ERROR CREDITS

Resourceful Reasoning about Error Bounds

Alejandro Aguirre Philipp Haselwarter Markus de Medeiros
Kwing Hei Li Simon Gregersen Joseph Tassarotti Lars Birkedal
alejandro@cs.au.dk

AARHUS
UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

June 4, 2024

SOLIDUM PETIT IN PROFUNDIS · UNIVERSITAS ARHUSIENSIS

```
Σ : gFunctors
irisGS0 : irisGS Σ
E : coPset
Φ : val → iPropI Σ
n, m : nat
Hnm : n < m
============================
---------------------------∗
⌜m ≤ n⌝
```

Introducing... error credits

$$\vdash \;\mid\!\Rrightarrow \lightning(0)$$

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

$$\lightning(1) \vdash \bot$$

```
Σ : gFunctors
irisGS0 : irisGS Σ
E : coPset
Φ : val → iPropI Σ
n, m : nat
Hnm : n < m
============================
"err" : ↯(1)
--------------------------*
⌜m ≤ n⌝
```

```
Σ : gFunctors
irisGS0 : irisGS Σ
E : coPset
Φ : val → iPropI Σ
n, m : nat
Hnm : n < m
============================
"err" : False
---------------------------*
⌜m ≤ n⌝
```

```
No more goals.
```

Randomized algorithms

Randomized algorithms

Monte Carlo algorithms

Randomized algorithms

Monte Carlo algorithms

Las Vegas algorithms

Randomized algorithms

Monte Carlo algorithms

- Efficient runtime
- Result might be wrong
- E.g. probabilistic primality tests

Las Vegas algorithms

Randomized algorithms

Monte Carlo algorithms

- Efficient runtime
- Result might be wrong
- E.g. probabilistic primality tests

Las Vegas algorithms

- Probabilistic runtime
- Correct result
- E.g. randomized Quicksort

Contributions

We introduce Eris, a logic to reason about probability of errors in probabilistic programs

- Error budget represented as a separation logic resource
- $\lightning(\varepsilon)$ allows specifications to fail with probability $\leq \varepsilon$
- The resource representation of error inherits the expresiveness of HO separation logic and enables new reasoning principles
- Partial and total correctness versions of the logic
- Fully mechanized in Coq and Iris

A probabilistic sequential language

$\lambda_{\mathsf{ref}}^{\mathsf{rand}}$: sequential HeapLang plus sampling

$$e \in \mathsf{Expr} ::= v \mid \mathsf{x} \mid \mathsf{rec}\,\mathsf{f}\,\mathsf{x} = e \mid e_1(e_2) \mid \mathsf{if}\ e\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2 \mid \mathsf{fst}(e) \mid \mathsf{snd}(e) \mid$$
$$\mathsf{ref}(e) \mid {!}\,e \mid e_1 \leftarrow e_2 \mid \cdots \mid \mathsf{rand}\ e$$

Probabilistic step function: $\mathrm{step} : \mathsf{Cfg} \to \mathcal{D}(\mathsf{Cfg})$.

$$\mathrm{step}((\mathsf{rec}\,\mathsf{f}\,\mathsf{x} = e)v, \sigma) = \{(e[v/\mathsf{x}][(\mathsf{rec}\,\mathsf{f}\,\mathsf{x} = e)/\mathsf{f}] \mapsto 1\}$$
$$\mathrm{step}(\mathsf{rand}\ n, \sigma) = \left\{(0,\sigma) \mapsto \frac{1}{n+1}, \ldots, (n,\sigma) \mapsto \frac{1}{n+1}\right\}$$

Partial probabilistic correctness

In Eris, WP has the following meaning:

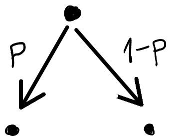
Assume φ is a pure Coq proposition, $\varepsilon \geq 0$. If

$$\lightning(\varepsilon) \vdash \mathsf{wp}\ e\ \{v.\varphi(v)\}$$

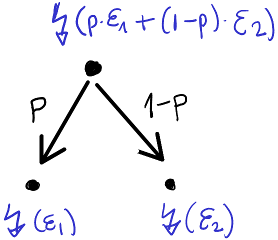
Then, with probability at least $1 - \varepsilon$, e will either diverge or return a result v satisfying $\varphi(v)$

