Approximate Relational Reasoning for Higher-Order Probabilistic Programs

Philipp G. Haselwarter[1], Kwing Hei Li[1], Alejandro Aguirre[1], Simon Oddershede Gregersen[2], Joseph Tassarotti[2], and Lars Birkedal[1]

[1]Aarhus University, [2]New York University

Motivation

Probabilistic Specifications for Probabilistic Programs

Correct randomized programs compute results approximately!

Goal: Bound error probability.

Probabilistic Specifications for Probabilistic Programs

Correct randomized programs compute results approximately!

Goal: Bound error probability.

Example from cryptography (2^n possible keys, usually $Q \ll 2^n$):

$$\left|\Pr\left[\mathcal{A}(\mathrm{enc}\,(\mathrm{keygen}\,())_{|Q}) = 1\right] - \Pr\left[\mathcal{A}(\mathrm{rand\_cipher}_{|Q}) = 1\right]\right| \leq \frac{Q^2}{2^{n+1}}$$

Specification:

“enc behaves (almost) like the uniform distribution on ciphertexts.”

Probabilistic Specifications for Probabilistic Programs

Correct randomized programs may take arbitrarily long to run!

Goal: Prove equivalence despite different internal use of randomness.

Probabilistic Specifications for Probabilistic Programs

Correct randomized programs may take arbitrarily long to run!

Goal: Prove equivalence despite different internal use of randomness.

Let $M < N$.

```
let direct _ =
  rand M
```

```
let reject _ =
  (rec sampler _ =
    let x = rand N in
    if x ≤ M then x else sampler ()) ()
```

Claim:

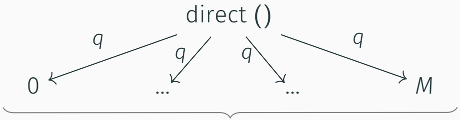
Both functions compute the uniform distribution on $\{0, \ldots, M\}$.

Probabilistic Specifications for Probabilistic Programs

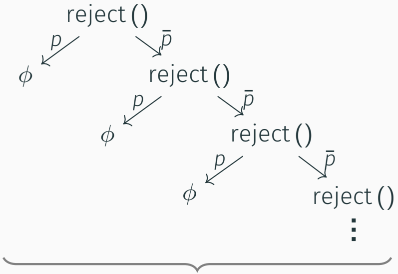
```
let direct _ =
    rand M
```

```
let reject _ =
  (rec sampler _ =
    let x = rand N in
    if x ≤ M then x else sampler ()) ()
```

Let $q = \frac{1}{M+1}$, $p = \frac{N-M}{N+1}$, and $\bar{p} = 1 - p$.



$\Pr[\phi] = 1$ where $\phi = \lambda k.0 \leq k \leq M$



$\Pr[\neg\phi] = \bar{p}^n$ after n rec. calls, goes to 0

$\Rightarrow \Pr[\phi]$ goes to 1 as n increases.

Higher Order Separation Logic and Probabilities

- Many success stories for probabilistic semantics & logics, in particular relational reasoning via *couplings*
- Higher-order functions and HO state still hard
- Iris: modular via HO separation logic (resource algebras, invariants, …)
- Clutch/Approxis: modular, local reasoning for randomisation
- All formalized in Coq on top of Iris

Program Semantics

The RandML language

A ML-like language with higher-order (recursive) functions, higher-order state, impredicative polymorphism, ..., and probabilistic uniform sampling.

$$v \in Val ::= z \in \mathbb{Z} \mid b \in \mathbb{B} \mid () \mid \ell \in Loc \mid \mathsf{rec}\, \mathsf{f}\, \mathsf{x} = e \mid \ \ldots$$

$$e \in Expr ::= v \mid \ \mathsf{ref}(e) \mid \ !\, e \mid e_1 \leftarrow e_2 \mid \ \ldots \ \mid \mathsf{rand}(e)$$

The RandML language

A ML-like language with higher-order (recursive) functions, higher-order state, impredicative polymorphism, ..., and probabilistic uniform sampling.

$$v \in Val ::= z \in \mathbb{Z} \mid b \in \mathbb{B} \mid () \mid \ell \in Loc \mid \mathsf{rec}\, \mathsf{f}\, \mathsf{x} = e \mid \ \ldots$$

$$e \in Expr ::= v \mid \ \mathsf{ref}(e) \mid \ ! \, e \mid e_1 \leftarrow e_2 \mid \ \ldots \ \mid \mathsf{rand}(e)$$

$$h \in Heap \triangleq Loc \xrightarrow{\text{fin}} Val$$

$$\rho \in Cfg \triangleq Expr \times Heap$$

The RandML language

A ML-like language with higher-order (recursive) functions, higher-order state, impredicative polymorphism, ..., and probabilistic uniform sampling.

$$v \in \mathit{Val} ::= z \in \mathbb{Z} \mid b \in \mathbb{B} \mid () \mid \ell \in \mathit{Loc} \mid \mathsf{rec}\, \mathsf{f}\, \mathsf{x} = e \mid \ldots$$

$$e \in \mathit{Expr} ::= v \mid \mathsf{ref}(e) \mid\ !\, e \mid e_1 \leftarrow e_2 \mid \ldots \mid \mathsf{rand}(e)$$

$$h \in \mathit{Heap} \triangleq \mathit{Loc} \xrightarrow{\text{fin}} \mathit{Val}$$

$$\rho \in \mathit{Cfg} \triangleq \mathit{Expr} \times \mathit{Heap}$$

$$\tau \in \mathit{Type} ::= \alpha \mid \mathsf{unit} \mid \mathsf{bool} \mid \mathsf{int} \mid \tau \times \tau \mid \tau + \tau \mid \tau \to \tau \mid \forall \alpha.\, \tau \mid \exists \alpha.\, \tau \mid \mu\, \alpha.\, \tau \mid \mathsf{ref}\, \tau$$

and a standard typing judgment $\Gamma \vdash e : \tau$.

Probabilities

A (discrete) *sub-distribution* $\mu \in \mathcal{D}(A)$ over a countable set A is a function $\mu : A \to [0,1]$ such that $\sum_{a \in A} \mu(a) \leq 1$.

