Tachis: Higher-Order Separation Logic with Credits for Expected Costs

Philipp G. Haselwarter[1], Kwing Hei Li[1], Markus de Medeiros[2], Simon Oddershede Gregersen[2], Alejandro Aguirre[1], Joseph Tassarotti[2], and Lars Birkedal[1]

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

Cost Analysis

Algorithms 101:

"Argue that $\mathcal{A}(n)$ runs in time $t(n)$."

Cost Analysis

Algorithms 101:

“Argue that $\mathcal{A}(n)$ runs in time $t(n)$.”

Algorithms 201:

“Show that $\mathcal{A}(n)$ runs in expected time $t(n)$. ”

Cost Analysis

Algorithms 101:

"Argue that $\mathcal{A}(n)$ runs in time $t(n)$."

Algorithms 201:

"Show that $\mathcal{A}(n)$ runs in expected time $t(n)$. "

Algorithms 301:

"Prove that $\mathcal{A}(n)$ runs in amortized expected time $t(n)$. "

Cost Analysis

Algorithms 101:
"Argue that $\mathcal{A}(n)$ runs in time $t(n)$."

Algorithms 201:
"Show that $\mathcal{A}(n)$ runs in expected time $t(n)$. "

Algorithms 301:
"Prove that $\mathcal{A}(n)$ runs in amortized expected time $t(n)$. "

Tachis:
"Formalize that $\mathcal{A}(n)$ runs in expected amortized time $t(n)$."

Algorithms 101:
"Argue that $\mathcal{A}(n)$ runs in time $t(n)$."

Algorithms 201:
"Show that $\mathcal{A}(n)$ runs in expected time $t(n)$. "

Algorithms 301:
"Prove that $\mathcal{A}(n)$ runs in amortized expected time $t(n)$. "

Tachis:
"Formalize that $\mathcal{A}(n)$ runs in expected amortized time $t(n)$."

Drawing Inspiration from Atkey's Time Credits

To reason about costs, Atkey proposed separation logic with time credit assertions $\$n$:

$$\frac{}{\{\$n\}\ \mathsf{tick}()\ \{\$(n-1)\}} \qquad \frac{}{\$(n_1 + n_2) \dashv\vdash \$n_1 * \$n_2}$$

Drawing Inspiration from Atkey's Time Credits

To reason about costs, Atkey proposed separation logic with time credit assertions $\$n$:

$$\frac{}{\{\$n\}\ \mathsf{tick}()\ \{\$(n-1)\}} \qquad \frac{}{\$(n_1 + n_2) \dashv\vdash \$n_1 * \$n_2}$$

The specification

$$\{\$n\}\ e\ \{Q\}$$

implies e does at most n calls to $\mathsf{tick}()$.

Drawing Inspiration from Atkey's Time Credits

To reason about costs, Atkey proposed separation logic with time credit assertions $\$n$:

$$\frac{}{\{\$n\}\ \mathsf{tick}()\ \{\$(n-1)\}} \qquad\qquad \frac{}{\$(n_1 + n_2) \dashv\vdash \$n_1 * \$n_2}$$

The specification

$$\{\$n\}\ e\ \{Q\}$$

implies e does at most n calls to tick().

Implemented in Rocq by Chargéraud and Pottier.
In Iris by Mével, Jourdan, Pottier.

Intro

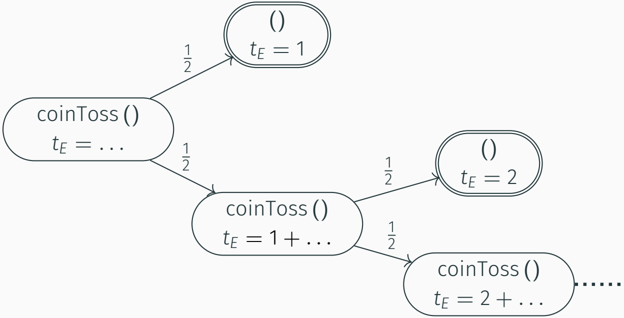
Expected Runtime Warm-Up

Consider $\mathsf{rec}\ \mathrm{coinToss}\ \_ = \mathsf{tick}\,1;\ \mathsf{if}\ \mathsf{flip}\ \mathsf{then}\ ()\ \mathsf{else}\ \mathrm{coinToss}\ ()$

Expected Runtime Warm-Up

Consider $\quad \mathsf{rec}\ \mathrm{coinToss}\ \_ = \mathsf{tick}\ 1;\ \mathsf{if}\ \mathsf{flip}\ \mathsf{then}\ ()\ \mathsf{else}\ \mathrm{coinToss}\ ()$

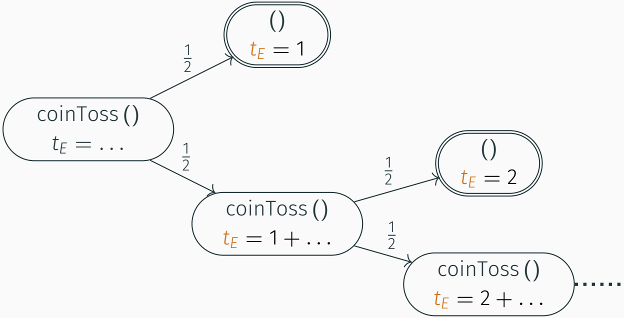
Evaluate:



Expected Runtime Warm-Up

Consider $\text{rec coinToss \_} = \text{tick } 1;\ \text{if flip then } ()\ \text{else coinToss } ()$

Evaluate:



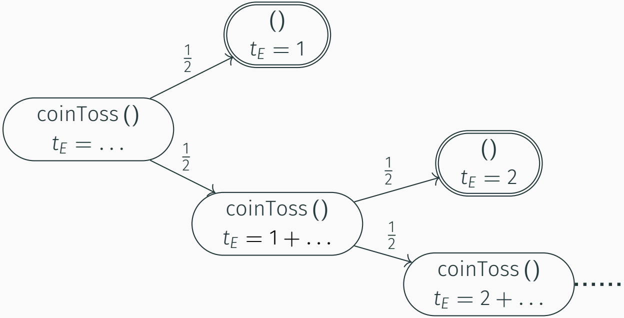
t_E depends on the value produced by flip!

