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

Motivating example

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

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

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

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

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

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

Operational semantics

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

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

$$\vdots$$

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

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

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

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

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

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

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

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

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.

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 $\Gamma \vdash e_1 \precsim_{\text{ctx}} e_2 : \tau$.

Reading

$$\frac{A_1 \qquad \cdots \qquad A_n}{B}$$

as $A_1 * \cdots * A_n \vdash B$, the judgment satisfies, e.g.,

Reading

$$\frac{A_1 \qquad \cdots \qquad A_n}{B}$$

as $A_1 * \cdots * A_n \vdash B$, the judgment satisfies, e.g.,

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

Reading

$$\frac{A_1 \qquad \cdots \qquad A_n}{B}$$

as $A_1 * \cdots * A_n \vdash B$, the judgment satisfies, e.g.,

$$\frac{e_1 \overset{\text{pure}}{\rightsquigarrow} e_1' \qquad \Gamma \vDash K[\, e_1' \,] \precsim e_2 : \tau}{\Gamma \vDash K[\, e_1 \,] \precsim e_2 : \tau} \qquad \frac{\ell \mapsto v \qquad \ell \mapsto v \mathrel{-\!\!*} \Gamma \vDash K[\, v \,] \precsim e_2 : \tau}{\Gamma \vDash K[\, !\ell \,] \precsim e_2 : \tau}$$

Reading

$$\frac{A_1 \qquad \cdots \qquad A_n}{B}$$

as $A_1 * \cdots * A_n \vdash B$, the judgment satisfies, e.g.,

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

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

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

Reading

$$\frac{A_1 \qquad \cdots \qquad A_n}{B}$$

as $A_1 * \cdots * A_n \vdash B$, the judgment satisfies, e.g.,

$$\frac{e_1 \overset{\text{pure}}{\rightsquigarrow} e_1' \qquad \Gamma \vDash K[\, e_1' \,] \precsim e_2 : \tau}{\Gamma \vDash K[\, e_1 \,] \precsim e_2 : \tau} \qquad \frac{\ell \mapsto v \qquad \ell \mapsto v -\!\!* \Gamma \vDash K[\, v \,] \precsim e_2 : \tau}{\Gamma \vDash K[\, !\ell \,] \precsim e_2 : \tau}$$

$$\frac{\forall b.\, \Gamma \vDash K[\, b \,] \precsim e_2 : \tau}{\Gamma \vDash K[\ \textsf{flip}\ ] \precsim e_2 : \tau} \qquad \frac{f \text{ bijection} \qquad \forall b.\, \Gamma \vDash K[\, b \,] \precsim K'[\, f(b) \,] : \tau}{\Gamma \vDash K[\ \textsf{flip}\ ] \precsim K'[\ \textsf{flip}\ ] : \tau}$$

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.

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

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.

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

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

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

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

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

As a consequence, labels and tapes can be erased!

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

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.,

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

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

Logically, we introduce a separation logic resource

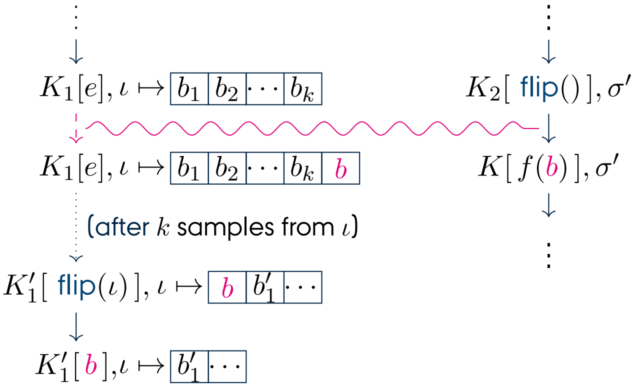
$$\iota \hookrightarrow \vec{b}$$

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

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

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

ElGamal public key encryption

$$
\begin{aligned}
keygen &\triangleq \lambda\,\_\,.\ \text{let } sk = \text{rand}(n) \text{ in} \\
&\qquad \text{let } pk = g^{sk} \text{ in} \\
&\qquad (sk, pk) \\
dec &\triangleq \lambda\, sk\ (B, X).\ \ X \cdot B^{-sk}
\end{aligned}
\qquad
\begin{aligned}
enc &\triangleq \lambda\, pk\ msg.\ \text{let } b = \text{rand}(n) \text{ in} \\
&\qquad \text{let } B = g^{b} \text{ in} \\
&\qquad \text{let } X = msg \cdot pk^{b} \text{ in} \\
&\qquad (B, X)
\end{aligned}
$$

where $G = (1,\ \cdot\,,\ -^{-1})$ is a finite cyclic group generated by g, and $n = |G| - 1$.