Probabilities

A (discrete) *sub-distribution* $\mu \in \mathcal{D}(A)$ over a countable set A is a function $\mu : A \to [0,1]$ such that $\sum_{a \in A} \mu(a) \leq 1$.

Let $\mu \in \mathcal{D}(A)$, $a \in A$, and $f : A \to \mathcal{D}(B)$. The *distribution monad* is given by

1. $\textbf{bind}(f, \mu)(b) \;\triangleq\; \sum_{a \in A} \mu(a) \cdot f(a)(b)$
2. $\textbf{ret}(a)(a') \;\triangleq\; 1 \;\text{ if }\; a = a', \;\; 0 \;\text{ otherwise.}$

Probabilistic computations compose!

Operational Semantics

A program e with heap h evaluates to a distribution on values: $\mathsf{exec}(e, h) \in \mathcal{D}(Val)$.

exec is defined by iterating $\mathsf{step} : Cfg \to \mathcal{D}(Cfg)$ via bind.

Write $(e, h) \to^p (e', h')$ if $\mathsf{step}(e, h)(e', h') = p$.

$$(\lambda x.\, e_1)\; e_2 \to^1 e_1[e_2/x]$$

$$\vdots$$

$$\mathsf{rand}(N) \to^{1/(N+1)} k \qquad\qquad \forall k \in \{0, 1, \ldots, N\}$$

Semantics examples

- exec $\text{flip} = \{\text{true} : 0.5, \text{false} : 0.5\}$

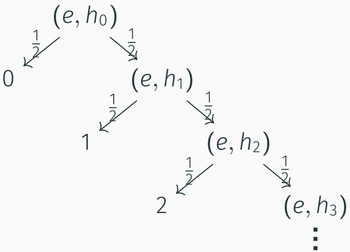
Semantics examples

- $\text{exec}\,\text{flip} = \{\text{true} : 0.5, \text{false} : 0.5\}$
- $\text{exec}(\text{not flip}) = \{\text{false} : 0.5, \text{true} : 0.5\}$

Semantics examples

- exec flip $= \{\text{true} : 0.5, \text{false} : 0.5\}$
- exec(not flip) $= \{\text{false} : 0.5, \text{true} : 0.5\}$
- Let ℓ be a location and write h_n for the heap $[\ell \mapsto n]$.

  Define $e \triangleq (\text{rec}\, f\, \_ = \text{if flip then } !\,\ell \text{ else } (\ell \leftarrow !\,\ell + 1; f()))\,()$.

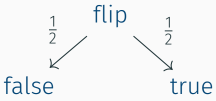


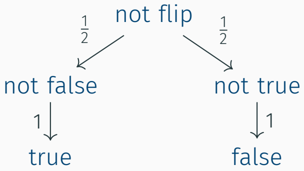
$$\text{exec}(e, h_0) \;=\; \{0 : 1/2,\; 1 : 1/4,\; 2 : 1/8,\; \ldots\}$$

Specifications & Couplings

Lifting Relations via Couplings

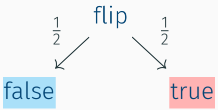
- Reasoning about equality of distributions directly is hard.
- “Coupling” proof technique: synchronize randomness

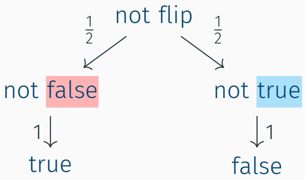




Lifting Relations via Couplings

- Reasoning about equality of distributions directly is hard.
- “Coupling” proof technique: synchronize randomness





Coupling-based Program Logics I

- Reasoning about operational semantics is hard, even without distributions.

Coupling-based Program Logics I

- Reasoning about operational semantics is hard, even without distributions.
- Build a program logic to construct couplings!

$$\mathrm{rwp}\ \ e \precsim e'\ \ \{\phi\}$$

Meaning: can align randomness s.t. ϕ holds.

Coupling-based Program Logics I

- Reasoning about operational semantics is hard, even without distributions.
- Build a program logic to construct couplings!

$$\mathrm{rwp}\ \ e \precsim e'\ \ \{\phi\}$$

  Meaning: can align randomness s.t. ϕ holds.
- (Lifted) Relational postconditions on *values* (not distributions).

Coupling-based Program Logics I

- Reasoning about operational semantics is hard, even without distributions.
- Build a program logic to construct couplings!

$$\text{rwp } e \precsim e' \; \{\phi\}$$

  Meaning: can align randomness s.t. ϕ holds.
- (Lifted) Relational postconditions on *values* (not distributions).
- Couplings compose:

$$\frac{\text{rwp } e_1 \precsim e_1' \; \{\psi\} \qquad \forall v, v'. \psi(v, v') \twoheadrightarrow \text{rwp } e_2[v/x] \precsim e_2'[v'/x] \; \{\phi\}}{\text{rwp } \textbf{let}\, x = e_1 \,\textbf{in}\, e_2 \precsim \textbf{let}\, x = e_1' \,\textbf{in}\, e_2' \; \{\phi\}}$$

Coupling-based Program Logics I

- Reasoning about operational semantics is hard, even without distributions.
- Build a program logic to construct couplings!

$$\text{rwp}\ \ e \precsim e'\ \ \{\phi\}$$

  Meaning: can align randomness s.t. ϕ holds.
- (Lifted) Relational postconditions on *values* (not distributions).
- Couplings compose:

$$\frac{\text{rwp}\ \ e_1 \precsim e'_1\ \ \{\psi\} \qquad \forall v, v'. \psi(v, v') \mathrel{-\!\!*} \text{rwp}\ \ e_2[v/x] \precsim e'_2[v'/x]\ \ \{\phi\}}{\text{rwp}\ \ \mathsf{let}\, x = e_1\, \mathsf{in}\, e_2 \precsim \mathsf{let}\, x = e'_1\, \mathsf{in}\, e'_2\ \ \{\phi\}}$$

- Postcondition ϕ can be any separation logic predicate. Today, we mostly use equality ("*eq*").
- rwp $e \precsim e'$ $\{\phi\}$ is defined as *refinement*. Today, think bi-refinement (equivalence).