No finite worst-case time bound.

Expected Runtime Warm-Up

Consider $\quad$ rec coinToss _ = tick 1; if flip then () else coinToss ()

Evaluate:

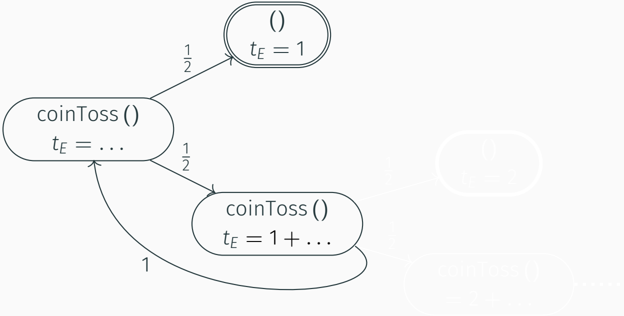


Expected time: $\quad t_E = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot (1 + (\frac{1}{2} \cdot 1 + \frac{1}{2} \ldots)) = \sum_{n=1}^{\infty} \frac{n}{2^n}$

Expected Runtime Warm-Up

Consider $\quad$ rec coinToss _ = tick 1; if flip then () else coinToss ()

Evaluate:



Expected time: $\quad t_E = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot (1 + t_E)$

Solve recurrence for t_E: $\quad t_E = 2\,.$

Our Contribution

Tachis: Higher-Order Separation Logic with Credits for Expected Costs

- Cost analysis for “randomized ML”
  $\rightarrow$ expressive language (higher order functions, local state, general recursion)

Our Contribution

Tachis: Higher-Order Separation Logic with Credits for Expected Costs

- Cost analysis for “randomized ML”
  $\rightarrow$ expressive language (higher order functions, local state, general recursion)
- Probabilistic cost credits (analogous to time credits)
  $\rightarrow$ amortized reasoning, local “expectation accounting”

Our Contribution

Tachis: Higher-Order Separation Logic with Credits for Expected Costs

- Cost analysis for “randomized ML”
  $\rightarrow$ expressive language (higher order functions, local state, general recursion)
- Probabilistic cost credits (analogous to time credits)
  $\rightarrow$ amortized reasoning, local “expectation accounting”
- General cost analysis (time complexity, entropy, “ticks”, ...)
  $\rightarrow$ user-definable cost models, *e.g.*, expected entropy use

Our Contribution

Tachis: Higher-Order Separation Logic with Credits for Expected Costs

- Cost analysis for “randomized ML”
  $\rightarrow$ expressive language (higher order functions, local state, general recursion)
- Probabilistic cost credits (analogous to time credits)
  $\rightarrow$ amortized reasoning, local “expectation accounting”
- General cost analysis (time complexity, entropy, “ticks”, …)
  $\rightarrow$ user-definable cost models, *e.g.*, expected entropy use
- “Natural proofs”: symbolic execution & solving recurrence relations
  $\rightarrow$ case studies (qSort, hash tables, F-Y shuffle, meldable heaps, …), tactics

Some Definitions

The RandML language

A (sequential) ML-like language with higher-order (recursive) functions, higher-order state, ..., and probabilistic uniform sampling.

$$e \in Expr ::= \ \ldots \mid \mathsf{rand}\, e \mid \mathsf{tick}\, e$$

The RandML language

A (sequential) ML-like language with higher-order (recursive) functions, higher-order state, ..., and probabilistic uniform sampling.

$$e \in \mathit{Expr} ::= \ \ldots \mid \mathsf{rand}\, e \mid \mathsf{tick}\, e$$

Semantics given by monadic iteration of $\mathbf{step} : (\mathit{Expr} \times \mathit{State}) \to \mathcal{D}(\mathit{Expr} \times \mathit{State})$.

$$\mathbf{step}((\lambda x.\, e_1)\ e_2, \sigma)(e', \sigma') \triangleq \begin{cases} 1 & \text{if } e' = e_1[e_2/x] \text{ and } \sigma' = \sigma, \\ 0 & \text{otherwise,} \end{cases}$$

$$\mathbf{step}(\mathsf{rand}\, N, \sigma)(k, \sigma) \triangleq \begin{cases} \frac{1}{N+1} & \text{for } k \in \{0, 1, \ldots, N\}, \\ 0 & \text{otherwise.} \end{cases}$$

Cost Models

Definition

A function $cost : Expr \to \mathbb{R}_{\geq 0}$ is a *cost model* if $cost(K[e]) = cost(e)$.

Cost Models

Definition
A function $cost : Expr \to \mathbb{R}_{\geq 0}$ is a *cost model* if $cost(K[e]) = cost(e)$.

Condition required for bind rule:

$$\frac{\vdash \{P\}\ e\ \{v.Q\} \qquad \vdash \forall v.\ \{Q\}\ K[v]\ \{R\}}{\vdash \{P\}\ K[e]\ \{R\}}\ \text{HT-BIND}$$

Cost Models

Definition

A function $cost : Expr \to \mathbb{R}_{\geq 0}$ is a *cost model* if $cost(K[e]) = cost(e)$.

Examples:

$cost_{\text{all}} \triangleq \lambda\_.\, 1$

Cost Models

Definition

A function $cost : Expr \to \mathbb{R}_{\geq 0}$ is a *cost model* if $cost(K[e]) = cost(e)$.

Examples:

$$cost_{\text{all}} \triangleq \lambda\_.\, 1$$

$$cost_{\text{app}} \triangleq \lambda e.\, 1 \quad \text{if } \mathbf{decomp}(e) = e_1\; e_2 \quad \text{for some } e_1, e_2, \text{ and } \quad 0 \quad \text{otherwise.}$$

Here, **decomp** picks out the "head redex", *e.g.*,

$$\mathbf{decomp}(\text{let}\, x = !\,\ell \,\text{in}\, x + 1) = !\,\ell\,.$$

Cost Models

Definition
A function $cost : Expr \to \mathbb{R}_{\geq 0}$ is a *cost model* if $cost(K[e]) = cost(e)$.

