Probabilistic Programs
A Program Analysis Perspective on Expected Sampling Times

Kevin Batz    Benjamin L. Kaminski
Joost-Pieter Katoen    Christoph Matheja

RWTH Aachen University

ROCKS Meeting 2017, Münster
The Official Motivation™

**Speedup**
The Official Motivation™

**Speedup**

**Cryptography**
The Official Motivation™

Speedup

Cryptography

Randomized Algorithms
The Official Motivation™

Speedup

Cryptography

Randomized Algorithms

Inference

Formal verification of simple programs
The Official Motivation™

Speedup

Inference

Cryptography

Machine Learning

Randomized Algorithms
The Official Motivation™

- Speedup
- Cryptography
- Randomized Algorithms
- Inference
- Machine Learning
- Probabilistic Graphical Models
The Official Motivation™

**Speedup**

**Cryptography**

**Randomized Algorithms**

**Inference**

**Machine Learning**

**Probabilistic Graphical Models**

Formal verification?
The Official Motivation™

- Speedup
- Cryptography
- Randomized Algorithms
- Inference
- Machine Learning
- Probabilistic Graphical Models

Formal verification?

of simple programs
The Official Motivation™

- Speedup
- Cryptography
- Randomized Algorithms
- Inference
- Machine Learning
- Probabilistic Graphical Models

Formal verification?

of simple programs
of graphical models
A probabilistic program takes an input state $\sigma \in \Sigma$ and computes a (sub-)distribution of output states.
Probabilistic Programs in a Nutshell

A probabilistic program takes an input state \( \sigma \in \Sigma \) and computes a (sub-)distribution of output states.

**Definition (Syntax of Probabilistic Programs)**

\[
\begin{align*}
C & \rightarrow \text{skip} \\
\end{align*}
\]
Probabilistic Programs in a Nutshell

A probabilistic program takes an input state $\sigma \in \Sigma$ and computes a (sub-)distribution of output states.

Definition (Syntax of Probabilistic Programs)

- $C \rightarrow \text{skip}$
- $| \ x : \approx \mu$

$\mu : \Sigma \rightarrow \text{Dist}(Q)$ is distribution expression.

Examples:
- $\mu = \frac{1}{2} \cdot \langle 3 \rangle + \frac{1}{2} \cdot \langle 7 \rangle$
- $\mu = \text{Unif}[1 \ldots 23]$, ...
Probabilistic Programs in a Nutshell

A probabilistic program takes an input state $\sigma \in \Sigma$ and computes a (sub-)distribution of output states.

**Definition (Syntax of Probabilistic Programs)**

$$
\begin{align*}
C & \quad \rightarrow \quad \text{skip} \\
& \quad | \quad x \sim \mu \\
\end{align*}
$$

- $\mu : \Sigma \rightarrow \text{Dist}(\mathbb{Q})$ is distribution expression.
Probabilistic Programs in a Nutshell

A probabilistic program takes an input state $\sigma \in \Sigma$ and computes a (sub-)distribution of output states.

### Definition (Syntax of Probabilistic Programs)

<table>
<thead>
<tr>
<th>$\mathcal{C}$</th>
<th>$\xrightarrow{}$</th>
<th>skip</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\mid$</td>
<td>$x \approx \mu$</td>
</tr>
</tbody>
</table>

- $\mu : \Sigma \rightarrow \text{Dist}(Q)$ is distribution expression.
- Examples: $\mu = \frac{1}{2} \cdot \langle 3 \rangle + \frac{1}{2} \cdot \langle 7 \rangle$, $\mu = \text{Unif}[1 \ldots 23]$, ...
A probabilistic program takes an input state $\sigma \in \Sigma$ and computes a (sub-)distribution of output states.

**Definition (Syntax of Probabilistic Programs)**

- $C \rightarrow \text{skip}$
- $\mid x \colon \approx \mu$
- $\mid C; C$

- $\mu : \Sigma \rightarrow \text{Dist}(\mathbb{Q})$ is distribution expression.

- Examples: $\mu = \frac{1}{2} \cdot \langle 3 \rangle + \frac{1}{2} \cdot \langle 7 \rangle$, $\mu = \text{Unif}[1 \ldots 23]$, ...
A probabilistic program takes an input state $\sigma \in \Sigma$ and computes a (sub-)distribution of output states.

**Definition (Syntax of Probabilistic Programs)**

- $C \rightarrow \text{skip}$
- $\mid x \approx \mu$
- $\mid C; C$
- $\mid \text{if} (\varphi) \{C\} \text{else} \{C\} \quad ([\varphi] : \Sigma \rightarrow \{0, 1\})$  

- $\mu : \Sigma \rightarrow \text{Dist}(\mathbb{Q})$ is distribution expression.
- Examples: $\mu = 1/2 \cdot \langle 3 \rangle + 1/2 \cdot \langle 7 \rangle$, $\mu = \text{Unif}[1 \ldots 23]$, ...
A probabilistic program takes an input state $\sigma \in \Sigma$ and computes a (sub-)distribution of output states.

**Definition (Syntax of Probabilistic Programs)**

- $C \rightarrow \text{skip}$
- $\mid x \approx \mu$
- $\mid C; C$
- $\mid \text{if}(\varphi)\{C\} \text{else}\{C\}$
  
  \begin{align*}
  ([\varphi] : \Sigma \rightarrow \{0, 1\})
  
  \end{align*}
- $\mid \text{while}(\varphi)\{C\}$
- $\mid \text{repeat}\{C\} \text{until}(\varphi)$

- $\mu : \Sigma \rightarrow \text{Dist}(\mathbb{Q})$ is distribution expression.

- Examples: $\mu = 1/2 \cdot \langle 3 \rangle + 1/2 \cdot \langle 7 \rangle$, $\mu = \text{Unif}[1 \ldots 23]$, \ldots
Semantics of Probabilistic Programs

Operational: infinite-state DTMCs [Gretz et al.]
Semantics of Probabilistic Programs

Operational: infinite-state DTMCs [Gretz et al.]

Denotational: weakest pre-expectation calculus (this talk)
Semantics of Probabilistic Programs

Operational: infinite-state DTMCs [Gretz et al.]

Denotational: weakest pre-expectation calculus (this talk)

An expectation is a random variable $f : \Sigma \rightarrow \mathbb{R}_{\geq 0}^{(\infty)}$
Semantics of Probabilistic Programs

Operational: infinite-state DTMCs [Gretz et al.]

Denotational: weakest pre-expectation calculus (this talk)

An expectation is a random variable \( f : \Sigma \rightarrow \mathbb{R}_{\geq 0}^{(\infty)} \)

E.g. \( x^2 \) or \( [x > y] \)
Semantics of Probabilistic Programs

Operational: infinite-state DTMCs [Gretz et al.]

Denotational: weakest pre-expectation calculus (this talk)

An expectation is a random variable \( f : \Sigma \rightarrow \mathbb{R}_{\geq 0}^{(\infty)} \)

E.g. \( x^2 \) or \([x > y] \)

\( \mathcal{E} \) is the set of expectations
Semantics of Probabilistic Programs

Operational: \textit{infinite-state} DTMCs [Gretz et al.]

Denotational: \textit{weakest pre-expectation calculus} (this talk)

An expectation is a random variable $f : \Sigma \rightarrow \mathbb{R}^{(\infty)}_{\geq 0}$

E.g. $x^2$ or $[x > y]$

$\mathbb{E}$ is the set of expectations

The Standard \textit{wp} Transformer [Kozen, McIver & Morgan]

$\text{wp}[C] : \mathbb{E} \rightarrow \mathbb{E}$ is a \textit{backwards moving} expectation transformer
Semantics of Probabilistic Programs

Operational: infinite-state DTMCs [Gretz et al.]

Denotational: weakest pre-expectation calculus (this talk)

An expectation is a random variable \( f : \Sigma \rightarrow \mathbb{R}^{(\infty)}_{\geq 0} \)

E.g. \( x^2 \) or \( [x > y] \)

\( \mathcal{E} \) is the set of expectations

The Standard wp Transformer [Kozen, McIver & Morgan]

\( \text{wp}[C] : \mathcal{E} \rightarrow \mathcal{E} \) is a backwards moving expectation transformer

\( C \)
Semantics of Probabilistic Programs

Operational: infinite-state DTMCs [Gretz et al.]

Denotational: weakest pre-expectation calculus (this talk)

An expectation is a random variable \( f : \Sigma \rightarrow \mathbb{R}_{\geq 0}^{(\infty)} \)

E.g. \( x^2 \) or \( [x > y] \)

\( \mathbb{E} \) is the set of expectations

The Standard wp Transformer [Kozen, McIver & Morgan]

\( \text{wp}[C] : \mathbb{E} \rightarrow \mathbb{E} \) is a backwards moving expectation transformer

\( C \)

\( f \)

post-expectation \( f \)

evaluated in final states
Semantics of Probabilistic Programs

Operational: infinite-state DTMCs [Gretz et al.]

