foundation of Korea (NRF) funded by the Korean Government MSIT (NRF-2018R1A5A1059921), and also by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT (2017M3C4A7068177).

REFERENCES


Towards Verified Stochastic Variational Inference for Probabilistic Programs


A PROOFS OF LEMMAS AND THEOREMS IN §4

Lemma 4.1. For all expressions $E$, $B$, and $S$, their semantics $⟦E⟧$, $⟦B⟧$ and $⟦S⟧$ are measurable functions from $\text{Store} \to \mathbb{R}$, $\mathbb{B}$ and $\text{Str}$, respectively.

Proof. We prove the lemma by structural induction. We present the proof for $E$ only, because those for $B$ and $S$ are similar. When $E$ is a constant, $⟦E⟧$ is a constant function and so it is measurable. When $E$ is a variable, $⟦x⟧$ is a projection of the $x$ component of the given store $s$. Since $\text{Store}$ is given the product $\sigma$-algebra, the projection $⟦x⟧$ is measurable. Finally, assume that $E$ is $f(E_0, \ldots, E_{n-1})$. Then, by induction hypothesis, the $⟦E_i⟧$ are measurable functions from $\text{Store} \to \mathbb{R}$. Thus,

$$
\lambda s. (⟦E_0⟧s, \ldots, ⟦E_{n-1}⟧s) : \text{Store} \to \mathbb{R}^n
$$

is measurable. Furthermore, $⟦f⟧ : \mathbb{R}^n \to \mathbb{R}$ is measurable. Thus, their composition is measurable as well. This implies the measurability of $⟦E⟧$ because

$$
⟦E⟧ = ⟦f⟧ \circ (\lambda s. (⟦E_0⟧s, \ldots, ⟦E_{n-1}⟧s)).
$$

\[\square\]

Lemma 4.2. $(\mathcal{K}, \sqsubseteq)$ is an $\omega$-complete partial order with the least element $\bot$.

Proof. The fact that $\sqsubseteq$ is a partial order immediately follows from its definition. The least element is the function $\lambda \sigma. \lambda A. 0$. It remains to show the $\omega$-completeness property. Consider an $\omega$-chain

$$
\kappa_0 \sqsubseteq \kappa_1 \sqsubseteq \ldots
$$

Let $\kappa = \sup_n \kappa_n$. We will show that $\kappa$ is the desired upper bound.

Pick a state $\sigma$. Then, $\kappa(\sigma)(A)$ is non-negative for all measurable subsets $A$, and $\kappa(\sigma)(\emptyset) = 0$. Furthermore, for every countable collection $\{A_j\}_j$ of disjoint measurable subsets of $\text{State} \times [0, \infty)$,

$$
\kappa(\sigma)(\bigcup_j A_j) = \sup_n \kappa_n(\sigma)(\bigcup_j A_j) = \lim_n \kappa_n(\sigma)(\bigcup_j A_j) = \lim_n \sum_j \kappa_n(\sigma)(A_j) = \sum_j \lim_n \kappa_n(\sigma)(A_j) = \sum_j \sup_n \kappa_n(\sigma)(A_j) = \sum_j \kappa(\sigma)(A_j).
$$

The exchange between sup and lim in the derivation uses the fact that the sequence involved is increasing. The third equality follows from the countable additivity of $\kappa_n(\sigma)$, and the fourth from the monotone convergence theorem applied to the counting measure on the set of natural numbers. Finally,

$$
\kappa(\sigma)(\text{State} \times [0, \infty)) = \sup_n \kappa_n(\sigma)(\text{State} \times [0, \infty)) \leq 1.
$$

We have just shown that $\kappa(\sigma)$ is a subprobability measure.

To show the measurability of $\kappa$, we use the fact that a map $f$ from a measurable space $(X, \Sigma)$ to $\text{Sp}(\text{State} \times [0, \infty))$ is measurable if and only if for all measurable subsets $A$ of $\text{State} \times [0, \infty)$ and reals $r \in \mathbb{R}$,

$$
f^{-1}(\{\mu \mid \mu(A) \leq r\}) \in \Sigma.
$$

Pick a measurable subset $A$ of $\text{State} \times [0, \infty)$ and a real $r \in \mathbb{R}$. Then,

$$
\kappa^{-1}(\{\mu \mid \mu(A) \leq r\}) = \{\sigma \mid \kappa(\sigma)(A) \leq r\}
$$
We will show these properties one by one. The first and the last equalities use the definition of integration, the second the characterisation of functions $h$.

Let us start with the proof of the first property. Pick a measurable subset $A$ of State $\times [0, \infty)$. For all non-negative measurable functions $h : \text{State} \times [0, \infty) \to [0, \infty)$ and all states $\sigma$,

$$\int \left( \bigcup_n \kappa_n \right) (\sigma)(d(\sigma', w')) h(\sigma', w') = \lim_n \int \kappa_n(\sigma)(d(\sigma', w')) h(\sigma', w').$$

**Proof.** We prove it by a simpler version of the so called monotone class theorem. This version says that in order to show the claimed equality for all non-negative measurable $h$, it suffices to show the following properties:

- The claimed equality holds if $h$ is the indicator function $\lambda(\sigma, w). \mathbb{1}_{[(\sigma, w) \in A]}$ for a measurable subset $A$ of State $\times [0, \infty)$.
- If non-negative measurable $h_1$ and $h_2$ satisfy the claimed equality and $r_1, r_2$ are non-negative real numbers, the linear combination $r_1 \cdot h_1 + r_2 \cdot h_2$ also satisfies the equality.
- If $\{h_m\}_m$ is a countable family of non-negative measurable functions such that each $h_m$ satisfies the claimed equality and $h_m \leq h_{m+1}$ for every $m$, then $\sup_m h_m$ also satisfies the claimed equality.

We will show these properties one by one.

Let us start with the proof of the first property. Pick a measurable subset $A$ of State $\times [0, \infty)$. Then,

$$\int \left( \bigcup_n \kappa_n \right) (\sigma)(d(\sigma', w')) \mathbb{1}_{[(\sigma', w') \in A]} = \left( \bigcup_n \kappa_n \right)(\sigma)(A) = \sup_n (\kappa_n(\sigma)(A)) = \lim_n (\kappa_n(\sigma)(A))$$

$$\int \kappa_n(\sigma)(d(\sigma', w')) \mathbb{1}_{[(\sigma', w') \in A]} = \lim_n \int \kappa_n(\sigma)(d(\sigma', w')) \mathbb{1}_{[(\sigma', w') \in A]}.$$

The first and the last equalities use the definition of integration, the second the characterisation of the least upper bound in $\mathcal{K}$, and the third the fact that $\{\kappa_n(\sigma)(A)\}_n$ is increasing.

For the second property, assume that the claimed equality holds for non-negative measurable functions $h_1$ and $h_2$. Let $r_1$ and $r_2$ be non-negative real numbers. Then,

$$\int \left( \bigcup_n \kappa_n \right)(\sigma)(d(\sigma', w')) (r_1 h_1(\sigma', w') + r_2 h_2(\sigma', w'))$$

$$= r_1 \int \left( \bigcup_n \kappa_n \right)(\sigma)(d(\sigma', w')) h_1(\sigma', w') + r_2 \int \left( \bigcup_n \kappa_n \right)(\sigma)(d(\sigma', w')) h_2(\sigma', w')$$

$$= r_1 \lim_n \int \kappa_n(\sigma)(d(\sigma', w')) h_1(\sigma', w') + r_2 \lim_n \int \kappa_n(\sigma)(d(\sigma', w')) h_2(\sigma', w')$$

$$= \lim_n \left( r_1 \int \kappa_n(\sigma)(d(\sigma', w')) h_1(\sigma', w') + r_2 \int \kappa_n(\sigma)(d(\sigma', w')) h_2(\sigma', w') \right)$$

$$= \lim_n \int \kappa_n(\sigma)(d(\sigma', w')) (r_1 h_1(\sigma', w') + r_2 h_2(\sigma', w')).$$
The first and the last equalities use the linearity of integration, the second the assumption on \( h_1 \) and \( h_2 \), and the third the continuity of constant multiplication and addition.

For the third property, consider an increasing sequence \( \{ h_m \} \) in the property. Then,

\[
\int \left( \bigcup_n \kappa_n \right)(\sigma)(d(\sigma', w')) \left( \sup_m h_m(\sigma', w') \right) = \int \left( \bigcup_n \kappa_n \right)(\sigma)(d(\sigma', w')) \sup_m h_m(\sigma', w') \\
= \int \left( \bigcup_n \kappa_n \right)(\sigma)(d(\sigma', w')) \lim_m h_m(\sigma', w') \\
= \lim_m \int \left( \bigcup_n \kappa_n \right)(\sigma)(d(\sigma', w')) h_m(\sigma', w') \\
= \lim \lim_m \int \kappa_n(\sigma)(d(\sigma', w')) h_m(\sigma', w').
\]

The third equality uses the monotone convergence theorem, and the fourth the assumption on the \( h_m \).

**Theorem 4.3.** For every command \( C \), its interpretation \( \| C \| \) is well-defined and it belongs to \( \mathcal{K} \).

**Proof.** We prove the theorem by induction on the structure of \( C \). In the proof, we use concepts and results from the measure theory that go beyond what we reviewed in §3.

We start with the case \( C \equiv (x := E) \). For every \( \sigma \in \text{State} \), \( \{ x := E \}(\sigma) \) is the Dirac measure at \( (\sigma[x \mapsto \{ E \} \sigma_3], 1) \). Thus, it suffices to show the measurability of the following function \( F \) for every measurable subset \( A \):

\[
F(\sigma) = 1_{\{ (\sigma[x \mapsto \{ E \} \sigma_3], 1) \in A \}}.
\]

This holds because the functions

\[
(\sigma, w) \mapsto 1_{\{ (\sigma, w) \in A \}} : \text{State} \times [0, \infty) \to [0, 1], \quad \sigma \mapsto (\sigma[x \mapsto \{ E \} \sigma_3], 1) : \text{State} \to \text{State} \times [0, \infty)
\]

are both measurable.

The case \( C \equiv \text{skip} \) follows from our argument for the previous case \( C \equiv (x := E) \), because \( \| \text{skip} \| = \{ x := x \} \).

Next we consider sequential composition. Assume that \( C \equiv (C_0; C_1) \). By induction hypothesis, we know that \( \| C_0 \| \) and \( \| C_1 \| \) are subprobability kernels. Then, for all \( \sigma \) and \( A \),

\[
\| C_0; C_1 \| (\sigma)(A) = \int \| C_0 \| (\sigma)(d(\sigma', w')) \int \| C_1 \| (\sigma')(d(\sigma'', w'')) 1_{\{ (\sigma'', w''), 1 \in A \}}
\]

\[
\leq \int \| C_0 \| (\sigma)(d(\sigma', w')) \leq 1.
\]

Furthermore, it is not difficult to show that \( \| C_0; C_1 \| (\sigma) \) is a measure. The measurability of \( \| C_0; C_1 \| \) follows from three facts. First, it suffices to show that for all measurable subsets \( A \), the function

\[
\sigma \mapsto \| C_0; C_1 \| (\sigma)(d(\sigma', w'))(A)
\]

is measurable. Second, for every measurable \( f \) and every subprobability kernel \( \kappa \), the function \( x \mapsto \int \kappa(x)(dy) f(y) \) is measurable. Second, the functions \( (\sigma, w) \mapsto \| C_1 \| (\sigma) \otimes \delta_w \) and \( ((\sigma'', w''), w) \mapsto (\sigma'', w', w'') \) are measurable. Here \( \delta_w \) is the Dirac measure at \( w \) and \( \otimes \) is the operator for product measure.

We move on to the case of \( C \equiv (B \{ C_0 \} \text{ else } \{ C_1 \}) \). We first show that the function \( \| C \| \) is a well-defined function from \( \text{State} \) to \( \text{Sp} (\text{State} \times [0, \infty)) \), that is, \( \| C \| (\sigma) \) is a subprobability measure.
for every $\sigma \in \text{State}$. The non-negativity and countable additivity of $\| C \| (\sigma)$ follow from the same properties of $\| C_0 \| (\sigma)$ and $\| C_1 \| (\sigma)$. Also,

$$\| C \| (\sigma)(A) = \| \text{if } B \{ C_0 \} \text{ else } \{ C_1 \} \| (\sigma)(A)$$

$$= \mathbb{1}_{\| \{B\}_{\sigma=true}\}} \cdot \| C_0 \| (\sigma)(A) + \mathbb{1}_{\| \{B\}_{\sigma\neq\true}\}} \cdot \| C_1 \| (\sigma)(A)$$

$$\leq \mathbb{1}_{\| \{B\}_{\sigma=true}\}} \cdot 1 + \mathbb{1}_{\| \{B\}_{\sigma\neq\true}\}} \cdot 1 = 1.$$ 

It remains to show that $\| C \|$ is measurable. This boils down to showing that

$$\{ \sigma \mid \mathbb{1}_{\| \{B\}_{\sigma=true}\}} \cdot \| C_0 \| (\sigma)(A) + \mathbb{1}_{\| \{B\}_{\sigma\neq\true}\}} \cdot \| C_1 \| (\sigma)(A) \leq r \}$$

is measurable for all $r \in \mathbb{R}$ and measurable subset $A$ of $\text{State} \times [0, \infty)$. But this follows easily from the induction hypothesis and the measurability of $\| B \|$.

The next case is $C \equiv (\text{while } B \{ C_0 \})$. Recall Lemma 4.2, which says that the set of subprobability kernels from $\text{State}$ to $\text{State} \times [0, \infty)$, denoted by $\mathcal{K}$, is an $\omega$-cpo when given the pointwise order $\subseteq$. Let $F$ be the function on $\mathcal{K}$ used in the semantics of $C$. That is, for all $\kappa \in \mathcal{K}$, $\sigma \in \text{State}$ and measurable $A \subseteq \text{Store} \times [0, \infty)$,

$$F(\kappa)(\sigma)(A) = \mathbb{1}_{\| \{B\}_{\sigma\neq\true}\}} \cdot \mathbb{1}_{\{[\sigma, 1] \in A\}}$$

$$+ \mathbb{1}_{\| \{B\}_{\sigma=true}\}} \cdot \int \| C_0 \| (\sigma)(d(\sigma', w')) \int \kappa(\sigma')(d(\sigma'', w'')) \mathbb{1}_{\{[\sigma'', w' \cdot w''] \in A\}}.$$

It suffices to show that $F$ is a continuous function on $\mathcal{K}$. The arguments used to handle the if and sequencing cases show that $F(\kappa)$ is a subprobability kernel. The monotonicity of $F$ follows from the fact that the integration is monotone with respect to the measure and its integrand. It remains to show that $F$ preserves the limits of $\omega$-chains. Consider an $\omega$-chain $\{\kappa_n\}_n$ of subprobability kernels in $\mathcal{K}$. Then, the least upper bound of this $\omega$-chain is $\kappa = \bigsqcup_n \kappa_n$. The following calculation shows that $F$ preserves this limit.

$$F(\kappa)(\sigma)(A)$$

$$= \mathbb{1}_{\| \{B\}_{\sigma\neq\true}\}} \cdot \mathbb{1}_{\{[\sigma, 1] \in A\}}$$

$$+ \mathbb{1}_{\| \{B\}_{\sigma=true}\}} \cdot \int \| C_0 \| (\sigma)(d(\sigma', w')) \int \kappa_n(\sigma')(d(\sigma'', w'')) \mathbb{1}_{\{[\sigma'', w' \cdot w''] \in A\}}$$

$$+ \mathbb{1}_{\| \{B\}_{\sigma\neq\true}\}} \cdot \mathbb{1}_{\{[\sigma, 1] \in A\}}$$

$$+ \mathbb{1}_{\| \{B\}_{\sigma=true}\}} \cdot \int \| C_0 \| (\sigma)(d(\sigma', w')) \lim_n \kappa_n(\sigma')(d(\sigma'', w'')) \mathbb{1}_{\{[\sigma'', w' \cdot w''] \in A\}}$$

$$+ \mathbb{1}_{\| \{B\}_{\sigma\neq\true}\}} \cdot \mathbb{1}_{\{[\sigma, 1] \in A\}}$$

$$+ \mathbb{1}_{\| \{B\}_{\sigma=true}\}} \cdot \lim_n \int \| C_0 \| (\sigma)(d(\sigma', w')) \kappa_n(\sigma')(d(\sigma'', w'')) \mathbb{1}_{\{[\sigma'', w' \cdot w''] \in A\}}$$

$$= \lim_n \left( \mathbb{1}_{\| \{B\}_{\sigma\neq\true}\}} \cdot \mathbb{1}_{\{[\sigma, 1] \in A\}} 

+ \mathbb{1}_{\| \{B\}_{\sigma=true}\}} \cdot \int \| C_0 \| (\sigma)(d(\sigma', w')) \kappa_n(\sigma')(d(\sigma'', w'')) \mathbb{1}_{\{[\sigma'', w' \cdot w''] \in A\}} \right)$$

$$= \lim_n F(\kappa_n)(\sigma)(A) = \left( \bigsqcup_n F(\kappa_n) \right)(\sigma)(A).$$
The third equality uses the continuity of integration with respect to the integrating measure and the non-negative integrand (Lemma A.1), and the fourth equality uses the monotone convergence theorem.

