Discrete Probabilistic Programming Languages VI: Sampling[1]

[1] CS7470 Fall 2023: Foundations of Probabilistic Programming.

Steven Holtzen
s.holtzen@northeastern.edu

October 10, 2023

Some logistics:

- Next week is Systems Week. Have a look at the systems week page! Start playing with some existing PPL systems.
- I am happy to meet about your project. Proposal is due Oct. 20.

1 *Sampling & approximate reasoning*

- Up until now we have been exclusively discussing *exact reasoning*: computing the exact probability that a program will output a particular value
- Problems with exact reasoning:
  - State-space explosion
  - Limited expressive power: how can we handle continuous probability, or loops that may never terminate?
  - "All-or-nothing": exact answer or nothing at all
- An alternative is *approximate reasoning*. Many of the most popular PPLs in use today support exclusively this mode of reasoning.[2]
- There is an entirely separate school of PPLs that reason by *sampling*.
- The crucial mechanism is the *sample mean*, which gives an estimate of the expectation of a random variable:

  Definition 1 (Expectation)**.** *Let* $(\Omega, \Pr)$ *be a probability space and* $f : \Omega \to \mathbb{R}$ *be a random variable. The* expectation *(or* average value*) of* f *with respect to* $\Pr$ *is defined:*

$$\mathbb{E}_{\Pr[f]} \triangleq \sum_{\omega \in \Omega} \Pr(\omega) f(\omega). \tag{1}$$

  Definition 2 (Sample mean)**.** *Let* $(\Omega, \Pr)$ *be a probability space and* $f : \Omega \to \mathbb{R}$ *be a random variable. Then, the* sample mean *of* f *with* N *samples is defined:*

$$\frac{1}{N} \sum_{\omega_i \sim \Pr}^{N} f(\omega_i), \tag{2}$$

[2] For example, Stan [Carpenter et al., 2017].

where the notation $\omega_i \sim \Pr$ *denotes drawing a sample* ω_i *from the probability distribution* Pr.

One may wonder *how quickly* a particular estimate of the mean approaches the true value (i.e., how many samples one must draw in order to have an accurate estimate with high probability). There are many bounds of this sort known broadly as *concentration inequalities*; Shalev-Shwartz and Ben-David [2014] has a nice summary of some of the useful concentration inequalities that arise in practice in the appendix.

- The reason why we use the sample estimator is that the *law of large numbers* guarantees that, as $N \to \infty$, the sample mean approaches the expectation, i.e.:

$$\lim_{N\to\infty} \frac{1}{N} \sum_{\omega_i \sim \Pr}^{N} f(\omega_i) = \mathbb{E}_{\Pr}[f]. \tag{3}$$

- What will do is give a semantics to programs in terms of expectations, and then use the expectation estimator in order to get an approximation for the program's behavior

2 *Sampling semantics for* Disc

This secton is based on the semantics given by Culpepper and Cobb [2017].

- **Goal**: Give a semantics that draws samples from $[\![\mathtt{e}]\!]$, where `e` is a (probabilistic) Disc term. Then, we can use the sample mean to approximate the semantics of a probabilistic program.
- We still want our semantics to be a *deterministic relation* on terms; how can we draw samples using a deterministic relation?
- Solution: Add a *source of randomness* to our context (just like how your computer has `/dev/rand`)
- To get our feet wet with this new style of semantics, let's consider a tiny sub-language of Disc with only the following syntax with only fair coin-flips and no observations:

```
e ::= flip 1/2 | x ← e; e | x | e ∧ e | e ∨ e | ¬ e
```

  The denotation of this sub-language is inherited from Disc.

