Probabilistic Programming

Lecture #13: Hardness of Almost-Sure Termination

Joost-Pieter Katoen

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

RWTH Lecture Series on Probabilistic Programming 2018

Overview

1. Motivation
2. Nuances of termination
3. Hardness of almost-sure termination — Π_2-complete
4. Hardness of positive almost-sure termination — Π_3 complete

Overview

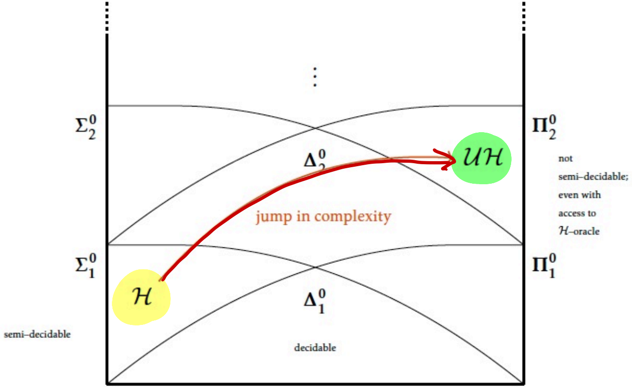
What we all know about termination

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

Complexity jump for termination



What if programs roll dice?

A radical change

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

$$\begin{array}{l} P \in UH \\ Q \in UH \end{array} \implies P\,;\,Q \in UH$$

A radical change

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

All these facts do not hold for probabilistic programs!

Overview

Certain termination

```
i := 100; while (i > 0) { i-- }
```

This program certainly terminates.

Almost-sure termination

$P \notin H$

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

P_i

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

This program does not always terminate. It almost surely terminates.

$\hookrightarrow P \overset{1-p}{\rightsquigarrow} c := true \overset{1-p}{\rightsquigarrow} c := true \rightsquigarrow \ldots\ldots \overset{1-p}{\rightsquigarrow} c := true$

Almost-sure termination

Do the following programs almost surely terminate?

(1)

```
P := (skip [0.5] call P)
```

(2)

```
P := (skip [0.5] call P; call P)
```

$t_p = \frac{1}{2} \cdot 1 + \frac{1}{2} t_p \times t_p = t_p = \frac{1}{2} + \frac{1}{2} t_p^2$

(3)

```
P := (skip [0.5] call P; call P; call P)
```

$t_p = \frac{1}{2} \cdot 1 + \frac{1}{2} t_p^3 \longrightarrow t_p = \frac{1-\sqrt{5}}{2}$

Positive almost-sure termination

P :

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

Geom(p)

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

$Pr\{i = N\} \sim Geom(p)$

This program almost surely terminates. In finite expected time.

Despite its possibility of divergence.

$\frac{1}{p}$ iterations on average

$$ert(P) = 2 + \frac{3}{p} < \infty$$

Null almost-sure termination

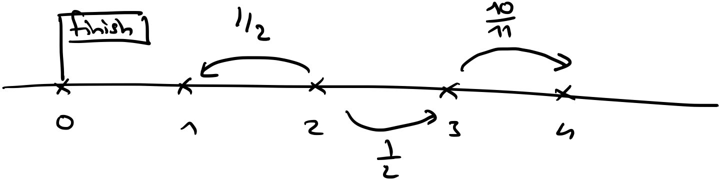
$< \frac{1}{2}$: AST + null

$> \frac{1}{2}$: no longer AST

Consider the one-dimensional (symmetric) random walk:

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

This program almost surely terminates
but requires an infinite expected time to do so.



