

Lower Bounds for Possibly Divergent Probabilistic Programs

Shenghua Feng, **Mingshuai Chen**, Han Su, Benjamin L. Kaminski,
Joost-Pieter Katoen, Naijun Zhan



浙江大学
ZHEJIANG UNIVERSITY



UNIVERSITÄT
DES SAARLANDES



OOPSLA · Lisbon · October 2023



浙江大学
ZHEJIANG UNIVERSITY

FICTION

A Fun Fact

"A drunk man will find his way home, but a drunk bird may get lost forever."

— Shizuo Kakutani



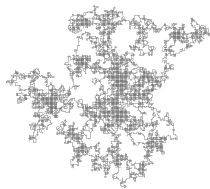
浙江大學
ZHEJIANG UNIVERSITY

FICTION

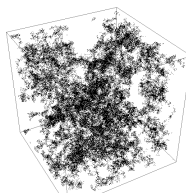
A Fun Fact

"A *drunk man* will find his way home, but a *drunk bird* may get lost forever."

— Shizuo Kakutani



©Wikipedia



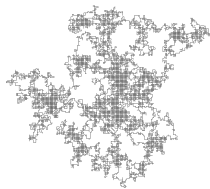
©StackExchange

A *2-D* symmetric random walk on a lattice returns to the origin *almost-surely*, yet *not* its *3-D* counterpart [Pólya, Math. Ann. '21].

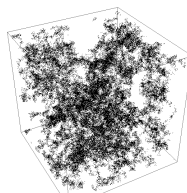
A Fun Fact

"A *drunk man* will find his way home, but a *drunk bird* may get lost forever."

— Shizuo Kakutani



©Wikipedia



©StackExchange

A *2-D* symmetric random walk on a lattice returns to the origin *almost-surely*, yet *not* its *3-D* counterpart [Pólya, Math. Ann. '21].

Question : How to compute *sound approx.* of the returning probability of the bird?



浙江大學
ZHEJIANG UNIVERSITY

FICTION

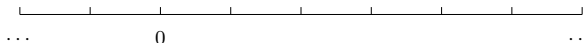
Probabilistic Programs

C_{brw} : `while ( $n > 0$ ) {  $n := n - 1$   $[1/3]$   $n := n + 1$  }`



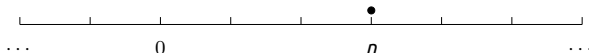
Probabilistic Programs

C_{brw} : $\text{while}(n > 0) \{ n := n - 1 \text{ } [1/3] \text{ } n := n + 1 \}$



Probabilistic Programs

C_{brw} : $\text{while}(n > 0) \{ n := n - 1 \text{ [1/3]} \ n := n + 1 \}$



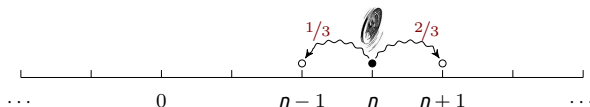
Probabilistic Programs

C_{brw} : $\text{while}(n > 0) \{ n := n - 1 \text{ [1/3]} \ n := n + 1 \}$



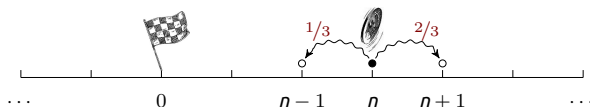
Probabilistic Programs

C_{brw} : $\text{while}(n > 0) \{ n := n - 1 \text{ } [1/3] \text{ } n := n + 1 \}$



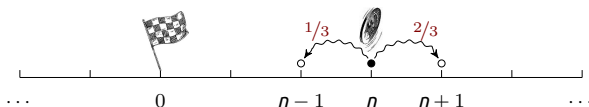
Probabilistic Programs

C_{brw} : $\text{while}(n > 0) \{ n := n - 1 \text{ } [1/3] \text{ } n := n + 1 \}$



Probabilistic Programs

C_{brw} : $\text{while}(n > 0) \{ n := n - 1 \text{ } [1/3] \text{ } n := n + 1 \}$

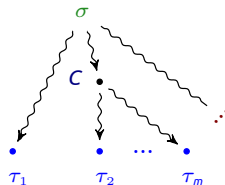


*"The crux of probabilistic programming is to treat normal-looking programs as if they were **probability distributions**."*

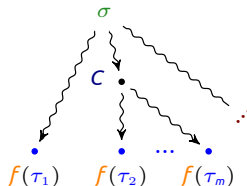
— Michael Hicks, The PL Enthusiast



Quantitative Reasoning about Probabilistic Loops [Kozen; McIver, Morgan; Kaminski]



Quantitative Reasoning about Probabilistic Loops [Kozen; McIver, Morgan; Kaminski]



Quantitative Reasoning about Probabilistic Loops [Kozen; McIver, Morgan; Kaminski]

$$\text{wp}[\mathbb{C}](f)(\sigma) \triangleq \text{Exp}[f(\tau_1) \quad f(\tau_2) \quad \dots \quad f(\tau_m)]$$



