TERMINATION
What we all know about termination

The halting problem
— does a program $P$ terminate on a given input state $s$? — is semi-decidable.

The universal halting problem
— does a program $P$ terminate on all input states? — is undecidable.

Alan Mathison Turing
On computable numbers, with an application to the Entscheidungsproblem
1937
What if programs roll dice?
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

\[ P \land Q \]
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!
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:

```c
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.
Positive almost-sure termination

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

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

This program almost surely terminates. In finite expected time. Despite its possibility of divergence.
Null almost-sure termination

Consider the symmetric one-dimensional random walk:

\begin{verbatim}
int x := 10; while (x > 0) { x-- [1/2] x++ }
\end{verbatim}

This program almost surely terminates.
It requires an infinite expected time to do so.
Non almost-sure termination

\[ x = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot x \cdot x \cdot x \]

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

This program terminates with probability \( \frac{1 + \sqrt{5}}{2} < 1 \).
Nuances of termination

...... certain termination

...... termination with probability one

⇒ almost-sure termination

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

⇒ “positive” almost-sure termination

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

⇒ “null” almost-sure termination
Three contributions

The hardness of the various notions of termination.
[MFCS 2015, Acta Informatica 2019]

A powerful proof rule for almost-sure termination.
[POPL 2018]

A weakest-precondition calculus for positive almost-sure termination.
[ESOP 2016, J. ACM 2018]
Hardness of termination

It is a known fact that deciding termination of ordinary programs is undecidable.

Our aim is to classify “how undecidable” (positive) almost-sure termination is.
Kleene and Mostovski

Stephen Kleene (1909–1994)

Andrzej Mostovski (1913–1975)
The Kleene-Mostovski hierarchy

The following inclusion diagram holds (all inclusions are strict):

\[
\begin{align*}
\Sigma_1^0 & \subseteq \Pi_1^0 \\
\Delta_2^0 & \subseteq \Pi_2^0 \\
\Sigma_2^0 & \subseteq \Pi_2^0 \\
\Sigma_3^0 & \subseteq \Pi_3^0
\end{align*}
\]
Hardness of almost sure termination
Proof idea: hardness of positive as-termination

\[ \overline{\text{UH}} \xrightarrow{\text{PAST}} \]

Reduction from the complement of the universal halting problem

For an ordinary program \( Q \), provide a probabilistic program \( P \) (depending on \( Q \)) and an input \( s \), such that

\( P \) terminates in a finite expected number of steps on \( s \) if and only if

\( Q \) does not terminate on some input
Let’s start simple

```c
bool c := true;
int nrflips := 0;
while (c) {
    nrflips++;
    (c := false [0.5] c := true);
}
```

Expected runtime (integral over the bars):

The $nrflips$-th iteration takes place with probability $\frac{1}{2^{nrflips}}$. 
Reducing an ordinary program to a probabilistic one

Assume an enumeration of all inputs for $Q$ is given

```c
bool c := true;
int nrflips := 0;
int i := 0;
while (c) {
    // simulate $Q$ for one (further) step on its $i$-th input
    if (Q terminates) {
        // take $2^{nrflips}$ effectless steps
        i++;
        // reset simulation of program $Q$
    }
    nrflips++;
    (c := false [0.5] c := true);
}
```

$P$ looses interest in further simulating $Q$ by a coin flip to decide for termination.
Q does not always halt

Let $i$ be the first input for which $Q$ does not terminate.

Expected runtime of $P$ (integral over the bars):

Finite **cheering** — finite expected runtime
$Q$ terminates on all inputs

Expected runtime of $P$ (integral over the bars):

Infinite cheering — infinite expected runtime
Hardness of almost sure termination

No change for non-deterministic probabilistic programs. No change when approximating termination probabilities.
Proving almost-sure termination

- **What?** Termination with probability one.

- **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

"[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."

Javier Esparza
CAV 2012
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 (aka: ranking) functions

$V : \Sigma \rightarrow \mathbb{R}_{\geq 0}$ 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:

   \[ V(t) \leq V(s) - \varepsilon \]

   for some fixed $\varepsilon > 0$, and

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

\[ V(s^i) \]

\[ \rightarrow \text{loop iterations} \]
Termination proofs

arrival at 0 guaranteed by well-foundedness of $\succ$
Examples

\[
\text{while } (x > 0) \{ x-- \}
\]

Ranking function \( V = x \).

\[
x := \ldots \ ; \ y := \ldots \quad // \ x \text{ and } y \text{ are positive}
\]
\[
\text{while } (x \neq y) \{
\quad \text{if } (x > y) \{ x := x-y \} \quad \text{else } \{ y := y-x \}
\}
\]

Ranking function \( V = x + y \).
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 $\varepsilon > 0$:

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

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

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

At the source code level.

No “descend” into the underlying probabilistic model.
Proving almost-sure termination

The symmetric random walk:

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

\[
\forall n \in \mathbb{N}, \mathbb{E}(V_{n+1}) < V_n - \varepsilon
\]

for some positive $\varepsilon$
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.
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 \(1/2\).
Are these programs almost surely terminating?

- Escaping spline:

```plaintext
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-\text{eps} ; \textbf{while} (x > 0) \{ x--1 [p] x++ \} \text{ X}
  ```
Are these programs almost surely terminating?

- Escaping spline:
  \[
  \text{while } (x > 0) \{ \ p := 1/(x+1); \ x := 0 \ [p] \ x++ \}
  \]

- A slightly unbiased random walk:
  \[
  0.5-\text{eps} ; \ \text{while } (x > 0) \{ \ x--1 \ [p] \ x++ \}
  \]

- A symmetric-in-the-limit random walk:
  \[
  \text{while } (x > 0) \{ \ p := x/(2*x+1) ; \ x-- \ [p] \ x++ \}
  \]
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
  ► $\mathbb{E}\{V(s_{n+1}) | V(s_0), \ldots, V(s_n)\} \leq V(s_n)$
  ► 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 $s^i$
  ► $V(s^i) = \nu$ decreases by $\geq d(\nu) > 0$ with probability $\geq p(\nu) > 0$
  ► 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

\[ V(s^i) \]

with prob. \( \geq p(V(s^1)) \)

\( V\approx x \)

\( p = \frac{1}{2} \)

\( d = 1 \)

\( \rightarrow \) loop iterations

Joost-Pieter Katoen
Principles of Probabilistic Programming
Proving almost-sure termination

\[ V(s^i) \]

with prob. \( \geq p(V(s^1)) \)

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

Joost-Pieter Katoen
Principles of Probabilistic Programming
Proving almost-sure termination

with prob. $\geq p(V(s^1))$

$V(s^i)$

$V(s^1)$ $\Rightarrow$ $d(V(s^1))$

$V(s^2)$ $\Rightarrow$ $d(V(s^2))$

$V(s^4)$ $\Rightarrow$ $V(s^4)$

$V(s^5)$ $\Rightarrow$ $d(V(s^5))$

$s^0$ $s^1$ $s^2$ $s^3$ $s^4$ $s^5$ $s^6$ $s^7$ $s^8$ $s^9$

$\rightarrow$ loop iterations
Proving almost-sure termination

$V(s^i)$

$V(s^1) \rightarrow V(s^2) \rightarrow V(s^4) \rightarrow V(s^5)$

$d(V(s^1))$ with prob. $\geq p(V(s^1))$

$d(V(s^4))$ with prob. $\geq p(V(s^4))$

$d(V1) \leq d(V4)$ by antitone $d$

$s^0 \rightarrow s^1 \rightarrow s^2 \rightarrow s^3 \rightarrow s^4 \rightarrow s^5 \rightarrow s^6 \rightarrow s^7 \rightarrow s^8 \rightarrow s^9$

→ loop iterations

Joost-Pieter Katoen

Principles of Probabilistic Programming

142/222
Proving almost-sure termination

\[ V(s^i) \]

with prob. \( \geq p(V(s^1)) \)

with prob. \( \geq p(V(s^4)) \)

\[ d(V(s^1)) \]

\[ d(V(s^4)) \]

\[ p(V1) \leq p(V4) \]

by antitone \( p \)

\[ d(V1) \leq d(V4) \]

by antitone \( d \)

a.s. arrival at 0 guaranteed by our proof rule

The closer to termination, the more \( V \) decreases and this becomes more likely.
Proving almost-sure termination

The closer to termination, the more $V$ decreases and this becomes more likely by antitone $p$.

$a.s.$ arrival at 0 guaranteed by our proof rule.

d($V_1$) ≤ d($V_4$) by antitone $d$.

with prob. ≥ $p(V(s_4))$ by antitone $p$.

with prob. ≥ $p(V(s_1))$.
The symmetric random walk

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

- Witness of almost-sure termination:
  - \( V = x \)
  - \( p(v) = \frac{1}{2} \) and \( d(v) = 1 \)

  That’s all you need to prove almost-sure termination!
The escaping spline

Consider the program:

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

Witness 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:

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

Witness of almost-sure termination:

- \( V = H_x \), where \( H_x \) is the \( x \)-th Harmonic number \( 1 + \frac{1}{2} + \ldots + \frac{1}{x} \)
- \( p(v) = \frac{1}{3} \) and \( d(v) = \begin{cases} \frac{1}{x} & \text{if } v > 0 \text{ and } H_{x-1} < v \leq H_x \\ 1 & \text{if } v = 0 \end{cases} \)