Semantics of Reactive Probabilistic Programming

Topos Colloquium, September 2024

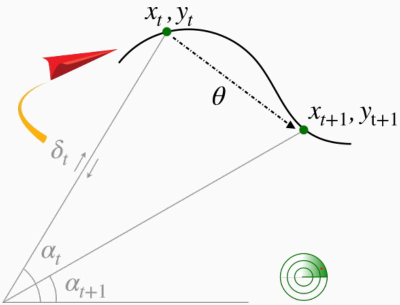
Guillaume Baudart - Louis Mandel - Christine Tasson

Introduction

Model a flight

Flight Tracker

Chopin & Papaspiliopoulos. An introduction to sequential Monte Carlo. 2020



Model evolution of the system

- Cruising speed and altitude
- Straight movement
- Radar tracks the plane

Bayesian inference

- Environment randomly influences the position
- Radar measures are noisy
- What are the conditional distributions of speed and position given radar observations?

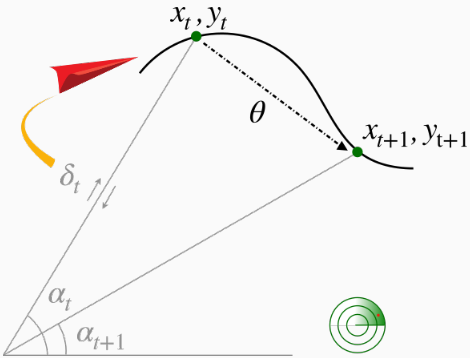
Goal

Study and apply semantics of **probabilistic reactive programming** language
Prove soundness of program transformations.

Reactive Programming

Example

Reactive Flight Tracker



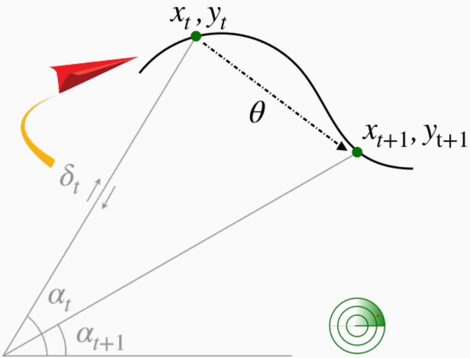
Straight movement

- Cruising altitude
- Constant speed θ
- $\text{pos}_{t+1} = \text{pos}_t + \theta$

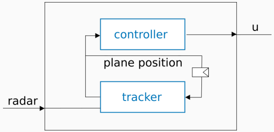
Radar measures: angle and delay

$$\begin{aligned} \text{rad}_t &= (\alpha_t, \delta_t) = f(\text{pos}_t) \text{ with} \\ \alpha_t &= \text{atan}(y_t/x_t) \\ \delta_t &= 2\sqrt{x_t^2+y_t^2}/c_{\ell ight} \end{aligned}$$

Synchronous Flight Tracker



Block diagrams (a la Simulink or Scade)



Synchronous program (a la Lustre or Zelus)

```
node tracker(rad_obs) = (pos, dif) where
  rec  init pos = pos_init
  and pos = last pos + theta
  and rad = f(pos)
  and dif = abs(rad - rad_obs)
node main(rad_obs) = u where
  rec (pos, dif) = tracker(rad_obs)
  and u = controller(pos, dif)
```

Reactive Programming

Synchronous Paradigm

Synchronous Programming

Paul Caspi & al. Lustre, 1987

A language with restricted expressivity, yet strong safety and well-defined semantics

- Synchronous hypothesis
  - simultaneous inputs
  - instantaneous computation
- Simply typed $\Gamma \vdash e : A$
- Productive Recursive Equations $\mathrm{e}\ \textbf{where rec}\ \mathrm{E}$ under fixpoint convergence criteria
- Causality: n-th element of the output stream depends on the n first elements of the input stream
- Deterministic: $[\![e]\!] : \mathsf{Stream}\ \Gamma \to \mathsf{Stream}\ A$

Example

```
node tracker(rad_obs) = (pos, dif)
  where rec  init pos = pos_init
  and pos = last pos + theta
  and rad = f(pos)
  and dif = abs(rad - rad_obs)
```

$$
\begin{aligned}
[\![\mathrm{tracker}]\!]\,(G)_n &= (p_n, d_n) \\
p_0 &= \mathrm{pos\_init} \\
p_n &= p_{n-1} + \theta = p_0 + n\theta, \\
d_n &= |f(p_0 + n\theta) - G_n(\mathrm{rad\_obs})|
\end{aligned}
$$