Quantitative Reasoning about Probabilistic Loops

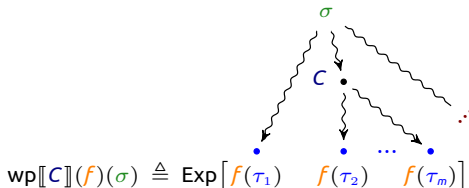
[Kozen; McIver, Morgan; Kaminski]

$$\text{wp}[\![C]\!](f)(\sigma) \triangleq \text{Exp} [f(\tau_1) \quad f(\tau_2) \quad \dots \quad f(\tau_m)]$$

$$\text{wp}[\![n := 5]\!](n) = 5$$



Quantitative Reasoning about Probabilistic Loops [Kozen; McIver, Morgan; Kaminski]

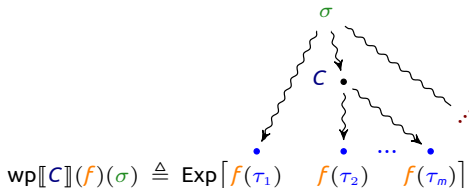


$$\text{wp}[[n := 5]](n) = 5$$

$$\text{wp}[[n := n - 1 \text{ } [1/3] \text{ } n := n + 1]](n) = 1/3 \cdot (n - 1) + 2/3 \cdot (n + 1) = n + 1/3$$



Quantitative Reasoning about Probabilistic Loops [Kozen ; McIver, Morgan ; Kaminski]



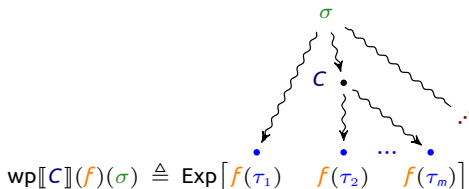
$$\text{wp}[[n := 5]](n) = 5$$

$$\text{wp}[[n := n - 1 \text{ [1/3]} \quad n := n + 1]](n) = \frac{1}{3} \cdot (n - 1) + \frac{2}{3} \cdot (n + 1) = n + \frac{1}{3}$$

$$\text{wp}[[\text{while } (n > 0) \{ n := n - 1 \text{ [1/3]} \quad n := n + 1 \}]](1) = ?$$



Quantitative Reasoning about Probabilistic Loops [Kozen; McIver, Morgan; Kaminski]



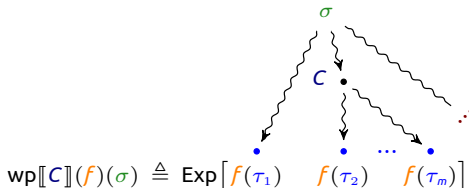
$$\text{wp}[[n := 5]](n) = 5$$

$$\text{wp}[[n := n - 1 \text{ [1/3]} \quad n := n + 1]](n) = 1/3 \cdot (n - 1) + 2/3 \cdot (n + 1) = n + 1/3$$

$$\text{wp}[[\text{while } (n > 0) \{ n := n - 1 \text{ [1/3]} \quad n := n + 1 \}]](1) = [n < 0] + [n \geq 0] \cdot (1/2)^n$$



Quantitative Reasoning about Probabilistic Loops [Kozen; McIver, Morgan; Kaminski]



$$\text{wp}[[n := 5]](n) = 5$$

$$\text{wp}[[n := n - 1 \text{ [1/3]} \quad n := n + 1]](n) = 1/3 \cdot (n - 1) + 2/3 \cdot (n + 1) = n + 1/3$$

$$\text{wp}[[\text{while } (n > 0) \{ n := n - 1 \text{ [1/3]} \quad n := n + 1 \}]](1) = [n < 0] + [n \geq 0] \cdot (1/2)^n$$

$$\text{wp}[[\text{while } (\varphi) \{ C \}]](f) = \text{lfp } \Phi_f = ?$$



Bounding the Least Fixed Point

$$l \preceq lfp \Phi_f \preceq u$$



Bounding the Least Fixed Point

$$l \preceq lfp \Phi_f \preceq u$$

■ Upper bounds (Park induction) :

$$\Phi_f(u) \preceq u \text{ implies } lfp \Phi_f \preceq u.$$



Bounding the Least Fixed Point

$$l \preceq lfp \Phi_f \preceq u$$

■ Upper bounds (Park induction) :

$$\Phi_f(u) \preceq u \text{ implies } lfp \Phi_f \preceq u. \quad u \bullet$$



Bounding the Least Fixed Point

$$l \preceq lfp \Phi_f \preceq u$$

■ Upper bounds (Park induction) :

$$\Phi_f(u) \preceq u \text{ implies } lfp \Phi_f \preceq u.$$

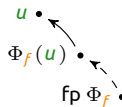


Bounding the Least Fixed Point

$$l \preceq \text{lfp } \Phi_f \preceq u$$

■ Upper bounds (Park induction):

$$\Phi_f(u) \preceq u \text{ implies } \text{lfp } \Phi_f \preceq u.$$

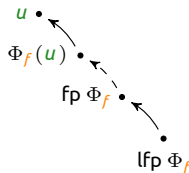


Bounding the Least Fixed Point

$$l \preceq \text{lfp } \Phi_f \preceq u$$

■ Upper bounds (Park induction):

$$\Phi_f(u) \preceq u \text{ implies } \text{lfp } \Phi_f \preceq u.$$



Bounding the Least Fixed Point

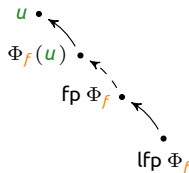
$$l \preceq \text{lfp } \Phi_f \preceq u$$

