Automated Deductive Verification of Probabilistic Programs

Kevin Batz

Software Modeling and Verification Chair

RWTH AACHEN UNIVERSITY

PhD Defense

December 20, 2024, Aachen, Germany

Deductive Program Verification

Deductive Program Verification
=
mathematically prove that programs do not fail

Deductive Program Verification
=
mathematically prove that programs do not fail

Why?

Deductive Program Verification
=
mathematically prove that programs do not fail

Why? **Because software is ubiquitous!**

Deductive Program Verification
=
mathematically prove that programs do not fail

Why? **Because software is ubiquitous!**

Software bugs cost tremendous amounts of money and have caused human deaths.

Probabilistic programs

Probabilistic programs = imperative `while`-programs + coin flips

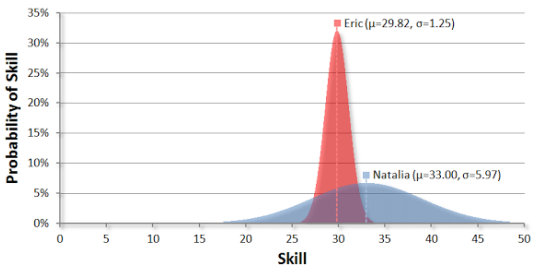
Probabilistic programs = imperative `while`-programs + coin flips

$$x := 0; y := 1; \texttt{while}\,(y = 1)\,\{\, y := 0 \;[1/2]\; x := x + 1 \,\}$$

Probabilistic programs = imperative `while`-programs + coin flips

$$x := 0; y := 1; \texttt{while}\,(y = 1)\,\{\, y := 0 \;[1/2]\; x := x + 1 \,\}$$

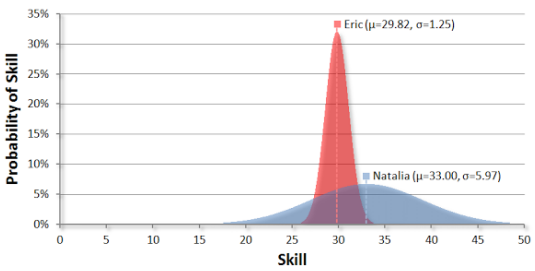
Machine Learning



Probabilistic programs = imperative `while`-programs + coin flips

$$x := 0; y := 1; \texttt{while}\,(y = 1)\,\{\, y := 0\ [1/2]\ x := x + 1 \,\}$$

Machine Learning



Security

The Bounded Retransmission Protocol [Helmink et al. '93, D'Argenio et al. '97]

Goal: Send file of $N = 8 \cdot 10^9$ packets via lossy channel.

The Bounded Retransmission Protocol [Helmink et al. '93, D'Argenio et al. '97]

Goal: Send file of $N = 8 \cdot 10^9$ packets via lossy channel.

Transmitting a packet fails with probability 0.01.

The Bounded Retransmission Protocol [Helmink et al. '93, D'Argenio et al. '97]

Goal: Send file of $N = 8 \cdot 10^9$ packets via lossy channel.

Transmitting a packet fails with probability 0.01.

Allow for at most $M = 10$ retransmissions per packet.

The Bounded Retransmission Protocol [Helmink et al. '93, D'Argenio et al. '97]

Goal: Send file of $N = 8 \cdot 10^9$ packets via lossy channel.

Transmitting a packet fails with probability 0.01.

Allow for at most $M = 10$ retransmissions per packet.

$$sent := 0; failed := 0;$$

$$\texttt{while}\,(sent < N \wedge failed \leq M)\,\texttt{do}$$

$$\{\underbrace{failed := failed + 1}_{\text{failed transmission}}\}[0.01]\{\underbrace{failed := 0; sent := sent + 1}_{\text{successful transmission}}\}$$

$$\texttt{end}$$

The Bounded Retransmission Protocol [Helmink et al. '93, D'Argenio et al. '97]

Goal: Send file of $N = 8 \cdot 10^9$ packets via lossy channel.

Transmitting a packet fails with probability 0.01.

Allow for at most $M = 10$ retransmissions per packet.

$$sent := 0; failed := 0;$$

$$\texttt{while}\,(sent < N \wedge failed \leq M)\,\texttt{do}$$

$$\{\underbrace{failed := failed + 1}_{\text{failed transmission}}\}[0.01]\{\underbrace{failed := 0; sent := sent + 1}_{\text{successful transmission}}\}$$

$$\texttt{end}$$

$$\textbf{Prove:} \quad \Pr[\Diamond\,(failed > M)] \leq 0.001$$

Outline & High-Level Goals

Outline:

The Weakest Preexpectation Calculus

Relatively Complete Verification [POPL '21]

Latticed k-Induction [CAV '21]

Inductive Synthesis of Inductive Invariants [TACAS '23]

Property Directed Reachability [CAV '20]

Outline & High-Level Goals

Outline:

The Weakest Preexpectation Calculus

Relatively Complete Verification [POPL '21]

Latticed k-Induction [CAV '21]

Inductive Synthesis of Inductive Invariants [TACAS '23]

Property Directed Reachability [CAV '20]

High-Level Contributions:

1. Provide theoretical **foundations for automated deductive verifiers**

Outline & High-Level Goals

Outline:

The Weakest Preexpectation Calculus

Relatively Complete Verification [POPL '21]

Latticed k-Induction [CAV '21]

Inductive Synthesis of Inductive Invariants [TACAS '23]

Property Directed Reachability [CAV '20]

High-Level Contributions:

1. Provide theoretical **foundations for automated deductive verifiers**
2. Advance the state-of-the-art for the **fully automatic verification of loops**

The Weakest Preexpectation Calculus

[Kozen '83, McIver '99, McIver & Morgan '05, Kaminski '19]

The Weakest Preexpectation Calculus

[Kozen '83, McIver '99, McIver & Morgan '05, Kaminski '19]

What is the **expected final value** of a program variable?

The Weakest Preexpectation Calculus

[Kozen '83, McIver '99, McIver & Morgan '05, Kaminski '19]

What is the **expected final value** of a program variable?

What is the **probability** of reaching a given final state?

The Weakest Preexpectation Calculus

- $\text{States} = \{\sigma \mid \sigma \colon \text{Vars} \to \mathbb{Q}_{\geq 0}\}$

The Weakest Preexpectation Calculus

- $\mathsf{States} = \{\sigma \mid \sigma \colon \mathsf{Vars} \to \mathbb{Q}_{\geq 0}\}$
- Complete lattice $(\mathbb{E}, \sqsubseteq)$ of *expectations*:

$$\mathbb{E} = \{f \mid f \colon \mathsf{States} \to \mathbb{R}^{\infty}_{\geq 0}\} \qquad \text{where} \qquad f \sqsubseteq g \quad \text{iff} \quad \forall \sigma \in \mathsf{States} \colon f(\sigma) \leq g(\sigma)$$

The Weakest Preexpectation Calculus

- $\mathsf{States} = \{\sigma \mid \sigma \colon \mathsf{Vars} \to \mathbb{Q}_{\geq 0}\}$
- Complete lattice $(\mathbb{E}, \sqsubseteq)$ of *expectations*:

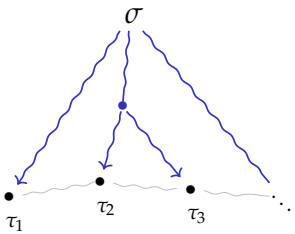
$$\mathbb{E} = \{f \mid f \colon \mathsf{States} \to \mathbb{R}^{\infty}_{\geq 0}\} \qquad \text{where} \qquad f \sqsubseteq g \quad \text{iff} \quad \forall \sigma \in \mathsf{States} \colon f(\sigma) \leq g(\sigma)$$

What is the expected value of f after running C on initial state σ?

- $\text{States} = \{\sigma \mid \sigma\colon \text{Vars} \to \mathbb{Q}_{\geq 0}\}$
- Complete lattice $(\mathbb{E}, \sqsubseteq)$ of *expectations*:

$$\mathbb{E} = \{f \mid f\colon \text{States} \to \mathbb{R}^{\infty}_{\geq 0}\} \qquad \text{where} \qquad f \sqsubseteq g \quad \text{iff} \quad \forall \sigma \in \text{States}\colon f(\sigma) \leq g(\sigma)$$

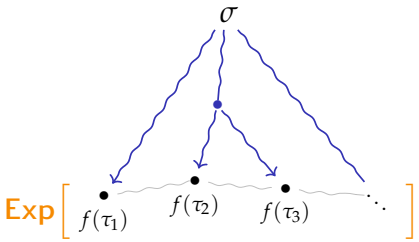
What is the expected value of f after running C on initial state σ?



- $\text{States} = \{\sigma \mid \sigma\colon \text{Vars} \to \mathbb{Q}_{\geq 0}\}$
- Complete lattice $(\mathbb{E}, \sqsubseteq)$ of *expectations*:

$$\mathbb{E} = \{f \mid f\colon \text{States} \to \mathbb{R}^{\infty}_{\geq 0}\} \qquad \text{where} \qquad f \sqsubseteq g \quad \text{iff} \quad \forall \sigma \in \text{States}\colon f(\sigma) \leq g(\sigma)$$

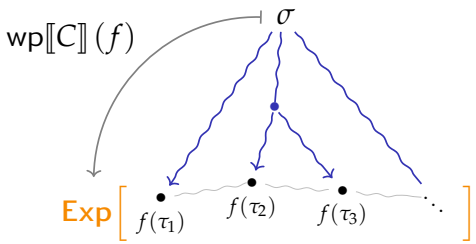
What is the expected value of f after running C on initial state σ?



- States $= \{\sigma \mid \sigma\colon \mathsf{Vars} \to \mathbb{Q}_{\geq 0}\}$
- Complete lattice $(\mathbb{E}, \sqsubseteq)$ of *expectations*:

$$\mathbb{E} = \{f \mid f\colon \mathsf{States} \to \mathbb{R}^{\infty}_{\geq 0}\} \qquad \text{where} \qquad f \sqsubseteq g \quad \text{iff} \quad \forall \sigma \in \mathsf{States}\colon f(\sigma) \leq g(\sigma)$$

What is the expected value of f after running C on initial state σ?



