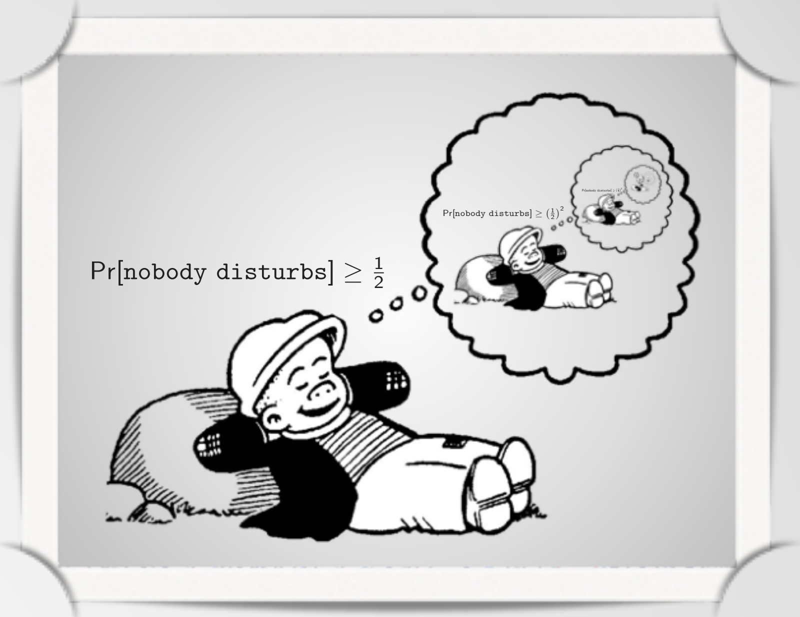
Reasoning about Recursive Probabilistic Programs



Federico Olmedo
Benjamin Kaminski
Joost-Pieter Katoen
Christoph Matheja

RWTH Aachen University, Germany

LICS 2016

July 8th — New York City

Randomization Leads to Intricate Behaviours

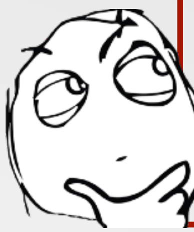
Randomization Leads to Intricate Behaviours

$P \triangleright \quad \{\texttt{skip}\}\ [1/2]\ \{\texttt{call}\ P\}$

Randomization Leads to Intricate Behaviours

$P \triangleright \; \{\texttt{skip}\} \; [1/2] \; \{\texttt{call}\ P\}$

Probability of Termination: 1

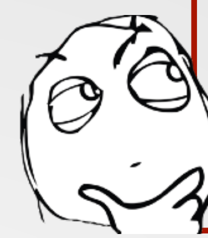

www.walldevil.com/

It terminates with probability 1, even though it admits arbitrarily long executions**!**

Randomization Leads to Intricate Behaviours

$P \triangleright \quad \{\texttt{skip}\} \ [1/2] \ \{\texttt{call } P\}$

Probability of Termination: 1


www.walldevil.com/

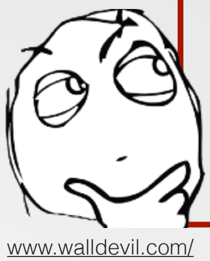
It terminates with probability 1, even though it admits arbitrarily long executions**!**

$P \triangleright \quad \{\texttt{skip}\} \ [1/2] \ \{\texttt{call } P; \ \texttt{call } P\}$

Randomization Leads to Intricate Behaviours

$P \triangleright \quad \{\texttt{skip}\}\ [1/2]\ \{\texttt{call}\ P\}$

Probability of Termination: 1



It terminates with probability 1, even though it admits arbitrarily long executions!

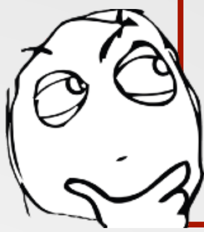
$P \triangleright \quad \{\texttt{skip}\}\ [1/2]\ \{\texttt{call}\ P;\ \texttt{call}\ P\}$

Probability of Termination: 1

Randomization Leads to Intricate Behaviours

$P \triangleright \quad \{\texttt{skip}\}\ [1/2]\ \{\texttt{call}\ P\}$

Probability of Termination: 1



www.walldevil.com/

It terminates with probability 1, even though it admits arbitrarily long executions!

$P \triangleright \quad \{\texttt{skip}\}\ [1/2]\ \{\texttt{call}\ P;\ \texttt{call}\ P\}$

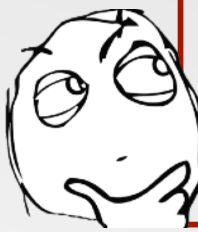
Probability of Termination: 1

Runtime:

Randomization Leads to Intricate Behaviours

$P \;\triangleright\; \{\texttt{skip}\}\ [1/2]\ \{\texttt{call}\ P\}$

Probability of Termination: 1



www.walldevil.com/

It terminates with probability 1, even though it admits arbitrarily long executions**!**

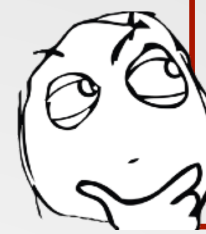
$P \;\triangleright\; \{\texttt{skip}\}\ [1/2]\ \{\texttt{call}\ P;\ \texttt{call}\ P\}$

Probability of Termination: 1
Runtime: 1 sec.

Randomization Leads to Intricate Behaviours

$P \triangleright \quad \{\texttt{skip}\}\ [{}^1\!/\!{}_2]\ \{\texttt{call}\ P\}$

Probability of Termination: 1



www.walldevil.com/

It terminates with probability 1, even though it admits arbitrarily long executions**!**

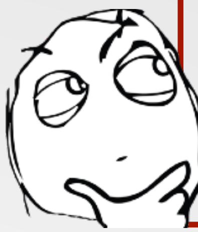
$P \triangleright \quad \{\texttt{skip}\}\ [{}^1\!/\!{}_2]\ \{\texttt{call}\ P;\ \texttt{call}\ P\}$

Probability of Termination: 1
Runtime: 1 min.

Randomization Leads to Intricate Behaviours

$P \;\triangleright\; \{\texttt{skip}\}\ [1/2]\ \{\texttt{call}\ P\}$

Probability of Termination: 1



www.walldevil.com/

It terminates with probability 1, even though it admits arbitrarily long executions!

$P \;\triangleright\; \{\texttt{skip}\}\ [1/2]\ \{\texttt{call}\ P;\ \texttt{call}\ P\}$

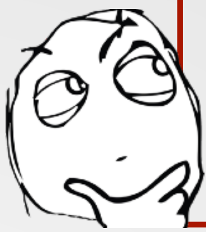
Probability of Termination: 1

Runtime: 1 hour

Randomization Leads to Intricate Behaviours

$P \triangleright \{\texttt{skip}\}\ [1/2]\ \{\texttt{call}\ P\}$

Probability of Termination: 1



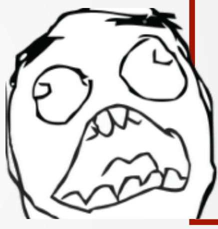
www.walldevil.com/

It terminates with probability 1, even though it admits arbitrarily long executions**!**

$P \triangleright \{\texttt{skip}\}\ [1/2]\ \{\texttt{call}\ P;\ \texttt{call}\ P\}$

Probability of Termination: 1

Runtime: ∞



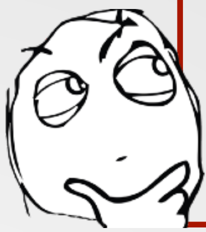
www.ragefaces.memesoftware.com/

It terminates with probability 1, but reaching termination takes (in average) infinite time**!**

Randomization Leads to Intricate Behaviours

$P \;\triangleright\; \{\texttt{skip}\}\ [1/2]\ \{\texttt{call}\ P\}$

Probability of Termination: 1

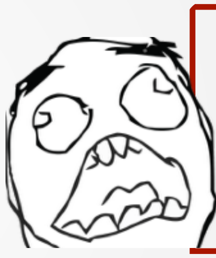

www.walldevil.com/

It terminates with probability 1, even though it admits arbitrarily long executions**!**

$P \;\triangleright\; \{\texttt{skip}\}\ [1/2]\ \{\texttt{call}\ P;\ \texttt{call}\ P\}$

Probability of Termination: 1

Runtime: ∞


www.ragefaces.memesoftware.com/

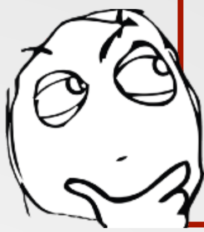
It terminates with probability 1, but reaching termination takes (in average) infinite time**!**

$P \;\triangleright\; \{\texttt{skip}\}\ [1/2]\ \{\texttt{call}\ P;\ \texttt{call}\ P;\ \texttt{call}\ P\}$

Randomization Leads to Intricate Behaviours

$P \triangleright \quad \{\texttt{skip}\}\ [1/2]\ \{\texttt{call}\ P\}$

Probability of Termination: 1



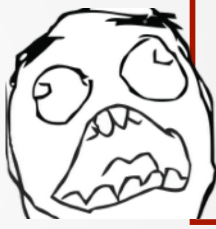
www.walldevil.com/

It terminates with probability 1, even though it admits arbitrarily long executions**!**

$P \triangleright \quad \{\texttt{skip}\}\ [1/2]\ \{\texttt{call}\ P;\ \texttt{call}\ P\}$

Probability of Termination: 1

Runtime: ∞

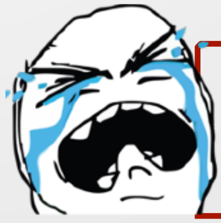


www.ragefaces.memesoftware.com/

It terminates with probability 1, but reaching termination takes (in average) infinite time**!**

$P \triangleright \quad \{\texttt{skip}\}\ [1/2]\ \{\texttt{call}\ P;\ \texttt{call}\ P;\ \texttt{call}\ P\}$

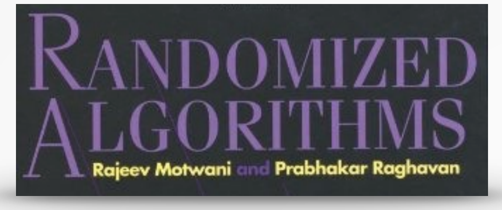
Probability of Termination: $\frac{\sqrt{5}-1}{2}$



www.gagfire.com/

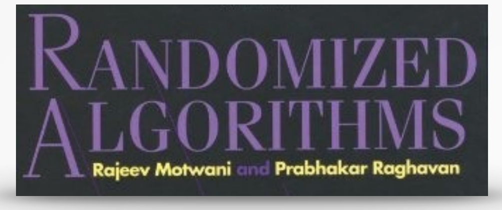
It terminates with an irrational probability**!**

Randomized (Recursive) Algorithms are Natural and Widespread



"For many applications, a randomized algorithm is **the simplest** algorithm available, or **the fastest**, or **both**." [Motwani & Raghavan]

Randomized (Recursive) Algorithms are Natural and Widespread



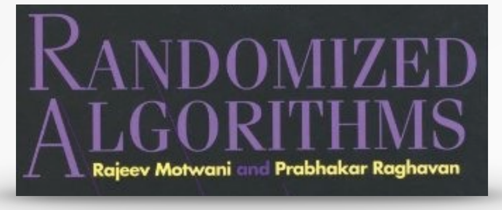
"For many applications, a randomized algorithm is **the simplest** algorithm available, or **the fastest**, or **both**." [Motwani & Raghavan]

Quicksort:

$$
\begin{aligned}
&\mathsf{QS}(A) \triangleq \\
&\quad \mathtt{if}\ (|A| \leq 1)\ \mathtt{then}\ \mathtt{return}\,(A); \\
&\quad i := \lfloor {}^{|A|}\!/_{2} \rfloor; \\
&\quad A_{<} := \{a' \in A \mid a' < A[i]\}; \\
&\quad A_{>} := \{a' \in A \mid a' > A[i]\}; \\
&\quad \mathtt{return}\ \big(\mathsf{QS}(A_{<}) \mathbin{+\!\!+} A[i] \mathbin{+\!\!+} \mathsf{QS}(A_{>})\big)
\end{aligned}
$$

