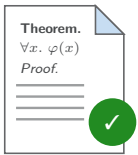
Automatically Analyzing Termination and Expected Runtime Complexity of Probabilistic Term Rewriting

PhD Defense

Jan-Christoph Kassing
RWTH Aachen University
01.10.2026

The Beginning...

1. **Verification of Software**
   Why? How?
2. **What Is Correctness?**
   Termination & Runtime
3. **Why Probabilities?**
   Randomized Algorithms







Why Is Software Correctness Important?



Mariner 1 (1962)

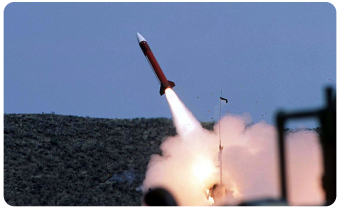
Rocket destroyed after 293 s

Why Is Software Correctness Important?



Mariner 1 (1962)

Rocket destroyed after 293 s



Patriot Missile (1991)

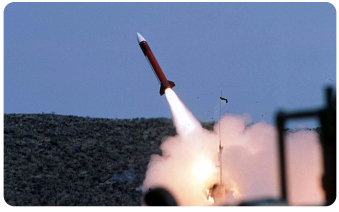
Failed interception

Why Is Software Correctness Important?



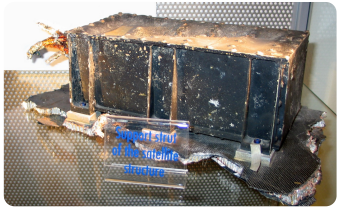
Mariner 1 (1962)

Rocket destroyed after 293 s



Patriot Missile (1991)

Failed interception



Ariane 5 (1996)

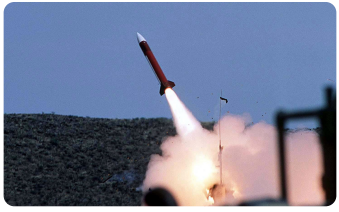
Self-destruct after 37 s

Why Is Software Correctness Important?



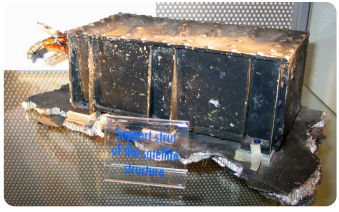
Mariner 1 (1962)

Rocket destroyed after 293 s



Patriot Missile (1991)

Failed interception



Ariane 5 (1996)

Self-destruct after 37 s

Today's World

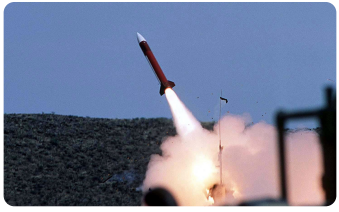
- Substantial fragments of software are generated automatically (e.g., AI-assisted development).

Why Is Software Correctness Important?



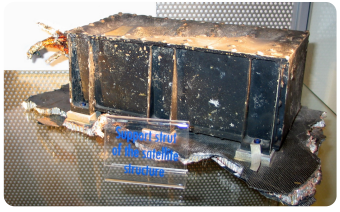
Mariner 1 (1962)

Rocket destroyed after 293 s



Patriot Missile (1991)

Failed interception



Ariane 5 (1996)

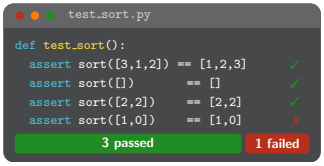
Self-destruct after 37 s

Today's World

- Substantial fragments of software are generated automatically (e.g., AI-assisted development).
- **Verifying correctness (and efficiency) of software is more pressing than ever!**

How to Verify Correctness of Software?

Testing



- Run program on chosen inputs
- ✓ Cheap, easy to automate
- ✗ Only covers the *tested* inputs

How to Verify Correctness of Software?

Testing

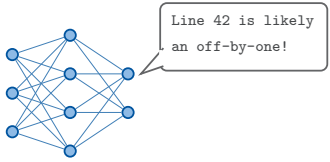
test_sort.py

```
def test_sort():
  assert sort([3,1,2]) == [1,2,3]   ✓
  assert sort([])      == []        ✓
  assert sort([2,2])   == [2,2]     ✓
  assert sort([1,0])   == [1,0]     ✗
```

3 passed | 1 failed

- Run program on chosen inputs
- ✓ Cheap, easy to automate
- ✗ Only covers the *tested* inputs

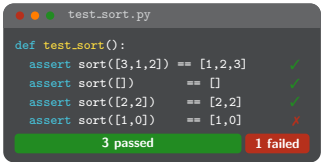
Machine Learning (LLMs)



- Trained on huge code bases
- Flag *statistically likely* errors
- ✗ No formal guarantees at all

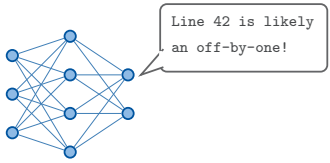
How to Verify Correctness of Software?

Testing



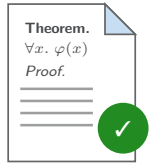
- Run program on chosen inputs
- ✓ Cheap, easy to automate
- ✗ Only covers the *tested* inputs

Machine Learning (LLMs)



- Trained on huge code bases
- Flag *statistically likely* errors
- ✗ No formal guarantees at all

Formal Verification



- Mathematical correctness proof
- ✓ Sound, covers *all* inputs
- ✗ Hard, undecidable in general

How to Verify Correctness of Software?

Testing

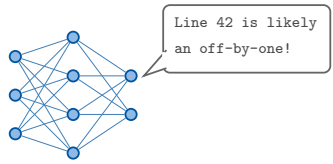
test_sort.py

```
def test_sort():
  assert sort([3,1,2]) == [1,2,3]   ✓
  assert sort([])      == []        ✓
  assert sort([2,2])   == [2,2]     ✓
  assert sort([1,0])   == [1,0]     ✗
```

3 passed | 1 failed

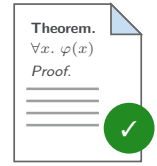
- Run program on chosen inputs
- ✓ Cheap, easy to automate
- ✗ Only covers the *tested* inputs

Machine Learning (LLMs)



- Trained on huge code bases
- Flag *statistically likely* errors
- ✗ No formal guarantees at all

Formal Verification



- Mathematical correctness proof
- ✓ Sound, covers *all* inputs
- ✗ Hard, undecidable in general

How to Formally Verify Correctness of Software?

Human-Made Proofs



Euclid's *Elements*, papyrus ca. 100 AD

- Handwritten proof per program
- Requires an expert every time
- ✗ Slow, error-prone, no scaling

How to Formally Verify Correctness of Software?

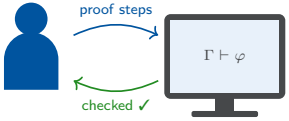
Human-Made Proofs



Euclid's *Elements*, papyrus ca. 100 AD

- Handwritten proof per program
- Requires an expert every time
- ✗ Slow, error-prone, no scaling

Interactive Proofs



Rocq · Isabelle · Lean · PVS · Dafny · Caesar

- Human writes, machine checks
- Proof assistants, SMT verifiers
- ✓ Trustworthy ✗ Expert needed

How to Formally Verify Correctness of Software?

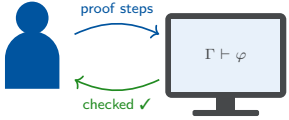
Human-Made Proofs



Euclid's *Elements*, papyrus ca. 100 AD

- Handwritten proof per program
- Requires an expert every time
- ✗ Slow, error-prone, no scaling

Interactive Proofs



Rocq · Isabelle · Lean · PVS · Dafny · Caesar

- Human writes, machine checks
- Proof assistants, SMT verifiers
- ✓ Trustworthy ✗ Expert needed

Fully Automatic Proofs



Push-button, no user interaction

- Machine finds *and* checks proof
- ✓ No user interaction needed
- ✗ Undecidable in general

How to Formally Verify Correctness of Software?

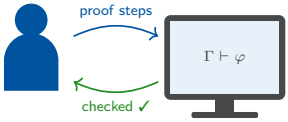
Human-Made Proofs



Euclid's *Elements*, papyrus ca. 100 AD

- Handwritten proof per program
- Requires an expert every time
- ✗ Slow, error-prone, no scaling

Interactive Proofs



Rocq · Isabelle · Lean · PVS · Dafny · Caesar

- Human writes, machine checks
- Proof assistants, SMT verifiers
- ✓ Trustworthy ✗ Expert needed

Fully Automatic Proofs



Push-button, no user interaction

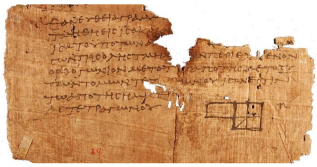
- Machine finds *and* checks proof
- ✓ No user interaction needed
- ✗ Undecidable in general

This Thesis

Fully automatic proofs of correctness for programs with probabilistic choices

How to Formally Verify Correctness of Software?

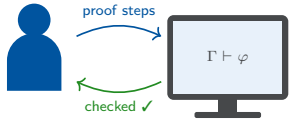
Human-Made Proofs



Euclid's *Elements*, papyrus ca. 100 AD

- Handwritten proof per program
- Requires an expert every time
- ✗ Slow, error-prone, no scaling

Interactive Proofs



Rocq · Isabelle · Lean · PVS · Dafny · Caesar

- Human writes, machine checks
- Proof assistants, SMT verifiers
- ✓ Trustworthy ✗ Expert needed

Fully Automatic Proofs



Push-button, no user interaction

- Machine finds *and* checks proof
- ✓ No user interaction needed
- ✗ Undecidable in general

This Thesis

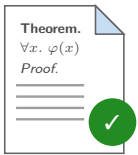
Fully automatic proofs of correctness for programs with probabilistic choices

The Beginning...

1 **Verification of Software**
Why? How?

2 **What Is Correctness?**
Termination & Runtime

3 **Why Probabilities?**
Randomized Algorithms







What Is Correctness?

A program is *correct* if it does what it is supposed to do, e.g.:

Functional correctness | Safety | Security | Termination | Runtime complexity

What Is Correctness?

A program is *correct* if it does what it is supposed to do, e.g.:

Functional correctness | Safety | Security | **Termination** | **Runtime complexity**

What Is Correctness?

A program is *correct* if it does what it is supposed to do, e.g.:

Functional correctness | Safety | Security | **Termination** | **Runtime complexity**

Linear $\mathcal{O}(n)$

$m \leftarrow x_1$
for $i = 2$ **to** n **do**
  if $x_i > m$ **then**
    $m \leftarrow x_i$
  end
end
return m



$n = 10$: **0 steps**

What Is Correctness?

A program is *correct* if it does what it is supposed to do, e.g.:

Functional correctness | Safety | Security | **Termination** | **Runtime complexity**

Linear $\mathcal{O}(n)$

$m \leftarrow x_1$
for $i = 2$ **to** n **do**
  if $x_i > m$ **then**
    $m \leftarrow x_i$
  end
end
return m



$n = 10$: **10 steps**

Exponential $\mathcal{O}(2^n)$

hanoi(n, a, b, c):
if $n > 0$ **then**
  hanoi$(n - 1, a, c, b)$
  move disk n from a to b
  hanoi$(n - 1, c, b, a)$
end



0 steps

What Is Correctness?

A program is *correct* if it does what it is supposed to do, e.g.:

Functional correctness | Safety | Security | **Termination** | **Runtime complexity**

Linear $\mathcal{O}(n)$

$m \leftarrow x_1$
for $i = 2$ **to** n **do**
  if $x_i > m$ **then**
    $m \leftarrow x_i$
  end
end
return m



$n = 10$: **10 steps**

Exponential $\mathcal{O}(2^n)$

hanoi(n, a, b, c):
if $n > 0$ **then**
  hanoi$(n - 1, a, c, b)$
  move disk n from a to b
  hanoi$(n - 1, c, b, a)$
end



1023 steps

Non-Terminating ∞

$x \leftarrow 1$
while $x \neq 0$ **do**
  $x \leftarrow x + 2$
end
return x



running

What Is Correctness?

A program is *correct* if it does what it is supposed to do, e.g.:

Functional correctness | Safety | Security | **Termination** | **Runtime complexity**

Linear $\mathcal{O}(n)$

$m \leftarrow x_1$
for $i = 2$ **to** n **do**
  if $x_i > m$ **then**
    $m \leftarrow x_i$
  end
end
return m



$n = 10$: **10 steps**

Exponential $\mathcal{O}(2^n)$

hanoi(n, a, b, c):
if $n > 0$ **then**
  hanoi$(n - 1, a, c, b)$
  move disk n from a to b
  hanoi$(n - 1, c, b, a)$
end



1023 steps

Non-Terminating ∞

$x \leftarrow 1$
while $x \neq 0$ **do**
  $x \leftarrow x + 2$
end
return x



running

This Thesis

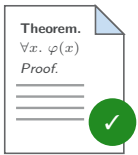
Termination: Is there a result at all? **Runtime Complexity:** How many steps until the result?

The Beginning...

1 **Verification of Software**
Why? How?

2 **What Is Correctness?**
Termination & Runtime

3 **Why Probabilities?**
Randomized Algorithms



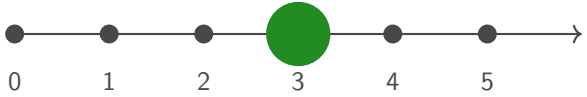




Random Walk

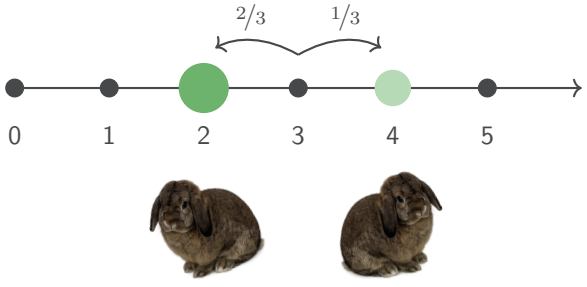


Random Walk

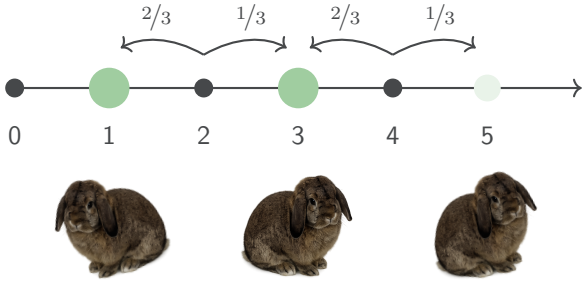




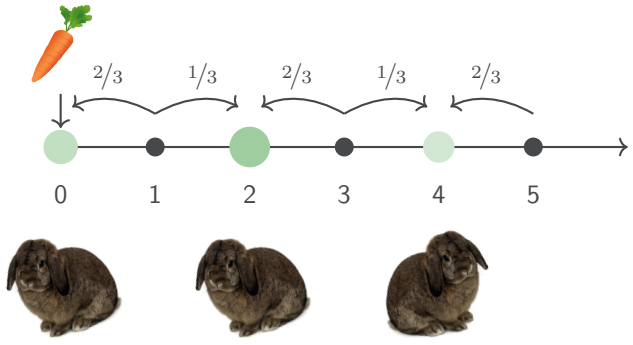
Random Walk



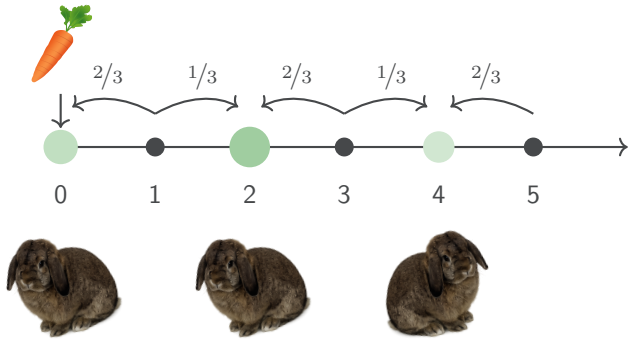
Random Walk



Random Walk

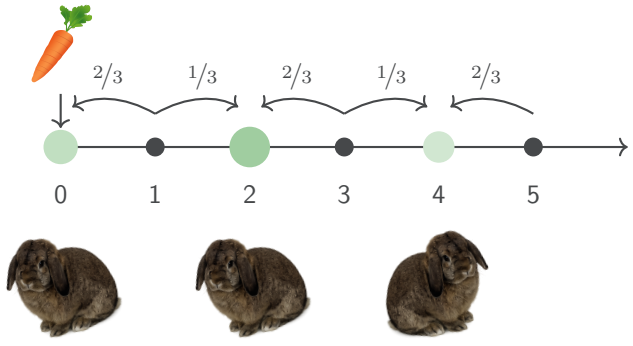


Random Walk



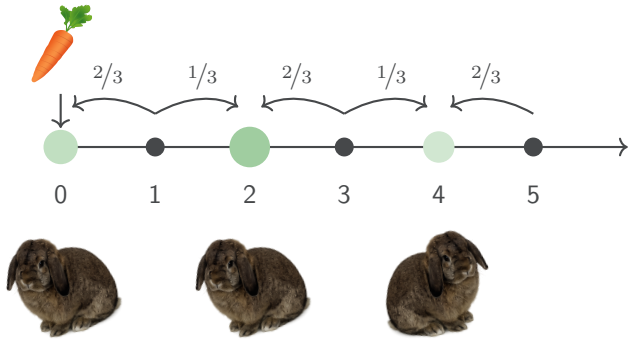
- Does the bunny (program) always reach the carrot (terminate)?

Random Walk



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

Random Walk

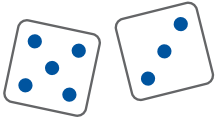


- Does the bunny (program) always reach the carrot (terminate)?
- What is the probability of reaching the carrot (probability of termination)?
- What is the expected number of steps it takes to reach the carrot (expected runtime)?

Two Important Questions

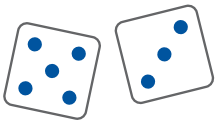
Two Important Questions

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?

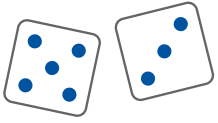


2. How to automatically analyze probabilistic programs?



Two Important Questions

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



2. How to automatically analyze probabilistic programs?



Probabilistic Quicksort

Order Natural Numbers

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

Problem: Sort the elements in ascending order.

Probabilistic Quicksort

Order Natural Numbers

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

Problem: Sort the elements in ascending order.

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: Sort the elements in ascending order.

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: Sort the elements in ascending order.

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

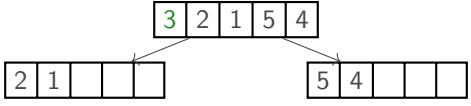
Order Natural Numbers

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

Problem: Sort the elements in ascending order.

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

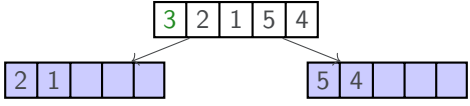
Order Natural Numbers

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

Problem: Sort the elements in ascending order.

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

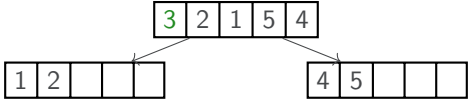
Order Natural Numbers

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

Problem: Sort the elements in ascending order.

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

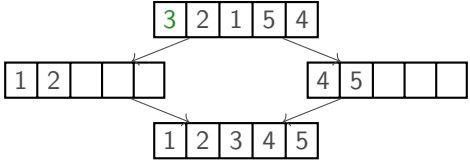
Order Natural Numbers

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

Problem: Sort the elements in ascending order.

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: Sort the elements in ascending order.

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: Sort the elements in ascending order.

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

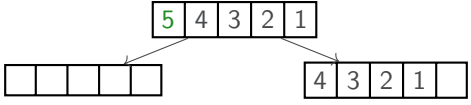
Problem: Sort the elements in ascending order.

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: Sort the elements in ascending order.

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

Problem: Sort the elements in ascending order.

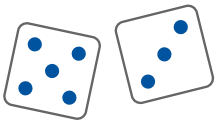
Las Vegas Algorithm:

- 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?