Topos of Trees

Birkedal & al. (. . .) step-indexing in the topos of trees. LMCS12

Tree $= [\mathbb{N}^{\mathsf{op}}, \mathbf{Set}]$

- $\mathbb{N}$ encodes the time steps.
- Presheaves encode the growing knowledge of the stream when time evolves
- Natural transformations encode causality: outputs depend only on previous inputs

$$\mathbb{N}^{\mathsf{op}} \qquad 0 \leq 1 \leq 2 \leq \ldots$$

$$\text{Stream } \texttt{bool} \qquad \{*\} \xleftarrow{\pi} \mathbb{2} \xleftarrow{\pi} \mathbb{2}^2 \longleftarrow \cdots$$

$$\begin{array}{ccccccc} \mathscr{G} & \qquad & \mathscr{G}(0) \xleftarrow{r^{\mathscr{G}}_{01}} & \mathscr{G}(1) \xleftarrow{r^{\mathscr{G}}_{12}} & \mathscr{G}(2) \longleftarrow \cdots \\ \big\downarrow f = [\![e]\!] & & \big\downarrow f_0 & \big\downarrow f_1 & \big\downarrow f_2 \\ \mathscr{A} & & \mathscr{A}(0) \xleftarrow{r^{\mathscr{A}}_{01}} & \mathscr{A}(1) \xleftarrow{r^{\mathscr{A}}_{12}} & \mathscr{A}(2) \longleftarrow \cdots \end{array}$$

The topos framework to reason on synchronous and guarded reactive languages

Guatto. A Generalized Modality for Recursion. LICS18

Synchronous Programming – Operational Semantics

Caspi & Pouzet, A Co-iterative Characterization of Synchronous Stream Functions, CMCS98

Labelled Transition System

States: Sta (History)

Inputs: $\gamma \in \Gamma$ (Labels)

Outputs: A (Observables)

$\Gamma \vdash e : A$

Projection: $[\![e]\!]^{\mathrm{obs}} : \mathsf{Sta} \to A$

Allocation: $[\![e]\!]^{\mathrm{init}} : \mathsf{Sta}$

Transition: $[\![e]\!]^{\mathrm{step}} : \mathsf{Sta} \times \Gamma \to \mathsf{Sta}$ denoted $S \xrightarrow{\gamma} S'$

Example

```
node tracker(rad_obs) = (pos, dif)
  where rec  init pos = pos_init
  and pos = last pos + theta
  and rad = g(pos)
  and dif = abs(rad - rad_obs)
```

$$[\![\mathrm{tracker}]\!]^{\mathrm{init}} = (\bot, p_0, \bot)$$

$$[\![\mathrm{tracker}]\!]^{\mathrm{step}} : (p_{-1}, p, d) \xrightarrow{\gamma} (p, p+\theta, |f(p+\theta) - g)|$$

with $g = \gamma(\mathrm{rad\_obs})$

$$[\![\mathrm{tracker}]\!]^{\mathrm{obs}} (p_{-1}, p, d) = (p, d)$$

Remark

Memory is bounded as only the last q steps in history are needed with q the number of last.

Synchronous Programming – Soundness and Adequacy

Denotational semantics: Stream function associated to $\Gamma \vdash e : A$.

$$[\![e]\!] : \mathsf{Stream}\ \Gamma \to \mathsf{Stream}\ A$$

Operational semantics: Labeled Transition System associated to $\Gamma \vdash e : A$.

$$[\![e]\!]^{\mathrm{step}} : \qquad \begin{array}{ccccccccccc} [\![e]\!]^{\mathrm{init}} = S_0 & \xrightarrow{\gamma_1} & S_1 & \xrightarrow{\gamma_2} & S_2 & \xrightarrow{\gamma_3} & \dots & \xrightarrow{\gamma_n} & S_n & \xrightarrow{\gamma_{n+1}} & \dots \\ & & {\scriptstyle [\![e]\!]^{\mathrm{obs}}}\downarrow & & \downarrow & & & & \downarrow & & \\ & & v_1 & & v_2 & & \dots & & v_n & & \end{array}$$

$$\text{Denote } \forall n \geq 1,\ [\![e]\!]^{\mathrm{run}}_n (\gamma_1, \dots, \gamma_n) = [\![e]\!]^{\mathrm{obs}} \left([\![e]\!]^{\mathrm{step}} (S_{n-1}, \gamma_n) \right) = v_n$$

Theorem (Equivalence between denotational and operational semantics).