Denotational: weakest pre-expectation calculus (this talk)

An expectation is a random variable $f : \Sigma \to \mathbb{R}^{\geq 0}$

E.g. $x^2$ or $[x > y]$  

$\mathbb{E}$ is the set of expectations

The Standard $wp$ Transformer [Kozen, Mclver & Morgan]

$wp[C] : \mathbb{E} \rightarrow \mathbb{E}$ is a backwards moving expectation transformer

$\text{wp} [C] (f)$  

$C$  

$f$  

weakest pre–expectation of $C$ with respect to $f$  

post–expectation $f$  

evaluated in initial states  

evaluated in final states
The \( \text{wp calculus} \) [Mclver & Morgan]

\[
C \quad \text{wp} \ [C] (f)
\]
<table>
<thead>
<tr>
<th>( C )</th>
<th>( \text{wp} \ [C] (f) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{skip}</td>
<td>( f )</td>
</tr>
</tbody>
</table>

The wp calculus [Mclver & Morgan]
The wp calculus [Mclver & Morgan]

<table>
<thead>
<tr>
<th>$C'$</th>
<th>$\text{wp } \llbracket C \rrbracket (f)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>skip</td>
<td>$f$</td>
</tr>
<tr>
<td>$x ::= \mu$</td>
<td>$\lambda \sigma \cdot E_{\llbracket \mu \rrbracket} (\lambda v \cdot f<a href="%5Csigma">x/v</a>)$</td>
</tr>
</tbody>
</table>
The wp calculus [Mclver & Morgan]

<table>
<thead>
<tr>
<th>( C )</th>
<th>( \text{wp} \ [C] \ (f) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>skip</td>
<td>( f )</td>
</tr>
<tr>
<td>( x : \approx \mu )</td>
<td>( \lambda \sigma \cdot E_{[\mu]}(\sigma) (\lambda v \cdot f<a href="%5Csigma">x/v</a>) )</td>
</tr>
<tr>
<td>if(( \varphi )) { ( C_1 ) } else { ( C_2 ) }</td>
<td>( [\varphi] \cdot \text{wp} \ [C_1] \ (f) + [\neg \varphi] \cdot \text{wp} \ [C_2] \ (f) )</td>
</tr>
</tbody>
</table>
The wp calculus [Mclver & Morgan]

<table>
<thead>
<tr>
<th>$C'$</th>
<th>$\text{wp } <a href="f">C'</a>$</th>
</tr>
</thead>
<tbody>
<tr>
<td>skip</td>
<td>$f$</td>
</tr>
<tr>
<td>$x \approx \mu$</td>
<td>$\lambda\sigma \cdot E_{[\mu]}(\sigma) (\lambda v \cdot f<a href="%5Csigma">x/v</a>)$</td>
</tr>
<tr>
<td>if$(\varphi){C_1} \text{ else } {C_2}$</td>
<td>$[\varphi] \cdot \text{wp } <a href="f">C_1</a> + [\neg \varphi] \cdot \text{wp } <a href="f">C_2</a>$</td>
</tr>
<tr>
<td>$C_1; C_2$</td>
<td>$\text{wp } [C_1](\text{wp } <a href="f">C_2</a>)$</td>
</tr>
</tbody>
</table>
### The wp calculus [McIver & Morgan]

<table>
<thead>
<tr>
<th>$C'$</th>
<th>$\text{wp } [C'] (f)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>skip</td>
<td>$f$</td>
</tr>
<tr>
<td>$x : \approx \mu$</td>
<td>$\lambda \sigma . \mathbf{E}_{[\mu]}(\sigma) (\lambda v . f<a href="%5Csigma">x/v</a>)$</td>
</tr>
<tr>
<td>if($\varphi$){$C_1$} else {$C_2$}</td>
<td>$[\varphi] \cdot \text{wp } [C_1] (f) + [\neg \varphi] \cdot \text{wp } [C_2] (f)$</td>
</tr>
<tr>
<td>$C_1; C_2$</td>
<td>$\text{wp } [C_1] (\text{wp } [C_2] (f))$</td>
</tr>
<tr>
<td>while($\varphi$){$C'$}</td>
<td>$\text{lfp } X . [\neg \varphi] \cdot f + [\varphi] \cdot \text{wp } [C'] (X)$</td>
</tr>
</tbody>
</table>
The wp calculus [Mclver & Morgan]

<table>
<thead>
<tr>
<th>$C'$</th>
<th>$\text{wp } [C] (f)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>skip</td>
<td>$f$</td>
</tr>
<tr>
<td>$x \approx \mu$</td>
<td>$\lambda \sigma \cdot \textbf{E}_{[\mu]}(\sigma) (\lambda v \cdot f<a href="%5Csigma">x/v</a>)$</td>
</tr>
<tr>
<td>if ($\varphi$) { $C_1$ } else { $C_2$ }</td>
<td>$[\varphi] \cdot \text{wp } [C_1] (f) + [\neg \varphi] \cdot \text{wp } [C_2] (f)$</td>
</tr>
<tr>
<td>$C_1; C_2$</td>
<td>$\text{wp } [C_1] (\text{wp } [C_2] (f))$</td>
</tr>
<tr>
<td>while ($\varphi$) { $C'$ }</td>
<td>$\text{lfp } X \cdot [\neg \varphi] \cdot f + [\varphi] \cdot \text{wp } [C'] (X)$</td>
</tr>
<tr>
<td>repeat { $C'$ } until ($\varphi$)</td>
<td>$\text{wp } [C'; \text{while } (\neg \varphi) { C' }] (f)$</td>
</tr>
</tbody>
</table>
Computing weakest pre-expectations

<table>
<thead>
<tr>
<th>$C$</th>
<th>$\text{wp } [C] (f)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x :\approx \mu$</td>
<td>$\lambda\sigma \cdot E_{\mu}[\sigma] (\lambda v \cdot f<a href="%5Csigma">x/v</a>)$</td>
</tr>
<tr>
<td>$\text{if}(\varphi){C_1} \text{ else } {C_2}$</td>
<td>$[\varphi] \cdot \text{wp } [C_1] (f) + [\neg \varphi] \cdot \text{wp } [C_2] (f)$</td>
</tr>
</tbody>
</table>

Example (Expected value of $x^2$ after program execution?)

if ($y > 0$) {skip} else { $y := 0$ };

$x :\approx 1/3 \cdot \langle 2y \rangle + 2/3 \cdot \langle x \rangle$
Computing weakest pre-expectations

<table>
<thead>
<tr>
<th>$C$</th>
<th>$\text{wp} \ [C] \ (f)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \approx \mu$</td>
<td>$\lambda \sigma \cdot \mathsf{E}_{\left[ \mu \right]}(\sigma) \left( \lambda v \cdot f<a href="%5Csigma">x/v</a> \right)$</td>
</tr>
<tr>
<td>$\text{if}(\varphi) {C_1} \text{ else } {C_2}$</td>
<td>$[\varphi] \cdot \text{wp} \ [C_1] \ (f) + [\neg \varphi] \cdot \text{wp} \ [C_2] \ (f)$</td>
</tr>
</tbody>
</table>

Example (Expected value of $x^2$ after program execution?)

if $(y > 0)$ \{ skip \} else \{ $y := 0$ \} ;

$$x \approx \frac{1}{3} \cdot \langle 2y \rangle + \frac{2}{3} \cdot \langle x \rangle$$

$$x^2$$
Computing weakest pre-expectations

<table>
<thead>
<tr>
<th>C</th>
<th>( \text{wp } <a href="f">C</a> )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x \approx \mu )</td>
<td>( \lambda \sigma \cdot E_{[\mu]}(\sigma) (\lambda v \cdot f<a href="%5Csigma">x/v</a>) )</td>
</tr>
<tr>
<td>( \text{if}(\varphi){C_1} \text{ else } {C_2} )</td>
<td>( [\varphi] \cdot \text{wp } <a href="f">C_1</a> + [\neg \varphi] \cdot \text{wp } <a href="f">C_2</a> )</td>
</tr>
</tbody>
</table>

Example (Expected value of \( x^2 \) after program execution?)

\[
\text{if}(y > 0)\{\text{skip}\} \text{ else } \{y := 0\};
\]
\[
\frac{1}{3} \cdot x^2[x/2y] + \frac{2}{3} \cdot x^2[x/x]
\]
\[
x \approx \frac{1}{3} \cdot \langle 2y \rangle + \frac{2}{3} \cdot \langle x \rangle
\]
\[
x^2
\]
Computing weakest pre-expectations

