A Calculus for Amortized Expected Runtimes

Kevin Batz, Benjamin Lucien Kaminski,
Joost-Pieter Katoen, Christoph Matheja, Lena Verscht

January 19, 2023, POPL 2023

Separation Logic-Style Reasoning for **Amortized Expected Runtimes**
of **Probabilistic Data Structures** à la Dijkstra and Tarjan

Separation Logic-Style Reasoning for **Amortized Expected Runtimes**
of **Probabilistic Data Structures** à la Dijkstra and Tarjan

Our calculus features

- **compositionality**,

Separation Logic-Style Reasoning for **Amortized Expected Runtimes**
of **Probabilistic Data Structures** à la Dijkstra and Tarjan

Our calculus features

- **compositionality**,
- **local reasoning** by utilizing Runtime Separation Logic [Matheja 2020, Haslbeck 2021],

Separation Logic-Style Reasoning for **Amortized Expected Runtimes**
of **Probabilistic Data Structures** à la Dijkstra and Tarjan

Our calculus features

- **compositionality**,
- **local reasoning** by utilizing Runtime Separation Logic [Matheja 2020, Haslbeck 2021],
- **invariant-based** analysis of (possibly unbounded) loops,

Separation Logic-Style Reasoning for **Amortized Expected Runtimes**
of **Probabilistic Data Structures** à la Dijkstra and Tarjan

Our calculus features

- **compositionality**,
- **local reasoning** by utilizing Runtime Separation Logic [Matheja 2020, Haslbeck 2021],
- **invariant-based** analysis of (possibly unbounded) loops,
- **soundness** w.r.t. operational Markov Decision Process semantics.

Amortized Expected Runtimes

Probabilistic operation C:

$$
\begin{array}{l}
\texttt{if}\ (c = s)\ \{ \\
\quad \{\, \texttt{tick}\,(2 \cdot s) \,\}\ [1/2]\ \{\, \texttt{skip} \,\}; \\
\quad s \coloneqq 2 \cdot s \\
\}; \\
\texttt{tick}\,(1); c \coloneqq c + 1
\end{array}
$$

Amortized Expected Runtimes

Probabilistic operation C:

$$
\begin{array}{l}
\texttt{if}\ (c = s)\ \{ \\
\quad \{\, \texttt{tick}\,(2 \cdot s) \,\}\ [1/2]\ \{\, \texttt{skip} \,\}; \\
\quad s \coloneqq 2 \cdot s \\
\}; \\
\texttt{tick}\,(1); c \coloneqq c + 1
\end{array}
$$

Worst case expected runtime of C is $s + 1$.

Amortized Expected Runtimes

Probabilistic operation C:

$$
\begin{array}{l}
\texttt{if}\ (c = s)\ \{ \\
\quad \{\, \texttt{tick}\,(2 \cdot s) \,\}\ [1/2]\ \{\, \texttt{skip} \,\}\,; \\
\quad s \coloneqq 2 \cdot s \\
\};\\
\texttt{tick}\,(1)\,; c \coloneqq c + 1
\end{array}
$$

Worst case expected runtime of C is $s + 1$.

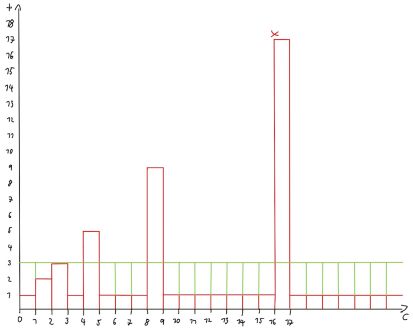
Amortized (= average) expected runtime of C in $\overbrace{C; \ldots; C}^{n \text{ times}}$ is:

$$\frac{\mathsf{ert}(C; \ldots; C)}{n} \leq 3 \qquad \text{(starting in } c = 0, s = 1)$$

Amortized Expected Runtimes

Probabilistic operation C:

```
if (c = s) {
    { tick(2 · s) } [1/2] { skip };
    s := 2 · s
};
tick(1); c := c + 1
```



Worst case expected runtime of C is $s + 1$.

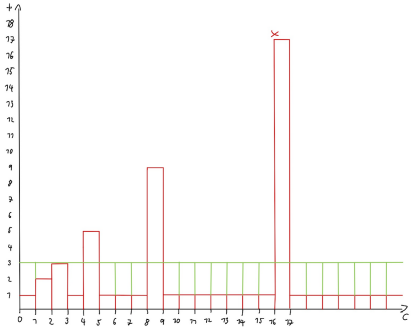
Amortized (= average) expected runtime of C in $\overbrace{C; \ldots; C}^{n \text{ times}}$ is:

$$\frac{\mathsf{ert}(C; \ldots; C)}{n} \leq 3 \qquad \text{(starting in } c = 0, s = 1)$$

Amortized Expected Runtimes

Probabilistic operation C:

```
if (c = s) {
    { tick(2 · s) } [1/2] { skip };
    s := 2 · s
};
tick(1); c := c + 1
```



Worst case expected runtime of C is $s + 1$.

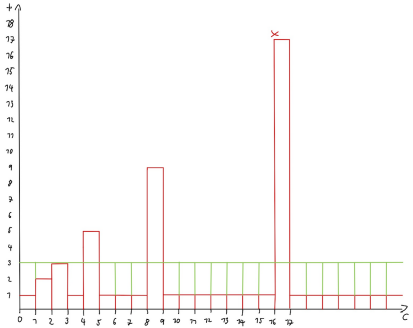
Amortized (= average) expected runtime of C in $\overbrace{C; \ldots; C}^{n \text{ times}}$ is:

$$\frac{\mathsf{ert}(C; \ldots; C)}{n} \leq 3 \qquad \text{(starting in } c = 0, s = 1)$$

Amortized Expected Runtimes

Probabilistic operation C:

```
if (c = s) {
    { tick(2 · s) } [1/2] { skip };
    s := 2 · s
};
tick(1); c := c + 1
```

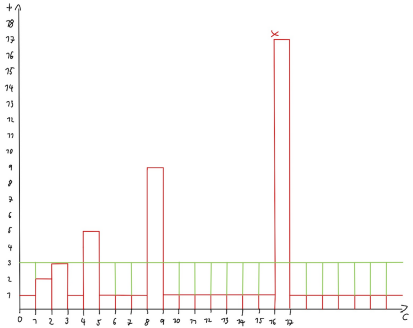


Redistribute runtimes by **potential function** π: States $\to \mathbb{R}_{\geq 0}$ **with** $\pi = 2 \cdot c \mathbin{\dot{-}} s$**.**

Amortized Expected Runtimes

Probabilistic operation C:

```
if (c = s) {
    { tick(2 · s) } [1/2] { skip };
    s := 2 · s
};
tick(1); c := c + 1
```



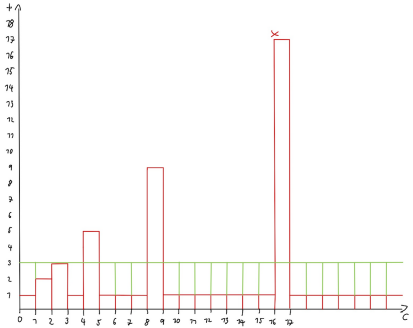
Redistribute runtimes by **potential function** $\pi\colon \mathsf{States} \to \mathbb{R}_{\geq 0}$ **with** $\pi = 2 \cdot c \mathbin{\dot{-}} s$**.**

$$\mathsf{aert}_\pi(C) \;=\; \mathsf{ert}(C) + \text{expected change of } \pi$$

Amortized Expected Runtimes

Probabilistic operation C:

```
if (c = s) {
    { tick(2 · s) } [1/2] { skip };
    s := 2 · s
};
tick(1); c := c + 1
```



