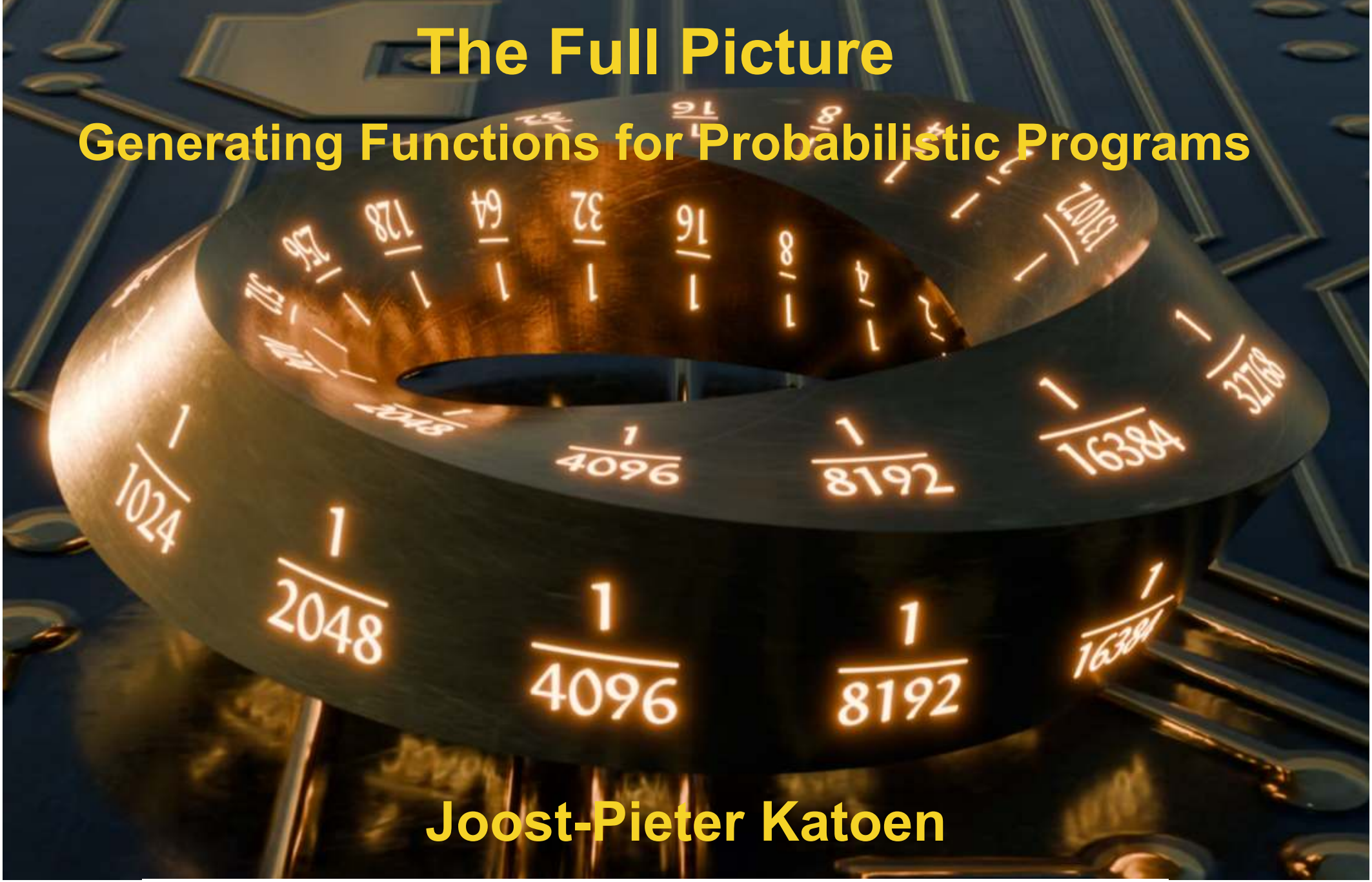


The Full Picture

Generating Functions for Probabilistic Programs

Joost-Pieter Katoen

42nd Conference on
Mathematical Foundations of Programming
Semantics
MFPS XLII, Ljubljana 2026

Motivation and objectives

Discrete Samplers

```
while (x = y) {
   x := 0 [p] x := 1;
   y := 0 [p] y := 1;
}
```

$0 < p < 1$

biased coin

```
c := 0; x := 0;
while (x ≠ 0) {
   x := 0 [1/2] x := 1;
   c := c+1
}
```

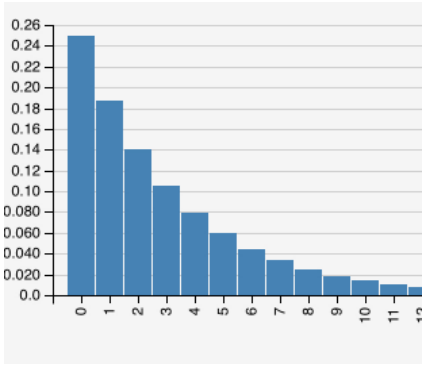
fair coin

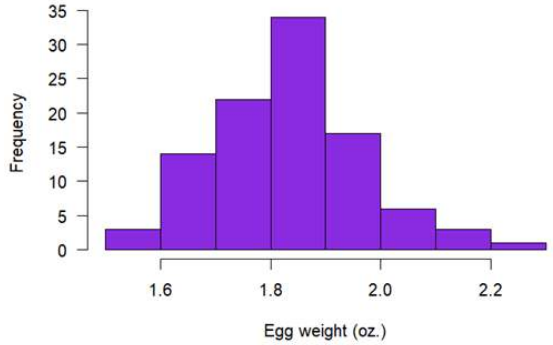
```
v := 1;  c := 0;
while (v < n) {
   v := 2v;
   c := 2c [1/2] c := 2c+1;
   if (c ⩾ n) {
         v := v-n;  c := c-n
   }
}
```

fair coin

Which probability distributions are computed by these programs? Exactly.

Programs := Distribution Transformers





Von Neumann's Sampler

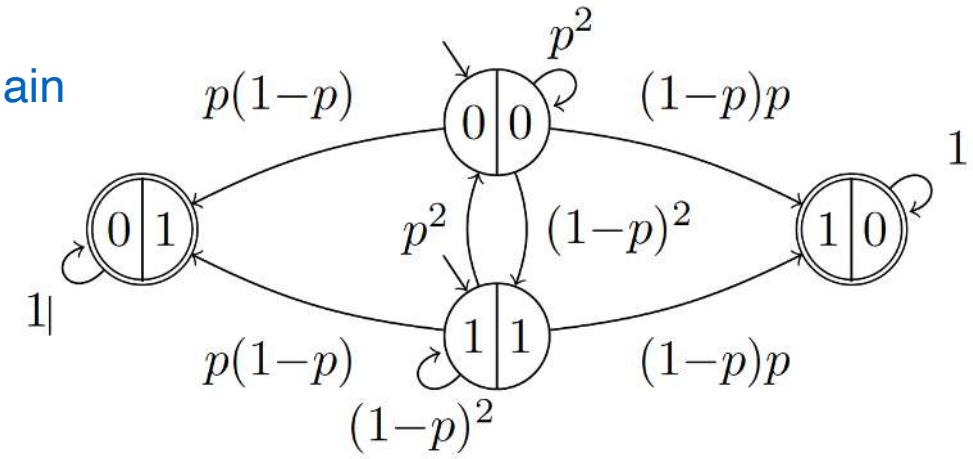
[Von Neumann, 1951]

```
while (x = y) {
   x := 0 [p] x := 1;
   y := 0 [p] y := 1;
}
```

$$0 < p < 1$$

Von Neumann's sampler

Its Markov chain



$$\begin{pmatrix} 0\,0 \\ 0\,1 \\ 1\,0 \\ 1\,1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}_0 \rightsquigarrow \begin{pmatrix} 0.16 \\ 0.24 \\ 0.24 \\ 0.36 \end{pmatrix}_1 \rightsquigarrow \begin{pmatrix} 0.0832 \\ 0.3648 \\ 0.3648 \\ 0.1872 \end{pmatrix}_2 \rightsquigarrow \cdots \rightsquigarrow \begin{pmatrix} 0.0004 \\ 0.4993 \\ 0.4993 \\ 0.0010 \end{pmatrix}_{10} \rightsquigarrow \cdots \begin{pmatrix} 0.0 \\ 0.5 \\ 0.5 \\ 0.0 \end{pmatrix}_{\omega}$$

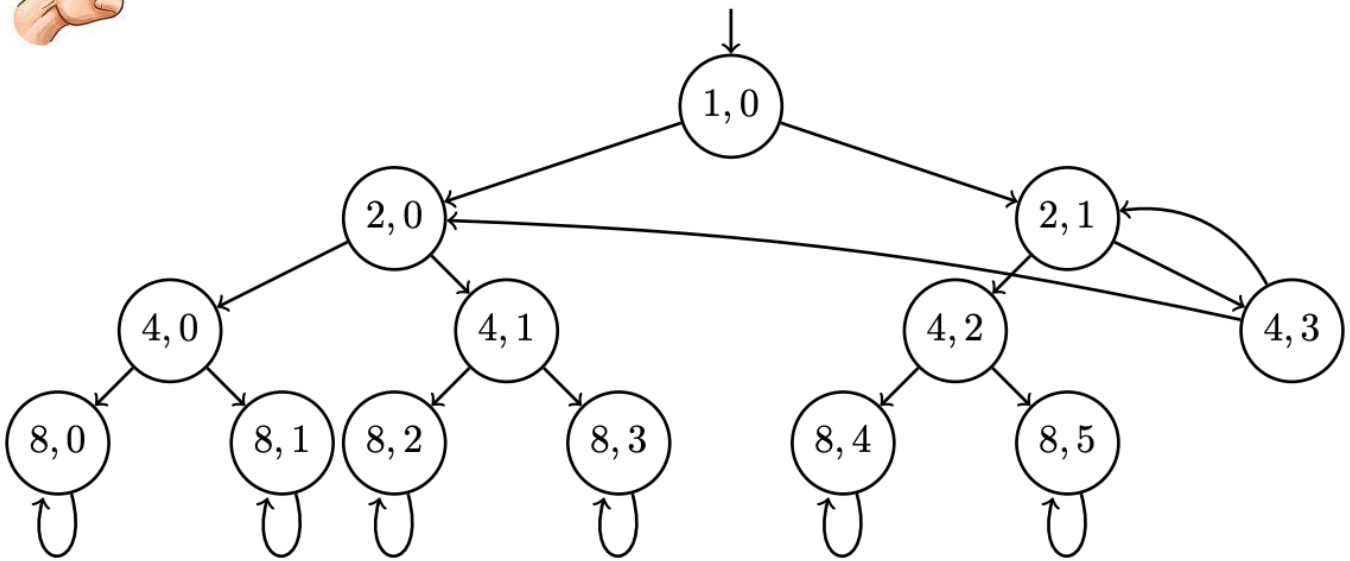
An infinite trajectory of reachable distributions

