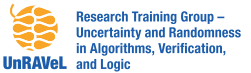
Reasoning about Expected Runtimes of Probabilistic Programs

(and Quantitative Separation Logic, time permitting)

Benjamin Lucien Kaminski



Dagstuhl Seminar 19371: **Deduction Beyond Satisfiability**

September 11, 2019, Schloss Dagstuhl, Germany

Probabilistic Program Applications

Probabilistic Program Applications

symmetry breaking

Probabilistic Program Applications

symmetry breaking

speed–up

Probabilistic Program Applications

symmetry breaking

speed–up

cryptography and security

Probabilistic Program Applications

symmetry breaking

speed–up

cryptography and security

machine learning & AI

Probabilistic Programs

Probabilistic Programs

- "Ordinary" programs with the additional ability to flip coins

$$\{\texttt{skip}\}\ [1/2]\ \{x := x + 2\}$$

Probabilistic Programs

- "Ordinary" programs with the additional ability to flip coins

$$\{\texttt{skip}\}\ [1/2]\ \{x := x + 2\}$$

- Control flow depends on outcome of coin flips

Probabilistic Programs

- "Ordinary" programs with the additional ability to flip coins

$$\{\texttt{skip}\}\ [1/2]\ \{x := x + 2\}$$

- Control flow depends on outcome of coin flips
- What does a probabilistic program C do?

Probabilistic Programs

- "Ordinary" programs with the additional ability to flip coins

$$\{\texttt{skip}\}\ [1/2]\ \{x := x + 2\}$$

- Control flow depends on outcome of coin flips
- What does a probabilistic program C do?
  - Run program C on initial state σ

Probabilistic Programs

- "Ordinary" programs with the additional ability to flip coins

$$\{\texttt{skip}\}\ [1/2]\ \{x := x + 2\}$$

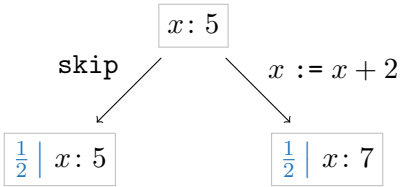
- Control flow depends on outcome of coin flips
- What does a probabilistic program C do?
  - Run program C on initial state σ
  - Obtain distribution $[\![C]\!]_\sigma$ over final states

Probabilistic Programs

- "Ordinary" programs with the additional ability to flip coins

$$\{\texttt{skip}\}\ [1/2]\ \{x := x + 2\}$$

- Control flow depends on outcome of coin flips
- What does a probabilistic program C do?
  - Run program C on initial state σ
  - Obtain distribution $[\![C]\!]_\sigma$ over final states

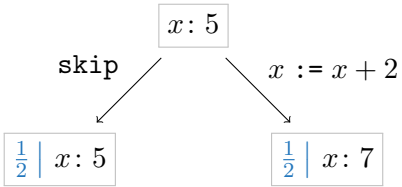


Probabilistic Programs

- "Ordinary" programs with the additional ability to flip coins

$$\{\texttt{skip}\}\ [1/2]\ \{x := x + 2\}$$

- Control flow depends on outcome of coin flips
- What does a probabilistic program C do?
  - Run program C on initial state σ
  - Obtain (sub-)distribution $[\![C]\!]_\sigma$ over final states



Expected Runtime Phenomena

Expected Runtime Phenomena

- ERT of C can be finite even if C admits infinite computations

Expected Runtime Phenomena

- ERT of C can be finite even if C admits infinite computations

---

$$x := 1;\ \texttt{while}\,(1/2)\ \{x := 2 \cdot x\}$$

Expected Runtime Phenomena

- ERT of C can be finite even if C admits infinite computations

$$x := 1;\ \texttt{while}\ (1/2)\ \{x := 2 \cdot x\}$$

Expected Runtime Phenomena

- ERT of C can be finite even if C admits infinite computations

---

$$x := 1;\ \texttt{while}\,(1/2)\ \{x := 2 \cdot x\}$$

Expected Runtime Phenomena

- ERT of C can be finite even if C admits infinite computations
- Positive almost–sure termination:

$$x := 1;\ \texttt{while}\,(1/2)\ \{x := 2 \cdot x\}$$

Expected Runtime Phenomena

- ERT of C can be finite even if C admits infinite computations
- Positive almost–sure termination:
  - ERT of C is finite

$$x := 1;\ \texttt{while}\,(1/2)\ \{x := 2 \cdot x\}$$

Expected Runtime Phenomena

- ERT of C can be finite even if C admits infinite computations
- Positive almost–sure termination:
  - ERT of C is finite

$$x := 1;\ \texttt{while}\,(1/2)\ \{x := 2 \cdot x\}\,;$$
$$\texttt{while}\,(x > 0)\ \{x := x - 1\}$$

Expected Runtime Phenomena

- ERT of C can be finite even if C admits infinite computations
- Positive almost–sure termination:
  - ERT of C is finite
  - Positive almost–sure termination not closed under sequential composition

---

$$x := 1;\ \texttt{while}\,(1/2)\ \{x := 2 \cdot x\}\,;$$
$$\texttt{while}\,(x > 0)\ \{x := x - 1\}$$

Expected Runtime Phenomena

- ERT of C can be finite even if C admits infinite computations
- Positive almost–sure termination:
  - ERT of C is finite
  - Positive almost–sure termination not closed under sequential composition
  - Reasoning about positive almost–sure termination is computationally very difficult:

$$x := 1;\ \texttt{while}\,(1/2)\ \{x := 2 \cdot x\}\,;$$
$$\texttt{while}\,(x > 0)\ \{x := x - 1\}$$

Expected Runtime Phenomena

- ERT of C can be finite even if C admits infinite computations
- Positive almost–sure termination:
  - ERT of C is finite
  - Positive almost–sure termination not closed under sequential composition
  - Reasoning about positive almost–sure termination is computationally very difficult:

