Probabilistic Program Verification via Inductive Synthesis of Inductive Invariants

Kevin Batz, Mingshuai Chen, Sebastian Junges,
Benjamin Lucien Kaminski, Joost-Pieter Katoen, Christoph Matheja

April 27, 2023, TACAS 2023

One-Slide Summary

Deductive Program Verification Techniques + CEGIS

enables the *fully automatic* verification of probabilistic integer loops

One-Slide Summary

Deductive Program Verification Techniques + CEGIS

enables the *fully automatic* verification of probabilistic integer loops

- Can scale to huge state spaces on interesting finite-state programs

One-Slide Summary

Deductive Program Verification Techniques + CEGIS

enables the *fully automatic* verification of probabilistic integer loops

- Can scale to huge state spaces on interesting finite-state programs
- Infinite-state programs: verify, e.g., infinite families of Markov chains

One-Slide Summary

Deductive Program Verification Techniques + CEGIS

enables the *fully automatic* verification of probabilistic integer loops

- Can scale to huge state spaces on interesting finite-state programs
- Infinite-state programs: verify, e.g., infinite families of Markov chains
- Competitive with modern invariant synthesis tools

One-Slide Summary

Deductive Program Verification Techniques + CEGIS

enables the *fully automatic* verification of probabilistic integer loops

- Can scale to huge state spaces on interesting finite-state programs
- Infinite-state programs: verify, e.g., infinite families of Markov chains
- Competitive with modern invariant synthesis tools
- Adaptable for the computation of expected runtime bounds

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 single packet fails with some probability, say 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 single packet fails with some probability, say 0.01.

Allow for at most $F = 10$ retransmissions per packet; otherwise fail.

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 single packet fails with some probability, say 0.01.

Allow for at most $F = 10$ retransmissions per packet; otherwise fail.

$$sent := 0;\ fail := 0;$$

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 single packet fails with some probability, say 0.01.

Allow for at most $F = 10$ retransmissions per packet; otherwise fail.

$$sent := 0;\ fail := 0;$$

$$\texttt{while}\,(\, sent < N \wedge fail < F \,)\,\{$$

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 single packet fails with some probability, say 0.01.

Allow for at most $F = 10$ retransmissions per packet; otherwise fail.

$$sent := 0;\ fail := 0;$$

$$\texttt{while}\,(\,sent < N \wedge fail < F\,)\,\{$$

$$\{\underbrace{fail := fail + 1}_{\text{failed transmission}}\}[0.01]$$

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 single packet fails with some probability, say 0.01.

Allow for at most $F = 10$ retransmissions per packet; otherwise fail.

$$sent := 0;\ fail := 0;$$

$$\texttt{while}\,(\,sent < N \wedge fail < F\,)\,\{$$

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

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 single packet fails with some probability, say 0.01.

Allow for at most $F = 10$ retransmissions per packet; otherwise fail.

$$sent := 0;\ fail := 0;$$

$$\texttt{while}\,(\,sent < N \wedge fail < F\,)\,\{$$

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

We verify $Pr(\Diamond fail = F) \leq 0.001$ in 11 seconds.

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

Goal: Send file of $N \leq 10^6$ packets via lossy channel.

Transmitting a single packet fails with some probability, say 0.01.

Allow for at most $F \geq 5$ retransmissions per packet; otherwise fail.

$$sent := 0;\ fail := 0;$$

$$\texttt{while}\,(\,sent < N \wedge fail < F\,)\,\{$$

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

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

Goal: Send file of $N \leq 10^6$ packets via lossy channel.

Transmitting a single packet fails with some probability, say 0.01.

Allow for at most $F \geq 5$ retransmissions per packet; otherwise fail.

$$sent := 0;\ fail := 0;$$

$$\texttt{while}\,(\,sent < N \wedge fail < F\,)\,\{$$

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

We verify $Pr(\Diamond fail = F) \leq 0.001$ for all instances in 10 seconds.

Race between Tortoise and Hare [Adapted from Ngo et al. 2017]

On every round, the tortoise moves 1 step.

The hare moves probabilistically between 0 and 10 steps.

Question: How long does the hare need to catch the tortoise?

Race between Tortoise and Hare [Adapted from Ngo et al. 2017]

On every round, the tortoise moves 1 step.

The hare moves probabilistically between 0 and 10 steps.

Question: How long does the hare need to catch the tortoise?

$$\texttt{while}\,(\,h \leq t\,)\,\{$$
$$\quad t := t + 1;$$
$$\quad \{\,\texttt{skip}\,\}\,[\,1/2\,]\,\{\,h := 1/11 \cdot [h+0\rangle + \ldots + 1/11 \cdot [h+10\rangle\,\}\,\}$$

Race between Tortoise and Hare [Adapted from Ngo et al. 2017]

On every round, the tortoise moves 1 step.

The hare moves probabilistically between 0 and 10 steps.

Question: How long does the hare need to catch the tortoise?

$$\texttt{while}\,(\,h \leq t\,)\,\{$$
$$\quad t := t + 1;$$
$$\quad \{\,\texttt{skip}\,\}\,[\,1/2\,]\,\{\,h := 1/11 \cdot [h+0\rangle + \ldots + 1/11 \cdot [h+10\rangle\,\}\,\}$$

We automatically compute a bound on the expected number of rounds in 1.3 seconds:

$$\frac{2}{3} \cdot (t - h) + \frac{13}{3}$$

Introduction

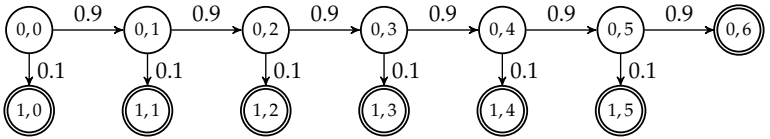
Quantitative Loop Invariants

Quantitative Loop Invariants

Given: $\texttt{while}\,(\,c = 0 \wedge x < 6\,)\,\{\,c \coloneqq 1\,[0.1]\,x \coloneqq x + 1\,\}$

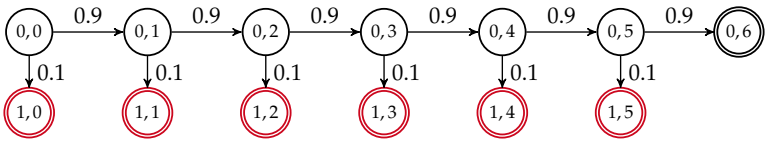
Quantitative Loop Invariants

Given: `while` $(c = 0 \wedge x < 6) \{ c \coloneqq 1 \; [0.1] \; x \coloneqq x + 1 \}$



