Discrete Probabilistic Programming Languages[1]

Steven Holtzen
s.holtzen@northeastern.edu

October 2, 2023

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

1 Disc: *A simple discrete PPL*

- Syntax:

```
  e ::=
  | x ← e; e
  | observe e; e
  | flip q                         // q is a rational value
  | if e then e else e
  | return e
  | true | false
  | e ∧ e | e ∨ e | ¬ e |
  | ( e )
p ::= e
```

- Disc looks very similar to a standard functional programming language, but has two some interesting new keywords: `flip`, `observe`, and `return`
- `flip` θ allocates a new random quantity that is `true` with probability θ and `false` with probability $1 - \theta$
- `return e` turns a non-probabilistic quantity into a probabilistic one, i.e. `return true` is a *probabilistic quantity* that is `true` with probability 1 and `false` with probability 0
- Example program and its interpretation:

```
x ← flip 0.5;
y ← flip 0.5;
return x ∧ y
```

  This program outputs the probability distribution $[\texttt{true} \mapsto 0.25, \texttt{false} \mapsto 0.75]$.

- `observe` is a powerful keyword that lets us *condition* the program. For instance, suppose I want to model the following scenario: "flip two coins and observe that at least one of them is heads. What is the probability that the first coin was heads?"

  We can encode this scenario as a Disc program:

```
x ← flip 0.5;
y ← flip 0.5;
observe x ∨ y;
return x
```

This program outputs the probability distribution:

$$[\texttt{true} \mapsto (0.25 + 0.25)/0.75, \texttt{false} \mapsto 0.25/0.75]$$

- **Type system**: terms can either be pure Booleans of type $\mathbb{B}$ or distributions on Booleans of type $\texttt{Dist}(\mathbb{B})$. So, we have the following type definition:

$$\tau ::= \mathbb{B} \mid \texttt{Dist}(\mathbb{B}). \tag{1}$$

- We define a typing judgment $\Gamma \vdash \texttt{e} : \tau$ that associates each term with a type. The typing context Γ is a map from identifiers to types.

$$\Gamma \vdash \texttt{true} : \mathbb{B} \qquad \Gamma \vdash \texttt{false} : \mathbb{B} \qquad \Gamma \vdash \texttt{flip}\ \theta : \texttt{Dist}(\mathbb{B})$$

$$\frac{\Gamma \vdash \texttt{e} : \mathbb{B}}{\Gamma \vdash \texttt{return e} : \texttt{Dist}(\mathbb{B})} \qquad \frac{\Gamma \vdash \texttt{e}_1 : \mathbb{B} \qquad \Gamma \vdash \texttt{e}_2 : \tau}{\Gamma \vdash \texttt{observe e}_1\texttt{;e}_2 : \tau}$$

$$\frac{\Gamma \vdash \texttt{e}_1 : \texttt{Dist}(\mathbb{B}) \qquad \Gamma \cup [x \mapsto \mathbb{B}] \vdash \texttt{e}_2 : \tau}{\Gamma \vdash x \leftarrow \texttt{e}_1\texttt{;e}_2 : \tau}$$

$$\frac{\Gamma \vdash \texttt{e}_1 : \mathbb{B} \qquad \Gamma \vdash \texttt{e}_2 : \tau \qquad \Gamma \vdash \texttt{e}_3 : \tau}{\Gamma \vdash \texttt{if e}_1 \texttt{ then e}_2 \texttt{ else e}_3 : \tau} \qquad \frac{\Gamma \vdash \texttt{e}_1 : \mathbb{B} \qquad \Gamma \vdash \texttt{e}_2 : \mathbb{B}}{\Gamma \vdash \texttt{e}_1 \wedge \texttt{e}_2 : \mathbb{B}}$$

