How long, O Bayesian network, will I sample thee?

A program analysis perspective on expected sampling times

Kevin Batz **Benjamin Lucien Kaminski**

Joost-Pieter Katoen Christoph Matheja

Research Training Group – Uncertainty and Randomness in Algorithms, Verification, and Logic

Software Modeling and Verification Chair

RWTH AACHEN UNIVERSITY

27th European Symposium on Programming

21st edition of the European Joint Conferences on Theory & Practice of Software

April 16, 2018, Thessaloniki, Greece

A program analysis perspective

Probabilistic Programs

$$\{\, x := 7 \,\}\ [1/3]\ \{\, x := 2 \,\}\,;$$

$$\texttt{if}\ (\, x > 5 \,)\ \{$$

$$\quad \texttt{skip}\,;\ \texttt{skip}\,;\ \texttt{skip}\,;\ \texttt{skip}$$

$$\}\,\texttt{else}\,\{$$

$$\quad \texttt{skip}\ \}$$

Probabilistic Programs

What does a probabilistic program C do?

$$\{\, x := 7 \,\}\ [1/3]\ \{\, x := 2 \,\};$$

```
if ( x > 5 ) {
    skip; skip; skip; skip
} else {
    skip }
```

Probabilistic Programs

What does a probabilistic program C do?

- Run program C on initial state σ

$$\{\, x := 7 \,\}\ [1/3]\ \{\, x := 2 \,\};$$

```
if (x > 5) {
    skip; skip; skip; skip
} else {
    skip }
```

Probabilistic Programs

What does a probabilistic program C do?

- Run program C on initial state σ
- Obtain a distribution over final states

$\{\, x := 7 \,\}\ [1/3]\ \{\, x := 2 \,\}$;

```
if (x > 5) {
    skip; skip; skip; skip
} else {
    skip }
```

Probabilistic Programs

What does a probabilistic program C do?

- Run program C on initial state σ
- Obtain a (sub–)distribution over final states

$$\{\, x := 7 \,\}\ [1/3]\ \{\, x := 2 \,\};$$

```
if (x > 5) {
    skip; skip; skip; skip
} else {
    skip }
```

Probabilistic Programs

What does a probabilistic program C do?

- Run program C on initial state σ
- Obtain a (sub–)distribution over final states

What is the runtime of C on input σ?

$$\{\, x := 7 \,\}\ [1/3]\ \{\, x := 2 \,\};$$

$$\begin{aligned}&\texttt{if}\ (\, x > 5 \,)\ \{\\ &\quad \texttt{skip}; \texttt{skip}; \texttt{skip}; \texttt{skip}\\ &\}\,\texttt{else}\,\{\\ &\quad \texttt{skip}\ \}\end{aligned}$$

Probabilistic Programs

What does a probabilistic program C do?

- Run program C on initial state σ
- Obtain a (sub–)distribution over final states

$$\{\, x := 7 \,\}\ [1/3]\ \{\, x := 2 \,\};$$

```
if (x > 5) {
    skip; skip; skip; skip
} else {
    skip }
```

What is the runtime of C on input σ?

- Behavior of C not entirely determined by σ

Probabilistic Programs

What does a probabilistic program C do?

- Run program C on initial state σ
- Obtain a (sub–)distribution over final states

$$\{\, x := 7 \,\}\ [1/3]\ \{\, x := 2 \,\};$$

```
if (x > 5) {
    skip; skip; skip; skip
} else {
    skip }
```

What is the runtime of C on input σ?

- Behavior of C not entirely determined by σ
- Probabilistic nature of C influences its runtime

Probabilistic Programs

What does a probabilistic program C do?

- Run program C on initial state σ
- Obtain a (sub–)distribution over final states

$$\{\, x := 7 \,\}\ [1/3]\ \{\, x := 2 \,\};$$

```
if (x > 5) {
    skip; skip; skip; skip
} else {
    skip }
```

What is the runtime of C on input σ?

- Behavior of C not entirely determined by σ
- Probabilistic nature of C influences its runtime

Better Question:

What is the expected runtime of C on input σ?

The ert Transformer [ESOP'16, LICS'16, JACM'18]

- Runtimes (random variables): $\mathbb{T} = \left\{ t \;\middle|\; t\colon \mathsf{States} \to \mathbb{R}^{\infty}_{\geq 0} \right\}$.

The ert Transformer [ESOP'16, LICS'16, JACM'18]

- Runtimes (random variables): $\mathbb{T} = \left\{ t \;\middle|\; t\colon \text{States} \to \mathbb{R}^{\infty}_{\geq 0} \right\}$.
- Use a continuation–passing style transformer $\text{ert}[C]\colon \; \mathbb{T} \to \mathbb{T}$.

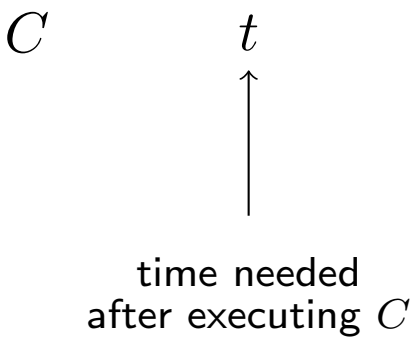
The ert Transformer [ESOP'16, LICS'16, JACM'18]

- Runtimes (random variables): $\mathbb{T} = \left\{ t \;\middle|\; t\colon \text{States} \to \mathbb{R}^{\infty}_{\geq 0} \right\}$.
- Use a continuation–passing style transformer $\text{ert}[C]\colon \; \mathbb{T} \to \mathbb{T}$.

$$C$$

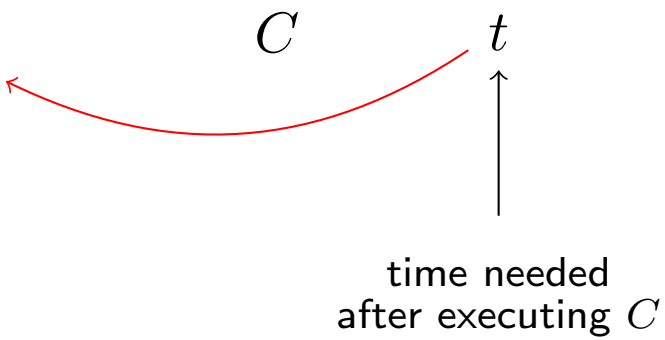
The ert Transformer [ESOP'16, LICS'16, JACM'18]

- Runtimes (random variables): $\mathbb{T} = \left\{ t \;\middle|\; t\colon \text{States} \to \mathbb{R}^{\infty}_{\geq 0} \right\}$.
- Use a continuation–passing style transformer $\text{ert}[C]\colon \; \mathbb{T} \to \mathbb{T}$.



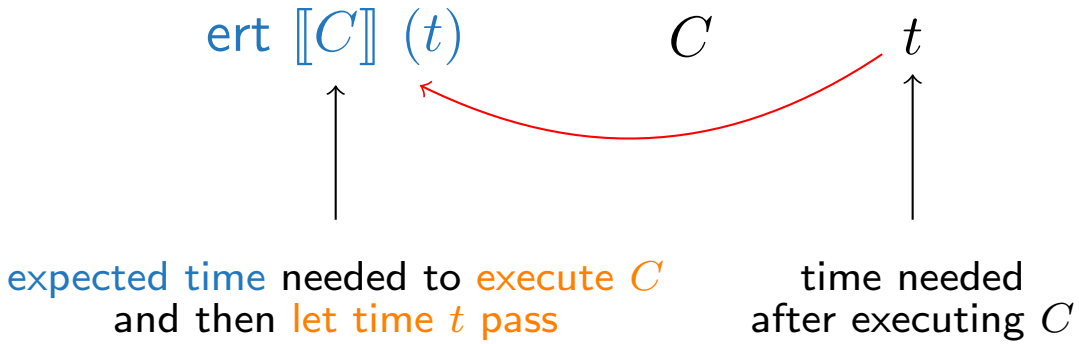
The ert Transformer [ESOP'16, LICS'16, JACM'18]

- Runtimes (random variables): $\mathbb{T} = \left\{ t \;\middle|\; t\colon \text{States} \to \mathbb{R}^{\infty}_{\geq 0} \right\}$.
- Use a continuation–passing style transformer $\text{ert}[C]\colon\ \mathbb{T} \to \mathbb{T}$.



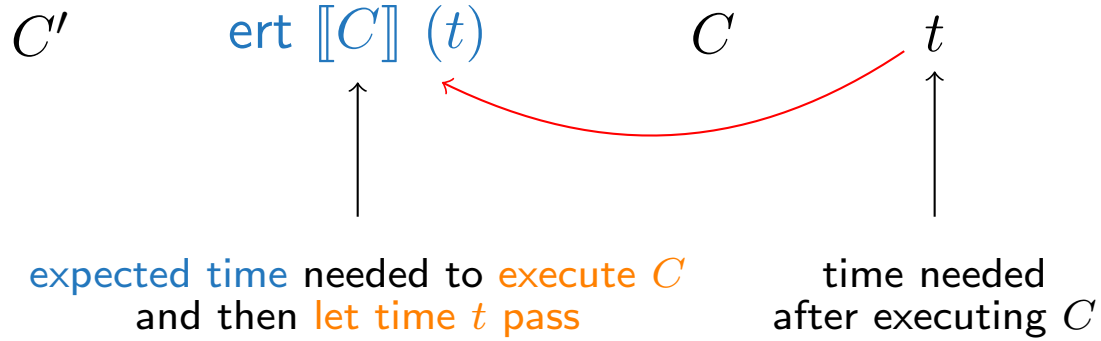
The ert Transformer [ESOP'16, LICS'16, JACM'18]

- Runtimes (random variables): $\mathbb{T} = \left\{ t \;\middle|\; t\colon \text{States} \to \mathbb{R}^{\infty}_{\geq 0} \right\}$.
- Use a continuation–passing style transformer $\text{ert}[C]\colon \; \mathbb{T} \to \mathbb{T}$.



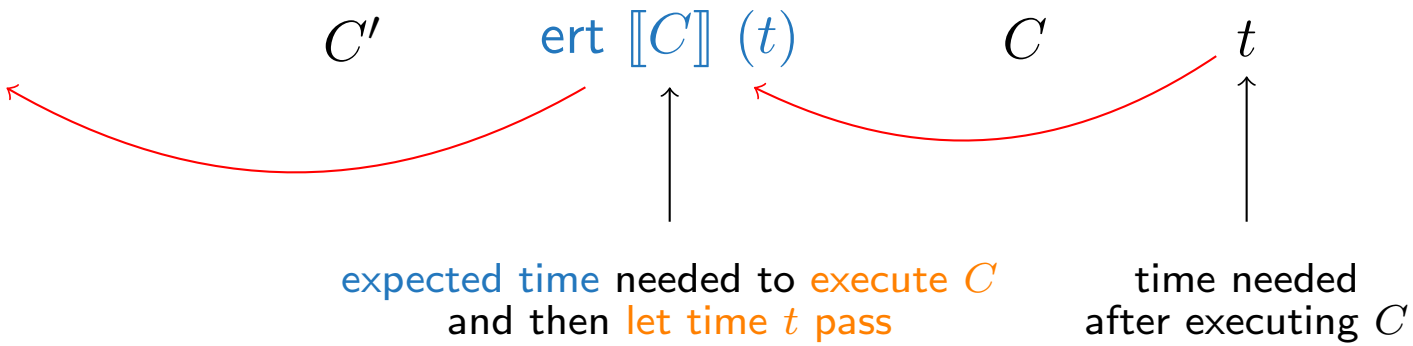
The ert Transformer [ESOP'16, LICS'16, JACM'18]

- Runtimes (random variables): $\mathbb{T} = \left\{ t \;\middle|\; t\colon \text{States} \to \mathbb{R}^{\infty}_{\geq 0} \right\}$.
- Use a continuation–passing style transformer $\text{ert}[C]\colon \; \mathbb{T} \to \mathbb{T}$.



