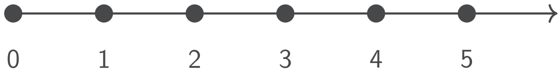
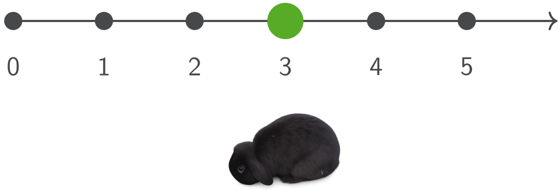
Automatically Analyzing Probabilistic Programs: Proving and Disproving Almost-Sure Termination of Probabilistic Term Rewriting.

Jan-Christoph Kassing
RWTH Aachen University
05.03.2026

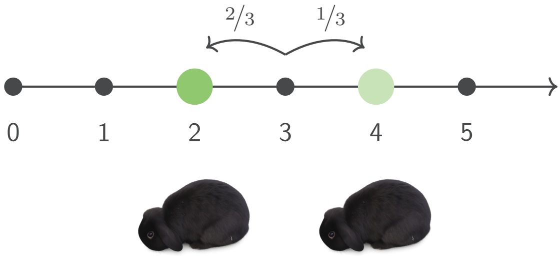
Random Walk



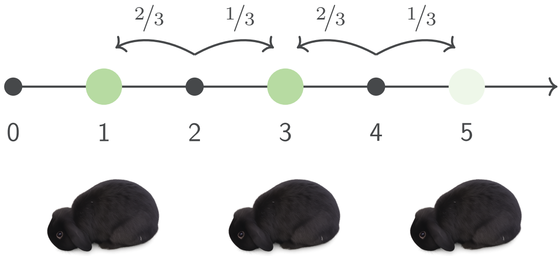
Random Walk



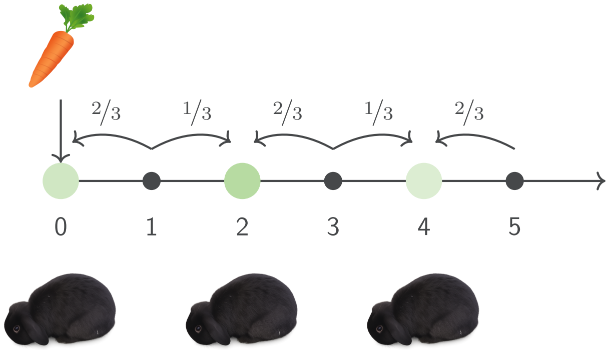
Random Walk



Random Walk

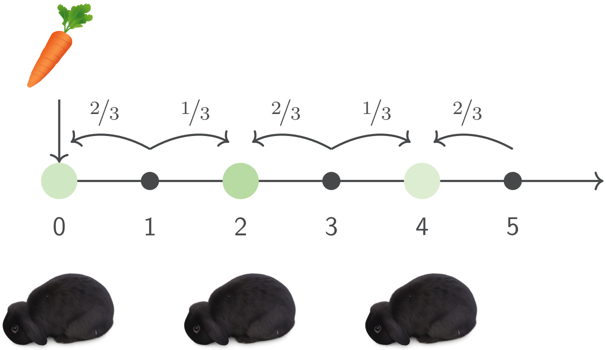


Random Walk



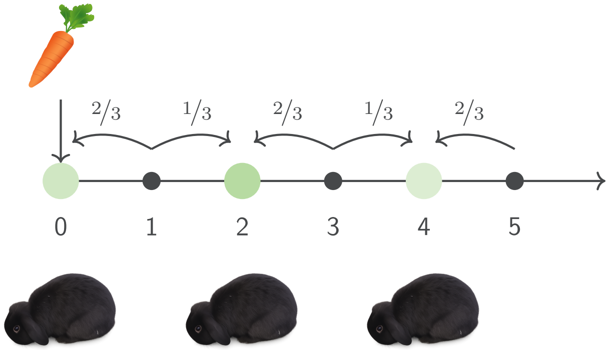
Random Walk

$x \leftarrow 3$
while $x > 0$ **do**
$\quad x \leftarrow x - 1 \oplus_{2/3} x \leftarrow x + 1;$



Random Walk

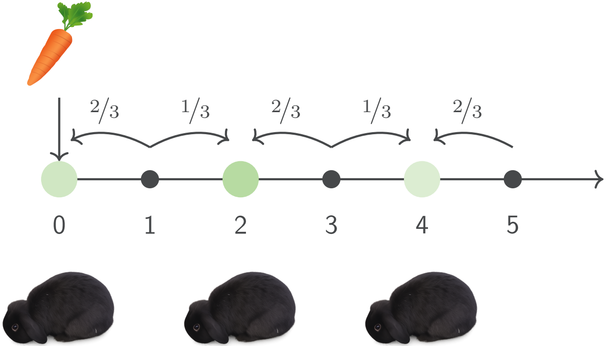
$x \leftarrow 3$
while $x > 0$ **do**
$\quad x \leftarrow x - 1 \oplus_{2/3} x \leftarrow x + 1;$



- Does the bunny (program) always reach the carrot (terminate)?

Random Walk

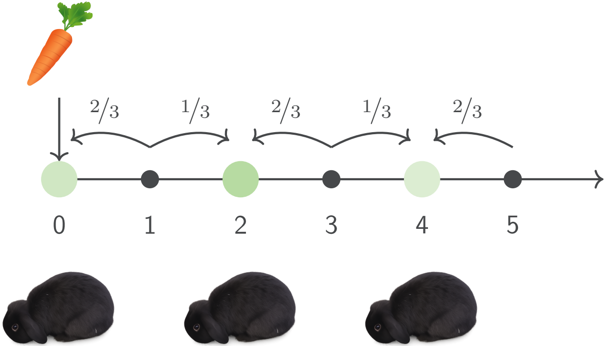
$x \leftarrow 3$
while $x > 0$ **do**
$\quad x \leftarrow x - 1 \oplus_{2/3} x \leftarrow x + 1;$



- Does the bunny (program) always reach the carrot (terminate)?
- What is the probability of reaching the carrot (probability of termination)?

Random Walk

$x \leftarrow 3$
while $x > 0$ **do**
  $x \leftarrow x - 1 \oplus_{2/3} x \leftarrow x + 1;$



- Does the bunny (program) always reach the carrot (terminate)?
- What is the probability of reaching the carrot (probability of termination)?
- What is the expected number of steps it takes to reach the carrot (expected runtime)?

Two Important Questions

Two Important Questions

1. What are applications of probabilistic programs? Why are they interesting?

Two Important Questions

1. What are applications of probabilistic programs? Why are they interesting?

2. How to analyze probabilistic programs?

Two Important Questions

1. What are applications of probabilistic programs? Why are they interesting?

2. How to analyze probabilistic programs?
    - Prove and disprove termination with probability 1

Two Important Questions

1. What are applications of probabilistic programs? Why are they interesting?

2. How to analyze probabilistic programs?
   - Prove and disprove termination with probability 1
   - Derive upper expected runtime bounds

Two Important Questions

1. What are applications of probabilistic programs? Why are they interesting?

2. How to analyze probabilistic programs?
   - Prove and disprove termination with probability 1
   - Derive upper expected runtime bounds

Different Types of Probabilistic Programs

Monte Carlo Algorithms

Las Vegas Algorithms

Different Types of Probabilistic Programs

Monte Carlo Algorithms

- Output may be incorrect
  (with a low probability)

Las Vegas Algorithms

Different Types of Probabilistic Programs

Monte Carlo Algorithms

- Output may be incorrect (with a low probability)

Las Vegas Algorithms

Monte Carlo Algorithm $\mathcal{P}$

Given input $s \in X$ and solution set $S \subseteq X$.

- If $\mathcal{P}$ on s returns *True*, then $s \in S$ with probability $p \geq 1/2$.
- If $\mathcal{P}$ on s returns *False*, then $s \notin S$ with probability 1.

Different Types of Probabilistic Programs

Monte Carlo Algorithms

- Output may be incorrect (with a low probability)

Las Vegas Algorithms

- Output always correct

Monte Carlo Algorithm $\mathcal{P}$

Given input $s \in X$ and solution set $S \subseteq X$.

- If $\mathcal{P}$ on s returns *True*, then $s \in S$ with probability $p \geq 1/2$.
- If $\mathcal{P}$ on s returns *False*, then $s \notin S$ with probability 1.

Different Types of Probabilistic Programs

Monte Carlo Algorithms

- Output may be incorrect (with a low probability)

Las Vegas Algorithms

- Output always correct
- Uses probabilities to prevent impactful worst-case performances

Monte Carlo Algorithm $\mathcal{P}$

Given input $s \in X$ and solution set $S \subseteq X$.

- If $\mathcal{P}$ on s returns *True*, then $s \in S$ with probability $p \geq 1/2$.
- If $\mathcal{P}$ on s returns *False*, then $s \notin S$ with probability 1.

Different Types of Probabilistic Programs

Monte Carlo Algorithms

- Output may be incorrect (with a low probability)

Las Vegas Algorithms

- Output always correct
- Uses probabilities to prevent impactful worst-case performances

Monte Carlo Algorithm $\mathcal{P}$

Given input $s \in X$ and solution set $S \subseteq X$.

- If $\mathcal{P}$ on s returns *True*, then $s \in S$ with probability $p \geq 1/2$.
- If $\mathcal{P}$ on s returns *False*, then $s \notin S$ with probability 1.

Las Vegas Algorithm $\mathcal{P}$

Given input $s \in X$ and solution set $S \subseteq X$.

- If $\mathcal{P}$ on s returns *True*, then $s \in S$ with probability 1.
- If $\mathcal{P}$ on s returns *False*, then $s \notin S$ with probability 1.

Different Types of Probabilistic Programs

Monte Carlo Algorithms

- Output may be incorrect (with a low probability)

Monte Carlo Algorithm $\mathcal{P}$

Given input $s \in X$ and solution set $S \subseteq X$.

- If $\mathcal{P}$ on s returns *True*, then $s \in S$ with probability $p \geq 1/2$.
- If $\mathcal{P}$ on s returns *False*, then $s \notin S$ with probability 1.

Las Vegas Algorithms

- Output always correct
- Uses probabilities to prevent impactful worst-case performances

Las Vegas Algorithm $\mathcal{P}$

Given input $s \in X$ and solution set $S \subseteq X$.

- If $\mathcal{P}$ on s returns *True*, then $s \in S$ with probability 1.
- If $\mathcal{P}$ on s returns *False*, then $s \notin S$ with probability 1.

Testing Polynomials to be Zero in Finite Fields

Polynomial Zero Testing

Given: $p(x_1, \ldots, x_n) \in \mathbb{F}_q[x_1, \ldots, x_n]$

Problem: $p(a_1, \ldots, a_n) = 0$ for all $(a_1, \ldots, a_n) \in \mathbb{F}_q^n$?

Testing Polynomials to be Zero in Finite Fields

Polynomial Zero Testing

Given: $p(x_1, \ldots, x_n) \in \mathbb{F}_q[x_1, \ldots, x_n]$

Problem: $p(a_1, \ldots, a_n) = 0$ for all $(a_1, \ldots, a_n) \in \mathbb{F}_q^n$?

Deterministic Algorithm:

Testing Polynomials to be Zero in Finite Fields

Polynomial Zero Testing

Given: $p(x_1, \ldots, x_n) \in \mathbb{F}_q[x_1, \ldots, x_n]$

Problem: $p(a_1, \ldots, a_n) = 0$ for all $(a_1, \ldots, a_n) \in \mathbb{F}_q^n$?

Deterministic Algorithm:
Try out all possible combinations for $(a_1, \ldots, a_n) \in \mathbb{F}_q^n$ and check $p(a_1, \ldots, a_n) = 0$.

Testing Polynomials to be Zero in Finite Fields

Polynomial Zero Testing

Given: $p(x_1, \dots, x_n) \in \mathbb{F}_q[x_1, \dots, x_n]$

Problem: $p(a_1, \dots, a_n) = 0$ for all $(a_1, \dots, a_n) \in \mathbb{F}_q^n$?

Deterministic Algorithm:
Try out all possible combinations for $(a_1, \dots, a_n) \in \mathbb{F}_q^n$ and check $p(a_1, \dots, a_n) = 0$.
$\Longrightarrow$ Runs in $\mathcal{O}(q^n)$.

Testing Polynomials to be Zero in Finite Fields

Polynomial Zero Testing

Given: $p(x_1, \dots, x_n) \in \mathbb{F}_q[x_1, \dots, x_n]$

Problem: $p(a_1, \dots, a_n) = 0$ for all $(a_1, \dots, a_n) \in \mathbb{F}_q^n$?

Deterministic Algorithm:
Try out all possible combinations for $(a_1, \dots, a_n) \in \mathbb{F}_q^n$ and check $p(a_1, \dots, a_n) = 0$.
$\Longrightarrow$ Runs in $\mathcal{O}(q^n)$.

Monte Carlo Algorithm:

Testing Polynomials to be Zero in Finite Fields

Polynomial Zero Testing

Given: $p(x_1, \ldots, x_n) \in \mathbb{F}_q[x_1, \ldots, x_n]$

Problem: $p(a_1, \ldots, a_n) = 0$ for all $(a_1, \ldots, a_n) \in \mathbb{F}_q^n$?

Deterministic Algorithm:
Try out all possible combinations for $(a_1, \ldots, a_n) \in \mathbb{F}_q^n$ and check $p(a_1, \ldots, a_n) = 0$.
$\implies$ Runs in $\mathcal{O}(q^n)$.

Monte Carlo Algorithm:
Pick $(a_1, \ldots, a_n) \in \mathbb{F}_q^n$ at random and check $p(a_1, \ldots, a_n) = 0$.

Testing Polynomials to be Zero in Finite Fields

Polynomial Zero Testing

Given: $p(x_1, \dots, x_n) \in \mathbb{F}_q[x_1, \dots, x_n]$

Problem: $p(a_1, \dots, a_n) = 0$ for all $(a_1, \dots, a_n) \in \mathbb{F}_q^n$?

Deterministic Algorithm:
Try out all possible combinations for $(a_1, \dots, a_n) \in \mathbb{F}_q^n$ and check $p(a_1, \dots, a_n) = 0$.
$\Longrightarrow$ Runs in $\mathcal{O}(q^n)$.

Monte Carlo Algorithm:
Pick $(a_1, \dots, a_n) \in \mathbb{F}_q^n$ at random and check $p(a_1, \dots, a_n) = 0$.
$\Longrightarrow$ Runs in $\mathcal{O}(1)$.

Testing Polynomials to be Zero in Finite Fields

Polynomial Zero Testing

Given: $p(x_1, \ldots, x_n) \in \mathbb{F}_q[x_1, \ldots, x_n]$

Problem: $p(a_1, \ldots, a_n) = 0$ for all $(a_1, \ldots, a_n) \in \mathbb{F}_q^n$?

Deterministic Algorithm:
Try out all possible combinations for $(a_1, \ldots, a_n) \in \mathbb{F}_q^n$ and check $p(a_1, \ldots, a_n) = 0$.
$\implies$ Runs in $\mathcal{O}(q^n)$.

Monte Carlo Algorithm:
Pick $(a_1, \ldots, a_n) \in \mathbb{F}_q^n$ at random and check $p(a_1, \ldots, a_n) = 0$.
$\implies$ Runs in $\mathcal{O}(1)$.

- If $p(a_1, \ldots, a_n) \neq 0$, then return *False*.

  This is correct with probability 1.

Testing Polynomials to be Zero in Finite Fields

Polynomial Zero Testing

Given: $p(x_1, \ldots, x_n) \in \mathbb{F}_q[x_1, \ldots, x_n]$

Problem: $p(a_1, \ldots, a_n) = 0$ for all $(a_1, \ldots, a_n) \in \mathbb{F}_q^n$?

Deterministic Algorithm:
Try out all possible combinations for $(a_1, \ldots, a_n) \in \mathbb{F}_q^n$ and check $p(a_1, \ldots, a_n) = 0$.
$\Longrightarrow$ Runs in $\mathcal{O}(q^n)$.

Monte Carlo Algorithm:
Pick $(a_1, \ldots, a_n) \in \mathbb{F}_q^n$ at random and check $p(a_1, \ldots, a_n) = 0$.
$\Longrightarrow$ Runs in $\mathcal{O}(1)$.

- If $p(a_1, \ldots, a_n) \neq 0$, then return *False*.

  This is correct with probability 1.

- If $p(a_1, \ldots, a_n) = 0$, then return *True*.

  This is correct with probability $\geq \frac{d}{q}$, where d is the degree of $p(x_1, \ldots, x_n)$.

Different Types of Probabilistic Programs

Monte Carlo Algorithms

- Output may be incorrect (with a low probability)

Las Vegas Algorithms

- Output always correct
- Uses probabilities to prevent impactful worst-case performances

Monte Carlo Algorithm $\mathcal{P}$

Given input s and solution set S.

- If $\mathcal{P}$ on s returns *True*, then $s \in S$ with probability $p < 1/2$.
- If $\mathcal{P}$ on s returns *False*, then $s \notin S$ with probability 1.

Las Vegas Algorithm $\mathcal{P}$

Given input s and solution set S.

- If $\mathcal{P}$ on s returns *True*, then $s \in S$ with probability 1.
- If $\mathcal{P}$ on s returns *False*, then $s \notin S$ with probability 1.

Probabilistic Quicksort

Order Natural Numbers

Given: $x_1, \ldots, x_n \in \mathbb{N}$

Problem: Find a bijective function $f : [1, n] \to [1, n]$ s. t. $x_{f(1)} \leq \ldots \leq x_{f(1)}$

Probabilistic Quicksort

Order Natural Numbers

Given: $x_1, \ldots, x_n \in \mathbb{N}$

Problem: Find a bijective function $f : [1, n] \to [1, n]$ s. t. $x_{f(1)} \leq \ldots \leq x_{f(1)}$

Deterministic Algorithm:

- Pick x_1 as *pivot element* and split the list into $L = \{x_i \mid x_i \leq x_1\}$ and $R = \{x_i \mid x_i > x_1\}$.
- Solve the problem for L and R recursively.
- Combine the two solutions and place x_1 in between.

Probabilistic Quicksort

Order Natural Numbers

Given: $x_1, \ldots, x_n \in \mathbb{N}$

Problem: Find a bijective function $f : [1, n] \to [1, n]$ s. t. $x_{f(1)} \leq \ldots \leq x_{f(1)}$

Deterministic Algorithm:

- Pick x_1 as *pivot element* and split the list into $L = \{x_i \mid x_i \leq x_1\}$ and $R = \{x_i \mid x_i > x_1\}$.
- Solve the problem for L and R recursively.
- Combine the two solutions and place x_1 in between.

3	2	1	5	4

Probabilistic Quicksort

Order Natural Numbers

Given: $x_1, \ldots, x_n \in \mathbb{N}$

Problem: Find a bijective function $f : [1, n] \to [1, n]$ s. t. $x_{f(1)} \leq \ldots \leq x_{f(1)}$

Deterministic Algorithm:

- Pick x_1 as *pivot element* and split the list into $L = \{x_i \mid x_i \leq x_1\}$ and $R = \{x_i \mid x_i > x_1\}$.
- Solve the problem for L and R recursively.
- Combine the two solutions and place x_1 in between.

3	2	1	5	4

Probabilistic Quicksort

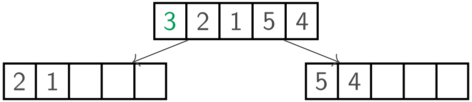
Order Natural Numbers

Given: $x_1, \ldots, x_n \in \mathbb{N}$

Problem: Find a bijective function $f : [1, n] \to [1, n]$ s. t. $x_{f(1)} \leq \ldots \leq x_{f(1)}$

Deterministic Algorithm:

- Pick x_1 as *pivot element* and split the list into $L = \{x_i \mid x_i \leq x_1\}$ and $R = \{x_i \mid x_i > x_1\}$.
- Solve the problem for L and R recursively.
- Combine the two solutions and place x_1 in between.



Probabilistic Quicksort

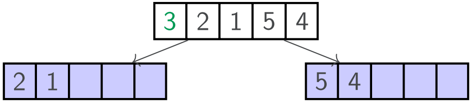
Order Natural Numbers

Given: $x_1, \ldots, x_n \in \mathbb{N}$

Problem: Find a bijective function $f : [1, n] \to [1, n]$ s. t. $x_{f(1)} \leq \ldots \leq x_{f(1)}$

Deterministic Algorithm:

- Pick x_1 as *pivot element* and split the list into $L = \{x_i \mid x_i \leq x_1\}$ and $R = \{x_i \mid x_i > x_1\}$.
- Solve the problem for L and R recursively.
- Combine the two solutions and place x_1 in between.



Probabilistic Quicksort

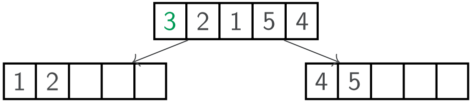
Order Natural Numbers

Given: $x_1, \ldots, x_n \in \mathbb{N}$

Problem: Find a bijective function $f : [1, n] \to [1, n]$ s. t. $x_{f(1)} \leq \ldots \leq x_{f(1)}$

Deterministic Algorithm:

- Pick x_1 as *pivot element* and split the list into $L = \{x_i \mid x_i \leq x_1\}$ and $R = \{x_i \mid x_i > x_1\}$.
- Solve the problem for L and R recursively.
- Combine the two solutions and place x_1 in between.



Probabilistic Quicksort

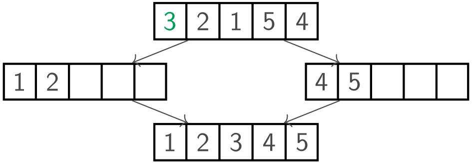
Order Natural Numbers

Given: $x_1, \ldots, x_n \in \mathbb{N}$

Problem: Find a bijective function $f : [1, n] \to [1, n]$ s. t. $x_{f(1)} \leq \ldots \leq x_{f(1)}$

Deterministic Algorithm:

- Pick x_1 as *pivot element* and split the list into $L = \{x_i \mid x_i \leq x_1\}$ and $R = \{x_i \mid x_i > x_1\}$.
- Solve the problem for L and R recursively.
- Combine the two solutions and place x_1 in between.