Generating and using error credits

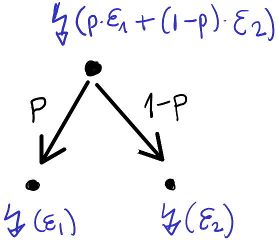
Generating and using error credits



Generating and using error credits

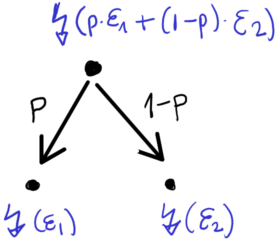


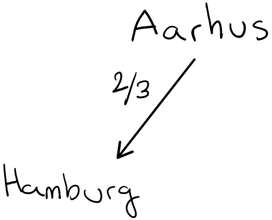
Generating and using error credits



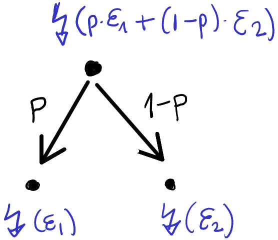
Aarhus

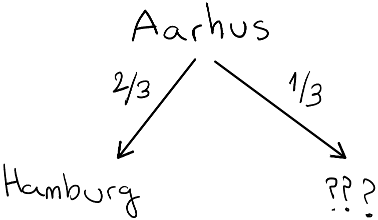
Generating and using error credits



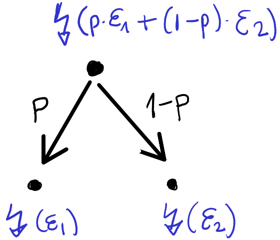


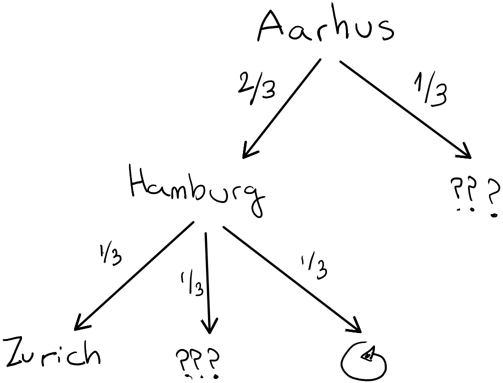
Generating and using error credits



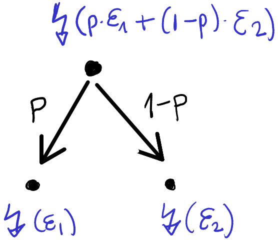


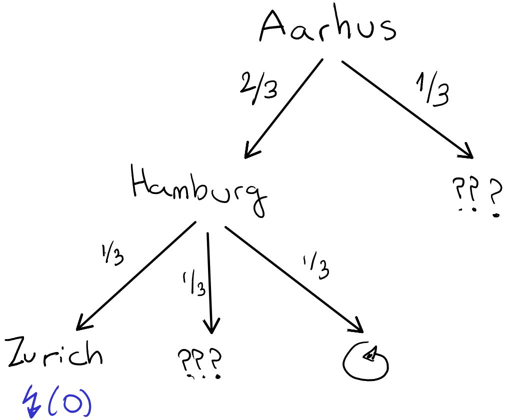
Generating and using error credits



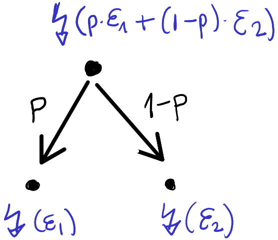


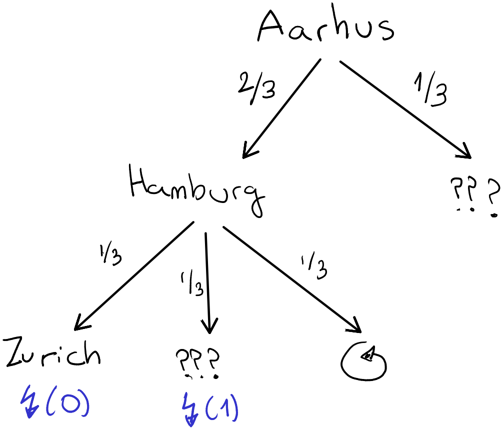
Generating and using error credits



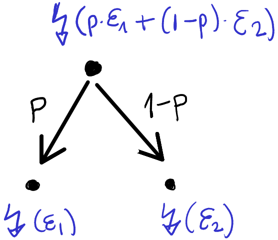


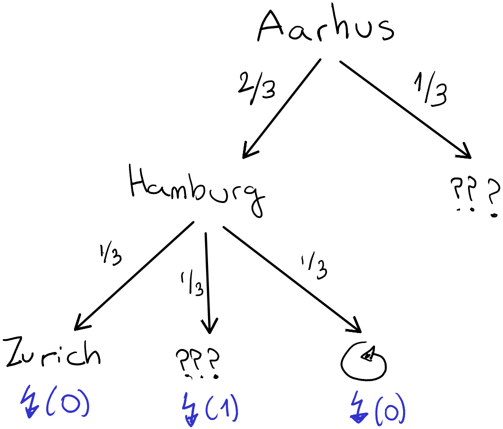
Generating and using error credits



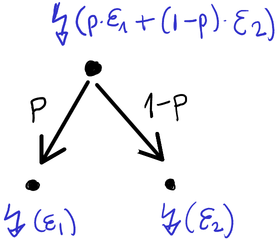


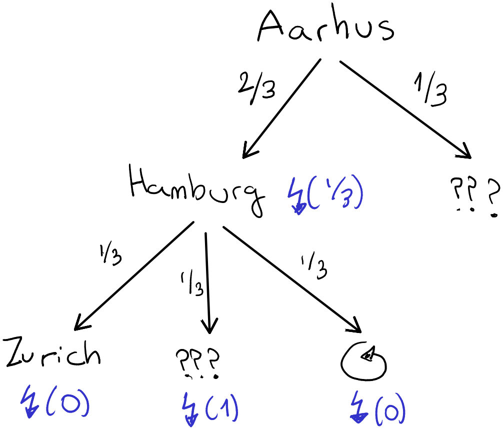
Generating and using error credits



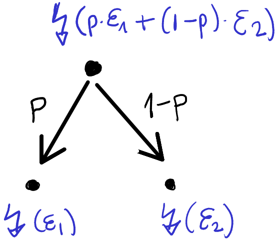


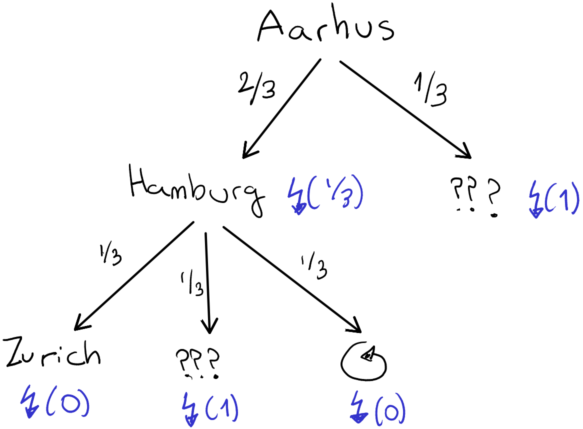
Generating and using error credits



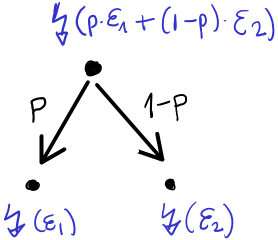


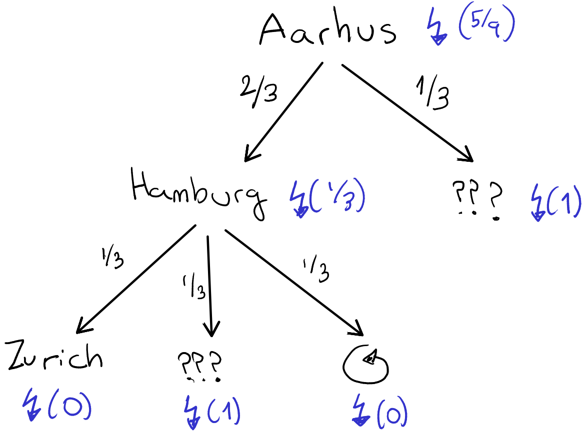
Generating and using error credits





Generating and using error credits

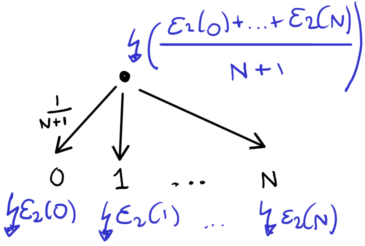




Generating and using error credits

$$\frac{\mathcal{E}_2 \colon \{0, \ldots, N\} \to \mathbb{R}^{\geq 0} \qquad \sum_{i=0}^{N} \dfrac{\mathcal{E}_2(i)}{N+1} = \varepsilon_1}{\vdash \{\lightning(\varepsilon_1)\}\ \textsf{rand}\ N\ \{n\ .\ \lightning(\mathcal{E}_2(n))\}}\ \textsf{ht-rand-exp}$$

Generating and using error credits



$$\frac{\mathcal{E}_2 \colon \{0, \ldots, N\} \to \mathbb{R}^{\geq 0} \qquad \sum_{i=0}^{N} \dfrac{\mathcal{E}_2(i)}{N+1} = \varepsilon_1}{\vdash \{\lightning(\varepsilon_1)\}\ \textsf{rand}\ N\ \{n\ .\ \lightning(\mathcal{E}_2(n))\}}\ \text{ht-rand-exp}$$