- $\mathsf{States} = \{\sigma \mid \sigma \colon \mathsf{Vars} \to \mathbb{Q}_{\geq 0}\}$
- Complete lattice $(\mathbb{E}, \sqsubseteq)$ of *expectations*:

$$\mathbb{E} = \{f \mid f \colon \mathsf{States} \to \mathbb{R}^{\infty}_{\geq 0}\} \qquad \text{where} \qquad f \sqsubseteq g \quad \text{iff} \quad \forall \sigma \in \mathsf{States} \colon f(\sigma) \leq g(\sigma)$$

What is the expected value of f after running C on initial state σ?

$$\mathsf{wp}[\![\{\,\mathtt{skip}\,\}\,[1/2]\,\{\,x := x + 2\,\}]\!]\,(x)$$

- $\mathsf{States} = \{\sigma \mid \sigma \colon \mathsf{Vars} \to \mathbb{Q}_{\geq 0}\}$
- Complete lattice $(\mathbb{E}, \sqsubseteq)$ of *expectations*:

$$\mathbb{E} = \{f \mid f \colon \mathsf{States} \to \mathbb{R}^{\infty}_{\geq 0}\} \qquad \text{where} \qquad f \sqsubseteq g \quad \text{iff} \quad \forall \sigma \in \mathsf{States} \colon f(\sigma) \leq g(\sigma)$$

What is the expected value of f after running C on initial state σ?

$$\mathsf{wp}[\![\{\,\mathtt{skip}\,\}\,[1/2]\,\{\,x := x+2\,\}]\!](x) \;=\; 1/2 \cdot x + 1/2 \cdot (x+2)$$

- $\mathsf{States} = \{\sigma \mid \sigma\colon \mathsf{Vars} \to \mathbb{Q}_{\geq 0}\}$
- Complete lattice $(\mathbb{E}, \sqsubseteq)$ of *expectations*:

$$\mathbb{E} = \{f \mid f\colon \mathsf{States} \to \mathbb{R}^{\infty}_{\geq 0}\} \qquad \text{where} \qquad f \sqsubseteq g \quad \text{iff} \quad \forall \sigma \in \mathsf{States}\colon f(\sigma) \leq g(\sigma)$$

What is the expected value of f after running C on initial state σ?

$$\mathsf{wp}[\![\{\,\mathtt{skip}\,\}\,[1/2]\,\{\,x := x+2\,\}]\!](x) \;=\; 1/2 \cdot x + 1/2 \cdot (x+2) \;=\; x+1$$

- $\mathsf{States} = \{\sigma \mid \sigma\colon \mathsf{Vars} \to \mathbb{Q}_{\geq 0}\}$
- Complete lattice $(\mathbb{E}, \sqsubseteq)$ of *expectations*:

$$\mathbb{E} = \{f \mid f\colon \mathsf{States} \to \mathbb{R}^{\infty}_{\geq 0}\} \qquad \text{where} \qquad f \sqsubseteq g \quad \text{iff} \quad \forall \sigma \in \mathsf{States}\colon f(\sigma) \leq g(\sigma)$$

What is the expected value of f after running C on initial state σ?

$$\mathsf{wp}[\![\{\,\mathtt{skip}\,\}\,[1/2]\,\{x := x+2\}]\!](x) \;=\; 1/2 \cdot x + 1/2 \cdot (x+2) \;=\; x+1$$

$$\mathsf{wp}[\![\{\,\mathtt{skip}\,\}\,[1/2]\,\{x := x+2\}]\!]([x=4])$$

- $\mathsf{States} = \{\sigma \mid \sigma\colon \mathsf{Vars} \to \mathbb{Q}_{\geq 0}\}$
- Complete lattice $(\mathbb{E}, \sqsubseteq)$ of *expectations*:

$$\mathbb{E} = \{f \mid f\colon \mathsf{States} \to \mathbb{R}^{\infty}_{\geq 0}\} \qquad \text{where} \qquad f \sqsubseteq g \quad \text{iff} \quad \forall \sigma \in \mathsf{States}\colon f(\sigma) \leq g(\sigma)$$

What is the expected value of f after running C on initial state σ?

$$\mathsf{wp}[\![\{\,\mathtt{skip}\,\}\,[1/2]\,\{\,x := x+2\,\}]\!](x) \;=\; 1/2 \cdot x + 1/2 \cdot (x+2) \;=\; x+1$$

$$\mathsf{wp}[\![\{\,\mathtt{skip}\,\}\,[1/2]\,\{\,x := x+2\,\}]\!]([x=4]) \;=\; 1/2 \cdot [x=4] + 1/2 \cdot [x=2]$$

C	$\mathsf{wp}\,[\![C]\!]\,(f)$
`skip`	f
$x := a$	$f\,[x/a]$
$C_1\,;\,C_2$	$\mathsf{wp}[\![C_1]\!]\,(\mathsf{wp}[\![C_2]\!]\,(f))$
$\{\,C_1\,\}\,[p]\,\{\,C_2\,\}$	$p \cdot \mathsf{wp}[\![C_1]\!]\,(f) + (1-p) \cdot \mathsf{wp}[\![C_2]\!]\,(f)$
`if` $(\varphi)\,\{\,C_1\,\}$ `else` $\{\,C_2\,\}$	$[\varphi] \cdot \mathsf{wp}[\![C_1]\!]\,(f) + [\neg\varphi] \cdot \mathsf{wp}[\![C_2]\!]\,(f)$
`while` $(\varphi)\,\{\,body\,\}$	$\mathsf{lfp}\ \lambda h.\ [\neg\varphi] \cdot f + [\varphi] \cdot \mathsf{wp}[\![body]\!]\,(h)$

The lfp is taken w.r.t. $\sqsubseteq$ on expectations.

C	$\mathsf{wp}\,[\![C]\!]\,(f)$
`skip`	f
$x := a$	$f\,[x/a]$
$C_1\,;\,C_2$	$\mathsf{wp}[\![C_1]\!]\,(\mathsf{wp}[\![C_2]\!]\,(f))$
$\{\,C_1\,\}\,[p]\,\{\,C_2\,\}$	$p \cdot \mathsf{wp}[\![C_1]\!]\,(f) + (1-p) \cdot \mathsf{wp}[\![C_2]\!]\,(f)$
`if` $(\varphi)\,\{\,C_1\,\}$ `else` $\{\,C_2\,\}$	$[\varphi] \cdot \mathsf{wp}[\![C_1]\!]\,(f) + [\neg\varphi] \cdot \mathsf{wp}[\![C_2]\!]\,(f)$
`while` $(\varphi)\,\{\,body\,\}$	$\mathsf{lfp}\ \lambda h.\ [\neg\varphi] \cdot f + [\varphi] \cdot \mathsf{wp}[\![body]\!]\,(h)$

The lfp is taken w.r.t. $\sqsubseteq$ on expectations.

Verification of loops is tackled via **quantitative loop invariants.**

$\mathsf{wp}[\![\texttt{while}\,(\varphi)\,\{\,\mathit{body}\,\}]\!]\,(f) = \mathsf{lfp}\;\Phi_f$, where

$$\Phi_f \colon \mathbb{E} \to \mathbb{E}, \qquad \Phi_f(h) = [\neg\varphi] \cdot f + [\varphi] \cdot \mathsf{wp}[\![\mathit{body}]\!]\,(h)$$

$\mathsf{wp}[\![\texttt{while}\,(\varphi)\,\{\,\mathit{body}\,\}]\!]\,(f) = \mathsf{lfp}\ \Phi_f$, where

$$\Phi_f \colon \mathbb{E} \to \mathbb{E}, \qquad \Phi_f(h) = [\neg\varphi] \cdot f + [\varphi] \cdot \mathsf{wp}[\![\mathit{body}]\!]\,(h)$$

Theorem (Park Induction for Loops [Kaminski '19])

Let $I \in \mathbb{E}$. We have

$$\underbrace{\Phi_f(I) \sqsubseteq I}_{I \text{ is inductive loop invariant}} \qquad \textit{implies} \qquad \mathsf{wp}[\![\texttt{while}\,(\varphi)\,\{\,\mathit{body}\,\}]\!]\,(f) \sqsubseteq I\,.$$

$\mathsf{wp}[\![\texttt{while}\,(\varphi)\,\{\,\mathit{body}\,\}]\!]\,(f) = \mathsf{lfp}\;\Phi_f$, where

$$\Phi_f \colon \mathbb{E} \to \mathbb{E}, \qquad \Phi_f(h) = [\neg\varphi] \cdot f + [\varphi] \cdot \mathsf{wp}[\![\mathit{body}]\!]\,(h)$$

Theorem (Park Induction for Loops [Kaminski '19])

Let $I \in \mathbb{E}$. We have

$$\underbrace{\Phi_f(I) \sqsubseteq I}_{I \text{ is inductive loop invariant}} \qquad \textit{implies} \qquad \mathsf{wp}[\![\texttt{while}\,(\varphi)\,\{\,\mathit{body}\,\}]\!]\,(f) \sqsubseteq I\,.$$

Loop invariants reduce reasoning about loops to **reasoning about one loop iteration.**

$$\mathsf{wp}[\![\texttt{while}\,(y = 1)\,\{\,y := 0\ [1/2]\ x := x + 1\,\}]\!]\,(x)$$

$$\mathsf{wp}[\![\texttt{while}\,(y=1)\,\{\,y:=0\ [1/2]\ x:=x+1\,\}]\!]\,(x)\ \sqsubseteq\ [y\neq 1]\cdot x+[y=1]\cdot(x+1)$$

$$\mathsf{wp}[\![\texttt{while}\,(y=1)\,\{\,y:=0\ [1/2]\ x:=x+1\,\}]\!]\,(x) \;\sqsubseteq\; \underbrace{[y\neq 1]\cdot x+[y=1]\cdot(x+1)}_{\text{candidate invariant } I}$$

$$\mathsf{wp}[\![\texttt{while}\,(y=1)\,\{\,y:=0\ [1/2]\ x:=x+1\,\}]\!]\,(x) \;\sqsubseteq\; \underbrace{[y\neq 1]\cdot x+[y=1]\cdot(x+1)}_{\text{candidate invariant } I}$$

$$\Phi_x(I) \;=\; [y\neq 1]\cdot x+[y=1]\cdot \mathsf{wp}[\![body]\!]\,(I)$$

$$\mathsf{wp}[\![\texttt{while}\,(y=1)\,\{\,y:=0\ [1/2]\ x:=x+1\,\}]\!]\,(x) \sqsubseteq \underbrace{[y\neq 1]\cdot x+[y=1]\cdot(x+1)}_{\text{candidate invariant } I}$$