$Pr[0\,0] = 1$ $\quad p = 2/5$

Lumbroso's Sampler

[Lumbroso, 2013]

```
v := 1;  c := 0;
while (v < n) {
   v := 2v;
   c := 2c [1/2] c := 2c+1;
   if (c ⩾ n) {
         v := v-n;  c := c-n
   }
}
```



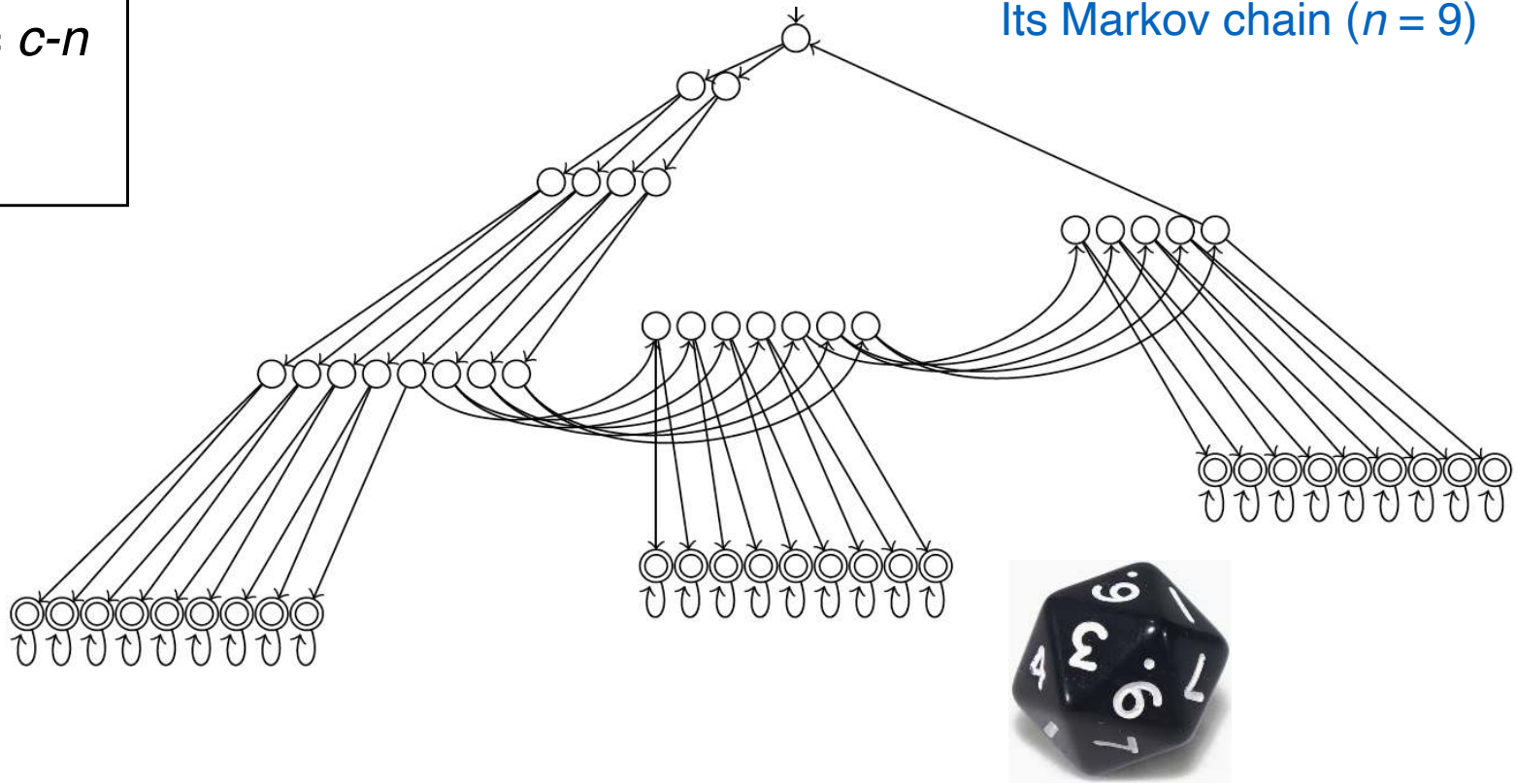
Its Markov chain ($n = 6$)

Lumbroso's Sampler

[Lumbroso, 2013]

```
v := 1;  c := 0;
while (v < n) {
   v := 2v;
   c := 2c [1/2] c := 2c+1;
   if (c ⩾ n) {
         v := v-n;  c := c-n
   }
}
```

Its Markov chain ($n = 9$)



Aim

Given an imperative probabilistic program P and a distributional Hoare triple

$$\{|\,\text{pre}\,|\}\ \ P\ \ \{|\,\text{post}\,|\}$$

prove that P, when executed on an input distribution satisfying pre

yields an output (sub-)distribution satisfying post

Example Distributional Hoare Triples

$\{|\, x = y \,|\}$

```
while (x = y) {
   x := 0 [p] x := 1;
   y := 0 [p] y := 1;
}
```

$\{|\, x \sim \mathrm{unif}(0,1) \,|\}$

$\{|\, c \sim \delta_0, x \sim \delta_0 \,|\}$

```
while (x ≠ 0) {
   x := 0 [1/2] x := 1;
   c := c + 1
}
```

$\{|\, c \sim \mathrm{geom}(1/2) \,|\}$

$\{|\, v \sim \delta_1, c \sim \delta_0, n \sim \delta_{\mathbf{n}} \mid \mathbf{n} > 0 \,|\}$

```
while (v < n) {
   v := 2v;
   c := 2c [1/2] c := 2c+1;
   if (c ⩾ n) {
         v := v-n;  c := c-n
   }
}
```

$\{|\, c \sim \mathrm{unif}(0,n) \,|\}$

Challenges

- exact output distributions required
- such distributions may only be reached in the limit
- parameterised samplers require reasoning about

  infinite families of distributions
- automation:
  - how to represent distributions?
  - what is still decidable?

computing expected values is Π^0_2 complete
[Kaminski & K., MFCS 2015]

PROBABILITY

GENERATING FUNCTIONS

Joost-Pieter Katoen

Formal Power Series

A univariate formal power series is an algebraic object

$$F(X) = a_0 \cdot X^0 + a_1 \cdot X^1 + \ldots = \sum_{n=0}^{\infty} a_n \cdot X^n$$

where the coefficients belong to some semiring, e.g. the reals

(Absolute) convergence of series irrrelevant, X is a formal symbol, operations are algebraic

A multivariate formal power series has the form

$$F(X_1, \ldots, X_k) = a_{0_1,\ldots,0_k} \cdot X_1^{0_1} \cdots X_k^{0_k} + a_{1_1,\ldots,1_k} \cdot X_1^{1_1} \cdots X_k^{1_k} + \ldots = \sum_{i \in \mathbb{N}^k} a_{i_1,\ldots,i_k} \cdot X_1^{i_1} \cdots X_k^{i_k}$$

Applications include: combinatorics algebra, algebraic geometry, number theory

Probability Generating Functions

A probability generating function is a specific analytic use of FPS associated with a discrete random variable

A univariate probability generating function (PGF) for random variable X is

$$F(X) = a_0 \cdot X^0 + a_1 \cdot X^1 + \ldots = \sum_{n=0}^{\infty} a_n \cdot X^n$$

where the real coefficients satisfy: $0 \leq a_i \leq 1$ and $\sum_i a_i = 1.$ Interpretation: $a_i = \Pr(X = i)$

The PGF for $X \sim \text{geom}(1/2)$ is:

value of X

$$\frac{1}{2} \cdot \sum_{i \in \mathbb{N}} \left(\frac{X}{2}\right)^i = \frac{1}{2} + \frac{1}{4}X + \frac{1}{8}X^2 + \frac{1}{16}X^3 + \frac{1}{32}X^4 + \frac{1}{64}X^5 + \frac{1}{128}X^6 + \frac{1}{256}X^7 + \ldots$$

$\Pr(X = 3)$

Closed Forms

PGFs are "infinite" polynomials $F(X) = a_0 \cdot X^0 + a_1 \cdot X^1 + \ldots = \sum_{n=0}^{\infty} a_n \cdot X^n$

A PGF F is in rational closed-form if $F = \frac{G}{H}$ where G, H are polynomials

Our example $\frac{1}{2-X} = \frac{1}{2}X^0 + \frac{1}{4}X^1 + \frac{1}{8}X^2 + \frac{1}{16}X^3 + \frac{1}{32}X^4 + \frac{1}{64}X^5 + \ldots$

Closed-form PGFs are concise representations of (possibly infinite-support) probability distributions

Distribution D	Closed-form PGF
bernoulli(p)	$(1-p)X^0 + pX^1$
binomial(p, n)	$((1-p)X^0 + pX^1)^n$
uniform(a, b)	$\frac{X^a - X^{b+a}}{1-X}$
geometric(p)	$\frac{1-p}{1-pX}$
poisson(λ)	$e^{\lambda(X-1)}$

closed form, not a rational closed form

Algebraic Operations

Let $G = \sum_{n \in \mathbb{N}} a_n X^n$ and $F = \sum_{n \in \mathbb{N}} b_n X^n$