Quantitative Loop Invariants

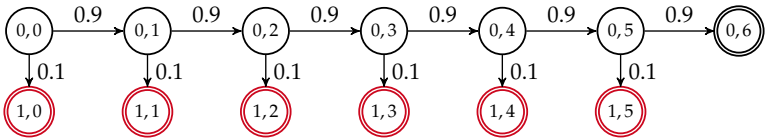
Given: `while` $(c = 0 \wedge x < 6) \{ c := 1 \; [0.1] \; x := x + 1 \}$ Target $\triangleq$ $c = 1$



Quantitative Loop Invariants

Given: `while` $(c = 0 \wedge x < 6) \{ c := 1 \ [0.1] \ x := x + 1 \}$ Target $\triangleq$ $c = 1$

Goal: Verify $Pr(0,0 \models \Diamond \text{Target}) \leq 0.9$

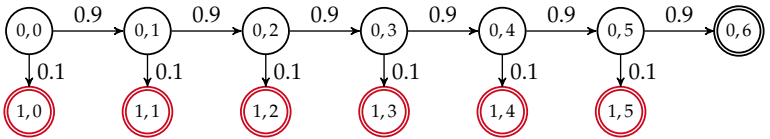


Quantitative Loop Invariants

Given: `while` $(c = 0 \wedge x < 6) \{ c := 1 \ [0.1] \ x := x + 1 \}$ Target $\triangleq$ $c = 1$

Goal: Verify $Pr(0,0 \models \Diamond \text{Target}) \leq 0.9$

Approach: Find safe inductive invariant I: States $\to \mathbb{R}$

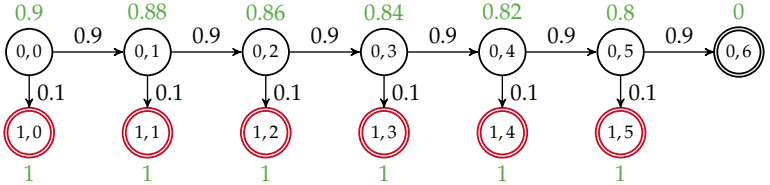


Quantitative Loop Invariants

Given: `while` $(c = 0 \wedge x < 6) \{ c := 1 \ [0.1] \ x := x + 1 \}$ Target $\triangleq$ $c = 1$

Goal: Verify $Pr(0,0 \models \Diamond \text{Target}) \leq 0.9$

Approach: Find safe inductive invariant I: States $\to \mathbb{R}$



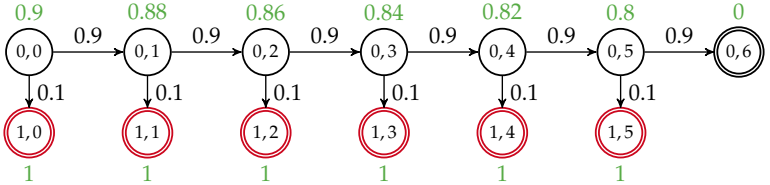
Quantitative Loop Invariants

Given: $\texttt{while}\,(\,c = 0 \wedge x < 6\,)\,\{\,c := 1\ [0.1]\ x := x + 1\,\}$ Target $\triangleq\ c = 1$

Goal: Verify $Pr(0,0 \models \Diamond \text{Target}) \leq 0.9$

Approach: Find safe inductive invariant I: States $\to \mathbb{R}$, i.e.

$$\underbrace{\forall s\colon I(s) \geq 0}_{\text{well-definedness}}$$



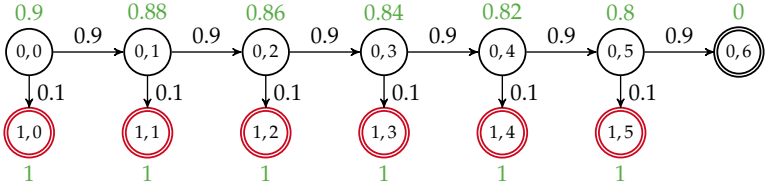
Quantitative Loop Invariants

Given: $\texttt{while}\,(\,c = 0 \wedge x < 6\,)\,\{\,c := 1\ [0.1]\ x := x + 1\,\}$ Target $\triangleq\ c = 1$

Goal: Verify $Pr(0,0 \models \Diamond \text{Target}) \leq 0.9$

Approach: Find safe inductive invariant I: States $\to \mathbb{R}$, i.e.

$$\underbrace{\forall s\colon I(s) \geq 0}_{\text{well-definedness}} \qquad \underbrace{\forall s\colon \Phi(I)(s) \leq I(s)}_{\text{inductivity}}$$



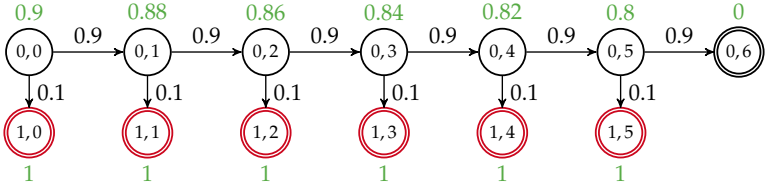
Given: `while` $(\, c = 0 \wedge x < 6 \,) \, \{\, c := 1 \; [0.1] \; x := x + 1 \,\}$ Target $\triangleq$ $c = 1$

Goal: Verify $Pr(0,0 \models \Diamond \text{Target}) \leq 0.9$

Approach: Find safe inductive invariant I: States $\to \mathbb{R}$, i.e.

$$\underbrace{\forall s\colon I(s) \geq 0}_{\text{well-definedness}} \qquad \underbrace{\forall s\colon \Phi(I)(s) \leq I(s)}_{\text{inductivity}}$$

$$\Phi(I)(0,0) \;=\; 0.9 \cdot 0.88 + 0.1 \cdot 1$$



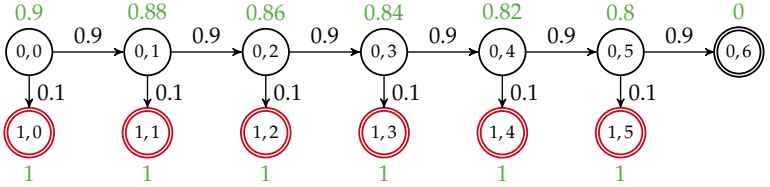
Quantitative Loop Invariants

Given: $\texttt{while}\,(\,c = 0 \wedge x < 6\,)\,\{\,c := 1\ [0.1]\ x := x + 1\,\}$ Target $\triangleq\ c = 1$