Sketch

let T_m = first time that the rw visits position m

$$p_{k\ell} = \Pr\{T_\ell < \infty \mid \underbrace{X_0 = k}_{\text{rw starts @ pos } k}\}$$

① claim $p_{12} = 1$. this follows from:

a. $$p_{12} = \underbrace{\tfrac{1}{2}}_{\substack{\text{directly} \\ \text{move } 1\to 2}} + \underbrace{\tfrac{1}{2}\, p_{02}}_{\text{move via } 0}$$

b. $$p_{02} = p_{01} \times p_{12}$$

c. $$p_{01} = p_{12}$$ ("strong Markov property")

$$\underbrace{p_{12}}_{p} = \frac{1}{2} + \frac{1}{2}\, p_{01} \times p_{12} \overset{(c)}{=} \frac{1}{2} + \frac{1}{2}\underbrace{p_{12}^2}_{p}$$

$$p = \tfrac{1}{2} + \tfrac{1}{2} p^2 \iff (p-1)^2 = 0$$

$\underline{2}$ $\underline{\text{claim} \quad m_{12} = \infty}$

$m_{kl} = \mathbb{E}(T_{kl})$

follows from:

a. $m_{12} = \underbrace{\frac{1}{2} \cdot 1}_{1 \to 2} + \underbrace{\frac{1}{2}(1 + m_{02})}_{\text{"via 0"}}$

b. $m_{02} = m_{01} + m_{12}$

c. $m_{01} = m_{12}$

Thus $\underbrace{m_{12}}_{=m} = \frac{1}{2} + \frac{1}{2}(1 + 2\underbrace{m_{12}}_{=m})$

$\Rightarrow \quad m = 1 + m$

$\Rightarrow \quad m = \infty$

Compositionality

Consider the two probabilistic programs:

```
int x := 1;
bool c := true;
while (c) {
    c := false [0.5] c := true;
    x := 2*x
}
```

Finite expected termination time

$\frac{1}{1/2}$ = 2 iterations on average

Compositionality

Consider the two probabilistic programs:

```
int x := 1;
bool c := true;
while (c) {
    c := false [0.5] c := true;
    x := 2*x
}
```

$\underbrace{\text{Finite expected termination time}}_{P}$

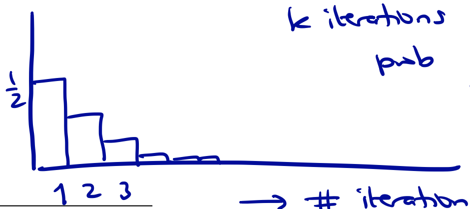
```
while (x > 0) {
    x--
}
```

Finite termination time

$\underbrace{\text{certainly terminate}}_{Q}$

P ; Q

Compositionality



k iterations with prob $\frac{1}{2^k}$

→ # iterations

P:

$$wp(P, x) = \sum_{k=0}^{\infty} 2^k \cdot \frac{1}{2^k} = \infty$$

P

```
int x := 1;
bool c := true;
while (c) {
    c := false [0.5] c := true;
    x := 2*x
}
```

Finite expected termination time

Q

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

Finite termination time

Running the right after the left program
yields an infinite expected termination time

P ; Q

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

Overview

1. Motivation
2. Nuances of termination
3. Hardness of almost-sure termination
4. Hardness of positive almost-sure termination

$Pr = 1$ — single input state
— for all input states

Computable approximations of such distributions

1. The (sub-)distribution $\llbracket P \rrbracket_s^{=k}$ of pGCL program P over final states on input s after exactly k computation steps is defined by:

$$\llbracket P \rrbracket_s^{=k}(t) = \sum_{\sigma \in \Sigma} q \text{ with } \Sigma = \{ \sigma = \langle \downarrow, t, k, \theta, q \rangle \mid \langle P, s, 0, \varepsilon, 1 \rangle \to^* \sigma \}$$

2. The k-the approximation of the weakest pre-expectation $wp(P, f)$ is defined by:

$$wp(P, f)^{=k}(s) \;=\; \sum_{t \in \Sigma_P} \llbracket P \rrbracket_s^{=k}(t) \cdot f(t)$$

3. The computable weakest pre-expectations are defined by:

$$wp(P, f)(s) \;=\; \sum_{k=0}^{\infty} wp(P, f)^{=k}(s)$$

Almost-sure termination

Similar to the halting H and the universal halting problem UH,
we define the decision problems AST and $UAST$

Almost-sure termination

Similar to the halting H and the universal halting problem UH,
we define the decision problems AST and $UAST$