Operation	Effect on Coefficients	Intuition
$X \cdot G$	$(0, a_0, a_1, a_2, \ldots)$	Shift in X dimension
$r \cdot G$	$(r \cdot a_0, r \cdot a_1, r \cdot a_2, \ldots)$	Scalar multiple
$F + G$	$(a_0 + b_0, a_1 + b_1, \ldots)$	Pointwise addition
$F \cdot G$	$(a_0 b_0, a_0 b_1 + a_1 b_0, \ldots)$	Discrete convolution
$\partial_X G$	$(a_1, 2a_2, 3a_3, \ldots)$	Weighted rev. X-shift
$G[X/0]$	$(a_0, 0, 0, \ldots)$	Drop dimension X
$G[X/1]$	$(\sum_{n \in \mathbb{N}} a_n, 0, 0 \ldots)$	Accumulate along X

Operations are structural changes fo the coefficients

Algebraic Operations

Let $G = \sum_{n\in\mathbb{N}} a_n X^n$ and $F = \sum_{n\in\mathbb{N}} b_n X^n$

$F \cdot G$	$(a_0 b_0, a_0 b_1 + a_1 b_0, \ldots)$	Discrete convolution

fair coin $F = \frac{1}{2}x^0 + \frac{1}{2}x^1 = \frac{1+x}{2}$

6-sided die $G = \frac{1}{6}x^0 + \ldots + \frac{1}{6}x^6 = \frac{x(1-x^6)}{6(1-x)}$

$$F \cdot G = \left(\frac{1+x}{2}\right)\left(\frac{x(1-x^6)}{6(1-x)}\right)$$

Operations on PGFs

Marginalize out X	$F[X/1]$
Probability of $x = k$	$\frac{1}{k!} \cdot \partial_X^k F[X/0]$
Expected value of x	$\partial_X F[X/1]$
Variance of x	$\partial_X (X \cdot \partial_X F)[X/1] - \partial_X F[X/1]$
Total probability mass	$F[X_0/1, \ldots, X_k/1]$

$$\frac{1}{2}X^0 + \frac{1}{4}X^1 + \frac{1}{8}X^2 + \frac{1}{16}X^3 + \frac{1}{32}X^4 + \frac{1}{64}X^5 + \ldots$$

Operations on PGFs

Expected value of x	$\partial_X F[X/1]$

$$F = \frac{1}{2}X^0 + \frac{1}{4}X^1 + \frac{1}{8}X^2 + \ldots = \frac{1}{2-X}$$

$$E[X] = \delta_X (2-X)^{-1} [X/1]$$

$$= (2-X)^{-2} [X/1] = 1$$

Operations on PGFs

Probability of $x = k$	$\frac{1}{k!} \cdot \partial_X^k F[X/0]$

$$F = \frac{1}{2}X^0 + \frac{1}{4}X^1 + \ldots + \frac{1}{2^{k+1}}X^k + \ldots$$

$\uparrow \Pr\{X=k\}$

$$\delta_x^k F[X/0] = k \cdot (k-1) \cdots 1 \cdot \frac{1}{2^{k+1}}$$

$$\Rightarrow \Pr\{X=k\} = \frac{1}{k!} \cdot \delta_x^k F[X/0]$$

PGF Composition Theorem

Let N be an $\mathbb{N}-$ valued random variable

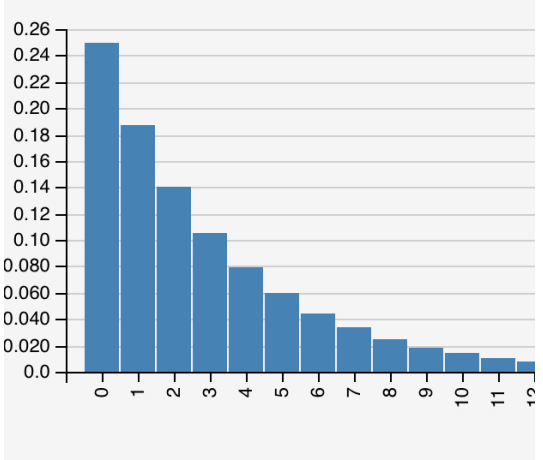
Let X_i be a series of i.i.d discrete random variables

Then: $\underbrace{X_1 + \ldots + X_N}_{N \text{ terms}} \sim F_N \circ F_X$

Iterate Randomly Often

[Flajolet et al., 2011]

```
x := geometric(1/4);
y := geometric(1/4);
t := x+y+1 [5/9] t := x+y;
r := 1;
for i in 1..3 {
   s := 0
   for j in 1..2t {
      s := s+1 [1/2] skip
   }
   r := (s == t)
}
```



What is the probability that r equals one on termination?

$$\pi^{-1}$$

PROGRAM SEMANTICS USING GENERATING FUNCTIONS

Joost-Pieter Katoen

Example Program Semantics


```
c := 0; x := 0;
while (x ≠ 0) {
   x := 0 [1/2] x := 1;
   c := c+ 1
}
```

Program with body *P*

$$/\!/\!/ 1 = X^0 C^0$$

$$\{x := 0\}[1/2]\{x := 1\}$$

$$/\!/\!/ \tfrac{1}{2} X^0 C^0 + \tfrac{1}{2} X^1 C^0$$

$$c := c + 1$$

$$/\!/\!/ \tfrac{1}{2} C + \tfrac{1}{2} CX$$

The semantics of the program body *P* for input *x*=0 and *c*=0 is

$$[\![P]\!](1) = \tfrac{1}{2} C + \tfrac{1}{2} CX$$

Program Syntax

skip

$x_i := E$

$\{P_1\}[p]\{P_2\}$

$\text{if}(B)\ \{P_1\}\ \text{else}\ \{P_2\}$

$P_1 \fatsemi P_2$

$x \mathrel{+}= \text{iid}(\mathcal{D}, y)$

$\text{while}(B)\ \{P\}$

program variables are natural numbers

program states $\sigma \in \mathbb{N}^k$

represent program states

as multivariate monomials

$X^2 \cdot Y^3$ denotes $\sigma = (2,3)$

$$F = \underbrace{\left(\frac{3}{4}, \frac{1}{4}\right)}_{[\sigma]_F} \underbrace{X^2 \cdot Y^3}_{\sigma}$$ distribution

Semantics of Straight-Line Programs

P	$[\![P]\!](F)$
skip	F
$x_i := E$	$\sum_{\sigma \in \mathbb{N}^k} [\sigma]_F \cdot X_1^{\sigma_1} \cdots X_i^{\mathsf{eval}_\sigma(E)} \cdots X_k^{\sigma_k}$
$\{P_1\}[p]\{P_2\}$	$p \cdot [\![P_1]\!](F) \;+\; (1-p) \cdot [\![P_2]\!](F)$
$\mathsf{if}(B)\ \{P_1\}\ \mathsf{else}\ \{P_2\}$	$[\![P_1]\!](\langle F \rangle_B) \;+\; [\![P_2]\!](\langle F \rangle_{\neg B})$
$P_1 \fatsemi P_2$	$[\![P_2]\!]([\![P_1]\!](F))$
$x \mathrel{+}= \mathsf{iid}(\mathcal{D}, y)$	$F\,[Y/Y\,[\![\mathcal{D}]\!]]$

set X_i to value of E in state σ

sum y times a sample from $\mathcal{D}$

filtering $\langle F \rangle_B = \sum_{x_i \vDash B} a_i \cdot X^i$

Semantics of Loops

Loops correspond to a least fixed point:

measure transformer

$$[\![\text{while}(B)\{\ P\ \}]\!](F) \;=\; \big(\text{lfp}\ \Phi_{B,P}\big)(F)$$

$$\text{where } \Phi_{B,P}(\varphi) : \;\; (\text{PGF} \longrightarrow \text{PGF}) \longrightarrow (\text{PGF} \longrightarrow \text{PGF})$$

$$\Phi_{B,P}(\varphi) \;=\; \lambda F\,.\,\langle F\rangle_{\neg B} \;+\; \varphi([\![P]\!](\langle F\rangle_{B})$$

Ordering of PGFs: $F = \sum_i a_i X^i \;\sqsubseteq\; G = \sum_i b_i X^i$ iff $a_i \leq b_i$ for all $i \in \mathbb{N}$

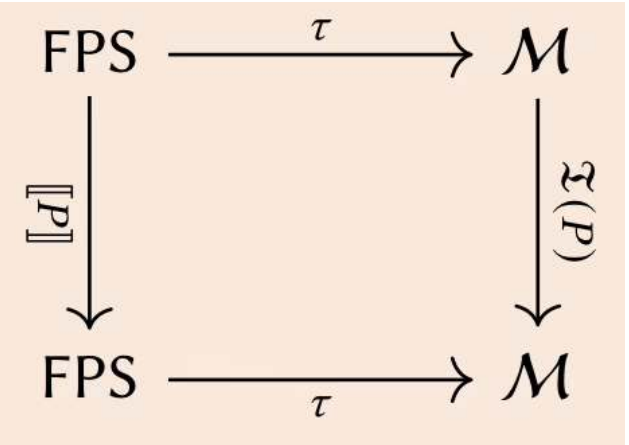
Properties PGF Semantics

$[\![\cdot]\!]$ is well-defined: $[\![\text{while}(B)\{\ P\ \}]\!] = \text{lfp}\ \Phi_{B,P} = \sup_n \Phi^n_{B,P}(\mathbf{0})$

$[\![\cdot]\!]$ is mass preserving: $|\,[\![P]\!](F)\,| \leqslant |\,F\,|$ for $F = \sum_i a_i X^i$, let $|\,F\,| = \sum_i a_i$

$[\![\cdot]\!]$ is linear: $[\![P]\!](r \cdot F + G) = r \cdot [\![P]\!](F) + [\![P]\!](G)$

$[\![\cdot]\!]$ corresponds to Kozen's semantics:



“Closed-Form” Loops

For loop $L = \text{while}\,(B)\{\,P\,\}$ and state σ, the PGF $[\![L]\!](X^\sigma)$ is a rational closed form

if (1) for all $\sigma \vDash B$, $[\![P]\!](X^\sigma)$ is a polynomial, and for each x_i either

(2) $[\![P]\!](X_i \cdot G) = X_i \cdot [\![P]\!](G)$ for each PGF *G*, and *B* does not depend on x_i, or

(3) for set $\{\,\sigma_i \mid \sigma \vDash B\,\}$ is finite.

```
while (x ≠ 0) {
   x := 0 [1/2] x := 1;
   c := c + 1
}
```

$$[\![L]\!](X) = \frac{1}{2}[\![L]\!](X) + \frac{1}{2}C \iff [\![L]\!](X) = \frac{2}{2-C}$$