- To draw samples from this language, we add to our evaluation judgment an *finite stream of fair coin-flips* $\sigma \in \mathbb{B}^k$ for some integer k.
- To handle binding, we will need a way to split this bit-stream up. We notate this π_L and π_R, which split σ into two disjoint random sets of bits. We can define these projections in a number of ways, but a simple way is to define π_R to take all even bits and π_L to take all odd bits:

$$\pi_R(v_1, v_2, v_3, \dots) \triangleq (v_2, v_4, v_6, \dots) \quad \pi_L(v_1, v_2, v_3, \dots) \triangleq (v_1, v_3, v_5, \dots)$$

- Then we can define a judgment $\sigma \vdash \mathtt{e} \Downarrow^S v$:

$$v :: \sigma \vdash \texttt{flip} \Downarrow^S v \qquad \frac{\pi_L(\sigma) \vdash \mathsf{e}_1 \Downarrow^S v_1 \qquad \pi_R(\sigma) \vdash \mathsf{e}_2[v_1/x] \Downarrow^S v_2}{\sigma \vdash x \leftarrow \mathsf{e}_1; \mathsf{e}_2 \Downarrow^S v_2}$$

This judgment simply gets *stuck* if there are not enough coin flips in our context.

- Since our semantics are deterministic, can define a function $ev(\sigma, \mathsf{e})$ that produces the value that the (closed) term e evaluates to for σ:
- Now we can establish an adequacy condition that relates an expectation (over finite sequences of coin tosses) to the denotation:

Theorem 1. *Let* Pr *be the distribution on the finite bit streams* $\mathbb{B}^k$*, and furthermore assume that* k *is "big enough".*[3] *Assume* $\Gamma \vdash \mathsf{e}$*. Then, for any* $\gamma \in [\![\Gamma]\!]$*,* $\mathbb{E}_{\sigma \sim \Pr}[\mathbb{1}(ev(\sigma, \mathsf{e}[\gamma]) = \mathit{true})] = [\![\mathsf{e}[\gamma]]\!]\,(\mathit{true})$.

[3] What does it mean for k to be big enough? Formally, we assume that $k > 2^n$, where n is the number of bindings in the term.

Proof. The non-probabilistic cases are straightforward. Let's see the `flip` case first. We can disregard γ since this term is closed.

$$\begin{aligned}
\mathbb{E}_{\sigma \sim \Pr}[\mathbb{1}(ev(\sigma, \texttt{flip 1/2}))] &= \sum_{(v_1, v_2, \ldots, v_k)} \Pr(v_1, v_2, \cdots, v_k) \mathbb{1}[ev((v_1, \ldots, v_k), \texttt{flip 1/2}) = \texttt{true})] \\
&= \sum_{v_2, \ldots, v_k} \Pr(v_2, \ldots, v_k) \times \sum_{v_1} \Pr(v_1) \mathbb{1}[ev(v_1, \texttt{flip 1/2}) = \texttt{true})] \\
&= \Pr(v = \texttt{true}) \mathbb{1}[\texttt{true} = \texttt{true}] + \Pr(v = \texttt{false}) \mathbb{1}[\texttt{false} = \texttt{true}] \\
&= 1/2 \\
&= [\![\texttt{flip 1/2}]\!]\,(\texttt{true}).
\end{aligned}$$

Now for the bind case, $x \leftarrow \mathsf{e}_1; \mathsf{e}_2$. Assume by induction that:

- $\mathbb{E}_{\sigma \sim \Pr}[\mathbb{1}(ev(\sigma, \mathsf{e}_1) = \texttt{true})] = [\![\mathsf{e}_1]\!]\,(\texttt{true})$.
- $\mathbb{E}_{\sigma \sim \Pr}[\mathbb{1}(ev(\sigma, \mathsf{e}_2) = \texttt{true})] = [\![\mathsf{e}_2]\!]\,(\texttt{true})$.

