ASYNCHRONOUS PROBABILISTIC COUPLINGS

in Higher-Order Separation Logic

Simon Oddershede Gregersen[1] Alejandro Aguirre[1] Philipp G. Haselwarter[1]
Joseph Tassarotti[2] Lars Birkedal[1]

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

AARHUS UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

Motivating example

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

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

pRHL approach

The usual coupling rules known from pRHL, e.g.,

pRHL-couple

$$\frac{}{\{P[v/x_1, v/x_2]\}\; x_1 \xleftarrow{\$} d \sim x_2 \xleftarrow{\$} d\; \{P\}}$$

require “synchronization” and thus do not suffice.

This work

Proving contextual equivalence of

… probabilistic programs written in an expressive programming language
… using a higher-order separation logic, called Clutch,
… and asynchronous probabilistic couplings

while mechanizing everything in the Coq proof assistant.

The $\mathbf{F}_{\mu,\mathsf{ref}}^{\mathsf{rand}}$ language

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

$$\begin{aligned} e \in \mathsf{Expr} &::= \;\ldots \mid \mathsf{rand}(e) \\ \tau \in \mathsf{Type} &::= \alpha \mid \mathsf{unit} \mid \mathsf{bool} \mid \mathsf{int} \mid \tau \times \tau \mid \tau + \tau \mid \tau \to \tau \mid \\ &\quad\; \forall \alpha.\, \tau \mid \exists \alpha.\, \tau \mid \mu\, \alpha.\, \tau \mid \mathsf{ref}\, \tau \end{aligned}$$

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

Operational semantics

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

$$\vdots$$

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

For this presentation we will just consider flip $\triangleq$ rand(1).

Contextual refinement

The property of interest is contextual refinement.

$$\Gamma \vdash e_1 \precsim_{\mathsf{ctx}} e_2 : \tau \quad \triangleq \quad \forall \tau', (\mathcal{C} : (\Gamma \vdash \tau) \Rightarrow (\emptyset \vdash \tau')), \sigma. \\ \mathrm{term}(\mathcal{C}[\, e_1 \,], \sigma) \leq \mathrm{term}(\mathcal{C}[\, e_2 \,], \sigma)$$

and $\Gamma \vdash e_1 \simeq_{\mathsf{ctx}} e_2 : \tau$ follows as refinement in both directions.

Proving contextual refinement

1. A probabilistic relational separation logic on top of Iris

2. A logical refinement judgment (a “logical” logical relation)

$$\Gamma \vDash e_1 \precsim e_2 : \tau$$

that implies contextual refinement.

Proving contextual refinement

1. A probabilistic relational separation logic on top of Iris

2. A logical refinement judgment (a “logical” logical relation)

$$\Gamma \vDash e_1 \precsim e_2 : \tau$$

   that implies contextual refinement.

Proving contextual refinement

1. A probabilistic relational separation logic on top of Iris

2. A logical refinement judgment (a "logical" logical relation)

$$\Gamma \vDash e_1 \precsim e_2 : \tau$$

that implies contextual refinement.

Refinement judgment

The judgment

$$\Gamma \vDash e_1 \precsim e_2 : \tau$$

should be read as "in env. Γ, expression e_1 refines expression e_2 at type τ".

Refinement judgment

The judgment

$$\Gamma \vDash e_1 \precsim e_2 : \tau$$

should be read as "in env. Γ, expression e_1 refines expression e_2 at type τ".

Theorem (Fundamental theorem)

If $\Gamma \vdash e : \tau$ then $\Gamma \vDash e \precsim e : \tau$.

Refinement judgment

The judgment

$$\Gamma \vDash e_1 \precsim e_2 : \tau$$

should be read as "in env. Γ, expression e_1 refines expression e_2 at type τ".

Theorem (Fundamental theorem)

If $\Gamma \vdash e : \tau$ then $\Gamma \vDash e \precsim e : \tau$.