The decision problems AST and $UAST$

Let P be a pGCL program, $s \in \mathbb{S}$ a variable valuation. Then:

$$
\begin{aligned}
(P, s) \in AST \quad &\text{iff} \quad wp(P, \mathbf{1})(s) = \mathbf{1} \\
P \in UAST \quad &\text{iff} \quad \forall s \in \mathbb{S}.\, (P, s) \in AST
\end{aligned}
$$

P terminates with prob 1
on input s

Almost-sure termination

Similar to the halting H and the universal halting problem UH, we define the decision problems AST and $UAST$

The decision problems AST and $UAST$

Let P be a pGCL program, $s \in \mathbb{S}$ a variable valuation. Then:

$$(P, s) \in AST \quad \text{iff} \quad wp(P, \mathbf{1})(s) = \mathbf{1}$$

$$P \in UAST \quad \text{iff} \quad \forall s \in \mathbb{S}. \, (P, s) \in AST$$

Examples

The geometric distribution program $\in UAST$, one-dimensional symmetric random walk $\in UAST$, one-dimensional asymmetric random walk $\notin UAST$, but for input 0 is in AST.

Hardness of almost-sure termination

$\forall \ldots \exists \ldots$

The decision problems *AST* and *UAST*

Let P be a pGCL program, $s \in \mathbb{S}$ a variable valuation. Then:

$$(P, s) \in AST \quad \text{iff} \quad wp(P, \mathbf{1})(s) = \mathbf{1}$$

$$P \in UAST \quad \text{iff} \quad \forall s \in \mathbb{S}.\, (P, s) \in AST$$

$\underbrace{\forall\forall}_{\forall} \ldots \exists \ldots$

$\forall \ldots \exists \ldots$

Hardness of almost-sure termination

AST and *UAST* are both Π_2-complete.

Proof.

For *AST* on the black board. *UAST*: straightforward from the definition of *UAST* and the fact that *AST* is Π_2-complete. □

AST is Π_2- hard.

$$r : UH \mapsto AST$$

$$r(Q) = (P, s)$$

$\uparrow$

GCL program

$P :=$

$$i := \mathrm{Geom}(\tfrac{1}{2});$$

$$S_Q(\underbrace{g_Q(i)}_{\text{the } i\text{-th input to program } Q})$$

simulate program Q

Correctness $\quad Q \in UH$ iff $(P, s) \in AST$

Interpreting this hardness result

Deciding almost-sure termination of a probabilistic program for a single input

is as hard as

deciding termination of an ordinary program for all inputs

is as hard as

deciding almost-sure termination of a probabilistic program for all inputs.

Overview

The expected run-time of a program

The expected run-time of a program

The expected run-time of pGCL program P on input state s is defined by:

$$ert(P, s) \;=\; \sum_{k=1}^{\infty} \Big(1 - \sum_{\langle \downarrow, \ldots, q \rangle \in \mathbb{C}^{<k}} q \Big)$$

where $\mathbb{C}^{<k}$ is the set of final configurations that can be reached in less than k steps by running P on input state s:

$$\mathbb{C}^{<k} \;=\; \{\, \sigma = \langle \downarrow, t, n, \theta, q \rangle \mid \langle P, s, 0, \varepsilon, 1 \rangle \rightarrow^{*} \sigma \text{ and } n < k \,\}$$

Computable approximations of expected run-times

The expected run-time of a program in k steps

The expected run-time of pGCL program P running on input state s for at most m steps is defined by:

$$ert^{\leq m}(P, s) \;=\; \sum_{k=1}^{m} \Big(1 - \sum_{\langle \downarrow, \ldots, q \rangle \in \mathbb{C}^{<k}} q\Big)$$

where $\mathbb{C}^{<k}$ is the set of final configurations that can be reached in less than k steps by running P on input state s.

It follows that $ert^{\leq m}(P, s)$ is computable[1]

[1] due to the Kleene Normal Form Theorem.