Derived rules

$$\frac{X \subseteq N \qquad \varepsilon = \dfrac{|X|}{N+1}}{\vdash \{\lightning(\varepsilon)\}\ \textsf{rand}\ N\ \{n\ .\ n \notin X\}}\ \text{ht-rand-err-sub}$$

Derived rules

$$\frac{X \subseteq N \qquad \varepsilon = \dfrac{|X|}{N+1}}{\vdash \{\lightning(\varepsilon)\}\ \textsf{rand}\ N\ \{n\ .\ n \notin X\}}\ \textsf{ht-rand-err-sub}$$

1. Apply ht − rand − exp with $\quad \mathcal{E}_2(x) \triangleq \mathrm{if}\ (x \in X)\ \mathrm{then}\ 1\ \mathrm{else}\ 0$

Derived rules

$$\frac{X \subseteq N \qquad \varepsilon = \dfrac{|X|}{N+1}}{\vdash \{ \lightning(\varepsilon) \} \ \textsf{rand}\ N \ \{ n \,.\, n \notin X \}} \ \textsf{ht-rand-err-sub}$$

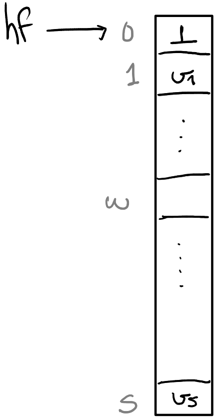
1. Apply ht – rand – exp with $\mathcal{E}_2(x) \triangleq \text{if } (x \in X) \text{ then } 1 \text{ else } 0$
2. Discard outcomes with 1 error credit

Collision-free hashing

$$
\begin{aligned}
\mathsf{query}\ hf\ w \triangleq\ & \mathsf{match}\ \mathsf{get}\ hf\ w\ \mathsf{with} \\
& \quad \mathsf{Some}(v) \Rightarrow v \\
& \mid \mathsf{None} \quad\ \ \Rightarrow \mathsf{let}\ v = \mathsf{rand}\ (S)\ \mathsf{in} \\
& \qquad\qquad\qquad\quad \mathsf{set}\ hf\ w\ v; \\
& \qquad\qquad\qquad\quad v \\
& \mathsf{end}
\end{aligned}
$$

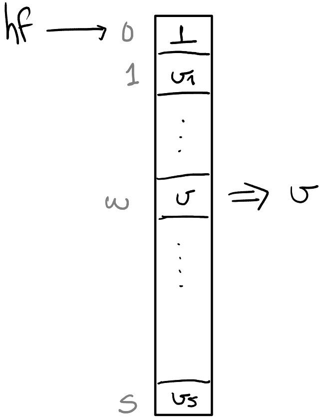
Collision-free hashing

query $hf\ w \triangleq$ match get $hf\ w$ with
  Some$(v) \Rightarrow v$
  | None $\Rightarrow$ let $v = \text{rand}\ (S)$ in
    set $hf\ w\ v$;
    v
end



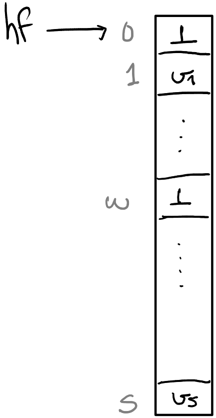
Collision-free hashing

$$
\begin{array}{l}
\text{query } hf\,w \triangleq \text{match get } hf\,w \text{ with} \\
\quad \text{Some}(v) \Rightarrow v \\
\quad |\ \text{None} \quad \Rightarrow \text{let } v = \text{rand } (S) \text{ in} \\
\qquad\qquad\qquad \text{set } hf\,w\,v; \\
\qquad\qquad\qquad v \\
\text{end}
\end{array}
$$