$$\texttt{while}\,(y=1)\,\{$$

$$\{\,y:=0\,\}\ [1/2]\ \{\,x:=x+1\,\}$$

$$\}$$

$$\Phi_x(I) \;=\; [y\neq 1]\cdot x+[y=1]\cdot \mathsf{wp}[\![\mathit{body}]\!]\,(I)$$

$$\mathsf{wp}[\![\texttt{while}\,(y=1)\,\{\,y:=0\ [1/2]\ x:=x+1\,\}]\!]\,(x) \sqsubseteq \underbrace{[y\neq 1]\cdot x+[y=1]\cdot(x+1)}_{\text{candidate invariant } I}$$

$$\texttt{while}\,(y=1)\,\{$$

$$\{\,y:=0\,\}\ [1/2]\ \{\,x:=x+1\,\}$$

$$\}$$

$$\Phi_x(I) \;=\; [y\neq 1]\cdot x+[y=1]\cdot \mathsf{wp}[\![\mathit{body}]\!]\,(I)$$

$$\mathsf{wp}[\![\texttt{while}\,(y=1)\,\{\,y:=0\ [1/2]\ x:=x+1\,\}]\!]\,(x) \sqsubseteq \underbrace{[y\neq 1]\cdot x+[y=1]\cdot(x+1)}_{\text{candidate invariant } I}$$

$$\texttt{while}\,(y=1)\,\{$$

$$\{\,y:=0\,\}\ [1/2]\ \{\,x:=x+1\,\}$$

$$\}$$

$$\Phi_x(I) \;=\; [y\neq 1]\cdot x+[y=1]\cdot \mathsf{wp}[\![body]\!]\,(I)$$

$$\mathsf{wp}[\![\texttt{while}\,(y=1)\,\{\,y:=0\ [1/2]\ x:=x+1\,\}]\!]\,(x) \;\sqsubseteq\; \underbrace{[y\neq 1]\cdot x+[y=1]\cdot(x+1)}_{\text{candidate invariant } I}$$

$$\texttt{while}\,(y=1)\,\{$$

$$\{\,y:=0\,\}\ [1/2]\ \{\,x:=x+1\,\}$$

$$/\!/\!/\ [y\neq 1]\cdot x+[y=1]\cdot(x+1)$$

$$\}$$

$$\Phi_x(I) \;=\; [y\neq 1]\cdot x+[y=1]\cdot \mathsf{wp}[\![\mathit{body}]\!]\,(I)$$

$$\mathsf{wp}[\![\texttt{while}\,(y=1)\,\{\,y:=0\ [1/2]\ x:=x+1\,\}]\!]\,(x) \sqsubseteq \underbrace{[y\neq 1]\cdot x+[y=1]\cdot(x+1)}_{\text{candidate invariant } I}$$

$$\texttt{while}\,(y=1)\,\{$$

$$\quad /\!/\!/\ 1/2\cdot(x+[y\neq 1]\cdot(x+1)+[y=1]\cdot(x+2))$$

$$\quad \{\,y:=0\,\}\ [1/2]\ \{\,x:=x+1\,\}$$

$$\quad /\!/\!/\ [y\neq 1]\cdot x+[y=1]\cdot(x+1)$$

$$\}$$

$$\Phi_x(I) \;=\; [y\neq 1]\cdot x+[y=1]\cdot \mathsf{wp}[\![\mathit{body}]\!]\,(I)$$

$$\mathsf{wp}[\![\texttt{while}\,(y=1)\,\{\,y:=0\ [1/2]\ x:=x+1\,\}]\!]\,(x) \sqsubseteq \underbrace{[y\neq 1]\cdot x+[y=1]\cdot(x+1)}_{\text{candidate invariant } I}$$

$$\texttt{while}\,(y=1)\,\{$$

$$\quad /\!/\!/\ 1/2\cdot(x+[y\neq 1]\cdot(x+1)+[y=1]\cdot(x+2))$$

$$\quad \{\,y:=0\,\}\ [1/2]\ \{\,x:=x+1\,\}$$

$$\quad /\!/\!/\ [y\neq 1]\cdot x+[y=1]\cdot(x+1)$$

$$\}$$

$$\begin{aligned}\Phi_x(I) &= [y\neq 1]\cdot x+[y=1]\cdot \mathsf{wp}[\![body]\!]\,(I)\\ &= [y\neq 1]\cdot x+[y=1]\cdot(1/2\cdot(x+[y\neq 1]\cdot(x+1)+[y=1]\cdot(x+2)))\end{aligned}$$

$$\mathsf{wp}[\![\texttt{while}\,(y=1)\,\{\,y:=0\ [1/2]\ x:=x+1\,\}]\!]\,(x) \sqsubseteq \underbrace{[y\neq 1]\cdot x+[y=1]\cdot(x+1)}_{\text{candidate invariant } I}$$

$$\texttt{while}\,(y=1)\,\{$$

$$\quad /\!/\!/\ 1/2\cdot(x+[y\neq 1]\cdot(x+1)+[y=1]\cdot(x+2))$$

$$\quad \{\,y:=0\,\}\ [1/2]\ \{\,x:=x+1\,\}$$

$$\quad /\!/\!/\ [y\neq 1]\cdot x+[y=1]\cdot(x+1)$$

$$\}$$

$$\begin{aligned}\Phi_x(I) &= [y\neq 1]\cdot x+[y=1]\cdot \mathsf{wp}[\![body]\!]\,(I)\\ &= [y\neq 1]\cdot x+[y=1]\cdot(1/2\cdot(x+[y\neq 1]\cdot(x+1)+[y=1]\cdot(x+2)))\sqsubseteq I\ \checkmark\end{aligned}$$

$$\mathsf{wp}[\![\texttt{while}\,(y=1)\,\{\,y:=0\ [1/2]\ x:=x+1\,\}]\!]\,(x) \sqsubseteq \underbrace{[y\neq 1]\cdot x+[y=1]\cdot(x+1)}_{\text{candidate invariant } I}$$

```
while (y = 1) {
    /// 1/2 · (x + [y ≠ 1] · (x + 1) + [y = 1] · (x + 2))
    { y := 0 } [1/2] { x := x + 1 }
    /// [y ≠ 1] · x + [y = 1] · (x + 1)
}
```

1. Compute $\Phi_x(I)$

$$\begin{aligned}\Phi_x(I) &= [y\neq 1]\cdot x+[y=1]\cdot \mathsf{wp}[\![body]\!]\,(I)\\ &= [y\neq 1]\cdot x+[y=1]\cdot(1/2\cdot(x+[y\neq 1]\cdot(x+1)+[y=1]\cdot(x+2)))\sqsubseteq I\ \checkmark\end{aligned}$$

$$\mathsf{wp}[\![\texttt{while}\,(y=1)\,\{\,y:=0\ [1/2]\ x:=x+1\,\}]\!]\,(x) \sqsubseteq \underbrace{[y\neq 1]\cdot x+[y=1]\cdot(x+1)}_{\text{candidate invariant } I}$$

$\texttt{while}\,(y=1)\,\{$

$\quad /\!/\!/\ 1/2\cdot(x+[y\neq 1]\cdot(x+1)+[y=1]\cdot(x+2))$

$\quad \{\,y:=0\,\}\ [1/2]\ \{\,x:=x+1\,\}$

$\quad /\!/\!/\ [y\neq 1]\cdot x+[y=1]\cdot(x+1)$

$\}$

1. Compute $\Phi_x(I)$
2. Check $\Phi_x(I)\sqsubseteq I$ via SMT

$$\begin{aligned}\Phi_x(I) &= [y\neq 1]\cdot x+[y=1]\cdot \mathsf{wp}[\![body]\!]\,(I)\\ &= [y\neq 1]\cdot x+[y=1]\cdot(1/2\cdot(x+[y\neq 1]\cdot(x+1)+[y=1]\cdot(x+2)))\sqsubseteq I\ \checkmark\end{aligned}$$

C	$\mathsf{wp}\,[\![C]\!]\,(f)$
`skip`	f
$x := a$	$f\,[x/a]$
$C_1\,;\,C_2$	$\mathsf{wp}[\![C_1]\!]\,(\mathsf{wp}[\![C_2]\!]\,(f))$
$\{\,C_1\,\}\,[p]\,\{\,C_2\,\}$	$p \cdot \mathsf{wp}[\![C_1]\!]\,(f) + (1-p) \cdot \mathsf{wp}[\![C_2]\!]\,(f)$
`if` $(\varphi)\,\{\,C_1\,\}$ `else` $\{\,C_2\,\}$	$[\varphi] \cdot \mathsf{wp}[\![C_1]\!]\,(f) + [\neg\varphi] \cdot \mathsf{wp}[\![C_2]\!]\,(f)$
`while` $(\varphi)\,\{\,body\,\}$	$\mathsf{lfp}\,\lambda h.\,[\neg\varphi] \cdot f + [\varphi] \cdot \mathsf{wp}[\![body]\!]\,(h)$

where $f\colon \mathsf{States} \to \mathbb{R}^{\infty}_{\geq 0}$

C	$\mathsf{wp}\,[\![C]\!]\,(f)$
`skip`	f
$x := a$	$f\,[x/a]$
$C_1\,;\,C_2$	$\mathsf{wp}[\![C_1]\!]\,(\mathsf{wp}[\![C_2]\!]\,(f))$
$\{\,C_1\,\}\,[p]\,\{\,C_2\,\}$	$p \cdot \mathsf{wp}[\![C_1]\!]\,(f) + (1-p) \cdot \mathsf{wp}[\![C_2]\!]\,(f)$
`if` $(\varphi)\,\{\,C_1\,\}$ `else` $\{\,C_2\,\}$	$[\varphi] \cdot \mathsf{wp}[\![C_1]\!]\,(f) + [\neg\varphi] \cdot \mathsf{wp}[\![C_2]\!]\,(f)$
`while` $(\varphi)\,\{\,body\,\}$	$\mathsf{lfp}\,\lambda h.\,[\neg\varphi] \cdot f + [\varphi] \cdot \mathsf{wp}[\![body]\!]\,(h)$

where $f\colon \mathsf{States} \to \mathbb{R}^{\infty}_{\geq 0}$

What is a suitable syntax for expectations?

Relatively Complete Verification [POPL '21]

Goal: Provide a quantitative assertion language for deductive verifiers.