The next case is \( C \equiv (x := \text{sample}_{\text{norm}}(S, E_1, E_2)) \). The requirement that
\[
\llbracket x := \text{sample}_{\text{norm}}(S, E_1, E_2) \rrbracket(\sigma)
\]
is a measure for every \( \sigma \) follows immediately from the semantic equation for this sample statement, which says that \( \llbracket x := \text{sample}_{\text{norm}}(S, E_1, E_2) \rrbracket(\sigma) \) is either the constant-0 measure or the pushforward of a normal distribution with mean \( \llbracket E_1 \rrbracket \sigma_s \) and standard deviation \( \llbracket E_2 \rrbracket \sigma_s \) by some measurable function. In fact, the following calculation shows that \( \llbracket x := \text{sample}_{\text{norm}}(S, E_1, E_2) \rrbracket \) is a subprobability measure:
\[
\llbracket x := \text{sample}_{\text{norm}}(S, E_1, E_2) \rrbracket(\sigma)(A)
= \mathbb{1}_{\llbracket S \rrbracket \sigma_s \notin \text{dom}(\sigma_r)} \cdot \mathbb{1}_{\llbracket E_2 \rrbracket \sigma_s \notin (0, \infty)} \cdot \int d\nu \left( \mathcal{N}(\nu; \llbracket E_1 \rrbracket \sigma_s, \llbracket E_2 \rrbracket \sigma_s) \cdot \mathbb{1}_{[(\sigma_s, |x|)\rightarrow \nu], \sigma_s[\llbracket E_2 \rrbracket \sigma_s, \nu, 1] \in A} \right)
\leq \mathbb{1}_{\llbracket S \rrbracket \sigma_s \notin \text{dom}(\sigma_r)} \cdot \mathbb{1}_{\llbracket E_2 \rrbracket \sigma_s \notin (0, \infty)} \cdot 1
\leq 1.
\]

Also, since all of \( \mathcal{N}, \llbracket S \rrbracket, \llbracket E_0 \rrbracket \) and \( \llbracket E_1 \rrbracket \) are measurable,
\[
\{ \sigma \mid \llbracket x := \text{sample}_{\text{norm}}(S, E_1, E_2) \rrbracket(\sigma)(A) \leq r \}
\]
is a measurable subset of \( \text{State} \) for all \( A \) and \( r \). This means that \( \llbracket x := \text{sample}_{\text{norm}}(S, E_1, E_2) \rrbracket \) is a measurable function. From what we have proved, it follows that \( \llbracket x := \text{sample}_{\text{norm}}(S, E_1, E_2) \rrbracket \) is a subprobability kernel as required.

The last case is \( C \equiv \text{score}_{\text{norm}}(E_0, E_1, E_2) \). Because of the semantic equation for this score statement, \( \llbracket \text{score}_{\text{norm}}(E_0, E_1, E_2) \rrbracket(\sigma) \) is a measure for all \( \sigma \). Furthermore, \( \llbracket \text{score}_{\text{norm}}(E_0, E_1, E_2) \rrbracket(\sigma)(A) \) is the product of two indicator functions, and so it is at most 1. That is, \( \llbracket \text{score}_{\text{norm}}(E_0, E_1, E_2) \rrbracket(\sigma) \) is a subprobability measure. By the semantic equation for the score statement again, the measurability of the normal density and the measurability of \( E_0, E_1 \) and \( E_2 \), the set \( \{ \sigma \mid \llbracket \text{score}_{\text{norm}}(E_0, E_1, E_2) \rrbracket(\sigma)(A) \leq r \} \) is measurable for all \( A \) and \( r \). This measurability implies that \( \llbracket \text{score}_{\text{norm}}(E_0, E_1, E_2) \rrbracket \) is a measurable function. By putting all these together, we can derive the required fact that \( \llbracket \text{score}_{\text{norm}}(E_0, E_1, E_2) \rrbracket \) is a subprobability kernel.

**Lemma 4.4.** For every measurable \( h : \text{RDB} \times \text{RDB} \times \text{RDB} \rightarrow \mathbb{R} \), the function \( (r, r') \mapsto \mathbb{1}_{[r \neq r']} \times h(r, r', r \cup r') \) from \( \text{RDB} \times \text{RDB} \) to \( \mathbb{R} \) is measurable.

**Proof.** Let \( * : \text{RDB} \times \text{RDB} \rightarrow \text{RDB} \) be the following extension of the partial function \( \cup : \text{RDB} \times \text{RDB} \rightarrow_{\text{partial}} \text{RDB} \) to a total function:
\[
r \ast r' = \begin{cases} r \cup r' & \text{if } r \neq r', \\
\emptyset & \text{otherwise}.
\end{cases}
\]

Then, for all \( r, r' \),
\[
\mathbb{1}_{[r \neq r']} \cdot h(r, r', r \cup r') = \mathbb{1}_{[r \neq r']} \cdot h(r, r', r \ast r')
\]
We will show that \( (r, r') \mapsto \mathbb{1}_{[r \neq r']} \) and \( (r, r') \mapsto r \ast r' \) are measurable. This is enough because the function on the RHS of the above equation (namely, \( (r, r') \mapsto \mathbb{1}_{[r \neq r']} \cdot h(r, r', r \ast r') \)) is constructed by applying pairing, composition and multiplication operators on measurable functions and so it is measurable.
Pick a real number $v \in \mathbb{R}$. We should show that $B = \{(r, r') \mid 1_{[r \# r']} \leq v\}$ is measurable. If $v \geq 1$, the set $B$ is $RDB \times RDB$, and so it is measurable. If $v < 0$, then $B$ is the empty set, so that it is measurable as well. If $0 \leq v < 1$, then

$$
\{(r, r') \mid 1_{[r \# r']} \leq v\} = \bigcup_{K \subseteq \text{Str}} \bigcup_{K' \subseteq \text{Str}} \{K \rightarrow \mathbb{R}\} \times \{K' \rightarrow \mathbb{R}\},
$$

which is measurable because the RHS of the above equation is the countable union of measurable sets.

Pick a measurable subset $A$ of $RDB$. We have to prove that $B = \{(r, r') \mid r \# r' \in A\}$ is measurable. If $A = \{\emptyset\}$, then $B = \{(r, r') \mid 1_{[r \# r']} \leq 0.5\} \cup \{([], [])\}$, so that it is measurable. If $\emptyset \notin A$, then

$$
\{(r, r') \mid r \# r' \in A\} = \bigcup_{K \subseteq \text{Str}} \bigcup_{K' \subseteq \text{Str}} \{(r, r') \in \{K \rightarrow \mathbb{R}\} \times \{K' \rightarrow \mathbb{R}\} \mid r \uplus r' \in \{r'' \in A \mid \text{dom}(r'') = K \cup K'\}\}.
$$

But $\{r'' \in A \mid \text{dom}(r'') = K \cup K'\}$ is measurable, and the set on the RHS of the above equation is the countable union of measurable sets. Thus, $B$ is measurable in this case as well. The remaining case is that $A$ contains $\emptyset$ and some other element. We can prove this case by splitting $A$ into $\{\emptyset\}$ and $A \setminus \{\emptyset\}$ and using what we proved in the previous two cases.

**Lemma 4.5.** ($\mathcal{D}, \sqsubseteq$) is an $\omega$-complete partial order and has the least element $a \mapsto \bot$. Thus, every continuous function $G$ on $\mathcal{D}$ has the least fixed point.

**Proof.** All constant functions from a measurable space to another are measurable. Thus, the constant function $a \mapsto \bot$ is measurable. It is immediate that this constant function is local and smaller than any other functions in $\mathcal{D}$.

Let $g_0 \leq g_1 \leq g_2 \leq \ldots$ be an $\omega$-chain. We show that the following function is the least upper bound of the sequence in $\mathcal{D}$: for all $s, r$,

$$(\bigsqcup_i g_i)(s, r) = \begin{cases} 
\bot & \text{if } g_i(s, r) = \bot \text{ for all } i, \\
c \neq \bot & \text{if } g_i(s, r) = c \neq \bot \text{ for some } i.
\end{cases}
$$

By definition, this function is the least upper bound in $\mathcal{D}$ that we are looking for, if it is measurable and local. To check the locality, suppose that $(\bigsqcup_i g_i)(s, r) = (s', r', w', p')$. Then, there exists some $i$ such that $g_i(s, r) = (s', r', w', p')$. By the locality of $g_i$, we have that

$$(\exists r'' \cdot r' \# r'' \land r = r' \uplus r'' \land g_i(s, r'') = (s', [], w', p'))
\land (\forall r'' \cdot r' \# r'' \implies g_i(s, r' \uplus r'') = (s', r' \uplus r'', w', p')).$$

Thus, the definition of $\bigsqcup_i g_i$ implies that the above property holds when we replace $g_i$ by $\bigsqcup_i g_i$. We have just shown that $\bigsqcup_i g_i$ is local.

It remains to show that $\bigsqcup_i g_i$ is measurable. Let $A \subseteq \{\bot\} \cup (\text{Store} \times RDB \times [0, \infty) \times [0, \infty))$ be a measurable set. We prove that $(\bigsqcup_i g_i)^{-1}(A)$ is a measurable set.

- Case $A = \{\bot\}$:

$$
(s, r) \in \left(\bigsqcup_i g_i\right)^{-1}(\{\bot\}) \iff g_i(s, r) = \bot \text{ for all } i \iff (s, r) \in g_i^{-1}(\{\bot\}) \text{ for all } i \iff (s, r) \in \bigsqcup_i g_i^{-1}(\{\bot\}).
$$
We say that \(16:40\) Wonyeol Lee, Hangyeol Yu, Xavier Rival, and Hongseok Yang

\[ D \]

The \(\omega\)-cpo \(D\) permits an operator for composing functions in it. We say that a function

\[ g : \left(\{\bot\} \cup (\text{Store} \times RDB \times [0, \infty) \times [0, \infty))\right) \rightarrow \left(\{\bot\} \cup (\text{Store} \times RDB \times [0, \infty) \times [0, \infty))\right) \]

is local if for all \((s, r, w, p), (s', r', w', p') \in \text{Store} \times RDB \times [0, \infty) \times [0, \infty),\)

\[ g(s, r, w, p) = (s', r', w', p') \implies (\exists r'' : r'' \# r' \land g(s, r'', w, p) = (s', [\_], w', p')) \land (\forall r' : r' \# r'' \implies g(s, r \cup r'', w, p) = (s', r' \cup r'', w', p')). \]

We say that \(g\) is strict if \(g(\bot) = \bot\). Let

\[ D^\bot \triangleq \left\{ g : \left(\{\bot\} \cup (\text{Store} \times RDB \times [0, \infty) \times [0, \infty))\right) \rightarrow \left(\{\bot\} \cup (\text{Store} \times RDB \times [0, \infty) \times [0, \infty))\right) \mid g \text{ is measurable, local and strict} \right\}. \]

Define a partial order \(\sqsubseteq\) on \(D^\bot\) by

\[ g \sqsubseteq g' \iff \forall a \in \{\bot\} \cup (\text{Store} \times RDB \times [0, \infty) \times [0, \infty)) \cdot (g(a) = \bot \lor g(a) = g'(a)). \]

**Lemma A.2.** \((D^\bot, \sqsubseteq)\) is an \(\omega\)-complete partial order and has the least element.

**Proof.** Let

\[ E \triangleq \left\{ g : \left(\text{Store} \times RDB \times [0, \infty) \times [0, \infty)\right) \rightarrow \left(\{\bot\} \cup (\text{Store} \times RDB \times [0, \infty) \times [0, \infty)\right) \mid g \text{ is measurable and local} \right\}. \]

Essentially by the same argument as the one used in the proof of Lemma 4.5, the set \(E\) with the pointwise order \(\sqsubseteq\) is an \(\omega\)-cpo and has the least element. But \((D^\bot, \sqsubseteq)\) and \((E, \sqsubseteq)\) are isomorphic as partially order sets. Thus, \((D^\bot, \sqsubseteq)\) has the property claimed by the lemma. \(\square\)

The following lemma says that the lifting \(g^\bot\) of \(g \in D\) in §4.3 is a well-defined function from \(D\) to this new domain.

**Lemma A.3.** The function \(g \mapsto g^\bot\) from \(D\) to \(D^\bot\) is well-defined and continuous.

Towards Verified Stochastic Variational Inference for Probabilistic Programs

Proof. We first show that the function is well-defined. Let \( g \in \mathcal{D} \). Then, \( g^\dagger(\bot) = \bot \) by the definition of \( g^\dagger \). The measurability of \( g^\dagger \) follows from the measurability of \( g \), projection, and multiplication and the fact that both the pairing of measurable functions (into a function to a product measurable space) and the function composition preserve measurability. The locality of \( g^\dagger \) follows from that of \( g \).

Next, we show the claimed continuity. The monotonicity is an immediate consequence of the definitions of the \( -^\dagger \) operator and the orders on \( \mathcal{D} \) and \( \mathcal{D}^\dagger \). It remains to show the limit preservation. Let \( g_0 \subseteq g_1 \subseteq g_2 \subseteq \ldots \) be an \( \omega \)-chain in \( \mathcal{D} \). Then, by the monotonicity of \( -^\dagger \), \( \{g_n^\dagger\}_n \) is also an \( \omega \)-chain in \( \mathcal{D}^\dagger \). We should show that

\[
\left( \bigsquare \ g_n \right)^\dagger(a) = \left( \bigsquare \ g_n^\dagger \right)(a)
\]

for all \( a \) from \( (\{\bot\} \cup \text{Store} \times \text{RDB} \times [0,\infty) \times [0,\infty)) \). Pick such an \( a \). If \( a = \bot \), the required holds because both sides become \( \bot \). If \( a \neq \bot \) but \( g_n(a) = \bot \) for all \( n \), then again both sides of the equality are \( \bot \), and so the equality holds. If neither of these two cases hold, there exists \( n \) such that \( g_n(a) \neq \bot \) and \( g_n(a) = g_m(a) \) for all \( m \geq n \). Let \((s',r',w',p') = g_n(a)\), and \((s,r) = a\). Then,

\[
\left( \bigsquare \ g_n \right)^\dagger(a) = \left( s', r', w \times w', p \times p' \right) = \left( \bigsquare \ g_n^\dagger \right)(a).
\]

\(\square\)

Lemma 4.6. For every command \( C \), its semantics \( \llbracket C \rrbracket_d \) is well-defined and belongs to \( \mathcal{D} \).

Proof. The proof that \( \llbracket C \rrbracket_d \) is well-defined and measurable proceeds by induction on the structure of commands \( C \).

- Case of \( \llbracket \text{skip} \rrbracket_d \):
  The semantics is clearly well-defined in this case. Note that \( \llbracket \text{skip} \rrbracket_d \) is essentially the pairing of the identity function and a constant function. Since both of these functions are measurable and the pairing preserves measurability, \( \llbracket \text{skip} \rrbracket_d \) is measurable. Also, since \( \llbracket \text{skip} \rrbracket_d \) neither changes nor depends on the \( r \) component, it is local.

- Case of \( \llbracket x := E \rrbracket_d \):
  The proof is identical to the previous case except that instead of the identity function, we should use the function \((s,r) \mapsto (s[x \mapsto \llbracket E \rrbracket])\), which is measurable because it can be constructed by composing and pairing three measurable functions, \( \llbracket E \rrbracket \), \((s,v) \mapsto s[x \mapsto v]\) and projection.

- Case of \( \llbracket \text{if } B \{ C_0 \} \text{ else } \{ C_1 \} \rrbracket_d \):
  The well-definedness and the locality are immediate consequences of the induction hypothesis on \( \llbracket C_0 \rrbracket_d \) and \( \llbracket C_1 \rrbracket_d \). Let \( A \subseteq \{\bot\} \cup \text{Store} \times \text{RDB} \times [0,\infty) \times [0,\infty) \) be a measurable set. Then, \( \llbracket \text{if } B \{ C_0 \} \text{ else } \{ C_1 \} \rrbracket_d^{-1}(A) \)

\[
= \left( \{(s,r) \mid \llbracket B \rrbracket s = \text{true} \} \cap \llbracket C_0 \rrbracket_d^{-1}(A) \right) \cup \left( \{(s,r) \mid \llbracket B \rrbracket s \neq \text{true} \} \cap \llbracket C_1 \rrbracket_d^{-1}(A) \right).
\]

Moreover, \( \{\text{true}\} \) is measurable, and so is its complement. Since \( \llbracket B \rrbracket \) is a measurable function, \( \{(s,r) \mid \llbracket B \rrbracket s = \text{true} \} \) and \( \{(s,r) \mid \llbracket B \rrbracket s \neq \text{true} \} \) are measurable. As a result the above set is measurable. So, \( \llbracket \text{if } B \{ C_0 \} \text{ else } \{ C_1 \} \rrbracket_d \) is well-defined, local and measurable as claimed.

- Case of \( \llbracket C_0; C_1 \rrbracket_d \):
  By induction hypothesis, both \( \llbracket C_0 \rrbracket_d \) and \( \llbracket C_1 \rrbracket_d \) are well-defined, local and measurable. This and Lemma A.3 imply that \( \llbracket C_1 \rrbracket_d^\dagger \) is also local and measurable. The function \( \llbracket C_0; C_1 \rrbracket_d \) is nothing but the composition of two well-defined measurable functions \( \llbracket C_1 \rrbracket_d^\dagger \) and \( \llbracket C_0 \rrbracket_d \), and
so it satisfies the claimed well-definedness and measurability. To see the locality of \( \|C_0; C_1\|_d \), suppose that
\[
(\|C_1\|_d^2 \circ \|C_0\|_d)(s, r) = (s'', r'', w'', p'').
\]
Then, there exist \( s', r', w', p' \) such that
\[
\|C_0\|_d(s, r) = (s', r', w', p') \land \|C_1\|_d^2(s', r', w', p') = (s'', r'', w'', p'').
\]
By the locality of \( \|C_0\|_d \) and \( \|C_1\|_d^2 \), there exist \( r'_0 \) and \( r''_0 \) such that
\[
r'_0 \# r' \land r'_0 \uplus r' = r \land \|C_0\|_d(s, r'_0) = (s', [], w', p')
\land r''_0 \# r'' \land r''_0 \uplus r'' = r' \land \|C_1\|_d^2(s', r''_0, w', p') = (s'', [], w'', p'').
\]
Note that \( r'_0 \# r''_0 \). Let \( r_0 \triangleq r'_0 \uplus r''_0 \). Then,
\[
r_0 \# r'' \land r_0 \uplus r'' = r \land (\|C_1\|_d^2 \circ \|C_0\|_d)(s, r_0) = \|C_1\|_d^2(s', r''_0, w', p') = (s'', [], w'', p'').
\]
The first equality from above uses the locality of \( \|C_0\|_d \). Thus, the first conjunct in the definition of locality holds if we use \( r_0 \) as a witness in the conjunct. To prove the second in the definition, consider \( r_1 \) such that \( r_1 \# r \). Then, \( r_1 \# r' \). Thus, by the locality of \( \|C_0\|_d \) and \( \|C_1\|_d^2 \), we have
\[
(\|C_1\|_d^2 \circ \|C_0\|_d)(s, r \uplus r_1) = \|C_1\|_d^2(s', r' \uplus r_1, w', p') = (s'', r'' \uplus r_1, w'', p'').
\]

- Case of \( \|\text{while } B \{C\}\|_d \): Since \( D \) is an \( \omega \)-cpo and has the least element, we simply need to prove that \( G \) in the definition of \( \|\text{while } B \{C\}\|_d \) is well-defined and continuous so that Kleene’s fixed point theorem applies (Lemma 4.5). By induction hypothesis, \( \|C\|_d \) is well-defined, local, and measurable. Thus, \( G \) is a well-defined function from \( D \) to \( D \); the proof that the image of \( G \) consists of local measurable functions is similar to the case of the if statement extended with the argument with the case of sequencing. Furthermore, \( G \) is monotone because both the lifting \( -\downarrow \) and the function composition are monotone. Thus, \( G \) transforms an \( \omega \)-chain into an \( \omega \)-chain. Finally, \( G \) preserves the least upper bound of such an \( \omega \)-chain. This is because both the lifting \( -\downarrow \) and the function composition preserve the least upper bound. We have just shown that \( G \) is well-defined and continuous.

