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

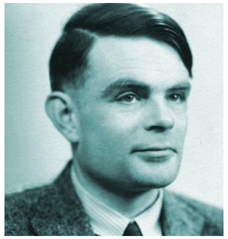
Overview

Overview

What we all know about termination

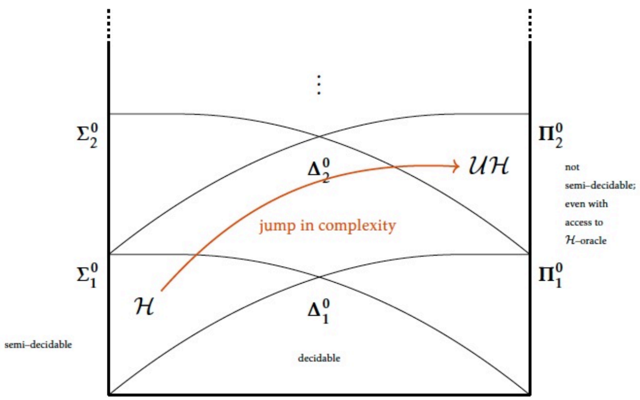
The halting problem
— does a program P terminate on a given input state s? —
is semi-decidable.

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

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

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 almost surely terminates.

Almost-sure termination

Do the following programs almost surely terminate?

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

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

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


Positive 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 almost surely terminates. In finite expected time.
Despite its possibility of divergence.

Null almost-sure termination

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.

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

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

Finite termination time

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

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

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 $UAST$

The decision problems AST and $UAST$

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

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

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

The decision problems AST and $UAST$

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

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

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. □

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]

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:

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

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

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).

□

Proof idea: hardness of positive as-termination

Reduction from the complement of the universal halting problem

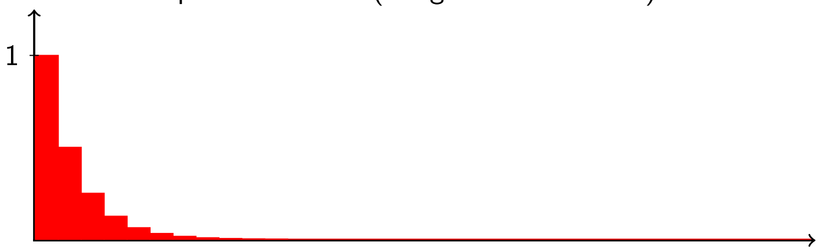
For an ordinary program Q, provide a probabilistic program P (depending on Q) and an input s, such that

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

```
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

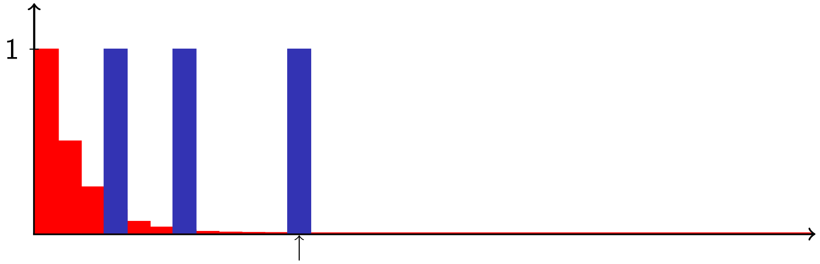
```
bool c := true;
int nrflips := 0;
int i := 0;
while (c) {
    // simulate Q for one (further) step on its i-th input
    if (Q terminates on its i-th input) {
        cheer; // take 2^nrflips effectless steps
        i++;
        // reset simulation of program Q
    }
    nrflips++;
    (c := false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

Expected runtime of P (integral over the bars):

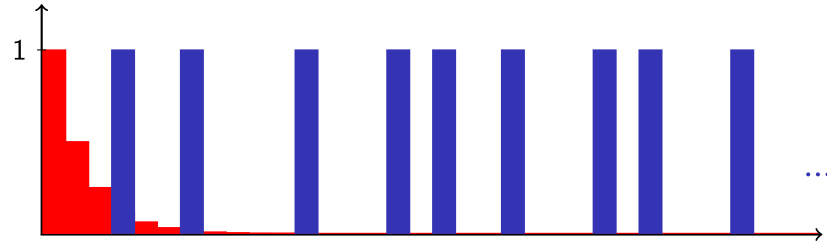


cheering on termination on (i−1)-th input

Finite cheering — finite expected runtime

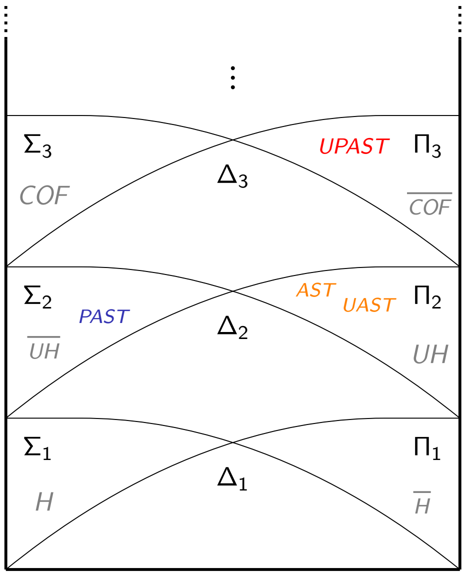
Q terminates on all inputs

Expected runtime of P (integral over the bars):

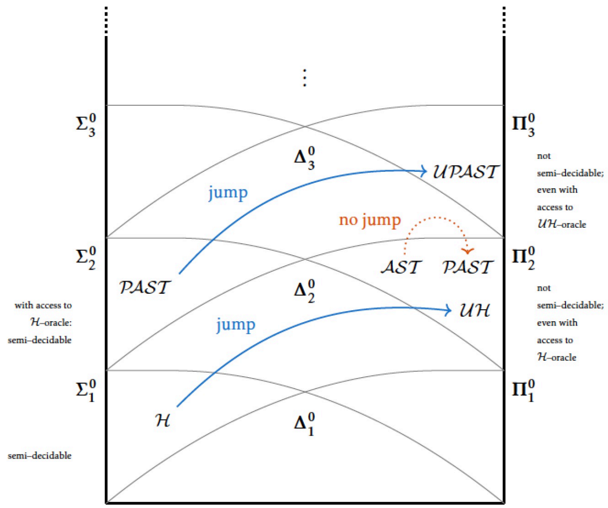


Infinite cheering — infinite expected runtime

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