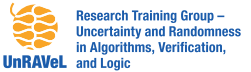
Quantitative Separation Logic

A Logic for Reasoning about Probabilistic Pointer Programs

Kevin Batz **Benjamin Lucien Kaminski**

Joost-Pieter Katoen Christoph Matheja Thomas Noll



7th Conference on **Highlights of Logic, Games and Automata (Highlights 2019)**

September 18, 2019, Warsaw, Poland

Garbage Collection

```
delete (x):
    if ( x ≠ 0 ) {
                                  l := [x] ; r := [x + 1] ;
                                  delete (l) ; delete (r) ;
                                  free(x) ; free(x + 1)
    }
```

Probabilistic Garbage Collection

$$
\begin{array}{l}
\texttt{delete}\ (x)\texttt{:} \\
\quad \texttt{if}\ (\, x \neq 0\,)\ \{ \\
\qquad \{\texttt{skip}\}\,[p]\ \left\{ \begin{array}{l} l := \overline{x}\ ;\ r := \overline{x+1}\ ; \\ \texttt{delete}\,(l)\ ;\ \texttt{delete}\,(r)\ ; \\ \texttt{free}(x)\ ;\ \texttt{free}(x+1) \end{array} \right\} \\
\quad \}
\end{array}
$$

Probabilistic Garbage Collection

$$
\begin{array}{l}
\texttt{delete}\ (x)\texttt{:} \\
\quad \texttt{if}\ (\, x \neq 0\,)\ \{ \\
\qquad \{\texttt{skip}\}\,[p] \left\{ \begin{array}{l} l := \overline{x}\ ;\ r := \overline{x+1}\ ; \\ \texttt{delete}\,(l)\ ;\ \texttt{delete}\,(r)\ ; \\ \texttt{free}(x)\ ;\ \texttt{free}(x+1) \end{array} \right\} \\
\quad \}
\end{array}
$$

What is the probability that the heap is empty?

Probabilistic Garbage Collection

$$
\begin{array}{l}
\texttt{delete}\ (x)\texttt{:} \\
\quad \texttt{if}\ (\, x \neq 0\,)\ \{ \\
\qquad \{\texttt{skip}\}\,[p]\ \left\{ \begin{array}{l} l := \overline{x}\ ;\ r := \overline{x+1}\ ; \\ \texttt{delete}\,(l)\ ;\ \texttt{delete}\,(r)\ ; \\ \texttt{free}(x)\ ;\ \texttt{free}(x+1) \end{array} \right\} \\
\quad \}
\end{array}
$$

What is the probability that the heap is empty?

What is the expected amount of garbage?

Quantitative Separation Logic
as an assertion language

Instead of truth, we measure a quantity!

Our Quantities

Our Quantities

- Associate quantities to program states

Our Quantities

- Associate quantities to program states

$$\mathbb{E} \;=\; \left\{ f \;\middle|\; f \colon \text{Stacks} \times \text{Heaps} \to \mathbb{R}^{\infty}_{\geq 0} \right\}$$

Our Quantities

- Associate quantities to program states

$$\mathbb{E} \;=\; \left\{ f \;\middle|\; f \colon \text{Stacks} \times \text{Heaps} \to \mathbb{R}^{\infty}_{\geq 0} \right\}$$

- Examples:

Our Quantities

- Associate quantities to program states

$$\mathbb{E} \;=\; \left\{ f \;\middle|\; f \colon \text{Stacks} \times \text{Heaps} \to \mathbb{R}^{\infty}_{\geq 0} \right\}$$

- Examples:
  - $x \;=\; \lambda(s, h).\;\; s(x)$

Our Quantities

- Associate quantities to program states

$$\mathbb{E} \;=\; \left\{ f \;\middle|\; f \colon \mathsf{Stacks} \times \mathsf{Heaps} \to \mathbb{R}^{\infty}_{\geq 0} \right\}$$

- Examples:
  - $x \;=\; \lambda(s,\, h).\;\; s(x)$
  - $\mathsf{heapSize} \;=\; \lambda(s,\, h).\;\; \left|\mathsf{dom}\,(h)\right|$

Quantitative Separating Conjunction

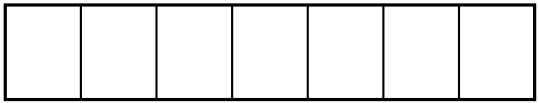
Classical separating conjunction:

$$(s,h) \models F \star G \qquad \text{iff}$$

$$\exists\, h_1, h_2 \colon \quad h = h_1 \uplus h_2 \quad \text{and} \quad (s,h_1) \models F \quad \text{and} \quad (s,h_2) \models G$$