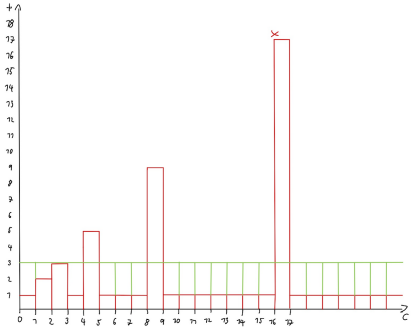
Redistribute runtimes by **potential function** $\pi\colon \mathsf{States} \to \mathbb{R}_{\geq 0}$ **with** $\pi = 2 \cdot c \mathbin{\dot{-}} s$**.**

$$\mathsf{aert}_\pi(C) \;=\; \mathsf{ert}(C) + \text{expected change of } \pi \;\leq\; 3$$

Amortized Expected Runtimes

Probabilistic operation C:

$$
\begin{array}{l}
\texttt{if}\ (c = s)\ \{ \\
\quad \{\, \texttt{tick}\,(2 \cdot s) \,\}\ [1/2]\ \{\, \texttt{skip} \,\}; \\
\quad s \coloneqq 2 \cdot s \\
\}; \\
\texttt{tick}\,(1); c \coloneqq c + 1
\end{array}
$$



Redistribute runtimes by **potential function** $\pi\colon \mathsf{States} \to \mathbb{R}_{\geq 0}$ **with** $\pi = 2 \cdot c \mathbin{\dot{-}} s$**.**

$$\mathsf{aert}_\pi(C) \;=\; \mathsf{ert}(C) + \text{expected change of } \pi \;\leq\; 3$$

If initially $\pi = 0$, then $\mathsf{ert}(C; \ldots; C) \;\leq\; \mathsf{aert}_\pi(C; \ldots; C) \;=\; 3 \cdot n$.

Motivating Case Study

The Insert-Delete-FindAny Problem [Brodal et al. '96]

Maintain a dictionary D of numbers providing operations:

- $insert(y)$
- $delete(x)$

The Insert-Delete-FindAny Problem [Brodal et al. '96]

Maintain a dictionary D of numbers providing operations:

- $insert(y)$
- $delete(x)$
- FindAny: return *arbitrary* element $y \in D$ and $rank(y) = 1 + |\{y' \in D \mid y' < y\}|$.

The Insert-Delete-FindAny Problem [Brodal et al. '96]

Maintain a dictionary D of numbers providing operations:

- $insert(y)$
- $delete(x)$
- FindAny: return *arbitrary* element $y \in D$ and $rank(y) = 1 + |\{y' \in D \mid y' < y\}|$.

No deterministic implementation can run in constant amortized runtime.

The Insert-Delete-FindAny Problem [Brodal et al. '96]

Maintain a dictionary D of numbers providing operations:

- $insert(y)$
- $delete(x)$
- FindAny: return *arbitrary* element $y \in D$ and $rank(y) = 1 + |\{y' \in D \mid y' < y\}|$.

No deterministic implementation can run in constant amortized runtime.

Utilizing **randomization: constant amortized expected runtime.**

The Insert-Delete-FindAny Problem [Brodal et al. '96]

Maintain a dictionary D of numbers providing operations:

- $insert(y)$
- $delete(x)$
- FindAny: return *arbitrary* element $y \in D$ and $rank(y) = 1 + |\{y' \in D \mid y' < y\}|$.

No deterministic implementation can run in constant amortized runtime.

Utilizing **randomization: constant amortized expected runtime.**

Randomization + Amortized Analysis + Pointers

Motivating Case Study

insert(y) :

add (y) ;
{
    $any \coloneqq H; Rank$
} $[1/len+1]$ {
    `if` $(len \geq 2)$ {
        $v_{any} \coloneqq \langle any + 2 \rangle$;
        `tick` (1) ;
        `if` $(y < v_{any})$ $\{rank \coloneqq rank + 1\}$
    } `else` {
        $any \coloneqq H$; $rank \coloneqq 1$
    }
}

delete(y) :

remove (x) ;
`if` $(len = 0)$ {
    $any \coloneqq 0$; $rank \coloneqq 0$
} `else` {
    `if` $(x = any)$ {
        Sample ; *Rank*
    } `else` {
        $v_x \coloneqq \langle x + 2 \rangle$; $v_{any} \coloneqq \langle any + 2 \rangle$;
        `tick` (1) ;
        `if` $(v_x < v_{any})$ $\{rank \coloneqq rank \dot{-} 1\}$
    }
}; `free`$(x, x+1, x+2)$

Motivating Case Study

insert (y) :

```
add (y);
{
    any := H; Rank
} [1/len+1] {
    if ( len ≥ 2 ) {
        v_any := ⟨any + 2⟩;
        tick (1);
        if ( y < v_any ) {rank := rank + 1 }
    } else {
        any := H ; rank := 1
    }
}
```

delete (y) :

```
remove (x);
if ( len = 0 ) {
    any := 0 ; rank := 0
} else {
    if ( x = any ) {
        Sample ; Rank
    } else {
        v_x := ⟨x + 2⟩ ; v_any := ⟨any + 2⟩;
        tick (1);
        if ( v_x < v_any ) {rank := rank ∸ 1 }
    }
};free(x, x + 1, x + 2)
```

insert (y) and *delete* (x) are **memory-safe** and run in **constant amortized expected time**.

The aert Calculus

The aert Calculus

$$
\begin{array}{lll}
C & \longrightarrow & \texttt{tick}\,(e) \qquad\qquad \text{time consumption} \\
& \mid & x \coloneqq a \qquad\qquad \text{assignment} \\
& \mid & \{\,C\,\}\,[\,p\,]\,\{\,C\,\} \qquad\qquad \text{probabilistic choice} \\
& \mid & x \coloneqq \texttt{alloc}\,(e) \qquad\qquad \text{memory allocation} \\
& \mid & x \coloneqq \langle e \rangle \qquad\qquad \text{heap lookup} \\
& \mid & \langle e \rangle \coloneqq e' \qquad\qquad \text{heap mutation} \\
& \mid & \texttt{free}(e) \qquad\qquad \text{memory deallocation} \\
& \mid & C\,;\,C \\
& \mid & \texttt{if}\,(\,\varphi\,)\,\{\,C\,\}\,\texttt{else}\,\{\,C\,\} \\
& \mid & \texttt{while}\,(\,\varphi\,)\,\{\,C'\,\}
\end{array}
$$

Fix a *potential function* π : States $\to \mathbb{R}_{\geq 0}$.

