On Proving Almost-Sure Termination

Joost-Pieter Katoen

Talk 2019 Meeting IFIP WG 2.2, Vienna

Termination of programs that roll dice?

Certain termination

```
while (i > 0) { i-- }
```

This program never diverges.
This holds for all integer inputs i.

Almost-sure termination

For $0 < p < 1$ an arbitrary probability:

```
bool c := true;
int i := 0;
while (c) {
    i++;
    (c := false [p] c := true)
}
```

This program does not always terminate.
It diverges with probability zero.
It almost surely terminates.

Non almost-sure termination

```
P :: skip [1/2] { call P; call P; call P }
```

This program terminates with probability $\frac{\sqrt{5}-1}{2} < 1$.

$x = \frac{1}{2} \cdot 1 + \frac{1}{2} x x x$

Nuances of termination

Olivier Bournez

Florent Garnier

. . . . . . certain termination

. . . . . . termination with probability one

$\Longrightarrow$ almost-sure termination

. . . . . . in an expected finite number of steps

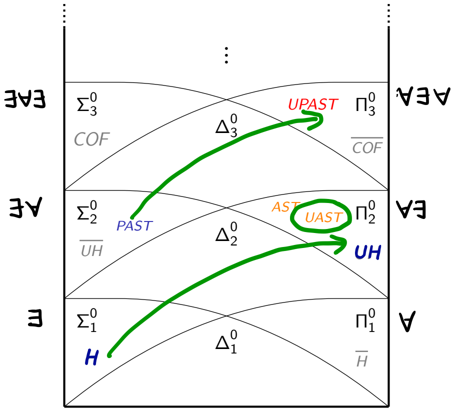
$\Longrightarrow$ "positive" almost-sure termination

. . . . . . in an expected infinite number of steps

$\Longrightarrow$ "null" almost-sure termination

Hardness of almost sure termination

[Kaminski & JPK, 2015]



Adding non-determinism does not change the picture.
Neither for approximating termination probabilities.

Proving almost-sure termination

- What?
  - Termination with probability one
  - For all possible inputs
- Why?
  - Reachability can be encoded as termination
  - Often a prerequisite for proving correctness
  - Often implicitly assumed
- Why is it hard in practice?
  - Requires proving lower bound 1 for termination probability

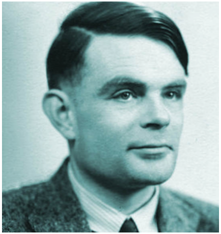
Almost-sure termination

Javier Esparza
CAV 2012

"[Ordinary] termination is a purely topological property [...], but almost-sure termination is not. [...] Proving almost–sure termination requires arithmetic reasoning not offered by termination provers."

How to prove termination?

Use a variant function on the program's state space
whose value — on each loop iteration — is monotonically decreasing
with respect to a (strict) well-founded relation.



Alan Mathison Turing
Checking a large routine
1949

Variant functions

$V : \Sigma \to \mathbb{R}_{\geq 0}$ for loop while$(G)\,P$ is variant function if every state s:

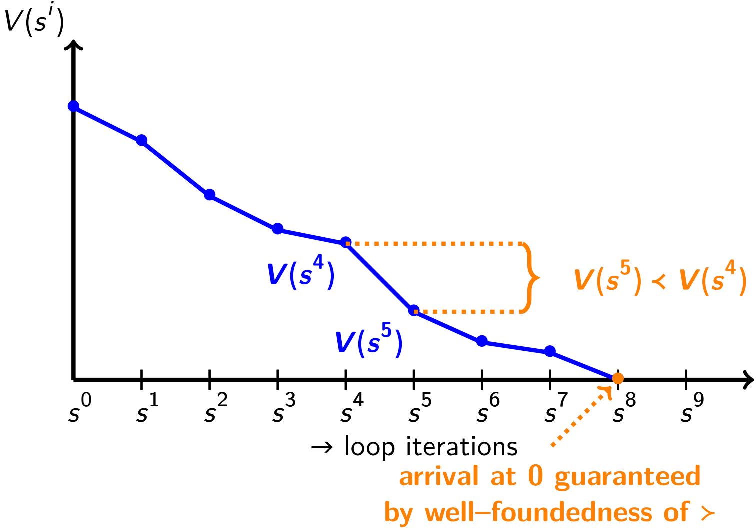
1. If $s \models G$, then P's execution on s terminates in a state t with:

$$V(t) \;\leq\; V(s) - \varepsilon \quad \text{for some fixed } \varepsilon > 0, \text{ and}$$

2. If $V(s) \leq 0$, then $s \not\models G$.

$(\mathbb{R}_{\geq 0}, <_\varepsilon)$ for $\varepsilon > 0$ is well-founded

Termination proofs



Examples

```
while (x > 0) { x-- }
```

Ranking function $V = x$.

```
x := ... ; y := ... // x and y are positive
while (x != y) {
   if (x > y) { x := x-y } else { y := y-x }
}
```

Ranking function $V = x + y$.

