A Denotational Semantics of a Probabilistic Stream-Processing Language

Yohei Miyamoto (Kyoto Univ.)
Kohei Suenaga (Kyoto Univ., JST PRESTO)
Koji Nakazawa (Nagoya Univ.)

Summary

- Denotational semantics of a stream-processing language with probabilistic behavior.
- Key idea: Extension of the Kahn-style semantics relying on the following theorem:

Theorem (Saheb-Djaromi [1]): If $(D, \leqq)$ is a cpo, then $(Prob(D), \leqq)$ is also a cpo, where Prob(D) is the set of *probabilistic distributions* over D and $\mu_1 \leqq \mu_2$ if $\mu_1(O) \leqq \mu_2(O)$ for all the Scott open set $O \subseteq D$.

Language

e (arithmetic streams) ::=
  x | c | e_1 aop e_2 | if b then e_1 else e_2
  | e_1 fby e_2
  | $\pi_k(f(e_1,\dots,e_n))$
b (Boolean streams) ::=
  true | false | $b_1 \wedge b_2$ | ¬b | e_1 rop e_2 | B^p
d (Node definitions) ::=
  node $f(x_1,\dots,x_m)$ return $(y_1,\dots,y_n)$
  with $(y_1,\dots,y_n,z_1,\dots,z_l) = (e_1,\dots,e_n,e_1',\dots,e_l)$

Head taken from e_1, tail from e_2

Call f with $e_1,\dots,e_n$ and takes the k-th of the returned streams

Stream where true occurs with probability p and false with 1-p

Definition by recursion on $y_1, \dots, y_n, z_1, \dots, z_l$

Examples

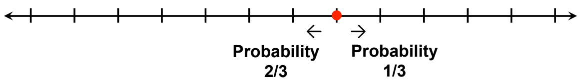
```
(* Takes (r_i)_{i∈Nat} and returns (r_1+…+r_j)_{j∈Nat}
   E.g., Sum(1 1 1 1 1 …) = 1 2 3 4 5 …*)
node Sum(x) returns y with
  y = 0 fby (x + y)
```

y[0] = 0
y[n+1] = (x + y)[n] = x[n] + y[n]

```
(* Random walk *)
node SumP() returns y with
  s = if B^{1/3} then 1 else -1
  y = Sum(s)
```

Chooses +1 with probability 1/3 and -1 with 2/3

Set y to $(s_1+\dots+s_j)_{j\in Nat}$




```
(* Nonterminating node *)
node Loop() returns x with x = x
(* Producing at least n elements with
   probability 1/2^n*)
node Geom() returns y with
  y = 1 fby (if B^{1/2} then Geom(n) else Loop())
```

Denotational Semantics

[[e]], [[b]]: Probability distributions over streams giving denotation of e and b

Remark: A probability distribution can be determined by fixing its value on open the sets of the form $s^\uparrow$ where $s^\uparrow := \{\omega \in Streams \mid s \text{ is a prefix of } \omega\}$.

The probability that x generates a stream with prefix s

The joint distribution of $[[e_1]]$ and $[[e_2]]$

$$[\![x]\!]_{\phi,\delta}(s^\uparrow) := \delta(x)(s^\uparrow) \qquad [\![c]\!]_{\phi,\delta}(s^\uparrow) := \begin{cases} 1 & (s = (c,c,\dots)) \\ 0 & (\text{otherwise}) \end{cases}$$

$$[\![e_1\,\mathbf{aop}\,e_2]\!]_{\phi,\delta}(s^\uparrow) := \sum_{\substack{s_1,s_2 \in D^* \\ s_1\,\mathbf{aop}\,s_2 = s}} P_{[\![e_1]\!]_{\phi,\delta},[\![e_2]\!]_{\phi,\delta}}(s_1^\uparrow, s_2^\uparrow)$$

$$[\![e_1\,\mathbf{fby}\,e_2]\!]_{\phi,\delta}(s^\uparrow) := \sum_{\substack{s_1,s_2 \in D^* \\ s_1\,\mathbf{fby}\,s_2 = s}} P_{[\![e_1]\!]_{\phi,\delta},[\![e_2]\!]_{\phi,\delta}}(s_1^\uparrow, s_2^\uparrow)$$

$$[\![\mathbf{if}\,b\,\mathbf{then}\,e_1\,\mathbf{else}\,e_2]\!](s^\uparrow) := \sum_{\substack{s' \in \mathbb{B}^* \quad s_1,s_2 \in D^* \\ s = \mathbf{if}\,s'\,\mathbf{then}\,s_1\,\mathbf{else}\,s_2}} P_{[\![b]\!]_{\phi,\delta},[\![e_1]\!]_{\phi,\delta},[\![e_2]\!]_{\phi,\delta}}(s'^\uparrow, s_1^\uparrow, s_2^\uparrow)$$

$$[\![\pi_k(f(e_1,\dots,e_m))]\!]_{\phi,\delta}(s^\uparrow) := \sum_{\substack{s_1,\dots,s_m \in D^* \\ s = \pi_k(\phi(f)(s_1,\dots,s_m))}} P_{[\![e_1]\!]_{\phi,\delta},\dots,[\![e_m]\!]_{\phi,\delta}}(s_1^\uparrow,\dots,s_m^\uparrow)$$

$$[\![\mathbf{B}^p]\!]_{\phi,\delta}(s^\uparrow) := 2^{-|s|} \quad (s \in \mathbb{B}^*)$$

$$[\![e_1\,\mathbf{rop}\,e_2]\!]_{\phi,\delta}(s^\uparrow)_\phi := \sum_{\substack{s_1,s_2 \in D^* \\ s = s_1\,\mathbf{rop}\,s_2}} P_{[\![e_1]\!]_{\phi,\delta},[\![e_2]\!]_{\phi,\delta}}(s_1^\uparrow, s_2^\uparrow)$$

Figure 1. Denotation of stream expressions.

Bernoulli distribution over {tt, ff} where tt is drawn with prob. p

The denotation of a recursive function as lfp; Saheb-Djaromi's theorem is for existence of the lfp

$$\left[\!\!\left[\begin{array}{l} \mathbf{node}\,f(x_1,\dots,x_m)\,\mathbf{returns}(y_1,\dots,y_n) \\ \mathbf{with}(\vec{y},\vec{z}) = (\vec{e},\vec{e'}) \end{array} \right]\!\!\right]_\phi :=$$

$$(p_1,\dots,p_m) \mapsto (\mu F_{\vec{p}}(y_1),\dots,\mu F_{\vec{p}}(y_n))$$

where $(p_1,\dots,p_m) \in \mathcal{D}(D^\omega \times \cdots \times D^\omega)$
$F_{\vec{p}}(\delta) := \{\vec{x} \mapsto \vec{p}, \vec{y} \mapsto [\![\vec{e}]\!]_{\phi,\delta}, \vec{z} \mapsto [\![\vec{e'}]\!]_{\phi,\delta}\}$
$\mu F_{\vec{p}}$ is the least fixed point of $F_{\vec{p}}$.

$$[\![[d_1,\dots,d_n;e]]\!] := [\![e]\!]_{\mu J,\emptyset}$$

where $J(\phi) := \{f_1 \mapsto [\![d_1]\!]_\phi,\dots,f_n \mapsto [\![d_n]\!]_\phi\}$
f_i is the node name defined by d_i

Figure 2. Denotation of node definitions and programs.

Future direction

- Simplification of the denotation of aop, fby, if-then-else, and function calls by analyzing dependencies among expressions
- Extension to continuous streams
- Small-step semantics?