Probabilistic Quicksort

Order Natural Numbers

Given: $x_1, \ldots, x_n \in \mathbb{N}$

Problem: Find a bijective function $f : [1, n] \to [1, n]$ s. t. $x_{f(1)} \leq \ldots \leq x_{f(1)}$

Deterministic Algorithm:

- Pick x_1 as *pivot element* and split the list into $L = \{x_i \mid x_i \leq x_1\}$ and $R = \{x_i \mid x_i > x_1\}$.
- Solve the problem for L and R recursively.
- Combine the two solutions and place x_1 in between.

$\Longrightarrow$ Average runtime in $\mathcal{O}(n \cdot \log(n))$.

Probabilistic Quicksort

Order Natural Numbers

Given: $x_1, \ldots, x_n \in \mathbb{N}$

Problem: Find a bijective function $f : [1, n] \to [1, n]$ s. t. $x_{f(1)} \leq \ldots \leq x_{f(1)}$

Deterministic Algorithm:

- Pick x_1 as *pivot element* and split the list into $L = \{x_i \mid x_i \leq x_1\}$ and $R = \{x_i \mid x_i > x_1\}$.
- Solve the problem for L and R recursively.
- Combine the two solutions and place x_1 in between.

$\Longrightarrow$ Average runtime in $\mathcal{O}(n \cdot \log(n))$.
$\Longrightarrow$ Worst-Case runtime in $\mathcal{O}(n^2)$.

Probabilistic Quicksort

Order Natural Numbers

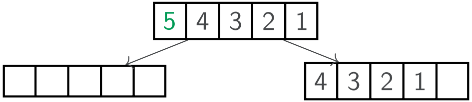
Given: $x_1, \ldots, x_n \in \mathbb{N}$

Problem: Find a bijective function $f : [1, n] \to [1, n]$ s. t. $x_{f(1)} \leq \ldots \leq x_{f(1)}$

Deterministic Algorithm:

- Pick x_1 as *pivot element* and split the list into $L = \{x_i \mid x_i \leq x_1\}$ and $R = \{x_i \mid x_i > x_1\}$.
- Solve the problem for L and R recursively.
- Combine the two solutions and place x_1 in between.

$\Longrightarrow$ Average runtime in $\mathcal{O}(n \cdot \log(n))$.
$\Longrightarrow$ Worst-Case runtime in $\mathcal{O}(n^2)$.



Probabilistic Quicksort

Order Natural Numbers

Given: $x_1, \ldots, x_n \in \mathbb{N}$

Problem: Find a bijective function $f : [1, n] \to [1, n]$ s. t. $x_{f(1)} \leq \ldots \leq x_{f(1)}$

Las Vegas Algorithm:

- Pick pivot element at random and split the list into $L = \{x_i \mid x_i \leq x_1\}$ and $R = \{x_i \mid x_i > x_1\}$.
- Solve the problem for L and R recursively.
- Combine the two solutions and place x_1 in between.

Probabilistic Quicksort

Order Natural Numbers

Given: $x_1, \dots, x_n \in \mathbb{N}$

Problem: Find a bijective function $f : [1, n] \to [1, n]$ s. t. $x_{f(1)} \leq \dots \leq x_{f(1)}$

Las Vegas Algorithm:

- Pick pivot element at random and split the list into $L = \{x_i \mid x_i \leq x_1\}$ and $R = \{x_i \mid x_i > x_1\}$.
- Solve the problem for L and R recursively.
- Combine the two solutions and place x_1 in between.

$\Longrightarrow$ Expected runtime in $\mathcal{O}(n \cdot \log(n))$.

Two Important Questions

1. What are applications of probabilistic programs? Why are they interesting?

2. How to analyze probabilistic programs?
   - Prove and disprove termination with probability 1
   - Derive upper expected runtime bounds

Two Important Questions

1. What are applications of probabilistic programs? Why are they interesting?
   - Increase the expected worst-case runtime
2. How to analyze probabilistic programs?
   - Prove and disprove termination with probability 1
   - Derive upper expected runtime bounds

Two Important Questions

1. What are applications of probabilistic programs? Why are they interesting?
   - Increase the expected worst-case runtime

2. How to analyze probabilistic programs?
   - Prove and disprove termination with probability 1
   - Derive upper expected runtime bounds

Termination and Complexity Analysis for Programs

Termination and Complexity Analysis for Programs

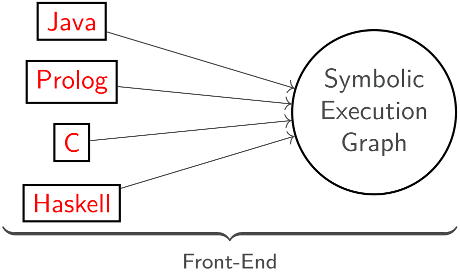
Java

Prolog

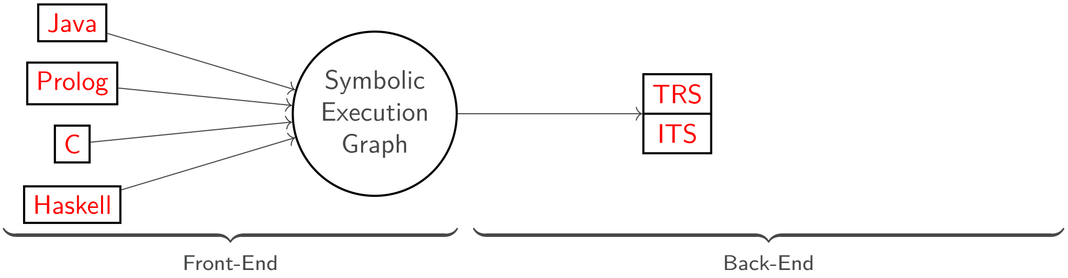
C

Haskell

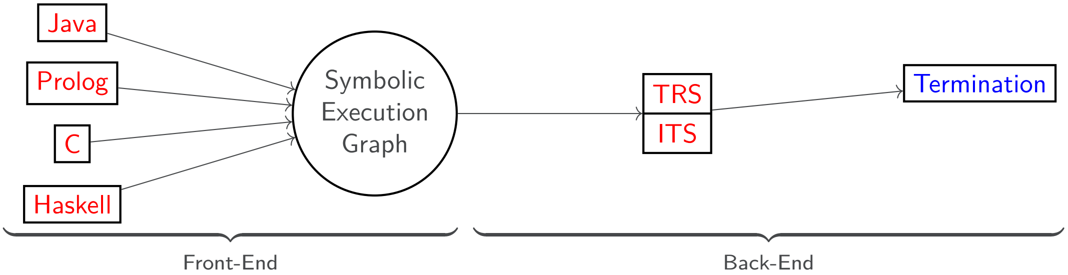
Termination and Complexity Analysis for Programs



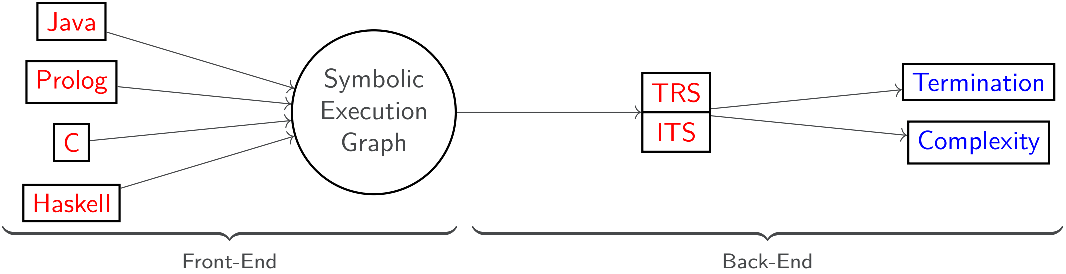
Termination and Complexity Analysis for Programs



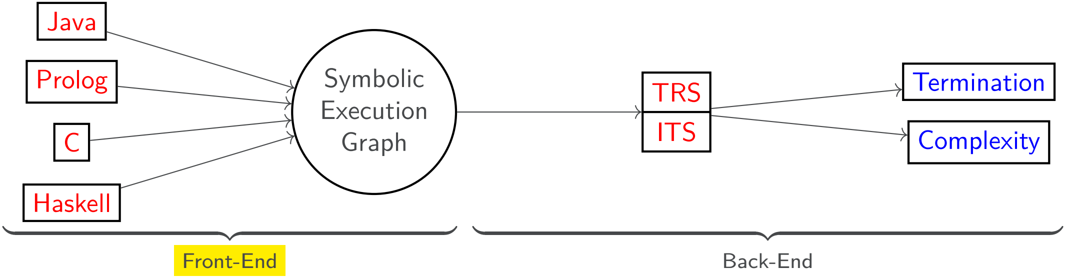
Termination and Complexity Analysis for Programs



Termination and Complexity Analysis for Programs

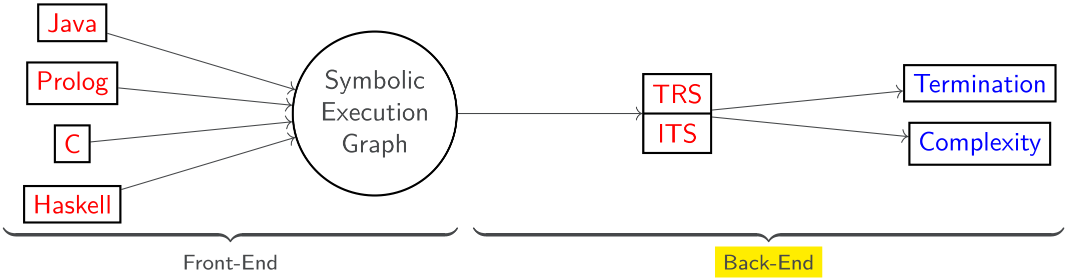


Termination and Complexity Analysis for Programs



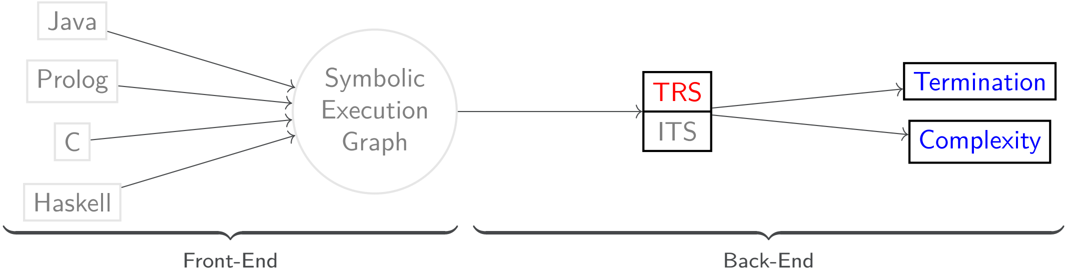
- language-specific features when generating symbolic execution graph

Termination and Complexity Analysis for Programs

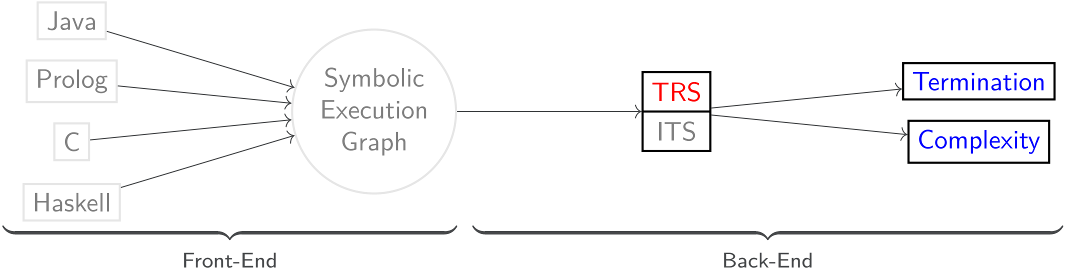


- language-specific features when generating symbolic execution graph
- back-end analyzes Term Rewrite Systems and/or Integer Transition Systems

Termination and Complexity Analysis for Programs

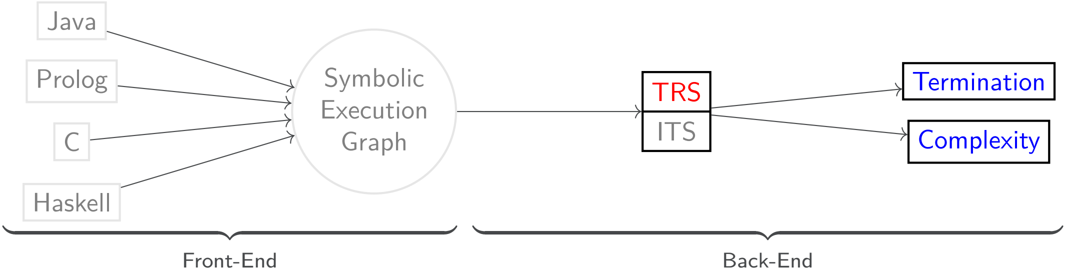


Termination and Complexity Analysis for Programs



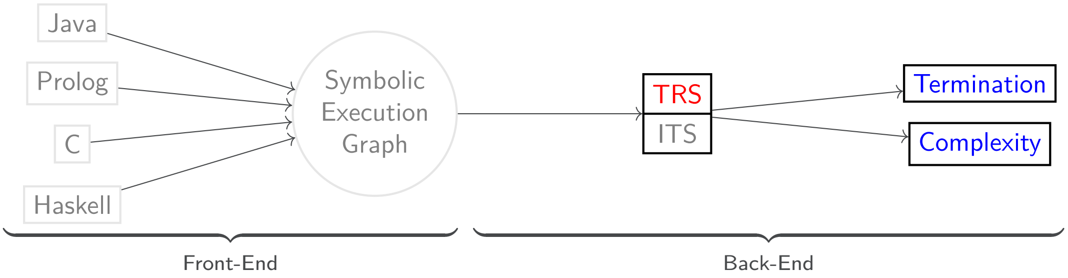
- Proving Termination and Complexity of TRSs

Termination and Complexity Analysis for Programs



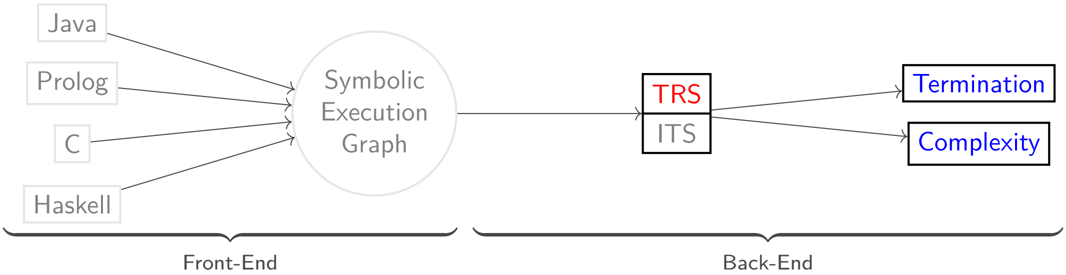
- Proving Termination and Complexity of TRSs
- Proving Termination and Complexity of Probabilistic TRSs

Termination and Complexity Analysis for Programs



- Proving Termination and Complexity of TRSs
- Proving Termination and Complexity of Probabilistic TRSs
- Disproving Termination of Probabilistic TRSs

Termination and Complexity Analysis for Programs



- Proving Termination and Complexity of TRSs
- Proving Termination and Complexity of Probabilistic TRSs
- Disproving Termination of Probabilistic TRSs

Complexity of TRSs

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))$$

Complexity of TRSs

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))$$

$\mathsf{double}(\mathsf{s}(\mathsf{s}(0)))$

Complexity of TRSs

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))$$

$$\mathsf{double}(\mathsf{s}(\mathsf{s}(0))) \to_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}(\mathsf{s}(\mathsf{double}(\mathsf{s}(0))))$$

Complexity of TRSs

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$

$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))$$

$$\mathsf{double}(\mathsf{s}(\mathsf{s}(0))) \to_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}(\mathsf{s}(\mathsf{double}(\mathsf{s}(0)))) \to_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(\mathsf{double}(0))$$

Complexity of TRSs

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))$$

$$\mathsf{double}(\mathsf{s}(\mathsf{s}(0))) \to_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}(\mathsf{s}(\mathsf{double}(\mathsf{s}(0)))) \to_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(\mathsf{double}(0)) \to_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(0)$$

Complexity of TRSs

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))$$

Termination

TRS $\mathcal{R}$ is *terminating* if there is no infinite evaluation $t_0 \to_{\mathcal{R}} t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} \dots$

$$\mathsf{double}(\mathsf{s}(\mathsf{s}(0))) \to_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}(\mathsf{s}(\mathsf{double}(\mathsf{s}(0)))) \to_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(\mathsf{double}(0)) \to_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(0)$$

Complexity of TRSs

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))$$

Runtime Complexity, $\mathrm{rc}_{\mathcal{R}}$

$$\mathrm{rc}_{\mathcal{R}}(n) = \sup\{\mathrm{dh}_{\mathcal{R}}(t) \mid \qquad |t| \leq n\}$$

$$\mathsf{double}(\mathsf{s}(\mathsf{s}(0))) \to_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}(\mathsf{s}(\mathsf{double}(\mathsf{s}(0)))) \to_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(\mathsf{double}(0)) \to_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(0)$$

Complexity of TRSs

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))$$

Runtime Complexity, $\mathrm{rc}_{\mathcal{R}}$

$$\mathrm{rc}_{\mathcal{R}}(n) = \sup\{\mathrm{dh}_{\mathcal{R}}(t) \mid \quad |t| \leq n\}$$

derivation **h**eight

$$\mathsf{double}(\mathsf{s}(\mathsf{s}(0))) \to_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}(\mathsf{s}(\mathsf{double}(\mathsf{s}(0)))) \to_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(\mathsf{double}(0)) \to_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(0)$$

$\mathrm{dh}_{\mathcal{R}}(\mathsf{double}(\mathsf{s}(\mathsf{s}(0)))) =$ "max number of steps" $= 3$

Complexity of TRSs

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))$$

Innermost Runtime Complexity, $\mathrm{rc}_{\mathcal{R}}$

$$\mathrm{rc}_{\mathcal{R}}(n) = \sup\{\mathrm{dh}_{\mathcal{R}}(t) \mid \quad |t| \leq n\}$$

$$\mathsf{double}(\mathsf{s}(\mathsf{s}(0))) \to_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}(\mathsf{s}(\mathsf{double}(\mathsf{s}(0)))) \to_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(\mathsf{double}(0)) \to_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(0)$$

$\mathrm{dh}_{\mathcal{R}}\,(\mathsf{double}(\mathsf{s}(\mathsf{s}(0)))) =$ "max number of steps" $= 3$

Complexity of TRSs

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))$$

Innermost Runtime Complexity, $\mathrm{rc}_{\mathcal{R}}$

$$\mathrm{rc}_{\mathcal{R}}(n) = \sup\{\mathrm{dh}_{\mathcal{R}}(t) \mid \qquad |t| \leq n\}$$

$$\mathsf{double}(\mathsf{s}(\mathsf{s}(0))) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}(\mathsf{s}(\mathsf{double}(\mathsf{s}(0)))) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(\mathsf{double}(0)) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(0)$$

$\mathrm{dh}_{\mathcal{R}}\,(\mathsf{double}(\mathsf{s}(\mathsf{s}(0)))) =$ "max number of steps" $= 3$

Complexity of TRSs

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))$$

Innermost Runtime Complexity, $\mathrm{rc}_{\mathcal{R}}$

$$\mathrm{rc}_{\mathcal{R}}(n) = \sup\{\mathrm{dh}_{\mathcal{R}}(t) \mid \qquad |t| \leq n\}$$

$$\mathsf{double}(\mathsf{s}(\mathsf{s}(0))) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}(\mathsf{s}(\mathsf{double}(\mathsf{s}(0)))) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(\mathsf{double}(0)) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(0)$$

$\mathrm{dh}_{\mathcal{R}}\,(\mathsf{double}(\mathsf{s}(\mathsf{s}(0)))) = \text{“max number of steps”} = 3$

$\mathsf{double}(\mathsf{double}(\mathsf{s}(0)))$

Complexity of TRSs

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))$$

Innermost Runtime Complexity, $\mathrm{rc}_{\mathcal{R}}$

$$\mathrm{rc}_{\mathcal{R}}(n) = \sup\{\mathrm{dh}_{\mathcal{R}}(t) \mid \qquad |t| \leq n\}$$

$$\mathsf{double}(\mathsf{s}(\mathsf{s}(0))) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}(\mathsf{s}(\mathsf{double}(\mathsf{s}(0)))) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(\mathsf{double}(0)) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(0)$$

$\mathrm{dh}_{\mathcal{R}}\,(\mathsf{double}(\mathsf{s}(\mathsf{s}(0)))) = \text{“max number of steps”} = 3$

$\mathsf{double}(\mathsf{double}(\mathsf{s}(0)))$

Complexity of TRSs

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))$$

Innermost Runtime Complexity, $\mathrm{rc}_{\mathcal{R}}$

$$\mathrm{rc}_{\mathcal{R}}(n) = \sup\{\mathrm{dh}_{\mathcal{R}}(t) \mid \quad |t| \leq n\}$$

$$\mathsf{double}(\mathsf{s}(\mathsf{s}(0))) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}(\mathsf{s}(\mathsf{double}(\mathsf{s}(0)))) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(\mathsf{double}(0)) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(0)$$

$\mathrm{dh}_{\mathcal{R}}\,(\mathsf{double}(\mathsf{s}(\mathsf{s}(0)))) = \text{“max number of steps”} = 3$

$$\mathsf{double}(\mathsf{double}(\mathsf{s}(0))) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{double}(\mathsf{s}(\mathsf{double}(0))) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{double}(\mathsf{s}(\mathsf{s}(0))) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \ldots$$

Complexity of TRSs

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))$$

Innermost Runtime Complexity, $\mathrm{rc}_{\mathcal{R}}$

$$\mathrm{rc}_{\mathcal{R}}(n) = \sup\{\mathrm{dh}_{\mathcal{R}}(t) \mid \qquad |t| \leq n\}$$

$$\mathsf{double}(\mathsf{s}(\mathsf{s}(0))) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}(\mathsf{s}(\mathsf{double}(\mathsf{s}(0)))) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(\mathsf{double}(0)) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(0)$$

$\mathrm{dh}_{\mathcal{R}}\,(\mathsf{double}(\mathsf{s}(\mathsf{s}(0)))) = \text{“max number of steps”} = 3$

Basic Terms, $\mathcal{T}_B$

Complexity of TRSs

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))$$

Innermost Runtime Complexity, $\mathrm{rc}_{\mathcal{R}}$

$$\mathrm{rc}_{\mathcal{R}}(n) = \sup\{\mathrm{dh}_{\mathcal{R}}(t) \mid \qquad |t| \leq n\}$$

Defined Symbols Σ_D: double,

$$\mathsf{double}(\mathsf{s}(\mathsf{s}(0))) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}(\mathsf{s}(\mathsf{double}(\mathsf{s}(0)))) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(\mathsf{double}(0)) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(0)$$

$\mathrm{dh}_{\mathcal{R}}(\mathsf{double}(\mathsf{s}(\mathsf{s}(0)))) =$ "max number of steps" $= 3$

Basic Terms, $\mathcal{T}_B$

Complexity of TRSs

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))$$

Innermost Runtime Complexity, $\mathrm{rc}_{\mathcal{R}}$

$$\mathrm{rc}_{\mathcal{R}}(n) = \sup\{\mathrm{dh}_{\mathcal{R}}(t) \mid \qquad |t| \leq n\}$$

Defined Symbols Σ_D: double, Constructors Σ_C: s, 0

$$\mathsf{double}(\mathsf{s}(\mathsf{s}(0))) \xrightarrow{\mathrm{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}(\mathsf{s}(\mathsf{double}(\mathsf{s}(0)))) \xrightarrow{\mathrm{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(\mathsf{double}(0)) \xrightarrow{\mathrm{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(0)$$

$\mathrm{dh}_{\mathcal{R}}(\mathsf{double}(\mathsf{s}(\mathsf{s}(0)))) =$ "max number of steps" $= 3$

Basic Terms, $\mathcal{T}_B$

Complexity of TRSs

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))$$

Innermost Runtime Complexity, $\mathrm{rc}_{\mathcal{R}}$

$$\mathrm{rc}_{\mathcal{R}}(n) = \sup\{\mathrm{dh}_{\mathcal{R}}(t) \mid t \in \mathcal{T}_B, |t| \leq n\}$$

Defined Symbols Σ_D: double, Constructors Σ_C: s, 0

$$\mathsf{double}(\mathsf{s}(\mathsf{s}(0))) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}(\mathsf{s}(\mathsf{double}(\mathsf{s}(0)))) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(\mathsf{double}(0)) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(0)$$

$\mathrm{dh}_{\mathcal{R}}(\mathsf{double}(\mathsf{s}(\mathsf{s}(0))))$ = "max number of steps" $= 3$

Basic Terms, $\mathcal{T}_B$

Terms $f(c_1, \ldots, c_n) \in \mathcal{T}_B$ are *basic* if $f \in \Sigma_D$ and all c_i are constructor terms.