<table>
<thead>
<tr>
<th>$C$</th>
<th>$\text{wp} \ [C] \ (f)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \approx \mu$</td>
<td>$\lambda \sigma \cdot \mathbb{E}_{[\mu]}(\sigma) (\lambda v \cdot f<a href="%5Csigma">x/v</a>)$</td>
</tr>
<tr>
<td>if($\varphi$) {$C_1$} else {$C_2$}</td>
<td>$[\varphi] \cdot \text{wp} [C_1] (f) + [\neg \varphi] \cdot \text{wp} [C_2] (f)$</td>
</tr>
</tbody>
</table>

Example (Expected value of $x^2$ after program execution?)

if($y > 0$) {skip} else {$y := 0$};

$1/3 \cdot (2y)^2 + 2/3 \cdot x^2$

$x :\approx 1/3 \cdot \langle 2y \rangle + 2/3 \cdot \langle x \rangle$

$x^2$
Computing weakest pre-expectations

<table>
<thead>
<tr>
<th>$C$</th>
<th>$\text{wp } [C] (f)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \approx \mu$</td>
<td>$\lambda \sigma \cdot E_{[\mu]}(\sigma) (\lambda v \cdot f<a href="%5Csigma">x/v</a>)$</td>
</tr>
<tr>
<td>if ($\varphi$) ${C_1}$ else ${C_2}$</td>
<td>$[\varphi] \cdot \text{wp } [C_1] (f) + [\neg \varphi] \cdot \text{wp } [C_2] (f)$</td>
</tr>
</tbody>
</table>

Example (Expected value of $x^2$ after program execution?)

$$[y > 0] \cdot \text{wp } [\text{skip}] (1/3 \ldots ) + [y \leq 0] \cdot \text{wp } [y := 0] (1/3 \ldots )$$

if ($y > 0$) $\{\text{skip}\}$ else $\{y := 0\}$;

$1/3 \cdot (2y)^2 + 2/3 \cdot x^2$

$x \approx 1/3 \cdot \langle 2y \rangle + 2/3 \cdot \langle x \rangle$

$x^2$
Computing weakest pre-expectations

<table>
<thead>
<tr>
<th>$C$</th>
<th>$\text{wp } \mathbb{E}<a href="f">C</a>$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x : \approx \mu$</td>
<td>$\lambda\sigma \cdot \mathbb{E}_{[\mu]}(\sigma)(\lambda v \cdot f<a href="%5Csigma">x/v</a>)$</td>
</tr>
<tr>
<td>$\text{if}(\varphi){C_1} \text{ else } {C_2}$</td>
<td>$[\varphi] \cdot \text{wp } \mathbb{E}<a href="f">C_1</a> + [\neg \varphi] \cdot \text{wp } \mathbb{E}<a href="f">C_2</a>$</td>
</tr>
</tbody>
</table>

Example (Expected value of $x^2$ after program execution?)

\[
[y > 0] \cdot (\frac{1}{3} \cdot (2y)^2 + \frac{2}{3} \cdot x^2) + [y \leq 0] \cdot \frac{2}{3} \cdot x^2
\]

if $(y > 0)$ \{skip\} else \{\(y := 0\)\};

$\frac{1}{3} \cdot (2y)^2 + \frac{2}{3} \cdot x^2$

\[
x : \approx \frac{1}{3} \cdot \langle 2y \rangle + \frac{2}{3} \cdot \langle x \rangle
\]

$x^2$
The ert calculus [Kaminski et al.]

- Similar transformer to reason about expected runtimes
The ert calculus [Kaminski et al.]

- Similar transformer to reason about expected runtimes
- ert$[C] : \mathbb{E} \rightarrow \mathbb{E}$ is a ”post-runtime” transformer
The ert calculus [Kaminski et al.]

- Similar transformer to reason about **expected runtimes**
- \( ert[C] : E \rightarrow E \) is a "post-runtime" transformer
- \( ert [C] (0) \) is the expected runtime of \( C \)
The ert calculus [Kaminski et al.]

- Similar transformer to reason about expected runtimes
- $\text{ert} [C] : E \rightarrow E$ is a "post-runtime" transformer
- $\text{ert} [C] (0)$ is the expected runtime of $C$

<table>
<thead>
<tr>
<th>$C$</th>
<th>$\text{ert} \ [C'] \ (f)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>skip</td>
<td>$1 + f$</td>
</tr>
<tr>
<td>$x : \approx \mu$</td>
<td>$1 + \lambda \sigma \cdot E_{[\mu]}(\sigma) (\lambda v \cdot f<a href="%5Csigma">x/v</a>)$</td>
</tr>
<tr>
<td>if $(\varphi) {C_1} \text{ else } {C_2}$</td>
<td>$1 + [\varphi] \cdot \text{ert} \ [C_1] \ (f) + [-\varphi] \cdot \text{ert} \ [C_2] \ (f)$</td>
</tr>
<tr>
<td>$C_1; C_2$</td>
<td>$\text{ert} \ [C_1] \ (\text{ert} \ [C_2] \ (f))$</td>
</tr>
<tr>
<td>while $(\varphi) {C'}$</td>
<td>lfp $X \cdot 1 + [-\varphi] \cdot f + [\varphi] \cdot \text{ert} \ [C'] \ (X)$</td>
</tr>
<tr>
<td>repeat ${C'}$ until $(\varphi)$</td>
<td>$\text{ert} \ [C'; \text{while } (\neg \varphi) {C'}] \ (f)$</td>
</tr>
</tbody>
</table>
Example: The Coupon Collector Problem [Erdös, 1961]

- There are $N$ different coupons hidden in boxes of cereal.
Example: The Coupon Collector Problem [Erdös, 1961]

- There are $N$ different coupons hidden in boxes of cereal.
- **Goal**: Collect all coupons to trade them for a prize.
Example: The Coupon Collector Problem [Erdös, 1961]

- There are $N$ different coupons hidden in boxes of cereal.
- **Goal**: Collect all coupons to trade them for a prize.
- What is the expected number of boxes we have to open?
Example: The Coupon Collector Problem [Erdös, 1961]

- There are $N$ different coupons hidden in boxes of cereal.
- **Goal**: Collect all coupons to trade them for a prize.
- What is the expected number of boxes we have to open?

```plaintext
remaining := N;
while (remaining > 0) {
    x := Unif[1...N];
    while (x > remaining) {x := Unif[1...N]};
    remaining := remaining − 1
}
```
Example: The Coupon Collector Problem [Erdös, 1961]

- There are $N$ different coupons hidden in boxes of cereal.
- **Goal:** Collect all coupons to trade them for a prize.
- What is the expected number of boxes we have to open?

```plaintext
remaining := N;
while (remaining > 0) {
  x := Unif[1...N];
  while (x > remaining) {x := Unif[1...N]};
  remaining := remaining – 1
}
```

- ert allows analysis on source code level [Kaminski et al.]
Example: The Coupon Collector Problem [Erdös, 1961]

- There are $N$ different coupons hidden in boxes of cereal.
- Goal: Collect all coupons to trade them for a prize.
- What is the expected number of boxes we have to open?

```plaintext
remaining := N;
while (remaining > 0) {
  x := Unif[1...N];
  while (x > remaining) { x := Unif[1...N] };
  remaining := remaining − 1
}
```

- ert allows analysis on source code level [Kaminski et al.]
- ert $\mathbb{E}[\text{coup. coll.}] (0) \in \Theta(N \cdot \log N)$
Applying ert to the Coupon Collector
Applying $e^{ert}$ to the Coupon Collector

\[ \text{cp} = [0, \ldots, 0], \]
\[ i = \text{uniform}(0, N), \]
\[ X = N; \]
\[ + \text{while } (X > 0) \{ \]
\[ x = \text{uniform}(0, N), \]
\[ \text{if } (\text{cp}[x] \neq 0) \{ \]
\[ \text{cp}[x] = 0; \]
\[ X = X - 1; \}
\[ \}
\[ \}