Termination probability? $[\![L]\!](X)[X/1,C/1] = \dfrac{1}{2-1} = 1$

$$\mathbf{E}[c] = \partial_C [\![L]\!](X)[X/1,C/1] = \frac{2}{(2-C)^2}[C/1] = 2$$

VERIFYING LOOP INVARIANTS USING GENERATING FUNCTIONS

Joost-Pieter Katoen

Park's Lemma

[Park, 1969]

Let $\Phi_{B,P}$ be the unfolding operator of a loop while $(B)\ \{P\}$ *and* $f : \mathsf{PGF} \to \mathsf{PGF}$. *Then*

$$\Phi_{B,P}(f) \sqsubseteq f \quad \text{implies} \quad [\![\text{while}\,(B)\ \{P\}]\!] \sqsubseteq f$$

```
while (x = 1) {
    {x := 0}[1/2]{x := 1};
    c := c + 1
}
```

Let $$f(X^i C^j) = C^j \cdot \begin{cases} \frac{C}{2-C}, & \text{if } i = 1; \\ X^i, & \text{if } i \neq 1. \end{cases}$$

Then $\Phi_{B,P}(f)(X^i C^j) \sqsubseteq f(X^i C^j)$ ✓

Since the program is UAST: $\Phi_{B,p} = f$

Thus $[\![\text{while}\,(B)\ \{P\}]\!] = f$

Proof

```
while (x = 1) {
    {x := 0}[1/2]{x := 1};
    c := c + 1
}
```

$$f(X^i C^j) = C^j \cdot \begin{cases} \frac{C}{2-C}, & \text{if } i = 1; \\ X^i, & \text{if } i \neq 1. \end{cases}$$

To check $\Phi_{B,P}(f) \equiv f$

let $i=1$: $\Phi_{B,P}(f)(X^1 C^j)$

$= f(\frac{1}{2} X^0 C^{j+1} + \frac{1}{2} X^1 C^{j+1})$

$= \frac{1}{2} f(X^0 C^{j+1}) + \frac{1}{2} f(X^1 C^{j+1})$

$= \frac{1}{2} X^0 C^{j+1} + \frac{1}{2} C^{j+1} \frac{C}{2-C}$

$= \frac{1}{2} C^{j+1} (1 + \frac{C}{2-C})$

$= \frac{1}{2} C^{j+1} (\frac{2}{2-C})$

$= C^{j+1} (\frac{1}{2-C}) = f(X^1 C^j)$ ⊠

✓

More Examples

```
while (x ≠ 0) {
  {x := x − 1}[1/2]{x := x + 1};
  c := c + 1
}
```

1D symmetric random walk

$$h(X^i C^j) = C^j \cdot \begin{cases} \left(\frac{1-\sqrt{1-C^2}}{C}\right)^i, & \text{if } i \geq 1 \\ 1, & \text{if } i = 0 \end{cases}$$

$$\Phi_{B,P}(h) = h$$ ✓

semi-automatically using SymPy

```
while (x > 0) {
  {x := x − 1}[1/x]{x := x − 1}
}
```

1D random walk that drifts to the right

$$f(X^i) = 1 - \frac{1}{e} \cdot \sum_{n=0}^{i-2} \frac{1}{n!}$$

$$\Phi_{B,P}(f) = f$$ ✓

not almost surely terminating

PROGRAM EQUIVALENCE USING GENERATING FUNCTIONS

Joost-Pieter Katoen

Our Aim in Terms of Program Equivalence

Program

```
while (x = y) {
   x := 0 [p] x := 1;
   y := 0 [p] y := 1;
}
```

$\{| x \sim \text{unif}(0,1) |\}$

```
while (x ≠ 0) {
   x := 0 [1/2] x := 1;
   c := c + 1
}
```

$\{| c \sim \text{geom}(1/2) |\}$

```
while (v < n) {
   v := 2v;
   c := 2c [1/2] c := 2c+1;
   if (c ⩾ n) {
         v := v-n;  c := c-n
   }
}
```

$\{| c \sim \text{unif}(0,n) |\}$

Specification

```
if (x ≠ y) {
    x := unif(0,1)
}
```

```
if (x ≠ 0) {
    c +:= geom(1/2);
    x := 0
    }
```

```
if (v = 1 ∧ c = 0 ∧ v < n) {
    c := unif(0,n)
}
```

Our Aim

Does loopy program L produce exactly the same distribution as specification S?

$$L \equiv I_S \iff \forall G \in \mathrm{PGF}: \quad \underbrace{[\![\texttt{while}(B)\,\{P\}]\!](G)}_{L} \stackrel{?}{=} [\![I_S]\!](G)$$

If L is UAST and I_S is loop free, then by Park's lemma it suffices to prove

$$\texttt{if(B)}\ \{P \fatsemi I_S\}\ \texttt{else}\ \{\texttt{skip}\} \equiv I_S \qquad (*)$$

Approach:

- identify a syntactic fragment for which (*) is decidable
- treat quantification over infinitely many distributions by generalising PGFs to "second-order" PGFs

[Barthe et al., 2020] limit the domain of program variables to finite fields

Representing Infinite Sets of Distributions

A second-order GF has the form of a PGF whose coefficients are PGFs

Example:

$$S_{dirac} = (1 - XU)^{-1} = 1 + XU + X^2U^2 + \ldots \in \mathbb{R}[[U, X]]$$

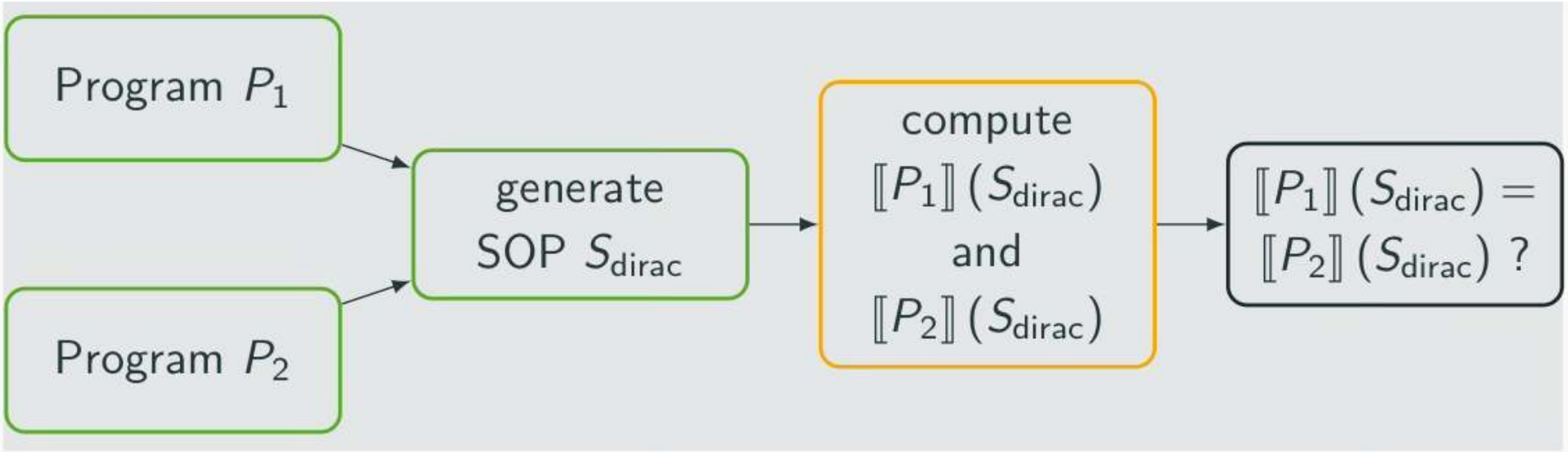
encodes all Dirac distributions $S_{dirac}(i) = X^i = \delta_i$

Then it holds:

$$\forall G \in \mathrm{PGF}.\ [\![P_1]\!](G) = [\![P_2]\!](G) \iff [\![P_1]\!](S_{dirac}) = [\![P_2]\!](S_{dirac})$$

program equivalence is about program semantics on second -order PGFs

Program Equivalence



How can we compute the semantics of P_i preserving closed-forms?

Rectangular Discrete Programs

ReDiP-program P	Semantics $[\![P]\!](G)$
$\mathrm{x} := n$	$G[X/1] \cdot X^n$
$\mathrm{x}{-}{-}$	$(G - G[X/0])X^{-1} + G[X/0]$
$\mathrm{x} \mathrel{+}= \mathrm{iid}(\mathcal{D}, y)$	$G[Y/Y\,[\![\mathcal{D}]\!]]$
$\mathrm{if}\,(\mathrm{x} < n)\,\{P_1\}\,\mathrm{else}\,\{P_2\}$	$[\![P_1]\!](G_{x<n}) + [\![P_2]\!](G - G_{x<n})$
$P_1 \fatsemi P_2$	$[\![P_2]\!]([\![P_1]\!](G))$
$\mathrm{while}\,(\mathrm{x} < n)\,\{P_1\}$	$[\mathrm{lfp}\ \Phi_{x<n,P_1}](G)$

closed form PGF

Each straight-line ReDiP program preserves rational closed forms

$$G_{x<n} = \sum_{i=0}^{n-1} \frac{1}{i!} \left(\partial_X^i G\right)[X/0] \cdot X^i$$

Toy example: Two dice roll programs

$/\!/\!/\ S_{\text{dirac}} = \frac{1}{1-DU}$

$\mathsf{d} := \texttt{uniform}(1, 6)$

$/\!/\!/\ \frac{1}{1-U} \cdot \frac{1}{6} \frac{D-D^7}{1-D}$

$/\!/\!/\ S_{\text{dirac}} = \frac{1}{1-DU}$

$\mathsf{d} := \texttt{uniform}(1, 3)$

$/\!/\!/\ \frac{1}{1-U} \cdot \frac{1}{3} \frac{D-D^4}{1-D}$

$\{\mathsf{skip}\}[1/2]\{\mathsf{d} := \mathsf{d} + 3\}$

$/\!/\!/\ \frac{1}{1-U} \cdot \frac{1}{6}\left(\frac{D-D^4}{1-D} + \frac{D^4-D^7}{1-D}\right)$