Complexity of TRSs

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))$$

Innermost Runtime Complexity, $\mathrm{rc}_{\mathcal{R}}$

$$\mathrm{rc}_{\mathcal{R}}(n) = \sup\{\mathrm{dh}_{\mathcal{R}}(t) \mid t \in \mathcal{T}_B, |t| \leq n\}$$

Defined Symbols Σ_D: double, Constructors Σ_C: $\mathsf{s}, 0$

$$\mathsf{double}(\mathsf{s}(\mathsf{s}(0))) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}(\mathsf{s}(\mathsf{double}(\mathsf{s}(0)))) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(\mathsf{double}(0)) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(0)$$

$\mathrm{dh}_{\mathcal{R}}(\mathsf{double}(\mathsf{s}(\mathsf{s}(0)))) =$ "max number of steps" $= 3$ $\quad \mathrm{rc}_{\mathcal{R}_{\mathsf{double}}}(n) = n - 1$

Basic Terms, $\mathcal{T}_B$

Terms $f(c_1, \ldots, c_n) \in \mathcal{T}_B$ are *basic* if $f \in \Sigma_D$ and all c_i are constructor terms.

Complexity of TRSs

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))$$

Innermost Runtime Complexity, $\mathrm{rc}_{\mathcal{R}}$

$$\mathrm{rc}_{\mathcal{R}}(n) = \sup\{\mathrm{dh}_{\mathcal{R}}(t) \mid t \in \mathcal{T}_B, |t| \leq n\}$$

Defined Symbols Σ_D: double, Constructors Σ_C: $\mathsf{s}, 0$

$$\mathsf{double}(\mathsf{s}(\mathsf{s}(0))) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}(\mathsf{s}(\mathsf{double}(\mathsf{s}(0)))) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(\mathsf{double}(0)) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \mathsf{s}^4(0)$$

$\mathrm{dh}_{\mathcal{R}}\left(\mathsf{double}(\mathsf{s}(\mathsf{s}(0)))\right) =$ "max number of steps" $= 3$ $\quad \mathrm{rc}_{\mathcal{R}_{\mathsf{double}}}(n) = n - 1 \in \mathcal{O}(n^1)$

Basic Terms, $\mathcal{T}_B$

Terms $f(c_1, \ldots, c_n) \in \mathcal{T}_B$ are *basic* if $f \in \Sigma_D$ and all c_i are constructor terms.

Complexity of TRSs

$\mathcal{R}_{\text{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))$$

Innermost Runtime Complexity, $\mathrm{rc}_{\mathcal{R}}$

$$\mathrm{rc}_{\mathcal{R}}(n) = \sup\{\mathrm{dh}_{\mathcal{R}}(t) \mid t \in \mathcal{T}_B, |t| \leq n\}$$

Defined Symbols Σ_D: double, Constructors Σ_C: s, 0

$$\mathsf{double}(\mathsf{s}(\mathsf{s}(0))) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\text{double}}} \mathsf{s}(\mathsf{s}(\mathsf{double}(\mathsf{s}(0)))) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\text{double}}} \mathsf{s}^4(\mathsf{double}(0)) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\text{double}}} \mathsf{s}^4(0)$$

$\mathrm{dh}_{\mathcal{R}}(\mathsf{double}(\mathsf{s}(\mathsf{s}(0)))) =$ "max number of steps" $= 3$ $\quad \mathrm{rc}_{\mathcal{R}_{\text{double}}}(n) = n - 1 \in \mathcal{O}(n^1)$ $\quad \mathrm{rc}_{\mathcal{R}_{\text{double}}} = \mathrm{Pol}_1$

Basic Terms, $\mathcal{T}_B$

Terms $f(c_1, \ldots, c_n) \in \mathcal{T}_B$ are *basic* if $f \in \Sigma_D$ and all c_i are constructor terms.

Proving Termination

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))$$

Proving Termination

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))$$

Natural & Monotonic Polynomial Interpretation $\mathcal{I}$

- natural: $\mathcal{I}_f(x_1, \ldots, x_n)$ is a polynomial with natural coefficients for every function symbol $f \in \Sigma$
- monotonic: $x > y$ implies $\mathcal{I}_f(\ldots, x, \ldots) > \mathcal{I}_f(\ldots, y, \ldots)$ for every function symbol $f \in \Sigma$

Proving Termination

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))$$

$$\mathcal{I}_0 = 1 \qquad \mathcal{I}_{\mathsf{s}}(x) = x + 1 \qquad \mathcal{I}_{\mathsf{double}}(x) = 2x + 1$$

Natural & Monotonic Polynomial Interpretation $\mathcal{I}$

- natural: $\mathcal{I}_f(x_1, \ldots, x_n)$ is a polynomial with natural coefficients for every function symbol $f \in \Sigma$
- monotonic: $x > y$ implies $\mathcal{I}_f(\ldots, x, \ldots) > \mathcal{I}_f(\ldots, y, \ldots)$ for every function symbol $f \in \Sigma$

Proving Termination

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))$$

$$\mathcal{I}_0 = 1 \qquad \mathcal{I}_{\mathsf{s}}(x) = x + 1 \qquad \mathcal{I}_{\mathsf{double}}(x) = 2x + 1$$

Proving Termination With Natural & Monotonic $\mathcal{I}$ [Lankford'79]

$\mathcal{R}$ is terminating if for all rules $\ell \to r : \mathcal{I}(\ell) > \mathcal{I}(r)$

Proving Termination

$\mathcal{R}_{\mathsf{double}}$:

$$\mathcal{I}(\mathsf{double}(0)) > \mathcal{I}(0)$$
$$\mathcal{I}(\mathsf{double}(\mathsf{s}(x))) > \mathcal{I}(\mathsf{s}(\mathsf{s}(\mathsf{double}(x))))$$

$$\mathcal{I}_0 = 1 \qquad \mathcal{I}_{\mathsf{s}}(x) = x + 1 \qquad \mathcal{I}_{\mathsf{double}}(x) = 2x + 1$$

Proving Termination With Natural & Monotonic $\mathcal{I}$ [Lankford'79]

$\mathcal{R}$ is terminating if for all rules $\ell \to r : \mathcal{I}(\ell) > \mathcal{I}(r)$

Proving Termination

$\mathcal{R}_{\mathsf{double}}$:

$$2 \cdot \mathcal{I}(0) + 1 > 1$$
$$2 \cdot \mathcal{I}(\mathsf{s}(x)) + 1 > \mathcal{I}(\mathsf{s}(\mathsf{double}(x))) + 1$$

$$\mathcal{I}_0 = 1 \qquad \mathcal{I}_{\mathsf{s}}(x) = x + 1 \qquad \mathcal{I}_{\mathsf{double}}(x) = 2x + 1$$

Proving Termination With Natural & Monotonic $\mathcal{I}$ [Lankford'79]

$\mathcal{R}$ is terminating if for all rules $\ell \to r : \mathcal{I}(\ell) > \mathcal{I}(r)$

Proving Termination

$\mathcal{R}_{\text{double}}$:

$$2 \cdot 1 + 1 > 1$$
$$2 \cdot (x + 1) + 1 > 2 \cdot x + 2$$

$$\mathcal{I}_0 = 1 \qquad \mathcal{I}_{\mathsf{s}}(x) = x + 1 \qquad \mathcal{I}_{\text{double}}(x) = 2x + 1$$

Proving Termination With Natural & Monotonic $\mathcal{I}$ [Lankford'79]

$\mathcal{R}$ is terminating if for all rules $\ell \to r : \mathcal{I}(\ell) > \mathcal{I}(r)$

Proving Termination

$\mathcal{R}_{\mathsf{double}}$:

$$3 > 1$$
$$2x + 3 > 2x + 2$$

$$\mathcal{I}_0 = 1 \qquad \mathcal{I}_{\mathsf{s}}(x) = x + 1 \qquad \mathcal{I}_{\mathsf{double}}(x) = 2x + 1$$

Proving Termination With Natural & Monotonic $\mathcal{I}$ [Lankford'79]

$\mathcal{R}$ is terminating if for all rules $\ell \to r : \mathcal{I}(\ell) > \mathcal{I}(r)$

Proving Termination

$\mathcal{R}_{\mathsf{double}}$:

$$3 > 1$$
$$2x + 3 > 2x + 2$$

$$\mathcal{I}_0 = 1 \qquad \mathcal{I}_{\mathsf{s}}(x) = x + 1 \qquad \mathcal{I}_{\mathsf{double}}(x) = 2x + 1$$

Proving Termination With Natural & Monotonic $\mathcal{I}$ [Lankford'79]

$\mathcal{R}$ is terminating if for all rules $\ell \to r : \mathcal{I}(\ell) > \mathcal{I}(r)$

$$t_0 \quad \overset{\mathsf{i}}{\to}_{\mathcal{R}_{\mathsf{double}}} \quad t_1 \quad \overset{\mathsf{i}}{\to}_{\mathcal{R}_{\mathsf{double}}} \quad t_2 \quad \overset{\mathsf{i}}{\to}_{\mathcal{R}_{\mathsf{double}}} \quad \dots$$

Proving Termination

$\mathcal{R}_{\mathsf{double}}$:

$$3 > 1$$
$$2x + 3 > 2x + 2$$

$$\mathcal{I}_0 = 1 \qquad \mathcal{I}_{\mathsf{s}}(x) = x + 1 \qquad \mathcal{I}_{\mathsf{double}}(x) = 2x + 1$$

Proving Termination With Natural & Monotonic $\mathcal{I}$ [Lankford'79]

$\mathcal{R}$ is terminating if for all rules $\ell \to r : \mathcal{I}(\ell) > \mathcal{I}(r)$

$$t_0 \quad \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \quad t_1 \quad \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \quad t_2 \quad \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{double}}} \quad \ldots$$

$$\mathcal{I}(t_0) \quad > \quad \mathcal{I}(t_1) \quad > \quad \mathcal{I}(t_2) \quad > \quad \ldots$$

Proving Complexity

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(\mathsf{0}) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{double}(x))$$

$$\mathcal{I}_{\mathsf{0}} = 1 \qquad \mathcal{I}_{\mathsf{s}}(x) = x + 1 \qquad \mathcal{I}_{\mathsf{double}}(x) = 2x + 1$$

Proving Complexity

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{double}(x))$$

$$\mathcal{I}_0 = 1 \qquad \mathcal{I}_{\mathsf{s}}(x) = x + 1 \qquad \mathcal{I}_{\mathsf{double}}(x) = 2x + 1$$

Goal: Infer complexity from the highest degree of $\mathcal{I}$ [Hofbauer & Lautemann'89]

Proving Complexity

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{double}(x))$$

$$\mathcal{I}_0 = 1 \qquad \mathcal{I}_{\mathsf{s}}(x) = x + 1 \qquad \mathcal{I}_{\mathsf{double}}(x) = 2x + 1$$

Goal: Infer complexity from the highest degree of $\mathcal{I}$ [Hofbauer & Lautemann'89]

For basic term $t = \mathsf{double}(\mathsf{s}^n(0))$:

Proving Complexity

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(\mathsf{0}) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{double}(x))$$

$$\mathcal{I}_{\mathsf{0}} = 1 \qquad \mathcal{I}_{\mathsf{s}}(x) = x + 1 \qquad \mathcal{I}_{\mathsf{double}}(x) = 2x + 1$$

Goal: Infer complexity from the highest degree of $\mathcal{I}$ [Hofbauer & Lautemann'89]

For basic term $t = \mathsf{double}(\mathsf{s}^n(\mathsf{0}))$: $\qquad \mathcal{I}(t) = 2 \cdot (n+1) + 1$

Proving Complexity

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{double}(x))$$

$$\mathcal{I}_0 = 1 \qquad \mathcal{I}_{\mathsf{s}}(x) = x + 1 \qquad \mathcal{I}_{\mathsf{double}}(x) = 2x + 1$$

Goal: Infer complexity from the highest degree of $\mathcal{I}$ [Hofbauer & Lautemann'89]

For basic term $t = \mathsf{double}(\mathsf{s}^n(0))$: $\mathcal{I}(t) = 2 \cdot (n+1) + 1$

$\rightsquigarrow$ at most linear runtime complexity

Proving Complexity

$\mathcal{R}_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{double}(x))$$

$$\mathcal{I}_0 = 1 \qquad \mathcal{I}_{\mathsf{s}}(x) = 2x \qquad \mathcal{I}_{\mathsf{double}}(x) = 2x + 1$$

Goal: Infer complexity from the highest degree of $\mathcal{I}$ [Hofbauer & Lautemann'89]

For basic term $t = \mathsf{double}(\mathsf{s}^n(0))$: $\mathcal{I}(t) = 2 \cdot (\; 2^n \;) + 1$

Proving Complexity

$\mathcal{R}_{\text{double}}$:

$$\text{double}(0) \to 0$$
$$\text{double}(\text{s}(x)) \to \text{s}(\text{double}(x))$$

$$\mathcal{I}_0 = 1 \qquad \mathcal{I}_{\text{s}}(x) = 2x \qquad \mathcal{I}_{\text{double}}(x) = 2x + 1$$

Goal: Infer complexity from the highest degree of $\mathcal{I}$ [Hofbauer & Lautemann'89]

For basic term $t = \text{double}(\text{s}^n(0))$: $\qquad \mathcal{I}(t) = 2 \cdot (\; 2^n \;) + 1$

Complexity Polynomial Interpretation (CPI)

$\mathcal{I}_f(x_1, \dots, x_n) = a_1 x_1 + \dots + a_n x_n + b$ for every constructor $f \in \Sigma_C$ with $b \in \mathbb{N}, a_i \in \{0, 1\}$

Proving Complexity

$\mathcal{R}_{\text{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{double}(x))$$

CPI: $\mathcal{I}_{0} = 1$ $\quad \mathcal{I}_{\mathsf{s}}(x) = x + 1$ $\quad \mathcal{I}_{\mathsf{double}}(x) = 2x + 1$

Goal: Infer complexity from the highest degree of $\mathcal{I}$ [Hofbauer & Lautemann'89]

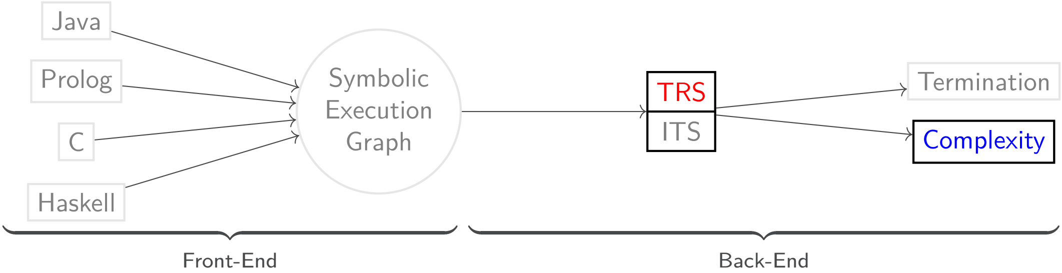
For basic term $t = \mathsf{double}(\mathsf{s}^n(0))$: $\quad \mathcal{I}(t) = 2 \cdot (n+1) + 1$

$\rightsquigarrow$ at most linear runtime complexity

Complexity Polynomial Interpretation (CPI)

$\mathcal{I}_f(x_1, \ldots, x_n) = a_1 x_1 + \ldots + a_n x_n + b$ for every constructor $f \in \Sigma_C$ with $b \in \mathbb{N}, a_i \in \{0, 1\}$

Termination and Complexity Analysis for Programs



- Proving Termination and Complexity of TRSs
- Proving Termination and Complexity of Probabilistic TRSs
- Disproving Termination of Probabilistic TRSs

Almost-Sure Termination Probabilistic TRSs

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \{\frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head}\}$

Almost-Sure Termination Probabilistic TRSs

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \left\{ \frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head} \right\}$

Distribution: $\mu = \{p_1 : t_1, \ldots, p_k : t_k\}$ with $p_1 + \cdots + p_k = 1$

Almost-Sure Termination Probabilistic TRSs

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \{\frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head}\}$

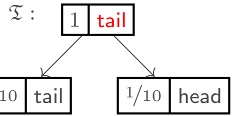
Distribution: $\mu = \{p_1 : t_1, \ldots, p_k : t_k\}$ with $p_1 + \cdots + p_k = 1$

$\mathfrak{T}$: | 1 | tail |

Almost-Sure Termination Probabilistic TRSs

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \{ \frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head} \}$

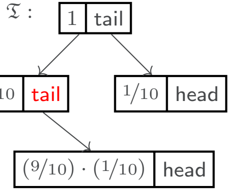
Distribution: $\mu = \{p_1 : t_1, \ldots, p_k : t_k\}$ with $p_1 + \cdots + p_k = 1$



Almost-Sure Termination Probabilistic TRSs

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \{\frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head}\}$

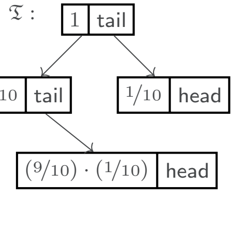
Distribution: $\mu = \{p_1 : t_1, \ldots, p_k : t_k\}$ with $p_1 + \cdots + p_k = 1$



Almost-Sure Termination Probabilistic TRSs

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \{\frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head}\}$

Distribution: $\mu = \{p_1 : t_1, \ldots, p_k : t_k\}$ with $p_1 + \cdots + p_k = 1$



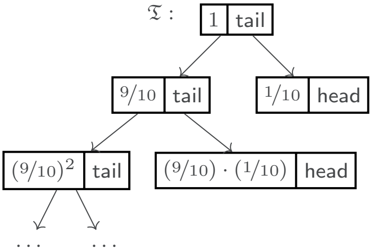
Almost-Sure Termination Probabilistic TRSs

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \{\frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head}\}$

Distribution: $\mu = \{p_1 : t_1, \ldots, p_k : t_k\}$ with $p_1 + \cdots + p_k = 1$

Probability of Termination:

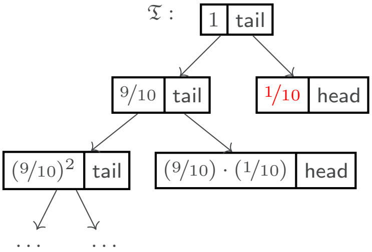
$|\mathfrak{T}|$



Almost-Sure Termination Probabilistic TRSs

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \left\{ \frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head} \right\}$

Distribution: $\mu = \{p_1 : t_1, \ldots, p_k : t_k\}$ with $p_1 + \cdots + p_k = 1$



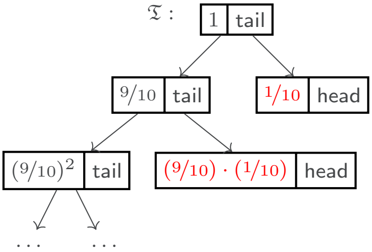
Probability of Termination:

$$|\mathfrak{T}| = {}^{1}\!/_{10} +$$

Almost-Sure Termination Probabilistic TRSs

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \{\frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head}\}$

Distribution: $\mu = \{p_1 : t_1, \ldots, p_k : t_k\}$ with $p_1 + \cdots + p_k = 1$



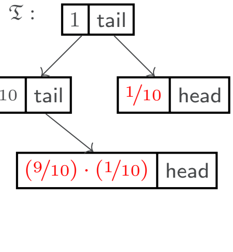
Probability of Termination:

$$|\mathfrak{T}| = {}^{1}/_{10} + {}^{9}/_{10} \cdot {}^{1}/_{10} +$$

Almost-Sure Termination Probabilistic TRSs

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \{\frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head}\}$

Distribution: $\mu = \{p_1 : t_1, \ldots, p_k : t_k\}$ with $p_1 + \cdots + p_k = 1$



Probability of Termination:

$$|\mathfrak{T}| = {}^1\!/_{10} + {}^9\!/_{10} \cdot {}^1\!/_{10} + ({}^9\!/_{10})^2 \cdot {}^1\!/_{10} + \ldots$$

Almost-Sure Termination Probabilistic TRSs

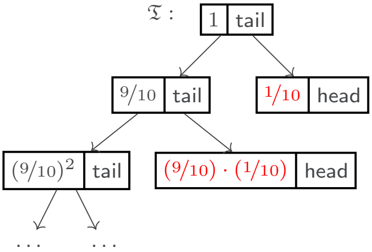
$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \{\frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head}\}$

Distribution: $\mu = \{p_1 : t_1, \ldots, p_k : t_k\}$ with $p_1 + \cdots + p_k = 1$

Probability of Termination:

$$|\mathfrak{T}| = {}^1\!/_{10} + {}^9\!/_{10} \cdot {}^1\!/_{10} + ({}^9\!/_{10})^2 \cdot {}^1\!/_{10} + \ldots$$

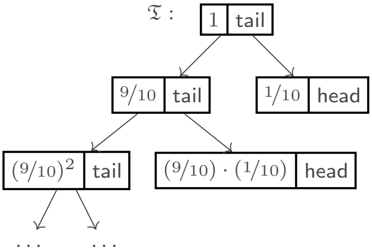
$$= {}^1\!/_{10} \cdot \sum_{n=0}^{\infty} (\tfrac{9}{10})^n = {}^1\!/_{10} \cdot 10 = 1$$



Almost-Sure Termination Probabilistic TRSs

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \left\{ \frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head} \right\}$

Distribution: $\mu = \{p_1 : t_1, \ldots, p_k : t_k\}$ with $p_1 + \cdots + p_k = 1$



Probability of Termination:

$$|\mathfrak{T}| = {}^1/_{10} + {}^9/_{10} \cdot {}^1/_{10} + ({}^9/_{10})^2 \cdot {}^1/_{10} + \ldots$$
$$= {}^1/_{10} \cdot \sum_{n=0}^{\infty} (\tfrac{9}{10})^n = {}^1/_{10} \cdot 10 = 1$$

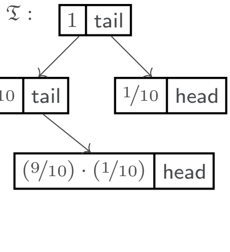
Almost-Sure Termination ($\mathtt{AST}$) [Avanzini & Dal Lago & Yamada'20]

PTRS $\mathcal{P}$ is $\mathtt{AST}$ if $|\mathfrak{T}| = 1$ for every $\mathfrak{T}$.