Proving almost-sure termination so far

Hart/Sharir/Pnueli: Termination of Probabilistic Concurrent Programs. POPL 1982

Bournez/Garnier: Proving Positive Almost-Sure Termination. RTA 2005

McIver/Morgan: Abstraction, Refinement and Proof for Probabilistic Systems. 2005

Esparza *et al.*: Proving Termination of Probabilistic Programs Using Patterns. CAV 2012

Chakarov/Sankaranarayanan: Probabilistic Program Analysis w. Martingales. CAV 2013

Fioriti/Hermanns: Probabilistic Termination: Soundness, Completeness, and Compositionality. POPL 2015

Chatterjee *et al.*: Algorithmic Termination of Affine Probabilistic Programs. POPL 2016

Agrawal/Chatterjee/Novotný: Lexicographic Ranking Supermartingales. POPL 2018

. . . . . .

Key ingredient: super- (or some form of) martingales

On super-martingales

A stochastic process $X_1, X_2, \ldots$ is a martingale whenever:

$$\mathbb{E}(X_{n+1} \mid X_1, \ldots, X_n) \;=\; X_n$$

It is a super-martingale whenever:

$$\mathbb{E}(X_{n+1} \mid X_1, \ldots, X_n) \;\leq\; X_n$$

Our aim

A powerful, simple proof rule for almost-sure termination.

At the source code level.

No “descend” into the underlying probabilistic model.

No severe restrictions on programs.

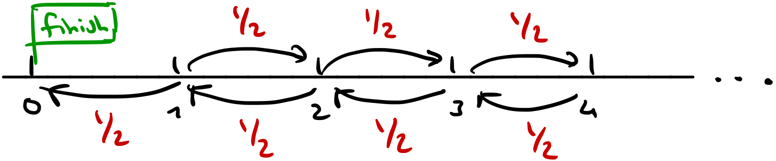
Proving almost-sure termination

$V = x$

$\exists \varepsilon > 0 .\ \mathbb{E}(V^{k+1}) \leq V^{k} - \varepsilon$

The symmetric random walk:

```
while (x > 0) { x := x-1 [0.5] x := x+1 }
```



Proving almost-sure termination

The symmetric random walk:

```
while (x > 0) { x := x-1 [0.5] x := x+1 }
```

Is out-of-reach for many proof rules.

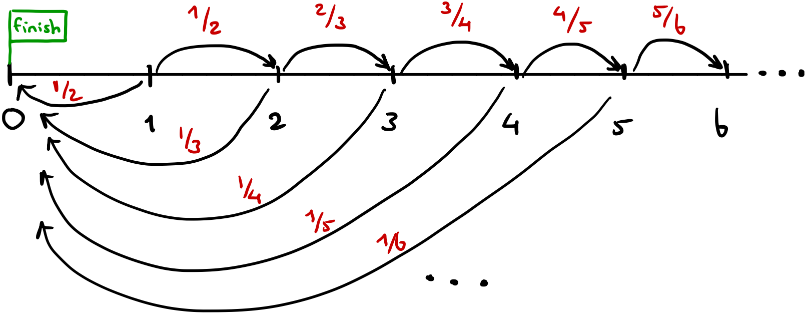
A loop iteration decreases x by one with probability $1/2$

This observation is enough to witness almost-sure termination!

Are these programs almost surely terminating?

- Escaping spline:

```
while (x > 0) { p := 1/(x+1); x := 0 [p] x++}
```



Are these programs almost surely terminating?