Deterministic version: $O(n^2)$ comparisons

Randomized (Recursive) Algorithms are Natural and Widespread



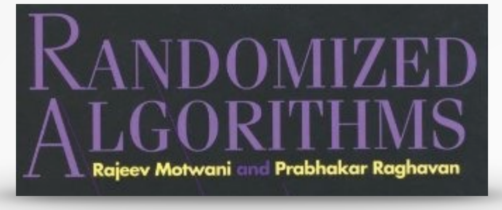
"For many applications, a randomized algorithm is **the simplest** algorithm available, or **the fastest**, or **both**." [Motwani & Raghavan]

Randomized Quicksort:

$$
\begin{aligned}
&\mathsf{rQS}(A) \triangleq \\
&\quad \mathtt{if}\,(|A| \leq 1)\,\mathtt{then}\ \mathtt{return}\,(A); \\
&\quad i := \mathsf{rand}[1 \ldots |A|]; \\
&\quad A_< := \{a' \in A \mid a' < A[i]\}; \\
&\quad A_> := \{a' \in A \mid a' > A[i]\}; \\
&\quad \mathtt{return}\ \big(\mathsf{QS}(A_<) \mathbin{+\!\!+} A[i] \mathbin{+\!\!+} \mathsf{QS}(A_>)\big)
\end{aligned}
$$

Randomized version: *O(n log(n))* comparisons

Randomized (Recursive) Algorithms are Natural and Widespread



"For many applications, a randomized algorithm is **the simplest** algorithm available, or **the fastest**, or **both**." [Motwani & Raghavan]

Randomized Quicksort:

$$
\begin{aligned}
&\mathrm{rQS}(A) \triangleq \\
&\qquad \texttt{if}\,(|A| \leq 1)\,\texttt{then return}\,(A); \\
&\qquad i := \mathrm{rand}[1 \ldots |A|]; \\
&\qquad A_{<} := \{a' \in A \mid a' < A[i]\}; \\
&\qquad A_{>} := \{a' \in A \mid a' > A[i]\}; \\
&\qquad \texttt{return}\,\big(\mathrm{QS}(A_{<}) \mathbin{+\!\!+} A[i] \mathbin{+\!\!+} \mathrm{QS}(A_{>})\big)
\end{aligned}
$$

Randomized version: *O(n log(n))* comparisons

Sample Randomized Recursive Algorithms:

- Quicksort
- Median finding
- Binary search
- Simple path of length *k*
- Euclidean matching
- …..

Current Analysis Approaches are Not Satisfactory

Current Analysis Approaches:

- Mathematical ad-hoc reasoning (on involved random variables)
- Probabilistic recurrence relations
- Dedicated techniques for D&C algorithms

Current Analysis Approaches are Not Satisfactory

Current Analysis Approaches:

- Mathematical ad-hoc reasoning (… n variables)
- Probabilistic re…
- Ded… D&C algorithms

- Cover only a fragment of the proof argument
- Non-trivial claims are taken for granted

Current Analysis Approaches are Not Satisfactory

Current Analysis Approaches:

- Mathematical ad-hoc reasoning (… n variables)
- Probabilistic re…
- Dec… D&C algorithms

- Cover only a fragment of the proof argument
- Non-trivial claims are taken for granted

Our Approach:

Formal verification

- using only first principles
- directly from the program code

Our Contribution

DEDUCTIVE VERIFICATION OF RANDOMIZED RECURSIVE ALGORITHMS

- Two calculi à la weakest pre-condition:
  - For reasoning about program outcomes, e.g. $\texttt{Pr}\left[x = x^{opt}\right] \geq 0.9$
  - For reasoning about program expected runtimes, e.g. $\mathsf{ert} \leq x + y$
- Soundness of the calculi w.r.t. an operational semantics
- Application: probabilistic binary search

For Program Outcomes

[Kozen '81]

probabilisitic program

$$\mathsf{wp}[c]\colon (\mathbb{S} \to [0,\, 1]) \to (\mathbb{S} \to [0,\, 1])$$

quantitative post-condition

quantitative pre-condition

$\mathsf{wp}[c](\mathbb{1}_Q)$: **probability** that c establishes post-condition Q.

Calculi — Basics

For Program Outcomes
[Kozen '81]

probabilisitic program

$$\mathrm{wp}[c]\colon (\mathbb{S} \to [0,\ 1]) \to (\mathbb{S} \to [0,\ 1])$$

quantitative post-condition

quantitative pre-condition

$\mathrm{wp}[c](\mathbb{1}_Q)$: **probability** that *c* establishes post-condition *Q*.

For Program Expected Runtimes
[ESOP '16]

$$\mathrm{ert}\,[c] \;:\; (\mathbb{S} \to \mathbb{R}^{\infty}_{\geq 0}) \to (\mathbb{S} \to \mathbb{R}^{\infty}_{\geq 0})$$

runtime of the computation following *c*

runtime of *c*, **plus** the computation following *c*

$\mathrm{ert}\,[c](\mathbf{0})$: **expected runtime** of c.

Calculi — Basics

For Program Outcomes

[Kozen '81]

probabilisitic program

$$\mathrm{wp}[c] : (\mathbb{S} \to [0,\ 1]) \to (\mathbb{S} \to [0,\ 1])$$

quantitative post-condition

quantitative pre-condition

$\mathrm{wp}[c](\mathbb{1}_Q)$: **probability** that c establishes post-condition Q.

$$\mathrm{wp}\big[\{c_1\}\ [p]\ \{c_2\}\big](\mathbb{1}_Q) \ = \ p \cdot \mathrm{wp}[c_1](\mathbb{1}_Q) + (1-p) \cdot \mathrm{wp}[c_2](\mathbb{1}_Q)$$

For Program Expected Runtimes

[ESOP '16]

$$\mathrm{ert}\,[c] \ : \ \left(\mathbb{S} \to \mathbb{R}_{\geq 0}^{\infty}\right) \to \left(\mathbb{S} \to \mathbb{R}_{\geq 0}^{\infty}\right)$$

runtime of the computation following c

runtime of c, **plus** the computation following c

$\mathrm{ert}\,[c](\mathbf{0})$: **expected runtime** of c.

$$\mathrm{ert}\,[\{c_1\}\ [p]\ \{c_2\}](t) \ = \ \mathbf{1} + p \cdot \mathrm{ert}\,[c_1](t) + (1-p) \cdot \mathrm{ert}\,[c_2](t)$$

Calculi — Proof Rules for Recursive Procedures

For procedure calls, we intuitively have

$$\text{“wp}[\texttt{call } P](\mathbb{1}_Q) \;=\; \text{wp}[\mathit{body}(P)](\mathbb{1}_Q)\text{”} \qquad\qquad \text{“ert}\,[\texttt{call } P](t) \;=\; \mathbf{1} + \text{ert}\,[\mathit{body}(P)](t)\text{”}$$

but formal definitions require (higher order) fixed points.

Calculi — Proof Rules for Recursive Procedures

For procedure calls, we intuitively have

$$\text{“wp}[\texttt{call } P](\mathbb{1}_Q) \;=\; \text{wp}[\mathit{body}(P)](\mathbb{1}_Q)\text{”} \qquad \text{“ert}\,[\texttt{call } P](t) \;=\; \mathbf{1} + \text{ert}\,[\mathit{body}(P)](t)\text{”}$$

but formal definitions require (higher order) fixed points.

Proof Rules for Procedure Calls

> "Prove the desired specification for the procedure's body assuming it already holds for the recursive calls in it."

- For upper bounds

$$\frac{\text{wp}[\texttt{call } P](\mathbb{1}_Q) \leq u \quad \Vdash \quad \text{wp}[\mathit{body}(P)](\mathbb{1}_Q) \leq u}{\text{wp}[\texttt{call } P](\mathbb{1}_Q) \leq u}$$

- For lower bounds

$$\frac{\begin{array}{c} l_0 = 0 \\ l_n \leq \text{wp}[\texttt{call } P](\mathbb{1}_Q) \quad \Vdash \quad l_{n+1} \leq \text{wp}[\mathit{body}(P)](\mathbb{1}_Q) \end{array}}{\sup_n l_n \leq \text{wp}[\texttt{call } P](\mathbb{1}_Q)}$$

  - Dual rule for upper bounds is also sound

Calculi — Proof Rules for Recursive Procedures

For procedure calls, we intuitively have

$$\text{“}\mathsf{wp}[\texttt{call } P](\mathbb{1}_Q) \;=\; \mathsf{wp}[\mathit{body}(P)](\mathbb{1}_Q)\text{”} \qquad\qquad \text{“}\mathsf{ert}\,[\texttt{call } P](t) \;=\; \mathbf{1} + \mathsf{ert}\,[\mathit{body}(P)](t)\text{”}$$

but formal definitions require (higher order) fixed points.

Proof Rules for Procedure Calls

> “Prove the desired specification for the procedure’s body assuming it already holds for the recursive calls in it.”

- For upper bounds

$$\frac{\mathsf{wp}[\texttt{call } P](\mathbb{1}_Q) \leq u \quad \Vdash \quad \mathsf{wp}[\mathit{body}(P)](\mathbb{1}_Q) \leq u}{\mathsf{wp}[\texttt{call } P](\mathbb{1}_Q) \leq u}$$

$$\frac{\mathsf{ert}\,[\texttt{call } P](t) \leq u + \mathbf{1} \quad \Vdash \quad \mathsf{ert}\,[\mathit{body}(P)](t) \leq u}{\mathsf{ert}\,[\texttt{call } P](t) \leq u + \mathbf{1}}$$

- For lower bounds

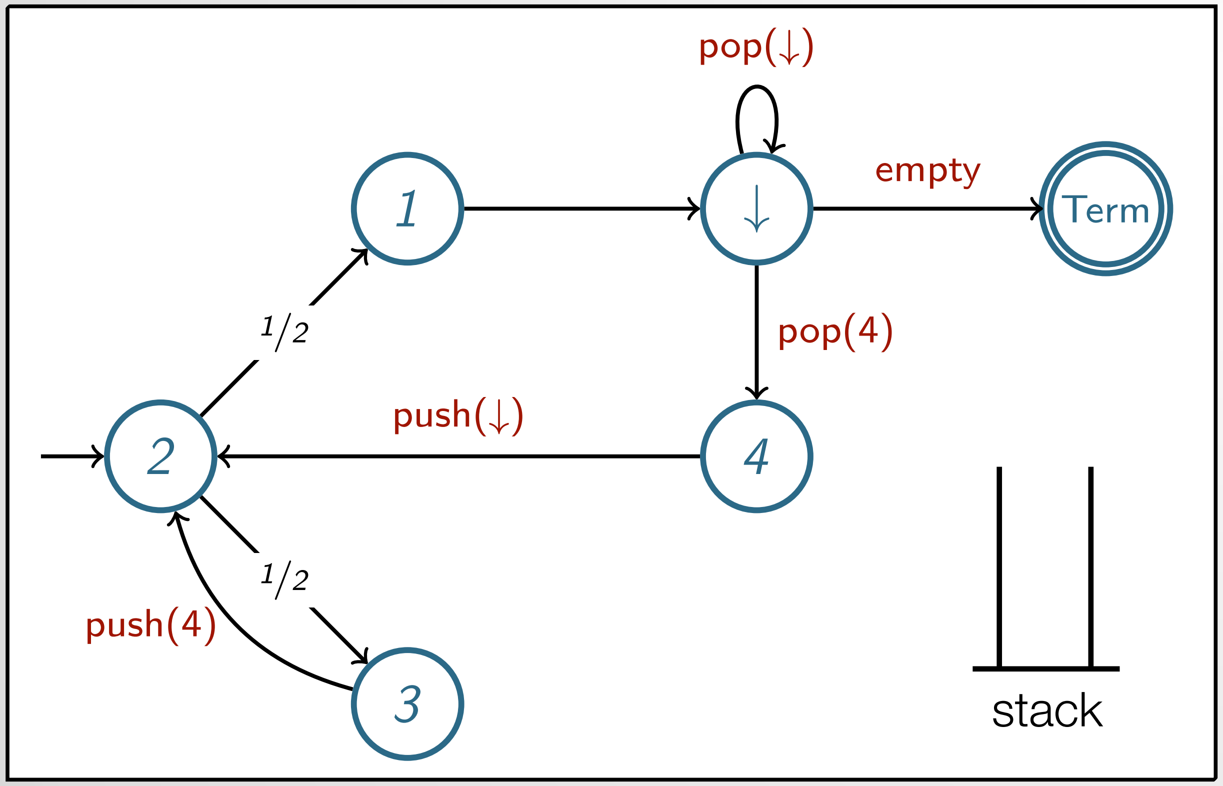
$$\frac{l_0 = 0 \qquad l_n \leq \mathsf{wp}[\texttt{call } P](\mathbb{1}_Q) \quad \Vdash \quad l_{n+1} \leq \mathsf{wp}[\mathit{body}(P)](\mathbb{1}_Q)}{\sup_n l_n \leq \mathsf{wp}[\texttt{call } P](\mathbb{1}_Q)}$$