Goal: Verify $Pr(0,0 \models \Diamond \text{Target}) \ \leq\ 0.9$

Approach: Find safe inductive invariant I: States $\to \mathbb{R}$, i.e.

$$\underbrace{\forall s\colon I(s) \geq 0}_{\text{well-definedness}} \qquad \underbrace{\forall s\colon \Phi(I)(s) \leq I(s)}_{\text{inductivity}} \qquad \underbrace{I(0,0) \leq 0.9}_{\text{safety}}\ .$$



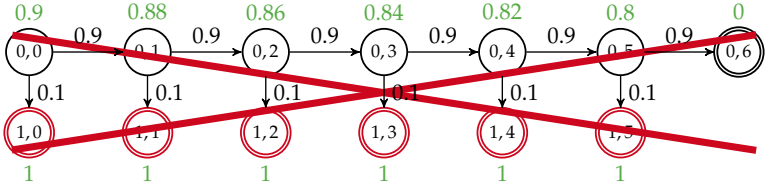
Quantitative Loop Invariants

Given: $\texttt{while}\,(\,c = 0 \wedge x < 6\,)\,\{\,c \coloneqq 1\ [0.1]\ x \coloneqq x + 1\,\}$ Target $\triangleq\ c = 1$

Goal: Verify $Pr(0,0 \models \Diamond \text{Target}) \ \leq\ 0.9$

Approach: Find safe inductive invariant I: States $\to \mathbb{R}$, i.e.

$$\underbrace{\forall s\colon I(s) \geq 0}_{\text{well-definedness}} \qquad \underbrace{\forall s\colon \Phi(I)(s) \leq I(s)}_{\text{inductivity}} \qquad \underbrace{I(0,0) \leq 0.9}_{\text{safety}}\ .$$



Quantitative Loop Invariants

Given: $\texttt{while}\,(\,c = 0 \wedge x < 6\,)\,\{\,c \coloneqq 1\ [0.1]\ x \coloneqq x + 1\,\}$ Target $\triangleq\ c = 1$

Goal: Verify $Pr(0,0 \models \Diamond \text{Target}) \ \leq\ 0.9$

Approach: Find safe inductive invariant $I\colon \text{States} \to \mathbb{R}$, i.e.

$$\underbrace{\forall s\colon I(s) \geq 0}_{\text{well-definedness}} \qquad \underbrace{\forall s\colon \Phi(I)(s) \leq I(s)}_{\text{inductivity}} \qquad \underbrace{I(0,0) \leq 0.9}_{\text{safety}}\ .$$

$$I \ = \ [c = 1] \cdot 1 + [c = 0 \wedge x < 6] \cdot (-0.02 \cdot x + 0.9)$$

Quantitative Loop Invariants

Given: $\texttt{while}\,(\,c = 0 \wedge x < 6\,)\,\{\,c := 1\ [0.1]\ x := x + 1\,\}$ Target $\triangleq\ c = 1$

Goal: Verify $Pr(0,0 \models \Diamond \text{Target}) \ \leq\ 0.9$

Approach: Find safe inductive invariant $I\colon \text{States} \to \mathbb{R}$, i.e.

$$\underbrace{\forall s\colon I(s) \geq 0}_{\text{well-definedness}} \qquad \underbrace{\forall s\colon \Phi(I)(s) \leq I(s)}_{\text{inductivity}} \qquad \underbrace{I(0,0) \leq 0.9}_{\text{safety}}\ .$$

$$I \ = \ [c = 1] \cdot 1 + [c = 0 \wedge x < 6] \cdot (-0.02 \cdot x + 0.9)$$

Finding I is hard. But:

> We can **effectively decide** whether I is a safe inductive invariant,
> **without constructing state spaces** [CAV'21].

Inductive Synthesis of Inductive Invariants

Inductive Synthesis of Inductive Invariants

Given: Linear loop `while` $(\varphi)\ \{ body \}$, LIA formula Target (e.g. $c = 1$)

Inductive Synthesis of Inductive Invariants

Given: Linear loop `while` $(\varphi)\ \{ body \}$, LIA formula Target (e.g. $c = 1$)

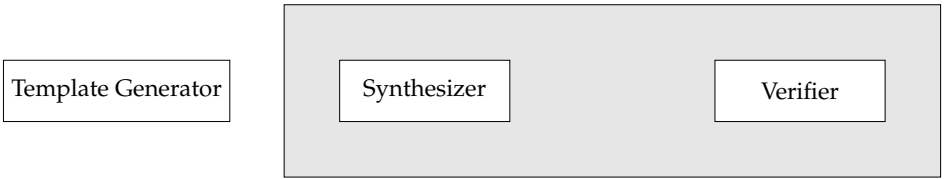
Goal: Verify $Pr(s_{\text{init}} \models \Diamond \text{Target}) \leq \lambda$

Inductive Synthesis of Inductive Invariants

Given: Linear loop `while` (φ) { *body* }, LIA formula Target (e.g. $c = 1$)

Goal: Verify $Pr(s_{\text{init}} \models \Diamond \text{Target}) \leq \lambda$

Approach: Compute safe inductive invariant $I : \text{States} \to \mathbb{R}$ via CEGIS:

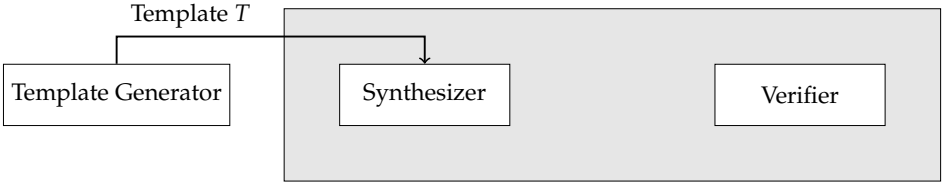


Inductive Synthesis of Inductive Invariants

Given: Linear loop `while` (φ) { *body* }, LIA formula Target (e.g. $c = 1$)

Goal: Verify $Pr(s_{\text{init}} \models \Diamond \text{Target}) \leq \lambda$

Approach: Compute safe inductive invariant $I : \text{States} \to \mathbb{R}$ via CEGIS:

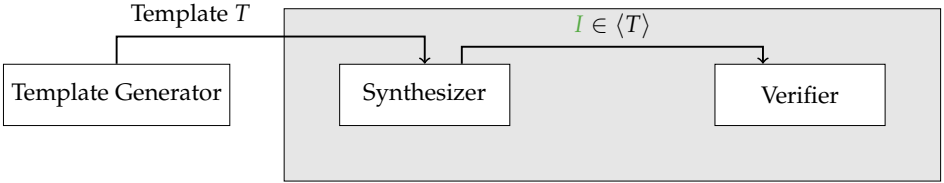


$$T \;=\; [c = 0 \wedge x < 6] \cdot (\alpha \cdot c + \beta \cdot x + \gamma) + [c = 1]$$