Proceeding:

$$\begin{aligned}
\mathbb{E}_{\sigma \sim \Pr}[\mathbb{1}(ev(x \leftarrow \mathsf{e}_1; \mathsf{e}_2) = \texttt{true})] &= \sum_{\sigma} \Pr(\sigma) \mathbb{1}(ev(x \leftarrow \mathsf{e}_1; \mathsf{e}_2) = \texttt{true}) && (4) \\
&= \sum_{\sigma} \Pr(\sigma) \sum_{v} \mathbb{1}[ev(\mathsf{e}_1, \pi_R(\sigma)) = v] \mathbf{1}[ev(\mathsf{e}_2[v/x], \pi_R(\sigma)) = \texttt{true}] && (5) \\
&= \sum_{v_1, v_3, \ldots} \Pr(v_1, v_3, \ldots)[ev(\mathsf{e}_1, v_1, v_3, \ldots) = v] \times \sum_{v_2, v_4, \ldots} \Pr(\sigma) \mathbb{1}[ev(\mathsf{e}_2[v/x], \sigma) = \texttt{true}]\, \mathrm{d}\sigma && (6) \\
&= \sum_{v} [\![\mathsf{e}_1]\!]\,(v)\, [\![\mathsf{e}_2[v/x]]\!]\,(\texttt{true}). && (7)
\end{aligned}$$

This final equality holds due to a natural Kripke-esque monotonicity property. □

- This directly gives us a procedure for sampling

Infinite streams

- **Problem**: How do we know how many coins to draw a-priori?
- One solution is to draw samples from an *infinite stream of fair coin-flips* $\sigma \in \mathbb{B}^{\mathbb{N}}$.
- Given access to this context, we can define a relation $\sigma \vdash \mathsf{e} \Downarrow^S v$ for our tiny language above (showing only the probabilistic terms):

$$v :: \sigma \vdash \texttt{flip} \Downarrow^S v \qquad \frac{\pi_L(\sigma) \vdash \mathsf{e}_1 \Downarrow^S v_1 \qquad \pi_R(\sigma) \vdash \mathsf{e}_2[v_1/x] \Downarrow^S v_2}{\sigma \vdash x \leftarrow \mathsf{e}_1; \mathsf{e}_2 \Downarrow^S v_2}$$

- Now we can run our program for different values of σ:

$$\dfrac{\texttt{true} :: \pi_L(\sigma) \vdash \texttt{flip}\ 1/2 \Downarrow^S \texttt{true} \qquad \dfrac{\texttt{false} :: \pi_L(\pi_R(\sigma)) \vdash \texttt{flip}\ 1/2 \Downarrow^S \texttt{false} \qquad \dfrac{\dfrac{\cdots}{\pi_R(\pi_R(\sigma)) \vdash \texttt{true} \vee \texttt{false} \Downarrow^S \texttt{true}}}{\pi_R(\pi_R(\sigma)) \vdash \texttt{return true} \vee \texttt{false} \Downarrow^S \texttt{true}}}{y \leftarrow \texttt{flip}\ 1/2; \texttt{return true} \vee y \Downarrow^S \texttt{true}}}{\texttt{true} :: \texttt{false} :: \sigma \vdash x \leftarrow \texttt{flip}\ 1/2; y \leftarrow \texttt{flip}\ 1/2; \texttt{return}\ x \vee y \Downarrow^S \texttt{true}}$$

3 *Adequacy*

- **Goal**: Prove the sampling relation $\sigma \vdash \mathsf{e} \Downarrow^S v$ "correctly sample" from $[\![\mathsf{e}]\!]$.
- To make this formal, we will need a way of representing a probability distribution on $\mathbb{B}^{\mathbb{N}}$. This is actually much trickier to define than it seems at first!

  Let's start by defining the probability of any *finite subsequence* of bits; we will refine this notion more later:

$$\Pr((v_1, v_2, \ldots, v_n)) = \frac{1}{2^n} \tag{8}$$

Theorem 2. *Let* Pr *be the distribution on the infinite bit streams* $\mathbb{B}^{\mathbb{N}}$. *Assume* $\Gamma \vdash \mathsf{e}$. *Then, for any* $\gamma \in [\![\Gamma]\!]$, $\mathbb{E}_{\sigma \sim \Pr}[\mathbb{1}(ev(\sigma, \mathsf{e}[\gamma]) = \mathit{true})] = [\![\mathsf{e}[\gamma]]\!]\,(\mathit{true})$.