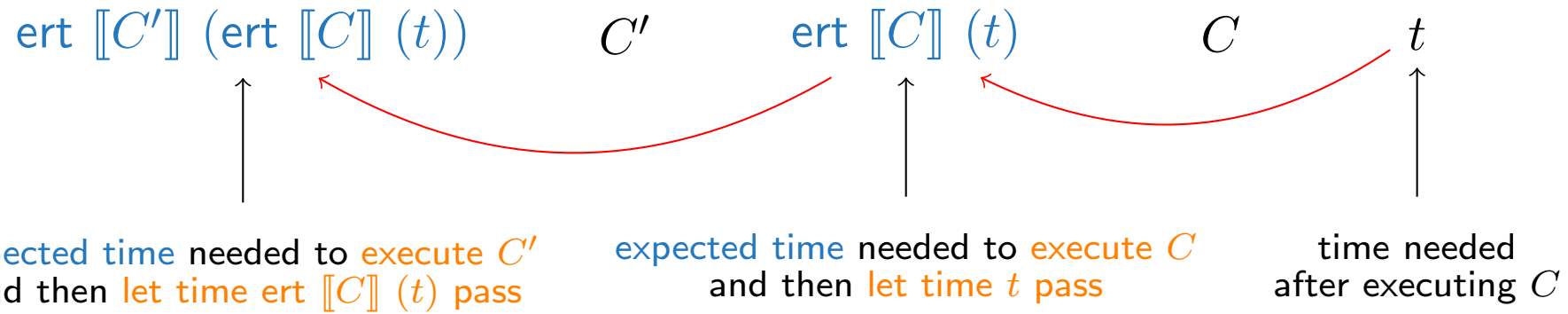
The ert Transformer [ESOP'16, LICS'16, JACM'18]

- Runtimes (random variables): $\mathbb{T} = \left\{ t \;\middle|\; t\colon \text{States} \to \mathbb{R}^{\infty}_{\geq 0} \right\}$.
- Use a continuation–passing style transformer $\text{ert}[C]\colon\ \mathbb{T} \to \mathbb{T}$.



The ert Transformer [ESOP'16, LICS'16, JACM'18]

- Runtimes (random variables): $\mathbb{T} = \left\{ t \;\middle|\; t\colon \text{States} \to \mathbb{R}^{\infty}_{\geq 0} \right\}$.
- Use a continuation–passing style transformer $\text{ert}[C]\colon\ \mathbb{T} \to \mathbb{T}$.



The ert Transformer [ESOP'16, LICS'16, JACM'18]

- Runtimes (random variables): $\mathbb{T} = \left\{ t \;\middle|\; t\colon \text{States} \to \mathbb{R}^{\infty}_{\geq 0} \right\}$.
- Use a continuation–passing style transformer $\text{ert}[C]\colon \; \mathbb{T} \to \mathbb{T}$.

expected time needed to execute C'
and then let time ert $[\![C]\!]$ (t) pass

in other words:

expected time needed to execute C',
then execute C
and then let time t pass

expected time needed to execute C
and then let time t pass

time needed
after executing C

$\{\, x := 7 \,\}\ [1/3]\ \{\, x := 2 \,\}\,;$

$\texttt{if}\ (\, x > 5 \,)\ \{$

$\quad \texttt{skip}\,;\ \texttt{skip}\,;\ \texttt{skip}\,;\ \texttt{skip}$

$\}\,\texttt{else}\,\{$

$\quad \texttt{skip}$

$\}\,;$

$\texttt{skip}$

$\{\, x := 7 \,\}\ [1/3]\ \{\, x := 2 \,\}$;

```
if (x > 5) {
    skip; skip; skip; skip
} else {
    skip
};
skip
```

// **0**

$\{\, x := 7 \,\}\ [1/3]\ \{\, x := 2 \,\}$;

if $(\, x > 5 \,)$ {

    skip; skip; skip; skip

} else {

    skip

};
// 1
skip
// 0

$\{\, x := 7 \,\}\ [1/3]\ \{\, x := 2 \,\}$;

```
if (x > 5) {

    skip; skip; skip; skip
    /// 1
}else{

    skip
    /// 1
};
/// 1
skip
/// 0
```

$\{\, x := 7 \,\}\ [1/3]\ \{\, x := 2 \,\}$;

```
if (x > 5) {

    skip; skip; skip; skip
    /// 1
}else{
    /// 2
    skip
    /// 1
};
/// 1
skip
/// 0
```

$\{\, x := 7 \,\}\ [1/3]\ \{\, x := 2 \,\}$;

```
if (x > 5) {
    /// 5
    skip; skip; skip; skip
    /// 1
}else{
    /// 2
    skip
    /// 1
};
/// 1
skip
/// 0
```

$\{\, x := 7 \,\}\ [1/3]\ \{\, x := 2 \,\}$;
// $\mathbf{1 \;+\; [x > 5] \cdot 5 \;+\; [x \leq 5] \cdot 2}$
if $(\, x > 5 \,)$ {
    // $\mathbf{5}$
    skip; skip; skip; skip
    // $\mathbf{1}$
} else {
    // $\mathbf{2}$
    skip
    // $\mathbf{1}$
};
// $\mathbf{1}$
skip
// $\mathbf{0}$

```
/// 2 + 1/3 · 5 + 2/3 · 2
{ x := 7 } [1/3] { x := 2 };
/// 1 + [x > 5] · 5 + [x ≤ 5] · 2
if ( x > 5 ) {
    /// 5
    skip; skip; skip; skip
    /// 1
} else {
    /// 2
    skip
    /// 1
};
/// 1
skip
/// 0
```

```
/// 5
{ x := 7 } [1/3] { x := 2 };
/// 1 + [x > 5] · 5 + [x ≤ 5] · 2
if ( x > 5 ) {
    /// 5
    skip; skip; skip; skip
    /// 1
} else {
    /// 2
    skip
    /// 1
};
/// 1
skip
/// 0
```

Expected Runtimes of Loops

Expected Runtimes of Loops

- Recall $\mathbb{T} = \left\{ t \;\middle|\; t \colon \text{States} \to \mathbb{R}^{\infty}_{\geq 0} \right\}$

Expected Runtimes of Loops

- Recall $\mathbb{T} = \left\{ t \;\middle|\; t\colon \text{States} \to \mathbb{R}^{\infty}_{\geq 0} \right\}$
- $(\mathbb{T}, \preceq)$ is a complete lattice with

$$s \preceq t \qquad \text{iff} \qquad \forall\, \sigma \in \text{States}\colon \quad s(\sigma) \;\leq\; t(\sigma)$$

Expected Runtimes of Loops

- Recall $\mathbb{T} = \{\, t \mid t\colon \text{States} \to \mathbb{R}^{\infty}_{\geq 0} \,\}$
- $(\mathbb{T}, \preceq)$ is a complete lattice with

$$s \preceq t \qquad \text{iff} \qquad \forall\, \sigma \in \text{States}\colon \quad s(\sigma) \;\leq\; t(\sigma)$$

Expected runtime of $\texttt{while}\,(\varphi)\,\{C\}$ and then let time t pass

Expected Runtimes of Loops

- Recall $\mathbb{T} = \left\{ t \;\middle|\; t\colon \text{States} \to \mathbb{R}^{\infty}_{\geq 0} \right\}$
- $(\mathbb{T}, \preceq)$ is a complete lattice with

$$s \preceq t \qquad \text{iff} \qquad \forall\, \sigma \in \text{States}\colon \quad s(\sigma) \;\leq\; t(\sigma)$$

Expected runtime of $\texttt{while}\,(\varphi)\,\{C\}$ and then let time t pass

Use a least fixed–point construct: Let

$$\Phi_t(X) \;=\; 1 \;+\; [\neg\varphi] \cdot t \;+\; [\varphi] \cdot \text{ert}\,[\![C]\!]\,(X) \;\;.$$

Expected Runtimes of Loops

- Recall $\mathbb{T} = \left\{ t \;\middle|\; t\colon \text{States} \to \mathbb{R}^{\infty}_{\geq 0} \right\}$
- $(\mathbb{T}, \preceq)$ is a complete lattice with

$$s \preceq t \qquad \text{iff} \qquad \forall\, \sigma \in \text{States}\colon \quad s(\sigma) \;\leq\; t(\sigma)$$

Expected runtime of $\texttt{while}\,(\varphi)\,\{C\}$ and then let time t pass

Use a least fixed–point construct: Let

$$\Phi_t(X) \;=\; 1 \;+\; [\neg\varphi]\cdot t \;+\; [\varphi]\cdot \text{ert}\,[\![C]\!]\,(X) \;\;.$$

Then $\quad \text{ert}\,[\![\texttt{while}\,(\varphi)\,\{C\}]\!]\,(t) \;=\; \text{lfp}\;\; \Phi_t \;\;.$

What is the expected runtime of $\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}$?

Let $\Phi_t(X) \;=\; 1 \;+\; [\neg\varphi]\cdot t \;+\; [\varphi]\cdot \mathsf{ert}\,[\![C]\!]\,(X)$.

Then $\quad \mathsf{ert}\,[\![\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}]\!]\,(t) \;=\; \mathsf{lfp}\;\;\Phi_t$.

What is the expected runtime of $\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}$?

Let $\Phi_t(X) \;=\; 1 \;+\; [\neg\varphi]\cdot t \;+\; [\varphi]\cdot \mathsf{ert}\,[\![C]\!]\,(X)$.

Then $\quad \mathsf{ert}\,[\![\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}]\!]\,(t) \;=\; \mathsf{lfp}\;\;\Phi_t$.

Induction Rule for Upper Bounds

What is the expected runtime of $\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}$?

Let $\Phi_t(X) \;=\; 1 \;+\; [\neg\varphi]\cdot t \;+\; [\varphi]\cdot \mathsf{ert}\,[\![C]\!]\,(X)$.

Then $\quad \mathsf{ert}\,[\![\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}]\!]\,(t) \;=\; \mathsf{lfp}\;\; \Phi_t$.

Induction Rule for Upper Bounds

Choose $I \in T$.

What is the expected runtime of $\texttt{while}\,(\varphi)\,\{C\}$?

Let $\Phi_t(X) = 1 + [\neg\varphi]\cdot t + [\varphi]\cdot \mathsf{ert}\,[\![C]\!]\,(X)$.

Then $\mathsf{ert}\,[\![\texttt{while}\,(\varphi)\,\{C\}]\!]\,(t) = \mathsf{lfp}\ \Phi_t$.