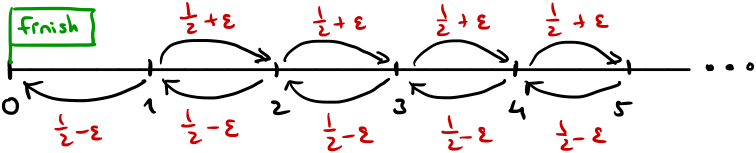
- Escaping spline:

```
while (x > 0) { p := 1/(x+1); x := 0 [p] x++}
```

✓

- A slightly unbiased random walk:

```
p := 0.5-eps ; while (x > 0) { x--  [p] x++ }
```



Are these programs almost surely terminating?

- Escaping spline:

```
while (x > 0) { p := 1/(x+1); x := 0 [p] x++}
```

✓

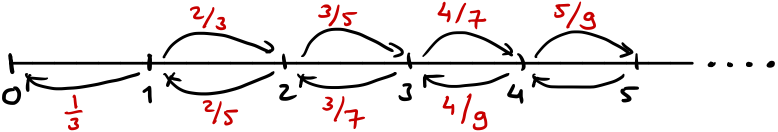
- A slightly unbiased random walk:

```
p := 0.5-eps ; while (x > 0) { x-- [p] x++ }
```

✗

- A symmetric-in-the-limit random walk:

```
while (x > 0) { p := x/(2*x+1) ; x-- [p] x++ }
```



Proving almost-sure termination

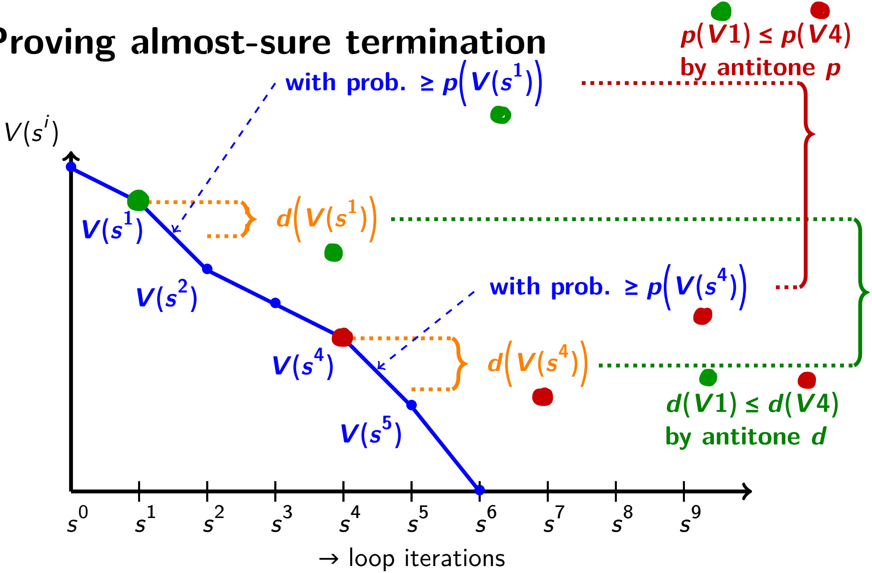
Goal: prove a.s.–termination of `while(G) P`, for all inputs

Ingredients:

- A supermartingale V mapping states onto non-negative reals
  - $\mathbb{E}\{V(s_{n+1}) \mid V(s_0), \ldots, V(s_n)\} \leq V(s_n)$
  - Running body `P` on state $s \models$ `G` does not increase $\mathbb{E}(V(s))$
  - Loop iteration ceases if $V(s) = 0$
- ...... and a progress condition: on each loop iteration in s^i
  - $V(s^i) = v$ decreases by $\geq d(v) > 0$ with probability $\geq p(v) > 0$
  - with antitone p ("probability") and d ("decrease") on V's values

Then: **`while(G) P` a.s.-terminates on every input**

Proving almost-sure termination



The closer to termination, the more V decreases and this becomes more likely

The symmetric random walk

- Recall:

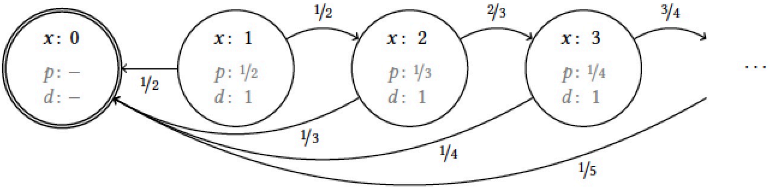
```
while (x > 0) { x := x-1 [0.5] x := x+1 }
```

+2

- Witnesses of almost-sure termination:
  - $V = x$
  - $p(v) = 1/2$ and $d(v) = 1$

That's all you need to prove almost-sure termination!