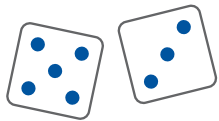


2. How to automatically analyze probabilistic programs?



Two Important Questions

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



2. How to automatically analyze probabilistic programs?



Push-button, no user interaction

Termination and Complexity Analysis for Ordinary Programs

Java

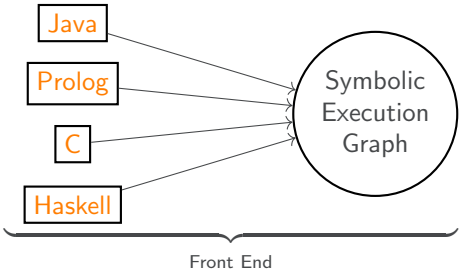
Prolog

C

Haskell

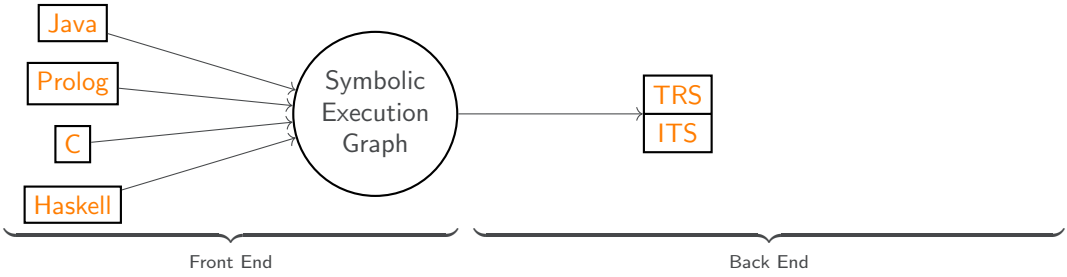


Termination and Complexity Analysis for Ordinary Programs



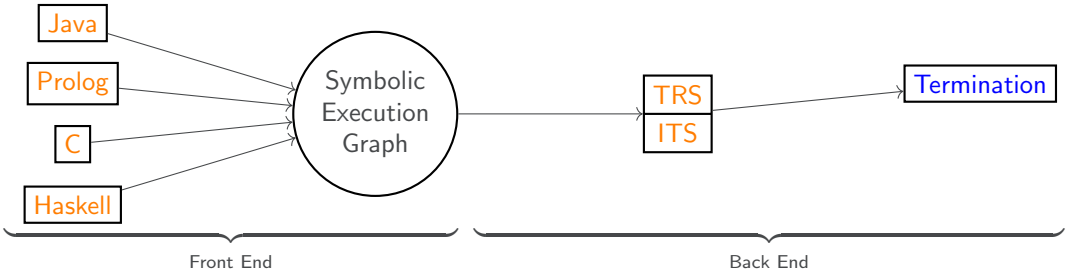
AProVE

Termination and Complexity Analysis for Ordinary Programs



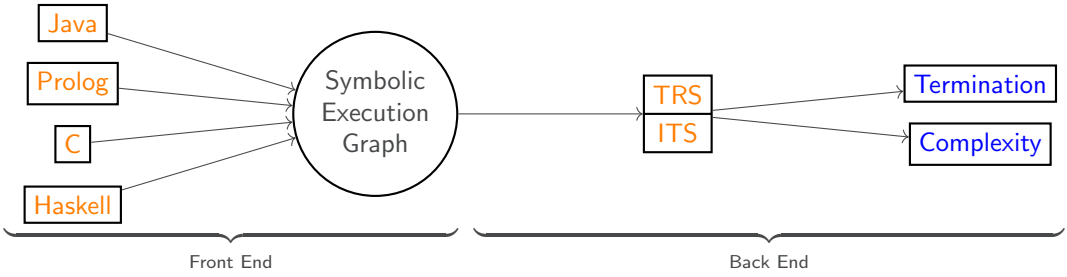
AProVE

Termination and Complexity Analysis for Ordinary Programs



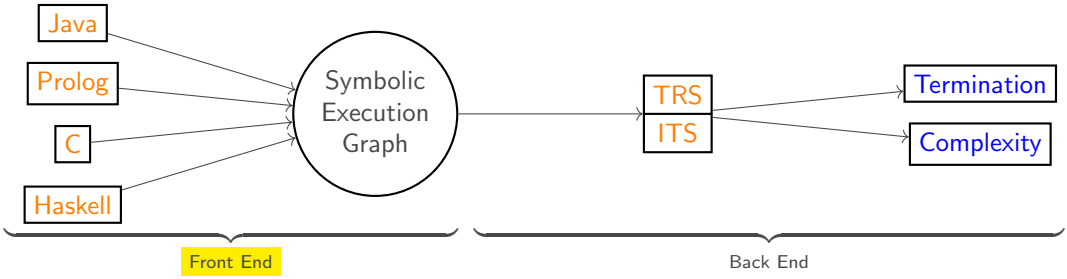
AProVE

Termination and Complexity Analysis for Ordinary Programs



AProVE

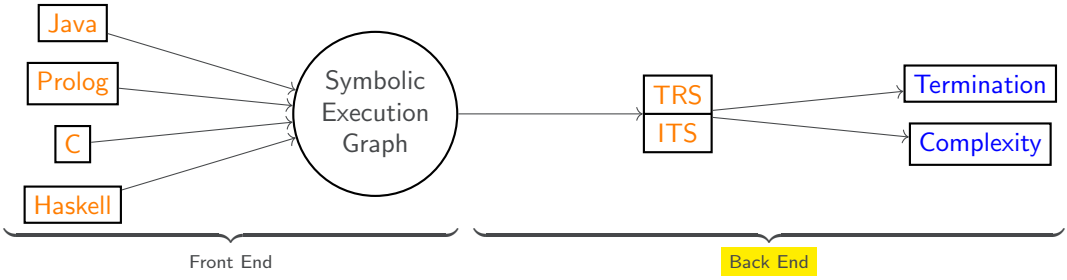
Termination and Complexity Analysis for Ordinary Programs



- Language-specific features when generating the symbolic execution graph

AProVE

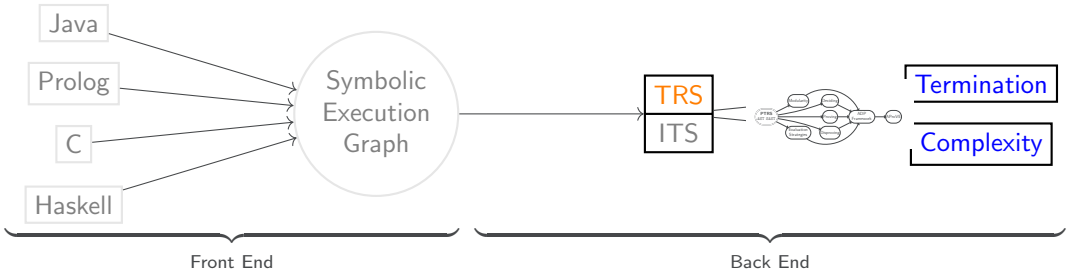
Termination and Complexity Analysis for Ordinary Programs



- Language-specific features when generating the symbolic execution graph
- Back end analyzes term rewrite systems and/or integer transition systems

AProVE

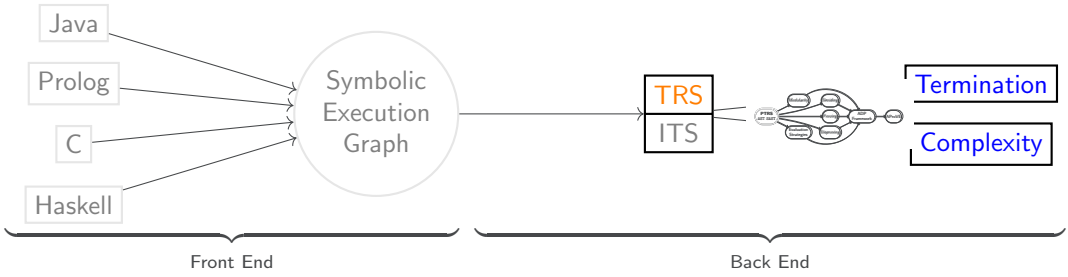
Termination and Complexity Analysis for Ordinary Programs



- Language-specific features when generating the symbolic execution graph
- Back end analyzes term rewrite systems and/or integer transition systems

AProVE

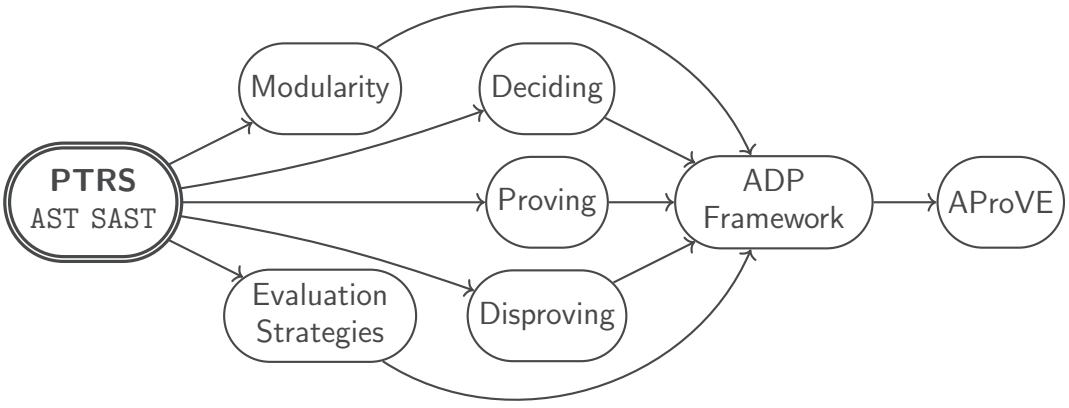
Termination and Complexity Analysis for Ordinary Programs



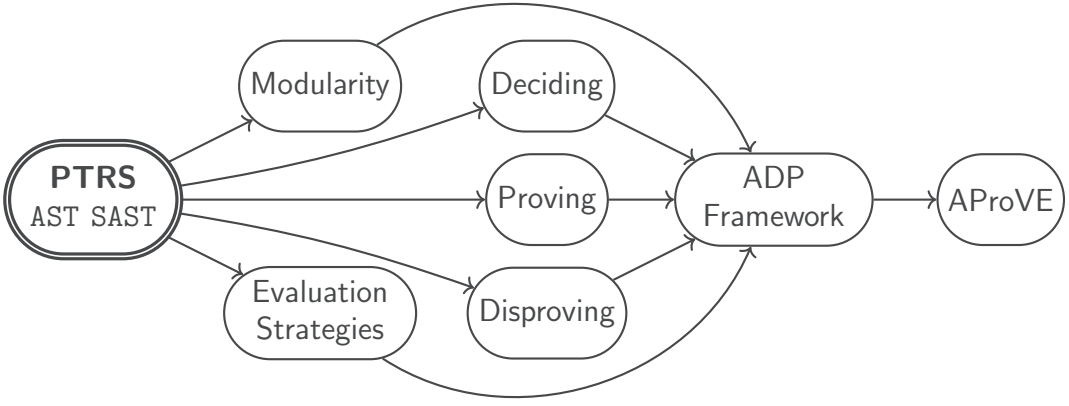
- Language-specific features when generating the symbolic execution graph
- Back end analyzes term rewrite systems and/or integer transition systems

AProVE

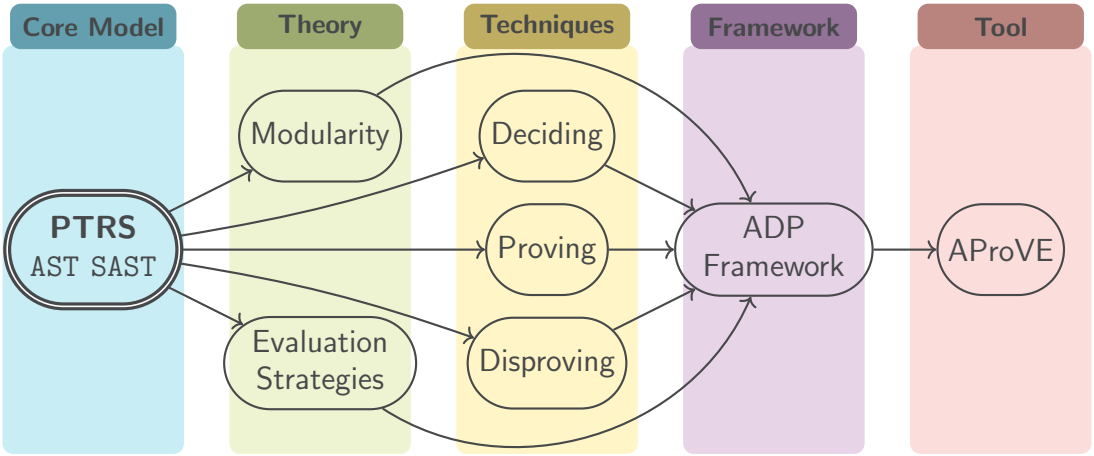
How to Automatically Analyze Probabilistic Programs?



How to Automatically Analyze Probabilistic Programs?

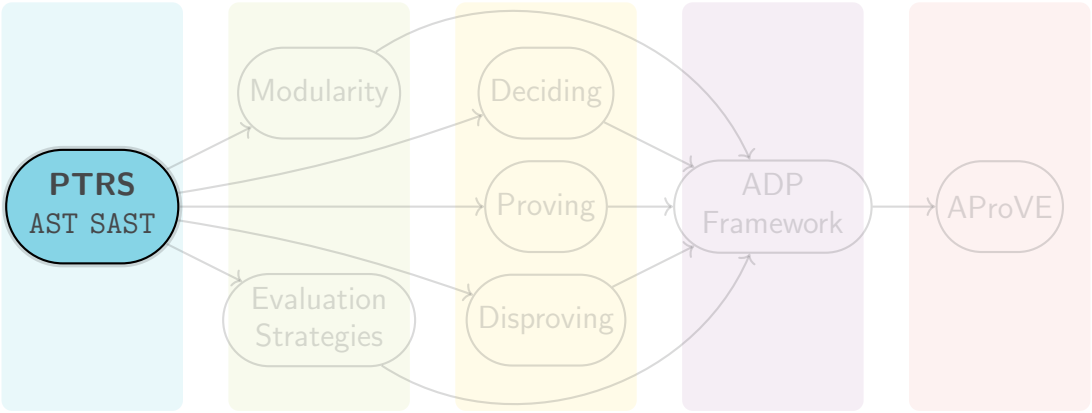


How to Automatically Analyze Probabilistic Programs?



Contributions of This Thesis

Probabilistic Term Rewriting



Probabilistic Term Rewriting: Core Intuition

Core Idea

A program is a set of rewrite rules applied by pattern matching.

$\mathcal{R}_{\mathsf{len}}$ $\quad \mathsf{len}([\,]) \to 0 \qquad \mathsf{len}(x : xs) \to \mathtt{s}(\mathsf{len}(xs))$

Probabilistic Term Rewriting: Core Intuition

Core Idea

A program is a set of rewrite rules applied by pattern matching.

$\mathcal{R}_{\mathsf{len}}$ $\quad \mathsf{len}(\texttt{[]}) \to 0 \qquad \mathsf{len}(x : xs) \to \mathtt{s}(\mathsf{len}(xs))$

(Apply a rule $\ell \to r$ at a matching subterm, the redex.)

$\mathsf{len}(\mathtt{a} : (\mathtt{b} : \texttt{[]}))$

Probabilistic Term Rewriting: Core Intuition

Core Idea

A program is a set of rewrite rules applied by pattern matching.

$\mathcal{R}_{\mathsf{len}}$ $\quad \mathsf{len}(\texttt{[]}) \to 0 \qquad \mathsf{len}(x : xs) \to \mathtt{s}(\mathsf{len}(xs))$

(Apply a rule $\ell \to r$ at a matching subterm, the redex.)

$\mathsf{len}(\mathtt{a} : (\mathtt{b} : \texttt{[]})) \to_{\mathcal{R}_{\mathsf{len}}} \mathtt{s}(\mathsf{len}(\mathtt{b} : \texttt{[]}))$

Probabilistic Term Rewriting: Core Intuition

Core Idea

A program is a set of rewrite rules applied by pattern matching.

$\mathcal{R}_{\mathsf{len}}$ $\quad \mathsf{len}(\texttt{[]}) \to 0 \qquad \mathsf{len}(x : xs) \to \mathtt{s}(\mathsf{len}(xs))$

(Apply a rule $\ell \to r$ at a matching subterm, the redex.)

$\mathsf{len}(\mathtt{a} : (\mathtt{b} : \texttt{[]})) \to_{\mathcal{R}_{\mathsf{len}}} \mathtt{s}(\mathsf{len}(\mathtt{b} : \texttt{[]})) \to_{\mathcal{R}_{\mathsf{len}}} \mathtt{s}(\mathtt{s}(\mathsf{len}(\texttt{[]})))$

Probabilistic Term Rewriting: Core Intuition

Core Idea

A program is a set of rewrite rules applied by pattern matching.

$\mathcal{R}_{\mathsf{len}}$ $\qquad \mathsf{len}([\,]) \to 0 \qquad\qquad \mathsf{len}(x : xs) \to \mathtt{s}(\mathsf{len}(xs))$

(Apply a rule $\ell \to r$ at a matching subterm, the redex.)

$$\mathsf{len}(\mathtt{a} : (\mathtt{b} : [\,])) \to_{\mathcal{R}_{\mathsf{len}}} \mathtt{s}(\mathsf{len}(\mathtt{b} : [\,])) \to_{\mathcal{R}_{\mathsf{len}}} \mathtt{s}(\mathtt{s}(\mathsf{len}([\,]))) \to_{\mathcal{R}_{\mathsf{len}}} \mathtt{s}(\mathtt{s}(0))$$

Probabilistic Term Rewriting: Core Intuition

Core Idea

A program is a set of rewrite rules applied by pattern matching.

$\mathcal{R}_{\mathsf{len}}$ $\quad \mathsf{len}([\,]) \to 0 \qquad \mathsf{len}(x : xs) \to \mathtt{s}(\mathsf{len}(xs))$

(Apply a rule $\ell \to r$ at a matching subterm, the redex.)

$$\mathsf{len}(\mathtt{a} : (\mathtt{b} : [\,])) \to_{\mathcal{R}_{\mathsf{len}}} \mathtt{s}(\mathsf{len}(\mathtt{b} : [\,])) \to_{\mathcal{R}_{\mathsf{len}}} \mathtt{s}(\mathtt{s}(\mathsf{len}([\,]))) \to_{\mathcal{R}_{\mathsf{len}}} \mathtt{s}(\overset{2}{\mathtt{s}}(0))$$

Probabilistic Term Rewriting: Core Intuition

Core Idea

A program is a set of rewrite rules applied by pattern matching.

$\mathcal{R}_{\mathsf{len}}$ $\qquad \mathsf{len}(\texttt{[]}) \to 0 \qquad\qquad \mathsf{len}(x : xs) \to \mathtt{s}(\mathsf{len}(xs))$

(Apply a rule $\ell \to r$ at a matching subterm, the redex.)

$$\mathsf{len}(\mathtt{a} : (\mathtt{b} : \texttt{[]})) \to_{\mathcal{R}_{\mathsf{len}}} \mathtt{s}(\mathsf{len}(\mathtt{b} : \texttt{[]})) \to_{\mathcal{R}_{\mathsf{len}}} \mathtt{s}(\mathtt{s}(\mathsf{len}(\texttt{[]}))) \to^{2}_{\mathcal{R}_{\mathsf{len}}} \mathtt{s}(\mathtt{s}(0))$$

TRSs are especially well suited for programs on data structures (lists, trees, ...).

Probabilistic Term Rewriting: Core Intuition

Core Idea

A program is a set of rewrite rules applied by pattern matching.

$\mathcal{R}_{\mathsf{len}}$ $\qquad \mathsf{len}(\texttt{[]}) \to 0 \qquad\qquad \mathsf{len}(x : xs) \to \mathtt{s}(\mathsf{len}(xs))$

(Apply a rule $\ell \to r$ at a matching subterm, the redex.)

$$\mathsf{len}(\mathtt{a} : (\mathtt{b} : \texttt{[]})) \to_{\mathcal{R}_{\mathsf{len}}} \mathtt{s}(\mathsf{len}(\mathtt{b} : \texttt{[]})) \to_{\mathcal{R}_{\mathsf{len}}} \mathtt{s}(\mathtt{s}(\mathsf{len}(\texttt{[]}))) \to^{2}_{\mathcal{R}_{\mathsf{len}}} \mathtt{s}(\mathtt{s}(0)) \implies \text{terminating}$$

TRSs are especially well suited for programs on data structures (lists, trees, ...).

Termination

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

Probabilistic Term Rewriting: Core Intuition

Core Idea

A program is a set of rewrite rules applied by pattern matching.

$\mathcal{R}_{\mathsf{len}}$ $\quad \mathsf{len}([\,]) \to 0 \qquad \mathsf{len}(x : xs) \to \mathtt{s}(\mathsf{len}(xs))$

(Apply a rule $\ell \to r$ at a matching subterm, the redex.)

$$\mathsf{len}(\mathtt{a} : (\mathtt{b} : [\,])) \to_{\mathcal{R}_{\mathsf{len}}} \mathtt{s}(\mathsf{len}(\mathtt{b} : [\,])) \to_{\mathcal{R}_{\mathsf{len}}} \mathtt{s}(\mathtt{s}(\mathsf{len}([\,]))) \to_{\mathcal{R}_{\mathsf{len}}}^{2} \mathtt{s}(\mathtt{s}(0)) \implies \text{terminating}$$

TRSs are especially well suited for programs on data structures (lists, trees, ...).

Termination

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

- Runtime Complexity: Count the rewrite steps.

