Coneris:

Modular Reasoning about Error Bounds for Concurrent Probabilistic Programs

Kwing Hei Li[1] Alejandro Aguirre[1] Simon Gregersen[2]
Philipp Haselwarter[1] Joseph Tassarotti[2] Lars Birkedal[1]

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

A simple probability problem

Heili is asking two different girls out on a date via text this weekend, *separately*.

A simple probability problem

Heili is asking two different girls out on a date via text this weekend, *separately*.

Each girl *independently* has a probability of 1/4 of saying no to his request (3/4 of saying yes).

A simple probability problem

Heili is asking two different girls out on a date via text this weekend, *separately*.

Each girl *independently* has a probability of 1/4 of saying no to his request (3/4 of saying yes).

What is the probability I will be rejected by both girls and have nothing to do this weekend?

A sequential probabilistic program

$$
\begin{aligned}
\mathit{twoAdd} \triangleq\ & \mathsf{let}\ l = \mathsf{ref}\ 0\ \mathsf{in} \\
& l \leftarrow (!\, l + \mathsf{rand}\ 3); \\
& l \leftarrow (!\, l + \mathsf{rand}\ 3); \\
& !\, l
\end{aligned}
$$

A sequential probabilistic program

$$\begin{aligned} \mathit{twoAdd} \triangleq\ & \mathsf{let}\ l = \mathsf{ref}\ 0\ \mathsf{in} \\ & l \leftarrow (!\, l + \mathsf{rand}\ 3); \\ & l \leftarrow (!\, l + \mathsf{rand}\ 3); \\ & !\, l \end{aligned}$$

$\mathsf{rand}\ N$ steps to any integer n between 0 and N uniformly with probability $1/(N+1)$

A sequential probabilistic program

$$\begin{aligned} \mathit{twoAdd} \triangleq\ & \mathsf{let}\ l = \mathsf{ref}\ 0\ \mathsf{in} \\ & l \leftarrow (!\, l + \mathsf{rand}\ 3); \\ & l \leftarrow (!\, l + \mathsf{rand}\ 3); \\ & !\, l \end{aligned}$$

$\mathsf{rand}\ N$ steps to any integer n between 0 and N uniformly with probability $1/(N+1)$

Aim: show *twoAdd* returns 0 (error result)
with probability at most $1/16$

Eris to the rescue

- **KEY IDEA: We internalize error as a separation logic resource, aka *error credit***

Eris to the rescue

- **KEY IDEA: We internalize error as a separation logic resource, aka *error credit***
- $↯(\varepsilon)$ asserts ownership of ε error credits, with $\varepsilon \in [0, 1]$

Eris to the rescue

- **KEY IDEA: We internalize error as a separation logic resource, aka *error credit***
- $↯(\varepsilon)$ asserts ownership of ε error credits, with $\varepsilon \in [0, 1]$
- Adequacy: $\{↯(\varepsilon)\}\, e\, \{v.\phi(v)\} \Rightarrow \Pr_{\mathsf{exec}\, e}[\neg\phi] \leqslant \varepsilon$

Eris to the rescue

- **KEY IDEA: We internalize error as a separation logic resource, aka *error credit***
- $\lightning(\varepsilon)$ asserts ownership of ε error credits, with $\varepsilon \in [0, 1]$
- Adequacy: $\{\lightning(\varepsilon)\}\, e\, \{v.\phi(v)\} \Rightarrow \mathrm{Pr}_{\mathsf{exec}\, e}[\neg\phi] \leqslant \varepsilon$

1. $\lightning(\varepsilon_1) * \lightning(\varepsilon_2) \dashv\vdash \lightning(\varepsilon_1 + \varepsilon_2)$

Eris to the rescue

- **KEY IDEA: We internalize error as a separation logic resource, aka *error credit***
- $\lightning(\varepsilon)$ asserts ownership of ε error credits, with $\varepsilon \in [0, 1]$
- Adequacy: $\{\lightning(\varepsilon)\}\, e\, \{v.\phi(v)\} \Rightarrow \mathrm{Pr}_{\mathrm{exec}\, e}[\neg\phi] \leqslant \varepsilon$

1. $\lightning(\varepsilon_1) * \lightning(\varepsilon_2) \dashv\vdash \lightning(\varepsilon_1 + \varepsilon_2)$
2. $\lightning(1) \vdash \bot$

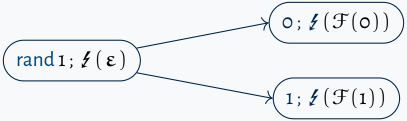
Eris to the rescue

- **KEY IDEA: We internalize error as a separation logic resource, aka *error credit***
- $\lightning(\varepsilon)$ asserts ownership of ε error credits, with $\varepsilon \in [0, 1]$
- Adequacy: $\{\lightning(\varepsilon)\}\, e\, \{v.\phi(v)\} \Rightarrow \Pr_{\text{exec}\, e}[\neg\phi] \leqslant \varepsilon$

1. $\lightning(\varepsilon_1) * \lightning(\varepsilon_2) \dashv\vdash \lightning(\varepsilon_1 + \varepsilon_2)$
2. $\lightning(1) \vdash \bot$
3. $$\frac{\sum_{i=0}^{N} \dfrac{\mathcal{F}(i)}{N+1} \leqslant \varepsilon}{\vdash \{\lightning(\varepsilon)\}\ \texttt{rand}\ N\, \{n\,.\, \lightning(\mathcal{F}(n))\}}\ \textsc{ht-rand-exp}$$

- **KEY IDEA: We internalize error as a separation logic resource, aka *error credit***
- $\lightning(\varepsilon)$ asserts ownership of ε error credits, with $\varepsilon \in [0, 1]$
- Adequacy: $\{\lightning(\varepsilon)\}\, e\, \{v.\phi(v)\} \Rightarrow \Pr_{\mathsf{exec}\, e}[\neg\phi] \leqslant \varepsilon$

1. $\lightning(\varepsilon_1) * \lightning(\varepsilon_2) \dashv\vdash \lightning(\varepsilon_1 + \varepsilon_2)$
2. $\lightning(1) \vdash \bot$
3. $$\frac{\sum_{i=0}^{N} \dfrac{\mathcal{F}(i)}{N+1} \leqslant \varepsilon}{\vdash \{\lightning(\varepsilon)\}\ \mathsf{rand}\, N\, \{n\,.\, \lightning(\mathcal{F}(n))\}}\ \text{HT-RAND-EXP}$$



$$\frac{\mathcal{F}(0) + \mathcal{F}(1)}{2} \leqslant \varepsilon$$

Eris in action

$$\{\lightning(1/16)\}$$

$$\begin{aligned}&\mathsf{let}\ l = \mathsf{ref}\ 0\ \mathsf{in}\\ &l \leftarrow (!\,l + \mathsf{rand}\ 3);\\ &l \leftarrow (!\,l + \mathsf{rand}\ 3);\\ &!\,l\end{aligned}$$

$$\{v.v > 0\}$$

By adequacy, the probability of the program returning a non-positive value (error result) is at most $1/16$.

Eris in action

$$\{\lightning(1/16) \;*\; l \mapsto 0\}$$

$$\quad l \leftarrow (!\,l + \mathsf{rand}\,3);$$
$$\quad l \leftarrow (!\,l + \mathsf{rand}\,3);$$
$$\quad !\,l$$

$$\{v.v > 0\}$$

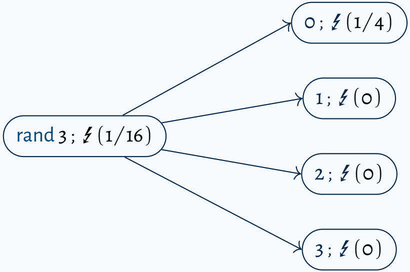
Allocate reference

Eris in action

$$\{↯(\text{if } x = 0 \text{ then } 1/4 \text{ else } 0) \ * \ l \mapsto 0\}$$

$$l \leftarrow (!\,l + x);$$
$$l \leftarrow (!\,l + \text{rand } 3);$$
$$!\,l$$

$$\{v.v > 0\}$$



$$\frac{1/4 + 0 + 0 + 0}{4} \leqslant \frac{1}{16}$$

Eris in action

$$\{\lightning(1/4) \ast l \mapsto 0\}$$
$$l \leftarrow (!\,l + 0);$$
$$l \leftarrow (!\,l + \mathsf{rand}\,3);$$
$$!\,l$$
$$\{v.v > 0\}$$

We continue with the case $x = 0$, otherwise it is trivial

Eris in action

$$\{\text{↯}(1/4) \ * \ l \mapsto 0\}$$
$$l \leftarrow (!\,l + \mathsf{rand}\ 3);$$
$$!\,l$$
$$\{v. v > 0\}$$

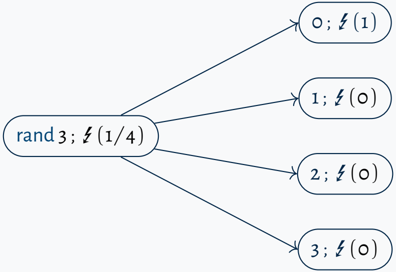
More steps...

