A Denotational Semantics of a Probabilistic Stream-Processing Language

Yohei Miyamoto[†1] Kohei Suenaga[†1,†2] Koji Nakazawa[†3]

†1: Kyoto University
†2: PRESTO, JST
†3: Nagoya University
{ymiyamoto,ksuenaga,knak}@fos.kuis.kyoto-u.ac.jp

1. Introduction

Stream-processing programming languages are used to specify computation on streams (i.e., infinite sequences of values). Since a stream can naturally express time-series data, these languages are widely used in industry to model the behavior of dynamical systems.

The formal semantics of such languages is not only a theoretically interesting topic; it is practically important as a foundation of the simulation of and reasoning about a model. To this end, several formal semantics of stream-processing languages have been proposed [1–3].

This paper describes a denotational semantics of a stream-processing language extended with *probabilistic behavior*. Although probabilistic behavior is useful in modeling uncertainty consisting in a model, to the best of our knowledge, the formal semantics of a probabilistic stream-processing language has been less investigated .

Our semantics extends one of our previous denotational semantics of a stream-processing language [5]. In the previous semantics, the denotation of streams was given by the domain of value streams. We now adapt it by using the set of *probability distributions over streams* as the domain. In our theoretical development, we rely on a theorem introduced by Saheb-Djahromi [4] stating that the set of probability distributions on a cpo constitutes a cpo with respect to a natural partial order.

As a first step toward a semantics of a full-fledged probabilistic stream-processing language, we consider a simple language, the probabilistic behavior of which is determined only by the term $\mathbf{B}^p$. This term produces a stream consisting of $\mathtt{tt}$ and $\mathtt{ff}$, where $\mathtt{tt}$ is drawn following the discrete Bernoulli distribution. It is found that we can give the semantics of this simple language as a straightforward extension of the original language. We leave the investigation of the semantics of more complex features as future work.

We outline our semantics without concrete definitions in this extended abstract. We plan to publish the concrete details in a forthcoming paper.

Preliminary Our semantics is parametrized over a domain of values $(D, \sqsubseteq)$. We assume D is countable[1]. We write $(D_\perp, \sqsubseteq)$ for the pointed domain with bottom $\perp$. We write $\mathbb{B}$ for the set of Boolean values $\{\mathtt{tt}, \mathtt{ff}\}$.

We write D^ω for the following set:

$$\{d \in D_\perp^{\mathbb{N}} \mid d(n) = \perp \land m \geq n \text{ implies } d(m) = \perp\}.$$

D^ω is the set of infinite sequences (i.e., streams), the members of which are elements of $D_\perp$ and all the members appearing after $\perp$ are $\perp$. The set $D_\perp^\omega$ is a cpo with respect to the partial order of the pointwise comparison; we abuse the symbol $\sqsubseteq$ for this order.

In order to define Borel sets on $D_\perp^\omega$, we define a topology on $D_\perp^\omega$. We write D^* for the set of finite sequences of the elements of D. For $s \in D^*$, we write $s^\uparrow$ for the set of streams $\{d \in D_\perp^\omega \mid s \text{ is a prefix of } d\}$. We write $\mathcal{O}$ for the family of sets $\{s^\uparrow \mid s \in D^*\}$; this family defines a topology on $D_\perp^\omega$. The measurable subset of $D_\perp^\omega$, written Ω, is the σ-algebra generated by $\mathcal{O}$. We write $\mathcal{D}(D_\perp^\omega)$ for the set of the probability distributions over Ω. Let $\mu_1, \mu_2 \in \mathcal{D}(D_\perp^\omega)$; we write $\mu_1 \sqsubseteq \mu_2$ if $\mu_1(S) \leq \mu_2(S)$ for every $S \in \mathcal{O}$. The set $\mathcal{D}(D_\perp^\omega)$ is a cpo with respect to the order $\sqsubseteq$ [4, Theorem 4]. Given two distributions μ_1 and μ_2, we write P_{μ_1,μ_2} for their joint distribution.

2. Language

We designate the set of *stream variables* $\mathbf{Var}$ ranged over by $x, y, z, \ldots$; a stream variable is bound to a stream of values. We also designate the set of *node names* $\mathbf{NdName}_{m,n} (m, n \in \mathbb{N})$ ranged over by $f, g, h, \ldots$; a node name represents a function from an m-tuple of streams to an n-tuple of streams.