Examples:

$$cost_{\mathsf{all}} \triangleq \lambda\_.\, 1$$

$$cost_{\mathsf{app}} \triangleq \lambda e.\, 1 \quad \text{if } \mathbf{decomp}(e) = e_1\ e_2 \quad \text{for some } e_1, e_2, \text{ and} \quad 0 \quad \text{otherwise.}$$

$$cost_{\mathsf{rand}} \triangleq \lambda e.\, \log_2(N+1) \quad \text{if } \mathbf{decomp}(e) = \mathsf{rand}\, N \quad \text{for some } N, \text{ and} \quad 0 \quad \text{otherwise.}$$

Here, **decomp** picks out the “head redex”, *e.g.*,

$$\mathbf{decomp}(\mathsf{let}\, x = !\,\ell\, \mathsf{in}\, x + 1) = !\,\ell\,.$$

Definition

A function $cost : Expr \to \mathbb{R}_{\geq 0}$ is a *cost model* if $cost(K[e]) = cost(e)$.

Examples:

$$
\begin{aligned}
cost_{\text{all}} &\triangleq \lambda\_.\, 1 \\
cost_{\text{app}} &\triangleq \lambda e.\, 1 \quad \text{if } \mathbf{decomp}(e) = e_1\ e_2 \quad \text{for some } e_1, e_2, \text{ and} \quad 0 \quad \text{otherwise.} \\
cost_{\text{rand}} &\triangleq \lambda e.\, \log_2(N+1) \quad \text{if } \mathbf{decomp}(e) = \mathsf{rand}\, N \quad \text{for some } N, \text{ and} \quad 0 \quad \text{otherwise.} \\
cost_{\text{tick}} &\triangleq \lambda e.\, |z| \quad \text{if } \mathbf{decomp}(e) = \mathsf{tick}\, z \quad \text{for some } z \in \mathbb{Z}, \text{ and} \quad 0 \quad \text{otherwise.}
\end{aligned}
$$

Here, **decomp** picks out the "head redex", *e.g.*,

$$\mathbf{decomp}(\mathsf{let}\, x = !\,\ell\ \mathsf{in}\ x + 1) = !\,\ell\,.$$

Expected Cost of a Program

Definition (Expected Cost)

$$\mathsf{EC}_n^{cost}(e,\sigma) \triangleq \begin{cases} 0 & \text{if } n = 0 \text{ or } e \in Val, \\ cost(e) + \mathbb{E}_{\mathsf{step}(e,\sigma)}\left[\mathsf{EC}_m^{cost}\right] & \text{if } n = m + 1. \end{cases}$$

Definition (Expected Cost)

$$\mathsf{EC}_n^{cost}(e,\sigma) \triangleq \begin{cases} 0 & \text{if } n = 0 \text{ or } e \in \mathit{Val}, \\ cost(e) + \mathbb{E}_{\mathsf{step}(e,\sigma)}\left[\mathsf{EC}_m^{cost}\right] & \text{if } n = m + 1. \end{cases}$$

$$\mathsf{EC}^{cost}(e,\sigma) \triangleq \sup_{n \in \omega} \mathsf{EC}_n^{cost}(e,\sigma) \quad \text{(or } +\infty \text{ if no bound exists)}$$

Expected Cost of a Program

Definition (Expected Cost)

$$\mathsf{EC}^{cost}_n(e,\sigma) \triangleq \begin{cases} 0 & \text{if } n = 0 \text{ or } e \in Val, \\ cost(e) + \mathbb{E}_{\mathsf{step}(e,\sigma)}\left[\mathsf{EC}^{cost}_m\right] & \text{if } n = m+1. \end{cases}$$

$$\mathsf{EC}^{cost}(e,\sigma) \triangleq \sup_{n\in\omega} \mathsf{EC}^{cost}_n(e,\sigma) \quad \text{(or } +\infty \text{ if no bound exists)}$$

where $\mathbb{E}_{\mathsf{step}(e,\sigma)}\left[\mathsf{EC}^{cost}_m\right] = \sum_{\rho\in Cfg} \mathsf{step}(e,\sigma)(\rho) \cdot \mathsf{EC}^{cost}_m(\rho)$

“The expectation of the random variable EC^{cost}_m over the distribution $\mathsf{step}(e,\sigma)$”

The Logic

Cost as a Resource

Cost resource algebra: $\text{Auth}(\mathbb{R}_{\geq 0}, +)$

For the user: Standard Iris plus one new assertion: $\$(x)$, fragmental part.

In the WP, use authoritative part ("cost interpretation"): $\$_\bullet(x)$.

Credit splitting rule

$$\$(x_1 + x_2) \dashv\vdash \$(x_1) \ast \$(x_2)$$

The Logic

- Standard Iris plus one new assertion: $\$(x)$.

The Logic

- Standard Iris plus one new assertion: $\$(x)$.
- Weakest precondition (and Hoare triples, rules) are parametrised by *cost*.

The Logic

- Standard Iris plus one new assertion: $\$(x)$.
- Weakest precondition (and Hoare triples, rules) are parametrised by *cost*.
- Standard rules! With added cost requirements.

$$\frac{}{\vdash \{\ell \mapsto v * \$(cost(!\,\ell))\}\ !\,\ell\ \{w.\, w = v * \ell \mapsto v\}}\ \text{HT-LOAD}$$

$$\frac{}{\{\$(cost(\mathsf{tick}\, z))\}\ \mathsf{tick}\, z\ \{().\, \mathrm{True}\}}\ \text{HT-TICK}$$

$$\frac{}{\vdash \{\$(cost(\mathsf{rand}\, N))\}\ \mathsf{rand}\, N\ \{n.\, 0 \leq n \leq N\}}\ \text{HT-RAND}$$

The Logic

- Standard Iris plus one new assertion: $\$(x)$.
- Weakest precondition (and Hoare triples, rules) are parametrised by *cost*.
- Standard rules! With added cost requirements.

$$\dfrac{}{\vdash \{\ell \mapsto v * \$(cost(!\,\ell))\}\ !\,\ell\ \{w.\ w = v * \ell \mapsto v\}}\ \text{HT-LOAD}$$

$$\dfrac{}{\{\$(cost(\mathsf{tick}\,z))\}\ \mathsf{tick}\,z\ \{().\ \mathsf{True}\}}\ \text{HT-TICK}$$

$$\dfrac{}{\vdash \{\$(cost(\mathsf{rand}\,N))\}\ \mathsf{rand}\,N\ \{n.\ 0 \le n \le N\}}\ \text{HT-RAND}$$

TOO WEAK! We expect better.