If all recursive equations have a unique solution for every inputs and the program is causal, then

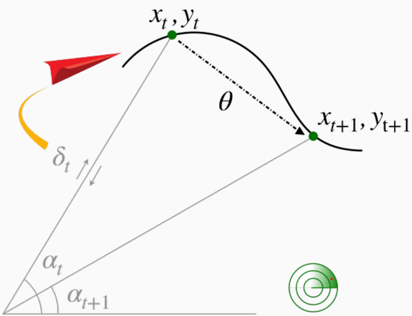
$$\forall G\ \forall n \geq 1,\ [\![e]\!] (G)_n = [\![e]\!]^{\mathrm{run}}_n (G_{\leq n})$$

Probabilistic Reactive Programming

Bayesian Inference

Bayesian Reactive Flight Tracker

Chopin & Papaspiliopoulos. An introduction to sequential Monte Carlo. 2020

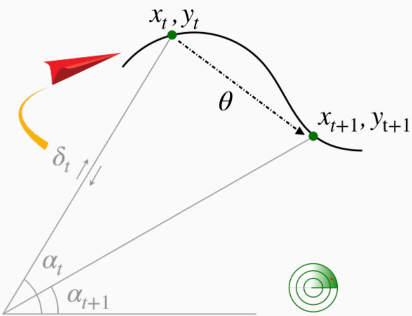


Random environment (prior)

$$
\begin{aligned}
z_t &= 10km \\
\text{pos}_{t+1} &\sim \mathcal{N}(\text{pos}_t + \theta, s_p)
\end{aligned}
$$

Bayesian Reactive Flight Tracker

Chopin & Papaspiliopoulos. An introduction to sequential Monte Carlo. 2020



Random environment (prior)

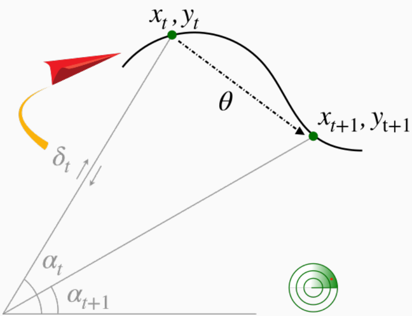
$$
\begin{aligned}
z_t &= 10km \\
\mathrm{pos}_{t+1} &\sim \mathcal{N}(\mathrm{pos}_t + \theta, s_p)
\end{aligned}
$$

Radar: noisy measures (likelihood)

$$
\begin{aligned}
\mathrm{rad}_t &= f(\mathrm{pos}_t) \\
\alpha_t &= \mathrm{atan}(y_t/x_t) \text{ (angle)} \\
\delta_t &= 2\sqrt{x_t^2+y_t^2}/c_{\mathrm{light}} \text{ (delay)} \\
\mathrm{rad\_obs}_t &\sim \mathcal{N}(\mathrm{rad}_t, s_r)
\end{aligned}
$$

Bayesian Reactive Flight Tracker

Chopin & Papaspiliopoulos. An introduction to sequential Monte Carlo. 2020



Random environment (prior)

$$
\begin{aligned}
z_t &= 10km \\
\mathrm{pos}_{t+1} &\sim \mathcal{N}(\mathrm{pos}_t + \theta, s_p)
\end{aligned}
$$

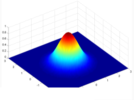
Radar: noisy measures (likelihood)

$$
\begin{aligned}
\mathrm{rad}_t &= f(\mathrm{pos}_t) \\
\alpha_t &= \mathrm{atan}(y_t/x_t) \text{ (angle)} \\
\delta_t &= 2\sqrt{x_t^2+y_t^2}/c_{\mathrm{light}} \text{ (delay)} \\
\mathrm{rad\_obs}_t &\sim \mathcal{N}(\mathrm{rad}_t, s_r)
\end{aligned}
$$