Quantitative Separating Conjunction

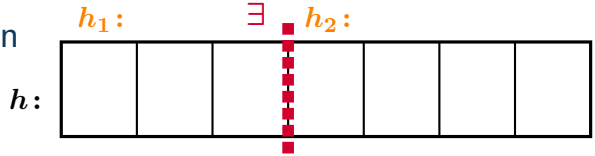
Classical separating conjunction:



$(s, h) \models F \star G$ iff

$\exists\, h_1, h_2 \colon \quad h = h_1 \uplus h_2$ and $(s, h_1) \models F$ and $(s, h_2) \models G$

Quantitative Separating Conjunction



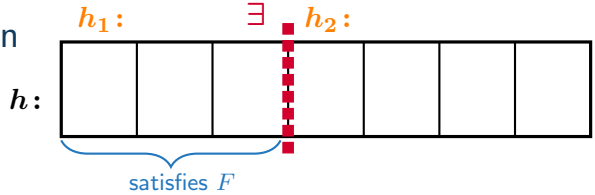
Classical separating conjunction:

$(s, h) \models F \star G$ iff

$\exists\, h_1, h_2 \colon \quad h = h_1 \uplus h_2$ and $(s, h_1) \models F$ and $(s, h_2) \models G$

Quantitative Separating Conjunction

Classical separating conjunction:



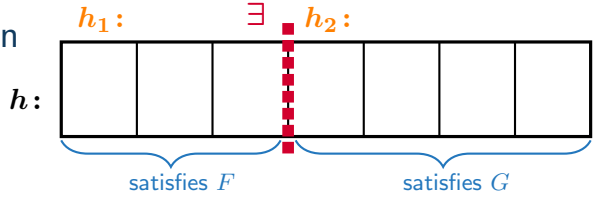
$$(s,h) \models F \star G \qquad \text{iff}$$

$$\exists\, h_1, h_2 \colon \quad h = h_1 \uplus h_2 \quad \text{and} \quad (s,h_1) \models F \quad \text{and} \quad (s,h_2) \models G$$

Quantitative Separating Conjunction

Classical separating conjunction:

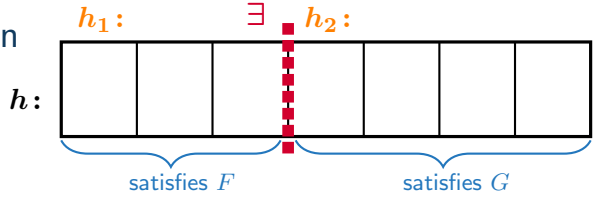
$(s, h) \models F \star G$ iff



$\exists\, h_1, h_2 \colon \quad h = h_1 \uplus h_2$ and $(s, h_1) \models F$ and $(s, h_2) \models G$

Quantitative Separating Conjunction

Classical separating conjunction:



$$(s,h) \models F \star G \qquad \text{iff}$$

$$\exists\, h_1, h_2 \colon \quad h = h_1 \uplus h_2 \quad \text{and} \quad (s,h_1) \models F \quad \text{and} \quad (s,h_2) \models G$$

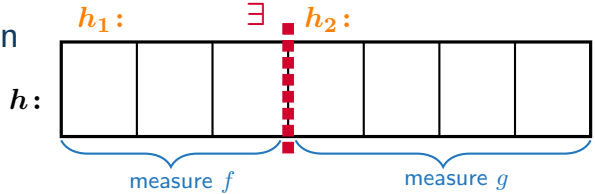
Naive quantitative separating conjunction:

$$f \star g \quad = \quad \lambda(s,h).\ \exists\, h_1, h_2 \colon\ [h = h_1 \uplus h_2] \cdot f(s,h_1) \cdot g(s,h_2)$$

Quantitative Separating Conjunction

Classical separating conjunction:

$(s, h) \models F \star G$ iff



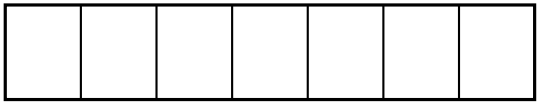
$\exists\, h_1, h_2\colon \quad h = h_1 \uplus h_2$ and $(s, h_1) \models F$ and $(s, h_2) \models G$