- Standard Iris plus one new assertion: $\$(x)$.
- Weakest precondition (and Hoare triples, rules) are parametrised by *cost*.
- Standard rules! With added cost requirements.

$$\frac{}{\vdash \{\ell \mapsto v * \$(cost(!\,\ell))\}\ !\,\ell\ \{w.\, w = v * \ell \mapsto v\}}\ \text{HT-LOAD}$$

$$\frac{}{\{\$(cost(\mathsf{tick}\, z))\}\ \mathsf{tick}\, z\ \{().\, \mathsf{True}\}}\ \text{HT-TICK}$$

$$\frac{}{\vdash \{\$(cost(\mathsf{rand}\, N))\}\ \mathsf{rand}\, N\ \{n.\, 0 \leq n \leq N\}}\ \text{HT-RAND}$$

TOO WEAK! We expect better.

Distributing Cost Credits in Expectation

$$\frac{cost(\text{rand}\ N) + \sum_{n=0}^{N} \frac{X_2(n)}{N+1} \leq x_1}{\{\$(x_1)\}\ \text{rand}\ N\ \{n.\ \$(X_2(n))\ *\ 0 \leq n \leq N\}}\ \text{HT-RAND-EXP}$$

$$\sum_{n=0}^{N} \frac{X_2(n)}{N+1} = \mathbb{E}_{\mathbf{step}(\text{rand}\ N)}[X_2]$$

Distributing Cost Credits in Expectation

$$\frac{cost(\mathsf{rand}\ N) + \sum_{n=0}^{N} \frac{X_2(n)}{N+1} \leq x_1}{\{\$(x_1)\}\ \mathsf{rand}\ N\ \{n.\ \$(X_2(n))\ *\ 0 \leq n \leq N\}}\ \textsc{HT-RAND-EXP}$$

$$\sum_{n=0}^{N} \frac{X_2(n)}{N+1} = \mathbb{E}_{\mathbf{step}(\mathsf{rand}\ N)}[X_2]$$

Derived rule:

$$\frac{\frac{1}{2} \cdot X_2(\mathsf{true}) + \frac{1}{2} \cdot X_2(\mathsf{false}) \leq x_1}{\{\$(cost(\mathsf{flip}))\ *\ \$(x_1)\}\ \mathsf{flip}\ \{b.\$(X_2(b))\}}\ \textsc{HT-FLIP-EXP}$$

Adequacy

Theorem

Let x be a non-negative real number and let φ be a predicate on values.
If $\vdash \{\$(x)\}\, e\, \{\varphi\}$ *then for any state σ,*

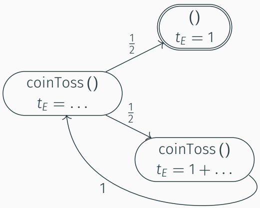
1. $\mathrm{EC}_{cost}(e, \sigma) \leq x$, *and*
2. $\forall v \in Val\,.\ \mathrm{exec}(e, \sigma)(v) > 0 \implies \varphi(v)$.

Examples

coinToss terminates in $t_E = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot (1 + t_E) = 2$

Let $\quad$ rec coinToss _ = tick 1; if flip then () else coinToss ()

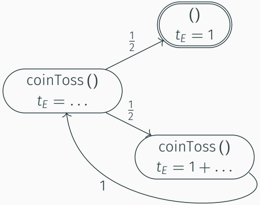
Evaluate:



coinToss terminates in $t_E = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot (1 + t_E) = 2$

Let $\quad$ rec coinToss _ = tick 1; if flip then () else coinToss ()

Evaluate:

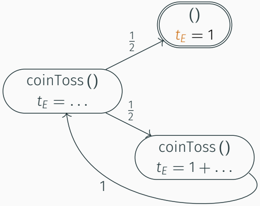


1. Instantiate Tachis with $cost_{\text{tick}}$ (could have used, *e.g.*, $cost_{\text{app}}$)
2. Prove $\{\$(2)\}$ coinToss () $\{\text{True}\}$.
3. By adequacy, $\mathbf{EC}_{cost_{\text{tick}}}(\text{coinToss}, \emptyset) \leq 2$.

coinToss terminates in $t_E = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot (1 + t_E) = 2$

Let $\quad \text{rec coinToss}\ \_ = \text{tick}\,1;\ \text{if flip then}\ ()\ \text{else coinToss}\ ()$

Evaluate:



1. Instantiate Tachis with $cost_{\text{tick}}$ (could have used, *e.g.*, $cost_{\text{app}}$)
2. Prove $\{\$(2)\}\ \text{coinToss}\ ()\ \{\text{True}\}$.
3. By adequacy, $\text{EC}_{cost_{\text{tick}}}(\text{coinToss}, \emptyset) \leq 2$.

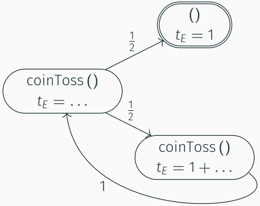
$$\frac{\frac{1}{2} \cdot X_2(\text{true}) + \frac{1}{2} \cdot X_2(\text{false}) \leq x_1}{\{\$(0)\ *\ \$(x_1)\}\ \text{flip}\ \{b.\,\$(X_2(b))\}}\ \text{HT-FLIP-EXP} \qquad \frac{}{\{\$(1)\}\ \text{tick}\,1\,\{().\,\text{True}\}}\ \text{HT-TICK}$$

Let $\quad X_2(b) \triangleq \text{if}\ b\ \text{then}\ 0\ \text{else}\ 2\,.$

coinToss terminates in $t_E = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot (1 + t_E) = 2$

Let $\quad \mathsf{rec\ coinToss\ \_ = tick\ 1;\ if\ flip\ then\ ()\ else\ coinToss\ ()}$

Evaluate:



1. Instantiate Tachis with $\mathit{cost}_{\mathsf{tick}}$ (could have used, *e.g.*, $\mathit{cost}_{\mathsf{app}}$)
2. Prove $\{\$(2)\}\ \mathsf{coinToss}\ ()\ \{\mathsf{True}\}$.
3. By adequacy, $\mathbf{EC}_{\mathit{cost}_{\mathsf{tick}}}(\mathsf{coinToss}, \emptyset) \leq 2$.