Coupling-based Program Logics II

- Expose probabilistic reasoning only via coupling rule for "alignment":

$$\frac{f \text{ bijection} \qquad \forall 0 \leq n \leq N, \text{rwp}\ \ n \precsim f\,n\ \ \{\phi\}}{\text{rwp}\ \ \text{rand}\, N \precsim \text{rand}\, N\ \ \{\phi\}}$$

Coupling-based Program Logics II

- Expose probabilistic reasoning only via coupling rule for "alignment":

$$\frac{f \text{ bijection} \qquad \forall 0 \le n \le N, \text{rwp } n \precsim f\,n\ \{\phi\}}{\text{rwp } \text{rand}\,N \precsim \text{rand}\,N\ \{\phi\}}$$

- Standard, familiar rules for state etc. remain valid! For example:

$$\frac{\ell \mapsto v \qquad \ell \mapsto v \mathrel{-\!\!*} \text{rwp } v \precsim e\ \{\phi\}}{\text{rwp } !\ell \precsim e\ \{\phi\}}\ \text{RWP-LOAD-L}$$

Coupling-based Program Logics II

- Expose probabilistic reasoning only via coupling rule for "alignment":

$$\frac{f \text{ bijection} \qquad \forall 0 \leq n \leq N, \text{rwp } n \precsim f\, n \ \{\phi\}}{\text{rwp } \ \text{rand}\, N \precsim \text{rand}\, N \ \{\phi\}}$$

- Standard, familiar rules for state etc. remain valid! For example:

$$\frac{\ell \mapsto v \qquad \ell \mapsto v \mathrel{-\!\!*} \text{rwp } v \precsim e \ \{\phi\}}{\text{rwp } \ !\ell \precsim e \ \{\phi\}} \ \text{RWP-LOAD-L}$$

But also Löb induction, impredicative invariants, logical relations, ...

Coupling-based Program Logics II

- Expose probabilistic reasoning only via coupling rule for "alignment":

$$\frac{f \text{ bijection} \qquad \forall 0 \leq n \leq N, \text{rwp } n \precsim f\,n\ \{\phi\}}{\text{rwp } \textsf{rand}\, N \precsim \textsf{rand}\, N\ \{\phi\}}$$

- Standard, familiar rules for state etc. remain valid! For example:

$$\frac{\ell \mapsto v \qquad \ell \mapsto v \twoheadrightarrow \text{rwp } v \precsim e\ \{\phi\}}{\text{rwp } !\ell \precsim e\ \{\phi\}}\ \text{RWP-LOAD-L}$$

  But also Löb induction, impredicative invariants, logical relations, ...

- Soundness theorem:

  If $\text{rwp } e \precsim e'\ \{eq\}$ then $\text{exec}(e, h)(v) \leq \text{exec}(e', h')(v)$ for all h, h', and v.

In Action

$$\dfrac{\neg : \mathbb{B} \to \mathbb{B} \text{ bij.} \qquad \dfrac{\overline{\text{rwp } \ \text{true} \precsim \text{not}\,(\neg\text{true}) \ \{eq\}} \qquad \overline{\text{rwp } \ \text{false} \precsim \text{not}\,(\neg\text{false}) \ \{eq\}}}{\forall b.\ \text{rwp } \ b \precsim \text{not}\ (\neg b) \ \{eq\}}}{\text{rwp } \ \text{flip} \precsim \text{not}\,\text{flip} \ \ \{eq\}}$$

Fancy Alignment

Aligning Randomness at Different Points

Two one-time samplers:

$$eager \triangleq \text{let } b = \text{flip in } \lambda\,\_.\; b$$

$$
\begin{aligned}
lazy \triangleq\; & \text{let } r = \text{ref None in} \\
& \lambda\,\_.\; \text{match } !r \text{ with} \\
& \quad\;\; \text{Some } b \Rightarrow b \\
& \quad\;\; |\; \text{None} \;\;\; \Rightarrow \text{let } b = \text{flip in} \\
& \qquad\qquad\qquad\quad r \leftarrow \text{Some } b; \\
& \qquad\qquad\qquad\quad b \\
& \quad\; \text{end}
\end{aligned}
$$

Aligning Randomness at Different Points

Two one-time samplers:

$$eager \triangleq \text{let } b = \text{flip in } \lambda\,\_.\; b$$

$$\begin{aligned} lazy \triangleq\ & \text{let } r = \text{ref None in} \\ & \lambda\,\_.\ \text{match } !r \text{ with} \\ & \quad\ \text{Some } b \Rightarrow b \\ & \quad \mid \text{None} \Rightarrow \text{let } b = \text{flip in} \\ & \qquad\qquad\quad r \leftarrow \text{Some } b; \\ & \qquad\qquad\quad b \\ & \quad \text{end} \end{aligned}$$

We expect

$$\text{rwp } C[lazy] \precsim C[eager]\ \{eq\}$$

Equivalence should hold for any (well-typed) context C evaluating to a boolean.

Note: Not the same distribution on values, but same observations!

Aligning Randomness at Different Points

Two one-time samplers:

$$eager \triangleq \text{let } b = \text{flip in } \lambda\,\_.\ b$$

$$\begin{aligned} lazy \triangleq\ & \text{let } r = \text{ref None in} \\ & \lambda\,\_.\ \text{match } !r \text{ with} \\ & \quad \text{Some } b \Rightarrow b \\ & \quad \mid \text{None} \Rightarrow \text{let } b = \text{flip in} \\ & \qquad\qquad r \leftarrow \text{Some } b; \\ & \qquad\qquad b \\ & \text{end} \end{aligned}$$

We expect

$$\text{rwp } C[lazy] \precsim C[eager]\ \{eq\}$$

$$\frac{f \text{ bijection} \qquad \forall 0 \leq n \leq N.\ \text{rwp } n \precsim f\ n\ \{\phi\}}{\text{rwp rand } N \precsim \text{rand } N\ \{\phi\}}$$

Aligning Randomness at Different Points

Two one-time samplers:

$$eager \triangleq \text{let } b = \text{flip in } \lambda\_.\ b$$

$$\begin{aligned} lazy \triangleq\ & \text{let } r = \text{ref None in} \\ & \lambda\_.\ \text{match } !r \text{ with} \\ & \quad \text{Some } b \Rightarrow b \\ & \quad \mid \text{None} \Rightarrow \text{let } b = \text{flip in} \\ & \qquad\qquad r \leftarrow \text{Some } b; \\ & \qquad\qquad b \\ & \text{end} \end{aligned}$$

We expect

$$\text{rwp } C[lazy] \precsim C[eager]\ \{eq\}$$

$$\frac{f \text{ bijection} \quad \forall 0 \le n \le N\,.\, \text{rwp } n \precsim f\ n\ \{\phi\}}{\text{rwp rand } N \precsim \text{rand } N\ \{\phi\}}$$

Does not apply: only allows coupling the *next* rand in both programs.