Naive quantitative separating conjunction:

$$f \star g \quad = \quad \lambda(s,h).\ \exists\, h_1, h_2\colon\ [h = h_1 \uplus h_2] \cdot f(s, h_1) \cdot g(s, h_2)$$

Quantitative Separating Conjunction

Classical separating conjunction:



$$(s,h) \models F \star G \qquad \text{iff}$$

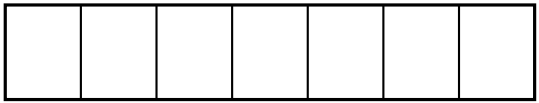
$$\exists\, h_1, h_2 \colon \quad h = h_1 \uplus h_2 \quad \text{and} \quad (s,h_1) \models F \quad \text{and} \quad (s,h_2) \models G$$

Naive quantitative separating conjunction:

$$f \star g \quad = \quad \lambda(s,h).\ \exists\, h_1, h_2 \colon\ [h = h_1 \uplus h_2] \cdot f(s,h_1) \cdot g(s,h_2)$$

Quantitative Separating Conjunction

Classical separating conjunction:



$$(s,h) \models F \star G \qquad \text{iff}$$

$$\exists\, h_1, h_2 \colon \quad h = h_1 \uplus h_2 \quad \text{and} \quad (s,h_1) \models F \quad \text{and} \quad (s,h_2) \models G$$

Naive quantitative separating conjunction:

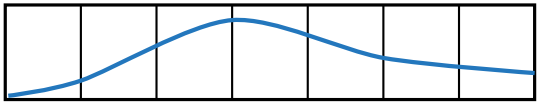
$$f \star g \quad = \quad \lambda(s,h).\ \exists\, h_1, h_2 \colon\ [h = h_1 \uplus h_2] \cdot f(s,h_1) \cdot g(s,h_2)$$

Meaningful quantitative separating conjunction [POPL'19]**:**

$$f \star g \ = \ \lambda(s,h).\ \max_{h_1,h_2} \left\{\, f(s,h_1) \cdot g(s,h_2) \;\middle|\; h = h_1 \uplus h_2 \,\right\}$$

Quantitative Separating Conjunction

Classical separating conjunction:



$$(s,h) \models F \star G \qquad \text{iff}$$

$$\exists\, h_1, h_2 \colon \quad h = h_1 \uplus h_2 \quad \text{and} \quad (s,h_1) \models F \quad \text{and} \quad (s,h_2) \models G$$

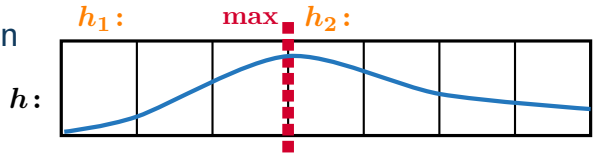
Naive quantitative separating conjunction:

$$f \star g \quad = \quad \lambda(s,h).\ \exists\, h_1, h_2 \colon\ [h = h_1 \uplus h_2] \cdot f(s,h_1) \cdot g(s,h_2)$$

Meaningful quantitative separating conjunction [POPL'19]**:**

$$f \star g \ = \ \lambda(s,h).\ \max_{h_1,h_2} \left\{\, f(s,h_1) \cdot g(s,h_2) \;\middle|\; h = h_1 \uplus h_2 \,\right\}$$

Quantitative Separating Conjunction



Classical separating conjunction:

$(s,h) \models F \star G$ iff

$\exists\, h_1, h_2\colon \quad h = h_1 \uplus h_2$ and $(s,h_1) \models F$ and $(s,h_2) \models G$

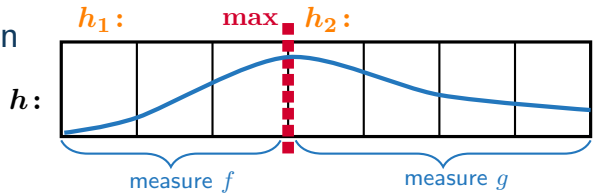
Naive quantitative separating conjunction:

$$f \star g \quad = \quad \lambda(s,h).\ \exists\, h_1, h_2\colon\ [h = h_1 \uplus h_2] \cdot f(s,h_1) \cdot g(s,h_2)$$

Meaningful quantitative separating conjunction [POPL'19]**:**

$$f \star g \ = \ \lambda(s,h).\ \max_{h_1,h_2} \left\{\, f(s,h_1) \cdot g(s,h_2) \;\middle|\; h = h_1 \uplus h_2 \,\right\}$$