  - Dual rule for upper bounds is also sound

Sample Program

$$P \triangleright \ \{\texttt{skip}^1\}\ [1/2]^2\ \{\texttt{call}\ P^3;\ \texttt{call}\ P^4\}$$

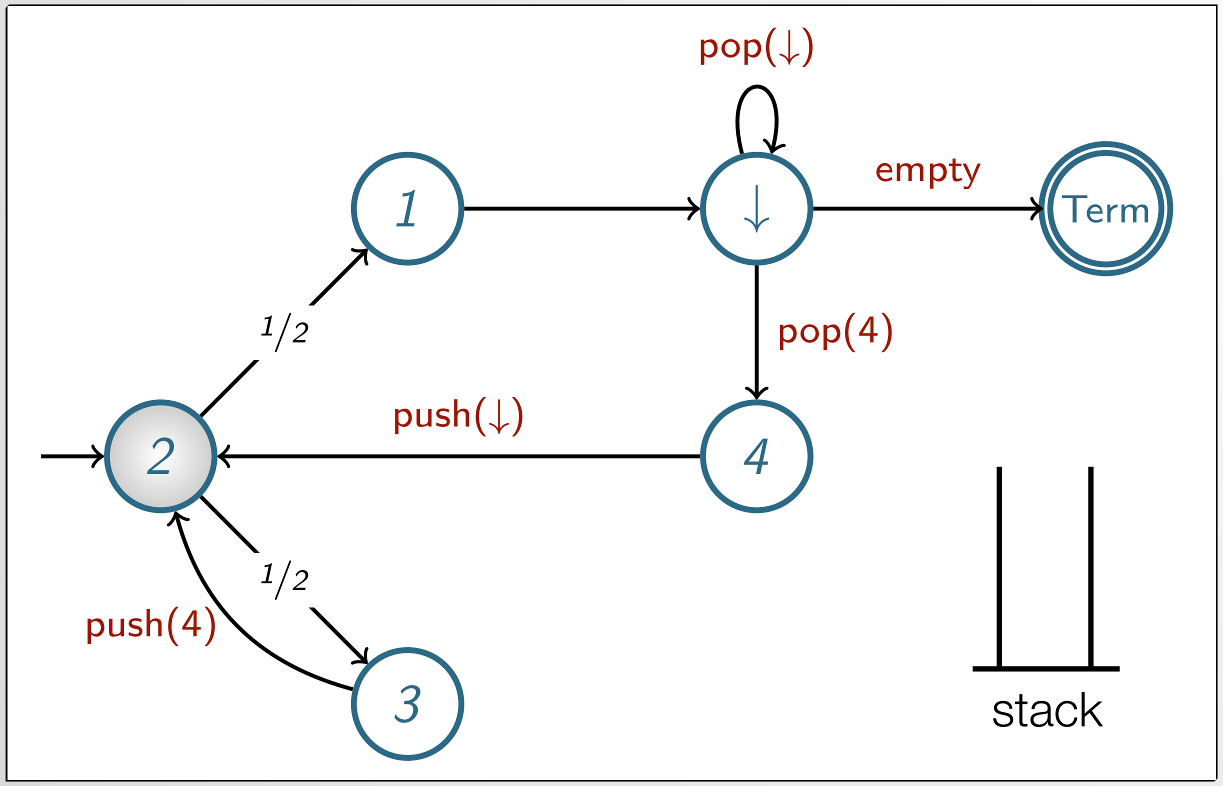
Associated Pushdown Markov Chain



Sample Program

$$P \triangleright \ \{\texttt{skip}^{1}\}\ [1/2]^{2}\ \{\texttt{call}\ P^{3};\ \texttt{call}\ P^{4}\}$$

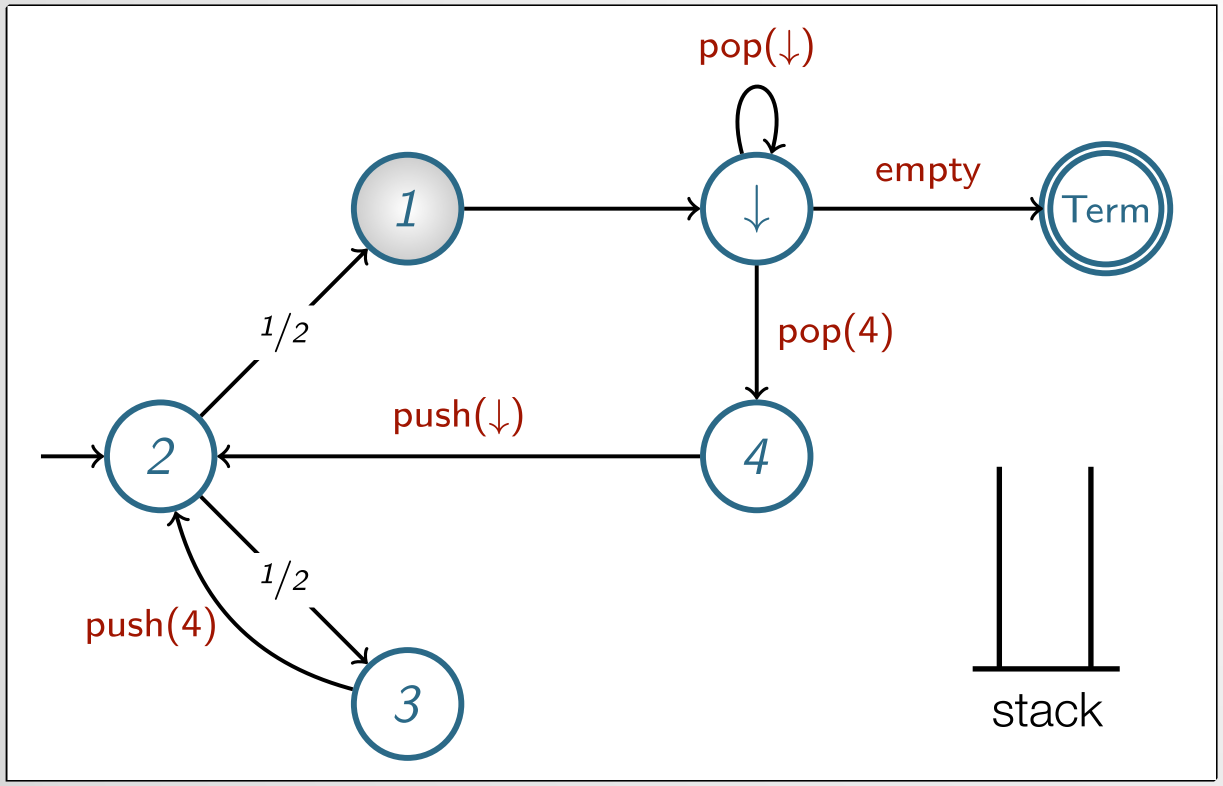
Associated Pushdown Markov Chain



Sample Program

$$P \triangleright \quad \{\texttt{skip}^{1}\}\ [1/2]^{2}\ \{\texttt{call}\ P^{3};\ \texttt{call}\ P^{4}\}$$

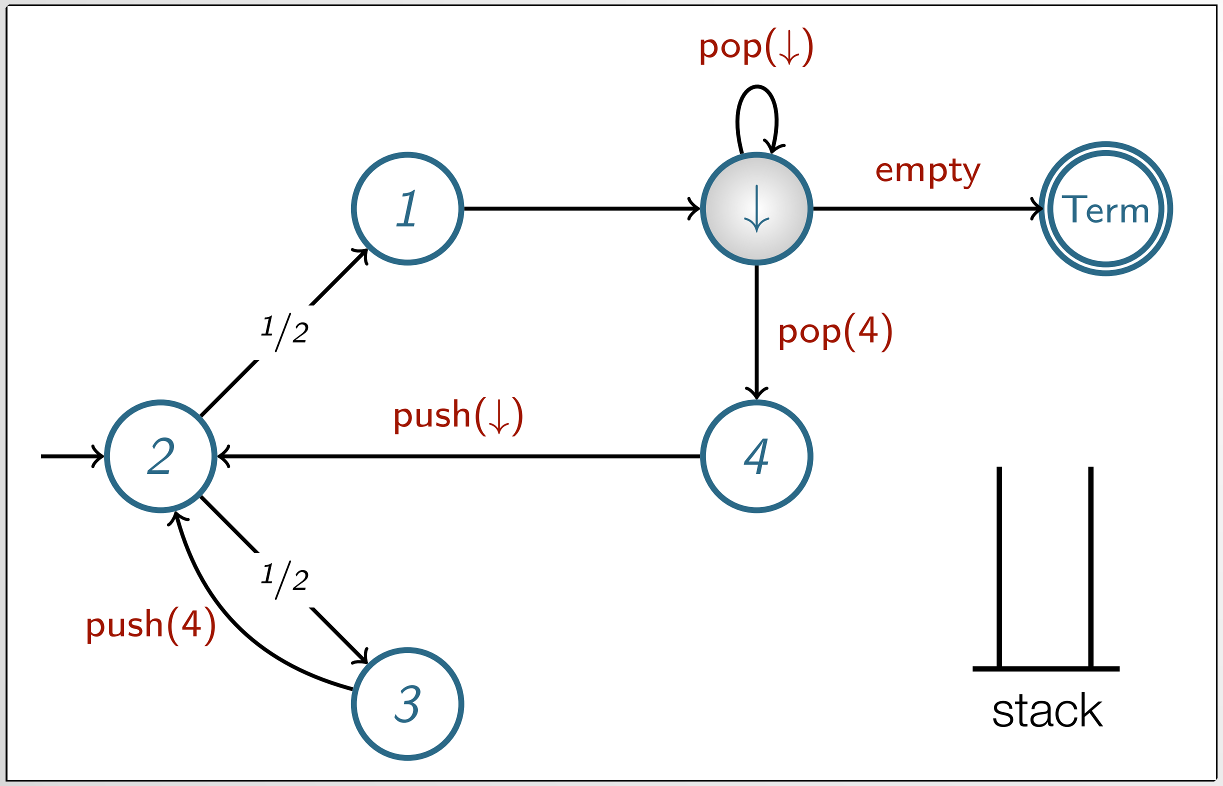
Associated Pushdown Markov Chain



Sample Program

$$P \triangleright \{\texttt{skip}^{1}\}\ [1/2]^{2}\ \{\texttt{call}\ P^{3};\ \texttt{call}\ P^{4}\}$$