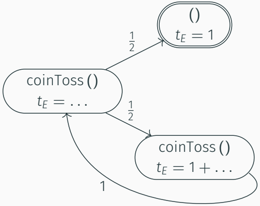
$$\frac{\frac{1}{2} \cdot X_2(\mathsf{true}) + \frac{1}{2} \cdot X_2(\mathsf{false}) \leq x_1}{\{\$(0) * \$(x_1)\}\ \mathsf{flip}\ \{b.\,\$(X_2(b))\}}\ \text{HT-FLIP-EXP} \qquad \frac{}{\{\$(1)\}\ \mathsf{tick}\ 1\ \{().\,\mathsf{True}\}}\ \text{HT-TICK}$$

Let $\quad X_2(b) \triangleq \text{if } b \text{ then } 0 \text{ else } 2\,.$

coinToss terminates in $t_E = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot (1 + t_E) = 2$

Let $\;\;\mathsf{rec}\ \mathrm{coinToss}\ \_ = \mathsf{tick}\,1;\ \mathsf{if}\ \mathsf{flip}\ \mathsf{then}\ ()\ \mathsf{else}\ \mathrm{coinToss}\ ()$

Evaluate:



1. Instantiate Tachis with $cost_{\mathsf{tick}}$ (could have used, *e.g.*, $cost_{\mathsf{app}}$)
2. Prove $\{\$(2)\}\ \mathrm{coinToss}\ ()\ \{\mathrm{True}\}$.
3. By adequacy, $\mathrm{EC}_{cost_{\mathsf{tick}}}(\mathrm{coinToss}, \emptyset) \leq 2$.

$$\frac{\frac{1}{2} \cdot X_2(\mathsf{true}) + \frac{1}{2} \cdot X_2(\mathsf{false}) \leq x_1}{\{\$(0) \ast \$(x_1)\}\ \mathsf{flip}\ \{b.\,\$(X_2(b))\}}\ \text{HT-FLIP-EXP} \qquad \frac{}{\{\$(1)\}\ \mathsf{tick}\,1\,\{().\,\mathrm{True}\}}\ \text{HT-TICK}$$

Let $\;\;X_2(b) \triangleq \text{if } b \text{ then } 0 \text{ else } 2\,.$

$\{\$(2)\}$

$(\mathsf{rec}\ \mathrm{coinToss}\ \_ = \mathsf{tick}\,1;\ \mathsf{if}\ \mathsf{flip}\ \mathsf{then}\ ()\ \mathsf{else}\ \mathrm{coinToss}\ ())\ ()$ split credits, Löb

$\{\$(2)\}$

$(\mathsf{rec}\ \mathsf{coinToss}\ \_ = \mathsf{tick}\,1;\ \mathsf{if}\ \mathsf{flip}\ \mathsf{then}\ ()\ \mathsf{else}\ \mathsf{coinToss}\ ())\ ()$ split credits, Löb

$\{\$(1) * \$(1) * \triangleright IH\}$ where $IH = \{\$(2)\}\ \mathsf{coinToss}\ ()\ \{\mathsf{True}\}$

$(\mathsf{rec}\ \mathsf{coinToss}\ \_ = \mathsf{tick}\,1;\ \mathsf{if}\ \mathsf{flip}\ \mathsf{then}\ ()\ \mathsf{else}\ \mathsf{coinToss}\ ())\ ()$ app, pay for tick

$\{\$(2)\}$

$(\mathsf{rec}\ \mathsf{coinToss}\ \_ = \mathsf{tick}\ 1;\ \mathsf{if}\ \mathsf{flip}\ \mathsf{then}\ ()\ \mathsf{else}\ \mathsf{coinToss}\ ())\ ()$ split credits, Löb

$\{\$(1) * \$(1) * \triangleright IH\}$ where $IH = \{\$(2)\}\ \mathsf{coinToss}\ ()\ \{\mathsf{True}\}$

$(\mathsf{rec}\ \mathsf{coinToss}\ \_ = \mathsf{tick}\ 1;\ \mathsf{if}\ \mathsf{flip}\ \mathsf{then}\ ()\ \mathsf{else}\ \mathsf{coinToss}\ ())\ ()$ app, pay for tick

$\{\$(1) * IH\}$

$\mathsf{if}\ \mathsf{flip}\ \mathsf{then}\ ()\ \mathsf{else}\ \mathsf{coinToss}\ ()$ HT-FLIP-EXP w/ X_2

$\{\$(2)\}$

$(\text{rec coinToss}\ \_ = \text{tick}\ 1;\ \text{if flip then}\ ()\ \text{else coinToss}\ ())\ ()$ — split credits, Löb

$\{\$(1) * \$(1) * \triangleright IH\}$ where $IH = \{\$(2)\}\ \text{coinToss}\ ()\ \{\text{True}\}$

$(\text{rec coinToss}\ \_ = \text{tick}\ 1;\ \text{if flip then}\ ()\ \text{else coinToss}\ ())\ ()$ — app, pay for tick

$\{\$(1) * IH\}$

$\text{if flip then}\ ()\ \text{else coinToss}\ ()$ — HT-FLIP-EXP w/ X_2

$b : \mathbb{B} \mid \{\$(\text{if}\ b\ \text{then}\ 0\ \text{else}\ 2) * IH\}$

$b : \mathbb{B} \mid \quad \text{if}\ b\ \text{then}\ ()\ \text{else coinToss}\ ()$ — case split on b

$\{\$(2)\}$

$(\text{rec coinToss } \_ = \text{tick } 1;\ \text{if flip then } ()\ \text{else coinToss } ())\ ()$ — split credits, Löb