Expected Runtime of Probabilistic TRSs

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \left\{ \frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head} \right\}$

Distribution: $\mu = \{p_1 : t_1, \ldots, p_k : t_k\}$ with $p_1 + \cdots + p_k = 1$



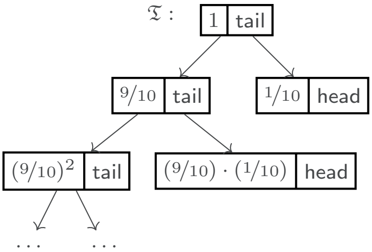
Expected Runtime of Probabilistic TRSs

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \left\{ \frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head} \right\}$

Distribution: $\mu = \{p_1 : t_1, \dots, p_k : t_k\}$ with $p_1 + \cdots + p_k = 1$

Expected Derivation Length:

$\mathrm{edl}(\mathfrak{T})$



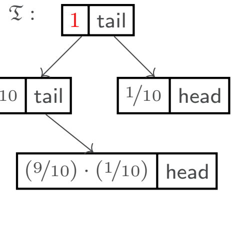
Expected Runtime of Probabilistic TRSs

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \left\{ \frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head} \right\}$

Distribution: $\mu = \{p_1 : t_1, \ldots, p_k : t_k\}$ with $p_1 + \cdots + p_k = 1$

Expected Derivation Length:

$$\mathrm{edl}(\mathfrak{T}) = 1 +$$



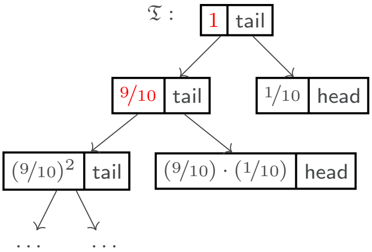
Expected Runtime of Probabilistic TRSs

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \left\{ \frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head} \right\}$

Distribution: $\mu = \{p_1 : t_1, \ldots, p_k : t_k\}$ with $p_1 + \cdots + p_k = 1$

Expected Derivation Length:

$$\mathrm{edl}(\mathfrak{T}) = 1 + \frac{9}{10} +$$



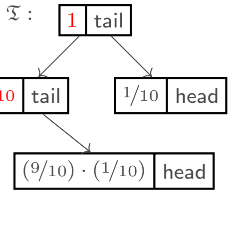
Expected Runtime of Probabilistic TRSs

$\mathcal{P}_{\mathsf{tail}}$: tail $\to \{ \frac{9}{10}$: tail, $\frac{1}{10}$: head$\}$

Distribution: $\mu = \{p_1 : t_1, \ldots, p_k : t_k\}$ with $p_1 + \cdots + p_k = 1$

Expected Derivation Length:

$$\mathrm{edl}(\mathfrak{T}) = 1 + \frac{9}{10} + \left(\frac{9}{10}\right)^2 + \ldots$$



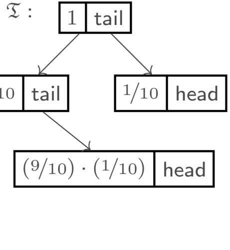
Expected Runtime of Probabilistic TRSs

$\mathcal{P}_{\mathsf{tail}}$: $\quad \mathsf{tail} \to \{\frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head}\}$

Distribution: $\mu = \{p_1 : t_1, \dots, p_k : t_k\}$ with $p_1 + \dots + p_k = 1$

Expected Derivation Length:

$$\mathrm{edl}(\mathfrak{T}) = 1 + \frac{9}{10} + (\frac{9}{10})^2 + \dots = \sum_{n=0}^{\infty} (\frac{9}{10})^n = 10$$



Expected Runtime of Probabilistic TRSs

$\mathcal{P}_{\mathsf{tail}}$: tail $\to \left\{ \frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head} \right\}$

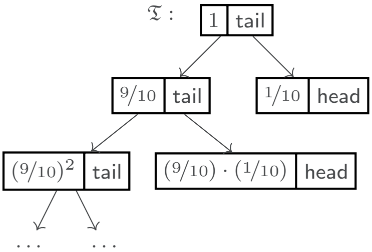
Distribution: $\mu = \{p_1 : t_1, \ldots, p_k : t_k\}$ with $p_1 + \cdots + p_k = 1$

Expected Derivation Length:

$$\mathrm{edl}(\mathfrak{T}) = 1 + \frac{9}{10} + \left(\frac{9}{10}\right)^2 + \ldots = \sum_{n=0}^{\infty} \left(\frac{9}{10}\right)^n = 10$$

Expected Derivation Height:

$$\mathrm{edh}_{\mathcal{P}_{\mathsf{tail}}}(\mathsf{tail}) = \sup\{\mathrm{edl}(\mathfrak{T}) \mid \mathfrak{T} \text{ starts with tail}\}$$



Expected Runtime of Probabilistic TRSs

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \left\{ \frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head} \right\}$

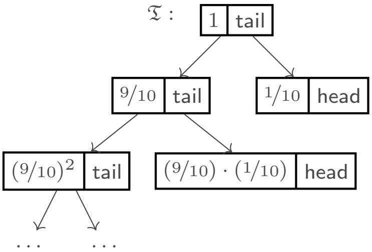
Distribution: $\mu = \{p_1 : t_1, \ldots, p_k : t_k\}$ with $p_1 + \cdots + p_k = 1$

Expected Derivation Length:

$$\mathrm{edl}(\mathfrak{T}) = 1 + \frac{9}{10} + \left(\frac{9}{10}\right)^2 + \ldots = \sum_{n=0}^{\infty} \left(\frac{9}{10}\right)^n = 10$$

Expected Derivation Height:

$$\mathrm{edh}_{\mathcal{P}_{\mathsf{tail}}}(\mathsf{tail}) = \sup\{\mathrm{edl}(\mathfrak{T}) \mid \mathfrak{T} \text{ starts with } \mathsf{tail}\}$$



Expected Runtime of Probabilistic TRSs

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \{\frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head}\}$

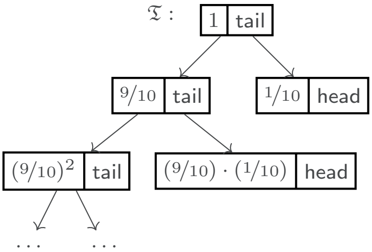
Distribution: $\mu = \{p_1 : t_1, \dots, p_k : t_k\}$ with $p_1 + \cdots + p_k = 1$

Expected Derivation Length:

$$\mathrm{edl}(\mathfrak{T}) = 1 + \frac{9}{10} + (\frac{9}{10})^2 + \dots = \sum_{n=0}^{\infty} (\frac{9}{10})^n = 10$$

Expected Derivation Height:

$$\mathrm{edh}_{\mathcal{P}_{\mathsf{tail}}}(\mathsf{tail}) = \sup\{\mathrm{edl}(\mathfrak{T}) \mid \mathfrak{T} \text{ starts with } \mathsf{tail}\} = 10$$



Expected Runtime of Probabilistic TRSs

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \{\frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head}\}$

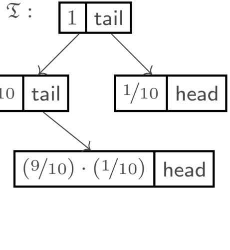
Distribution: $\mu = \{p_1 : t_1, \ldots, p_k : t_k\}$ with $p_1 + \cdots + p_k = 1$

Expected Derivation Length:

$$\mathrm{edl}(\mathfrak{T}) = 1 + \frac{9}{10} + (\frac{9}{10})^2 + \ldots = \sum_{n=0}^{\infty} (\frac{9}{10})^n = 10$$

Expected Derivation Height:

$$\mathrm{edh}_{\mathcal{P}_{\mathsf{tail}}}(\mathsf{tail}) = \sup\{\mathrm{edl}(\mathfrak{T}) \mid \mathfrak{T} \text{ starts with } \mathsf{tail}\} = 10$$



Expected Runtime Complexity:

Expected Runtime of Probabilistic TRSs

$\mathcal{P}_{\mathsf{tail}}$: $\quad \mathsf{tail} \to \{\frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head}\}$

Distribution: $\mu = \{p_1 : t_1, \ldots, p_k : t_k\}$ with $p_1 + \cdots + p_k = 1$

Expected Derivation Length:

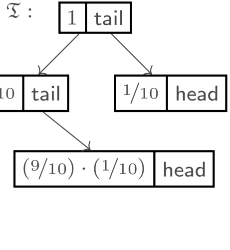
$$\mathrm{edl}(\mathfrak{T}) = 1 + \frac{9}{10} + \left(\frac{9}{10}\right)^2 + \ldots = \sum_{n=0}^{\infty} \left(\frac{9}{10}\right)^n = 10$$

Expected Derivation Height:

$$\mathrm{edh}_{\mathcal{P}_{\mathsf{tail}}}(\mathsf{tail}) = \sup\{\mathrm{edl}(\mathfrak{T}) \mid \mathfrak{T} \text{ starts with } \mathsf{tail}\} = 10$$

Expected Runtime Complexity:

$$\mathrm{erc}_{\mathcal{P}_{\mathsf{tail}}}(n) = \sup\{\mathrm{edh}_{\mathcal{P}_{\mathsf{tail}}}(t) \mid t \in \mathcal{T}_B, |t| \le n\}$$



Expected Runtime of Probabilistic TRSs

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \{\frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head}\}$

Distribution: $\mu = \{p_1 : t_1, \ldots, p_k : t_k\}$ with $p_1 + \cdots + p_k = 1$

Expected Derivation Length:

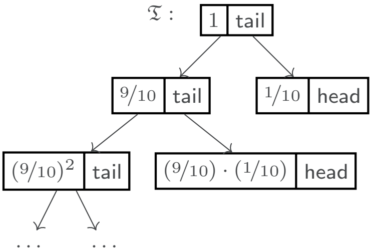
$$\mathrm{edl}(\mathfrak{T}) = 1 + \frac{9}{10} + (\frac{9}{10})^2 + \ldots = \sum_{n=0}^{\infty} (\frac{9}{10})^n = 10$$

Expected Derivation Height:

$$\mathrm{edh}_{\mathcal{P}_{\mathsf{tail}}}(\mathsf{tail}) = \sup\{\mathrm{edl}(\mathfrak{T}) \mid \mathfrak{T} \text{ starts with } \mathsf{tail}\} = 10$$

Expected Runtime Complexity:

$$\mathrm{erc}_{\mathcal{P}_{\mathsf{tail}}}(n) = \sup\{\mathrm{edh}_{\mathcal{P}_{\mathsf{tail}}}(t) \mid t \in \mathcal{T}_B, |t| \leq n\}$$



Strong Almost-Sure Termination (SAST) [Avanzini & Dal Lago & Yamada'20]

PTRS $\mathcal{P}$ is SAST if $\mathrm{edh}_{\mathcal{P}}(t)$ is finite for every start term t.

Expected Runtime of Probabilistic TRSs

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \{\frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head}\}$ constant complexity: Pol_0

Distribution: $\mu = \{p_1 : t_1, \ldots, p_k : t_k\}$ with $p_1 + \cdots + p_k = 1$

Expected Derivation Length:

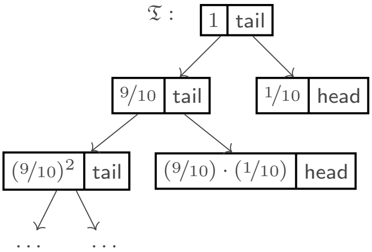
$$\mathrm{edl}(\mathfrak{T}) = 1 + \frac{9}{10} + (\frac{9}{10})^2 + \ldots = \sum_{n=0}^{\infty} (\frac{9}{10})^n = 10$$

Expected Derivation Height:

$$\mathrm{edh}_{\mathcal{P}_{\mathsf{tail}}}(\mathsf{tail}) = \sup\{\mathrm{edl}(\mathfrak{T}) \mid \mathfrak{T} \text{ starts with } \mathsf{tail}\} = 10$$

Expected Runtime Complexity:

$$\mathrm{erc}_{\mathcal{P}_{\mathsf{tail}}}(n) = \sup\{\mathrm{edh}_{\mathcal{P}_{\mathsf{tail}}}(t) \mid t \in \mathcal{T}_B, |t| \leq n\}$$



Strong Almost-Sure Termination (SAST) [Avanzini & Dal Lago & Yamada'20]

PTRS $\mathcal{P}$ is SAST if $\mathrm{edh}_{\mathcal{P}}(t)$ is finite for every start term t.

Proving Almost-Sure Termination

$\mathcal{P}_{\mathsf{tail}}$: $\quad \mathsf{tail} \to \left\{ \frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head} \right\}$

Proving Almost-Sure Termination

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \left\{ \frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head} \right\}$

Multilinear Polynomials

$x \cdot y$ is multilinear but x^2 is not.

Proving Almost-Sure Termination

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \left\{ \frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head} \right\}$

$\mathcal{I}_{\mathsf{tail}} = 1 \qquad \mathcal{I}_{\mathsf{head}} = 0$

Multilinear Polynomials

$x \cdot y$ is multilinear but x^2 is not.

Proving Almost-Sure Termination

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \left\{ \frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head} \right\}$

$\mathcal{I}_{\mathsf{tail}} = 1 \qquad \mathcal{I}_{\mathsf{head}} = 0$

Multilinear Polynomials

$x \cdot y$ is multilinear but x^2 is not.

Theorem: Natural & Monotonic & Multilinear $\mathcal{I}$ [Kassing & Giesl'24]

For all $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{R}$ let

- $\mathit{Pol}(\ell) > \mathit{Pol}(r_j)$ for some $1 \leq j \leq k$
- $\mathit{Pol}(\ell) \geq p_1 \cdot \mathit{Pol}(r_1) + \ldots + p_k \cdot \mathit{Pol}(r_k)$

Then $\mathcal{R}$ is AST.

Proving Almost-Sure Termination

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \left\{ \frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head} \right\}$

$\mathcal{I}_{\mathsf{tail}} = 1 \qquad \mathcal{I}_{\mathsf{head}} = 0$

Multilinear Polynomials

$x \cdot y$ is multilinear but x^2 is not.

Theorem: Natural & Monotonic & Multilinear $\mathcal{I}$ [Kassing & Giesl'24]

For all $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{R}$ let

- $Pol(\ell) > Pol(r_j)$ for some $1 \leq j \leq k$
- $Pol(\ell) \geq p_1 \cdot Pol(r_1) + \ldots + p_k \cdot Pol(r_k)$

Then $\mathcal{R}$ is AST.

Proving Almost-Sure Termination

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \left\{ \frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head} \right\}$

$\mathcal{I}_{\mathsf{tail}} = 1 \qquad \mathcal{I}_{\mathsf{head}} = 0$

Multilinear Polynomials

$x \cdot y$ is multilinear but x^2 is not.

Theorem: Natural & Monotonic & Multilinear $\mathcal{I}$ [Kassing & Giesl'24]

For all $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{R}$ let

- $Pol(\ell) > Pol(r_j)$ for some $1 \leq j \leq k$
- $Pol(\ell) \geq p_1 \cdot Pol(r_1) + \ldots + p_k \cdot Pol(r_k)$

Then $\mathcal{R}$ is AST.

Proving Almost-Sure Termination

$\mathcal{P}_{\text{tail}}$: $\mathcal{I}(\text{tail}) > \mathcal{I}(\text{head})$

$\mathcal{I}_{\text{tail}} = 1$ $\mathcal{I}_{\text{head}} = 0$

Multilinear Polynomials

$x \cdot y$ is multilinear but x^2 is not.

Theorem: Natural & Monotonic & Multilinear $\mathcal{I}$ [Kassing & Giesl'24]

For all $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{R}$ let

- $Pol(\ell) > Pol(r_j)$ for some $1 \leq j \leq k$
- $Pol(\ell) \geq p_1 \cdot Pol(r_1) + \ldots + p_k \cdot Pol(r_k)$

Then $\mathcal{R}$ is AST.

Proving Almost-Sure Termination

$\mathcal{P}_{\mathsf{tail}}$: $1 > 0$

$\mathcal{I}_{\mathsf{tail}} = 1 \qquad \mathcal{I}_{\mathsf{head}} = 0$

Multilinear Polynomials

$x \cdot y$ is multilinear but x^2 is not.

Theorem: Natural & Monotonic & Multilinear $\mathcal{I}$ [Kassing & Giesl'24]

For all $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{R}$ let

- $Pol(\ell) > Pol(r_j)$ for some $1 \leq j \leq k$
- $Pol(\ell) \geq p_1 \cdot Pol(r_1) + \ldots + p_k \cdot Pol(r_k)$

Then $\mathcal{R}$ is AST.

Proving Almost-Sure Termination

$\mathcal{P}_{\mathsf{tail}}$: $\mathcal{I}(\mathsf{tail}) > \mathbb{E}_{\mathcal{I}}(\{\frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head}\})$

$\mathcal{I}_{\mathsf{tail}} = 1 \qquad \mathcal{I}_{\mathsf{head}} = 0$

Multilinear Polynomials

$x \cdot y$ is multilinear but x^2 is not.

Theorem: Natural & Monotonic & Multilinear $\mathcal{I}$ [Kassing & Giesl'24]

For all $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{R}$ let

- $Pol(\ell) > Pol(r_j)$ for some $1 \leq j \leq k$
- $Pol(\ell) \geq p_1 \cdot Pol(r_1) + \ldots + p_k \cdot Pol(r_k)$

Then $\mathcal{R}$ is AST.

Proving Almost-Sure Termination

$\mathcal{P}_{\text{tail}}$: $\mathcal{I}(\text{tail}) > \frac{9}{10} \cdot \mathcal{I}(\text{tail}) + \frac{1}{10} \cdot \mathcal{I}(\text{head})$

$\mathcal{I}_{\text{tail}} = 1 \qquad \mathcal{I}_{\text{head}} = 0$

Multilinear Polynomials

$x \cdot y$ is multilinear but x^2 is not.

Theorem: Natural & Monotonic & Multilinear $\mathcal{I}$ [Kassing & Giesl'24]

For all $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{R}$ let

- $Pol(\ell) > Pol(r_j)$ for some $1 \leq j \leq k$
- $Pol(\ell) \geq p_1 \cdot Pol(r_1) + \ldots + p_k \cdot Pol(r_k)$

Then $\mathcal{R}$ is AST.

Proving Almost-Sure Termination

$\mathcal{P}_{\mathsf{tail}}$: $1 > \frac{9}{10}$

$\mathcal{I}_{\mathsf{tail}} = 1 \qquad \mathcal{I}_{\mathsf{head}} = 0$

Multilinear Polynomials

$x \cdot y$ is multilinear but x^2 is not.

Theorem: Natural & Monotonic & Multilinear $\mathcal{I}$ [Kassing & Giesl'24]

For all $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{R}$ let

- $Pol(\ell) > Pol(r_j)$ for some $1 \leq j \leq k$
- $Pol(\ell) \geq p_1 \cdot Pol(r_1) + \ldots + p_k \cdot Pol(r_k)$

Then $\mathcal{R}$ is AST.

Proving Almost-Sure Termination

$\mathcal{P}_{\mathsf{tail}}$: $1 > \frac{9}{10}$

$\mathcal{I}_{\mathsf{tail}} = 1 \qquad \mathcal{I}_{\mathsf{head}} = 0$

Multilinear Polynomials

$x \cdot y$ is multilinear but x^2 is not.

Theorem: Natural & Monotonic & Multilinear $\mathcal{I}$ [Kassing & Giesl'24]

For all $\ell \to \{p_1 : r_1, \ldots, p_k : r_k\} \in \mathcal{R}$ let

- $Pol(\ell) > Pol(r_j)$ for some $1 \leq j \leq k$
- $Pol(\ell) \geq p_1 \cdot Pol(r_1) + \ldots + p_k \cdot Pol(r_k)$

Then $\mathcal{R}$ is AST.

$\Rightarrow$ proves AST

Proving Expected Runtime Complexity

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \left\{ \frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head} \right\}$

Proving Expected Runtime Complexity

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \left\{ \frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head} \right\}$

Multilinear Polynomials

$x \cdot y$ is multilinear but x^2 is not.

