Probabilistic Lazy PCF with Real-Valued Choice

David Sabel
Hochschule RheinMain
Wiesbaden

Manfred Schmidt-Schauß
Goethe-University
Frankfurt am Main

WPTE 2025, Birmingham, UK, 20.07.2025

Motivation

Probabilistic Programming + **Call-by-Need Functional Programming Languages**

- probabilistic programs represent stochastic models
- program execution is performing a probabilistic experiment
- reasoning on program semantics is reasoning on the models

- declarative, high-level programming allowing equational reasoning
- efficient implementation of lazy evaluation
- semantics is different from call-by-name and call-by-value

$\longrightarrow$ **Investigate the semantics of probabilistic call-by-need functional languages**

Evaluation Strategies

let $(m \oplus n)$ represent fair probabilistic choice

Example: $(\lambda x, y.x + x)\ (1 \oplus 2)\ (3 \oplus \bot)$

Possible results with their respective probabilities

Result	Call-by-Name	Call-by-Value	Call-By-Need
2	0.25	0.25	0.5
3	0.5	impossible	impossible
4	0.25	0.25	0.5
$\bot$	impossible	0.5	impossible

$\rightarrow$ **all three strategies are different**

Previous Work

Probablistic call-by-need calculus with recursive let [PPDP 2022]

- correctness of program transformations
- proof techniques for proving contextual equivalences

Probabilistic Lazy PCF [WPTE 2022, JLAMP 2023]

- PCF: simply typed λ-calculus $+$ numbers $+$ fix-point operator
- call-by-need-evaluation with explicit sharing by `let`
- probabilistic fair choice $s \oplus t$ evaluates to s or t both with probability 0.5
- result: distribution equivalence $=$ contextual equivalence on programs of type nat

Goals

- Add probabilistic choice $(s \overset{r}{\oplus} t)$ with (computable) **real-valued probability** r:
  - s is chosen with probability r
  - t with probability $1 - r$

  Does this change the **expressivity** of the language?

  Do former **results on the program semantics** still hold?

- Develop techniques to approximate distribution equivalence (work in progress)

Syntax of Probabilistic Lazy PCF and the Extension

$ProbPCF^{need}$	$ProbPCF_{\mathbb{R}}^{need}$

Expressions: $s, t \in \mathit{Exp} ::= x \mid \lambda x.s \mid (s\ t) \mid \textbf{fix}\ \ s \mid \texttt{if}\ s\ \texttt{then}\ t_1\ \texttt{else}\ t_2$
$\mid \texttt{pred}\ s \mid \texttt{succ}\ s \mid \texttt{let}\ x = s\ \texttt{in}\ t \mid n$ where $n \in \mathbb{N}$

$ProbPCF^{need}$	$ProbPCF_{\mathbb{R}}^{need}$
$\mid (s \oplus t)$	$\mid (s \overset{r}{\oplus} t)$ where $r \in (0,1)$ is computable

Types: $\tau, \sigma \in \mathit{Typ} ::= nat \mid \tau \to \sigma$

Type check: standard monomorphic type system, $s \in \mathit{Exp}$ is well-typed iff $s : \tau$

Operational Semantics: Small-Step Reduction $\xrightarrow{sr}$

$ProbPCF^{need}$	$ProbPCF^{need}_{\mathbb{R}}$

(*sr,lbeta*)	$R[(\lambda x.s)\ t] \xrightarrow{sr} R[\texttt{let}\ x = t\ \texttt{in}\ s]$
(*sr,if-0*)	$R[\texttt{if}\ 0\ \texttt{then}\ s\ \texttt{else}\ t] \xrightarrow{sr} R[s]$
(*sr,if-not-0*)	$R[\texttt{if}\ n\ \texttt{then}\ s\ \texttt{else}\ t] \xrightarrow{sr} R[t]$ if $n > 0$
...	...