Induction Rule for Upper Bounds

Choose $I \in T$. Then

$$\Phi_t(I) \preceq I \quad \text{implies} \quad \mathsf{ert}\,[\![\texttt{while}\,(\varphi)\,\{C\}]\!]\,(t) \preceq I$$

What is the expected runtime of $\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}$?

Let $\Phi_t(X) \;=\; 1 \;+\; [\neg\varphi]\cdot t \;+\; [\varphi]\cdot \mathsf{ert}\,[\![C]\!]\,(X)$.

Then $\quad \mathsf{ert}\,[\![\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}]\!]\,(t) \;=\; \mathsf{lfp}\;\; \Phi_t$.

Induction Rule for Upper Bounds

Choose $I \in T$. Then

$$\Phi_t(I) \;\preceq\; I \quad \text{implies} \quad \mathsf{ert}\,[\![\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}]\!]\,(t) \;\preceq\; I$$

Rule for Lower Bounds

What is the expected runtime of $\texttt{while}\,(\varphi)\,\{C\}$?

Let $\Phi_t(X) \;=\; 1 \;+\; [\neg\varphi]\cdot t \;+\; [\varphi]\cdot \mathsf{ert}\,[\![C]\!]\,(X)$.

Then $\quad \mathsf{ert}\,[\![\texttt{while}\,(\varphi)\,\{C\}]\!]\,(t) \;=\; \mathsf{lfp}\;\; \Phi_t$.

Induction Rule for Upper Bounds

Choose $I \in T$. Then

$$\Phi_t(I) \;\preceq\; I \quad \text{implies} \quad \mathsf{ert}\,[\![\texttt{while}\,(\varphi)\,\{C\}]\!]\,(t) \;\preceq\; I$$

Rule for Lower Bounds

Choose sequence $0 = I_0 \preceq I_1 \preceq I_2 \preceq \cdots \subset \mathbb{T}$.

What is the expected runtime of $\texttt{while}\,(\varphi)\,\{C\}$?

Let $\Phi_t(X) \;=\; 1 \;+\; [\neg\varphi]\cdot t \;+\; [\varphi]\cdot \mathsf{ert}\,[\![C]\!]\,(X)$.

Then $\quad \mathsf{ert}\,[\![\texttt{while}\,(\varphi)\,\{C\}]\!]\,(t) \;=\; \mathsf{lfp}\;\; \Phi_t$.

Induction Rule for Upper Bounds

Choose $I \in T$. Then

$$\Phi_t(I) \;\preceq\; I \quad \text{implies} \quad \mathsf{ert}\,[\![\texttt{while}\,(\varphi)\,\{C\}]\!]\,(t) \;\preceq\; I$$

Rule for Lower Bounds

Choose sequence $0 = I_0 \preceq I_1 \preceq I_2 \preceq \cdots \subset \mathbb{T}$. Then

$$I_{n+1} \;\preceq\; \Phi_t(I_n) \quad \text{implies} \quad \sup_n\; I_n \;\preceq\; \mathsf{ert}\,[\![\texttt{while}\,(\varphi)\,\{C\}]\!]\,(t)$$

What is the expected runtime of $\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}$?

Let $\Phi_t(X) \;=\; 1 \;+\; [\neg\varphi]\cdot t \;+\; [\varphi]\cdot \mathsf{ert}\,[\![C]\!]\,(X)$.

Then $\quad \mathsf{ert}\,[\![\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}]\!]\,(t) \;=\; \mathsf{lfp}\;\; \Phi_t$.

Induction Rule for Upper Bounds

Choose $I \in T$. Then

$$\Phi_t(I) \;\preceq\; I \quad \text{implies} \quad \mathsf{ert}\,[\![\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}]\!]\,(t) \;\preceq\; I$$

Rule for Lower Bounds

Choose sequence $0 = I_0 \preceq I_1 \preceq I_2 \preceq \cdots \subset \mathbb{T}$. Then

$$I_{n+1} \;\preceq\; \Phi_t(I_n) \quad \text{implies} \quad \sup_n\; I_n \;\preceq\; \mathsf{ert}\,[\![\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}]\!]\,(t)$$

What is the expected runtime of $\texttt{while}\,(\varphi)\,\{C\}$?

Let $\Phi_t(X) \;=\; 1 \;+\; [\neg\varphi]\cdot t \;+\; [\varphi]\cdot \mathsf{ert}\,[\![C]\!]\,(X)$.

Then $\quad \mathsf{ert}\,[\![\texttt{while}\,(\varphi)\,\{C\}]\!]\,(t) \;=\; \mathsf{lfp}\;\; \Phi_t$.

Induction Rule for Upper Bounds

Choose $I \in T$. Then

$$\Phi_t(I) \;\preceq\; I \quad \text{implies} \quad \mathsf{ert}\,[\![\texttt{while}\,(\varphi)\,\{C\}]\!]\,(t) \;\preceq\; I$$

Rule for Lower Bounds

Choose sequence $0 = I_0 \preceq I_1 \preceq I_2 \preceq \cdots \subset \mathbb{T}$. Then

$$I_{n+1} \;\preceq\; \Phi_t(I_n) \quad \text{implies} \quad \sup_n\; I_n \;\preceq\; \mathsf{ert}\,[\![\texttt{while}\,(\varphi)\,\{C\}]\!]\,(t)$$

Proving upper bounds is “easy”.

(induction)

Proving upper bounds is “easy”.

(induction)

Proving lower bounds is hard.

(the nasty sequence thing)

Proving upper bounds is “easy”.
(induction)

Proving lower bounds is hard.
(the nasty sequence thing)

Proving exact runtimes is also hard.

A new proof rule for exact expected runtimes

The Coupon Collector

$$
\begin{aligned}
&cp := [0, \ldots, 0] \,;\, i := 1 \,;\, x := N \\
&\texttt{while}\,(\, x > 0 \,)\,\{ \\
&\qquad \texttt{while}\,(\, cp[i] \neq 0 \,)\,\{ \\
&\qquad\qquad i :\approx \texttt{Unif}[1 \ldots N] \\
&\qquad \}\,; \\
&\qquad cp[i] := 1 \,;\, x := x - 1 \\
&\}
\end{aligned}
$$

The Coupon Collector

$$
\begin{aligned}
&cp := [0, \ldots, 0] \,;\, i := 1 \,;\, x := N \\
&\texttt{while}\,(\, x > 0 \,)\,\{ \\
&\qquad \texttt{while}\,(\, cp[i] \neq 0 \,)\,\{ \\
&\qquad\qquad i :\approx \texttt{Unif}[1 \ldots N] \\
&\qquad \}\,; \\
&\qquad cp[i] := 1 \,;\, x := x - 1 \\
&\}
\end{aligned}
$$

- Expected runtime analysis quite involved

The Coupon Collector

$$
\begin{aligned}
&cp := [0, \ldots, 0] \,;\, i := 1 \,;\, x := N \\
&\texttt{while}\,(\, x > 0 \,)\,\{ \\
&\qquad \texttt{while}\,(\, cp[i] \neq 0 \,)\,\{ \\
&\qquad\qquad i :\approx \texttt{Unif}[1 \ldots N] \\
&\qquad \}\,; \\
&\qquad cp[i] := 1 \,;\, x := x - 1 \\
&\}
\end{aligned}
$$

- Expected runtime analysis quite involved
- Part of the problem: Nested loops!

The Coupon Collector

$$cp := [0, \ldots, 0] \,;\, i := 1 \,;\, x := N$$

$$
\begin{aligned}
&F_f(J^f_n) \\
&= \mathbf{1} + [cp[i] = 0] \cdot f + [cp[i] \neq 0] \cdot \left(\mathbf{1} + \frac{1}{N} \cdot \sum_{k=1}^{N} J^f_n[i/k]\right) \\
&= \mathbf{1} + [cp[i] = 0] \cdot f + [cp[i] \neq 0] \cdot (\mathbf{1} + \frac{1}{N} \cdot \sum_{k=1}^{N} (\mathbf{1} + [cp[k] \\
&\quad + [cp[k] \neq 0] \cdot \sum_{\ell=0}^{n} \left(\frac{\#col}{N}\right)^{\ell} \cdot \left(2 + \frac{G(f)[i/k]}{N}\right))) \\
&= \mathbf{1} + [cp[i] = 0] \cdot f + [cp[i] \neq 0] \cdot (\mathbf{1} + \frac{1}{N} \cdot \sum_{k=1}^{N} (\mathbf{1} + [cp[k] \\
&\quad + [cp[k] \neq 0] \cdot \sum_{\ell=0}^{n} \left(\frac{\#col}{N}\right)^{\ell} \cdot \left(2 + \frac{G(f)}{N}\right)) \quad (k \text{ does n} \\
&= \mathbf{1} + [cp[i] = 0] \cdot f + \mathbf{2} \cdot [cp[i] \neq 0] + \frac{[cp[i] \neq 0]}{N} \cdot \sum_{k=1}^{N} ([cp[\\
&\quad + [cp[k] \neq 0] \cdot \sum_{\ell=0}^{n} \left(\frac{\#col}{N}\right)^{\ell} \cdot \left(2 + \frac{G(f)}{N}\right)) \\
&= \mathbf{1} + [cp[i] = 0] \cdot f + \mathbf{2} \cdot [cp[i] \neq 0] \\
&\quad + \frac{[cp[i] \neq 0]}{N} \cdot \sum_{k=1}^{N} ([cp[k] = 0] \cdot f[i/k]) \\
&\quad + \frac{[cp[i] \neq 0]}{N} \cdot \sum_{k=1}^{N} [cp[k] \neq 0] \cdot \sum_{\ell=0}^{n} \left(\frac{\#col}{N}\right)^{\ell} \cdot \left(2 + \frac{G(f}{N}\right. \\
&= \mathbf{1} + [cp[i] = 0] \cdot f + \mathbf{2} \cdot [cp[i] \neq 0] \\
&\quad + \frac{[cp[i] \neq 0]}{N} \cdot \sum_{k=1}^{N} ([cp[k] = 0] \cdot f[i/k]) \\
&\quad + \frac{[cp[i] \neq 0]\#col}{N} \cdot \sum_{\ell=0}^{n} \left(\frac{\#col}{N}\right)^{\ell} \cdot \left(2 + \frac{G(f)}{N}\right) \qquad (\text{Def. } \#col) \\
&= \mathbf{1} + [cp[i] = 0] \cdot f + [cp[i] \neq 0] \left(2 + \frac{G(f)}{N}\right) \\
&\quad + \frac{[cp[i] \neq 0]\#col}{N} \cdot \sum_{\ell=0}^{n} \left(\frac{\#col}{N}\right)^{\ell} \cdot \left(2 + \frac{G(f)}{N}\right) \qquad (\text{Def. } G) \\
&= \mathbf{1} + [cp[i] = 0] \cdot f + [cp[i] \neq 0] \left(2 + \frac{G(f)}{N}\right) \\
&\quad + [cp[i] \neq 0] \cdot \sum_{\ell=1}^{n+1} \left(\frac{\#col}{N}\right)^{\ell} \cdot \left(2 + \frac{G(f)}{N}\right) \\
&= \mathbf{1} + [cp[i] = 0] \cdot f + [cp[i] \neq 0] \cdot \sum_{\ell=0}^{n+1} \left(\frac{\#col}{N}\right)^{\ell} \cdot \left(2 + \frac{G(f)}{N}\right) \\
&= J^f_{n+1}
\end{aligned}
$$

Now, by Theorem 5, we obtain

$$J^g = \lim_{n\to\infty} J^g_n \preceq \mathsf{ert}\,[C_{\text{in}}]\,(g) \preceq \lim_{n\to\infty} J^g_n = J^g.$$

The Coupon Collector

$$
\begin{aligned}
&cp := [0, \ldots, 0] \,;\, i := 1 \,;\, x := N \\
&\texttt{while}\,(\, x > 0 \,)\,\{ \\
&\qquad \texttt{while}\,(\, cp[i] \neq 0 \,)\,\{ \\
&\qquad\qquad i :\approx \texttt{Unif}[1 \ldots N] \\
&\qquad \}\,; \\
&\qquad cp[i] := 1 \,;\, x := x - 1 \\
&\}
\end{aligned}
$$

- Expected runtime analysis quite involved
- Part of the problem: Nested loops!