    Strictly more difficult than termination of non–probabilistic programs [MFCS'15]

$$x := 1;\ \texttt{while}\,(1/2)\ \{x := 2 \cdot x\}\,;$$
$$\texttt{while}\,(x > 0)\ \{x := x - 1\}$$

[1] i.e. with probability 1

Expected Runtime Phenomena

- ERT of C can be finite even if C admits infinite computations
- Positive almost–sure termination:
  - ERT of C is finite
  - Positive almost–sure termination not closed under sequential composition
  - Reasoning about positive almost–sure termination is computationally very difficult:

    Strictly more difficult than termination of non–probabilistic programs [MFCS'15]
- ERT of C can be infinite, even if C terminates almost–surely[1]

$$x := 1;\ \texttt{while}\,(1/2)\ \{x := 2 \cdot x\}\,;$$
$$\texttt{while}\,(x > 0)\ \{x := x - 1\}$$

[1] i.e. with probability 1

Weakest Precondition Reasoning for Expected Runtimes

Domain for Expected Runtimes

Domain for Expected Runtimes

- ERT of C on input σ:

$$\sum_{i=1}^{\infty} i \cdot \Pr\left(\begin{array}{c} \text{“}C \text{ terminates after} \\ i \text{ steps on input } \sigma\text{”} \end{array} \right)$$

Domain for Expected Runtimes

- ERT of C on input σ:

$$\sum_{i=1}^{\infty} i \cdot \Pr\left(\begin{array}{c} \text{“}C \text{ terminates after} \\ i \text{ steps on input } \sigma\text{”} \end{array} \right)$$

- Set of runtimes $\mathbb{T} \;=\; \left\{ t \;\middle|\; t\colon \text{ States} \to \mathbb{R}_{\geq 0}^{\infty} \right\}$

Domain for Expected Runtimes

- ERT of C on input σ:

$$\sum_{i=1}^{\infty} i \cdot \Pr\left(\begin{array}{c} \text{“}C \text{ terminates after} \\ i \text{ steps on input } \sigma\text{”} \end{array} \right)$$

- Set of runtimes $\mathbb{T} \;=\; \left\{ t \;\middle|\; t\colon \text{ States} \to \mathbb{R}^{\infty}_{\geq 0} \right\}$

- associate runtimes (quantities) to program states

Domain for Expected Runtimes

- ERT of C on input σ:

$$\sum_{i=1}^{\infty} i \cdot \Pr\left(\begin{array}{c} \text{“}C \text{ terminates after} \\ i \text{ steps on input } \sigma\text{”} \end{array} \right)$$

- Set of runtimes $\mathbb{T} \;=\; \left\{ t \;\middle|\; t\colon \text{ States} \to \mathbb{R}^{\infty}_{\geq 0} \right\}$, a complete lattice with partial order

$$r \preceq t \qquad \text{iff} \qquad \forall\, \sigma \in \text{States}\colon \quad r(\sigma) \;\leq\; t(\sigma)$$

- associate runtimes (quantities) to program states

Weakest Precondition Reasoning for Expected Runtimes

The ert Transformer [ESOP'16, J.ACM'18]

Weakest Precondition Reasoning for Expected Runtimes

The ert Transformer [ESOP'16, J.ACM'18]

Use a continuation passing style ERT transformer $\mathrm{ert}[\![C]\!] \colon \mathbb{T} \to \mathbb{T}$.

Weakest Precondition Reasoning for Expected Runtimes

The ert Transformer [ESOP'16, J.ACM'18]

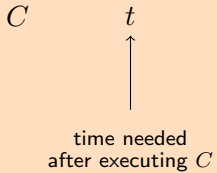
Use a continuation passing style ERT transformer $\mathrm{ert}[\![C]\!] \colon \mathbb{T} \to \mathbb{T}$.

$$C$$

Weakest Precondition Reasoning for Expected Runtimes

The ert Transformer [ESOP'16, J.ACM'18]

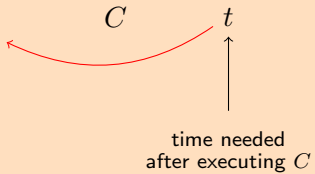
Use a continuation passing style ERT transformer $\mathsf{ert}[\![C]\!]\colon\ \mathbb{T} \to \mathbb{T}$.



Weakest Precondition Reasoning for Expected Runtimes

The ert Transformer [ESOP'16, J.ACM'18]

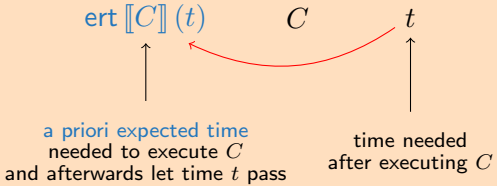
Use a continuation passing style ERT transformer $\mathsf{ert}[\![C]\!]\colon\ \mathbb{T} \to \mathbb{T}$.



Weakest Precondition Reasoning for Expected Runtimes

The ert Transformer [ESOP'16, J.ACM'18]

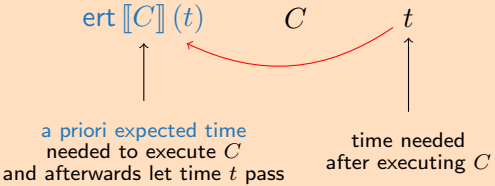
Use a continuation passing style ERT transformer $\mathrm{ert}[\![C]\!] \colon \mathbb{T} \to \mathbb{T}$.



Weakest Precondition Reasoning for Expected Runtimes

The ert Transformer [ESOP'16, J.ACM'18]

Use a continuation passing style ERT transformer $\mathsf{ert}[\![C]\!] \colon \mathbb{T} \to \mathbb{T}$.