$\{\$(1) * \$(1) * \triangleright IH\}$ where $IH = \{\$(2)\}\ \text{coinToss } ()\ \{\text{True}\}$

$(\text{rec coinToss } \_ = \text{tick } 1;\ \text{if flip then } ()\ \text{else coinToss } ())\ ()$ — app, pay for tick

$\{\$(1) * IH\}$

$\text{if flip then } ()\ \text{else coinToss } ()$ — HT-FLIP-EXP w/ X_2

$b : \mathbb{B} \mid \{\$(\text{if } b \text{ then } 0 \text{ else } 2) * IH\}$

$b : \mathbb{B} \mid \quad \text{if } b \text{ then } ()\ \text{else coinToss } ()$ — case split on b

$b = \text{true}: \quad \{\$(0) * IH\}$

$()$ — done

$\{\$(2)\}$

$\quad (\text{rec coinToss}\ \_ = \text{tick}\ 1;\ \text{if flip then}\ ()\ \text{else coinToss}\ ())\ ()$ — split credits, Löb

$\{\$(1) * \$(1) * \triangleright IH\}$ — where $IH = \{\$(2)\}\ \text{coinToss}\ ()\ \{\text{True}\}$

$\quad (\text{rec coinToss}\ \_ = \text{tick}\ 1;\ \text{if flip then}\ ()\ \text{else coinToss}\ ())\ ()$ — app, pay for tick

$\{\$(1) * IH\}$

$\quad \text{if flip then}\ ()\ \text{else coinToss}\ ()$ — HT-FLIP-EXP w/ X_2

$b : \mathbb{B} \mid \{\$(\text{if}\ b\ \text{then}\ 0\ \text{else}\ 2) * IH\}$

$b : \mathbb{B} \mid \quad \text{if}\ b\ \text{then}\ ()\ \text{else coinToss}\ ()$ — case split on b

$b = \text{true}$: $\{\$(0) * IH\}$

$\qquad ()$ — done

$b = \text{false}$: $\{\$(2) * \{\$(2)\}\ \text{coinToss}\ ()\ \{\text{True}\}\}$

$\qquad \text{coinToss}\ ()$ — by IH with $\$(2)$

$\{\text{True}\}$

$\{\$(2)\}$

$\quad (\mathsf{rec}\ \mathsf{coinToss}\ \_ = \mathsf{tick}\ 1;\ \mathsf{if}\ \mathsf{flip}\ \mathsf{then}\ ()\ \mathsf{else}\ \mathsf{coinToss}\ ())\ ()$ split credits, Löb

$\{\$(1) * \$(1) * \triangleright IH\}$ where $IH = \{\$(2)\}\ \mathsf{coinToss}\ ()\ \{\mathsf{True}\}$

$\quad (\mathsf{rec}\ \mathsf{coinToss}\ \_ = \mathsf{tick}\ 1;\ \mathsf{if}\ \mathsf{flip}\ \mathsf{then}\ ()\ \mathsf{else}\ \mathsf{coinToss}\ ())\ ()$ app, pay for tick

$\{\$(1) * IH\}$

$\quad \mathsf{if}\ \mathsf{flip}\ \mathsf{then}\ ()\ \mathsf{else}\ \mathsf{coinToss}\ ()$ HT-FLIP-EXP w/ X_2

$b : \mathbb{B} \mid \{\$(\text{if } b \text{ then } 0 \text{ else } 2) * IH\}$

$b : \mathbb{B} \mid \quad \mathsf{if}\ b\ \mathsf{then}\ ()\ \mathsf{else}\ \mathsf{coinToss}\ ()$ case split on b

$b = \mathsf{true}$: $\{\$(0) * IH\}$

$\qquad ()$ done

$b = \mathsf{false}$: $\{\$(2) * \{\$(2)\}\ \mathsf{coinToss}\ ()\ \{\mathsf{True}\}\}$

$\qquad \mathsf{coinToss}\ ()$ by IH with $\$(2)$

$\{\mathsf{True}\}$ **Qed.**

Entropy Usage in Rejection Sampling

Want: Uniform distribution on three elements $\mathsf{unif}(0,2) = \{0 : \frac{1}{3}, 1 : \frac{1}{3}, 2 : \frac{1}{3}\}$.

Entropy Usage in Rejection Sampling

Want: Uniform distribution on three elements $\mathbf{unif}(0, 2) = \{0 : \frac{1}{3}, 1 : \frac{1}{3}, 2 : \frac{1}{3}\}$.

Implement via naïve rejection sampler:

$$\begin{aligned}\text{rec sampleThree } \_ = \text{let } v = (\text{rand } 1) + 2 * (\text{rand } 1) \text{ in}\\ \text{if } v < 3 \text{ then } v \text{ else sampleThree } ()\end{aligned}$$

Entropy Usage in Rejection Sampling

Want: Uniform distribution on three elements $\textbf{unif}(0,2) = \{0 : \frac{1}{3}, 1 : \frac{1}{3}, 2 : \frac{1}{3}\}$.

Implement via naïve rejection sampler:

$$\text{rec sampleThree } \_ = \text{let } v = \Big(\text{rand } 1\Big) + 2 * \Big(\text{rand } 1\Big) \text{ in}$$

$$\text{if } v < 3 \text{ then } v \text{ else sampleThree } ()$$

Entropy Usage in Rejection Sampling

Want: Uniform distribution on three elements $\mathbf{unif}(0,2) = \{0 : \frac{1}{3}, 1 : \frac{1}{3}, 2 : \frac{1}{3}\}$.

Implement via naïve rejection sampler:

$$\text{rec sampleThree } \_ = \text{let } v = \Big(\text{rand } 1\Big) + 2 * \Big(\text{rand } 1\Big) \text{ in}$$

$$\text{if } v < 3 \text{ then } v \text{ else sampleThree } ()$$

Entropy of a distribution μ is defined as $H(\mu) = -\sum_{x \in |\mu|} \mu(x) \cdot \log_2 \mu(x)$.

$\mathbf{unif}(0,2)$ has entropy $\log_2 3 \approx 1.6$.