Theorem (Soundness)

If $\Gamma \vDash e_1 \precsim e_2 : \tau$ then $\Gamma \vdash e_1 \precsim_{\mathsf{ctx}} e_2 : \tau$.

$$\frac{e_1 \overset{\text{pure}}{\rightsquigarrow} e_1' \qquad \Gamma \vDash K[\, e_1' \,] \precsim e_2 : \tau}{\Gamma \vDash K[\, e_1 \,] \precsim e_2 : \tau}$$

$$\frac{e_1 \overset{\text{pure}}{\rightsquigarrow} e_1' \qquad \Gamma \vDash K[\, e_1' \,] \precsim e_2 : \tau}{\Gamma \vDash K[\, e_1 \,] \precsim e_2 : \tau}$$

$$\frac{\ell \mapsto v \qquad \ell \mapsto v \twoheadrightarrow \Gamma \vDash K[\, v \,] \precsim e_2 : \tau}{\Gamma \vDash K[\, !\ell \,] \precsim e_2 : \tau}$$

$$\frac{e_1 \stackrel{\text{pure}}{\rightsquigarrow} e_1' \qquad \Gamma \vDash K[\, e_1' \,] \precsim e_2 : \tau}{\Gamma \vDash K[\, e_1 \,] \precsim e_2 : \tau}$$

$$\frac{\ell \mapsto v \qquad \ell \mapsto v \twoheadrightarrow \Gamma \vDash K[\, v \,] \precsim e_2 : \tau}{\Gamma \vDash K[\, !\ell \,] \precsim e_2 : \tau}$$

$$\frac{\forall b.\, \Gamma \vDash K[\, b \,] \precsim e_2 : \tau}{\Gamma \vDash K[\, \textsf{flip} \,] \precsim e_2 : \tau}$$

$$\frac{e_1 \overset{\text{pure}}{\rightsquigarrow} e_1' \qquad \Gamma \vDash K[\, e_1' \,] \precsim e_2 : \tau}{\Gamma \vDash K[\, e_1 \,] \precsim e_2 : \tau}$$

$$\frac{\ell \mapsto v \qquad \ell \mapsto v \mathrel{-\!\!*} \Gamma \vDash K[\, v \,] \precsim e_2 : \tau}{\Gamma \vDash K[\, !\ell \,] \precsim e_2 : \tau}$$

$$\frac{\forall b.\, \Gamma \vDash K[\, b \,] \precsim e_2 : \tau}{\Gamma \vDash K[\ \mathsf{flip}\ ] \precsim e_2 : \tau}$$

$$\frac{f \text{ bijection} \qquad \forall b.\, \Gamma \vDash K[\, b \,] \precsim K'[\, f(b) \,] : \tau}{\Gamma \vDash K[\ \mathsf{flip}\ ] \precsim K'[\ \mathsf{flip}\ ] : \tau}$$

Clutch

Clutch is built on top of the (Boolean-valued!) Iris separation logic

$$
\begin{aligned}
P, Q \in \text{iProp} ::= {} & \text{True} \mid \text{False} \mid P \wedge Q \mid P \vee Q \mid P \Rightarrow Q \mid && \text{(propositional)} \\
& \forall x.\, P \mid \exists x.\, P \mid && \text{(higher-order)} \\
& P * Q \mid P \mathrel{-\!\!*} Q \mid \ell \mapsto v \mid && \text{(separation)} \\
& \square\, P \mid \triangleright P \mid \boxed{a} \mid \boxed{P} \mid \ldots \mid && \text{(Iris)} \\
& \text{wp}\; e\; \{\Phi\} \mid \text{spec}(e) \mid \ldots && \text{(Clutch)}
\end{aligned}
$$

from which we derive $\Gamma \vDash e_1 \precsim e_2 : \tau$.