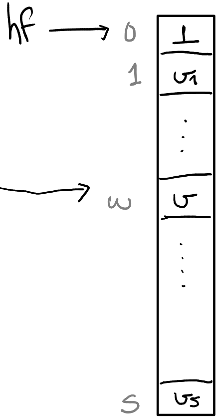
Collision-free hashing

query $hf\,w \triangleq$ match get $hf\,w$ with
  Some$(v) \Rightarrow v$
  | None $\Rightarrow$ let v = rand (S) in
    set $hf\,w\,v$;
    v
  end



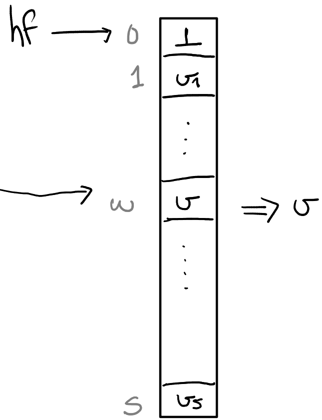
Collision-free hashing

$$\begin{aligned} \text{query}\ hf\ w \triangleq\ & \text{match get}\ hf\ w\ \text{with} \\ & \quad \text{Some}(v) \Rightarrow v \\ & \mid \text{None} \quad \Rightarrow \text{let}\ v = \text{rand}\ (S)\ \text{in} \\ & \qquad\qquad\quad \text{set}\ hf\ w\ v; \\ & \qquad\qquad\quad v \\ & \text{end} \end{aligned}$$



Collision-free hashing

$$
\begin{aligned}
\text{query } hf\, w \triangleq\ & \textsf{match get } hf\, w \textsf{ with} \\
& \quad \textsf{Some}(v) \Rightarrow v \\
& \mid \textsf{None} \quad\ \Rightarrow \textsf{let } v = \textsf{rand } (S) \textsf{ in} \\
& \qquad\qquad\qquad\ \textsf{set } hf\, w\, v; \\
& \qquad\qquad\qquad\ v \\
& \textsf{end}
\end{aligned}
$$



Collision-free hashing

$$\left\{n \notin \mathrm{dom}\ m * \mathsf{collFree}(hf, \mathsf{m}) * ↯\left(\frac{|\mathsf{m}|}{\mathsf{S}+1}\right)\right\}\ \mathsf{query}\ hf\ n\ \{v.\ \mathsf{collFree}(hf, \mathsf{m}[\mathsf{n} \leftarrow \mathsf{v}])\}$$

Collision-free hashing

$$\left\{ n \notin \mathsf{dom}\ m * \mathsf{collFree}(hf, \mathsf{m}) * ↯\left(\frac{|\mathsf{m}|}{\mathsf{S}+1}\right) \right\} \mathsf{query}\ hf\ n\ \{v.\ \mathsf{collFree}(hf, \mathsf{m}[\mathsf{n} \leftarrow \mathsf{v}])\}$$

$$\mathsf{collFree}(hf, \mathsf{m}) \triangleq \begin{array}{l} (\forall i < S, \\ \qquad ((hf +_{\ell} i \mapsto \mathsf{None} \wedge i \notin \mathsf{dom}(m)) \vee \\ \qquad (\exists v, hf +_{\ell} i \mapsto \mathsf{Some}\ v \wedge m[i] = v))) * \\ \mathsf{injective}\ \mathsf{m} \end{array}$$

Amortized error

Suppose we execute a sequence of M insertions. The total error will be

$$0 + \frac{1}{S+1} + \frac{2}{S+1} + \cdots + \frac{M-1}{S+1}$$

Amortized error

Suppose we execute a sequence of M insertions. The total error will be

$$0 + \frac{1}{S+1} + \frac{2}{S+1} + \cdots + \frac{M-1}{S+1}$$

We can prove a stronger specification tracking amortized error

$$\left\{ \begin{array}{c} n \notin \mathsf{dom}\ m\ * \\ |m| < M\ * \\ \mathsf{collFreeAE}(hf, \mathsf{m})\ * \\ ↯\left(\frac{M-1}{2(S+1)}\right) \end{array} \right\} \mathsf{query}\ hf\ n\ \{v.\ \mathsf{collFreeAE}(hf, \mathsf{m}[\mathsf{n} \leftarrow \mathsf{v}])\}$$

Amortized error

$$\left\{\begin{array}{c} n \notin \mathsf{dom}\ m \ * \\ |m| < M \ * \\ \mathsf{collFreeAE}(hf, \mathsf{m}) \ * \\ \lightning\left(\frac{M-1}{2(S+1)}\right) \end{array}\right\} \mathsf{query}\ hf\ \ n\ \{v.\,\mathsf{collFreeAE}(hf, \mathsf{m}[\mathsf{n} \leftarrow \mathsf{v}])\}$$