\[ H(Y) = \frac{1}{Y} \left( 3 + \sum_{k=0}^{\infty} \left( \frac{1}{N} \right)^k \cdot 2 + 2 + 1 \right) \]
\[ + \left[ x = 0 \right] \left( \frac{1}{N} \sum_{k=0}^{\infty} \left( \frac{1}{N} \right)^k \cdot 2 + 2 + 1 \right) \]
\[ + \left[ x = 1 \right] \left( \frac{1}{N} \sum_{k=0}^{\infty} \left( \frac{1}{N} \right)^k \cdot 2 + 2 + 1 \right) \]
\[ + \left[ x = 2 \right] \left( \frac{1}{N} \sum_{k=0}^{\infty} \left( \frac{1}{N} \right)^k \cdot 2 + 2 + 1 \right) \]
\[ + \left[ x = 3 \right] \left( \frac{1}{N} \sum_{k=0}^{\infty} \left( \frac{1}{N} \right)^k \cdot 2 + 2 + 1 \right) \]
\[ \frac{A}{\lambda} + 1 \left( \frac{1}{N} \sum_{k=0}^{\infty} \left( \frac{1}{N} \right)^k \cdot 2 + 2 + 1 \right) \]
\[ + 1 \left( \frac{1}{N} \sum_{k=0}^{\infty} \left( \frac{1}{N} \right)^k \cdot 2 + 2 + 1 \right) \]
\[ + 1 + 1 \left( \frac{1}{N} \sum_{k=0}^{\infty} \left( \frac{1}{N} \right)^k \cdot 2 + 2 + 1 \right) \]
Applying ert to the Coupon Collector

\[ I_{\text{out}} = 1 + \sum_{x=0}^{\infty} \left( 4 + \sum_{k=0}^{\infty} \frac{(\text{col } x)^k}{N^k} \cdot 2 \right) \]

\[ + \sum_{x=0}^{\infty} \sum_{k=0}^{\infty} \frac{(\text{col } x)^k}{N^k} \cdot 2 \]

\[ = 1 + \sum_{x=0}^{\infty} \left( 4 + \sum_{k=0}^{\infty} \frac{(\text{col } x)^k}{N^k} \cdot 2 \right) + \sum_{x=0}^{\infty} \sum_{k=0}^{\infty} \frac{(\text{col } x)^k}{N^k} \cdot 2 \]

\[ = 1 + \sum_{x=0}^{\infty} \left( 4 + \sum_{k=0}^{\infty} \frac{(\text{col } x)^k}{N^k} \cdot 2 \right) + \sum_{x=0}^{\infty} \sum_{k=0}^{\infty} \frac{(\text{col } x)^k}{N^k} \cdot 2 \]

\[ = I_{\text{out}} + \sum_{x=0}^{\infty} \left( 4 + \sum_{k=0}^{\infty} \frac{(\text{col } x)^k}{N^k} \cdot 2 \right) + \sum_{x=0}^{\infty} \sum_{k=0}^{\infty} \frac{(\text{col } x)^k}{N^k} \cdot 2 \]

\[ = I_{\text{out}} + \sum_{x=0}^{\infty} \left( 4 + \sum_{k=0}^{\infty} \frac{(\text{col } x)^k}{N^k} \cdot 2 \right) + \sum_{x=0}^{\infty} \sum_{k=0}^{\infty} \frac{(\text{col } x)^k}{N^k} \cdot 2 \]

\[ = I_{\text{out}} + \sum_{x=0}^{\infty} \left( 4 + \sum_{k=0}^{\infty} \frac{(\text{col } x)^k}{N^k} \cdot 2 \right) + \sum_{x=0}^{\infty} \sum_{k=0}^{\infty} \frac{(\text{col } x)^k}{N^k} \cdot 2 \]

\[ = I_{\text{out}} + \sum_{x=0}^{\infty} \left( 4 + \sum_{k=0}^{\infty} \frac{(\text{col } x)^k}{N^k} \cdot 2 \right) + \sum_{x=0}^{\infty} \sum_{k=0}^{\infty} \frac{(\text{col } x)^k}{N^k} \cdot 2 \]

\[ = I_{\text{out}} + \sum_{x=0}^{\infty} \left( 4 + \sum_{k=0}^{\infty} \frac{(\text{col } x)^k}{N^k} \cdot 2 \right) + \sum_{x=0}^{\infty} \sum_{k=0}^{\infty} \frac{(\text{col } x)^k}{N^k} \cdot 2 \]

\[ = I_{\text{out}} + \sum_{x=0}^{\infty} \left( 4 + \sum_{k=0}^{\infty} \frac{(\text{col } x)^k}{N^k} \cdot 2 \right) + \sum_{x=0}^{\infty} \sum_{k=0}^{\infty} \frac{(\text{col } x)^k}{N^k} \cdot 2 \]
Applying ert to the Coupon Collector

\[
\begin{align*}
\Gamma_{\omega} & = 1 + \sum_{k=0}^{\infty} [x > k] \left( 1 + \sum_{\omega \in \Omega} \left( \frac{x \cdot \text{col} + \varepsilon}{N} \right)^k \right) \left( 1 + \sum_{\omega \in \Omega} \left( \frac{\text{col} + x \cdot \varepsilon}{N} \right)^k \right)^{-1} \\
H(\Gamma_{\omega}) & = 1 + \sum_{k=0}^{\infty} \left[ x > k \right] \left( 1 + \sum_{\omega \in \Omega} \left( \frac{x \cdot \text{col} + \varepsilon}{N} \right)^k \right) \left( 1 + \sum_{\omega \in \Omega} \left( \frac{\text{col} + x \cdot \varepsilon}{N} \right)^k \right) \\
& + \sum_{k=0}^{\infty} \left[ x > k \right] \left( 1 + \sum_{\omega \in \Omega} \left( \frac{x \cdot \text{col} + \varepsilon}{N} \right)^k \right) \left( 1 + \sum_{\omega \in \Omega} \left( \frac{\text{col} + x \cdot \varepsilon}{N} \right)^k \right)^{-1} \\
& = 1 + \sum_{k=0}^{\infty} \left( 1 + \sum_{\omega \in \Omega} \left( \frac{x \cdot \text{col} + \varepsilon}{N} \right)^k \right) \left( 1 + \sum_{\omega \in \Omega} \left( \frac{\text{col} + x \cdot \varepsilon}{N} \right)^k \right)^{-1}. \\
\end{align*}
\]
Applying ert to the Coupon Collector

\[ F_{j}(J_{n}) \]

\[ = 1 + [ cp[i] = 0 ] \cdot f + [ cp[i] \neq 0 ] \cdot \left(1 + \frac{1}{N} \cdot \sum_{k=1}^{N} J_{f}^{k}[i/k]\right) \]

\[ = 1 + [ cp[i] = 0 ] \cdot f + [ cp[i] \neq 0 ] \cdot (1 + \frac{1}{N} \cdot \sum_{k=1}^{N} (1 + [ 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)\) (k does not)

\[ = 1 + [ cp[i] = 0 ] \cdot f + 2 \cdot [ cp[i] \neq 0 ] + \frac{[ cp[i] \neq 0 ]}{N} \cdot \sum_{k=1}^{N} ([ cp[k] = 0 ] \cdot f[i/k]) \]

\[ + [ cp[k] \neq 0 ] \cdot \sum_{\ell=0}^{n} \left( \frac{\#col}{N} \right)^{\ell} \cdot \left(2 + \frac{G(f)}{N}\right) \) (Def. \#col)

\[ = 1 + [ cp[i] = 0 ] \cdot f + [ cp[i] \neq 0 ] \cdot \left(2 + \frac{G(f)}{N}\right) \)

\[ + [ cp[i] \neq 0 ] \cdot \sum_{\ell=0}^{n} \left( \frac{\#col}{N} \right)^{\ell} \cdot \left(2 + \frac{G(f)}{N}\right) \) (Def. G)

\[ = 1 + [ cp[i] = 0 ] \cdot f + [ cp[i] \neq 0 ] \cdot \left(2 + \frac{G(f)}{N}\right) \]

\[ + [ cp[i] \neq 0 ] \cdot \sum_{\ell=0}^{n+1} \left( \frac{\#col}{N} \right)^{\ell} \cdot \left(2 + \frac{G(f)}{N}\right) \)

\[ = [ 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) \) (n+1)