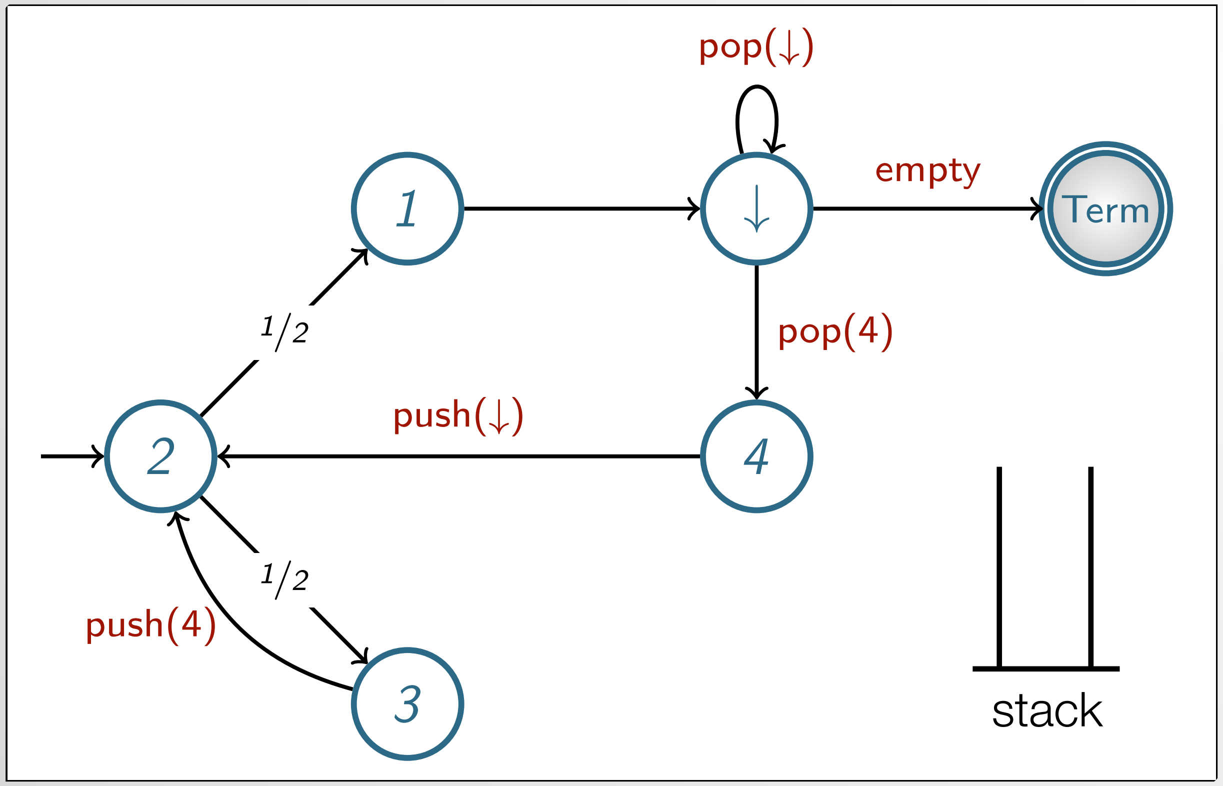
Associated Pushdown Markov Chain



Sample Program

$P \triangleright \quad \{\texttt{skip}^1\} \; [1/2]^2 \; \{\texttt{call } P^3;\ \texttt{call } P^4\}$

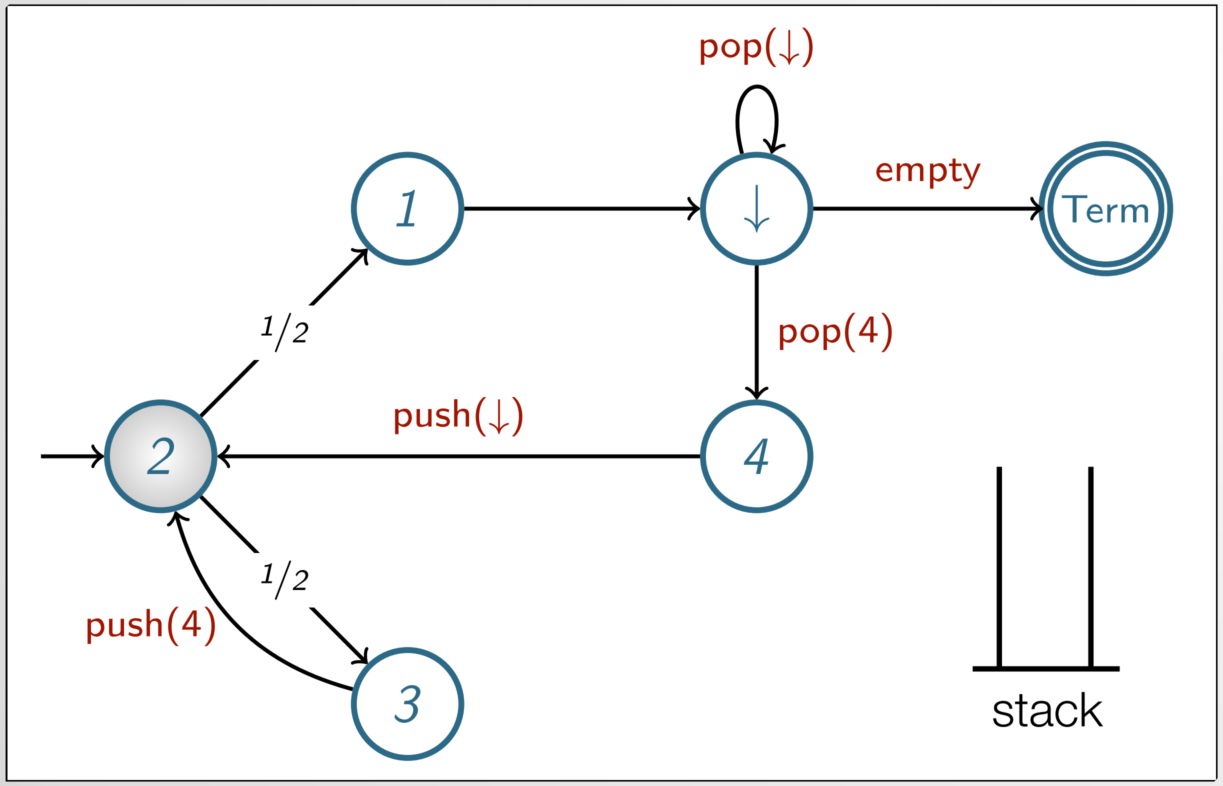
Associated Pushdown Markov Chain



Sample Program

$$P \triangleright \quad \{\texttt{skip}^{1}\}\ [1/2]^{2}\ \{\texttt{call}\ P^{3};\ \texttt{call}\ P^{4}\}$$

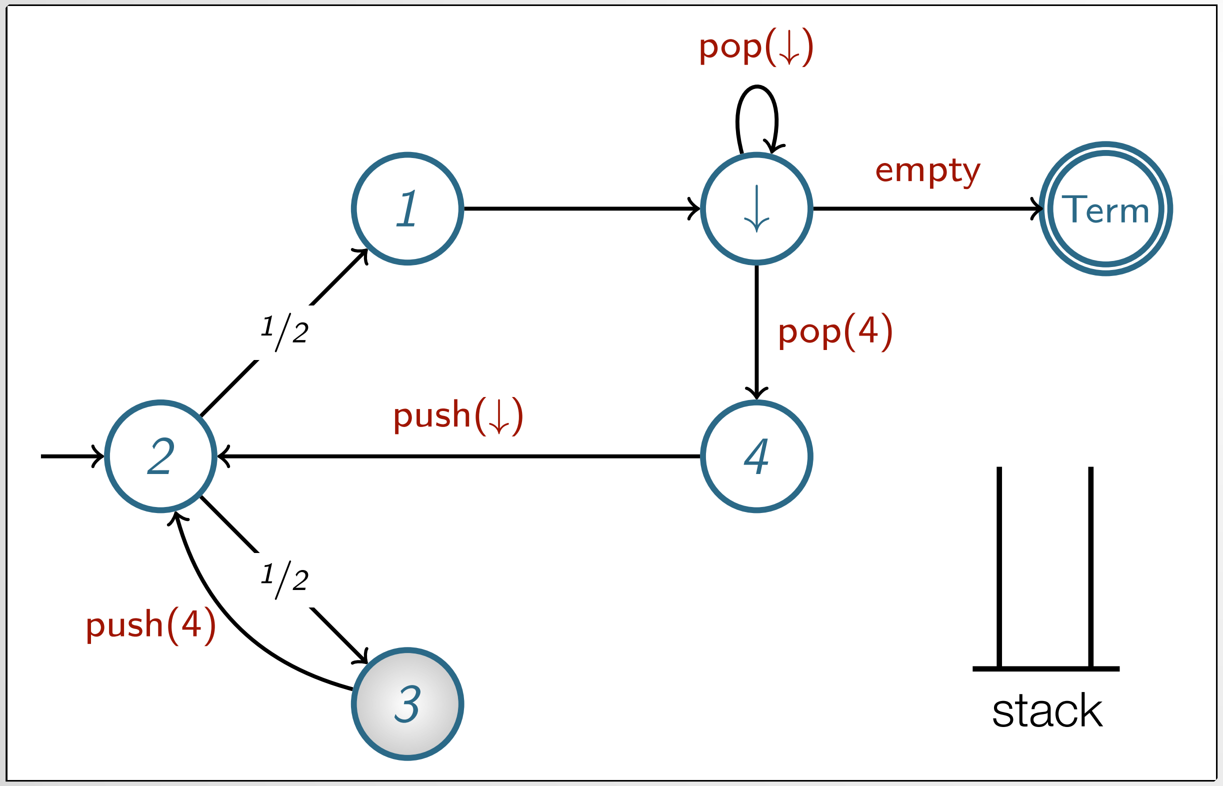
Associated Pushdown Markov Chain



Sample Program

$$P \triangleright \quad \{\texttt{skip}^1\} \; [1/2]^2 \; \{\texttt{call } P^3;\ \texttt{call } P^4\}$$

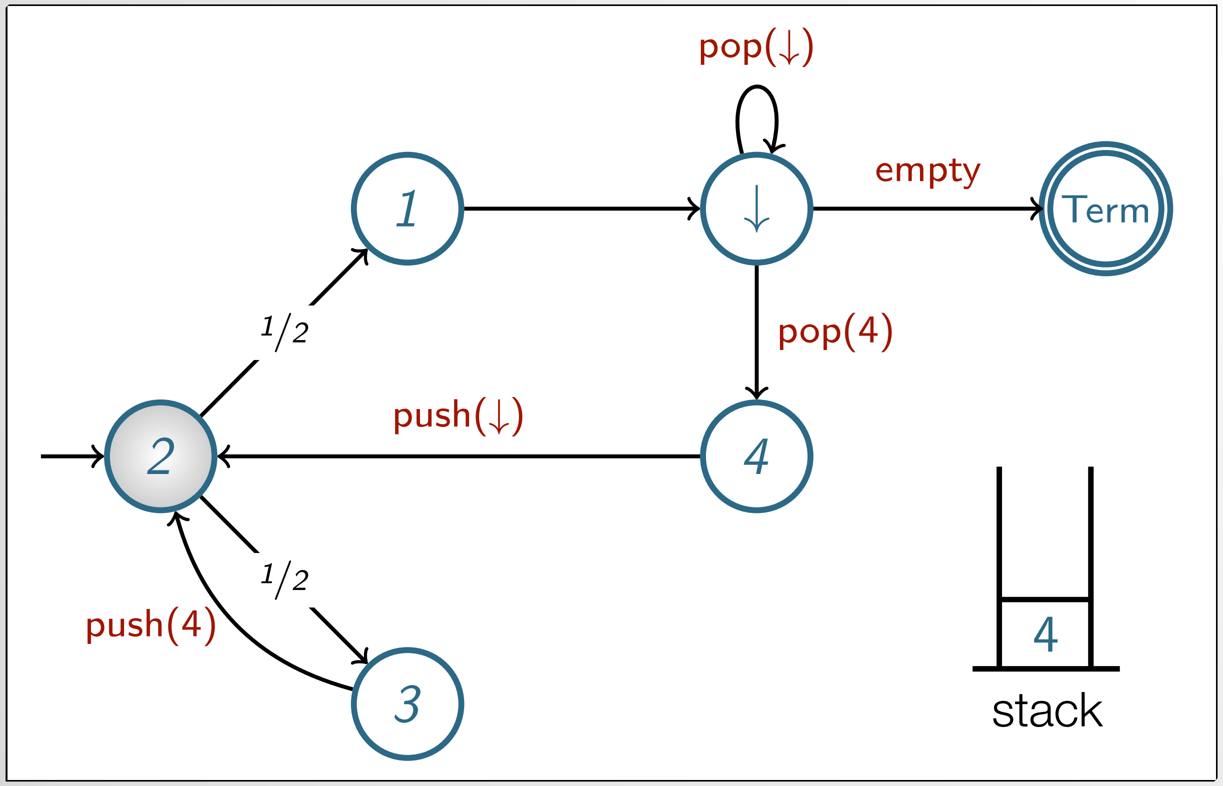
Associated Pushdown Markov Chain



Sample Program

$$P \triangleright \quad \{\texttt{skip}^1\}\ [1/2]^2\ \{\texttt{call}\ P^3;\ \texttt{call}\ P^4\}$$