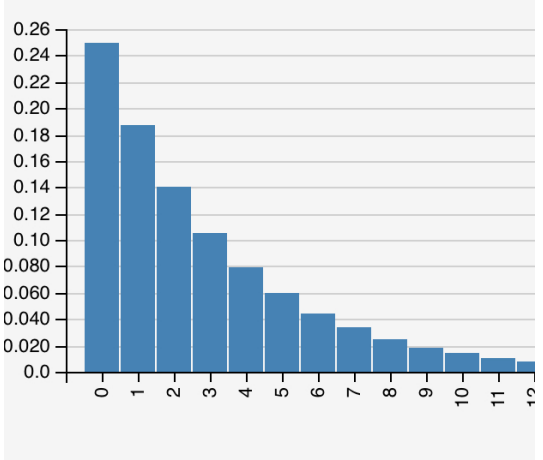
$$[\![P_{left}]\!](S_{dirac}) = [\![P_{right}]\!](S_{dirac}) \implies \underbrace{\forall G \in \mathsf{PGF} : [\![P_{left}]\!](G) = [\![P_{left}]\!](G)}_{P_{left} \equiv P_{right}}$$

A Non-ReDiP Program

[Flajolet et al., 2011]

```
x := geometric(1/4);
y := geometric(1/4);
t := x + y;
{ t := t + 1 } [5/9] { skip };
r := 1;
repeat 3 times {
    s := iid(bernoulli(1/2), 2t);
    if (s ≠ t) { r := 0 }
}
```

non rectangular



$$\Pr\{r = 1\} = \pi^{-1}$$

Example: Verifying A Sampler

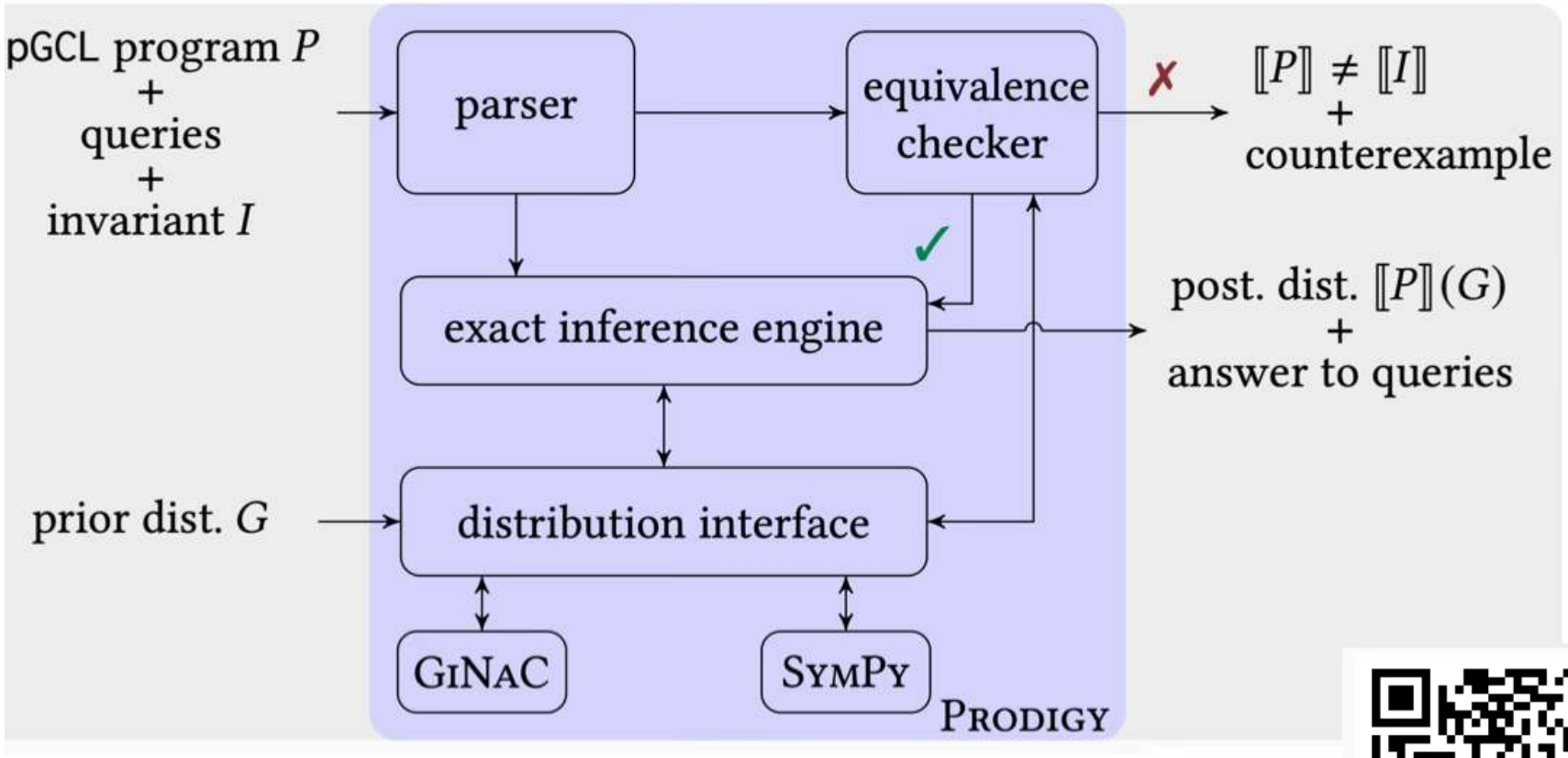
$$\underbrace{\texttt{while(n} > 0)\{\ \{\texttt{n := n - 1}\}[1/2]\{\texttt{c := c + 1}\}\ \}}_{L} \quad \underbrace{\texttt{c += iid(geometric(1/2), n)}\,;\ \texttt{n := 0}}_{I}$$

if $(n > 0)\{\ (n := n - 1\,[1/2]\,c := c + 1)\,;\ I\ \}$ else skip $\equiv\ I$?

proven using second-order PGF $\quad G_{N,C} = (1 - NU)^{-1}(1 - CV)^{-1}$

This validates the correctness of the sampler *L*

Our Tool ProDiGy



PRObability **DI**stributions via **G**eneratingfunctionolog**Y**

Joost-Pieter Katoen

A Nested Sampler

```
while(x > 0) {
    y := 1 ⨾
    while(y = 1) {
        {y := 0}[1/2]{x := x + 1} } ⨾
    x := x - 1 ⨾
    c += 1 ⨾
}
```

Nested sampler L

- prove equivalence by an innermost-first strategy
- infinite expected run-time and algebraic PGF
- ProDiGy verifies this < 30ms

```
if(y = 1) {
    x += geometric(1/2) ⨾
    y := 0 ⨾
}
```

Inner invariant I_{in}

```
if(x > 0) {
    c := iid(catalan(1/2), x) ⨾
    x := 0 ⨾
    y := 0
}
```

Outer invariant I_{out}

$y = 1$ is equivalent to $0 < y$ and $y < 2$, as y is a natural

GENERATING FUNCTIONS MEET OCCUPATION TIMES

Joost-Pieter Katoen

Occupation Measures

Given probabilistic loop $\mathsf{while}\,(B)\ \{P\}$ and input distribution G

Goal: Automatically infer output distribution $[\![\mathsf{while}\,(B)\ \{P\}]\!]\,(G)$

1: capture semantics by means of occupation measures

[Sharir et al., 1984]

2: use loop invariants to approximate output distribution (as before)

[Park, 1969]

3: use template-based invariant synthesis for closed-form PGFs

What is an Occupation Measure?

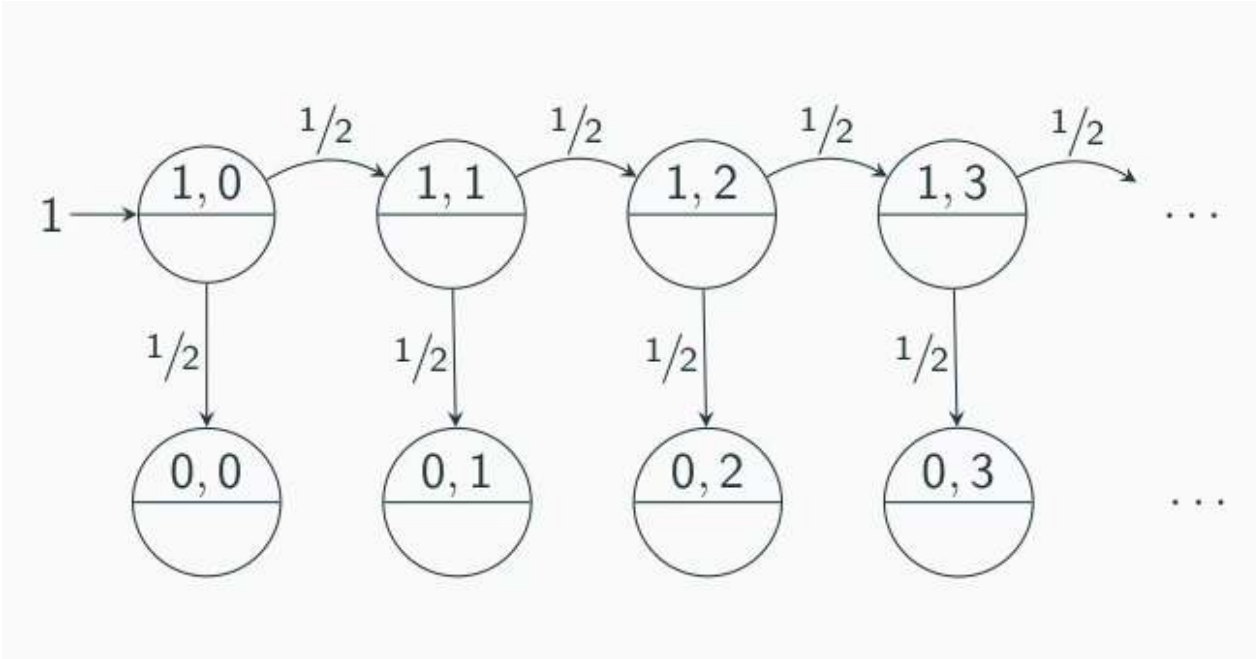
$/\!/\!/\ G = 1X^1C^0$

while $(x = 1)\{$

$\quad \{x := 0\}[1/2]\{c := c+1\}$

$\}$

$/\!/\!/\ \frac{1}{2} + \frac{1}{4}C + \frac{1}{8}C^2 + \ldots$



occupation measure of a state σ = expected number of visits to σ

What is an Occupation Measure?