Fix a *potential function* $\pi\colon \mathsf{States} \to \mathbb{R}_{\geq 0}$. Complete lattice $(\mathbb{A}_\pi, \leq)$ of π-runtimes:

$$\mathbb{A}_\pi \;=\; \{X\colon \mathsf{States} \to \mathbb{R} \cup \{\infty\} \;\mid\; \forall \sigma\colon\; -\pi(\sigma) \leq X(\sigma)\}$$

The aert Calculus

Fix a *potential function* $\pi\colon \mathsf{States} \to \mathbb{R}_{\geq 0}$. Complete lattice $(\mathbb{A}_\pi, \leq)$ of π-runtimes:

$$\mathbb{A}_\pi \;=\; \{X\colon \mathsf{States} \to \mathbb{R} \cup \{\infty\} \mid \forall \sigma\colon -\pi(\sigma) \leq X(\sigma)\} \qquad X \leq Y \text{ iff } \forall \sigma\colon X(\sigma) \leq Y(\sigma)$$

The aert Calculus

Fix a *potential function* $\pi\colon \mathsf{States} \to \mathbb{R}_{\geq 0}$. Complete lattice $(\mathbb{A}_\pi, \leq)$ of π-runtimes:

$$\mathbb{A}_\pi \;=\; \{X\colon \mathsf{States} \to \mathbb{R} \cup \{\infty\} \mid \forall \sigma\colon -\pi(\sigma) \leq X(\sigma)\} \qquad X \leq Y \text{ iff } \forall \sigma\colon X(\sigma) \leq Y(\sigma)$$

Goal:

$$\mathsf{aert}_\pi\,[\![C]\!] : \mathbb{A}_\pi \to \mathbb{A}_\pi,$$

The aert Calculus

Fix a *potential function* $\pi\colon \mathsf{States} \to \mathbb{R}_{\geq 0}$. Complete lattice $(\mathbb{A}_\pi, \leq)$ of π-runtimes:

$$\mathbb{A}_\pi \;=\; \{X\colon \mathsf{States} \to \mathbb{R} \cup \{\infty\} \mid \forall \sigma\colon -\pi(\sigma) \leq X(\sigma)\} \qquad X \leq Y \text{ iff } \forall \sigma\colon X(\sigma) \leq Y(\sigma)$$

Goal:

$$\mathsf{aert}_\pi\,[\![C]\!] : \mathbb{A}_\pi \to \mathbb{A}_\pi, \qquad \mathsf{aert}_\pi\,[\![C]\!]\,(0)$$

The aert Calculus

Fix a *potential function* $\pi\colon \mathsf{States} \to \mathbb{R}_{\geq 0}$. Complete lattice $(\mathbb{A}_\pi, \leq)$ of π-runtimes:

$$\mathbb{A}_\pi \;=\; \{X\colon \mathsf{States} \to \mathbb{R} \cup \{\infty\} \mid \forall \sigma\colon -\pi(\sigma) \leq X(\sigma)\} \qquad X \leq Y \text{ iff } \forall \sigma\colon X(\sigma) \leq Y(\sigma)$$

Goal:

$$\mathsf{aert}_\pi\,[\![\mathrm{C}]\!] : \mathbb{A}_\pi \to \mathbb{A}_\pi, \qquad \mathsf{aert}_\pi\,[\![\mathrm{C}]\!]\,(0)\,(\sigma)$$

The aert Calculus

Fix a *potential function* $\pi\colon \mathsf{States} \to \mathbb{R}_{\geq 0}$. Complete lattice $(\mathbb{A}_\pi, \leq)$ of π-runtimes:

$$\mathbb{A}_\pi \;=\; \{X\colon \mathsf{States} \to \mathbb{R} \cup \{\infty\} \mid \forall \sigma\colon -\pi(\sigma) \leq X(\sigma)\} \qquad X \leq Y \text{ iff } \forall \sigma\colon X(\sigma) \leq Y(\sigma)$$

Goal:

$$\mathsf{aert}_\pi\,[\![C]\!] : \mathbb{A}_\pi \to \mathbb{A}_\pi, \qquad \mathsf{aert}_\pi\,[\![C]\!]\,(0)\,(\sigma) \;=\; \begin{array}{c}\text{amortized expected runtime of } C \\ \text{w.r.t. } \pi \text{ on input } \sigma\end{array}$$

Fix a *potential function* $\pi\colon \mathsf{States} \to \mathbb{R}_{\geq 0}$. Complete lattice $(\mathbb{A}_\pi, \leq)$ of π-runtimes:

$$\mathbb{A}_\pi \;=\; \{X\colon \mathsf{States} \to \mathbb{R} \cup \{\infty\} \mid \forall \sigma\colon\; -\pi(\sigma) \leq X(\sigma)\} \qquad X \leq Y \text{ iff } \forall \sigma\colon X(\sigma) \leq Y(\sigma)$$

$$C_1\,; \qquad C_2$$

Fix a *potential function* $\pi\colon \mathsf{States} \to \mathbb{R}_{\geq 0}$. Complete lattice $(\mathbb{A}_\pi, \leq)$ of π-runtimes:

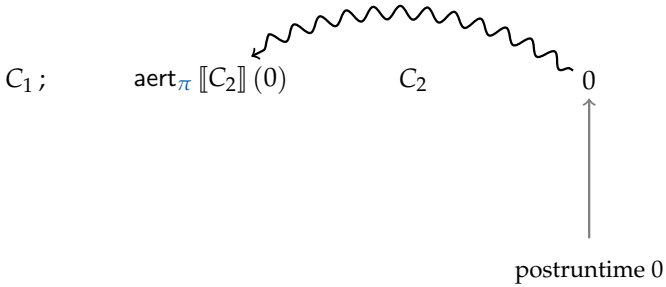
$$\mathbb{A}_\pi \;=\; \{X\colon \mathsf{States} \to \mathbb{R} \cup \{\infty\} \mid \forall \sigma\colon\; -\pi(\sigma) \leq X(\sigma)\} \qquad X \leq Y \text{ iff } \forall \sigma\colon X(\sigma) \leq Y(\sigma)$$

$C_1\,;$ C_2 0

postruntime 0

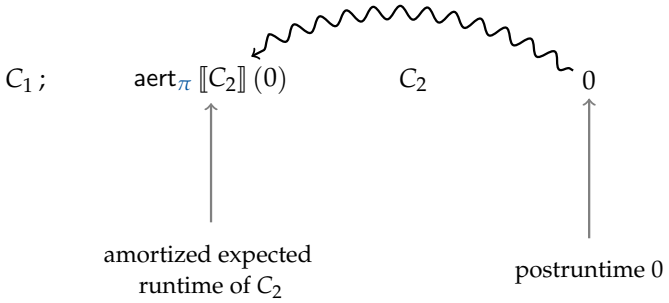
Fix a *potential function* $\pi\colon \mathsf{States} \to \mathbb{R}_{\geq 0}$. Complete lattice $(\mathbb{A}_\pi, \leq)$ of π-runtimes:

$$\mathbb{A}_\pi \;=\; \{X\colon \mathsf{States} \to \mathbb{R} \cup \{\infty\} \mid \forall \sigma\colon\; -\pi(\sigma) \leq X(\sigma)\} \qquad X \leq Y \text{ iff } \forall \sigma\colon X(\sigma) \leq Y(\sigma)$$