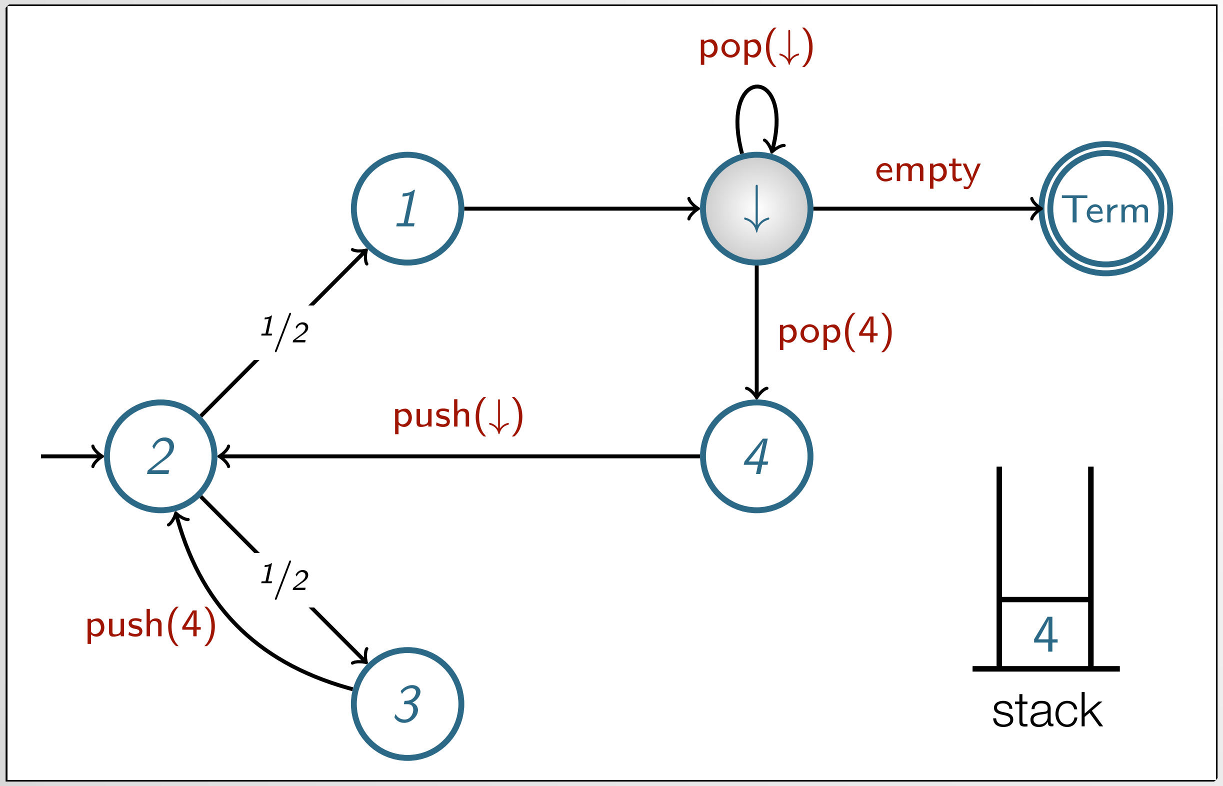
Associated Pushdown Markov Chain



Sample Program

$$P \triangleright \; \{\texttt{skip}^{1}\} \; [1/2]^{2} \; \{\texttt{call } P^{3};\ \texttt{call } P^{4}\}$$

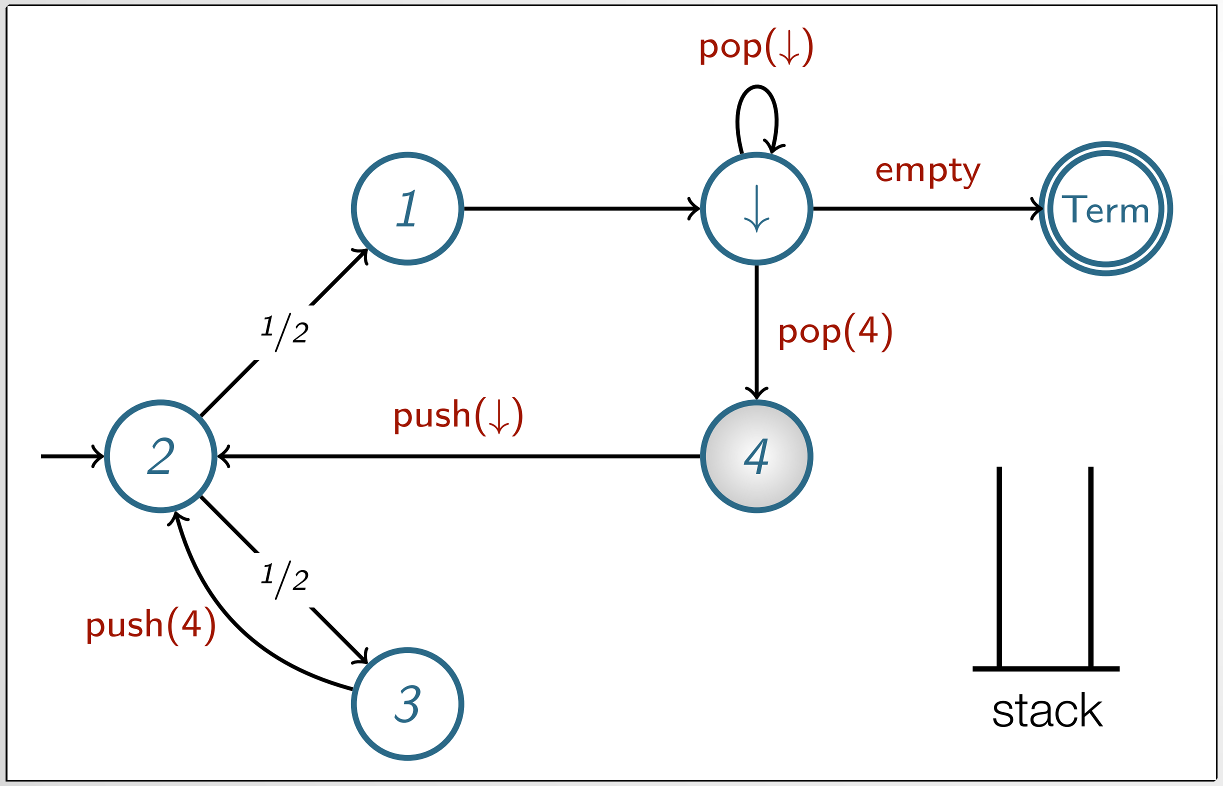
Associated Pushdown Markov Chain



Sample Program

$$P \triangleright \quad \{\texttt{skip}^{1}\}\ [1/2]^{2}\ \{\texttt{call}\ P^{3};\ \texttt{call}\ P^{4}\}$$

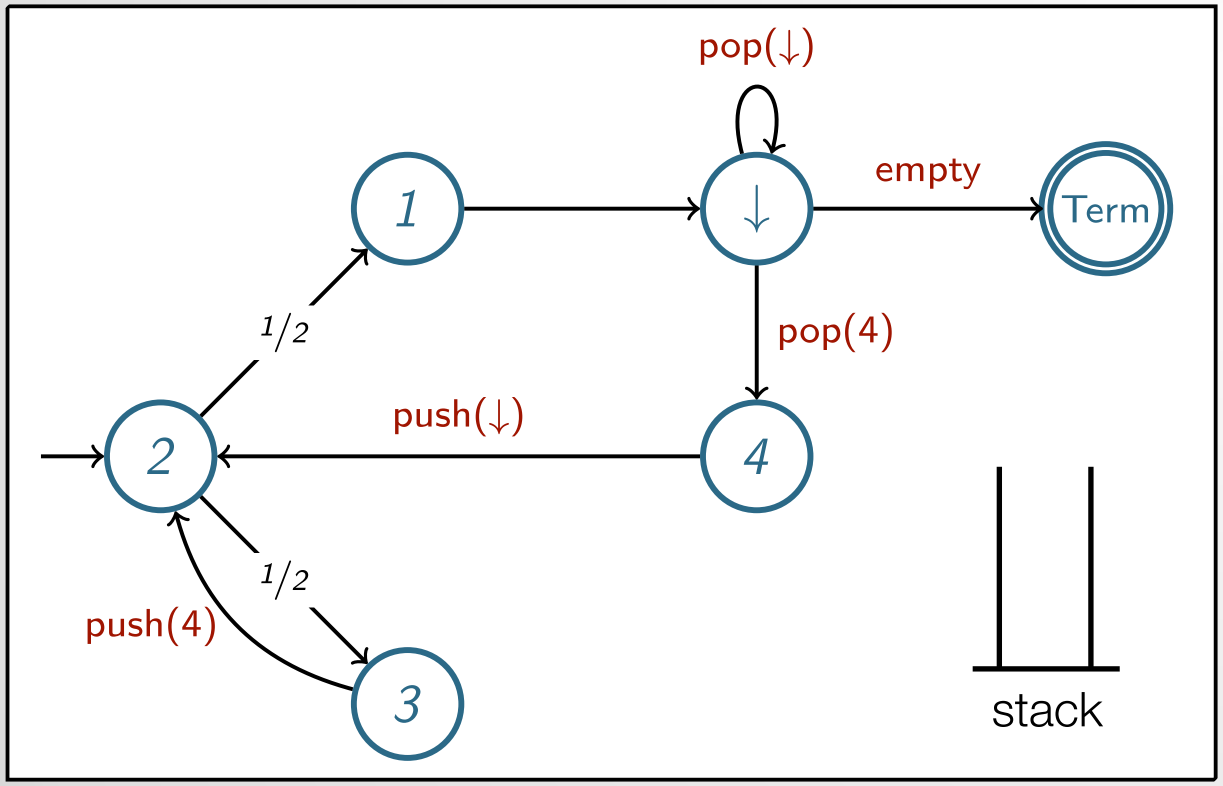
Associated Pushdown Markov Chain



Sample Program

$$P \triangleright \quad \{\texttt{skip}^1\}\ [1/2]^2\ \{\texttt{call}\ P^3;\ \texttt{call}\ P^4\}$$