- Case of \( \|x := \text{sample}_{\text{norm}}(S, E_1, E_2)\|_d \): The well-definedness and the locality are immediate. To prove the measurability, pick a measurable subset \( A \) of \( \{\bot\} \cup (\mathit{Store} \times RDB \times [0, \infty) \times [0, \infty) \). Then,
\[
\{(s, r) \mid \|x := \text{sample}_{\text{norm}}(S, E_1, E_2)\|_d(s, r) \in A\} = A_0 \cup A_1 \cup A_2 \cup A_3
\]
where
- \( A_0 = \{(s, r) \mid \bot \in A \land \|S\|, s \notin \text{dom}(r)\} \),
- \( A_1 = \{(s, r) \mid \bot \in A \land \|E_2\|, s \notin (0, \infty)\} \),
- \( A_2 = \{(s, r) \mid \|S\|, s \in \text{dom}(r) \land \|E_2\|, s \in (0, \infty)\} \),
- \( A_3 = A_2 \cap \{(s, r) \mid (s[x \mapsto r(\|S\|s)], r \setminus \|S\|s), 1, \mathcal{N}(r(\|S\|s); \|E_1\|s, \|E_2\|s)) \in A\} \).

We show that all of \( A_0, A_1, A_2, \) and \( A_3 \) are measurable. The measurability of the first three sets follow from the measurability of the following two sets:
\[
A'_0 = \{(s, r) \mid \|E_2\|, s \in (0, \infty)\} \quad \text{and} \quad A'_1 = \{(s, r) \mid \|S\|, s \in \text{dom}(r)\}.
\]
The set $A'_1$ is measurable because it is the inverse image of the composition of two measurable functions (namely, $\llbracket E_1 \rrbracket$ and projection) to the measurable set $(0, \infty)$. For the measurability of $A'_1$, note that

$$\{(s, r) \mid \llbracket S \rrbracket s \in \text{dom}(r)\} = \bigcup_{K \subseteq \text{fin Str}} \{(s, r) \mid \text{dom}(r) = K \} \cap \{(s, r) \mid \llbracket S \rrbracket s \in K\}.$$  

The set on the RHS is measurable, because it is a countable union of the intersection of two measurable subsets; the measurability of $\{(s, r) \mid \text{dom}(r) = K\}$ comes from the fact that $\text{RDB}$ is built by the disjoint-union construction over $[K \to \mathbb{R}]$ for all finite $K$, and the measurability of $\{(s, r) \mid \llbracket S \rrbracket s \in K\}$ follows from the measurability of the function $\llbracket S \rrbracket$ and the set $K$. It remains to show the measurability of $A_3$. To do so, we note that $A_3$ is the inverse image of a function constructed by applying measurability-preserving operators on $\llbracket S \rrbracket$, $\llbracket E_1 \rrbracket$, $\llbracket E_2 \rrbracket$ and $N$, and the application and restriction functions on random databases. Since the first four are measurable, it is sufficient to show the measurability of the last two, which we spell out below:

$$f_1 : \text{RDB} \times \text{Str} \to \mathbb{R}, \quad f_1(r, \alpha) \triangleq \begin{cases} r(\alpha) & \text{if } \alpha \in \text{dom}(r), \\ 0 & \text{otherwise}, \end{cases}$$

$$f_2 : \text{RDB} \times \text{Str} \to \text{RDB}, \quad f_2(r, \alpha)(\beta) \triangleq \begin{cases} \text{undefined} & \text{if } \alpha = \beta, \\ r(\beta) & \text{otherwise}. \end{cases}$$

For any measurable subset $A$ of $\mathbb{R}$,

$$f_1^{-1}(A) = \bigcup_{\alpha \in \text{Str}} \bigcup_{K \subseteq \text{fin Str}} \left( \begin{array}{c} \{(r, \beta) \mid \text{dom}(r) = K \land \beta = \alpha \land \alpha \notin K \land 0 \in A\} \\ \cup \{(r, \beta) \mid \text{dom}(r) = K \land \beta = \alpha \land \alpha \in K \land r(\alpha) \in A\} \end{array} \right).$$

All the explicitly-defined sets on the RHS of this equation are measurable, and the RHS only uses finite or countable union. Thus, $f_1^{-1}(A)$ is measurable. Now consider a measurable subset $A'$ of $\text{RDB}$. Then,

$$f_2^{-1}(A') = \bigcup_{\alpha \in \text{Str}} \bigcup_{K \subseteq \text{fin Str}} \left( \begin{array}{c} \{(r, \beta) \mid \beta = \alpha \land \text{dom}(r) = K \land \alpha \notin K \land r \in A'\} \\ \cup \{(r, \beta) \mid \beta = \alpha \land \text{dom}(r) = K \land \alpha \in K \land r \notin A'\} \end{array} \right).$$

Again all the explicitly-defined sets on the RHS of this equation are measurable, and the RHS only uses finite or countable union. Thus, $f_2^{-1}(A')$ is measurable.

- Case of $\llbracket \text{score}_{\text{norm}}(E_0, E_1, E_2) \rrbracket_d^{-1}$:
  
  The well-definedness and the locality are immediate in this case. Pick a measurable set

  $$A \subseteq \{\bot\} \cup (\text{Store} \times \text{RDB} \times [0, \infty) \times [0, \infty)).$$

  Then,

  $$\llbracket \text{score}_{\text{norm}}(E_0, E_1, E_2) \rrbracket_d^{-1}(A) = \{(s, r) \mid (s, r, N(\llbracket E_0 \rrbracket s; \llbracket E_1 \rrbracket s, \llbracket E_2 \rrbracket s), 1) \in A\} \cup \{(s, r) \mid \bot \in A \land \llbracket E_2 \rrbracket s \notin (0, \infty)\}$$

  Both sets in the union here are measurable because all of $N$, $\llbracket E_0 \rrbracket$, $\llbracket E_1 \rrbracket$ and $\llbracket E_2 \rrbracket$ are measurable and the composition and the pairing operators on functions preserve measurability. Thus, the union itself is measurable. 

$\square$
Lemma 4.7. For all \( g \in \mathcal{D} \), the following functions from Store \( \times \) RDB to \( \mathbb{R} \) and \( \{ \bot \} \cup \) Store are measurable: \((s, r) \mapsto \text{dens}(g, s)(r)\) and \((s, r) \mapsto \text{get}(g, s)(r)\).

Proof. We have to show that for every real \( v \in \mathbb{R} \) and measurable subset \( A \subseteq \{ \bot \} \cup \) Store,

\[
B_0 \triangleq \{(s, r) \mid \text{dens}(g, s)(r) \leq v\} \quad \text{and} \quad B_1 \triangleq \{(s, r) \mid \text{get}(g, s)(r) \in A\}
\]

are measurable. The set \( B_0 \) is measurable because

\[
\{(s, r) \mid \text{dens}(g, s)(r) \leq v\} = \{(s, r) \mid g(s, r) = \bot \land 0 \leq v\}
\]

and the three sets on the RHS of the equation are all measurable. The measurability of the other \( B_1 \) follows from a similar argument based on case split. That is,

\[
\{(s, r) \mid \text{get}(g, s)(r) \in A\} = \{(s, r) \mid g(s, r) = \bot \land \bot \in A\}
\]

and all three sets on the RHS of the equation are measurable. Thus, \( B_1 \) is a measurable subset. \( \Box \)

In the next lemma, we regard \( \{ \bot \} \cup \) Store as a partially-ordered set with the expected order: \( a \sqsubseteq b \iff a = \bot \lor a = b \), for all \( a, b \in \{ \bot \} \cup \) Store.

Lemma A.4. Let \( \{g_n\}_n \) be an \( \omega \)-chain in \( \mathcal{D} \). Then, for all stores \( s \) and random databases \( r \), the sequences \( \{\text{dens}(g_n, s)(r)\}_n \) and \( \{\text{get}(g_n, s)(r)\}_n \) are increasing.

\[
\lim_n \text{dens}(g_n, s)(r) = \text{dens}\left(\bigsqcap_n g_n, s\right)(r) \quad \text{and} \quad \lim_n \text{get}(g_n, s)(r) = \text{get}\left(\bigsqcap_n g_n, s\right)(r).
\]

Proof. Let \( s \) be a store, \( r \) be a random database, and \( \{g_n\}_n \) be an \( \omega \)-chain in \( \mathcal{D} \).

We first show that the sequences \( \{\text{dens}(g_n, s)(r)\}_n \) and \( \{\text{get}(g_n, s)(r)\}_n \) are increasing. We do so by showing the monotonicity of \( \text{dens}(−, s)(r)\) and \( \text{get}(−, s)(r)\). Consider \( g, g' \in \mathcal{D} \) such that \( g \sqsubseteq g' \). Then, \( g(s, r) = \bot \) or \( g(s, r) = g'(s, r) \). In the former case, \( \text{dens}(g, s)(r) = 0 \leq \text{dens}(g', s)(r) \) and \( \text{get}(g, s)(r) = \bot \sqsubseteq \text{get}(g', s)(r) \). In the latter case, \( \text{dens}(g, s)(r) = \text{dens}(g', s)(r) \) and \( \text{get}(g, s)(r) = \text{get}(g', s)(r) \). Thus, \( \text{dens}(−, s)(r) \) and \( \text{get}(-, s)(r) \) are monotone.

Next we show that \( \text{dens}(−, s)(r) \) and \( \text{get}(−, s)(r) \) preserve the limit. By what we have already shown, the sequence \( \{g_n(s, r)\}_n \) is monotone. But this sequence is ultimately stable. The stable value becomes the least upper bound \( \bigsqcap_n g_n(s, r) \), which is also equal to \( (\bigsqcap_n g_n)(s, r) \). Thus, \( \{\text{dens}(g_n, s)(r)\}_n \) is ultimately stable as well, and its limit is

\[
\lim_n \text{dens}(g_n, s)(r) = \text{dens}\left(\bigsqcap_n g_n, s\right)(r).
\]

By the same reason, \( \{\text{get}(g_n, s)(r)\}_n \) is ultimately stable, and has the limit

\[
\lim_n \text{get}(g_n, s)(r) = \text{get}\left(\bigsqcap_n g_n, s\right)(r).
\]
LEMMA 4.8. For all non-negative bounded measurable functions $h : (\{\bot\} \cup \text{Store}) \times \text{RDB} \to \mathbb{R}$, stores $s$, and functions $g_1, g_2 \in \mathcal{D}$, we have that

$$
\int \rho(dr) \left( \text{dens}(g_2^s \circ g_1, s)(r) \cdot h\left( g_2^s \circ g_1, s\right)(r) \right)
= \int \rho(dr_1) \left( \text{dens}(g_1, s)(r_1) \cdot 1_{[g_1, s]) \neq \bot}
\cdot \int \rho(dr_2) \left( \text{dens}(g_2, g_1, s)(r_1)(r_2) \cdot 1_{[r_1 \cup r_2]} \cdot h\left( g_2, g_1, s\right)(r_1)(r_2)\right)\right).
$$

PROOF. For $g \in \mathcal{D}$ and $s \in \text{Store}$, let

$$
R_1(g, s) : \text{RDB} \to \text{RDB},
$$

$$
R_1(g, s)(r) = \begin{cases}
  r \setminus \text{dom}(r') & \text{if } g(s, r) = (s', r', w', p') \text{ for some } (s', r', w', p'),
  \\
  \emptyset & \text{otherwise},
\end{cases}
$$

$$
R_2(g, s) : \text{RDB} \to \text{RDB},
$$

$$
R_2(g, s)(r) = \begin{cases}
  r' & \text{if } g(s, r) = (s', r', w', p') \text{ for some } (s', r', w', p'),
  \\
  \emptyset & \text{otherwise}.
\end{cases}
$$

Here $r \setminus K$ means the restriction of $r$ to $\text{dom}(r) \setminus K$. Both $R_1(g, s)$ and $R_2(g, s)$ are measurable. The measurability of the latter is an immediate consequence of the measurability of $g$. For the former, we note that for every measurable subset $A$ of $\text{RDB}$ with $\emptyset \not\in A$,

$$
R_1(g, s)^{-1}(A) = \bigcup_{K_1, K_2 \subseteq \text{finStr}} \left\{ r_1 \cup r_2 \mid r_1 \in (A \cap [K_1 \to \mathbb{R}]) \land r_2 \in [K_2 \to \mathbb{R}] \cap g^{-1}(s, -)(\text{Store} \times [K_2 \to \mathbb{R}] \times [0, \infty) \times [0, \infty)) \right\}.
$$

The partially applied function $g(s, -)$ is measurable, so the RHS of this equation is the countable union of measurable subsets. If $A = \{\emptyset\}$, then

$$
R_1(g, s)^{-1}(A) = g^{-1}(s, -)(\{\bot\}) \cup \left( A \cap g^{-1}(s, -)(\text{Store} \times \{\emptyset\}) \times [0, \infty) \times [0, \infty)\right).
$$

Thus, $R_1(g, s)^{-1}(A)$ is measurable. The remaining case is that $\emptyset \in A$. We can handle this case by splitting $A$ into $\{\emptyset\}$ and $A \setminus \{\emptyset\}$, and dealing with the split cases separately.

For $g \in \mathcal{D}$, $s \in \text{Store}$, and $r \in \text{RDB}$, let consumed$(g, s, r)$ be the predicate defined by

$$
\text{consumed}(g, s, r) \iff \exists s', w', p'. (g(s, r) = (s', s, w', p')).
$$

Then, $r \mapsto 1_{[\text{consumed}(g, s, r)]}$ is a measurable function.

Now we note a few useful equalities about the entities $h, s, g_1, g_2$ assumed in the lemma. First, for all $r$, if

$$
\text{consumed}(g_2^s \circ g_1, s, r),
$$

then

$$
r = R_1(g_1, s)(r) \cup R_2(g_1, s)(r),
$$

$$
\text{get}(g_2^s \circ g_1, s)(r) = \text{get}\left( g_2, \text{get}(g_1, s)(R_1(g_1, s)(r))\right)(R_2(g_1, s)(r)),
$$

$$
\text{dens}(g_2^s \circ g_1, s)(r) = \text{dens}(g_1, s)(R_1(g_1, s)(r)) \cdot \text{dens}\left( g_2, \text{get}(g_1, s)(R_1(g_1, s)(r))\right)(R_2(g_1, s)(r)).
$$

The first equality is just an immediate consequence of the definitions of $R_1$ and $R_2$, and the other two equalities hold mainly because $g_1$ and $g_2$ are local.
Next, consider the density functions $f_1$ on $RDB$ and $f_2$ on $RDB \times RDB$:

\[
\begin{align*}
f_1 : RDB &\to [0, \infty), \\
f_1(r) &\triangleq \mathbbm{1}_{[\text{consumed}(g_2^+ \circ g_1, s, r)]}, \\
f_2 : RDB \times RDB &\to [0, \infty) \\
f_2(r_1, r_2) &\triangleq \mathbbm{1}_{[\text{consumed}(g_1, s, r_1)]} \cdot \mathbbm{1}_{[\text{consumed}(g_2, \text{get}(g_1, s)(r_1), r_2)]} \
&\quad \cdot \mathbbm{1}_{[r_1 \neq r_2]}.
\end{align*}
\]

The densities are taken with respect to $\rho$ and $\rho \otimes \rho$, respectively. There is a measurable bijection $\beta$ between the supports of $f_1$ and $f_2$, and the pushforward of the measure of $f_1$ by this bijection is the measure of $f_2$. Concretely, the function $\beta$ is:

\[
\beta : RDB \to RDB \times RDB, \\
\beta(r) = \begin{cases} \\
(R_1(g_1, s)(r), R_2(g_1, s)(r)) & \text{if consumed}(g_2^+ \circ g_1, s, r) \\
(r_d, r_d) & \text{otherwise},
\end{cases}
\]

where $r_d \in RDB$ is ["s" → nil]. Note that $\beta$ is not a bijection. But the restriction of $\beta$ to $A_1 = \{ r \mid f_1(r) > 0 \}$ is injective, and its image is $A_2 = \{ (r_1, r_2) \mid f_2(r_1, r_2) > 0 \}$. These properties follow from the definitions of consumed, $R_1$ and $R_2$ and the locality of $g$. The function $\beta$ is measurable because $R_1(g_1, s)$ and $R_2(g_2, s)$ are measurable functions and $\{ r \mid \text{consumed}(g_2^+ \circ g_1, s, r) \}$ is a measurable set. It remains to show the measure preservation of $\beta$. Pick a measurable subset $A'$ of $A_2$. We have to show that

\[
(\rho \otimes \rho)(A') = \rho(\beta^{-1}(A')).
\]