The Coupon Collector

$$
\begin{aligned}
&cp := [0, \ldots, 0] \fatsemi i := 1 \fatsemi x := N \\
&\texttt{while}\,(\,x > 0\,)\,\{ \\
&\qquad \texttt{while}\,(\,cp[i] \neq 0\,)\,\{ \\
&\qquad\qquad i :\approx \texttt{Unif}[1 \ldots N] \\
&\qquad \}\fatsemi \\
&\qquad cp[i] := 1 \fatsemi x := x - 1 \\
&\}
\end{aligned}
$$

- Expected runtime analysis quite involved
- Part of the problem: Nested loops!
- But the inner loop is simple!

The Coupon Collector's Inner Loop

$$\begin{array}{l}\texttt{while}\,(\,cp[i] \neq 0\,)\,\{\\ \qquad i \,:\!\approx\, \texttt{Unif}[1 \ldots N]\\ \}\end{array}$$

The Coupon Collector's Inner Loop

$$\begin{array}{l}\texttt{while}\,(\,cp[i] \neq 0\,)\,\{\\ \qquad i :\approx \texttt{Unif}[1 \ldots N]\\ \}\end{array}$$

- **Desideratum:** Closed form for ert of inner loop

The Coupon Collector's Inner Loop

$$\begin{array}{l}\texttt{while}\,(\,cp[i] \neq 0\,)\,\{\\ \qquad i :\approx \texttt{Unif}[1 \ldots N]\\ \}\end{array}$$

- **Desideratum:** Closed form for ert of inner loop
- **Observations:**

The Coupon Collector's Inner Loop

$$\begin{array}{l}\texttt{while}\,(\,cp[i] \neq 0\,)\,\{\\ \qquad i :\approx \texttt{Unif}[1 \ldots N]\\ \}\end{array}$$

- **Desideratum:** Closed form for ert of inner loop
- **Observations:**
  1. No information flow across loop iterations!

The Coupon Collector's Inner Loop

$$\begin{array}{l}\texttt{while}\,(\,cp[i] \neq 0\,)\,\{\\ \qquad i :\approx \texttt{Unif}[1 \ldots N]\\ \}\end{array}$$

- **Desideratum:** Closed form for ert of inner loop
- **Observations:**
  1. No information flow across loop iterations!
  2. Loop terminates with "constant" probability after each iteration.

The Coupon Collector's Inner Loop

$$\texttt{while}\,(\,cp[i] \neq 0\,)\,\{$$
$$\qquad i :\approx \texttt{Unif}[1 \ldots N]$$
$$\}$$

- **Desideratum:** Closed form for ert of inner loop
- **Observations:**
  1. No information flow across loop iterations!
  2. Loop terminates with "constant" probability after each iteration.
- **Simple loops should have simple runtime proofs!**

Weakest Preexpectations

Let $f \in \mathbb{T}$ (think: f is a random variable).

Weakest Preexpectations

Let $f \in \mathbb{T}$ (think: f is a random variable).

Weakest preexpectation of C with respect to f:

Weakest Preexpectations

Let $f \in \mathbb{T}$ (think: f is a random variable).

Weakest preexpectation of C with respect to f:

$$\text{wp}\,[\![C]\!]\,(f): \quad \text{States} \to \mathbb{R}^{\infty}_{\geq 0},$$

Weakest Preexpectations

Let $f \in \mathbb{T}$ (think: f is a random variable).

Weakest preexpectation of C with respect to f:

$$\text{wp}\,[\![C]\!]\,(f): \quad \text{States} \to \mathbb{R}^{\infty}_{\geq 0},$$

$$\text{wp}\,[\![C]\!]\,(f)\,(\sigma) \;=\; \text{EV of } f \text{ after executing } C \text{ on } \sigma$$

Weakest Preexpectations

Let $f \in \mathbb{T}$ (think: f is a random variable).

Weakest preexpectation of C with respect to f:

$$\text{wp}\,[\![C]\!]\,(f): \quad \text{States} \to \mathbb{R}^{\infty}_{\geq 0},$$

$$\text{wp}\,[\![C]\!]\,(f)\,(\sigma) \;=\; \text{EV of } f \text{ after executing } C \text{ on } \sigma$$

Example:

$$\text{wp}\,[\![\{\, x := 5 \,\}\,[1/2]\,\{\, x := 3 \,\}]\!]\,(x)$$

Weakest Preexpectations

Let $f \in \mathbb{T}$ (think: f is a random variable).

Weakest preexpectation of C with respect to f:

$$\text{wp}\,[\![C]\!]\,(f): \quad \text{States} \to \mathbb{R}^{\infty}_{\geq 0},$$

$$\text{wp}\,[\![C]\!]\,(f)\,(\sigma) \;=\; \text{EV of } f \text{ after executing } C \text{ on } \sigma$$

Example:

$$\text{wp}\,[\![\{\, x := 5 \,\}\,[1/2]\,\{\, x := 3 \,\}]\!]\,(x) \;=\; \tfrac{1}{2}\cdot 5 + \tfrac{1}{2}\cdot 3$$

Weakest Preexpectations

Let $f \in \mathbb{T}$ (think: f is a random variable).

Weakest preexpectation of C with respect to f:

$$\text{wp}\,[\![C]\!]\,(f): \quad \text{States} \to \mathbb{R}^{\infty}_{\geq 0},$$

$$\text{wp}\,[\![C]\!]\,(f)\,(\sigma) \;=\; \text{EV of } f \text{ after executing } C \text{ on } \sigma$$

Example:

$$\text{wp}\,[\![\{\, x := 5 \,\}\,[1/2]\,\{\, x := 3 \,\}]\!]\,(x) \;=\; 4$$

Weakest Preexpectations

Let $f \in \mathbb{T}$ (think: f is a random variable).

Weakest preexpectation of C with respect to f:

$$\text{wp}\,[\![C]\!]\,(f): \quad \text{States} \to \mathbb{R}^{\infty}_{\geq 0},$$

$$\text{wp}\,[\![C]\!]\,(f)\,(\sigma) \;=\; \text{EV of } f \text{ after executing } C \text{ on } \sigma$$

Example:

$$\text{wp}\,[\![\{\, x := 5 \,\}\,[1/2]\,\{\, x := 3 \,\}]\!]\,(x) \;=\; 4$$

$$\text{wp}\,[\![\{\, x := 5 \,\}\,[1/2]\,\{\, x := x + 5 \,\}]\!]\,(x)$$

Weakest Preexpectations

Let $f \in \mathbb{T}$ (think: f is a random variable).

Weakest preexpectation of C with respect to f:

$$\text{wp}\,[\![C]\!]\,(f): \quad \text{States} \to \mathbb{R}^{\infty}_{\geq 0},$$

$$\text{wp}\,[\![C]\!]\,(f)\,(\sigma) \;=\; \text{EV of } f \text{ after executing } C \text{ on } \sigma$$

Example:

$$\text{wp}\,[\![\{\, x := 5 \,\}\,[1/2]\,\{\, x := 3 \,\}]\!]\,(x) \;=\; 4$$

$$\text{wp}\,[\![\{\, x := 5 \,\}\,[1/2]\,\{\, x := x + 5 \,\}]\!]\,(x) \;=\; \tfrac{1}{2} \cdot 5 + \tfrac{1}{2} \cdot (x + 5)$$

Weakest Preexpectations

Let $f \in \mathbb{T}$ (think: f is a random variable).

Weakest preexpectation of C with respect to f:

$$\text{wp}\,[\![C]\!]\,(f): \quad \text{States} \to \mathbb{R}^{\infty}_{\geq 0},$$

$$\text{wp}\,[\![C]\!]\,(f)\,(\sigma) \;=\; \text{EV of } f \text{ after executing } C \text{ on } \sigma$$

Example:

$$\text{wp}\,[\![\{\, x := 5 \,\}\,[1/2]\,\{\, x := 3 \,\}]\!]\,(x) \;=\; 4$$

$$\text{wp}\,[\![\{\, x := 5 \,\}\,[1/2]\,\{\, x := x + 5 \,\}]\!]\,(x) \;=\; \tfrac{x}{2} + 5$$

Theorem: A New Proof Rule for Expected Runtimes

Theorem: A New Proof Rule for Expected Runtimes

- Let $\texttt{while}\,(\varphi)\,\{C\}$ be t–i.i.d.

Theorem: A New Proof Rule for Expected Runtimes

- Let $\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}$ be t–i.i.d.
  - Every loop iteration has equal termination probability

Theorem: A New Proof Rule for Expected Runtimes

- Let $\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}$ be t–i.i.d.
  - Every loop iteration has equal termination probability
  - EV of t determined by a single loop iteration

Theorem: A New Proof Rule for Expected Runtimes

- Let $\texttt{while}\,(\varphi)\,\{C\}$ be t–i.i.d.
  - Every loop iteration has equal termination probability
  - EV of t determined by a single loop iteration
- C terminates almost–surely (with probability 1)

Theorem: A New Proof Rule for Expected Runtimes

- Let $\texttt{while}\,(\varphi)\,\{C\}$ be t–i.i.d.
  - Every loop iteration has equal termination probability
  - EV of t determined by a single loop iteration
- C terminates almost–surely (with probability 1)
- Every loop iteration takes equal expected time

Theorem: A New Proof Rule for Expected Runtimes

- Let $\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}$ be t–i.i.d.
  - Every loop iteration has equal termination probability
  - EV of t determined by a single loop iteration
- C terminates almost–surely (with probability 1)
- Every loop iteration takes equal expected time

Then $\mathsf{ert}\,[\![\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}]\!]\,(t)$ is given by

$$1 \;+\; [\neg\varphi]\cdot t \;+\; [\varphi]\cdot\frac{1+\mathbf{ert}\,[\![C]\!]\,([\neg\varphi]\cdot t)}{1-\mathbf{wp}\,[\![C]\!]\,([\varphi])}$$

Theorem: A New Proof Rule for Expected Runtimes

- Let $\texttt{while}\,(\varphi)\,\{C\}$ be t–i.i.d.
  - Every loop iteration has equal termination probability
  - EV of t determined by a single loop iteration
- C terminates almost–surely (with probability 1)
- Every loop iteration takes equal expected time

Then $\text{ert}\,[\![\texttt{while}\,(\varphi)\,\{C\}]\!]\,(t)$ is given by

$$1 \;+\; [\neg\varphi]\cdot t \;+\; [\varphi]\cdot\frac{1+\textbf{ert}\,[\![C]\!]\,([\neg\varphi]\cdot t)}{1-\textbf{wp}\,[\![C]\!]\,([\varphi])}$$

Closed form for exact expected runtime!