Now, by Theorem 5, we obtain

\[ J^{3} = \lim_{n \to \infty} J_{n}^{3} \leq ert \{ C_{in} \} (g) \leq \lim_{n \to \infty} J_{n}^{3} = J^{3}. \]
Applying ert to the Coupon Collector

\[ F_j(J^0) = 1 + \left[ cp[i] = 0 \right] \cdot f + \left[ cp[i] \neq 0 \right] \cdot \left( 1 + \frac{1}{N} \cdot \sum_{k=1}^{N} J^i_{k} \cdot \frac{\#\text{col}}{N} \right) \]

\[ = 1 + \left[ cp[i] = 0 \right] \cdot f + \left[ cp[i] \neq 0 \right] \cdot \left( 1 + \frac{1}{N} \cdot \sum_{k=1}^{N} (1 + \left| cp[k] \right|) \cdot \frac{\#\text{col}}{N} \right) \]

\[ = 1 + \left[ cp[i] = 0 \right] \cdot f + \left[ cp[i] \neq 0 \right] \cdot \left( 1 + \frac{1}{N} \cdot \sum_{k=1}^{N} \sum_{\ell=0}^{n} \left( \frac{\#\text{col}}{N} \right)^\ell \cdot \left( 2 + \frac{G(f)}{N} \right) \right) \]

\[ = 1 + \left[ cp[i] = 0 \right] \cdot f + \left[ cp[i] \neq 0 \right] \cdot \sum_{k=1}^{n} \left( \frac{\#\text{col}}{N} \right)^\ell \cdot \left( 2 + \frac{G(f)}{N} \right) \]

\[ = 1 + \left[ cp[i] = 0 \right] \cdot f + \left[ cp[i] \neq 0 \right] \cdot \frac{\#\text{col}}{N} \cdot \sum_{\ell=0}^{n} \left( \frac{\#\text{col}}{N} \right)^\ell \cdot \left( 2 + \frac{G(f)}{N} \right) \]

\[ = 1 + \left[ cp[i] = 0 \right] \cdot f + \left[ cp[i] \neq 0 \right] \cdot \frac{\#\text{col}}{N} \cdot \sum_{\ell=0}^{n} \left( \frac{\#\text{col}}{N} \right)^\ell \cdot \left( 2 + \frac{G(f)}{N} \right) \]

\[ = 1 + \left[ cp[i] = 0 \right] \cdot f + \left[ cp[i] \neq 0 \right] \cdot \sum_{\ell=0}^{n} \left( \frac{\#\text{col}}{N} \right)^\ell \cdot \left( 2 + \frac{G(f)}{N} \right) \]

This is an excerpt of the proof of the inner loop

Now, by Theorem 5, we obtain

\[ J^3 = \lim_{n \to \infty} J^3_n \leq \text{ert} \cdot \text{G_M} \cdot (g) \leq \lim_{n \to \infty} J^3_n = J^3. \]
The actual motivation

There should be simple proofs for simple programs!
Independent and Identically Distributed Loops

Definition (Universally i.i.d. Loops)

A loop $\text{while } (\phi) \{ C \}$ is universally i.i.d. iff

$$\text{wp}_{J\cdot C}K(\phi) = \text{wp}_{J\cdot C}K(f).$$

Example

while $(x > r)$ \{ $x \approx \text{Unif}[1 \ldots N]$ \} is universally i.i.d.
Definition (Universally i.i.d. Loops)

A loop $\text{while}(\phi) \{$ $C$ $\}$ is universally i.i.d. iff for all $f \in E$: $wp \{ J_C K \phi \} \cdot wp \{ J_C K f \} = wp \{ J_C K \phi \} \cdot wp \{ J_C K f \}.$

Example: $\text{while}(x > r) \{$ $x := \text{Unif}[1...N] \}$ is universally i.i.d.
**Definition (Universally i.i.d. Loops)**

A loop while \((\varphi) \{C\}\) is universally i.i.d. iff
Independent and Identically Distributed Loops

Definition (Universally i.i.d. Loops)

A loop while (ϕ) \{C\} is universally i.i.d. iff for all \( f \in \mathbb{E} \):

\[
wp_J \left[ \phi \right] \cdot wp_J (f) = wp_J \left[ \phi \right] \cdot wp_J (f).
\]
Definition (Universally i.i.d. Loops)

A loop while \((\varphi) \{ C \}\) is universally i.i.d. iff for all \(f \in \mathbb{E}\):

\[
wp[C](\varphi \cdot wp[C](f)) =
\]

Example

while \((x > r)\) \{ \(x \approx \text{Unif}[1 \ldots N]\) \} is universally i.i.d.
Definition (Universally i.i.d. Loops)

A loop while $(\varphi) \{ C \}$ is universally i.i.d. iff for all $f \in E$:

$$ wp [C] ( [\varphi] \cdot wp [C] (f) ) = wp [C] ([\varphi]) . $$
Independent and Identically Distributed Loops

Definition (Universally i.i.d. Loops)

A loop \( \text{while} (\varphi) \{ C \} \) is \textit{universally i.i.d.} iff for all \( f \in \mathbb{E} \):

\[
\wp [C] ( [\varphi] \cdot \wp [C] (f) ) = \wp [C] ([\varphi]) \cdot \wp [C] (f) .
\]
Independent and Identically Distributed Loops

Definition (Universally i.i.d. Loops)

A loop \( \text{while} (\varphi) \{ C \} \) is universally i.i.d. iff for all \( f \in \mathbb{E} \):

\[
wp[C](\varphi \cdot wp[C](f)) = wp[C](\varphi) \cdot wp[C](f).
\]

Example

\( \text{while} (x > r) \{ x \approx \text{Unif}[1 \ldots N] \} \) is universally i.i.d.
Theorem

Let $C$ be a program, $\phi$ be a guard, and $f \in E$. If

1. $\text{while}(\phi)\{C\}$ is universally i.i.d.,
2. $wp_{JCK}(1) = 1$, and
3. $\text{ert}_{JC}(0) = 2 \cdot \text{ert}_{JC}(0)$,

then the expected runtime $\text{ert}_{\text{while}(\phi)\{C\}}(f)$ is given by:

$$\text{ert}_{\text{first guard evaluation}} + [\phi] \cdot \text{runtime of one iteration}$$

$$\# \text{iterations} + [\neg \phi] \cdot f,$$

where we define $0 := 0$ and $a := \infty$ for $a \neq 0$. 

A Proof Rule for the Expected Runtime of i.i.d. Loops
A Proof Rule for the Expected Runtime of i.i.d. Loops

**Theorem**

Let $C$ be a program,
A Proof Rule for the Expected Runtime of i.i.d. Loops

Theorem

Let $C$ be a program, $\varphi$ be a guard.

\[\text{expected runtime of } C \text{ is given by:}\]

\[\text{first guard evaluation} + \left[\varphi\right] \cdot \text{runtime of one iteration}\]

\[\text{# iterations} + \left[\neg \varphi\right] \cdot f,
\]

where we define $0 := 0$ and $a := \infty$ for $a \neq 0$. 
A Proof Rule for the Expected Runtime of i.i.d. Loops

**Theorem**

Let $C$ be a program, $\varphi$ be a guard, and $f \in E$. If

1. while $(\varphi)$ \{ $C$ \} is universally i.i.d.,
2. $wp_J C_K (1) = 1$,
3. $ert_J C_K (0) = 2 \cdot ert_J C_K (0)$,

then the expected runtime $ert_J$ while $(\varphi)$ \{ $C$ \} $K$ $(f)$ is given by:

$$1 \quad \text{first guard evaluation} + \left[ \varphi \right] \cdot \approx \text{runtime of one iteration} \quad \text{# iterations} + \left[ \neg \varphi \right] \cdot f,$$

where we define $0 := 0$ and $a := \infty$ for $a \neq 0$. 


A Proof Rule for the Expected Runtime of i.i.d. Loops

**Theorem**

Let $C$ be a program, $\varphi$ be a guard, and $f \in \mathbb{F}$. If $\text{while}(\varphi)\{C\}$ is universally i.i.d.,

1. 

$$
\text{ert}_J \text{while}(\varphi)\{C\}(f) = \text{ert}_J C(1) + [\varphi] \cdot (1 + \text{ert}_J C(\neg \varphi) \cdot f) \cdot (1 - \text{wp}_J C(\varphi)) + [\neg \varphi] \cdot f,
$$

where we define $0_0 = 0$ and $a_0 = \infty$ for $a \neq 0$. 