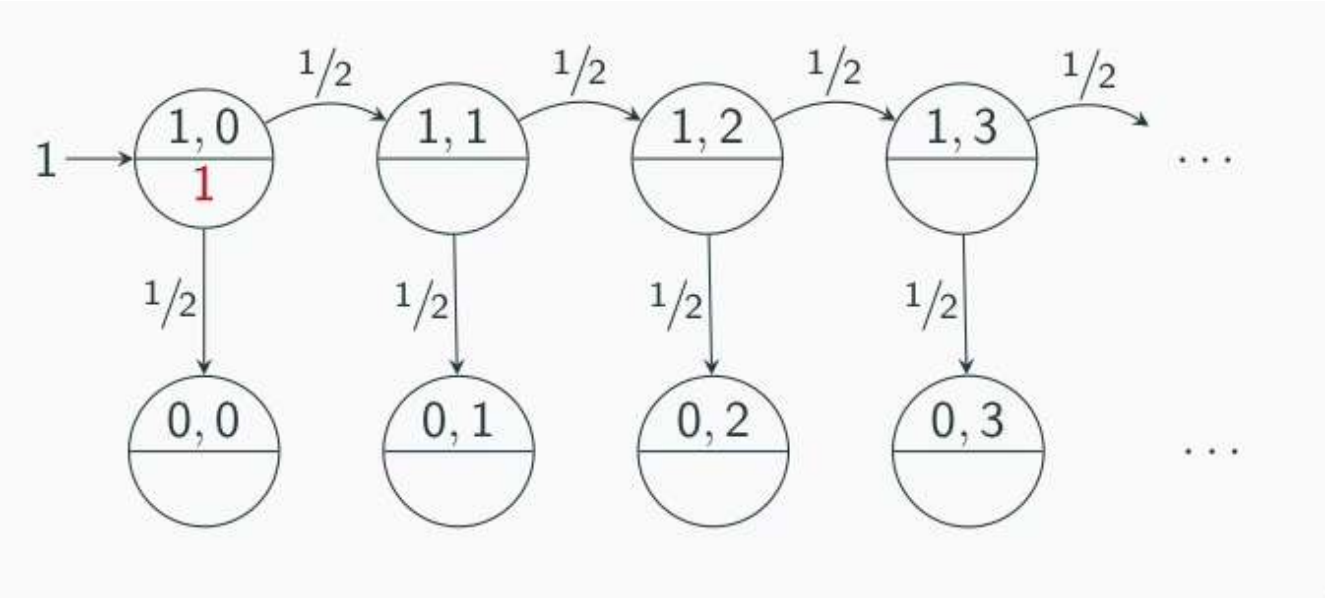
$/\!/\!/\ G = 1X^1C^0$

while $(x = 1)\{$

$\quad \{x := 0\}[1/2]\{c := c+1\}$

$\}$

$/\!/\!/\ \frac{1}{2} + \frac{1}{4}C + \frac{1}{8}C^2 + \ldots$



occupation measure of a state σ = expected number of visits to σ

What is an Occupation Measure?

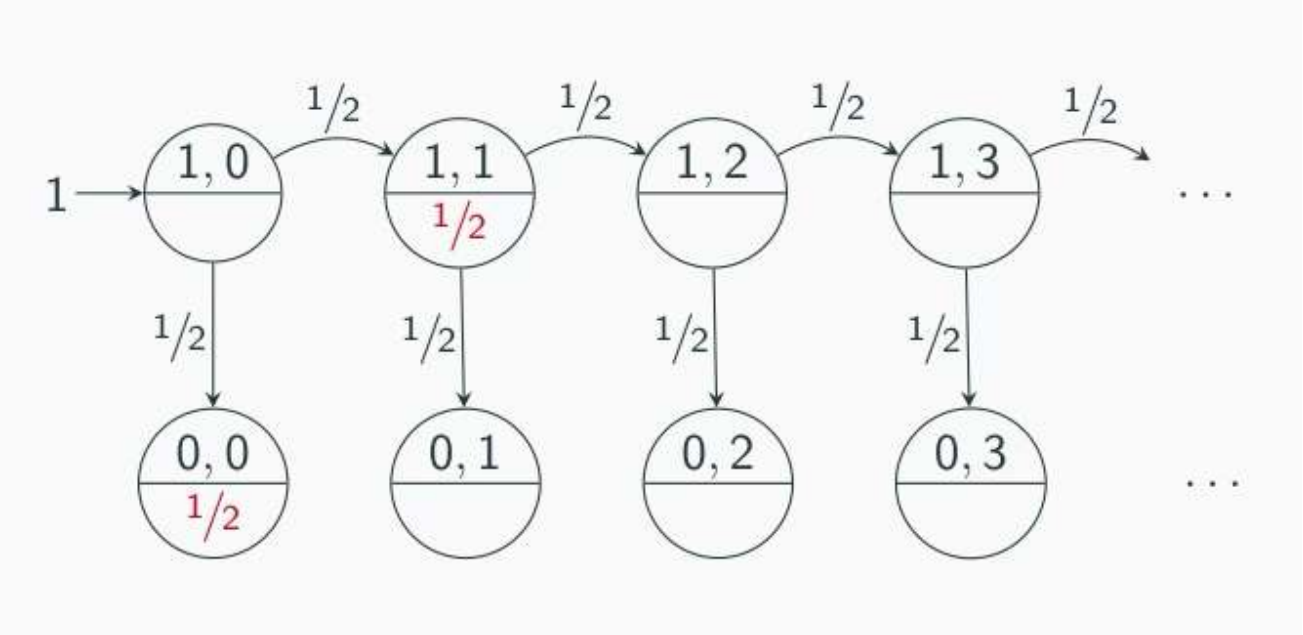
$/\!/\!/\ G = 1X^1C^0$

while $(x = 1)\{$

$\quad \{x := 0\}[1/2]\{c := c+1\}$

$\}$

$/\!/\!/\ \frac{1}{2} + \frac{1}{4}C + \frac{1}{8}C^2 + \ldots$



occupation measure of a state σ = expected number of visits to σ

What is an Occupation Measure?

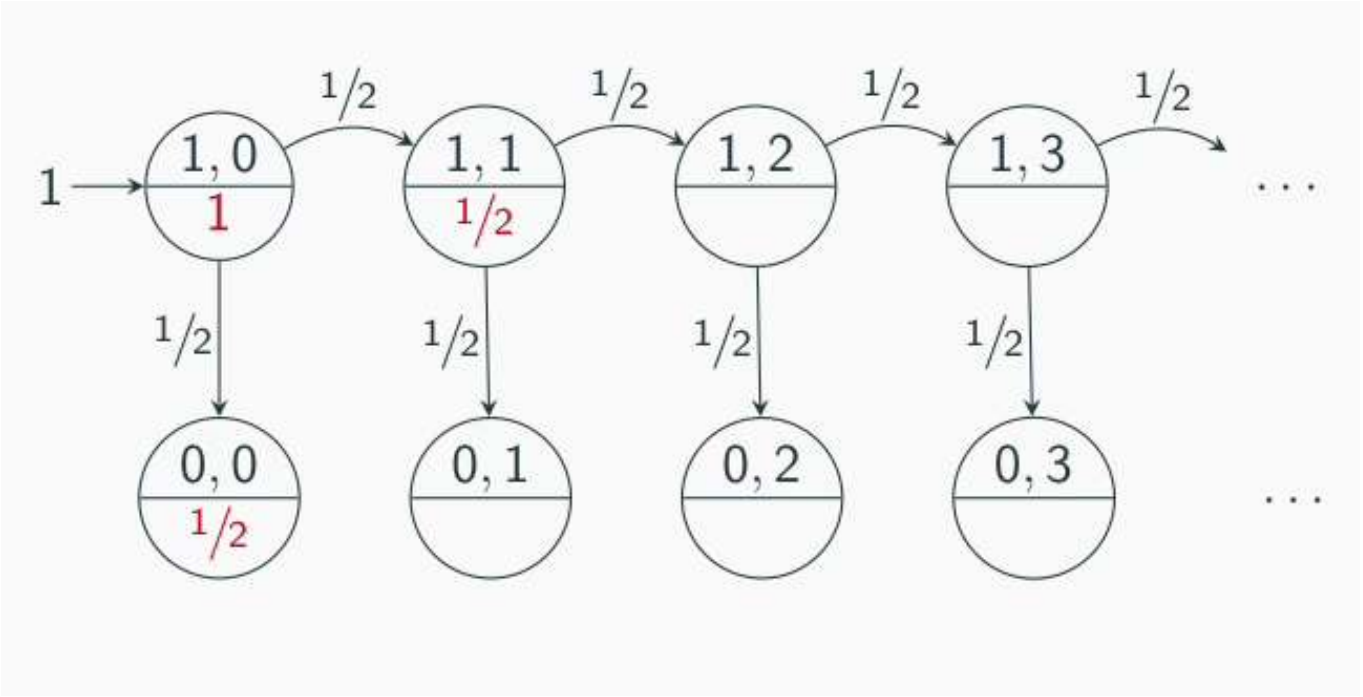
$/\!/\!/\ G = 1X^1C^0$

while $(x = 1)\{$

$\quad \{x := 0\}[1/2]\{c := c+1\}$

$\}$

$/\!/\!/\ \frac{1}{2} + \frac{1}{4}C + \frac{1}{8}C^2 + \ldots$



occupation measure of a state σ = expected number of visits to σ

What is an Occupation Measure?