ERT in Terms of ert

$$\mathsf{ert}[\![C]\!](\mathbf{0})(\sigma) \;=\; \text{“ERT of } C \text{ on input } \sigma\text{”}$$

ert of Skipping

ert of Skipping

$$\mathsf{ert}\,[\![\texttt{skip}]\!]\,(t) \;=\; 1 + t$$

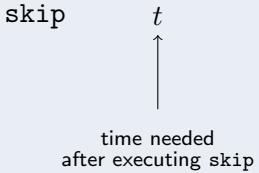
ert of Skipping

$$\mathsf{ert}\,[\![\texttt{skip}]\!]\,(t) \;=\; 1 + t$$

$$\texttt{skip}$$

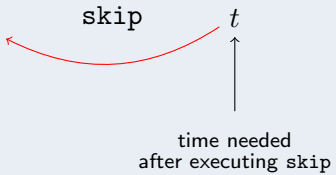
ert of Skipping

$$\text{ert}\,[\![\texttt{skip}]\!]\,(t) \;=\; 1 + t$$



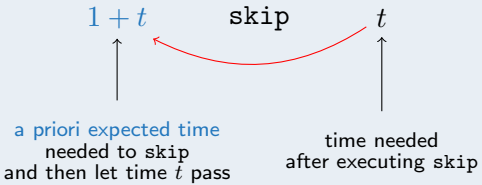
ert of Skipping

$$\mathsf{ert}\,[\![\texttt{skip}]\!]\,(t) \;=\; 1 + t$$



ert of Skipping

$$\text{ert}\,[\![\texttt{skip}]\!]\,(t) \;=\; 1 + t$$



ert of Assignments

ert of Assignments

$$\mathsf{ert}\,[\![x := E]\!]\,(t) \;=\; 1 + t\,[x/E]$$

ert of Assignments

$$\mathsf{ert}\,[\![x := E]\!]\,(t) \;=\; 1 + t\,[x/E]$$

$$x := z + 1$$

ert of Assignments

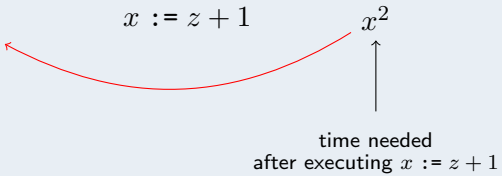
$$\text{ert}\,[\![x := E]\!]\,(t) \;=\; 1 + t\,[x/E]$$

$x := z + 1$ $\qquad$ x^2

time needed
after executing $x := z + 1$

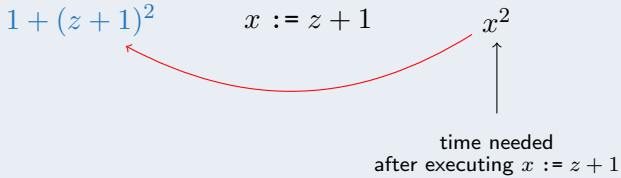
ert of Assignments

$$\text{ert}\,[\![x := E]\!]\,(t) \;=\; 1 + t\,[x/E]$$



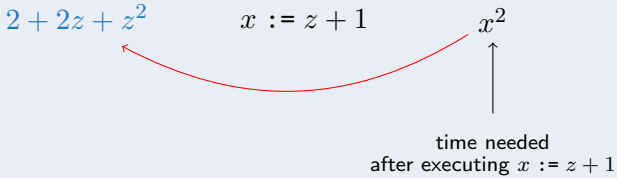
ert of Assignments

$$\text{ert}\,[\![x := E]\!]\,(t) \;=\; 1 + t\,[x/E]$$



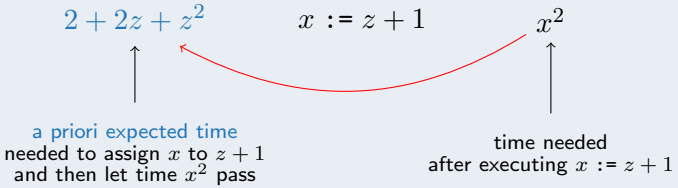
ert of Assignments

$$\text{ert}\,[\![x := E]\!]\,(t) \;=\; 1 + t\,[x/E]$$



ert of Assignments

$$\text{ert}\,[\![x := E]\!]\,(t) \;=\; 1 + t\,[x/E]$$



Rules for the ert Transformer

C	$\mathbf{ert}\,[\![C]\!]\,(t)$

Rules for the ert Transformer

C	$\mathbf{ert}\,[\![C]\!]\,(t)$
`skip`	$1 + t$
$x := E$	$1 + t\,[x/E]$

Rules for the ert Transformer

C	$\mathsf{ert}\,[\![C]\!]\,(t)$
`skip`	$1 + t$
$x := E$	$1 + t\,[x/E]$
$C_1 \fatsemi C_2$	$\mathsf{ert}\,[\![C_1]\!]\,(\mathsf{ert}\,[\![C_2]\!]\,(t))$

Rules for the ert Transformer

C	$\mathsf{ert}\,[\![C]\!]\,(t)$
$\texttt{skip}$	$1 + t$
$x := E$	$1 + t\,[x/E]$
$C_1 \fatsemi C_2$	$\mathsf{ert}\,[\![C_1]\!]\,(\mathsf{ert}\,[\![C_2]\!]\,(t))$
$\{C_1\}\,[p]\,\{C_2\}$	$1 + p \cdot \mathsf{ert}\,[\![C_1]\!]\,(t) + (1-p) \cdot \mathsf{ert}\,[\![C_2]\!]\,(t)$

Rules for the ert Transformer

C	$\mathsf{ert}\,[\![C]\!]\,(t)$
$\texttt{skip}$	$1 + t$
$x := E$	$1 + t\,[x/E]$
$C_1 ; C_2$	$\mathsf{ert}\,[\![C_1]\!]\,(\mathsf{ert}\,[\![C_2]\!]\,(t))$
$\{C_1\}\,[p]\,\{C_2\}$	$1 + p \cdot \mathsf{ert}\,[\![C_1]\!]\,(t) + (1-p) \cdot \mathsf{ert}\,[\![C_2]\!]\,(t)$
$\texttt{if}\,(\varphi)\,\{C_1\}\,\texttt{else}\,\{C_2\}$	$1 + [\varphi] \cdot \mathsf{ert}\,[\![C_1]\!]\,(t) + [\neg\varphi] \cdot \mathsf{ert}\,[\![C_2]\!]\,(t)$