ProbPCF^need:

		probability
(*sr,probl*)	$R[s \oplus t] \xrightarrow{sr} R[s]$	0.5
(*sr,probr*)	$R[s \oplus t] \xrightarrow{sr} R[t]$	0.5

ProbPCF^need_R:

		probability
(*sr,probl*)	$R[s \overset{r}{\oplus} t] \xrightarrow{sr} R[s]$	r
(*sr,probr*)	$R[s \overset{r}{\oplus} t] \xrightarrow{sr} R[t]$	$1 - r$

"prob-steps"

where **reduction contexts** are

$R ::= \mathrm{LR}[A] \mid \mathrm{LR}[\texttt{let}\ x = A\ \texttt{in}\ R[x]]$ $\qquad$ $\mathrm{LR} ::= [\cdot] \mid \texttt{let}\ x = s\ \texttt{in}\ \mathrm{LR}$

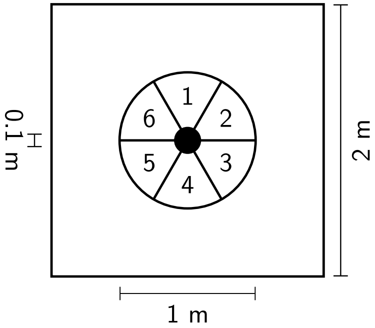
$A ::= [\cdot] \mid (A\ s) \mid \texttt{if}\ A\ \texttt{then}\ s\ \texttt{else}\ t \mid \texttt{pred}\ A \mid \texttt{succ}\ A \mid \texttt{fix}\ A$

Expected Convergence

- An **evaluation** S of s: $s \xrightarrow{sr,a_1} \cdots \xrightarrow{sr,a_n} t$ where $t = LR[v]$ is a weak head normal form ($LR ::= [\cdot] \mid \texttt{let}\ x = s\ \texttt{in}\ LR$ and v is a number n or an abstraction $\lambda x.s$).
- **Probability of an evaluation** $\mathsf{P}(S)$: product of all probability measures of all prob-steps in $s \xrightarrow{sr,a_1} \cdots \xrightarrow{sr,a_n} t$
- **Expected convergence** $\mathrm{ExCv}(s)$ = sum of the probabilities of all evaluations of s
- **Expected value convergence** $\mathrm{ExVCv}(s,n)$ = sum of the probabilities of all evaluations of s ending in the number n

$$\mathrm{ExCv}(s) := \sum_{S \in \mathit{Eval}(s)} \mathsf{P}(S) \quad \text{and} \quad \mathrm{ExVCv}(s,n) := \sum_{\substack{S \in \mathit{Eval}(s),\\ val(WHNF(s,S)) = n}} \mathsf{P}(S)$$

Example: Randomly Throwing Darts (Simplified)



$$
\begin{aligned}
&\mathit{throwDart} = \\
&\quad \texttt{let } \mathit{wall} = 0 \texttt{ in} \\
&\quad \texttt{let } \mathit{segment} = 1 \overset{1/6}{\oplus} (2 \overset{1/5}{\oplus} (3 \overset{1/4}{\oplus} (4 \overset{1/3}{\oplus} (5 \overset{1/2}{\oplus} 6)))) \texttt{ in} \\
&\quad \texttt{let } \mathit{bullseye} = 10 \texttt{ in} \\
&\quad \texttt{let } \mathit{board} = \mathit{bullseye} \overset{1/100}{\oplus} \mathit{segment} \\
&\quad \texttt{in } \mathit{board} \overset{\pi/16}{\oplus} \mathit{wall}
\end{aligned}
$$

Some expected convergences:

$\mathrm{ExCv}(\mathit{throwDart}) = 1$

Context C tests if the board is hit:
$C = \texttt{if } [\cdot] \texttt{ then } \bot \texttt{ else } 1$

$\mathrm{ExCv}(C[\mathit{throwDart}]) = \pi/16 \approx 19.63\%$

Expected value convergences:

$$
\begin{aligned}
\mathrm{ExVCv}(\mathit{throwDart}, 0) &= 1 - \pi/16 && \approx 80.37\% \\
\mathrm{ExVCv}(\mathit{throwDart}, 1) &= \ldots = \\
\mathrm{ExVCv}(\mathit{throwDart}, 6) &= \pi/16 \cdot 99/100 \cdot 1/6 && \approx 3.24\% \\
\mathrm{ExVCv}(\mathit{throwDart}, 10) &= \pi/16 \cdot 1/100 && \approx 0.2\% \\
\mathrm{ExVCv}(\mathit{throwDart}, i) &= 0 \text{ for } i \notin \{0,1,2,3,4,5,6,10\}
\end{aligned}
$$