Theorem: A New Proof Rule for Expected Runtimes

- Let $\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}$ be t–i.i.d.
  - Every loop iteration has equal termination probability
  - EV of t determined by a single loop iteration
- C terminates almost–surely (with probability 1)
- Every loop iteration takes equal expected time

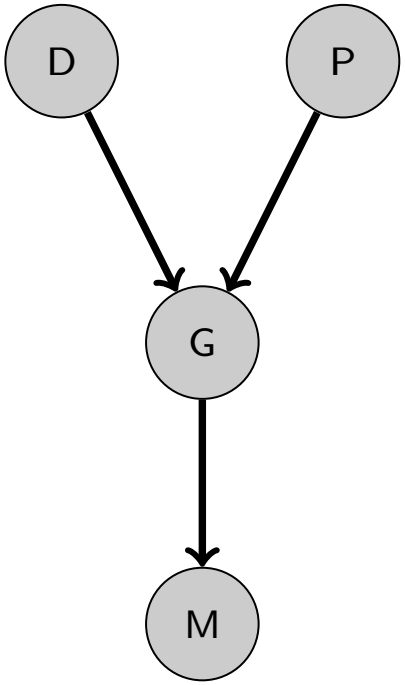
Then $\text{ert}\,[\![\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}]\!]\,(t)$ is given by

$$1 \;+\; [\neg\varphi]\cdot t \;+\; [\varphi]\cdot\frac{1+\textbf{ert}\,[\![C]\!]\,([\neg\varphi]\cdot t)}{1-\textbf{wp}\,[\![C]\!]\,([\varphi])}$$

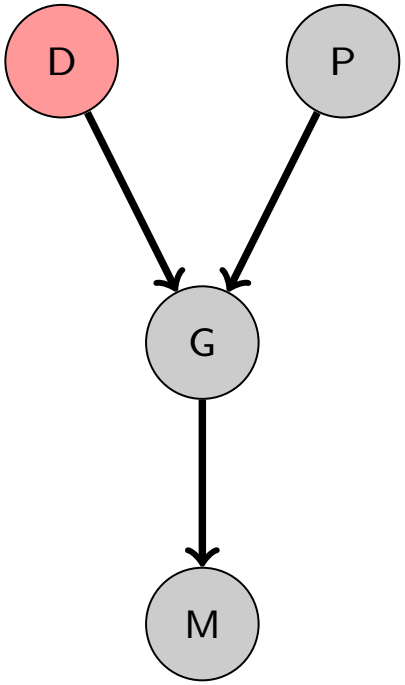
Closed form for exact expected runtime!
No (sequence) guessing needed!

How long, O Bayesian network, will I sample thee?

Bayesian Network → Probabilistic Program



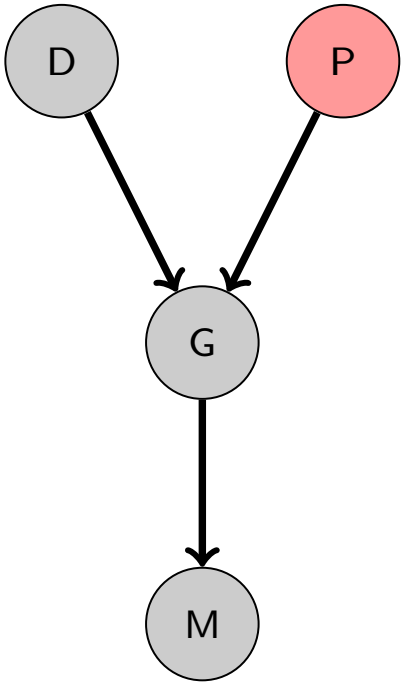
Bayesian Network → Probabilistic Program



$D = 0$	$D = 1$
0.95	0.05

$x_D :\approx 0.95 \cdot \langle 0 \rangle + 0.05 \cdot \langle 1 \rangle;$

Bayesian Network → Probabilistic Program

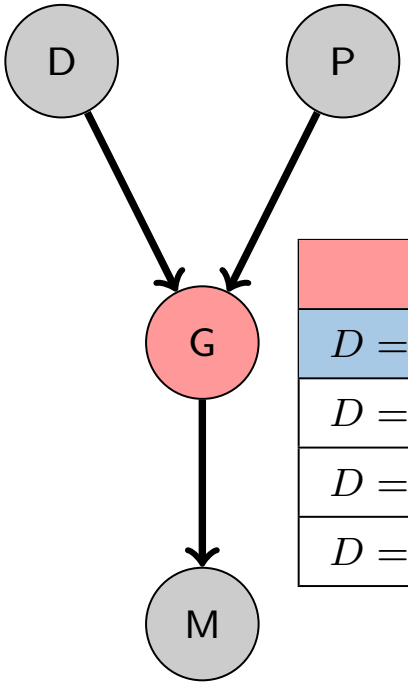


$P = 0$	$P = 1$
0.98	0.02

$x_D :\approx 0.95 \cdot \langle 0 \rangle + 0.05 \cdot \langle 1 \rangle;$

$x_P :\approx 0.98 \cdot \langle 0 \rangle + 0.02 \cdot \langle 1 \rangle;$

Bayesian Network → Probabilistic Program



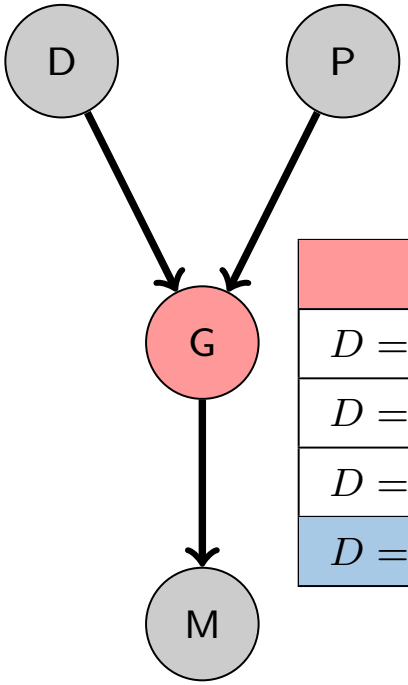
	$G = 0$	$G = 1$
$D = 0,\ P = 0$	0.99	0.01
$D = 0,\ P = 1$	0.29	0.71
$D = 1,\ P = 0$	0.06	0.94
$D = 1,\ P = 1$	0.01	0.99

$$x_D :\approx 0.95 \cdot \langle 0 \rangle + 0.05 \cdot \langle 1 \rangle;$$

$$x_P :\approx 0.98 \cdot \langle 0 \rangle + 0.02 \cdot \langle 1 \rangle;$$

$$\text{if}(x_D = 0 \wedge x_P = 0)\{$$
$$\quad x_G :\approx 0.99 \cdot \langle 0 \rangle + 0.01 \cdot \langle 1 \rangle;$$
$$\vdots$$

Bayesian Network → Probabilistic Program



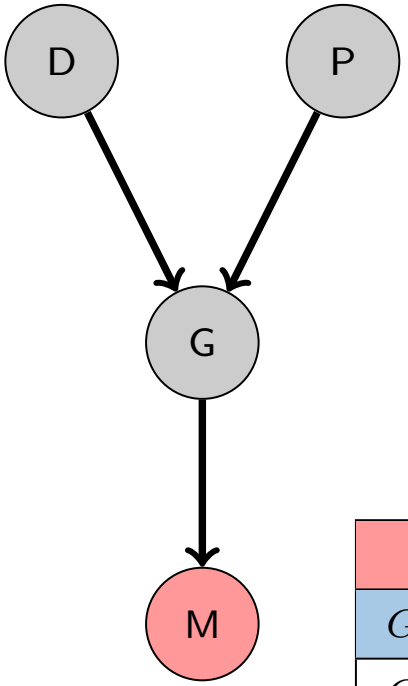
	$G=0$	$G=1$
$D=0,\ P=0$	0.99	0.01
$D=0,\ P=1$	0.29	0.71
$D=1,\ P=0$	0.06	0.94
$D=1,\ P=1$	0.01	0.99

$$x_D :\approx 0.95 \cdot \langle 0 \rangle + 0.05 \cdot \langle 1 \rangle;$$

$$x_P :\approx 0.98 \cdot \langle 0 \rangle + 0.02 \cdot \langle 1 \rangle;$$

$$\text{if}(x_D = 0 \wedge x_P = 0)\{$$
$$\quad x_G :\approx 0.99 \cdot \langle 0 \rangle + 0.01 \cdot \langle 1 \rangle;$$
$$\vdots$$
$$\} \text{ else } \{$$
$$\quad x_G :\approx 0.01 \cdot \langle 0 \rangle + 0.99 \cdot \langle 1 \rangle\};$$

Bayesian Network → Probabilistic Program



	$M=0$	$M=1$
$G=0$	0.9	0.1
$G=1$	0.3	0.7

$x_D :\approx 0.95 \cdot \langle 0 \rangle + 0.05 \cdot \langle 1 \rangle;$

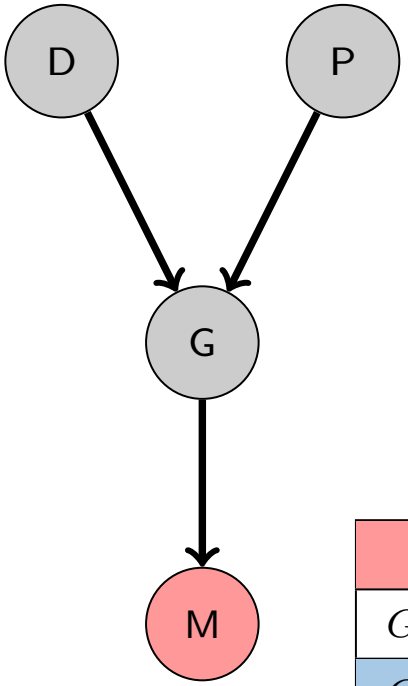
$x_P :\approx 0.98 \cdot \langle 0 \rangle + 0.02 \cdot \langle 1 \rangle;$

if$(x_D = 0 \wedge x_P = 0)\{$
$\quad x_G :\approx 0.99 \cdot \langle 0 \rangle + 0.01 \cdot \langle 1 \rangle;$
$\vdots$

$\}$ else $\{$
$\quad x_G :\approx 0.01 \cdot \langle 0 \rangle + 0.99 \cdot \langle 1 \rangle\};$

if$(x_G = 0)\{$
$\quad x_M :\approx 0.9 \cdot \langle 0 \rangle + 0.1 \cdot \langle 1 \rangle;$