Ordinary Programs

Assertions are $F\colon \text{States} \to \{0,1\}$

Ordinary Programs

Assertions are $F\colon \text{States} \to \{0,1\}$

$F \in \text{FO-Arithmetic}$

Ordinary Programs

Assertions are $F\colon \text{States} \to \{0,1\}$

$F \in \text{FO-Arithmetic}$

$x = 2 \wedge y = x + 1$

Ordinary Programs

Assertions are $F\colon \text{States} \to \{0,1\}$

$F \in \text{FO-Arithmetic}$

$x = 2 \land y = x + 1$

$\exists i\colon i \cdot 2 = z$

Ordinary Programs

Assertions are $F\colon \text{States} \to \{0,1\}$

$$F \in \mathsf{FO\text{-}Arithmetic}$$
$$\text{implies}^1$$
$$\mathsf{wp}[\![C]\!]\,(F) \in \mathsf{FO\text{-}Arithmetic}$$

$$x = 2 \land y = x + 1$$
$$\exists i\colon i \cdot 2 = z$$

[1] [Clarke '76, Cook '78, Olderog '80]

Ordinary Programs

Assertions are $F\colon \text{States} \to \{0,1\}$

Probabilistic Programs

"Assertions" are $f\colon \text{States} \to \mathbb{R}^{\infty}_{\geq 0}$

$F \in \text{FO-Arithmetic}$

implies[1]

$\text{wp}[\![C]\!](F) \in \text{FO-Arithmetic}$

$x = 2 \land y = x + 1$

$\exists i\colon i \cdot 2 = z$

[1] [Clarke '76, Cook '78, Olderog '80]

Ordinary Programs

Assertions are $F\colon \text{States} \to \{0,1\}$

$F \in \text{FO-Arithmetic}$

implies[1]

$\text{wp}[\![C]\!](F) \in \text{FO-Arithmetic}$

$x = 2 \land y = x + 1$

$\exists i\colon i \cdot 2 = z$

Probabilistic Programs

"Assertions" are $f\colon \text{States} \to \mathbb{R}^{\infty}_{\geq 0}$

$f \in \text{SomeSyntax}$

[1] [Clarke '76, Cook '78, Olderog '80]

Ordinary Programs

Assertions are $F\colon \text{States} \to \{0,1\}$

$F \in \text{FO-Arithmetic}$

implies[1]

$\text{wp}[\![C]\!](F) \in \text{FO-Arithmetic}$

Probabilistic Programs

"Assertions" are $f\colon \text{States} \to \mathbb{R}^{\infty}_{\geq 0}$

$f \in \text{SomeSyntax}$

implies

$\text{wp}[\![C]\!](f) \in \text{SomeSyntax}$

$x = 2 \land y = x + 1$

$\exists i\colon i \cdot 2 = z$

[1] [Clarke '76, Cook '78, Olderog '80]

Ordinary Programs

Assertions are $F\colon \text{States} \to \{0,1\}$

$F \in \text{FO-Arithmetic}$

implies[1]

$\text{wp}[\![C]\!](F) \in \text{FO-Arithmetic}$

Probabilistic Programs

"Assertions" are $f\colon \text{States} \to \mathbb{R}^{\infty}_{\geq 0}$

$f \in \text{SomeSyntax}$

implies

$\text{wp}[\![C]\!](f) \in \text{SomeSyntax}$

$x = 2 \land y = x + 1$

$\exists i\colon i \cdot 2 = z$

We provide a simple, yet expressive, language of expectations.

[1] [Clarke '76, Cook '78, Olderog '80]

[Flajolet, Pelletier, Soria '09]

$x := \mathsf{geometric}(1/4); y := \mathsf{geometric}(1/4);$

$\{t := x + y + 1\}\ [5/9]\ \{t := x + y\};$

$r := 1;$

`repeat 3 times {`

draw sequence of $2 \cdot t$ coin flips;

`if` (#heads $\neq$ #tails) $\{r := 0\};$

}

/// $[r = 1]$

$/\!/\!/\ 1/\pi$ [Flajolet, Pelletier, Soria '09]

$x := \mathsf{geometric}(1/4); y := \mathsf{geometric}(1/4);$

$\{t := x + y + 1\}\ [5/9]\ \{t := x + y\};$

$r := 1;$

`repeat 3 times {`

draw sequence of $2 \cdot t$ coin flips;

`if` (#heads $\neq$ #tails) $\{r := 0\};$

}

$/\!/\!/\ [r = 1]$

$/\!/\!/\ 1/\pi$ [Flajolet, Pelletier, Soria '09]

$x := \mathsf{geometric}(1/4); y := \mathsf{geometric}(1/4);$

$\{t := x + y + 1\}\ [5/9]\ \{t := x + y\};$

$r := 1;$

`repeat 3 times {`

draw sequence of $2 \cdot t$ coin flips;

`if` (#heads $\neq$ #tails) $\{r := 0\};$

}

$/\!/\!/\ [r = 1]$

Simple postexpectations can yield irrational, non-algebraic,... preexpectations

Arithmetic Expressions:

$$a \quad \longrightarrow \quad q \in \mathbb{Q}_{\geq 0} \mid \underbrace{x \in \mathsf{Vars}}_{\mathbb{Q}_{\geq 0}\text{-valued}} \mid a + a \mid a \cdot a \mid a \dot{-} a$$

Arithmetic Expressions:

$$a \quad \longrightarrow \quad q \in \mathbb{Q}_{\geq 0} \mid \underbrace{x \in \mathsf{Vars}}_{\mathbb{Q}_{\geq 0}\text{-valued}} \mid a + a \mid a \cdot a \mid a \dot{-} a$$

Boolean Expressions:

$$\varphi \quad \longrightarrow \quad a < a \mid \varphi \wedge \varphi \mid \neg\varphi$$

Arithmetic Expressions:

$$a \quad \longrightarrow \quad q \in \mathbb{Q}_{\geq 0} \mid \underbrace{x \in \mathsf{Vars}}_{\mathbb{Q}_{\geq 0}\text{-valued}} \mid a + a \mid a \cdot a \mid a \dot{-} a$$

Boolean Expressions:

$$\varphi \quad \longrightarrow \quad a < a \mid \varphi \wedge \varphi \mid \neg\varphi$$

Our Language Exp of Expectations:

$$f \quad \longrightarrow \quad a$$

Arithmetic Expressions:

$$a \quad \longrightarrow \quad q \in \mathbb{Q}_{\geq 0} \mid \underbrace{x \in \mathsf{Vars}}_{\mathbb{Q}_{\geq 0}\text{-valued}} \mid a + a \mid a \cdot a \mid a \dot{-} a$$

Boolean Expressions:

$$\varphi \quad \longrightarrow \quad a < a \mid \varphi \wedge \varphi \mid \neg\varphi$$

Our Language Exp **of Expectations:**

$$f \quad \longrightarrow \quad a$$

x

Arithmetic Expressions:

$$a \quad \longrightarrow \quad q \in \mathbb{Q}_{\geq 0} \mid \underbrace{x \in \mathsf{Vars}}_{\mathbb{Q}_{\geq 0}\text{-valued}} \mid a + a \mid a \cdot a \mid a \dot{-} a$$

Boolean Expressions:

$$\varphi \quad \longrightarrow \quad a < a \mid \varphi \wedge \varphi \mid \neg\varphi$$

Our Language Exp **of Expectations:**

$$f \quad \longrightarrow \quad a \mid [\varphi]$$

x

Arithmetic Expressions:

$$a \quad \longrightarrow \quad q \in \mathbb{Q}_{\geq 0} \mid \underbrace{x \in \mathsf{Vars}}_{\mathbb{Q}_{\geq 0}\text{-valued}} \mid a + a \mid a \cdot a \mid a \dot{-} a$$

Boolean Expressions:

$$\varphi \quad \longrightarrow \quad a < a \mid \varphi \wedge \varphi \mid \neg\varphi$$

Our Language Exp **of Expectations:**

$$f \quad \longrightarrow \quad a \mid [\varphi] \mid f \cdot f$$

x

Arithmetic Expressions:

$$a \quad \longrightarrow \quad q \in \mathbb{Q}_{\geq 0} \mid \underbrace{x \in \mathsf{Vars}}_{\mathbb{Q}_{\geq 0}\text{-valued}} \mid a + a \mid a \cdot a \mid a \dot{-} a$$

Boolean Expressions:

$$\varphi \quad \longrightarrow \quad a < a \mid \varphi \wedge \varphi \mid \neg\varphi$$

Our Language Exp **of Expectations:**

$$f \quad \longrightarrow \quad a \mid [\varphi] \mid f \cdot f$$

$$[x \cdot x < y] \cdot x$$

Arithmetic Expressions:

$$a \quad \longrightarrow \quad q \in \mathbb{Q}_{\geq 0} \mid \underbrace{x \in \mathsf{Vars}}_{\mathbb{Q}_{\geq 0}\text{-valued}} \mid a + a \mid a \cdot a \mid a \dot{-} a$$

Boolean Expressions:

$$\varphi \quad \longrightarrow \quad a < a \mid \varphi \wedge \varphi \mid \neg\varphi$$

Our Language Exp **of Expectations:**

$$f \quad \longrightarrow \quad a \mid [\varphi] \mid f \cdot f \mid f + f$$

$$[x \cdot x < y] \cdot x$$

Arithmetic Expressions:

$$a \quad \longrightarrow \quad q \in \mathbb{Q}_{\geq 0} \mid \underbrace{x \in \mathsf{Vars}}_{\mathbb{Q}_{\geq 0}\text{-valued}} \mid a + a \mid a \cdot a \mid a \dot{-} a$$

Boolean Expressions:

$$\varphi \quad \longrightarrow \quad a < a \mid \varphi \wedge \varphi \mid \neg\varphi$$

Our Language Exp of Expectations:

$$f \quad \longrightarrow \quad a \mid [\varphi] \mid f \cdot f \mid f + f \mid \mathfrak{D}x\colon f \mid \mathfrak{C}x\colon f$$

$$[x \cdot x < y] \cdot x$$

Arithmetic Expressions:

$$a \quad \longrightarrow \quad q \in \mathbb{Q}_{\geq 0} \mid \underbrace{x \in \mathsf{Vars}}_{\mathbb{Q}_{\geq 0}\text{-valued}} \mid a + a \mid a \cdot a \mid a \dot{-} a$$

Boolean Expressions:

$$\varphi \quad \longrightarrow \quad a < a \mid \varphi \wedge \varphi \mid \neg\varphi$$