We calculate the equation as follows:

\[
\begin{align*}
(\rho \otimes \rho)(A') &= (\rho \otimes \rho)\left( \bigcup_{K_1, K_2 \subseteq \text{fin}, \text{Str}} \left( K_1 \to [\mathbb{R}] \times K_2 \to [\mathbb{R}] \right) \cap A' \right) \\
&= \bigcup_{K_1, K_2 \subseteq \text{fin}, \text{Str}} \left( \bigotimes_{a_1 \in K_1} \rho_{a_1} \right) \otimes \left( \bigotimes_{a_2 \in K_2} \rho_{a_2} \right) \left( A' \cap \left( K_1 \to [\mathbb{R}] \times K_2 \to [\mathbb{R}] \right) \right) \\
&= \bigcup_{K_1, K_2 \subseteq \text{fin}, \text{Str}} \left( \bigotimes_{a \in K_1 \cup K_2} \rho_{a} \right) \left( \beta^{-1}(A') \cap \left( K_1 \to [\mathbb{R}] \times K_2 \to [\mathbb{R}] \right) \right) \\
&= \rho \left( \beta^{-1}(A') \cap \left( K_1 \to [\mathbb{R}] \times K_2 \to [\mathbb{R}] \right) \right) \\
&= \rho \left( \beta^{-1}(A') \right).
\end{align*}
\]

Using what we have shown so far, we calculate the claim of the lemma. In the calculation we often omit the $g_1$ and $s$ parameters from $R_1(g_1, s)(r)$ and $R_2(g_1, s)(r)$, and just write $R_1(r)$ and $R_2(r)$.

\[
\begin{align*}
\int \rho(dr) \left( \text{dens}(g_2^+ \circ g_1, s)(r) \cdot h\left( \text{get}(g_2^+ \circ g_1, s)(r), r \right) \right) \\
= \int \rho(dr) \left( \mathbbm{1}_{[\text{consumed}(g_2^+ \circ g_1, s, r)]} \cdot \text{dens}(g_2^+ \circ g_1, s)(r) \cdot h(\text{get}(g_2^+ \circ g_1, s)(r), r) \right) \\
= \int \rho(dr) \left( \mathbbm{1}_{[\text{consumed}(g_2^+ \circ g_1, s, r)]} \cdot \text{dens}(g_2^+ \circ g_1, s)(r) \cdot \mathbbm{1}_{[\text{get}(g_1, s)(R_1(r)) \neq \perp]} \cdot h(\text{get}(g_2^+ \circ g_1, s)(r), r) \right)
\end{align*}
\]
\[
\frac{\int \rho(\text{dr}) \left( \mathbb{1}_{[\text{consumed}(g_1, s, r)]} \cdot \text{dens}(g_1, s)(R_1(r)) \cdot \mathbb{1}_{[\text{get}(g_1, s)\langle R_1(r)\rangle \neq \perp]} \right)
\cdot \text{dens}(g_2, \text{get}(g_1, s)(R_1(r)))(R_2(r)) \cdot h(\text{get}(g_2, \text{get}(g_1, s)(R_1(r)))(R_2(r)), R_1(r) \uplus R_2(r))}{\int \rho(\text{dr}_1) \int \rho(\text{dr}_2) \left( \mathbb{1}_{[\text{consumed}(g_1, s, r_1)]} \cdot \mathbb{1}_{[\text{consumed}(g_2, \text{get}(g_1, s)(r_1), r_2)]} \cdot \mathbb{1}_{[r_1 \uplus r_2]}
\cdot \text{dens}(g_1, s)(r_1) \cdot \mathbb{1}_{[\text{get}(g_1, s)(r_1) \neq \perp]} \cdot \text{dens}(g_2, \text{get}(g_1, s)(r_1))(r_2)
\cdot h(\text{get}(g_2, \text{get}(g_1, s)(r_1))(r_2), r_1 \uplus r_2) \right)
\]
\[
= \int \rho(\text{dr}_1) \int \rho(\text{dr}_2) \left( \text{dens}(g_1, s)(r_1) \cdot \mathbb{1}_{[\text{get}(g_1, s)(r_1) \neq \perp]} \right)
\cdot \int \rho(\text{dr}_2) \left( \text{dens}(g_2, \text{get}(g_1, s)(r_1))(r_2) \cdot \mathbb{1}_{[r_1 \uplus r_2]} \cdot h\left(\text{get}(g_2, \text{get}(g_1, s)(r_1))(r_2), r_1 \uplus r_2\right) \right)
\]
\]

**Theorem 4.9.** For all non-negative bounded measurable monotone functions \( h : (\{\perp\} \cup \text{Store}) \times \mathbb{R}^\text{DDB} \rightarrow \mathbb{R} \) and states \( \sigma \),
\[
\int \mathbb{C}\|\mathbb{C}(\mathbb{C}(\mathbf{d}(\sigma', w')) (w' \cdot h(\sigma'_s, \sigma'_r))) = \int \rho(\text{dr}') \left( \text{dens}(C, \sigma_s)(r') \cdot \mathbb{1}_{[r' \uplus \sigma_r]} \cdot h(\text{get}(C, \sigma_s)(r'), r' \uplus \sigma_r) \right).
\]

**Proof.** We prove the theorem by induction on the structure of \( C \).
When \( C \equiv \text{skip} \), we derive the claimed equality as follows:
\[
\int \mathbb{C}(\mathbf{d}(\sigma', w')) (w' \cdot h(\sigma'_s, \sigma'_r))
= h(\sigma_s, \sigma_r)
= \int \rho(\text{dr}') \left( \mathbb{1}_{[r' = \perp]} \cdot \mathbb{1}_{[\perp \uplus \sigma_r]} \cdot h(\sigma_s, \perp \uplus \sigma_r) \right)
= \int \rho(\text{dr}') \left( \mathbb{1}_{[r' = \perp]} \cdot \mathbb{1}_{[r' \uplus \sigma_r]} \cdot h(\sigma_s, r' \uplus \sigma_r) \right)
= \int \rho(\text{dr}') \left( \text{dens}(\text{skip}, \sigma_s)(r') \cdot \mathbb{1}_{[r' \uplus \sigma_r]} \cdot h(\text{get}(\text{skip}, \sigma_s)(r'), r' \uplus \sigma_r) \right).
\]

Next we handle the case that \( C \equiv (x := E) \).
\[
\int \mathbb{C}(\mathbf{d}(\sigma', w')) (w' \cdot h(\sigma'_s, \sigma'_r))
= h(\sigma_s, \sigma_r)
= \int \rho(\text{dr}') \left( \mathbb{1}_{[r' = \perp]} \cdot \mathbb{1}_{[\perp \uplus \sigma_r]} \cdot h(\sigma_s, \perp \uplus \sigma_r) \right)
= \int \rho(\text{dr}') \left( \mathbb{1}_{[r' = \perp]} \cdot \mathbb{1}_{[r' \uplus \sigma_r]} \cdot h(\sigma_s, \perp \uplus \sigma_r) \right)
= \int \rho(\text{dr}') \left( \text{dens}(x := E, \sigma_s)(r') \cdot \mathbb{1}_{[r' \uplus \sigma_r]} \cdot h(\text{get}(x := E, \sigma_s)(r'), r' \uplus \sigma_r) \right).
\]
We move on to the case that $C \equiv (\text{if } B \{C_0\} \text{ else } \{C_1\})$, and prove the desired equality:

$$
\int [\text{if } B \{C_0\} \text{ else } \{C_1\}] (\sigma)(d(\sigma', w')) (w' \cdot h(\sigma'_s, \sigma'_r))
$$

$$
= \mathbb{I}_{[B]\sigma_s=\text{true}} \cdot \int C_0 (\sigma)(d(\sigma', w')) (w' \cdot h(\sigma'_s, \sigma'_r))
+ \mathbb{I}_{[B]\sigma_s=\text{true}} \cdot \int C_1 (\sigma)(d(\sigma', w')) (w' \cdot h(\sigma'_s, \sigma'_r))
$$

$$
= \mathbb{I}_{[B]\sigma_s=\text{true}} \cdot \int \rho(dr') \left( \text{dens}(C_0, \sigma_s)(r') \cdot \mathbb{I}_{[r'\not\in \sigma_s]} \cdot h(\text{get}(C_0, \sigma_s)(r'), r' \not\in \sigma_r) \right)
+ \mathbb{I}_{[B]\sigma_s=\text{true}} \cdot \int \rho(dr') \left( \text{dens}(C_1, \sigma_s)(r') \cdot \mathbb{I}_{[r'\not\in \sigma_s]} \cdot h(\text{get}(C_1, \sigma_s)(r'), r' \not\in \sigma_r) \right)
$$

$$
= \int \rho(dr') \left( \text{dens}(\text{if } B \{C_0\} \text{ else } \{C_1\}, \sigma_s)(r') \cdot \mathbb{I}_{[r'\not\in \sigma_s]} \cdot h(\text{get}(\text{if } B \{C_0\} \text{ else } \{C_1\}, \sigma_s)(r'), r' \not\in \sigma_r) \right).
$$

Now we prove the claimed equality for $C \equiv (C_0; C_1)$.

$$
\int [C_0; C_1] (\sigma)(d(\sigma', w')) (w' \cdot h(\sigma'_s, \sigma'_r))
$$

$$
= \int C_0 (\sigma)(d(\sigma', w')) \int C_1 (\sigma')(d(\sigma'', w'')) (w' \cdot w'' \cdot h(\sigma''_s, \sigma''_r))
$$

$$
= \int C_0 (\sigma)(d(\sigma', w')) \left( w' \cdot \int C_1 (\sigma')(d(\sigma'', w'')) (w'' \cdot h(\sigma''_s, \sigma''_r)) \right)
$$

$$
= \int C_0 (\sigma)(d(\sigma', w')) \left( w' \cdot \int \rho(dr_2) \left( \text{dens}(C_1, \sigma'_s)(r_2) \cdot \mathbb{I}_{[r_2\not\in \sigma'_s]} \cdot h(\text{get}(C_1, \sigma'_s)(r_2), r_2 \not\in \sigma'_r) \right) \right)
$$

$$
= \int \rho(dr_1) \left( \text{dens}(C_0, \sigma_s)(r_1) \cdot \mathbb{I}_{[r_1\not\in \sigma_r]} 
\cdot \mathbb{I}_{[\text{get}(C_0, \sigma_s)(r_1)\not\in \perp]} \cdot \int \rho(dr_2) \left( \text{dens}(C_1, \text{get}(C_0, \sigma_s)(r_1))(r_2) \cdot \mathbb{I}_{[r_2\not\in \sigma_r]} \cdot h(\text{get}(C_1, \text{get}(C_0, \sigma_s)(r_1))(r_2), r_2 \not\in (r_1 \not\in \sigma_r) \not\in \sigma_r) \right) \right)
$$

$$
= \int \rho(dr_1) \left( \text{dens}(C_0, \sigma_s)(r_1) \cdot \mathbb{I}_{[\text{get}(C_0, \sigma_s)(r_1)\not\in \perp]} 
\cdot \int \rho(dr_2) \left( \text{dens}(C_1, \text{get}(C_0, \sigma_s)(r_1))(r_2) \cdot \mathbb{I}_{[r_1\not\in \sigma_r]} \cdot \mathbb{I}_{[r_2\not\in \sigma_r]} \cdot h(\text{get}(C_1, \text{get}(C_0, \sigma_s)(r_1))(r_2), (r_1 \not\in \sigma_r) \not\in \sigma_r) \right) \right)
$$

$$
= \int \rho(dr) \left( \text{dens}(C_0; C_1)(r) \cdot \mathbb{I}_{[r\not\in \sigma_r]} \cdot h(\text{get}(C_0; C_1)(r), r \not\in \sigma_r) \right).
$$

The third and fifth equalities follow from the induction hypothesis, and the last equality from Lemma 4.8.

The next case is $C \equiv (\text{while } B \{C_0\})$. Let $\mathcal{R} \subseteq \mathcal{K} \times \mathcal{D}$ be the relation defined by

$$
\kappa [\mathcal{R}] g \iff \text{for all states } \sigma \text{ and non-negative bounded measurable monotone functions } h,
$$
The first equation follows from Lemma A.1, and the third from the monotone convergence theorem. This is because for all stores \( \sigma \), \( \kappa \) and non-negative bounded measurable functions \( h \),

\[
\int \kappa(\sigma)(d(\sigma', w'))(w' \cdot h(\sigma'_s, \sigma'_r)) \\
= \int \rho(dr') \left( \text{dens}(g_n, \sigma_s)(r') \cdot 1_{[r' \in \sigma_s]} \cdot h\left(\text{get}(g_n, \sigma_s)(r'), r' \cup \sigma_r\right)\right).
\]

Note that the least elements from \( \mathcal{K} \) and \( \mathcal{D} \) are related by \( \mathcal{R} \) because they make both sides of the equality in the definition of \( \mathcal{R} \) be zero. Furthermore, for all \( \omega \)-chains \( \{\kappa_n\}_n \) and \( \{g_n\}_n \) in \( \mathcal{K} \) and \( \mathcal{D} \), if \( \kappa_n [\mathcal{R}] g_n \) for all \( n \), then

\[
\bigg\lfloor \lim_n \kappa_n [\mathcal{R}] \bigg\rfloor = \bigg\lfloor g_n \bigg\rfloor.
\]

This is because for all stores \( \sigma \) and non-negative bounded measurable functions \( h \),

\[
\int \left( \bigg\lfloor \kappa_n \bigg\rfloor(\sigma)(d(\sigma', w'))(w' \cdot h(\sigma'_s, \sigma'_r)) \right) \\
= \lim_n \int \kappa_n(\sigma)(d(\sigma', w'))(w' \cdot f(\sigma'_r)) \\
= \lim_n \int \rho(dr') \left( \text{dens}(g_n, \sigma_s)(r') \cdot 1_{[r' \in \sigma_s]} \cdot h\left(\text{get}(g_n, \sigma_s)(r'), r' \cup \sigma_r\right)\right) \\
= \int \rho(dr') \lim_n \left( \text{dens}\left( \bigg\lfloor g_n, \sigma_s \bigg\rfloor(r') \cdot 1_{[r' \in \sigma_s]} \cdot h\left(\bigg\lfloor g_n, \sigma_s \bigg\rfloor(r'), r' \cup \sigma_r\right)\right)\right).
\]

The first equation follows from Lemma A.1, and the third from the monotone convergence theorem and Lemma A.4. The last equality holds because of Lemma A.4 again.

Thus, we can complete the proof of this case if we show one more thing. Let \( K \) and \( G \) be the functions on \( \mathcal{K} \) and \( \mathcal{D} \) used in the interpretation of the loop (namely, while \( B \{C_0\} \)) in the two semantics. What we have to show is:

\[
\kappa [\mathcal{R}] g \implies K(\kappa) [\mathcal{R}] G(g).
\]

Pick \( \kappa \) and \( g \) with \( \kappa [\mathcal{R}] g \). Then, for all \( \sigma \) and non-negative bounded measurable \( h \),

\[
\int (K(\kappa))(\sigma)(d(\sigma', w'))(w' \cdot h(\sigma'_s, \sigma'_r)) \\
= \|B\|_{\sigma_s = \text{true}} \cdot h(\sigma_s, \sigma_r) \\
+ \|B\|_{\sigma_s = \text{true}} \cdot \int C(\sigma)(d(\sigma', w')) \int K(\kappa)(d(\sigma'', w''))(w'' \cdot h(\sigma''_s, \sigma''_r)) \\
= \|B\|_{\sigma_s = \text{true}} \cdot h(\sigma_s, \sigma_r) \\
+ \|B\|_{\sigma_s = \text{true}} \cdot \int C(\sigma)(d(\sigma', w')) \left( w' \cdot \int K(\kappa)(d(\sigma'', w''))(w'' \cdot h(\sigma''_s, \sigma''_r)) \right).
\]

But

\[
\int C(\sigma)(d(\sigma', w')) \left( w' \cdot \int K(\kappa)(d(\sigma'', w''))(w'' \cdot h(\sigma''_s, \sigma''_r)) \right) \\
= \int C(\sigma)(d(\sigma', w')) \left( w' \cdot \rho(dr'') \left( \text{dens}(g, \sigma'_s)(r'') \cdot 1_{[r'' \in \sigma'_s]} \cdot h\left(\text{get}(g, \sigma'_s)(r''), r'' \cup \sigma_r\right)\right) \right) \\
= \int C(\sigma)(d(\sigma', w')).
\]

\[
\left( \omega' \cdot I_{[\sigma' \neq \perp]} \cdot \int \rho(d\omega') \left( \text{dens}(g, \sigma'_s)(\omega') \cdot I_{[r'' \neq \sigma'_r]} \cdot h(g, \sigma'_s)(\omega'), r'' \cup \sigma'_r) \right) \right)
\]

\[
= \int \rho(d\omega') \left( \text{dens}(C, \sigma_s)(\omega') \cdot I_{[r'' \neq \sigma_r]} \cdot I_{[\omega' \neq \sigma_r]} \cdot \int \rho(d\omega'') \left( \text{dens}(g, \omega') \cdot \int \kappa(\omega')(d(\omega'', \omega'))(\omega'' \cdot h(\sigma'_s, \omega'')) \right) \right)
\]

\[
\cdot h(g, \sigma_s)(\omega'), r'' \cup (r' \cup \sigma_r)) \right)
\]

\[
= \int \rho(d\omega') \left( \text{dens}(g^\circ \cdot \text{C}_d, \sigma_s)(r) \cdot I_{[r'' \neq \sigma_r]} \cdot h(g^\circ \cdot \text{C}_d, \sigma_s)(r, r \cup \sigma_r) \right)
\]