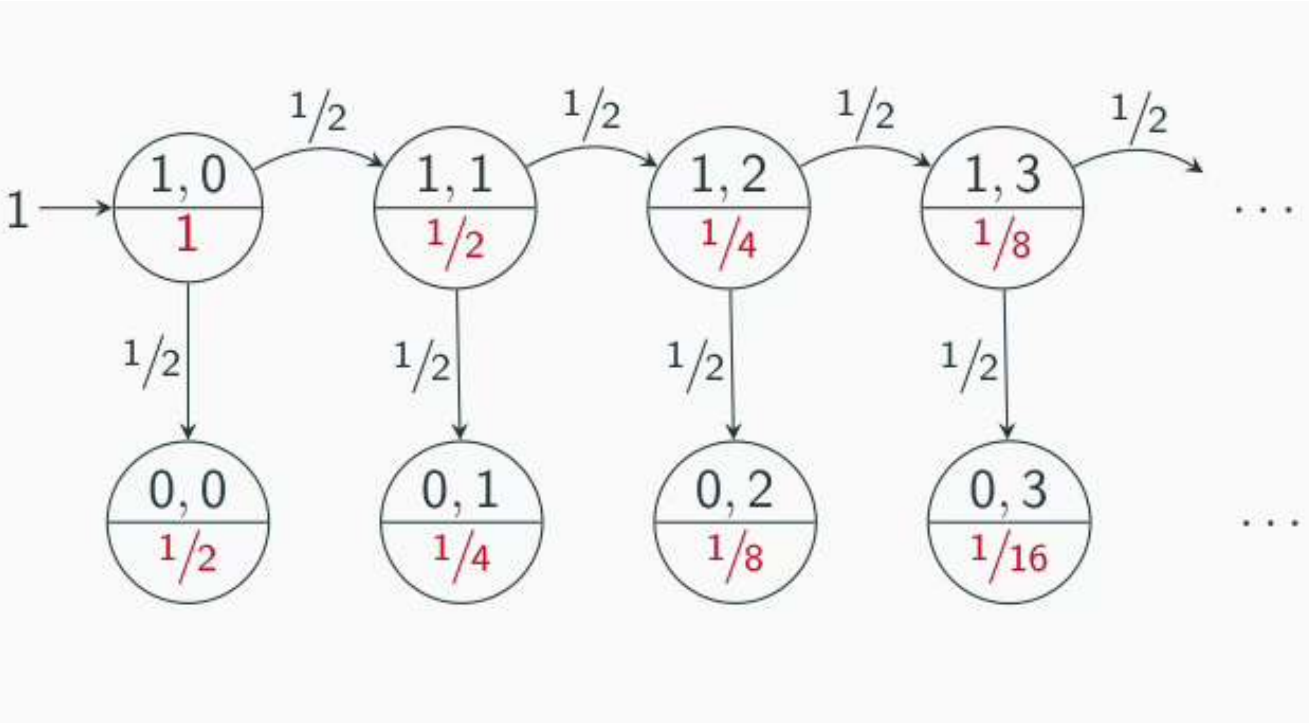
$/\!/\!/\ G = 1X^1C^0$

while $(x = 1)\{$

$\quad \{x := 0\}[1/2]\{c := c+1\}$

$\}$

$/\!/\!/\ \frac{1}{2} + \frac{1}{4}C + \frac{1}{8}C^2 + \ldots$



occupation measure of a state σ = expected number of visits to σ

Occupation Measures versus Reachability Probabilities

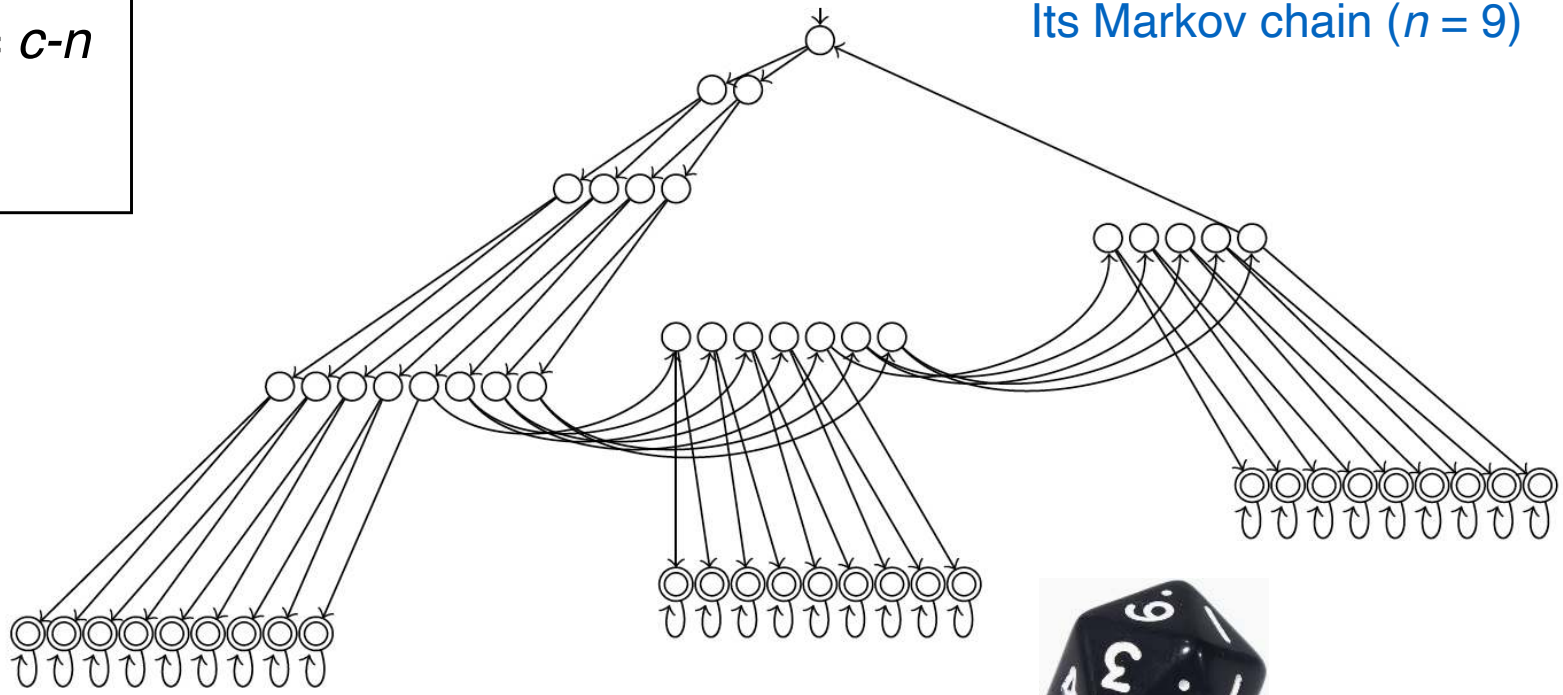
[Lumbroso, 2013]

```
v := 1;  c := 0;
while (v < n) {
   v := 2v;
   c := 2c [1/2] c := 2c+1;
   if (c ⩾ n) {
         v := v-n;  c := c-n
   }
}
```

$$\Pr(\Diamond\, 0) = \frac{1}{9} \wedge \ldots \wedge \Pr(\Diamond\, 8) = \frac{1}{9}$$

$\Longrightarrow$ solve 9 linear equation systems

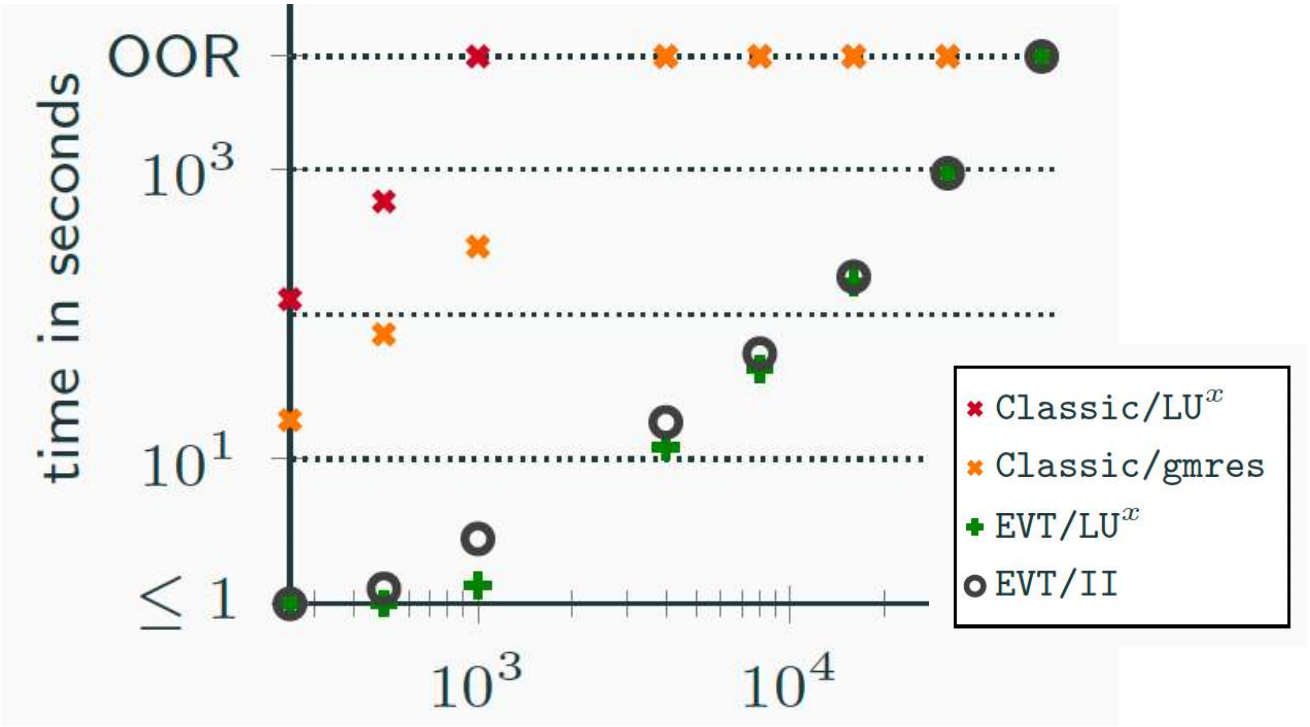
Its Markov chain ($n = 9$)



Occupation Measures vs. Reachability Probabilities

[Mertens *et al*, JAR 2025]

Storm
www.stormchecker.org



Our solution: solve **1** linear equation system

Simpler Fixed Points by Occupation Measures

$\Phi_{B,P}\colon (\mathrm{PGF} \to \mathrm{PGF}) \to (\mathrm{PGF} \to \mathrm{PGF})$