Proof. The non-probabilistic cases are straightforward. Let's see the `flip` case first. We can disregard γ since this term is closed.

$$\begin{aligned}
\mathbb{E}_{\sigma \sim \Pr}[\mathbb{1}(ev(\sigma, \mathsf{e}))] &= \int \Pr(v :: \sigma)\mathbb{1}[ev(v :: \sigma, \mathsf{e}) = \texttt{true})]\ \mathrm{d}\sigma \\
&= \Pr(v = \texttt{true})\mathbb{1}[\texttt{true} = \texttt{true}] + \Pr(v = \texttt{false})\mathbb{1}[\texttt{false} = \texttt{true}] \\
&= 1/2 = [\![\texttt{flip 1/2}]\!]\,(\texttt{true}).
\end{aligned}$$

Now for the bind case, $x \leftarrow \mathtt{e}_1; \mathtt{e}_2$. Assume by induction that:

- $\mathbb{E}_{\sigma \sim \Pr}[\mathbb{1}(ev(\sigma, \mathtt{e}_1) = \mathtt{true})] = [\![\mathtt{e}_1]\!](\mathtt{true})$.
- $\mathbb{E}_{\sigma \sim \Pr}[\mathbb{1}(ev(\sigma, \mathtt{e}_2) = \mathtt{true})] = [\![\mathtt{e}_2]\!](\mathtt{true})$.

Proceeding:

$$\mathbb{E}_{\sigma \sim \Pr}[\mathbb{1}(ev(x \leftarrow \mathtt{e}_1; \mathtt{e}_2) = \mathtt{true})] = \int \Pr(\sigma) \mathbb{1}(ev(x \leftarrow \mathtt{e}_1; \mathtt{e}_2) = \mathtt{true}) \, \mathrm{d}\sigma \tag{9}$$

$$= \int \Pr(\sigma) \sum_v \mathbb{1}[ev(\mathtt{e}_1, \pi_R(\sigma)) = v] \mathbf{1}[ev(\mathtt{e}_2[v/x], \pi_R(\sigma)) = \mathtt{true}] \, \mathrm{d}\sigma \tag{10}$$

$$= \sum_v \int \Pr(\sigma)[ev(\mathtt{e}_1, \sigma) = v] \, \mathrm{d}\sigma \times \int \Pr(\sigma) \mathbb{1}[ev(\mathtt{e}_2[v/x], \sigma) = \mathtt{true}] \, \mathrm{d}\sigma \tag{11}$$

$$= \sum_v [\![\mathtt{e}_1]\!](v) \, [\![\mathtt{e}_2[v/x]]\!](\mathtt{true}). \tag{12}$$

□

- Now we are left with one final problem: how can we "run these semantics" on a computer? I.e., how can we effectively sample from an infinite stream of random bits, which seems to require infinite memory (and in which each sample has probability 0)? We can *sample the bits lazily*: each time a fresh random bit is required, flip a fair coin to sample it.
- Now, let's compare sampling against exact: when might one prefer sampling over exact, and vise versa?

The uniform distribution on infinite bit streams

- **Goal**: Define the probability space $(\mathbb{B}^{\mathbb{N}}, \Pr)$
- **Problem**: It is surprisingly hard to give a formal description of this space! Observe: every $\sigma \in \mathbb{B}^{\mathbb{N}}$ has a probability of 0. This seems broken.
- We will need to venture into the machinery of *measure theory* to give a formal description of this probability distribution.

Some recommended measure theory textbooks: Axler [2020], Rosenthal [2006], Pollard [2002].

- We need a new definition of probability space. The main insight is that, while individual elements of the sample space might have 0 probability, *events* may not have 0 probability. For instance, consider the event A_1 which is the collection of all infinite bit-streams that begin with the first bit being `true`:

  $$A_1 = \{\mathtt{true}, a_1, a_2, \dots\}$$

  Intuitively, this event should have probability $1/2$, $\Pr(A_1) = 1/2$, even though every individual element of the sample space inside it has probability 0.

- How do we resolve this paradox? We introduce a new *structure on events*, and define our probability *measure* on that event structure instead of directly on the sample space.