Eris in action

$$\{ ↯ (\text{if } x = 0 \text{ then } 1 \text{ else } 0) \ * \ l \mapsto 0 \}$$

$$\begin{array}{l} \quad l \leftarrow (!\, l + x); \\ \quad !\, l \end{array}$$

$$\{ v.\, v > 0 \}$$



$$\frac{1 + 0 + 0 + 0}{4} \leqslant \frac{1}{4}$$

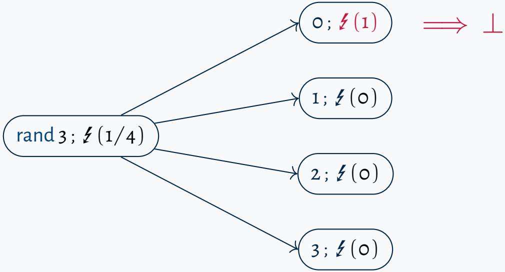
Eris in action

$$\{\lightning(\text{if } x = 0 \text{ then } 1 \text{ else } 0) \;*\; l \mapsto 0\}$$

$$\quad l \leftarrow (!\,l + x);$$

$$\quad !\,l$$

$$\{v. v > 0\}$$



$$\frac{1+0+0+0}{4} \leqslant \frac{1}{4}$$

We can reason about error bounds of sequential **probabilistic** programs.

When it comes to texting girls,

When it comes to texting girls, I don't do it sequentially.

A concurrent probabilistic program

$$
\begin{aligned}
\mathit{conTwoAdd} \triangleq\ & \mathsf{let}\ l = \mathsf{ref}\ 0\ \mathsf{in} \\
& (\mathsf{faa}\ l\ (\mathsf{rand}\ 3) \mid\mid\mid \mathsf{faa}\ l\ (\mathsf{rand}\ 3))\ ; \\
& !\, l
\end{aligned}
$$

A concurrent probabilistic program

$$
\begin{aligned}
\mathit{conTwoAdd} \triangleq\ & \mathsf{let}\ l = \mathsf{ref}\ 0\ \mathsf{in} \\
& (\mathsf{faa}\ l\ (\mathsf{rand}\ 3) \mathrel{|||} \mathsf{faa}\ l\ (\mathsf{rand}\ 3))\ ; \\
& !\, l
\end{aligned}
$$

$\mathsf{faa}\ l\ x$ reads from reference l and increments it by x *atomically*

Introducing CONERIS

Coneris = Concurrency + Eris (+ *much more stuff* we will soon see)

Introducing CONERIS

Coneris = Concurrency + Eris (+ *much more stuff* we will soon see)

Error credit rules are the same, e.g. we still have HT-RAND-EXP.

Introducing CONERIS

Coneris = Concurrency + Eris (+ *much more stuff* we will soon see)

Error credit rules are the same, e.g. we still have HT-RAND-EXP.

Operational semantics of language extended to thread pools – decision of which thread to step is decided by a probabilistic scheduler

Introducing CONERIS

Coneris = Concurrency + Eris (+ *much more stuff* we will soon see)

Error credit rules are the same, e.g. we still have HT-RAND-EXP.

Operational semantics of language extended to thread pools – decision of which thread to step is decided by a probabilistic scheduler

Adequacy: $\{↯(\varepsilon)\}\, e\, \{v.\phi(v)\} \Rightarrow$ ***for all possible schedulers*** s, $\Pr_{\text{exec}\, s,e}[\neg\phi] \leqslant \varepsilon$

Coming up with an invariant

$$I(\gamma_1, \gamma_2) \triangleq$$
$$\exists (s_1\, s_2 : T).\ \boxed{\bullet s_1}^{\gamma_1} * \boxed{\bullet s_2}^{\gamma_2} *$$

$$(\exists n.\ l \mapsto n *$$
$$n = 0 * \text{no\_thread\_added}\ s_1\ s_2 \vee$$
$$\text{one\_thread\_added}\ s_1\ s_2 \vee$$
$$n > 0 * \text{both\_threads\_added}\ s_1\ s_2) *$$

$$(\lightning(1/16) * \text{no\_thread\_sampled}\ s_1\ s_2 \vee$$
$$\lightning(1/4) * \text{one\_thread\_sampled\_zero}\ s_1\ s_2 \vee$$
$$\lightning(0) * \text{at\_least\_one\_thread\_sampled\_nonzero}\ s_1\ s_2 \vee$$
$$\lightning(1))$$

$$\left\{ \boxed{I(\gamma_1, \gamma_2)}^{\iota} * \boxed{\circ S_0}^{\gamma_1} * \boxed{\circ S_0}^{\gamma_2} \right\}$$

$$(\text{faa}\ l\ (\text{rand}\ 3) \mid\mid\mid \text{faa}\ l\ (\text{rand}\ 3))\ ;$$
$$!\, l$$

$$\{v. v > 0\}$$

Coming up with an invariant

$$I(\gamma_1, \gamma_2) \triangleq$$
$$\exists (s_1\, s_2 : T).\ \boxed{\bullet s_1}^{\gamma_1} * \boxed{\bullet s_2}^{\gamma_2} *$$

$$(\exists n.\ l \mapsto n *$$
$$n = 0 * \text{no\_thread\_added}\ s_1\ s_2 \vee$$
$$\text{one\_thread\_added}\ s_1\ s_2 \vee$$
$$n > 0 * \text{both\_threads\_added}\ s_1\ s_2) *$$

$$(\lightning(1/16) * \text{no\_thread\_sampled}\ s_1\ s_2 \vee$$
$$\lightning(1/4) * \text{one\_thread\_sampled\_zero}\ s_1\ s_2 \vee$$
$$\lightning(0) * \text{at\_least\_one\_thread\_sampled\_nonzero}\ s_1\ s_2 \vee$$
$$\lightning(1))$$

$$\left\{ \boxed{I(\gamma_1, \gamma_2)}^{\iota} * \boxed{\circ S_0}^{\gamma_1} * \boxed{\circ S_0}^{\gamma_2} \right\}$$

$$(\text{faa}\ l\ (\text{rand}\ 3) \ ||| \ \text{faa}\ l\ (\text{rand}\ 3));$$
$$!\, l$$

$$\{v. v > 0\}$$

Coming up with an invariant

$$I(\gamma_1, \gamma_2) \triangleq$$
$$\exists (s_1\, s_2 : T).\ \boxed{\bullet s_1}^{\gamma_1} * \boxed{\bullet s_2}^{\gamma_2} *$$

$$(\exists n.\ l \mapsto n *$$
$$n = 0 * \text{no\_thread\_added}\ s_1\ s_2 \lor$$
$$\text{one\_thread\_added}\ s_1\ s_2 \lor$$
$$n > 0 * \text{both\_threads\_added}\ s_1\ s_2) *$$

$$(\text{↯}(1/16) * \text{no\_thread\_sampled}\ s_1\ s_2 \lor$$
$$\text{↯}(1/4) * \text{one\_thread\_sampled\_zero}\ s_1\ s_2 \lor$$
$$\text{↯}(0) * \text{at\_least\_one\_thread\_sampled\_nonzero}\ s_1\ s_2 \lor$$
$$\text{↯}(1))$$

$$\left\{ \boxed{I(\gamma_1, \gamma_2)}^{\iota} * \boxed{\circ S_0}^{\gamma_1} * \boxed{\circ S_0}^{\gamma_2} \right\}$$
$$(\text{faa}\ l\ (\text{rand}\ 3)\ |||\ \text{faa}\ l\ (\text{rand}\ 3));$$
$$!\,l$$
$$\{v. v > 0\}$$

Coming up with an invariant

$$I(\gamma_1, \gamma_2) \triangleq$$
$$\exists (s_1\, s_2 : T).\ \boxed{\bullet s_1}^{\gamma_1} * \boxed{\bullet s_2}^{\gamma_2} *$$

$$(\exists n.\, l \mapsto n *$$
$$n = 0 * \mathsf{no\_thread\_added}\, s_1\, s_2 \lor$$
$$\mathsf{one\_thread\_added}\, s_1\, s_2 \lor$$
$$n > 0 * \mathsf{both\_threads\_added}\, s_1\, s_2) *$$

$$(⚡(1/16) * \mathsf{no\_thread\_sampled}\, s_1\, s_2 \lor$$
$$⚡(1/4) * \mathsf{one\_thread\_sampled\_zero}\, s_1\, s_2 \lor$$
$$⚡(0) * \mathsf{at\_least\_one\_thread\_sampled\_nonzero}\, s_1\, s_2 \lor$$
$$⚡(1))$$

$$\left\{ \boxed{I(\gamma_1, \gamma_2)}^{\iota} * \boxed{\circ S_0}^{\gamma_1} * \boxed{\circ S_0}^{\gamma_2} \right\}$$
$$(\mathsf{faa}\ l\ (\mathsf{rand}\ 3)\ |||\ \mathsf{faa}\ l\ (\mathsf{rand}\ 3))\ ;$$
$$!\, l$$
$$\{v.v > 0\}$$

We can reason about error bounds of ~~sequential~~ **concurrent** probabilistic programs.