Aligning Randomness at Different Points

Two one-time samplers:

$$eager \triangleq \text{let } b = \text{flip in } \lambda\_.\ b$$

$$\begin{aligned} lazy \triangleq\ & \text{let } r = \text{ref None in} \\ & \lambda\_.\ \text{match } !r \text{ with} \\ & \quad \text{Some } b \Rightarrow b \\ & \quad \mid \text{None} \Rightarrow \text{let } b = \text{flip in} \\ & \qquad\qquad r \leftarrow \text{Some } b; \\ & \qquad\qquad b \\ & \text{end} \end{aligned}$$

We expect

$$\text{rwp}\ \ C[lazy] \precsim C[eager]\ \{eq\}$$

$$\frac{f \text{ bijection} \qquad \forall 0 \leq n \leq N.\, \text{rwp}\ \ n \precsim f\ n\ \{\phi\}}{\text{rwp}\ \ \text{rand}\ N \precsim \text{rand}\ N\ \{\phi\}}$$

Does not apply: only allows coupling the *next* rand in both programs.

Q: Why bother? A: Simplified example from ElGamal encryption scheme.

- Goal: $\forall C : (unit \to bool) \Rightarrow bool, \text{rwp}\ \ C[lazy] \precsim C[eager]\ \ \{eq\}$

Aligning Asynchronous Samplings

- Goal: $\forall C : (\mathit{unit} \to \mathit{bool}) \Rightarrow \mathit{bool}, \mathrm{rwp}\;\; C[\mathit{lazy}] \precsim C[\mathit{eager}]\;\; \{\mathit{eq}\}$
- Limitation: No “scoped” / local reasoning for randomness.

Aligning Asynchronous Samplings

- Goal: $\forall C : (unit \to bool) \Rightarrow bool, \text{rwp}\ \ C[lazy] \precsim C[eager]\ \ \{eq\}$
- Limitation: No "scoped" / local reasoning for randomness.
- Idea:
  - "Presampling tapes" de-couple construction of coupling from operational semantics by introducing a resource for "logical randomness".
  - "Tape allocation" confers *ownership* of a fresh (logical) source of randomness.

Syntax

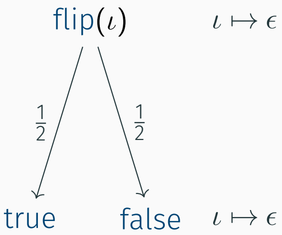
Modify RandML as follows

$$
\begin{array}{lccl}
 & & \vdots & \\
\mathit{Val} & v & ::= & \ldots \mid \iota \in \mathit{Label} \\
\mathit{Expr} & e & ::= & \ldots \mid \mathsf{rand}(e_1, e_2) \mid \mathsf{tape}\, e \\
\mathit{TapeMap} & & = & \mathit{Label} \xrightarrow{\text{fin}} \mathit{Tape} \\
\mathit{State} & \sigma & \in & \mathit{Heap} \times \mathit{TapeMap} \\
\mathit{Cfg} & \rho & ::= & (\sigma, e) \\
 & & & \\
\mathit{Type} & \tau & ::= & \ldots \mid \mathsf{tape}
\end{array}
$$

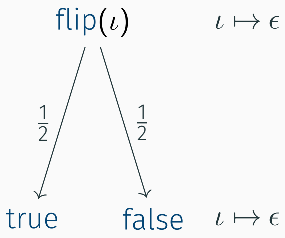
Presampling Tapes I

$\mathsf{flip}(\iota) \qquad \iota \mapsto \epsilon$

Presampling Tapes I

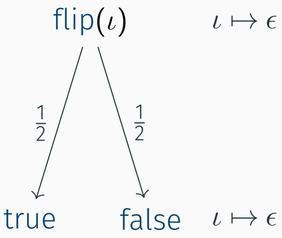


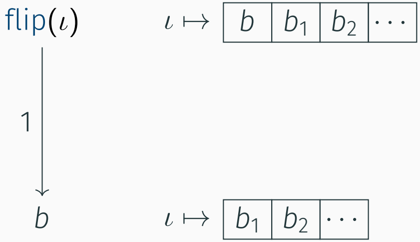
Presampling Tapes I



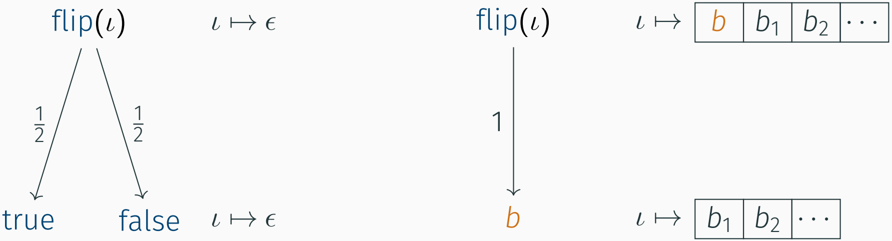
$$\text{flip}(\iota) \qquad \iota \mapsto \boxed{b} \boxed{b_1} \boxed{b_2} \boxed{\cdots}$$

Presampling Tapes I





Presampling Tapes I



Presampling Tapes II

… but operationally, no language primitives add values to the tapes!

$$\iota : \mathsf{tape} \vdash \mathsf{rwp}\ \ \mathsf{flip} \precsim \mathsf{flip}(\iota)\ \ \{eq\}$$

Presampling Tapes II

… but operationally, no language primitives add values to the tapes!

$$\iota : \mathsf{tape} \vdash \mathsf{rwp}\ \ \mathsf{flip} \precsim \mathsf{flip}(\iota)\ \ \{eq\}$$

Instead, tapes will be populated with fresh samples via a logical operation.