■ Upper bounds (Park induction) :

$$\Phi_f(u) \preceq u \text{ implies } \text{lfp } \Phi_f \preceq u.$$

■ Lower bounds :

$$l \preceq \Phi_f(l) \text{ implies } l \preceq \text{lfp } \Phi_f.$$



Bounding the Least Fixed Point

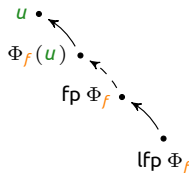
$$l \preceq \text{lfp } \Phi_f \preceq u$$

■ Upper bounds (Park induction) :

$$\Phi_f(u) \preceq u \text{ implies } \text{lfp } \Phi_f \preceq u.$$

■ Lower bounds :

$$l \preceq \Phi_f(l) \text{ implies } l \preceq \text{lfp } \Phi_f.$$



Bounding the Least Fixed Point

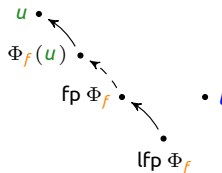
$$l \preceq \text{lfp } \Phi_f \preceq u$$

■ Upper bounds (Park induction) :

$$\Phi_f(u) \preceq u \text{ implies } \text{lfp } \Phi_f \preceq u.$$

■ Lower bounds :

$$l \preceq \Phi_f(l) \text{ implies } l \preceq \text{lfp } \Phi_f.$$



Bounding the Least Fixed Point

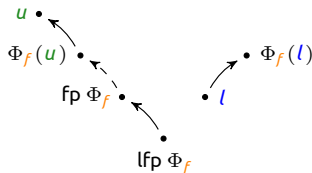
$$l \preceq \text{lfp } \Phi_f \preceq u$$

■ Upper bounds (Park induction) :

$$\Phi_f(u) \preceq u \text{ implies } \text{lfp } \Phi_f \preceq u.$$

■ Lower bounds :

$$l \preceq \Phi_f(l) \text{ implies } l \preceq \text{lfp } \Phi_f.$$



Bounding the Least Fixed Point

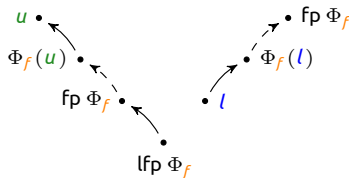
$$l \preceq \text{lfp } \Phi_f \preceq u$$

■ Upper bounds (Park induction) :

$$\Phi_f(u) \preceq u \text{ implies } \text{lfp } \Phi_f \preceq u.$$

■ Lower bounds :

$$l \preceq \Phi_f(l) \text{ implies } l \preceq \text{lfp } \Phi_f.$$



Bounding the Least Fixed Point

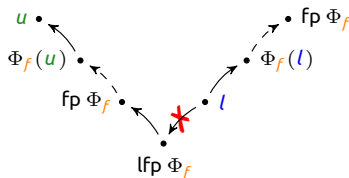
$$l \preceq \text{lfp } \Phi_f \preceq u$$

■ Upper bounds (Park induction) :

$$\Phi_f(u) \preceq u \text{ implies } \text{lfp } \Phi_f \preceq u.$$

■ Lower bounds :

$$l \preceq \Phi_f(l) \text{ implies } l \preceq \text{lfp } \Phi_f.$$



Bounding the Least Fixed Point

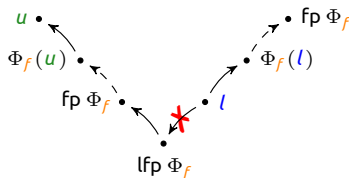
$$l \preceq \text{lfp } \Phi_f \preceq u$$

■ Upper bounds (Park induction) :

$$\Phi_f(u) \preceq u \text{ implies } \text{lfp } \Phi_f \preceq u.$$

■ Lower bounds ([Hark et al., POPL '20]) :

$$l \preceq \Phi_f(l) \wedge l \text{ is uni. int. implies } l \preceq \text{lfp } \Phi_f.$$



Bounding the Least Fixed Point

$$l \preceq lfp \Phi_f \preceq u$$

■ Upper bounds (Park induction) :

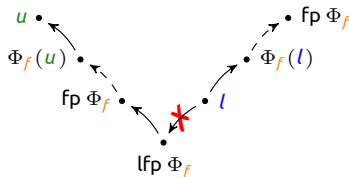
$$\Phi_f(u) \preceq u \text{ implies } lfp \Phi_f \preceq u.$$

■ Lower bounds ([Hark et al., POPL '20]) :

$$l \preceq \Phi_f(l) \wedge \boxed{l \text{ is uni. int.}} \text{ implies } l \preceq lfp \Phi_f.$$

almost-sure termination (AST)
bounded expectations

...