Clutch

Clutch is built on top of the (Boolean-valued!) Iris separation logic

$$
\begin{aligned}
P, Q \in \text{iProp} ::= {} & \text{True} \mid \text{False} \mid P \wedge Q \mid P \vee Q \mid P \Rightarrow Q \mid && \text{(propositional)} \\
& \forall x.\, P \mid \exists x.\, P \mid && \text{(higher-order)} \\
& P * Q \mid P \mathrel{-\!\!*} Q \mid \ell \mapsto v \mid && \text{(separation)} \\
& \square\, P \mid \triangleright P \mid \boxed{a} \mid \boxed{P} \mid \ldots \mid && \text{(Iris)} \\
& \text{wp}\ e\ \{\Phi\} \mid \text{spec}(e) \mid \ldots && \text{(Clutch)}
\end{aligned}
$$

from which we derive $\Gamma \vDash e_1 \precsim e_2 : \tau$.

The connectives $\mathsf{wp}\ e\ \{\Phi\}$ and $\mathsf{spec}(e)$ form a coupling logic.

The connectives wp $e\ \{\Phi\}$ and spec(e) form a coupling logic.

A pRHL-style Hoare quadruple $\{P\}\ e_1 \sim e_2\ \{Q\}$ can be encoded as

$$P \twoheadrightarrow \mathsf{spec}(e_2) \twoheadrightarrow \mathsf{wp}\ e_1\ \{v_1.\ \mathsf{spec}(v_2) * Q(v_1, v_2)\}$$

The connectives wp $e \{\Phi\}$ and spec(e) form a coupling logic.

A pRHL-style Hoare quadruple $\{P\}\, e_1 \sim e_2\, \{Q\}$ can be encoded as

$$P \mathrel{-\!\!*} \mathsf{spec}(e_2) \mathrel{-\!\!*} \mathsf{wp}\; e_1 \; \{v_1.\, \mathsf{spec}(v_2) * Q(v_1, v_2)\}$$

Adequacy of the program logic will allow us to conclude that there exists probabilistic coupling of the execution of e_1 and e_2.

Asynchronous couplings

To support asynchronous couplings we augment the programming language with presampling tapes.

Asynchronous couplings

To support asynchronous couplings we augment the programming language with presampling tapes.

$$\mathsf{tape}, \sigma \to^1 \iota, \sigma[\iota \mapsto \epsilon] \qquad \text{if } \iota = \mathrm{fresh}(\sigma)$$

Asynchronous couplings

To support asynchronous couplings we augment the programming language with presampling tapes.

$$\mathsf{tape}, \sigma \to^{1} \iota, \sigma[\iota \mapsto \epsilon] \qquad \text{if } \iota = \mathrm{fresh}(\sigma)$$

$$\mathsf{flip}(), \sigma \to^{1/2} b, \sigma \qquad b \in \{\mathsf{true}, \mathsf{false}\}$$

Asynchronous couplings

To support asynchronous couplings we augment the programming language with presampling tapes.

$$\mathsf{tape}, \sigma \to^{1} \iota, \sigma[\iota \mapsto \epsilon] \qquad \text{if } \iota = \mathrm{fresh}(\sigma)$$

$$\mathsf{flip}(), \sigma \to^{1/2} b, \sigma \qquad b \in \{\mathsf{true}, \mathsf{false}\}$$

$$\mathsf{flip}(\iota), \sigma \to^{1/2} b, \sigma \qquad \text{if } \sigma(\iota) = \epsilon \text{ and } b \in \{\mathsf{true}, \mathsf{false}\}$$

Asynchronous couplings

To support asynchronous couplings we augment the programming language with presampling tapes.