Given: Linear loop `while` $(\varphi)\ \{\ body\ \}$, LIA formula Target (e.g. $c = 1$)

Goal: Verify $Pr(s_{\text{init}} \models \Diamond \text{Target}) \leq \lambda$

Approach: Compute safe inductive invariant $I : \text{States} \to \mathbb{R}$ via CEGIS:



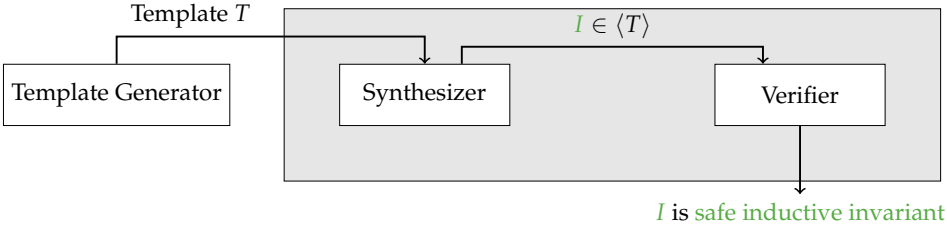
$$T = [c = 0 \land x < 6] \cdot (\alpha \cdot c + \beta \cdot x + \gamma) + [c = 1]$$

$$\langle T \rangle \ni I = [c = 0 \land x < 6] \cdot (0 \cdot c - 9/110 \cdot x + 28/55) + [c = 1]$$

Given: Linear loop `while` $(\varphi)\ \{\ body\ \}$, LIA formula Target (e.g. $c = 1$)

Goal: Verify $Pr(s_{\text{init}} \models \Diamond \text{Target}) \leq \lambda$

Approach: Compute safe inductive invariant I : States $\to \mathbb{R}$ via CEGIS:



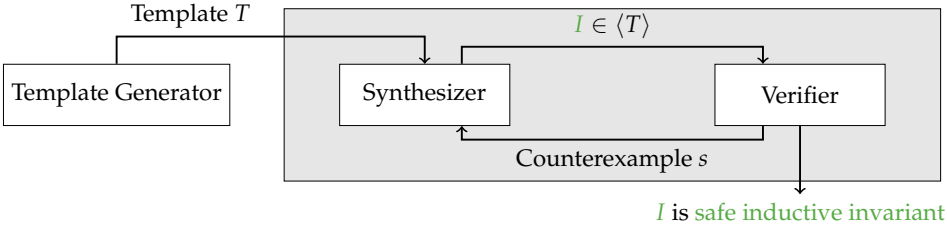
$$T = [c = 0 \land x < 6] \cdot (\alpha \cdot c + \beta \cdot x + \gamma) + [c = 1]$$

$$\langle T \rangle \ni I = [c = 0 \land x < 6] \cdot (0 \cdot c - 9/110 \cdot x + 28/55) + [c = 1]$$

Given: Linear loop `while` $(\varphi)\ \{\ body\ \}$, LIA formula Target (e.g. $c = 1$)

Goal: Verify $Pr(s_{\text{init}} \models \Diamond \text{Target}) \leq \lambda$

Approach: Compute safe inductive invariant $I : \text{States} \to \mathbb{R}$ via CEGIS:



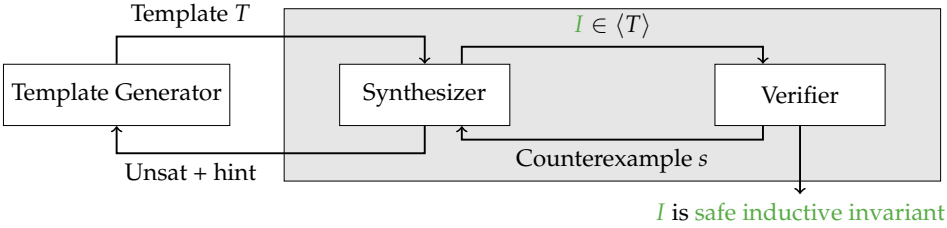
$$T = [c = 0 \land x < 6] \cdot (\alpha \cdot c + \beta \cdot x + \gamma) + [c = 1]$$

$$\langle T \rangle \ni I = [c = 0 \land x < 6] \cdot (0 \cdot c - 9/110 \cdot x + 28/55) + [c = 1]$$

Given: Linear loop `while` $(\varphi)\ \{\ body\ \}$, LIA formula Target (e.g. $c = 1$)

Goal: Verify $Pr(s_{\text{init}} \models \Diamond \text{Target}) \leq \lambda$

Approach: Compute safe inductive invariant $I : \text{States} \to \mathbb{R}$ via CEGIS:



$$T = [c = 0 \land x < 6] \cdot (\alpha \cdot c + \beta \cdot x + \gamma) + [c = 1]$$

$$\langle T \rangle \ni I = [c = 0 \land x < 6] \cdot (0 \cdot c - 9/110 \cdot x + 28/55) + [c = 1]$$

Inductive Synthesis of Inductive Invariants

Given: $\texttt{while}\,(\,c = 0 \wedge x < 6\,)\,\{\,c \coloneqq 1\ [0.1]\ x \coloneqq x + 1\,\}$ $\quad$ Target $\triangleq\ c = 1$

Goal: Verify $Pr(0, 0 \models \Diamond \text{Target}) \ \leq\ 0.9$

Inductive Synthesis of Inductive Invariants

Given: `while` $(c = 0 \wedge x < 6) \{ c := 1 \; [0.1] \; x := x + 1 \}$ Target $\triangleq$ $c = 1$

Goal: Verify $Pr(0, 0 \models \Diamond \text{Target}) \leq 0.9$

Template Generator:

$T = [c = 0 \wedge x < 6] \cdot (\alpha \cdot c + \beta \cdot x + \gamma) + [c = 1]$

Inductive Synthesis of Inductive Invariants

Given: `while` $(c = 0 \wedge x < 6) \{ c := 1 \; [0.1] \; x := x + 1 \}$ Target $\triangleq$ $c = 1$

Goal: Verify $Pr(0, 0 \models \Diamond \text{Target}) \leq 0.9$

Template Generator:

$T = [c = 0 \wedge x < 6] \cdot (\alpha \cdot c + \beta \cdot x + \gamma) + [c = 1]$

Inductive Synthesis of Inductive Invariants

Given: $\texttt{while}\,(\,c = 0 \wedge x < 6\,)\,\{\,c \coloneqq 1\ [0.1]\ x \coloneqq x + 1\,\}$ Target $\triangleq$ $c = 1$

