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

$\text{let } b = \text{flip in}$

$\lambda \_ .\, b$

Motivating example

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

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

Motivating example

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

Motivating example

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

$$\simeq$$

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

While this example seems esoteric, the pattern shows up in many places: cryptographic security, hash functions, lazily-sampled big integers, ...

The pRHL approach

For such properties, people have developed probabilistic relational Hoare logics, where sampling statements are related through so-called coupling rules.

pRHL-couple

$$\frac{}{\{\text{True}\}\ \text{flip} \sim \text{flip}\ \{v_1, v_2.\ \exists(b : \mathbb{B}).\ b = v_1 = v_2\}}$$

However, it requires you to "synchronize" the probabilistic choices.

Motivating example

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

Motivating example

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

Motivating example

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

This work

Goal: prove 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

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

$$\vdots$$

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

For this presentation, we only 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

Proving contextual refinement

1. A probabilistic relational separation logic on top of Iris

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)

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)

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

Theorem (Soundness)

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

Symbolic execution rules

Symbolic execution rules

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

Symbolic execution rules

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

Symbolic execution rules

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

Symbolic execution rules

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

Symbolic execution rules

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

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

Symbolic execution rules

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

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

Symbolic execution rules

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

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

Symbolic execution rules

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

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

Symbolic execution rules

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

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

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

Symbolic execution rules

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

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

Symbolic execution rules

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

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

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

Relational separation logic

Clutch is built on top of the Iris separation logic framework

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

within 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 $\mathsf{wp}\ e\ \{\Phi\}$ and $\mathsf{spec}(e)$ form a coupling logic.
- A pRHL-style quadruple $\{P\}\ e_1 \sim e_2\ \{Q\}$ can be encoded as

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

- The connectives wp $e\ \{\Phi\}$ and spec(e) form a coupling logic.
- A pRHL-style quadruple $\{P\}\ e_1 \sim e_2\ \{Q\}$ can be encoded as

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

- Soundness of the coupling logic will allow us to conclude that there exists a probabilistic coupling of the execution of e_1 and e_2.

Asynchronous couplings

To support asynchronous couplings, we introduce ghost presampling tapes.

Asynchronous couplings

To support asynchronous couplings, we introduce ghost presampling tapes.

Operationally, we extend the execution state with dynamically allocated tapes.

$$\rho \in \mathsf{Cfg} ::= \mathsf{Expr} \times \mathsf{State}$$
$$\sigma \in \mathsf{State} ::= \mathsf{Heap} \times \mathsf{Tapes}$$

Asynchronous couplings

To support asynchronous couplings, we introduce ghost presampling tapes.

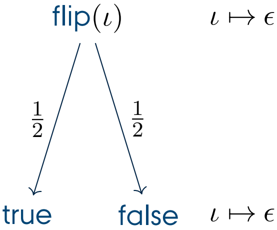
Operationally, we extend the execution state with dynamically allocated tapes.

$$\begin{aligned} \rho &\in \mathsf{Cfg} ::= \mathsf{Expr} \times \mathsf{State} \\ \sigma &\in \mathsf{State} ::= \mathsf{Heap} \times \mathsf{Tapes} \\ e &\in \mathsf{Expr} ::= \ \dots \mid \mathsf{flip} \mid \mathsf{flip}(e) \mid \mathsf{tape} \end{aligned}$$

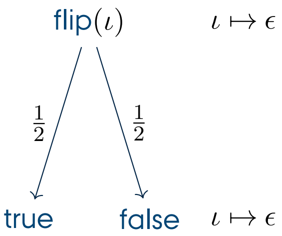
Presampling tapes

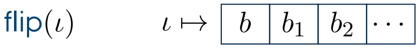
$\mathsf{flip}(\iota)$ $\qquad$ $\iota \mapsto \epsilon$

