Probabilistic Programming Homework 2

Submit your solutions to the TA in the homework submission box in the third floor of the E3-1 building by 2:00pm on 22 November 2017 (Wednesday). If you type up your solutions, you can email them to him (bskim90@kaist.ac.kr).

Question 1

This question is concerned with operational semantics for probabilistic programs. Throughout the question, we will use the definitions of value, evaluation context and the $\rightsquigarrow$ relation for the likelihood weighted importance sampler in the note "Implementing Inference Algorithm for Probabilistic Programs" in the course web page.

(a) In the lecture, we introduced `let` as macro that gets expanded as follows:

$$(\texttt{let}\ [x\ e]\ e') \equiv ((\texttt{fn}\ [x]\ e')\ e).$$

Suppose that we do not want to treat `let` as macro, but we want to regard it as one of the core constructs in the language. Extend the definitions of evaluation context C and the $\rightsquigarrow$ relation such that `let` expressions are evaluated directly by the $\rightsquigarrow$ relation, instead of first being macro-expanded and then being evaluated. Hint: You need to add one new case to the current definition of evaluation context C and one new rule for $\rightsquigarrow$.

(b) Consider the following program:

$$\begin{array}{l}(\texttt{let}\ [x\ (\texttt{sample}\ (\texttt{uniform-continuous}\ 0.1\ 0.9))]\\ \quad(\texttt{let}\ [y\ (\texttt{observe}\ (\texttt{flip}\ x)\ \texttt{true})]\\ \quad\quad(\texttt{if}\ (>\ x\ 0.5)\ 0\ 1)))\end{array}$$

Let us use e to denote this program. There are many ways of evaluating $(e, 1)$ to a value-weight pair using the $\rightsquigarrow$ relation extended in Part (a). For instance, the following repeated application of the the $\rightsquigarrow$ relation leads to the pair of value 1 and weight 0.2:

$$\begin{array}{l}\left(\begin{array}{l}(\texttt{let}\ [x\ (\texttt{sample}\ (\texttt{uniform-continuous}\ 0.1\ 0.9))]\\ \quad(\texttt{let}\ [y\ (\texttt{observe}\ (\texttt{flip}\ x)\ \texttt{true})]\\ \quad\quad(\texttt{if}\ (>\ x\ 0.5)\ 0\ 1)))\end{array}\ ,\ 1\right)\\ \rightsquigarrow \left(\begin{array}{l}(\texttt{let}\ [x\ 0.2]\\ \quad(\texttt{let}\ [y\ (\texttt{observe}\ (\texttt{flip}\ x)\ \texttt{true})]\\ \quad\quad(\texttt{if}\ (>\ x\ 0.5)\ 0\ 1)))\end{array}\ ,\ 1\right)\\ \rightsquigarrow \left(\begin{array}{l}(\texttt{let}\ [y\ (\texttt{observe}\ (\texttt{flip}\ 0.2)\ \texttt{true})]\\ \quad(\texttt{if}\ (>\ 0.2\ 0.5)\ 0\ 1))\end{array}\ ,\ 1\right) \rightsquigarrow \left(\begin{array}{l}(\texttt{let}\ [y\ \texttt{true}]\\ \quad(\texttt{if}\ (>\ 0.2\ 0.5)\ 0\ 1))\end{array}\ ,\ 0.2\right)\\ \rightsquigarrow ((\texttt{if}\ (>\ 0.2\ 0.5)\ 0\ 1),\ 0.2) \rightsquigarrow ((\texttt{if}\ \texttt{false}\ 0\ 1),\ 0.2) \rightsquigarrow (1,\ 0.2)\end{array}$$

(Here I used an unexplained rule of $\rightsquigarrow$ for `let`) Using your rule for `let`, give two other possible execution sequences of the program e. Make it sure that the final values of your execution sequences are 0 and 1, respectively.

Question 2

The lightweight Metropolis-Hastings algorithm uses the following acceptance ratio:

$$\alpha((v_{n-1}, w_{n-1}, S_{n-1}), (v', w', S')) = \min\left(1, \frac{w' \cdot |S_{n-1}|}{w_{n-1} \cdot |S'|}\right).$$

Here $(v_{n-1}, w_{n-1}, S_{n-1})$ is the current state consisting of value v_{n-1}, weight w_{n-1}, and sequence of random choices S_{n-1}, and (v', w', S') is a newly proposed state. With the probability $\alpha((v_{n-1}, w_{n-1}, S_{n-1}), (v', w', S'))$, we accept this new state and set the next state (v_n, w_n, S_n) to be (v', w', S'). For detail, look at the note "Implementing Inference Algorithm for Probabilistic Programs" in the course web page.

Suppose that instead of the correct acceptance ratio above, we used the following variant:

$$\alpha'((v_{n-1}, w_{n-1}, S_{n-1}), (v', w', S')) = \min\left(1, \frac{w'}{w_{n-1}}\right).$$

Explain why this variant computes a wrong estimate for the following program:

```
(if (sample (flip 0.5))
  (or true (sample (flip 0.5)))
  false)
```

This (slightly silly) program does not contain any observe. Thus, its prior and posterior distributions coincide. In fact, the posterior distribution on boolean values assigns 0.5 to `true` and 0.5 to `false`. Hint: One way to answer this question is to focus on what the loop body of the variant of the lightweight MH algorithm does on v_n and to show that the target posterior distribution is not an invariant distribution (or stationary distribution) of the loop body.

Question 3

In the lectures, we learnt about the denotational semantics of a simple first-order programming language with discrete random choices. One big benefit of having such a semantics is that we can formally prove the equalities of programs; such equalities can form a theoretical basis for compiler optimisation. Proving the equation:

$$[\![\Gamma \vdash (\texttt{if false } e_1\ e_2) : t]\!] = [\![\Gamma \vdash e_2 : t]\!]$$

The proof is very similar to what was written in the lecture note. An optional question is to prove the following equation:

$$[\![\Gamma \vdash (\texttt{if (sample (flip 0.7)) false true}) : \texttt{bool}]\!] = [\![\Gamma \vdash (\texttt{sample (flip 0.3)}) : \texttt{bool}]\!]$$

The proof of this equation is also similar to what you find in the lecture note. Checking how you need to modify the proof in the lecture will help you to understand what goes on there. Of course, proving this second equation is optional, and you don't have to do it.