A New Proof Rule for Lower Bounds

Theorem (Guard-Strengthening Rule)

$$C_{\text{loop}}: \text{while}(\varphi)\{C\} \rightsquigarrow C'_{\text{loop}}: \text{while}(\varphi')\{C\}$$

A New Proof Rule for Lower Bounds

Theorem (Guard-Strengthening Rule)

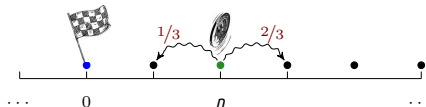
$$\begin{array}{c}
 C_{\text{loop}}: \text{ while } (\varphi) \{ C \} \quad \rightsquigarrow \quad C'_{\text{loop}}: \text{ while } (\varphi') \{ C \} \\
 \varphi' \Rightarrow \varphi \quad l \preceq \text{wp} \llbracket C'_{\text{loop}} \rrbracket ([\neg \varphi] \cdot f) \\
 \hline
 l \preceq \text{wp} \llbracket C_{\text{loop}} \rrbracket (f) \quad \text{(Guard-Strengthening)}
 \end{array}$$

A New Proof Rule for Lower Bounds

Theorem (Guard-Strengthening Rule)

$$\begin{array}{c}
 C_{\text{loop}}: \text{ while } (\varphi) \{ C \} \quad \rightsquigarrow \quad C'_{\text{loop}}: \text{ while } (\varphi') \{ C \} \\
 \frac{\varphi' \Rightarrow \varphi \quad l \preceq \text{wp} \llbracket C'_{\text{loop}} \rrbracket ([\neg \varphi] \cdot f)}{l \preceq \text{wp} \llbracket C_{\text{loop}} \rrbracket (f)} \quad (\text{Guard-Strengthening})
 \end{array}$$

$C_{\text{brw}}: \text{ while } (0 < n) \{$
 $\quad n := n - 1 \ [1/3] \ n := n + 1 \}$

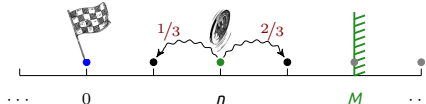


A New Proof Rule for Lower Bounds

Theorem (Guard-Strengthening Rule)

$$\begin{array}{c}
 C_{\text{loop}}: \text{ while } (\varphi) \{ C \} \quad \rightsquigarrow \quad C'_{\text{loop}}: \text{ while } (\varphi') \{ C \} \\
 \frac{\varphi' \Rightarrow \varphi \quad l \preceq \text{wp}[C'_{\text{loop}}](\lceil \neg \varphi \rceil \cdot f)}{l \preceq \text{wp}[C_{\text{loop}}](f)} \quad (\text{Guard-Strengthening})
 \end{array}$$

$$\begin{array}{c}
 C_{\text{brw}}: \text{ while } (0 < n) \{ \\
 \quad n := n - 1 \ [1/3] \ n := n + 1 \} \quad \rightsquigarrow \quad C^M_{\text{brw}}: \text{ while } (0 < n < M) \{ \\
 \quad n := n - 1 \ [1/3] \ n := n + 1 \}
 \end{array}$$



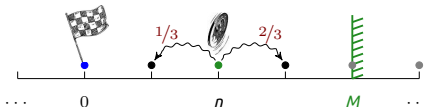
A New Proof Rule for Lower Bounds

Theorem (Guard-Strengthening Rule)

$$\begin{array}{c}
 C_{\text{loop}}: \text{ while } (\varphi) \{ C \} \quad \rightsquigarrow \quad C'_{\text{loop}}: \text{ while } (\varphi') \{ C \} \\
 \frac{\varphi' \Rightarrow \varphi \quad l \preceq \text{wp} \llbracket C'_{\text{loop}} \rrbracket ([\neg \varphi] \cdot f)}{l \preceq \text{wp} \llbracket C_{\text{loop}} \rrbracket (f)} \quad (\text{Guard-Strengthening})
 \end{array}$$

$$\begin{array}{c}
 C_{\text{brw}}: \text{ while } (0 < n) \{ \\
 \quad n := n - 1 \ [1/3] \ n := n + 1 \} \\
 \rightsquigarrow \quad C^M_{\text{brw}}: \text{ while } (0 < n < M) \{ \\
 \quad n := n - 1 \ [1/3] \ n := n + 1 \}
 \end{array}$$

$$\text{wp} \llbracket C^M_{\text{brw}} \rrbracket ([n \leq 0] \cdot 1) \preceq \text{wp} \llbracket C_{\text{brw}} \rrbracket (1)$$



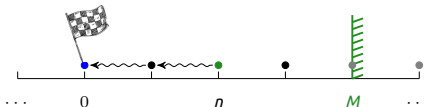
A New Proof Rule for Lower Bounds

Theorem (Guard-Strengthening Rule)

$$\begin{array}{c}
 C_{\text{loop}}: \text{ while } (\varphi) \{ C \} \quad \rightsquigarrow \quad C'_{\text{loop}}: \text{ while } (\varphi') \{ C \} \\
 \frac{\varphi' \Rightarrow \varphi \quad l \preceq \text{wp}[C'_{\text{loop}}]([\neg\varphi] \cdot f)}{l \preceq \text{wp}[C_{\text{loop}}](f)} \quad (\text{Guard-Strengthening})
 \end{array}$$

$$\begin{array}{c}
 C_{\text{brw}}: \text{ while } (0 < n) \{ \\
 \quad n := n - 1 \ [1/3] \ n := n + 1 \} \quad \rightsquigarrow \quad C^M_{\text{brw}}: \text{ while } (0 < n < M) \{ \\
 \quad \quad \quad n := n - 1 \ [1/3] \ n := n + 1 \}
 \end{array}$$

$$\text{wp}[C^M_{\text{brw}}]([n \leq 0] \cdot 1) \preceq \text{wp}[C_{\text{brw}}](1)$$