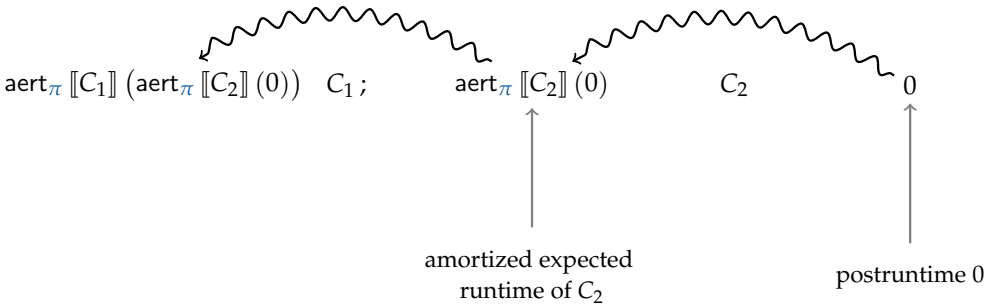
Fix a *potential function* $\pi\colon \mathsf{States} \to \mathbb{R}_{\geq 0}$. Complete lattice $(\mathbb{A}_\pi, \leq)$ of π-runtimes:

$$\mathbb{A}_\pi \;=\; \{X\colon \mathsf{States} \to \mathbb{R} \cup \{\infty\} \mid \forall \sigma\colon\; -\pi(\sigma) \leq X(\sigma)\} \qquad X \leq Y \text{ iff } \forall \sigma\colon X(\sigma) \leq Y(\sigma)$$



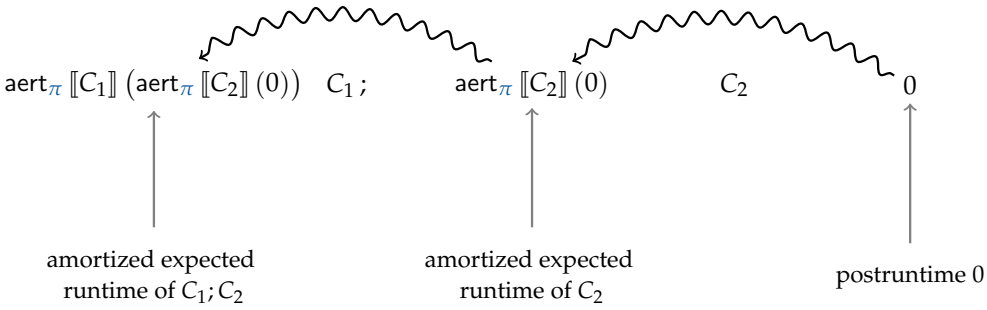
Fix a *potential function* $\pi\colon \mathsf{States} \to \mathbb{R}_{\geq 0}$. Complete lattice $(\mathbb{A}_\pi, \leq)$ of π-runtimes:

$$\mathbb{A}_\pi \;=\; \{X\colon \mathsf{States} \to \mathbb{R} \cup \{\infty\} \mid \forall \sigma\colon\; -\pi(\sigma) \leq X(\sigma)\} \qquad X \leq Y \text{ iff } \forall \sigma\colon X(\sigma) \leq Y(\sigma)$$



Fix a *potential function* $\pi\colon \mathsf{States} \to \mathbb{R}_{\geq 0}$. Complete lattice $(\mathbb{A}_\pi, \leq)$ of π-runtimes:

$$\mathbb{A}_\pi \;=\; \{X\colon \mathsf{States} \to \mathbb{R} \cup \{\infty\} \mid \forall \sigma\colon\; -\pi(\sigma) \leq X(\sigma)\} \qquad X \leq Y \text{ iff } \forall \sigma\colon X(\sigma) \leq Y(\sigma)$$



The aert Calculus

Fix a *potential function* $\pi\colon \mathsf{States} \to \mathbb{R}_{\geq 0}$. Complete lattice $(\mathbb{A}_\pi, \leq)$ of π-runtimes:

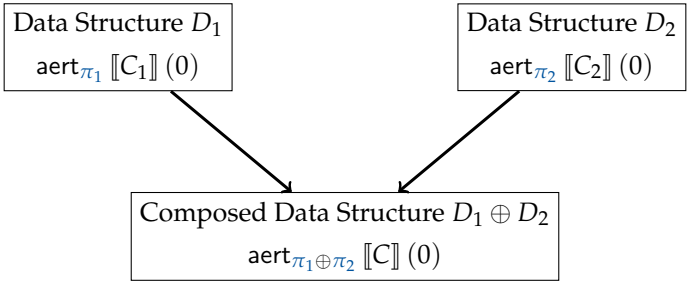
$$\mathbb{A}_\pi \;=\; \{X\colon \mathsf{States} \to \mathbb{R} \cup \{\infty\} \mid \forall \sigma\colon -\pi(\sigma) \leq X(\sigma)\} \qquad X \leq Y \text{ iff } \forall \sigma\colon X(\sigma) \leq Y(\sigma)$$

Data Structure D_1
$\mathsf{aert}_{\pi_1}\,[\![C_1]\!]\,(0)$

Data Structure D_2
$\mathsf{aert}_{\pi_2}\,[\![C_2]\!]\,(0)$

Fix a *potential function* $\pi\colon \mathsf{States} \to \mathbb{R}_{\geq 0}$. Complete lattice $(\mathbb{A}_\pi, \leq)$ of π-runtimes:

$$\mathbb{A}_\pi \;=\; \{X\colon \mathsf{States} \to \mathbb{R} \cup \{\infty\} \mid \forall \sigma\colon -\pi(\sigma) \leq X(\sigma)\} \qquad X \leq Y \text{ iff } \forall \sigma\colon X(\sigma) \leq Y(\sigma)$$



The aert Calculus

Fix a *potential function* π: States $\to \mathbb{R}_{\geq 0}$.

C	$\mathsf{aert}_\pi\ [\![C]\!]\ (X)$
$\texttt{tick}\,(e)$	
$x \coloneqq a$	
$\{\,C_1\,\}\,[\,p\,]\,\{\,C_2\,\}$	
$C_1\,;\,C_2$	
$\texttt{if}\ (\,\varphi\,)\ \{\,C_1\,\}\ \texttt{else}\ \{\,C_2\,\}$	
$\texttt{while}\,(\,\varphi\,)\,\{\,C'\,\}$	

The aert Calculus

Fix a *potential function* π: States $\to \mathbb{R}_{\geq 0}$.

C	$\mathbf{aert}_\pi\ [\![C]\!]\ (X)$
$\texttt{tick}\,(e)$	$e + X$
$x \coloneqq a$	
$\{\, C_1 \,\}\ [\, p \,]\ \{\, C_2 \,\}$	
$C_1\,;\, C_2$	
$\texttt{if}\ (\,\varphi\,)\ \{\, C_1 \,\}\ \texttt{else}\ \{\, C_2 \,\}$	
$\texttt{while}\,(\,\varphi\,)\,\{\, C' \,\}$	

The aert Calculus

Fix a *potential function* π: States $\to \mathbb{R}_{\geq 0}$.

C	$\mathbf{aert}_\pi\ [\![C]\!]\ (X)$
$\texttt{tick}\,(e)$	$e + X$
$x \coloneqq a$	$X\,[x/a] + (\pi\,[x/a] - \pi)$
$\{\,C_1\,\}\,[\,p\,]\,\{\,C_2\,\}$	
$C_1\,;\,C_2$	
$\texttt{if}\ (\,\varphi\,)\ \{\,C_1\,\}\ \texttt{else}\ \{\,C_2\,\}$	
$\texttt{while}\,(\,\varphi\,)\,\{\,C'\,\}$	

The aert Calculus

Fix a *potential function* $\pi\colon \mathsf{States} \to \mathbb{R}_{\geq 0}$.