Computable approximations of expected run-times

The expected run-time of a program in k steps

The expected run-time of pGCL program P running on input state s for at most m steps is defined by:

$$ert^{\leq m}(P, s) \;=\; \sum_{k=1}^{m} \Bigl(1 - \sum_{\langle \downarrow, \ldots, q \rangle \in \mathbb{C}^{<k}} q\Bigr)$$

where $\mathbb{C}^{<k}$ is the set of final configurations that can be reached in less than k steps by running P on input state s.

It follows that $ert^{\leq m}(P, s)$ is computable[1]

Moreover, we have: $ert(P, s) \;=\; \sup_{m \in \mathbb{N}} ert^{\leq m}(P, s)$

[1] due to the Kleene Normal Form Theorem.

Positive almost-sure termination

The decision problems *PAST* and *UPAST*

Let P be a pGCL program, $s \in \mathbb{S}$ a variable valuation. Then:

$$\begin{aligned} (P, s) \in \mathit{PAST} \quad & \text{iff} \quad \mathit{ert}(P, s) < \infty \\ P \in \mathit{UPAST} \quad & \text{iff} \quad \forall s \in \mathbb{S}.\, (P, s) \in \mathit{PAST} \end{aligned}$$

It follows that $\mathit{PAST} \subsetneq \mathit{AST}$ and $\mathit{UPAST} \subsetneq \mathit{UAST}$.

Positive almost-sure termination

Hardness of positive almost-sure termination

1. *PAST* is Σ_2-complete.
2. *UPAST* is Π_3-complete.

Proof.

1. *PAST* $\in \Sigma_2$: on black board; Σ_2-hardness: sketch on next slides.
2. See the lecture notes (on the web page).

$\square$

Proof idea: hardness of positive as-termination

Σ_2 -hard

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

$$r : \underbrace{\overline{UH}}_{\Sigma_2\text{- complete}} \mapsto PAST$$

Correctness:

$Q \in \overline{UH}$

iff

$r(Q) \in PAST$

$$\mathrm{PAST} \in \Sigma_2$$

$$(P, s) \in \mathrm{PAST}$$

iff

$$\mathrm{ert}(P, s) < \infty$$

iff

$$\exists c : \mathrm{ert}(P, s) < c$$

iff

$$\exists c : \sup_m \mathrm{ert}^{\leq m}(P, s) \leq c$$

iff

$$\underbrace{\exists c . \forall \ell \ \mathrm{ert}^{\leq \ell}(P, s)}_{\Sigma_2\text{-formula.}} < c$$

$\boxtimes$

Let's start simple

```
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 $1/2^{\texttt{nrflips}}$.

Reducing an ordinary program to a probabilistic one

Assume an enumeration of all inputs for Q is given

$Q \in \overline{UH}$ iff $r(Q) \in PAST$

```
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 on its i-th input) {
        cheer; // take 2^nrflips effectless steps
        i++;
        // reset simulation of program Q
    }
    nrflips++;
    (c := false [0.5] c := true);
}
```

Reducing an ordinary program to a probabilistic one

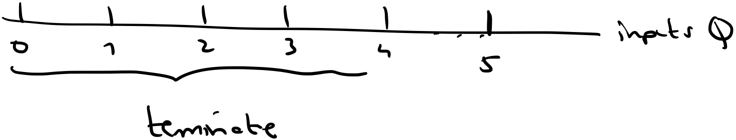
Assume an enumeration of all inputs for Q is given

```
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 on its i-th input) {
        cheer; // 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

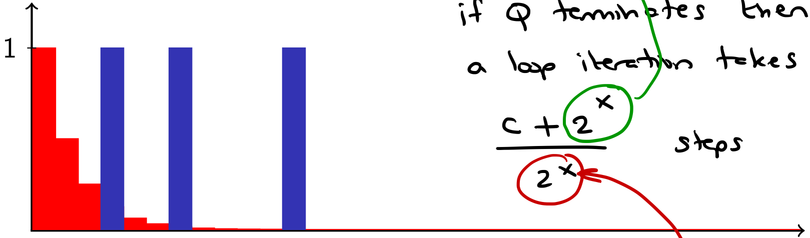
Let i be the first input for which Q does not terminate.



