Probabilistic Programming

Lecture #14: Proving Almost-Sure Termination

Joost-Pieter Katoen

Software Modeling and Verification Chair | RWTH AACHEN UNIVERSITY

RWTH Lecture Series on Probabilistic Programming 2018

Overview

Overview

Proving almost-sure termination

- What? Termination with probability one.
- Why?
  - Termination is an elementary liveness property
  - Reachability can be encoded as termination
  - Often a prerequisite for proving correctness
- Why is it hard in practice?
  - Requires proving lower bound 1 for termination probability
  - Lower bounds (AST) are harder to prove than upper bounds (positive AST)
  - This is especially true for null-terminating programs

Our aim

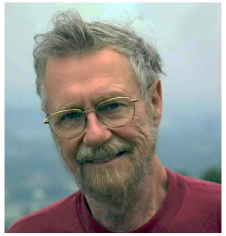
A powerful proof rule at the source code level.

No "descend" into the underlying probabilistic model.

Overview

Termination by weakest preconditions

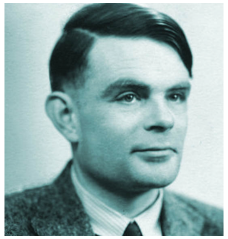
Determine $wp(P, \text{true})$ for program P and postcondition true.



Edsger Wybe Dijkstra
A Discipline of Programming
1976

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

Overview

Well-founded relation

Well-founded relation

Let $(D, \sqsubset)$ be a strict partial order. The relation $\sqsubset$ is well-founded if there is no infinite sequence $d_1, d_2, d_3, \ldots$ with $d_i \in D$ such that $d_i \sqsubset d_{i+1}$ for all $i \in \mathbb{N}$.

Examples

- $(\mathbb{N}, <)$
- $(\mathbb{R}^+, <_\varepsilon)$ for $\varepsilon > 0$ where $x <_\varepsilon y$ iff $x \leq y - \varepsilon$
- $(\mathbb{L}, <)$ for lists $\mathbb{L}$ where $\ell_1 < \ell_2$ iff $|\ell_1| < |\ell_2|$.

A relation $\sqsubset$ is Noetherian on D, if the converse relation $\sqsupset$ is well-founded on D.

A Noetherian relation is also called terminating.

Variant functions

Variant function

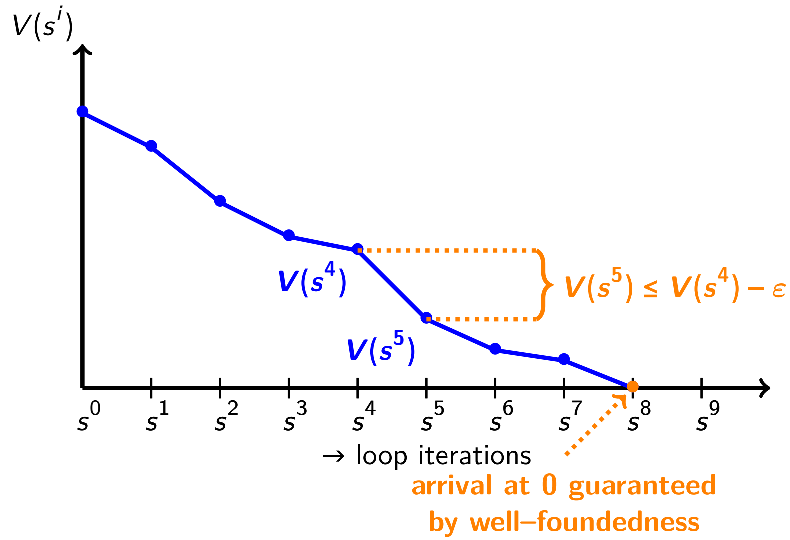
A variant (aka: ranking) function $V : \mathbb{S} \to \mathbb{R}$ for GCL-loop `while`$(G)\, P$ is a function that satisfies for every $s \in \mathbb{S}$:

1. If $s \models G$, then the execution of P 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 $s \not\models G$.

Variant (aka: ranking) functions



Termination

Every (universally) terminating loop `while`$(G)\,P$ has a variant function.

Proof.

(Sketch.)