Presampling tapes

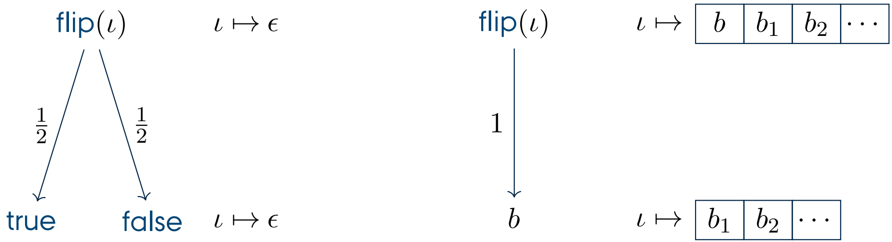


Presampling tapes

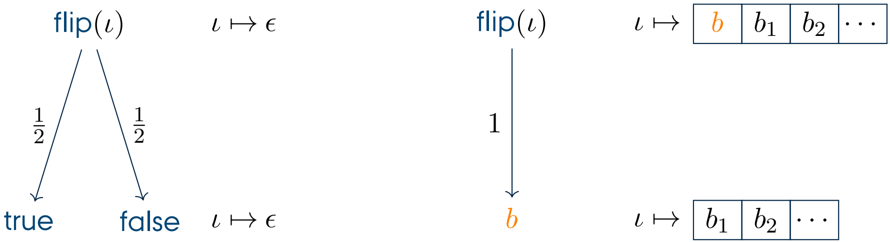




Presampling tapes



Presampling tapes



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

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

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

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

Instead, tapes will non-deterministically be populated with fresh samples.

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

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

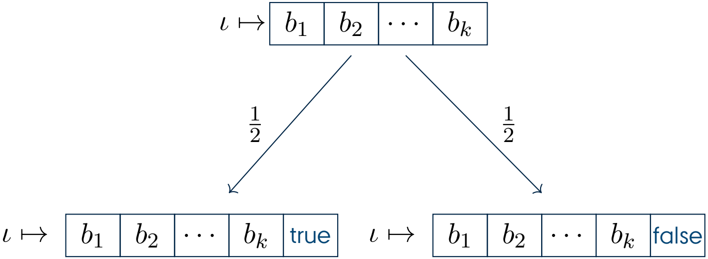
Instead, tapes will non-deterministically be populated with fresh samples.

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

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

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

Instead, tapes will non-deterministically be populated with fresh samples.



Logically, we introduce a separation logic resource

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

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

Logically, we introduce a separation logic resource

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

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

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

Logically, we introduce a separation logic resource

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

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

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

Logically, we introduce a separation logic resource

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

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

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

Logically, we introduce a separation logic resource

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

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

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

Logically, we introduce a separation logic resource

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

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

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

Logically, we introduce a separation logic resource

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

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

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

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

Logically, we introduce a separation logic resource

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

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

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

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

Logically, we introduce a separation logic resource

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

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

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

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

Logically, we introduce a separation logic resource

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

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

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

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

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

to couple program samplings asynchronously!

Asynchronous couplings

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

$$\frac{\iota \hookrightarrow \vec{b} \qquad \forall b.\, \iota \hookrightarrow \vec{b} \cdot b \twoheadrightarrow \Gamma \vDash e \precsim K[\, b \,] : \tau}{\Gamma \vDash e \precsim K[\, \mathsf{flip} \,] : \tau}$$

to couple program samplings asynchronously!

Asynchronous couplings

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

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

to couple program samplings asynchronously!

Asynchronous couplings

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

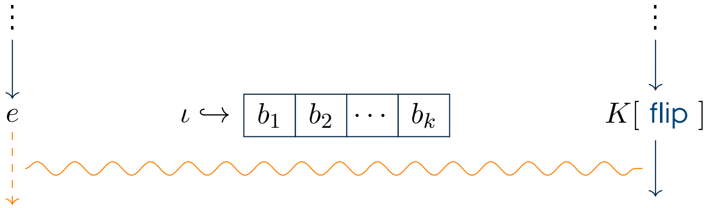
$$\frac{\iota \hookrightarrow \vec{b} \qquad \forall b.\, \boxed{\iota \hookrightarrow \vec{b} \cdot b} \twoheadrightarrow \Gamma \vDash e \precsim \boxed{K[\, b \,]} : \tau}{\Gamma \vDash e \precsim K[\ \text{flip}\ ] : \tau}$$

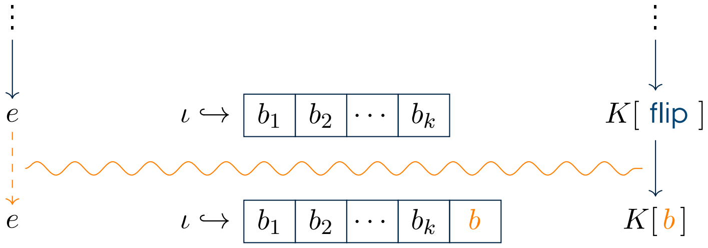
to couple program samplings asynchronously!