Entropy Usage in Rejection Sampling

Want: Uniform distribution on three elements $\mathbf{unif}(0,2) = \{0 : \frac{1}{3}, 1 : \frac{1}{3}, 2 : \frac{1}{3}\}$.

Implement via naïve rejection sampler:

$$\text{rec sampleThree } \_ = \text{let } v = (\text{rand } 1) + 2 * (\text{rand } 1) \text{ in}$$

$$\text{if } v < 3 \text{ then } v \text{ else sampleThree } ()$$

Entropy of a distribution μ is defined as $H(\mu) = -\sum_{x \in |\mu|} \mu(x) \cdot \log_2 \mu(x)$.

$\mathbf{unif}(0,2)$ has entropy $\log_2 3 \approx 1.6$.

Let $E = \mathbf{EC}\,(\text{sampleThree}\,())$.

Recurrence: $E = \frac{3}{4}2 + \frac{1}{4}(2 + E) = \frac{4}{3}(\frac{6}{4} + \frac{2}{4}) = \frac{8}{3} = 2.666\ldots$

With $cost_{\text{rand}}$, we can prove $\{\$\left(\frac{8}{3}\right)\}$ sampleThree () $\{n.\, 0 \leq n \leq 2\}$.

Amortized Expected Entropy: Batch Sampling

```
rec prefetch mem =
  let v = flipN 8 in
  if  v < 243  then
    v
  else prefetch m
```

Idea: to batch 5 samplings, pick $v \in [0, 3^5[$. Flip 8 coins.

Since $3^5 = 243$ is close to $2^8 = 256$, not much entropy is wasted.

Usual recurrence analysis gives $E = 243/256 \cdot 8 + 13/256 \cdot (8 + E) = \frac{256*8}{243} \approx 8.4$.

Amortized Expected Entropy: Batch Sampling

$\{\$ \left(\frac{256 \cdot 8}{243}\right)\}$

```
rec prefetch mem =
  let v = flipN 8 in
  if v < 243 then
    v
  else prefetch m
```

$() \quad \{n. (0 \leq n < 243)\}$

Idea: to batch 5 samplings, pick $\mathsf{v} \in [0, 3^5[$. Flip 8 coins.

Since $3^5 = 243$ is close to $2^8 = 256$, not much entropy is wasted.

Usual recurrence analysis gives $E = 243/256 \cdot 8 + 13/256 \cdot (8 + E) = \frac{256*8}{243} \approx 8.4$.

Again with $cost_{\mathsf{rand}}$, we can prove this in Tachis.

Amortized Expected Entropy: Batch Sampling

$$
\begin{aligned}
\mathsf{initSampler} \triangleq\ & \mathsf{let\ mem} = \mathsf{ref}\,0\ \mathsf{in} \\
& \mathsf{let\ cnt} = \mathsf{ref}\,0\ \mathsf{in} \\
& \lambda\ \_.\ \mathsf{if}\ !\,\mathsf{cnt} == 0\ \mathsf{then} \\
& \qquad (\mathsf{mem} \leftarrow \mathsf{prefetch}\ ()\,;\ \mathsf{cnt} \leftarrow 5)\,; \\
& \quad \mathsf{let\ v} = !\,\mathsf{mem}\ \mathsf{in} \\
& \quad \mathsf{cnt} \leftarrow !\,\mathsf{cnt} - 1\,; \\
& \quad \mathsf{mem} \leftarrow \mathsf{v}\ \grave{}\mathsf{quot}\grave{}\ 3\,; \\
& \quad \mathsf{v}\ \grave{}\mathsf{mod}\grave{}\ 3
\end{aligned}
$$

Idea: draw 5 samples at once, reveal one at a time.

Amortized expected cost should be $E/5 = \frac{256 \cdot 8}{243 \cdot 5} \approx 1.7$, much closer to 1.6 than 2.66... was.

Amortized Reasoning via First-Class Credits

Extracting a sample in $\{0, 1, 2\}$ does not have constant costs:

$$E, 0, 0, 0, 0, E, 0, 0, \ldots$$

Worst-case bound is higher than naive rejection sampling, but the average is lower.

Amortized Reasoning via First-Class Credits

Extracting a sample in $\{0, 1, 2\}$ does not have constant costs:

$$E, 0, 0, 0, 0, E, 0, 0, \ldots$$

Worst-case bound is higher than naive rejection sampling, but the average is lower.

With amortized point of view:

$$\text{Setup: } \frac{4 \cdot E}{5}, \text{ then: } \frac{E}{5}, \frac{E}{5}, \frac{E}{5}, \frac{E}{5}, \frac{E}{5}, \frac{E}{5}, \ldots$$

Intuition: pay "extra" per operation, to pay for costly E operations.

Amortized Expected Entropy: Batch Sampling

Again with $cost_{\text{rand}}$, we can prove

$$\{\$\left(4 \cdot \tfrac{256 \cdot 8}{243 \cdot 5}\right)\}\ \text{initSampler}\ \{f.\ \pi * \{\$\left(\tfrac{256 \cdot 8}{243 \cdot 5}\right) * \pi\}\ f\ ()\ \{n.\ (0 \leq n < 3) * \pi\}\}$$

where $\pi \triangleq \exists \text{cnt}, c, \text{mem}, m.\ \text{cnt} \mapsto c * c < 5 * \text{mem} \mapsto m * m < 3^c * \$\left((4 - c) \cdot \tfrac{256 \cdot 8}{243 \cdot 5}\right)$.

$$\begin{aligned}
\text{initSampler} \triangleq\ & \textbf{let}\ \text{mem} = \textbf{ref}\ 0\ \textbf{in} \\
& \textbf{let}\ \text{cnt} = \textbf{ref}\ 0\ \textbf{in} \\
& \lambda\ _.\ \textbf{if}\ !\,\text{cnt} == 0\ \textbf{then} \\
& \qquad (\text{prefetch mem};\ \text{cnt} \leftarrow 5); \\
& \quad \textbf{let}\ \text{v} = !\,\text{mem}\ \textbf{in} \\
& \quad \text{cnt} \leftarrow !\,\text{cnt} - 1; \\
& \quad \text{mem} \leftarrow \text{v}\ \grave{}\text{quot}\grave{}\ 3; \\
& \quad \text{v}\ \grave{}\text{mod}\grave{}\ 3
\end{aligned}$$