Probabilistic Term Rewrite Systems

$\mathcal{R}_{\text{geo}}$: $\quad \mathtt{geo}(\mathtt{x}) \to \left\{ {}^{1/2}\mathtt{x}, {}^{1/2}\mathtt{geo}(\mathtt{0}:\mathtt{x}) \right\}$

Probabilistic Term Rewrite Systems

$\mathcal{R}_{\text{geo}}$: $\quad \mathtt{geo(x)} \to \left\{ {}^{1/2}\mathtt{x},\ {}^{1/2}\mathtt{geo(0:x)} \right\}$

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

Probabilistic Term Rewrite Systems

$\mathcal{R}_{\text{geo}}$: $\quad \mathtt{geo(x)} \to \left\{ {}^{1/2}\mathtt{x}, {}^{1/2}\mathtt{geo(0:x)} \right\}$

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

$\mathfrak{T}$:

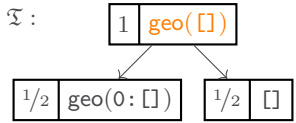
1	geo([])

Probabilistic Term Rewrite Systems

$\mathcal{R}_{\text{geo}}$: $\quad \mathtt{geo(x)} \to \left\{ {}^{1/2}\mathtt{x},\ {}^{1/2}\mathtt{geo(0:x)} \right\}$

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

$\mathfrak{T}$:

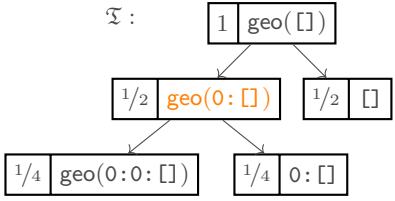


Probabilistic Term Rewrite Systems

$\mathcal{R}_{\text{geo}}$: $\quad \mathtt{geo(x)} \to \left\{ {}^{1/2}\mathtt{x},\ {}^{1/2}\mathtt{geo(0:x)} \right\}$

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

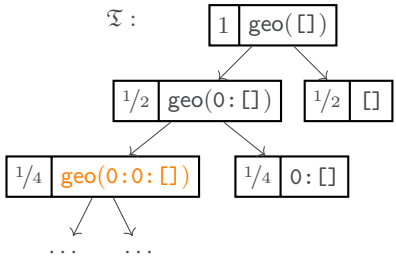
$\mathfrak{T}$:



Probabilistic Term Rewrite Systems

$\mathcal{R}_{\text{geo}}$: $\quad \text{geo}(x) \to \left\{ {}^{1/2}x,\ {}^{1/2}\text{geo}(0:x) \right\}$

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



Probabilistic Term Rewrite Systems

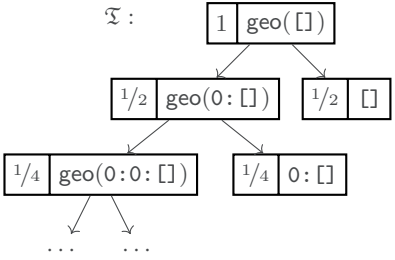
$\mathcal{R}_{\text{geo}}$: $\quad \mathtt{geo(x)} \to \left\{ {}^{1/2}\mathtt{x}, {}^{1/2}\mathtt{geo(0:x)} \right\}$

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

Probability of Termination:

$|\mathfrak{T}|$

$\mathfrak{T}$:



Probabilistic Term Rewrite Systems

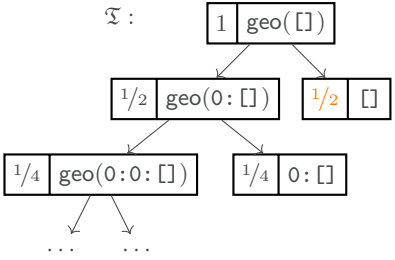
$\mathcal{R}_{\text{geo}}$: $\quad \mathtt{geo(x)} \to \left\{ {}^{1/2}\mathtt{x}, {}^{1/2}\mathtt{geo(0:x)} \right\}$

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

Probability of Termination:

$|\mathfrak{T}| = 1/2 +$

$\mathfrak{T}$:



Probabilistic Term Rewrite Systems

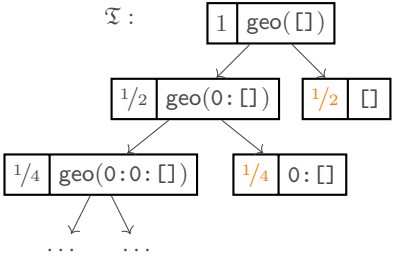
$\mathcal{R}_{\text{geo}}$: $\quad \mathtt{geo(x)} \to \left\{ {}^{1/2}\mathtt{x},\ {}^{1/2}\mathtt{geo(0:x)} \right\}$

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

Probability of Termination:

$$|\mathfrak{T}| = 1/2 + 1/4 +$$

$\mathfrak{T}$:



Probabilistic Term Rewrite Systems

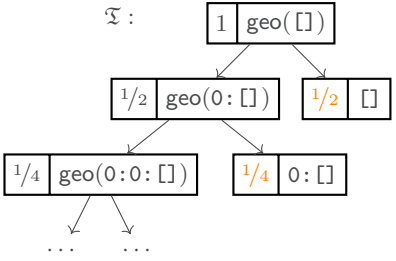
$\mathcal{R}_{\text{geo}}$: $\quad \mathtt{geo(x)} \to \left\{ {}^{1/2}\mathtt{x},\ {}^{1/2}\mathtt{geo(0:x)} \right\}$

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

Probability of Termination:

$$|\mathfrak{T}| = 1/2 + 1/4 + 1/8 + \ldots$$

$\mathfrak{T}$:



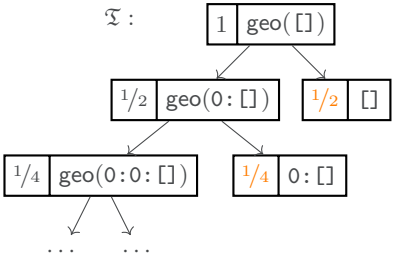
Probabilistic Term Rewrite Systems

$\mathcal{R}_{\text{geo}}$: $\quad \mathtt{geo(x)} \to \left\{ {}^{1/2}\mathtt{x},\ {}^{1/2}\mathtt{geo(0:x)} \right\}$

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

Probability of Termination:

$$|\mathfrak{T}| = 1/2 + 1/4 + 1/8 + \ldots = \sum_{n=0}^{\infty} \frac{1}{2^{n+1}} = 1$$



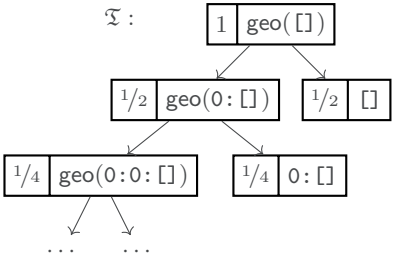
Probabilistic Term Rewrite Systems

$\mathcal{R}_{\text{geo}}$: $\quad \mathtt{geo(x)} \to \left\{ {}^{1/2}\mathtt{x}, {}^{1/2}\mathtt{geo(0:x)} \right\}$

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

Probability of Termination:

$$|\mathfrak{T}| = 1/2 + 1/4 + 1/8 + \ldots = \sum_{n=0}^{\infty} \tfrac{1}{2^{n+1}} = 1$$



Almost-Sure Termination (AST)

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

Probabilistic Term Rewrite Systems

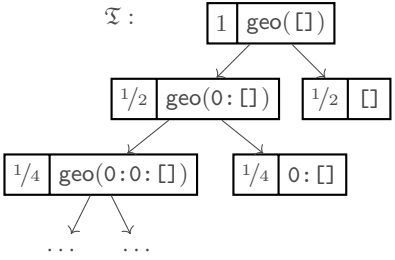
$\mathcal{R}_{\text{geo}}$: $\quad \mathtt{geo(x)} \to \left\{ {}^{1/2}\mathtt{x}, {}^{1/2}\mathtt{geo(0:x)} \right\}$ $\quad$ AST

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

Probability of Termination:

$$|\mathfrak{T}| = 1/2 + 1/4 + 1/8 + \ldots = \sum_{n=0}^{\infty} \tfrac{1}{2^{n+1}} = 1$$

$\mathfrak{T}$:



Almost-Sure Termination (AST)

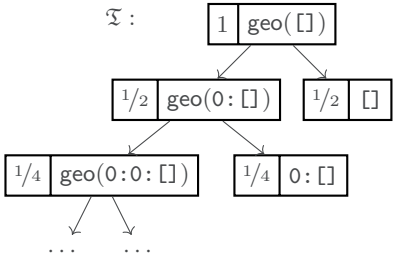
PTRS $\mathcal{R}$ is AST if $|\mathfrak{T}| = 1$ for every $\mathfrak{T}$.

Probabilistic Term Rewrite Systems

$\mathcal{R}_{\text{geo}}$: $\quad \mathtt{geo(x)} \to \left\{ {}^{1/2}\mathtt{x},\ {}^{1/2}\mathtt{geo(0:x)} \right\}$

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

$\mathfrak{T}$:



Probabilistic Term Rewrite Systems

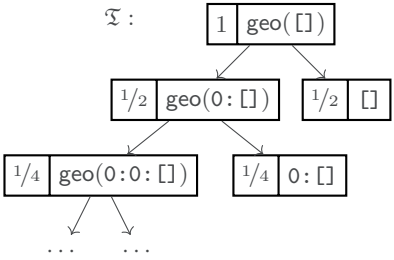
$\mathcal{R}_{\text{geo}}$: $\quad \mathtt{geo(x)} \to \left\{ {}^{1/2}\mathtt{x},\ {}^{1/2}\mathtt{geo(0:x)} \right\}$

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

Expected Derivation Length:

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

$\mathfrak{T}$:



Probabilistic Term Rewrite Systems

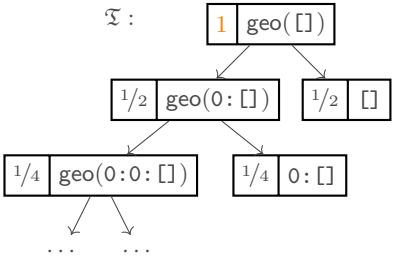
$\mathcal{R}_{\text{geo}}$: $\quad \mathtt{geo(x)} \to \left\{ {}^{1/2}\mathtt{x},\ {}^{1/2}\mathtt{geo(0:x)} \right\}$

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

Expected Derivation Length:

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

$\mathfrak{T}$:



Probabilistic Term Rewrite Systems

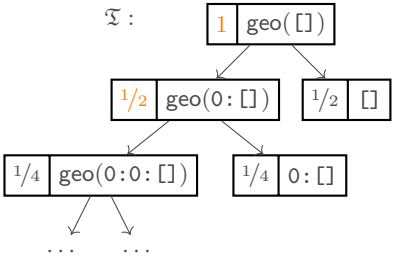
$\mathcal{R}_{\text{geo}}$: $\quad \mathtt{geo(x)} \to \left\{ {}^{1/2}\mathtt{x},\ {}^{1/2}\mathtt{geo(0:x)} \right\}$

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

Expected Derivation Length:

$$\mathrm{edl}(\mathfrak{T}) = 1 + 1/2 +$$

$\mathfrak{T}$:



Probabilistic Term Rewrite Systems

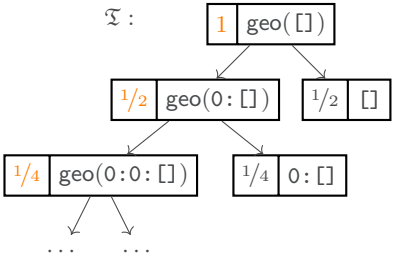
$\mathcal{R}_{\text{geo}}$: $\quad \mathtt{geo}(\mathtt{x}) \to \left\{ {}^{1/2}\mathtt{x},\ {}^{1/2}\mathtt{geo}(\mathtt{0}:\mathtt{x}) \right\}$

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

Expected Derivation Length:

$$\mathrm{edl}(\mathfrak{T}) = 1 + 1/2 + 1/4 + \ldots$$

$\mathfrak{T}$:



Probabilistic Term Rewrite Systems

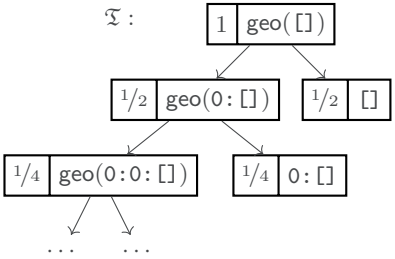
$\mathcal{R}_{\text{geo}}$: $\quad \mathtt{geo(x)} \to \left\{ {}^{1/2}\mathtt{x},\ {}^{1/2}\mathtt{geo(0:x)} \right\}$

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

Expected Derivation Length:

$$\mathrm{edl}(\mathfrak{T}) = 1 + 1/2 + 1/4 + \ldots = \sum_{n=0}^{\infty} (1/2)^n = 2$$

$\mathfrak{T}$:



Probabilistic Term Rewrite Systems

$\mathcal{R}_{\text{geo}}$: $\quad \mathtt{geo(x)} \to \left\{ {}^{1/2}\mathtt{x},\ {}^{1/2}\mathtt{geo(0:x)} \right\}$

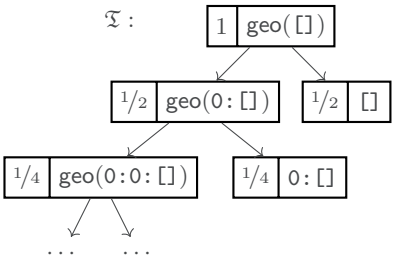
Distribution: $\mu = \{{}^{p_1}t_1, \ldots, {}^{p_k}t_k\}$ with $p_1 + \cdots + p_k = 1$

Expected Derivation Length:

$$\mathrm{edl}(\mathfrak{T}) = 1 + 1/2 + 1/4 + \ldots = \sum_{n=0}^{\infty} (1/2)^n = 2$$

Expected Derivation Height:

$$\mathrm{edh}_{\mathcal{R}_{\text{geo}}}(\mathtt{geo([])}) = \sup\{\mathrm{edl}(\mathfrak{T}') \mid \mathfrak{T}' \text{ starts with } \mathtt{geo([])}\}$$



Probabilistic Term Rewrite Systems

$\mathcal{R}_{\text{geo}}$: $\quad \mathtt{geo(x)} \to \left\{ {}^{1/2}\mathtt{x},\ {}^{1/2}\mathtt{geo(0:x)} \right\}$

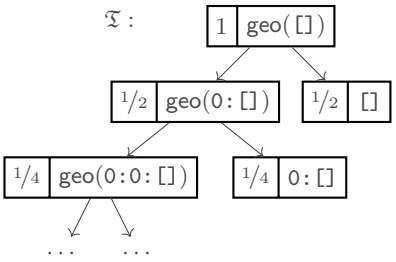
Distribution: $\mu = \{{}^{p_1}t_1, \ldots, {}^{p_k}t_k\}$ with $p_1 + \cdots + p_k = 1$

Expected Derivation Length:

$$\mathrm{edl}(\mathfrak{T}) = 1 + 1/2 + 1/4 + \ldots = \sum_{n=0}^{\infty} (1/2)^n = 2$$

Expected Derivation Height:

$$\mathrm{edh}_{\mathcal{R}_{\text{geo}}}(\mathtt{geo([])}) = \sup\{\mathrm{edl}(\mathfrak{T}') \mid \mathfrak{T}' \text{ starts with } \mathtt{geo([])}\}$$



Probabilistic Term Rewrite Systems

$\mathcal{R}_{\text{geo}}$: $\quad \mathtt{geo(x)} \to \left\{ {}^{1/2}\mathtt{x},\ {}^{1/2}\mathtt{geo(0:x)} \right\}$

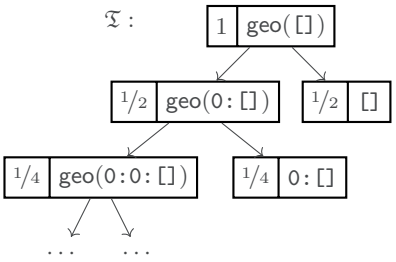
Distribution: $\mu = \{{}^{p_1}t_1, \ldots, {}^{p_k}t_k\}$ with $p_1 + \cdots + p_k = 1$

Expected Derivation Length:

$$\mathrm{edl}(\mathfrak{T}) = 1 + 1/2 + 1/4 + \ldots = \sum_{n=0}^{\infty} (1/2)^n = 2$$

Expected Derivation Height:

$$\mathrm{edh}_{\mathcal{R}_{\text{geo}}}(\mathtt{geo([])}) = \sup\{\mathrm{edl}(\mathfrak{T}') \mid \mathfrak{T}' \text{ starts with } \mathtt{geo([])}\} = 2$$



Probabilistic Term Rewrite Systems

$\mathcal{R}_{\text{geo}}$: $\quad \mathtt{geo}(\mathtt{x}) \to \left\{ {}^{1/2}\mathtt{x},\ {}^{1/2}\mathtt{geo}(\mathtt{0}:\mathtt{x}) \right\}$

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

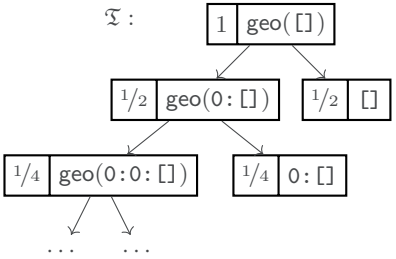
Expected Derivation Length:

$$\mathrm{edl}(\mathfrak{T}) = 1 + 1/2 + 1/4 + \ldots = \sum_{n=0}^{\infty} (1/2)^n = 2$$

Expected Derivation Height:

$$\mathrm{edh}_{\mathcal{R}_{\text{geo}}}(\mathtt{geo}([])) = \sup\{\mathrm{edl}(\mathfrak{T}') \mid \mathfrak{T}' \text{ starts with } \mathtt{geo}([])\} = 2$$

$\mathfrak{T}$:



Strong Almost-Sure Termination (SAST)[ADY20]

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

[ADY20] Avanzini, Dal Lago, and Yamada. On Probabilistic Term Rewriting (SCICO 2020)

Probabilistic Term Rewrite Systems

$\mathcal{R}_{\text{geo}}$: $\quad \mathtt{geo(x)} \to \left\{ {}^{1/2}\mathtt{x},\ {}^{1/2}\mathtt{geo(0:x)} \right\}$ $\quad$ SAST

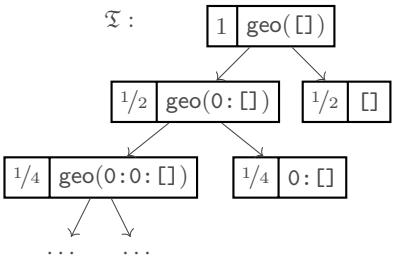
Distribution: $\mu = \{{}^{p_1}t_1, \ldots, {}^{p_k}t_k\}$ with $p_1 + \cdots + p_k = 1$

Expected Derivation Length:

$$\mathrm{edl}(\mathfrak{T}) = 1 + 1/2 + 1/4 + \ldots = \sum_{n=0}^{\infty} (1/2)^n = 2$$

Expected Derivation Height:

$$\mathrm{edh}_{\mathcal{R}_{\text{geo}}}(\mathtt{geo([])}) = \sup\{\mathrm{edl}(\mathfrak{T}') \mid \mathfrak{T}' \text{ starts with } \mathtt{geo([])}\} = 2$$



Strong Almost-Sure Termination (SAST)[ADY20]

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

[ADY20] Avanzini, Dal Lago, and Yamada. On Probabilistic Term Rewriting (SCICO 2020)

Probabilistic Term Rewrite Systems

$\mathcal{R}_{\text{geo}}$: $\quad \mathtt{geo(x)} \to \left\{ {}^{1/2}\mathtt{x},\ {}^{1/2}\mathtt{geo(0:x)} \right\}$ $\quad$ SAST

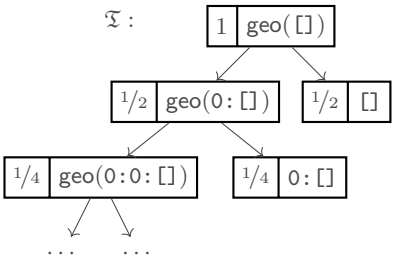
Distribution: $\mu = \{{}^{p_1}t_1, \ldots, {}^{p_k}t_k\}$ with $p_1 + \cdots + p_k = 1$

Expected Derivation Length:

$$\mathrm{edl}(\mathfrak{T}) = 1 + 1/2 + 1/4 + \ldots = \sum_{n=0}^{\infty} (1/2)^n = 2$$

Expected Derivation Height:

$$\mathrm{edh}_{\mathcal{R}_{\text{geo}}}(\mathtt{geo([])}) = \sup\{\mathrm{edl}(\mathfrak{T}') \mid \mathfrak{T}' \text{ starts with } \mathtt{geo([])}\} = 2$$



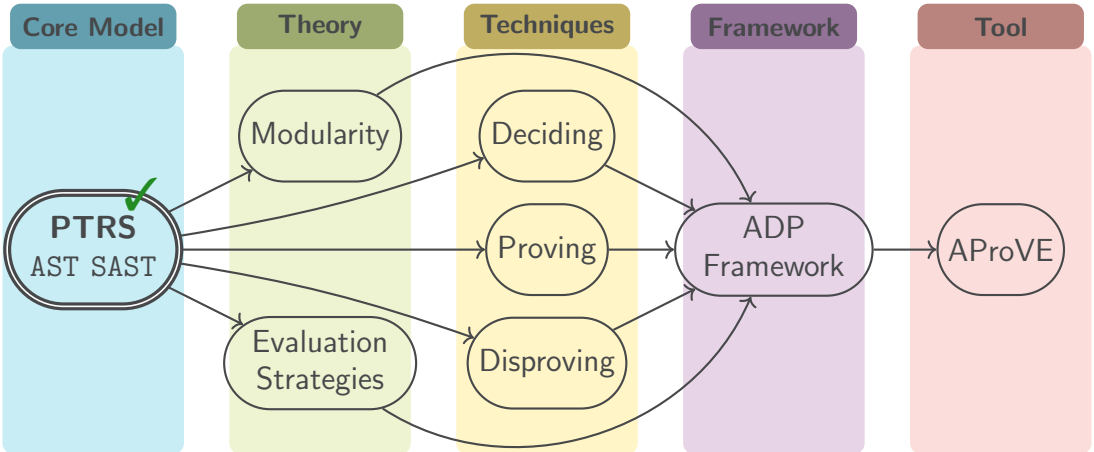
Strong Almost-Sure Termination (SAST)[ADY20]

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

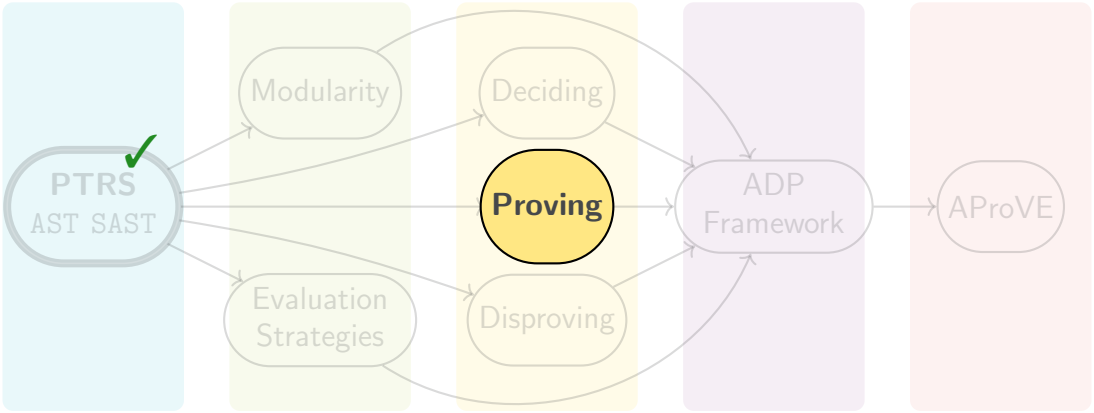
$$\mathtt{SAST} \Rightarrow \mathtt{AST} \qquad \text{and} \qquad \mathtt{AST} \not\Rightarrow \mathtt{SAST}$$

[ADY20] Avanzini, Dal Lago, and Yamada. On Probabilistic Term Rewriting (SCICO 2020)

How to Automatically Analyze Probabilistic Programs?



Ranking Functions



Ranking Functions: Core Intuition

Core Idea

Assign a number $\mathrm{Pol}(t)$ to every term t (a ranking function) that decreases with every rewrite step and is bounded from below.

Ranking Functions: Core Intuition

Core Idea

Assign a number $\mathrm{Pol}(t)$ to every term t (a ranking function) that decreases with every rewrite step and is bounded from below.

Termination: strict decrease



Ranking Functions: Core Intuition

Core Idea

Assign a number $\mathrm{Pol}(t)$ to every term t (a ranking function) that decreases with every rewrite step and is bounded from below.

Termination: strict decrease



Ranking Functions: Core Intuition

Core Idea

Assign a number $\mathrm{Pol}(t)$ to every term t (a ranking function) that decreases with every rewrite step and is bounded from below.