A Proof Rule for the Expected Runtime of i.i.d. Loops

Theorem

Let $C$ be a program, $\varphi$ be a guard, and $f \in F$. If

1. $\text{while} (\varphi) \{C\}$ is universally i.i.d.,
2. $\text{wp} \left[ C \right] (1) = 1$, and

then the expected runtime $\text{ert} \left[ \text{while} (\varphi) \{C\} \right] (f)$ is given by:

\[
\text{ert} \left[ \text{while} (\varphi) \{C\} \right] (f) = \text{first guard evaluation} + \left[ \varphi \right] \cdot \text{runtime of one iteration} \cdot (1 + \text{ert} \left[ C \right] (\neg \varphi \cdot f))^{1 - \text{wp} \left[ C \right] (\varphi)}.
\]

Where we define $0^0 := 0$ and $a^0 := \infty$ for $a \neq 0$. 
Theorem

Let $C$ be a program, $\varphi$ be a guard, and $f \in \mathbb{E}$. If

1. while $(\varphi) \{C\}$ is universally i.i.d.,
2. $\text{wp} [C] (1) = 1$, and
3. $\text{ert} [C; C] (0) = 2 \cdot \text{ert} [C] (0)$.

then
A Proof Rule for the Expected Runtime of i.i.d. Loops

**Theorem**

Let $C$ be a program, $\varphi$ be a guard, and $f \in \mathbb{E}$. If

1. while $(\varphi) \{ C \}$ is universally i.i.d.,
2. $wp [C'] (1) = 1$, and
3. $ert [C' ; C'] (0) = 2 \cdot ert [C'] (0)$.

then the expected runtime $ert [while (\varphi) \{ C \}] (f)$ is given by:
A Proof Rule for the Expected Runtime of i.i.d. Loops

**Theorem**

Let $C$ be a program, $\varphi$ be a guard, and $f \in E$. If

1. the body $\{C\}$ is universally i.i.d.,
2. $wp \lceil C \rceil (1) = 1$, and
3. $\text{ert} \lceil C ; C \rceil (0) = 2 \cdot \text{ert} \lceil C \rceil (0)$.

then the expected runtime $\text{ert} \lceil \text{while}(\varphi)\{C\} \rceil (f)$ is given by:

\[
1 + \underbrace{\text{first guard evaluation}}_{\text{first guard evaluation}}
\]
A Proof Rule for the Expected Runtime of i.i.d. Loops

**Theorem**

Let \( C \) be a program, \( \varphi \) be a guard, and \( f \in \mathbb{E} \). If

1. \( \text{while} (\varphi) \{ C \} \) is universally i.i.d.,
2. \( \text{wp} \left[ C \right] (1) = 1 \), and
3. \( \text{ert} \left[ C ; C \right] (0) = 2 \cdot \text{ert} \left[ C \right] (0) \).

then the expected runtime \( \text{ert} \left[ \text{while} (\varphi) \{ C \} \right] (f) \) is given by:

\[
\underbrace{1}_{\text{first guard evaluation}} + [\varphi] \cdot \underbrace{\text{runtime of one iteration}}_{\text{first guard evaluation}} + [\neg \varphi] \cdot f ,
\]
Theorem

Let $C$ be a program, $\varphi$ be a guard, and $f \in E$. If

1. while $(\varphi) \{ C \}$ is universally i.i.d.,
2. $wp \llbracket C \rrbracket (1) = 1$, and
3. $ert \llbracket C; C \rrbracket (0) = 2 \cdot ert \llbracket C \rrbracket (0)$.

then the expected runtime $ert \llbracket \text{while} (\varphi) \{ C \} \rrbracket (f)$ is given by:

$$
\begin{align*}
\text{first guard evaluation} \\
1 + [\varphi] \cdot \frac{\approx \text{runtime of one iteration}}{(1 + ert \llbracket C \rrbracket ([\neg \varphi] \cdot f))} + [\neg \varphi] \cdot f,
\end{align*}
$$

where we define $0^0 := 0$ and $a^0 := \infty$ for $a \neq 0$. 

A Proof Rule for the Expected Runtime of i.i.d. Loops
### Theorem

Let $C$ be a program, $\varphi$ be a guard, and $f \in \mathbb{E}$. If

1. $\text{while}(\varphi)\{C\}$ is universally i.i.d.,
2. $\text{wp}[C](1) = 1$, and
3. $\text{ert}[C; C](0) = 2 \cdot \text{ert}[C](0)$.

then the expected runtime $\text{ert[while}(\varphi)\{C\}] (f)$ is given by:

\[
\begin{align*}
\text{first guard evaluation} & \quad + \quad [\varphi] \cdot \frac{(1 + \text{ert}[C]([\neg \varphi] \cdot f))}{1 - \text{wp}[C]([\varphi])} \\
\text{runtime of one iteration} & \quad + \quad [\neg \varphi] \cdot f,
\end{align*}
\]

where we define $0 := 0$ and $a := \infty$ for $a \neq 0$. 

A Proof Rule for the Expected Runtime of i.i.d. Loops

**Theorem**

Let $C$ be a program, $\varphi$ be a guard, and $f \in \mathbb{E}$. If

1. while $(\varphi) \{ C \}$ is universally i.i.d.,
2. $wp[C](1) = 1$, and
3. $ert[C; C](0) = 2 \cdot ert[C](0)$.

then the expected runtime $ert[\text{while}(\varphi) \{ C \}](f)$ is given by:

\[
\begin{align*}
&\mathbf{1} \quad \text{first guard evaluation} \\
\approx & \text{runtime of one iteration} \\
&+ [\varphi] \cdot \frac{(1 + ert[C][[\neg \varphi] \cdot f])}{1 - wp[C][[\varphi]]} \\
&+ [\neg \varphi] \cdot f ,
\end{align*}
\]

where we define $\frac{0}{0} := 0$ and $\frac{a}{0} := \infty$ for $a \neq 0$. 
Semantics of i.i.d. Loops

**Theorem**

*If* \( \text{while}(\varphi)\{C\} \) *is universally i.i.d.*, then for all \( f \in \mathbb{E}, \sigma \in \Sigma \):

\[
\begin{align*}
\text{wp}\left[\text{while}(\varphi)\{C\}\right](f)(\sigma) &= [\varphi](\sigma) \cdot \frac{\text{wp}\left[C\right]([\neg \varphi] \cdot f)(\sigma)}{1 - \text{wp}\left[C\right](\varphi)(\sigma)} + [\neg \varphi](\sigma) \cdot f(\sigma).
\end{align*}
\]

\[
\begin{cases}
= 0 & \text{if } \text{wp}[C](\varphi)(\sigma) = 1
\end{cases}
\]
A Syntactic Notion of i.i.d. Loops

Definition

A loop $\text{while}(\phi)\{C\}$ is $f$–i.i.d. if $1 \\wp J_C K(\[\phi\])$ is unaffected by $C$, and $2 \\wp J_C K(\[\neg \phi \cdot f\])$ is unaffected by $C$.

Theorem

The proof rules on expected values and expected runtimes for universally i.i.d. loops also hold for $f$–i.i.d. loops.

Universal i.i.d. loops and $f$–i.i.d. loops are incomparable.
A Syntactic Notion of i.i.d. Loops

\[ f \in \mathbb{E} \text{ is unaffected by program } C \text{ iff } \text{Vars}(f) \cap \text{Mod}(C) = \emptyset. \]
A Syntactic Notion of i.i.d. Loops

\( f \in \mathbb{E} \) is unaffected by program \( C \) iff \( \text{Vars}(f) \cap \text{Mod}(C) = \emptyset \).

**Definition**

A loop \( \text{while}(\varphi) \{ C \} \) is \( f \)-i.i.d. if

1. \( \exp(\text{while}(\varphi) \{ C \}, f) \) is unaffected by \( C \), and
2. \( \exp(\text{while}(\neg \varphi) \{ f \} \cdot \varphi) \) is unaffected by \( C \).

Theorem

The proof rules on expected values and expected runtimes for universally i.i.d. loops also hold for \( f \)-i.i.d. loops.

Universal i.i.d. loops and \( f \)-i.i.d. loops are incomparable.
A Syntactic Notion of i.i.d. Loops

\[ f \in \mathbb{F} \text{ is unaffected by program } C \text{ iff } \text{Vars}(f) \cap \text{Mod}(C) = \emptyset. \]

**Definition**

A loop \( \text{while}(\varphi)\{C\} \) is \( f \)-i.i.d. if

1. \( \text{wp}[C][[\varphi]] \) is unaffected by \( C \), and
A Syntactic Notion of i.i.d. Loops

\[ f \in \mathbb{E} \text{ is unaffected by program } C \text{ iff } \text{Vars}(f) \cap \text{Mod}(C) = \emptyset. \]

**Definition**

A loop while (\( \varphi \)) \{\( C \)\} is \( f \)-i.i.d. if

1. \( \text{wp}[C]\{\varphi\} \) is unaffected by \( C \), and
2. \( \text{wp}[C]\{\neg\varphi \cdot f\} \) is unaffected by \( C \).
A Syntactic Notion of i.i.d. Loops

\[ f \in \mathbb{E} \text{ is unaffected by program } C \text{ iff } \text{Vars}(f) \cap \text{Mod}(C) = \emptyset. \]

**Definition**

A loop while \((\varphi) \{C\}\) is \(f\)-i.i.d. if

1. \(\text{wp} \left[ C \right] (\varphi)\) is unaffected by \(C\), and
2. \(\text{wp} \left[ C \right] (\neg \varphi \cdot f)\) is unaffected by \(C\).

**Theorem**

*The proof rules on expected values and expected runtimes for universally i.i.d. loops also hold for \(f\)-i.i.d. loops.*
A Syntactic Notion of i.i.d. Loops

\( f \in \mathbb{E} \) is unaffected by program \( C \) iff \( \text{Vars}(f) \cap \text{Mod}(C) = \emptyset \).