C	$\mathsf{aert}_\pi\,[\![C]\!]\,(X)$
$\texttt{tick}\,(e)$	$e + X$
$x \coloneqq a$	$X\,[x/a] + (\pi\,[x/a] - \pi)$
$\{\,C_1\,\}\,[\,p\,]\,\{\,C_2\,\}$	$p \cdot \mathsf{aert}_\pi\,[\![C_1]\!]\,(X) + (1-p) \cdot \mathsf{aert}_\pi\,[\![C_2]\!]\,(X)$
$C_1\,;\,C_2$	
$\texttt{if}\,(\,\varphi\,)\,\{\,C_1\,\}\,\texttt{else}\,\{\,C_2\,\}$	
$\texttt{while}\,(\,\varphi\,)\,\{\,C'\,\}$	

The aert Calculus

Fix a *potential function* $\pi\colon \mathsf{States} \to \mathbb{R}_{\geq 0}$.

C	$\mathsf{aert}_\pi\,[\![C]\!]\,(X)$
$\texttt{tick}\,(e)$	$e + X$
$x \coloneqq a$	$X\,[x/a] + (\pi\,[x/a] - \pi)$
$\{\,C_1\,\}\,[\,p\,]\,\{\,C_2\,\}$	$p \cdot \mathsf{aert}_\pi\,[\![C_1]\!]\,(X) + (1-p) \cdot \mathsf{aert}_\pi\,[\![C_2]\!]\,(X)$
$C_1\,;\,C_2$	$\mathsf{aert}_\pi\,[\![C_1]\!]\,\big(\mathsf{aert}_\pi\,[\![C_2]\!]\,(X)\big)$
$\texttt{if}\ (\,\varphi\,)\ \{\,C_1\,\}\ \texttt{else}\ \{\,C_2\,\}$	
$\texttt{while}\,(\,\varphi\,)\,\{\,C'\,\}$	

The aert Calculus

Fix a *potential function* $\pi\colon \mathsf{States} \to \mathbb{R}_{\geq 0}$.

C	$\mathsf{aert}_\pi\,[\![C]\!]\,(X)$
$\texttt{tick}\,(e)$	$e + X$
$x \coloneqq a$	$X\,[x/a] + (\pi\,[x/a] - \pi)$
$\{\,C_1\,\}\,[\,p\,]\,\{\,C_2\,\}$	$p \cdot \mathsf{aert}_\pi\,[\![C_1]\!]\,(X) + (1-p) \cdot \mathsf{aert}_\pi\,[\![C_2]\!]\,(X)$
$C_1\,;\,C_2$	$\mathsf{aert}_\pi\,[\![C_1]\!]\,\big(\mathsf{aert}_\pi\,[\![C_2]\!]\,(X)\big)$
$\texttt{if}\,(\,\varphi\,)\,\{\,C_1\,\}\,\texttt{else}\,\{\,C_2\,\}$	$[\varphi] \cdot \mathsf{aert}_\pi\,[\![C_1]\!]\,(X) + [\neg\varphi] \cdot \mathsf{aert}_\pi\,[\![C_2]\!]\,(X)$
$\texttt{while}\,(\,\varphi\,)\,\{\,C'\,\}$	

The aert Calculus

Fix a *potential function* $\pi\colon \mathsf{States} \to \mathbb{R}_{\geq 0}$.

C	$\mathsf{aert}_\pi\,[\![C]\!]\,(X)$
$\texttt{tick}\,(e)$	$e + X$
$x \coloneqq a$	$X\,[x/a] + (\pi\,[x/a] - \pi)$
$\{\,C_1\,\}\,[\,p\,]\,\{\,C_2\,\}$	$p \cdot \mathsf{aert}_\pi\,[\![C_1]\!]\,(X) + (1-p) \cdot \mathsf{aert}_\pi\,[\![C_2]\!]\,(X)$
$C_1\,;\,C_2$	$\mathsf{aert}_\pi\,[\![C_1]\!]\,\big(\mathsf{aert}_\pi\,[\![C_2]\!]\,(X)\big)$
$\texttt{if}\,(\,\varphi\,)\,\{\,C_1\,\}\,\texttt{else}\,\{\,C_2\,\}$	$[\varphi] \cdot \mathsf{aert}_\pi\,[\![C_1]\!]\,(X) + [\neg\varphi] \cdot \mathsf{aert}_\pi\,[\![C_2]\!]\,(X)$
$\texttt{while}\,(\,\varphi\,)\,\{\,C'\,\}$	$\mathsf{lfp}\;Y.\;[\neg\varphi] \cdot X + [\varphi] \cdot \mathsf{aert}_\pi\,[\![C']\!]\,(Y)$

The lfp is taken w.r.t. $\leq$ on π-runtimes.

The aert Calculus

Fix a *potential function* $\pi\colon \mathsf{States} \to \mathbb{R}_{\geq 0}$.

C	$\mathsf{aert}_\pi\,[\![C]\!]\,(X)$
$\texttt{tick}\,(e)$	$e + X$
$x \coloneqq a$	$X\,[x/a] + (\pi\,[x/a] - \pi)$
$\{\,C_1\,\}\,[\,p\,]\,\{\,C_2\,\}$	$p \cdot \mathsf{aert}_\pi\,[\![C_1]\!]\,(X) + (1-p) \cdot \mathsf{aert}_\pi\,[\![C_2]\!]\,(X)$
$C_1\,;\,C_2$	$\mathsf{aert}_\pi\,[\![C_1]\!]\,\big(\mathsf{aert}_\pi\,[\![C_2]\!]\,(X)\big)$
$\texttt{if}\,(\,\varphi\,)\,\{\,C_1\,\}\,\texttt{else}\,\{\,C_2\,\}$	$[\varphi] \cdot \mathsf{aert}_\pi\,[\![C_1]\!]\,(X) + [\neg\varphi] \cdot \mathsf{aert}_\pi\,[\![C_2]\!]\,(X)$
$\texttt{while}\,(\,\varphi\,)\,\{\,C'\,\}$	$\mathsf{lfp}\;Y.\;[\neg\varphi] \cdot X + [\varphi] \cdot \mathsf{aert}_\pi\,[\![C']\!]\,(Y)$

The lfp is taken w.r.t. $\leq$ on π-runtimes.

Analysis of loops is tackled by **quantitative runtime invariants.**

Let $\pi = x$.

$$\{$$

$$\texttt{tick}\,(x)$$

$$\}\ [1/2]\ \{$$

$$x := 0$$

$$\}$$

$$x := x + 1$$

Let $\pi = x$.

$$\{$$

$$\texttt{tick}\,(x)$$

$$\}\ [1/2]\ \{$$

$$x := 0$$

$$\}$$

$$x := x + 1$$

$$/\!/\!/\ 0$$

0 is the postruntime

Let $\pi = x$.

$\{$

$\quad \texttt{tick}\,(x)$

$\}\ [1/2]\ \{$

$\quad x := 0$

$\}$

$/\!/\!/\ 0\,[x/x+1] + (x+1-x)$

$x := x+1$

$/\!/\!/\ 0$

0 is the postruntime

Let $\pi = x$.

$\{$

$\quad \texttt{tick}\,(x)$

$\}\ [1/2]\ \{$

$\quad x := 0$

$\}$

$/\!/\!/\ 1$

$x := x + 1$

$/\!/\!/\ 0$

0 is the postruntime

Let $\pi = x$.