Proving Expected Runtime Complexity

$\mathcal{P}_{\mathsf{tail}}$:

$$\mathsf{tail} \to \left\{ \tfrac{9}{10} : \mathsf{tail}, \tfrac{1}{10} : \mathsf{head} \right\}$$

$$\mathcal{I}_{\mathsf{tail}} = 1 \qquad \mathcal{I}_{\mathsf{head}} = 0$$

Multilinear Polynomials

$x \cdot y$ is multilinear but x^2 is not.

Proving Expected Runtime Complexity

$\mathcal{P}_{\mathsf{tail}}$: $\mathsf{tail} \to \left\{ \frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head} \right\}$

$$\mathcal{I}_{\mathsf{tail}} = 1 \qquad \mathcal{I}_{\mathsf{head}} = 0$$

Theorem: Natural & Monotonic & Multilinear $\mathcal{I}$ [Avanzini & Dal Lago & Yamada'20]

$\mathcal{P}$ is `SAST` if for all rules $\ell \to \mu$: $\mathcal{I}(\ell) > \mathbb{E}_{\mathcal{I}}(\mu)$

Proving Expected Runtime Complexity

$\mathcal{P}_{\mathsf{tail}}$: tail $\to \{ \frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head} \}$

$$\mathcal{I}_{\mathsf{tail}} = 1 \qquad \mathcal{I}_{\mathsf{head}} = 0$$

Theorem: Natural & Monotonic & Multilinear $\mathcal{I}$ [Avanzini & Dal Lago & Yamada'20]

$\mathcal{P}$ is SAST if for all rules $\ell \to \mu$: $\mathcal{I}(\ell) > \mathbb{E}_{\mathcal{I}}(\mu) = \sum_{1 \leq j \leq k} p_j \cdot \mathcal{I}(r_j)$ for $\mu = \{p_1 : r_1, \ldots, p_k : r_k\}$

Proving Expected Runtime Complexity

$\mathcal{P}_{\mathsf{tail}}$: $\mathcal{I}(\mathsf{tail}) > \mathbb{E}_{\mathcal{I}}(\{\frac{9}{10} : \mathsf{tail}, \frac{1}{10} : \mathsf{head}\})$

$\mathcal{I}_{\mathsf{tail}} = 1 \qquad \mathcal{I}_{\mathsf{head}} = 0$

Theorem: Natural & Monotonic & Multilinear $\mathcal{I}$ [Avanzini & Dal Lago & Yamada'20]

$\mathcal{P}$ is SAST if for all rules $\ell \to \mu$: $\mathcal{I}(\ell) > \mathbb{E}_{\mathcal{I}}(\mu) = \sum_{1 \leq j \leq k} p_j \cdot \mathcal{I}(r_j)$ for $\mu = \{p_1 : r_1, \ldots, p_k : r_k\}$

Proving Expected Runtime Complexity

$\mathcal{P}_{\mathsf{tail}}$: $\mathcal{I}(\mathsf{tail}) > \frac{9}{10} \cdot \mathcal{I}(\mathsf{tail}) + \frac{1}{10} \cdot \mathcal{I}(\mathsf{head})$

$\mathcal{I}_{\mathsf{tail}} = 1 \qquad \mathcal{I}_{\mathsf{head}} = 0$

Theorem: Natural & Monotonic & Multilinear $\mathcal{I}$ [Avanzini & Dal Lago & Yamada'20]

$\mathcal{P}$ is SAST if for all rules $\ell \to \mu$: $\mathcal{I}(\ell) > \mathbb{E}_{\mathcal{I}}(\mu) = \sum_{1 \leq j \leq k} p_j \cdot \mathcal{I}(r_j)$ for $\mu = \{p_1 : r_1, \ldots, p_k : r_k\}$

Proving Expected Runtime Complexity

$\mathcal{P}_{\mathsf{tail}}$: $$1 > \frac{9}{10}$$

$$\mathcal{I}_{\mathsf{tail}} = 1 \qquad \mathcal{I}_{\mathsf{head}} = 0$$

Theorem: Natural & Monotonic & Multilinear $\mathcal{I}$ [Avanzini & Dal Lago & Yamada'20]

$\mathcal{P}$ is SAST if for all rules $\ell \to \mu$: $\mathcal{I}(\ell) > \mathbb{E}_{\mathcal{I}}(\mu) = \sum_{1 \leq j \leq k} p_j \cdot \mathcal{I}(r_j)$ for $\mu = \{p_1 : r_1, \ldots, p_k : r_k\}$

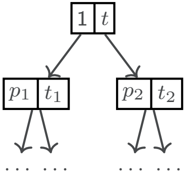
Proving Expected Runtime Complexity

$\mathcal{P}_{\mathsf{tail}}$: $1 > \frac{9}{10}$

$\mathcal{I}_{\mathsf{tail}} = 1 \qquad \mathcal{I}_{\mathsf{head}} = 0$

Theorem: Natural & Monotonic & Multilinear $\mathcal{I}$ [Avanzini & Dal Lago & Yamada'20]

$\mathcal{P}$ is SAST if for all rules $\ell \to \mu$: $\mathcal{I}(\ell) > \mathbb{E}_{\mathcal{I}}(\mu) = \sum_{1 \leq j \leq k} p_j \cdot \mathcal{I}(r_j)$ for $\mu = \{p_1 : r_1, \ldots, p_k : r_k\}$



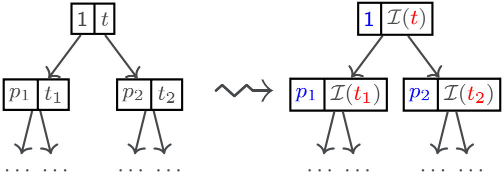
Proving Expected Runtime Complexity

$\mathcal{P}_{\text{tail}}$: $1 > \frac{9}{10}$

$\mathcal{I}_{\text{tail}} = 1 \qquad \mathcal{I}_{\text{head}} = 0$

Theorem: Natural & Monotonic & Multilinear $\mathcal{I}$ [Avanzini & Dal Lago & Yamada'20]

$\mathcal{P}$ is SAST if for all rules $\ell \to \mu$: $\mathcal{I}(\ell) > \mathbb{E}_{\mathcal{I}}(\mu) = \sum_{1 \leq j \leq k} p_j \cdot \mathcal{I}(r_j)$ for $\mu = \{p_1 : r_1, \ldots, p_k : r_k\}$



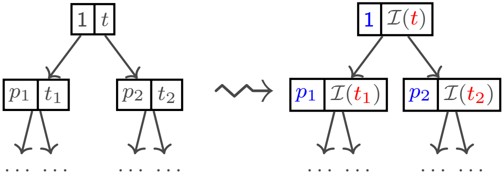
Proving Expected Runtime Complexity

$\mathcal{P}_{\text{tail}}$: $1 > \frac{9}{10}$

$\mathcal{I}_{\text{tail}} = 1 \qquad \mathcal{I}_{\text{head}} = 0$

Theorem: Natural & Monotonic & Multilinear $\mathcal{I}$ [Avanzini & Dal Lago & Yamada'20]

$\mathcal{P}$ is SAST if for all rules $\ell \to \mu$: $\mathcal{I}(\ell) > \mathbb{E}_{\mathcal{I}}(\mu) = \sum_{1 \leq j \leq k} p_j \cdot \mathcal{I}(r_j)$ for $\mu = \{p_1 : r_1, \ldots, p_k : r_k\}$



$\mathbb{E}_{\mathcal{I}}(\mu_0) = 1 \cdot \mathcal{I}(t)$

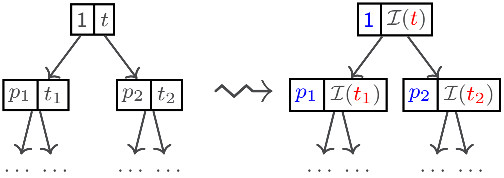
Proving Expected Runtime Complexity

$\mathcal{P}_{\mathsf{tail}}$: $1 > \frac{9}{10}$

$\mathcal{I}_{\mathsf{tail}} = 1 \qquad \mathcal{I}_{\mathsf{head}} = 0$

Theorem: Natural & Monotonic & Multilinear $\mathcal{I}$ [Avanzini & Dal Lago & Yamada'20]

$\mathcal{P}$ is SAST if for all rules $\ell \to \mu$: $\mathcal{I}(\ell) > \mathbb{E}_{\mathcal{I}}(\mu) = \sum_{1 \leq j \leq k} p_j \cdot \mathcal{I}(r_j)$ for $\mu = \{p_1 : r_1, \ldots, p_k : r_k\}$



$\mathbb{E}_{\mathcal{I}}(\mu_0) = 1 \cdot \mathcal{I}(t)$

$\mathbb{E}_{\mathcal{I}}(\mu_1) = p_1 \cdot \mathcal{I}(t_1) + p_2 \cdot \mathcal{I}(t_2)$

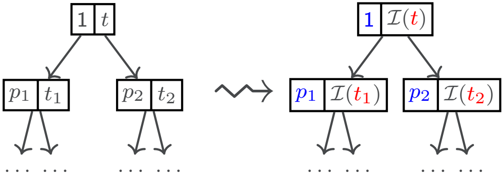
Proving Expected Runtime Complexity

$\mathcal{P}_{\text{tail}}$: $1 > \frac{9}{10}$

$\mathcal{I}_{\text{tail}} = 1 \qquad \mathcal{I}_{\text{head}} = 0$

Theorem: Natural & Monotonic & Multilinear $\mathcal{I}$ [Avanzini & Dal Lago & Yamada'20]

$\mathcal{P}$ is SAST if for all rules $\ell \to \mu$: $\mathcal{I}(\ell) > \mathbb{E}_{\mathcal{I}}(\mu) = \sum_{1 \leq j \leq k} p_j \cdot \mathcal{I}(r_j)$ for $\mu = \{p_1 : r_1, \ldots, p_k : r_k\}$



$\mathbb{E}_{\mathcal{I}}(\mu_0) = 1 \cdot \mathcal{I}(t)$

$\mathbb{E}_{\mathcal{I}}(\mu_1) = p_1 \cdot \mathcal{I}(t_1) + p_2 \cdot \mathcal{I}(t_2)$

$\ldots$

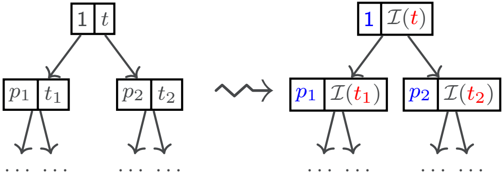
Proving Expected Runtime Complexity

$\mathcal{P}_{\text{tail}}$: $\quad 1 > \frac{9}{10} \quad \epsilon = {}^1\!/_{10}$

$\mathcal{I}_{\text{tail}} = 1 \qquad \mathcal{I}_{\text{head}} = 0$

Theorem: Natural & Monotonic & Multilinear $\mathcal{I}$ [Avanzini & Dal Lago & Yamada'20]

$\mathcal{P}$ is SAST if for all rules $\ell \to \mu$: $\mathcal{I}(\ell) > \mathbb{E}_{\mathcal{I}}(\mu) = \sum_{1 \leq j \leq k} p_j \cdot \mathcal{I}(r_j)$ for $\mu = \{p_1 : r_1, \ldots, p_k : r_k\}$



$\mathbb{E}_{\mathcal{I}}(\mu_0) = 1 \cdot \mathcal{I}(t)$

$> \epsilon \cdot 1 \qquad$ ϵ - minimal decrease for all rules

$\mathbb{E}_{\mathcal{I}}(\mu_1) = p_1 \cdot \mathcal{I}(t_1) + p_2 \cdot \mathcal{I}(t_2)$

$\ldots$

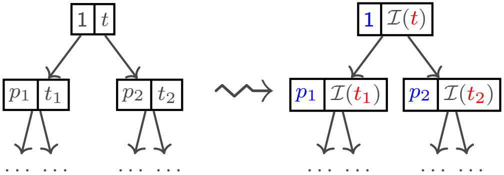
Proving Expected Runtime Complexity

$\mathcal{P}_{\text{tail}}$: $\quad 1 > \frac{9}{10} \quad \epsilon = 1/10$

$\mathcal{I}_{\text{tail}} = 1 \qquad \mathcal{I}_{\text{head}} = 0$

Theorem: Natural & Monotonic & Multilinear $\mathcal{I}$ [Avanzini & Dal Lago & Yamada'20]

$\mathcal{P}$ is SAST if for all rules $\ell \to \mu$: $\mathcal{I}(\ell) > \mathbb{E}_{\mathcal{I}}(\mu) = \sum_{1 \leq j \leq k} p_j \cdot \mathcal{I}(r_j)$ for $\mu = \{p_1 : r_1, \ldots, p_k : r_k\}$



$\mathbb{E}_{\mathcal{I}}(\mu_0) = 1 \cdot \mathcal{I}(t)$

$> \epsilon \cdot 1 \qquad \epsilon$ - minimal decrease for all rules

$\mathbb{E}_{\mathcal{I}}(\mu_1) = p_1 \cdot \mathcal{I}(t_1) + p_2 \cdot \mathcal{I}(t_2)$

$> \epsilon \cdot (p_1 + p_2)$

$\ldots$

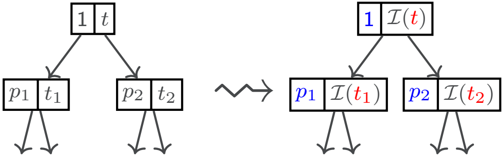
Proving Expected Runtime Complexity

$\mathcal{P}_{\mathsf{tail}}$: $1 > \frac{9}{10}$ $\epsilon = {}^{1}/_{10}$

$\mathcal{I}_{\mathsf{tail}} = 1 \qquad \mathcal{I}_{\mathsf{head}} = 0$

Theorem: Natural & Monotonic & Multilinear $\mathcal{I}$ [Avanzini & Dal Lago & Yamada'20]

$\mathcal{P}$ is SAST if for all rules $\ell \to \mu$: $\mathcal{I}(\ell) > \mathbb{E}_{\mathcal{I}}(\mu) = \sum_{1 \leq j \leq k} p_j \cdot \mathcal{I}(r_j)$ for $\mu = \{p_1 : r_1, \ldots, p_k : r_k\}$



$\mathbb{E}_{\mathcal{I}}(\mu_0) = 1 \cdot \mathcal{I}(t)$

$> \epsilon \cdot 1$ $\qquad$ ϵ - minimal decrease for all rules

$\mathbb{E}_{\mathcal{I}}(\mu_1) = p_1 \cdot \mathcal{I}(t_1) + p_2 \cdot \mathcal{I}(t_2)$

$> \epsilon \cdot (p_1 + p_2)$

Goal: Infer expected complexity from the highest degree of $\mathcal{I}$.

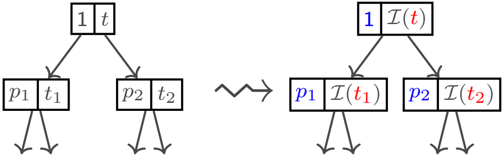
Proving Expected Runtime Complexity

$\mathcal{P}_{\text{tail}}$: $1 > \frac{9}{10}$ $\epsilon = {}^{1}/_{10}$

$\mathcal{I}_{\text{tail}} = 1$ $\mathcal{I}_{\text{head}} = 0$ $\rightsquigarrow$ constant complexity: Pol_0

Theorem: Natural & Monotonic & Multilinear $\mathcal{I}$ [Avanzini & Dal Lago & Yamada'20]

$\mathcal{P}$ is SAST if for all rules $\ell \to \mu$: $\mathcal{I}(\ell) > \mathbb{E}_{\mathcal{I}}(\mu) = \sum_{1 \leq j \leq k} p_j \cdot \mathcal{I}(r_j)$ for $\mu = \{p_1 : r_1, \ldots, p_k : r_k\}$



$\mathbb{E}_{\mathcal{I}}(\mu_0) = 1 \cdot \mathcal{I}(t)$

$> \epsilon \cdot 1$ ϵ - minimal decrease for all rules

$\mathbb{E}_{\mathcal{I}}(\mu_1) = p_1 \cdot \mathcal{I}(t_1) + p_2 \cdot \mathcal{I}(t_2)$

$> \epsilon \cdot (p_1 + p_2)$

Goal: Infer expected complexity from the highest degree of $\mathcal{I}$.

- Restrict to basic start terms and CPI

Wrap Up (Proving Termination and Complexity of Probabilistic TRSs)

Wrap Up (Proving Termination and Complexity of Probabilistic TRSs)

- **Almost-Sure Termination (AST)**
  - Termination with probability 1.
  - Proved via $\mathcal{I}(\ell) \geq \mathbb{E}(\mathcal{I}(\mu))$ and $\exists j. \mathcal{I}(\ell) > \mathcal{I}(r_j)$.

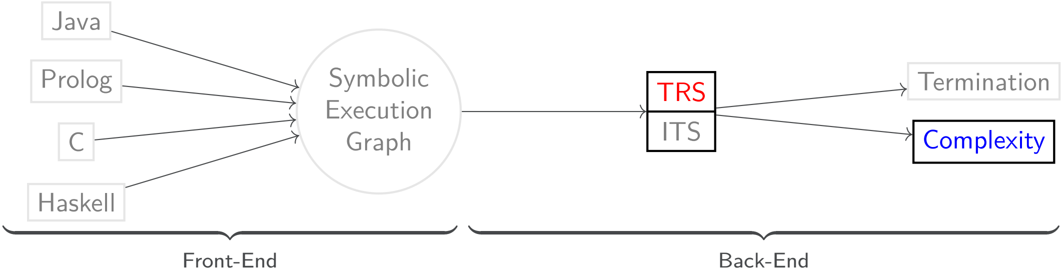
Wrap Up (Proving Termination and Complexity of Probabilistic TRSs)

- **Almost-Sure Termination (AST)**
  - Termination with probability 1.
  - Proved via $\mathcal{I}(\ell) \geq \mathbb{E}(\mathcal{I}(\mu))$ and $\exists j. \mathcal{I}(\ell) > \mathcal{I}(r_j)$.
- **Strong AST (SAST)**
  - Finite expected runtime.
  - Proved via $\mathcal{I}(\ell) > \mathbb{E}(\mathcal{I}(\mu))$.

Wrap Up (Proving Termination and Complexity of Probabilistic TRSs)

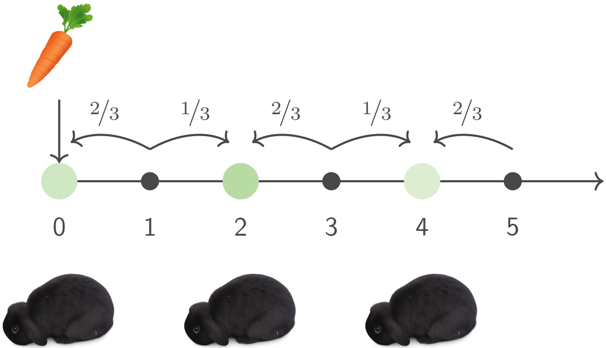
- **Almost-Sure Termination (AST)**
  - Termination with probability 1.
  - Proved via $\mathcal{I}(\ell) \geq \mathbb{E}(\mathcal{I}(\mu))$ and $\exists j. \mathcal{I}(\ell) > \mathcal{I}(r_j)$.
- **Strong AST (SAST)**
  - Finite expected runtime.
  - Proved via $\mathcal{I}(\ell) > \mathbb{E}(\mathcal{I}(\mu))$.
- **Expected Runtime Complexity**
  - Upper bound derived from degree of $\mathcal{I}$ for basic start terms and CPIs.

Termination and Complexity Analysis for Programs



- Proving Termination and Complexity of TRSs
- Proving Termination and Complexity of Probabilistic TRSs
- Disproving Termination of Probabilistic TRSs

Random Walk



$x \leftarrow 3$
while $x > 0$ **do**
$\quad x \leftarrow x - 1 \oplus_{2/3} x \leftarrow x + 1;$

- Does the bunny (program) always reach the carrot (terminate)?
- What is the probability of reaching the carrot (probability of termination)?
- What is the expected number of steps it takes to reach the carrot (expected runtime)?

Characterization of Random Walks

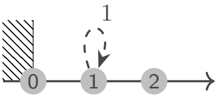
Random Walk μ

A function $\mu : \mathbb{Z} \to \mathbb{R}_{\geq 0}$ s. t. $\mathrm{Supp}(\mu) = \{x \in \mathbb{Z} \mid \mu(x) > 0\}$ is finite and $\sum_{x \in \mathrm{Supp}(\mu)} \mu(x) = 1$.
By $\mathbb{E}(\mu) = \sum_{x \in \mathrm{Supp}(\mu)} x \cdot \mu(x)$ we denote its *expected change*.

Characterization of Random Walks

Random Walk μ

A function $\mu : \mathbb{Z} \to \mathbb{R}_{\geq 0}$ s. t. $\mathrm{Supp}(\mu) = \{x \in \mathbb{Z} \mid \mu(x) > 0\}$ is finite and $\sum_{x \in \mathrm{Supp}(\mu)} \mu(x) = 1$.
By $\mathbb{E}(\mu) = \sum_{x \in \mathrm{Supp}(\mu)} x \cdot \mu(x)$ we denote its *expected change*.



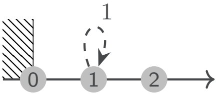
$\mu_1(0) = 1$

Loop Walk μ_1

Characterization of Random Walks

Random Walk μ

A function $\mu : \mathbb{Z} \to \mathbb{R}_{\geq 0}$ s. t. $\mathrm{Supp}(\mu) = \{x \in \mathbb{Z} \mid \mu(x) > 0\}$ is finite and $\sum_{x \in \mathrm{Supp}(\mu)} \mu(x) = 1$.
By $\mathbb{E}(\mu) = \sum_{x \in \mathrm{Supp}(\mu)} x \cdot \mu(x)$ we denote its *expected change*.



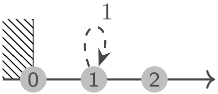
$\mu_1(0) = 1$

Loop Walk μ_1
($\mu(0) = 1$)

Characterization of Random Walks

Random Walk μ

A function $\mu : \mathbb{Z} \to \mathbb{R}_{\geq 0}$ s. t. $\mathrm{Supp}(\mu) = \{x \in \mathbb{Z} \mid \mu(x) > 0\}$ is finite and $\sum_{x \in \mathrm{Supp}(\mu)} \mu(x) = 1$.
By $\mathbb{E}(\mu) = \sum_{x \in \mathrm{Supp}(\mu)} x \cdot \mu(x)$ we denote its *expected change*.