Presampling Tapes II

… but operationally, no language primitives add values to the tapes!

$$\iota : \mathsf{tape} \vdash \mathsf{rwp}\ \ \mathsf{flip} \precsim \mathsf{flip}(\iota)\ \ \{eq\}$$

Instead, tapes will be populated with fresh samples via a logical operation.

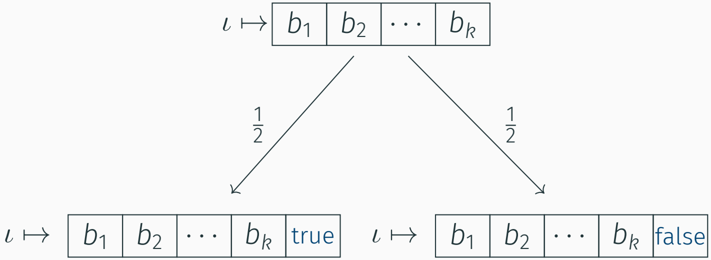
$$\iota \mapsto \boxed{b_1}\boxed{b_2}\boxed{\cdots}\boxed{b_k}$$

Presampling Tapes II

… but operationally, no language primitives add values to the tapes!

$$\iota : \text{tape} \vdash \text{rwp}\ \ \text{flip} \precsim \text{flip}(\iota)\ \ \{eq\}$$

Instead, tapes will be populated with fresh samples via a logical operation.



Presampling Tapes III

Logically, we introduce a separation logic resource

$$\iota \hookrightarrow \vec{b}$$

that denotes ownership of a tape ι and its contents $\vec{b}$.

Presampling Tapes III

Logically, we introduce a separation logic resource

$$\iota \hookrightarrow \vec{b}$$

that denotes ownership of a tape ι and its contents $\vec{b}$.

$$\frac{\forall \iota.\, \iota \hookrightarrow \epsilon \twoheadrightarrow \mathsf{rwp}\ \iota \precsim e\ \{\tau\}}{\mathsf{rwp}\ \mathsf{tape} \precsim e\ \{\tau\}}$$

Presampling Tapes III

Logically, we introduce a separation logic resource

$$\iota \hookrightarrow \vec{b}$$

that denotes ownership of a tape ι and its contents $\vec{b}$.

$$\frac{\forall \iota.\, \iota \hookrightarrow \epsilon \twoheadrightarrow \mathsf{rwp}\ \iota \precsim e\ \{\tau\}}{\mathsf{rwp}\ \boxed{\mathsf{tape}} \precsim e\ \{\tau\}}$$

Presampling Tapes III

Logically, we introduce a separation logic resource

$$\iota \hookrightarrow \vec{b}$$

that denotes ownership of a tape ι and its contents $\vec{b}$.

$$\frac{\boxed{\forall \iota.\, \iota \hookrightarrow \epsilon} \mathrel{-\!\!*} \text{rwp}\ \iota \precsim e\ \{\tau\}}{\text{rwp}\ \text{tape} \precsim e\ \{\tau\}}$$

Presampling Tapes III

Logically, we introduce a separation logic resource

$$\iota \hookrightarrow \vec{b}$$

that denotes ownership of a tape ι and its contents $\vec{b}$.

$$\frac{\forall \iota.\, \iota \hookrightarrow \epsilon \twoheadrightarrow \text{rwp}\ \ \iota \precsim e\ \{\tau\}}{\text{rwp}\ \ \text{tape}\ \precsim e\ \{\tau\}} \qquad \frac{\iota \hookrightarrow b \cdot \vec{b} \qquad \iota \hookrightarrow \vec{b} \twoheadrightarrow \text{rwp}\ \ b \precsim e_2\ \{\tau\}}{\text{rwp}\ \ \text{flip}(\iota) \precsim e_2\ \{\tau\}}$$

Presampling Tapes III

Logically, we introduce a separation logic resource

$$\iota \hookrightarrow \vec{b}$$

that denotes ownership of a tape ι and its contents $\vec{b}$.

$$\frac{\forall \iota.\, \iota \hookrightarrow \epsilon \twoheadrightarrow \text{rwp}\ \ \iota \precsim e\ \ \{\tau\}}{\text{rwp}\ \ \text{tape}\ \ \precsim e\ \ \{\tau\}} \qquad \frac{\iota \hookrightarrow b \cdot \vec{b} \qquad \iota \hookrightarrow \vec{b} \twoheadrightarrow \text{rwp}\ \ b \precsim e_2\ \ \{\tau\}}{\text{rwp}\ \ \text{flip}(\iota) \precsim e_2\ \ \{\tau\}}$$

Presampling Tapes III

Logically, we introduce a separation logic resource

$$\iota \hookrightarrow \vec{b}$$

that denotes ownership of a tape ι and its contents $\vec{b}$.

$$\frac{\forall \iota.\, \iota \hookrightarrow \epsilon \twoheadrightarrow \text{rwp } \iota \precsim e\ \{\tau\}}{\text{rwp } \textsf{tape} \precsim e\ \{\tau\}} \qquad \frac{\iota \hookrightarrow b \cdot \vec{b} \qquad \iota \hookrightarrow \vec{b} \twoheadrightarrow \text{rwp } b \precsim e_2\ \{\tau\}}{\text{rwp } \textsf{flip}(\iota) \precsim e_2\ \{\tau\}}$$

Presampling Tapes III

Logically, we introduce a separation logic resource

$$\iota \hookrightarrow \vec{b}$$

that denotes ownership of a tape ι and its contents $\vec{b}$.

$$\frac{\forall \iota.\, \iota \hookrightarrow \epsilon \twoheadrightarrow \text{rwp } \iota \precsim e \ \{\tau\}}{\text{rwp } \texttt{tape} \precsim e \ \{\tau\}} \qquad \frac{\iota \hookrightarrow b \cdot \vec{b} \qquad \boxed{\iota \hookrightarrow \vec{b}} \twoheadrightarrow \text{rwp } b \precsim e_2 \ \{\tau\}}{\text{rwp } \texttt{flip}(\iota) \precsim e_2 \ \{\tau\}}$$

Logically, we introduce a separation logic resource

$$\iota \hookrightarrow \vec{b}$$

that denotes ownership of a tape ι and its contents $\vec{b}$.