$PK_{real} \triangleq$

let $(sk, pk) = keygen()$ in

let $count =$ ref 0 in

let $query = \lambda\, msg.$

  if $!\, count \neq 0$ then

    None

  else

    $count \leftarrow 1;$

    let $(B, X) =$ $enc\; pk\; msg$ in

    Some (B, X)

in $(pk, query)$

$PK_{rand} \triangleq$

let $(sk, pk) = keygen()$ in

let $count =$ ref 0 in

let $query = \lambda\, msg.$

  if $!\, count \neq 0$ then

    None

  else

    $count \leftarrow 1;$

    let $b =$ rand(n) in

    let $x =$ rand(n) in

    let $(B, X) = (g^b, g^x)$ in

    Some (B, X)

in $(pk, query)$

Security reduction

The security of ElGamal encryption can be reduced to the DDH assumption.

$$DH_{real} \triangleq \text{let } a = \text{rand}(n) \text{ in} \\ \text{let } b = \text{rand}(n) \text{ in} \\ (g^a, g^b, g^{ab})$$

$$DH_{rand} \triangleq \text{let } a = \text{rand}(n) \text{ in} \\ \text{let } b = \text{rand}(n) \text{ in} \\ \text{let } c = \text{rand}(n) \text{ in} \\ (g^a, g^b, g^c)$$

are “indistinguishable” for certain groups and adversaries.

By exhibiting a PPT context $\mathcal{C}$ such that

$$\vdash PK_{real} \simeq_{\textsf{ctx}} \mathcal{C}[DH_{real}] : \tau_{PK}$$
$$\vdash PK_{rand} \simeq_{\textsf{ctx}} \mathcal{C}[DH_{rand}] : \tau_{PK}$$

we can complete the reduction outside of Clutch.

$$
\begin{aligned}
C[-] \triangleq\ & \mathsf{let}\ (pk, B, C) = -\ \mathsf{in} \\
& \mathsf{let}\ count = \mathsf{ref}\ 0\ \mathsf{in} \\
& \mathsf{let}\ query = \lambda\, msg. \\
& \quad \mathsf{if}\ \ !\, count \neq 0\ \mathsf{then} \\
& \qquad \mathsf{None} \\
& \quad \mathsf{else} \\
& \qquad count \leftarrow 1; \\
& \qquad \mathsf{let}\ X = msg \cdot C\ \mathsf{in} \\
& \qquad \mathsf{Some}\ (B, X) \\
& \mathsf{in}\ (pk, query)
\end{aligned}
$$

$PK_{real} \quad \simeq_{\mathsf{ctx}} \quad PK^{tape}_{real} \quad \simeq_{\mathsf{ctx}} \quad \mathcal{C}[DH_{real}]$

PK_{real}

```
let sk = rand(n) in
let pk = g^sk in

let count = ref 0 in
let query = λ msg.
  if !count ≠ 0 then
    None
  else
    count ← 1;
    let b = rand(n) in
    let B = g^b in

    let X = msg · pk^b in
    Some (B, X)
in (pk, query)
```

PK^{tape}_{real}

```
let β = tape(n) in
let sk = rand(n) in
let pk = g^sk in

let count = ref 0 in
let query = λ msg.
  if !count ≠ 0 then
    None
  else
    count ← 1;
    let b = rand(n, β) in
    let B = g^b in
    let C = pk^b in
    let X = msg · C in
    Some (B, X)
in (pk, query)
```

$\mathcal{C}[DH_{real}]$

```
let (pk, B, C) =
  let a = rand(n) in
  let b = rand(n) in
  (g^a, g^b, g^ab) in
let count = ref 0 in
let query = λ msg.
  if !count ≠ 0 then
    None
  else
    count ← 1;



    let X = msg · C in
    Some (B, X)
in (pk, query)
```

Clutch

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

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

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

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

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

The connectives wp $e\,\{\Phi\}$ and 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

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

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

Couplings

Goal

A relational program logic that proves the existence of a probabilistic coupling between the two programs.