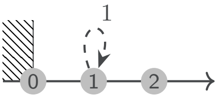
$\mu_1(0) = 1$

Loop Walk μ_1
$(\mu(0) = 1)$
$\Longrightarrow$ not AST and not SAST

Characterization of Random Walks

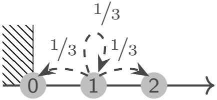
Random Walk μ

A function $\mu : \mathbb{Z} \to \mathbb{R}_{\geq 0}$ s. t. $\text{Supp}(\mu) = \{x \in \mathbb{Z} \mid \mu(x) > 0\}$ is finite and $\sum_{x \in \text{Supp}(\mu)} \mu(x) = 1$.
By $\mathbb{E}(\mu) = \sum_{x \in \text{Supp}(\mu)} x \cdot \mu(x)$ we denote its *expected change*.



$\mu_1(0) = 1$

Loop Walk μ_1
$(\mu(0) = 1)$
$\Longrightarrow$ not AST and not SAST



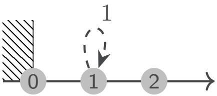
$\mu_2(-1) = 1/3$
$\mu_2(0) = 1/3$
$\mu_2(1) = 1/3$

Symmetric Random Walk μ_2

Characterization of Random Walks

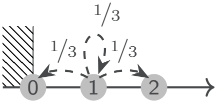
Random Walk μ

A function $\mu : \mathbb{Z} \to \mathbb{R}_{\geq 0}$ s. t. $\mathrm{Supp}(\mu) = \{x \in \mathbb{Z} \mid \mu(x) > 0\}$ is finite and $\sum_{x \in \mathrm{Supp}(\mu)} \mu(x) = 1$.
By $\mathbb{E}(\mu) = \sum_{x \in \mathrm{Supp}(\mu)} x \cdot \mu(x)$ we denote its *expected change*.



$\mu_1(0) = 1$

Loop Walk μ_1
$(\mu(0) = 1)$
$\implies$ not AST and not SAST



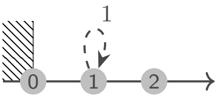
$\mu_2(-1) = 1/3$
$\mu_2(0) = 1/3$
$\mu_2(1) = 1/3$

Symmetric Random Walk μ_2
$(\mu(0) < 1$ and $\mathbb{E}(\mu) = 0)$

Characterization of Random Walks

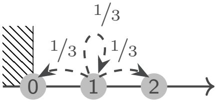
Random Walk μ

A function $\mu : \mathbb{Z} \to \mathbb{R}_{\geq 0}$ s. t. $\mathrm{Supp}(\mu) = \{x \in \mathbb{Z} \mid \mu(x) > 0\}$ is finite and $\sum_{x \in \mathrm{Supp}(\mu)} \mu(x) = 1$.
By $\mathbb{E}(\mu) = \sum_{x \in \mathrm{Supp}(\mu)} x \cdot \mu(x)$ we denote its *expected change*.



$\mu_1(0) = 1$

Loop Walk μ_1
$(\mu(0) = 1)$
$\Longrightarrow$ not AST and not SAST



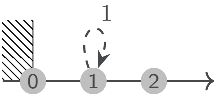
$\mu_2(-1) = 1/3$
$\mu_2(0) = 1/3$
$\mu_2(1) = 1/3$

Symmetric Random Walk μ_2
$(\mu(0) < 1$ and $\mathbb{E}(\mu) = 0)$
$\Longrightarrow$ AST and not SAST

Characterization of Random Walks

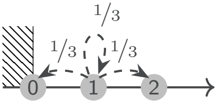
Random Walk μ

A function $\mu : \mathbb{Z} \to \mathbb{R}_{\geq 0}$ s. t. $\mathrm{Supp}(\mu) = \{x \in \mathbb{Z} \mid \mu(x) > 0\}$ is finite and $\sum_{x \in \mathrm{Supp}(\mu)} \mu(x) = 1$.
By $\mathbb{E}(\mu) = \sum_{x \in \mathrm{Supp}(\mu)} x \cdot \mu(x)$ we denote its *expected change*.



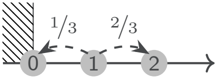
$\mu_1(0) = 1$

Loop Walk μ_1
$(\mu(0) = 1)$
$\Longrightarrow$ not AST and not SAST



$\mu_2(-1) = 1/3$
$\mu_2(0) = 1/3$
$\mu_2(1) = 1/3$

Symmetric Random Walk μ_2
$(\mu(0) < 1$ and $\mathbb{E}(\mu) = 0)$
$\Longrightarrow$ AST and not SAST



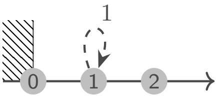
$\mu_3(-1) = 1/3$
$\mu_3(1) = 2/3$

Positively Biased Random Walk μ_3

Characterization of Random Walks

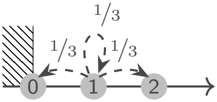
Random Walk μ

A function $\mu : \mathbb{Z} \to \mathbb{R}_{\geq 0}$ s. t. $\text{Supp}(\mu) = \{x \in \mathbb{Z} \mid \mu(x) > 0\}$ is finite and $\sum_{x \in \text{Supp}(\mu)} \mu(x) = 1$.
By $\mathbb{E}(\mu) = \sum_{x \in \text{Supp}(\mu)} x \cdot \mu(x)$ we denote its *expected change*.



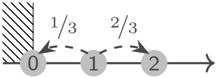
$\mu_1(0) = 1$

Loop Walk μ_1
$(\mu(0) = 1)$
$\Longrightarrow$ not AST and not SAST



$\mu_2(-1) = 1/3$
$\mu_2(0) = 1/3$
$\mu_2(1) = 1/3$

Symmetric Random Walk μ_2
$(\mu(0) < 1$ and $\mathbb{E}(\mu) = 0)$
$\Longrightarrow$ AST and not SAST



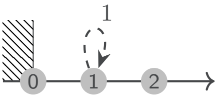
$\mu_3(-1) = 1/3$
$\mu_3(1) = 2/3$

Positively Biased Random Walk μ_3
$(\mu(0) < 1$ and $\mathbb{E}(\mu) > 0)$

Characterization of Random Walks

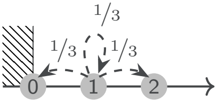
Random Walk μ

A function $\mu : \mathbb{Z} \to \mathbb{R}_{\geq 0}$ s. t. $\mathrm{Supp}(\mu) = \{x \in \mathbb{Z} \mid \mu(x) > 0\}$ is finite and $\sum_{x \in \mathrm{Supp}(\mu)} \mu(x) = 1$.
By $\mathbb{E}(\mu) = \sum_{x \in \mathrm{Supp}(\mu)} x \cdot \mu(x)$ we denote its *expected change*.



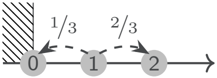
$\mu_1(0) = 1$

Loop Walk μ_1
($\mu(0) = 1$)
$\Longrightarrow$ not AST and not SAST



$\mu_2(-1) = 1/3$
$\mu_2(0) = 1/3$
$\mu_2(1) = 1/3$

Symmetric Random Walk μ_2
($\mu(0) < 1$ and $\mathbb{E}(\mu) = 0$)
$\Longrightarrow$ AST and not SAST



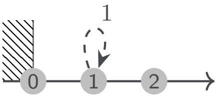
$\mu_3(-1) = 1/3$
$\mu_3(1) = 2/3$

Positively Biased Random Walk μ_3
($\mu(0) < 1$ and $\mathbb{E}(\mu) > 0$)
$\Longrightarrow$ not AST and not SAST

Characterization of Random Walks

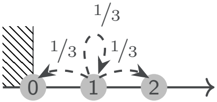
Random Walk μ

A function $\mu : \mathbb{Z} \to \mathbb{R}_{\geq 0}$ s. t. $\text{Supp}(\mu) = \{x \in \mathbb{Z} \mid \mu(x) > 0\}$ is finite and $\sum_{x \in \text{Supp}(\mu)} \mu(x) = 1$.
By $\mathbb{E}(\mu) = \sum_{x \in \text{Supp}(\mu)} x \cdot \mu(x)$ we denote its *expected change*.



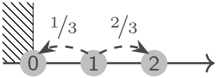
$\mu_1(0) = 1$

Loop Walk μ_1
$(\mu(0) = 1)$
$\implies$ not AST and not SAST



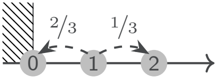
$\mu_2(-1) = 1/3$
$\mu_2(0) = 1/3$
$\mu_2(1) = 1/3$

Symmetric Random Walk μ_2
$(\mu(0) < 1$ and $\mathbb{E}(\mu) = 0)$
$\implies$ AST and not SAST



$\mu_3(-1) = 1/3$
$\mu_3(1) = 2/3$

Positively Biased Random Walk μ_3
$(\mu(0) < 1$ and $\mathbb{E}(\mu) > 0)$
$\implies$ not AST and not SAST



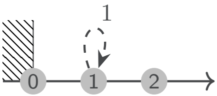
$\mu_4(-1) = 2/3$
$\mu_4(1) = 1/3$

Negatively Biased Random Walk μ_4

Characterization of Random Walks

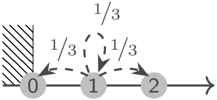
Random Walk μ

A function $\mu : \mathbb{Z} \to \mathbb{R}_{\geq 0}$ s. t. $\mathrm{Supp}(\mu) = \{x \in \mathbb{Z} \mid \mu(x) > 0\}$ is finite and $\sum_{x \in \mathrm{Supp}(\mu)} \mu(x) = 1$.
By $\mathbb{E}(\mu) = \sum_{x \in \mathrm{Supp}(\mu)} x \cdot \mu(x)$ we denote its *expected change*.



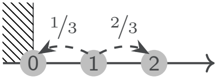
$\mu_1(0) = 1$

Loop Walk μ_1
$(\mu(0) = 1)$
$\implies$ not AST and not SAST



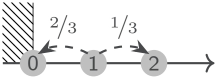
$\mu_2(-1) = 1/3$
$\mu_2(0) = 1/3$
$\mu_2(1) = 1/3$

Symmetric Random Walk μ_2
$(\mu(0) < 1$ and $\mathbb{E}(\mu) = 0)$
$\implies$ AST and not SAST



$\mu_3(-1) = 1/3$
$\mu_3(1) = 2/3$

Positively Biased Random Walk μ_3
$(\mu(0) < 1$ and $\mathbb{E}(\mu) > 0)$
$\implies$ not AST and not SAST



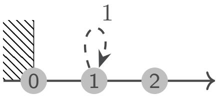
$\mu_4(-1) = 2/3$
$\mu_4(1) = 1/3$

Negatively Biased Random Walk μ_4
$(\mu(0) < 1$ and $\mathbb{E}(\mu) < 0)$

Characterization of Random Walks

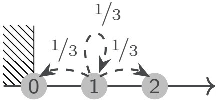
Random Walk μ

A function $\mu : \mathbb{Z} \to \mathbb{R}_{\geq 0}$ s. t. $\mathrm{Supp}(\mu) = \{x \in \mathbb{Z} \mid \mu(x) > 0\}$ is finite and $\sum_{x \in \mathrm{Supp}(\mu)} \mu(x) = 1$.
By $\mathbb{E}(\mu) = \sum_{x \in \mathrm{Supp}(\mu)} x \cdot \mu(x)$ we denote its *expected change*.



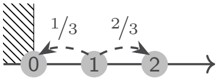
$\mu_1(0) = 1$

Loop Walk μ_1
($\mu(0) = 1$)
$\Longrightarrow$ not AST and not SAST



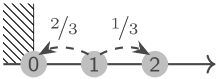
$\mu_2(-1) = 1/3$
$\mu_2(0) = 1/3$
$\mu_2(1) = 1/3$

Symmetric Random Walk μ_2
($\mu(0) < 1$ and $\mathbb{E}(\mu) = 0$)
$\Longrightarrow$ AST and not SAST



$\mu_3(-1) = 1/3$
$\mu_3(1) = 2/3$

Positively Biased Random Walk μ_3
($\mu(0) < 1$ and $\mathbb{E}(\mu) > 0$)
$\Longrightarrow$ not AST and not SAST



$\mu_4(-1) = 2/3$
$\mu_4(1) = 1/3$

Negatively Biased Random Walk μ_4
($\mu(0) < 1$ and $\mathbb{E}(\mu) < 0$)
$\Longrightarrow$ AST and SAST

Disproving AST and SAST of a PTRS

Disproving Termination of a TRS:

Disproving AST and SAST of a PTRS

Disproving Termination of a TRS: Find t, context C, and substitution σ s. t.

Disproving AST and SAST of a PTRS

Disproving Termination of a TRS: Find t, context C, and substitution σ s. t.

$$t \to C[t\sigma]$$

Disproving AST and SAST of a PTRS

Disproving Termination of a TRS: Find t, context C, and substitution σ s. t.

$$
\begin{array}{c}
t \\
\downarrow \\
C[t\sigma] \\
\downarrow \\
C[C[t\sigma]\sigma] \\
\downarrow \\
\ldots
\end{array}
$$

Disproving AST and SAST of a PTRS

Disproving Termination of a TRS: Find t, context C, and substitution σ s. t.

$$
\begin{array}{c}
t \\
\downarrow \\
C[t\sigma] \\
\downarrow \\
C[C[t\sigma]\sigma] \\
\downarrow \\
\dots
\end{array}
$$

$\mathcal{P}'_{\mathsf{double}}$:

$$
\begin{aligned}
\mathsf{double}(0) &\to 0 \\
\mathsf{double}(\mathsf{s}(x)) &\to \mathsf{s}(\mathsf{double}(\mathsf{s}(\mathsf{s}(x))))
\end{aligned}
$$

Disproving AST and SAST of a PTRS

Disproving Termination of a TRS: Find t, context C, and substitution σ s. t.

$$
\begin{array}{ccc}
t & \qquad & \mathsf{double}(\mathsf{s}(x)) \\
\downarrow & & \downarrow \\
C[t\sigma] & & \mathsf{s}(\mathsf{double}(\mathsf{s}(\mathsf{s}(x)))) \\
\downarrow & & \downarrow \\
C[C[t\sigma]\sigma] & & \mathsf{s}(\mathsf{s}(\mathsf{double}(\mathsf{s}(\mathsf{s}(\mathsf{s}(x)))))) \\
\downarrow & & \downarrow \\
\dots & & \dots
\end{array}
$$

$\mathcal{P}'_{\mathsf{double}}$:

$$
\begin{aligned}
\mathsf{double}(0) &\to 0 \\
\mathsf{double}(\mathsf{s}(x)) &\to \mathsf{s}(\mathsf{double}(\mathsf{s}(\mathsf{s}(x))))
\end{aligned}
$$

Disproving AST and SAST of a PTRS

Disproving Termination of a TRS: Find t, context C, and substitution σ s. t.

$$
\begin{array}{ccc}
t & \quad & \mathsf{double}(\mathsf{s}(x)) \\
\downarrow & & \downarrow \\
C[t\sigma] & & \mathsf{s}(\mathsf{double}(\mathsf{s}(\mathsf{s}(x)))) \\
\downarrow & & \downarrow \\
C[C[t\sigma]\sigma] & & \mathsf{s}(\mathsf{s}(\mathsf{double}(\mathsf{s}(\mathsf{s}(\mathsf{s}(x)))))) \\
\downarrow & & \downarrow \\
\ldots & & \ldots
\end{array}
$$

$\mathcal{P}'_{\mathsf{double}}$:

$$
\begin{aligned}
\mathsf{double}(0) &\to 0 \\
\mathsf{double}(\mathsf{s}(x)) &\to \mathsf{s}(\mathsf{double}(\mathsf{s}(\mathsf{s}(x))))
\end{aligned}
$$

Disproving AST and SAST of a PTRS

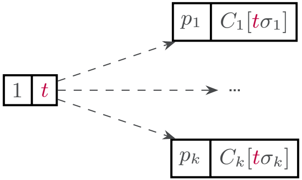
Disproving (S)AST of a PTRS (1.Idea):

Disproving AST and SAST of a PTRS

Disproving (S)AST of a PTRS (1.Idea): Find t, $C_1, \ldots, C_k$, and $\sigma_1, \ldots, \sigma_k$ s. t.

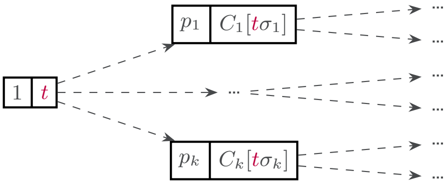
Disproving AST and SAST of a PTRS

Disproving (S)AST of a PTRS (1.Idea): Find t, $C_1, \ldots, C_k$, and $\sigma_1, \ldots, \sigma_k$ s. t.



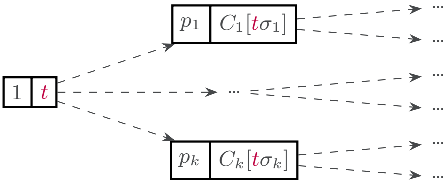
Disproving AST and SAST of a PTRS

Disproving (S)AST of a PTRS (1.Idea): Find t, $C_1, \ldots, C_k$, and $\sigma_1, \ldots, \sigma_k$ s. t.



Disproving AST and SAST of a PTRS

Disproving (S)AST of a PTRS (1.Idea): Find t, $C_1, \ldots, C_k$, and $\sigma_1, \ldots, \sigma_k$ s. t.

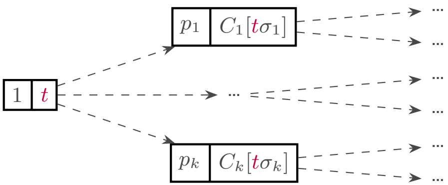


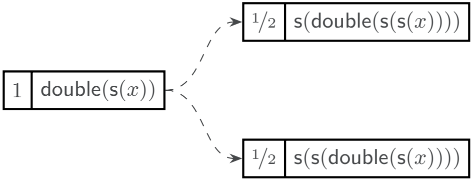
$\mathcal{P}'_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \{1/2 : \mathsf{s}(\mathsf{double}(\mathsf{s}(\mathsf{s}(x)))),\ 1/2 : \mathsf{s}(\mathsf{s}(\mathsf{double}(\mathsf{s}(x))))\}$$

Disproving AST and SAST of a PTRS

Disproving (S)AST of a PTRS (1.Idea): Find t, $C_1, \ldots, C_k$, and $\sigma_1, \ldots, \sigma_k$ s. t.





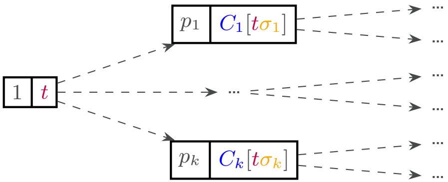
$\mathcal{P}'_{\mathsf{double}}$:

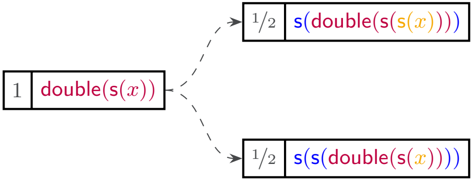
$$\mathsf{double}(0) \to 0$$

$$\mathsf{double}(\mathsf{s}(x)) \to \{1/2 : \mathsf{s}(\mathsf{double}(\mathsf{s}(\mathsf{s}(x)))),\ 1/2 : \mathsf{s}(\mathsf{s}(\mathsf{double}(\mathsf{s}(x))))\}$$

Disproving AST and SAST of a PTRS

Disproving (S)AST of a PTRS (1.Idea): Find t, $C_1, \ldots, C_k$, and $\sigma_1, \ldots, \sigma_k$ s. t.



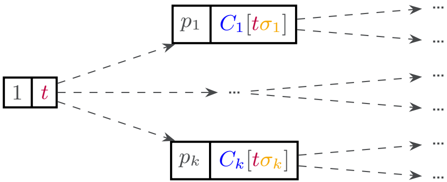


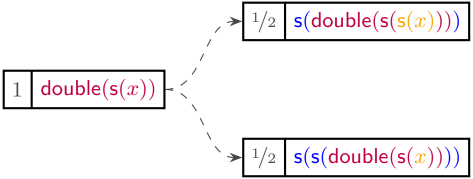
$\mathcal{P}'_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \{1/2 : \mathsf{s}(\mathsf{double}(\mathsf{s}(\mathsf{s}(x)))),\ 1/2 : \mathsf{s}(\mathsf{s}(\mathsf{double}(\mathsf{s}(x))))\}$$

Disproving AST and SAST of a PTRS

Disproving (S)AST of a PTRS (1.Idea): Find t, $C_1, \ldots, C_k$, and $\sigma_1, \ldots, \sigma_k$ s. t.

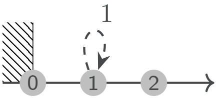




$\mathcal{P}'_{\mathsf{double}}$:

$$\mathsf{double}(0) \to 0$$
$$\mathsf{double}(\mathsf{s}(x)) \to \{1/2 : \mathsf{s}(\mathsf{double}(\mathsf{s}(\mathsf{s}(x)))),\ 1/2 : \mathsf{s}(\mathsf{s}(\mathsf{double}(\mathsf{s}(x))))\}$$

$\Rightarrow$ Embedding loop walks



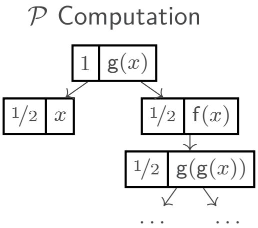
$\mu_1(0) = 1$

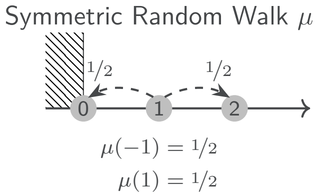
Disproving AST and SAST of a PTRS

Disproving (S)AST of a PTRS (2.Idea): Try to embed symmetric or positively biased random walks

Disproving AST and SAST of a PTRS

Disproving (S)AST of a PTRS (2.Idea): Try to embed symmetric or positively biased random walks

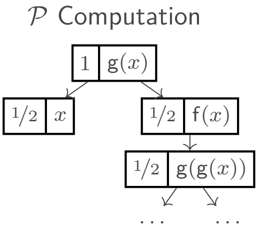


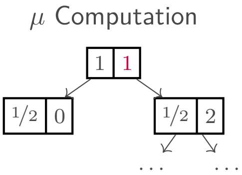


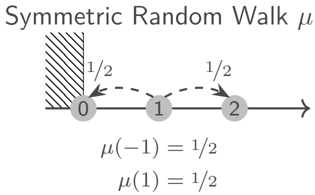
- What does it mean to find a random walk?