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):



cheering

if Q terminates then a loop iteration takes

$$\frac{c + 2^x}{2^x} \text{ steps}$$

x = nrflips

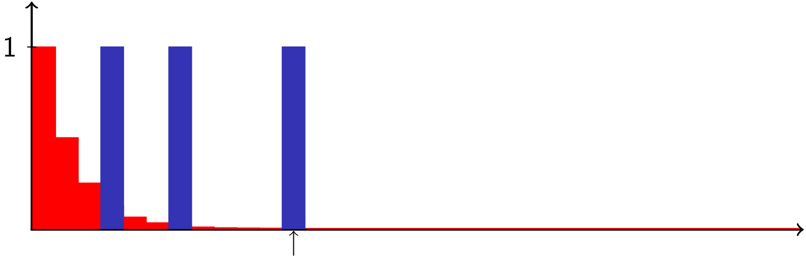
$$\frac{1}{2^x}$$

Q does not always halt

$Q \in \overline{UH}$

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

Expected runtime of P (integral over the bars):

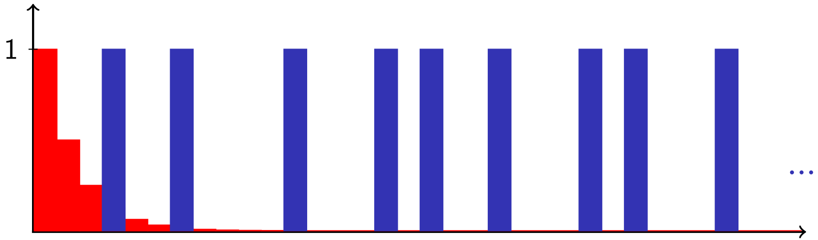


cheering on termination on (i−1)-th input

Finite cheering — finite expected runtime

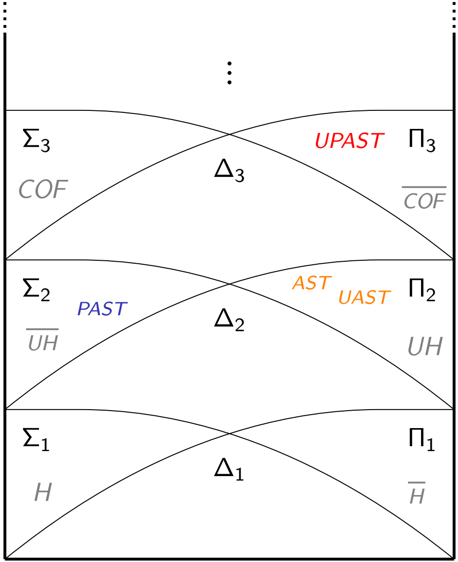
Q terminates on all inputs

Expected runtime of P (integral over the bars):

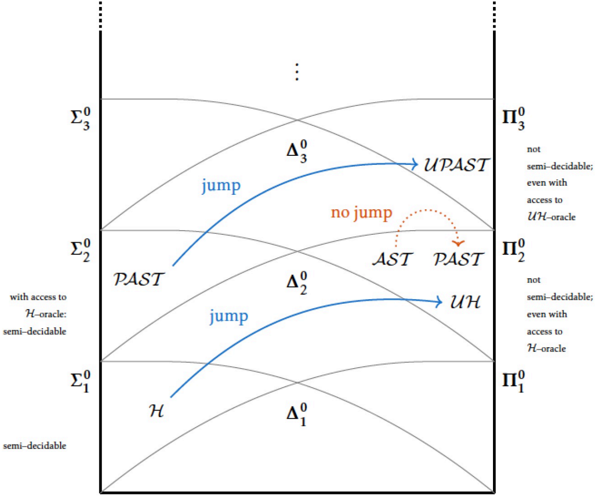


Infinite cheering — infinite expected runtime

Hardness of almost sure termination



Complexity landscape



Interpretation of these results

There is a complexity gap
between termination on one or all inputs

but not

between almost-sure termination on one or all inputs

but again

between positive almost-sure termination on one or all inputs