Our Language Exp of Expectations:

$$f \quad \longrightarrow \quad a \mid [\varphi] \mid f \cdot f \mid f + f \mid \mathfrak{D}x\colon f \mid \mathfrak{C}x\colon f$$

$$\mathfrak{D}x\colon\ [x \cdot x < y] \cdot x$$

Arithmetic Expressions:

$$a \quad \longrightarrow \quad q \in \mathbb{Q}_{\geq 0} \mid \underbrace{x \in \mathsf{Vars}}_{\mathbb{Q}_{\geq 0}\text{-valued}} \mid a + a \mid a \cdot a \mid a \dot{-} a$$

Boolean Expressions:

$$\varphi \quad \longrightarrow \quad a < a \mid \varphi \wedge \varphi \mid \neg\varphi$$

Our Language Exp of Expectations:

$$f \quad \longrightarrow \quad a \mid [\varphi] \mid f \cdot f \mid f + f \mid \mathfrak{2}x\colon f \mid \mathfrak{l}x\colon f$$

$$(\mathfrak{2}x\colon\ [x \cdot x < y] \cdot x)(\sigma)$$

Arithmetic Expressions:

$$a \quad \longrightarrow \quad q \in \mathbb{Q}_{\geq 0} \mid \underbrace{x \in \mathsf{Vars}}_{\mathbb{Q}_{\geq 0}\text{-valued}} \mid a + a \mid a \cdot a \mid a \dot{-} a$$

Boolean Expressions:

$$\varphi \quad \longrightarrow \quad a < a \mid \varphi \wedge \varphi \mid \neg\varphi$$

Our Language Exp **of Expectations:**

$$f \quad \longrightarrow \quad a \mid [\varphi] \mid f \cdot f \mid f + f \mid \mathfrak{Q}x\colon f \mid \mathfrak{L}x\colon f$$

$$(\mathfrak{Q}x\colon\ [x \cdot x < y] \cdot x)(\sigma) \quad = \quad \sup\{q \in \mathbb{Q}_{\geq 0} \mid q \cdot q < \sigma(y)\}$$

Arithmetic Expressions:

$$a \quad \longrightarrow \quad q \in \mathbb{Q}_{\geq 0} \mid \underbrace{x \in \mathsf{Vars}}_{\mathbb{Q}_{\geq 0}\text{-valued}} \mid a + a \mid a \cdot a \mid a \dot{-} a$$

Boolean Expressions:

$$\varphi \quad \longrightarrow \quad a < a \mid \varphi \wedge \varphi \mid \neg\varphi$$

Our Language Exp **of Expectations:**

$$f \quad \longrightarrow \quad a \mid [\varphi] \mid f \cdot f \mid f + f \mid \mathfrak{Q}x\colon f \mid \mathfrak{L}x\colon f$$

$$(\mathfrak{Q}x\colon\ [x \cdot x < y] \cdot x)(\sigma) \quad = \quad \sup\{q \in \mathbb{Q}_{\geq 0} \mid q \cdot q < \sigma(y)\} \quad = \quad \sqrt{\sigma(y)}$$

Our Language Exp **of Expectations:**

$$f \quad \longrightarrow \quad a \mid [\varphi] \mid f \cdot f \mid f + f \mid \mathfrak{Q} x \colon f \mid \mathfrak{L} x \colon f$$

Theorem (POPL '21)

The language Exp *is expressive, i.e.,*

$$\forall \text{ programs } C \colon \ \forall f \in \mathsf{Exp} \colon \ \mathsf{wp}[\![C]\!](f) \in \mathsf{Exp}\,.$$

Our Language Exp **of Expectations:**

$$f \quad \longrightarrow \quad a \mid [\varphi] \mid f \cdot f \mid f + f \mid \mathfrak{D}x\colon f \mid \mathfrak{C}x\colon f$$

Theorem (POPL '21)

The language Exp *is expressive, i.e.,*

$$\forall \text{ programs } C\colon\ \forall f \in \mathsf{Exp}\colon\ \mathsf{wp}[\![C]\!](f) \in \mathsf{Exp}\,.$$

Proof.

By induction on C. Most intricate part: loops.

Combines Gödel's β-function [Gödel 1931] with Dedekind cuts [Bertrand 1849]. □

Our Language Exp of Expectations:

$$f \quad \longrightarrow \quad a \mid [\varphi] \mid f \cdot f \mid f + f \mid \mathfrak{S} x \colon f \mid \mathfrak{L} x \colon f$$

Theorem (POPL '21)

The language Exp *is expressive, i.e.,*

$$\forall \text{ programs } C \colon \ \forall f \in \mathsf{Exp} \colon \ \mathsf{wp}[\![C]\!](f) \in \mathsf{Exp}\,.$$

Proof.

By induction on C. Most intricate part: loops.

Combines Gödel's β-function [Gödel 1931] with Dedekind cuts [Bertrand 1849]. □

Exp is the basis of our modern automated deductive verifier CAESAR [OOPSLA '23].

Automatic Verification of Loops

Automatic Verification of Loops

Given: $C = \texttt{while}\,(\varphi)\,\{\mathit{body}\}$ and expectations f, g

Automatic Verification of Loops

Given: $C = \texttt{while}\,(\varphi)\,\{\,body\,\}$ and expectations f, g

Goal: **Automatically** prove $\mathsf{wp}[\![C]\!]\,(f) \sqsubseteq g$

Automatic Verification of Loops

Given: $C = \texttt{while}\,(\varphi)\,\{\,body\,\}$ and expectations f, g

Goal: **Automatically** prove $\mathsf{wp}[\![C]\!]\,(f) \sqsubseteq g$

1. **Latticed k-Induction [CAV '21]**
2. **Inductive Synthesis of Inductive Invariants [TACAS '23]**
3. **Property Directed Reachability [CAV '20]**

Automatic Verification of Loops - The Challenge

$\mathsf{wp}[\![\texttt{while}\,(\varphi)\,\{\,body\,\}]\!]\,(f) = \mathsf{lfp}\;\Phi_f$, where

$$\Phi_f \colon \mathbb{E} \to \mathbb{E}, \qquad \Phi_f(h) = [\neg\varphi] \cdot f + [\varphi] \cdot \mathsf{wp}[\![body]\!]\,(h)$$

Theorem (Park Induction for Loops [Kaminski '19])

Let $I \in \mathbb{E}$. We have

$$\underbrace{\Phi_f(I) \sqsubseteq I}_{I \text{ is inductive loop invariant}} \qquad \textit{implies} \qquad \mathsf{wp}[\![\texttt{while}\,(\varphi)\,\{\,body\,\}]\!]\,(f) \sqsubseteq I\,.$$

Automatic Verification of Loops - The Challenge

$\mathsf{wp}[\![\texttt{while}\,(\varphi)\,\{\,body\,\}]\!]\,(f) = \mathsf{lfp}\;\Phi_f$, where

$$\Phi_f \colon \mathbb{E} \to \mathbb{E}, \qquad \Phi_f(h) = [\neg\varphi] \cdot f + [\varphi] \cdot \mathsf{wp}[\![body]\!]\,(h)$$

Theorem (Park Induction for Loops [Kaminski '19])

Let $I \in \mathbb{E}$. We have

$$\underbrace{\Phi_f(I) \sqsubseteq I}_{I \text{ is inductive loop invariant}} \qquad \textit{implies} \qquad \mathsf{wp}[\![\texttt{while}\,(\varphi)\,\{\,body\,\}]\!]\,(f) \sqsubseteq I\,.$$

$\mathsf{wp}[\![\texttt{while}\,(\varphi)\,\{\,body\,\}]\!]\,(f) = \mathsf{lfp}\;\Phi_f$, where

$$\Phi_f \colon \mathbb{E} \to \mathbb{E}, \qquad \Phi_f(h) = [\neg\varphi] \cdot f + [\varphi] \cdot \mathsf{wp}[\![body]\!]\,(h)$$

Theorem (Park Induction for Loops [Kaminski '19])

Let $I \in \mathbb{E}$. We have

$$\underbrace{\Phi_f(I) \sqsubseteq I}_{I \text{ is inductive loop invariant}} \qquad \textit{implies} \qquad \mathsf{wp}[\![\texttt{while}\,(\varphi)\,\{\,body\,\}]\!]\,(f) \sqsubseteq I\,.$$

Problem: The converse direction does not hold in general!

Latticed k-Induction [CAV '21]

Improves upon classic Park induction.

Latticed k-Induction [CAV '21]

Improves upon classic Park induction.

Latticed k-Induction [CAV '21]

Improves upon classic Park induction.

- Symbolic model checking technique for transition systems [Sheeran et al. '00]

Latticed k-Induction [CAV '21]

Improves upon classic Park induction.

- Symbolic model checking technique for transition systems [Sheeran et al. '00]
- Prominent hard- and software verification technique. [Krishnan et al. '18]:

Latticed k-Induction [CAV '21]

Improves upon classic Park induction.

- Symbolic model checking technique for transition systems [Sheeran et al. '00]
- Prominent hard- and software verification technique. [Krishnan et al. '18]:

"The simplicity of applying k-induction made it
the go-to technique for SMT-based infinite-state model checking."

Latticed k-Induction [CAV '21]

Improves upon classic Park induction.

- Symbolic model checking technique for transition systems [Sheeran et al. '00]
- Prominent hard- and software verification technique. [Krishnan et al. '18]:

"The simplicity of applying k-induction made it
the go-to technique for SMT-based infinite-state model checking."

We generalize k-induction to make it applicable to probabilistic programs,

Latticed k-Induction [CAV '21]

Improves upon classic Park induction.

- Symbolic model checking technique for transition systems [Sheeran et al. '00]
- Prominent hard- and software verification technique. [Krishnan et al. '18]:

"The simplicity of applying k-induction made it
the go-to technique for SMT-based infinite-state model checking."

We generalize k-induction to make it applicable to probabilistic programs, probabilistic pushdown systems, stochastic games, ...