Disproving AST and SAST of a PTRS

Disproving (S)AST of a PTRS (2.Idea): Try to embed symmetric or positively biased random walks



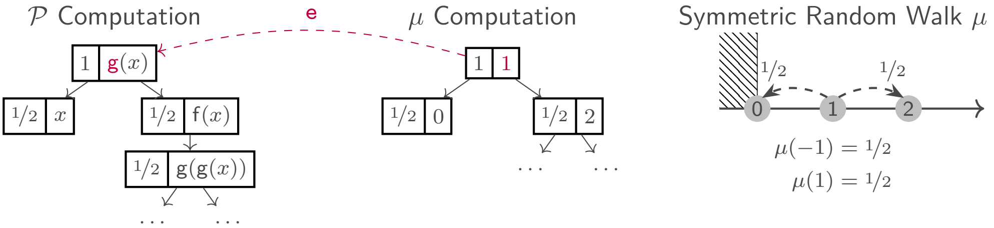




- What does it mean to find a random walk?

Disproving AST and SAST of a PTRS

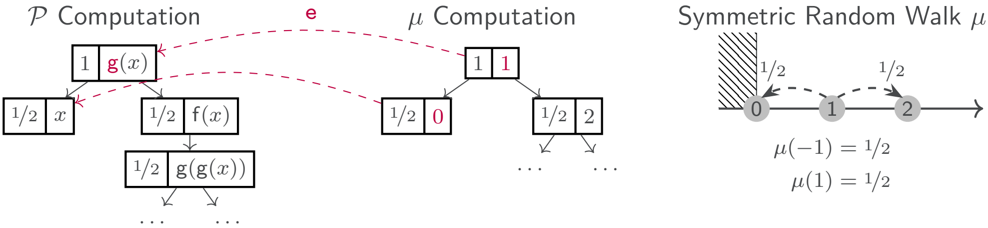
Disproving (S)AST of a PTRS (2.Idea): Try to embed symmetric or positively biased random walks



- What does it mean to find a random walk?

Disproving AST and SAST of a PTRS

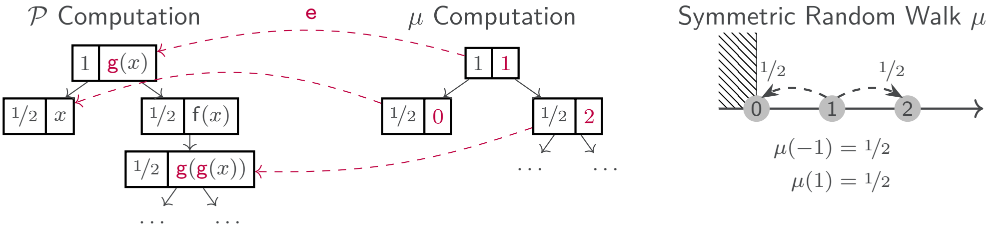
Disproving (S)AST of a PTRS (2.Idea): Try to embed symmetric or positively biased random walks



- What does it mean to find a random walk?

Disproving AST and SAST of a PTRS

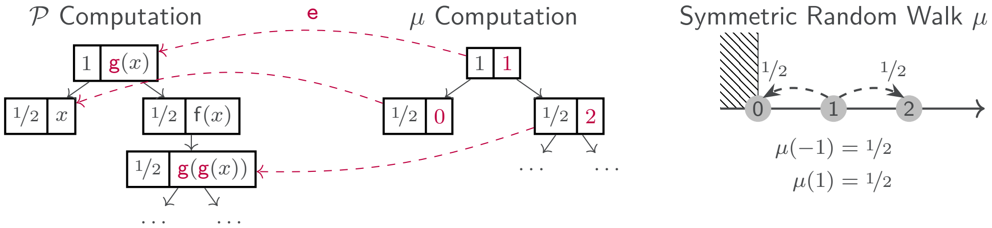
Disproving (S)AST of a PTRS (2.Idea): Try to embed symmetric or positively biased random walks



- What does it mean to find a random walk?

Disproving AST and SAST of a PTRS

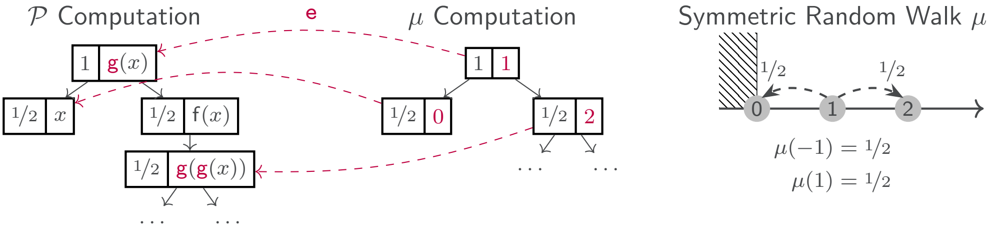
Disproving (S)AST of a PTRS (2.Idea): Try to embed symmetric or positively biased random walks



- What does it mean to find a random walk?
  ⇝ Embedding e from computation of μ to computation of $\mathcal{P}$

Disproving AST and SAST of a PTRS

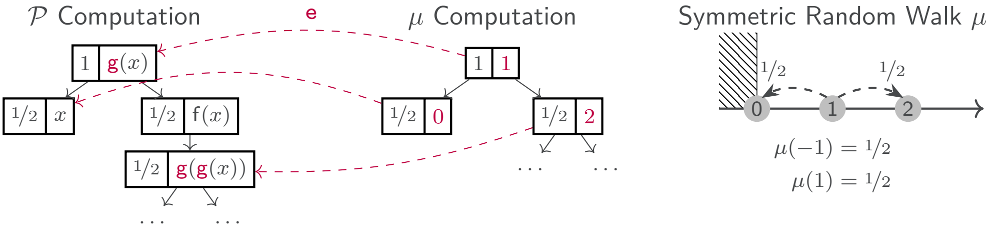
Disproving (S)AST of a PTRS (2.Idea): Try to embed symmetric or positively biased random walks



- What does it mean to find a random walk?
  $\rightsquigarrow$ Embedding e from computation of μ to computation of $\mathcal{P}$
- What and How to Count?

Disproving AST and SAST of a PTRS

Disproving (S)AST of a PTRS (2.Idea): Try to embed symmetric or positively biased random walks



- What does it mean to find a random walk?
  $\rightsquigarrow$ Embedding e from computation of μ to computation of $\mathcal{P}$
- What and How to Count? $\rightsquigarrow$ Count term occurrences

Counting Term Occurrences

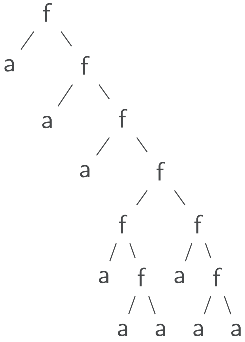
- Find the maximal number t occurs in s (denoted $\mathrm{maxNO}(t, s)$)

Counting Term Occurrences

- Find the maximal number t occurs in s (denoted $\mathrm{maxNO}(t, s)$)

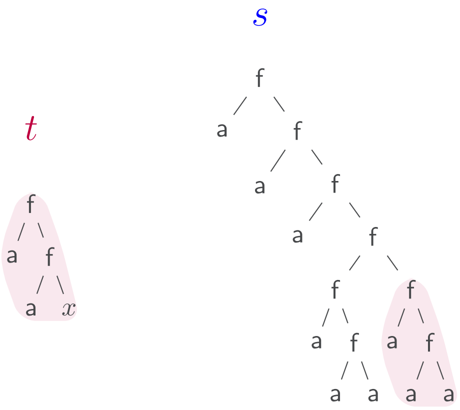
t

s



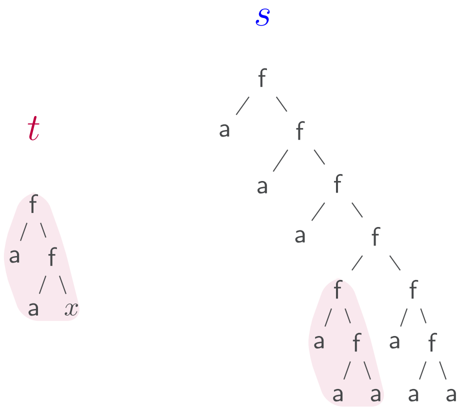
Counting Term Occurrences

- Find the maximal number t occurs in s (denoted $\mathrm{maxNO}(t, s)$)



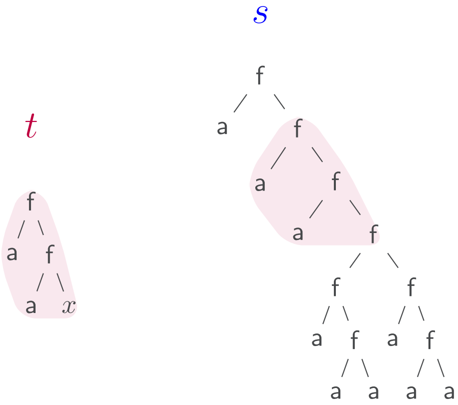
Counting Term Occurrences

- Find the maximal number t occurs in s (denoted $\mathrm{maxNO}(t, s)$)



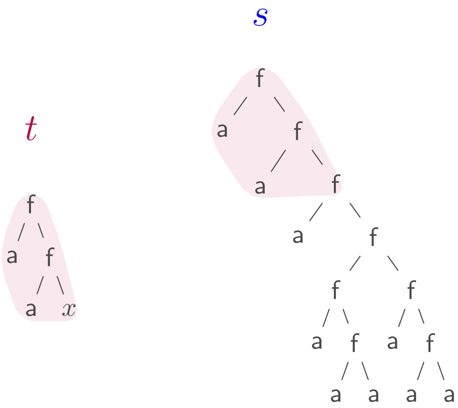
Counting Term Occurrences

- Find the maximal number t occurs in s (denoted $\mathrm{maxNO}(t, s)$)



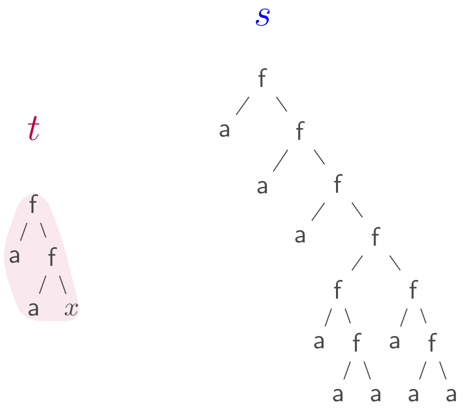
Counting Term Occurrences

- Find the maximal number t occurs in s (denoted $\mathrm{maxNO}(t, s)$)



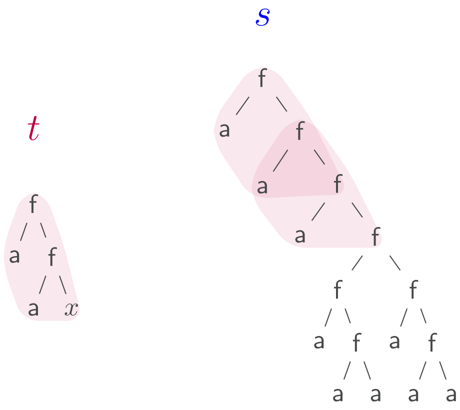
Counting Term Occurrences

- Find the maximal number t occurs in s (denoted $\mathrm{maxNO}(t, s)$)
- Only count non-overlapping occurrences!



Counting Term Occurrences

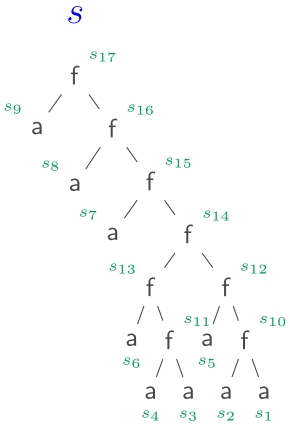
- Find the maximal number t occurs in s (denoted $\mathrm{maxNO}(t, s)$)
- Only count non-overlapping occurrences!



Counting Term Occurrences

- Find the maximal number t occurs in s (denoted $\mathrm{maxNO}(t, s)$)
- Only count non-overlapping occurrences!

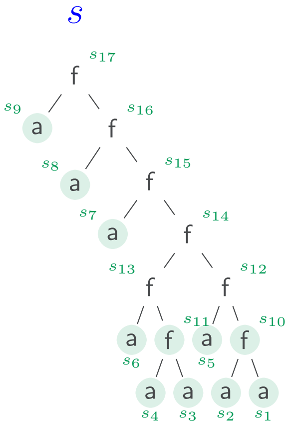
Subterm $s' \lhd s$	$\mathrm{maxNO}(t, s')$



Counting Term Occurrences

- Find the maximal number t occurs in s (denoted $\mathrm{maxNO}(t, s)$)
- Only count non-overlapping occurrences!

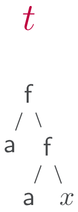
Subterm $s' \lhd s$	$\mathrm{maxNO}(t, s')$
s_1–s_{11}	0

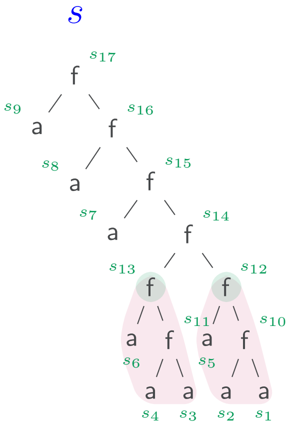


Counting Term Occurrences

- Find the maximal number t occurs in s (denoted $\mathrm{maxNO}(t, s)$)
- Only count non-overlapping occurrences!

Subterm $s' \lhd s$	$\mathrm{maxNO}(t, s')$
s_1–s_{11}	0
s_{12}–s_{13}	1

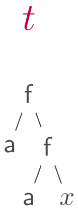


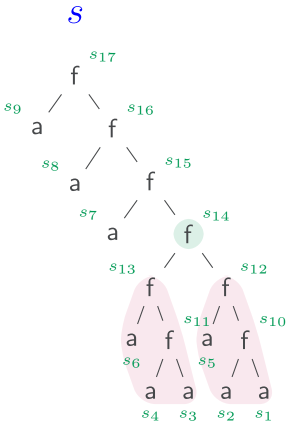


Counting Term Occurrences

- Find the maximal number t occurs in s (denoted $\mathrm{maxNO}(t, s)$)
- Only count non-overlapping occurrences!

Subterm $s' \lhd s$	$\mathrm{maxNO}(t, s')$
s_1–s_{11}	0
s_{12}–s_{13}	1
s_{14}	2

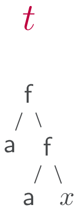


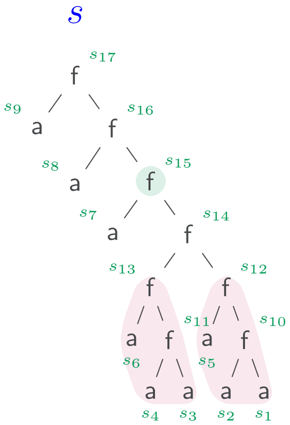


Counting Term Occurrences

- Find the maximal number t occurs in s (denoted $\mathrm{maxNO}(t, s)$)
- Only count non-overlapping occurrences!

Subterm $s' \lhd s$	$\mathrm{maxNO}(t, s')$
s_1–s_{11}	0
s_{12}–s_{13}	1
s_{14}	2
s_{15}	2

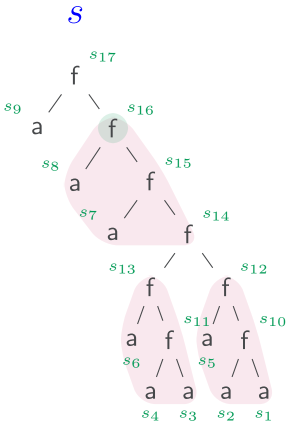




Counting Term Occurrences

- Find the maximal number t occurs in s (denoted $\mathrm{maxNO}(t, s)$)
- Only count non-overlapping occurrences!

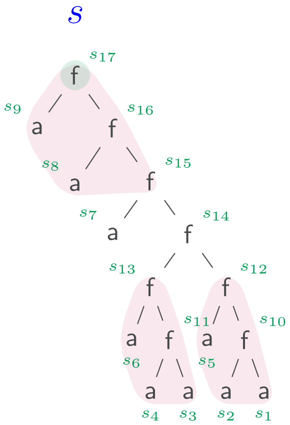
Subterm $s' \lhd s$	$\mathrm{maxNO}(t, s')$
s_1–s_{11}	0
s_{12}–s_{13}	1
s_{14}	2
s_{15}	2
s_{16}	3



Counting Term Occurrences

- Find the maximal number t occurs in s (denoted $\mathrm{maxNO}(t, s)$)
- Only count non-overlapping occurrences!

Subterm $s' \lhd s$	$\mathrm{maxNO}(t, s')$
s_1–s_{11}	0
s_{12}–s_{13}	1
s_{14}	2
s_{15}	2
s_{16}	3
s_{17}	3

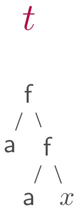


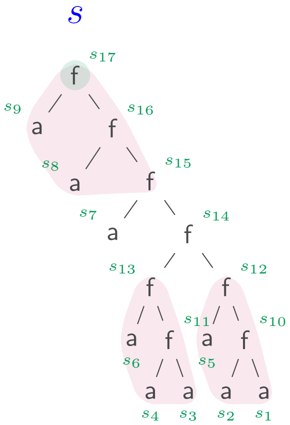
Counting Term Occurrences

- Find the maximal number t occurs in s (denoted $\mathrm{maxNO}(t, s)$)
- Only count non-overlapping occurrences!

$$\mathrm{maxNO}(t, s) = \alpha_s = \alpha_{s_{17}} = 3$$

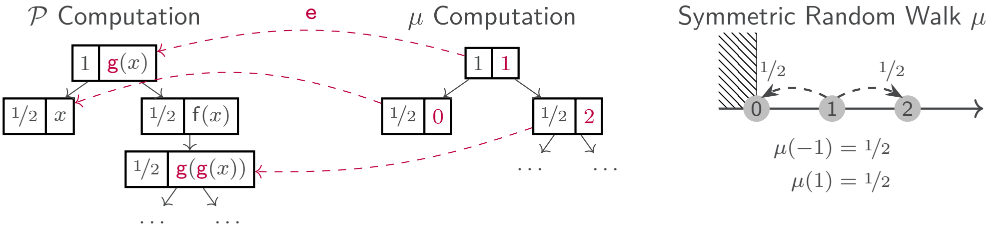
Subterm $s' \lhd s$	$\mathrm{maxNO}(t, s')$
s_1–s_{11}	0
s_{12}–s_{13}	1
s_{14}	2
s_{15}	2
s_{16}	3
s_{17}	3





Disproving AST and SAST of a PTRS

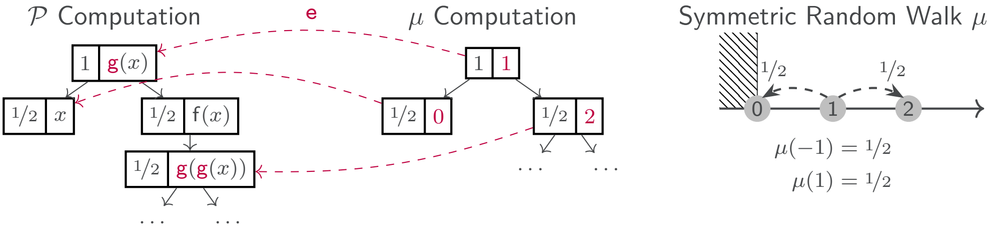
Disproving (S)AST of a PTRS (2.Idea): Try to find random walks within the computation



- What does it mean to find a random walk?
  $\rightsquigarrow$ Embedding e from computation of μ to computation of $\mathcal{P}$
  - What and How to Count? $\rightsquigarrow$ Count term occurrences

Disproving AST and SAST of a PTRS

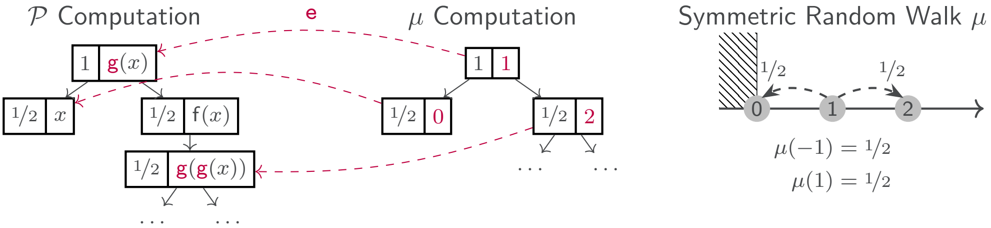
Disproving (S)AST of a PTRS (2.Idea): Try to find random walks within the computation



- What does it mean to find a random walk?
  $\rightsquigarrow$ Embedding e from computation of μ to computation of $\mathcal{P}$
  - What and How to Count? $\rightsquigarrow$ Count term occurrences
- How to construct the infinite computation?

Disproving AST and SAST of a PTRS

Disproving (S)AST of a PTRS (2.Idea): Try to find random walks within the computation



- What does it mean to find a random walk?
  $\rightsquigarrow$ Embedding e from computation of μ to computation of $\mathcal{P}$
  - What and How to Count? $\rightsquigarrow$ Count term occurrences
- How to construct the infinite computation? $\rightsquigarrow$ Iteratively rewrite the innermost occurrence

Construct the Infinite Computation

Theorem: Embedding Random Walks [Kassing & Nagel & Schlecht & Giesl (In Review)]

Construct the Infinite Computation

Theorem: Embedding Random Walks [Kassing & Nagel & Schlecht & Giesl (In Review)]

$\mathcal{P}$ a PTRS

Construct the Infinite Computation

Theorem: Embedding Random Walks [Kassing & Nagel & Schlecht & Giesl (In Review)]

$\mathcal{P}$ a PTRS, t a linear term

Construct the Infinite Computation

Theorem: Embedding Random Walks [Kassing & Nagel & Schlecht & Giesl (In Review)]

$\mathcal{P}$ a PTRS, t a linear term, and (finite) computation $\mathfrak{T}$ starting with t.