Contextual Equivalence and Distribution Equivalence

For expressions $s, t : \sigma$:

Contextual preorder	$s \leq_c t$	iff	$\forall C[\cdot_\sigma] : nat$: $\mathrm{ExCv}(C[s]) \leq \mathrm{ExCv}(C[t])$
Contextual equivalence	$s \sim_c t$	iff	$s \leq_c t$ and $t \leq_c s$

For closed expressions $s, t : nat$:

Distribution approximation	$s \leq_d t$	iff	$\forall i \in \mathbb{N} : \mathrm{ExVCv}(s, i) \leq_d \mathrm{ExVCv}(t, i)$
Distribution equivalence	$s \sim_d t$	iff	$s \leq_d t$ and $t \leq_d s$

Example: $a \overset{1/6}{\oplus} (b \overset{1/5}{\oplus} (c \overset{1/4}{\oplus} (d \overset{1/3}{\oplus} (e \overset{1/2}{\oplus} f)))) \sim_d ((a \overset{1/2}{\oplus} b) \overset{2/3}{\oplus} c) \overset{1/2}{\oplus} (d \overset{1/3}{\oplus} (e \overset{1/2}{\oplus} f))$

Theorem

For closed $s, t : nat$: $s \sim_c t \iff s \sim_d t$

Conjecture (work in progress)

For closed $s, t : nat$: $s \leq_c t \iff s \leq_d t$

Conservativity

We also use distribution equivalence to compare expressions in both calculi

Theorem

For every closed $s : nat$ in $ProbPCF_{\mathbb{R}}^{need}$ there exists a distribution-equivalent closed $s' : nat$ in $ProbPCF^{need}$.

Requires to encode $(s \overset{r}{\oplus} t)$ using fair choice $(s \oplus t)$ only.

Approach: Use bitwise fair choice to simulate arbitrary probabilistic choice (well-known, e.g. Arora & Barak, 2009 for Probabilistic Turing Machines)

Encoding Real-Valued Choice with Fair Choice

Ideas:

- use the bit-expansion of $r \in (0,1) = 0.b_1 b_2 \ldots$ (where $r = \sum_{i=1}^{\infty} \frac{b_i}{2^i}$)
- since r is computable, the bit-expansion is computable
- simulate $(s \overset{r}{\oplus} t)$ by calling $g\ 1$ where g is the recursive function

$$
\begin{aligned}
g\ i = \texttt{if}\ b_i = 1\ \texttt{then}\ & s \oplus (g\ (i+1)) \\
\texttt{else}\ & t \oplus (g\ (i+1))
\end{aligned}
$$

- $g\ 1$ unfolds to $u_1 \oplus (u_2 \oplus (u_3 \ldots$ where $u_i = \begin{cases} s, & \text{if } b_i = 1 \\ t, & \text{if } b_i = 0 \end{cases}$
- in call-by-need: s and t are shared (no duplication)

The Encoding in Probabilistic Lazy PCF

Encoding $enc : ProbPCF^{need}_{\mathbb{R}} \to ProbPCF^{need}$

$enc(F\ s_1 \ldots s_n) = F\ enc(s_1)\ \ldots\ enc(s_n)$ for all language constructs $F \neq \overset{r}{\oplus}$

$$
\begin{aligned}
enc(s \overset{r}{\oplus} t) = \ & \texttt{let}\ f_r = \ldots\ \texttt{in} \\
& \textbf{fix}\ (\lambda g, i, x, y.\ \texttt{if}\ (f_r\ i)\ \texttt{then}\ x \oplus (g\ (\texttt{succ}\ i)\ x\ y) \\
& \qquad\qquad\qquad\qquad\qquad \texttt{else}\ y \oplus (g\ (\texttt{succ}\ i)\ x\ y)) \\
& \quad 1\ enc(s)\ enc(t)
\end{aligned}
$$

where f_r computes the inverted bit expansion $f_r(i) = 1 - b_i$ of $r = \sum_{i=1}^{\infty} \frac{b_i}{2^i}$