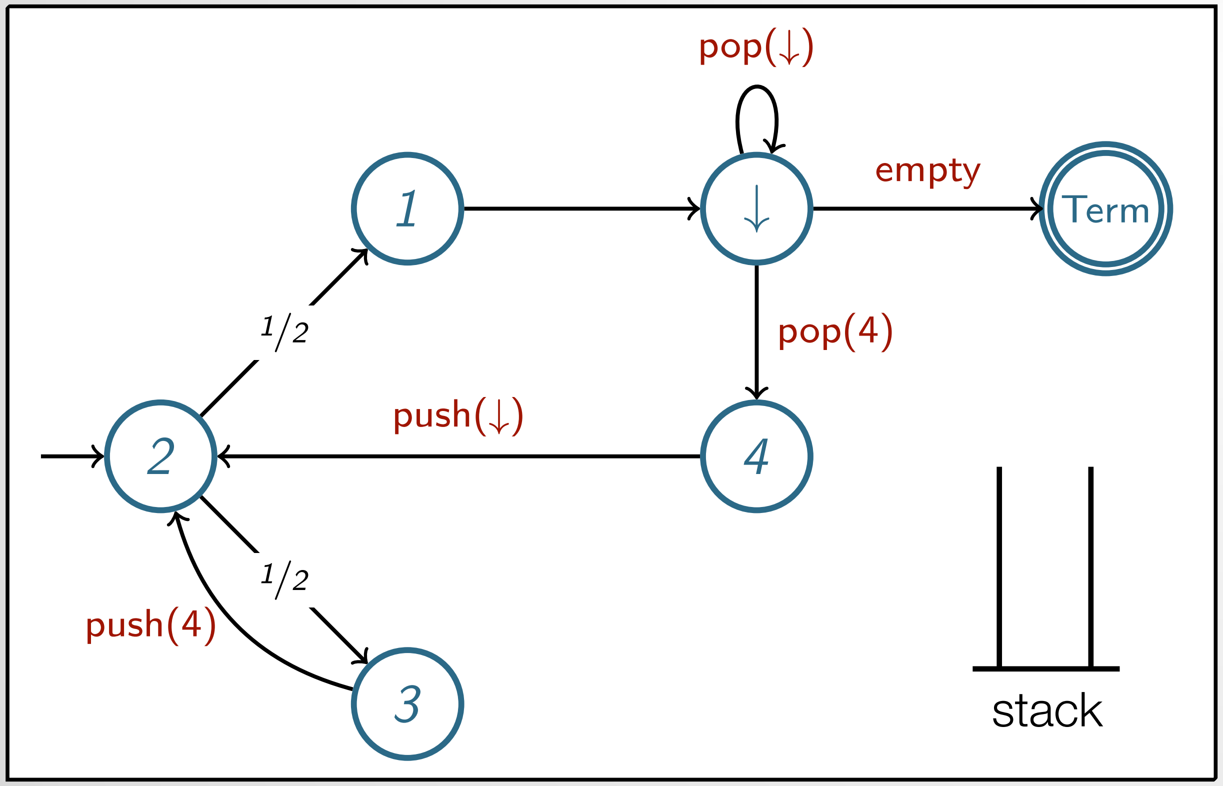
Associated Pushdown Markov Chain



Sample Program

$$P \triangleright \quad \{\texttt{skip}^{1}\} \; [1/2]^{2} \; \{\texttt{call } P^{3};\ \texttt{call } P^{4}\}$$

Associated Pushdown Markov Chain



SOUNDNESS RESULT

$$\mathrm{wp}[\texttt{call } P](\mathbb{1}_{\mathrm{true}}) \;=\; \Pr(\Diamond \mathrm{Term})$$

Case Study: Probabilistic Binary Search

Input: sorted array $a[left\ldots right]$, value val to search in the array

Output: index of the array containing val (if any)

$$
\begin{aligned}
&\mathrm{PBS} \triangleq\\
&\quad pivot := \mathtt{rand}[left\ldots right];\\
&\quad \mathtt{if}\ (left < right)\\
&\qquad \mathtt{if}\ (a[pivot] < val)\\
&\qquad\quad left := \min\{pivot+1, right\};\\
&\qquad\quad \mathtt{call}\ \mathrm{PBS}\\
&\qquad \mathtt{if}\ (a[pivot] > val)\\
&\qquad\quad right := \max\{pivot-1, left\};\\
&\qquad\quad \mathtt{call}\ \mathrm{PBS}
\end{aligned}
$$

Formal Verification of Correctness & Expected Runtime

CORRECTNESS FOR CASE $val \in a[left\ldots right]$

$$\mathbf{1} \cdot \mathbb{1}_{G} \;\leq\; \mathrm{wp}[\mathtt{call}\ \mathrm{PBS}]\left(\mathbb{1}_{a[pivot]=val}\right)$$

G: $left \leq right \;\wedge\; \mathtt{sorted}(a[left\ldots right]) \;\wedge\; val \in a[left\ldots right]$

RUNTIME FOR CASE $val \notin a[left\ldots right]$

$$\mathrm{ert}\,[\mathtt{call}\ \mathrm{PBS}](\mathbf{0}) \;\leq\; \mathbf{4} + \mathbb{1}_{\neg G} \cdot \infty + \mathbb{1}_{G} \cdot \overbrace{(6\,H_n - 2.5)}^{\in\,\Theta(\log n)}$$

G: $left \leq right \;\wedge\; \mathtt{sorted}(a[left\ldots right]) \;\wedge\; val \notin a[left\ldots right]$

H_n: $\sum_{i=1}^{n} 1/i$ with $n = right - left + 1$

What Else is on the Paper?

- Algebraic properties of both transformers $\mathsf{wp}[\cdot]$ and $\mathsf{ert}\,[\cdot]$, e.g.

$$\mathsf{wp}[c](a \cdot f + b \cdot g) \;=\; a \cdot \mathsf{wp}[c](f) \;+\; b \cdot \mathsf{wp}[c](g)$$
$$\mathsf{ert}\,[c](\boldsymbol{k} + t) \;=\; \boldsymbol{k} \;+\; \mathsf{ert}\,[c](t)$$
$$\mathsf{ert}\,[c](t) \;=\; \mathsf{ert}\,[c](\mathbf{0}) \;+\; \mathsf{wp}[c](t)$$

- Relation between finite expected runtime and program termination

$$\mathsf{ert}\,[c](\mathbf{0})(s) < \infty \implies \mathsf{wp}[c](\mathbf{1})(s) = 1$$

- Extension to mutual recursion

Summary

What we have done:

Deductive approach for the formal verification of randomized recursive algorithms

- Two calculi for reasoning about the outcome and runtime of programs
- Set of proof rules for reasoning about recursive programs
- Soundness w.r.t. an operational semantics
- Application: probabilistic binary search

What we would like to do:

- Automate the verification process
- More challenging case studies (e.g. randomized Quicksort)

Summary

What we have done:

Deductive approach for the formal verification of randomized recursive algorithms

- Two calculi for reasoning about the outcome and runtime of programs
- Set of proof rules for reasoning about recursive programs
- Soundness w.r.t. an operational semantics
- Application: probabilistic binary search

What we would like to do:

- Automate the verification process
- More challenging case studies (e.g. randomized Quicksort)

Thanks!

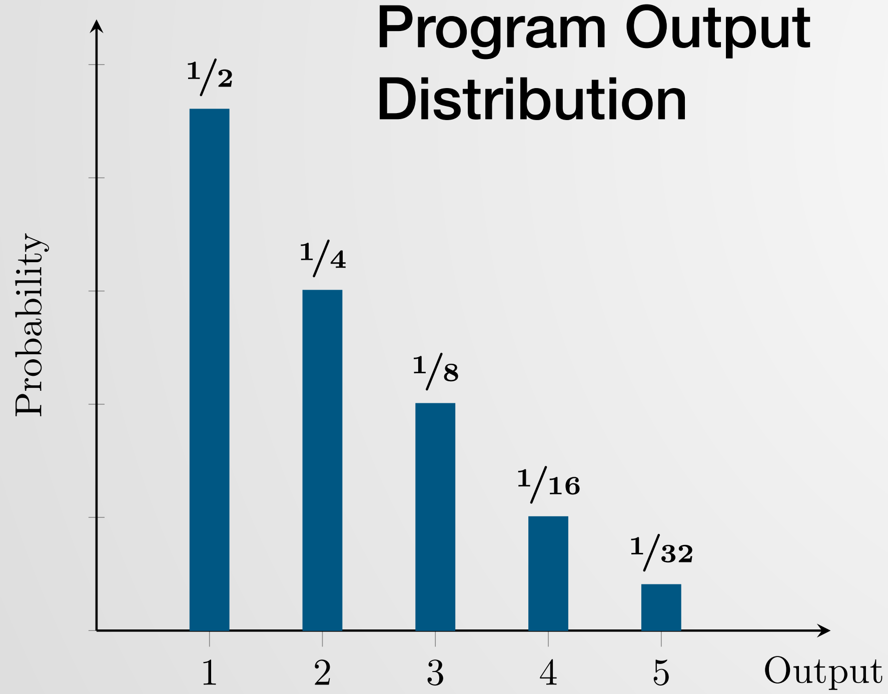
BACKUP SLIDES

What is a Probabilistic Program?

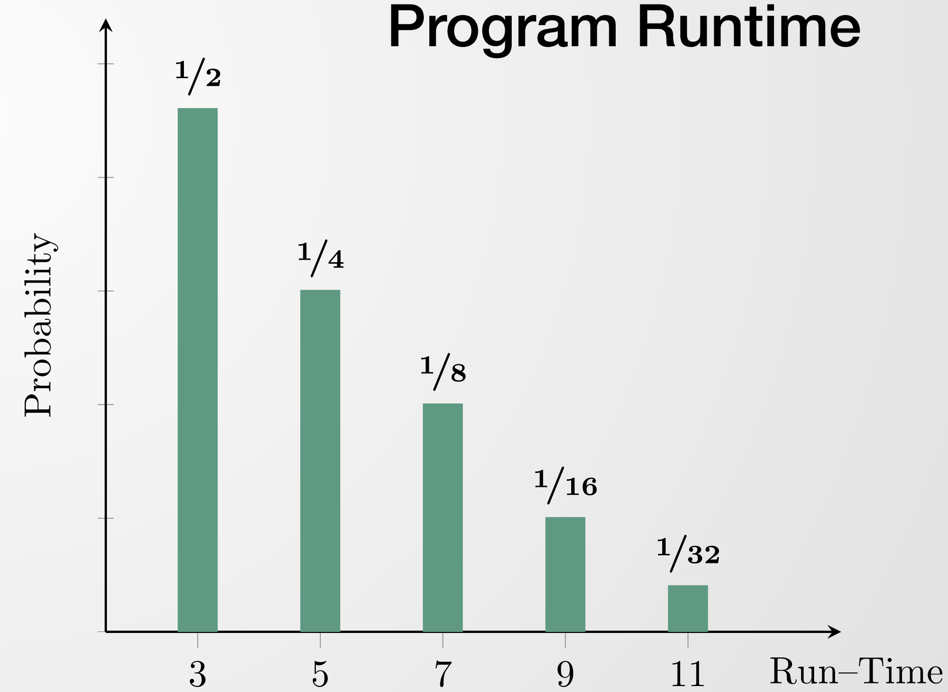
Probabilistic program that simulates a geometric distribution

C_{geo}:
```
n := 0;
repeat
   n := n + 1;
   c := coin_flip(0.5)
until (c=heads);
return n
```

Program Output Distribution



Program Runtime



Average (or Expected) Runtime:

$$3 \cdot \frac{1}{2} + 5 \cdot \frac{1}{4} + \cdots + (2n+1) \cdot \frac{1}{2^n} + \cdots = \mathbf{5}$$

Our Programming Model

Language Syntax

$$
\begin{array}{lcll}
\mathcal{C} & := & \texttt{skip} & \text{nop} \\
 & \mid & \texttt{abort} & \text{abortion} \\
 & \mid & x := E & \text{assignment} \\
 & \mid & \texttt{if}\,(G)\,\texttt{then}\,\{\mathcal{C}\}\,\texttt{else}\,\{\mathcal{C}\} & \text{conditional} \\
 & \mid & \{\mathcal{C}\}\ [p]\ \{\mathcal{C}\} & \textbf{probabilistic choice} \\
 & \mid & \texttt{call}\ P & \textbf{procedure call} \\
 & \mid & \mathcal{C};\ \mathcal{C} & \text{sequence}
\end{array}
$$

- We assume only one procedure P
- No argument passing or return expression in P (it manipules the *global* program state).