**Definition**

A loop \( \text{while}(\varphi) \{C\} \) is \( f \)-i.i.d. if

1. \( \text{wp}[[C]]([\varphi]) \) is unaffected by \( C \), and
2. \( \text{wp}[[C]]([\neg \varphi] \cdot f) \) is unaffected by \( C \).

**Theorem**

The proof rules on expected values and expected runtimes for universally i.i.d. loops also hold for \( f \)-i.i.d. loops.

Universal i.i.d. loops and \( f \)-i.i.d. loops are incomparable.
**Application: Bayesian Networks**

#### Difficulty

<table>
<thead>
<tr>
<th>$D = 0$</th>
<th>$D = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6</td>
<td>0.4</td>
</tr>
</tbody>
</table>

#### Preparation

<table>
<thead>
<tr>
<th>$P = 0$</th>
<th>$P = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.7</td>
<td>0.3</td>
</tr>
</tbody>
</table>

#### Grade

<table>
<thead>
<tr>
<th>$G = 0$</th>
<th>$G = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D = 0, P = 0$</td>
<td>0.95</td>
</tr>
<tr>
<td>$D = 1, P = 1$</td>
<td>0.05</td>
</tr>
<tr>
<td>$D = 0, P = 1$</td>
<td>0.5</td>
</tr>
<tr>
<td>$D = 1, P = 0$</td>
<td>0.6</td>
</tr>
</tbody>
</table>

#### Mood

<table>
<thead>
<tr>
<th>$M = 0$</th>
<th>$M = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G = 0$</td>
<td>0.9</td>
</tr>
<tr>
<td>$G = 1$</td>
<td>0.3</td>
</tr>
</tbody>
</table>

**Inference problem:** Derive probabilities from observed evidence.

"What is the probability that a poorly prepared student ends up with a good mood?"

Exact and approximative inference are NP-hard.

Use sampling methods instead, e.g. rejection sampling. This corresponds to running a probabilistic program!
Inference problem: Derive probabilities from observed evidence
Application: Bayesian Networks

- Inference problem: Derive probabilities from observed evidence
- "What is the probability that a poorly prepared student ends up with a good mood?"
Inference problem: Derive probabilities from observed evidence

"What is the probability that a poorly prepared student ends up with a good mood?"

Exact and approximative inference are NP-hard.
Application: Bayesian Networks

- **Inference problem**: Derive probabilities from observed evidence.

- "What is the probability that a poorly prepared student ends up with a good mood?"

- Exact and approximative inference are NP–hard.

- Use sampling methods instead, e.g. rejection sampling.
Application: Bayesian Networks

<table>
<thead>
<tr>
<th>Difficulty</th>
<th>Preparation</th>
<th>Grade</th>
<th>Mood</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D = 0$</td>
<td>$P = 0$</td>
<td>$G = 0$</td>
<td>$M = 0$</td>
</tr>
<tr>
<td>0.6</td>
<td>0.7</td>
<td>0.95</td>
<td>0.9</td>
</tr>
<tr>
<td>$D = 0$</td>
<td>$P = 1$</td>
<td>$G = 1$</td>
<td>$M = 0$</td>
</tr>
<tr>
<td>0.4</td>
<td>0.3</td>
<td>0.05</td>
<td>0.1</td>
</tr>
<tr>
<td>$D = 1$</td>
<td>$P = 0$</td>
<td>$G = 0$</td>
<td>$M = 1$</td>
</tr>
<tr>
<td>0.7</td>
<td>0.3</td>
<td>0.5</td>
<td>0.7</td>
</tr>
<tr>
<td>$D = 1$</td>
<td>$P = 1$</td>
<td>$G = 1$</td>
<td>$M = 1$</td>
</tr>
<tr>
<td>0.4</td>
<td>0.3</td>
<td>0.5</td>
<td>0.7</td>
</tr>
</tbody>
</table>

- **Inference problem:** Derive probabilities from observed evidence
- "What is the probability that a poorly prepared student ends up with a good mood?"
- Exact and approximative inference are NP-hard.
- Use sampling methods instead, e.g. rejection sampling.
- This corresponds to running a probabilistic program!
Bayesian Network → Probabilistic Program

\[
\text{Pr}(G=0, D=1, M=0, P=1) \\
\text{Conditioning? } \text{Pr}(D, M=1 | G, P=0) \\
D=0 \quad 0.95 \quad D=1 \quad 0.05 \\
P=0 \quad 0.98 \quad P=1 \quad 0.02 \\
G=0 \quad 0.99 \quad G=1 \quad 0.01 \\
D=0, P=0 \quad D=1, P=0 \quad D=1, P=1 \\
\text{repeat} \\
x_D: \approx 0.95 \cdot \langle 0 \rangle + 0.05 \cdot \langle 1 \rangle; \\
x_P: \approx 0.98 \cdot \langle 0 \rangle + 0.02 \cdot \langle 1 \rangle; \\
\text{if } (x_D=0 \land x_P=0) \\
x_G: \approx 0.99 \cdot \langle 0 \rangle + 0.01 \cdot \langle 1 \rangle; \\
\text{else} \\
x_G: \approx 0.01 \cdot \langle 0 \rangle + 0.99 \cdot \langle 1 \rangle; \\
\text{if } (x_G=0) \\
x_M: \approx 0.9 \cdot \langle 0 \rangle + 0.1 \cdot \langle 1 \rangle; \\
\text{else} \\
x_M: \approx 0.3 \cdot \langle 0 \rangle + 0.7 \cdot \langle 1 \rangle; \\
\text{until } (x_G=0 \land x_P=0)
\]
Bayesian Network → Probabilistic Program

\[
x_D \approx 0.95 \cdot \langle 0 \rangle + 0.05 \cdot \langle 1 \rangle;
\]
Bayesian Network → Probabilistic Program

\[ \text{Conditioning? } \Pr(D,M = 1 | G,P = 0) \]

\[ D = 0 \]
\[ D = 1 \] 0.95 0.05
\[ P = 0 \]
\[ P = 1 \] 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

\[
x_D : \approx 0.95 \cdot \langle 0 \rangle + 0.05 \cdot \langle 1 \rangle;
\]
\[
x_P : \approx 0.98 \cdot \langle 0 \rangle + 0.02 \cdot \langle 1 \rangle;
\]
\[
\text{if}(x_D = 0 \land x_P = 0) \{
    x_G : \approx 0.99 \cdot \langle 0 \rangle + 0.01 \cdot \langle 1 \rangle;
\}
\]

<table>
<thead>
<tr>
<th>(D, P)</th>
<th>(G = 0)</th>
<th>(G = 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(D = 0, P = 0)</td>
<td>0.99</td>
<td>0.01</td>
</tr>
<tr>
<td>(D = 0, P = 1)</td>
<td>0.29</td>
<td>0.71</td>
</tr>
<tr>
<td>(D = 1, P = 0)</td>
<td>0.06</td>
<td>0.94</td>
</tr>
<tr>
<td>(D = 1, P = 1)</td>
<td>0.01</td>
<td>0.99</td>
</tr>
</tbody>
</table>
Bayesian Network → Probabilistic Program

<table>
<thead>
<tr>
<th></th>
<th>$G = 0$</th>
<th>$G = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D = 0$, $P = 0$</td>
<td>0.99</td>
<td>0.01</td>
</tr>
<tr>
<td>$D = 0$, $P = 1$</td>
<td>0.29</td>
<td>0.71</td>
</tr>
<tr>
<td>$D = 1$, $P = 0$</td>
<td>0.06</td>
<td>0.94</td>
</tr>
<tr>
<td>$D = 1$, $P = 1$</td>
<td>0.01</td>
<td>0.99</td>
</tr>
</tbody>
</table>

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

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

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

    
    } else {
    
    $x_G \approx 0.01 \cdot \langle 0 \rangle + 0.99 \cdot \langle 1 \rangle$;
Bayesian Network → Probabilistic Program

\[
\text{if}(x_D = 0 \land x_P = 0) \{
    x_G : \approx 0.99 \cdot \langle 0 \rangle + 0.01 \cdot \langle 1 \rangle;
\}
\text{else} \{
    x_G : \approx 0.01 \cdot \langle 0 \rangle + 0.99 \cdot \langle 1 \rangle;
\}
\]

\[
x_D : \approx 0.95 \cdot \langle 0 \rangle + 0.05 \cdot \langle 1 \rangle;
\]
\[
x_P : \approx 0.98 \cdot \langle 0 \rangle + 0.02 \cdot \langle 1 \rangle;
\]

\[
\begin{array}{c|cc}
G & M = 0 & M = 1 \\
\hline
G = 0 & 0.9 & 0.1 \\
G = 1 & 0.3 & 0.7 \\
\end{array}
\]

\[
\text{if}(x_G = 0) \{
    x_M : \approx 0.9 \cdot \langle 0 \rangle + 0.1 \cdot \langle 1 \rangle;
\}
\]

\[
\text{if}(x_G = 1) \{
    x_M : \approx 0.1 \cdot \langle 0 \rangle + 0.9 \cdot \langle 1 \rangle;
\}
\]
$$\Pr(G = 0, D = 1, M = 0, P = 1)$$