Goal: Verify $Pr(0,0 \models \Diamond \text{Target}) \leq 0.9$

Template Generator:

$T = [c = 0 \wedge x < 6] \cdot (\alpha \cdot c + \beta \cdot x + \gamma) + [c = 1]$

Synthesizer:

$I = [c = 0 \wedge x < 6] \cdot (0 \cdot c + 0 \cdot x + 0) + [c = 1]$

Given: $\texttt{while}\,(\,c = 0 \wedge x < 6\,)\,\{\,c \coloneqq 1\,[0.1]\,x \coloneqq x + 1\,\}$ Target $\triangleq$ $c = 1$

Goal: Verify $Pr(0,0 \models \Diamond \text{Target}) \leq 0.9$

Template Generator:

$T = [c = 0 \wedge x < 6] \cdot (\alpha \cdot c + \beta \cdot x + \gamma) + [c = 1]$

Synthesizer:

$I = [c = 0 \wedge x < 6] \cdot (0 \cdot c + 0 \cdot x + 0) + [c = 1]$

Verifier:

Not inductive: $\Phi(I)(0,5) \not\leq I(0,5)$

Given: $\texttt{while}\,(\,c = 0 \wedge x < 6\,)\,\{\,c \coloneqq 1\ [0.1]\ x \coloneqq x + 1\,\}$ Target $\triangleq\ c = 1$

Goal: Verify $Pr(0,0 \models \Diamond \text{Target}) \leq 0.9$

Template Generator:

$T = [c = 0 \wedge x < 6] \cdot (\alpha \cdot c + \beta \cdot x + \gamma) + [c = 1]$

Synthesizer:

$I = [c = 0 \wedge x < 6] \cdot (0 \cdot c + 0 \cdot x + 0) + [c = 1]$

Verifier:

Not inductive: $\Phi(I)(0,5) \not\leq I(0,5)$

Learn constraint: $\Phi(T)(0,5) = \quad 0.1 \leq \alpha \cdot 0 + \beta \cdot 5 + \gamma \quad = T(0,5)$

Given: `while` $(c = 0 \wedge x < 6) \{ c := 1 \; [0.1] \; x := x + 1 \}$ Target $\triangleq c = 1$

Goal: Verify $Pr(0, 0 \models \Diamond \text{Target}) \leq 0.9$

Template Generator:

$T = [c = 0 \wedge x < 6] \cdot (\alpha \cdot c + \beta \cdot x + \gamma) + [c = 1]$

Synthesizer:

Verifier:

Learned Constraints (linear!):

- $0.1 \leq \beta \cdot 5 + \gamma$

Given: `while` $(c = 0 \wedge x < 6) \{ c := 1 \; [0.1] \; x := x + 1 \}$ Target $\triangleq c = 1$

Goal: Verify $Pr(0, 0 \models \Diamond \text{Target}) \leq 0.9$

Template Generator:

$T = [c = 0 \wedge x < 6] \cdot (\alpha \cdot c + \beta \cdot x + \gamma) + [c = 1]$

Synthesizer:

$I = [c = 0 \wedge x < 6] \cdot (0 \cdot c + 0 \cdot x + 1/10) + [c = 1]$

Verifier:

Learned Constraints (linear!):

- $0.1 \leq \beta \cdot 5 + \gamma$

Given: $\texttt{while}\,(\,c = 0 \wedge x < 6\,)\,\{\,c \coloneqq 1\,[0.1]\,x \coloneqq x + 1\,\}$ Target $\triangleq$ $c = 1$

Goal: Verify $Pr(0,0 \models \Diamond \text{Target}) \leq 0.9$

Template Generator:

$T = [c = 0 \wedge x < 6] \cdot (\alpha \cdot c + \beta \cdot x + \gamma) + [c = 1]$

Synthesizer:

$I = [c = 0 \wedge x < 6] \cdot (0 \cdot c + 0 \cdot x + 1/10) + [c = 1]$

Verifier:

Not inductive: $\Phi(I)(0,3) \not\leq I(0,3)$

Learned Constraints (linear!):

- $0.1 \leq \beta \cdot 5 + \gamma$

Given: $\texttt{while}\,(\,c = 0 \wedge x < 6\,)\,\{\,c \coloneqq 1\ [0.1]\ x \coloneqq x + 1\,\}$ Target $\triangleq$ $c = 1$

Goal: Verify $Pr(0,0 \models \Diamond \text{Target}) \leq 0.9$

Template Generator:

$T = [c = 0 \wedge x < 6] \cdot (\alpha \cdot c + \beta \cdot x + \gamma) + [c = 1]$

Synthesizer:

$I = [c = 0 \wedge x < 6] \cdot (0 \cdot c + 0 \cdot x + 1/10) + [c = 1]$

Verifier:

Not inductive: $\Phi(I)(0,3) \not\leq I(0,3)$

Learned Constraints (linear!):

- $0.1 \leq \beta \cdot 5 + \gamma$

Learn constraint: $\Phi(T)(0,3) = \quad 0.1 + 0.9 \cdot (\beta \cdot 4 + \gamma) \leq \beta \cdot 3 + \gamma \quad = T(0,3)$

Given: `while` $(c = 0 \wedge x < 6) \{ c := 1 \ [0.1] \ x := x + 1 \}$ Target $\triangleq c = 1$

Goal: Verify $Pr(0,0 \models \Diamond \text{Target}) \leq 0.9$

Template Generator:

$T = [c = 0 \wedge x < 6] \cdot (\alpha \cdot c + \beta \cdot x + \gamma) + [c = 1]$

Synthesizer:

Verifier:

Learned Constraints (linear!):

- $0.1 \leq \beta \cdot 5 + \gamma$
- $0.1 + 0.9 \cdot (\beta \cdot 4 + \gamma) \leq \beta \cdot 3 + \gamma$

Inductive Synthesis of Inductive Invariants

Given: $\texttt{while}\,(\,c = 0 \wedge x < 6\,)\,\{\,c \coloneqq 1\,[0.1]\,x \coloneqq x + 1\,\}$ Target $\triangleq$ $c = 1$

Goal: Verify $Pr(0,0 \models \Diamond \text{Target}) \leq 0.9$

Template Generator:

$T = [c = 0 \wedge x < 6] \cdot (\alpha \cdot c + \beta \cdot x + \gamma) + [c = 1]$

Synthesizer:

Verifier:

Learned Constraints (linear!):