Termination: strict decrease

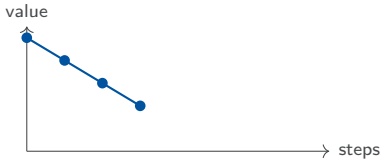


Ranking Functions: Core Intuition

Core Idea

Assign a number $\mathrm{Pol}(t)$ to every term t (a ranking function) that decreases with every rewrite step and is bounded from below.

Termination: strict decrease

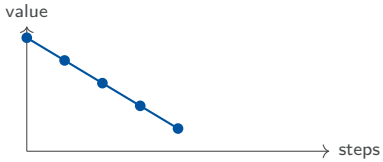


Ranking Functions: Core Intuition

Core Idea

Assign a number $\mathrm{Pol}(t)$ to every term t (a ranking function) that decreases with every rewrite step and is bounded from below.

Termination: strict decrease

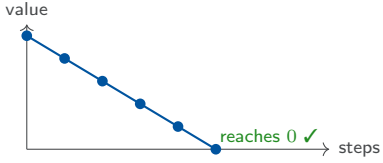


Ranking Functions: Core Intuition

Core Idea

Assign a number $\mathrm{Pol}(t)$ to every term t (a ranking function) that decreases with every rewrite step and is bounded from below.

Termination: strict decrease



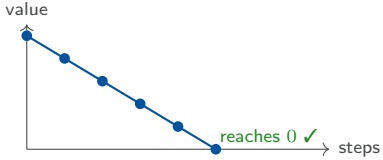
$\mathrm{Pol}(\ell) > \mathrm{Pol}(r)$
for every rule $\ell \to r$

Ranking Functions: Core Intuition

Core Idea

Assign a number $\mathrm{Pol}(t)$ to every term t (a ranking function) that decreases with every rewrite step and is bounded from below.

Termination: strict decrease



$\mathrm{Pol}(\ell) > \mathrm{Pol}(r)$
for every rule $\ell \to r$

`AST` / `SAST`: decrease in expectation

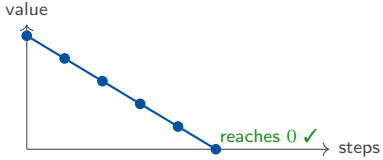


Ranking Functions: Core Intuition

Core Idea

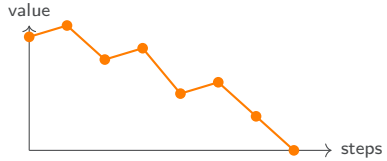
Assign a number $\mathrm{Pol}(t)$ to every term t (a ranking function) that decreases with every rewrite step and is bounded from below.

Termination: strict decrease



$\mathrm{Pol}(\ell) > \mathrm{Pol}(r)$
for every rule $\ell \to r$

`AST` / `SAST`: decrease in expectation

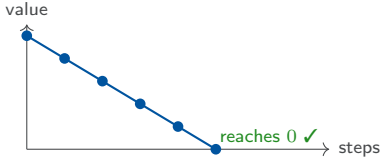


Ranking Functions: Core Intuition

Core Idea

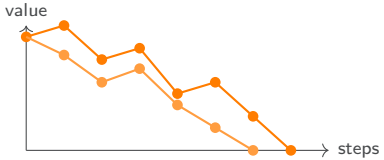
Assign a number $\mathrm{Pol}(t)$ to every term t (a ranking function) that decreases with every rewrite step and is bounded from below.

Termination: strict decrease



$\mathrm{Pol}(\ell) > \mathrm{Pol}(r)$
for every rule $\ell \to r$

`AST` / `SAST`: decrease in expectation

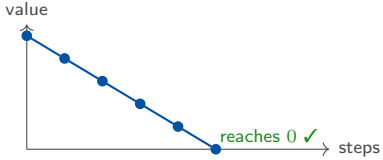


Ranking Functions: Core Intuition

Core Idea

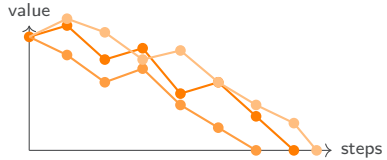
Assign a number $\mathrm{Pol}(t)$ to every term t (a ranking function) that decreases with every rewrite step and is bounded from below.

Termination: strict decrease



$\mathrm{Pol}(\ell) > \mathrm{Pol}(r)$
for every rule $\ell \to r$

AST / SAST: decrease in expectation

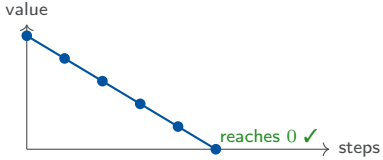


Ranking Functions: Core Intuition

Core Idea

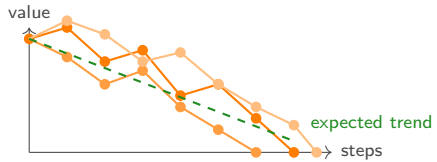
Assign a number $\mathrm{Pol}(t)$ to every term t (a ranking function) that decreases with every rewrite step and is bounded from below.

Termination: strict decrease



$\mathrm{Pol}(\ell) > \mathrm{Pol}(r)$
for every rule $\ell \to r$

`AST` / `SAST`: decrease in expectation



$\mathrm{Pol}(\ell) \geq \sum_{j=1}^{k} p_j \cdot \mathrm{Pol}(r_j)$
for every rule $\ell \to \{{}^{p_1}r_1, \ldots, {}^{p_k}r_k\}$

Proving Almost-Sure Termination

Theorem: Proving AST[CADE '23][MMKK17]

Let Pol be a natural & monotonic & multilinear polynomial interpretation.
$\mathcal{R}$ is AST if for all $\ell \to \{{}^{p_1}r_1, \ldots, {}^{p_k}r_k\} \in \mathcal{R}$:

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

[MMKK17] McIver, Morgan, Kaminski, and Katoen. A New Proof Rule for Almost-Sure Termination (POPL 2017)
[CADE '23] Kassing and Giesl. Proving Almost-Sure Innermost Termination [...] (CADE 2023)

Proving Almost-Sure Termination

Theorem: Proving AST[CADE '23][MMKK17]

Let Pol be a natural & monotonic & multilinear polynomial interpretation.
$\mathcal{R}$ is AST if for all $\ell \to \{{}^{p_1}r_1, \ldots, {}^{p_k}r_k\} \in \mathcal{R}$:

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

$\mathcal{R}_{\mathsf{geo}}$: $\quad \mathsf{geo}(\mathtt{x}) \to \left\{{}^{1/2}\mathtt{x}, {}^{1/2}\mathsf{geo}(\mathtt{0}:\mathtt{x})\right\}$

[MMKK17] McIver, Morgan, Kaminski, and Katoen. A New Proof Rule for Almost-Sure Termination (POPL 2017)
[CADE '23] Kassing and Giesl. Proving Almost-Sure Innermost Termination [...] (CADE 2023)

Proving Almost-Sure Termination

Theorem: Proving AST[CADE '23][MMKK17]

Let Pol be a natural & monotonic & multilinear polynomial interpretation.
$\mathcal{R}$ is AST if for all $\ell \to \{{}^{p_1}r_1, \ldots, {}^{p_k}r_k\} \in \mathcal{R}$:

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

$\mathcal{R}_{\mathsf{geo}}$: $\quad \mathsf{geo}(\mathtt{x}) \to \{{}^{1/2}\mathtt{x}, {}^{1/2}\mathsf{geo}(\mathtt{0}:\mathtt{x})\}$

$$\mathrm{Pol}_{\mathsf{geo}}(x) = x + 2 \qquad \mathrm{Pol}_{\mathtt{0}} = 0 \qquad \mathrm{Pol}_{:}(x, y) = x + y + 1 \qquad \mathrm{Pol}_{[]} = 0$$

[MMKK17] McIver, Morgan, Kaminski, and Katoen. A New Proof Rule for Almost-Sure Termination (POPL 2017)
[CADE '23] Kassing and Giesl. Proving Almost-Sure Innermost Termination [...] (CADE 2023)

Proving Almost-Sure Termination

Theorem: Proving AST[CADE '23][MMKK17]

Let Pol be a natural & monotonic & multilinear polynomial interpretation.
$\mathcal{R}$ is AST if for all $\ell \to \{{}^{p_1}r_1, \ldots, {}^{p_k}r_k\} \in \mathcal{R}$:

- $\mathrm{Pol}(\ell) > \mathrm{Pol}(r_j)$ for some $1 \le j \le k$
- $\mathrm{Pol}(\ell) \ge p_1 \cdot \mathrm{Pol}(r_1) + \ldots + p_k \cdot \mathrm{Pol}(r_k)$

$\mathcal{R}_{\mathsf{geo}}$: $\mathsf{geo}(\mathtt{x}) \to \{{}^{1/2}\mathtt{x}, {}^{1/2}\mathsf{geo}(\mathtt{0}:\mathtt{x})\}$

$$\mathrm{Pol}_{\mathsf{geo}}(x) = x + 2 \qquad \mathrm{Pol}_{\mathtt{0}} = 0 \qquad \mathrm{Pol}_{:}(x, y) = x + y + 1 \qquad \mathrm{Pol}_{[]} = 0$$

$$\mathrm{Pol}(\mathsf{geo}(\mathtt{x})) \;>\; \mathrm{Pol}(\mathtt{x})$$

[MMKK17] McIver, Morgan, Kaminski, and Katoen. A New Proof Rule for Almost-Sure Termination (POPL 2017)
[CADE '23] Kassing and Giesl. Proving Almost-Sure Innermost Termination [...] (CADE 2023)

Proving Almost-Sure Termination

Theorem: Proving AST[CADE '23][MMKK17]

Let Pol be a natural & monotonic & multilinear polynomial interpretation.
$\mathcal{R}$ is AST if for all $\ell \to \{{}^{p_1}r_1, \ldots, {}^{p_k}r_k\} \in \mathcal{R}$:

- $\mathrm{Pol}(\ell) > \mathrm{Pol}(r_j)$ for some $1 \le j \le k$
- $\mathrm{Pol}(\ell) \ge p_1 \cdot \mathrm{Pol}(r_1) + \ldots + p_k \cdot \mathrm{Pol}(r_k)$

$\mathcal{R}_{\mathsf{geo}}$: $\mathsf{geo}(\mathtt{x}) \to \{{}^{1/2}\mathtt{x}, {}^{1/2}\mathsf{geo}(\mathtt{0}:\mathtt{x})\}$

$$\mathrm{Pol}_{\mathsf{geo}}(x) = x + 2 \qquad \mathrm{Pol}_{\mathtt{0}} = 0 \qquad \mathrm{Pol}_{:}(x, y) = x + y + 1 \qquad \mathrm{Pol}_{[]} = 0$$

$$\begin{array}{rcl} \mathrm{Pol}(\mathsf{geo}(\mathtt{x})) & > & \mathrm{Pol}(\mathtt{x}) \\ x + 2 & > & x \quad \checkmark \end{array}$$

[MMKK17] McIver, Morgan, Kaminski, and Katoen. A New Proof Rule for Almost-Sure Termination (POPL 2017)
[CADE '23] Kassing and Giesl. Proving Almost-Sure Innermost Termination [...] (CADE 2023)

Proving Almost-Sure Termination

Theorem: Proving AST[CADE '23][MMKK17]

Let Pol be a natural & monotonic & multilinear polynomial interpretation.
$\mathcal{R}$ is AST if for all $\ell \to \{{}^{p_1}r_1, \ldots, {}^{p_k}r_k\} \in \mathcal{R}$:

- $\mathrm{Pol}(\ell) > \mathrm{Pol}(r_j)$ for some $1 \le j \le k$
- $\mathrm{Pol}(\ell) \ge p_1 \cdot \mathrm{Pol}(r_1) + \ldots + p_k \cdot \mathrm{Pol}(r_k)$

$\mathcal{R}_{\mathsf{geo}}$: $\mathtt{geo(x)} \to \{{}^{1/2}\mathtt{x}, {}^{1/2}\mathtt{geo(0:x)}\}$

$$\mathrm{Pol}_{\mathsf{geo}}(x) = x + 2 \qquad \mathrm{Pol}_{\mathtt{0}} = 0 \qquad \mathrm{Pol}_{\mathtt{:}}(x, y) = x + y + 1 \qquad \mathrm{Pol}_{\mathtt{[]}} = 0$$

$$\mathrm{Pol}(\mathtt{geo(x)}) > \mathrm{Pol}(\mathtt{x}) \qquad\qquad \mathrm{Pol}(\mathtt{geo(x)}) \ge 1/2 \cdot \mathrm{Pol}(\mathtt{x}) + 1/2 \cdot \mathrm{Pol}(\mathtt{geo(0:x)})$$

$$x + 2 > x \quad \checkmark$$

[MMKK17] McIver, Morgan, Kaminski, and Katoen. A New Proof Rule for Almost-Sure Termination (POPL 2017)
[CADE '23] Kassing and Giesl. Proving Almost-Sure Innermost Termination [...] (CADE 2023)

Proving Almost-Sure Termination

Theorem: Proving AST[CADE '23][MMKK17]

Let Pol be a natural & monotonic & multilinear polynomial interpretation.
$\mathcal{R}$ is AST if for all $\ell \to \{{}^{p_1}r_1, \ldots, {}^{p_k}r_k\} \in \mathcal{R}$:

- $\mathrm{Pol}(\ell) > \mathrm{Pol}(r_j)$ for some $1 \le j \le k$
- $\mathrm{Pol}(\ell) \ge p_1 \cdot \mathrm{Pol}(r_1) + \ldots + p_k \cdot \mathrm{Pol}(r_k)$

$\mathcal{R}_{\mathsf{geo}}$: $\mathtt{geo(x)} \to \{{}^{1/2}\mathtt{x}, {}^{1/2}\mathtt{geo(0:x)}\}$

$$\mathrm{Pol}_{\mathsf{geo}}(x) = x + 2 \qquad \mathrm{Pol}_{\mathtt{0}} = 0 \qquad \mathrm{Pol}_{\mathtt{:}}(x, y) = x + y + 1 \qquad \mathrm{Pol}_{[]} = 0$$

$$\mathrm{Pol}(\mathtt{geo(x)}) > \mathrm{Pol}(\mathtt{x}) \qquad x + 2 > x \quad ✓$$

$$\mathrm{Pol}(\mathtt{geo(x)}) \ge 1/2 \cdot \mathrm{Pol}(\mathtt{x}) + 1/2 \cdot \mathrm{Pol}(\mathtt{geo(0:x)}) \qquad x + 2 \ge x + 3/2 \quad ✓$$

[MMKK17] McIver, Morgan, Kaminski, and Katoen. A New Proof Rule for Almost-Sure Termination (POPL 2017)
[CADE '23] Kassing and Giesl. Proving Almost-Sure Innermost Termination [...] (CADE 2023)

Proving Almost-Sure Termination

Theorem: Proving AST[CADE '23][MMKK17]

Let Pol be a natural & monotonic & multilinear polynomial interpretation.
$\mathcal{R}$ is AST if for all $\ell \to \{{}^{p_1}r_1, \ldots, {}^{p_k}r_k\} \in \mathcal{R}$:

- $\mathrm{Pol}(\ell) > \mathrm{Pol}(r_j)$ for some $1 \le j \le k$
- $\mathrm{Pol}(\ell) \ge p_1 \cdot \mathrm{Pol}(r_1) + \ldots + p_k \cdot \mathrm{Pol}(r_k)$

$\mathcal{R}_{\mathsf{geo}}$: $\quad \mathsf{geo}(\mathtt{x}) \to \{{}^{1/2}\mathtt{x}, {}^{1/2}\mathsf{geo}(\mathtt{0}:\mathtt{x})\}$

$$\mathrm{Pol}_{\mathsf{geo}}(x) = x + 2 \qquad \mathrm{Pol}_{\mathtt{0}} = 0 \qquad \mathrm{Pol}_{:}(x, y) = x + y + 1 \qquad \mathrm{Pol}_{[]} = 0$$

$$\mathrm{Pol}(\mathsf{geo}(\mathtt{x})) > \mathrm{Pol}(\mathtt{x}) \qquad x + 2 > x \quad ✓$$

$$\mathrm{Pol}(\mathsf{geo}(\mathtt{x})) \ge 1/2 \cdot \mathrm{Pol}(\mathtt{x}) + 1/2 \cdot \mathrm{Pol}(\mathsf{geo}(\mathtt{0}:\mathtt{x})) \qquad x + 2 \ge x + 3/2 \quad ✓$$

$\Longrightarrow \mathcal{R}_{\mathsf{geo}}$ is AST

[MMKK17] McIver, Morgan, Kaminski, and Katoen. A New Proof Rule for Almost-Sure Termination (POPL 2017)
[CADE '23] Kassing and Giesl. Proving Almost-Sure Innermost Termination [...] (CADE 2023)

Proving Strong Almost-Sure Termination

Theorem: Proving SAST[ADY20]

Let Pol be a natural & monotonic & multilinear polynomial interpretation.
$\mathcal{R}$ is SAST if for all $\ell \to \{{}^{p_1}r_1, \ldots, {}^{p_k}r_k\} \in \mathcal{R}$: $\quad \mathrm{Pol}(\ell) > p_1 \cdot \mathrm{Pol}(r_1) + \ldots + p_k \cdot \mathrm{Pol}(r_k)$

[ADY20] Avanzini, Dal Lago, and Yamada. On Probabilistic Term Rewriting (SCICO 2020)

Proving Strong Almost-Sure Termination

Theorem: Proving SAST[ADY20]

Let Pol be a natural & monotonic & multilinear polynomial interpretation.
$\mathcal{R}$ is SAST if for all $\ell \to \{{}^{p_1}r_1, \ldots, {}^{p_k}r_k\} \in \mathcal{R}$: $\quad \mathrm{Pol}(\ell) > p_1 \cdot \mathrm{Pol}(r_1) + \ldots + p_k \cdot \mathrm{Pol}(r_k)$

$\mathcal{R}_{\mathsf{geo}}$: $\qquad \mathsf{geo}(\mathsf{x}) \to \left\{{}^{1/2}\mathsf{x}, {}^{1/2}\mathsf{geo}(\mathsf{0}:\mathsf{x})\right\}$

[ADY20] Avanzini, Dal Lago, and Yamada. On Probabilistic Term Rewriting (SCICO 2020)

Proving Strong Almost-Sure Termination

Theorem: Proving SAST[ADY20]

Let Pol be a natural & monotonic & multilinear polynomial interpretation.
$\mathcal{R}$ is SAST if for all $\ell \to \{{}^{p_1}r_1, \ldots, {}^{p_k}r_k\} \in \mathcal{R}$: $\quad \mathrm{Pol}(\ell) > p_1 \cdot \mathrm{Pol}(r_1) + \ldots + p_k \cdot \mathrm{Pol}(r_k)$

$\mathcal{R}_{\mathsf{geo}}$: $\qquad \mathsf{geo}(\mathtt{x}) \to \left\{{}^{1/2}\mathtt{x}, {}^{1/2}\mathsf{geo}(\mathtt{0}:\mathtt{x})\right\}$

$$\mathrm{Pol}_{\mathsf{geo}}(x) = x + 2 \qquad \mathrm{Pol}_{\mathtt{0}} = 0 \qquad \mathrm{Pol}_{\mathtt{:}}(x, y) = x + y + 1 \qquad \mathrm{Pol}_{[]} = 0$$

[ADY20] Avanzini, Dal Lago, and Yamada. On Probabilistic Term Rewriting (SCICO 2020)

Proving Strong Almost-Sure Termination

Theorem: Proving $\mathtt{SAST}$[ADY20]

Let Pol be a natural & monotonic & multilinear polynomial interpretation.
$\mathcal{R}$ is $\mathtt{SAST}$ if for all $\ell \to \{{}^{p_1}r_1, \ldots, {}^{p_k}r_k\} \in \mathcal{R}$: $\quad \mathrm{Pol}(\ell) > p_1 \cdot \mathrm{Pol}(r_1) + \ldots + p_k \cdot \mathrm{Pol}(r_k)$

$\mathcal{R}_{\mathtt{geo}}$: $\qquad \mathtt{geo(x)} \to \left\{{}^{1/2}\mathtt{x}, {}^{1/2}\mathtt{geo(0:x)}\right\}$

$$\mathrm{Pol}_{\mathtt{geo}}(x) = x + 2 \qquad \mathrm{Pol}_{\mathtt{0}} = 0 \qquad \mathrm{Pol}_{\mathtt{:}}(x, y) = x + y + 1 \qquad \mathrm{Pol}_{\mathtt{[]}} = 0$$

$$\begin{aligned} \mathrm{Pol}(\mathtt{geo(x)}) &> {}^{1}/_{2} \cdot \mathrm{Pol}(\mathtt{x}) + {}^{1}/_{2} \cdot \mathrm{Pol}(\mathtt{geo(0:x)}) \\ x + 2 &> x + {}^{3}/_{2} \quad \checkmark \end{aligned}$$

[ADY20] Avanzini, Dal Lago, and Yamada. On Probabilistic Term Rewriting (SCICO 2020)

Proving Strong Almost-Sure Termination

Theorem: Proving SAST[ADY20]

Let Pol be a natural & monotonic & multilinear polynomial interpretation.
$\mathcal{R}$ is SAST if for all $\ell \to \{{}^{p_1}r_1, \ldots, {}^{p_k}r_k\} \in \mathcal{R}$: $\quad \mathrm{Pol}(\ell) > p_1 \cdot \mathrm{Pol}(r_1) + \ldots + p_k \cdot \mathrm{Pol}(r_k)$

$\mathcal{R}_{\mathsf{geo}}$: $\qquad \mathsf{geo}(\mathtt{x}) \to \left\{{}^{1/2}\mathtt{x}, {}^{1/2}\mathsf{geo}(\mathtt{0}:\mathtt{x})\right\}$

$$\mathrm{Pol}_{\mathsf{geo}}(x) = x + 2 \qquad \mathrm{Pol}_{\mathtt{0}} = 0 \qquad \mathrm{Pol}_{:}(x, y) = x + y + 1 \qquad \mathrm{Pol}_{[]} = 0$$

$$\begin{aligned} \mathrm{Pol}(\mathsf{geo}(\mathtt{x})) &> {}^1/_2 \cdot \mathrm{Pol}(\mathtt{x}) + {}^1/_2 \cdot \mathrm{Pol}(\mathsf{geo}(\mathtt{0}:\mathtt{x})) \\ x + 2 &> x + {}^3/_2 \quad \checkmark \\ &\Longrightarrow \mathcal{R}_{\mathsf{geo}} \text{ is SAST} \end{aligned}$$