Quantitative Separating Conjunction



Classical separating conjunction:

$$(s,h) \models F \star G \qquad \text{iff}$$

$$\exists\, h_1, h_2\colon \quad h = h_1 \uplus h_2 \quad \text{and} \quad (s,h_1) \models F \quad \text{and} \quad (s,h_2) \models G$$

Naive quantitative separating conjunction:

$$f \star g \quad = \quad \lambda(s,h).\ \exists\, h_1, h_2\colon\ [h = h_1 \uplus h_2] \cdot f(s,h_1) \cdot g(s,h_2)$$

Meaningful quantitative separating conjunction [POPL'19]**:**

$$f \star g \ = \ \lambda(s,h).\ \max_{h_1,h_2} \left\{\, f(s,h_1) \cdot g(s,h_2) \;\middle|\; h = h_1 \uplus h_2 \,\right\}$$

Quantitative Separating Implication [POPL'19]

$$f \mathrel{-\!\!*} g$$

Quantitative Separating Implication [POPL'19]

Classical separating implication:

$$(s,h) \models F \mathrel{-\!\!\star} G \quad \text{iff} \quad \forall\, h' \text{ with } h' \perp h \wedge (s,h') \models F\colon \quad (s, h \star h') \models G$$

Quantitative separating implication:

$$f \mathrel{-\!\!\star} g \quad = \quad \lambda(s,h).\ \inf_{h'} \left\{ \frac{g(s,h\star h')}{f(s,h')} \;\middle|\; \begin{array}{l} h' \perp h \text{ and } f(s,h') > 0 \text{ and} \\ \underline{\text{not}}\ \ f(s,h') = \infty = g(s,h\star h') \end{array} \right\}$$

Notice:

$$[F] \mathrel{-\!\!\star} g \quad = \quad \lambda(s,h).\ \inf_{h'} \left\{ g(s,h\star h') \;\middle|\; h' \perp h \text{ and } (s,h') \models F \right\}$$

Quantitative Separating Implication [POPL'19]

Classical separating implication:

$$(s,h) \models F \mathrel{-\!\!*} G \quad \text{iff} \quad \forall h' \text{ with } h' \perp h \wedge (s,h') \models F\colon \quad (s, h \star h') \models G$$

Quantitative separating implication:

$$f \mathrel{-\!\!*} g \quad = \quad \lambda(s,h).\ \inf_{h'} \left\{ \frac{g(s, h \star h')}{f(s,h')} \;\middle|\; \begin{array}{l} h' \perp h \text{ and } f(s,h') > 0 \text{ and} \\ \underline{\text{not}}\ \ f(s,h') = \infty = g(s, h \star h') \end{array} \right\}$$

Notice:

$$[F] \mathrel{-\!\!*} g \quad = \quad \lambda(s,h).\ \inf_{h'} \left\{ g(s, h \star h') \;\middle|\; h' \perp h \text{ and } (s,h') \models F \right\}$$

Adjointness of $\longrightarrow\!\!\!\!\star$ and $\star$

Quantitative Adjointness:

$$f \star g \preceq j \quad \text{iff} \quad f \preceq g \longrightarrow\!\!\!\!\star\, j$$

Adjointness of $\longrightarrow\!\!\!\!\star$ and $\star$

Quantitative Adjointness:

$$f \star g \;\preceq\; j \qquad \text{iff} \qquad f \;\preceq\; g \longrightarrow\!\!\!\!\star\, j$$

My personal Aha Erlebnis:

$$a \cdot b \;\leq\; c \qquad \text{iff}^* \qquad a \;\leq\; c \,/\, b$$

Modus Ponens

$$f \star (f \mathrel{-\!\!*} g) \quad \preceq \quad g$$

Quantitative Separation Logic
as a Program Logic

A teaser.

```
delete (x):
    if (x ≠ 0) {
        {skip} [p] {
            l := ⌜x⌝ ; r := ⌜x + 1⌝ ;

            delete (l) ;

            delete (r) ;

            free(x) ; free(x + 1)
    }}
```

```
delete (x):
    if (x ≠ 0) {
        {skip} [p] {
            l := ⌜x⌝ ; r := ⌜x + 1⌝ ;

            delete (l) ;

            delete (r) ;

            free(x) ; free(x + 1)
    }}
```

Probability of empty heap after executing `delete`?

```
delete (x):
    if (x ≠ 0) {
        {skip} [p] {
            l := ⌈x⌉ ; r := ⌈x + 1⌉ ;

            delete (l) ;

            delete (r) ;

            free(x) ; free(x + 1)
    }}
/// [emp]
```