Now let's refactor stuff into a randomized concurrent counter

$$
\begin{aligned}
\mathit{conTwoAdd} \triangleq\ & \mathsf{let}\ l = \mathsf{ref}\ 0\ \mathsf{in} \\
& (\mathsf{faa}\ l\ (\mathsf{rand}\ 3) \mid\mid\mid \mathsf{faa}\ l\ (\mathsf{rand}\ 3))\ ; \\
& !\, l
\end{aligned}
$$

Now let's refactor stuff into a randomized concurrent counter

$$\begin{aligned} \mathit{conTwoAdd} \triangleq\ & \mathsf{let}\ l = \mathsf{ref}\ 0\ \mathsf{in} \\ & (\mathsf{faa}\ l\ (\mathsf{rand}\ 3)\ |||\ \mathsf{faa}\ l\ (\mathsf{rand}\ 3))\ ; \\ & !\,l \end{aligned} \quad \Rightarrow$$

$$\begin{aligned} \mathit{createCntr} &\triangleq \lambda\_.\ \mathsf{ref}\ 0 \\ \mathit{readCntr} &\triangleq \lambda l.\ !\,l \\ \mathit{incrCntr} &\triangleq \lambda l.\ \mathsf{faa}\ l\ (\mathsf{rand}\ 3) \end{aligned}$$

$$\begin{aligned} \mathit{conTwoAdd} \triangleq\ & \mathsf{let}\ l = \mathit{createCntr}\ ()\ \mathsf{in} \\ & (\mathit{incrCntr}\ l\ |||\ \mathit{incrCntr}\ l)\ ; \\ & \mathit{readCntr}\ l \end{aligned}$$

Urgh... problem with invariants

$$\left\{ \boxed{I(\gamma_1, \gamma_2)}^{\iota} * \left[\circ S_\circ\right]^{\gamma_1} \right\}$$

$$\mathit{incrCntr}\, l \triangleq (\lambda l.\ \mathsf{faa}\, l\, (\mathsf{rand}\, 3))\, l$$

$$\left\{ \exists n.\ \left[\circ S_2(n)\right]^{\gamma_1} \right\}$$

HT-INV-OPEN

$$\frac{e \text{ atomic} \qquad \{\triangleright I * P\}\, e\, \{\triangleright I * Q\}_{\mathcal{E}}}{\left\{ \boxed{I}^{\iota} * P \right\} e\, \{Q\}_{\mathcal{E} \uplus \{\iota\}}}$$

HOCAP approach

- We adopt the HOCAP approach to parametrize preconditions with view shifts written with $_{\mathcal{E}_1}\Rrightarrow_{\mathcal{E}_2}$ that describes how logical state of the counter changes at linearization point

HOCAP approach

- We adopt the HOCAP approach to parametrize preconditions with view shifts written with ${}_{\mathcal{E}_1}\!\Rrightarrow_{\mathcal{E}_2}$ that describes how logical state of the counter changes at linearization point
- ${}_{\mathcal{E}_1}\!\Rrightarrow_{\mathcal{E}_2} P$ denotes a resource that together with the invariants in $\mathcal{E}_1$, can be updated and split into two disjoint parts: P and one satisfying invariants in $\mathcal{E}_2$

HOCAP specification of *incrcounter*

$$
\begin{array}{l}
\forall \mathcal{E}, \iota, c, Q. \\
\left\{ \begin{array}{l}
\mathit{counter}\, \iota\, c \ast \\
{}_{\mathcal{E}}{\Rrightarrow}_{\emptyset} \left(\begin{array}{l}
\exists \varepsilon, \mathcal{F}.\ ↯(\varepsilon) \ast (\mathbb{E}_{\mathfrak{U}_3}[\mathcal{F}] \leqslant \varepsilon) \ast \\
\qquad \forall x \in \{0..3\}.\ ↯(\mathcal{F}(x)) \twoheadrightarrow \\
\qquad \left({}_{\emptyset}{\Rrightarrow}_{\mathcal{E}} (\forall z.\ \mathit{cauth}\, z \twoheadrightarrow \right. \\
\qquad\quad \left. {\Rrightarrow}_{\mathcal{E}}\, \mathit{cauth}\,(z + x) \ast Q\, \varepsilon\, \mathcal{F}\, x\, z) \right)
\end{array} \right)
\end{array} \right\} \\
\quad \mathit{incrCntr}\, c \triangleq (\lambda l.\ \mathsf{faa}\, l\, (\mathsf{rand}\, 3))\, c \\
\{z.\ \exists \varepsilon, \mathcal{F}, x.\ Q\, \varepsilon\, \mathcal{F}\, x\, z\}_{\mathcal{E} \uplus \{\iota\}}
\end{array}
$$

HOCAP specification of *incrcounter*

$$
\begin{array}{l}
\forall \mathcal{E}, \iota, c, Q. \\
\left\{ \begin{array}{l}
\mathit{counter}\,\iota\,c \ast \\
{}_{\mathcal{E}}{\Rrightarrow}_{\emptyset} \left(\begin{array}{l}
\exists \varepsilon, \mathcal{F}.\ \lightning(\varepsilon) \ast (\mathbb{E}_{\mathfrak{U}_3}[\mathcal{F}] \leqslant \varepsilon) \ast \\
\qquad \forall x \in \{0..3\}.\ \lightning(\mathcal{F}(x)) \mathrel{-\!\!\ast} \\
\qquad \left({}_{\emptyset}{\Rrightarrow}_{\mathcal{E}} (\forall z.\ \mathit{cauth}\, z \mathrel{-\!\!\ast} \right. \\
\qquad\qquad \left. {\Rrightarrow}_{\mathcal{E}}\ \mathit{cauth}\,(z + x) \ast Q\,\varepsilon\,\mathcal{F}\,x\,z) \right)
\end{array} \right)
\end{array} \right\} \\
\quad \mathit{incrCntr}\, c \triangleq (\lambda l.\ \mathsf{faa}\, l\, (\mathsf{rand}\, 3))\, c \\
\{z.\ \exists \varepsilon, \mathcal{F}, x.\ Q\,\varepsilon\,\mathcal{F}\,x\,z\}_{\mathcal{E} \uplus \{\iota\}}
\end{array}
$$

HOCAP specification of *incrcounter*

$$
\begin{array}{l}
\forall \mathcal{E}, \iota, c, Q. \\
\left\{ \begin{array}{l} \mathit{counter}\, \iota\, c \ast \\ {}_{\mathcal{E}}{\Rrightarrow}_{\emptyset} \left(\begin{array}{l} \exists \varepsilon, \mathcal{F}.\ ↯(\varepsilon) \ast (\mathbb{E}_{\mathfrak{U}_3}[\mathcal{F}] \leqslant \varepsilon) \ast \\ \quad \forall x \in \{0..3\}.\ ↯(\mathcal{F}(x)) \mathrel{-\!\!\ast} \\ \quad ({}_{\emptyset}{\Rrightarrow}_{\mathcal{E}} (\forall z.\ \mathit{cauth}\, z \mathrel{-\!\!\ast} \\ \quad\quad {\Rrightarrow}_{\mathcal{E}}\, \mathit{cauth}\,(z + x) \ast Q\, \varepsilon\, \mathcal{F}\, x\, z)) \end{array} \right) \end{array} \right\} \\
\quad \mathit{incrCntr}\, c \triangleq (\lambda l.\ \mathsf{faa}\, l\, (\mathsf{rand}\, 3))\, c \\
\{z.\ \exists \varepsilon, \mathcal{F}, x.\ Q\, \varepsilon\, \mathcal{F}\, x\, z\}_{\mathcal{E} \uplus \{\iota\}}
\end{array}
$$

HOCAP specification of *incrcounter*

$$
\begin{array}{l}
\forall \mathcal{E}, \iota, c, Q. \\
\left\{ \begin{array}{l} \mathit{counter}\,\iota\, c \ast \\
{}_{\mathcal{E}}\!\Rrightarrow_{\emptyset} \left(\begin{array}{l} \exists \varepsilon, \mathcal{F}.\ ↯(\varepsilon) \ast (\mathbb{E}_{\mathfrak{U}3}[\mathcal{F}] \leqslant \varepsilon) \ast \\
\quad \forall x \in \{0..3\}.\ ↯(\mathcal{F}(x)) \mathrel{-\!\ast} \\
\quad \big({}_{\emptyset}\!\Rrightarrow_{\mathcal{E}} (\forall z.\ \mathit{cauth}\, z \mathrel{-\!\ast} \\
\qquad \Rrightarrow_{\mathcal{E}} \mathit{cauth}\,(z + x) \ast Q\,\varepsilon\,\mathcal{F}\,x\,z)\big) \end{array} \right) \end{array} \right\} \\
\quad \mathit{incrCntr}\, c \triangleq (\lambda l.\ \mathsf{faa}\, l\, (\mathsf{rand}\, 3))\, c \\
\{z.\ \exists \varepsilon, \mathcal{F}, x.\ Q\,\varepsilon\,\mathcal{F}\,x\,z\}_{\mathcal{E} \uplus \{\iota\}}
\end{array}
$$