[ADY20] Avanzini, Dal Lago, and Yamada. On Probabilistic Term Rewriting (SCICO 2020)

Proving Strong Almost-Sure Termination

Theorem: Proving SAST[ADY20]

Let Pol be a natural & monotonic & multilinear polynomial interpretation.
$\mathcal{R}$ is SAST if for all $\ell \to \{{}^{p_1}r_1, \ldots, {}^{p_k}r_k\} \in \mathcal{R}$: $\quad \mathrm{Pol}(\ell) > p_1 \cdot \mathrm{Pol}(r_1) + \ldots + p_k \cdot \mathrm{Pol}(r_k)$

$\mathcal{R}_{\mathsf{geo}}$: $\qquad \mathsf{geo}(\mathtt{x}) \to \left\{{}^{1/2}\mathtt{x}, {}^{1/2}\mathsf{geo}(\mathtt{0}:\mathtt{x})\right\}$

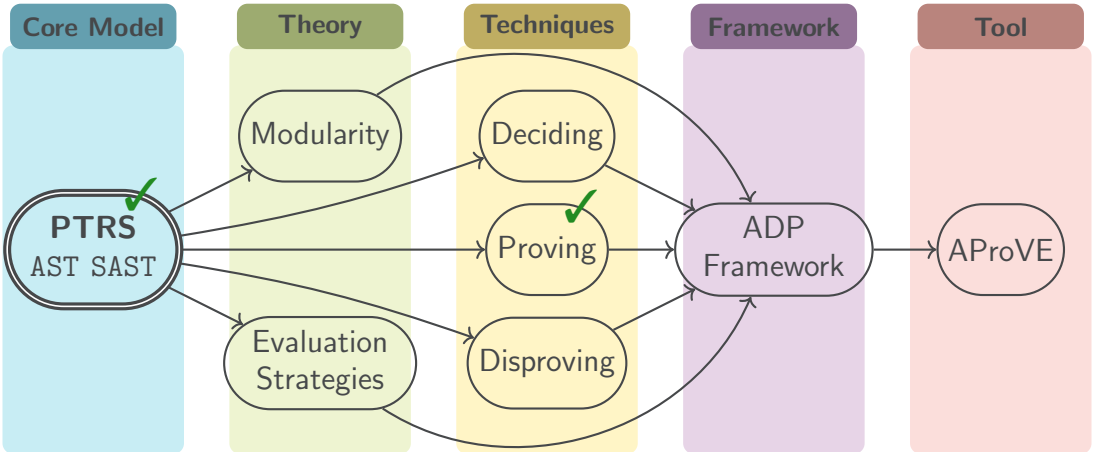
$$\mathrm{Pol}_{\mathsf{geo}}(x) = x + 2 \qquad \mathrm{Pol}_{\mathtt{0}} = 0 \qquad \mathrm{Pol}_{:}(x, y) = x + y + 1 \qquad \mathrm{Pol}_{[]} = 0$$

$$\begin{aligned} \mathrm{Pol}(\mathsf{geo}(\mathtt{x})) &> 1/2 \cdot \mathrm{Pol}(\mathtt{x}) + 1/2 \cdot \mathrm{Pol}(\mathsf{geo}(\mathtt{0}:\mathtt{x})) \\ x + 2 &> x + 3/2 \quad \checkmark \\ &\Longrightarrow \mathcal{R}_{\mathsf{geo}} \text{ is SAST} \end{aligned}$$

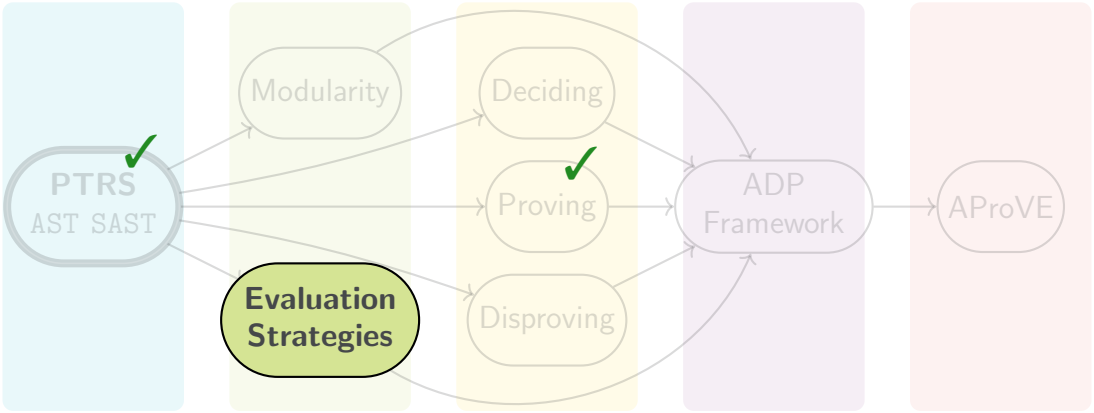
Goal: Infer expected complexity from the highest degree of Pol.[PPDP '25]

[PPDP '25] Kassing, Spitzer, and Giesl. Dependency Pairs for Expected Innermost Runtime Complexity [...] (PPDP 2025)
[ADY20] Avanzini, Dal Lago, and Yamada. On Probabilistic Term Rewriting (SCICO 2020)

How to Automatically Analyze Probabilistic Programs?



Evaluation Strategies



Evaluation Strategies: Core Intuition

Core Idea

Innermost evaluation is the easiest to analyze: when does it suffice?

$\mathcal{R}$:

$$\begin{aligned} \mathtt{f}(\mathtt{g}(\mathtt{[]})) &\to \mathtt{f}(\mathtt{g}(\mathtt{[]})) \\ \mathtt{g}(\mathtt{[]}) &\to \mathtt{[]} \end{aligned}$$

Evaluation Strategies: Core Intuition

Core Idea

Innermost evaluation is the easiest to analyze: when does it suffice?

$\mathcal{R}$:

$$\begin{aligned} \mathtt{f}(\mathtt{g}(\mathtt{[]})) &\to \mathtt{f}(\mathtt{g}(\mathtt{[]})) \\ \mathtt{g}(\mathtt{[]}) &\to \mathtt{[]} \end{aligned}$$

Full Rewriting (any redex): **Terminating? No**

Evaluation Strategies: Core Intuition

Core Idea

Innermost evaluation is the easiest to analyze: when does it suffice?

$\mathcal{R}$:

$$\begin{aligned} \mathsf{f}(\mathsf{g}(\square)) &\to \mathsf{f}(\mathsf{g}(\square)) \\ \mathsf{g}(\square) &\to \square \end{aligned}$$

Full Rewriting (any redex): **Terminating? No**

$\mathsf{f}(\mathsf{g}(\square))$

Evaluation Strategies: Core Intuition

Core Idea

Innermost evaluation is the easiest to analyze: when does it suffice?

$\mathcal{R}$:

$$\begin{aligned} \mathtt{f}(\mathtt{g}(\mathtt{[]})) &\to \mathtt{f}(\mathtt{g}(\mathtt{[]})) \\ \mathtt{g}(\mathtt{[]}) &\to \mathtt{[]} \end{aligned}$$

Full Rewriting (any redex): **Terminating? No**

$\mathtt{f}(\mathtt{g}(\mathtt{[]})) \to_{\mathcal{R}} \mathtt{f}(\mathtt{g}(\mathtt{[]}))$

Evaluation Strategies: Core Intuition

Core Idea

Innermost evaluation is the easiest to analyze: when does it suffice?

$\mathcal{R}$:

$$\begin{aligned} \mathsf{f}(\mathsf{g}(\square)) &\to \mathsf{f}(\mathsf{g}(\square)) \\ \mathsf{g}(\square) &\to \square \end{aligned}$$

Full Rewriting (any redex): **Terminating? No**

$$\mathsf{f}(\mathsf{g}(\square)) \to_{\mathcal{R}} \mathsf{f}(\mathsf{g}(\square)) \to_{\mathcal{R}} \dots$$

Evaluation Strategies: Core Intuition

Core Idea

Innermost evaluation is the easiest to analyze: when does it suffice?

$\mathcal{R}$:

$$\begin{aligned} \mathsf{f}(\mathsf{g}(\square)) &\to \mathsf{f}(\mathsf{g}(\square)) \\ \mathsf{g}(\square) &\to \square \end{aligned}$$

Full Rewriting (any redex): **Terminating? No**

$$\mathsf{f}(\mathsf{g}(\square)) \to_{\mathcal{R}} \mathsf{f}(\mathsf{g}(\square)) \to_{\mathcal{R}} \dots$$

Innermost (arguments first): **Terminating? Yes**

Evaluation Strategies: Core Intuition

Core Idea

Innermost evaluation is the easiest to analyze: when does it suffice?

$\mathcal{R}$:

$$\begin{aligned} \mathsf{f}(\mathsf{g}(\square)) &\to \mathsf{f}(\mathsf{g}(\square)) \\ \mathsf{g}(\square) &\to \square \end{aligned}$$

Full Rewriting (any redex): **Terminating? No**

$$\mathsf{f}(\mathsf{g}(\square)) \to_{\mathcal{R}} \mathsf{f}(\mathsf{g}(\square)) \to_{\mathcal{R}} \dots$$

Innermost (arguments first): **Terminating? Yes**

$$\mathsf{f}(\mathsf{g}(\square))$$

Evaluation Strategies: Core Intuition

Core Idea

Innermost evaluation is the easiest to analyze: when does it suffice?

$\mathcal{R}$:

$$\begin{aligned} \mathsf{f}(\mathsf{g}(\square)) &\to \mathsf{f}(\mathsf{g}(\square)) \\ \mathsf{g}(\square) &\to \square \end{aligned}$$

Full Rewriting (any redex): **Terminating? No**

$$\mathsf{f}(\mathsf{g}(\square)) \to_{\mathcal{R}} \mathsf{f}(\mathsf{g}(\square)) \to_{\mathcal{R}} \dots$$

Innermost (arguments first): **Terminating? Yes**

$$\mathsf{f}(\mathsf{g}(\square)) \overset{\mathsf{i}}{\to}_{\mathcal{R}} \mathsf{f}(\square)$$

Evaluation Strategies: Core Intuition

Core Idea

Innermost evaluation is the easiest to analyze: when does it suffice?

$\mathcal{R}$:

$$\begin{aligned} \mathsf{f}(\mathsf{g}(\square)) &\to \mathsf{f}(\mathsf{g}(\square)) \\ \mathsf{g}(\square) &\to \square \end{aligned}$$

Full Rewriting (any redex): **Terminating? No**

$\mathsf{f}(\mathsf{g}(\square)) \to_{\mathcal{R}} \mathsf{f}(\mathsf{g}(\square)) \to_{\mathcal{R}} \ldots$

Innermost (arguments first): **Terminating? Yes**

$\mathsf{f}(\mathsf{g}(\square)) \overset{\mathsf{i}}{\to}_{\mathcal{R}} \mathsf{f}(\square) \leftarrow$ normal form

Evaluation Strategies: Core Intuition

Core Idea

Innermost evaluation is the easiest to analyze: when does it suffice?

$\mathcal{R}$:

$$\begin{aligned} \mathtt{f}(\mathtt{g}(\mathtt{[]})) &\to \mathtt{f}(\mathtt{g}(\mathtt{[]})) \\ \mathtt{g}(\mathtt{[]}) &\to \mathtt{[]} \end{aligned}$$

Full Rewriting (any redex): **Terminating? No**

$\mathtt{f}(\mathtt{g}(\mathtt{[]})) \to_{\mathcal{R}} \mathtt{f}(\mathtt{g}(\mathtt{[]})) \to_{\mathcal{R}} \ldots$

Innermost (arguments first): **Terminating? Yes**

$\mathtt{f}(\mathtt{g}(\mathtt{[]})) \xrightarrow{\mathsf{i}}_{\mathcal{R}} \mathtt{f}(\mathtt{[]})$ ← normal form

Termination $\implies$ innermost termination, but not vice versa

Evaluation Strategies: Core Intuition

Core Idea

Innermost evaluation is the easiest to analyze: when does it suffice?

$\mathcal{R}$:

$$\begin{aligned} \mathtt{f(g([]))} &\to \mathtt{f(g([]))} \\ \mathtt{g([])} &\to \mathtt{[]} \end{aligned}$$

Full Rewriting (any redex): **Terminating? No**

$\mathtt{f(g([]))} \to_{\mathcal{R}} \mathtt{f(g([]))} \to_{\mathcal{R}} \ldots$

Innermost (arguments first): **Terminating? Yes**

$\mathtt{f(g([]))} \xrightarrow{\mathrm{i}}_{\mathcal{R}} \mathtt{f([])} \leftarrow$ normal form

Termination $\implies$ innermost termination, but not vice versa

Goal: decidable conditions such that $\mathtt{iAST} \implies \mathtt{AST}$ and $\mathtt{iSAST} \implies \mathtt{SAST}$

Innermost Termination vs. Termination

Required Conditions

Innermost Termination vs. Termination

Required Conditions

- **Non-Overlapping:** No left-hand sides overlap.

Innermost Termination vs. Termination

Required Conditions

- **Non-Overlapping:** No left-hand sides overlap.

  $\mathtt{f}(\mathtt{[]}) \to \ldots$ and $\mathtt{f}(\mathtt{[]}) \to \ldots$ overlap

Innermost Termination vs. Termination

Required Conditions

- **Non-Overlapping:** No left-hand sides overlap.

  $\texttt{f([])} \to \ldots$ and $\texttt{f([])} \to \ldots$ overlap $\texttt{f([])} \to \ldots$ and $\texttt{f(x:xs)} \to \ldots$ do not overlap

Innermost Termination vs. Termination

Required Conditions

- **Non-Overlapping:** No left-hand sides overlap.

  $\mathtt{f}(\mathtt{[]}) \to \ldots$ and $\mathtt{f}(\mathtt{[]}) \to \ldots$ overlap $\mathtt{f}(\mathtt{[]}) \to \ldots$ and $\mathtt{f}(\mathtt{x:xs}) \to \ldots$ do not overlap

- **Linear:** Each variable at most once in a left-hand side and in each term of the right-hand side.

Innermost Termination vs. Termination

Required Conditions

- **Non-Overlapping:** No left-hand sides overlap.

  $\mathtt{f}(\mathtt{[]}) \to \ldots$ and $\mathtt{f}(\mathtt{[]}) \to \ldots$ overlap $\mathtt{f}(\mathtt{[]}) \to \ldots$ and $\mathtt{f}(\mathtt{x:xs}) \to \ldots$ do not overlap

- **Linear:** Each variable at most once in a left-hand side and in each term of the right-hand side.

  $\mathrm{g}(x, x) \to \ldots$ not linear

Innermost Termination vs. Termination

Required Conditions

- **Non-Overlapping:** No left-hand sides overlap.

  $\mathtt{f}(\mathtt{[]}) \to \ldots$ and $\mathtt{f}(\mathtt{[]}) \to \ldots$ overlap $\mathtt{f}(\mathtt{[]}) \to \ldots$ and $\mathtt{f}(\mathtt{x:xs}) \to \ldots$ do not overlap

- **Linear:** Each variable at most once in a left-hand side and in each term of the right-hand side.

  $\mathsf{g}(x, x) \to \ldots$ not linear $\mathsf{f}(x) \to \{^1\mathsf{g}(x, x)\}$ not linear

Innermost Termination vs. Termination

Required Conditions

- **Non-Overlapping:** No left-hand sides overlap.

  $\mathtt{f}(\mathtt{[]}) \to \ldots$ and $\mathtt{f}(\mathtt{[]}) \to \ldots$ overlap $\mathtt{f}(\mathtt{[]}) \to \ldots$ and $\mathtt{f}(\mathtt{x:xs}) \to \ldots$ do not overlap

- **Linear:** Each variable at most once in a left-hand side and in each term of the right-hand side.

  $\mathsf{g}(x, x) \to \ldots$ not linear $\mathsf{f}(x) \to \{^{1}\mathsf{g}(x, x)\}$ not linear $\mathsf{f}(x) \to \{^{1/2}\mathsf{f}(x), ^{1/2}\mathsf{h}(x)\}$ linear

Innermost Termination vs. Termination

Required Conditions

- **Non-Overlapping:** No left-hand sides overlap.

  $\mathtt{f}(\mathtt{[]}) \to \ldots$ and $\mathtt{f}(\mathtt{[]}) \to \ldots$ overlap $\mathtt{f}(\mathtt{[]}) \to \ldots$ and $\mathtt{f}(\mathtt{x:xs}) \to \ldots$ do not overlap

- **Linear:** Each variable at most once in a left-hand side and in each term of the right-hand side.

  $\mathsf{g}(x, x) \to \ldots$ not linear $\mathsf{f}(x) \to \{^1\mathsf{g}(x, x)\}$ not linear $\mathsf{f}(x) \to \{^{1/2}\mathsf{f}(x), ^{1/2}\mathsf{h}(x)\}$ linear

Theorem [FoSSaCS '24][LMCS '25]

If $\mathcal{R}$ is non-overlapping, and linear, then: $\mathcal{R}$ is AST $\iff$ $\mathcal{R}$ is iAST $\mathcal{R}$ is SAST $\iff$ $\mathcal{R}$ is iSAST.

[LMCS '25] Kassing and Giesl. From Innermost to Full Probabilistic Term Rewriting: AST, Complexity, and Modularity (LMCS 2025)
[FoSSaCS '24] Kassing, Frohn, and Giesl. From Innermost to Full Almost-Sure Termination [...] (FoSSaCS 2024)

Innermost Termination vs. Termination

Required Conditions

- **Non-Overlapping:** No left-hand sides overlap.

  $\mathtt{f}(\mathtt{[]}) \to \ldots$ and $\mathtt{f}(\mathtt{[]}) \to \ldots$ overlap $\mathtt{f}(\mathtt{[]}) \to \ldots$ and $\mathtt{f}(\mathtt{x:xs}) \to \ldots$ do not overlap

- **Linear:** Each variable at most once in a left-hand side and in each term of the right-hand side.

  $\mathsf{g}(x, x) \to \ldots$ not linear $\mathsf{f}(x) \to \{^1\mathsf{g}(x, x)\}$ not linear $\mathsf{f}(x) \to \{^{1/2}\mathsf{f}(x), ^{1/2}\mathsf{h}(x)\}$ linear

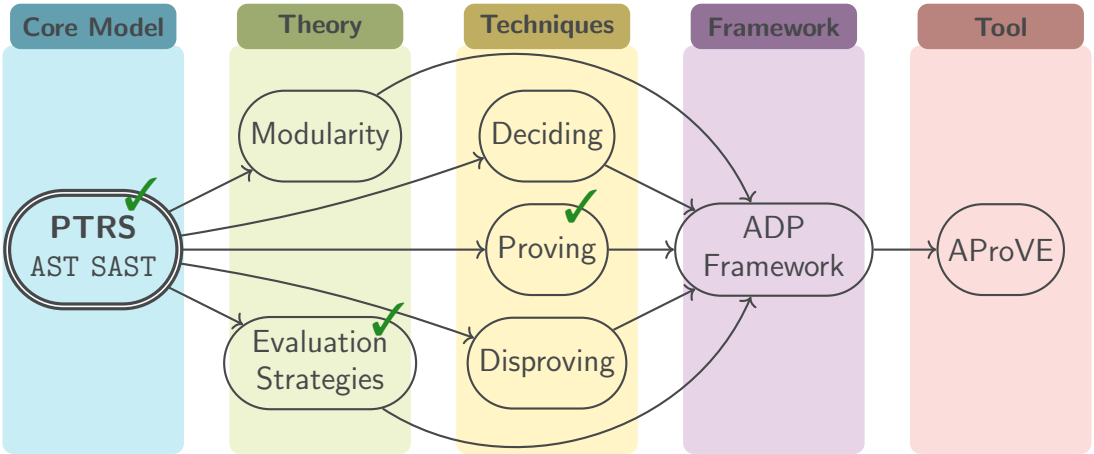
Theorem[FoSSaCS '24][LMCS '25]

If $\mathcal{R}$ is non-overlapping, and linear, then: $\mathcal{R}$ is AST $\iff \mathcal{R}$ is iAST $\mathcal{R}$ is SAST $\iff \mathcal{R}$ is iSAST.

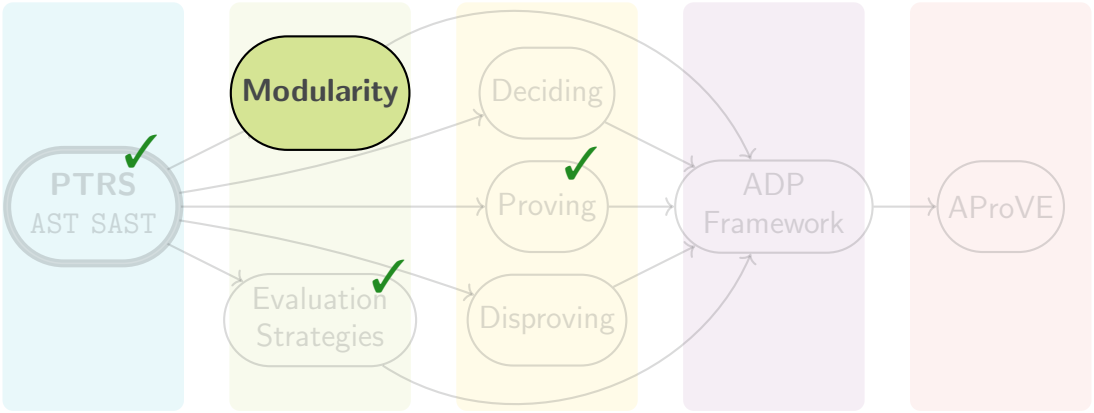
For TRSs, non-overlapping suffices: $\mathcal{R}$ terminating $\iff \mathcal{R}$ innermost terminating.[Gra95]

[Gra95] Gramlich. Abstract Relations between Restricted Termination and Confluence Properties [...] (Fundam. Inform. 1995)
[LMCS '25] Kassing and Giesl. From Innermost to Full Probabilistic Term Rewriting: AST, Complexity, and Modularity (LMCS 2025)
[FoSSaCS '24] Kassing, Frohn, and Giesl. From Innermost to Full Almost-Sure Termination [...] (FoSSaCS 2024)

How to Automatically Analyze Probabilistic Programs?



