Probabilistic Functional Programming

Donnacha Oisín Kidney

July 8, 2018

Modeling Probability

How do we model stochastic and probabilistic processes in programming languages?

The Boy-Girl Paradox

1. Mr. Jones has two children. The older child is a girl. What is the probability that both children are girls?
2. Mr. Smith has two children. At least one of them is a boy. What is the probability that both children are boys?

The Boy-Girl Paradox

1. Mr. Jones has two children. The older child is a girl. What is the probability that both children are girls?
2. Mr. Smith has two children. At least one of them is a boy. What is the probability that both children are boys?

Is the answer to 2 $\frac{1}{3}$ or $\frac{1}{2}$?

The Boy-Girl Paradox

1. Mr. Jones has two children. The older child is a girl. What is the probability that both children are girls?
2. Mr. Smith has two children. At least one of them is a boy. What is the probability that both children are boys?

Is the answer to 2 $\frac{1}{3}$ or $\frac{1}{2}$?

Part of the difficulty in the question is that it's ambiguous: can we use programming languages to lend some precision?

An Ad-Hoc Solution i

Using normal features built in to the language.

```
from random import randrange, choice

class Child:
    def __init__(self):
        self.gender = choice(("boy", "girl"))
        self.age = randrange(18)
```

An Ad-Hoc Solution ii

```
from operator import attrgetter

def mr_jones():
    child_1 = Child()
    child_2 = Child()
    eldest = max(child_1, child_2,
                 key=attrgetter('age'))
    assert eldest.gender == 'girl'
    return [child_1, child_2]
```

An Ad-Hoc Solution iii

```python
def mr_smith():
    child_1 = Child()
    child_2 = Child()
    assert child_1.gender == 'boy' or \
           child_2.gender == 'boy'
    return [child_1, child_2]
```

Unclear semantics

What contracts are guaranteed by probabilistic functions?

What does it mean *exactly* for a function to be probabilistic?

Why isn't the following[1] "random"?

```
int getRandomNumber()
{
  return 4; // chosen by fair dice roll.
            // guaranteed to be random.
}
```

[1]Randall Munroe. *Xkcd: Random Number*. en. Title text: RFC 1149.5 specifies 4 as the standard IEEE-vetted random number. Feb. 2007. URL: https://xkcd.com/221/ (visited on 07/06/2018).

What about this?

```
children_1 = [Child(), Child()]
children_2 = [Child()] * 2
```

How can we describe the difference between `children_1` and `children_2`?

Underpowered

There are many more things we may want to do with probability distributions.

What about expectations?

```python
def expect(predicate, process, iterations=100):
    success, tot = 0, 0
    for _ in range(iterations):
        try:
            success += predicate(process())
            tot += 1
        except AssertionError:
            pass
    return success / tot
```

The Ad-Hoc Solution

```
expect(lambda children: all(child.gender == 'girl'
                            for child in children),
       mr_jones)
expect(lambda children: all(child.gender == 'boy'
                            for child in children),
       mr_smith)
```

Monadic Modeling

A DSL

What we're approaching is a DSL, albeit an unspecified one.

Three questions for this DSL:

- Why should we implement it? What is it useful for?
- How should we implement it? How can it be made efficient?
- Can we glean any insights on the nature of probabilistic computations from the language? Are there any interesting symmetries?

The Erwig And Kollmansberger Approach

First approach[2]:

$$\textbf{newtype}\ \mathit{Dist}\ a = \mathit{Dist}\ \{\mathit{runDist} :: [(a, \mathit{Rational})]\}$$

A distribution is a list of possible events, each tagged with a probability.

[2] Martin Erwig and Steve Kollmansberger. “Functional Pearls: Probabilistic Functional Programming in Haskell”. In: *Journal of Functional Programming* 16.1 (2006), pp. 21–34. ISSN: 1469-7653, 0956-7968. DOI: `10.1017/S0956796805005721`. URL: `http://web.engr.oregonstate.edu/~erwig/papers/abstracts.html%5C#JFP06a` (visited on 09/29/2016).

A random integer, then, is:

$$\textbf{type}\ \mathit{RandInt} = \mathit{Dist\ Int}$$

This lets us encode (in the types) the difference between:

$$\mathit{children\_1} :: [\mathit{Dist\ Child}]$$
$$\mathit{children\_2} :: \mathit{Dist}\ [\mathit{Child}]$$

As we will use this as a DSL, we need to define the language features we used above:

```
def mr_smith():
    child_1 = Child()
    child_2 = Child()
    assert child_1.gender == 'boy' or \
           child_2.gender == 'boy'
    return [child_1, child_2]
```

1. `=` (assignment)
2. `assert`
3. `return`

Assignment

This is encapsulated by the "monadic bind":

$$(\gg\!=) :: \mathit{Dist}\ a \to (a \to \mathit{Dist}\ b) \to \mathit{Dist}\ b$$

When we assign to a variable in a probabilistic computation, everything that comes later is conditional on the result of that assignment. We are therefore looking for the probability of the continuation given the left-hand-side; this is encapsulated by multiplication:

$$\begin{aligned} xs \gg\!= f = \mathit{Dist}\ & [\,(y, xp \times yp) \\ & \mid (x, xp) \leftarrow \mathit{runDist}\ xs \\ & ,\ (y, yp) \leftarrow \mathit{runDist}\ (f\ x)] \end{aligned}$$

Assertion

Assertion is a kind of conditioning: given a statement about an event, it either occurs or it doesn't.

$$
\begin{aligned}
&\mathit{guard} :: \mathit{Bool} \rightarrow \mathit{Dist}\ ()\\
&\mathit{guard}\ \mathit{True}\ \ = \mathit{Dist}\ [((), 1)]\\
&\mathit{guard}\ \mathit{False} = \mathit{Dist}\ [\,]
\end{aligned}
$$

Return

Return is the “unit” value for a distribution; the certain event, the unconditional distribution.

$$\mathit{return} :: a \to \mathit{Dist}\ a$$
$$\mathit{return}\ x = \mathit{Dist}\ [(x, 1)]$$

Putting it all Together

$$
\begin{aligned}
&\mathit{mrSmith} :: \mathit{Dist}\ [\mathit{Child}]\\
&\mathit{mrSmith} = \mathbf{do}\\
&\quad \mathit{child1} \leftarrow \mathit{child}\\
&\quad \mathit{child2} \leftarrow \mathit{child}\\
&\quad \mathit{guard}\ (\mathit{gender}\ \mathit{child1} \equiv \mathit{Boy} \vee \mathit{gender}\ \mathit{child2} \equiv \mathit{Boy})\\
&\quad \mathit{return}\ [\mathit{child1}, \mathit{child2}]
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{expect} :: (a \rightarrow \mathit{Rational}) \rightarrow \mathit{Dist}\ a \rightarrow \mathit{Rational}\\
&\mathit{expect}\ p\ \mathit{xs} = \frac{\mathit{sum}\ [p\ x \times \mathit{xp} \mid (x, \mathit{xp}) \leftarrow \mathit{runDist}\ \mathit{xs}]}{\mathit{sum}\ [\mathit{xp} \mid (\_, \mathit{xp}) \leftarrow \mathit{runDist}\ \mathit{xs}]}
\end{aligned}
$$

Theoretical Basis