Bayesian Network → Probabilistic Program



	$M=0$	$M=1$
$G=0$	0.9	0.1
$G=1$	0.3	0.7

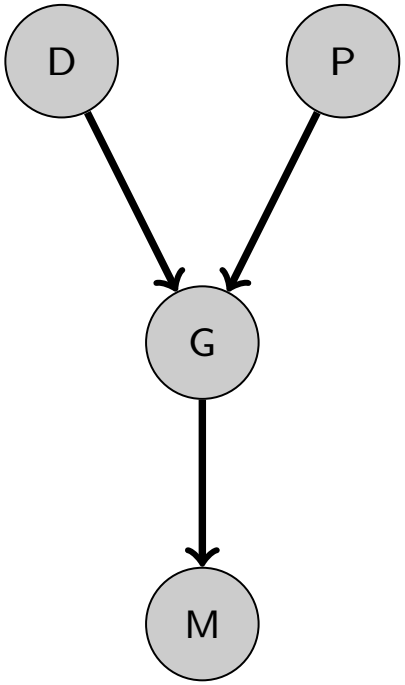
$$x_D :\approx 0.95 \cdot \langle 0 \rangle + 0.05 \cdot \langle 1 \rangle;$$

$$x_P :\approx 0.98 \cdot \langle 0 \rangle + 0.02 \cdot \langle 1 \rangle;$$

$$\begin{aligned}
&\mathsf{if}(x_D = 0 \wedge x_P = 0)\{ \\
&\quad x_G :\approx 0.99 \cdot \langle 0 \rangle + 0.01 \cdot \langle 1 \rangle; \\
&\vdots \\
&\} \ \mathsf{else}\ \{ \\
&\quad x_G :\approx 0.01 \cdot \langle 0 \rangle + 0.99 \cdot \langle 1 \rangle\};
\end{aligned}$$

$$\begin{aligned}
&\mathsf{if}(x_G = 0)\{ \\
&\quad x_M :\approx 0.9 \cdot \langle 0 \rangle + 0.1 \cdot \langle 1 \rangle; \\
&\} \ \mathsf{else}\ \{ \\
&\quad x_M :\approx 0.3 \cdot \langle 0 \rangle + 0.7 \cdot \langle 1 \rangle\};
\end{aligned}$$

Bayesian Network $\rightarrow$ Probabilistic Program



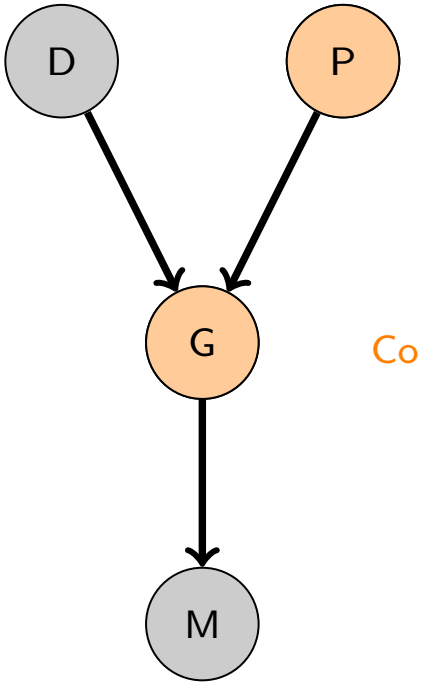
$$x_D :\approx 0.95 \cdot \langle 0 \rangle + 0.05 \cdot \langle 1 \rangle;$$

$$x_P :\approx 0.98 \cdot \langle 0 \rangle + 0.02 \cdot \langle 1 \rangle;$$

$$\begin{aligned}
&\mathsf{if}(x_D = 0 \wedge x_P = 0)\{ \\
&\quad x_G :\approx 0.99 \cdot \langle 0 \rangle + 0.01 \cdot \langle 1 \rangle; \\
&\vdots \\
&\} \ \mathsf{else} \ \{ \\
&\quad x_G :\approx 0.01 \cdot \langle 0 \rangle + 0.99 \cdot \langle 1 \rangle\};
\end{aligned}$$

$$\begin{aligned}
&\mathsf{if}(x_G = 0)\{ \\
&\quad x_M :\approx 0.9 \cdot \langle 0 \rangle + 0.1 \cdot \langle 1 \rangle; \\
&\} \ \mathsf{else} \ \{ \\
&\quad x_M :\approx 0.3 \cdot \langle 0 \rangle + 0.7 \cdot \langle 1 \rangle\};
\end{aligned}$$

Bayesian Network $\rightarrow$ Probabilistic Program



Conditioning: $G \stackrel{!}{=} 0$ and $P \stackrel{!}{=} 0$

$x_D :\approx 0.95 \cdot \langle 0 \rangle + 0.05 \cdot \langle 1 \rangle;$

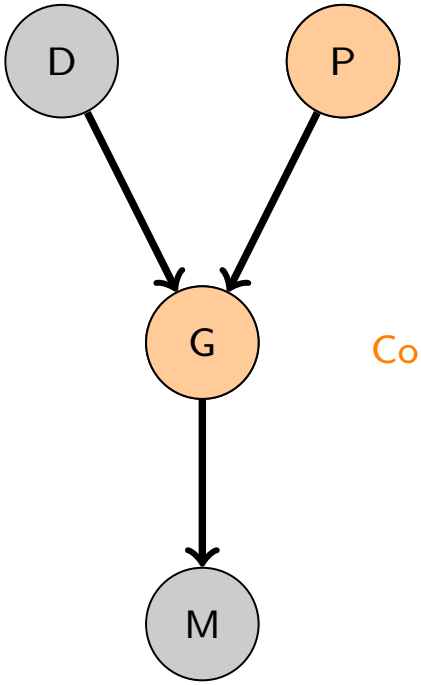
$x_P :\approx 0.98 \cdot \langle 0 \rangle + 0.02 \cdot \langle 1 \rangle;$

if$(x_D = 0 \wedge x_P = 0)\{$
    $x_G :\approx 0.99 \cdot \langle 0 \rangle + 0.01 \cdot \langle 1 \rangle;$
$\vdots$

$\}$ else $\{$
    $x_G :\approx 0.01 \cdot \langle 0 \rangle + 0.99 \cdot \langle 1 \rangle\};$

if$(x_G = 0)\{$
    $x_M :\approx 0.9 \cdot \langle 0 \rangle + 0.1 \cdot \langle 1 \rangle;$
$\}$ else $\{$
    $x_M :\approx 0.3 \cdot \langle 0 \rangle + 0.7 \cdot \langle 1 \rangle\};$

Bayesian Network $\rightarrow$ Probabilistic Program



Conditioning: $G \stackrel{!}{=} 0$ and $P \stackrel{!}{=} 0$

repeat {

$x_D :\approx 0.95 \cdot \langle 0 \rangle + 0.05 \cdot \langle 1 \rangle;$

$x_P :\approx 0.98 \cdot \langle 0 \rangle + 0.02 \cdot \langle 1 \rangle;$

if$(x_D = 0 \wedge x_P = 0)\{$
$\quad x_G :\approx 0.99 \cdot \langle 0 \rangle + 0.01 \cdot \langle 1 \rangle;$
$\vdots$

} else {
$\quad x_G :\approx 0.01 \cdot \langle 0 \rangle + 0.99 \cdot \langle 1 \rangle\};$

if$(x_G = 0)\{$
$\quad x_M :\approx 0.9 \cdot \langle 0 \rangle + 0.1 \cdot \langle 1 \rangle;$
} else {
$\quad x_M :\approx 0.3 \cdot \langle 0 \rangle + 0.7 \cdot \langle 1 \rangle\};$

} until$(x_G = 0 \wedge x_P = 0)$

Translation: Bayesian Network to Probabilistic Program

Translation: Bayesian Network to Probabilistic Program

- Our translation $BN \to C$ is *correct*:

Translation: Bayesian Network to Probabilistic Program

- Our translation $BN \rightarrow C$ is *correct*:
  - Distribution established by C corresponds to the *conditional* distribution of BN *given observed evidence*

Translation: Bayesian Network to Probabilistic Program

- Our translation $BN \rightarrow C$ is *correct*:
  - Distribution established by C corresponds to the *conditional* distribution of BN *given observed evidence*
- Executing C corresponds to sampling BN:

Translation: Bayesian Network to Probabilistic Program

- Our translation $BN \rightarrow C$ is *correct*:
  - Distribution established by C corresponds to the *conditional* distribution of BN *given observed evidence*
- Executing C corresponds to sampling BN:
  - Running C once corresponds to obtaining a *single sample* from BN that *satisfies the observed evidence*

Translation: Bayesian Network to Probabilistic Program

- Our translation $BN \to C$ is *correct*:
  - Distribution established by C corresponds to the *conditional* distribution of BN *given observed evidence*
- Executing C corresponds to sampling BN:
  - Running C once corresponds to obtaining a *single sample* from BN that *satisfies the observed evidence*
  - **Expected runtime of C** corresponds to **expected time to obtain a single sample from BN** using rejection sampling

Translation: Bayesian Network to Probabilistic Program

- Our translation $BN \to C$ is *correct*:
  - Distribution established by C corresponds to the *conditional* distribution of BN *given observed evidence*
- Executing C corresponds to sampling BN:
  - Running C once corresponds to obtaining a *single sample* from BN that *satisfies the observed evidence*
  - **Expected runtime of C** corresponds to **expected time to obtain a single sample from BN** using rejection sampling
- The `repeat...until`–loop in C is always t–i.i.d. for any t

Translation: Bayesian Network to Probabilistic Program

- Our translation $BN \to C$ is *correct*:
  - Distribution established by C corresponds to the *conditional* distribution of BN *given observed evidence*
- Executing C corresponds to sampling BN:
  - Running C once corresponds to obtaining a *single sample* from BN that *satisfies the observed evidence*
  - **Expected runtime of C** corresponds to **expected time to obtain a single sample from BN** using rejection sampling
- The `repeat...until`–loop in C is always t–i.i.d. for any t
- **Our closed form proof rule applies to C!**

Translation: Bayesian Network to Probabilistic Program

- Our translation $BN \to C$ is *correct*:
  - Distribution established by C corresponds to the *conditional* distribution of BN *given observed evidence*
- Executing C corresponds to sampling BN:
  - Running C once corresponds to obtaining a *single sample* from BN that *satisfies the observed evidence*
  - **Expected runtime of C** corresponds to **expected time to obtain a single sample from BN** using rejection sampling
- The `repeat...until`–loop in C is always t–i.i.d. for any t
- **Our closed form proof rule applies to C!**
- **Formal analysis of a probabilistic program yields expected sampling time of a Bayesian network.**

BN	#obs	Time	EST	#obs	Time	EST
`hailfinder`	*#nodes: 56, #edges: 66, avg. Markov Blanket: 3.54*					
	5	0.63	$5.016 \cdot 10^5$	9	0.46	$9.048 \cdot 10^6$
`hepar2`	*#nodes: 70, #edges: 123, avg. Markov Blanket: 4.51*					
	1	1.84	$1.579 \cdot 10^2$	2	MO	—
`win95pts`	*#nodes: 76, #edges: 112, avg. Markov Blanket: 5.92*					
	1	0.36	$2.284 \cdot 10^3$	3	0.36	$4.296 \cdot 10^5$
	12	0.42	$3.973 \cdot 10^7$	17	61.73	$1.110 \cdot 10^{15}$
`pathfinder`	*#nodes: 135, #edges: 200, avg. Markov Blanket: 3.04*					
	1	0.53	$1.050 \cdot 10^4$	3	31.31	$2.872 \cdot 10^4$
	7	5.44	∞	7	480.83	∞
`andes`	*#nodes: 223, #edges: 338, avg. Markov Blanket: 5.61*					
	1	MO	—	3	1.66	$5.251 \cdot 10^3$
	7	0.99	$8.904 \cdot 10^4$	9	0.90	$6.637 \cdot 10^5$
`pigs`	*#nodes: 441, #edges: 592, avg. Markov Blanket: 3.66*					
	1	0.74	$2.952 \cdot 10^3$	3	0.88	$2.362 \cdot 10^3$
	7	1.02	$1.511 \cdot 10^6$	8	MO	—
`munin`	*#nodes: 1041, #edges: 1397, avg. Markov Blanket: 3.54*					
	1	1.47	$3.648 \cdot 10^4$	3	1.37	$1.824 \cdot 10^7$
	9	1.79	$1.824 \cdot 10^{16}$	10	65.64	$1.153 \cdot 10^{18}$