HOCAP specification of *incrcounter*

$$\forall \mathcal{E}, \iota, c, Q.$$

$$\left\{ \begin{array}{l} \mathit{counter}\,\iota\,c \ast \\ {}_{\mathcal{E}}{\Rrightarrow}_{\emptyset} \left(\begin{array}{l} \exists \varepsilon, \mathcal{F}.\ ↯(\varepsilon) \ast (\mathbb{E}_{\mathcal{U}_3}[\mathcal{F}] \leqslant \varepsilon) \ast \\ \qquad \forall x \in \{0..3\}.\ ↯(\mathcal{F}(x)) \twoheadrightarrow \\ \qquad ({}_{\emptyset}{\Rrightarrow}_{\mathcal{E}} (\forall z.\ \mathit{cauth}\,z \twoheadrightarrow \\ \qquad\quad {\Rrightarrow}_{\mathcal{E}}\, \mathit{cauth}\,(z+x) \ast Q\,\varepsilon\,\mathcal{F}\,x\,z)) \end{array} \right) \end{array} \right\}$$

$$\mathit{incrCntr}\,c \triangleq (\lambda l.\ \mathsf{faa}\,l\,(\mathsf{rand}\,3))\,c$$

$$\{z.\ \exists \varepsilon, \mathcal{F}, x.\ Q\,\varepsilon\,\mathcal{F}\,x\,z\}_{\mathcal{E} \uplus \{\iota\}}$$

HOCAP specification of *incrcounter*

$$
\begin{aligned}
&\forall \mathcal{E}, \iota, c, Q.\\
&\left\{ \begin{array}{l} \mathit{counter}\,\iota\,c\ * \\ {}_{\mathcal{E}}\!\Rrightarrow_{\emptyset} \left(\begin{array}{l} \exists \varepsilon, \mathcal{F}.\ \lightning(\varepsilon) \ * \ (\mathbb{E}_{\mathfrak{U}_3}[\mathcal{F}] \leqslant \varepsilon) \ * \\ \qquad \forall x \in \{0..3\}.\ \lightning(\mathcal{F}(x)) \ -\!\!* \\ \qquad ({}_{\emptyset}\!\Rrightarrow_{\mathcal{E}} (\forall z.\ \mathit{cauth}\,z \ -\!\!* \\ \qquad\quad \Rrightarrow_{\mathcal{E}} \mathit{cauth}\,(z + x) \ * \ Q\,\varepsilon\,\mathcal{F}\,x\,z)) \end{array} \right) \end{array} \right\}\\
&\quad \mathit{incrCntr}\,c \triangleq (\lambda l.\ \mathsf{faa}\,l\,(\mathsf{rand}\,3))\,c\\
&\{z.\ \exists \varepsilon, \mathcal{F}, x.\ Q\,\varepsilon\,\mathcal{F}\,x\,z\}_{\mathcal{E} \uplus \{\iota\}}
\end{aligned}
$$

We can reason about error bounds of
concurrent probabilistic programs **modularly**.

Problem 1

$$incrCntr_1 \triangleq \lambda l.\; \mathsf{faa}\; l\, (\mathsf{rand}\; 3)$$

Problem 1

$$incrCntr_1 \triangleq \lambda l.\ \mathsf{faa}\ l\ (\mathsf{rand}\ 3)$$

$$incrCntr_2 \triangleq \lambda l.\ \mathsf{faa}\ l\ ((\mathsf{rand}\ 1) * 2 + \mathsf{rand}\ 1)$$

Problem 1

$$incrCntr_1 \triangleq \lambda l.\ \mathsf{faa}\, l\, (\mathsf{rand}\, 3)$$

$$incrCntr_2 \triangleq \lambda l.\ \mathsf{faa}\, l\, ((\mathsf{rand}\, 1) * 2 + \mathsf{rand}\, 1)$$

$$\begin{aligned} &incrCntr_3 \triangleq \\ &\quad \mathsf{rec}\, f\ l = \mathsf{let}\, x = \mathsf{rand}\, 4\ \mathsf{in} \\ &\qquad \mathsf{if}\, x < 4\ \mathsf{then}\ \mathsf{faa}\, l\, x\ \mathsf{else}\, f l \end{aligned}$$

Problem 1

$$incrCntr_1 \triangleq \lambda l.\ \mathsf{faa}\, l\, (\mathsf{rand}\, 3)$$

$$incrCntr_2 \triangleq \lambda l.\ \mathsf{faa}\, l\, ((\mathsf{rand}\, 1) * 2 + \mathsf{rand}\, 1)$$

$$\begin{aligned} &incrCntr_3 \triangleq \\ &\quad \mathsf{rec}\, f\, l = \mathsf{let}\, x = \mathsf{rand}\, 4\, \mathsf{in} \\ &\qquad \mathsf{if}\, x < 4\, \mathsf{then}\ \mathsf{faa}\, l\, x\, \mathsf{else}\, f l \end{aligned}$$

The probabilistic sampling operations in $incrCntr_2$ and $incrCntr_3$ are not atomic.

Problem 2

$$
\begin{aligned}
\mathit{twoIncr}\,\_ \triangleq\ & \mathsf{let}\ c = \mathit{createCntr}\,()\ \mathsf{in} \\
& \mathit{incrCntr}\ c; \\
& \mathsf{let}\ v_1 = \mathit{readCntr}\ c\ \mathsf{in} \\
& \mathit{incrCntr}\ c; \\
& \mathsf{let}\ v_2 = \mathit{readCntr}\ c - v_1\ \mathsf{in} \\
& 4 * v_1 + v_2
\end{aligned}
$$

Problem 2

$$\begin{aligned} \mathit{twoIncr}\ \_ \triangleq\ & \mathsf{let}\ c = \mathit{createCntr}\ ()\ \mathsf{in} \\ & \mathit{incrCntr}\ c; \\ & \mathsf{let}\ v_1 = \mathit{readCntr}\ c\ \mathsf{in} \\ & \mathit{incrCntr}\ c; \\ & \mathsf{let}\ v_2 = \mathit{readCntr}\ c - v_1\ \mathsf{in} \\ & 4 * v_1 + v_2 \end{aligned}$$

$$\forall \iota\, Q\, \mathcal{E}.\ \left\{ \begin{array}{l} ({}_{\mathcal{E}}{\Rrightarrow}_{\emptyset}\ \exists \varepsilon\, \mathcal{F}. \\ \quad ↯(\varepsilon) * (\mathbb{E}_{\mathfrak{U}15}[\mathcal{F}] \leqslant \varepsilon) * \\ \quad (\forall x.\ ↯(\mathcal{F}(x)) \mathrel{-\!\!*} {}_{\emptyset}{\Rrightarrow}_{\mathcal{E}}\ Q\, \varepsilon\, \mathcal{F}\, x)) \end{array} \right\}$$

$$\mathit{twoIncr}\ ()\ \{z. \exists \varepsilon\, \mathcal{F}.\ Q\, \varepsilon\, \mathcal{F}\, z\}_{\mathcal{E} \uplus \{\iota\}}$$

twoIncr acts like an atomic rand 15

Problem 2

$$
\begin{aligned}
\mathit{twoIncr\_} \triangleq\ & \mathsf{let}\ c = \mathit{createCntr}\,()\ \mathsf{in} \\
& \mathit{incrCntr}\ c; \\
& \mathsf{let}\ v_1 = \mathit{readCntr}\ c\ \mathsf{in} \\
& \mathit{incrCntr}\ c; \\
& \mathsf{let}\ v_2 = \mathit{readCntr}\ c - v_1\ \mathsf{in} \\
& 4 * v_1 + v_2
\end{aligned}
$$

$$
\forall \iota\, Q\, \mathcal{E}.\ \left\{ \begin{array}{l} ({}_{\mathcal{E}}{\Rrightarrow}_{\emptyset}\ \exists \varepsilon\, \mathcal{F}. \\ \lightning(\varepsilon) * (\mathbb{E}_{\mathfrak{U}15}[\mathcal{F}] \leq \varepsilon) * \\ (\forall x.\ \lightning(\mathcal{F}(x)) \mathrel{-\!\!*} {}_{\emptyset}{\Rrightarrow}_{\mathcal{E}}\ Q\, \varepsilon\, \mathcal{F}\, x)) \end{array} \right\}
$$

$$
\mathit{twoIncr}\,()\ \{z. \exists \varepsilon\, \mathcal{F}.\ Q\, \varepsilon\, \mathcal{F}\, z\}_{\mathcal{E} \uplus \{\iota\}}
$$

twoIncr acts like an atomic rand 15

We cannot combine the probabilistic part of the two view shifts of *incrCntr* into one single one

We need to capture "randomized logical atomicity"

Key idea: We capture a notion of *randomized logical atomicity*

We need to capture "randomized logical atomicity"

Key idea: We capture a notion of *randomized logical atomicity*

Intuitively describes how modules commit to some probabilistic choice in a logically atomic manner

We need to capture "randomized logical atomicity"

Key idea: We capture a notion of *randomized logical atomicity*

Intuitively describes how modules commit to some probabilistic choice in a logically atomic manner

Two new ingredients:

We need to capture "randomized logical atomicity"

Key idea: We capture a notion of *randomized logical atomicity*

Intuitively describes how modules commit to some probabilistic choice in a logically atomic manner

Two new ingredients:

1. Presampling tapes (first introduced in Clutch)

We need to capture “randomized logical atomicity”

Key idea: We capture a notion of *randomized logical atomicity*

Intuitively describes how modules commit to some probabilistic choice in a logically atomic manner