At each time step, what is the (posterior) **conditional distribution** of the position given the observed radar measures ? $\forall n \in \mathbb{N},\ \mathbb{P}(\mathrm{pos}|\mathrm{rad\_obs})_n$

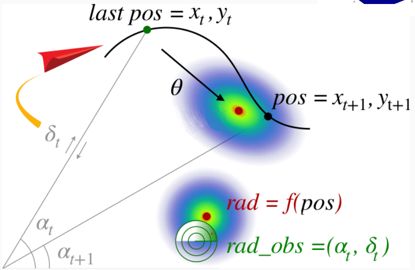
Probabilistic Synchronous Language

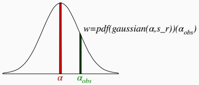
Baudart & al. Reactive Probabilistic Programming, PLDI20



ProbZelus (syntax a la Zelus, Pyro or Stan)

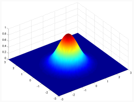
```
proba tracker(rad_obs) = pos where
   rec  init pos = pos_init
   (* prior *)
   and pos = sample(gaussian(last pos+theta, s_p))
   and rad = f(pos)
   (* likelihood / conditionning *)
   and () = observe(gaussian(rad, s_r), rad_obs)

node main(rad_obs) = u where
   (* posterior *)
   rec pos_dist = infer (tracker (rad_obs))
   and u = controller(pos_dist)
```





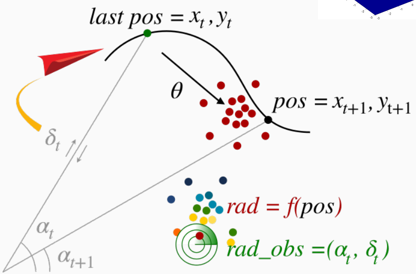
Probabilistic Synchronous Language

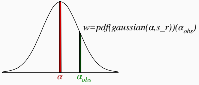
Baudart & al. Reactive Probabilistic Programming, PLDI20



ProbZelus (syntax a la Zelus, Pyro or Stan)

```
proba tracker(rad_obs) = pos where
  rec  init pos = pos_init
  (* prior *)
  and pos = sample(gaussian(last pos+theta, s_p))
  and rad = f(pos)
  (* likelihood / conditionning *)
  and () = observe(gaussian(rad, s_r), rad_obs)

node main(rad_obs) = u where
  (* posterior *)
  rec pos_dist = infer (tracker (rad_obs))
  and u = controller(pos_dist)
```





Sequential Monte-Carlo Inference

sample: $[(pos^0,\ 1\ ), \ldots, (pos^n,\ 1\ )]$

observe: $\underbrace{[(pos^0, w^0), \ldots, (pos^n, w^n)]}_{\text{categorical distribution}}$

Probabilistic Reactive Programming

Semantics

Probabilistic Synchronous Programming – Denotational Semantics

Stream of probabilistic measures

$$[\![\Gamma \vdash \texttt{infer}\ e : \text{Prob}\ A]\!] : \text{Stream}\ \Gamma \to \text{Stream}\ (\text{Prob}\ A)$$

Solving recursive equations towards a schedule-agnostic semantics

- inherited from block diagrams that are standard in the industry,
- manually scheduling is not modular.

Problem to compute fixpoints in the measure semantics:

```
e =  (x, y) where
       rec  x = sample(gaussian(42, 1))
       and  y = x
```

Wanted semantics:

$[\![e]\!] = \int_{\mathbb{R}} \delta_{(x,x)}\, \mathcal{N}(42,1)(x) dx$

Yet, in the measure semantics, the least element (and least fixpoint) is the null measure.

Jones & Plotkin. A Probabilistic Powerdomain of Evaluations. 1998

Solution: externalize random seeds and compute fixpoint in the value domain

Vakar & al. A domain Theory for Statistical Probabilistic Programming. POPL2019

Probabilistic Synchronous Programming – Denotational Semantics

Stream of probabilistic measures

$$[\![\Gamma \vdash \texttt{infer}\ e : \text{Proba}\ A]\!] : \text{Stream}\ \Gamma \rightarrow \text{Stream}\ (\text{Proba}\ A)$$

Externalize randomness in order to solve recursive equations:
If probability distributions have density wrt the counting or the Lebesgue measures, then

$$\rho(U) = \int_{[0,1]} \delta_{icdf_{\rho}(r) \in U} dr$$

with $r \in [0,1]$ a random seed and $icdf_{\rho}(r)$ its inverse cumulative distribution function.