- $0.1 \leq \beta \cdot 5 + \gamma$
- $0.1 + 0.9 \cdot (\beta \cdot 4 + \gamma) \leq \beta \cdot 3 + \gamma$
- $0.1 + 0.9 \cdot (\beta \cdot 5 + \gamma) \leq \beta \cdot 4 + \gamma$

Given: $\texttt{while}\,(\,c = 0 \wedge x < 6\,)\,\{\,c \coloneqq 1\,[0.1]\,x \coloneqq x + 1\,\}$ Target $\triangleq$ $c = 1$

Goal: Verify $Pr(0,0 \models \Diamond \text{Target}) \leq 0.9$

Template Generator:

$T = [c = 0 \wedge x < 6] \cdot (\alpha \cdot c + \beta \cdot x + \gamma) + [c = 1]$

Synthesizer:

$I = [c = 0 \wedge x < 6] \cdot (0 \cdot c - 1/50 \cdot x + 9/10) + [c = 1]$

Verifier:

Learned Constraints (linear!):

- $0.1 \leq \beta \cdot 5 + \gamma$
- $0.1 + 0.9 \cdot (\beta \cdot 4 + \gamma) \leq \beta \cdot 3 + \gamma$
- $0.1 + 0.9 \cdot (\beta \cdot 5 + \gamma) \leq \beta \cdot 4 + \gamma$

Given: $\texttt{while}\,(\,c = 0 \wedge x < 6\,)\,\{\,c := 1\ [0.1]\ x := x + 1\,\}$ Target $\triangleq\ c = 1$

Goal: Verify $Pr(0,0 \models \Diamond \text{Target}) \leq 0.9$

Template Generator:

$T = [c = 0 \wedge x < 6] \cdot (\alpha \cdot c + \beta \cdot x + \gamma) + [c = 1]$

Synthesizer:

$I = [c = 0 \wedge x < 6] \cdot (0 \cdot c - 1/50 \cdot x + 9/10) + [c = 1]$

Verifier:

Safe inductive invariant

Learned Constraints (linear!):

- $0.1 \leq \beta \cdot 5 + \gamma$
- $0.1 + 0.9 \cdot (\beta \cdot 4 + \gamma) \leq \beta \cdot 3 + \gamma$
- $0.1 + 0.9 \cdot (\beta \cdot 5 + \gamma) \leq \beta \cdot 4 + \gamma$

Inductive Synthesis of Inductive Invariants

Given: $\texttt{while}\,(\,c = 0 \wedge x < 6\,)\,\{\,c \coloneqq 1\ [0.1]\ x \coloneqq x + 1\,\}$ $\quad$ Target $\triangleq\ c = 1$

Goal: Verify $Pr(0, 0 \models \Diamond \text{Target}) \ \leq\ 0.5$

Given: $\texttt{while}\,(\,c = 0 \wedge x < 6\,)\,\{\,c \coloneqq 1\ [0.1]\ x \coloneqq x + 1\,\}$ Target $\triangleq\ c = 1$

Goal: Verify $Pr(0,0 \models \Diamond \text{Target}) \leq 0.5$

Template Generator:

$T = [c = 0 \wedge x < 6] \cdot (\alpha \cdot c + \beta \cdot x + \gamma) + [c = 1]$

Synthesizer:

Verifier:

Learned Constraints (linear!):

- $0.1 \leq \beta \cdot 5 + \gamma$
- $0.1 + 0.9 \cdot (\beta \cdot 4 + \gamma) \leq \beta \cdot 3 + \gamma$
- $0.1 + 0.9 \cdot (\beta \cdot 5 + \gamma) \leq \beta \cdot 4 + \gamma$

Given: $\texttt{while}\,(\,c = 0 \wedge x < 6\,)\,\{\,c \coloneqq 1\ [0.1]\ x \coloneqq x + 1\,\}$ Target $\triangleq$ $c = 1$

Goal: Verify $Pr(0,0 \models \Diamond \text{Target}) \leq 0.5$

Template Generator:

$T = [c = 0 \wedge x < 6] \cdot (\alpha \cdot c + \beta \cdot x + \gamma) + [c = 1]$

Synthesizer:

Verifier:

Learned Constraints (linear!):

- $0.1 \leq \beta \cdot 5 + \gamma$
- $0.1 + 0.9 \cdot (\beta \cdot 4 + \gamma) \leq \beta \cdot 3 + \gamma$
- $0.1 + 0.9 \cdot (\beta \cdot 5 + \gamma) \leq \beta \cdot 4 + \gamma$
- $\gamma \leq 0.5$

Given: `while` $(\, c = 0 \wedge x < 6 \,) \, \{ \, c \coloneqq 1 \; [0.1] \; x \coloneqq x + 1 \, \}$ Target $\triangleq \; c = 1$

Goal: Verify $Pr(0,0 \models \Diamond \text{Target}) \; \leq \; 0.5$

Template Generator:

$T = [c = 0 \wedge x < 6] \cdot (\alpha \cdot c + \beta \cdot x + \gamma) + [c = 1]$

Synthesizer:

UNSAT

Verifier:

Learned Constraints (linear!):

- $0.1 \leq \beta \cdot 5 + \gamma$
- $0.1 + 0.9 \cdot (\beta \cdot 4 + \gamma) \leq \beta \cdot 3 + \gamma$
- $0.1 + 0.9 \cdot (\beta \cdot 5 + \gamma) \leq \beta \cdot 4 + \gamma$
- $\gamma \leq 0.5$

Initial Template: $T_1 = [c = 0 \wedge x < 6] \cdot (\alpha \cdot c + \beta \cdot x + \gamma) + [c = 1]$

Heuristics for Template Refinement

Initial Template: $T_1 = [c = 0 \wedge x < 6] \cdot (\alpha \cdot c + \beta \cdot x + \gamma) + [c = 1]$

Heuristics for Template Refinement

- Static refinement for finite-state programs:

$$\begin{aligned} T_2 \;=\; & [c = 0 \wedge x < 3] \cdot (\alpha_1 \cdot c + \beta_1 \cdot x + \gamma_1) \\ & + [c = 0 \wedge x \geq 3 \wedge x < 6] \cdot (\alpha_2 \cdot c + \beta_2 \cdot x + \gamma_2) + [c = 1] \end{aligned}$$

Initial Template: $T_1 = [c = 0 \wedge x < 6] \cdot (\alpha \cdot c + \beta \cdot x + \gamma) + [c = 1]$

Heuristics for Template Refinement

- Static refinement for finite-state programs:

$$\begin{aligned} T_2 \;=\; & [c = 0 \wedge x < 3] \cdot (\alpha_1 \cdot c + \beta_1 \cdot x + \gamma_1) \\ & + [c = 0 \wedge x \geq 3 \wedge x < 6] \cdot (\alpha_2 \cdot c + \beta_2 \cdot x + \gamma_2) + [c = 1] \end{aligned}$$