Rules for the ert Transformer

C	$\mathsf{ert}\,[\![C]\!]\,(t)$
$\texttt{skip}$	$1 + t$
$x := E$	$1 + t\,[x/E]$
$C_1 ; C_2$	$\mathsf{ert}\,[\![C_1]\!]\,(\mathsf{ert}\,[\![C_2]\!]\,(t))$
$\{C_1\}\,[p]\,\{C_2\}$	$1 + p \cdot \mathsf{ert}\,[\![C_1]\!]\,(t) + (1-p) \cdot \mathsf{ert}\,[\![C_2]\!]\,(t)$
$\texttt{if}\,(\varphi)\,\{C_1\}\,\texttt{else}\,\{C_2\}$	$1 + [\varphi] \cdot \mathsf{ert}\,[\![C_1]\!]\,(t) + [\neg\varphi] \cdot \mathsf{ert}\,[\![C_2]\!]\,(t)$
$\texttt{while}\,(\varphi)\,\{C'\}$	$\mathsf{lfp}\,X \bullet \quad 1 + [\neg\varphi] \cdot t + [\varphi] \cdot \mathsf{ert}\,[\![C']\!]\,(X)$

The ert Transformer in Action

$$\{x \texttt{ := } 5\}\ [1/2]\ \{x \texttt{ := } 2\}\,;$$

$$\texttt{while}\,(x > 0)\ \big\{\ \{x \texttt{ := } x - 1\}\ [1/2]\ \{\texttt{skip}\}\ \big\}\,;$$

$$\texttt{skip}$$

The ert Transformer in Action

$$\{x := 5\}\ [1/2]\ \{x := 2\};$$

$$\texttt{while}\,(x > 0)\ \big\{\,\{x := x - 1\}\ [1/2]\ \{\texttt{skip}\}\,\big\};$$

$$\texttt{skip}$$

The ert Transformer in Action

$$\{x := 5\}\ [1/2]\ \{x := 2\};$$

$$\texttt{while}\,(x > 0)\ \{\{x := x - 1\}\ [1/2]\ \{\texttt{skip}\}\};$$

$$\texttt{skip}$$

$$/\!/\ \mathbf{0}$$

The ert Transformer in Action

$\{x := 5\}\ [1/2]\ \{x := 2\};$

$\texttt{while}\,(x > 0)\ \{\{x := x - 1\}\ [1/2]\ \{\texttt{skip}\}\};$

$/\!/\!/\ \mathbf{1 + 0}$

$\texttt{skip}$

$/\!/\!/\ \mathbf{0}$

The ert Transformer in Action

$\{x := 5\}\ [1/2]\ \{x := 2\}\,;$

$\texttt{while}\,(x > 0)\ \{\,\{x := x - 1\}\ [1/2]\ \{\texttt{skip}\}\,\}\,;$

$/\!/\!/\ \mathbf{1}$

$\texttt{skip}$

$/\!/\!/\ \mathbf{0}$

The ert Transformer in Action

$\{x := 5\}\ [1/2]\ \{x := 2\}\,;$

$\texttt{while}\,(x > 0)\ \big\{\,\{x := x - 1\}\ [1/2]\ \{\texttt{skip}\}\,\big\}\,;$

$/\!/\!/\ \mathbf{1}$

$\texttt{skip}$

$/\!/\!/\ \mathbf{0}$

The ert Transformer in Action

$$\{x := 5\}\ [1/2]\ \{x := 2\}\,;$$

$$/\!/\!/\ \mathbf{2 + [x > 0] \cdot 6x}$$

$$\texttt{while}\,(x > 0)\ \big\{\,\{x := x - 1\}\ [1/2]\ \{\texttt{skip}\}\,\big\}\,;$$

$$/\!/\!/\ \mathbf{1}$$

$$\texttt{skip}$$

$$/\!/\!/\ \mathbf{0}$$

The ert Transformer in Action

$$\{x := 5\}\ [1/2]\ \{x := 2\};$$

$$/\!/\!/\ \mathbf{2 + [x > 0] \cdot 6x}$$

$$\texttt{while}\,(x > 0)\ \{\{x := x - 1\}\ [1/2]\ \{\texttt{skip}\}\};$$

$$/\!/\!/\ \mathbf{1}$$

$$\texttt{skip}$$

$$/\!/\!/\ \mathbf{0}$$

The ert Transformer in Action

$$/\!/\!/\ \mathbf{1} \ + \ \tfrac{\mathbf{1}}{\mathbf{2}}\big(\mathbf{2} + [\mathbf{5} > \mathbf{0}] \cdot \mathbf{6} \cdot \mathbf{5} \ + \ \mathbf{2} + [\mathbf{2} > \mathbf{0}] \cdot \mathbf{6} \cdot \mathbf{2}\big)$$

$$\{x := 5\}\ [1/2]\ \{x := 2\};$$

$$/\!/\!/\ \mathbf{2} + [\boldsymbol{x} > \mathbf{0}] \cdot \mathbf{6}\boldsymbol{x}$$

$$\texttt{while}\,(x > 0)\ \big\{\, \{x := x - 1\}\ [1/2]\ \{\texttt{skip}\} \,\big\};$$

$$/\!/\!/\ \mathbf{1}$$

$$\texttt{skip}$$

$$/\!/\!/\ \mathbf{0}$$

The ert Transformer in Action

$$/\!/\!/ \; \mathbf{24}$$

$$\{x \coloneqq 5\} \; [1/2] \; \{x \coloneqq 2\};$$

$$/\!/\!/ \; \mathbf{2 + [x > 0] \cdot 6x}$$

$$\texttt{while}\,(x > 0) \; \big\{ \{x \coloneqq x - 1\} \; [1/2] \; \{\texttt{skip}\} \big\};$$