The last equality uses Lemma 4.8. Thus, if we continue our calculation that we paused momentarily, we get

\[
1_{[\text{B}, \sigma_s \neq \text{true}]} \cdot h(\sigma_s, \sigma_r)
\]

\[
+ 1_{[\text{B}, \sigma_s \neq \text{true}]} \cdot \int \text{C}(\sigma)(d(\sigma', \omega'))(w' \cdot \int \kappa(\sigma')(d(\sigma'', \omega''))(w'' \cdot h(\sigma'_s, \omega'')))
\]

\[
= 1_{[\text{B}, \sigma_s \neq \text{true}]} \cdot h(\sigma_s, \sigma_r)
\]

\[
+ 1_{[\text{B}, \sigma_s \neq \text{true}]} \cdot \int \rho(d\omega') \left( \text{dens}(g^\circ \cdot \text{C}_d, \sigma_s)(r) \cdot I_{[r'' \neq \sigma_r]} \cdot h(g^\circ \cdot \text{C}_d, \sigma_s)(r, r \cup \sigma_r) \right)
\]

\[
= 1_{[\text{B}, \sigma_s \neq \text{true}]} \cdot \int \rho(d\omega') \left( 1_{[r'' = []]} \cdot 1_{[\text{B}, \sigma_s \neq \text{true}]} \cdot h(\sigma_s, r' \cup \sigma_r) \right)
\]

\[
= 1_{[\text{B}, \sigma_s \neq \text{true}]} \cdot \int \rho(d\omega') \left( 1_{[r'' = []]} \cdot 1_{[\text{B}, \sigma_s \neq \text{true}]} \cdot h(\sigma_s, r' \cup \sigma_r) \right)
\]

\[
= \int \rho(d\omega') \left( \text{dens}(G(g), \sigma_s)(r') \cdot I_{[r'' \neq \sigma_r]} \cdot h(G(g), \sigma_s)(r', r' \cup \sigma_r) \right)
\]

Next we handle the case that \( C \equiv (x := \text{sample}_{\text{norm}}(S, E_1, E_2)). \)

\[
\int \text{C}(\sigma)(d(\sigma', \omega'))(w' \cdot h(\sigma'_s, \omega'))
\]

\[
= 1_{[S, \sigma_s \neq \text{dom}(\sigma_s)]} \cdot 1_{[E_2, \sigma_s \in (0, \infty)]} \cdot \int d\nu \left( \text{N}(\nu; E_1 \parallel S, \parallel E_2 \parallel S) \cdot h(\sigma_s[x \mapsto \nu], \sigma_r[[S] \sigma_s \mapsto \nu]) \right)
\]

\[
= 1_{[E_2, \sigma_s \in (0, \infty)]} \cdot \int d\nu \left( \text{N}(\nu; E_1 \parallel S, \parallel E_2 \parallel S) \cdot 1_{[[S] \sigma_s \mapsto \nu \neq \sigma_s]} \cdot h(\sigma_s[x \mapsto \nu], [S] \sigma_s \mapsto \nu \cup \sigma_r) \right)
\]
\[
\begin{align*}
&= I_{[E_2 \sigma_s \in (0, \infty)]} \cdot \int \rho(\text{dr}') \left( I_{[\text{dom}(\text{dr}') = [S \sigma_s]]} \cdot N(\text{r}'(\text{S}[\sigma_s]), E_1 \sigma_s, E_2 \sigma_s) \cdot I_{[r' \in [\sigma_s]]} \cdot h(\sigma_s, [r' \in [\sigma_s]]) \right)
$
\]

Towards Verified Stochastic Variational Inference for Probabilistic Programs

**Proof of Theorem in §5**

\[= \int \rho(\text{dr}') \left( I_{[E_2 \sigma_s \in (0, \infty)]} \cdot N(\text{r}'(\text{S}[\sigma_s]), E_1 \sigma_s, E_2 \sigma_s) \cdot I_{[r' \in [\sigma_s]]} \cdot h(\sigma_s, [r' \in [\sigma_s]]) \right)
\]

It remains to prove the case that \( C \equiv \text{score}_\text{norm}(E_0, E_1, E_2) \). Here is our proof for this last case:

\[
\int \rho(\text{dr}') \left( I_{[r' \in [\sigma_s]]} \cdot N(\text{r}'(\text{S}[\sigma_s]), E_1 \sigma_s, E_2 \sigma_s) \cdot I_{[r' \in [\sigma_s]]} \cdot h(\sigma_s, [r' \in [\sigma_s]]) \right)
\]

**Corollary 4.10.** \( \text{Mea}(C, A) = \int \rho(\text{dr}') \left( I_{[r' \in A]} \cdot \text{dens}(C, s_i)(r') \right) \) for all \( C \) and all measurable \( A \).

**Proof.** We instantiate Theorem 4.9 with \( h(a, r) = I_{[r' \in A]} \) and \( \sigma = \sigma_f \). Recall that \( (\sigma_i)_s = s_i \) and \( (\sigma_j)_r = [r] \). Thus, the conclusion of Theorem 4.9 in this case says that

\[
\int C(\sigma_i)(d(\sigma', w'))(w' \cdot I_{[\sigma_i \in A]}) = \int \rho(\text{dr}') \left( \text{dens}(C, s_i)(r') \cdot I_{[r' \in [\sigma_i]]} \cdot I_{[r' \in [\sigma_i]]} \right)
\]

This gives the equation claimed by the corollary.

**B PROOF OF THEOREM IN §5**

**Theorem 5.1.** Let \( C \) be a model, \( D_0 \) be a guide, and \( N \neq 0 \in \mathbb{N} \). Define \( \text{KL}_{\langle \_ \_ \rangle} : \mathbb{R}^p \to \mathbb{R}_{\geq 0} \) as \( \text{KL}_{\langle \_ \_ \rangle} \triangleq \text{KL}(\text{dens}(D_0, s_i) || \text{dens}(C, s_i) / Z_C) \). Then, \( \text{KL}_{\langle \_ \_ \rangle} \) is well-defined and continuously differentiable with

\[
\nabla_\theta \text{KL}_{\langle \_ \_ \rangle} = \mathbb{E}_{[r, \text{dens}(D_0, s_i)]}(r) \left[ \frac{1}{N} \sum_{i=1}^{N} \left( \nabla_\theta \text{log} \text{dens}(D_0, s_i)(r_i) \log \frac{\text{dens}(D_0, s_i)(r_i)}{\text{dens}(C, s_i)(r_i)} \right) \right]
\]

\[(R1) \ \text{dens}(C, s_i)(r) = 0 \implies \text{dens}(D_0, s_i)(r) = 0, \text{ for all } r \in \text{RDB} \text{ and } \theta \in \mathbb{R}^p; \]

\[(R2) \text{ for all } (r, \theta, j) \in \text{RDB} \times \mathbb{R}^p \times [p], \text{ the function } v \mapsto \text{dens}(D_{\theta[j; \cdot]}, s_i)(r) \text{ on } \mathbb{R} \text{ is differentiable.} \]

(R3) for all $\theta \in \mathbb{R}^p$,
\[ \int \rho(\text{dr}) \left( \text{dens}(D_{\theta}, s_i)(r) \cdot \log \frac{\text{dens}(D_{\theta}, s_i)(r)}{\text{dens}(C, s_i)(r)} \right) < \infty; \]

(R4) for all $(\theta, j) \in \mathbb{R}^p \times [p]$, the function
\[ \nu \mapsto \int \rho(\text{dr}) \left( \text{dens}(D_{\theta[j:v]}, s_i)(r) \cdot \log \frac{\text{dens}(D_{\theta[j:v]}, s_i)(r)}{\text{dens}(C, s_i)(r)} \right) \]
on $\mathbb{R}$ is continuously differentiable;

(R5) for all $\theta \in \mathbb{R}^p$,
\[ \nabla_\theta \int \rho(\text{dr}) \left( \text{dens}(D_{\theta}, s_i)(r) \cdot \log \frac{\text{dens}(D_{\theta}, s_i)(r)}{\text{dens}(C, s_i)(r)} \right) \]
\[ = \int \rho(\text{dr}) \nabla_\theta \left( \text{dens}(D_{\theta}, s_i)(r) \cdot \log \frac{\text{dens}(D_{\theta}, s_i)(r)}{\text{dens}(C, s_i)(r)} \right) \];

(R6) for all $\theta \in \mathbb{R}^p$,
\[ \int \rho(\text{dr}) \nabla_\theta \text{dens}(D_{\theta}, s_i)(r) = \nabla_\theta \int \rho(\text{dr}) \text{dens}(D_{\theta}, s_i)(r). \]

As we explained in the main text already, the theorem (except the continuous differentiability of $\text{KL}_\theta$) is well-known with a well-known proof. We include its proof for completeness and also to help the reader to see why each of the requirements in the theorem is needed.

**Proof.** $\text{KL}_\theta$ is well-defined because of the following derivation with R1 and R3:
\[ \text{KL}_\theta = \text{KL} \left( \text{dens}(D_{\theta}, s_i) \bigg| \frac{\text{dens}(C, s_i)}{\mathbb{Z}_C} \right) \]
\[ = \int \rho(\text{dr}) \left( \text{dens}(D_{\theta}, s_i)(r) \cdot \log \left( \frac{\text{dens}(D_{\theta}, s_i)(r)}{\text{dens}(C, s_i)(r)} \right) \right) \]
\[ = \int \rho(\text{dr}) \left( \text{dens}(D_{\theta}, s_i)(r) \cdot \left( \log \mathbb{Z}_C + \log \frac{\text{dens}(D_{\theta}, s_i)(r)}{\text{dens}(C, s_i)(r)} \right) \right) \]
\[ = \log \mathbb{Z}_C + \int \rho(\text{dr}) \left( \text{dens}(D_{\theta}, s_i)(r) \cdot \log \frac{\text{dens}(D_{\theta}, s_i)(r)}{\text{dens}(C, s_i)(r)} \right). \]

It remains to show the continuous differentiability of $\text{KL}_\theta$ and Equation (6). Recall the fact that for any function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$, if all partial derivatives of $f$ exist and are continuous, $f$ is continuously differentiable. Because of the fact and the linearity of expectation, it suffices to show that all partial derivatives of $\text{KL}_\theta$ are well-defined and continuous, and
\[ \nabla_\theta \text{KL}_\theta = \mathbb{E}_{\text{dens}(D_{\theta}, s_i)(r)} \left[ \left( \nabla_\theta \log \text{dens}(D_{\theta}, s_i)(r) \right) \log \frac{\text{dens}(D_{\theta}, s_i)(r)}{\text{dens}(C, s_i)(r)} \right]. \quad (13) \]

Here is a well-known derivation of (13):
\[ \nabla_\theta \text{KL}_\theta \]
\[ = \nabla_\theta \left( \log \mathbb{Z}_C + \int \rho(\text{dr}) \left( \text{dens}(D_{\theta}, s_i)(r) \cdot \log \frac{\text{dens}(D_{\theta}, s_i)(r)}{\text{dens}(C, s_i)(r)} \right) \right) \]
\[ = \nabla_\theta \int \rho(\text{dr}) \left( \text{dens}(D_{\theta}, s_i)(r) \cdot \log \frac{\text{dens}(D_{\theta}, s_i)(r)}{\text{dens}(C, s_i)(r)} \right) \quad (14) \]
\[
\begin{align*}
\mathbb{E}_{\text{dens}(D_\theta, s_1)}(r) = \int \rho(\mathbf{r}) \nabla_\theta \left( \text{dens}(D_\theta, s_1) \right) \cdot \log \frac{\text{dens}(D_\theta, s_1)}{\text{dens}(C, s_1)} \\
\mathbb{E}_{\text{dens}(D_\theta, s_1)}(r) = \int \rho(\mathbf{r}) \left( \nabla_\theta \text{dens}(D_\theta, s_1) \right) \cdot \log \frac{\text{dens}(D_\theta, s_1)}{\text{dens}(C, s_1)} + \int \rho(\mathbf{r}) \nabla_\theta \text{dens}(D_\theta, s_1)(\mathbf{r}) \\
\mathbb{E}_{\text{dens}(D_\theta, s_1)}(r) = \int \rho(\mathbf{r}) \left( \nabla_\theta \text{dens}(D_\theta, s_1) \right) \cdot \log \frac{\text{dens}(D_\theta, s_1)}{\text{dens}(C, s_1)} + \nabla_\theta 1 \\
\mathbb{E}_{\text{dens}(D_\theta, s_1)}(r) = \mathbb{E}_{\text{dens}(D_\theta, s_1)} \left( \nabla_\theta \log \text{dens}(D_\theta, s_1) \right) \cdot \log \frac{\text{dens}(D_\theta, s_1)}{\text{dens}(C, s_1)}.
\end{align*}
\]

The well-definedness and the continuity of \((\nabla_\theta \text{KL}_{\theta})_j\) for all \(j \in [p]\) follow from (14) and R4, which concludes the proof.

Note that R2 is used to guarantee the well-definedness of \(\nabla_\theta \log \text{dens}(D_\theta, s_1)\) in (13); R5 and R6 are used in (15) and (16), respectively.

\section*{C PROOFS OF EXAMPLES, LEMMA, AND THEOREMS IN §6}

\begin{theorem}
Under our assumption in §6.1,
\[
\mathbb{E}_{\text{dens}(D_\theta, s_1)}(r) \left[ \log \text{dens}(D_\theta, s_1) \right] < \infty.
\]
\end{theorem}

\begin{proof}
For \(i \in [M]\) and \(r \in A_i\), define
\[
g_{i, \theta}(r) = \prod_{\alpha \in K_i} N(r(\alpha); \mu_{i, \alpha}(\theta), \sigma_{i, \alpha}(\theta)).
\]

Then,
\[
\mathbb{E}_{\text{dens}(D_\theta, s_1)}(r) \left[ \log \text{dens}(D_\theta, s_1) \right]
\begin{align*}
&= \int \rho(\mathbf{r}) \left( \sum_{i=1}^{M} \mathbb{I}_{[r \in A_i]} \cdot g_{i, \theta}(r) \right) \cdot \log \sum_{i=1}^{M} \mathbb{I}_{[r \in A_i]} \cdot g_{i, \theta}(r) \\
&= \sum_{i=1}^{M} \int \rho(\mathbf{r}) \left( \mathbb{I}_{[r \in A_i]} \cdot g_{i, \theta}(r) \right) \cdot \log g_{i, \theta}(r) \\
&\leq \sum_{i=1}^{M} \int \rho(\mathbf{r}) \left( \mathbb{I}_{[r \in K_i \rightarrow \mathbb{R}]} \cdot g_{i, \theta}(r) \right) \cdot \log g_{i, \theta}(r) \\
&= \sum_{i=1}^{M} \mathbb{E}_{g_{i, \theta}(r)} \left[ \log g_{i, \theta}(r) \right]
\end{align*}
< \infty.
\]

The last inequality uses a well-known result that the differential entropy of any multivariate normal distribution is finite.
\end{proof}
Example 6.2. Consider guides $D_{i,θ}$ defined as follows ($i = 1, 2$):

$$D_{i,θ} \equiv (x_1 := \text{sample}_{\text{norm}}(a_1^n, θ_1, 1); x_2 := \text{sample}_{\text{norm}}(a_2^n, θ_2, E_i[x_1]))$$

where for some $n \geq 1$ and $c \neq 0 \in \mathbb{R}$,

$$E_i[x_1] \equiv \text{if } (x_i = 0) \text{ then } 1 \text{ else } \exp(-1/|x_i|^n), \quad E_2[x_1] \equiv \exp(c \cdot x_2^3).$$

Then, the entropies of $\text{dens}(D_{i,θ}, s_i)$'s are all undefined.

Proof. We prove that

$$\mathbb{E}_{\text{dens}(D_{i,θ}, s_i)(r)}[\log \text{dens}(D_{i,θ}, s_i)(r)] = \infty \quad \text{for all } i = 1, 2$$

by bounding the the expectation as follows.

$$\mathbb{E}_{\text{dens}(D_{i,θ}, s_i)(r)}[\log \text{dens}(D_{i,θ}, s_i)(r)]$$

$$= \mathbb{E}_{\mathcal{N}(x_1; θ_1, 1) \cdot \mathcal{N}(x_2; θ_2, E_i[x_1])}[\log \mathcal{N}(x_1; θ_1, 1) + \log \mathcal{N}(x_2; θ_2, E_i[x_1])]$$

$$\geq -\mathbb{E}_{\mathcal{N}(x_1; θ_1, 1) \cdot \mathcal{N}(x_2; θ_2, E_i[x_1])}[\log \mathcal{N}(x_1; θ_1, 1)]$$

$$+ \mathbb{E}_{\mathcal{N}(x_1; θ_1, 1) \cdot \mathcal{N}(x_2; θ_2, E_i[x_1])}[\log \mathcal{N}(x_2; θ_2, E_i[x_1])]$$

Here the two terms in the last equation are bounded as follows.

$$-\mathbb{E}_{\mathcal{N}(x_1; θ_1, 1) \cdot \mathcal{N}(x_2; θ_2, E_i[x_1])}[\log \mathcal{N}(x_1; θ_1, 1)]$$

$$= -\mathbb{E}_{\mathcal{N}(x_1; θ_1, 1)}[\log \sqrt{2π} - \frac{1}{2}(x_1 - θ_1)^2]$$

$$\geq -\mathbb{E}_{\mathcal{N}(x_1; θ_1, 1)}[\log \sqrt{2π} - \frac{1}{2} \mathbb{E}_{\mathcal{N}(x_1; θ_1, 1)}(x_1 - θ_1)^2]$$

$$= -\log \sqrt{2π} - \frac{1}{2} \cdot 1^2$$

$$\mathbb{E}_{\mathcal{N}(x_1; θ_1, 1) \cdot \mathcal{N}(x_2; θ_2, E_i[x_1])}[\log \mathcal{N}(x_2; θ_2, E_i[x_1])]

= \mathbb{E}_{\mathcal{N}(x_1; θ_1, 1) \cdot \mathcal{N}(x_2; θ_2, E_i[x_1])}[-\log \sqrt{2π} - \frac{(x_2 - θ_2)^2}{2E_i[x_1]^2} \cdot \log E_i[x_1]]$$

$$\geq -\mathbb{E}_{\mathcal{N}(x_1; θ_1, 1)}[\log \sqrt{2π} - \frac{1}{2} \mathbb{E}_{\mathcal{N}(x_1; θ_1, 1)}(x_2 - θ_2)^2 \cdot \frac{1}{E_i[x_1]^2}]$$

$$+ \mathbb{E}_{\mathcal{N}(x_1; θ_1, 1)}[\log E_i[x_1]]$$

$$= -\log \sqrt{2π} - \frac{1}{2} + \mathbb{E}_{\mathcal{N}(x_1; θ_1, 1)}[\log E_i[x_1]]$$