Latticed k-Induction [CAV '21]

Given: Complete lattice $(D, \sqsubseteq)$, monotonic $\Phi\colon D \to D$, and $I \in D$

Latticed k-Induction [CAV '21]

Given: Complete lattice $(D, \sqsubseteq)$, monotonic $\Phi\colon D \to D$, and $I \in D$

Goal: Prove $\mathsf{lfp}\, \Phi \sqsubseteq I$

Latticed k-Induction [CAV '21]

Given: Complete lattice $(D, \sqsubseteq)$, monotonic $\Phi\colon D \to D$, and $I \in D$

Goal: Prove $\mathsf{lfp}\ \Phi \sqsubseteq I$

Theorem (Latticed k-Induction [CAV '21])

- 1-*induction:* $\Phi(I) \sqsubseteq I$ *implies* $\mathsf{lfp}\ \Phi \sqsubseteq I$

Latticed k-Induction [CAV '21]

Given: Complete lattice $(D, \sqsubseteq)$, monotonic $\Phi\colon D \to D$, and $I \in D$

Goal: Prove $\mathsf{lfp}\ \Phi \sqsubseteq I$

Theorem (Latticed k-Induction [CAV '21])

- 1-*induction:* $\Phi(I) \sqsubseteq I$ *implies* $\mathsf{lfp}\ \Phi \sqsubseteq I$
- 2-*induction:* $\Phi(\Phi(I) \sqcap I) \sqsubseteq I$ *implies* $\mathsf{lfp}\ \Phi \sqsubseteq I$

Latticed k-Induction [CAV '21]

Given: Complete lattice $(D, \sqsubseteq)$, monotonic $\Phi\colon D \to D$, and $I \in D$

Goal: Prove $\mathsf{lfp}\ \Phi \sqsubseteq I$

Theorem (Latticed k-Induction [CAV '21])

- 1-*induction:* $\Phi(I) \sqsubseteq I$ *implies* $\mathsf{lfp}\ \Phi \sqsubseteq I$
- 2-*induction:* $\Phi(\Phi(I) \sqcap I) \sqsubseteq I$ *implies* $\mathsf{lfp}\ \Phi \sqsubseteq I$
- 3-*induction:* $\Phi(\Phi(\Phi(I) \sqcap I) \sqcap I) \sqsubseteq I$ *implies* $\mathsf{lfp}\ \Phi \sqsubseteq I$

Latticed k-Induction [CAV '21]

Given: Complete lattice $(D, \sqsubseteq)$, monotonic $\Phi\colon D \to D$, and $I \in D$

Goal: Prove $\mathsf{lfp}\ \Phi \sqsubseteq I$

Theorem (Latticed k-Induction [CAV '21])

- 1-*induction:* $\Phi(I) \sqsubseteq I$ *implies* $\mathsf{lfp}\ \Phi \sqsubseteq I$
- 2-*induction:* $\Phi(\Phi(I) \sqcap I) \sqsubseteq I$ *implies* $\mathsf{lfp}\ \Phi \sqsubseteq I$
- 3-*induction:* $\Phi(\Phi(\Phi(I) \sqcap I) \sqcap I) \sqsubseteq I$ *implies* $\mathsf{lfp}\ \Phi \sqsubseteq I$
- $\ldots$

Latticed k-Induction [CAV '21]

Given: Complete lattice $(D, \sqsubseteq)$, monotonic $\Phi\colon D \to D$, and $I \in D$

Goal: Prove $\mathsf{lfp}\ \Phi \sqsubseteq I$

Theorem (Latticed k-Induction [CAV '21])

- 1-*induction:* $\Phi(I) \sqsubseteq I$ *implies* $\mathsf{lfp}\ \Phi \sqsubseteq I$
- 2-*induction:* $\Phi(\Phi(I) \sqcap I) \sqsubseteq I$ *implies* $\mathsf{lfp}\ \Phi \sqsubseteq I$
- 3-*induction:* $\Phi(\Phi(\Phi(I) \sqcap I) \sqcap I) \sqsubseteq I$ *implies* $\mathsf{lfp}\ \Phi \sqsubseteq I$
- …

Leveraged by:

- [Winkler & Katoen '23]: Probabilistic Pushdown Automata
- [Yang et al. '24]: Static Analysis of Probabilistic Programs

Latticed k-Induction [CAV '21]

Given: $(\mathbb{E}, \sqsubseteq)$ and $\Phi_f \colon \mathbb{E} \to \mathbb{E}$ (obtained from $C = \texttt{while}\,(\varphi)\,\{body\}$ and f), $I \in \mathbb{E}$

Goal: Prove $\mathsf{wp}[\![C]\!]\,(f) = \mathsf{lfp}\ \Phi_f \sqsubseteq I$

Latticed k-Induction [CAV '21]

Given: $(\mathbb{E}, \sqsubseteq)$ and $\Phi_f : \mathbb{E} \to \mathbb{E}$ (obtained from $C = \texttt{while}\,(\varphi)\,\{body\}$ and f), $I \in \mathbb{E}$

Goal: Prove $\mathsf{wp}[\![C]\!](f) = \mathsf{lfp}\ \Phi_f \sqsubseteq I$

Corollary (k-Induction for Probabilistic Programs [CAV '21])

- 1-*induction:* $\Phi_f(I) \sqsubseteq I$ *implies* $\mathsf{wp}[\![C]\!](f) \sqsubseteq I$
- 2-*induction:* $\Phi_f(\Phi_f(I) \sqcap I) \sqsubseteq I$ *implies* $\mathsf{wp}[\![C]\!](f) \sqsubseteq I$
- 3-*induction:* $\Phi_f(\Phi_f(\Phi_f(I) \sqcap I) \sqcap I) \sqsubseteq I$ *implies* $\mathsf{wp}[\![C]\!](f) \sqsubseteq I$
- $\ldots$

where $g \sqcap h = \lambda\sigma.\, \min\{g(\sigma), h(\sigma)\}$

Latticed k-Induction [CAV '21]

Given: $(\mathbb{E}, \sqsubseteq)$ and $\Phi_f : \mathbb{E} \to \mathbb{E}$ (obtained from $C = \texttt{while}\,(\varphi)\,\{body\}$ and f), $I \in \mathbb{E}$

Goal: Prove $\mathsf{wp}[\![C]\!](f) = \mathsf{lfp}\ \Phi_f \sqsubseteq I$

Corollary (k-Induction for Probabilistic Programs [CAV '21])

- 1-*induction:* $\Phi_f(I) \sqsubseteq I$ *implies* $\mathsf{wp}[\![C]\!](f) \sqsubseteq I$
- 2-*induction:* $\Phi_f(\Phi_f(I) \sqcap I) \sqsubseteq I$ *implies* $\mathsf{wp}[\![C]\!](f) \sqsubseteq I$
- 3-*induction:* $\Phi_f(\Phi_f(\Phi_f(I) \sqcap I) \sqcap I) \sqsubseteq I$ *implies* $\mathsf{wp}[\![C]\!](f) \sqsubseteq I$
- $\ldots$

where $g \sqcap h = \lambda\sigma.\, \min\{g(\sigma), h(\sigma)\}$

k-inductivity is decidable for linear loop and piecewise linear expectations.

The Bounded Retransmission Protocol [Helmink et al. '93, D'Argenio et al. '97]

Goal: Send file of N packets via lossy channel.

Transmitting a packet fails with probability 0.1.

Allow for at most M retransmissions per packet.

$$\texttt{while}\,(\mathit{sent} < N \land \mathit{failed} \leq M)\,\{$$
$$\{\underbrace{\mathit{failed} := \mathit{failed} + 1;\, \mathit{totalFailed} := \mathit{totalFailed} + 1}_{\text{failed transmission}}\}\,[0.1]\,\{\underbrace{\mathit{failed} := 0;\, \mathit{sent} := \mathit{sent} + 1}_{\text{successful transmission}}\}\}$$

The Bounded Retransmission Protocol [Helmink et al. '93, D'Argenio et al. '97]

Goal: Send file of N packets via lossy channel.

Transmitting a packet fails with probability 0.1.

Allow for at most M retransmissions per packet.

$$\texttt{while}\,(\mathit{sent} < N \land \mathit{failed} \leq M)\,\{$$
$$\{\underbrace{\mathit{failed} := \mathit{failed} + 1;\, \mathit{totalFailed} := \mathit{totalFailed} + 1}_{\text{failed transmission}}\}\,[0.1]\,\{\underbrace{\mathit{failed} := 0;\, \mathit{sent} := \mathit{sent} + 1}_{\text{successful transmission}}\}\}$$

$$\mathsf{wp}[\![C]\!]\,(\mathit{totalFailed}) \;\sqsubseteq\; [N \leq 4] \cdot (\mathit{totalFailed} + 1/2) + [N > 4] \cdot \infty$$

The Bounded Retransmission Protocol [Helmink et al. '93, D'Argenio et al. '97]

Goal: Send file of N packets via lossy channel.

Transmitting a packet fails with probability 0.1.

Allow for at most M retransmissions per packet.

$$\texttt{while}\,(\mathit{sent} < N \land \mathit{failed} \leq M)\,\{$$
$$\{\underbrace{\mathit{failed} := \mathit{failed} + 1;\, \mathit{totalFailed} := \mathit{totalFailed} + 1}_{\text{failed transmission}}\}\,[0.1]\,\{\underbrace{\mathit{failed} := 0;\, \mathit{sent} := \mathit{sent} + 1}_{\text{successful transmission}}\}\}$$

Our tool proves

$$\mathsf{wp}[\![C]\!]\,(\mathit{totalFailed}) \;\sqsubseteq\; [N \leq 4] \cdot (\mathit{totalFailed} + 1/2) + [N > 4] \cdot \infty$$

fully automatically by means of 4-induction.

Problem: k-induction is incomplete

Problem: k-induction is incomplete

The property

$$\mathsf{wp}[\![\texttt{while}\,(y = 1)\,\{\,y := 0\ [1/2]\ x := x + 1\,\}]\!](x) \sqsubseteq \mathbf{2x + 1}$$

is not k-inductive for any k.

Problem: k-induction is incomplete

The property

$$\mathsf{wp}[\![\texttt{while}\,(y = 1)\,\{\,y := 0\ [1/2]\ x := x + 1\,\}]\!](x) \sqsubseteq \mathbf{2x + 1}$$

is not k-inductive for any k.