Couplings can be constructed compositionally and will allow us to prove equality between distributions.

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 relation $R : A \times B$ 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.

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 relation $R : A \times B$ 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.

For example, the distribution $\mu_{\text{coins}} \in \mathcal{D}(\mathbb{B} \times \mathbb{B})$ where

$$\mu_{\text{coins}}(b_1, b_2) \triangleq \begin{cases} \frac{1}{2} & \text{if } b_1 = b_2 \\ 0 & \text{otherwise} \end{cases}$$

is a witness of a coupling $\mu_{\text{coin}} \sim \mu_{\text{coin}} : (=)$ as can be easily verified.

Lemma (Composition of couplings)

Let $R : A \times B$, $S : A' \times B'$, $\mu_1 \in \mathcal{D}(A)$, $\mu_2 \in \mathcal{D}(B)$, $f_1 : A \to \mathcal{D}(A')$, and $f_2 : B \to \mathcal{D}(B')$.

1. If $(a, b) \in R$ then $\mathrm{ret}(a) \sim \mathrm{ret}(b) : R$.
2. If $\mu_1 \sim \mu_2 : R$ and $\forall (a, b) \in R.\ f_1(a) \sim f_2(b) : S$ then $\mu_1 \gg\!= f_1 \sim \mu_2 \gg\!= f_2 : S$

Lemma (Equality coupling)

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

Logical refinement

We define the logical refinement judgment for closed terms

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

which we extend to open terms by closing substitutions as usual

$$\Gamma \vDash e_1 \precsim e_2 : \tau \triangleq \forall \vec{v}, \vec{w}.\ [\![\Gamma]\!](\vec{v}, \vec{w}) \twoheadrightarrow\!\!* \vDash e_1[\vec{v}/\Gamma] \precsim e_2[\vec{w}/\Gamma] : \tau$$

Peeling the onion (layer 1)

The structure of the refinement judgment is "the usual" one:

$$\vDash e_1 \precsim e_2 : \tau \quad \triangleq \quad \forall K.\, \mathsf{specCtx} \mathrel{-\!\!*} \mathsf{spec}(K[\,e_2\,]) \mathrel{-\!\!*} \\ \mathsf{wp}\ e_1\ \{v_1.\exists v_2.\, \mathsf{spec}(K[\,v_2\,]) * [\![\tau]\!](v_1, v_2)\}$$

All the magic happens in the weakest precondition predicate.

Peeling the onion (layer 2)

The intuitive reading of the weakest precondition is that the execution of e_1 can be coupled with the execution of some other program.

$$
\begin{aligned}
\mathsf{wp}\ e_1\ \{\Phi\} \triangleq\ & (e_1 \in \mathsf{Val} \land \Phi(e_1)) \lor \\
& (e_1 \notin \mathsf{Val} \land \forall \sigma_1, \rho_1.\ S(\sigma_1) * G(\rho_1) \mathrel{-\!\!*} \\
& \quad \mathsf{execCoupl}(e_1, \sigma_1, \rho_1)(\lambda e_2, \sigma_2, \rho_2. \\
& \quad\quad \triangleright S(\sigma_2) * G(\rho_2) * \mathsf{wp}\ e_2\ \{\Phi\}))
\end{aligned}
$$

Peeling the onion (layer 3)

$$\frac{\mathsf{red}(\rho_1) \qquad \mathsf{prim\_step}(\rho_1) \sim \mathrm{ret}(\rho_1') : R \qquad \forall \rho_2.\, R(\rho_2, \rho_1') \twoheadrightarrow Z(\rho_2, \rho_1')(Z)}{\mathsf{execCoupl}(\rho_1, \rho_1')(Z)}$$

$$\frac{\mathrm{ret}(\rho_1) \sim \mathsf{prim\_step}(\rho_1') : R \qquad \forall \rho_2'.\, R(\rho_1, \rho_2') \twoheadrightarrow \mathsf{execCoupl}(\rho_1, \rho_2')(Z)}{\mathsf{execCoupl}(\rho_1, \rho_1')(Z)}$$

$$\frac{\begin{array}{c}\mathsf{red}(\rho_1) \\ \mathsf{prim\_step}(\rho_1) \sim \mathsf{prim\_step}(\rho_1') : R \qquad \forall \rho_2, \rho_2'.\, R(\rho_2, \rho_2') \twoheadrightarrow Z(\rho_2, \rho_2')\end{array}}{\mathsf{execCoupl}(\rho_1, \rho_1')(Z)}$$