Sampling semantics: if k is the number of samples, then

$$(\!|e|\!) : \text{Stream}\ \Gamma \times \text{Stream}\ [0,1]^k \rightarrow \text{Stream}\ A \times \text{Stream}\ \mathbb{R}^+$$

Stochastic semantics: if $(v_n, w_n) = (\!|e|\!)\,(G, R)_n$, then

$$\forall \vec{\gamma},\ \forall n,\ [\![e]\!]\,(G)_n = \int_{([0,1]^k)^{\mathbb{N}}} \delta_{v_n} w_n\, dR = \int_{([0,1]^k)^n} \delta_{v_n} w_n\, dR_{\leq n}$$

Probabilistic Synchronous Programming – Operational Semantics

Sampling Labelled Transition System

States: $\mathsf{Sta} \times \mathbb{R}^+$

(History and score)

Inputs: $\gamma \in \Gamma$ (Labels)

Outputs: A (Observables)

Projection: $(\!| e |\!)^{\mathrm{obs}} : \mathsf{Sta} \times \mathbb{R}^+ \to A \times \mathbb{R}^+$

Allocation: $(\!| e |\!)^{\mathrm{init}} : \mathsf{Sta} \times \mathbb{R}^+$

Sampling Transition: $(\!| e |\!)^{\mathrm{step}} : (S, w) \xrightarrow{\gamma, r} (S', w')$

with $\gamma \in \Gamma$, $r \in [0,1]^k$ and $w, w' \in \mathbb{R}^+$

Stochastic Labelled Transition System: if $(S', w') = (\!| e |\!)^{\mathrm{step}}(S, w, \gamma, r)$, then

$$[\![e]\!]^{\mathrm{step}} : S \in \mathsf{Sta} \xrightarrow{\gamma} \int_{[0,1]^k} \delta_{S'}\, w'\, dr \in \mathsf{Prob\ Sta}$$

Probabilistic Synchronous Programming – Example

Syntax

```
node tracker(rad_obs) = pos
  where rec  init pos = pos_init
  and pos = sample(gaussian(last pos + theta, s_p))
  and rad = f(pos)
  and () = observe(gaussian(rad, s_r), rad_obs)
```

Operational semantics: with states $(\mathrm{pos\_last}, \mathrm{pos}) \in \mathsf{Sta}$

$$[\![\mathrm{tracker}]\!]^{\mathrm{obs}} : \quad (p_{-1}, p) \mapsto \quad p$$

$$[\![\mathrm{tracker}]\!]^{\mathrm{init}} : \quad (\bot, p_0), 1$$

$$[\![\mathrm{tracker}]\!]^{\mathrm{step}} : \quad (p_{-1}, p), w \xrightarrow{\gamma, r} \begin{cases} S' = (p, p' + \theta) & \text{with } p' = icdf_{\mathcal{N}(p, s_p)}(r) \text{ in} \\ w' = w * \mathcal{N}(f(p' + \theta), s_r)(g) & \text{with } g = \gamma(\mathrm{rad\_obs}) \end{cases}$$

Probabilistic Reactive Semantics – Soundness and Adequacy

Denotational semantics: Stream function associated to $\Gamma \vdash e : \text{Meas A}$

$$(\!|e|\!) : \text{Stream } \Gamma \to \text{Stream } A \times \text{Stream } \mathbb{R}^+$$

Operational semantics: Labeled Transition System associated to $\Gamma \vdash e : \text{Meas A}$

$$\begin{array}{llllllll} (\!|e|\!)^{\text{step}} : (\!|e|\!)^{\text{init}} = S_0, 1 & \xrightarrow{\gamma_1, R_1} & S_1, w_1 & \xrightarrow{\gamma_2, R_2} & S_2, w_2 & \xrightarrow{\gamma_3, R_3} \dots \xrightarrow{\gamma_n, R_n} & S_n, w_n & \xrightarrow{\gamma_{n+1}, R_{n+1}} \dots \\ & & (\!|e|\!)^{\text{obs}} \downarrow & & \downarrow & & \downarrow & \\ & & v_1, w_1 & & v_2, w_2 & \dots & v_n, w_n & \end{array}$$