Two new ingredients:

1. Presampling tapes (first introduced in Clutch)
2. A novel probabilistic update modality

We need to capture “randomized logical atomicity”

Key idea: We capture a notion of *randomized logical atomicity*

Intuitively describes how modules commit to some probabilistic choice in a logically atomic manner

Two new ingredients:

1. Presampling tapes (first introduced in Clutch)
2. A novel probabilistic update modality (not the same modality from Lohse et al ☺)

Presampling tapes I

$$\sigma \in \mathit{State} \triangleq (\mathit{Loc} \xrightarrow{\text{fin}} \mathit{Val}) \times (\mathit{Label} \xrightarrow{\text{fin}} \mathit{Tape})$$
$$t \in \mathit{Tape} \triangleq \{(N, \vec{n}) \mid N \in \mathbb{N} \wedge \vec{n} \in \mathbb{N}^*_{\leqslant N}\}$$

Presampling tapes I

$$\sigma \in \mathit{State} \triangleq (\mathit{Loc} \xrightarrow{\text{fin}} \mathit{Val}) \times (\mathit{Label} \xrightarrow{\text{fin}} \mathit{Tape})$$

$$t \in \mathit{Tape} \triangleq \{(N, \vec{n}) \mid N \in \mathbb{N} \wedge \vec{n} \in \mathbb{N}^{*}_{\leqslant N}\}$$

$$\text{step}(\text{tape}\ N, \sigma) = \text{ret}(\kappa, \sigma[\kappa := (N, \epsilon)], []) \quad \text{(where } \kappa \text{ is fresh w.r.t. } \sigma\text{)}$$

$$\text{step}(\text{rand}\ \kappa\ N, \sigma) = \lambda(n, \sigma, []) \,.\, \tfrac{1}{N+1} \quad \text{if } \sigma[\kappa] = (N, \epsilon) \wedge n \in \{0, \ldots, N\} \quad \text{and} \quad 0 \text{ otherwise}$$

Presampling tapes I

$$\sigma \in \mathit{State} \triangleq (\mathit{Loc} \xrightharpoonup{\text{fin}} \mathit{Val}) \times (\mathit{Label} \xrightharpoonup{\text{fin}} \mathit{Tape})$$

$$t \in \mathit{Tape} \triangleq \{(N, \vec{n}) \mid N \in \mathbb{N} \wedge \vec{n} \in \mathbb{N}^*_{\leqslant N}\}$$

$$\text{step}(\text{tape}\ N, \sigma) = \text{ret}(\kappa, \sigma[\kappa := (N, \epsilon)], []) \quad \text{(where } \kappa \text{ is fresh w.r.t. } \sigma\text{)}$$

$$\text{step}(\text{rand}\ \kappa\ N, \sigma) = \lambda(n, \sigma, []) .\ \tfrac{1}{N+1} \quad \text{if } \sigma[\kappa] = (N, \epsilon) \wedge n \in \{0, \ldots, N\} \quad \text{and} \quad 0 \text{ otherwise}$$

There are no steps in operational semantics to *write* contents into a tape!

Rewriting randomized concurrent counter module

$$\begin{aligned} \mathit{createCntr} &\triangleq \lambda\_.\ \ \mathsf{ref}\ 0 \\ \mathit{readCntr} &\triangleq \lambda l.\ \ !\,l \\ \mathit{incrCntr} &\triangleq \lambda l.\ \ \mathsf{faa}\ l\ (\mathsf{rand}\ 3) \end{aligned}$$

$$\begin{aligned} \mathit{conTwoAdd} \triangleq\ & \mathsf{let}\ l = \mathit{createCntr}\,()\ \mathsf{in} \\ & (\mathit{incrCntr}\ l \mid\mid\mid \mathit{incrCntr}\ l)\ ; \\ & \mathit{readCntr}\ l \end{aligned}$$

Rewriting randomized concurrent counter module

$$\begin{aligned} \textit{createCntr} &\triangleq \lambda\_.\ \mathsf{ref}\,0 \\ \textit{readCntr} &\triangleq \lambda l.\ !\,l \\ \textit{incrCntr} &\triangleq \lambda l.\ \mathsf{faa}\,l\,(\mathsf{rand}\,3) \end{aligned}$$

$$\begin{aligned} \textit{conTwoAdd} \triangleq\ & \mathsf{let}\ l = \textit{createCntr}\,()\ \mathsf{in} \\ & (\textit{incrCntr}\,l \mid\mid\mid \textit{incrCntr}\,l)\,; \\ & \textit{readCntr}\,l \end{aligned}$$

$$\Rightarrow$$

$$\begin{aligned} \textit{createCntr} &\triangleq \lambda\_.\ \mathsf{ref}\,0 \\ \textit{readCntr} &\triangleq \lambda l.\ !\,l \\ \textit{createCtape} &\triangleq \lambda().\ \mathsf{tape}\,3 \\ \textit{incrCntr} &\triangleq \lambda l\,\kappa.\ \mathsf{faa}\,l\,(\mathsf{rand}\,\kappa\,3) \end{aligned}$$

$$\begin{aligned} \textit{conTwoAdd} \triangleq\ & \mathsf{let}\ c = \textit{createCntr}()\ \mathsf{in} \\ & \left(\begin{array}{l} \mathsf{let}\ \kappa = \textit{createCtape}()\ \mathsf{in} \\ \textit{incrCntr}\,c\,\kappa \end{array} \mid\mid\mid \ \ldots \right); \\ & \textit{readCntr}\,c \end{aligned}$$

Presampling tapes II

HT-ALLOC-TAPE

$$\frac{}{\{\text{True}\}\ \texttt{tape}\ N\ \{\kappa.\ \kappa \hookrightarrow (N, \epsilon)\}}$$

Presampling tapes II

HT-ALLOC-TAPE

$$\frac{}{\{\mathsf{True}\}\ \mathsf{tape}\ N\,\{\kappa.\ \kappa \hookrightarrow (N, \epsilon)\}}$$

HT-RAND-TAPE

$$\frac{}{\{\kappa \hookrightarrow (N, n \cdot \vec{n})\}\ \mathsf{rand}\ \kappa\ N\,\{x.\ x = n * \kappa \hookrightarrow (N, \vec{n})\}}$$

Presampling tapes II

HT-ALLOC-TAPE

$$\frac{}{\{\mathsf{True}\}\ \mathsf{tape}\ N\, \{\kappa.\, \kappa \hookrightarrow (N, \epsilon)\}}$$

HT-RAND-TAPE

$$\frac{}{\{\kappa \hookrightarrow (N, n \cdot \vec{n})\}\ \mathsf{rand}\ \kappa\, N\, \{x.\, x = n * \kappa \hookrightarrow (N, \vec{n})\}}$$

Wait?! How do you presample onto a tape in the logic?

Presampling tapes II

HT-ALLOC-TAPE

$$\frac{}{\{\mathsf{True}\}\ \mathtt{tape}\ N\,\{\kappa.\ \kappa \hookrightarrow (N, \epsilon)\}}$$

HT-RAND-TAPE

$$\frac{}{\{\kappa \hookrightarrow (N, n \cdot \vec{n})\}\ \mathtt{rand}\ \kappa\, N\,\{x.\ x = n * \kappa \hookrightarrow (N, \vec{n})\}}$$

Wait?! How do you presample onto a tape in the logic?

We do it with the probabilistic update modality!

Rules of the probabilistic update modality

${}_{\mathcal{E}_1}\Rrightarrow_{\mathcal{E}_2} P$ denotes a resource together with the invariants in $\mathcal{E}_1$, can perform *a randomized logical atomic* operation and split into two parts: P and one satisfying invariants in $\mathcal{E}_2$

Rules of the probabilistic update modality

${}_{\mathcal{E}_1}\!\rightsquigarrow\!\rightsquigarrow_{\mathcal{E}_2} P$ denotes a resource together with the invariants in $\mathcal{E}_1$, can perform *a randomized logical atomic* operation and split into two parts: P and one satisfying invariants in $\mathcal{E}_2$

PUPD-RET

$$\frac{P}{\rightsquigarrow\!\rightsquigarrow_{\mathcal{E}} P}$$

Rules of the probabilistic update modality

${}_{\mathcal{E}_1}\overset{\leadsto}{\leadsto}{}_{\mathcal{E}_2}\, P$ denotes a resource together with the invariants in $\mathcal{E}_1$, can perform *a randomized logical atomic* operation and split into two parts: P and one satisfying invariants in $\mathcal{E}_2$

PUPD-RET

$$\frac{P}{\overset{\leadsto}{\leadsto}_{\mathcal{E}}\, P}$$

PUPD-BIND

$$\frac{{}_{\mathcal{E}_1}\overset{\leadsto}{\leadsto}{}_{\mathcal{E}_2}\, P \qquad P \twoheadrightarrow {}_{\mathcal{E}_2}\overset{\leadsto}{\leadsto}{}_{\mathcal{E}_3}\, Q}{{}_{\mathcal{E}_1}\overset{\leadsto}{\leadsto}{}_{\mathcal{E}_3}\, Q}$$