1. As V is a variant function, from every state $s \models G$, the execution of the loop body P reaches a state t whose ranking is at least by ε smaller than s's ranking, and
2. ensures that if the ranking hits 0 or drops below, this falsifies the loop guard G and thus causes the loop to terminate.

Therefore, from every state s, no infinite chain of successor states with ever decreasing ranking can be formed by iterated execution of the loop body P without eventually falsifying the loop guard G. Since the length of such a chain is bounded by $\lceil V(s)/\varepsilon \rceil$, this ensures certain termination of the loop within at most $\lceil V(s)/\varepsilon \rceil$ loop iterations. □

Examples

```
while (x > 0) { x := x-1 }
```

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 $V = x + y$.

Ranking functions for probabilistic programs

```
while (x > 0) {
    { x := x-1 } [1/2] { skip }
}
```

Ranking function $V = x$ does not guarantee to decrease x.

But every loop iteration decreases x "in expectation".

A proof rule for positive almost-sure termination

Proving positive almost-sure termination **[Chakarov *et al.*, 2013]**

Let `while`$(G)P$ be a loop where P terminates universally certainly (e.g., P is loop-free), and let $I \in \mathbb{E}$ be a ranking super-invariant of the loop w.r.t. expectation $\mathbf{0}$, i.e., $I \leq \infty$ and for some constants ε and K with $0 < \varepsilon < K$ it holds:

$$[\neg G] \cdot I \leq K \quad \text{and} \quad [G] \cdot K \leq [G] \cdot I + [\neg G] \quad \text{and} \quad \Phi(I) \leq [G] \cdot (I - \varepsilon).$$

Then: `while`$(G)P$ terminates universally positively almost surely.

Example

On the black board.

Overview

AST by weakest preconditions

Determine $wp(P, \mathbf{1})$ for program P and postcondition $\mathbf{1}$.

Dexter Kozen
A probabilistic PDL
1983

A zero-one law for termination

Zero-one law for probabilistic termination

Let $I \in \mathbb{P}$ such that $[I]$ is a wp-subinvariant of $\texttt{while}(G)\,P$ with respect to post-expectation $[I]$. Furthermore, let $\varepsilon > 0$ a constant such that:

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

Then:

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

Proof.

On the black board. □

A special case is obtained for invariant I equals true.

A large body of existing works

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:

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

A historical perspective

A countable Markov process is "non-dissipative"
if almost every infinite path eventually enters
— and remains in — positive recurrent states.

A sufficient condition for being non-dissipative is:

$$\sum_{j \geq 0} j \cdot p_{ij} \;\leq\; i \quad \text{for all states } i$$

F. Gordon Foster

Born	24 February 1921 Belfast, United Kingdom
Died	20 December 2010 (aged 89) Dublin, Ireland
Nationality	Irish
Known for	Foster's theorem
Scientific career	
Doctoral advisor	David George Kendall

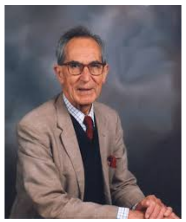
Frederic Gordon Foster
Markoff chains with an enumerable number of states
and a class of cascade processes
1951

Kendall's variation

A Markov process is non-dissipative if for some function $V : \Sigma \to \mathbb{R}$:

$$\sum_{j \geq 0} V(j) \cdot p_{ij} \;\leq\; V(i) \quad \text{for all states } i$$

and for each r there are finitely many states i with $V(i) \leq r$



David George Kendall
On non-dissipative Markoff chains
with an enumerable infinity of states
1951

On positive recurrence

Every irreducible positive recurrent Markov chain is non-dissipative.

A Markov process is positive recurrent iff there is a Lyapunov function
$V : \Sigma \to \mathbb{R}_{\geq 0}$ with for finite $F \subseteq \Sigma$ and $\varepsilon > 0$:

$$\sum_j V(j) \cdot p_{ij} \;<\; \infty \quad \text{for } i \in F, \text{ and}$$
$$\sum_j V(j) \cdot p_{ij} \;<\; V(j) - \varepsilon \quad \text{for } i \notin F.$$

Markov Chains pp 167-193 | Cite as

Lyapunov Functions and Martingales

Authors | Authors and affiliations

Pierre Brémaud

Pierre Brémaud 1999

Frederic Gordon Foster
On the stochastic matrices associated
with certain queuing processes
1953

Proving almost-sure termination

The symmetric random walk:

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

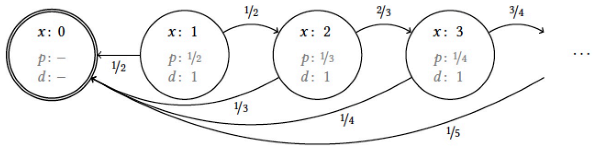
Is out-of-reach for many proof rules.

