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-Djahromi [1]): If $D$ is a cpo, then $(\text{Prob}(D), \preceq)$ is also a cpo, where $\text{Prob}(D)$ is the set of probabilistic distributions over $D$ and $\mu_1 \preceq \mu_2$ if $\mu_1(O) \preceq \mu_2(O)$ for all the Scott open set $O \subseteq D$.

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 $s'$ where $s' := (\omega \text{Streams} \mid s \text{ is a prefix of } \omega)$.

Definition by recursion on $Y_V \cdots Y_N: z_i \rightarrow z_i$

Language

- **arithmetic streams**: $e := x \mid c \mid e \circ aop e \mid e \circ fby e_1 \mid \pi_k(e_1, \ldots, e_n)$
- **Boolean streams**: $d := \text{true} \mid \text{false} \mid b_1 \land b_2 \mid \neg b \mid e_1 \circ \text{rop } e_2 \mid B^n$
- **Node definitions**: $f := \text{node } f(x_1, \ldots, x_m) \text{ return } y_1, \ldots, y_n$ with $(y_1, \ldots, y_n, z_1, \ldots, z_l) = (e_1, \ldots, e_n, e_1', \ldots, e_l')$

Examples

(* Takes $(r_i)_{i \in \mathbb{N}}$ and returns $(r_1 + \cdots + r_i)_{i \in \mathbb{N}}$ E.g., Sum$(111111 \ldots) = 12345 \ldots$*)

node Sum$(x)$ returns $y$ with $y = 0$ if $x = 1$ else $y = y + (x + y)$

(* Random walk *)

node Sum$(P)$ returns $y$ with $y = 0$ if $B^{1/2}$ then 1 else -1 and -1 with probability $1/3$ and 2 with $2/3$

(* Nonterminating node *)

node Loop$(c)$ returns $x$ with $x = x$

(* Producing at least $n$ elements with probability $1/2^n$*)

node Geom$(n)$ returns $y$ with $y = 1$ if $B^{1/2}$ then $\text{Geom}(n)$ else Loop$(c)$

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?