Set $\forall n \geq 1,\ (\!|e|\!)^{\text{run}}_n (\gamma_1, \dots, \gamma_n, R_1, \dots, R_n) = (\!|e|\!)^{\text{obs}} ((\!|e|\!)^{\text{step}} (S_{n-1}, w_{n-1}, \gamma_n, R_n)) = v_n, w_n$

Theorem (Equivalence between denotational and operational semantics)

If all recursive equations have a unique solution for every inputs and the program is causal, then for any input stream G, and for any random seeds stream R,

$$\forall n \geq 1,\ (\!|e|\!) (G)_n = (\!|e|\!)^{\text{run}}_n (G_{\leq n}, R_{\leq n})$$

Thus, the denotational and operational output probability measures coincide at each time step.

Program Equivalence

Observational Equivalence

Observational equivalence (operational)

$$\texttt{sample}(e_1) + \texttt{sample}(e_2) \overset{\texttt{obs}}{\simeq} x + y \;\texttt{where rec}\; x = \texttt{sample}(e_2) \;\texttt{and}\; y = \texttt{sample}(e_1)$$

Definition: $e_1 \overset{\texttt{obs}}{\simeq} e_2$ if for all input stream G, $[\![e_1]\!](G) = [\![e_2]\!](G)$.

Stochastic bisimulation: $e_1 \sim e_2$ if there is $\mathscr{C} \subseteq \text{Sta} \times \text{Sta}$ such that for all γ, for all $s_1 \mathscr{C} s_2$, if $s_1 \xrightarrow[(\!| e_1 |\!)]{\gamma} \varphi_1$, then there is φ_2 with $s_2 \xrightarrow[(\!| e_2 |\!)]{\gamma} \varphi_2$ such that

- there is a coupling $C \in \text{Proba}\,(\text{Sta} \times \text{Sta})$ with marginals φ_1 and φ_2
- there is a measurable relation on pair of states $\mathscr{C}' \subseteq \mathscr{C}$ such that

$$C(\mathscr{C}') = 1 \qquad \forall s'_1 \mathscr{C}' s'_2,\; \texttt{obs}_{(\!| e_1 |\!)}(s'_1) = \texttt{obs}_{(\!| e_2 |\!)}(s'_2)$$

et vice versa.

Theorem: If $e_1 \sim e_2$, then $e_1 \overset{\texttt{obs}}{\simeq} e_2$.

Proof: consequence of adequacy.

Observational Equivalence (Denotational)

$$\texttt{sample}(e_1) + \texttt{sample}(e_2) \overset{\texttt{obs}}{\simeq} x + y \;\texttt{where rec}\; x = \texttt{sample}(e_2) \;\texttt{and}\; y = \texttt{sample}(e_1)$$

Sampling bisimulation: $e_1 \overset{\texttt{sam}}{\simeq} e_2$ if there is $\psi : [0,1]^{k_1} \to [0,1]^{k_2}$

- preserving uniform distribution $\psi_*(\lambda^{k_1}) = \lambda^{k_2}$
- $\forall G, R \in \text{Stream } (\Gamma \times [0,1]^{k_1})$, $(\!|e_1|\!)\,(G,R) = (\!|e_2|\!)\,(G, \psi(R))$ with $\psi(R) = (\psi(R_n))_{n \in \mathbb{N}}$

Theorem: If $e_1 \overset{\texttt{sam}}{\simeq} e_2$, then $e_1 \overset{\texttt{obs}}{\simeq} e_2$.

Proof: We apply the change of variable formula along ψ, set $s_i(G,R), w_i(G,R) = (\!|e_i|\!)\,(G,R)$

$$\begin{aligned}
[\![e_1]\!]\,(G) = \int_{([0,1]^{k_1})^{\mathbb{N}}} w_1(G,R)\delta_{s_1(G,R)} d\lambda^{k_1}(R) &= \int_{([0,1]^{k_1})^{\mathbb{N}}} w_2(G,\psi(R))\delta_{s_2(G,\psi(R))} d\lambda^{k_1}(R) \\
&= \int_{([0,1]^{k_2})^{\mathbb{N}}} w_2(G,R')\delta_{s_2(G,R')} d\lambda^{k_2}(R') \\
&= [\![e_2]\!]\,(G)
\end{aligned}$$