Rules of the probabilistic update modality

${}_{\mathcal{E}_1}\!\mid\!\!\rightsquigarrow_{\mathcal{E}_2} P$ denotes a resource together with the invariants in $\mathcal{E}_1$, can perform *a randomized logical atomic* operation and split into two parts: P and one satisfying invariants in $\mathcal{E}_2$

PUPD-RET

$$\frac{P}{\mid\!\!\rightsquigarrow_{\mathcal{E}} P}$$

PUPD-BIND

$$\frac{{}_{\mathcal{E}_1}\!\mid\!\!\rightsquigarrow_{\mathcal{E}_2} P \qquad P \mathrel{-\!\!*} {}_{\mathcal{E}_2}\!\mid\!\!\rightsquigarrow_{\mathcal{E}_3} Q}{{}_{\mathcal{E}_1}\!\mid\!\!\rightsquigarrow_{\mathcal{E}_3} Q}$$

PUPD-FUPD

$$\frac{{}_{\mathcal{E}_1}\!\mid\!\!\Rrightarrow_{\mathcal{E}_2} P}{{}_{\mathcal{E}_1}\!\mid\!\!\rightsquigarrow_{\mathcal{E}_2} P}$$

Rules of the probabilistic update modality

$_{\mathcal{E}_1}\rightsquigarrow_{\mathcal{E}_2} P$ denotes a resource together with the invariants in $\mathcal{E}_1$, can perform *a randomized logical atomic* operation and split into two parts: P and one satisfying invariants in $\mathcal{E}_2$

$$\textsc{pupd-ret}\quad \frac{P}{\rightsquigarrow_{\mathcal{E}} P}$$

$$\textsc{pupd-bind}\quad \frac{_{\mathcal{E}_1}\rightsquigarrow_{\mathcal{E}_2} P \qquad P \mathrel{-\!\!*} {}_{\mathcal{E}_2}\rightsquigarrow_{\mathcal{E}_3} Q}{_{\mathcal{E}_1}\rightsquigarrow_{\mathcal{E}_3} Q}$$

$$\textsc{pupd-fupd}\quad \frac{_{\mathcal{E}_1}\Rrightarrow_{\mathcal{E}_2} P}{_{\mathcal{E}_1}\rightsquigarrow_{\mathcal{E}_2} P}$$

$$\textsc{pupd-presample-exp}\quad \frac{\kappa \hookrightarrow (N, \vec{n}) \qquad \lightning(\varepsilon) \qquad \mathbb{E}_{\mathfrak{U}N}[\mathcal{F}] \leq \varepsilon}{\rightsquigarrow_{\mathcal{E}} (\exists n.\, \kappa \hookrightarrow (N, \vec{n} \cdot n) * \lightning(\mathcal{F}(n)))}$$

Rules of the probabilistic update modality

${}_{\mathcal{E}_1}{\rightsquigarrow\!\!\!\!\!\rightsquigarrow}_{\mathcal{E}_2}\, P$ denotes a resource together with the invariants in $\mathcal{E}_1$, can perform *a randomized logical atomic* operation and split into two parts: P and one satisfying invariants in $\mathcal{E}_2$

$$\textsc{pupd-ret}\quad \frac{P}{{\rightsquigarrow\!\!\!\!\!\rightsquigarrow}_{\mathcal{E}}\, P}$$

$$\textsc{pupd-bind}\quad \frac{{}_{\mathcal{E}_1}{\rightsquigarrow\!\!\!\!\!\rightsquigarrow}_{\mathcal{E}_2}\, P \qquad P \mathrel{-\!\!*} {}_{\mathcal{E}_2}{\rightsquigarrow\!\!\!\!\!\rightsquigarrow}_{\mathcal{E}_3}\, Q}{{}_{\mathcal{E}_1}{\rightsquigarrow\!\!\!\!\!\rightsquigarrow}_{\mathcal{E}_3}\, Q}$$

$$\textsc{pupd-fupd}\quad \frac{{}_{\mathcal{E}_1}{\Rrightarrow}_{\mathcal{E}_2}\, P}{{}_{\mathcal{E}_1}{\rightsquigarrow\!\!\!\!\!\rightsquigarrow}_{\mathcal{E}_2}\, P}$$

$$\textsc{pupd-presample-exp}\quad \frac{\kappa \hookrightarrow (N, \vec{n}) \qquad ↯(\varepsilon) \qquad \mathbb{E}_{\mathfrak{U}N}[\mathcal{F}] \leqslant \varepsilon}{{\rightsquigarrow\!\!\!\!\!\rightsquigarrow}_{\mathcal{E}}\, (\exists n.\, \kappa \hookrightarrow (N, \vec{n} \cdot n) * ↯(\mathcal{F}(n)))}$$

Rules of the probabilistic update modality

${}_{\mathcal{E}_1}\!\rightsquigarrow_{\mathcal{E}_2} P$ denotes a resource together with the invariants in $\mathcal{E}_1$, can perform *a randomized logical atomic* operation and split into two parts: P and one satisfying invariants in $\mathcal{E}_2$

PUPD-RET

$$\frac{P}{\rightsquigarrow_{\mathcal{E}} P}$$

PUPD-BIND

$$\frac{{}_{\mathcal{E}_1}\!\rightsquigarrow_{\mathcal{E}_2} P \qquad P \twoheadrightarrow {}_{\mathcal{E}_2}\!\rightsquigarrow_{\mathcal{E}_3} Q}{{}_{\mathcal{E}_1}\!\rightsquigarrow_{\mathcal{E}_3} Q}$$

PUPD-FUPD

$$\frac{{}_{\mathcal{E}_1}\!\Rrightarrow_{\mathcal{E}_2} P}{{}_{\mathcal{E}_1}\!\rightsquigarrow_{\mathcal{E}_2} P}$$

PUPD-PRESAMPLE-EXP

$$\frac{\kappa \hookrightarrow (N, \vec{n}) \qquad ⚡(\varepsilon) \qquad \mathbb{E}_{\mathfrak{U}N}[\mathcal{F}] \leqslant \varepsilon}{\rightsquigarrow_{\mathcal{E}} (\exists n.\, \kappa \hookrightarrow (N, \vec{n} \cdot n) * ⚡(\mathcal{F}(n)))}$$

Rules of the probabilistic update modality

${}_{\mathcal{E}_1}{\rightsquigarrow}_{\mathcal{E}_2} P$ denotes a resource together with the invariants in $\mathcal{E}_1$, can perform *a randomized logical atomic* operation and split into two parts: P and one satisfying invariants in $\mathcal{E}_2$

PUPD-RET

$$\frac{P}{{\rightsquigarrow}_{\mathcal{E}}\, P}$$

PUPD-BIND

$$\frac{{}_{\mathcal{E}_1}{\rightsquigarrow}_{\mathcal{E}_2} P \qquad P \mathrel{-\!\!*} {}_{\mathcal{E}_2}{\rightsquigarrow}_{\mathcal{E}_3} Q}{{}_{\mathcal{E}_1}{\rightsquigarrow}_{\mathcal{E}_3} Q}$$

PUPD-FUPD

$$\frac{{}_{\mathcal{E}_1}{\mathrel{|}\!\!\Rrightarrow}_{\mathcal{E}_2} P}{{}_{\mathcal{E}_1}{\rightsquigarrow}_{\mathcal{E}_2} P}$$

PUPD-PRESAMPLE-EXP

$$\frac{\kappa \hookrightarrow (N, \vec{n}) \qquad ↯(\varepsilon) \qquad \mathbb{E}_{\mathfrak{U}N}[\mathcal{F}] \leq \varepsilon}{{\rightsquigarrow}_{\mathcal{E}} (\exists n.\ \kappa \hookrightarrow (N, \vec{n} \cdot n) \ast ↯(\mathcal{F}(n)))}$$

Rules of the probabilistic update modality

${}_{\mathcal{E}_1}\!\rightsquigarrow_{\mathcal{E}_2} P$ denotes a resource together with the invariants in $\mathcal{E}_1$, can perform *a randomized logical atomic* operation and split into two parts: P and one satisfying invariants in $\mathcal{E}_2$

PUPD-RET
$$\frac{P}{\rightsquigarrow_{\mathcal{E}} P}$$

PUPD-BIND
$$\frac{{}_{\mathcal{E}_1}\!\rightsquigarrow_{\mathcal{E}_2} P \qquad P \twoheadrightarrow {}_{\mathcal{E}_2}\!\rightsquigarrow_{\mathcal{E}_3} Q}{{}_{\mathcal{E}_1}\!\rightsquigarrow_{\mathcal{E}_3} Q}$$

PUPD-FUPD
$$\frac{{}_{\mathcal{E}_1}\!\Rrightarrow_{\mathcal{E}_2} P}{{}_{\mathcal{E}_1}\!\rightsquigarrow_{\mathcal{E}_2} P}$$

PUPD-PRESAMPLE-EXP
$$\frac{\kappa \hookrightarrow (N, \vec{n}) \qquad \lightning(\varepsilon) \qquad \mathbb{E}_{\mathfrak{U}N}[\mathcal{F}] \leqslant \varepsilon}{\rightsquigarrow_{\mathcal{E}} (\exists n.\, \kappa \hookrightarrow (N, \vec{n} \cdot n) * \lightning(\mathcal{F}(n)))}$$