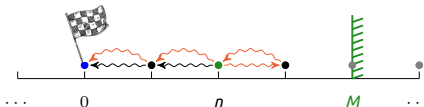
A New Proof Rule for Lower Bounds

Theorem (Guard-Strengthening Rule)

$$\begin{array}{c}
 C_{\text{loop}}: \text{ while } (\varphi) \{ C \} \quad \rightsquigarrow \quad C'_{\text{loop}}: \text{ while } (\varphi') \{ C \} \\
 \frac{\varphi' \Rightarrow \varphi \quad l \preceq \text{wp}[C'_{\text{loop}}]([\neg\varphi] \cdot f)}{l \preceq \text{wp}[C_{\text{loop}}](f)} \quad (\text{Guard-Strengthening})
 \end{array}$$

$$\begin{array}{c}
 C_{\text{brw}}: \text{ while } (0 < n) \{ \\
 \quad n := n - 1 \ [1/3] \ n := n + 1 \} \quad \rightsquigarrow \quad C^M_{\text{brw}}: \text{ while } (0 < n < M) \{ \\
 \quad \quad \quad n := n - 1 \ [1/3] \ n := n + 1 \}
 \end{array}$$

$$\text{wp}[C^M_{\text{brw}}]([n \leq 0] \cdot 1) \preceq \text{wp}[C_{\text{brw}}](1)$$



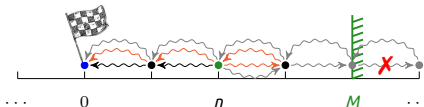
A New Proof Rule for Lower Bounds

Theorem (Guard-Strengthening Rule)

$$\begin{array}{c}
 C_{\text{loop}}: \text{ while } (\varphi) \{ C \} \quad \rightsquigarrow \quad C'_{\text{loop}}: \text{ while } (\varphi') \{ C \} \\
 \frac{\varphi' \Rightarrow \varphi \quad l \preceq \text{wp}[C'_{\text{loop}}]([\neg\varphi] \cdot f)}{l \preceq \text{wp}[C_{\text{loop}}](f)} \quad (\text{Guard-Strengthening})
 \end{array}$$

$$\begin{array}{c}
 C_{\text{brw}}: \text{ while } (0 < n) \{ \\
 \quad n := n - 1 \ [1/3] \ n := n + 1 \} \quad \rightsquigarrow \quad C^M_{\text{brw}}: \text{ while } (0 < n < M) \{ \\
 \quad n := n - 1 \ [1/3] \ n := n + 1 \}
 \end{array}$$

$$\text{wp}[C^M_{\text{brw}}]([n \leq 0] \cdot 1) \preceq \text{wp}[C_{\text{brw}}](1)$$



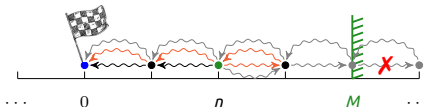
A New Proof Rule for Lower Bounds

Theorem (Guard-Strengthening Rule)

$$\begin{array}{c}
 C_{\text{loop}}: \text{ while } (\varphi) \{ C \} \quad \rightsquigarrow \quad C'_{\text{loop}}: \text{ while } (\varphi') \{ C \} \\
 \frac{\varphi' \Rightarrow \varphi \quad l \preceq \text{wp} \llbracket C'_{\text{loop}} \rrbracket ([\neg \varphi] \cdot f)}{l \preceq \text{wp} \llbracket C_{\text{loop}} \rrbracket (f)} \quad (\text{Guard-Strengthening})
 \end{array}$$

$$\begin{array}{c}
 C_{\text{brw}}: \text{ while } (0 < n) \{ \\
 \quad n := n - 1 \ [1/3] \ n := n + 1 \} \\
 \rightsquigarrow \quad C^M_{\text{brw}}: \text{ while } (0 < n < M) \{ \\
 \quad n := n - 1 \ [1/3] \ n := n + 1 \}
 \end{array}$$

$$\text{wp} \llbracket C^M_{\text{brw}} \rrbracket ([n \leq 0] \cdot 1) \preceq \text{wp} \llbracket C_{\text{brw}} \rrbracket (1)$$



- C_{loop} features a **stronger** termination property (e.g., becoming AST).
- **Easier** to verify the uni. int. of l and the boundedness of expectations.

Behind the Proof Rule

Theorem (wp-Difference)

$$\text{wp}[\llbracket C_{\text{loop}} \rrbracket](f) - \text{wp}[\llbracket C'_{\text{loop}} \rrbracket](f) =$$

$$\begin{aligned} & \text{wp}[\text{while}(\varphi \wedge \varphi')\{C\}](\lceil \neg\varphi \wedge \varphi' \rceil \cdot f) + \lambda\sigma \cdot \int_A f_{C_{\text{loop}}} d(\sigma\mathbb{P}) - \\ & \text{wp}[\text{while}(\varphi \wedge \varphi')\{C\}](\lceil \varphi \wedge \neg\varphi' \rceil \cdot f) - \lambda\sigma \cdot \int_B f_{C'_{\text{loop}}} d(\sigma\mathbb{P}) \end{aligned}$$

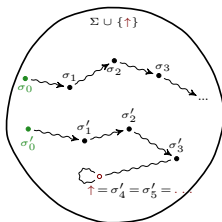


Figure – Infinite prog. traces.