$$/\!/\!/ \; \mathbf{1}$$

$$\texttt{skip}$$

$$/\!/\!/ \; \mathbf{0}$$

Induction for Loops

Induction for ert of Loops

Induction for ert of Loops

- Definition of ert:

$$\mathsf{ert}\,[\![\texttt{while}\,(\varphi)\ \{C\}]\!]\,(t) \;=\; \mathsf{lfp}\,X.\ \underbrace{1 \;+\; [\neg\varphi]\cdot t \;+\; [\varphi]\cdot \mathsf{ert}\,[\![C]\!]\,(X)}_{=:\ \Phi(X)}$$

$$=\; \mathsf{lfp}\,\Phi$$

Induction for ert of Loops

- Definition of ert:

$$\mathsf{ert}\,[\![\texttt{while}\,(\varphi)\ \{C\}]\!]\,(t) \;=\; \mathsf{lfp}\,X.\ \underbrace{1 \;+\; [\neg\varphi]\cdot t \;+\; [\varphi]\cdot \mathsf{ert}\,[\![C]\!]\,(X)}_{=:\ \Phi(X)}$$

$$= \mathsf{lfp}\,\Phi$$

- Φ monotonic implies lfp exists [Knaster'28, Tarski'49, Kleene'62, *Folklore*]
- Problem: $\mathsf{lfp}\,\Phi$ highly non–computable! [MFCS'15, Acta Informatica'18]

Induction for ert of Loops

- Definition of ert:

$$\text{ert}\,[\![\texttt{while}\,(\varphi)\ \{C\}]\!]\,(t) \;=\; \text{lfp}\,X.\ \underbrace{1 \;+\; [\neg\varphi]\cdot t \;+\; [\varphi]\cdot \text{ert}\,[\![C]\!]\,(X)}_{=:\ \Phi(X)}$$

$$= \text{lfp}\,\Phi$$

- Φ monotonic implies lfp exists [Knaster'28, Tarski'49, Kleene'62, *Folklore*]
- Problem: $\text{lfp}\,\Phi$ highly non–computable! [MFCS'15, Acta Informatica'18]
- Reasoning about lfp needed: Induction! [Park '69]

$$\Phi(I) \preceq I \quad \text{implies} \quad \text{lfp}\,\Phi \preceq I$$

Induction Example

$$\texttt{while}\,(x > 0)$$
$$\quad \{x \texttt{ := } x - 1\}\,[1/2]\,\{\texttt{skip}\}$$
$$\}$$

Induction Example

$$\texttt{while}\,(x > 0)$$
$$\quad \{x \texttt{ := } x - 1\}\ [1/2]\ \{\texttt{skip}\}$$
$$\}$$

Postruntime: 1 (expected time of executing the loop and then needing 1 time unit)

Induction Example

$$\texttt{while}\,(x > 0)\;\{x \texttt{ := } x-1\}\,[1/2]\,\{\texttt{skip}\}$$

Postruntime: 1 (expected time of executing the loop and then needing 1 time unit)

Characteristic function $\Phi(X) \;=\; 2 \;+\; [x>0]\left(1+\frac{1}{2}\big(X\,[x/x-1]+X\big)\right)$

Induction Example

$$\texttt{while}\,(x > 0)$$
$$\quad \{x := x - 1\}\,[1/2]\,\{\texttt{skip}\}$$
$$\}$$

Postruntime: 1 (expected time of executing the loop and then needing 1 time unit)

Characteristic function $\Phi(X) \;=\; 2 \;+\; [x > 0]\left(1 + \frac{1}{2}\big(X\,[x/x-1] + X\big)\right)$

Candidate for upper bound: $\boldsymbol{I \;=\; 2 + [x > 0]\,6x}$

Induction Example

$$\texttt{while}\,(x > 0)$$
$$\quad \{x := x - 1\}\,[1/2]\,\{\texttt{skip}\}$$
$$\}$$

Postruntime: 1 (expected time of executing the loop and then needing 1 time unit)

Characteristic function $\Phi(X) \;=\; 2 \;+\; [x > 0]\left(1 + \frac{1}{2}\big(X\,[x/x-1] + X\big)\right)$

Candidate for upper bound: $\boldsymbol{I \;=\; 2 + [x > 0]\,6x}$

Induction:

Induction Example

$$\texttt{while}\,(x > 0)$$
$$\quad \{x \texttt{ := } x - 1\}\,[1/2]\,\{\texttt{skip}\}$$
$$\}$$

Postruntime: 1 (expected time of executing the loop and then needing 1 time unit)

Characteristic function $\Phi(X) \;=\; 2 \;+\; [x > 0]\left(1 + \frac{1}{2}\big(X\,[x/x-1] + X\big)\right)$

Candidate for upper bound: $\boldsymbol{I \;=\; 2 + [x > 0]\,6x}$

Induction: $\boldsymbol{\Phi(I)}$

Induction Example

$$\texttt{while}\,(x > 0)\;\{x := x - 1\}\,[1/2]\,\{\texttt{skip}\}$$

Postruntime: 1 (expected time of executing the loop and then needing 1 time unit)

Characteristic function $\Phi(X) \;=\; 2 \;+\; [x > 0]\left(1 + \frac{1}{2}\big(X\,[x/x-1] + X\big)\right)$

Candidate for upper bound: $\boldsymbol{I \;=\; 2 + [x > 0]\,6x}$

Induction: $\boldsymbol{\Phi(I)} \;=\; 2 \;+\; [x > 0]\left(1 + \frac{1}{2}\big(\quad\big)\right)$

Induction Example

$$\texttt{while}\,(x > 0)\;\{x := x - 1\}\,[1/2]\,\{\texttt{skip}\}$$

Postruntime: 1 (expected time of executing the loop and then needing 1 time unit)