{

`tick` (x)

/// 1

} $[1/2]$ {

$x := 0$

/// 1

}

/// 1

$x := x + 1$

/// 0

0 is the postruntime

Let $\pi = x$.

$$
\begin{array}{l}
\{ \\
\quad \texttt{tick}\,(x) \\
\quad /\!/\!/\ 1 \\
\}\ [1/2]\ \{ \\
\quad /\!/\!/\ 1\,[x/0] + (0 - x) \\
\quad x := 0 \\
\quad /\!/\!/\ 1 \\
\} \\
/\!/\!/\ 1 \\
x := x + 1 \\
/\!/\!/\ 0
\end{array}
$$

0 is the postruntime

Let $\pi = x$.

$$
\begin{array}{l}
\{ \\
\quad \texttt{tick}\,(x) \\
\quad /\!/\!/\ 1 \\
\} \ [1/2] \ \{ \\
\quad /\!/\!/\ 1 - x \qquad \text{possibly negative} \\
\quad x := 0 \\
\quad /\!/\!/\ 1 \\
\} \\
/\!/\!/\ 1 \\
x := x + 1 \\
/\!/\!/\ 0 \qquad 0 \text{ is the postruntime}
\end{array}
$$

Let $\pi = x$.

$$
\begin{array}{l}
\{ \\
\quad /\!/\!/\ x+1 \\
\quad \texttt{tick}\,(x) \\
\quad /\!/\!/\ 1 \\
\}\ [1/2]\ \{ \\
\quad /\!/\!/\ 1-x \qquad \text{possibly negative} \\
\quad x := 0 \\
\quad /\!/\!/\ 1 \\
\} \\
/\!/\!/\ 1 \\
x := x+1 \\
/\!/\!/\ 0 \qquad 0 \text{ is the postruntime}
\end{array}
$$

The aert Calculus

Let $\pi = x$.

$/\!/\!/\ 1/2 \cdot (x+1) + 1/2 \cdot (1-x)$ this is $\mathsf{aert}_\pi [\![C]\!] (0)$

{

$/\!/\!/\ x+1$

$\texttt{tick}\,(x)$

$/\!/\!/\ 1$

} [1/2] {

$/\!/\!/\ 1-x$ possibly negative

$x := 0$

$/\!/\!/\ 1$

}

$/\!/\!/\ 1$

$x := x+1$

$/\!/\!/\ 0$ 0 is the postruntime

The aert Calculus

Let $\pi = x$.

$/\!/\!/\ 1$ — this is $\mathsf{aert}_{\pi}\,[\![C]\!]\,(0)$

{

$/\!/\!/\ x + 1$

$\texttt{tick}\,(x)$

$/\!/\!/\ 1$

} $[1/2]$ {

$/\!/\!/\ 1 - x$ — possibly negative

$x := 0$

$/\!/\!/\ 1$

}

$/\!/\!/\ 1$

$x := x + 1$

$/\!/\!/\ 0$ — 0 is the postruntime

Incorporating Dynamic Memory

by utilizing **Runtime Separation Logic** [Matheja 2020, Haslbeck 2021]

Incorporating Dynamic Memory

$$
\begin{array}{lcll}
C & \longrightarrow & \texttt{tick}\,(e) & \text{time consumption} \\
& \mid & x \coloneqq a & \text{assignment} \\
& \mid & x \coloneqq \texttt{alloc}\,(e) & \text{memory allocation} \\
& \mid & x \coloneqq \langle e \rangle & \text{heap lookup} \\
& \mid & \langle e \rangle \coloneqq e' & \text{heap mutation} \\
& \mid & \texttt{free}(e) & \text{memory deallocation} \\
& \mid & C\,;C \mid \{\,C\,\}\,[\,p\,]\,\{\,C\,\} \mid & \\
& \mid & \texttt{if}\,(\,\varphi\,)\,\{\,C\,\}\,\texttt{else}\,\{\,C\,\} \mid \texttt{while}\,(\,\varphi\,)\,\{\,C'\,\} &
\end{array}
$$

Incorporating Dynamic Memory

$$
\begin{array}{rcll}
C & \longrightarrow & \texttt{tick}\,(e) & \text{time consumption} \\
& \mid & x \coloneqq a & \text{assignment} \\
& \mid & x \coloneqq \texttt{alloc}\,(e) & \text{memory allocation} \\
& \mid & x \coloneqq \langle e \rangle & \text{heap lookup} \\
& \mid & \langle e \rangle \coloneqq e' & \text{heap mutation} \\
& \mid & \texttt{free}(e) & \text{memory deallocation} \\
& \mid & C\,;C \mid \{\,C\,\}\,[\,p\,]\,\{\,C\,\} \mid & \\
& \mid & \texttt{if}\,(\,\varphi\,)\,\{\,C\,\}\,\texttt{else}\,\{\,C\,\} \mid \texttt{while}\,(\,\varphi\,)\,\{\,C'\,\} &
\end{array}
$$

$$
\mathsf{States} \;=\; \{(\mathfrak{s},\mathfrak{h}) \mid \mathfrak{s}\colon \mathsf{Vars} \to \mathbb{N},\ \mathfrak{h}\colon \mathsf{Loc} \to_{\text{fin}} \mathbb{N}\}
$$

Incorporating Dynamic Memory

$$
\begin{array}{rcll}
C & \longrightarrow & \texttt{tick}\,(e) & \text{time consumption} \\
& \mid & x \coloneqq a & \text{assignment} \\
& \mid & x \coloneqq \texttt{alloc}\,(e) & \text{memory allocation} \\
& \mid & x \coloneqq \langle e \rangle & \text{heap lookup} \\
& \mid & \langle e \rangle \coloneqq e' & \text{heap mutation} \\
& \mid & \texttt{free}(e) & \text{memory deallocation} \\
& \mid & C\,;C \mid \{\,C\,\}\,[\,p\,]\,\{\,C\,\} \mid & \\
& \mid & \texttt{if}\,(\,\varphi\,)\,\{\,C\,\}\,\texttt{else}\,\{\,C\,\} \mid \texttt{while}\,(\,\varphi\,)\,\{\,C'\,\} &
\end{array}
$$

$$
\mathsf{States} \;=\; \{(\mathfrak{s},\mathfrak{h}) \mid \mathfrak{s}\colon \mathsf{Vars} \to \mathbb{N},\ \mathfrak{h}\colon \mathsf{Loc} \to_{\text{fin}} \mathbb{N}\}
$$

Memory faults cause an *infinite* runtime

Incorporating Dynamic Memory

Define **separating connectives** on runtimes $f, g\colon \text{States} \to \mathbb{R}^{\infty}_{\geq 0}$:

$$f \oplus g \;=\; \lambda(\mathfrak{s},\mathfrak{h}).\; \min\left\{ f(\mathfrak{s},\mathfrak{h}_1) + g(\mathfrak{s},\mathfrak{h}_2) \;\middle|\; \mathfrak{h} = \mathfrak{h}_1 \star \mathfrak{h}_2 \right\}$$