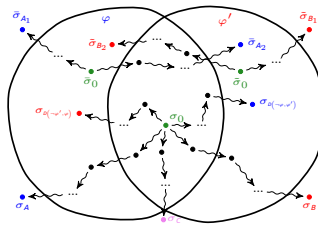


Figure – Illustration of wp-Difference.

- Potentially applicable to *sensitivity analysis* and *model repair*.

Properties of the Proof Rule

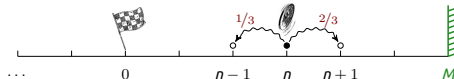
$$\frac{\varphi' \Rightarrow \varphi \quad l \preceq \text{wp}[\![C'_{\text{loop}}]\!]([\neg\varphi] \cdot f)}{l \preceq \text{wp}[\![C_{\text{loop}}]\!](f)} \quad (\text{Guard-Strengthening})$$

- (Trivially) **complete** : where there's an l , there's a φ' (albeit not “good” enough).

Properties of the Proof Rule

$$\frac{\varphi' \Rightarrow \varphi \quad l \preceq \text{wp}[\![C'_{\text{loop}}]\!]([\neg\varphi] \cdot f)}{l \preceq \text{wp}[\![C_{\text{loop}}]\!](f)} \quad \text{(Guard-Strengthening)}$$

- (Trivially) **complete** : where there's an l , there's a φ' (albeit not “good” enough).
- **General** : applicable to *possibly divergent* C_{loop} and unbounded expectations f, l :



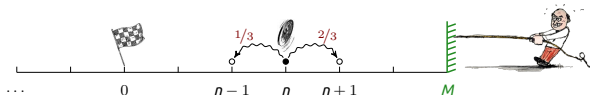
$$l_M = [n < 0] + [0 \leq n \leq M] \cdot \left((1/2)^n - (1/2)^M \right)$$

$$\forall M \in \mathbb{N}: \quad l_M \stackrel{\text{Hark}}{\preceq} \text{wp}[\![C_{\text{brw}}^M]\!]([n \leq 0] \cdot 1) \stackrel{\text{Stren.}}{\preceq} \text{wp}[\![C_{\text{brw}}]\!](1)$$

Properties of the Proof Rule

$$\frac{\varphi' \Rightarrow \varphi \quad l \preceq \text{wp}[\![C_{\text{loop}}]\!]([\neg\varphi] \cdot f)}{l \preceq \text{wp}[\![C_{\text{loop}}]\!](f)} \quad (\text{Guard-Strengthening})$$

- (Trivially) **complete** : where there's an l , there's a φ' (albeit not “good” enough).
- **General** : applicable to *possibly divergent* C_{loop} and unbounded expectations f, l :



$$l_M = [n < 0] + [0 \leq n \leq M] \cdot \left((1/2)^n - (1/2)^M \right)$$

$$\forall M \in \mathbb{N}: \quad l_M \stackrel{\text{Hark}}{\preceq} \text{wp}[\![C_{\text{brw}}^M]\!]([n \leq 0] \cdot 1) \stackrel{\text{Stren.}}{\preceq} \text{wp}[\![C_{\text{brw}}]\!](1)$$

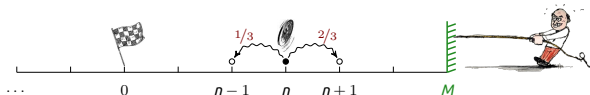
- **Tight** : the underapproximation error approaches 0 as $\varphi' \rightarrow \varphi$:

$$[n < 0] + [n \geq 0] \cdot (1/2)^n = \lim_{M \rightarrow \infty} l_M \stackrel{\text{Stren.}}{\preceq} \text{wp}[\![C_{\text{brw}}]\!](1)$$

Properties of the Proof Rule

$$\frac{\varphi' \Rightarrow \varphi \quad l \preceq \text{wp}[\![C_{\text{loop}}]\!]([\neg\varphi] \cdot f)}{l \preceq \text{wp}[\![C_{\text{loop}}]\!](f)} \quad (\text{Guard-Strengthening})$$

- (Trivially) **complete** : where there's an l , there's a φ' (albeit not “good” enough).
- **General** : applicable to *possibly divergent* C_{loop} and unbounded expectations f, l :



$$l_M = [n < 0] + [0 \leq n \leq M] \cdot \left((1/2)^n - (1/2)^M \right)$$

$$\forall M \in \mathbb{N}: \quad l_M \stackrel{\text{Hark}}{\preceq} \text{wp}[\![C_{\text{brw}}^M]\!]([n \leq 0] \cdot 1) \stackrel{\text{Stren.}}{\preceq} \text{wp}[\![C_{\text{brw}}]\!](1)$$

- **Tight** : the underapproximation error approaches 0 as $\varphi' \rightarrow \varphi$:

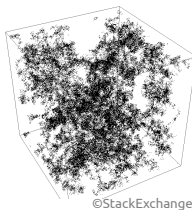
$$[n < 0] + [n \geq 0] \cdot (1/2)^n = \lim_{M \rightarrow \infty} l_M \stackrel{\text{Park}}{=} \text{wp}[\![C_{\text{brw}}]\!](1)$$