New specification that exposes presampling

$$\forall \iota, c.\ \{\mathit{counter}\ \iota\ c\}\ \mathit{createCtape}(\,)\ \left\{\kappa.\ \mathit{ctape}\ \kappa\ \epsilon\right\}$$

New specification that exposes presampling

$$\forall \iota, c.\ \{\mathit{counter}\ \iota\ c\}\ \mathit{createCtape}(\,)\ \{\kappa.\ \mathit{ctape}\ \kappa\ \epsilon\}$$

$$\begin{array}{l} \forall \mathcal{E}, \iota, c, n, \vec{n}, Q. \\ \left\{ \begin{array}{l} \mathit{counter}\ \iota\ c * \mathit{ctape}\ \kappa\ (n \cdot \vec{n}) * \\ \quad (\forall z.\ \mathit{cauth}\ z \mathrel{-\!\!*} \Rrightarrow_{\mathcal{E}} \mathit{cauth}\ (z + n) * Q\ z) \end{array} \right\} \\ \quad \mathit{incrCntr}\ c\ \kappa \\ \{z.\mathit{ctape}\ \kappa\ \vec{n} * Q\ z\}_{\mathcal{E} \uplus \{\iota\}} \end{array}$$

New specification that exposes presampling

$$\forall \iota, c.\ \{\mathit{counter}\ \iota\ c\}\ \mathit{createCtape}(\,)\ \{\kappa.\ \mathit{ctape}\ \kappa\ \epsilon\}$$

$$\forall \mathcal{E}, \iota, c, n, \vec{n}, Q.$$

$$\left\{ \begin{array}{l} \mathit{counter}\ \iota\ c * \mathit{ctape}\ \kappa\ (n \cdot \vec{n}) * \\ (\forall z.\ \mathit{cauth}\ z \mathrel{-\!\!*} \Rrightarrow_{\mathcal{E}} \mathit{cauth}\ (z + n) * Q\ z) \end{array} \right\}$$

$$\mathit{incrCntr}\ c\ \kappa$$

$$\{z. \mathit{ctape}\ \kappa\ \vec{n} * Q\ z\}_{\mathcal{E} \uplus \{\iota\}}$$

$$\forall \mathcal{E}, \varepsilon, \mathcal{F}, \vec{n}, \kappa.$$

$$(\lightning(\varepsilon) * (\mathbb{E}_{\mathcal{U}_3}[\mathcal{F}] \leqslant \varepsilon) * \mathit{ctape}\ \kappa\ \vec{n} \mathrel{-\!\!*} \rightsquigarrow_{\mathcal{E}} \exists n \in \{0..3\}.\ \lightning(\mathcal{F}(n)) * \mathit{ctape}\ \kappa\ (\vec{n} \cdot [n]))$$

Take home message

Motto: We can reason about error bounds of concurrent probabilistic programs

Take home message

Motto: We can reason about error bounds of concurrent probabilistic programs modularly

Take home message

Motto: We can reason about error bounds of concurrent probabilistic programs modularly by capturing randomized logical atomicity.

Take home message

Motto: We can reason about error bounds of concurrent probabilistic programs modularly by capturing randomized logical atomicity.

Other stuff in the paper:

- How to prove that all three implementations satisfy this specification? (Not trivial)

Take home message

Motto: We can reason about error bounds of concurrent probabilistic programs modularly by capturing randomized logical atomicity.

Other stuff in the paper:

- How to prove that all three implementations satisfy this specification? (Not trivial)
- How exactly do we use the specification to prove *conTwoAdd*

Take home message

Motto: We can reason about error bounds of concurrent probabilistic programs modularly by capturing randomized logical atomicity.

Other stuff in the paper:

- How to prove that all three implementations satisfy this specification? (Not trivial)
- How exactly do we use the specification to prove *conTwoAdd*
- Other examples: thread safe hash function, concurrent implementation of bloom filter...

Take home message

Motto: We can reason about error bounds of concurrent probabilistic programs modularly by capturing randomized logical atomicity.

Other stuff in the paper:

- How to prove that all three implementations satisfy this specification? (Not trivial)
- How exactly do we use the specification to prove *conTwoAdd*
- Other examples: thread safe hash function, concurrent implementation of bloom filter...
- Definition of weakestpre + probabilistic update modality

HOCAP specification of *createCntr* and *readCntr*

$\{\mathsf{True}\}\ \mathit{createCntr}(\,)\ \{c.\ \exists\iota.\ \mathit{counter}\ \iota\ c * \mathit{cfrag}\ 1\ 0\}$

$\forall \mathcal{E}, \iota, c, Q.$

$\quad \{\mathit{counter}\ \iota\ c * (\forall z.\ \mathit{cauth}\ z \twoheadrightarrow\!* \ \Rrightarrow_{\mathcal{E}} \mathit{cauth}\ z * Q\ z)\}$

$\qquad \mathit{readCntr}\ c$

$\quad \{z.\ Q\ z\}_{\mathcal{E} \uplus \{\iota\}}$

- *counter* ι c captures the fact that c is a counter with invariant name ι

HOCAP specification of *createCntr* and *readCntr*

$$\{\mathsf{True}\}\ \mathit{createCntr}()\ \{c.\ \exists\iota.\ \mathit{counter}\,\iota\,c * \mathit{cfrag}\,1\,0\}$$

$$\begin{array}{l}\forall\mathcal{E},\iota,c,Q.\\ \quad\{\mathit{counter}\,\iota\,c * (\forall z.\ \mathit{cauth}\,z \mathrel{-\!\!*} \Rrightarrow_{\mathcal{E}} \mathit{cauth}\,z * Q\,z)\}\\ \qquad \mathit{readCntr}\,c\\ \quad\{z.\ Q\,z\}_{\mathcal{E}\uplus\{\iota\}}\end{array}$$

- *counter* ι c captures the fact that c is a counter with invariant name ι
- *cauth* and *cfrag* provides *authoritative* and *fragmental* views of the counter

HOCAP specification of *createCntr* and *readCntr*

$\{\mathsf{True}\}\ \mathit{createCntr}()\ \{c.\ \exists\iota.\ \mathit{counter}\ \iota\ c * \mathit{cfrag}\ 1\ 0\}$

$$\forall\mathcal{E}, \iota, c, Q.$$
$$\{\mathit{counter}\ \iota\ c * (\forall z.\ \mathit{cauth}\ z \mathrel{-\!\!*} \Rrightarrow_{\mathcal{E}} \mathit{cauth}\ z * Q\ z)\}$$
$$\mathit{readCntr}\ c$$
$$\{z.\ Q\ z\}_{\mathcal{E} \uplus \{\iota\}}$$

- *counter* ι c captures the fact that c is a counter with invariant name ι
- *cauth* and *cfrag* provides *authoritative* and *fragmental* views of the counter
- Many conditions of these abstract predicates not shown, e.g. $\mathit{cauth}\ n * \mathit{cfrag}\ q\ m \vdash \Rrightarrow_{\mathcal{E}} \mathit{cauth}\ (n+p) * \mathit{cfrag}\ q\ (m+p)$

Second attempt in verifying *conTwoAdd*

$$\{ \text{↯}(1/16) \ast l \mapsto 0\}$$

$$(\text{faa}\ l\ (\text{rand}\ 3) \mid\mid\mid \text{faa}\ l\ (\text{rand}\ 3))\ ;$$

$$!\,l$$

$$\{v.v > 0\}$$

Where we left off...

Second attempt in verifying *conTwoAdd*

$$\left\{ \boxed{I(\gamma_1, \gamma_2)}^{\iota} * \boxed{\circ S_0}^{\gamma_1} * \boxed{\circ S_0}^{\gamma_2} \right\}$$

$$\quad (\mathsf{faa}\ l\ (\mathsf{rand}\ 3) \ ||| \ \mathsf{faa}\ l\ (\mathsf{rand}\ 3))\ ;$$

$$\quad !\, l$$

$$\{v. v > 0\}$$

Allocating invariants and resources:

$$⚡(1/16) * l \mapsto 0 -\!\!* \;\Rrightarrow \exists \gamma_1\, \gamma_2.\, I(\gamma_1, \gamma_2) * \boxed{\circ S_0}^{\gamma_1} * \boxed{\circ S_0}^{\gamma_2}$$

Second attempt in verifying *conTwoAdd*

$$\left\{ \boxed{I(\gamma_1, \gamma_2)}^{\iota} * \left[\circ S_0\right]^{\gamma_1} \right\} \ \mathsf{faa}\ l\ (\mathsf{rand}\ 3)\ \left\{ \exists n.\ \left[\circ S_2(n)\right]^{\gamma_1} \right\}$$

$$\left\{ \boxed{I(\gamma_1, \gamma_2)}^{\iota} * \left[\circ S_0\right]^{\gamma_2} \right\} \ \mathsf{faa}\ l\ (\mathsf{rand}\ 3)\ \left\{ \exists n.\ \left[\circ S_2(n)\right]^{\gamma_2} \right\}$$

$$\left\{ \boxed{I(\gamma_1, \gamma_2)}^{\iota} * \left[\circ S_2(n_1)\right]^{\gamma_1} * \left[\circ S_2(n_2)\right]^{\gamma_2} \right\}\ !\,l\ \{v.\ v > 0\}$$