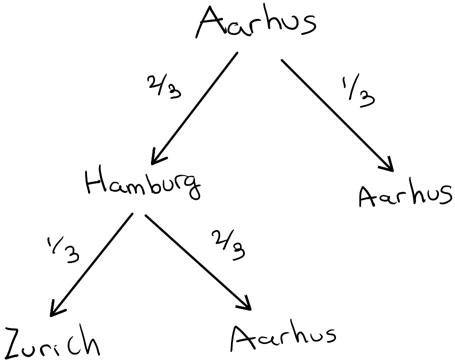
Amortized error

$$\left\{ \begin{array}{c} n \notin \mathsf{dom}\ m \ast \\ |m| < M \ast \\ \mathsf{collFreeAE}(hf, \mathsf{m}) \ast \\ ↯\left(\frac{M-1}{2(S+1)}\right) \end{array} \right\} \mathsf{query}\ hf\ n\ \{v.\, \mathsf{collFreeAE}(hf, \mathsf{m}[\mathsf{n} \leftarrow \mathsf{v}])\}$$

$$\mathsf{collFreeAE}(hf, \mathsf{m}) \quad \triangleq \quad \exists \varepsilon.\, ↯(\varepsilon) \ast \ulcorner \varepsilon \geq \sum_{\mathsf{i}=|\mathsf{m}|}^{\mathsf{M}} \frac{\mathsf{i}}{\mathsf{S}+1} \urcorner \ast \mathsf{collFree}(hf, \mathsf{m})$$

Reasoning about termination

Reasoning about termination



A Total WP

Eris also provides a total WP with the following meaning:

Assume φ pure Coq proposition, $\varepsilon \geq 0$. If

$$↯(\varepsilon) \vdash \mathsf{twp}\ e\ \{v.\varphi(v)\}$$

Then, with probability at least $1 - \varepsilon$, e will terminate and return a result v satisfying $\varphi(v)$

Refresher: Löb induction

$$\frac{\triangleright(\mathsf{wp}\ e\ \{\Phi\}) \vdash \mathsf{wp}\ e\ \{\Phi\}}{\vdash \mathsf{wp}\ e\ \{\Phi\}}\ \text{Löb}$$

Refresher: Löb induction

$$\frac{\triangleright(\mathsf{wp}\ e\ \{\Phi\}) \vdash \mathsf{wp}\ e\ \{\Phi\}}{\vdash \mathsf{wp}\ e\ \{\Phi\}}\ \textsf{Löb}$$

$$\frac{(\pounds(1) \mathrel{-\!\!*} \mathsf{wp}\ e\ \{\Phi\}) \vdash \mathsf{wp}\ e\ \{\Phi\}}{\vdash \mathsf{wp}\ e\ \{\Phi\}}\ \textsf{LöbLC}$$

Refresher: Löb induction

$$\frac{\triangleright(\mathsf{wp}\ e\ \{\Phi\}) \vdash \mathsf{wp}\ e\ \{\Phi\}}{\vdash \mathsf{wp}\ e\ \{\Phi\}}\ \text{Löb}$$

$$\frac{(£(1) \twoheadrightarrow \mathsf{wp}\ e\ \{\Phi\}) \vdash \mathsf{wp}\ e\ \{\Phi\}}{\vdash \mathsf{wp}\ e\ \{\Phi\}}\ \text{LöbLC}$$

- We get an induction hypothesis guarded by $£(1)$
- Later credits obtained by execution steps, but **only under partial WP**

Error induction

$$\frac{\varepsilon > 0 \qquad k > 1 \\ \forall \varepsilon', (\lightning\left(k \cdot \varepsilon'\right) \mathrel{-\!\!*} \textsf{twp}\ e\ \{\Phi\}) * \lightning\left(\varepsilon'\right) \vdash \textsf{twp}\ e\ \{\Phi\}}{\lightning(\varepsilon) \vdash \textsf{twp}\ e\ \{\Phi\}}\ \textsf{err-ind}$$

Error induction

$$\frac{\varepsilon > 0 \qquad {\color{magenta} k > 1} \qquad \forall \varepsilon', (\text{\textit{↯}}(k \cdot \varepsilon') \mathrel{-\!\!*} \mathsf{twp}\ e\ \{\Phi\}) * \text{\textit{↯}}(\varepsilon') \vdash \mathsf{twp}\ e\ \{\Phi\}}{\text{\textit{↯}}(\varepsilon) \vdash \mathsf{twp}\ e\ \{\Phi\}}\ \textsf{err-ind}$$

- We choose an amplification factor $k > 1$

Error induction

$$\frac{\varepsilon > 0 \qquad k > 1 \qquad \forall \varepsilon', (\text{↯}(k \cdot \varepsilon') \mathrel{-\!\!*} \mathsf{twp}\ e\ \{\Phi\}) * \text{↯}(\varepsilon') \vdash \mathsf{twp}\ e\ \{\Phi\}}{\text{↯}(\varepsilon) \vdash \mathsf{twp}\ e\ \{\Phi\}}\ \textsf{err-ind}$$

- We choose an amplification factor $k > 1$
- We get an arbitrary amount of credits ε'

Error induction

$$\frac{\varepsilon > 0 \qquad k > 1 \qquad \forall \varepsilon', (\lightning(k \cdot \varepsilon') \mathrel{-\!\!*} \textsf{twp}\ e\ \{\Phi\}) * \lightning(\varepsilon') \vdash \textsf{twp}\ e\ \{\Phi\}}{\lightning(\varepsilon) \vdash \textsf{twp}\ e\ \{\Phi\}}\ \textsf{err-ind}$$

- We choose an amplification factor $k > 1$
- We get an arbitrary amount of credits ε'
- We get an induction hypothesis guarded by $\lightning(k \cdot \varepsilon')$

Error induction

$$\frac{\varepsilon > 0 \qquad k > 1 \qquad \forall \varepsilon', (\lightning(k \cdot \varepsilon') \mathrel{-\!\!*} \mathsf{twp}\ e\ \{\Phi\}) * \lightning(\varepsilon') \vdash \mathsf{twp}\ e\ \{\Phi\}}{\lightning(\varepsilon) \vdash \mathsf{twp}\ e\ \{\Phi\}}\ \textsf{err-ind}$$

- We choose an amplification factor $k > 1$
- We get an arbitrary amount of credits ε'
- We get an induction hypothesis guarded by $\lightning(k \cdot \varepsilon')$
- Error credits can be obtained by taking probabilistic choices