- Inductivity-guided refinement: Uses obtained counterexamples

Initial Template: $T_1 = [c = 0 \wedge x < 6] \cdot (\alpha \cdot c + \beta \cdot x + \gamma) + [c = 1]$

Heuristics for Template Refinement

- Static refinement for finite-state programs:

$$\begin{aligned} T_2 \;=\; & [c = 0 \wedge x < 3] \cdot (\alpha_1 \cdot c + \beta_1 \cdot x + \gamma_1) \\ & + [c = 0 \wedge x \geq 3 \wedge x < 6] \cdot (\alpha_2 \cdot c + \beta_2 \cdot x + \gamma_2) + [c = 1] \end{aligned}$$

- Inductivity-guided refinement: Uses obtained counterexamples
- Dynamic refinement:

$$\begin{aligned} T_2 \;=\; & [c = 0 \wedge x < \delta_1] \cdot (\alpha_1 \cdot c + \beta_1 \cdot x + \gamma_1) \\ & + [c = 0 \wedge x \geq \delta_1 \wedge x < 6] \cdot (\alpha_2 \cdot c + \beta_2 \cdot x + \gamma_2) + [c = 1] \end{aligned}$$

Empirical Results

Our tool *cegispro2* implements our approach:

`https://github.com/moves-rwth/cegispro2`

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

		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

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

		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

- Few counterexamples often suffice.

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

		STORM			CEGISPRO2											
					INDUCT.-GUIDED				STATIC				DYNAMIC			
Prog	$\lvert S\rvert$	SP	DD	BEST	$\lvert S'\rvert$	$\lvert I\rvert$	t_s%	t	$\lvert S'\rvert$	$\lvert I\rvert$	t_s%	t	$\lvert S'\rvert$	$\lvert I\rvert$	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

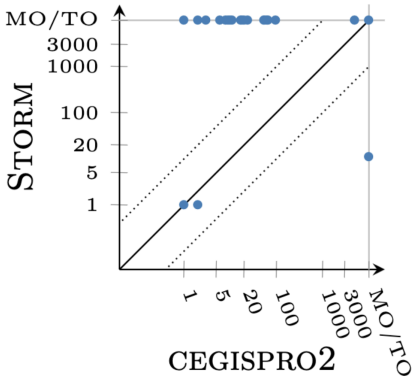
- Few counterexamples often suffice.
- Performance depends on tightness of threshold.

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

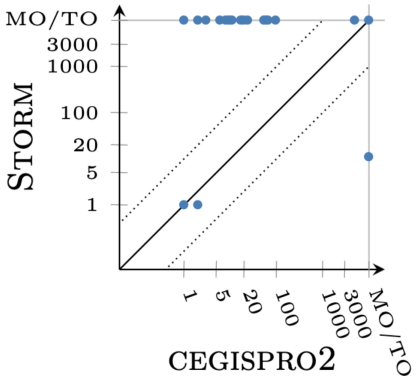
		STORM			CEGISPRO2											
					INDUCT.-GUIDED				STATIC				DYNAMIC			
Prog	$\|S\|$	SP	DD	BEST	$\|S'\|$	$\|I\|$	t_s%	t	$\|S'\|$	$\|I\|$	t_s%	t	$\|S'\|$	$\|I\|$	t_s%	t
boundedrwmultistep	$1 \cdot 10^5$	MO	TO	3	33	10	40	3	–	–	–	TO	–	–	–	TO
		MO	TO	10	55	16	36	10	–	–	–	TO	–	–	–	TO
		MO	TO	–	–	–	–	TO	–	–	–	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

- Few counterexamples often suffice.
- Performance depends on tightness of threshold.
- No clear winner between template refinements.

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

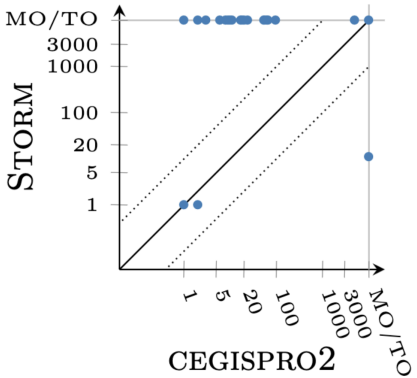


Finite-State Programs (Runtimes in seconds. TO=2h, MO=8gb)



***cegispro2* can verify interesting finite-state programs.**

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



***cegispro2* can verify interesting finite-state programs.**
Piecewise linear invariants can be sufficiently expressive.

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

$$\texttt{while}\,(\,a < 1000 \wedge b < 1000\,)\,\{\,a := a + 1\,[0.5]\,b := b + 1\,\}$$

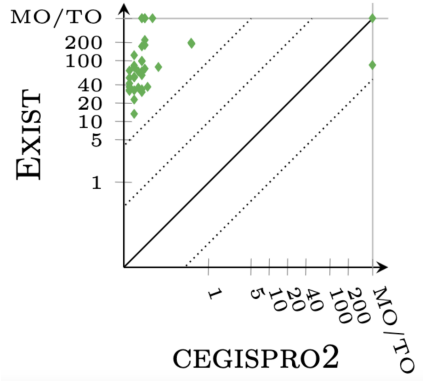
Finite-State Programs (Runtimes in seconds. TO=2h, MO=8gb)

$$\texttt{while}\,(\,a < 1000 \wedge b < 1000\,)\,\{\,a := a + 1\,[0.5]\,b := b + 1\,\}$$

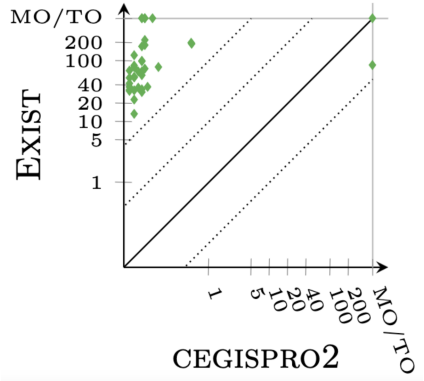
For $Pr(0, 0 \models \Diamond b = 1000) \leq 0.8$, *cegispro2* times out.
Storm terminates after 11 seconds.