Interesting and realistic examples:

- Coupon Collector (ht-rand-exp)
- Fisher-Yates Shuffle (expected entropy w/o $\log_2$ in lang)
- Batch Sampling (expected entropy, amortization)
- quicksort (time and entropy, reusable recurrence reasoning)
- hash map (”deref cost” via tick, amortised for put & get)
- meldable heaps (nr. of cmp)
- k-way merge (heap client)

The Model

The Weakest Precondition

$$\begin{aligned}\mathrm{wp}\ e_1\ \{\Phi\} \triangleq\ & (e_1 \in Val \wedge \Phi(e_1)) \\ & \vee (e_1 \notin Val \wedge \forall \sigma_1, x_1.\, S(\sigma_1) * \$_{\bullet}(x_1) \mathrel{-\!\!*} \\ & \quad \mathrm{ECM}(\underbrace{(e_1,\ \sigma_1)}_{\rho_1},\ x_1,\ \underbrace{(\lambda\, e_2, \sigma_2, x_2\ .\ \triangleright(S(\sigma_2) * \$_{\bullet}(x_2)\ * \mathrm{wp}\ e_2\ \{\Phi\}))}_{Z})\end{aligned}$$

$\$_{\bullet}(x_1)$ connects $\$(x)$ to operational semantics of e_1.

The Weakest Precondition

$$\begin{aligned}\mathrm{wp}\ e_1\ \{\Phi\} \triangleq\ & (e_1 \in \mathit{Val} \wedge \Phi(e_1)) \\ & \vee (e_1 \notin \mathit{Val} \wedge \forall \sigma_1, x_1.\ S(\sigma_1) * \$_{\bullet}(x_1) \mathrel{-\!\!*} \\ & \quad \mathrm{ECM}(\underbrace{(e_1,\ \sigma_1)}_{\rho_1},\ x_1,\ \underbrace{(\lambda\, e_2, \sigma_2, x_2\ .\ \triangleright(S(\sigma_2) * \$_{\bullet}(x_2)\ * \mathrm{wp}\ e_2\ \{\Phi\}))}_{Z}))\end{aligned}$$

$\$_{\bullet}(x_1)$ connects $\$(x)$ to operational semantics of e_1.

$$\text{where } \mathrm{ECM}(\rho_1, x_1, Z) \triangleq \exists\ (X_2 : \mathit{Cfg} \to \mathbb{R}_{\geq 0})\ . \tag{1}$$

$$\mathrm{red}(\rho_1) * \exists r.\ \forall \rho_2.\ X_2(\rho_2) \leq r\ * \tag{2}$$

$$\mathit{cost}(\rho_1) + \textstyle\sum_{\rho_2 \in \mathit{Cfg}} \mathsf{step}(\rho_1)(\rho_2) \cdot X_2(\rho_2) \leq x_1\ * \tag{3}$$

$$\forall \rho_2.\ \mathsf{step}(\rho_1)(\rho_2) > 0 \mathrel{-\!\!*} Z(\rho_2, X_2(\rho_2)) \tag{4}$$

The Weakest Precondition

$$
\begin{aligned}
\mathrm{wp}\; e_1\,\{\Phi\} \triangleq\; & (e_1 \in Val \wedge \Phi(e_1)) \\
& \vee (e_1 \notin Val \wedge \forall \sigma_1, x_1.\, S(\sigma_1) * \$_\bullet(x_1) \mathrel{-\!\!*} \\
& \quad \mathrm{ECM}(\underbrace{(e_1,\ \sigma_1)}_{\rho_1},\ x_1,\ \underbrace{(\lambda\, e_2, \sigma_2, x_2\ .\ \triangleright(S(\sigma_2) * \$_\bullet(x_2)\ * \mathrm{wp}\; e_2\,\{\Phi\})))}_{Z}
\end{aligned}
$$

$\$_\bullet(x_1)$ connects $\$(x)$ to operational semantics of e_1.

$$
\begin{aligned}
\text{where } \mathrm{ECM}(\rho_1, x_1, Z) \triangleq\; & \exists\, (X_2 : Cfg \to \mathbb{R}_{\geq 0})\ . && (1) \\
& \mathrm{red}(\rho_1) * \exists r.\, \forall \rho_2.\, X_2(\rho_2) \leq r\, * && (2) \\
& cost(\rho_1) + \textstyle\sum_{\rho_2 \in Cfg} \mathrm{step}(\rho_1)(\rho_2) \cdot X_2(\rho_2) \leq x_1\, * && (3) \\
& \forall \rho_2.\ \mathrm{step}(\rho_1)(\rho_2) > 0 \mathrel{-\!\!*} Z(\rho_2, X_2(\rho_2)) && (4)
\end{aligned}
$$

Future Work

Read the Tachis paper at arxiv:2405.20083 !

- Contextual equivalence for probabilistic polynomial time
- Work complexity for concurrency (instead of randomisation)
  our cost models should be flexible enough to deal, e.g., spin locks
- (work,span) for fork/join parallelism à la Parallel ML
- Are there evaluation context sensitive cost models that anyone cares for?
- Cost *variance* instead of expectation?
- Tail bounds (“with high probability, the cost is below some bound”)
- Unified story for “composition in expectation” (Eris / Tachis / ...)

Cost Resource Algebra Laws

Unital Resource Algebra: $\text{Auth}(\mathbb{R}_{\geq 0}, +)$
Cost Interpretation: $\$_{\bullet}(x_1)$
Cost Budget $\$(x)$

Agreement rule $\$(x_1) \ast \$_{\bullet}(x_2) \vdash x_1 \preceq x_2$

Spending rule: update $\$(x_1) \ast \$_{\bullet}(x_1 + x_2)$ to $\$(x_2)$

Acquisition rule: updating $\$_{\bullet}(x_1)$ to $\$(x_2) \ast \$_{\bullet}(x_1 + x_2)$.

Splitting rule: $\$(x_1 + x_2) \dashv\vdash \$(x_1) \ast \$(x_2)$