The escaping spline

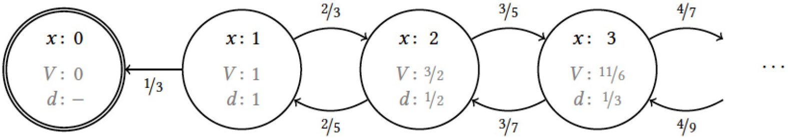


- Consider the program:

```
while (x > 0) { p := 1/(x+1); x := 0 [p] x++}
```

- Witnesses of almost-sure termination:
  - $V = x$
  - $p(v) = \frac{1}{v+1}$ and $d(v) = 1$

A symmetric-in-the-limit random walk



- Consider the program:

```
while (x > 0) { p := x/(2*x+1) ; x-- [p] x++ }
```

V = ln(x)

- Witnesses of almost-sure termination:
  - $V = H_x$, where H_x is x-th Harmonic number $1 + 1/2 + \ldots + 1/x$
  - $p(v) = 1/3$ and $d(v) = \begin{cases} 1/x & \text{if } v > 0 \text{ and } H_{x-1} < v \leq H_x \\ 1 & \text{if } v = 0 \end{cases}$

Formal proof rule

Let I be a predicate, variant function $V : \Sigma \to \mathbb{R}_{\geq 0}$, probability function $p : \mathbb{R}_{\geq 0} \to (0, 1]$ be antitone, decrease function $d : \mathbb{R}_{\geq 0} \to \mathbb{R}_{>0}$ be antitone. If:

1. $[I]$ is a wp-subinvariant of $\texttt{while}(G)\, P$ w.r.t. $[I]$
2. V is a super-invariant of $\texttt{while}(G)\, P$ w.r.t. V
3. $V = 0$ indicates termination, i.e. $[\neg G] = [V = 0]$
4. V satisfies the progress condition:

$$p \circ (V \cdot [G] \cdot [I]) \;\leq\; \lambda s.\, wp(P, \big[V \leq V(s) - d\,(V(s))\big])(s)$$

Then: the loop $\texttt{while}(G)\, P$ terminates from any state s with $s \models I$, i.e.,

$$[I] \;\leq\; wp(\texttt{while}(G)\, P, \mathbf{1})\ .$$

Some remarks

Checking if V, p and d satisfy the sufficient conditions is simple.

This proof rule covers many a.s.-terminating programs
that are out-of-reach for many existing proof rules

The proof rule is applicable to program with nondeterminism too

Questions and discussion

- Are/can similar proof techniques be used elsewhere?
- Completeness? For a certain set of programs?
- Synthesis of functions V, p, and d?
- Complexity issues
- PAST is harder than AST, but AST seems more difficult. Why?
- Automation?

Common knowledge

- A program either terminates or not (on a given input)
- Terminating programs have a finite run-time
- Having a finite run-time is compositional

$$\left.\begin{array}{l} rt(P) < \infty \\ rt(Q) < \infty \end{array}\right\} \quad rt(P;Q) < \infty$$

A radical change

- A program either terminates or not (on a given input)
- Terminating programs have a finite run-time
- Having a finite run-time is compositional

All these facts do not hold for probabilistic programs!

Epilogue

Take-home messages

- Flavours of termination for probabilistic programs
- Positive almost-sure termination is difficult
- A powerful proof rule for almost-sure termination

Extensions

- Expected run-times
- Non-determinism
- Conditioning
- Pointer programs

A big thanks to my co-authors!

Benjamin Kaminski, Christoph Matheja,

Annabelle McIver, Carroll Morgan

Federico Olmedo

Further reading

- B. Kaminski, JPK, C. Matheja.
  On the hardness of analysing probabilistic programs. MFCS 2015/Acta Inf. 2019.
- B. Kaminski, JPK, C. Matheja, and F. Olmedo.
  Expected run-time analysis of probabilistic programs. ESOP 2016/J. ACM 2018.
- A. McIver, C. Morgan, B. Kaminski, JPK.
  A new proof rule for almost-sure termination. POPL 2018.
- M. Hark, B. Kaminski, J. Giesl, JPK.
  Aiming low is harder: Induction for lower bounds in probabilistic program verification. POPL 2020?