$enc(s \overset{r}{\oplus} t)$ unfolds to $\begin{pmatrix} \texttt{let}\ x = enc(s)\ \texttt{in} \\ \texttt{let}\ y = enc(t)\ \texttt{in} \\ (z_1 \oplus (z_2 \oplus \ldots)) \end{pmatrix}$ where $z_i = \begin{cases} x, & \text{if } b_i = 1 \\ y, & \text{if } b_i = 0 \end{cases}$

Example

$(m \overset{1/3}{\oplus} n)$

- the bit-expansion of $1/3$ is $0.01010101010101\ldots$
- the inverted sequence can be computed by $f_{1/3} = \lambda i.(i \bmod 2)$
- the encoding $s = enc(m \overset{1/3}{\oplus} n) = \texttt{let}\ f_{1/3} = \lambda i.(i \bmod 2)\ \texttt{in}\ \textbf{fix}\ \ \ldots$

  unfolds to $n \oplus (m \oplus (n \oplus (m \oplus (n \oplus \ldots$
- as expected:

$$\mathrm{ExVCv}(s, m) = \sum_{i \in \mathbb{N}} \frac{1}{2^{2(i+1)}} = \frac{1}{3} \text{ and } \mathrm{ExVCv}(s, n) = \sum_{i \in \mathbb{N}} \frac{1}{2^{2i+1}} = \frac{2}{3}$$

Conservativity

Theorem

For every closed $s : nat$ in $ProbPCF_{\mathbb{R}}^{need}$ there exists a distribution-equivalent closed $s' : nat$ in $ProbPCF^{need}$.

Proof:

- iteratively replaces each $s \overset{r}{\oplus} t$ with $enc(s \overset{r}{\oplus} t)$.
- each step requires the equation:

$$\text{for prob-free } s, t\text{: } C[s \overset{r}{\oplus} t] \sim_d C[enc(s \overset{r}{\oplus} t)]$$

Proving $C[s \overset{r}{\oplus} t] \sim_d C[enc(s \overset{r}{\oplus} t)]$

Proposition (Equation in Reduction Contexts)

$R[s \overset{r}{\oplus} t] \sim_d R[enc(s \overset{r}{\oplus} t)]$ if s, t are prob-free and $R[s \overset{r}{\oplus} t] : nat$ is closed.

$\mathrm{ExVCv}(R[s \overset{r}{\oplus} t], n) = \mathrm{ExVCv}(R[enc(s \overset{r}{\oplus} t)], n)$ is proved by:

1. For all $k \in \mathbb{N}$: $\mathrm{ExVCv}(R[enc(s \overset{r}{\oplus} t)], n, k) \leq \mathrm{ExVCv}(R[s \overset{r}{\oplus} t], n)$
2. $\forall \varepsilon > 0 : \exists k$: $\mathrm{ExVCv}(R[s \overset{r}{\oplus} t], n) - \mathrm{ExVCv}(R[enc(s \overset{r}{\oplus} t)], n, k) < \varepsilon$

Additional parameter k: at most k prob-steps are permitted

Proposition (Context Lemma for $\sim_d$)

Let $s, t : \sigma$ and for all closing $R[\cdot_\sigma] : nat$: $R[s] \sim_d R[t]$. Then $C[s, \ldots, s] \leq_d C[t, \ldots, t]$, if $C[\cdot_{1,\sigma} \ldots, \cdot_{n,\sigma}] : nat$ is closing.

Again: the proof uses $\mathrm{ExVCv}(\cdot, \cdot, k)$ where k restricts the number of prob-steps.

Conclusion

Summary

- extension by real-valued probabilistic choice is conservative w.r.t. $\sim_d$ in Probabilistic Lazy PCF
- we applied the well-known technique exploiting the computable bit-expansion
- proofs on $\sim_d$: enable inductive proofs by restricting the number of prob-steps

Future Work

- prove the conjecture $\leq_c = \leq_d$
- investigate algorithmic approximations of probabilistic (closed) programs:
  - again by restricting the number of prob-steps in evaluations
  - by restricting the number of sr-steps and perhaps stopping with no result
  - by encodings that stop after performing a limit of prob-steps

Thank You!