Our Programming Model

Language Syntax

$$
\begin{array}{lcll}
\mathcal{C} & := & \texttt{skip} & \text{nop} \\
& \mid & \texttt{abort} & \text{abortion} \\
& \mid & x := E & \text{assignment} \\
& \mid & \texttt{if}\,(G)\,\texttt{then}\,\{\mathcal{C}\}\,\texttt{else}\,\{\mathcal{C}\} & \text{conditional} \\
& \mid & \{\mathcal{C}\}\ [p]\ \{\mathcal{C}\} & \textbf{probabilistic choice} \\
& \mid & \texttt{call}\ P & \textbf{procedure call} \\
& \mid & \mathcal{C};\ \mathcal{C} & \text{sequence}
\end{array}
$$

- We assume only one procedure P
- No argument passing or return expression in P (it manipules the *global* program state).

Example: Factorial

$$
\begin{array}{l}
P \rhd\ \texttt{if}\ (x \leq 0)\ \texttt{then}\ \{y := 1\}\ \texttt{else} \\
\qquad \{\, x := x-1;\ \texttt{call}\ P; \\
\qquad\ \ x := x+1;\ y := y \cdot x \}
\end{array}
$$

Our Programming Model

Language Syntax

$$
\begin{array}{lcll}
\mathcal{C} & := & \texttt{skip} & \text{nop} \\
& \mid & \texttt{abort} & \text{abortion} \\
& \mid & x := E & \text{assignment} \\
& \mid & \texttt{if}\,(G)\,\texttt{then}\,\{\mathcal{C}\}\,\texttt{else}\,\{\mathcal{C}\} & \text{conditional} \\
& \mid & \{\mathcal{C}\}\ [p]\ \{\mathcal{C}\} & \textbf{probabilistic choice} \\
& \mid & \texttt{call}\ P & \textbf{procedure call} \\
& \mid & \mathcal{C};\ \mathcal{C} & \text{sequence}
\end{array}
$$

- We assume only one procedure P
- No argument passing or return expression in P (it manipules the *global* program state).

Example: Faulty factorial

$$
\begin{array}{l}
P \triangleright\ \texttt{if}\ (x \leq 0)\ \texttt{then}\ \{y := 1\}\ \texttt{else} \\
\qquad \{\, x := x-1;\ \texttt{call}\ P; \\
\qquad\quad x := x+1;\ \{y := y \cdot x\}[1/2]\{\texttt{skip}\}\,\}
\end{array}
$$

The Probabilistic Predicate Transformer — Inductive Definition

$$
\begin{aligned}
\mathrm{wp}[\mathtt{skip}](f) &= f \\
\mathrm{wp}[\mathtt{abort}](f) &= \mathbf{0} \\
\mathrm{wp}[x := E](f) &= f[x/E] \\
\mathrm{wp}[\mathtt{if}\,(G)\,\mathtt{then}\,\{c_1\}\,\mathtt{else}\,\{c_2\}](f) &= [G]\cdot \mathrm{wp}[c_1](f) + [\neg G]\cdot \mathrm{wp}[c_2](f) \\
\mathrm{wp}[\{c_1\}\ [p]\ \{c_2\}](f) &= p\cdot \mathrm{wp}[c_1](f) + (1{-}p)\cdot \mathrm{wp}[c_2](f) \\
\mathrm{wp}[c_1; c_2](f) &= (\mathrm{wp}[c_1]\circ \mathrm{wp}[c_2])(f) \\
\mathrm{wp}[\mathtt{call}\ P] &= \sup_n\ \mathrm{wp}[\,\mathtt{call}_n\ P\,]
\end{aligned}
$$

n-inlining of *P*

$$
\begin{aligned}
\mathtt{call}_0\ P &= \mathtt{abort} \\
\mathtt{call}_{n+1}\ P &= \mathit{body}(P)[\mathtt{call}\ P/\mathtt{call}_n\ P]
\end{aligned}
$$

The Expected Runtime Transformer — Inductive Definition

$$\begin{aligned}
\mathsf{ert}\,[\mathtt{skip}](t) &= \mathbf{1} + t \\
\mathsf{ert}\,[\mathtt{abort}](t) &= \mathbf{0} \\
\mathsf{ert}\,[x := E](t) &= \mathbf{1} + t[x/E] \\
\mathsf{ert}\,[\mathtt{if}\,(G)\,\mathtt{then}\,\{c_1\}\,\mathtt{else}\,\{c_2\}](t) &= \mathbf{1} + [G]\cdot \mathsf{ert}\,[c_1](t) \;+\; [\neg G]\cdot \mathsf{ert}\,[c_2](t) \\
\mathsf{ert}\,[\{c_1\}\ [p]\ \{c_2\}](t) &= \mathbf{1} + p\cdot \mathsf{ert}\,[c_1](t) \;+\; (1-p)\cdot \mathsf{ert}\,[c_2](t) \\
\mathsf{ert}\,[c_1; c_2](t) &= (\mathsf{ert}\,[c_1] \circ \mathsf{ert}\,[c_2])\,(t) \\
\mathsf{ert}\,[\mathtt{call}\ P](t) &= \mathit{lfp}\Big(\lambda\eta \bullet\ \underline{\mathbf{1}} \oplus \mathsf{ert}\,[\mathit{body}(P)]^{\sharp}_{\eta}\Big)(t)
\end{aligned}$$

“$\mathsf{ert}\,[\mathtt{call}\ P](t) \;=\; \mathbf{1} + \mathsf{ert}\,[\mathit{body}(P)](t)$”

Probabilistic Predicate Transformer — Calculation Example

Example 3. Reconsider the procedure P_{rec_3} with declaration

$$\mathcal{D}(P_{\mathsf{rec}_3}) : \quad \{\mathsf{skip}\}\ [1/2]\ \{\mathsf{call}\ P_{\mathsf{rec}_3};\ \mathsf{call}\ P_{\mathsf{rec}_3};\ \mathsf{call}\ P_{\mathsf{rec}_3}\}$$

presented in the introduction. We prove that it terminates with probability *at most* $\varphi = \frac{\sqrt{5}-1}{2}$ from any initial state. Formally, this is captured by $\mathsf{wp}[\mathsf{call}\ P, \mathcal{D}](\mathbf{1}) \preceq \boldsymbol{\varphi}$. To prove this, we apply rule [wp-rec]. We must then establish the derivability claim

$$\mathsf{wp}[\mathsf{call}\ P](\mathbf{1}) \preceq \boldsymbol{\varphi} \Vdash \mathsf{wp}[\mathcal{D}(P_{\mathsf{rec}_3})](\mathbf{1}) \preceq \boldsymbol{\varphi}\ .$$

The derivation goes as follows:

$$\begin{aligned}
& \mathsf{wp}[\mathcal{D}(P_{\mathsf{rec}_3})](\mathbf{1}) \\
= \quad & \{\text{def. of wp}\} \\
& \tfrac{1}{2} \cdot \mathsf{wp}[\mathsf{skip}](\mathbf{1}) + \tfrac{1}{2} \cdot \mathsf{wp}[\mathsf{call}\ P_{\mathsf{rec}_3};\ \mathsf{call}\ P_{\mathsf{rec}_3};\ \mathsf{call}\ P_{\mathsf{rec}_3}](\mathbf{1}) \\
= \quad & \{\text{def. of wp}\} \\
& \tfrac{\mathbf{1}}{\mathbf{2}} + \tfrac{1}{2} \cdot \mathsf{wp}[\mathsf{call}\ P_{\mathsf{rec}_3};\ \mathsf{call}\ P_{\mathsf{rec}_3}]\big(\mathsf{wp}[\mathsf{call}\ P_{\mathsf{rec}_3}](\mathbf{1})\big) \\
\preceq \quad & \{\text{assumption, monot. of wp}\} \\
& \tfrac{\mathbf{1}}{\mathbf{2}} + \tfrac{1}{2} \cdot \mathsf{wp}[\mathsf{call}\ P_{\mathsf{rec}_3}; \mathsf{call}\ P_{\mathsf{rec}_3}](\boldsymbol{\varphi}) \\
= \quad & \{\text{def. of wp, scalab. of wp twice}\} \\
& \tfrac{\mathbf{1}}{\mathbf{2}} + \tfrac{1}{2}\, \varphi \cdot \mathsf{wp}[\mathsf{call}\ P_{\mathsf{rec}_3}]\big(\mathsf{wp}[\mathsf{call}\ P_{\mathsf{rec}_3}](\mathbf{1})\big) \\
\preceq \quad & \{\text{assumption, monot. of wp}\} \\
& \tfrac{\mathbf{1}}{\mathbf{2}} + \tfrac{1}{2}\, \varphi \cdot \mathsf{wp}[\mathsf{call}\ P_{\mathsf{rec}_3}](\boldsymbol{\varphi}) \\
= \quad & \{\text{scalab. of wp}\} \\
& \tfrac{\mathbf{1}}{\mathbf{2}} + \tfrac{1}{2}\, \varphi^2 \cdot \mathsf{wp}[\mathsf{call}\ P_{\mathsf{rec}_3}](\mathbf{1}) \\
\preceq \quad & \{\text{assumption, monot. of wp}\} \\
& \tfrac{\mathbf{1}}{\mathbf{2}} + \tfrac{\mathbf{1}}{\mathbf{2}}\, \varphi^3 \\
= \quad & \{\text{algebra}\} \\
& \boldsymbol{\varphi}
\end{aligned}$$

$\triangle$

The Probabilistic Predicate Transformers — Proof Rules for Recursion

- Proof rule for upper bounds

"Prove the desired specification for the procedure's body assuming it already holds for the recursive calls in it."

$$\frac{\mathrm{wp}[\texttt{call}\ P](f) \leq u \quad \Vdash \quad \mathrm{wp}[\mathit{body}(P)](f) \leq u}{\mathrm{wp}[\texttt{call}\ P](f) \leq u}$$

$$\begin{array}{rl} & \langle u \rangle \\ & \\ \mathit{body}(P) & \left\{ \begin{array}{l} \vdots \\ \texttt{call}\ P \\ \vdots \end{array} \right. \\ & \\ & \langle f \rangle \end{array}$$

The Probabilistic Predicate Transformers — Proof Rules for Recursion

- Proof rule for upper bounds

"Prove the desired specification for the procedure's body assuming it already holds for the recursive calls in it."

$$\frac{\mathsf{wp}[\texttt{call}\ P](f) \leq u \quad \Vdash \quad \mathsf{wp}[\mathit{body}(P)](f) \leq u}{\mathsf{wp}[\texttt{call}\ P](f) \leq u}$$

$$\begin{array}{c} \langle u \rangle \\ \mathit{body}(P) \left\{ \begin{array}{c} \vdots \\ \texttt{call}\ P \\ \langle f' \rangle \\ \vdots \end{array} \right. \\ \langle f \rangle \end{array}$$

- Proof rule for upper bounds

"Prove the desired specification for the procedure's body assuming it already holds for the recursive calls in it."

$$\frac{\mathsf{wp}[\texttt{call } P](f) \leq u \quad \Vdash \quad \mathsf{wp}[\mathit{body}(P)](f) \leq u}{\mathsf{wp}[\texttt{call } P](f) \leq u}$$

$$\begin{array}{lcl} & \langle u \rangle & \\ & & \\ \mathit{body}(P) \left\{ \begin{array}{c} \vdots \\ \langle u \rangle \\ \texttt{call } P \\ \langle f' \rangle \\ \vdots \end{array} \right. & & f' \leq f \\ & & \\ & \langle f \rangle & \end{array}$$

- Proof rule for upper bounds

“Prove the desired specification for the procedure’s body assuming it already holds for the recursive calls in it.”