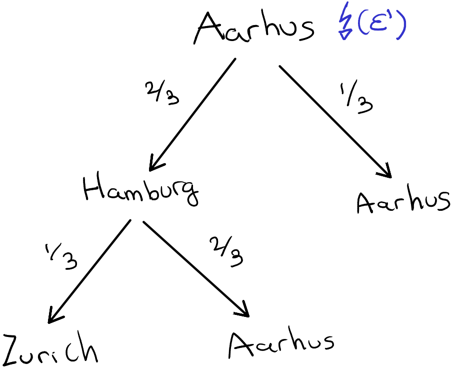
Error induction

$$\frac{\varepsilon > 0 \qquad k > 1 \qquad \forall \varepsilon', (\lightning(k \cdot \varepsilon') \mathrel{-\!\!*} \mathsf{twp}\ e\ \{\Phi\}) * \lightning(\varepsilon') \vdash \mathsf{twp}\ e\ \{\Phi\}}{\lightning(\varepsilon) \vdash \mathsf{twp}\ e\ \{\Phi\}}\ \text{err-ind}$$

- We choose an amplification factor $k > 1$
- We get an arbitrary amount of credits ε'
- We get an induction hypothesis guarded by $\lightning(k \cdot \varepsilon')$
- Error credits can be obtained by taking probabilistic choices

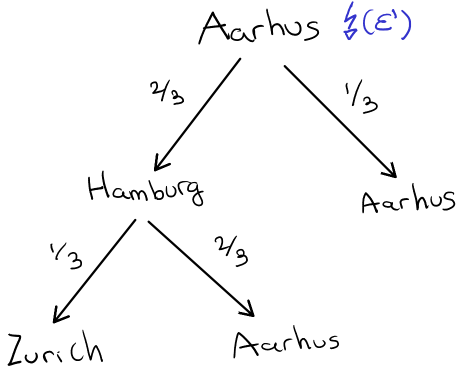
Intuitively: by amplifying ε by k enough times, we can get to $\lightning(1)$.

Proving termination



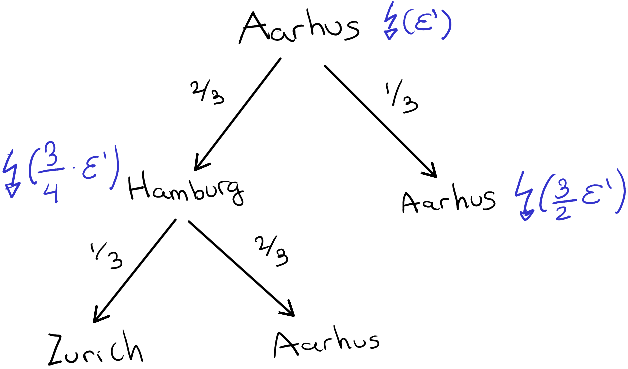
Proving termination

$K = 9/8 > 1$



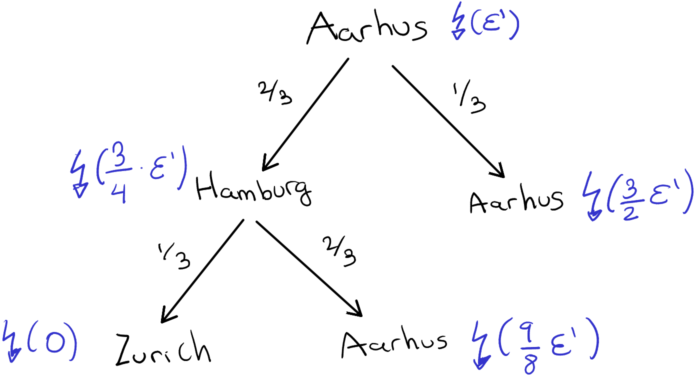
Proving termination

$K = 9/8 > 1$



Proving termination

$K = 9/8 > 1$



Adequacy in the limit

By the completeness of the reals, we can prove a stronger adequacy result:

Theorem (Limit adequacy for total correctness)

Let φ be a pure Coq proposition. If for every $\varepsilon > 0$ we can prove

$$↯(\varepsilon) \vdash \mathsf{twp}\; e\; \{v.\varphi(v)\}$$

then, with probability 1, e will return a result v satisfying $\varphi(v)$

Constructing the model

An error-tracking WP

Our WP tracks the authoritative view $\lightning_\bullet(\varepsilon_1)$ of the error budget, and threads it through a Graded Lifting Modality (GLM)

An error-tracking WP

Our WP tracks the authoritative view $\lightning_\bullet(\varepsilon_1)$ of the error budget, and threads it through a Graded Lifting Modality (GLM)

$$
\begin{aligned}
\mathsf{wp}\ e_1\ \{\Phi\} \triangleq\ & (e_1 \in \mathsf{Val} \wedge \Phi(e_1)) \\
& \vee (e_1 \notin \mathsf{Val} \wedge \forall \sigma_1, \varepsilon_1 .\, S(\sigma_1) * \lightning_\bullet(\varepsilon_1) \mathrel{-\!\!*} \\
& \quad \mathsf{GLM}(e_1, \sigma_1, \varepsilon_1,\ (\lambda\, e_2, \sigma_2, \varepsilon_2 \,.\, \triangleright(S(\sigma_2) * \lightning_\bullet(\varepsilon_2)\ * \mathsf{wp}\ e_2\ \{\Phi\}))))
\end{aligned}
$$

The Graded Lifting Modality

The GLM pairs logic and error usage with the operational semantics

$$\mathsf{GLM} \colon \mathsf{Cfg} \to \mathbb{R}^{\geq 0} \to (\mathsf{Cfg} \to \mathbb{R}^{\geq 0} \to \mathsf{iProp}) \to \mathsf{iProp}$$