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

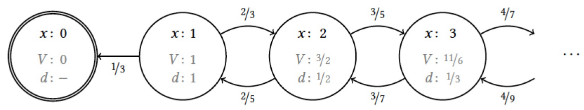
This observation is enough to witness almost-sure termination!

Do these programs almost surely terminate?

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




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



Proving almost-sure termination

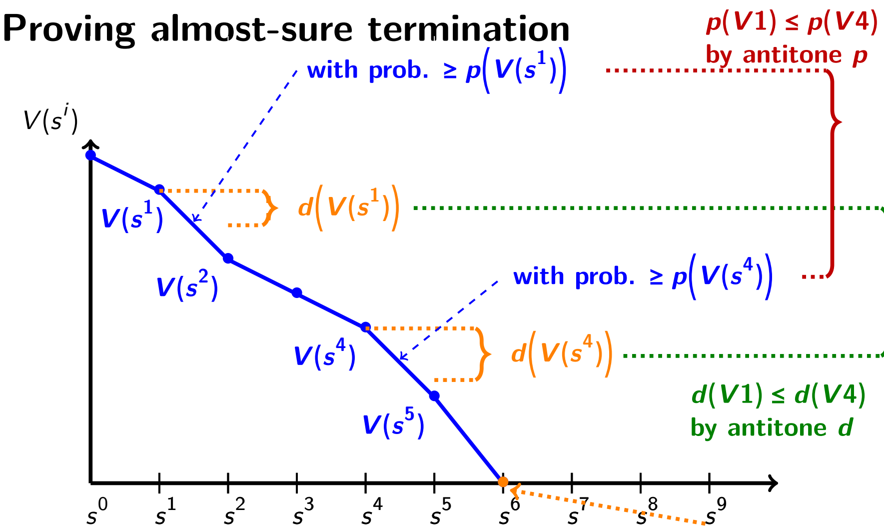
Goal: prove a.s.–termination of `while(G) P`

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 $\geq d(v)$ with probability $\geq p(v)$
  - 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



→ loop iterations a.s. arrival at 0 guaranteed

by our proof rule

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

The formal proof rule for almost-sure termination

Proof rule for almost-sure termination [McIver *et al.*, 2018]

Let $I \in \mathbb{P}$, (variant) function $V : \mathbb{S} \to \mathbb{R}_{\geq 0}$, (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. $V = 0$ indicates termination, i.e. $[\neg G] = [V = 0]$
3. V is a super-invariant of $\texttt{while}(G)\,P$ w.r.t. V
4. V satisfies the progress condition:

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

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

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

The symmetric random walk

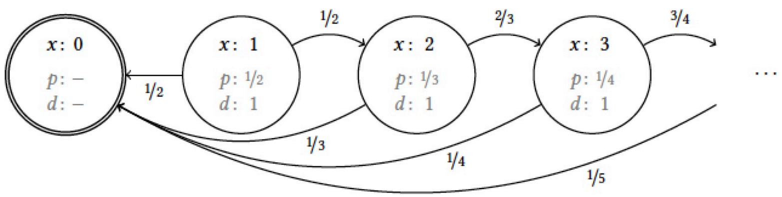
- Recall:

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

- 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

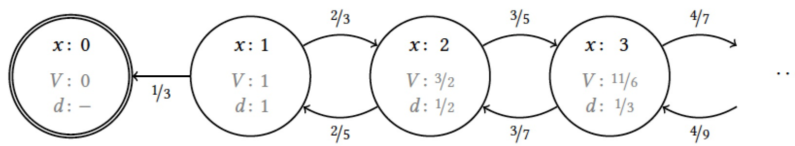


- Consider the program:

```
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++ }
```

- 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 $d(v) = \begin{cases} 1/x & \text{if } v > 0 \text{ and } H_{x-1} < v \leq H_x \\ 1 & \text{if } v = 0 \end{cases}$

Expressiveness

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