$$e$$

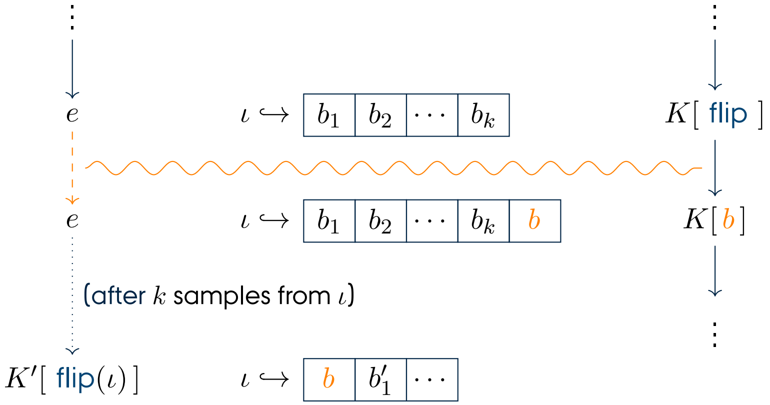
$$\iota \hookrightarrow \boxed{b_1}\boxed{b_2}\boxed{\cdots}\boxed{b_k}$$

$$K[\ \mathsf{flip}\ ]$$

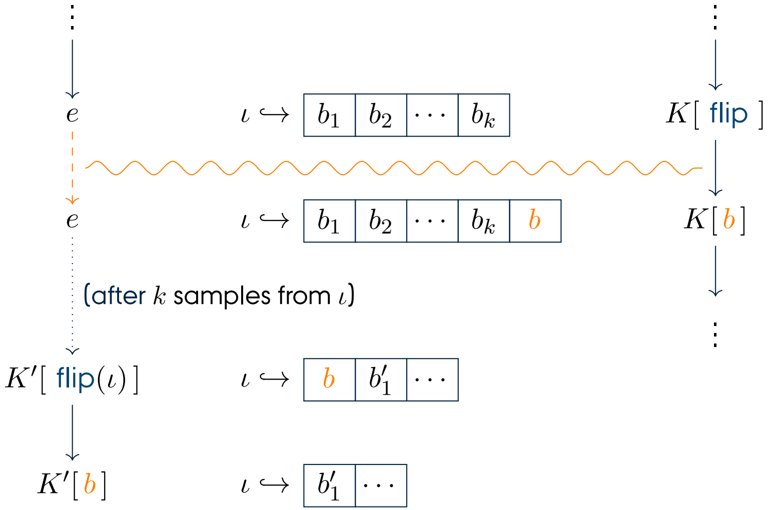

e
ι ↪ b1 b2 ⋯ bk
K[flip]



e
ι ↪ b1 b2 ⋯ bk
K[flip]
e
ι ↪ b1 b2 ⋯ bk b
K[b]



e
$\iota \hookrightarrow$ b_1 b_2 $\cdots$ b_k
$K[$ flip $]$
e
$\iota \hookrightarrow$ b_1 b_2 $\cdots$ b_k b
$K[b]$
(after k samples from ι)
$K'[$ flip(ι) $]$
$\iota \hookrightarrow$ b b'_1 $\cdots$


e
$\iota \hookrightarrow$ b_1 b_2 $\cdots$ b_k
$K[\ \text{flip}\ ]$
e
$\iota \hookrightarrow$ b_1 b_2 $\cdots$ b_k b
$K[\ b\ ]$
(after k samples from ι)
$K'[\ \text{flip}(\iota)\ ]$
$\iota \hookrightarrow$ b b'_1 $\cdots$
$K'[\ b\ ]$
$\iota \hookrightarrow$ b'_1 $\cdots$

Motivating example cont'd

$$\begin{aligned}
&\textsf{let } r = \textsf{ref}(\textsf{None}) \textsf{ in} \\
&\lambda_.\ \textsf{match } !r \textsf{ with} \\
&\quad \textsf{Some}(b) \Rightarrow b \\
&\quad \mid \textsf{None} \Rightarrow \textsf{let } b = \textsf{flip in} \\
&\qquad\qquad r \leftarrow \textsf{Some}(b); \\
&\qquad\qquad b \\
&\quad \textsf{end}
\end{aligned}
\quad \precsim_{\textsf{ctx}} \quad
\begin{aligned}
&\textsf{let } b = \textsf{flip in} \\
&\lambda_.\ b
\end{aligned}$$

Motivating example cont'd

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

Motivating example cont'd

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

Motivating example cont'd

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

Motivating example cont'd

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

Summary

- A higher-order probabilistic relational separation logic, Clutch, for proving contextual refinement of higher-order probabilistic programs.
- A proof method for asynchronous couplings.
- Many more examples in the paper:
  → Cryptographic security, hash functions, lazily-sampled big integers, ...
- Full mechanization of all results in the Coq proof assistant.

Contact gregersen@cs.au.dk
Paper `https://doi.org/10.1145/3632868`
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

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