Modularity Results



Modularity: Core Intuition

Core Idea

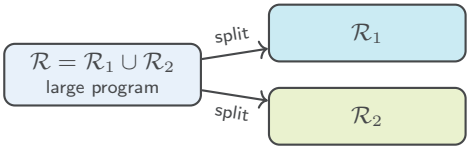
Divide and conquer: split the rule set into parts, analyze each part on its own, and combine the results.

$\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$
large program

Modularity: Core Intuition

Core Idea

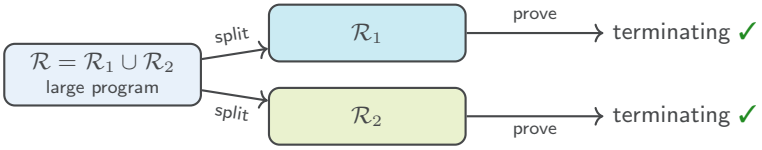
Divide and conquer: split the rule set into parts, analyze each part on its own, and combine the results.



Modularity: Core Intuition

Core Idea

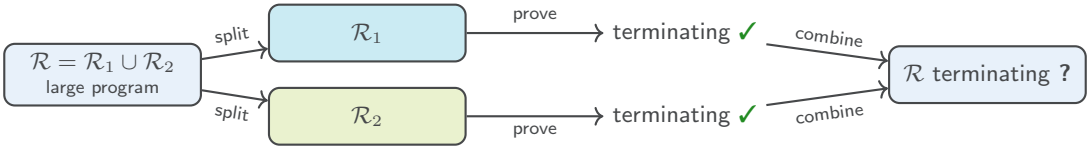
Divide and conquer: split the rule set into parts, analyze each part on its own, and combine the results.



Modularity: Core Intuition

Core Idea

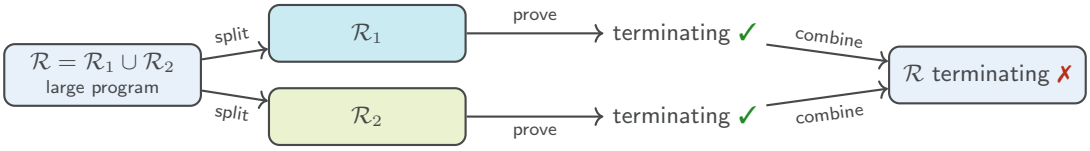
Divide and conquer: split the rule set into parts, analyze each part on its own, and combine the results.



Modularity: Core Intuition

Core Idea

Divide and conquer: split the rule set into parts, analyze each part on its own, and combine the results.



Termination Is Not Modular in General[Toy87]

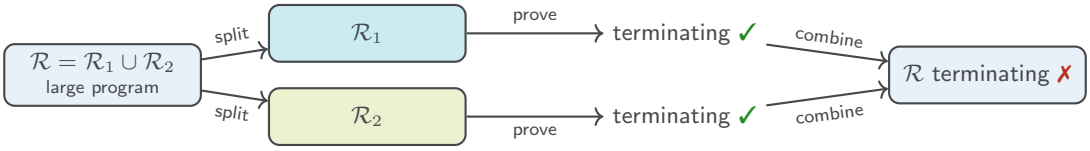
$\mathcal{R}_1$: $\mathsf{f}(0, 1, x) \to \mathsf{f}(x, x, x)$ $\mathcal{R}_2$: $\mathsf{g}(x, y) \to x, \quad \mathsf{g}(x, y) \to y$ both terminating ✓

[Toy87] Toyama. Counterexamples to Termination for the Direct Sum of Term Rewriting Systems (IPL 1987)

Modularity: Core Intuition

Core Idea

Divide and conquer: split the rule set into parts, analyze each part on its own, and combine the results.



Termination Is Not Modular in General[Toy87]

$\mathcal{R}_1$: $\mathsf{f}(0,1,x) \to \mathsf{f}(x,x,x)$ $\quad$ $\mathcal{R}_2$: $\mathsf{g}(x,y) \to x, \quad \mathsf{g}(x,y) \to y$ $\quad$ both terminating ✓

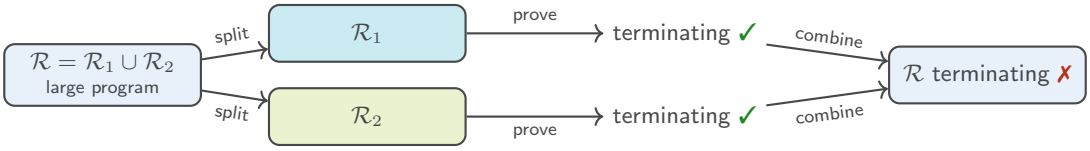
$\mathsf{f}(0,1,\mathsf{g}(0,1)) \to_{\mathcal{R}_1} \mathsf{f}(\mathsf{g}(0,1),\mathsf{g}(0,1),\mathsf{g}(0,1)) \to_{\mathcal{R}_2} \mathsf{f}(\mathsf{g}(0,1),1,\mathsf{g}(0,1)) \to_{\mathcal{R}_2} \mathsf{f}(0,1,\mathsf{g}(0,1)) \to_{\mathcal{R}_1} \cdots$ ✗

[Toy87] Toyama. Counterexamples to Termination for the Direct Sum of Term Rewriting Systems (IPL 1987)

Modularity: Core Intuition

Core Idea

Divide and conquer: split the rule set into parts, analyze each part on its own, and combine the results.



Termination Is Not Modular in General[Toy87]

$\mathcal{R}_1$: $\mathsf{f}(0,1,x) \to \mathsf{f}(x,x,x)$ $\quad$ $\mathcal{R}_2$: $\mathsf{g}(x,y) \to x, \quad \mathsf{g}(x,y) \to y$ $\quad$ both terminating ✓

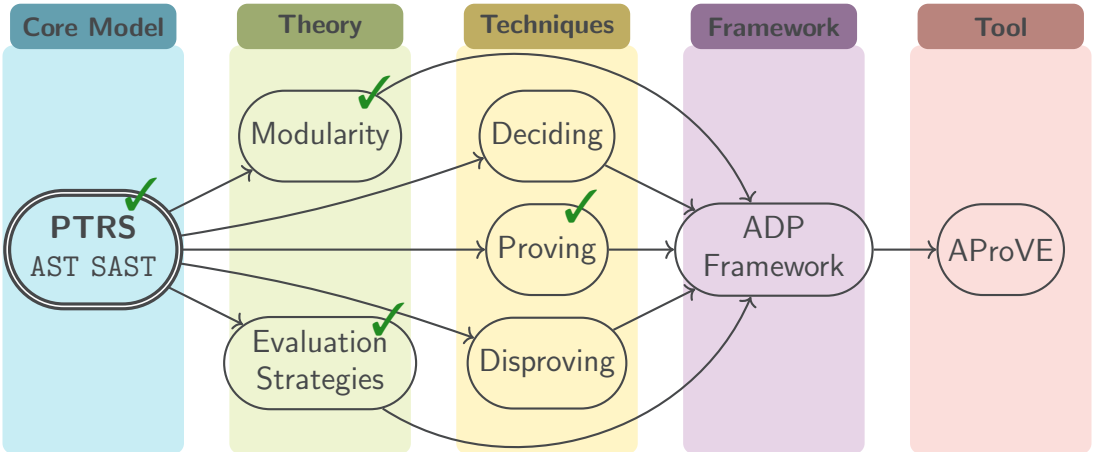
$\mathsf{f}(0,1,\mathsf{g}(0,1)) \to_{\mathcal{R}_1} \mathsf{f}(\mathsf{g}(0,1),\mathsf{g}(0,1),\mathsf{g}(0,1)) \to_{\mathcal{R}_2} \mathsf{f}(\mathsf{g}(0,1),1,\mathsf{g}(0,1)) \to_{\mathcal{R}_2} \mathsf{f}(0,1,\mathsf{g}(0,1)) \to_{\mathcal{R}_1} \cdots$ ✗

Solution: innermost rewriting + further restrictions[LMCS '25]

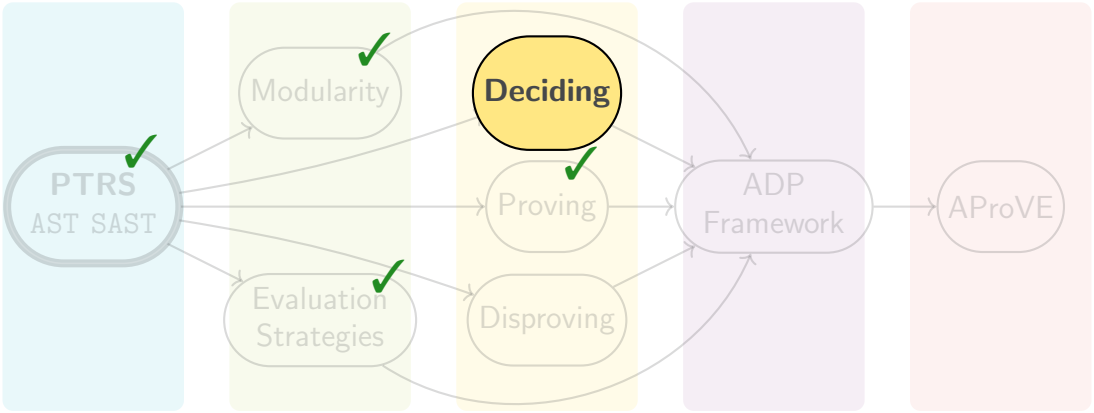
[LMCS '25] Kassing and Giesl. From Innermost to Full Probabilistic Term Rewriting: AST, Complexity, and Modularity (LMCS 2025)

[Toy87] Toyama. Counterexamples to Termination for the Direct Sum of Term Rewriting Systems (IPL 1987)

How to Automatically Analyze Probabilistic Programs?



Decidability



Decidability Results: Where Is AST Decidable?

Core Idea

Restrict the rules of a PTRS such that AST becomes decidable.

Decidability Results: Where Is AST Decidable?

Core Idea

Restrict the rules of a PTRS such that AST becomes decidable.

random walks

Decidability Results: Where Is AST Decidable?

Core Idea

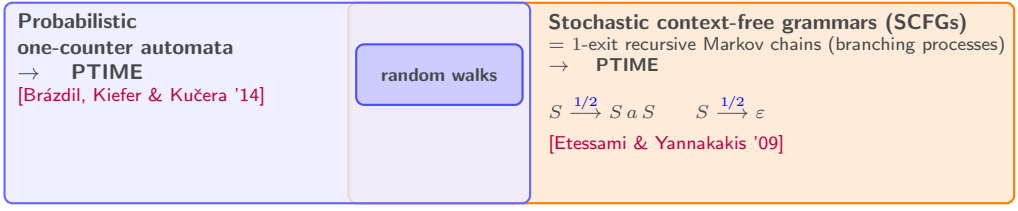
Restrict the rules of a PTRS such that AST becomes decidable.



Decidability Results: Where Is AST Decidable?

Core Idea

Restrict the rules of a PTRS such that AST becomes decidable.



Decidability Results: Where Is AST Decidable?

Core Idea

Restrict the rules of a PTRS such that AST becomes decidable.

Probabilistic pushdown automata (pPDAs) = recursive Markov chains = order-1 pHORS → **PSPACE**
[Esparza, Kučera & Mayr '06], [Etessami & Yannakakis '09], [Kobayashi, Dal Lago & Grellois '20]

Probabilistic one-counter automata
→ **PTIME**
[Brázdil, Kiefer & Kučera '14]

random walks

Stochastic context-free grammars (SCFGs)
= 1-exit recursive Markov chains (branching processes)
→ **PTIME**

$S \xrightarrow{1/2} S\,a\,S \qquad S \xrightarrow{1/2} \varepsilon$

[Etessami & Yannakakis '09]

Decidability Results: Where Is AST Decidable?

Core Idea

Restrict the rules of a PTRS such that AST becomes decidable.

Affine probabilistic higher-order recursion schemes (pHORS) → **EXPTIME**
[Dal Lago, Fiorillo & Pistone '26]

Probabilistic pushdown automata (pPDAs) = recursive Markov chains = order-1 pHORS → **PSPACE**
[Esparza, Kučera & Mayr '06], [Etessami & Yannakakis '09], [Kobayashi, Dal Lago & Grellois '20]

Probabilistic one-counter automata
→ **PTIME**
[Brázdil, Kiefer & Kučera '14]

random walks

Stochastic context-free grammars (SCFGs)
= 1-exit recursive Markov chains (branching processes)
→ **PTIME**

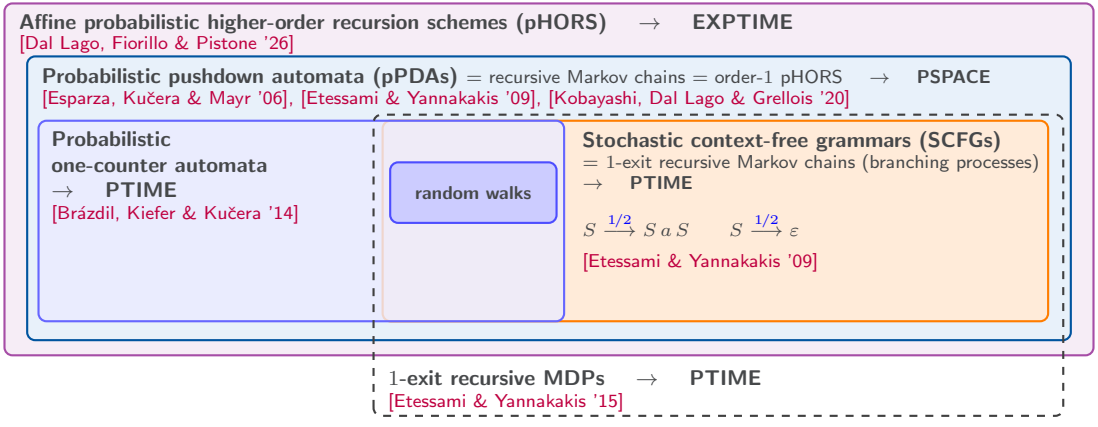
$S \xrightarrow{1/2} S\,a\,S \qquad S \xrightarrow{1/2} \varepsilon$

[Etessami & Yannakakis '09]

Decidability Results: Where Is AST Decidable?

Core Idea

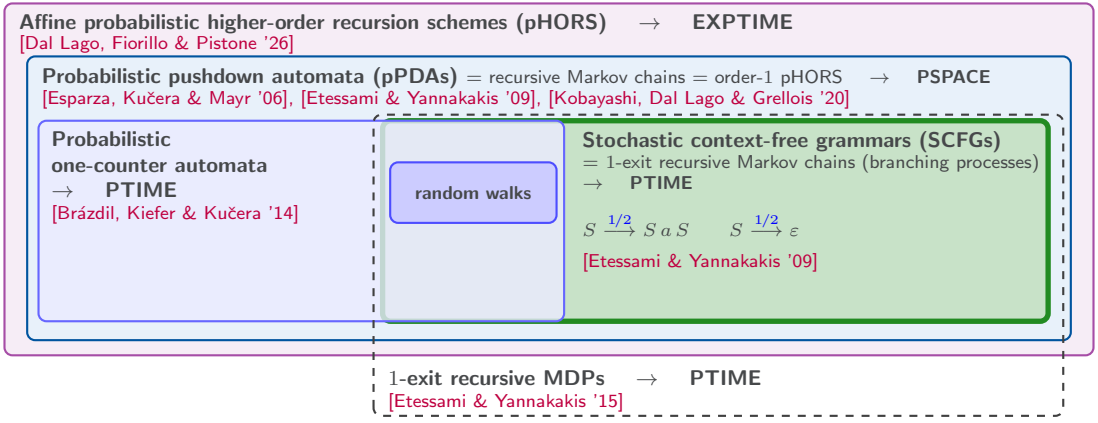
Restrict the rules of a PTRS such that AST becomes decidable.



Decidability Results: Where Is AST Decidable?

Core Idea

Restrict the rules of a PTRS such that AST becomes decidable.



Decidability Results for PTRSs

The Class RNT: **R**ight-Ground

Right-Ground:
no variables in right-hand sides

✓ $\mathsf{g} \to \{^1\mathsf{c}(\mathsf{g},\mathsf{g})\}$ ✗ $\mathsf{f}(x) \to \{^1\mathsf{f}(x)\}$

Decidability Results for PTRSs

The Class RNT: **R**ight-Ground + **N**on-Overlapping

Right-Ground:
no variables in right-hand sides

✓ $\mathsf{g} \to \{^1\mathsf{c}(\mathsf{g},\mathsf{g})\}$ ✗ $\mathsf{f}(x) \to \{^1\mathsf{f}(x)\}$

Non-Overlapping:
left-hand sides do not unify

✓ a, $\mathsf{f}(\mathsf{b})$ ✗ a, $\mathsf{f}(\mathsf{a})$

Decidability Results for PTRSs

The Class RNT: **R**ight-Ground + **N**on-Overlapping + **T**ail-Recursive

Right-Ground:
no variables in right-hand sides

✓ $\mathsf{g} \to \{^1\mathsf{c}(\mathsf{g},\mathsf{g})\}$ ✗ $\mathsf{f}(x) \to \{^1\mathsf{f}(x)\}$

Non-Overlapping:
left-hand sides do not unify

✓ a, $\mathsf{f}(\mathsf{b})$ ✗ a, $\mathsf{f}(\mathsf{a})$

Tail-Recursive:
no defined symbol below another

✗ $\mathsf{f}(x) \to \{^1\mathsf{f}(\mathsf{g}(\mathsf{f}(x)))\}$

Decidability Results for PTRSs

The Class RNT: **R**ight-Ground + **N**on-Overlapping + **T**ail-Recursive

Right-Ground:
no variables in right-hand sides

✓ $\mathsf{g} \to \{^1\mathsf{c}(\mathsf{g},\mathsf{g})\}$ ✗ $\mathsf{f}(x) \to \{^1\mathsf{f}(x)\}$

Non-Overlapping:
left-hand sides do not unify

✓ a, $\mathsf{f}(\mathsf{b})$ ✗ a, $\mathsf{f}(\mathsf{a})$

Tail-Recursive:
no defined symbol below another

✗ $\mathsf{f}(x) \to \{^1\mathsf{f}(\mathsf{g}(\mathsf{f}(x)))\}$

$\mathcal{P} \in$ RNT: right-ground + NO + TR

Decidability Results for PTRSs

The Class RNT: **R**ight-Ground + **N**on-Overlapping + **T**ail-Recursive

Right-Ground:
no variables in right-hand sides

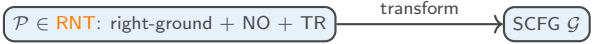
✓ $\mathtt{g} \to \{^1\mathtt{c}(\mathtt{g},\mathtt{g})\}$ ✗ $\mathtt{f}(x) \to \{^1\mathtt{f}(x)\}$

Non-Overlapping:
left-hand sides do not unify

✓ $\mathtt{a}$, $\mathtt{f}(\mathtt{b})$ ✗ $\mathtt{a}$, $\mathtt{f}(\mathtt{a})$

Tail-Recursive:
no defined symbol below another

✗ $\mathtt{f}(x) \to \{^1\mathtt{f}(\mathtt{g}(\mathtt{f}(x)))\}$



Decidability Results for PTRSs

The Class RNT: **R**ight-Ground + **N**on-Overlapping + **T**ail-Recursive

Right-Ground:
no variables in right-hand sides

✓ $\mathtt{g} \to \{^1\mathtt{c}(\mathtt{g},\mathtt{g})\}$ ✗ $\mathtt{f}(x) \to \{^1\mathtt{f}(x)\}$

Non-Overlapping:
left-hand sides do not unify

✓ $\mathtt{a}$, $\mathtt{f}(\mathtt{b})$ ✗ $\mathtt{a}$, $\mathtt{f}(\mathtt{a})$

Tail-Recursive:
no defined symbol below another

✗ $\mathtt{f}(x) \to \{^1\mathtt{f}(\mathtt{g}(\mathtt{f}(x)))\}$



[EY09] Etessami and Yannakakis. Recursive Markov Chains, Stochastic Grammars, and Monotone Systems [...] (JACM 2009)

Decidability Results for PTRSs

The Class RNT: **R**ight-Ground + **N**on-Overlapping + **T**ail-Recursive

Right-Ground:
no variables in right-hand sides

✓ $\mathtt{g} \to \{^1\mathtt{c}(\mathtt{g},\mathtt{g})\}$ ✗ $\mathtt{f}(x) \to \{^1\mathtt{f}(x)\}$

Non-Overlapping:
left-hand sides do not unify

✓ $\mathtt{a},\ \mathtt{f}(\mathtt{b})$ ✗ $\mathtt{a},\ \mathtt{f}(\mathtt{a})$

Tail-Recursive:
no defined symbol below another

✗ $\mathtt{f}(x) \to \{^1\mathtt{f}(\mathtt{g}(\mathtt{f}(x)))\}$



Non-Probabilistic Setting: right-ground alone suffices, and termination is decidable.[Der81]

[Der81] Dershowitz. Termination of Linear Rewriting Systems (ICALP 1981)

[EY09] Etessami and Yannakakis. Recursive Markov Chains, Stochastic Grammars, and Monotone Systems [...] (JACM 2009)

Decidability Results for PTRSs

The Class RNT: **R**ight-Ground + **N**on-Overlapping + **T**ail-Recursive

Right-Ground:
no variables in right-hand sides

✓ $\mathsf{g} \to \{^1\mathsf{c}(\mathsf{g},\mathsf{g})\}$ ✗ $\mathsf{f}(x) \to \{^1\mathsf{f}(x)\}$

Non-Overlapping:
left-hand sides do not unify

✓ a, $\mathsf{f}(\mathsf{b})$ ✗ a, $\mathsf{f}(\mathsf{a})$

Tail-Recursive:
no defined symbol below another

✗ $\mathsf{f}(x) \to \{^1\mathsf{f}(\mathsf{g}(\mathsf{f}(x)))\}$



Non-Probabilistic Setting: right-ground alone suffices, and termination is decidable.[Der81]

Probabilistic Setting: right-ground alone does not suffice, and AST is undecidable already for ground PTRSs.