Stream Sampling Semantics

adapted from Bourke et al. Velus, 2017

Inference system (selected rules): $G, R \vdash e \downarrow s, w$

$$\frac{F, G \vdash e \downarrow s}{F, G, [] \vdash e \Downarrow (s, 1)} \qquad \frac{F, G \vdash e \downarrow s_\mu}{F, G, [R] \vdash \texttt{sample}(e) \Downarrow (icdf_{s_\mu}(R), 1)} \qquad \frac{F, G \vdash e \downarrow w}{F, G, [] \vdash \texttt{factor}(e) \Downarrow ((), w)}$$

$$\frac{F, G, R_e \vdash e \downarrow (s_e, w_e) \qquad F(f) = \texttt{proba}\ f\ x = e_f \qquad F, [x \leftarrow s_e], R_f \vdash e_f \Downarrow (s, w)}{F, G, [R_e : R_f] \vdash f(e) \Downarrow (s, w * w_e)}$$

$$\frac{F, G + G_E, R_E \vdash E : w_E \qquad F, G + G_E, R_e \vdash e \Downarrow (s, w)}{F, G, [R_e : R_E] \vdash e\ \texttt{where rec}\ E \Downarrow (s, w * w_E)} \qquad \frac{F, G, R \vdash e \Downarrow (G(x), w)}{F, G, R \vdash x = e : w}$$

$$\frac{F, G, R \vdash e \Downarrow (i \cdot s, w_i \cdot w) \qquad G(x.\mathrm{last}) = i \cdot G(x)}{F, G, R \vdash \texttt{init}\ x = e : w_i \cdot 1} \qquad \frac{F, G, R_1 \vdash E_1 : w_1 \qquad F, G, R_2 \vdash E_2 : w_2}{F, G, [R_1 : R_2] \vdash E_1\ \texttt{and}\ E_2 : w_1 * w_2}$$

$$\frac{p = \mathrm{RV}(e) \qquad [F, G, R \vdash e \Downarrow (s, w) \qquad \overline{w} = \Pi\ w]_{R \in ([0,1]^\omega)^p}}{F, G \vdash \texttt{infer}(e) \downarrow integ_p\ \overline{w}\ s}$$

Soundness: $G, R \vdash e \downarrow s, w$ if and only if $(s, w) = [\![e]\!](G, R)$

Program Equivalence – Commutativity

$$\texttt{sample}(e_1) + \texttt{sample}(e_2) \overset{\texttt{obs}}{\simeq} x + y \;\texttt{where rec}\; x = \texttt{sample}(e_2) \;\texttt{and}\; y = \texttt{sample}(e_1)$$

Program Equivalence – Commutativity

$$\texttt{sample}(e_1) + \texttt{sample}(e_2) \overset{\texttt{obs}}{\simeq} x + y \texttt{ where rec } x = \texttt{sample}(e_2) \texttt{ and } y = \texttt{sample}(e_1)$$

$$\frac{G, R_1 \vdash \texttt{sample}(e_1) \Downarrow (s_1, w_1) \qquad G, R_2 \vdash \texttt{sample}(e_2) \Downarrow (s_2, w_2)}{G, [R_1 : R_2] \vdash \texttt{sample}(e_1) + \texttt{sample}(e_2) \Downarrow (s_1 + s_2, w_1 w_2)}$$

Program Equivalence – Commutativity

$$\texttt{sample}(e_1) + \texttt{sample}(e_2) \overset{\texttt{obs}}{\simeq} x + y \texttt{ where rec } x = \texttt{sample}(e_2) \texttt{ and } y = \texttt{sample}(e_1)$$

$$\frac{G, R_1 \vdash \texttt{sample}(e_1) \Downarrow (s_1, w_1) \qquad G, R_2 \vdash \texttt{sample}(e_2) \Downarrow (s_2, w_2)}{G, [R_1 : R_2] \vdash \texttt{sample}(e_1) + \texttt{sample}(e_2) \Downarrow (s_1 + s_2, w_1 w_2)}$$