DEFINITION 1. *The syntax of language* SPROC^P *is defined by the following BNF:*

$$\begin{array}{rcl}
e & ::= & x \mid c \mid e_1 \,\mathbf{aop}\, e_2 \mid e_1 \,\mathbf{fby}\, e_2 \\
 & \mid & \mathbf{if}\, b \,\mathbf{then}\, e_1 \,\mathbf{else}\, e_2 \\
 & \mid & \pi_k(f(e_1, \ldots, e_m)) \\
 & & \quad \textit{where } f \in \mathbf{NdName}_{m,n} \textit{ and } k \in \{1, \ldots, n\} \\
b & ::= & \mathbf{true} \mid \mathbf{false} \mid b_1 \land b_2 \mid \neg b \mid e_1 \,\mathbf{rop}\, e_2 \\
 & \mid & \mathbf{B}^p \quad \textit{where } p \in [0, 1] \\
d & ::= & \mathbf{node}\, f(x_1, \ldots, x_m) \,\mathbf{returns}(y_1, \ldots, y_n) \\
 & \mid & \mathbf{with}(y_1, \ldots, y_n, z_1, \ldots, z_l) = (e_1, \ldots, e_n, e'_1, \ldots, e'_l) \\
 & & \quad \textit{where } f \in \mathbf{NdName}_{m,n}. \\
pg & ::= & [d_1, \ldots, d_n; e]
\end{array}$$

[1] For uncountable D, the probability distribution over D does not form a cpo in general; see Saheb-Djahromi [4].

$$\llbracket x \rrbracket_{\phi,\delta}(s^{\uparrow}) := \delta(x)(s^{\uparrow}) \quad \llbracket c \rrbracket_{\phi,\delta}(s^{\uparrow}) := \begin{cases} 1 & (s = (c, c, \dots)) \\ 0 & \text{(otherwise)} \end{cases}$$

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

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

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

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

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

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

Figure 1. Denotation of stream expressions.

SPROC^P is an extension of the SPROC given by Suenaga and Hasuo [5] with expression $\mathbf{B}^p$; for a detailed explanation, see their paper.

The metavariable e ranges over *arithmetic stream expressions*, the denotation of which are given by $D^{\omega}_{\perp}$. The symbol c represents constant streams. The symbol $\mathbf{aop}$ ranges over pointwise arithmetic operations between two streams. The expression $e_1 \,\mathbf{fby}\, e_2$ evaluates to a stream, the head of which is e_1 and the tail is e_2; this expression is used for delaying a stream. The expression $\pi_k(f(e_1, \dots, e_m))$ takes the k-th stream from the n-tuple returned by $f(e_1, \dots, e_m)$.

The metavariable b is for *Boolean stream expressions* that evaluate to streams of Boolean values. The symbol $\mathbf{rop}$ is for operations between arithmetic values.

The metavariable d is for *node definitions*; an (m, n)-*node* (or simply *node*) is a function from an m-tuple of arithmetic streams to an n-tuple of arithmetic streams. The body $(y_1, \dots, y_n, z_1, \dots, z_l) = (e_1, \dots, e_n, e'_1, \dots, e'_l)$ of a node definition is mutually recursive definitions of stream variables.

EXAMPLE 1. *The following node Sum takes a stream of rational numbers $(r_i)_{i \in \mathbb{N}}$ and returns $(\sum_{j=0}^{i-1} r_j)_{i \in \mathbb{N}}$:*

$$\mathbf{node}\, \mathit{Sum}(x) \,\mathbf{returns}\, y \,\mathbf{with}\, y = 0 \,\mathbf{fby}(x + y).$$

3. Semantics

We abuse our notation of the language constructors as operations on finite sequences. For example, the following equations hold: $(1, 2, 3) + (4, 5, 6) = (5, 7, 9)$; $(1, 2, 3) \,\mathbf{fby}(4, 5) = (1, 4, 5)$; $\mathbf{if}(\mathtt{tt}, \mathtt{ff}) \,\mathbf{then}(1, 2) \,\mathbf{else}(4, 5) = (1, 5)$; and $(1, 2, 3) < (1, 4, 5) = (\mathtt{ff}, \mathtt{tt}, \mathtt{tt})$. We write $|s|$ for the length of the finite sequence s.

Figure 1 defines the denotation $\llbracket e \rrbracket_{\phi,\delta}$ for arithmetic stream expression e, $\llbracket b \rrbracket_{\phi,\delta}$ for Boolean stream expression b, and $\llbracket d \rrbracket_{\phi}$ for node definition d. Here, ϕ is a *node environment*, a map from node names to their denotation; δ is a *stream environment*, a map from variables to their denotation. The denotation $\llbracket e \rrbracket_{\phi,\delta}$ (resp. $\llbracket b \rrbracket_{\phi,\delta}$) is given by a probability distribution on $D^{\omega}_{\perp}$ (resp. $\mathbb{B}^{\omega}_{\perp}$).