Incorporating Dynamic Memory

Define **separating connectives** on runtimes $f, g\colon \mathsf{States} \to \mathbb{R}^{\infty}_{\geq 0}$:

$$f \oplus g \;=\; \lambda(\mathfrak{s},\mathfrak{h}).\ \min\{f(\mathfrak{s},\mathfrak{h}_1) + g(\mathfrak{s},\mathfrak{h}_2) \mid \mathfrak{h} = \mathfrak{h}_1 \star \mathfrak{h}_2\}$$

$$f \mathrel{-\!\!\!-\!\!\ominus} g \;=\; \text{(adjoint: monus, omitted)}$$

Incorporating Dynamic Memory

Define **separating connectives** on runtimes f, g: States $\to \mathbb{R}^{\infty}_{\geq 0}$:

$$f \oplus g \;=\; \lambda(\mathfrak{s}, \mathfrak{h}).\ \min\{f(\mathfrak{s}, \mathfrak{h}_1) + g(\mathfrak{s}, \mathfrak{h}_2) \mid \mathfrak{h} = \mathfrak{h}_1 \star \mathfrak{h}_2\}$$

$$f \mathrel{-\!\!\!-\!\!\ominus} g \;=\; \text{(adjoint: monus, omitted)}$$

Typical **runtime specification**:

$$\mathsf{aert}_{\pi}\,[\![C]\!]\,(0) \;\leq\; \wr x \mapsto 1\wr \oplus \wr y \mapsto 2\wr \oplus \underline{3}$$

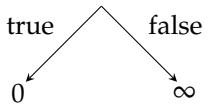
Define **separating connectives** on runtimes $f, g\colon \text{States} \to \mathbb{R}^{\infty}_{\geq 0}$:

$$f \oplus g \;=\; \lambda(\mathfrak{s},\mathfrak{h}).\ \min\{f(\mathfrak{s},\mathfrak{h}_1) + g(\mathfrak{s},\mathfrak{h}_2) \mid \mathfrak{h} = \mathfrak{h}_1 \star \mathfrak{h}_2\}$$

$$f \mathrel{-\!\!\!-\!\!\ominus} g \;=\; \text{(adjoint: monus, omitted)}$$

Typical **runtime specification**:

$$\mathsf{aert}_{\pi}[\![C]\!](0) \;\leq\; [\![x \mapsto 1]\!] \oplus [\![y \mapsto 2]\!] \oplus \underline{3} \;=\; \lambda(\mathfrak{s},\mathfrak{h}).\ \begin{cases} 3, & \text{if } (\mathfrak{s},\mathfrak{h}) \models x \mapsto 1 \star y \mapsto 2 \\ \infty, & \text{otherwise.} \end{cases}$$



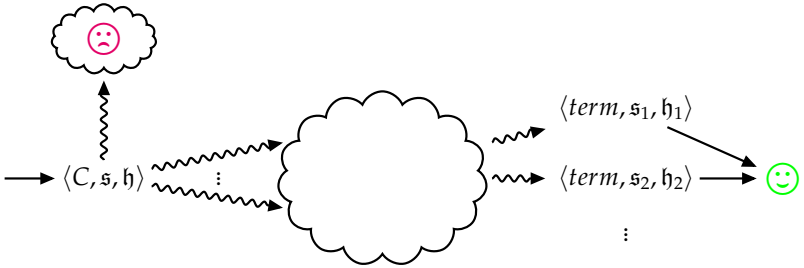
Incorporating Dynamic Memory

C	$\mathbf{aert}_\pi\ [\![C]\!]\ (X)$
$x \coloneqq \texttt{alloc}\,(e)$	$\sup v\colon \left(\bigoplus_{i=1}^{e} \wr v+i-1 \mapsto 0 \wr\right) \multimap (X\,[x/v] + \pi\,[x/v]) \quad - \quad \pi$
$x \coloneqq \langle e \rangle$	$\inf v\colon \wr e \mapsto v \wr \oplus \left(\wr e \mapsto v \wr \multimap (X\,[x/v] + \pi\,[x/v])\right) \quad - \quad \pi$
$\langle e \rangle \coloneqq e'$	$\wr e \mapsto\ - \wr \oplus \left(\wr e \mapsto e' \wr \multimap (X + \pi)\right) \quad - \quad \pi$
$\texttt{free}(e)$	$\wr e \mapsto\ - \wr \oplus (X + \pi) \quad - \quad \pi$

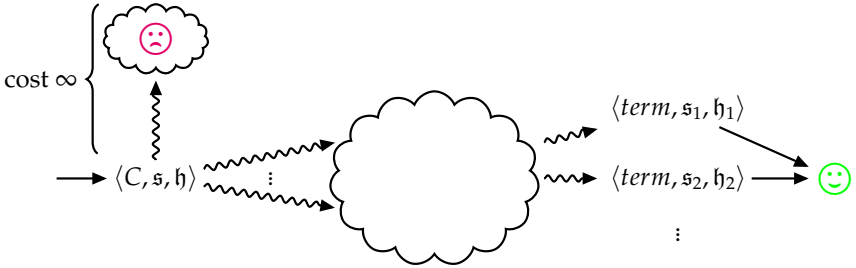
C	$\mathsf{aert}_\pi\,[\![C]\!]\,(X)$
$x \coloneqq \mathtt{alloc}\,(e)$	$\sup v\colon \left(\bigoplus_{i=1}^{e} \wr v+i-1 \mapsto 0 \int\right) \multimap (X\,[x/v] + \pi\,[x/v]) \;-\; \pi$
$x \coloneqq \langle e \rangle$	$\inf v\colon \wr e \mapsto v \int \oplus \left(\wr e \mapsto v \int \multimap (X\,[x/v] + \pi\,[x/v])\right) \;-\; \pi$
$\langle e \rangle \coloneqq e'$	$\wr e \mapsto\; - \int \oplus \left(\wr e \mapsto e' \int \multimap (X + \pi)\right) \;-\; \pi$
$\mathtt{free}(e)$	$\wr e \mapsto\; - \int \oplus (X + \pi) \;-\; \pi$

$$\mathsf{aert}_\pi\,[\![C]\!]\,(0)\,(\mathfrak{s},\mathfrak{h}) \;=\; \begin{cases} \text{amort. exp. runtime of } C \text{ on } (\mathfrak{s},\mathfrak{h}), & \text{if } C \text{ memory-safe on } (\mathfrak{s},\mathfrak{h}) \\ \infty, & \text{otherwise} \end{cases}$$

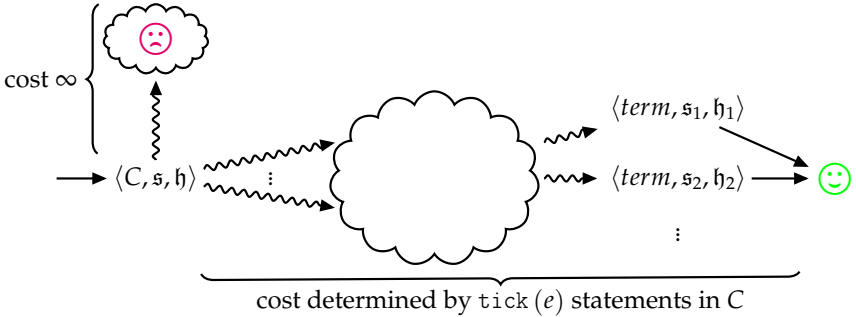
Incorporating Dynamic Memory



Incorporating Dynamic Memory



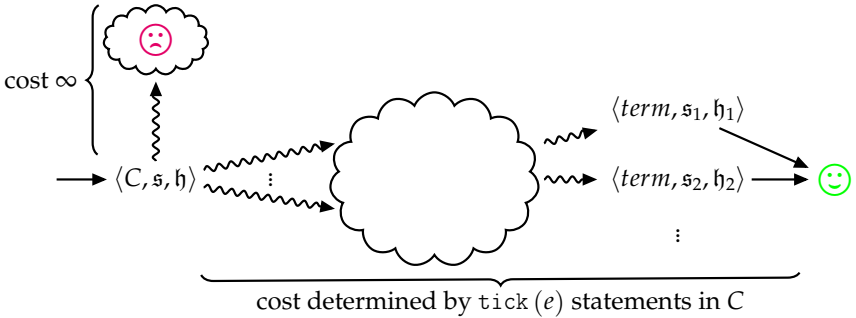
Incorporating Dynamic Memory



Theorem (Soundness of aert)