$$\begin{aligned}
\mathsf{tape}, \sigma &\to^{1} \iota, \sigma[\iota \mapsto \epsilon] && \text{if } \iota = \mathrm{fresh}(\sigma) \\
\mathsf{flip}(), \sigma &\to^{1/2} b, \sigma && b \in \{\mathsf{true}, \mathsf{false}\} \\
\mathsf{flip}(\iota), \sigma &\to^{1/2} b, \sigma && \text{if } \sigma(\iota) = \epsilon \text{ and } b \in \{\mathsf{true}, \mathsf{false}\} \\
\mathsf{flip}(\iota), \sigma[\iota \mapsto b \cdot \vec{b}] &\to^{1} b, \sigma[\iota \mapsto \vec{b}]
\end{aligned}$$

Asynchronous couplings

To support asynchronous couplings we augment the programming language with presampling tapes.

$$\textsf{tape}, \sigma \to^{1} \iota, \sigma[\iota \mapsto \epsilon] \qquad \text{if } \iota = \mathrm{fresh}(\sigma)$$

$$\textsf{flip}(), \sigma \to^{1/2} b, \sigma \qquad b \in \{\textsf{true}, \textsf{false}\}$$

$$\textsf{flip}(\iota), \sigma \to^{1/2} b, \sigma \qquad \text{if } \sigma(\iota) = \epsilon \text{ and } b \in \{\textsf{true}, \textsf{false}\}$$

$$\textsf{flip}(\iota), \sigma[\iota \mapsto b \cdot \vec{b}] \to^{1} b, \sigma[\iota \mapsto \vec{b}]$$

... but operationally, it is not possible to (pre-)sample to the tapes!

However, labels and tapes can be erased through refinement!

$$\iota : \mathsf{tape} \vdash \mathsf{flip}() \simeq_{\mathsf{ctx}} \mathsf{flip}(\iota) : \mathsf{bool}$$

Logically, we introduce a separation logic resource

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

that satisfies, e.g.,

Logically, we introduce a separation logic resource

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

that satisfies, e.g.,

$$\frac{\forall \iota.\ \iota \hookrightarrow \epsilon \mathrel{-\!\!*} \Gamma \vDash K[\,\iota\,] \precsim e : \tau}{\Gamma \vDash K[\ \text{tape}\ ] \precsim e : \tau}$$

Logically, we introduce a separation logic resource

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

that satisfies, e.g.,

$$\frac{\forall \iota.\ \iota \hookrightarrow \epsilon \mathrel{-\!\!*} \Gamma \vDash K[\,\iota\,] \precsim e : \tau}{\Gamma \vDash K[\ \mathsf{tape}\ ] \precsim e : \tau} \qquad \frac{\iota \hookrightarrow b \cdot \vec{b} \qquad \iota \hookrightarrow \vec{b} \mathrel{-\!\!*} \Gamma \vDash K[\,b\,] \precsim e_2 : \tau}{\Gamma \vDash K[\ \mathsf{flip}(\iota)\,] \precsim e_2 : \tau}$$

Logically, we introduce a separation logic resource

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

that satisfies, e.g.,

$$\frac{\forall \iota.\ \iota \hookrightarrow \epsilon \mathrel{-\!\!*} \Gamma \vDash K[\,\iota\,] \precsim e : \tau}{\Gamma \vDash K[\ \text{tape}\ ] \precsim e : \tau} \qquad \frac{\iota \hookrightarrow b \cdot \vec{b} \qquad \iota \hookrightarrow \vec{b} \mathrel{-\!\!*} \Gamma \vDash K[\,b\,] \precsim e_2 : \tau}{\Gamma \vDash K[\ \text{flip}(\iota)\,] \precsim e_2 : \tau}$$

$$\frac{f \text{ bijection} \qquad \iota \hookrightarrow \vec{b} \qquad \forall b.\ \iota \hookrightarrow \vec{b} \cdot b \mathrel{-\!\!*} \Gamma \vDash e \precsim K'[\,f(b)\,] : \tau}{\Gamma \vDash e \precsim K'[\ \text{flip}()\,] : \tau}$$