$$\frac{\forall \iota.\ \iota \hookrightarrow \epsilon \twoheadrightarrow \text{rwp}\ \ \iota\ \precsim\ e\ \{\tau\}}{\text{rwp}\ \ \texttt{tape}\ \precsim\ e\ \{\tau\}}
\qquad
\frac{\iota \hookrightarrow b \cdot \vec{b} \qquad \iota \hookrightarrow \vec{b} \twoheadrightarrow \text{rwp}\ \boxed{b} \precsim\ e_2\ \{\tau\}}{\text{rwp}\ \ \texttt{flip}(\iota)\ \precsim\ e_2\ \{\tau\}}$$

Presampling Tapes III

Logically, we introduce a separation logic resource

$$\iota \hookrightarrow \vec{b}$$

that denotes ownership of a tape ι and its contents $\vec{b}$.

$$\frac{\forall \iota.\, \iota \hookrightarrow \epsilon \mathrel{-\!\!*} \text{rwp} \;\; \iota \precsim e \;\; \{\tau\}}{\text{rwp} \;\; \text{tape} \;\; \precsim e \;\; \{\tau\}} \qquad \frac{\iota \hookrightarrow b \cdot \vec{b} \qquad \iota \hookrightarrow \vec{b} \mathrel{-\!\!*} \text{rwp} \;\; b \precsim e_2 \;\; \{\tau\}}{\text{rwp} \;\; \text{flip}(\iota) \precsim e_2 \;\; \{\tau\}}$$

It—locally—turns reasoning about probabilistic choice into reasoning about state.

Asynchronous Couplings

With presampling tapes, we can synchronously couple tape samplings with program samplings

$$\frac{f \text{ bijection} \qquad \iota \hookrightarrow \vec{b} \qquad \forall b.\, \iota \hookrightarrow \vec{b} \cdot b \mathrel{-\!\!*} \text{rwp } \; e \precsim f(b) \;\; \{\phi\}}{\text{rwp } \; e \precsim \text{flip} \;\; \{\phi\}}$$

to couple program samplings asynchronously!

Asynchronous Couplings

With presampling tapes, we can synchronously couple tape samplings with program samplings

$$\frac{f \text{ bijection} \qquad \iota \hookrightarrow \vec{b} \qquad \forall b.\, \iota \hookrightarrow \vec{b} \cdot b \twoheadrightarrow \text{rwp } e \precsim f(b)\ \{\phi\}}{\text{rwp } e \precsim \text{flip } \{\phi\}}$$

to couple program samplings asynchronously!

Asynchronous Couplings

With presampling tapes, we can synchronously couple tape samplings with program samplings

$$\frac{f \text{ bijection} \qquad \boxed{\iota \hookrightarrow \vec{b}} \qquad \forall b.\, \iota \hookrightarrow \vec{b} \cdot b \twoheadrightarrow* \text{rwp } e \precsim f(b) \;\{\phi\}}{\text{rwp } e \precsim \text{flip } \{\phi\}}$$

to couple program samplings asynchronously!

With presampling tapes, we can synchronously couple tape samplings with program samplings

$$\frac{f \text{ bijection} \qquad \iota \hookrightarrow \vec{b} \qquad \forall b.\, \iota \hookrightarrow \vec{b} \cdot b \mathrel{-\!\!*} \text{rwp } e \precsim f(b)\ \{\phi\}}{\text{rwp } e \precsim \text{flip } \{\phi\}}$$

to couple program samplings asynchronously!

Lazy / Eager Coin with Tapes

$\text{let } b = \boxed{\text{flip}} \text{ in } \lambda\_.\ b$

$$
\begin{array}{l}
\text{let } r = \text{ref None in} \\
\boxed{\text{let } \iota = \text{tape}} \text{ in} \\
\lambda\_.\ \text{match } !\,r \text{ with} \\
\quad \text{Some } b \Rightarrow b \\
\quad \mid \text{None} \Rightarrow \text{let } b = \boxed{\text{flip } \iota} \text{ in} \\
\qquad\qquad\qquad r \leftarrow \text{Some } b; \\
\qquad\qquad\qquad b \\
\quad \text{end}
\end{array}
$$

Proof:

- asynchronously couple flip and tape ι
- invariant: $(\iota \hookrightarrow (1, b) * \ell \mapsto \text{None}) \vee \ell \mapsto \text{Some}(b)$
- case distinction on value of $!\,r$

Approximate Reasoning

Approximate Equivalence

Sampling with replacement:

$$\begin{aligned}&\text{let } x_0 = \text{rand } N \text{ in}\\&\text{let } x_1 = \text{rand } N \text{ in}\\&\text{let } x_2 = \text{rand } N \text{ in}\\&(x_0, x_1, x_2)\end{aligned}$$

without replacement:

$$\begin{aligned}&\text{let } x_0 = \text{rand } N \text{ in}\\&\text{let } x_1 = \text{rand } N \setminus \{x_0\} \text{ in}\\&\text{let } x_2 = \text{rand } N \setminus \{x_0, x_1\} \text{ in}\\&(x_0, x_1, x_2)\end{aligned}$$

Approximate Equivalence

Sampling with replacement:

$$\begin{aligned}&\text{let } x_0 = \text{rand } N \text{ in}\\&\text{let } x_1 = \text{rand } N \text{ in}\\&\text{let } x_2 = \text{rand } N \text{ in}\\&(x_0, x_1, x_2)\end{aligned}$$

without replacement:

$$\begin{aligned}&\text{let } x_0 = \text{rand } N \text{ in}\\&\text{let } x_1 = \text{rand } N \setminus \{x_0\} \text{ in}\\&\text{let } x_2 = \text{rand } N \setminus \{x_0, x_1\} \text{ in}\\&(x_0, x_1, x_2)\end{aligned}$$

- We want to align distributions that aren't equal.