  Definition 3. *Let Ω be a sample space. A* σ-algebra $\mathcal{F}$ *on* Ω *is a collection of subsets of* Ω *that:*

  1. *Contains* Ω*;*
  2. *Is closed under complementation: if* $A \in \mathcal{F}$ *then* $\overline{A} \in \mathcal{F}$*;*
  3. *Is closed under countable union: If* $\{A_i\} \in \mathcal{F}$*, then* $\bigcup_i A_i \in \mathcal{F}$*.*

- The simplest kind of σ algebra is the *power-set σ-algebra*. Let Ω be a finite set, and 2^Ω be the set of all subsets formed out of Ω. Then, 2^Ω is a σ-algebra.

- Now we can define a generalized probability space:

  Definition 4. *Let Ω be a sample space, $\mathcal{F}$ be a σ-algebra on Ω, and $\mu : \mathcal{F} \to [0,1]$ be a map. We call μ a* probability measure *if it satisfies:*

  1. $\mu(\Omega) = 1$*;*
  2. *Countable additivity: for disjoint* $\{A_i\} \in \mathcal{F}$*, it holds that* $\mu(\bigcup_i A_i) = \sum_i \mu(A_i)$*.*

  We call the triple $(\Omega, \mathcal{F}, \mu)$ a probability space*.*

- Note: up until now, we have been implicitly working in a finite probability space where Ω is finite, $\mathcal{F}$ is the power-set σ-algebra, and μ is defined only on $\omega \in \Omega$. We can extend this old definition to a generalized probability space by countable additivity.

See Rosenthal [2006, Section 2.6] for a detailed discussion on coin-flipping measures.

- Now the key step for resolving our coin-flipping dilemma: we define a collection of events and their probability measure according to our intuition about how finite sequences of coin tosses behave:

$$\mathcal{J} = \{A_{a_1 a_2 \ldots a_n} \mid n \in \mathbb{N}, a_i \in \{0,1\}\} \cup \{\emptyset, \Omega\} \tag{13}$$

  Then, for any $A_{a_1,a_2,\ldots,a_n} \in \mathcal{J}$, we define $\mu(A_{a_1,a_2,\ldots,a_n}) = 1/2^n$, $\mu(\emptyset) = 0, \mu(\Omega) = 1$.

- Clearly $\mathcal{J}$ is not yet a σ-algebra (it is not closed under countable unions), and μ is not a fully-defined probability measure since it is not defined on a valid σ-algebra. [4] So, let's define μ to satisfy countable additivity. Let $\{A_i\}$ be disjoint elements of $\mathcal{J}$. Then, we define:

$$\mu\left(\bigcup_i A_i\right) = \sum_i \mu(A_i). \tag{14}$$

[4] The set $\mathcal{J}$ is what is known as a *semi-algebra* on Ω: it is a collection of subsets of Ω that contains $\emptyset$ and Ω, is closed under *finite* intersection, and the complement of any element is equal to a *finite* disjoint union of elements.

- Now, there is a well-known theorem in measure theory called the *extension theorem* that says that (1) there exists a sigma algebra that contains $\mathcal{J}$; and (2) there is a countably additive probability measure μ^* on $\mathcal{F}$ that agrees with μ above, i.e. for any $A \in \mathcal{J}$, we have that $\mu^*(A) = \mu(A)$.
- This resolves all of our paradoxes: we now have a probability measure that lets us (1) reason about probabilities of finite subsequences; (2) assigns zero probability to all infinite sequences

References

Sheldon Axler. *Measure, integration & real analysis*. Springer Nature, 2020.

Bob Carpenter, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus A Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of statistical software*, 76, 2017.

Ryan Culpepper and Andrew Cobb. Contextual equivalence for probabilistic programs with continuous random variables and scoring. In *European Symposium on Programming*, pages 368–392. Springer, 2017.

David Pollard. *A user's guide to measure theoretic probability*. Number 8. Cambridge University Press, 2002.

Jeffrey S Rosenthal. *First Look At Rigorous Probability Theory, A*. World Scientific Publishing Company, 2006.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.