Variant	$[\![\mathsf{while}\,(B)\ \{P\}]\!]\,(G)$
Kozen	$\underbrace{\mathsf{lfp}\,\varphi.\ [\lambda F.\ \langle F\rangle_{\neg B}\ +\ \varphi\,(\ [\![P]\!]\,(\langle F\rangle_{B})\ )]}_{\text{a measure transformer (FPS} \to \text{FPS)}}(G)$
Sharir et al.	$\underbrace{\langle\ \mathsf{lfp}\,F.\ G\ +\ [\![P]\!]\,(\langle F\rangle_{B})\ \rangle_{\neg B}}_{\text{a measure (FPS)}}$

Kozen's backward wp semantics = forward occupation measure semantics

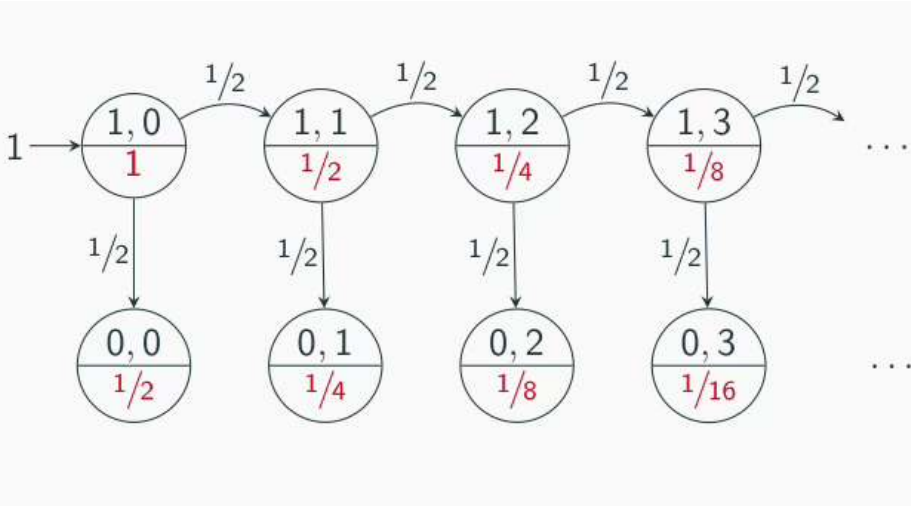
[Sharir et al., 1984] proved this on Markov chains; we inductively on program syntax

Expected Runtimes

$$\langle \text{lfp}\, F \,.\, \underbrace{G \;+\; [\![\, P \,]\!]\big(\langle F \rangle_B\big)}_{=\Phi} \rangle_{\neg B}$$

$$\big(\text{lfp}\ \Phi\big)_{loop}(X^\sigma) \;=\; \text{OM}_{MC[\![\, loop \,]\!]}(\sigma)$$

```
/// G = 1X^1C^0
while (x = 1){ {x := 0}[1/2]{c := c+1} }
```



Then it follows

$$\big|\big(\text{lfp}\ \Phi\big)(X^\sigma)\big| \;=\; \text{ert}(\sigma)$$

$$\big|\big(\text{lfp}\ \Phi\big)(X^\sigma)\big| \;<\; \infty \quad \text{iff} \quad \text{while}\ (B)\{\, P \,\}\ \text{is PAST on input}\ \sigma$$

deciding PAST is Σ_2^0 complete

Template-Based Invariant Synthesis

Construct a template (= PGF with symbolic coefficients)

- geometric distribution for c on termination
- scaled geometric distribution while running loop

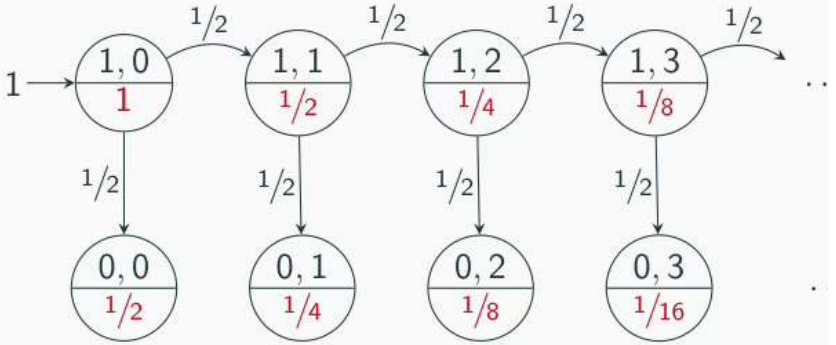
$$I[a_1, b_1, a_2, b_2] \;=\; a_1 \cdot X^0 \frac{1}{1 - b_1 \cdot C} + a_2 \cdot X^1 \frac{1}{1 - b_2 \cdot C}$$

```
/// G = 1X^1C^0
while (x = 1){
    {x := 0}[1/2]{c := c+1}
}
```

Find a loop invariant

- solving $\Phi(I) \;=\; I$ yields $\tau = (1/2, 1/2, 1, 1/2)$
- $I[\tau](x \neq 1) \;= 1 \;\geqslant 0.$

Thus: $[\![P]\!](G) \;=\; \frac{1}{2} \cdot X^0 \frac{1}{1 - \frac{1}{2} \cdot C}$



Template-Based Invariant Synthesis

Let while $(B)\{\ P\ \}$ with *P* a closed-form preserving body and *F* a closed-form PGF

- pick a template $I[\,\cdot\,] \;=\; \dfrac{G}{H}$
  - manual specification, or
  - enumerate closed forms by increasing degrees
  - ……
- find an invariant: solve $\Phi(I[\tau]) \;=\; I[\tau]$ by comparing polynomials in $\mathbb{R}[X]$
- check positivity $I[\tau]) \;\geqslant\; 0$ heuristically
  - geometric fractions
  - checking dominant roots
  - ……

decidability is open
[Ouaknine & Worrell, 2014]

implemented in ProDiGy

Invariant Synthesis with ProDiGy

Benchmark	Time (s)	Auto/User	Additional Comments
subdist_enter	0.568013	A	
cond_and	0.017394	A	
faulty_decrement	0.120409	A	
geometric	0.026373	A	
geometric_counter	0.124910	A	
modulo_geometric	0.035103	U	$a = 4f, b = 2f, c = f, d = 8f,$ $e = 6f$
thirds_geometric	6.681842	A	Used Z3 to solve
random_walk	0.118271	A	
random_walk_counter	0.106519	U	$a = 1, b = 1, d = -1$
fast_dice_roller	0.155679	U	$w_0 = \frac{1}{3}, w_1 = 0, w_2 = \frac{2}{3}, w_3 = 1,$ $w_4 = \frac{1}{6}, w_5 = 0, w_6 = 0, w_m = 0$
nontermination	Failed	A	Actual invariant is $\infty \cdot X + \frac{1}{2} \cdot X^2$
sequential_loops	Failed	A	Cannot determine pos. of $\frac{2C}{C^2-3C+2} + 1$

Lumbroso's Sampler

Construct a template

observe that $c \sim \text{unif}(0, \min(v, n) - 1)$

$$\sum_{i<n} a_i \cdot F^0 V^i \frac{1 - C^i}{1 - C} \; + \; \sum_{n \leqslant i < 2n} a_i \cdot F^1 V^i \frac{1 - C^n}{1 - C}$$

```
f := 0, v := 1;  c := 0;
while (f = 0) {
   v := 2v;
   c := 2c [1/2] c := 2c+1;
   if (n ⩽ v) {
      if (c < n) {
         f := 1 }
      else {
         v := v-n;  c := c-n
      }
   }
}
```

Find a loop invariant

solving $\Phi(I) \;=\; I$ yields τ with $\displaystyle\sum_{n \leqslant i < 2n} \tau(a_i) = \frac{1}{n}$ and $|I[\tau]| < \infty$

Marginalising out v and f yields: $\quad [\![P]\!](G)[V/1, F/1] \;=\; \dfrac{1}{N} \cdot \dfrac{1 - C^n}{1 - C}$

RELATED WORK AND EPILOGUE

Related Work

- Formal power series in coinductive algebraic semantics [Boreale & Gadducci, TCS 2006]
- Inference by automatic differentation [Zaiser, Murawski & Ong, NeuRIPS 2023] [Zaiser, Murawski & Ong, POPL 2025]
- PGFs as intermediate compilation target for exact inference [Li & Zhang, OOPSLA 2025]
- PGFs and weighted automata [Geissler & Winkler, ICTAC 2025]
- Weakest-precondition reasoning [McIver & Morgan, 2005] [Kaminski , K. et al., 2016]
- Computer algebra for probabilistic reasoning [Moosbrugger, Bartocci & Kovacs, 2024]
- Hoare logic on distributions [Barthe et al., ESOP 2018] [Zilken, K. et al., FM 2026]

PGFs for Probabilistic Programs: Pros and Cons

- provide the full picture
- rich quantitative querying
- compactness
- algebraic compositionality
- symbolic parameters
- adequate for diverging loops

- dependence on closed forms
- no continuous probabilities
- scalability trade-offs
- unknown decidability
- nondeterminism?

Epilogue

- Program semantics using PGFs
  - forward transformer agrees wirh Kozen's semantics
  - semantic characterisation of loops with closed-form PGFs
  - verifying loop invariants := algebraic manipulation
- Program equivalence using PGFs
  - syntactic program fragment with closed-form PGFs
  - "second-order" PGFs for families of distributions
  - extension to exact Bayesian inference (not covered)
- Invariant synthesis using PGFs and occupation measures
  - simpler fixed point characterisation
  - simpler loop invariant synthesis
  - connection to expected runtimes

Literature

- Program semantics using PGFs [Klinkenberg et al, LOPSTR 2020]
- Program equivalence using PGFs [Chen et al, CAV 2022]
- PGFs for exact Bayesian inference [Chen et al., OOPSLA 2024]
- PGFs meet occupation measures [Haase et al., ESOP 2026]
- PGFs for probabilistic programs [Klinkenberg, PhD Dissertation 2025]

ProDiGy Tool

My Co-Authors

Kevin Batz (Cornell)

Mingshuai Chen (Zhejiang)

Adrian Gallus (Aachen)

Darion Haase (RWTH)

Benjamin Kaminski (Saarland/UCL)

Lutz Klinkenberg (RWTH)

Joshua Moerman (Open University)

Tobias Winkler (RWTH)