Construct the Infinite Computation

Theorem: Embedding Random Walks [Kassing & Nagel & Schlecht & Giesl (In Review)]

$\mathcal{P}$ a PTRS, t a linear term, and (finite) computation $\mathfrak{T}$ starting with t.
If we have $\sum_{v \in \mathrm{Leaf}(\mathfrak{T})} p_v \cdot \mathrm{maxNO}(t, t_v) \underset{(-)}{\geq} 1$, then $\mathcal{P}$ is not (S)AST.

Construct the Infinite Computation

Theorem: Embedding Random Walks [Kassing & Nagel & Schlecht & Giesl (In Review)]

$\mathcal{P}$ a PTRS, t a linear term, and (finite) computation $\mathfrak{T}$ starting with t.
If we have $\sum_{v \in \mathrm{Leaf}(\mathfrak{T})} p_v \cdot \mathrm{maxNO}(t, t_v) \underset{(-)}{\geq} 1$, then $\mathcal{P}$ is not (S)AST.

Why do we need non-overlapping occurrences?

Construct the Infinite Computation

Theorem: Embedding Random Walks [Kassing & Nagel & Schlecht & Giesl (In Review)]

$\mathcal{P}$ a PTRS, t a linear term, and (finite) computation $\mathfrak{T}$ starting with t.
If we have $\sum_{v \in \mathrm{Leaf}(\mathfrak{T})} p_v \cdot \mathrm{maxNO}(t, t_v) \underset{(-)}{\geq} 1$, then $\mathcal{P}$ is not (S)AST.

Why do we need non-overlapping occurrences?

$\mathcal{P}$:

$$\mathsf{g}(\mathsf{g}(x)) \to \{1/3 : x, 2/3 : \mathsf{g}(\mathsf{g}(\mathsf{g}(x)))\}$$

Construct the Infinite Computation

Theorem: Embedding Random Walks [Kassing & Nagel & Schlecht & Giesl (In Review)]

$\mathcal{P}$ a PTRS, t a linear term, and (finite) computation $\mathfrak{T}$ starting with t.
If we have $\sum_{v \in \mathrm{Leaf}(\mathfrak{T})} p_v \cdot \mathrm{maxNO}(t, t_v) \underset{(-)}{\geq} 1$, then $\mathcal{P}$ is not (S)AST.

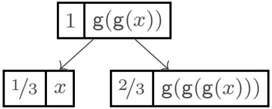
Why do we need non-overlapping occurrences?

$\mathcal{P}$:

$$\mathsf{g}(\mathsf{g}(x)) \to \{1/3 : x, 2/3 : \mathsf{g}(\mathsf{g}(\mathsf{g}(x)))\}$$

$t = \mathsf{g}(\mathsf{g}(x))$

Computation $\mathfrak{T}$



Construct the Infinite Computation

Theorem: Embedding Random Walks [Kassing & Nagel & Schlecht & Giesl (In Review)]

$\mathcal{P}$ a PTRS, t a linear term, and (finite) computation $\mathfrak{T}$ starting with t.
If we have $\sum_{v \in \mathrm{Leaf}(\mathfrak{T})} p_v \cdot \mathrm{maxNO}(t, t_v) \underset{(-)}{\geq} 1$, then $\mathcal{P}$ is not (S)AST.

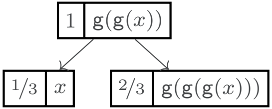
Why do we need non-overlapping occurrences?

$\mathcal{P}$:

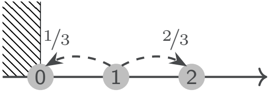
$$\mathsf{g}(\mathsf{g}(x)) \to \{1/3 : x, 2/3 : \mathsf{g}(\mathsf{g}(\mathsf{g}(x)))\}$$

$t = \mathsf{g}(\mathsf{g}(x))$

Computation $\mathfrak{T}$



Positively Biased Random Walk μ



$\mu(-1) = 1/3$

$\mu(1) = 2/3$

Construct the Infinite Computation

Theorem: Embedding Random Walks [Kassing & Nagel & Schlecht & Giesl (In Review)]

$\mathcal{P}$ a PTRS, t a linear term, and (finite) computation $\mathfrak{T}$ starting with t.
If we have $\sum_{v \in \mathrm{Leaf}(\mathfrak{T})} p_v \cdot \mathrm{maxNO}(t, t_v) \underset{(-)}{\geq} 1$, then $\mathcal{P}$ is not (S)AST.

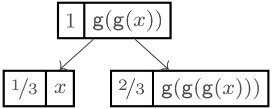
Why do we need non-overlapping occurrences?

$\mathcal{P}$:

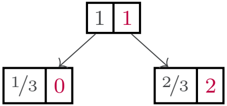
$$\mathsf{g}(\mathsf{g}(x)) \to \{1/3 : x, 2/3 : \mathsf{g}(\mathsf{g}(\mathsf{g}(x)))\}$$

$t = \mathsf{g}(\mathsf{g}(x))$

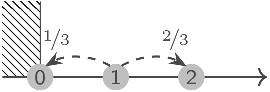
Computation $\mathfrak{T}$



μ Computation



Positively Biased Random Walk μ



$\mu(-1) = 1/3$

$\mu(1) = 2/3$

Construct the Infinite Computation

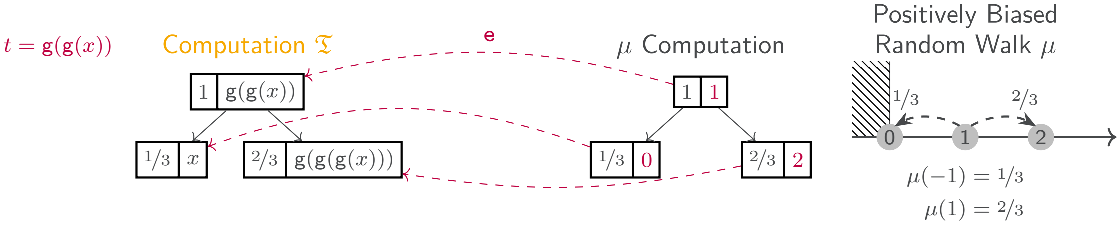
Theorem: Embedding Random Walks [Kassing & Nagel & Schlecht & Giesl (In Review)]

$\mathcal{P}$ a PTRS, t a linear term, and (finite) computation $\mathfrak{T}$ starting with t.
If we have $\sum_{v \in \mathrm{Leaf}(\mathfrak{T})} p_v \cdot \mathrm{maxNO}(t, t_v) \underset{(-)}{\geq} 1$, then $\mathcal{P}$ is not (S)AST.

Why do we need non-overlapping occurrences?

$\mathcal{P}$:

$$\mathsf{g}(\mathsf{g}(x)) \to \{{}^1/_3 : x, {}^2/_3 : \mathsf{g}(\mathsf{g}(\mathsf{g}(x)))\}$$



Construct the Infinite Computation

Theorem: Embedding Random Walks [Kassing & Nagel & Schlecht & Giesl (In Review)]

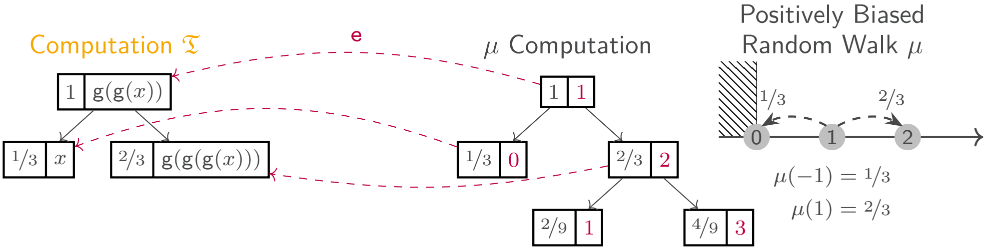
$\mathcal{P}$ a PTRS, t a linear term, and (finite) computation $\mathfrak{T}$ starting with t.
If we have $\sum_{v \in \mathrm{Leaf}(\mathfrak{T})} p_v \cdot \mathrm{maxNO}(t, t_v) \underset{(-)}{\geq} 1$, then $\mathcal{P}$ is not (S)AST.

Why do we need non-overlapping occurrences?

$\mathcal{P}$:

$$\mathsf{g}(\mathsf{g}(x)) \to \{1/3 : x, 2/3 : \mathsf{g}(\mathsf{g}(\mathsf{g}(x)))\}$$

$t = \mathsf{g}(\mathsf{g}(x))$



Construct the Infinite Computation

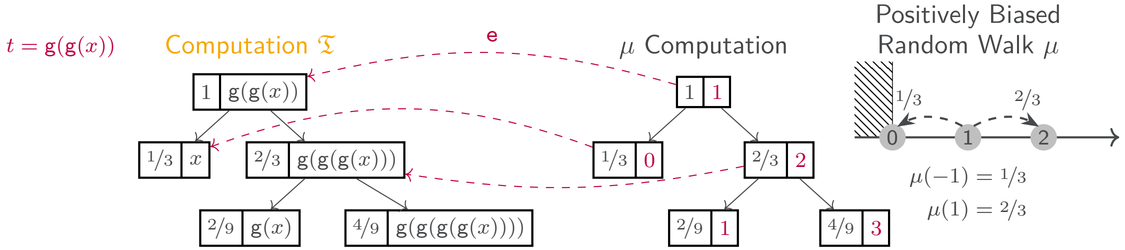
Theorem: Embedding Random Walks [Kassing & Nagel & Schlecht & Giesl (In Review)]

$\mathcal{P}$ a PTRS, t a linear term, and (finite) computation $\mathfrak{T}$ starting with t.
If we have $\sum_{v \in \mathrm{Leaf}(\mathfrak{T})} p_v \cdot \mathrm{maxNO}(t, t_v) \underset{(-)}{\geq} 1$, then $\mathcal{P}$ is not (S)AST.

Why do we need non-overlapping occurrences?

$\mathcal{P}$:

$$\mathsf{g}(\mathsf{g}(x)) \rightarrow \{1/3 : x, 2/3 : \mathsf{g}(\mathsf{g}(\mathsf{g}(x)))\}$$



Construct the Infinite Computation

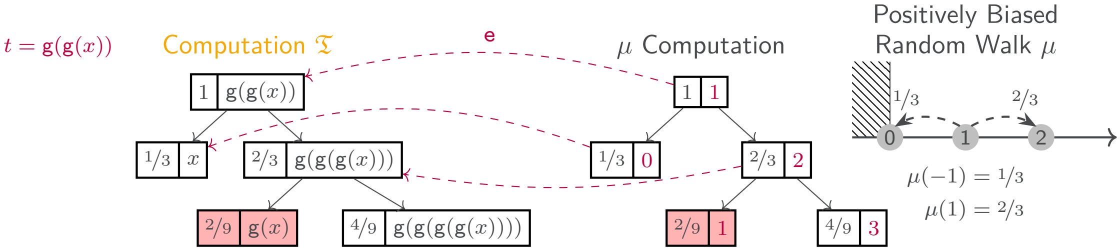
Theorem: Embedding Random Walks [Kassing & Nagel & Schlecht & Giesl (In Review)]

$\mathcal{P}$ a PTRS, t a linear term, and (finite) computation $\mathfrak{T}$ starting with t.
If we have $\sum_{v \in \mathrm{Leaf}(\mathfrak{T})} p_v \cdot \mathrm{maxNO}(t, t_v) \underset{(-)}{\geq} 1$, then $\mathcal{P}$ is not (S)AST.

Why do we need non-overlapping occurrences?

$\mathcal{P}$:

$$\mathsf{g}(\mathsf{g}(x)) \to \{1/3 : x, 2/3 : \mathsf{g}(\mathsf{g}(\mathsf{g}(x)))\}$$



How to Find Such Computations and Random Walks?

1. Find loop in $\mathsf{np}(\mathcal{P})$ [Giesl & Thiemann & Schneider-Kamp'05]

How to Find Such Computations and Random Walks?

1. Find loop in $\mathrm{np}(\mathcal{P})$ [Giesl & Thiemann & Schneider-Kamp'05]

$\mathcal{P}$:

$$\mathsf{g}(x) \to \{{}^{1}/_{3} : \mathsf{a}(x), {}^{1}/_{3} : \mathsf{b}(x), {}^{1}/_{3} : \mathsf{c}(x)\}$$

$$\mathsf{a}(x) \to \{1 : \mathsf{g}(x)\} \qquad \mathsf{b}(x) \to \{1 : \mathsf{g}(\mathsf{g}(x))\} \qquad \mathsf{c}(x) \to \{1 : x\}$$

How to Find Such Computations and Random Walks?

1. Find loop in $\mathsf{np}(\mathcal{P})$ [Giesl & Thiemann & Schneider-Kamp'05]

$\mathcal{P}$:

$$\mathsf{g}(x) \to \{1/3 : \mathsf{a}(x), 1/3 : \mathsf{b}(x), 1/3 : \mathsf{c}(x)\}$$

$$\mathsf{a}(x) \to \{1 : \mathsf{g}(x)\} \qquad \mathsf{b}(x) \to \{1 : \mathsf{g}(\mathsf{g}(x))\} \qquad \mathsf{c}(x) \to \{1 : x\}$$

$\mathsf{np}(\mathcal{P})$:

$$\mathsf{g}(x) \to \mathsf{a}(x) \qquad \mathsf{g}(x) \to \mathsf{b}(x) \qquad \mathsf{g}(x) \to \mathsf{c}(x)$$

$$\mathsf{a}(x) \to \mathsf{g}(x) \qquad \mathsf{b}(x) \to \mathsf{g}(\mathsf{g}(x)) \qquad \mathsf{c}(x) \to x$$

How to Find Such Computations and Random Walks?

1. Find loop in $\mathrm{np}(\mathcal{P})$ [Giesl & Thiemann & Schneider-Kamp'05]

$\mathcal{P}$:

$$\mathsf{g}(x) \to \{1/3 : \mathsf{a}(x), 1/3 : \mathsf{b}(x), 1/3 : \mathsf{c}(x)\}$$

$$\mathsf{a}(x) \to \{1 : \mathsf{g}(x)\} \qquad \mathsf{b}(x) \to \{1 : \mathsf{g}(\mathsf{g}(x))\} \qquad \mathsf{c}(x) \to \{1 : x\}$$

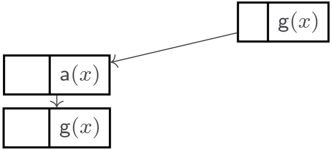
$\mathrm{np}(\mathcal{P})$:

$$\mathsf{g}(x) \to \mathsf{a}(x) \qquad \mathsf{g}(x) \to \mathsf{b}(x) \qquad \mathsf{g}(x) \to \mathsf{c}(x)$$

$$\mathsf{a}(x) \to \mathsf{g}(x) \qquad \mathsf{b}(x) \to \mathsf{g}(\mathsf{g}(x)) \qquad \mathsf{c}(x) \to x$$

$\mathrm{np}(\mathcal{P})$ Loop

$t = \mathsf{g}(x)$



How to Find Such Computations and Random Walks?

1. Find loop in $\mathrm{np}(\mathcal{P})$ [Giesl & Thiemann & Schneider-Kamp'05]
2. Reconstruct computation

$\mathcal{P}$:

$$\mathsf{g}(x) \to \{1/3 : \mathsf{a}(x), 1/3 : \mathsf{b}(x), 1/3 : \mathsf{c}(x)\}$$

$$\mathsf{a}(x) \to \{1 : \mathsf{g}(x)\} \qquad \mathsf{b}(x) \to \{1 : \mathsf{g}(\mathsf{g}(x))\} \qquad \mathsf{c}(x) \to \{1 : x\}$$

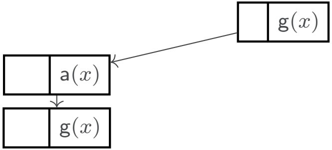
$\mathrm{np}(\mathcal{P})$:

$$\mathsf{g}(x) \to \mathsf{a}(x) \qquad \mathsf{g}(x) \to \mathsf{b}(x) \qquad \mathsf{g}(x) \to \mathsf{c}(x)$$

$$\mathsf{a}(x) \to \mathsf{g}(x) \qquad \mathsf{b}(x) \to \mathsf{g}(\mathsf{g}(x)) \qquad \mathsf{c}(x) \to x$$

$\mathrm{np}(\mathcal{P})$ Loop

$t = \mathsf{g}(x)$



How to Find Such Computations and Random Walks?

1. Find loop in $\mathrm{np}(\mathcal{P})$ [Giesl & Thiemann & Schneider-Kamp'05]
2. Reconstruct computation

$\mathcal{P}$:

$$\mathsf{g}(x) \to \{1/3 : \mathsf{a}(x), 1/3 : \mathsf{b}(x), 1/3 : \mathsf{c}(x)\}$$

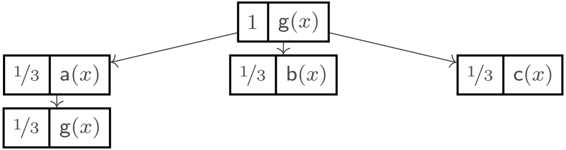
$$\mathsf{a}(x) \to \{1 : \mathsf{g}(x)\} \qquad \mathsf{b}(x) \to \{1 : \mathsf{g}(\mathsf{g}(x))\} \qquad \mathsf{c}(x) \to \{1 : x\}$$

$\mathrm{np}(\mathcal{P})$:

$$\mathsf{g}(x) \to \mathsf{a}(x) \qquad \mathsf{g}(x) \to \mathsf{b}(x) \qquad \mathsf{g}(x) \to \mathsf{c}(x)$$

$$\mathsf{a}(x) \to \mathsf{g}(x) \qquad \mathsf{b}(x) \to \mathsf{g}(\mathsf{g}(x)) \qquad \mathsf{c}(x) \to x$$

$\mathcal{P}$ Computation $\qquad t = \mathsf{g}(x)$



How to Find Such Computations and Random Walks?

1. Find loop in $\mathrm{np}(\mathcal{P})$ [Giesl & Thiemann & Schneider-Kamp'05]
2. Reconstruct computation
3. Rewrite...

$\mathcal{P}$:

$$\mathsf{g}(x) \to \{1/3 : \mathsf{a}(x), 1/3 : \mathsf{b}(x), 1/3 : \mathsf{c}(x)\}$$

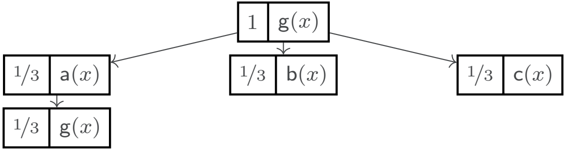
$$\mathsf{a}(x) \to \{1 : \mathsf{g}(x)\} \qquad \mathsf{b}(x) \to \{1 : \mathsf{g}(\mathsf{g}(x))\} \qquad \mathsf{c}(x) \to \{1 : x\}$$

$\mathrm{np}(\mathcal{P})$:

$$\mathsf{g}(x) \to \mathsf{a}(x) \qquad \mathsf{g}(x) \to \mathsf{b}(x) \qquad \mathsf{g}(x) \to \mathsf{c}(x)$$

$$\mathsf{a}(x) \to \mathsf{g}(x) \qquad \mathsf{b}(x) \to \mathsf{g}(\mathsf{g}(x)) \qquad \mathsf{c}(x) \to x$$

$\mathcal{P}$ Computation $\qquad t = \mathsf{g}(x)$



How to Find Such Computations and Random Walks?

1. Find loop in $\mathsf{np}(\mathcal{P})$ [Giesl & Thiemann & Schneider-Kamp'05]
2. Reconstruct computation
3. Rewrite...

$\mathcal{P}$:

$$\mathsf{g}(x) \to \{1/3 : \mathsf{a}(x), 1/3 : \mathsf{b}(x), 1/3 : \mathsf{c}(x)\}$$

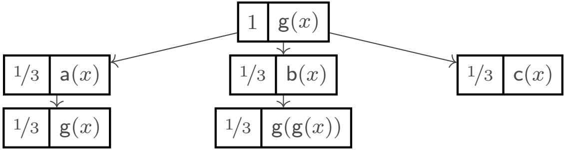
$$\mathsf{a}(x) \to \{1 : \mathsf{g}(x)\} \qquad \mathsf{b}(x) \to \{1 : \mathsf{g}(\mathsf{g}(x))\} \qquad \mathsf{c}(x) \to \{1 : x\}$$

$\mathsf{np}(\mathcal{P})$:

$$\mathsf{g}(x) \to \mathsf{a}(x) \qquad \mathsf{g}(x) \to \mathsf{b}(x) \qquad \mathsf{g}(x) \to \mathsf{c}(x)$$

$$\mathsf{a}(x) \to \mathsf{g}(x) \qquad \mathsf{b}(x) \to \mathsf{g}(\mathsf{g}(x)) \qquad \mathsf{c}(x) \to x$$

$\mathcal{P}$ Computation $\qquad t = \mathsf{g}(x)$



Conclusion

1. What are applications of probabilistic programs? Why are they interesting?
   - Monte Carlo and Las Vegas algorithms
   - Increase the expected worst-case runtime (prevent adversarial attacks)

2. How to analyze probabilistic programs?
   - Transform to a simpler backend language (term rewriting)
   - Prove $\mathtt{AST}$ and $\mathtt{SAST}$ via ranking functions
   - Derive upper expected runtime bounds via ranking functions
   - Disprove $\mathtt{AST}$ and $\mathtt{SAST}$ via embeddings of random walks

   Further techniques:

   - Modularize proofs via dependency pairs
   - Transform the rules to simplify proofs
   - ...

UnRAVeL Symposium 2026

UnRAVeL – UNcertainty and Randomness in Algorithms, VErification and Logic

Date: May 26–29, 2026
Location: RWTH Aachen University, Aachen
Topics: Discrete Structures, Verification, Control, Railway, Combinatorial Optimization, and more!

Keynote Speakers:

- Michal Pilipczuk (University of Warsaw)
- Rupak Majumdar (MPI-SWS)
- Maurice Heemels (TU Eindhoven)
- Francesca Parise (Cornell University)
- Rico Zenklusen (ETH Zürich)
- And further talks by UnRAVeL members and alumni

More information: `https://www.unravel.rwth-aachen.de`
(**Registration (including dinner) is free!**)

Implementation

- Fully implemented in AProVE
- Evaluated on 158 benchmarks

Category	AProVE
AST	70
$\neg$ AST	24
Unknown	68

Category	AProVE
SAST	49
$\neg$ SAST	31
Unknown	78