Logically, we introduce a separation logic resource

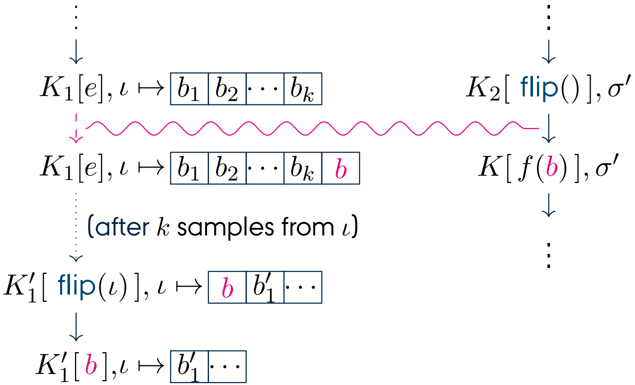
$$\iota \hookrightarrow \vec{b}$$

that satisfies, e.g.,

$$\frac{\forall \iota.\ \iota \hookrightarrow \epsilon \mathrel{-\!\!*} \Gamma \vDash K[\,\iota\,] \precsim e : \tau}{\Gamma \vDash K[\ \textsf{tape}\ ] \precsim e : \tau} \qquad \frac{\iota \hookrightarrow b \cdot \vec{b} \qquad \iota \hookrightarrow \vec{b} \mathrel{-\!\!*} \Gamma \vDash K[\,b\,] \precsim e_2 : \tau}{\Gamma \vDash K[\ \textsf{flip}(\iota)\,] \precsim e_2 : \tau}$$

$$\frac{f \text{ bijection} \qquad \iota \hookrightarrow \vec{b} \qquad \forall b.\ \iota \hookrightarrow \vec{b} \cdot b \mathrel{-\!\!*} \Gamma \vDash e \precsim K'[\,f(b)\,] : \tau}{\Gamma \vDash e \precsim K'[\ \textsf{flip}()\,] : \tau}$$

Effectively, we turn reasoning about prob. choice into reasoning about state!


$K_1[e], \iota \mapsto$ b_1 b_2 $\cdots$ b_k
$K_2[$ flip() $], \sigma'$
$K_1[e], \iota \mapsto$ b_1 b_2 $\cdots$ b_k b
$K[\, f(b) \,], \sigma'$
(after k samples from ι)
$K_1'[$ flip(ι) $], \iota \mapsto$ b b_1' $\cdots$
$K_1'[\, b \,], \iota \mapsto$ b_1' $\cdots$

$$\textsf{let}\ b = \textsf{flip in}\ \lambda\_.\ b \quad \precsim_{\textsf{ctx}} \quad \begin{array}{l} \textsf{let}\ r = \textsf{ref}(\textsf{None})\ \textsf{in} \\ \lambda\_.\ \textsf{match}\ !\,r\ \textsf{with} \\ \quad \textsf{Some}(b) \Rightarrow b \\ \quad \mid \textsf{None} \Rightarrow \textsf{let}\ b = \textsf{flip in} \\ \qquad\qquad\quad r \leftarrow \textsf{Some}(b); \\ \qquad\qquad\quad b \\ \quad \textsf{end} \end{array}$$

$$
\begin{array}{l}
\textsf{let } b = \textsf{flip in} \\
\lambda\_.\, b
\end{array}
\quad \precsim_{\textsf{ctx}} \quad
\begin{array}{l}
\textsf{let } \iota = \textsf{tape}(1) \textsf{ in} \\
\textsf{let } r = \textsf{ref}(\textsf{None}) \textsf{ in} \\
\lambda\_.\ \textsf{match } !r \textsf{ with} \\
\quad \textsf{Some}(b) \Rightarrow b \\
\quad \mid \textsf{None} \Rightarrow \textsf{let } b = \textsf{flip}(\iota) \textsf{ in} \\
\qquad\qquad r \leftarrow \textsf{Some}(b); \\
\qquad\qquad b \\
\textsf{end}
\end{array}
\quad \precsim_{\textsf{ctx}} \quad
\begin{array}{l}
\textsf{let } r = \textsf{ref}(\textsf{None}) \textsf{ in} \\
\lambda\_.\ \textsf{match } !r \textsf{ with} \\
\quad \textsf{Some}(b) \Rightarrow b \\
\quad \mid \textsf{None} \Rightarrow \textsf{let } b = \textsf{flip in} \\
\qquad\qquad r \leftarrow \textsf{Some}(b); \\
\qquad\qquad b \\
\textsf{end}
\end{array}
$$