For every C and initial state $(\mathfrak{s}, \mathfrak{h})$,

$$\mathsf{ExpectedCost}\left(\mathsf{MDP}[\![C, \mathfrak{s}, \mathfrak{h}]\!]\right) \;\leq\; \mathsf{aert}_{\pi}\,[\![C]\!]\,(0)\,(\mathfrak{s}, \mathfrak{h}) + \pi(\mathfrak{s}, \mathfrak{h})\,.$$



Case Study

The Insert-Delete-FindAny Problem [Brodal et al. '96]

Case Study: The Insert-Delete-FindAny Problem [Brodal et al. '96]

$\underline{insert(y):}$

$add\,(y)\,;$

$\{$

$\quad any \coloneqq H; Rank$

$\}\ [1/_{len+1}]\ \{$

$\quad \texttt{if}\ (\,len \geq 2\,)\ \{$

$\quad\quad v_{any} \coloneqq \langle any + 2\rangle;$

$\quad\quad \texttt{tick}\,(1)\,;$

$\quad\quad \texttt{if}\ \left(\,y < v_{any}\,\right)\ \{rank \coloneqq rank + 1\,\}$

$\quad\}\ \texttt{else}\ \{$

$\quad\quad any \coloneqq H\,;\, rank \coloneqq 1$

$\quad\}$

$\}$

$\underline{delete(y):}$

$remove\,(x)\,;$

$\texttt{if}\ (\,len = 0\,)\ \{$

$\quad any \coloneqq 0\,;\, rank \coloneqq 0$

$\}\ \texttt{else}\ \{$

$\quad \texttt{if}\ (\,x = any\,)\ \{$

$\quad\quad Sample\,;\, Rank$

$\quad\}\ \texttt{else}\ \{$

$\quad\quad v_x \coloneqq \langle x + 2\rangle\,;\, v_{any} \coloneqq \langle any + 2\rangle;$

$\quad\quad \texttt{tick}\,(1)\,;$

$\quad\quad \texttt{if}\ \left(\,v_x < v_{any}\,\right)\ \{rank \coloneqq rank \dot{-} 1\,\}$

$\quad\}$

$\};\texttt{free}(x, x+1, x+2)$

Case Study: The Insert-Delete-FindAny Problem [Brodal et al. '96]

Theorem

insert (y) *and delete* (x) *are* ***memory-safe*** *and run in* ***constant amortized expected time****.*

Case Study: The Insert-Delete-FindAny Problem [Brodal et al. '96]

Theorem

insert (y) *and delete* (x) *are* ***memory-safe*** *and run in* ***constant amortized expected time****.*

Ingredients:

- **Inductive runtime invariants** for loops:

$$\zeta v, end, len_1, len_2 : \mathsf{dll}\,(H,\, v,\, 0,\, len_1,\, c) \oplus \mathsf{dll}\,(c,\, end,\, v,\, len_2,\, 0) \oplus \underline{len_2}$$

Case Study: The Insert-Delete-FindAny Problem [Brodal et al. '96]

Theorem

insert (y) *and delete* (x) *are* ***memory-safe*** *and run in* ***constant amortized expected time****.*

Ingredients:

- **Inductive runtime invariants** for loops:

$$\wr v, end, len_1, len_2 \colon \mathsf{dll}\,(H,\, v,\, 0,\, len_1,\, c) \oplus \mathsf{dll}\,(c,\, end,\, v,\, len_2,\, 0) \oplus \underline{len_2}$$

- The aert **frame rule**:

$$\mathsf{aert}_\pi\,[\![C]\!]\,(f \oplus g) \overset{\text{morally}}{\leq} \mathsf{aert}_\pi\,[\![C]\!]\,(f) \oplus g$$

Case Study: The Insert-Delete-FindAny Problem [Brodal et al. '96]

Theorem

insert (y) *and delete* (x) *are* ***memory-safe*** *and run in* ***constant amortized expected time****.*

Ingredients:

- **Inductive runtime invariants** for loops:

$$\wr v, end, len_1, len_2 \colon \mathsf{dll}\,(H,\, v,\, 0,\, len_1,\, c) \oplus \mathsf{dll}\,(c,\, end,\, v,\, len_2,\, 0) \oplus \underline{len_2}$$

- The aert **frame rule**:

$$\mathsf{aert}_\pi\,[\![C]\!]\,(f \oplus g) \quad \overset{\text{morally}}{\leq} \quad \mathsf{aert}_\pi\,[\![C]\!]\,(f) \oplus g$$

- Rules for local reasoning (avoiding $\multimap\!\!\ominus$)

Theorem

insert (y) *and delete* (x) *are* ***memory-safe*** *and run in* ***constant amortized expected time***.

Ingredients:

- **Inductive runtime invariants** for loops:

$$\wr\, v, end, len_1, len_2 \colon \mathsf{dll}\,(H,\, v,\, 0,\, len_1,\, c) \oplus \mathsf{dll}\,(c,\, end,\, v,\, len_2,\, 0) \oplus \underline{len_2}$$

- The aert **frame rule**:

$$\mathsf{aert}_\pi\,[\![C]\!]\,(f \oplus g) \overset{\text{morally}}{\leq} \mathsf{aert}_\pi\,[\![C]\!]\,(f) \oplus g$$

- Rules for local reasoning (avoiding $\multimap$)

Thank you!

Theorem (aert Frame Rule)

$$\mathsf{Mod}\,(C) \cap \mathsf{Vars}(g) = \emptyset \qquad \textit{implies}$$
$$\mathsf{aert}_{\pi}\,[\![C]\!]\,((f \oplus g) - \pi) \;\leq\; \big((\mathsf{aert}_{\pi}\,[\![C]\!]\,(f - \pi) + \pi) \oplus g\big) - \pi\,.$$

Theorem (Compositionality w.r.t. π)

Let π_1, π_2 be potentials with $\pi_1 \preceq \pi_1 \oplus \pi_2$.

$$\mathsf{Mod}\,(C) \cap \mathsf{Vars}(\pi_2) = \emptyset \quad \textit{and} \quad X + (\pi_1 \oplus \pi_2) \leq (X + \pi_1) \oplus \pi_2$$
$$\textit{implies} \quad \mathsf{aert}_{\pi_1 \oplus \pi_2}\,[\![C]\!]\,(X) \leq (\mathsf{aert}_{\pi_1}\,[\![C]\!]\,(X) + \pi_1) \oplus \pi_2 \quad - \quad \pi_1 \oplus \pi_2\,.$$