[Der81] Dershowitz. Termination of Linear Rewriting Systems (ICALP 1981)

[EY09] Etessami and Yannakakis. Recursive Markov Chains, Stochastic Grammars, and Monotone Systems [...] (JACM 2009)

Decidability Results for PTRSs

The Class RNT: **R**ight-Ground + **N**on-Overlapping + **T**ail-Recursive

Right-Ground:
no variables in right-hand sides

✓ $\mathtt{g} \to \{^1\mathtt{c}(\mathtt{g},\mathtt{g})\}$ ✗ $\mathtt{f}(x) \to \{^1\mathtt{f}(x)\}$

Non-Overlapping:
left-hand sides do not unify

✓ $\mathtt{a}$, $\mathtt{f}(\mathtt{b})$ ✗ $\mathtt{a}$, $\mathtt{f}(\mathtt{a})$

Tail-Recursive:
no defined symbol below another

✗ $\mathtt{f}(x) \to \{^1\mathtt{f}(\mathtt{g}(\mathtt{f}(x)))\}$



Non-Probabilistic Setting: right-ground alone suffices, and termination is decidable.[Der81]

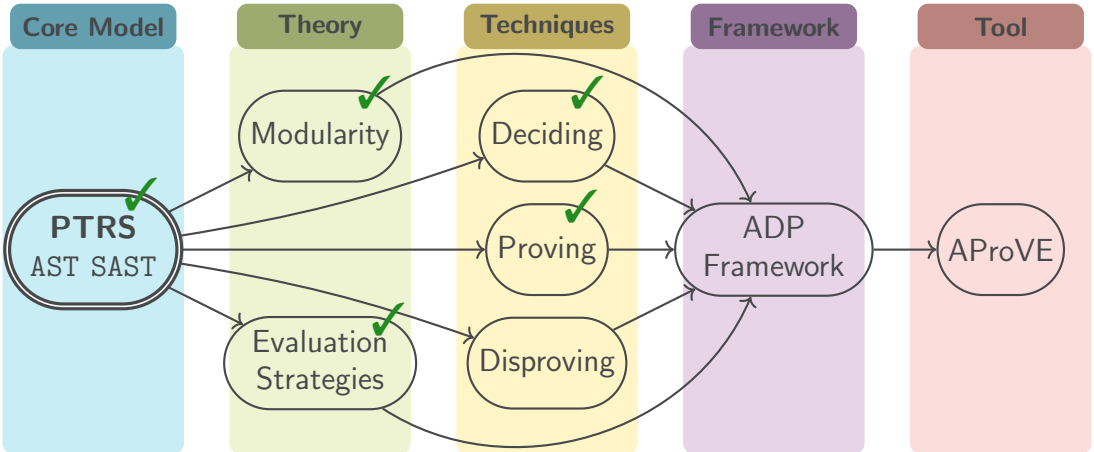
Probabilistic Setting: right-ground alone does not suffice, and AST is undecidable already for ground PTRSs.

$\Longrightarrow$ Decidability and undecidability results to be submitted.

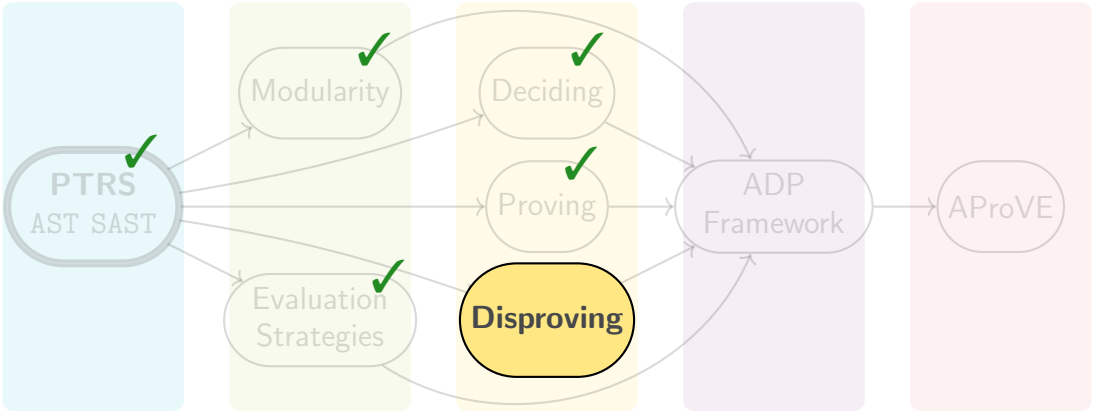
[Der81] Dershowitz. Termination of Linear Rewriting Systems (ICALP 1981)

[EY09] Etessami and Yannakakis. Recursive Markov Chains, Stochastic Grammars, and Monotone Systems [...] (JACM 2009)

How to Automatically Analyze Probabilistic Programs?



Disproving


PTRS
AST SAST
Modularity
Deciding
Proving
Evaluation Strategies
Disproving
ADP Framework
AProVE

Disproving AST and SAST of a PTRS

Core Idea

Disproving finds a bug: look for a self-reproducing pattern — a loop or an embedded random walk.

Disproving Termination of a TRS:

Disproving AST and SAST of a PTRS

Core Idea

Disproving finds a bug: look for a self-reproducing pattern — a loop or an embedded random walk.

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

Disproving AST and SAST of a PTRS

Core Idea

Disproving finds a bug: look for a self-reproducing pattern — a loop or an embedded random walk.

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

$$t \downarrow C[t\sigma]$$

Disproving AST and SAST of a PTRS

Core Idea

Disproving finds a bug: look for a self-reproducing pattern — a loop or an embedded random walk.

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

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

Disproving AST and SAST of a PTRS

Core Idea

Disproving finds a bug: look for a self-reproducing pattern — a loop or an embedded random walk.

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

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

$\mathcal{R}'_{\mathsf{f}}$:

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

Disproving AST and SAST of a PTRS

Core Idea

Disproving finds a bug: look for a self-reproducing pattern — a loop or an embedded random walk.

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

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

$$\mathtt{f(s(x))} \to \mathtt{s(f(s(s(x))))} \to \mathtt{s(s(f(s(s(x)))))} \to \cdots$$

$\mathcal{R}'_{\mathsf{f}}$:

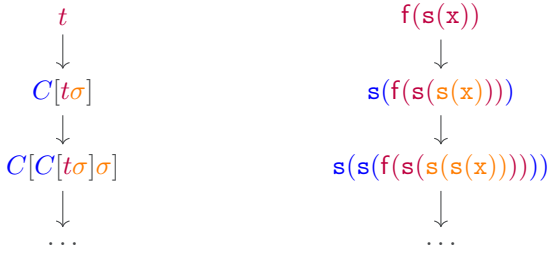
$$\begin{aligned} \mathtt{f(0)} &\to \mathtt{0} \\ \mathtt{f(s(x))} &\to \mathtt{s(f(s(s(x))))} \end{aligned}$$

Disproving AST and SAST of a PTRS

Core Idea

Disproving finds a bug: look for a self-reproducing pattern — a loop or an embedded random walk.

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



$\mathcal{R}'_{\mathsf{f}}$:

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

Disproving AST and SAST of a PTRS

Core Idea

Disproving finds a bug: look for a self-reproducing pattern — a loop or an embedded random walk.

Disproving (S)AST of a PTRS[IJCAR '26]**:** Try to embed random walks:

[IJCAR '26] Kassing, Nagel, Schlecht, and Giesl. Disproving [...] via Random Walks (IJCAR 2026)

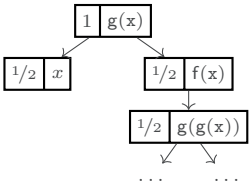
Disproving AST and SAST of a PTRS

Core Idea

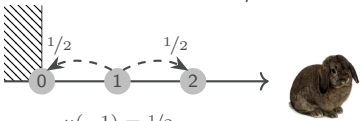
Disproving finds a bug: look for a self-reproducing pattern — a loop or an embedded random walk.

Disproving (S)AST of a PTRS[IJCAR '26]**:** Try to embed random walks:

$\mathcal{R}$ Computation



Symmetric Random Walk μ



$\mu(-1) = 1/2$

$\mu(1) = 1/2$

- What does it mean to find a random walk?

[IJCAR '26] Kassing, Nagel, Schlecht, and Giesl. Disproving [...] via Random Walks (IJCAR 2026)

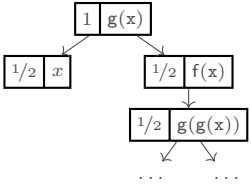
Disproving AST and SAST of a PTRS

Core Idea

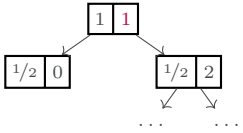
Disproving finds a bug: look for a self-reproducing pattern — a loop or an embedded random walk.

Disproving (S)AST of a PTRS[IJCAR '26]**:** Try to embed random walks:

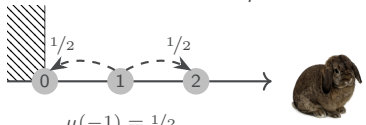
$\mathcal{R}$ Computation



μ Computation



Symmetric Random Walk μ



$\mu(-1) = 1/2$

$\mu(1) = 1/2$

- What does it mean to find a random walk?

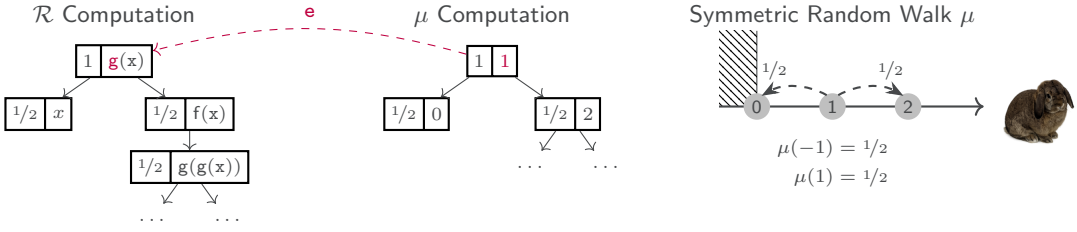
[IJCAR '26] Kassing, Nagel, Schlecht, and Giesl. Disproving [...] via Random Walks (IJCAR 2026)

Disproving AST and SAST of a PTRS

Core Idea

Disproving finds a bug: look for a self-reproducing pattern — a loop or an embedded random walk.

Disproving (S)AST of a PTRS[IJCAR '26]**:** Try to embed random walks:



- What does it mean to find a random walk?

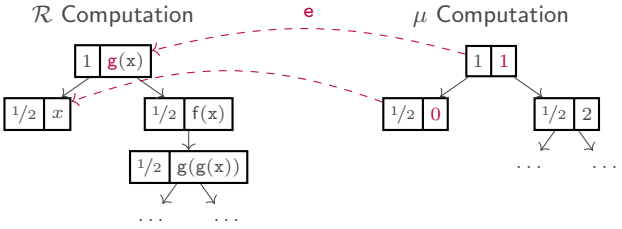
[IJCAR '26] Kassing, Nagel, Schlecht, and Giesl. Disproving [...] via Random Walks (IJCAR 2026)

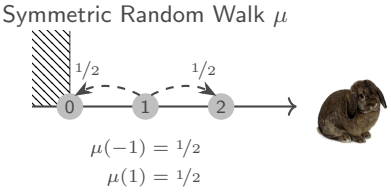
Disproving AST and SAST of a PTRS

Core Idea

Disproving finds a bug: look for a self-reproducing pattern — a loop or an embedded random walk.

Disproving (S)AST of a PTRS[IJCAR '26]**:** Try to embed random walks:





- What does it mean to find a random walk?

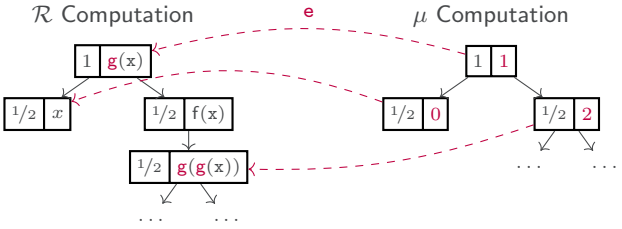
[IJCAR '26] Kassing, Nagel, Schlecht, and Giesl. Disproving [...] via Random Walks (IJCAR 2026)

Disproving AST and SAST of a PTRS

Core Idea

Disproving finds a bug: look for a self-reproducing pattern — a loop or an embedded random walk.

Disproving (S)AST of a PTRS[IJCAR '26]**:** Try to embed random walks:



- What does it mean to find a random walk?

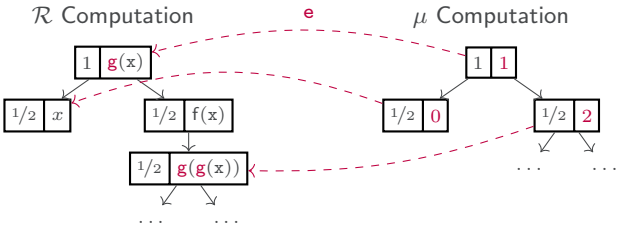
[IJCAR '26] Kassing, Nagel, Schlecht, and Giesl. Disproving [...] via Random Walks (IJCAR 2026)

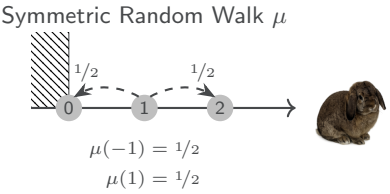
Disproving AST and SAST of a PTRS

Core Idea

Disproving finds a bug: look for a self-reproducing pattern — a loop or an embedded random walk.

Disproving (S)AST of a PTRS[IJCAR '26]**:** Try to embed random walks:

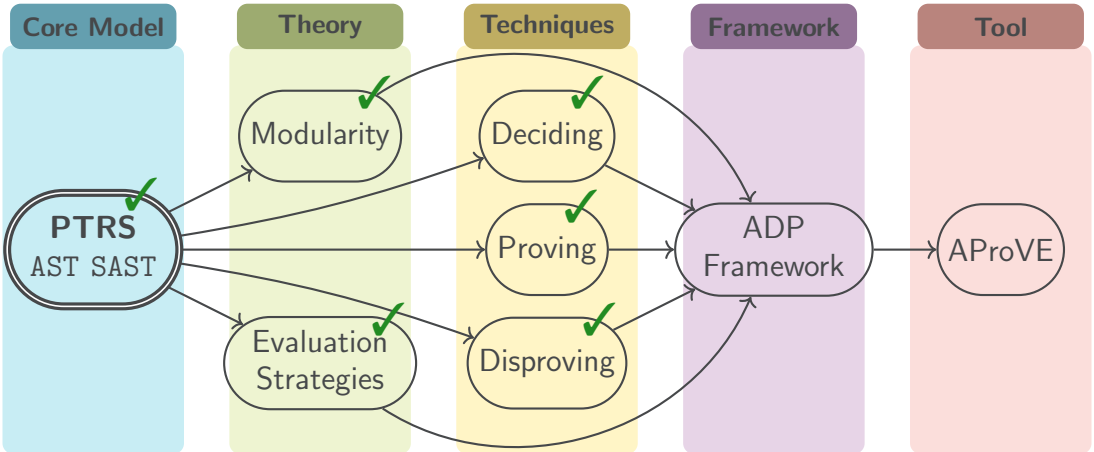




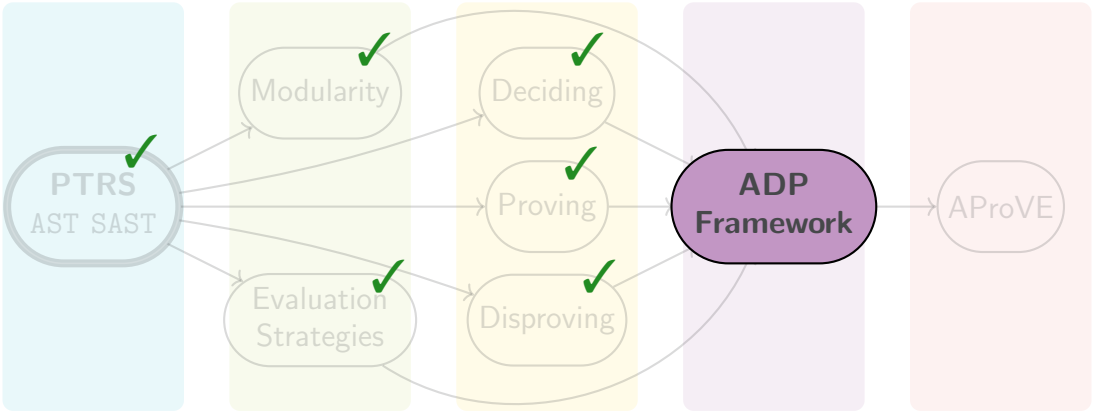
- What does it mean to find a random walk?
  $\rightsquigarrow$ Embedding e from computation of μ to computation of $\mathcal{R}$

[IJCAR '26] Kassing, Nagel, Schlecht, and Giesl. Disproving [...] via Random Walks (IJCAR 2026)

How to Automatically Analyze Probabilistic Programs?



ADP Framework



Dependency Pairs: Core Intuition

Core Idea

A recursive function terminates if its arguments decrease w.r.t. a well-founded order.

Dependency Pairs: Core Intuition

Core Idea

A recursive function terminates if its arguments decrease w.r.t. a well-founded order.

Classical Dependency Pairs[AG00]: keep only the recursive calls and compare their arguments

$$\mathtt{f}(\mathtt{s}(x)) \to \{{}^{1/2}\mathtt{f}(x),\ {}^{1/2}\mathtt{c}(\mathtt{f}(x),\mathtt{f}(x))\} \quad \rightsquigarrow \quad \mathsf{F}(\mathtt{s}(x)) \to \mathsf{F}(x)$$

[AG00] Arts and Giesl. Termination of Term Rewriting Using Dependency Pairs (TCS 2000)

Dependency Pairs: Core Intuition

Core Idea

A recursive function terminates if its arguments decrease w.r.t. a well-founded order.

Classical Dependency Pairs[AG00]: keep only the recursive calls and compare their arguments

$$\mathsf{f}(\mathsf{s}(x)) \to \{^{1/2}\mathsf{f}(x),\ ^{1/2}\mathsf{c}(\mathsf{f}(x),\mathsf{f}(x))\} \quad \rightsquigarrow \quad \mathsf{F}(\mathsf{s}(x)) \to \mathsf{F}(x) \quad \text{with} \quad \mathsf{s}(x) \succ x \ \checkmark$$

[AG00] Arts and Giesl. Termination of Term Rewriting Using Dependency Pairs (TCS 2000)

Dependency Pairs: Core Intuition

Core Idea

A recursive function terminates if its arguments decrease w.r.t. a well-founded order.

Classical Dependency Pairs[AG00]: keep only the recursive calls and compare their arguments

$$\mathsf{f}(\mathsf{s}(x)) \to \{{}^{1/2}\mathsf{f}(x),\ {}^{1/2}\mathsf{c}(\mathsf{f}(x),\mathsf{f}(x))\} \quad \rightsquigarrow \quad \mathsf{F}(\mathsf{s}(x)) \to \mathsf{F}(x) \quad \text{with} \quad \mathsf{s}(x) \succ x \ \checkmark$$

. . . but for AST the probabilities and the number of calls in one term matter!

[AG00] Arts and Giesl. Termination of Term Rewriting Using Dependency Pairs (TCS 2000)

Probabilistic Annotated Dependency Pair Framework

Core Idea

Annotate the recursive calls whose almost-sure termination still has to be shown.

Probabilistic Annotated Dependency Pair Framework

Core Idea

Annotate the recursive calls whose almost-sure termination still has to be shown.

Annotated Dependency Pairs[FLOPS '24][SCP '26]: keep the whole rule and annotate the recursive calls

$$\mathsf{f}(\mathsf{s}(x)) \to \{{}^{1/2}\mathsf{f}(x),\ {}^{1/2}\mathsf{c}(\mathsf{f}(x),\mathsf{f}(x))\} \quad \rightsquigarrow \quad \mathsf{f}(\mathsf{s}(x)) \to \{{}^{1/2}\mathsf{F}(x),\ {}^{1/2}\mathsf{c}(\mathsf{F}(x),\mathsf{F}(x))\}$$

[SCP '26] Kassing and Giesl. The Annotated Dependency Pair Framework for Almost-Sure Termination [...] (SCP 2026)

[FLOPS '24] Kassing, Dollase, and Giesl. A Complete Dependency Pair Framework for Almost-Sure Innermost Termination [...] (FLOPS 2024)

Probabilistic Annotated Dependency Pair Framework

Core Idea

Annotate the recursive calls whose almost-sure termination still has to be shown.

Annotated Dependency Pairs[FLOPS '24][SCP '26]: keep the whole rule and annotate the recursive calls

$$\mathsf{f}(\mathbf{s}(x)) \to \{{}^{1/2}\mathsf{f}(x),\ {}^{1/2}\mathsf{c}(\mathsf{f}(x),\mathsf{f}(x))\} \quad \rightsquigarrow \quad \mathsf{f}(\mathbf{s}(x)) \to \{{}^{1/2}\mathsf{F}(x),\ {}^{1/2}\mathsf{c}(\mathsf{F}(x),\mathsf{F}(x))\}$$

- Show that arguments have to decrease in expectation over all annotated calls.

[SCP '26] Kassing and Giesl. The Annotated Dependency Pair Framework for Almost-Sure Termination [...] (SCP 2026)
[FLOPS '24] Kassing, Dollase, and Giesl. A Complete Dependency Pair Framework for Almost-Sure Innermost Termination [...] (FLOPS 2024)

Probabilistic Annotated Dependency Pair Framework

Core Idea

Annotate the recursive calls whose almost-sure termination still has to be shown.

Annotated Dependency Pairs[FLOPS '24][SCP '26]: keep the whole rule and annotate the recursive calls

$$\mathsf{f}(\mathbf{s}(x)) \to \{^{1/2}\mathsf{f}(x),\ ^{1/2}\mathsf{c}(\mathsf{f}(x),\mathsf{f}(x))\} \quad \rightsquigarrow \quad \mathsf{f}(\mathbf{s}(x)) \to \{^{1/2}\mathsf{F}(x),\ ^{1/2}\mathsf{c}(\mathsf{F}(x),\mathsf{F}(x))\}$$

- Show that arguments have to decrease in expectation over all annotated calls.
- No annotations left $\implies$ AST

[SCP '26] Kassing and Giesl. The Annotated Dependency Pair Framework for Almost-Sure Termination [...] (SCP 2026)
[FLOPS '24] Kassing, Dollase, and Giesl. A Complete Dependency Pair Framework for Almost-Sure Innermost Termination [...] (FLOPS 2024)

Probabilistic Annotated Dependency Pair Framework

Core Idea

Annotate the recursive calls whose almost-sure termination still has to be shown.