`bnlearn.com/bnrepository`

BN	#obs	Time	EST	#obs	Time	EST
`hailfinder`	*#nodes: 56, #edges: 66, avg. Markov Blanket: 3.54*					
	5	0.63	$5.016 \cdot 10^5$	9	0.46	$9.048 \cdot 10^6$
`hepar2`	*#nodes: 70, #edges: 123, avg. Markov Blanket: 4.51*					
	1	1.84	$1.579 \cdot 10^2$	2	MO	—
`win95pts`	*#nodes: 76, #edges: 112, avg. Markov Blanket: 5.92*					
	1	0.36	$2.284 \cdot 10^3$	3	0.36	$4.296 \cdot 10^5$
	12	0.42	$3.973 \cdot 10^7$	17	61.73	$1.110 \cdot 10^{15}$
`pathfinder`	*#nodes: 135, #edges: 200, avg. Markov Blanket: 3.04*					
	1	0.53	$1.050 \cdot 10^4$	3	31.31	$2.872 \cdot 10^4$
	7	5.44	∞	7	480.83	∞
`andes`	*#nodes: 223, #edges: 338, avg. Markov Blanket: 5.61*					
	1	MO	—	3	1.66	$5.251 \cdot 10^3$
	7	0.99	$8.904 \cdot 10^4$	9	0.90	$6.637 \cdot 10^5$
`pigs`	*#nodes: 441, #edges: 592, avg. Markov Blanket: 3.66*					
	1	0.74	$2.952 \cdot 10^3$	3	0.88	$2.362 \cdot 10^3$
	7	1.02	$1.511 \cdot 10^6$	8	MO	—
`munin`	*#nodes: 1041, #edges: 1397, avg. Markov Blanket: 3.54*					
	1	1.47	$3.648 \cdot 10^4$	3	1.37	$1.824 \cdot 10^7$
	9	1.79	$1.824 \cdot 10^{16}$	10	65.64	$1.153 \cdot 10^{18}$

`bnlearn.com/bnrepository`

BN	#obs	Time	EST	#obs	Time	EST
hailfinder	*#nodes: 56, #edges: 66, avg. Markov Blanket: 3.54*					
	5	0.63	$5.016 \cdot 10^5$	9	0.46	$9.048 \cdot 10^6$
hepar2	*#nodes: 70, #edges: 123, avg. Markov Blanket: 4.51*					
	1	1.84	$1.579 \cdot 10^2$	2	MO	—
win95pts	*#nodes: 76, #edges: 112, avg. Markov Blanket: 5.92*					
	1	0.36	$2.284 \cdot 10^3$	3	0.36	$4.296 \cdot 10^5$
	12	0.42	$3.973 \cdot 10^7$	17	61.73	$1.110 \cdot 10^{15}$
pathfinder	*#nodes: 135, #edges: 200, avg. Markov Blanket: 3.04*					
	1	0.53	$4.050 \cdot 10^4$	3	31.31	$2.872 \cdot 10^4$
	7	3.64	∞	7	480.83	∞
andes	*#nodes: 223, #edges: 338, avg. Markov Blanket: 5.61*					
	1	0.9	—	3	1.66	$5.251 \cdot 10^3$
	7	0.99	$1.904 \cdot 10^4$	9	0.90	$6.637 \cdot 10^5$
pigs	*#nodes: 441, #edges: 592, avg. Markov Blanket: 3.66*					
	1	0.74	$2.952 \cdot 10^3$	3	0.88	$2.362 \cdot 10^3$
	7	1.12	$1.511 \cdot 10^6$	8	MO	—
munin	*#nodes: 1041, #edges: 1397, avg. Markov Blanket: 3.54*					
	1	1.47	$3.648 \cdot 10^4$	3	1.37	$1.824 \cdot 10^7$
	9	1.79	$1.824 \cdot 10^{16}$	10	65.64	$1.153 \cdot 10^{18}$

bnlearn.com/bnrepository

BN	#obs	Time	EST	#obs	Time	EST
`hailfinder`	*#nodes: 56, #edges: 66, avg. Markov Blanket: 3.54*					
	5	0.63	$5.016 \cdot 10^5$	9	0.46	$9.048 \cdot 10^6$
`hepar2`	*#nodes: 70, #edges: 123, avg. Markov Blanket: 4.51*					
	1	1.84	$1.579 \cdot 10^2$	2	MO	—
`win95pts`	*#nodes: 76, #edges: 112, avg. Markov Blanket: 5.92*					
	1	0.36	$2.284 \cdot 10^3$	3	0.36	$4.296 \cdot 10^5$
	12	0.42	$3.973 \cdot 10^7$	17	61.73	$1.110 \cdot 10^{15}$
`pathfinder`	*#nodes: 135, #edges: 200, avg. Markov Blanket: 3.04*					
	1	0.53	$1.050 \cdot 10^4$	3	31.31	$2.872 \cdot 10^4$
	7	5.44	∞	7	480.83	∞
`andes`	*#nodes: 223, #edges: 338, avg. Markov Blanket: 5.61*					
	1	MO	—	3	1.66	$5.251 \cdot 10^3$
	7	0.99	$8.904 \cdot 10^4$	9	0.90	$6.637 \cdot 10^5$
`pigs`	*#nodes: 441, #edges: 592, avg. Markov Blanket: 3.66*					
	1	0.74	$2.952 \cdot 10^3$	3	0.88	$2.362 \cdot 10^3$
	7	1.02	$1.511 \cdot 10^6$	8	MO	—
`munin`	*#nodes: 1041, #edges: 1397, avg. Markov Blanket: 3.54*					
	1	1.47	$3.648 \cdot 10^4$	3	1.37	$1.824 \cdot 10^7$
	9	1.79	$1.824 \cdot 10^{16}$	10	65.64	$1.153 \cdot 10^{18}$

`bnlearn.com/bnrepository`

BN	#obs	Time	EST	#obs	Time	EST
`hailfinder`	*#nodes: 56, #edges: 66, avg. Markov Blanket: 3.54*					
	5	0.63	$5.016 \cdot 10^5$	9	0.46	$9.048 \cdot 10^6$
`hepar2`	*#nodes: 70, #edges: 123, avg. Markov Blanket: 4.51*					
	1	1.84	$1.579 \cdot 10^2$	2	MO	—
`win95pts`	*#nodes: 76, #edges: 112, avg. Markov Blanket: 5.92*					
	1	0.36	$2.284 \cdot 10^3$	3	0.36	$4.296 \cdot 10^5$
	12	0.42	$3.973 \cdot 10^7$	17	61.73	$1.110 \cdot 10^{15}$
`pathfinder`	*#nodes: 135, #edges: 200, avg. Markov Blanket: 3.04*					
	1	0.53	$1.050 \cdot 10^4$	3	31.31	$2.872 \cdot 10^4$
	7	5.44	∞	7	480.83	∞
`andes`	*#nodes: 223, #edges: 338, avg. Markov Blanket: 5.61*					
	1	MO	—	3	1.66	$5.251 \cdot 10^3$
	7	0.99	$8.904 \cdot 10^4$	9	0.90	$6.637 \cdot 10^5$
`pigs`	*#nodes: 441, #edges: 592, avg. Markov Blanket: 3.66*					
	1	0.74	$2.952 \cdot 10^3$	3	0.88	$2.362 \cdot 10^3$
	7	1.02	$1.511 \cdot 10^6$	8	MO	—
`munin`	*#nodes: 1041, #edges: 1397, avg. Markov Blanket: 3.54*					
	1	1.47	$3.648 \cdot 10^4$	3	1.37	$1.824 \cdot 10^7$
	9	1.79	$1.824 \cdot 10^{16}$	10	65.64	$1.153 \cdot 10^{18}$

`bnlearn.com/bnrepository`

BN	#obs	Time	EST	#obs	Time	EST
hailfinder	*#nodes: 56, #edges: 66, avg. Markov Blanket: 3.54*					
	5	0.63	$5.016 \cdot 10^5$	9	0.46	$9.048 \cdot 10^6$
hepar2	*#nodes: 70, #edges: 123, avg. Markov Blanket: 4.51*					
	1	1.84	$1.579 \cdot 10^2$	2	MO	—
win95pts	*#nodes: 76, #edges: 112, avg. Markov Blanket: 5.92*					
	1	0.36	$2.284 \cdot 10^3$	3	0.36	$4.296 \cdot 10^5$
	12	0.42	$3.973 \cdot 10^7$	17	61.73	$1.110 \cdot 10^{15}$
pathfinder	*#nodes: 135, #edges: 200, avg. Markov Blanket: 3.04*					
	1	0.53	$1.050 \cdot 10^4$	3	31.31	$2.872 \cdot 10^4$
	7	5.44	∞	7	480.83	∞
andes	*#nodes: 223, #edges: 338, avg. Markov Blanket: 5.61*					
	1	MO	—	3	1.66	$5.251 \cdot 10^3$
	7	0.99	$8.904 \cdot 10^4$	9	0.90	$6.637 \cdot 10^5$
pigs	*#nodes: 441, #edges: 592, avg. Markov Blanket: 3.66*					
	1	0.74	$2.952 \cdot 10^3$	3	0.88	$2.362 \cdot 10^3$
	7	1.02	$1.511 \cdot 10^6$	8	MO	—
munin	*#nodes: 1041, #edges: 1397, avg. Markov Blanket: 3.54*					
	1	1.47	$3.648 \cdot 10^4$	3	1.37	$1.824 \cdot 10^7$
	9	1.79	$1.824 \cdot 10^{16}$	10	65.64	$1.153 \cdot 10^{18}$

bnlearn.com/bnrepository

BN	#obs	Time	EST	#obs	Time	EST
`hailfinder`	*#nodes: 56, #edges: 66, avg. Markov Blanket: 3.54*					
	5	0.63	$5.016 \cdot 10^5$	9	0.46	$9.048 \cdot 10^6$
`hepar2`	*#nodes: 70, #edges: 123, avg. Markov Blanket: 4.51*					
	1	1.84	$1.579 \cdot 10^2$	2	MO	—
`win95pts`	*#nodes: 76, #edges: 112, avg. Markov Blanket: 5.92*					
	1	0.36	$2.284 \cdot 10^3$	3	0.36	$4.296 \cdot 10^5$
	12	0.42	$3.973 \cdot 10^7$	17	61.73	$1.110 \cdot 10^{15}$
`pathfinder`	*#nodes: 135, #edges: 200, avg. Markov Blanket: 3.04*					
	1	0.53	$1.050 \cdot 10^4$	3	31.31	$2.872 \cdot 10^4$
	7	5.44	∞	7	480.83	∞
`andes`	*#nodes: 223, #edges: 338, avg. Markov Blanket: 5.61*					
	1	MO	—	3	1.66	$5.251 \cdot 10^3$
	7	0.99	$8.904 \cdot 10^4$	9	0.90	$6.637 \cdot 10^5$
`pigs`	*#nodes: 441, #edges: 592, avg. Markov Blanket: 3.66*					
	1	0.74	$2.952 \cdot 10^3$	3	0.88	$2.362 \cdot 10^3$
	7	1.02	$1.511 \cdot 10^6$	8	MO	—
`munin`	*#nodes: 1041, #edges: 1397, avg. Markov Blanket: 3.54*					
	1	1.47	$3.648 \cdot 10^4$	3	1.37	$1.824 \cdot 10^7$
	9	1.79	$1.824 \cdot 10^{16}$	10	65.64	$1.153 \cdot 10^{18}$