Summary

- **Clutch**: a higher-order relational separation logic for proving contextual refinement of probabilistic programs
- Asynchronous probabilistic couplings
- More examples in the paper
  - → ElGamal security, lazy hash functions, lazy big integers, ...
- Full mechanization of all results in Coq

Thank you!

Contact	gregersen@cs.au.dk
Paper	`https://arxiv.org/abs/2301.10061`
Coq dev.	`https://github.com/logsem/clutch`

Extras

Let $\mathrm{step}(\rho) \in \mathcal{D}(\mathsf{Cfg})$ be the distribution of single step reduction of $\rho \in \mathsf{Cfg}$.

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

$$\mathrm{exec}(\rho)(v) \triangleq \lim_{n\to\infty} \mathrm{exec}_n(\rho)(v)$$

$$\mathrm{term}(\rho) \triangleq \sum_{v \in \mathsf{Val}} \mathrm{exec}(\rho)(v)$$

Definition (Coupling)

Let $\mu_1 \in \mathcal{D}(A)$, $\mu_2 \in \mathcal{D}(B)$. A sub-distribution $\mu \in \mathcal{D}(A \times B)$ is a coupling of μ_1 and μ_2 if

1. $\forall a.\ \sum_{b \in B} \mu(a, b) = \mu_1(a)$
2. $\forall b.\ \sum_{a \in A} \mu(a, b) = \mu_2(b)$

Given a relation $R \subseteq A \times B$ we say μ is an R-coupling if furthermore $\mathrm{supp}(\mu) \subseteq R$. We write $\mu_1 \sim \mu_2 : R$ if there exists an R-coupling of μ_1 and μ_2.

Lemma

If $\mu_1 \sim \mu_2 : (=)$ then $\mu_1 = \mu_2$.

Definition (Left-Partial Coupling)

Let $\mu_1 \in \mathcal{D}(A), \mu_2 \in \mathcal{D}(B)$. A sub-distribution $\mu \in \mathcal{D}(A \times B)$ is a left-partial coupling of μ_1 and μ_2 if

1. $\forall a.\ \sum_{b \in B} \mu(a, b) = \mu_1(a)$
2. $\forall b.\ \sum_{a \in A} \mu(a, b) \leq \mu_2(b)$

Given a relation $R \subseteq A \times B$ we say μ is an R-left-partial-coupling if furthermore $\mathrm{supp}(\mu) \subseteq R$. We write $\mu_1 \lesssim \mu_2 : R$ if there exists an R-left-partial-coupling of μ_1 and μ_2.

Lemma

If $\mu_1 \sim \mu_2 : R$ then $\mu_1 \lesssim \mu_2 : R$.

Lemma

If $\mu_1 \lesssim \mu_2 : (=)$ then $\forall a.\ \mu_1(a) \leq \mu_2(a)$.

The adequacy theorem relies on the fact that presampling does not matter.

Lemma (Erasure)

If $\sigma_1(\iota) \in \mathsf{dom}(\sigma_1)$ then

$$\mathrm{exec}_n(e_1, \sigma_1) \sim (\mathrm{step}_\iota(\sigma_1) \gg\!= \lambda \sigma_2.\ \mathrm{exec}_n(e_1, \sigma_2)) : (=)$$