Applying HT-PAR-COMP

Second attempt in verifying *conTwoAdd* – First two Hoare triples

$$\left\{ \boxed{I(\gamma_1, \gamma_2)}^{\iota} * \left[\circ S_\circ\right]^{\gamma_1} \right\}$$

$$\mathsf{faa}\; l\; (\mathsf{rand}\; 3)$$

$$\left\{ \exists n.\, \left[\circ S_2(n)\right]^{\gamma_1} \right\}$$

First Hoare triple

(second Hoare triple is proven similarly)

Recall invariant opening rule:

HT-INV-OPEN

$$\frac{e \text{ atomic} \qquad \{I * P\}\, e\, \{I * Q\}}{\left\{\boxed{I} * P\right\}\, e\, \left\{\boxed{I} * Q\right\}}$$

Second attempt in verifying *conTwoAdd* – First two Hoare triples

$$\left\{ \boxed{I(\gamma_1, \gamma_2)}^{\iota} * \boxed{\circ S_1(n)}^{\gamma_1} \right\}$$

$\text{faa}\ l\ n$

$$\left\{ \exists n.\ \boxed{\circ S_2(n)}^{\gamma_1} \right\}$$

rand 3 is atomic

We can open invariants temporarily and update ghost resources to track n sampled

Second attempt in verifying *conTwoAdd* – First two Hoare triples

$$\left\{ \boxed{I(\gamma_1, \gamma_2)}^{\iota} * \boxed{\circ S_1(n)}^{\gamma_1} \right\}$$

$$\texttt{faa}\ l\ n$$

$$\left\{ \exists n.\ \boxed{\circ S_2(n)}^{\gamma_1} \right\}$$

rand 3 is atomic

We can open invariants temporarily and update ghost resources to track n sampled

We can do the same again with faa l n

Second attempt in verifying *conTwoAdd* – last Hoare triples

Last Hoare triple

$$\left\{ \boxed{I(\gamma_1, \gamma_2)}^{\iota} * \boxed{\circ S_2(n_1)}^{\gamma_1} * \boxed{\circ S_2(n_2)}^{\gamma_2} \right\}$$

$$!\,l$$

$$\{v.\, v > 0\}$$

Second attempt in verifying *conTwoAdd* – last Hoare triples

$$\left\{ \boxed{I(\gamma_1, \gamma_2)}^{\iota} * \boxed{\circ S_2(n_1)}^{\gamma_1} * \boxed{\circ S_2(n_2)}^{\gamma_2} \right\}$$

$$!\, l$$

$$\{v.\, v > 0\}$$

Last Hoare triple

- But nothing too surprising!

Second attempt in verifying *conTwoAdd* – last Hoare triples

$$\left\{ \boxed{I(\gamma_1, \gamma_2)}^{\iota} * \boxed{\circ S_2(n_1)}^{\gamma_1} * \boxed{\circ S_2(n_2)}^{\gamma_2} \right\}$$

$$!\, l$$

$$\{v.\, v > 0\}$$

Last Hoare triple

- But nothing too surprising!
- $!\,l$ is atomic, so we can open invariants and do a case split on value of n_1 and n_2.

Second attempt in verifying *conTwoAdd* – last Hoare triples

$$\left\{ \boxed{I(\gamma_1, \gamma_2)}^{\iota} * \boxed{\circ S_2(n_1)}^{\gamma_1} * \boxed{\circ S_2(n_2)}^{\gamma_2} \right\}$$
$$!\,l$$
$$\{v.\, v > 0\}$$

Last Hoare triple

- But nothing too surprising!
- $!\,l$ is atomic, so we can open invariants and do a case split on value of n_1 and n_2.
- If both are 0, we get $\lightning(1)$ and can derive $\bot$!

Weakest-pre

$$
\begin{aligned}
\mathsf{wp}\ e_1\,\{\Phi\} \triangleq \forall \sigma_1, \varepsilon_1.\ S(\sigma_1, \varepsilon_1) \mathrel{-\!\!*}\ {}_{\top}{\Rrightarrow}_{\emptyset}\ \mathsf{sstep}\ \sigma_1\ \varepsilon_1\,\{\sigma_2, \varepsilon_2.\\
&\big(e_1 \in Val * {}_{\emptyset}{\Rrightarrow}_{\top}\, S(\sigma_2, \varepsilon_2) * \Phi(e_1)\big) \vee \\
&\big(e_1 \notin Val * \mathsf{pstep}\ (e_1, \sigma_2)\ \varepsilon_2\,\{e_2, \sigma_3, l, \varepsilon_3.\\
&\quad \triangleright\, \mathsf{sstep}\ \sigma_3\ \varepsilon_3\,\{\sigma_4, \varepsilon_4.\ {}_{\emptyset}{\Rrightarrow}_{\top}\, S(\sigma_4, \varepsilon_4) * \mathsf{wp}\ e_2\,\{\Phi\} * \mathop{\Huge *}_{e' \in l} \mathsf{wp}\ e'\,\{\mathsf{True}\}\}\}\big)\}
\end{aligned}
$$

State and program step precondition

STATE-STEP-ERR-1

$$\frac{1 \leqslant \varepsilon}{\mathsf{sstep}\ \sigma\ \varepsilon\ \{\Phi\}}$$

STATE-STEP-RET

$$\frac{\Phi(\sigma, \varepsilon)}{\mathsf{sstep}\ \sigma\ \varepsilon\ \{\Phi\}}$$

STATE-STEP-CONTINUOUS

$$\frac{\forall \varepsilon'.\ \varepsilon < \varepsilon' \twoheadrightarrow \mathsf{sstep}\ \sigma\ \varepsilon'\ \{\Phi\}}{\mathsf{sstep}\ \sigma\ \varepsilon\ \{\Phi\}}$$

STATE-STEP-EXP

$$\frac{\mathbb{E}_{\mu}[\mathcal{F}] \leqslant \varepsilon \qquad \mathsf{schErasable}(\mu, \sigma_1) \qquad \forall \sigma_2.\ 0 < \mu(\sigma_2) \twoheadrightarrow \mathsf{sstep}\ \sigma_2\ (\mathcal{F}(\sigma_2))\ \{\Phi\}}{\mathsf{sstep}\ \sigma_1\ \varepsilon\ \{\Phi\}}$$

PROG-STEP-EXP

$$\frac{\mathsf{red}(e_1, \sigma_1) \qquad \mathbb{E}_{\mathsf{step}(e_1, \sigma_1)}[\mathcal{F}] \leqslant \varepsilon \qquad \forall (e_2, \sigma_2, l).\ 0 < \mathsf{step}(e_1, \sigma_1)(e_2, \sigma_2, l) \twoheadrightarrow \Phi(e_2, \sigma_2, l, \mathcal{F}(e_2, \sigma_2, l))}{\mathsf{pstep}\ (e_1, \sigma_1)\ \varepsilon\ \{\Phi\}}$$

Probabilistic update modality

$$ {}_{\mathcal{E}_1}\!\rightsquigarrow_{\mathcal{E}_2} P \triangleq \forall \sigma_1, \varepsilon_1.\, S(\sigma_1, \varepsilon_1) \mathrel{-\!\!*} {}_{\mathcal{E}_1}\!\Rrightarrow_{\emptyset} \mathsf{sstep}\ \sigma_1\ \varepsilon_1 \left\{\sigma_2, \varepsilon_2.\ {}_{\emptyset}\!\Rrightarrow_{\mathcal{E}_2} S(\sigma_2, \varepsilon_2) * P\right\} $$

PUPD-ELIM

$$ \frac{\{P * Q\}\, e\, \{R\}_{\mathcal{E}}}{\{(\rightsquigarrow_{\mathcal{E}} P) * Q\}\, e\, \{R\}_{\mathcal{E}}} $$

PUPD-RET

$$ \frac{P}{\rightsquigarrow_{\mathcal{E}} P} $$

PUPD-BIND

$$ \frac{{}_{\mathcal{E}_1}\!\rightsquigarrow_{\mathcal{E}_2} P \qquad P \mathrel{-\!\!*} {}_{\mathcal{E}_2}\!\rightsquigarrow_{\mathcal{E}_3} Q}{{}_{\mathcal{E}_1}\!\rightsquigarrow_{\mathcal{E}_3} Q} $$

PUPD-FUPD

$$ \frac{{}_{\mathcal{E}_1}\!\Rrightarrow_{\mathcal{E}_2} P}{{}_{\mathcal{E}_1}\!\rightsquigarrow_{\mathcal{E}_2} P} $$

PUPD-PRESAMPLE-EXP

$$ \frac{\mathbb{E}_{\mathfrak{U}N}[\mathcal{F}] \leqslant \varepsilon \qquad ↯(\varepsilon) \qquad \kappa \hookrightarrow (N, \vec{n})}{\rightsquigarrow_{\mathcal{E}} (\exists n.\, \kappa \hookrightarrow (N, \vec{n} \cdot n) * ↯(\mathcal{F}(n)))} $$

PUPD-ERR

$$ \frac{}{\rightsquigarrow_{\mathcal{E}} (\exists \varepsilon.\, 0 < \varepsilon * ↯(\varepsilon))} $$