`bnlearn.com/bnrepository`

BN	#obs	Time	EST	#obs	Time	EST
`hailfinder`	*#nodes: 56, #edges: 66, avg. Markov Blanket: 3.54*					
	5	0.63	$5.016 \cdot 10^5$	9	0.46	$9.048 \cdot 10^6$
`hepar2`	*#nodes: 70, #edges: 123, avg. Markov Blanket: 4.51*					
	1	1.84	$1.579 \cdot 10^2$	2	MO	—
`win95pts`	*#nodes: 76, #edges: 112, avg. Markov Blanket: 5.92*					
	1	0.36	$2.284 \cdot 10^3$	3	0.36	$4.296 \cdot 10^5$
	12	0.42	$3.973 \cdot 10^7$	17	61.73	$1.110 \cdot 10^{15}$
`pathfinder`	*#nodes: 135, #edges: 200, avg. Markov Blanket: 3.04*					
	1	0.53	$1.050 \cdot 10^4$	3	31.31	$2.872 \cdot 10^4$
	7	5.44	∞	7	480.83	∞
`andes`	*#nodes: 223, #edges: 338, avg. Markov Blanket: 5.61*					
	1	MO	—	3	1.66	$5.251 \cdot 10^3$
	7	0.99	$8.904 \cdot 10^4$	9	0.90	$6.637 \cdot 10^5$
`pigs`	*#nodes: 441, #edges: 592, avg. Markov Blanket: 3.66*					
	1	0.74	$2.952 \cdot 10^3$	3	0.88	$2.362 \cdot 10^3$
	7	1.02	$1.511 \cdot 10^6$	8	MO	—
`munin`	*#nodes: 1041, #edges: 1397, avg. Markov Blanket: 3.54*					
	1	1.47	$3.648 \cdot 10^4$	3	1.37	$1.824 \cdot 10^7$
	9	1.79	$1.824 \cdot 10^{16}$	10	65.64	$1.153 \cdot 10^{18}$

`bnlearn.com/bnrepository`

Summary

The program analysis perspective:

The Bayesian Network perspective:

Summary

The program analysis perspective:

- A fairly checkable condition (t–i.i.d.–ness)

The Bayesian Network perspective:

Summary

The program analysis perspective:

- A fairly checkable condition (t–i.i.d.–ness)
- A proof rule yielding exact expected runtimes.

The Bayesian Network perspective:

Summary

The program analysis perspective:

- A fairly checkable condition (t–i.i.d.–ness)
- A proof rule yielding exact expected runtimes.
  No need for guessing invariants, supermartingales, or alike

The Bayesian Network perspective:

Summary

The program analysis perspective:

- A fairly checkable condition (t–i.i.d.–ness)
- A proof rule yielding exact expected runtimes.
  No need for guessing invariants, supermartingales, or alike
- A syntactic language fragment of probabilistic programs
  to which our proof rule *automatically* applies

The Bayesian Network perspective:

Summary

The program analysis perspective:

- A fairly checkable condition (t–i.i.d.–ness)
- A proof rule yielding exact expected runtimes.
  No need for guessing invariants, supermartingales, or alike
- A syntactic language fragment of probabilistic programs
  to which our proof rule *automatically* applies

The Bayesian Network perspective:

- A method to obtain expected sampling times
  using techniques from program analysis

Summary

The program analysis perspective:

- A fairly checkable condition (t–i.i.d.–ness)
- A proof rule yielding exact expected runtimes.
  No need for guessing invariants, supermartingales, or alike
- A syntactic language fragment of probabilistic programs
  to which our proof rule *automatically* applies

The Bayesian Network perspective:

- A method to obtain expected sampling times
  using techniques from program analysis
- We are not solving an NP–hard problem more efficiently

Summary

The program analysis perspective:

- A fairly checkable condition (t–i.i.d.–ness)
- A proof rule yielding exact expected runtimes.
  No need for guessing invariants, supermartingales, or alike
- A syntactic language fragment of probabilistic programs
  to which our proof rule *automatically* applies

The Bayesian Network perspective:

- A method to obtain expected sampling times
  using techniques from program analysis
- We are not solving an NP–hard problem more efficiently
- We have shown that our method works on "very large"
  and even "massive" Bayesian networks

Backup: t–i.i.d.–ness

Backup: t–i.i.d.–ness

- $\mathrm{Vars}\,(t) \;=\; \{\, x \mid \exists\, \sigma, v, v' \colon t(\sigma\,[x \mapsto v]) \;\neq\; t(\sigma\,[x \mapsto v'])\,\}$

Backup: t–i.i.d.–ness

- $\text{Vars}(t) = \{ x \mid \exists \sigma, v, v' : t(\sigma[x \mapsto v]) \neq t(\sigma[x \mapsto v']) \}$
- $\text{Mod}(C)$: set of variables that occur on left hand side of an assignment in program C

Backup: t–i.i.d.–ness

- $\mathrm{Vars}(t) = \{ x \mid \exists \sigma, v, v' : t(\sigma[x \mapsto v]) \neq t(\sigma[x \mapsto v']) \}$
- $\mathrm{Mod}(C)$: set of variables that occur on left hand side of an assignment in program C
- $t \not\perp\!\!\!\perp C$ iff $\mathrm{Vars}(f) \cap \mathrm{Mod}(C) = \emptyset$

Backup: t–i.i.d.–ness

- $\mathrm{Vars}\,(t) \;=\; \{\, x \mid \exists\, \sigma, v, v' \colon t(\sigma\,[x \mapsto v]) \;\neq\; t(\sigma\,[x \mapsto v'])\,\}$
- $\mathrm{Mod}\,(C)$: set of variables that occur on left hand side of an assignment in program C
- $t \not\!\!\pitchfork C$ iff $\mathrm{Vars}\,(f) \cap \mathrm{Mod}\,(C) = \emptyset$

f–i.i.d.–ness

The loop $\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}$ is called t–i.i.d. iff

Backup: t–i.i.d.–ness

- $\mathrm{Vars}\,(t) \;=\; \{\, x \mid \exists\, \sigma, v, v' \colon t(\sigma\,[x \mapsto v]) \;\neq\; t(\sigma\,[x \mapsto v'])\,\}$
- $\mathrm{Mod}\,(C)$: set of variables that occur on left hand side of an assignment in program C
- $t \not\perp\!\!\!\perp C$ iff $\mathrm{Vars}\,(f) \cap \mathrm{Mod}\,(C) = \emptyset$

f–i.i.d.–ness

The loop $\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}$ is called t–i.i.d. iff

$$\mathrm{wp}\;[\![C]\!]\;([\varphi]) \not\perp\!\!\!\perp C \qquad \text{and} \qquad \mathrm{wp}\;[\![C]\!]\;([\neg\varphi] \cdot t) \not\perp\!\!\!\perp C\;.$$

Backup: The Coupon Collector's Inner Loop is i–i.i.d.

$$\begin{array}{l}\texttt{while}\,(\,cp[i] \neq 0\,)\,\{\\ \qquad i :\approx \texttt{Unif}[1 \ldots N]\\ \}\end{array}$$

Backup: The Coupon Collector's Inner Loop is i–i.i.d.

$$\begin{array}{l}\texttt{while}\,(\,cp[i] \neq 0\,)\,\{\\ \qquad i :\approx \texttt{Unif}[1 \ldots N]\\ \}\end{array}$$

- The coupon collector's inner loop is i–i.i.d.

Backup: The Coupon Collector's Inner Loop is i–i.i.d.

$$\begin{array}{l}\texttt{while}\,(\,cp[i] \neq 0\,)\,\{\\ \qquad i :\approx \texttt{Unif}[1 \ldots N]\\ \}\end{array}$$

- The coupon collector's inner loop is i–i.i.d.
  - wp $[\![i :\approx \ldots]\!]\,([cp[i] \neq 0]) \;=\; \frac{1}{N}\sum_{k=1}^{N}[cp[k] \neq 0] \;\not\preccurlyeq\; C$

Backup: The Coupon Collector's Inner Loop is i–i.i.d.

$$\begin{array}{l}\texttt{while}\,(\,cp[i] \neq 0\,)\,\{\\ \qquad i :\approx \texttt{Unif}[1 \ldots N]\\ \}\end{array}$$

- The coupon collector's inner loop is i–i.i.d.
  - wp $[\![i :\approx \ldots]\!]\,([cp[i] \neq 0]) \;=\; \frac{1}{N}\sum_{k=1}^{N}[cp[k] \neq 0] \;\not\Rightarrow\; C$
  - wp $[\![i :\approx \ldots]\!]\,([cp[i] = 0] \cdot i) \;=\; \frac{1}{N}\sum_{k=1}^{N}[cp[k] = 0] \cdot k \;\not\Rightarrow\; C$

Backup: The Coupon Collector's Inner Loop is i–i.i.d.

$$\begin{array}{l}\texttt{while}\,(\,cp[i] \neq 0\,)\,\{\\ \qquad i :\approx \texttt{Unif}[1 \ldots N]\\ \}\end{array}$$

- The coupon collector's inner loop is i–i.i.d.
  - wp $[\![i :\approx \ldots]\!]\,([cp[i] \neq 0]) \;=\; \frac{1}{N}\sum_{k=1}^{N}[cp[k] \neq 0] \;\not\Rightarrow\; C$
  - wp $[\![i :\approx \ldots]\!]\,([cp[i] = 0] \cdot i) \;=\; \frac{1}{N}\sum_{k=1}^{N}[cp[k] = 0] \cdot k \;\not\Rightarrow\; C$
- **In fact:** The inner loop is f–i.i.d. for any f

Backup: The Bayesian Network Language

$$
\begin{aligned}
C &\longrightarrow \mathit{Seq} \mid \texttt{repeat}\,\{\,\psi\,\}\,\texttt{until}\ (\,\mathit{Seq}\,) \mid C \fatsemi C \\
\mathit{Seq} &\longrightarrow \mathit{Seq} \fatsemi \mathit{Seq} \mid B_{x_1} \mid B_{x_2} \mid \dots \\
B_{x_i} &\longrightarrow x_i :\approx \sum_{j=1}^{n} p_j \cdot \langle a_j \rangle \mid \texttt{if}\ (\,\varphi\,)\,\{\,x_i :\approx \mu\,\}\,\texttt{else}\,\{\,B_{x_i}\,\}
\end{aligned}
$$

(rule exists for all $x_i \in \mathsf{Vars}$)