Properties of the Proof Rule

$$\frac{\varphi' \Rightarrow \varphi \quad l \preceq \text{wp}[\![C_{\text{loop}}]\!]([\neg\varphi] \cdot f)}{l \preceq \text{wp}[\![C_{\text{loop}}]\!](f)} \quad (\text{Guard-Strengthening})$$

- **Automatable** : reducible to *probabilistic model checking* for finite-state C_{loop} :

```
while (x ≠ 0 ∨ y ≠ 0 ∨ z ≠ 0) {
  x := x - 1 ⊕ x := x + 1 ⊕ y := y - 1 ⊕ y := y + 1 ⊕ z := z - 1 ⊕ z := z + 1 }
```

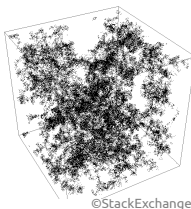


Properties of the Proof Rule

$$\frac{\varphi' \Rightarrow \varphi \quad l \preceq \text{wp}[\![C_{\text{loop}}]\!]([\neg\varphi] \cdot f)}{l \preceq \text{wp}[\![C_{\text{loop}}]\!](f)} \quad (\text{Guard-Strengthening})$$

- **Automatable** : reducible to *probabilistic model checking* for finite-state C_{loop} :

while $(x \neq 0 \vee y \neq 0 \vee z \neq 0) \{$
 $x := x - 1 \oplus x := x + 1 \oplus y := y - 1 \oplus y := y + 1 \oplus z := z - 1 \oplus z := z + 1 \}$



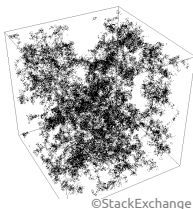
$$\mathcal{P} = 1 - \left(\frac{3}{(2\pi)^3} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} \frac{dx \, dy \, dz}{3 - \cos x - \cos y - \cos z} \right)^{-1} = 0.3405373296 \dots$$

Properties of the Proof Rule

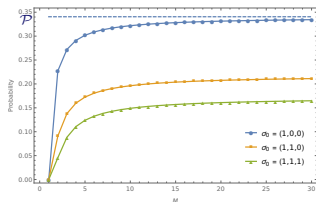
$$\frac{\varphi' \Rightarrow \varphi \quad l \preceq \text{wp}[\![C_{\text{loop}}]\!]([\neg\varphi] \cdot f)}{l \preceq \text{wp}[\![C_{\text{loop}}]\!](f)} \quad (\text{Guard-Strengthening})$$

- **Automatable**: reducible to *probabilistic model checking* for finite-state C_{loop} :

while $(x \neq 0 \vee y \neq 0 \vee z \neq 0) \{$
 $x := x - 1 \oplus x := x + 1 \oplus y := y - 1 \oplus y := y + 1 \oplus z := z - 1 \oplus z := z + 1 \}$



©StackExchange



$$\mathcal{P} = 1 - \left(\frac{3}{(2\pi)^3} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} \frac{dx \, dy \, dz}{3 - \cos x - \cos y - \cos z} \right)^{-1} = 0.3405373296 \dots$$

A “Real” Application : Zeroconf Protocol [Bohnenkamp et al. 2003]

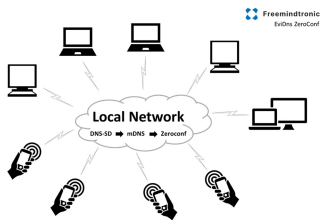
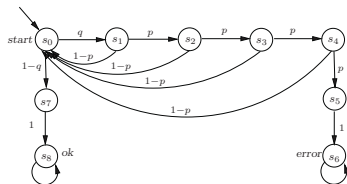


Figure – Self-configuring IP network.



©[Baier & Katoen 2008]

Figure – Markov-chain snippet ($N = 4$).

```

 $C_{zc}$ :   $start = 1 \ ; \ established = 0 \ ; \ probe = 0 \ ;$ 
        while (  $start \leq 1 \wedge established \leq 0 \wedge probe < N \wedge N \geq 4$  ) {
            if (  $start = 1$  ) {
                {  $start := 0$  } [0.5] {  $start := 0 \ ; \ established := 1$  } }
            else { {  $probe := probe + 1$  } [0.001] {  $start := 1 \ ; \ probe := 0$  } } }

```

A “Real” Application : Zeroconf Protocol [Bohnenkamp et al. 2003]

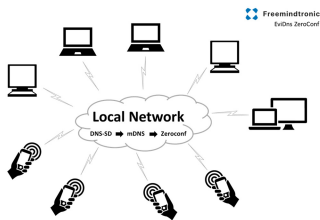
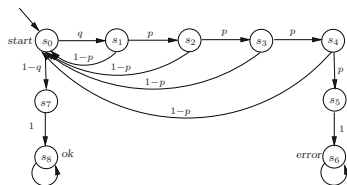


Figure – Self-configuring IP network.