Note that

$$\mathbb{E}_{\mathcal{N}(x_1; θ_1, 1)}[\log E_i[x_1]] = \left\{ \begin{array}{ll}
\mathbb{E}_{\mathcal{N}(x_1; θ_1, 1)}[\mathbb{I}_{x_1=0} \cdot 1/|x_1|^n] & \text{for } i = 1 \\
\mathbb{E}_{\mathcal{N}(x_1; θ_1, 1)}[\exp(c \cdot x_1^3)] & \text{for } i = 2
\end{array} \right\} = \infty$$

because $\int_{[0,ε]} 1/|v|^n dv = \infty$ for any $ε > 0$ and $n \geq 1$, and $\mathbb{E}_{\mathcal{N}(v; μ, σ)}[\exp(c \cdot v^3)] = \infty$ for any $μ ∈ \mathbb{R}$ and $σ ∈ \mathbb{R}_{>0}$. Hence, we obtain the desired result. \(\square\)

Theorem 6.3. Under our assumption in §6.1, the condition (9) implies

$$\mathbb{E}_{\text{dens}(D_θ, s_i)(r)}[\log \text{dens}(C, s_i)(r)] < \infty,$$

and thus the requirement R3 (i.e., the objective in (5) is well-defined).
We show that the inequalities are indeed true. By the condition (9) and the parts (a) and (b) of Lemma C.1, we have that
\[ g_{i,\theta}(r) \triangleq \prod_{\beta \in K_i} \mathcal{N}(r(\beta); \mu_{i,\beta}(\theta), \sigma_{i,\beta}(\theta)), \]
\[ h_{i,\alpha}(r) \triangleq \mathcal{N}(r(\alpha); \mu'_{i,\alpha}(r), \sigma'_{i,\alpha}(r)), \quad k_{i,\beta}(r) \triangleq \mathcal{N}(\epsilon_{i,\beta}; \mu''_{i,\beta}(r), \sigma''_{i,\beta}(r)). \]
We note that
\[ \int_{\alpha} \int_{\beta} \mathcal{N}(r(\alpha); \mu'_{i,\alpha}(r), \sigma'_{i,\alpha}(r)) \mathcal{N}(r(\beta); \mu_{i,\beta}(\theta), \sigma_{i,\beta}(\theta)) \]
\[ = \int \rho(d\beta) \left( \prod_{i=1}^{M} \mathbb{1}_{[r \in A_i]} \cdot g_{i,\theta}(r) \right) \cdot \log \left( \prod_{i=1}^{M} \mathbb{1}_{[r \in A_i]} \cdot \prod_{\beta \in K_i} h_{i,\alpha}(r) \cdot \prod_{j \in [N_i]} k_{i,j}(r) \right) \]
\[ = \int \rho(d\beta) \left( \prod_{i=1}^{M} \mathbb{1}_{[r \in A_i]} \cdot g_{i,\theta}(r) \right) \cdot \log \left( \prod_{\beta \in K_i} h_{i,\beta}(r) \cdot \prod_{j \in [N_i]} k_{i,j}(r) \right) \]
\[ = \sum_{i=1}^{M} \int \rho(d\beta) \left( \mathbb{1}_{[r \in A_i]} \cdot g_{i,\theta}(r) \right) \cdot \sum_{\beta \in K_i} \log h_{i,\beta}(r) + \sum_{j \in [N_i]} \log k_{i,j}(r) \]
Thus, we can have
\[ \mathbb{E}_{g_{i,\theta}(r)} \left[ \log h_{i,\alpha}(r) \right] < \infty \quad \text{(17)} \]
if the following two inequalities hold:
\[ \mathbb{E}_{g_{i,\theta}(r)} \left[ \log \sigma'_{i,\alpha}(r) \right] < \infty \quad \text{and} \quad \mathbb{E}_{g_{i,\theta}(r)} \left[ \frac{(r(\alpha) - \mu'_{i,\alpha}(r))^2}{\sigma'_{i,\alpha}(r)^2} \right] < \infty. \quad \text{(18)} \]
We show that the inequalities are indeed true. By the condition (9) and the parts (a) and (b) of Lemma C.1 there exist functions \( f_i, u_i \in \mathcal{A}_i \) such that for every \( r \in A_i \),
\[ \log \sigma'_{i,\alpha}(r) \leq \exp(f_i(r)) \quad \text{and} \quad \frac{(r(\alpha) - \mu'_{i,\alpha}(r))^2}{\sigma'_{i,\alpha}(r)^2} \leq \exp(u_i(r)). \quad \text{(19)} \]
For every \( f \in \mathcal{A}_i \), by the part (e) of Lemma C.1 we have that
\[ \mathbb{E}_{g_{i,\theta}(r)} \left[ \exp(f(r)) \right] < \infty. \quad \text{(20)} \]
The inequalities in (18) follow from the properties in (19) and (20).
By an argument similar to what we have just given, we also have that
\[ \mathbb{E}_{g_{i,\theta}(r)} \left[ \log k_{i,j}(r) \right] < \infty. \quad \text{(21)} \]
Using (17) and (21), we calculate the claim of the theorem:
\[ \mathbb{E}_{\text{dens}(D_\theta, s_i)(r)} \left[ \log \text{dens}(C, s_i)(r) \right] \]
\[ = \int \rho(d\beta) \left( \prod_{i=1}^{M} \mathbb{1}_{[r \in A_i]} \cdot g_{i,\theta}(r) \right) \cdot \log \left( \prod_{i=1}^{M} \mathbb{1}_{[r \in A_i]} \cdot \prod_{\beta \in K_i} h_{i,\beta}(r) \cdot \prod_{j \in [N_i]} k_{i,j}(r) \right) \]
\[ = \int \rho(d\beta) \left( \prod_{i=1}^{M} \mathbb{1}_{[r \in A_i]} \cdot g_{i,\theta}(r) \right) \cdot \log \left( \prod_{\beta \in K_i} h_{i,\beta}(r) \cdot \prod_{j \in [N_i]} k_{i,j}(r) \right) \]
\[ = \sum_{i=1}^{M} \int \rho(d\beta) \left( \mathbb{1}_{[r \in A_i]} \cdot g_{i,\theta}(r) \right) \cdot \sum_{\beta \in K_i} \log h_{i,\beta}(r) + \sum_{j \in [N_i]} \log k_{i,j}(r) \]
The last inequality follows from (17) and (21). Finally, the requirement R3 follows from the above and Theorem 6.1:

\[
\int \rho(dr) \left| \frac{\text{dens}(D_\theta, s_j)(r)}{\text{dens}(C, s_j)(r)} \right| \log \left| \frac{\text{dens}(D_\theta, s_j)(r)}{\text{dens}(C, s_j)(r)} \right| \leq \mathbb{E}_{\text{dens}(D_\theta, s_j)(r)} \left[ \log \text{dens}(D_\theta, s_j)(r) \right] + \mathbb{E}_{\text{dens}(D_\theta, s_j)(r)} \left[ \log \text{dens}(C, s_j)(r) \right] < \infty.
\]

**Example 6.4.** Consider models \(C_1, \ldots, C_4\) and a guide \(D_\theta\) defined as follows:

\[
\begin{align*}
C_1 & \equiv (x_1 := \text{sample}_{\text{norm}}("a_1", 0, 1); x_2 := \text{sample}_{\text{norm}}("a_2", E_1[x_1], 1)) \quad \text{for } i = 1, 2 \\
C_2 & \equiv (x_1 := \text{sample}_{\text{norm}}("a_1", 0, 1); x_2 := \text{sample}_{\text{norm}}("a_2", 0, E_1[x_1]) \quad \text{for } i = 3, 4 \\
D_\theta & \equiv (x_1 := \text{sample}_{\text{norm}}("a_1", \theta_1, 1); x_2 := \text{sample}_{\text{norm}}("a_2", \theta_2, 1))
\end{align*}
\]

where for some \(n \geq 1\) and \(c \neq 0 \in \mathbb{R},\)

\[
E_1[x_1] \equiv 0 \text{ if } (x_1 = 0) \text{ then } 1/|x_1|^{n}, \quad E_2[x_1] \equiv 0 \text{ if } (x_1 = 0) \text{ then } 1/|x_1|^{n}.
\]

Then, the requirement R3 does not hold (i.e., the objective in (5) is undefined) for all \(i = 1, \ldots, 4\).

**Proof.** Since we have

\[
\int \rho(dr) \left| \frac{\text{dens}(D_\theta, s_j)(r)}{\text{dens}(C_1, s_j)(r)} \right| \geq -\mathbb{E}_{\text{dens}(D_\theta, s_j)(r)} \left[ \log \text{dens}(D_\theta, s_j)(r) \right] + \mathbb{E}_{\text{dens}(D_\theta, s_j)(r)} \left[ \log \text{dens}(C_1, s_j)(r) \right]
\]

it suffices to show that

\[
\mathbb{E}_{\text{dens}(D_\theta, s_j)(r)} \left[ \log \text{dens}(C_1, s_j)(r) \right] < \infty,
\]

which implies

\[
\mathbb{E}_{\text{dens}(D_\theta, s_j)(r)} \left[ \log \text{dens}(C_1, s_j)(r) \right] = \infty \quad \text{for all } i = 1, \ldots, 4.
\]

**Case of \(i = 1, 2\):** The quantity in (22) is bounded as follows.

\[
\mathbb{E}_{\text{dens}(D_\theta, s_j)(r)} \left[ \log \text{dens}(C_1, s_j)(r) \right] = \mathbb{E}_{\mathcal{N}(x_1; \theta_1, 1)} \cdot \mathbb{E}_{\mathcal{N}(x_2; \theta_2, 1)} \left[ \log \mathcal{N}(x_1; 0, 1) + \log \mathcal{N}(x_2; E_1[x_1], 1) \right]
\]
We bound the last term in the last equation as follows, assuming we additionally use the following fact:

\[
\begin{align*}
&\frac{\log{\mathcal{N}(x_1; 0, 1)}}{\log{\mathcal{N}(x_2; E_i[x_1], 1)}}
\end{align*}
\]

Here the two terms in the last equation are bounded as follows.

\[
\begin{align*}
-\mathbb{E}_{N(x_1; \theta_1, 1), N(x_2; \theta_2, 1)}[\log{\mathcal{N}(x_1; 0, 1)}] &
-\mathbb{E}_{N(x_1; \theta_1, 1), N(x_2; \theta_2, 1)}[\log{\mathcal{N}(x_2; E_i[x_1], 1)}]
\end{align*}
\]

We bound the last term in the last equation as follows, assuming \( c > 0 \).

\[
\begin{align*}
&\mathbb{E}_{N(x_1; \theta_1, 1), N(x_2; \theta_2, 1)}[E_i[x_1]^2 - 2x_2 \cdot E_i[x_1]]
\end{align*}
\]

Here the last inequality comes from that the two expectations over \( x_2 \) are finite while the other two over \( x_1 \) are both \( \infty \), because \( \int_{0}^{\infty} 1/v^a \, dv = \infty \) for any \( \epsilon > 0 \) and \( n \geq 1 \), and \( \mathbb{E}_{N(\nu; \mu, \sigma)}[\exp(c \cdot v^3)] = \infty \) for any \( \mu \in \mathbb{R} \) and \( \sigma \in \mathbb{R_{>0}} \). Note that we can use a similar calculation for the case of \( c < 0 \). This proves the equality in (22).

**Case of \( i = 3, 4 \):** The proof is similar to the above case, so we omit it. Remark that in the proof, we additionally use the following fact: \( \int_{0}^{1} \log{v} \, dv < \infty \). □

**Lemma C.1.** Pick \( i \in [M] \). Let \((\alpha_1, \ldots, \alpha_J)\) be an enumeration of the elements in \( K_i \), and \( \bar{r} \equiv (r(\alpha_1), \ldots, r(\alpha_J)) \in \mathbb{R}^J \) for \( r \in A_i \).

(a) For every \( l_1, l_2 \in \mathcal{A}_i \) and every \( \odot \in \{+, -, \times, \div, \max\} \), there exists \( l \in \mathcal{A}_i \) such that \( \exp(l_1(r)) \odot \exp(l_2(r)) \leq \exp(l(r)) \) for all \( r \in A_i \).

(b) For every polynomial \( \text{poly} : \mathbb{R}^J \to \mathbb{R} \), there exists \( l \in \mathcal{A}_i \) such that \( |\text{poly}(\bar{r})| \leq \exp(l(r)) \) for all \( r \in A_i \).
In particular, for every \( l \in A_i \), there exists \( l' \in A_i \) such that
\[
|l(r)| \leq \exp(l'(r)) \text{ for all } r \in A_i.
\]

(c) For every affine-bounded neural network \( nn : \mathbb{R}^l \to \mathbb{R} \), there exists \( l \in A_i \) such that
\[
|nn(r)| \leq l(r) \text{ for all } r \in A_i.
\]

(d) There exist affine functions \( l_1, l_2 : \mathbb{R} \to \mathbb{R} \) such that
\[
\exp(l_1(|v|)) \leq \text{softplus}(v) \leq \exp(l_2(|v|)) \text{ for all } v \in \mathbb{R}.
\]

(e) For every \( l \in A_i \) and every \( J \)-dimensional normal distribution \( q \),
\[
\mathbb{E}_{q(r(\alpha_1), \ldots, r(\alpha_J))}[\exp(l(r))] < \infty.
\]

**Proof of (a).** We split the cases.
- Case \( \odot \in \{\times, /\} \): Take \( l = l_1 \pm l_2 \).
- Case \( \odot = \max \): For \( m = 1, 2 \), let
\[
l_m(r) = c_{m,0} + \sum_{j=1}^{J} c_{m,j} \cdot |r(\alpha_j)|
\]
where \( c_{m,j} \in \mathbb{R} \). For all \( r \in A_i \), we have
\[
\max(l_1(r), l_2(r)) \leq \max \left( c_{1,0} + \sum_{j=1}^{J} |c_{1,j}| \cdot |r(\alpha_j)|, \ c_{2,0} + \sum_{j=1}^{J} |c_{2,j}| \cdot |r(\alpha_j)| \right) \leq l(r),
\]
where
\[
l(r) \equiv \max(c_{1,0}, c_{2,0}) + \sum_{j=1}^{J} \max(|c_{1,j}|, |c_{2,j}|) \cdot |r(\alpha_j)| \in A_i.
\]
Hence, for every \( r \in A_i \),
\[
\max(\exp(l_1(r)), \exp(l_2(r))) = \exp(\max(l_1(r), l_2(r))) \leq \exp(l(r)).
\]
- Case \( \odot \in \{+, -\} \): Using the case \( \odot = \max \), we have the following for some \( l \in A_i \):
\[
\exp(l_1(r)) + \exp(l_2(r)) \leq 2 \cdot \max(\exp(l_1(r)), \exp(l_2(r)))
\]
\[
\leq \exp(\log 2) \cdot \exp(l(r))
\]
\[
= \exp(l(r) + \log 2).
\]

**Proof of (b).** Let \( n \) be an index in \( \{1, \ldots, J\} \) such that
\[
\text{poly}(r(\alpha_1), \ldots, r(\alpha_J)) = \text{poly}'(r(\alpha_1), \ldots, r(\alpha_n))
\]
for some polynomial \( \text{poly}' : \mathbb{R}^n \to \mathbb{R} \). We show that there exists \( l \in A_i \) satisfying
\[
\text{poly}'(r(\alpha_1), \ldots, r(\alpha_n)) \leq \exp(l(r)).
\]
Note that the claim of the lemma follows from this. Our proof is by induction on \( n \).
- Case \( n = 1 \): Let \( v \in \mathbb{R} \) and \( \text{poly}'(v) = \sum_{d=0}^{D} c_d \cdot v^d \) where \( c_d \in \mathbb{R} \). We use the following fact.
**FACT:** For every \( d \in \mathbb{N} \), there exists an affine function \( l_d : \mathbb{R} \to \mathbb{R} \) such that
\[
|v|^d \leq \exp(l_d(|v|)) \text{ for all } v \in \mathbb{R}.
\]
We prove the claim of the lemma by induction on the depth $d$. Towards Verified Stochastic Variational Inference for Probabilistic Programs 16:59

such that (i) $n_1 = 1$ and $n_{d+1} = 1$; (ii) $nn = f_d \circ \cdots \circ f_1$; and (iii) for all $1 \leq j \leq d$ and $v \in \mathbb{R}^{n_j}$,

$$\sum_{k=1}^{n_{j+1}} |f_j(v)_k| \leq l'_j(|v_1|, \ldots, |v_{n_j}|).$$

We prove the claim of the lemma by induction on the depth $d$. For $d = 1$, the claim follows immediately from the defining property (iii) of the affine-bounded neural network $nn$. Assume

\textbf{Proof of (c).} By the definition of affine-bounded neural network, there exist functions $f_1 : \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$, $f_2 : \mathbb{R}^{n_2} \to \mathbb{R}^{n_3}$, \ldots, $f_d : \mathbb{R}^{n_d} \to \mathbb{R}^{n_{d+1}}$

and affine functions $l'_j : \mathbb{R}^{n_j} \to \mathbb{R}$ for all $1 \leq j \leq d$

such that (i) $n_1 = 1$ and $n_{d+1} = 1$; (ii) $nn = f_d \circ \cdots \circ f_1$; and (iii) for all $1 \leq j \leq d$ and $v \in \mathbb{R}^{n_j}$,
\[ d > 1. \text{By induction hypothesis, there exist affine functions } l_1, \ldots, l_{n_d} : \mathbb{R}^J \rightarrow \mathbb{R} \text{ such that for all } k \in \{1, \ldots, n_d\} \text{ and } v \in \mathbb{R}^J, \]
\[ |(f_{d-1} \circ \cdots \circ f_1)(v)_k| \leq l_k([v_1], \ldots, [v_J]). \]

This is because the \((f_{d-1} \circ \cdots \circ f_1)(v)_k\) are affine-bounded neural networks of depth \(d - 1\). Let
\[ l'_d(v_1, \ldots, v_{n_d}) = c_0 + \sum_{k=1}^{n_d} c_k \cdot |v_k|. \]

Using what we have proved so far, we calculate:
\[ |nn(v)| = |f_d((f_{d-1} \circ \cdots \circ f_1)(v))| \]
\[ \leq l'_d((f_{d-1} \circ \cdots \circ f_1)(v)_1, \ldots, (f_{d-1} \circ \cdots \circ f_1)(v)_{n_d}) \]
\[ = c_0 + \sum_{k=1}^{n_d} c_k \cdot |(f_{d-1} \circ \cdots \circ f_1)(v)_k| \]
\[ \leq c_0 + \sum_{k=1}^{n_d} |c_k| \cdot |(f_{d-1} \circ \cdots \circ f_1)(v)_k| \]
\[ \leq c_0 + \sum_{k=1}^{n_d} |c_k| \cdot l_k([v_1], \ldots, [v_J]). \]

Let \( l(r) = c_0 + \sum_{j=1}^{n_d} |c_j| \cdot l_k([r(a_1)], \ldots, [r(a_J)]) \). The function \( l \) is essentially the composition of two affine functions, and so it belongs to \( \mathcal{A}_l \). This and the conclusion of the above calculation show that \( l \) is the function in \( \mathcal{A}_l \) that we look for.