The Graded Lifting Modality

The GLM pairs logic and error usage with the operational semantics

$$\mathsf{GLM} \colon \mathsf{Cfg} \to \mathbb{R}^{\geq 0} \to (\mathsf{Cfg} \to \mathbb{R}^{\geq 0} \to \mathsf{iProp}) \to \mathsf{iProp}$$

Intended meaning: $\mathsf{GLM}((e, \sigma), \varepsilon, Z)$ iff

The Graded Lifting Modality

The GLM pairs logic and error usage with the operational semantics

$$\mathsf{GLM} : \mathsf{Cfg} \to \mathbb{R}^{\geq 0} \to (\mathsf{Cfg} \to \mathbb{R}^{\geq 0} \to \mathsf{iProp}) \to \mathsf{iProp}$$

Intended meaning: $\mathsf{GLM}((e, \sigma), \varepsilon, Z)$ iff

- Starting from (e, σ) with ε error budget,

The Graded Lifting Modality

The GLM pairs logic and error usage with the operational semantics

$$\mathsf{GLM} \colon \mathsf{Cfg} \to \mathbb{R}^{\geq 0} \to (\mathsf{Cfg} \to \mathbb{R}^{\geq 0} \to \mathsf{iProp}) \to \mathsf{iProp}$$

Intended meaning: $\mathsf{GLM}((e, \sigma), \varepsilon, Z)$ iff

- Starting from (e, σ) with ε error budget,
- we can take a (probabilistic) execution step, and

The Graded Lifting Modality

The GLM pairs logic and error usage with the operational semantics

$$\mathsf{GLM} \colon \mathsf{Cfg} \to \mathbb{R}^{\geq 0} \to (\mathsf{Cfg} \to \mathbb{R}^{\geq 0} \to \mathsf{iProp}) \to \mathsf{iProp}$$

Intended meaning: $\mathsf{GLM}((e, \sigma), \varepsilon, Z)$ iff

- Starting from (e, σ) with ε error budget,
- we can take a (probabilistic) execution step, and
- ensure we get (e', σ') and residual error ε' s.t. $Z((e', \sigma'), \varepsilon')$ or $\varepsilon' \geq 1$

$$\frac{\mathsf{red}(e_1, \sigma_1) \quad \varepsilon_1 + \varepsilon_2 \leq \varepsilon \quad \Pr_{\mathrm{step}(e_1,\sigma_1)}[\neg R] \leq \varepsilon_1 \\ \forall e_2, \sigma_2.\, R(e_2, \sigma_2) \twoheadrightarrow Z(e_2, \sigma_2, \varepsilon_2)}{\mathsf{GLM}((e_1, \sigma_1), \varepsilon, Z)}\ \textsf{step-simple}$$

$$\frac{\mathsf{red}(e_1, \sigma_1) \quad \varepsilon_1 + \varepsilon_2 \leq \varepsilon \quad \Pr_{\mathrm{step}(e_1,\sigma_1)}[\neg R] \leq \varepsilon_1 \\ \forall e_2, \sigma_2.\, R(e_2, \sigma_2) \mathrel{-\!\!*} Z(e_2, \sigma_2, \varepsilon_2)}{\mathsf{GLM}((e_1, \sigma_1), \varepsilon, Z)}\ \text{step-simple}$$

First a simple setting:

1. We split ε into ε_1 and ε_2

$$\frac{\mathsf{red}(e_1, \sigma_1) \quad \varepsilon_1 + \varepsilon_2 \le \varepsilon \quad \Pr_{\mathsf{step}(e_1,\sigma_1)}[\neg R] \le \varepsilon_1 \\ \forall e_2, \sigma_2.\, R(e_2, \sigma_2) \mathrel{-\!\!*} Z(e_2, \sigma_2, \varepsilon_2)}{\mathsf{GLM}((e_1, \sigma_1), \varepsilon, Z)}\ \text{step-simple}$$

First a simple setting:

1. We split ε into ε_1 and ε_2
2. We spend ε_1 to make sure we step into R

$$\frac{\mathsf{red}(e_1, \sigma_1) \quad \varepsilon_1 + \varepsilon_2 \leq \varepsilon \quad \Pr_{\mathrm{step}(e_1,\sigma_1)}[\neg R] \leq \varepsilon_1 \qquad \forall e_2, \sigma_2.\, R(e_2, \sigma_2) \twoheadrightarrow Z(e_2, \sigma_2, \varepsilon_2)}{\mathsf{GLM}((e_1, \sigma_1), \varepsilon, Z)} \;\textsf{step-simple}$$

First a simple setting:

1. We split ε into ε_1 and ε_2
2. We spend ε_1 to make sure we step into R
3. Any configuration (e_2, σ_2) in R gets ε_2

$$\frac{\mathsf{red}(e_1, \sigma_1) \quad \varepsilon_1 + \varepsilon_2 \leq \varepsilon \quad \Pr_{\mathrm{step}(e_1,\sigma_1)}[\neg R] \leq \varepsilon_1 \\ \forall e_2, \sigma_2.\ R(e_2, \sigma_2) \twoheadrightarrow Z(e_2, \sigma_2, \varepsilon_2)}{\mathsf{GLM}((e_1, \sigma_1), \varepsilon, Z)} \text{ step-simple}$$

First a simple setting:

1. We split ε into ε_1 and ε_2
2. We spend ε_1 to make sure we step into R
3. Any configuration (e_2, σ_2) in R gets ε_2
4. From R, we have to prove $Z((e_2, \sigma_2), \varepsilon_2)$

$$\dfrac{\mathsf{red}(e_1, \sigma_1) \qquad \Pr_{\mathrm{step}(e_1,\sigma_1)}[\neg R] \leq \varepsilon_1 \\ \varepsilon_1 + \sum_{\rho_2 \in \mathsf{Cfg}} \mathrm{step}(\rho_1)(\rho_2) \cdot \mathcal{E}_2(\rho_2) \leq \varepsilon \\ \forall (e_2, \sigma_2).\, R(e_2, \sigma_2) \mathrel{-\!\!*} \mathcal{E}_2((e_2, \sigma_2)) \geq 1 \vee Z((e_2, \sigma_2), (\mathcal{E}_2((e_2, \sigma_2))))}{\mathsf{GLM}((e_1, \sigma_1), \varepsilon, Z)} \ \textsf{step-exp}$$

$$\frac{\mathsf{red}(e_1, \sigma_1) \qquad \Pr_{\mathrm{step}(e_1,\sigma_1)}[\neg R] \leq \varepsilon_1 \qquad \varepsilon_1 + \sum_{\rho_2 \in \mathsf{Cfg}} \mathrm{step}(\rho_1)(\rho_2) \cdot \mathcal{E}_2(\rho_2) \leq \varepsilon \qquad \forall (e_2, \sigma_2).\, R(e_2, \sigma_2) \twoheadrightarrow \mathcal{E}_2((e_2, \sigma_2)) \geq 1 \vee Z((e_2, \sigma_2), (\mathcal{E}_2((e_2, \sigma_2))))}{\mathsf{GLM}((e_1, \sigma_1), \varepsilon, Z)} \ \text{step-exp}$$

Using expected value composition:

1. We split ε into ε_1 and the rest, ensuring the average is below the total

$$\mathsf{red}(e_1, \sigma_1) \qquad \mathrm{Pr}_{\mathrm{step}(e_1,\sigma_1)}[\neg R] \leq \varepsilon_1$$

$$\dfrac{\begin{array}{c}\varepsilon_1 + \sum_{\rho_2 \in \mathsf{Cfg}} \mathrm{step}(\rho_1)(\rho_2) \cdot \mathcal{E}_2(\rho_2) \leq \varepsilon \\ \forall (e_2, \sigma_2).\, R(e_2, \sigma_2) \twoheadrightarrow \mathcal{E}_2((e_2, \sigma_2)) \geq 1 \vee Z((e_2, \sigma_2), (\mathcal{E}_2((e_2, \sigma_2))))\end{array}}{\mathsf{GLM}((e_1, \sigma_1), \varepsilon, Z)} \ \textsf{step-exp}$$

Using expected value composition:

1. We split ε into ε_1 and the rest, ensuring the average is below the total
2. We spend ε_1 to make sure we step into R

$$\frac{\mathsf{red}(e_1, \sigma_1) \qquad \Pr_{\mathrm{step}(e_1,\sigma_1)}[\neg R] \le \varepsilon_1 \qquad \varepsilon_1 + \sum_{\rho_2 \in \mathsf{Cfg}} \mathrm{step}(\rho_1)(\rho_2) \cdot \mathcal{E}_2(\rho_2) \le \varepsilon \qquad \forall (e_2, \sigma_2).\, R(e_2, \sigma_2) \mathrel{-\!\!*} \mathcal{E}_2((e_2, \sigma_2)) \ge 1 \vee Z((e_2, \sigma_2), (\mathcal{E}_2((e_2, \sigma_2))))}{\mathsf{GLM}((e_1, \sigma_1), \varepsilon, Z)} \; \textsf{step-exp}$$

Using expected value composition:

1. We split ε into ε_1 and the rest, ensuring the average is below the total
2. We spend ε_1 to make sure we step into R
3. Any configuration (e_2, σ_2) in R gets $\mathcal{E}_2(e_2, \sigma_2)$

$$\dfrac{\mathsf{red}(e_1, \sigma_1) \qquad \Pr_{\mathrm{step}(e_1,\sigma_1)}[\neg R] \le \varepsilon_1 \qquad \varepsilon_1 + \sum_{\rho_2 \in \mathsf{Cfg}} \mathrm{step}(\rho_1)(\rho_2) \cdot \mathcal{E}_2(\rho_2) \le \varepsilon \qquad \forall (e_2, \sigma_2).\, R(e_2, \sigma_2) \twoheadrightarrow \mathcal{E}_2((e_2, \sigma_2)) \ge 1 \lor Z((e_2, \sigma_2), (\mathcal{E}_2((e_2, \sigma_2))))}{\mathsf{GLM}((e_1, \sigma_1), \varepsilon, Z)} \text{ step-exp}$$

Using expected value composition:

1. We split ε into ε_1 and the rest, ensuring the average is below the total
2. We spend ε_1 to make sure we step into R
3. Any configuration (e_2, σ_2) in R gets $\mathcal{E}_2(e_2, \sigma_2)$
4. From R, we have to prove $1 \le \varepsilon_2$ or $Z((e_2, \sigma_2), \varepsilon_2$

Adequacy

Theorem (Adequacy for partial correctness)

Let φ be a pure Coq proposition, $\varepsilon \geq 0$. If

$$↯(\varepsilon) \vdash \mathsf{wp}\; e\; \{v.\varphi(v)\}$$

Then, with probability at least $1 - \varepsilon$, e will either diverge or return a result v satisfying $\varphi(v)$

Open questions

- Continuous WP wrt the error entirely within the logic?
- Is there a deeper induction principle behind error amplification?

Conclusions

We introduced Eris, a logic to reason about probability of errors in probabilistic programs

- Error budget represented as a separation logic resource
- New reasoning principles: amortized error, error induction, and more!
- Partial and total correctness versions of the logic

Conclusions

We introduced Eris, a logic to reason about probability of errors in probabilistic programs

- Error budget represented as a separation logic resource
- New reasoning principles: amortized error, error induction, and more!
- Partial and total correctness versions of the logic

Read our draft! `https://arxiv.org/pdf/2404.14223`
We are on Github! `https://github.com/logsem/clutch`