Empirical Results

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)



***cegispro2* can verify infinite-state programs and is competitive with modern tools.**

Upper-bounding expected runtimes

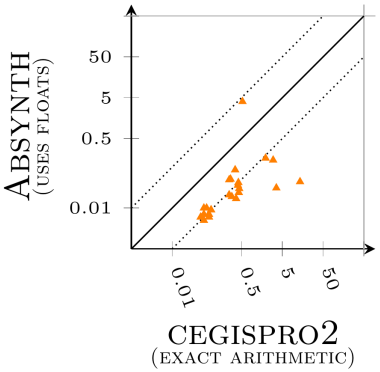
By adapting inductivity [Kaminski et al. '16]:

$$\forall s\colon 1 + \Phi(I)(s) \leq I(s)$$

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

By adapting inductivity [Kaminski et al. '16]:

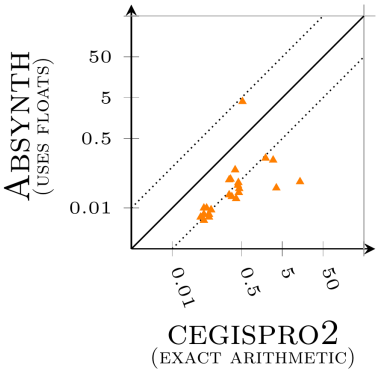
$$\forall s\colon 1 + \Phi(I)(s) \leq I(s)$$



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

By adapting inductivity [Kaminski et al. '16]:

$$\forall s\colon 1 + \Phi(I)(s) \leq I(s)$$



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

Conclusion

- Automatic and simple yet effective invariant synthesis technique

Conclusion

- Automatic and simple yet effective invariant synthesis technique
- Competitive performance. Few counterexamples often suffice

Conclusion

- Automatic and simple yet effective invariant synthesis technique
- Competitive performance. Few counterexamples often suffice
- Extension for computing bounds on ERTs

Conclusion

- Automatic and simple yet effective invariant synthesis technique
- Competitive performance. Few counterexamples often suffice
- Extension for computing bounds on ERTs

Future work

- More expressive invariants

Conclusion

- Automatic and simple yet effective invariant synthesis technique
- Competitive performance. Few counterexamples often suffice
- Extension for computing bounds on ERTs

Future work

- More expressive invariants
- Better template refinements

Conclusion

- Automatic and simple yet effective invariant synthesis technique
- Competitive performance. Few counterexamples often suffice
- Extension for computing bounds on ERTs

Future work

- More expressive invariants
- Better template refinements
- Continuous sampling?

Conclusion

- Automatic and simple yet effective invariant synthesis technique
- Competitive performance. Few counterexamples often suffice
- Extension for computing bounds on ERTs

Future work

- More expressive invariants
- Better template refinements
- Continuous sampling?

Thank you!

Backup Slides

Prog	Absynth t	Absynth bound	$\|S'\|$	cegispro2 t	cegispro2 bound
bayesiannetwork	0.15	1 + 2 max(0, n)	1	2.99	(n * 2.0)
ber	≤0.01	1 + 2 max(0, n - x)	3	0.08	((x * -2.0) + (n * 3.0))
cowboyduel	≤0.01	1 + 1.2 max(0, flag)	1	0.06	11/5
C4Bt09	0.02	1 + max(0, -j + x)	15	0.37	max[((j * -1.0) + x + 1.0), ((j * -1.0) + x + 1.0)]
C4Bt13	0.02	1 + 2 max(0, x) + max(0, y)	8	0.25	((x * 2.0) + y)
C4Bt19	0.04	3 + max(0, 51 + i + k) + 2 max(0, i)	16	0.42	((i * 2.0) + k + -46.0)
C4Bt61	0.02	2 + max(0, l)	15	0.43	(l + -3.0)
condand	≤0.01	1 + 2 max(0, m)	4	0.09	((m * 2.0) + 1.0)
coupon	0.05	10 max(0, 5 - i)	5	0.25	max[149/12, 137/12, 61/6, 17/2, (i * 3/2)]
fcall	≤0.01	1 + 4 max(0, n - x)	3	0.09	((x * -3.0) + (n * 3.0))
fillingvol	0.09	1 + 0.666667 max(0, 10 - vM + vTF)	10	0.35	max[((vM * -1/36) + (vTF * 1/36) + 2.0), ((vM * -2/3) + (vTF * 2/3) + -583/180)]
geo	≤0.01	3	1	0.06	3.0
linear01	≤0.01	1 + 0.6 max(0, x)	1	0.06	x

Backup Slides

Prog	EXIST t	EXIST I	CEGISPRO2 $\|S'\|$	CEGISPRO2 t	CEGISPRO2 I
BiasDir1_0	78.31	$x + [x = y] \cdot (-0.5 \cdot x + -0.5 \cdot y + 0.1)$	0	0.15	$[\text{loop guard}] \cdot 0 + [\neg\text{loop guard}] \cdot x$
BiasDir1_1	97.29	$x + [x = y] \cdot (-0.5 \cdot x + -0.5 \cdot y + 0.5)$	1	0.08	$[\text{loop guard}] \cdot 0.5 + [\neg\text{loop guard}] \cdot x$
BiasDir2_0	66.42	$x + [x = y] \cdot (-0.5 \cdot x + -0.5 \cdot y + 0.1)$	0	0.07	$[\text{loop guard}] \cdot 0 + [\neg\text{loop guard}] \cdot x$
BiasDir2_1	171.83	$x + [x = y] \cdot (-0.5 \cdot x + -0.5 \cdot y + 0.5)$	1	0.08	$[\text{loop guard}] \cdot 0.5 + [\neg\text{loop guard}] \cdot x$
BiasDir3_0	66.55	$x + [x = y] \cdot (-0.3 \cdot x + -0.3 \cdot y)$	0	0.08	$[\text{loop guard}] \cdot 0 + [\neg\text{loop guard}] \cdot x$
BiasDir3_1	99.23	$x + [x = y] \cdot (-0.5 \cdot x + -0.5 \cdot y + 0.5)$	1	0.08	$[\text{loop guard}] \cdot 0.5 + [\neg\text{loop guard}] \cdot x$
Bin01_0	53.19	$x + [n > 0] \cdot 0$	1	0.06	$[\text{loop guard}] \cdot x + [\neg\text{loop guard}] \cdot x$
Bin02_0	83.04	$x + [n > 0] \cdot 0$	1	0.06	$[\text{loop guard}] \cdot x + [\neg\text{loop guard}] \cdot x$
Bin03_0	53.13	$x + [n > 0] \cdot 0$	1	0.06	$[\text{loop guard}] \cdot x + [\neg\text{loop guard}] \cdot x$
Bin11_0	32.52	$x + [n < M] \cdot 0$	1	0.06	$[\text{loop guard}] \cdot x + [\neg\text{loop guard}] \cdot x$