Solution: Invariant synthesis

Inductive Synthesis of Inductive Invariants [TACAS ’23]

Inductive Synthesis of Inductive Invariants [TACAS '23]

Given: Linear loop $\texttt{while}\,(\varphi)\,\{body\}$, piecewise linear f, g

Inductive Synthesis of Inductive Invariants [TACAS '23]

Given: Linear loop $\texttt{while}\,(\varphi)\,\{body\}$, piecewise linear f, g

Goal: **Automatically** prove $\mathsf{wp}[\![C]\!]\,(f) \sqsubseteq g$

Inductive Synthesis of Inductive Invariants [TACAS '23]

Given: Linear loop $\texttt{while}\,(\varphi)\,\{body\}$, piecewise linear f, g

Goal: **Automatically** prove $\mathsf{wp}[\![C]\!](f) \sqsubseteq g$

Approach: Compute piecewise linear I with $\underbrace{\Phi_f(I) \sqsubseteq I}_{\text{inductivity}}$ and $\underbrace{I \sqsubseteq g}_{\text{safety}}$

Inductive Synthesis of Inductive Invariants [TACAS '23]

Given: Linear loop $\texttt{while}\,(\varphi)\,\{body\}$, piecewise linear f, g

Goal: **Automatically** prove $\mathsf{wp}[\![C]\!](f) \sqsubseteq g$

Approach: Compute piecewise linear I with $\underbrace{\underbrace{\Phi_f(I) \sqsubseteq I}_{\text{inductivity}} \text{ and } \underbrace{I \sqsubseteq g}_{\text{safety}}}_{\text{implies } \mathsf{wp}[\![C]\!](f) \sqsubseteq g}$

Inductive Synthesis of Inductive Invariants [TACAS '23]

Given: Linear loop $\texttt{while}\,(\varphi)\,\{body\}$, piecewise linear f, g

Goal: **Automatically** prove $\mathsf{wp}[\![C]\!](f) \sqsubseteq g$

Approach: Compute piecewise linear I with $\underbrace{\Phi_f(I) \sqsubseteq I}_{\text{inductivity}}$ and $\underbrace{I \sqsubseteq g}_{\text{safety}}$

Given: Linear loop $\texttt{while}(\varphi)\{body\}$, piecewise linear f, g

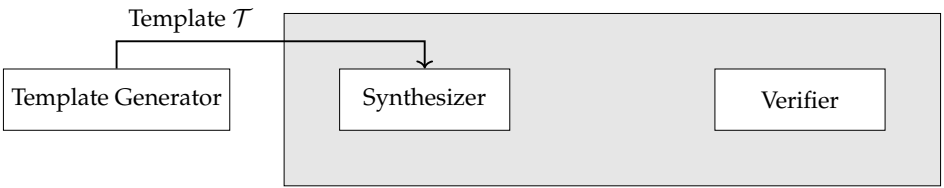
Goal: **Automatically** prove $\mathsf{wp}[\![C]\!](f) \sqsubseteq g$

Approach: Compute piecewise linear I with $\underbrace{\Phi_f(I) \sqsubseteq I}_{\text{inductivity}}$ and $\underbrace{I \sqsubseteq g}_{\text{safety}}$

Given: Linear loop $\texttt{while}(\varphi)\{body\}$, piecewise linear f, g

Goal: **Automatically** prove $\mathsf{wp}[\![C]\!](f) \sqsubseteq g$

Approach: Compute piecewise linear I with $\underbrace{\Phi_f(I) \sqsubseteq I}_{\text{inductivity}}$ and $\underbrace{I \sqsubseteq g}_{\text{safety}}$

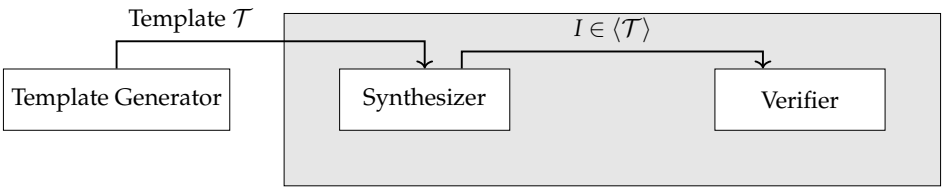


$$\mathcal{T} \;=\; [y = 0 \wedge x < 6] \cdot (\alpha \cdot x + \beta \cdot y + \gamma) + [y = 1]$$

Given: Linear loop `while` $(\varphi)\,\{body\}$, piecewise linear f, g

Goal: **Automatically** prove $\mathsf{wp}[\![\mathrm{C}]\!](f) \sqsubseteq g$

Approach: Compute piecewise linear I with $\underbrace{\Phi_f(I) \sqsubseteq I}_{\text{inductivity}}$ and $\underbrace{I \sqsubseteq g}_{\text{safety}}$



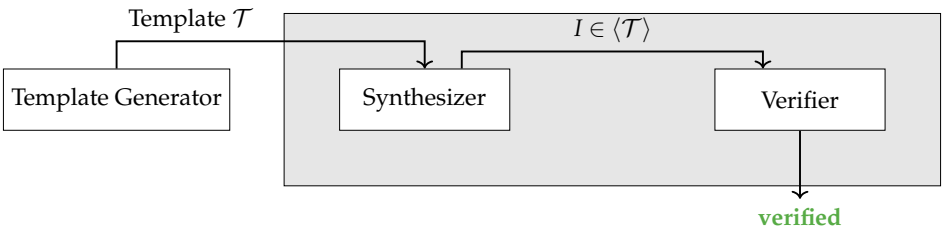
$$\mathcal{T} = [y = 0 \wedge x < 6] \cdot (\alpha \cdot x + \beta \cdot y + \gamma) + [y = 1]$$

$$\langle \mathcal{T} \rangle \ni I = [y = 0 \wedge x < 6] \cdot (0 \cdot x - 9/110 \cdot y + 28/55) + [y = 1]$$

Given: Linear loop `while` $(\varphi)\{body\}$, piecewise linear f, g

Goal: **Automatically** prove $\mathsf{wp}[\![\mathrm{C}]\!](f) \sqsubseteq g$

Approach: Compute piecewise linear I with $\underbrace{\Phi_f(I) \sqsubseteq I}_{\text{inductivity}}$ and $\underbrace{I \sqsubseteq g}_{\text{safety}}$



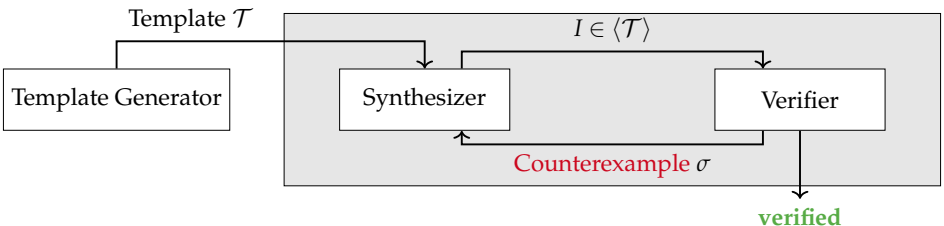
$$\mathcal{T} = [y = 0 \wedge x < 6] \cdot (\alpha \cdot x + \beta \cdot y + \gamma) + [y = 1]$$

$$\langle \mathcal{T} \rangle \ni I = [y = 0 \wedge x < 6] \cdot (0 \cdot x - 9/110 \cdot y + 28/55) + [y = 1]$$

Given: Linear loop `while` $(\varphi)\{body\}$, piecewise linear f, g

Goal: **Automatically** prove $\mathsf{wp}[\![C]\!](f) \sqsubseteq g$

Approach: Compute piecewise linear I with $\underbrace{\Phi_f(I) \sqsubseteq I}_{\text{inductivity}}$ and $\underbrace{I \sqsubseteq g}_{\text{safety}}$

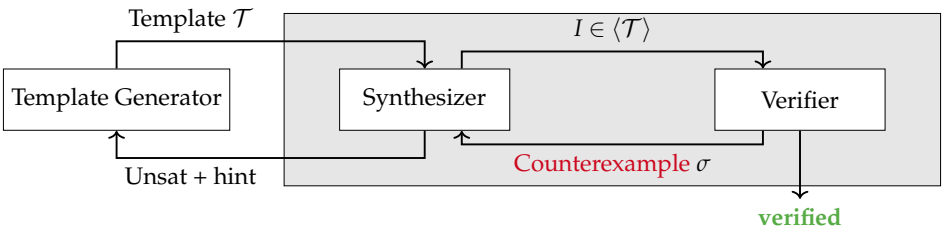


$$\mathcal{T} \;=\; [y = 0 \wedge x < 6] \cdot (\alpha \cdot x + \beta \cdot y + \gamma) + [y = 1]$$

Given: Linear loop $\texttt{while}(\varphi)\{body\}$, piecewise linear f, g

Goal: **Automatically** prove $\mathsf{wp}[\![C]\!](f) \sqsubseteq g$

Approach: Compute piecewise linear I with $\underbrace{\Phi_f(I) \sqsubseteq I}_{\text{inductivity}}$ and $\underbrace{I \sqsubseteq g}_{\text{safety}}$

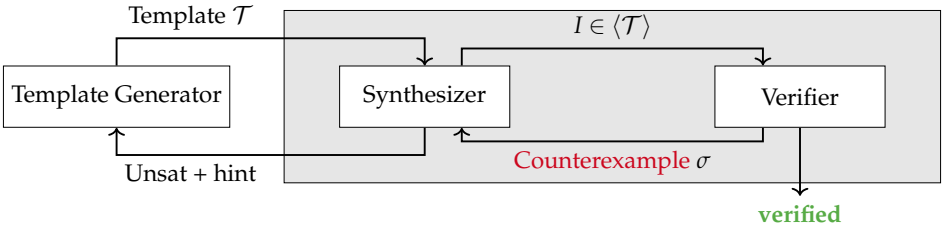


$$\mathcal{T} \;=\; [y = 0 \wedge x < 6] \cdot (\alpha \cdot x + \beta \cdot y + \gamma) + [y = 1]$$

Given: Linear loop $\texttt{while}(\varphi)\{body\}$, piecewise linear f, g

Goal: **Automatically** prove $\mathsf{wp}[\![C]\!](f) \sqsubseteq g$

Approach: Compute piecewise linear I with $\underbrace{\Phi_f(I) \sqsubseteq I}_{\text{inductivity}}$ and $\underbrace{I \sqsubseteq g}_{\text{safety}}$



$$\mathcal{T} = [y = 0 \land x < 3] \cdot (\alpha_1 \cdot x + \beta_1 \cdot y + \gamma_1) + [y = 0 \land x \geq 3 \land x \leq 6] \cdot (\alpha_2 \cdot x + \beta_2 \cdot y + \gamma_2)$$

The Bounded Retransmission Protocol [Helmink et al. '93, D'Argenio et al. '97]

Goal: Send file of $N = 8 \cdot 10^9$ packets via lossy channel.

Transmitting a packet fails with probability 0.01.