$\texttt{delete}\ (x)\colon$

$\quad \texttt{if}\ (x \neq 0)\ \{$

$\quad\quad \{\texttt{skip}\}\ [p]\ \{$

$\quad\quad\quad l := \overline{x\,|}\ ;\ r := \overline{x+1\,|}\ ;$

$\quad\quad\quad \mathsf{delete}\,(l)\ ;$

$\quad\quad\quad \mathsf{delete}\,(r)\ ;$

$\quad\quad\quad /\!/\!/\ \ [x \mapsto -, -] \star [\mathbf{emp}] \star [\mathbf{emp}]$

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

$\quad \}\,\}$

$/\!/\!/\ \ [\mathbf{emp}]$

```
delete (x):
    if (x ≠ 0) {
        {skip} [p] {
            l := ⌜x⌝ ; r := ⌜x + 1⌝ ;

            delete (l) ;
            /// [x ↦ −, −] ⋆ [emp] ⋆ (1 − p)^spt(r)        (invariant & frame rule)
            delete (r) ;
            /// [x ↦ −, −] ⋆ [emp] ⋆ [emp]
            free(x) ; free(x + 1)
    } }
/// [emp]
```

$$
\begin{array}{l}
\texttt{delete}\ (x)\colon \\
\quad \texttt{if}\ (x \neq 0)\ \{ \\
\qquad \{\texttt{skip}\}\ [p]\ \{ \\
\qquad\quad l := \overline{x\,|}\ ;\ r := \overline{x+1\,|}\ ; \\
\qquad\quad /\!/\!/\ \ [x \mapsto -, -] \star (1-p)^{\mathbf{spt}(l)} \star (1-p)^{\mathbf{spt}(r)} \qquad \text{(invariant \& frame rule)} \\
\qquad\quad \texttt{delete}\,(l)\ ; \\
\qquad\quad /\!/\!/\ \ [x \mapsto -, -] \star [\mathbf{emp}] \star (1-p)^{\mathbf{spt}(r)} \qquad \text{(invariant \& frame rule)} \\
\qquad\quad \texttt{delete}\,(r)\ ; \\
\qquad\quad /\!/\!/\ \ [x \mapsto -, -] \star [\mathbf{emp}] \star [\mathbf{emp}] \\
\qquad\quad \texttt{free}(x)\ ;\ \texttt{free}(x+1) \\
\quad \}\,\} \\
/\!/\!/\ \ [\mathbf{emp}]
\end{array}
$$

$/\!/ \quad (1-p)^{\mathbf{spt}(x)}$

```
delete (x):
    if (x ≠ 0) {
        {skip} [p] {
            l := |x| ; r := |x + 1| ;
```

$/\!/ \quad [x \mapsto -, -] \star (1-p)^{\mathbf{spt}(l)} \star (1-p)^{\mathbf{spt}(r)}$ (invariant & frame rule)

```
            delete (l) ;
```

$/\!/ \quad [x \mapsto -, -] \star [\mathbf{emp}] \star (1-p)^{\mathbf{spt}(r)}$ (invariant & frame rule)

```
            delete (r) ;
```

$/\!/ \quad [x \mapsto -, -] \star [\mathbf{emp}] \star [\mathbf{emp}]$

```
            free(x) ; free(x + 1)
    } }
```

$/\!/ \quad [\mathbf{emp}]$

Future work on Quantitative Separation Logic:

Future work on Quantitative Separation Logic:

- A syntax for quantitative separation logic

Future work on Quantitative Separation Logic:

- A syntax for quantitative separation logic
- Automation: Quantitative entailment? Quantitative symbolic heaps?

Future work on Quantitative Separation Logic:

- A syntax for quantitative separation logic
- Automation: Quantitative entailment? Quantitative symbolic heaps?
- A QSL–based calculus for expected runtimes / proving termination

Future work on Quantitative Separation Logic:

- A syntax for quantitative separation logic
- Automation: Quantitative entailment? Quantitative symbolic heaps?
- A QSL–based calculus for expected runtimes / proving termination
- Concurrency

Future work on Quantitative Separation Logic:

- A syntax for quantitative separation logic
- Automation: Quantitative entailment? Quantitative symbolic heaps?
- A QSL–based calculus for expected runtimes / proving termination
- Concurrency

Thank you for your kind attention!