Approximate Equivalence

Sampling with replacement:

$$
\begin{aligned}
& && \mathsf{let}\ x_0 = \mathsf{rand}\ N\ \mathsf{in} \\
& ↯\left(\tfrac{1}{N+1}\right) && \mathsf{let}\ x_1 = \mathsf{rand}\ N\ \mathsf{in} \\
& ↯\left(\tfrac{2}{N+1}\right) && \mathsf{let}\ x_2 = \mathsf{rand}\ N\ \mathsf{in} \\
& && (x_0, x_1, x_2)
\end{aligned}
$$

without replacement:

$$
\begin{aligned}
& \mathsf{let}\ x_0 = \mathsf{rand}\ N\ \mathsf{in} \\
& \mathsf{let}\ x_1 = \mathsf{rand}\ N \setminus \{x_0\}\ \mathsf{in} \\
& \mathsf{let}\ x_2 = \mathsf{rand}\ N \setminus \{x_0, x_1\}\ \mathsf{in} \\
& (x_0, x_1, x_2)
\end{aligned}
$$

- We want to align distributions that aren't equal.
- "Error credits" logically bound the distance between aligned distributions.

Error Credits

- $\lightning(\varepsilon)$ asserts ownership of ε error credits, where $\varepsilon \in [0, 1]$.

Error Credits

- $\lightning(\varepsilon)$ asserts ownership of ε error credits, where $\varepsilon \in [0, 1]$.
- Error credits obey the following laws:

$$\vdash \lightning(0) \qquad \lightning(\varepsilon_1) * \lightning(\varepsilon_2) \dashv\vdash \lightning(\varepsilon_1 + \varepsilon_2) \qquad \lightning(1) \vdash \bot$$

Error Credits

- $↯(\varepsilon)$ asserts ownership of ε error credits, where $\varepsilon \in [0,1]$.
- Error credits obey the following laws:

$$\vdash ↯(0) \qquad ↯(\varepsilon_1) * ↯(\varepsilon_2) \dashv\vdash ↯(\varepsilon_1 + \varepsilon_2) \qquad ↯(1) \vdash \bot$$

- "Mismatched" samplings *consume* error:

$$\frac{↯\left(\frac{1}{N+2}\right) \qquad \forall n \leq N.\ \text{rwp}\ \ n \precsim n\ \ \{\Phi\}}{\text{rwp}\ \ \text{rand}\, N \precsim \text{rand}\ (N+1)\ \ \{\Phi\}}$$

Error Credits

- $\lightning(\varepsilon)$ asserts ownership of ε error credits, where $\varepsilon \in [0,1]$.
- Error credits obey the following laws:

$$\vdash \lightning(0) \qquad \lightning(\varepsilon_1) * \lightning(\varepsilon_2) \dashv\vdash \lightning(\varepsilon_1 + \varepsilon_2) \qquad \lightning(1) \vdash \bot$$

- "Mismatched" samplings *consume* error:

$$\frac{\lightning\left(\frac{1}{N+2}\right) \qquad \forall n \leq N.\ \mathrm{rwp}\ \ n \precsim n\ \ \{\Phi\}}{\mathrm{rwp}\ \ \mathsf{rand}\, N \precsim \mathsf{rand}\ (N+1)\ \ \{\Phi\}}$$

- More generally (also: variant for tapes) :

$$\frac{f : \mathbb{N}_{\leq N} \to \mathbb{N}_{\leq M}\ \text{injection} \qquad \lightning\left(\frac{M-N}{M+1}\right) \qquad N \leq M \qquad \forall n \leq N.\ \mathrm{rwp}\ \ n \precsim f(n)\ \ \{\Phi\}}{\mathrm{rwp}\ \ \mathsf{rand}\, N \precsim \mathsf{rand}\, M\ \ \{\Phi\}}$$

- Semantic model requires a different notion of *approximate* coupling.
- Compatible with all previous probabilistic and non-probabilistic features.
- Soundness theorem:

  If $\lightning(\varepsilon) \vdash \text{rwp}\ \ e \precsim e'\ \ \{eq\}$ then the distributions induced by executing e and e' are at distance at most ε.

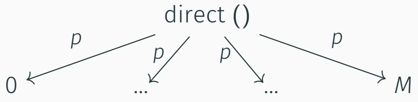
  NB: $\varepsilon = 0$ means equality, recovering the previous logic.

Application: Equivalence by Approximation

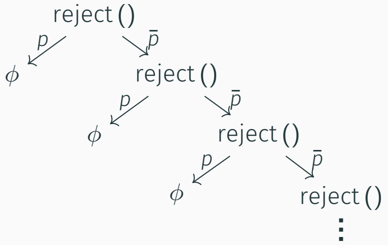
```
let direct _ =
  let ι_d = tape M in
  rand M ι_d
```

```
let reject _ =
  let ι_r = tape N in
  (rec sampler _ =
    let x = rand N ι_r in
    if x ≤ M then x else sampler ()) ()
```

Let $p = \frac{1}{M+1}$ and $\bar{p} = 1 - p$.



$\Pr[\phi] = 1$ where $\phi = \lambda k.0 \leq k \leq M$



$\Pr[\neg\phi] = \bar{p}^n$ after n rec. calls, goes to 0

Application: Security of a PRF-based Symmetric Encryption Scheme

$$\begin{array}{lll} \text{let enc key msg} & = & \text{let r} = \text{rand } N \text{ in} \\ & & \text{let pad} = \mathit{prf} \text{ key r in} \\ & & \text{let c} = \text{xor msg pad in} \\ & & (\text{r}, \text{c}) \end{array} \qquad \begin{array}{lll} \text{let keygen ()} & = & \text{rand } N \\ \text{let dec key (r, c)} & = & \text{let pad} = \mathit{prf} \text{ key r in} \\ & & \text{let msg} = \text{xor c pad in} \\ & & \text{msg} \end{array}$$

We prove, for all (well-typed) adversaries $\mathcal{A}$:

$$\lightning\left(\frac{Q^2}{2^{n+1}}\right) \twoheadrightarrow \text{ rwp } \mathcal{A}\,(\text{enc}\,(\text{keygen}\,())_{|Q}) \precsim \mathcal{A}\,(\text{rand\_cipher}_{|Q})\ \{eq\}$$

By soundness theorem:

$$\left|\Pr\left[\mathcal{A}(\text{enc}\,(\text{keygen}\,())_{|Q}) = 1\right] - \Pr\left[\mathcal{A}(\text{rand\_cipher}_{|Q}) = 1\right]\right| \leq \frac{Q^2}{2^{n+1}}$$

The Bigger Picture

Preprint available at `arxiv:2407.14107` (will be at POPL'25)
Code & other papers at `https://github.com/logsem/clutch`

- Clutch: refinement, asynchronous sampling via tapes
- Eris: unary, bound probability of "bad events" via error credits
- Caliper: termination-preserving refinement
- Tachis: expected cost (e.g., time or entropy) via cost credits
- Approxis: approximate refinement

- ongoing: concurrency, continuous distributions
- future: differential privacy, unifying framework, tail bounds, more security, fair schedulers, distributed systems...