Allow for at most $M = 10$ retransmissions per packet.

$$sent := 0; failed := 0;$$

$$\texttt{while}\,(sent < N \wedge failed \leq M)\,\texttt{do}$$

$$\{\underbrace{failed := failed + 1}_{\text{failed transmission}}\}[0.01]\{\underbrace{failed := 0; sent := sent + 1}_{\text{successful transmission}}\}$$

$$\texttt{end}$$

The Bounded Retransmission Protocol [Helmink et al. '93, D'Argenio et al. '97]

Goal: Send file of $N = \mathbf{8 \cdot 10^9}$ packets via lossy channel.

Transmitting a packet fails with probability 0.01.

Allow for at most $M = \mathbf{10}$ retransmissions per packet.

$$\mathit{sent} := 0; \mathit{failed} := 0;$$

$$\texttt{while}\,(\mathit{sent} < N \wedge \mathit{failed} \leq M)\,\texttt{do}$$

$$\{\underbrace{\mathit{failed} := \mathit{failed} + 1}_{\text{failed transmission}}\}[0.01]\{\underbrace{\mathit{failed} := 0; \mathit{sent} := \mathit{sent} + 1}_{\text{successful transmission}}\}$$

$$\texttt{end}$$

Our tool proves $\mathsf{wp}[\![C]\!]\,([\mathit{failed} > M]) \sqsubseteq 0.001$ in **11** seconds.

The Bounded Retransmission Protocol [Helmink et al. '93, D'Argenio et al. '97]

Goal: Send file of $N = \mathbf{8 \cdot 10^9}$ packets via lossy channel.

Transmitting a packet fails with probability 0.01.

Allow for at most $M = \mathbf{10}$ retransmissions per packet.

$$sent := 0; failed := 0;$$

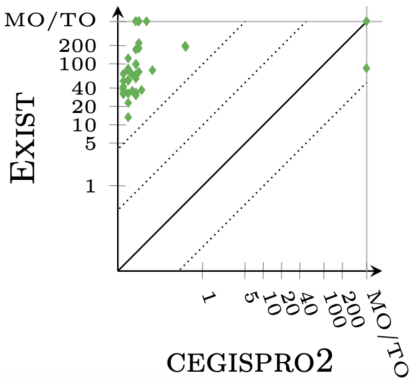
$$\texttt{while}\,(sent < N \wedge failed \leq M)\,\texttt{do}$$

$$\{\underbrace{failed := failed + 1}_{\text{failed transmission}}\}[0.01]\{\underbrace{failed := 0; sent := sent + 1}_{\text{successful transmission}}\}$$

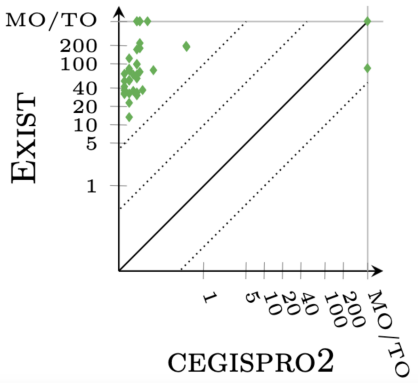
$$\texttt{end}$$

Automatic deductive program verification complements probabilistic model checking.

Mostly **infinite-state programs** from [Bao et al. '22] (TO=5min, MO=8gb)



Mostly **infinite-state programs** from [Bao et al. '22] (TO=5min, MO=8gb)



Our approach is competitive with the state-of-the-art invariant synthesis techniques.

Property Directed Reachability [CAV '20]

Property Directed Reachability [CAV '20]

- PDR: State-of-the-art **hardware model checking** algorithm [Bradley '11, Een et al. '11]

Property Directed Reachability [CAV '20]

- PDR: State-of-the-art **hardware model checking** algorithm [Bradley '11, Een et al. '11]
- Highly tailored to transition system verification

Property Directed Reachability [CAV '20]

- PDR: State-of-the-art **hardware model checking** algorithm [Bradley '11, Een et al. '11]
- Highly tailored to transition system verification

We have generalized PDR to the probabilistic setting

Property Directed Reachability [CAV '20]

- PDR: State-of-the-art **hardware model checking** algorithm [Bradley '11, Een et al. '11]
- Highly tailored to transition system verification

We have generalized PDR to the probabilistic setting

- **PDR-style quantitative invariant synthesis technique**

Property Directed Reachability [CAV '20]

- PDR: State-of-the-art **hardware model checking** algorithm [Bradley '11, Een et al. '11]
- Highly tailored to transition system verification

We have generalized PDR to the probabilistic setting

- **PDR-style quantitative invariant synthesis technique**
- **Fully backward compatible**

Property Directed Reachability [CAV '20]

- PDR: State-of-the-art **hardware model checking** algorithm [Bradley '11, Een et al. '11]
- Highly tailored to transition system verification

We have generalized PDR to the probabilistic setting

- **PDR-style quantitative invariant synthesis technique**
- **Fully backward compatible**
- **Tool support. Performance not yet competitive.**

Property Directed Reachability [CAV '20]

- PDR: State-of-the-art **hardware model checking** algorithm [Bradley '11, Een et al. '11]
- Highly tailored to transition system verification

We have generalized PDR to the probabilistic setting

- **PDR-style quantitative invariant synthesis technique**
- **Fully backward compatible**
- **Tool support. Performance not yet competitive.**
- **Recent advances by [Kori et al. '22,'23]. Future work: Join forces!**

Scientific Contributions

1. **Computing Sampling Times [ESOP '18]**
2. **Quantitative Separation Logic [POPL '19]**
3. **Property Directed Reachability [CAV '20]**
4. **Generating Functions for PPs [LOPSTR '21]**
5. **Relatively Complete Verification [POPL '21]**
6. **Latticed k-Induction [CAV '21]**
7. **Entailment Checking in QSL [ESOP '22]**
8. **Weighted Programming [OOPSLA '22]**
9. **Amortized Expected Runtimes [POPL '23]**
10. **Invariant Synthesis [TACAS '23]**
11. **Verification Infrastructure for PPs [OOPSLA '23]**
12. **Programmatic Strategy Synthesis [POPL '24]**
13. **J.-P.: MDP. FP. PP [Festschrift JPK]**
14. **Verifying Continuous Programs [under sub.]**
15. **Quant. Quantifier Elimination [under sub.]**

Conclusion

- **Relatively Complete Verification [POPL '21]**
- **Latticed k-Induction [CAV '21]**
- **Inductive Synthesis of Inductive Invariants [TACAS '23]**
- **Property Directed Reachability [CAV '20]**

Conclusion

- **Relatively Complete Verification [POPL '21]**
- **Latticed k-Induction [CAV '21]**
- **Inductive Synthesis of Inductive Invariants [TACAS '23]**
- **Property Directed Reachability [CAV '20]**

Future work:

- **Nondeterministic** probabilistic programs (Planning/AI: [POPL '24])

Conclusion

- **Relatively Complete Verification [POPL '21]**
- **Latticed k-Induction [CAV '21]**
- **Inductive Synthesis of Inductive Invariants [TACAS '23]**
- **Property Directed Reachability [CAV '20]**

Future work:

- **Nondeterministic** probabilistic programs (Planning/AI: [POPL '24])
- **Continuous** sampling

Conclusion

- **Relatively Complete Verification [POPL '21]**
- **Latticed k-Induction [CAV '21]**
- **Inductive Synthesis of Inductive Invariants [TACAS '23]**
- **Property Directed Reachability [CAV '20]**

Future work:

- **Nondeterministic** probabilistic programs (Planning/AI: [POPL '24])
- **Continuous** sampling

Thank you!

Conclusion

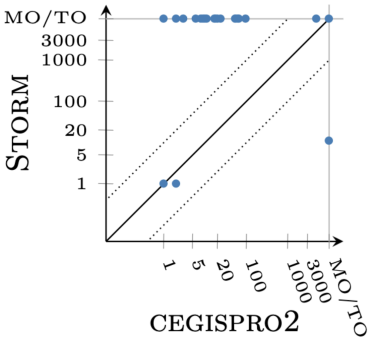
- **Relatively Complete Verification [POPL '21]**
- **Latticed k-Induction [CAV '21]**
- **Inductive Synthesis of Inductive Invariants [TACAS '23]**
- **Property Directed Reachability [CAV '20]**

Future work:

- **Nondeterministic** probabilistic programs (Planning/AI: [POPL '24])
- **Continuous** sampling

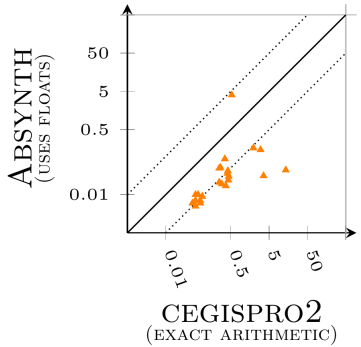
Thank you!
Arigato gozaimasu!

Finite-State Programs (Runtimes in seconds. TO=2h, MO=8gb)



Invariant synthesis can advance the state-of-the-art in probabilistic model checking.

Upper-bounding expected runtimes Programs from [Ngo et al. '18] (TO=5min, MO=8gb)



***cegispro2* can compute ERTs of infinite-state programs.**
Bounds are of similar quality. No floating-point issues.

Backup

		STORM			CEGISPRO2										
					INDUCT.-GUIDED				STATIC				DYNAMIC			
Prog	$\|S\|$	SP	DD	BEST	$\|S'\|$	$\|I\|$	t_s%	t	$\|S'\|$	$\|I\|$	t_s%	t	$\|S'\|$	$\|I\|$	t_s%	t
boundedrwmultistep	$1 \cdot 10^5$	MO	TO	3	33	10	40	3	–	–	–	TO	–	–	–	TO
		MO	TO	10	55	16	36	10	–	–	–	TO	–	–	–	TO
		MO	TO	–	–	–	–	TO	–	–	–	TO	–	–	–	TO
brp	$1 \cdot 10^{10}$	MO	TO	11	56	23	40	11	70	10	35	18	–	–	–	TO
		MO	TO	54	138	42	63	253	125	17	27	54	–	–	–	TO
		MO	TO	56	104	41	54	111	122	17	30	56	–	–	–	TO
brpfinitefamily	$16 \cdot 10^{13}$	TO	TO	8	53	7	72	10	67	7	44	8	54	9	52	29
		TO	TO	17	64	13	74	17	215	19	66	373	–	–	–	TO
		TO	TO	18	68	12	68	18	231	19	80	731	–	–	–	TO