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

$$\begin{array}{l} (p_1, \dots, p_m) \mapsto (\mu F_{\vec{p}}(y_1), \dots, \mu F_{\vec{p}}(y_n)) \\ \text{where } \ (p_1, \dots, p_m) \in \mathcal{D}(D^{\omega} \times \cdots \times D^{\omega}) \\ \qquad F_{\vec{p}}(\delta) := \{\vec{x} \mapsto \vec{p}, \vec{y} \mapsto \llbracket \vec{e} \rrbracket_{\phi,\delta}, \vec{z} \mapsto \llbracket \vec{e'} \rrbracket_{\phi,\delta}\} \\ \qquad \mu F_{\vec{p}} \text{ is the least fixed point of } F_{\vec{p}}. \end{array}$$

$$\begin{array}{l} \llbracket [d_1, \dots, d_n; e] \rrbracket := \llbracket e \rrbracket_{\mu J, \emptyset} \\ \text{where } \ J(\phi) := \{f_1 \mapsto \llbracket d_1 \rrbracket_{\phi}, \dots, f_n \mapsto \llbracket d_n \rrbracket_{\phi}\} \\ \qquad f_i \text{ is the node name defined by } d_i \end{array}$$

Figure 2. Denotation of node definitions and programs.

The denotation $\llbracket d \rrbracket_{\phi}$, where d defines an (m, n)-node, is given by a function from $(D^{\omega}_{\perp})^m$ to $(D^{\omega}_{\perp})^n$.

In order to define a probability distribution on Borel-measurable sets, it suffices to give the probability on each open set. Since $\mathcal{O}$ consists of the set of streams of the form $s^{\uparrow}$, where s is a finite sequence, Figure 1 defines $\llbracket e \rrbracket_{\phi,\delta}$ by giving the probability of $\llbracket e \rrbracket_{\phi,\delta}$ at $s^{\uparrow}$; this probability can be interpreted as "the probability of s appearing as a prefix if e is evaluated under node environment ϕ and stream environment δ."

The definition of $\llbracket e_1 \,\mathbf{aop}\, e_2 \rrbracket_{\phi,\delta}(s^{\uparrow})$ can be understood as follows. Given a finite sequence s, the probability of s appearing as a prefix of $e_1 \,\mathbf{aop}\, e_2$ is the sum of the probabilities of the events that satisfy the following two conditions: (1) $s = s_1 + s_2$ for some s_1 and s_2, and (2) s_1 appears as a prefix of e_1 and s_2 appears as a prefix of e_2. Since the distribution $\llbracket e_1 \rrbracket_{\phi,\delta}$ is in general not independent of $\llbracket e_2 \rrbracket_{\phi,\delta}$, the probability of (2) above is expressed using the joint probability $P_{\llbracket e_1 \rrbracket_{\phi,\delta}, \llbracket e_2 \rrbracket_{\phi,\delta}}$.

Figure 2 defines the denotation of node definitions and programs. It is the same as that in SPROC [5], except that the domain is set to probabilistic distributions. Since a node definition may contain mutual recursion among stream variables, it prepares a generating function $F_{\vec{p}}$ and uses its least fixed point as the semantics of the output of the node. The existence of the least fixed point is guaranteed by the theorem of Saheb-Djahromi [4] mentioned in Section 1.

EXAMPLE 2. *Consider the following node definition:*

$$\begin{array}{rcl} \multicolumn{3}{l}{\mathbf{node}\, \mathit{SumP}() \,\mathbf{returns}\, y \,\mathbf{with}} \\ s & = & \mathbf{if}\, \mathbf{B}^{\frac{1}{3}} \,\mathbf{then}\, 1 \,\mathbf{else}\, {-1} \\ y & = & \mathit{Sum}(s). \end{array}$$