$$\frac{\mathsf{wp}[\texttt{call}\ P](f) \leq u \quad \Vdash \quad \mathsf{wp}[\mathit{body}(P)](f) \leq u}{\mathsf{wp}[\texttt{call}\ P](f) \leq u}$$

$$\begin{array}{lcl} & \langle u \rangle & \\ & \langle u' \rangle & \\ \mathit{body}(P) \left\{ \begin{array}{c} \vdots \\ \langle u \rangle \\ \texttt{call}\ P \\ \langle f' \rangle \\ \vdots \end{array} \right. & & f' \leq f \\ & \langle f \rangle & \end{array}$$

- Proof rule for upper bounds

"Prove the desired specification for the procedure's body assuming it already holds for the recursive calls in it."

$$\frac{\mathsf{wp}[\texttt{call } P](f) \leq u \quad \Vdash \quad \mathsf{wp}[\mathit{body}(P)](f) \leq u}{\mathsf{wp}[\texttt{call } P](f) \leq u}$$

$$\begin{array}{rcl} & \langle u \rangle & \\ & \langle u' \rangle & u' \leq u \\ \mathit{body}(P) \left\{ \begin{array}{c} \vdots \\ \langle u \rangle \\ \texttt{call } P \\ \langle f' \rangle \\ \vdots \end{array} \right. & & f' \leq f \\ & \langle f \rangle & \end{array}$$

The Probabilistic Predicate Transformers — Proof Rules for Recursion

- Proof rule for upper bounds

“Prove the desired specification for the procedure’s body assuming it already holds for the recursive calls in it.”

$$\frac{\mathrm{wp}[\texttt{call}\ P](f) \leq u \quad \Vdash \quad \mathrm{wp}[\mathit{body}(P)](f) \leq u}{\mathrm{wp}[\texttt{call}\ P](f) \leq u}$$

$$\begin{array}{rcl}
 & \langle u \rangle & \\
 & \langle u' \rangle & u' \leq u \\
\mathit{body}(P) \left\{ \begin{array}{c} \vdots \\ \langle u \rangle \\ \texttt{call}\ P \\ \langle f' \rangle \\ \vdots \end{array} \right. & & f' \leq f \\
 & \langle f \rangle &
\end{array}$$

- Proof rule for lower bounds

$$\frac{\begin{array}{c} l_0 = 0 \\ l_n \leq \mathrm{wp}[\texttt{call}\ P](f) \quad \Vdash \quad l_{n+1} \leq \mathrm{wp}[\mathit{body}(P)](f) \end{array}}{\sup_n l_n \leq \mathrm{wp}[\texttt{call}\ P](f)}$$

  - Dual rule for upper bounds is also sound

The Expected Runtime Transformers — Proof Rules for Recursion

Rules from the wp—calculus can be easily adapted for the ert—calculus

- Proof rule for upper bounds

$$\frac{\mathsf{ert}\left[\texttt{call } P\right](t) \leq u + \mathbf{1} \quad \Vdash \quad \mathsf{ert}\left[\mathit{body}(P)\right](t) \leq u}{\mathsf{ert}\left[\texttt{call } P\right](t) \leq u + \mathbf{1}}$$

- Proof rule for upper bounds

$$\frac{l_0 = 0 \qquad l_n + \mathbf{1} \leq \mathsf{ert}\left[\texttt{call } P\right](t) \quad \Vdash \quad l_{n+1} \leq \mathsf{ert}\left[\mathit{body}(P)\right](t)}{\sup_n l_n + \mathbf{1} \leq \mathsf{ert}\left[\texttt{call } P\right](t)}$$

Proof Rule for Mutually Recursive Procedures

$$\frac{\begin{array}{ccc} \mathsf{wp}[\texttt{call } P_1](f_1) \leq g_1, \ldots, \mathsf{wp}[\texttt{call } P_m](f_m) \leq g_m & \Vdash & \mathsf{wp}[\mathit{body}(P_1)](f_1) \leq g_1 \\ & \vdots & \\ \mathsf{wp}[\texttt{call } P_1](f_1) \leq g_1, \ldots, \mathsf{wp}[\texttt{call } P_m](f_m) \leq g_m & \Vdash & \mathsf{wp}[\mathit{body}(P_m)](f_m) \leq g_m \end{array}}{\mathsf{wp}[\texttt{call } P_i](f_i) \leq g_i \quad \text{for all } i = 1 \ldots m}$$

Operational Semantics

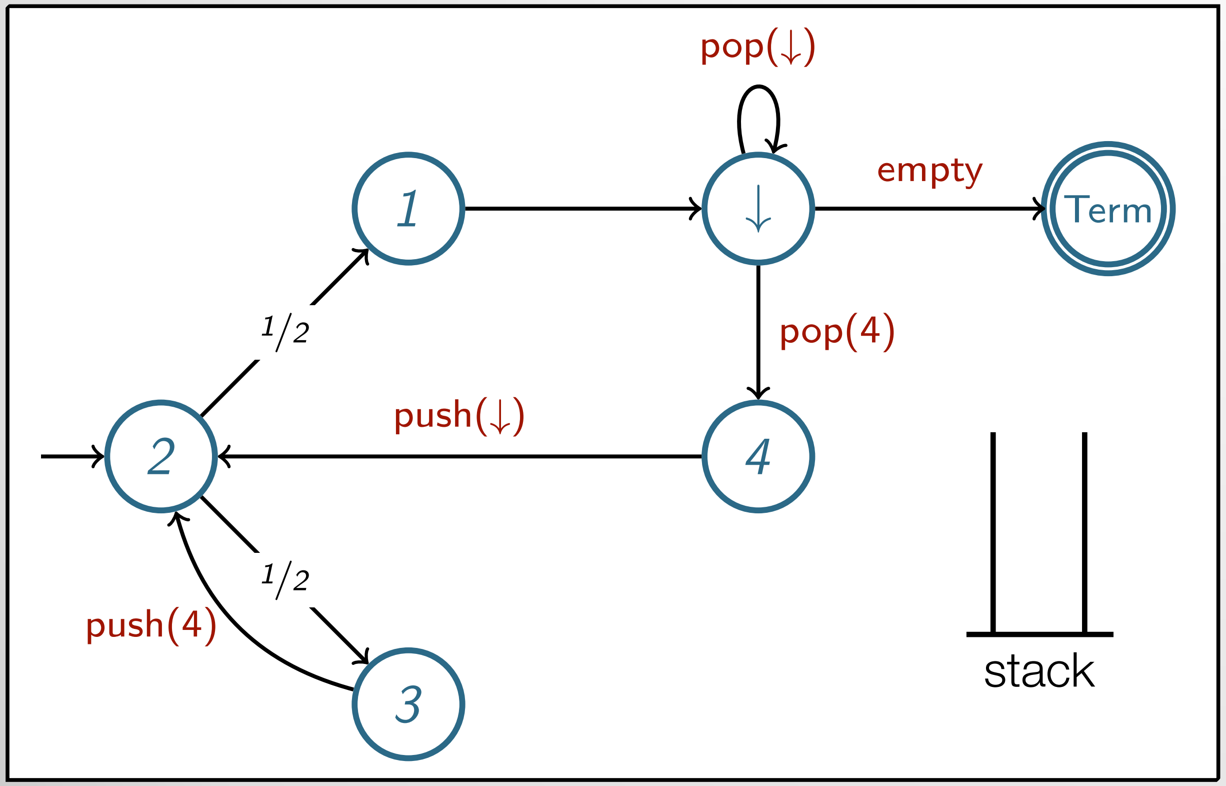
- To each program c, initial state s_0 and post-condition f we associate a **reward pushdown Markov chain** $\mathfrak{M}^{f}_{s_0}[\![c]\!]$
- We prove that the weakest pre-condition $\mathsf{wp}[c](f)(s_0)$ coincides with the **expected reward** $\mathbf{ER}(\Diamond \mathsf{Term})$ upon reaching a terminal state in the Markov chain

$$\mathsf{wp}[c](f)(s_0) \;=\; \mathbf{ER}(\Diamond \mathsf{Term})$$

SOUNDNESS RESULT

Example:

$$P \triangleright \quad \{\texttt{skip}^1\} \; [1/2]^2 \; \{\texttt{call } P^3;\; \texttt{call } P^4\}$$



$$\mathbf{ER}(\Diamond \mathsf{Term}) \;=\; \sum_{\pi : \langle \ell_0, s_0\rangle \rightsquigarrow \langle \mathsf{Term}, s'\rangle} \Pr(\pi)\cdot f(s') \;\overset{f=\mathbf{1}}{=}\; \frac{1}{2} + \frac{1}{2}\cdot\left(\frac{1}{2}\right)^2 + \cdots$$

Operational Semantics

$$\frac{\mathsf{stmt}\,(\ell) = \mathsf{skip} \quad \mathsf{succ}_1\,(\ell) = \ell'}{\langle \ell,\, s\rangle \xrightarrow{\gamma,\,1,\,\gamma} \langle \ell',\, s\rangle}\,[\text{skip}] \qquad \frac{\mathsf{stmt}\,(\ell) = x := E \quad \mathsf{succ}_1\,(\ell) = \ell'}{\langle \ell,\, s\rangle \xrightarrow{\gamma,\,1,\,\gamma} \big\langle \ell',\, s\big[x \mapsto s(E)\big]\big\rangle}\,[\text{assign}] \qquad \frac{\mathsf{stmt}\,(\ell) = \mathsf{abort}}{\langle \ell,\, s\rangle \xrightarrow{\gamma,\,1,\,\gamma} \langle \ell,\, s\rangle}\,[\text{abort}]$$

$$\frac{\mathsf{stmt}\,(\ell) = \mathsf{if}\,(G)\,\{c_1\}\,\mathsf{else}\,\{c_2\} \quad s \models G \quad \mathsf{succ}_1\,(\ell) = \ell'}{\langle \ell,\, s\rangle \xrightarrow{\gamma,\,1,\,\gamma} \langle \ell',\, s\rangle}\,[\text{if1}] \qquad \frac{\mathsf{stmt}\,(\ell) = \mathsf{if}\,(G)\,\{c_1\}\,\mathsf{else}\,\{c_2\} \quad s \not\models G \quad \mathsf{succ}_2\,(\ell) = \ell'}{\langle \ell,\, s\rangle \xrightarrow{\gamma,\,1,\,\gamma} \langle \ell',\, s\rangle}\,[\text{if2}]$$

$$\frac{\mathsf{stmt}\,(\ell) = \{c_1\}\,[p]\,\{c_2\} \quad \mathsf{succ}_1\,(\ell) = \ell'}{\langle \ell,\, s\rangle \xrightarrow{\gamma,\,p,\,\gamma} \langle \ell',\, s\rangle}\,[\text{prob1}] \qquad \frac{\mathsf{stmt}\,(\ell) = \{c_1\}\,[p]\,\{c_2\} \quad \mathsf{succ}_2\,(\ell) = \ell'}{\langle \ell,\, s\rangle \xrightarrow{\gamma,\,1-p,\,\gamma} \langle \ell',\, s\rangle}\,[\text{prob2}]$$

$$\frac{\mathsf{stmt}\,(\ell) = \mathsf{call}\,P \quad \mathsf{succ}_1\,(\ell) = \ell'}{\langle \ell,\, s\rangle \xrightarrow{\gamma,\,1,\,\gamma\cdot\ell'} \big\langle \mathsf{init}\big(\mathcal{D}(P)\big),\, s\big\rangle}\,[\text{call}] \qquad \frac{}{\langle \downarrow,\, s\rangle \xrightarrow{\ell',\,1,\,\varepsilon} \langle \ell',\, s\rangle}\,[\text{return}] \qquad \frac{}{\langle \downarrow,\, s\rangle \xrightarrow{\gamma_0,\,1,\,\gamma_0} \langle \mathsf{Term},\, s\rangle}\,[\text{terminate}]$$

Figure 3. Rules for defining an operational semantics for pRGCL programs. For sequential composition there is no dedicated rule as the control flow is encoded via the succ_1 and the succ_2 functions.