©[Baier & Katoen 2008]

Figure – Markov-chain snippet ($N = 4$).

```

 $C_{zc}$ :   $start = 1$  ;  $established = 0$  ;  $probe = 0$  ;
        while (  $start \leq 1 \wedge established \leq 0 \wedge probe < N \wedge N \geq 4$  ) {
            if (  $start = 1$  ) {
                {  $start := 0$  } [0.5] {  $start := 0$  ;  $established := 1$  }
            }
            else { {  $probe := probe + 1$  } [0.001] {  $start := 1$  ;  $probe := 0$  } }
        }
```

$\Pr(\text{“starting within the loop guard, } C_{zc} \text{ terminates with } established = 1\text{”}) \geq 0.9999999999$

[Bohnenkamp et al. 2003]



Figure – Self-configuring IP network.



Figure – Markov-chain snippet ($N = 4$).

```

CZC:  start = 1; established = 0; probe = 0;
       while ( start ≤ 1 ∧ established ≤ 0 ∧ probe < N ∧ N ≥ 4 ) {
         if ( start = 1 ) {
           { start := 0 } [0.5] { start := 0; established := 1 } }
         else { { probe := probe + 1 } [0.001] { start := 1; probe := 0 } } }

```

$$\Pr(\text{"starting within the loop guard, } C_{zc} \text{ terminates with } \textit{established} = 1") \stackrel{?}{>} 0.999999999999$$

A "Real" Application : Zeroconf Protocol [Bohnenkamp et al. 2003]

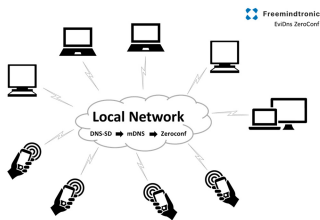
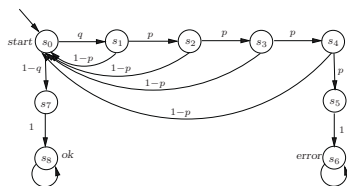


Figure – Self-configuring IP network.



©[Baier & Katoen 2008]

Figure – Markov-chain snippet ($N = 4$).

```

CZC:  start = 1; established = 0; probe = 0;
while ( start ≤ 1 ∧ established ≤ 0 ∧ probe < N ∧ N ≥ 4 ∧ N ≤ 10 ) {
  if ( start = 1 ) {
    { start := 0 } [0.5] { start := 0; established := 1 } }
  else { { probe := probe + 1 } [0.001] { start := 1; probe := 0 } } }

```

$\Pr(\text{"starting within the loop guard, } C_{zc} \text{ terminates with } \textit{established} = 1") > 0.9999999999999999$ ✓

Summary

$$\frac{\varphi' \Rightarrow \varphi \quad l \preceq \text{wp}[\![C'_{\text{loop}}]\!]([\neg\varphi] \cdot f)}{l \preceq \text{wp}[\![C_{\text{loop}}]\!](f)} \quad (\text{Guard-Strengthening})$$

- a new lower bound rule based on **wp-difference** and **guard-strengthening**;

⇒ Feng, Chen, Su, Kaminski, Katoen, Zhan : *Lower Bounds for Poss. Divergent Prob. Prog.* OOPSLA '23.



Summary

$$\frac{\varphi' \Rightarrow \varphi \quad l \preceq \text{wp}[\![C'_{\text{loop}}]\!]([\neg\varphi] \cdot f)}{l \preceq \text{wp}[\![C_{\text{loop}}]\!](f)} \quad (\text{Guard-Strengthening})$$

- a new lower bound rule based on **wp-difference** and **guard-strengthening**;
- first lower bound rule admitting **divergent loops** with **unbounded expectations**;

⇒ Feng, Chen, Su, Kaminski, Katoen, Zhan : *Lower Bounds for Poss. Divergent Prob. Prog.* OOPSLA '23.



Summary

$$\frac{\varphi' \Rightarrow \varphi \quad l \preceq \text{wp}[\![C'_{\text{loop}}]\!]([\neg\varphi] \cdot f)}{l \preceq \text{wp}[\![C_{\text{loop}}]\!](f)} \quad (\text{Guard-Strengthening})$$

- a new lower bound rule based on **wp-difference** and **guard-strengthening**;
- first lower bound rule admitting **divergent loops** with **unbounded expectations**;
- tight lower bounds for **3-D random walks on $\mathbb{Z}^3$** and the **Zeroconf** protocol.

⇒ Feng, Chen, Su, Kaminski, Katoen, Zhan : *Lower Bounds for Poss. Divergent Prob. Prog.* OOPSLA '23.



Summary

$$\frac{\varphi' \Rightarrow \varphi \quad l \preceq \text{wp}[\![C_{\text{loop}}]\!]([\neg\varphi] \cdot f)}{l \preceq \text{wp}[\![C_{\text{loop}}]\!](f)} \quad (\text{Guard-Strengthening})$$

- a new lower bound rule based on **wp-difference** and **guard-strengthening**;
- first lower bound rule admitting **divergent loops** with **unbounded expectations**;
- tight lower bounds for **3-D random walks on $\mathbb{Z}^3$** and the **Zeroconf protocol**.

More in the paper :

- how to **find a “good” strengthening** $\varphi' \Rightarrow \varphi$?
- how to **generate a non-trivial lower bound** for C_{loop} ?
- corner cases where guard strengthening is **insufficient**;
- ...



⇒ Feng, Chen, Su, Kaminski, Katoen, Zhan : *Lower Bounds for Poss. Divergent Prob. Prog.* OOPSLA '23.