**Proof of (d).** Since \( \log(1 + v) \geq v/2 \) for all \( v \in [0, 1] \), we have
\[ \log(1 + \exp(v)) \geq \exp(v)/2 = \exp(-|v| - \log 2) \quad \text{for all } v \in (-\infty, 0], \quad \text{and} \]
\[ \log(1 + \exp(v)) \leq \log(2 \cdot \exp(v)) = |v| + \log 2 \quad \text{for all } v \in [0, \infty). \]

Let \( l_1(v) = -v - \log 2 \) and \( l_2(v) = v + \log 2 \). As \( \text{softplus}(\cdot) \) is increasing, we have
\[ \exp(l_1(|v|)) \leq \text{softplus}(v) \leq \exp(l_2(|v|)) \quad \text{for all } v \in \mathbb{R}. \]

**Proof of (e).** Let \( l(r) = c_0 + \sum_{j=1}^{J} c_j \cdot |r(a_j)| \), where \( c_j \in \mathbb{R} \). The moment generating function \( G_p : \mathbb{R}^J \rightarrow \mathbb{R} \) of a random variable of density \( p(v) \) on \( \mathbb{R}^J \) is defined by \( G_p(t) = \mathbb{E}_p(v) [\exp(t \cdot v)] \) where \( t \cdot v \) is the inner product of \( t \) and \( v \). It is well-known that the moment generating functions are well defined for multivariate normal distributions \( q(v) \) on \( \mathbb{R}^J \) in the lemma, i.e., \( G_q(t) < \infty \) for all \( t \in \mathbb{R}^J \). From this fact and that \( \exp(|v|) \leq \exp(v) + \exp(-v) \), it follows that
\[ \mathbb{E}_q(r(a_1), \ldots, r(a_J)) [\exp(l(r))] = \exp(c_0) \cdot \mathbb{E}_q(r(a_1), \ldots, r(a_J)) \left[ \exp \left( \sum_{j=1}^{J} c_j \cdot |r(a_j)| \right) \right] \]
\[ \leq \exp(c_0) \cdot \sum_{k_1, \ldots, k_J \in \{0, 1\}} G_q \left( (-1)^{k_1} c_1, \ldots, (-1)^{k_J} c_J \right) \]
\[ < \infty. \]

The proof of Theorem 6.6 uses two more ingredients: Theorem C.2 which, compared to Theorem 6.7, additionally guarantees the continuity of \( \nabla_v \int f_v \, d\mu \) by assuming the continuity of \( \nabla_v f_v \);
and Lemma C.3 which bounds the densities of multiple normal distributions using the density of a single, fixed normal distribution.

**Theorem C.2.** Let $V \subset \mathbb{R}$ be an open interval, and $(X, \Sigma, \mu)$ be a measure space. Suppose that a measurable function $f : V \times X \rightarrow \mathbb{R}$ satisfies the following conditions:

(a) The function $\nu \in V \mapsto \int \mu(dx) f_\nu(x) \in \mathbb{R}$ is well-defined.

(b) For almost all $x \in X$ (w.r.t. $\mu$), the function $\nu \in V \mapsto \nabla_v f_\nu(x) \in \mathbb{R}$ is well-defined, continuous.

(c) There is a measurable function $h : X \rightarrow \mathbb{R}$ such that $\int \mu(dx) h(x)$ is well-defined and $|\nabla_v f_\nu(x)| \leq h(x)$ for all $\nu \in V$ and almost all $x \in X$ (w.r.t. $\mu$).

Then, for all $\nu \in V$, both sides of the below equation are well-defined, and the equality holds:

$$\nabla_v \int \mu(dx) f_\nu(x) = \int \mu(dx) \nabla_v f_\nu(x).$$

Moreover, the function $\nu \in V \mapsto \nabla_v \int \mu(dx) f_\nu(x)$ is continuous.

**Proof.** Theorem 6.7 implies the former part of the conclusion: for all $\nu \in V$, each side of (23) is well-defined and the equation (23) holds. To prove the latter part of the conclusion, define $G : V \rightarrow \mathbb{R}$ and $g : V \times X \rightarrow \mathbb{R}$ by

$$G(\nu) \triangleq \nabla_v \int \mu(dx) f_\nu(x) = \int \mu(dx) g(\nu, x), \quad g(\nu, x) \triangleq \nabla_v f_\nu(x),$$

and let $(\nu_n)_{n \in \mathbb{N}}$ be any sequence in $V$ such that $\lim_{n \rightarrow \infty} \nu_n = \nu^* \in V$. Then, it suffices to show that

$$\lim_{n \rightarrow \infty} G(\nu_n) = G(\nu^*).$$

To apply the dominated convergence theorem, define $g_n, g^* : X \rightarrow \mathbb{R}$ by $g_n(x) \triangleq g(\nu_n, x)$ and $g^*(x) \triangleq g(\nu^*, x)$. By (b), the function $g(\cdot, x) : V \rightarrow \mathbb{R}$ is continuous for almost all $x \in X$ (w.r.t. $\mu$), so

$$\lim_{n \rightarrow \infty} g_n(x) = \lim_{n \rightarrow \infty} g(\nu_n, x) = g(\nu^*, x) = g^*(x) \quad \text{for almost all } x \in X \text{ (w.r.t. } \mu).$$

By (c), for all $n \in \mathbb{N}$, $|g_n(x)| \leq h(x)$ for almost all $x \in X$ (w.r.t. $\mu$), and $h(x)$ is integrable (w.r.t. $\mu$). Therefore, we can apply the dominated convergence theorem to $(g_n)_{n \in \mathbb{N}}$ and $g^*$ with $h$, which gives:

$$\lim_{n \rightarrow \infty} G(\nu_n) = \lim_{n \rightarrow \infty} \int \mu(dx) g_n(x) = \int \mu(dx) g^*(x) = G(\nu^*).$$

**Lemma C.3.** Suppose $[\mu_1, \mu_2] \subset \mathbb{R}$ and $[\sigma_1, \sigma_2] \subset (0, \infty)$ Then, there exists $c \in \mathbb{R}$ such that

$$\mathcal{N}(\nu; \mu, \sigma) \leq c \cdot \mathcal{N}(\nu; 0, 2\sigma_2)$$

for all $\nu \in \mathbb{R}$, $\mu \in [\mu_1, \mu_2]$, and $\sigma \in [\sigma_1, \sigma_2]$.

**Proof.** We claim that $c = \frac{2\sigma_2}{\sigma_1} \exp\left(\frac{1}{6\sigma_1^2} \max(\mu_1^2, \mu_2^2)\right)$ is the desired coefficient. For the constant $c$, the claim we need show is that

$$\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\nu - \mu)^2}{2\sigma^2}\right) \leq \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\nu^2}{8\sigma^2} + \frac{1}{6\sigma_1^2} \max(\mu_1^2, \mu_2^2)\right)
$$

for any $\nu \in \mathbb{R}$, $\mu \in [\mu_1, \mu_2]$, and $\sigma \in [\sigma_1, \sigma_2]$. Since $\frac{1}{\sqrt{2\pi}} \leq \frac{1}{\sqrt{2\pi} \sigma}$, it suffices to show

$$\frac{\nu^2}{8\sigma^2} + \frac{1}{6\sigma_1^2} \max(\mu_1^2, \mu_2^2) \geq \frac{(\nu - \mu)^2}{2\sigma^2}
$$

$$\iff \frac{1}{2\sigma^2} \left(a\nu^2 - 2\nu \mu + \mu^2 + \frac{\sigma^2}{3\sigma_1^2} \max(\mu_1^2, \mu_2^2)\right) \geq 0,$$

where $a = 1 - \frac{\sigma^2}{4\sigma_1^2}$.
\[ a \left( \frac{\mu}{a} \right)^2 + \left( 1 - \frac{1}{a} \right) \mu^2 + \frac{\sigma^2}{3\sigma_i^2} \max(\mu_i^2, \mu_2^2) \geq 0 \]
\[ \left( 1 - \frac{1}{a} \right) \mu^2 + \frac{\sigma^2}{3\sigma_i^2} \max(\mu_i^2, \mu_2^2) \geq 0, \quad \text{since} \quad a = 1 - \frac{\sigma^2}{4\sigma_i^2} \geq \frac{3}{4} \]

The last inequality holds because

\[ \left( 1 - \frac{1}{a} \right) \mu^2 + \frac{\sigma^2}{3\sigma_i^2} \max(\mu_i^2, \mu_2^2) \geq -\frac{1}{3} \mu^2 + \frac{1}{3} \max(\mu_i^2, \mu_2^2) \geq 0. \]

\[ \square \]

**Theorem 6.6.** If both our assumption in §6.1 and the condition (9) hold, then the condition (10) implies the requirements R4-R6.

**Proof.** Before staring the proof, we note that by the condition (10), there is an open interval \( V_{(i,a,j,\theta)} \subseteq \mathbb{R} \) for all \( i \in [M], \alpha \in K_i, j \in [p] \) and \( \theta \in \mathbb{R}^p \), such that

- \( V_{(i,a,j,\theta)} \) contains \( \theta_j \);
- the function \( \nu \in \mathbb{R} \mapsto \mu_{(i,a)}(\theta | j : y) \) is continuously differentiable in \( V_{(i,a,j,\theta)} \);
- the function \( \nu \in \mathbb{R} \mapsto \sigma_{(i,a)}(\theta | j : y) \) is continuously differentiable in \( V_{(i,a,j,\theta)} \), and \( \inf \{ \sigma_{(i,a)}(\theta | j : y) \mid \nu \in V_{(i,a,j,\theta)} \} > 0 \).

Let \( F_1 \) and \( F_2 \) be functions from \( \mathbb{R}^p \times \text{RDB} \) to \( \mathbb{R} \) defined as follows: for all \( \theta \in \mathbb{R}^p \) and \( r \in \text{RDB} \),

\[ F_1(\theta, r) \triangleq \text{dens}(D_{\theta}, s_i)(r) \cdot \log \frac{\text{dens}(D_{\theta}, s_i)(r)}{\text{dens}(C, s_i)(r)} \quad \text{and} \quad F_2(\theta, r) \triangleq \text{dens}(D_{\theta}, s_i)(r). \]

We have to show that for every \( \theta \in \mathbb{R}^p \) and \( j \in [p] \),

- the partial derivatives
  \[ \nabla_{\theta_j} \int \rho(dr) F_1(\theta, r) \quad \text{and} \quad \nabla_{\theta_j} \int \rho(dr) F_2(\theta, r) \]
  are well-defined and are continuous functions of \( \theta_j \in \mathbb{R} \);
- these derivatives commute with integration:
  \[ \nabla_{\theta_j} \int \rho(dr) F_1(\theta, r) = \int \rho(dr) \nabla_{\theta_j} F_1(\theta, r) \quad \text{and} \quad \nabla_{\theta_j} \int \rho(dr) F_2(\theta, r) = \int \rho(dr) \nabla_{\theta_j} F_2(\theta, r). \]

We discharge these proof obligations using Theorem C.2. Pick \( \theta^* \in \mathbb{R}^p \) and \( j \in [p] \). We instantiate Theorem C.2 twice for \( F_1 \) and \( F_2 \) as follows.

- For both cases of \( F_1 \) and \( F_2 \), we set \( V \) to be an open interval \( (a, b) \) such that both \( a \) and \( b \) are finite, the closure \( [a, b] \) is contained in \( V_{(j, \theta^*)} = \bigcap_{i \in [M]} \bigcap_{\alpha \in K_i} V_{(i, a, j, \theta^*)} \), and \( (a, b) \) contains \( \theta_j^* \). Note that such an open interval exists because \( V_{(j, \theta^*)} \) is open and contains \( \theta_j^* \).
- The measurable space \( (X, \Sigma) \) in the theorem is RDB with its \( \sigma \)-algebra in both cases.
- The measure \( \mu \) is the reference measure \( \rho \) on RDB in both cases.
- The measurable function \( f \) is instantiated twice for \( (\nu, r) \mapsto F_i(\theta^* | j : y), r \) with \( i = 1, 2 \).

Under this instantiation, the conclusion of Theorem C.2 implies all of the requirements R4-R6.

It remains to show that both instantiations satisfy the three provisos of Theorem C.2 marked (a), (b) and (c). We tackle the provisos one-by-one. We use the notations in §6.1 without recalling them explicitly. For \( i \in [M], l \in [N_i], \alpha \in K_i, \nu \in \mathbb{R}, r \in A_i \) and \( \theta \in \mathbb{R}^p \), let

\[ g_{(i, \alpha, \theta)}(\nu) \triangleq \mathcal{N}(\nu; \mu_{(i, \alpha)}(\theta), \sigma_{(i, \alpha)}(\theta)), \quad g_{(i, \theta)}(r) \triangleq \prod_{\beta \in K_i} g_{(i, \beta, \theta)}(r(\beta)). \]
Towards Verified Stochastic Variational Inference for Probabilistic Programs

\[ h_{(i,\alpha)}(r) \triangleq \mathcal{N}(r(\alpha) ; \mu_{(i,\alpha)}(r), \sigma'_{(i,\alpha)}(r)), \quad h_{(r)}(r) \triangleq \prod_{\beta \in K_i} h_{(i,\beta)}(r), \]

\[ k_{(i,t)}(r) \triangleq \mathcal{N}(c_{(i,t)} ; \mu'_{(i,t)}(r), \sigma''_{(i,t)}(r)), \quad k_{(r)}(r) \triangleq \prod_{m=1}^{N_i} k_{(i,m)}(r). \]

**Proof of (a).** Since we assume the condition 9, Theorem 6.3 ensures that the integral \( \int \mu(dr) \mathcal{F}_1(\theta^*[j : v]) \) is well-defined for all \( v \in V \). The other integral involving \( \mathcal{F}_2 \) is the integral of the density of \( D_{\theta^*[j : v]} \). Since \( D_{\theta^*[j : v]} \) is a guide, the integral is well-defined and has the value 1.

**Proof of (b).** Note that

\[
\begin{align*}
\mathcal{F}_1(\theta^*[j : v], r) &= \text{dens}(D_{\theta^*[j : v]}, s_I)(r) \cdot \log \frac{\text{dens}(D_{\theta^*[j : v]}, s_I)(r)}{\text{dens}(C, s_I)(r)} \\
&= \sum_{i=1}^{M} \mathbb{I}_{[r \in A_i]} \cdot g(i, \theta^*[j : v])(r) \cdot \log \frac{g(i, \theta^*[j : v])(r)}{h_i(r) \cdot k_i(r)} \\
&= \sum_{i=1}^{M} \mathbb{I}_{[r \in A_i]} \cdot g(i, \theta^*[j : v])(r) \cdot \left( -\log(h_i(r) \cdot k_i(r)) + \log(g(i, \theta^*[j : v])(r)) \right) \\
&= \sum_{i=1}^{M} \mathbb{I}_{[r \in A_i]} \cdot \prod_{\alpha \in K_i} \mathcal{N}(r(\alpha) ; \mu_{(i,\alpha)}(\theta^*[j : v]), \sigma_{(i,\alpha)}(\theta^*[j : v])) \\
&\quad \cdot \left( -\log(h_i(r) \cdot k_i(r)) + \sum_{\alpha \in K_i} \log \mathcal{N}(r(\alpha) ; \mu_{(i,\alpha)}(\theta^*[j : v]), \sigma_{(i,\alpha)}(\theta^*[j : v])) \right),
\end{align*}
\]

\[
\mathcal{F}_2(\theta^*[j : v], r) = \text{dens}(D_{\theta^*[j : v]}, s_I)(r) \\
&= \sum_{i=1}^{M} \mathbb{I}_{[r \in A_i]} \cdot \prod_{\alpha \in K_i} \mathcal{N}(r(\alpha) ; \mu_{(i,\alpha)}(\theta^*[j : v]), \sigma_{(i,\alpha)}(\theta^*[j : v])).
\]

Recall that every \( r \in A_i \) is a map from \( K_i \) to \( \mathbb{R} \). The normal density \( \mathcal{N}(\nu' ; \mu', \sigma') \) and its log are smooth functions on \( (\nu', \mu', \sigma') \in \mathbb{R} \times \mathbb{R} \times (0, \infty) \). Also, by the condition (10), the functions

\[
\nu \in V_{(i,\alpha,j,\theta^*)} \mapsto \mu_{(i,\alpha)}(\theta^*[j : v]) \quad \text{and} \quad \nu \in V_{(i,\alpha,j,\theta^*)} \mapsto \sigma_{(i,\alpha)}(\theta^*[j : v])
\]

are continuously differentiable for all \( i \in [M] \) and \( \alpha \in K_i \). Thus, they are also continuously differentiable when we restrict their domains to \( V \). One more thing to notice is that neither \( \mathbb{I}_{[r \in A_i]} \) nor \( -\log(h_i(r) \cdot k_i(r)) \) depends on \( v \), so that they can be viewed as constant functions on \( v \), which are obviously smooth. Now note that both \( \nu \in V \mapsto \mathcal{F}_1(\theta^*[j : v], r) \) and \( \nu \in V \mapsto \mathcal{F}_2(\theta^*[j : v], r) \) are obtained from these smooth or continuously differentiable functions by function composition, addition and multiplication. Thus, they are continuously differentiable.