$$\frac{G + G_E, [] \vdash x + y \Downarrow (s_2 + s_1, 1) \qquad \dfrac{\dfrac{G + G_E, R_2 \vdash \texttt{sample}(e_2) \Downarrow (s_2, w_2)}{G + G_E, R_2 \vdash x = \texttt{sample}(e_2) : w_2} \qquad \dfrac{G + G_E, R_1 \vdash \texttt{sample}(e_1) \Downarrow (s_1, w_1)}{G + G_E, R_1 \vdash y = \texttt{sample}(e_1) : w_1}}{G + G_E, [R_2 : R_1] \vdash x = \texttt{sample}(e_2) \texttt{ and } y = \texttt{sample}(e_1) : w_1 w_2}}{G, [R_2 : R_1] \vdash x + y \texttt{ where rec } x = \texttt{sample}(e_2) \texttt{ and } y = \texttt{sample}(e_1) \Downarrow (s_2 + s_1, w_1 w_2)}$$

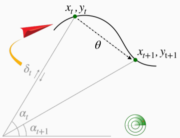
where $G_E = [x \leftarrow s_2, y \leftarrow s_1]$.

Program Equivalence

Application – Assumed Parameter Filter

Assumed Parameter Filter (APF) Inference

Erol & al. A nearly-black-box online algorithm for joint parameter and state estimation in temporal models, 2017




```
proba f(pre_x) = pre_x + theta where
  rec init theta = sample(gaussian(zeros, st))
  and theta = last theta

proba tracker(rad_obs) = pos where
  rec  init pos = pos_init
  and pos = sample(gaussian(f(last pos), sp))
  and rad = g(pos)
  and () = observe(gaussian(rad, sr), rad_obs)

node main(rad_obs) = u where
  rec pos_dist = infer (tracker (rad_obs))
  and msg = controller(pos_dist)
```

At each time step, different methods for

- state parameters
  sequential Monte-Carlo inference
- constant parameters
  symbolic inference and optimization

APF necessitates a program transformation to extract constant parameters.

Program Transformation for APF – Soundness

```
proba f(pre_x) = pre_x + theta where
  rec init theta = sample(gaussian(zeros, st))
  and theta = last theta

proba tracker(rad_obs) = pos where
  rec  init pos = pos_init
  and pos = sample(gaussian(f(last pos), sp))
  and rad = g(pos)
  and () = observe(gaussian(rad, sr), rad_obs)

node main(rad_obs) = u where
  rec pos_dist = infer (tracker (rad_obs))
  and msg = controller(pos_dist)
```

```
let f_prior = gaussian(zeros, st)
proba f_model(theta, pre_pos) = pre_pos + theta

let tracker_prior = f_prior

proba tracker_model(theta, rad_obs) = pos where
  rec init pos = pos_init
  and pos = sample(gaussian(f_prior(theta, last pos), sp))
  and rad = g(pos)
  and () = observe(gaussian(rad, sr), rad_obs)

node main(rad_obs) = msg where
  rec pos_dist = APF.infer(tracker_model, tracker_prior, rad_obs)
  and msg = controller(pos_dist)
```

APF Inference definition

$$\texttt{APF.infer}(f.\mathrm{model},\ f.\mathrm{prior},\ e) \stackrel{\Delta}{=} \texttt{infer}(f.\mathrm{model}(\theta, e)\ \texttt{where rec init}\ \theta = \texttt{sample}(f.\mathrm{prior}))$$

Soundness: $F, G \vdash \texttt{infer}(f(e)) \downarrow d \quad \text{iff} \quad F^{+}, G \vdash \texttt{APF.infer}(f.\mathrm{model},\ f.\mathrm{prior},\ e) \downarrow d$

Proofs: By sampling bisimulation (using stream functions) or stochastic bisimulation (using states and labeled transition systems).

Probabilistic Reactive Programming

arXiv Baudart, Mandel, Tasson, Density-Based Semantics for Reactive Probabilistic Programming, 2023

Equivalent Semantics for Probabilistic Reactive Programming,
with observational equivalence characterization

- Operational semantics (sLTS), with stochastic bisimulation
- Sampling semantics (stream functions), with sampling bisimulation

Proofs of Equivalence of Probabilistic Reactive Programs

- Basic equations
- Transformation of programs

G. Kahn, The Semantics of a Simple Language for Parallel Programming, 1974

Future works

- Recursive equations in Probabilistic Programming
- Probabilistic distance between inference algorithms