Annotated Dependency Pairs[FLOPS '24][SCP '26]: keep the whole rule and annotate the recursive calls

$$\mathsf{f}(\mathsf{s}(x)) \to \{^{1/2}\mathsf{f}(x),\ ^{1/2}\mathsf{c}(\mathsf{f}(x),\mathsf{f}(x))\} \quad \rightsquigarrow \quad \mathsf{f}(\mathsf{s}(x)) \to \{^{1/2}\mathsf{F}(x),\ ^{1/2}\mathsf{c}(\mathsf{F}(x),\mathsf{F}(x))\}$$

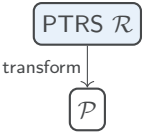
- Show that arguments have to decrease in expectation over all annotated calls.
- No annotations left $\implies$ AST
  (same idea with polynomial bounds yields expected complexity[PPDP '25])

[PPDP '25] Kassing, Spitzer, and Giesl. Dependency Pairs for Expected Innermost Runtime Complexity [...] (PPDP 2025)
[SCP '26] Kassing and Giesl. The Annotated Dependency Pair Framework for Almost-Sure Termination [...] (SCP 2026)
[FLOPS '24] Kassing, Dollase, and Giesl. A Complete Dependency Pair Framework for Almost-Sure Innermost Termination [...] (FLOPS 2024)

The ADP Framework for AST: Proof Trees

ADP Problem $\mathcal{P}$

A set of annotated rules



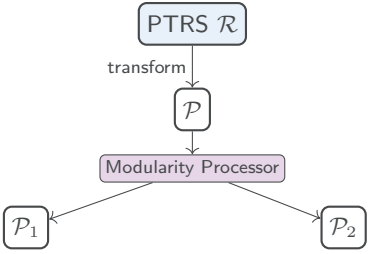
The ADP Framework for AST: Proof Trees

ADP Problem $\mathcal{P}$

A set of annotated rules

Processor $\mathrm{Proc}(\mathcal{P}) = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$

Sound: $\mathcal{P}_1, \ldots, \mathcal{P}_n$ AST $\implies$ $\mathcal{P}$ AST
Complete: $\mathcal{P}$ AST $\implies$ $\mathcal{P}_1, \ldots, \mathcal{P}_n$ AST



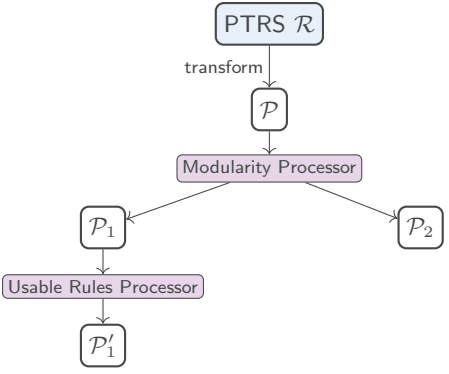
The ADP Framework for AST: Proof Trees

ADP Problem $\mathcal{P}$

A set of annotated rules

Processor $\mathrm{Proc}(\mathcal{P}) = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$

Sound: $\mathcal{P}_1, \ldots, \mathcal{P}_n$ AST $\implies$ $\mathcal{P}$ AST
Complete: $\mathcal{P}$ AST $\implies$ $\mathcal{P}_1, \ldots, \mathcal{P}_n$ AST



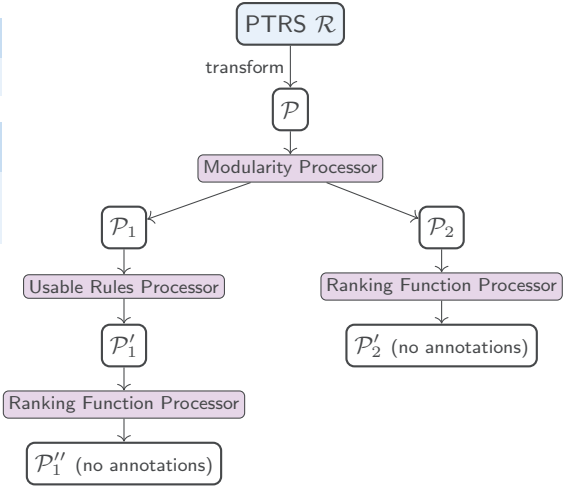
The ADP Framework for AST: Proof Trees

ADP Problem $\mathcal{P}$

A set of annotated rules

Processor $\mathrm{Proc}(\mathcal{P}) = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$

Sound: $\mathcal{P}_1, \ldots, \mathcal{P}_n$ AST $\implies$ $\mathcal{P}$ AST
Complete: $\mathcal{P}$ AST $\implies$ $\mathcal{P}_1, \ldots, \mathcal{P}_n$ AST



The ADP Framework for AST: Proof Trees

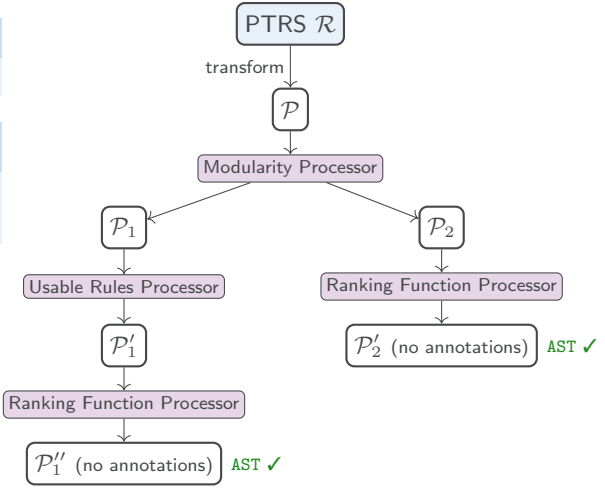
ADP Problem $\mathcal{P}$

A set of annotated rules

Processor $\mathrm{Proc}(\mathcal{P}) = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$

Sound: $\mathcal{P}_1, \ldots, \mathcal{P}_n$ AST $\implies$ $\mathcal{P}$ AST
Complete: $\mathcal{P}$ AST $\implies$ $\mathcal{P}_1, \ldots, \mathcal{P}_n$ AST

- Problems without annotations are AST.



The ADP Framework for AST: Proof Trees

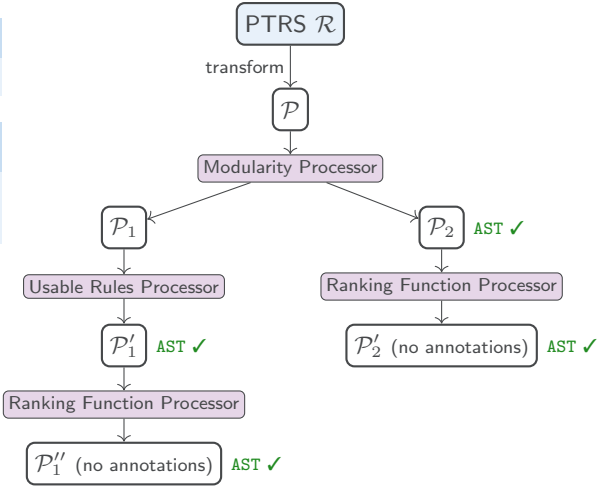
ADP Problem $\mathcal{P}$

A set of annotated rules

Processor $\mathrm{Proc}(\mathcal{P}) = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$

Sound: $\mathcal{P}_1, \ldots, \mathcal{P}_n$ AST $\implies$ $\mathcal{P}$ AST
Complete: $\mathcal{P}$ AST $\implies$ $\mathcal{P}_1, \ldots, \mathcal{P}_n$ AST

- Problems without annotations are AST.
- Soundness: pushes AST **upwards** to the root



The ADP Framework for AST: Proof Trees

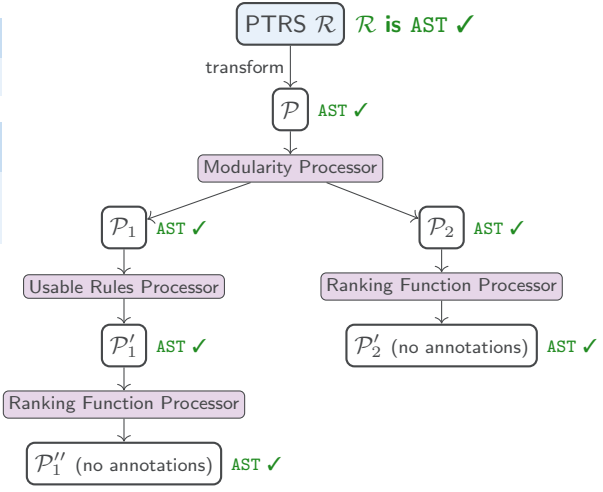
ADP Problem $\mathcal{P}$

A set of annotated rules

Processor $\mathrm{Proc}(\mathcal{P}) = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$

Sound: $\mathcal{P}_1, \ldots, \mathcal{P}_n$ AST $\implies$ $\mathcal{P}$ AST
Complete: $\mathcal{P}$ AST $\implies$ $\mathcal{P}_1, \ldots, \mathcal{P}_n$ AST

- Problems without annotations are AST.
- Soundness: pushes AST **upwards** to the root



The ADP Framework for AST: Proof Trees

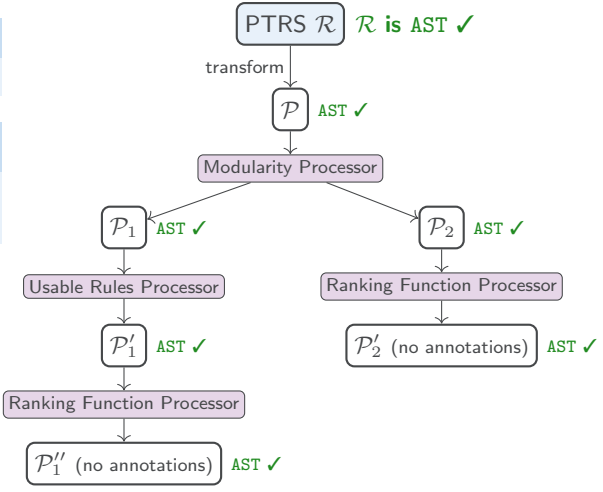
ADP Problem $\mathcal{P}$

A set of annotated rules

Processor $\mathrm{Proc}(\mathcal{P}) = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$

Sound: $\mathcal{P}_1, \ldots, \mathcal{P}_n$ AST $\implies$ $\mathcal{P}$ AST
Complete: $\mathcal{P}$ AST $\implies$ $\mathcal{P}_1, \ldots, \mathcal{P}_n$ AST

- Problems without annotations are AST.
- Soundness: pushes AST **upwards** to the root
- Completeness: a failing leaf refutes AST



The ADP Framework for AST: Proof Trees

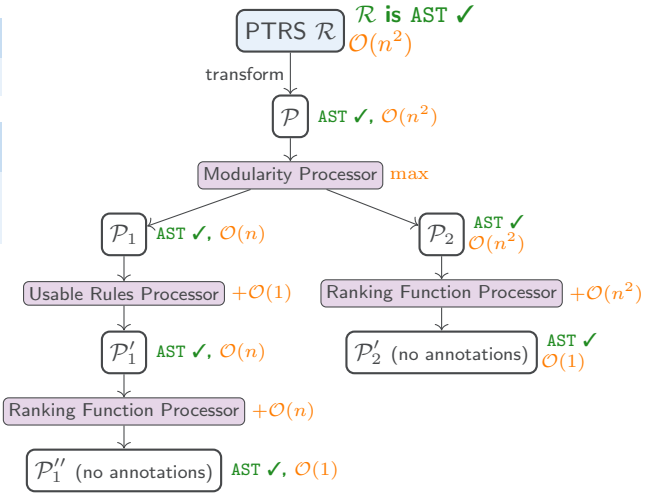
ADP Problem $\mathcal{P}$

A set of annotated rules

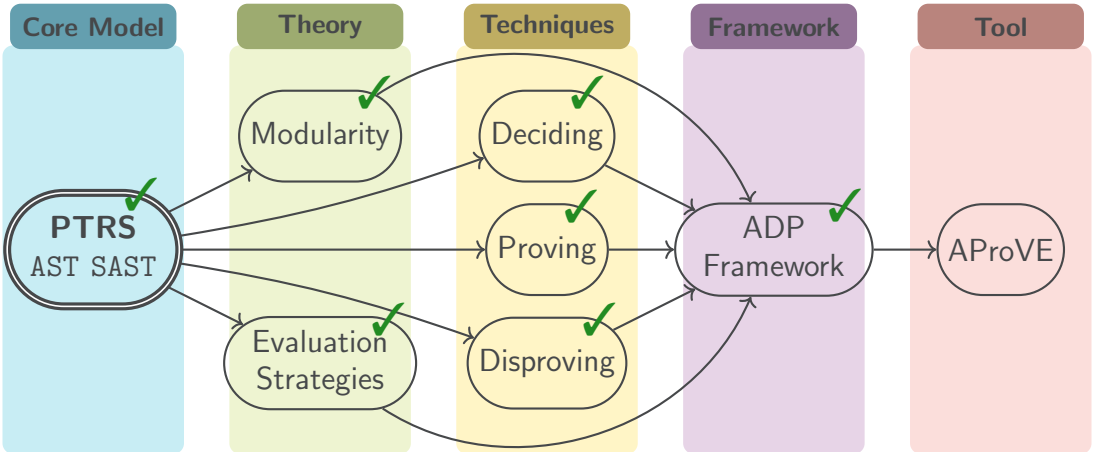
Processor $\mathrm{Proc}(\mathcal{P}) = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$

Sound: $\mathcal{P}_1, \ldots, \mathcal{P}_n$ AST $\implies$ $\mathcal{P}$ AST
Complete: $\mathcal{P}$ AST $\implies$ $\mathcal{P}_1, \ldots, \mathcal{P}_n$ AST

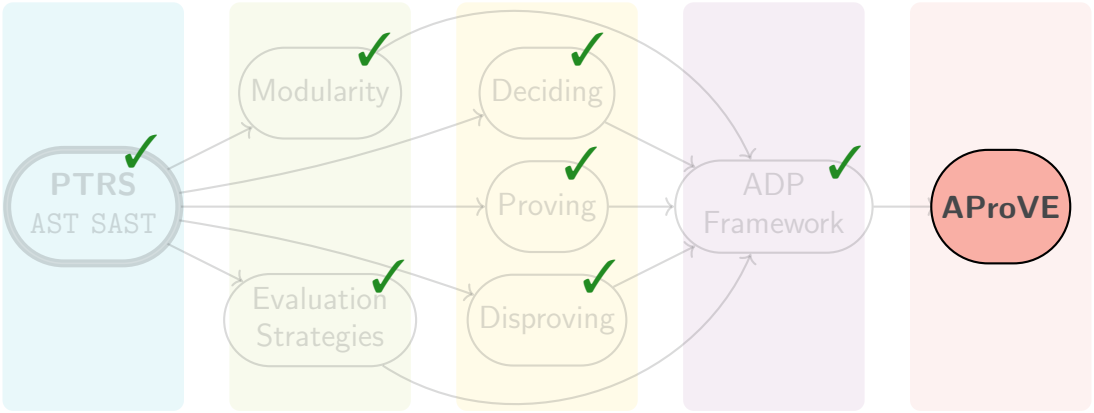
- Problems without annotations are AST.
- Soundness: pushes AST **upwards** to the root
- Completeness: a failing leaf refutes AST
- Also works for runtime!

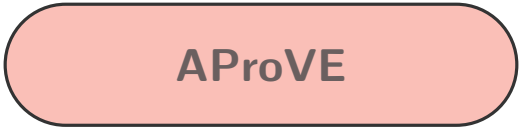


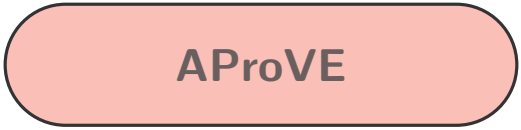
How to Automatically Analyze Probabilistic Programs?



AProVE


PTRS
AST SAST
Modularity
Evaluation Strategies
Deciding
Proving
Disproving
ADP Framework
AProVE

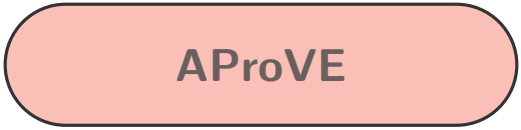

AProVE



Termination and Complexity Competition (termCOMP)[TC26]

AProVE has taken part in the annual competition since its start in 2004.

[TC26] Termination and Complexity Competition 2026, termcomp.github.io/Y2026



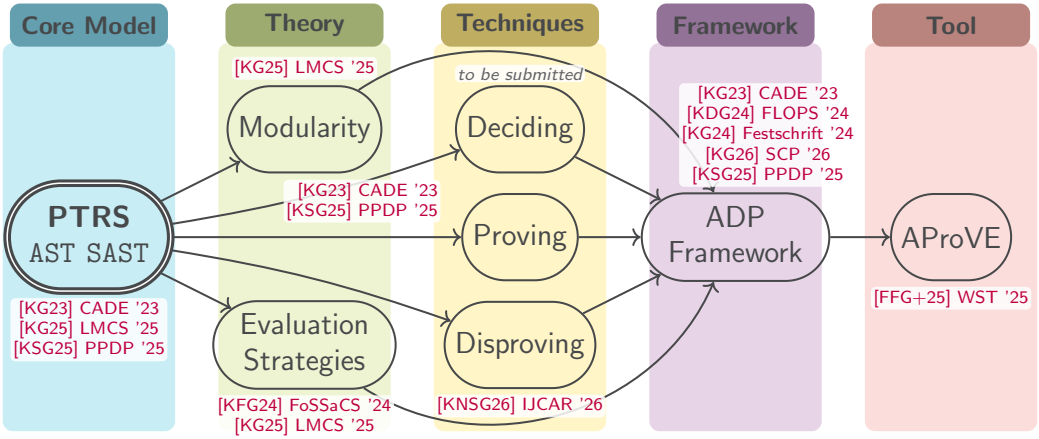
Termination and Complexity Competition (termCOMP)[TC26]

AProVE has taken part in the annual competition since its start in 2004.

Evaluation on the TPDB (175 PTRSs)	Full Evaluation	Innermost Evaluation
AST proved	81 (46.3%)	139 (79.4%)
AST disproved	32 (18.3%)	18 (10.3%)
SAST proved	50 (28.6%)	62 (35.4%)
SAST disproved	62 (35.4%)	9 (5.2%)

[TC26] Termination and Complexity Competition 2026, termcomp.github.io/Y2026

Publications



[KG23] Kassing, Giesl. CADE '23
[KG25] Kassing, Giesl. LMCS '25
[KNSG26] Kassing, Nagel, Schlecht, Giesl. IJCAR '26
[KDG24] Kassing, Dollase, Giesl. FLOPS '24
[KG26] Kassing, Giesl. Sci. Comput. Program. '26
[FFG+25] Frohn, Fuhs, Giesl, Kassing, Lommen. WST '25
[KFG24] Kassing, Frohn, Giesl. FoSSaCS '24
[KG24] Kassing, Giesl. Katoen Festschrift '24
[KSG25] Kassing, Spitzer, Giesl. PPDP '25

Further Publications

Weighted Semantics

[AKGK25] Ahrens, Kassing, Giesl, Katoen. *Weighted Rewriting: Semiring Semantics for Abstract Reduction Systems*. FSCD '25

Relative Rewriting

[KVG24] Kassing, Vartanyan, Giesl. *A Dependency Pair Framework for Relative Termination of Term Rewriting*. IJCAR '24

✓ Solves Problem #106 of the *RTA List of Open Problems*: "Can we use the dependency pair method to prove relative termination?"

(Probabilistic) Confluence

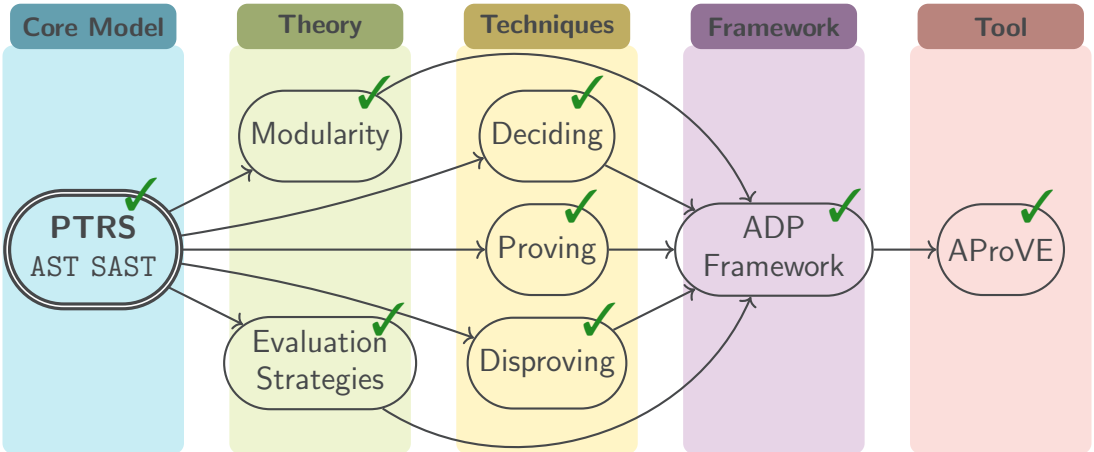
[KS25] Kassing, Sokolowski. *AProVE25: Confluence Analysis in a Termination Tool*. IWC '25

[KSSG26] Kassing, Schüßler, Sokolowski, Giesl. *AProVE'26: Confluence Analysis in a Termination Tool*. IWC '26

Probabilistic Reachability

[KLNG26] Kassing, Leven Rosarius, Nagel, Giesl. *Disproving Reachability in Probabilistic Term Rewriting*. IWC '26

Conclusion


Core Model
Theory
Techniques
Framework
Tool
PTRS
AST SAST
Modularity
Evaluation Strategies
Deciding
Proving
Disproving
ADP Framework
AProVE