**Proof of (c).** For all \( i \in [M] \), we will construct measurable function \( H_i, H_i' : A_i \to [0, \infty) \) such that

- both \( \int \rho(dr) (\mathbb{I}_{[r \in A_i]} \cdot H_i(r)) \) and \( \int \rho(dr) (\mathbb{I}_{[r \in A_i]} \cdot H_i'(r)) \) are well-defined;
- for all \( r \in A_i \) and \( \nu \in V \),

\[
\nabla_\nu \left( g(i, \theta^*[j : v])(r) \cdot \log \frac{g(i, \theta^*[j : v])(r)}{h_i(r) \cdot k_i(r)} \right) \leq H_i(r);
\]

- for all \( r \in A_i \) and \( \nu \in V \),

\[
\nabla_\nu g(i, \theta^*[j : v])(r) \leq H_i'(r).
\]
Once we have such $H_i$’s, we can use them to define desired measurable functions $H, H' : RDB \rightarrow \mathbb{R}$ as follows:

$$H(r) = \sum_{i=1}^{M} 1_{[r \in A_i]} \cdot H_i(r) \quad \text{and} \quad H'(r) = \sum_{i=1}^{M} 1_{[r \in A_i]} \cdot H'_i(r).$$

Since both $H_i$ and $H'_i$ are integrable for all $i$, so are $H$ and $H'$. We prove this only for $H$ below. The case for $H'$ is similar.

$$\int \rho(dr) |H(r)| \leq \int \rho(dr) \sum_{i=1}^{M} 1_{[r \in A_i]} \cdot |H_i(r)| = \sum_{i=1}^{M} \int \rho(dr) 1_{[r \in A_i]} \cdot |H_i(r)| < \infty.$$ 

Both $H_i$ and $H'_i$ bound the magnitudes of the partial derivatives involving $g(i, \theta^*[j:v])$ for every $i$. Using this, we show that $H$ and $H'$ meet the condition about bounding the magnitude of a derivative:

$$|\nabla_v F_1(\theta^*[j:v], r)| = \left| \nabla_v \sum_{i=1}^{M} 1_{[r \in A_i]} \cdot \left( g(i, \theta^*[j:v])(r) \cdot \left( - \log(\alpha_i(r) \cdot k_i(r)) + \log(g(i, \theta^*[j:v])(r)) \right) \right) \right|$$

$$\leq \sum_{i=1}^{M} 1_{[r \in A_i]} \cdot |\nabla_v g(i, \theta^*[j:v])(r)|$$

$$\leq \sum_{i=1}^{M} 1_{[r \in A_i]} \cdot |H_i(r)| = H(r).$$

$$|\nabla_v F_2(\theta^*[j:v], r)| = \left| \nabla_v \sum_{i=1}^{M} 1_{[r \in A_i]} \cdot g(i, \theta^*[j:v])(r) \right|$$

$$\leq \sum_{i=1}^{M} 1_{[r \in A_i]} \cdot |\nabla_v g(i, \theta^*[j:v])(r)\right|$$

$$\leq \sum_{i=1}^{M} 1_{[r \in A_i]} \cdot |H'_i(r)| = H'(r).$$

It remains to construct measurable $H_i, H'_i : A_i \rightarrow \mathbb{R}$ with the required properties for every $i \in [M]$. Pick $i \in [M]$. Note that

$$\nabla_v \left( g(i, \theta^*[j:v])(r) \cdot \log \frac{g(i, \theta^*[j:v])(r)}{h_i(r) \cdot k_i(r)} \right)$$

$$= \left( \nabla_v g(i, \theta^*[j:v])(r) \right) \cdot \log \frac{g(i, \theta^*[j:v])(r)}{h_i(r) \cdot k_i(r)} + \left( g(i, \theta^*[j:v])(r) \cdot \nabla_v \log(g(i, \theta^*[j:v])(r)) \right)$$

$$= \left( \nabla_v g(i, \theta^*[j:v])(r) \right) \cdot \left( \log g(i, \theta^*[j:v])(r) \frac{g(i, \theta^*[j:v])(r)}{h_i(r) \cdot k_i(r)} + 1 \right).$$

We will build $H_i$ and $H'_i$ by finding good bounds for $|\nabla_v g(i, \theta^*[j:v])(r)|$, $|\log g(i, \theta^*[j:v])(r)|$ and $|\log h_i(r) - k_i(r)|$ and then combining these bounds. We do so in three steps.

First, we build functions that bound $|\nabla_v g(i, \theta^*[j:v])(r)|$ and $|\log g(i, \theta^*[j:v])(r)|$ over $v \in V$. For every $\alpha \in K_i$,

$$\nabla_v \log g(i, \alpha, \theta^*[j:v]) = \nabla_v \left( - \log \sqrt{2\pi} - \log \sigma(i, \alpha)(\theta^*[j:v]) - \frac{(w - \mu(i, \alpha)(\theta^*[j:v]))^2}{2\sigma(i, \alpha)(\theta^*[j:v])^2} \right)$$

$$= \nabla_v \left( \frac{\alpha(i, \alpha)(\theta^*[j:v]) - w}{\sigma(i, \alpha)(\theta^*[j:v])^2} \right) - \left( \frac{\alpha(i, \alpha)(\theta^*[j:v]) - w}{\sigma(i, \alpha)(\theta^*[j:v])^2} \right) \nabla_v \mu(i, \alpha)(\theta^*[j:v])$$

Towards Verified Stochastic Variational Inference for Probabilistic Programs

\[
\begin{align*}
\hat{\nabla}_v g_{(i, \theta^*[j:v])}(r) &= \hat{\nabla}_v \prod_{\alpha \in K_i} g_{(i, \alpha, \theta^*[j:v])}(r(\alpha)) \\
&= \sum_{\alpha \in K_i} \hat{\nabla}_v g_{(i, \alpha, \theta^*[j:v])}(r(\alpha)) \cdot \prod_{\beta \in K_i, \beta \neq \alpha} g_{(i, \beta, \theta^*[j:v])}(r(\beta)) \\
&= \sum_{\alpha \in K_i} g_{(i, \alpha, \theta^*[j:v])}(r(\alpha)) \cdot \hat{\nabla}_v \log g_{(i, \alpha, \theta^*[j:v])}(r(\alpha)) \cdot \prod_{\beta \in K_i, \beta \neq \alpha} g_{(i, \beta, \theta^*[j:v])}(r(\beta)) \\
&= g_{(i, \theta^*[j:v])}(r) \cdot \sum_{\alpha \in K_i} \hat{\nabla}_v \log g_{(i, \alpha, \theta^*[j:v])}(r(\alpha)) \\
&= g_{(i, \theta^*[j:v])}(r) \cdot \sum_{\alpha \in K_i} \left( \hat{\nabla}_v \sigma_{(i, \alpha)}(\theta^*[j:v]) \cdot \frac{(r(\alpha) - \mu_{(i, \alpha)}(\theta^*[j:v]))^2 - \sigma^2_{(i, \alpha)}(\theta^*[j:v])}{\sigma_{(i, \alpha)}(\theta^*[j:v])^3} \\
&\quad + \hat{\nabla}_v \mu_{(i, \alpha)}(\theta^*[j:v]) \cdot \frac{\sigma_{(i, \alpha)}(\theta^*[j:v])}{\sigma_{(i, \alpha)}(\theta^*[j:v])^2} \right) \\
&= g_{(i, \theta^*[j:v])}(r) \cdot \sum_{\alpha \in K_i} \sum_{n=0}^2 G_{(1, \alpha, n)}(v) \cdot r(\alpha)^n
\end{align*}
\]

for some continuous functions \(G_{(1, \alpha, n)} : V_{(i, \theta^*)} \to \mathbb{R}\) that do not depend on \(r\). The continuity of \(G_{(1, \alpha, n)}\) follows from our condition (10). Also,

\[
\log g_{(i, \theta^*[j:v])}(r) = \sum_{\alpha \in K_i} \log g_{(i, \alpha, \theta^*[j:v])}(r(\alpha))
\]

\[
= \sum_{\alpha \in K_i} \left( -\log \sqrt{2\pi} - \log \sigma_{(i, \alpha)}(\theta^*[j:v]) - \frac{(r(\alpha) - \mu_{(i, \alpha)}(\theta^*[j:v]))^2}{2\sigma^2_{(i, \alpha)}(\theta^*[j:v])} \right)
\]

\[
= \sum_{\alpha \in K_i} \sum_{n=0}^2 G_{(2, \alpha, n)}(v) \cdot r(\alpha)^n
\]

for some continuous functions \(G_{(2, \alpha, n)} : V_{(j, \theta^*)} \to \mathbb{R}\) that do not depend on \(r\). The continuity of \(G_{(2, \alpha, n)}\) follows from the condition (10). Since \(\overline{V} \subset V_{(j, \theta^*)}\), the functions \(G_{(m, \alpha, n)}\) are all continuous on the closed (bounded) interval \(\overline{V}\), where \(\overline{V}\) is the closure of \(V\). Thus, by the extreme value theorem,

\[
C_{\max} \triangleq \max_{(m, \alpha, n)} \sup_{v \in \overline{V}} |G_{(m, \alpha, n)}(v)| < \infty.
\]
Since the term \( g_{i, \theta^*[j:v]}(r) \) in (24) still depends on \( v \), we bound it over \( v \in V \) using Lemma C.3 which is instantiated with
\[
\mu_1 \triangleq \min_{a \in K_i} \inf_{v \in \overline{V}} \mu_{i,a}(\theta^*[j:v]) \quad \mu_2 \triangleq \max_{a \in K_i} \sup_{v \in \overline{V}} \mu_{i,a}(\theta^*[j:v]),
\]

\[
\sigma_1 \triangleq \min_{a \in K_i} \inf_{v \in \overline{V}} \sigma_{i,a}(\theta^*[j:v]) \quad \sigma_2 \triangleq \max_{a \in K_i} \sup_{v \in \overline{V}} \sigma_{i,a}(\theta^*[j:v]).
\]

We check that the assumptions of the lemma are satisfied: since \( \overline{V} \subset V_{(j, \theta^*)} \) and the functions
\[
v \in V_{(j, \theta^*)} \mapsto \mu_{i,a}(\theta^*[j:v]) \quad \text{and} \quad v \in V_{(j, \theta^*)} \mapsto \sigma_{i,a}(\theta^*[j:v])
\]
are continuous by the condition (10), \( \mu_m \) and \( \sigma_m \) (\( m = 1, 2 \)) are all finite by the extreme value theorem; moreover, \( \sigma_1 \) is positive by the choice of \( V_{(i,a,j,\theta^*)} \). Thus, by Lemma C.3, there exists \( c \in \mathbb{R} \) such that
\[
g_{i, \theta^*[j:v]}(r) = \prod_{a \in K_i} N(r(\alpha); \mu_{i,a}(\theta^*[j:v]), \sigma_{i,a}(\theta^*[j:v])) \leq \prod_{a \in K_i} c \cdot N(r(\alpha); 0, 2\sigma_2) = c' \cdot g'(r)
\]
for all \( v \in V \), where \( c' \triangleq c^{|K_i|} \) and \( g'(r) \triangleq \prod_{a \in K_i} N(r(\alpha); 0, 2\sigma_2) \).

Combining what we have proved, we obtain the following bounds: for all \( v \in V \),
\[
|\nabla_v g_{i, \theta^*[j:v]}(r)| = g_{i, \theta^*[j:v]}(r) \cdot \left| \sum_{a \in K_i} \sum_{n=0}^{2} G_{(1,a,n)}(v) \cdot r(\alpha)^n \right| 
\]
\[
\leq c' \cdot g'(r) \cdot \sum_{a \in K_i} \sum_{n=0}^{2} C_{\text{max}} \cdot |r(\alpha)^n| 
\]
\[
\leq c' \cdot g'(r) \cdot \exp(l_1(r)) \quad \text{for some } l_1 \in A_i \text{ (Lemma C.1 (a) and (b)),}
\]
\[
|\log g_{i, \theta^*[j:v]}(r)| = \left| \sum_{a \in K_i} \sum_{n=0}^{2} G_{(2,a,n)}(v) \cdot r(\alpha)^n \right| 
\]
\[
\leq \sum_{a \in K_i} \sum_{n=0}^{2} C_{\text{max}} \cdot |r(\alpha)^n| \leq \exp(l_1(r)).
\]

Second, we bound \( |\log h_i(r) \cdot k_i(r)| \) by \( \exp(l_2(r)) \) for some \( l_2 \in A_i \).
\[
|\log h_i(r) \cdot k_i(r)| \leq \sum_{a \in K_i} |\log h_{i,a}(r)| + \sum_{l=1}^{N_l} |\log k_{i,l}(r)| 
\]
\[
= \sum_{a \in K_i} \left| -\log \sqrt{2\pi} - \log \sigma'_{i,a}(r) - \frac{(r(\alpha) - \mu'_{i,a}(r))^2}{2\sigma'_{i,a}(r)^2} \right| 
\]
\[
+ \sum_{l=1}^{N_l} \left| -\log \sqrt{2\pi} - \log \sigma''_{i,l}(r) - \frac{(c_{i,l} - \mu''_{i,l}(r))^2}{2\sigma''_{i,l}(r)^2} \right|
\]
Finally, we define $H_i$ and $H_i'$ by combining the bounds that we got so far. Here are our definitions for them:

\[ H_i(r) \doteq c' \cdot g'(r) \cdot \exp(l_1(r)) \cdot (\exp(l_1(r)) + \exp(l_2(r)) + 1), \]
\[ H_i'(r) \doteq c' \cdot g'(r) \cdot \exp(l_1(r)). \]

Then, by the choice of $l_1$ and $l_2$, for all $r \in A_i$ and $v \in V$,

\[ \left| \nabla_v \left( g_{i, \theta^*[f;\emptyset]}(r) \cdot \log \frac{g_{i, \theta^*[f;\emptyset]}(r)}{h_i(r) \cdot k_i(r)} \right) \right| \leq H_i(r) \quad \text{and} \quad \left| \nabla_v g_{i, \theta^*[f;\emptyset]}(r) \right| \leq H_i'(r). \]

Also, we have

\[
\int \rho(dr) \left| \mathbb{1}_{[r \in A_i]} \cdot H_i(r) \right| \\
= \int \rho(dr) \left( \mathbb{1}_{[r \in A_i]} \cdot g'(r) \cdot c' \cdot \exp(l_1(r)) \cdot (\exp(l_1(r)) + \exp(l_2(r)) + 1) \right) \\
\leq \int \rho(dr) \left( \mathbb{1}_{[r \in A_i]} \cdot g'(r) \cdot \exp(l_3(r)) \right) \quad \text{for some } l_3 \in A_i \text{ (Lemma C.1 (a))} \\
= \mathbb{E}_{\Pi_{r \in K_i} \mathcal{N}(r(\alpha); 0, 2\sigma_2)} [\exp(l_3(r))] < \infty \quad \text{(Lemma C.1 (e)),}
\]

and

\[
\int \rho(dr) \left| \mathbb{1}_{[r \in A_i]} \cdot H_i'(r) \right| \\
= \int \rho(dr) \left( \mathbb{1}_{[r \in A_i]} \cdot g'(r) \cdot c' \cdot \exp(l_1(r)) \right) \\
\leq \int \rho(dr) \left( \mathbb{1}_{[r \in A_i]} \cdot g'(r) \cdot \exp(l_4(r)) \right) \quad \text{for some } l_4 \in A_i \text{ (Lemma C.1 (a))} \\
= \mathbb{E}_{\Pi_{r \in K_i} \mathcal{N}(r(\alpha); 0, 2\sigma_2)} [\exp(l_4(r))] < \infty \quad \text{(Lemma C.1 (e)),}
\]

We have just shown that $H_i$ and $H_i'$ are the functions that we are looking for. \( \square \)

D PROOF OF THEOREM IN §7

THEOREM 7.1 (Soundness). For all commands $C$, we have $\llbracket C \rrbracket_d \in \gamma(\llbracket C \rrbracket_\#)$.

Proof. We prove the theorem by structural induction on $C$. Every case except the one for the loop follows from the assumption made for an abstract element in $T_\#$ or an operator on it that corresponds to $C$. The proof of the case of $C \equiv \text{while } E \{ C_0 \}$ goes as follows. Let $t_{fix} \doteq (wfix T)$. Then, by the assumption on the concretisation, $\gamma(t_{fix})$ is an admissible subset of $\mathcal{D}$. Thus, it suffices to prove that when $G$ is the function in the density semantics of $\llbracket \text{while } E \{ C_0 \} \rrbracket_d$, the image of $G$
on $\gamma(t_{\text{fix}})$ is included in $\gamma(t_{\text{fix}})$. To do so, pick $g \in \gamma(t_{\text{fix}})$. We have to show that $G(g) \in \gamma(t_{\text{fix}})$. By the induction hypothesis and the assumption on the abstract composition,

$$(g^+ \circ \llbracket C_0 \rrbracket_d) \in \gamma(t_{\text{fix}} \circ \# \llbracket C_0 \rrbracket^\#).$$

Also, $\llbracket \text{skip} \rrbracket_d \in \gamma(\text{skip}^\#)$. Thus,

$$G(g) = \lambda(s, r). \text{if } (\llbracket E \rrbracket s = \text{true}) \text{ then } \llbracket \text{skip} \rrbracket_d(s, r) \text{ else } (g^+ \circ \llbracket C_0 \rrbracket_d)(s, r)$$

$$\in \gamma(\text{cond}(E)^\#(t_{\text{fix}} \circ \# \llbracket C_0 \rrbracket^\#, \text{skip}^\#)) = \gamma(T(t_{\text{fix}})) = \gamma(t_{\text{fix}}).$$

The set membership holds because of the assumption on the abstract conditional operator. \qed