Peeling the onion (layer 3) cont'd

$$\frac{\mathrm{step}_\iota(\sigma_1) \sim \textsf{prim\_step}(\rho'_1) : R \qquad \forall \sigma_2, \rho'_2.\ R(\sigma_2, \rho'_2) \twoheadrightarrow \textsf{execCoupl}((e_1, \sigma_2), \rho'_2)(Z)}{\textsf{execCoupl}((e_1, \sigma_1), \rho'_1)(Z)}$$

$$\frac{\mathrm{step}_\iota(\sigma_1) \sim \mathrm{step}_{\iota'}(\sigma'_1) : R \qquad \forall \sigma_2, \sigma'_2.\ R(\sigma_2, \sigma'_2) \twoheadrightarrow \textsf{execCoupl}((e_1, \sigma_2), (e'_1, \sigma'_2))(Z)}{\textsf{execCoupl}((e_1, \sigma_1, (e'_1, \sigma'_1))(Z)}$$

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

Lemma (Erasure)

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

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

Soundness

Theorem (Adequacy)

Let $\varphi : \mathsf{Val} \times \mathsf{Val} \to \mathsf{Prop}$ be a predicate on values in the meta-logic. If

$$\mathsf{specCtx} * \mathsf{spec}(e') \vdash \mathsf{wp}\ e\ \left\{v.\exists v'.\, \mathsf{spec}(v') * \varphi(v, v')\right\}$$

is provable then $\forall n.\ \mathrm{exec}_n(e, \sigma) \lesssim \mathrm{exec}(e', \sigma') : \varphi$.

Soundness

Theorem (Adequacy)

Let $\varphi : \mathsf{Val} \times \mathsf{Val} \to \mathsf{Prop}$ be a predicate on values in the meta-logic. If

$$\mathsf{specCtx} * \mathsf{spec}(e') \vdash \mathsf{wp}\ e \left\{v.\exists v'.\, \mathsf{spec}(v') * \varphi(v, v')\right\}$$

is provable then $\forall n.\ \mathrm{exec}_n(e, \sigma) \lesssim \mathrm{exec}(e', \sigma') : \varphi$.

Due to Löb induction, the LHS program may not terminate, i.e., in some execution paths the distribution may not have mass. For this reason, what we show in the end is a left-partial coupling.

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 relation $R : A \times B$ 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.

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 relation $R : A \times B$ 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 $\mu_1 \lesssim \mu_2 : R$.

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 relation $R : A \times B$ 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 $\forall a.\ \mu_1(a) \leq \mu_2(a)$.

Summary

- **Clutch**: a higher-order relational separation logic for proving contextual refinement of probabilistic programs
- Asynchronous probabilistic couplings
- More examples in the paper
  - $\rightarrow$ 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`

Peeling the onion (layer 4)

$$\vDash e_1 \precsim e_2 : \tau \triangleq \forall K.\, \mathsf{specCtx} \twoheadrightarrow \mathsf{spec}_\circ(K[\,e_2\,]) \twoheadrightarrow \\ \mathsf{wp}\ e_1\ \{v_1.\exists v_2.\, \mathsf{spec}_\circ(K[\,v_2\,]) * [\![\tau]\!](v_1, v_2)\}$$

$$G(\rho) \triangleq \mathsf{specInterp}_\bullet(\rho)$$

$$\mathsf{specInv} \triangleq \exists \rho, e, \sigma, n.\, \mathsf{specInterp}_\circ(\rho) * \mathsf{spec}_\bullet(e) * \mathsf{heaps}(\sigma) * \\ \mathrm{execConf}_n(\rho)(e, \sigma) = 1$$

$$\mathsf{specCtx} \triangleq \boxed{\mathsf{specInv}}^{\mathcal{N}.\mathsf{spec}}$$

This allows the right-hand side to “run ahead”, e.g.,

$$\frac{\mathsf{specCtx} \qquad e \overset{\text{pure}}{\rightsquigarrow} e' \qquad \mathcal{N}.\mathsf{spec} \subseteq \mathcal{E}}{\mathsf{spec}(K[\,e\,]) \vdash \Rrightarrow \mathsf{spec}(K[\,e'\,])}$$

In the adequacy theorem and when coupling program steps, the program in the weakest precondition first “catches up”.