Characteristic function $\Phi(X) \;=\; 2 \;+\; [x > 0]\left(1 + \frac{1}{2}\left(X\,[x/x-1] + X\right)\right)$

Candidate for upper bound: $\boldsymbol{I \;=\; 2 + [x > 0]\,6x}$

Induction: $\boldsymbol{\Phi(I)} \;=\; 2 \;+\; [x > 0]\left(1 + \frac{1}{2}\left(2 + [x - 1 > 0]\,6(x-1)\right)\right)$

Induction Example

$$\texttt{while}\,(x > 0)\;\{x \texttt{ := } x-1\}\,[1/2]\,\{\texttt{skip}\}$$

Postruntime: 1 (expected time of executing the loop and then needing 1 time unit)

Characteristic function $\Phi(X) \;=\; 2 \;+\; [x>0]\left(1+\frac{1}{2}\big(X\,[x/x-1]+X\big)\right)$

Candidate for upper bound: $\boldsymbol{I \;=\; 2+[x>0]\,6x}$

Induction: $\boldsymbol{\Phi(I)} \;=\; 2 \;+\; [x>0]\left(1+\frac{1}{2}\big(2+[x-1>0]\,6(x-1) \;+\; 2+[x>0]\,6x\big)\right)$

Induction Example

$$\texttt{while}\,(x > 0)\;\{x := x - 1\}\,[1/2]\,\{\texttt{skip}\}$$

Postruntime: 1 (expected time of executing the loop and then needing 1 time unit)

Characteristic function $\Phi(X) \;=\; 2 \;+\; [x > 0]\left(1 + \frac{1}{2}\big(X\,[x/x-1] + X\big)\right)$

Candidate for upper bound: $\boldsymbol{I \;=\; 2 + [x > 0]\,6x}$

Induction: $\boldsymbol{\Phi(I)} \;=\; 2 \;+\; [x > 0]\left(1 + \frac{1}{2}\big(2 + [x-1 > 0]\,6(x-1) \;+\; 2 + [x > 0]\,6x\big)\right)$

$$= \; 2 + [x > 0]\,6x$$

Induction Example

$$\texttt{while}\,(x > 0)\;\{x := x-1\}\,[1/2]\,\{\texttt{skip}\}$$

Postruntime: 1 (expected time of executing the loop and then needing 1 time unit)

Characteristic function $\Phi(X) \;=\; 2 \;+\; [x>0]\left(1+\frac{1}{2}\big(X\,[x/x-1]+X\big)\right)$

Candidate for upper bound: $\boldsymbol{I \;=\; 2+[x>0]\,6x}$

Induction: $\boldsymbol{\Phi(I)} \;=\; 2 \;+\; [x>0]\left(1+\frac{1}{2}\big(2+[x-1>0]\,6(x-1) \;+\; 2+[x>0]\,6x\big)\right)$

$$=\; 2+[x>0]\,6x \;=\; I$$

Induction Example

$$\texttt{while}\,(x > 0)\;\{x := x - 1\}\,[1/2]\,\{\texttt{skip}\}$$

Postruntime: 1 (expected time of executing the loop and then needing 1 time unit)

Characteristic function $\Phi(X) \;=\; 2 \;+\; [x > 0]\left(1 + \frac{1}{2}\big(X\,[x/x-1] + X\big)\right)$

Candidate for upper bound: $\boldsymbol{I \;=\; 2 + [x > 0]\,6x}$

Induction: $\boldsymbol{\Phi(I)} \;=\; 2 \;+\; [x > 0]\left(1 + \frac{1}{2}\big(2 + [x - 1 > 0]\,6(x-1) \;+\; 2 + [x > 0]\,6x\big)\right)$

$$= \; 2 + [x > 0]\,6x \;=\; I \;\preceq\; \boldsymbol{I}$$

Induction Example

$$\texttt{while}\,(x > 0)\;\{x := x - 1\}\,[1/2]\,\{\texttt{skip}\}$$

Postruntime: 1 (expected time of executing the loop and then needing 1 time unit)

Characteristic function $\Phi(X) \;=\; 2 \;+\; [x > 0]\left(1 + \frac{1}{2}\big(X\,[x/x-1] + X\big)\right)$

Candidate for upper bound: $\boldsymbol{I \;=\; 2 + [x > 0]\,6x}$

Induction: $\boldsymbol{\Phi(I)} \;=\; 2 \;+\; [x > 0]\left(1 + \frac{1}{2}\big(2 + [x - 1 > 0]\,6(x-1) \;+\; 2 + [x > 0]\,6x\big)\right)$

$$= 2 + [x > 0]\,6x \;=\; I \;\preceq\; \boldsymbol{I}$$

Induction Example

$$\texttt{while}\,(x > 0)\;\{x := x - 1\}\,[1/2]\,\{\texttt{skip}\}$$

Postruntime: 1 (expected time of executing the loop and then needing 1 time unit)

Characteristic function $\Phi(X) \;=\; 2 \;+\; [x > 0]\left(1 + \frac{1}{2}\big(X\,[x/x-1] + X\big)\right)$

Candidate for upper bound: $\boldsymbol{I \;=\; 2 + [x > 0]\,6x}$

Induction: $\boldsymbol{\Phi(I)} \;=\; 2 \;+\; [x > 0]\left(1 + \frac{1}{2}\big(2 + [x - 1 > 0]\,6(x-1) \;+\; 2 + [x > 0]\,6x\big)\right)$

$= 2 + [x > 0]\,6x \;=\; I \;\preceq\; \boldsymbol{I}$ ✓

Induction Example

$$\texttt{while}\,(x > 0)\;\{x := x-1\}\,[1/2]\,\{\texttt{skip}\}$$

Postruntime: 1 (expected time of executing the loop and then needing 1 time unit)

Characteristic function $\Phi(X) \;=\; 2 \;+\; [x>0]\left(1+\frac{1}{2}\big(X\,[x/x-1]+X\big)\right)$