Configuration reduction

$$
\begin{aligned}
(h, e) &\to^1 (h, e') && \text{if } e \overset{\text{pure}}{\rightsquigarrow} e' \\
(h, \mathsf{ref}(v)) &\to^1 (h[\ell \mapsto v], \ell) && \text{where } \ell = \mathsf{freshLoc}(\mathrm{dom}(h)) \\
(h, !\,\ell) &\to^1 (h, h(\ell)) && \text{if } \ell \in \mathrm{dom}(h) \\
(h, \ell \leftarrow v) &\to^1 (h[\ell \mapsto v], ()) && \text{if } \ell \in \mathrm{dom}(h) \\
(h, \mathsf{rand}\, N) &\to^p (h, k) && p = \frac{1}{N+1} \text{ and } 0 \le k \le N \\
(h, E[e]) &\to^p (h', E[e']) && \text{if } (h, e) \to^p (h', e')
\end{aligned}
$$

Definition (n-step execution)

$$\mathrm{execVal}_n(e,\sigma) \triangleq \begin{cases} \mathbf{0} & \text{if } e \notin \mathit{Val} \text{ and } n = 0 \\ \mathbf{ret}(e) & \text{if } e \in \mathit{Val} \\ \mathbf{step}(e,\sigma) \ggg= \mathrm{execVal}_{(n-1)} & \text{otherwise} \end{cases}$$

Operational semantics

Definition (n-step execution)

$$\text{execVal}_n(e,\sigma) \triangleq \begin{cases} 0 & \text{if } e \notin \mathit{Val} \text{ and } n = 0 \\ \text{ret}(e) & \text{if } e \in \mathit{Val} \\ \text{step}(e,\sigma) \gg\!= \text{execVal}_{(n-1)} & \text{otherwise} \end{cases}$$

We can take the limit since execVal is monotone and bounded.

$$\text{execVal}(\rho)(v) \triangleq \lim_{n\to\infty} \text{execVal}_n(\rho)(v)$$

A program thus induces a distribution on values.

Couplings

A *coupling* for distributions $\mu_1 : \mathcal{D}(A), \mu_2 : \mathcal{D}(B)$, is a distribution μ on $A \times B$ such that $\lambda a. \sum_{b \in B} \mu(a, b) = \mu_1$ and $\lambda b. \sum_{a \in A} \mu(a, b) = \mu_2$.

A coupling μ lifts a relation R if for all (a, b) s.t. $\mu(a, b) > 0$, $R(a, b)$ holds.

Definition (Approximate Coupling)
Let $\mu_1 \in \mathcal{D}(A)$ and $\mu_2 \in \mathcal{D}(B)$. Given some approximation error $\varepsilon \in [0, 1]$ and a relation $R \subseteq A \times B$, we say that there exists an (ε, R)-coupling of μ_1 and μ_2 if for all $[0, 1]$-valued random variables $X : A \to [0, 1]$ and $Y : B \to [0, 1]$, such that $(a, b) \in R$ implies $X(a) \leq Y(b)$, the expected value of X exceeds the expected value of Y by at most ε, *i.e.*, $\mathbb{E}_{\mu_1}[X] \leq \mathbb{E}_{\mu_2}[Y] + \varepsilon$. We write $\mu_1 \lesssim_\varepsilon \mu_2 : R$ if an (ε, R)-coupling exists between μ_1 and μ_2.

$$\frac{0 < \varepsilon \qquad 1 < k \qquad \forall \varepsilon'.(\lightning(k \cdot \varepsilon') \mathrel{-\!\!*} P) * \lightning(\varepsilon') \vdash P}{\lightning(\varepsilon) \vdash P}\ \textsc{err-amp}$$

$$\frac{\begin{array}{c} 0 < \varepsilon \qquad 1 < k \\ \forall \varepsilon'.(\lightning(k \cdot \varepsilon') \mathrel{-\!\!*} \mathsf{rwp}\ e \precsim e'\ \{\Phi\}) * \lightning(\varepsilon') \vdash \mathsf{rwp}\ e \precsim e'\ \{\Phi\} \end{array}}{\lightning(\varepsilon) \vdash \mathsf{rwp}\ e \precsim e'\ \{\Phi\}}\ \textsc{wp-err-amp}$$

Fragmented Couplings

WP-FRAGMENTED-R-EXP

$$\frac{f : \mathbb{N}_{\leq N} \to \mathbb{N}_{\leq M} \text{ injection} \qquad N < M \qquad ↯(\varepsilon) \qquad \iota \hookrightarrow (N, \vec{n}) \qquad \iota' \hookrightarrow_{s} (M, \vec{m}) \qquad \forall m \leq M.\, \iota' \hookrightarrow_{s} (M, \vec{m} \cdot m) * \left(\text{if } m \in \text{img}(f) \begin{array}{l} \text{then } \iota \hookrightarrow (N, \vec{n} \cdot f^{-1}(m)) \\ \text{else } \iota \hookrightarrow (N, \vec{n}) * ↯\left(\frac{M+1}{M-N} \cdot \varepsilon\right) \end{array} \right) \twoheadrightarrow \text{rwp } e_1 \precsim e_2 \; \{\Phi\}}{\text{rwp } e_1 \precsim e_2 \; \{\Phi\}}$$

Bounded Oracles

$$
\begin{aligned}
&\text{let } \mathrm{q\_calls}\ (Q : \mathrm{int})\ (f : \alpha \rightarrow \beta) : \alpha \rightarrow \beta\ \mathrm{option} = \\
&\quad \text{let } \mathrm{counter} = \mathrm{ref}\ 0\ \mathrm{in} \\
&\quad \lambda x.\ \mathrm{if}\ (!\,\mathrm{counter} < Q)\ \ \mathrm{then}\ \ \mathrm{incr\ counter}\,;\ \mathrm{Some}\ (f\ x)\ \ \mathrm{else}\ \ \mathrm{None}
\end{aligned}
$$