Denotational semantics of DISC

- Associates each term with an *unnormalized probability distribution* (i.e., the total probability mass may be less than 1).
- Has the type $[\![\texttt{e}]\!] : \texttt{Bool} \to [0,1]$, and has the following inductive definition:

$$[\![\texttt{flip}\ \theta]\!](v) = \begin{cases} \theta & \text{if } v = T \\ 1 - \theta & \text{if } v = F \end{cases}$$

$$[\![\texttt{return}\ e]\!](v) = \begin{cases} 1 & \text{if } v = [\![e]\!] \\ 0 & \text{otherwise} \end{cases}$$

$$[\![x \leftarrow e_1; e_2]\!](v) = \sum_{v'} [\![e_1]\!](v') \times [\![e_2[x \mapsto v']]\!](v)$$

$$[\![\texttt{observe}\ e_1; e_2]\!](v) = \sum_{\{v' \mid [\![e_1(v') = T\}]\!]} [\![e_2]\!](v)$$

- The semantics for non-probabilistic terms is standard. The semantic evaluation has type $[[e]]$: `Bool` for closed terms.
- These semantics give an unnormalized distribution. The main semantic object of interest is the normalized distribution, which is given by the **normalized semantics**:

$$[\![e]\!]_D(T) = \frac{[\![e]\!](T)}{[\![e]\!](T) + [\![e]\!](F)},$$

  defined analogously for the false case.

2 *Observation*

3 *Compiling* Disc *to* Prop

- **Goal**: give a semantics-preserving compilation $\rightsquigarrow$ that compiles Disc to Prop
- In order to handle observations, we will compile Disc programs into *two* Prop programs: one that computes the unnormalized probability of returning `true`, and one that computes the probability of evidence (i.e. normalizing constant)
- Inductive description has the shape $\mathsf{e} \rightsquigarrow (\mathsf{p}_1, \mathsf{p}_2)$. We want to define this relation to satisfy the following **adequacy condition**:

$$[\![\mathsf{e}]\!]_D\,(\mathtt{true}) = \frac{[\![\mathsf{p}_1]\!]}{[\![\mathsf{p}_2]\!]}. \tag{2}$$

  We will shorten this description to $\mathsf{e} \rightsquigarrow (\varphi, \varphi_A, w)$ and assume that the two formulae share a common w. The adequacy condition then becomes:

$$[\![\mathsf{e}]\!]_D\,(\mathtt{true}) = \frac{[\![(\varphi, w)]\!]}{[\![(\varphi_A, w)]\!]}. \tag{3}$$

- Compilation relation:

$$\mathtt{true} \rightsquigarrow (\mathtt{true}, \mathtt{true}, \emptyset) \qquad \mathtt{false} \rightsquigarrow (\mathtt{false}, \mathtt{true}, \emptyset)$$

$$x \rightsquigarrow (x, \mathtt{true}, \emptyset) \qquad \frac{\mathtt{fresh}\ x}{\mathtt{flip}\ \theta \rightsquigarrow (x, \mathtt{true}, [x \mapsto \theta, \overline{x} \mapsto 1-\theta])}$$

$$\frac{\mathsf{e}_1 \rightsquigarrow (\varphi, \varphi_A, w) \qquad \mathsf{e}_2 \rightsquigarrow (\varphi', \varphi'_A, w')}{x \leftarrow \mathsf{e}_1; \mathsf{e}_2 \rightsquigarrow (\varphi'[\varphi/x], \varphi'_A[\varphi/x] \wedge \varphi_A, w_1 \cup w_2)}$$

$$\frac{\mathsf{e}_1 \rightsquigarrow (\varphi, \varphi_A, w) \qquad \mathsf{e}_2 \rightsquigarrow (\varphi', \varphi'_A, w')}{\mathtt{observe}\ \mathsf{e}_1; \mathsf{e}_2 \rightsquigarrow (\varphi', \varphi'_A \wedge \varphi_A, w_1 \cup w_2)}$$

Theorem 1 (Adequacy). *For well-typed term* $\mathtt{e}$*, assume* $\mathtt{e} \rightsquigarrow (\varphi, \varphi_A, w)$*.*
Then, $[\![\mathtt{e}]\!]_D\,(\mathtt{true}) = [\![(\varphi, w)]\!] / [\![(\varphi_A, w)]\!]$.

References