We show that the probability of $\mathit{SumP}()$ generating a prefix 01, which is given by $\mathit{SumP}()(01^{\uparrow})$, is equal to $\frac{1}{3}$. By definition, this is equal to $\mathit{Sum}(s)(01^{\uparrow})$; in order to calculate this value, one needs to evaluate $\llbracket 0 \,\mathbf{fby}(x + y) \rrbracket_{\phi,\delta}(01^{\uparrow})$, which is equal to $\llbracket x + y \rrbracket_{\phi,\delta}(1^{\uparrow})$, where $\delta(x) = s$ and $\delta(y) = \llbracket 0 \,\mathbf{fby}(x + y) \rrbracket_{\phi,\delta}$. This is equal to $\delta(x)(1^{\uparrow})$, since $\delta(y)(0^{\uparrow}) = \llbracket 0 \,\mathbf{fby}(x + y) \rrbracket_{\phi,\delta}(0^{\uparrow}) = 1$ by definition; hence, $\mathit{SumP}()(01^{\uparrow})$ is equal to $\delta(x)(1^{\uparrow}) = \left\llbracket \mathbf{if}\, \mathbf{B}^{\frac{1}{3}} \,\mathbf{then}\, 1 \,\mathbf{else}\, {-1} \right\rrbracket_{\phi,\delta}(1^{\uparrow}) = \frac{1}{3}$.

EXAMPLE 3. *Consider the following definition of the node Geom:*

$$\begin{array}{l} \mathbf{node}\, \mathit{Geom}(n) \,\mathbf{returns}\, y \,\mathbf{with} \\ \quad y \;=\; n \,\mathbf{fby}(\mathbf{if}\, \mathbf{B}^{\frac{1}{2}} \,\mathbf{then}\, \mathit{Geom}(n) \,\mathbf{else}\, \perp), \end{array}$$

where $\perp$ above is the shorthand for an expression having the denotation $\perp$. We show $\llbracket \mathit{Geom}(n) \rrbracket ((p_1 \dots p_m)^{\uparrow}) = \frac{1}{2^m}$ if $n = p_1 = \cdots = p_m$ by induction on m. The base case $\llbracket \mathit{Geom}(n) \rrbracket (\epsilon^{\uparrow}) = 1$

because $\epsilon^{\uparrow} = D^{\omega}$ by definition. The induction case is

$$
\begin{aligned}
& [\![Geom(n)]\!]\,(p\vec{p}) \\
= \;& \left[\!\!\left[n\,\mathbf{fby}(\mathbf{if}\,\mathbf{B}^{\frac{1}{2}}\,\mathbf{then}\,Geom(n)\,\mathbf{else}\,\bot) \right]\!\!\right](p\vec{p}) \\
& (\because p = n \text{ and } [\![n]\!] \text{ is independent of the other distributions}) \\
= \;& \left[\!\!\left[\mathbf{if}\,\mathbf{B}^{\frac{1}{2}}\,\mathbf{then}\,Geom(n)\,\mathbf{else}\,\bot \right]\!\!\right](\vec{p}) \\
& (\because \left[\!\!\left[\mathbf{B}^{\frac{1}{2}}\right]\!\!\right] \text{ and } [\![\bot]\!] \text{ are independent of the other distributions}) \\
= \;& \tfrac{1}{2}\,[\![Geom(n)]\!]\,(\vec{p}) \\
& (\because \text{I.H.}) \\
= \;& \tfrac{1}{2^{|\vec{p}|+1}}.
\end{aligned}
$$

4. Conclusion

We presented a denotational semantics of a probabilistic stream-processing programming language. The semantics exploits the cpo-structure of $\mathcal{D}(D_{\bot}^{\omega})$.

We recognize that the current semantics is a cleaver for cracking a nut; for a node definition that contains $\mathbf{B}^{p}$, by replacing it with a stream variable x and adding it as an argument of the definition, we can hoist every probabilistic expression to the top level. For such a translated program, one can give denotations without using the cpo structure of $\mathcal{D}(D_{\bot}^{\omega})$. However, we believe that the current idea extends well for more complex probabilistic expressions, such as those having parameters that change dynamically.

References

[1] E. A. Ashcroft and W. W. Wadge. Lucid - A formal system for writing and proving programs. *SIAM J. Comput.*, 5(3):336–354, 1976. . URL `http://dx.doi.org/10.1137/0205029`.

[2] P. Caspi, D. Pilaud, N. Halbwachs, and J. Plaice. Lustre: A declarative language for programming synchronous systems. In *POPL 1987*, pages 178–188, 1987.

[3] G. Kahn. The semantics of simple language for parallel programming. In *IFIP Congress*, pages 471–475, 1974.

[4] N. Saheb-Djahromi. Cpo's of measures for nondeterminism. *Theor. Comput. Sci.*, 12:19–37, 1980.

[5] K. Suenaga, H. Sekine, and I. Hasuo. Hyperstream processing systems: nonstandard modeling of continuous-time signals. In *POPL 2013*, pages 417–430, 2013.