Candidate for upper bound: $\boldsymbol{I \;=\; 2+[x>0]\,6x}$

Induction: $\boldsymbol{\Phi(I)} \;=\; 2 \;+\; [x>0]\left(1+\frac{1}{2}\big(2+[x-1>0]\,6(x-1) \;+\; 2+[x>0]\,6x\big)\right)$

$$= 2+[x>0]\,6x \;=\; I \;\preceq\; \boldsymbol{I} \quad \checkmark$$

Thus: $\boldsymbol{\mathsf{ert}\,[\![\texttt{while}\ldots]\!]\,(1) \quad \preceq \quad 2+[x>0]\,6x}$

Induction Example

$$\texttt{while}\,(x > 0)\;\{x \texttt{ := } x-1\}\,[1/2]\,\{\texttt{skip}\}$$

Postruntime: 1 (expected time of executing the loop and then needing 1 time unit)

Characteristic function $\Phi(X) \;=\; 2 \;+\; [x>0]\left(1+\frac{1}{2}\big(X\,[x/x-1]+X\big)\right)$

Candidate for upper bound: $\boldsymbol{I \;=\; 2+[x>0]\,6x}$

Induction: $\boldsymbol{\Phi(I)} \;=\; 2 \;+\; [x>0]\left(1+\frac{1}{2}\big(2+[x-1>0]\,6(x-1) \;+\; 2+[x>0]\,6x\big)\right)$

$$= 2+[x>0]\,6x \;=\; I \;\preceq\; \boldsymbol{I} \quad \checkmark$$

Thus: $\quad \textbf{ert}\,[\![\texttt{while}\ldots]\!]\,(1) \quad \preceq \quad \boldsymbol{2+[x>0]\,6x} \quad \preceq \quad \boldsymbol{7x}$

Work building on the ert calculus:

Work building on the ert calculus:

- ert calculus for recursive probabilistic programs [LICS'16]

Work building on the ert calculus:

- ert calculus for recursive probabilistic programs [LICS'16]

ert calculus has been Isabelle/HOL certified [Hölzl'16]

Work building on the ert calculus:

- ert calculus for recursive probabilistic programs [LICS'16]

ert calculus has been Isabelle/HOL certified [Hölzl'16]

- ert calculus was used to prove soundness of an automated technique for inferring expected runtimes [Ngo *et al.*'18]

Work building on the ert calculus:

- ert calculus for recursive probabilistic programs [LICS'16]

ert calculus has been Isabelle/HOL certified [Hölzl'16]

- ert calculus was used to prove soundness of an automated technique for inferring expected runtimes [Ngo *et al.*'18]
- A fairly simple (but incomplete) rule for lower bounds [{arXiv'19} [ε] {POPL'20}]

Work building on the ert calculus:

- ert calculus for recursive probabilistic programs [LICS'16]

ert calculus has been Isabelle/HOL certified [Hölzl'16]

- ert calculus was used to prove soundness of an automated technique for inferring expected runtimes [Ngo *et al.*'18]
- A fairly simple (but incomplete) rule for lower bounds [{arXiv'19} [ε] {POPL'20}]

Open Problems (for this community?):

Work building on the ert calculus:

- ert calculus for recursive probabilistic programs [LICS'16]

ert calculus has been Isabelle/HOL certified [Hölzl'16]

- ert calculus was used to prove soundness of an automated technique for inferring expected runtimes [Ngo *et al.*'18]
- A fairly simple (but incomplete) rule for lower bounds [{arXiv'19} [ε] {POPL'20}]

Open Problems (for this community?):

- **Quantitative "entailment" checking: Does $\Phi(I) \preceq I$ hold?**

Work building on the ert calculus:

- ert calculus for recursive probabilistic programs [LICS'16]

ert calculus has been Isabelle/HOL certified [Hölzl'16]

- ert calculus was used to prove soundness of an automated technique for inferring expected runtimes [Ngo *et al.*'18]
- A fairly simple (but incomplete) rule for lower bounds [{arXiv'19} [ε] {POPL'20}]

Open Problems (for this community?):

- **Quantitative "entailment" checking: Does $\Phi(I) \preceq I$ hold?**
- **Synthesis of quantitative invariants: Find I, such that $\Phi(I) \preceq I$!**

Work building on the ert calculus:

- ert calculus for recursive probabilistic programs [LICS'16]
- 
  ert calculus has been Isabelle/HOL certified [Hölzl'16]
- ert calculus was used to prove soundness of an automated technique for inferring expected runtimes [Ngo *et al.*'18]
- A fairly simple (but incomplete) rule for lower bounds [{arXiv'19} [ε] {POPL'20}]

Open Problems (for this community?):

- **Quantitative "entailment" checking: Does $\Phi(I) \preceq I$ hold?**
- **Synthesis of quantitative invariants: Find I, such that $\Phi(I) \preceq I$!**
- **Verifying lower bounds: Does $I \preceq \mathrm{lfp}\ \Phi$ hold?**

Quantitative Separation Logic

Quantitative Separating Conjunction [POPL'19]

Classical separating conjunction:

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

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

Quantitative Separating Conjunction [POPL'19]

Classical separating conjunction:

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

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

Naive quantitative separating conjunction:

$$\big(f \star g\big)(s,\, h) \quad = \quad \exists\, h_1, h_2 \colon \; [h = h_1 \star h_2] \cdot f(s,h_1) \cdot g(s,h_2)$$

Quantitative Separating Conjunction [POPL'19]

Classical separating conjunction:

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

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

Naive quantitative separating conjunction:

$$\big(f \star g\big)(s,\, h) \quad = \quad \exists\, h_1, h_2 \colon \; [h = h_1 \star h_2] \cdot f(s,h_1) \cdot g(s,h_2)$$

Meaningful quantitative separating conjunction:

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

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 [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:

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

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:

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

Modus Ponens

Classical modus ponens:

$$F \wedge (F \Longrightarrow G) \quad \Longrightarrow \quad G$$

Modus Ponens

Classical modus ponens:

$$F \wedge (F \implies G) \qquad \Longrightarrow \qquad G$$

Modus ponens of classical separation logic:

$$F \star (F \mathrel{-\!\!\star} G) \qquad \Longrightarrow \qquad G$$

Modus Ponens

Classical modus ponens:

$$F \wedge (F \implies G) \qquad \Longrightarrow \qquad G$$

Modus ponens of classical separation logic:

$$F \star (F \mathrel{-\!\!\star} G) \qquad \Longrightarrow \qquad G$$

Modus ponens of quantitative separation logic:

$$f \star \big(f \mathrel{-\!\!\star} g\big) \qquad \preceq \qquad g$$

Adjointness of $\mathrel{-\!\!*}$ and $\star$

Classical adjointness:

$$\big((F \star G) \implies J\big) \quad \iff \quad \big(F \implies (G \mathrel{-\!\!*} J)\big)$$

Adjointness of $\mathrel{-\!\!*}$ and $\star$

Classical adjointness:

$$\big((F \star G) \implies J\big) \qquad \iff \qquad \big(F \implies (G \mathrel{-\!\!*} J)\big)$$

Quantitative Adjointness:

$$f \star g \;\preceq\; j \qquad \text{iff} \qquad f \;\preceq\; g \mathrel{-\!\!*} j$$

Adjointness of $\mathrel{-\!\!\star}$ and $\star$

Classical adjointness:

$$\big((F \star G) \implies J\big) \qquad \iff \qquad \big(F \implies (G \mathrel{-\!\!\star} J)\big)$$

Quantitative Adjointness:

$$f \star g \;\preceq\; j \qquad \text{iff} \qquad f \;\preceq\; g \mathrel{-\!\!\star} j$$

My personal Aha Erlebnis:

$$a - b \;\preceq\; c \qquad \text{iff} \qquad a \;\preceq\; b + c$$

$$a \cdot b \;\preceq\; c \qquad \text{iff} \qquad a \;\preceq\; c \,/\, b$$

Overview of QSL:

[2]Bozga, Iosif, Perarnau. Quantitative Separation Logic and Programs with Lists. IJCAR'08.

Overview of QSL:

- Conservative extension of classical separation

[2]Bozga, Iosif, Perarnau. Quantitative Separation Logic and Programs with Lists. IJCAR'08.

Overview of QSL:

- Conservative extension of classical separation
- Weakest–precondition–style reasoning about probabilistic programs with pointers

[2] Bozga, Iosif, Perarnau. Quantitative Separation Logic and Programs with Lists. IJCAR'08.

Overview of QSL:

- Conservative extension of classical separation
- Weakest–precondition–style reasoning about probabilistic programs with pointers
  - Not possible with the "Quantitative Separation Logic" of [Bozga,Iosif,Perarnau'08][2]

[2] Bozga, Iosif, Perarnau. Quantitative Separation Logic and Programs with Lists. IJCAR'08.

Overview of QSL:

- Conservative extension of classical separation
- Weakest–precondition–style reasoning about probabilistic programs with pointers
  - Not possible with the "Quantitative Separation Logic" of [Bozga,Iosif,Perarnau'08][2]
- Supports inductive definitions of quantities

[2]Bozga, Iosif, Perarnau. Quantitative Separation Logic and Programs with Lists. IJCAR'08.

Overview of QSL:

- Conservative extension of classical separation
- Weakest–precondition–style reasoning about probabilistic programs with pointers
  - Not possible with the "Quantitative Separation Logic" of [Bozga,Iosif,Perarnau'08][2]
- Supports inductive definitions of quantities

Future Work on QSL:

[2] Bozga, Iosif, Perarnau. Quantitative Separation Logic and Programs with Lists. IJCAR'08.

Overview of QSL:

- Conservative extension of classical separation
- Weakest–precondition–style reasoning about probabilistic programs with pointers
  - Not possible with the "Quantitative Separation Logic" of [Bozga,Iosif,Perarnau'08][2]
- Supports inductive definitions of quantities

Future Work on QSL:

- A syntax for quantitative separation logic

[2] Bozga, Iosif, Perarnau. Quantitative Separation Logic and Programs with Lists. IJCAR'08.

Overview of QSL:

- Conservative extension of classical separation
- Weakest–precondition–style reasoning about probabilistic programs with pointers
  - Not possible with the "Quantitative Separation Logic" of [Bozga,Iosif,Perarnau'08][2]
- Supports inductive definitions of quantities

Future Work on QSL:

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

[2] Bozga, Iosif, Perarnau. Quantitative Separation Logic and Programs with Lists. IJCAR'08.

Overview of QSL:

- Conservative extension of classical separation
- Weakest–precondition–style reasoning about probabilistic programs with pointers
  - Not possible with the "Quantitative Separation Logic" of [Bozga,Iosif,Perarnau'08][2]
- Supports inductive definitions of quantities

Future Work on QSL:

- A syntax for quantitative separation logic
- Automation: Quantitative entailment? Quantitative symbolic heaps?
- Cyclic proofs for inductive quantities?
- Concurrency

[2] Bozga, Iosif, Perarnau. Quantitative Separation Logic and Programs with Lists. IJCAR'08.

Overview of QSL:

- Conservative extension of classical separation
- Weakest–precondition–style reasoning about probabilistic programs with pointers
  - Not possible with the "Quantitative Separation Logic" of [Bozga,Iosif,Perarnau'08][2]
- Supports inductive definitions of quantities

Future Work on QSL:

- A syntax for quantitative separation logic
- Automation: Quantitative entailment? Quantitative symbolic heaps?
- Cyclic proofs for inductive quantities?
- Concurrency

Thank you for your kind attention!

[2] Bozga, Iosif, Perarnau. Quantitative Separation Logic and Programs with Lists. IJCAR'08.