Conditioning: $$\Pr(D, M = 1 | G, P = 0)$$

<table>
<thead>
<tr>
<th></th>
<th>$$M = 0$$</th>
<th>$$M = 1$$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$$G = 0$$</td>
<td>0.9</td>
<td>0.1</td>
</tr>
<tr>
<td>$$G = 1$$</td>
<td>0.3</td>
<td>0.7</td>
</tr>
</tbody>
</table>

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

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

if $$(x_D = 0 \land x_P = 0)$$

$$x_G \approx 0.99 \cdot \langle 0 \rangle + 0.01 \cdot \langle 1 \rangle;$$

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

$$x_M \approx 0.3 \cdot \langle 0 \rangle + 0.7 \cdot \langle 1 \rangle;$$

else

$$x_M \approx 0.3 \cdot \langle 0 \rangle + 0.7 \cdot \langle 1 \rangle;$$
Bayesian Network $\rightarrow$ Probabilistic Program

$$\Pr(G = 0, D = 1, M = 0, P = 1) \sqrt{\text{Conditioning?}}$$

$$\begin{align*}
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 \land x_P = 0) \{ \\
&\quad x_G \approx 0.99 \cdot \langle 0 \rangle + 0.01 \cdot \langle 1 \rangle; \\
&\quad \vdots \\
&\} \quad \text{else} \quad \{ \\
&\quad x_G \approx 0.01 \cdot \langle 0 \rangle + 0.99 \cdot \langle 1 \rangle; \\
&\} \quad \text{if}(x_G = 0) \{ \\
&\quad x_M \approx 0.9 \cdot \langle 0 \rangle + 0.1 \cdot \langle 1 \rangle; \\
&\} \quad \text{else} \quad \{ \\
&\quad x_M \approx 0.3 \cdot \langle 0 \rangle + 0.7 \cdot \langle 1 \rangle; \\
\end{align*}$$
Bayesian Network → Probabilistic Program

Pr($G = 0, D = 1, M = 0, P = 1$) √

Conditioning? Pr($D, M = 1 | G, P = 0$)

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

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

if ($x_D = 0 \land x_P = 0$) {
    $x_G : \approx 0.99 \cdot \langle 0 \rangle + 0.01 \cdot \langle 1 \rangle$;
    :
    :
} 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 → Probabilistic Program

\[ \Pr(G = 0, D = 1, M = 0, P = 1) \]

Conditioning? \[ \Pr(D, M = 1 \mid G, P = 0) \]

- \( x_D \approx 0.95 \cdot \langle 0 \rangle + 0.05 \cdot \langle 1 \rangle; \)
- \( x_P \approx 0.98 \cdot \langle 0 \rangle + 0.02 \cdot \langle 1 \rangle; \)
- if \((x_D = 0 \land x_P = 0)\) { \( x_G \approx 0.99 \cdot \langle 0 \rangle + 0.01 \cdot \langle 1 \rangle; \) 
  : 
} 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 { 
    \( x_M \approx 0.3 \cdot \langle 0 \rangle + 0.7 \cdot \langle 1 \rangle; \) 
  } 
} until \((x_G = 0 \land x_P = 0)\)
The Bayesian Network Language (BNL)

\[ C \to Seq \mid \text{repeat}\{Seq\}\text{until}(\psi) \mid C; C \]
The Bayesian Network Language (BNL)

\[
\begin{align*}
C & \rightarrow \begin{array}{c} \text{Seq} \mid \text{repeat\{Seq\} until(ψ)} \mid \ C; \ C \\
\end{array} \\
\text{Seq} & \rightarrow \begin{array}{c} \text{Seq; Seq} \mid B_{x_1} \mid B_{x_2} \mid \ldots \end{array}
\end{align*}
\]
The Bayesian Network Language (BNL)

\[ C \rightarrow \text{Seq} \mid \text{repeat}\{\text{Seq}\}\text{until}(\psi) \mid C; C \]

\[ \text{Seq} \rightarrow \text{Seq}; \text{Seq} \mid B_{x_1} \mid B_{x_2} \mid \ldots \]

\[ B_{x_i} \rightarrow x_i : \approx \mu \mid \text{if}(\varphi)\{x_i : \approx \mu\}\text{else}\{B_{x_i}\} \]
The Bayesian Network Language (BNL)

\[
C \rightarrow \text{Seq} \mid \text{repeat} \{\text{Seq}\} \text{until} (\psi) \mid C; C
\]

\[
\text{Seq} \rightarrow \text{Seq}; \text{Seq} \mid B_{x_1} \mid B_{x_2} \mid \ldots
\]

\[
B_{x_i} \rightarrow x_i : \approx \mu \mid \text{if} (\varphi) \{x_i : \approx \mu\} \text{else} \{B_{x_i}\}
\]

where \( \mu = \sum_{j=1}^{n} p_j \cdot \langle a_j \rangle \) with \( a_j \in \mathbb{Q} \) for \( 1 \leq j \leq n \).
The Bayesian Network Language (BNL)

\[ C \longrightarrow \text{Seq} \mid \text{repeat}\{\text{Seq}\}\text{until}(\psi) \mid C; C \]

\[ \text{Seq} \longrightarrow \text{Seq}; \text{Seq} \mid B_{x_1} \mid B_{x_2} \mid \ldots \]

\[ B_{x_i} \longrightarrow x_i : \approx \mu \mid \text{if}(\varphi)\{x_i : \approx \mu\}\text{else}\{B_{x_i}\} \]

where \( \mu = \sum_{j=1}^{n} p_j \cdot \langle a_j \rangle \) with \( a_j \in \mathbb{Q} \) for \( 1 \leq j \leq n \).

\textbf{Theorem}

Every Bayesian network \( \mathcal{B} \) with observed nodes \( O \subseteq V \) can be translated into an equivalent BNL program:
The Bayesian Network Language (BNL)

\[
\begin{align*}
C & \rightarrow \text{Seq} \mid \text{repeat}\{\text{Seq}\}\text{until}(\psi) \mid C; C \\
\text{Seq} & \rightarrow \text{Seq}; \text{Seq} \mid B_{x_1} \mid B_{x_2} \mid \ldots \\
B_{x_i} & \rightarrow x_i : \approx \mu \mid \text{if}(\varphi)\{x_i : \approx \mu\}\text{else}\{B_{x_i}\}
\end{align*}
\]

where \( \mu = \sum_{j=1}^{n} p_j \cdot \langle a_j \rangle \) with \( a_j \in Q \) for \( 1 \leq j \leq n \).

**Theorem**

Every Bayesian network \( \mathcal{B} \) with observed nodes \( O \subseteq V \) can be translated into an equivalent BNL program:

\[
\text{wp} [\text{BNL}(\mathcal{B})] \left( \begin{array}{c}

\end{array} \right)
\]
The Bayesian Network Language (BNL)

\[
\begin{align*}
C & \longrightarrow \text{Seq} \mid \text{repeat \{Seq\} until}(\psi) \mid C; C \\
\text{Seq} & \longrightarrow \text{Seq; Seq} \mid B_{x_1} \mid B_{x_2} \mid \ldots \\
B_{x_i} & \longrightarrow x_i \approx \mu \mid \text{if}(\varphi) \{x_i \approx \mu\} \text{ else } \{B_{x_i}\}
\end{align*}
\]

where \( \mu = \sum_{j=1}^{n} p_j \cdot \langle a_j \rangle \) with \( a_j \in \mathbb{Q} \) for \( 1 \leq j \leq n \).

**Theorem**

*Every Bayesian network \( \mathcal{B} \) with observed nodes \( O \subseteq V \) can be translated into an equivalent BNL program:*

\[
\text{wp}[\text{BNL}(\mathcal{B})]\left(\prod_{v \in V \setminus O} [x_v = v] \right)
\]
The Bayesian Network Language (BNL)

\[
\begin{align*}
C & \rightarrow \text{Seq} \mid \text{repeat}\{\text{Seq}\}\text{until}(\psi) \mid C; C \\
\text{Seq} & \rightarrow \text{Seq}; \text{Seq} \mid B_{x_1} \mid B_{x_2} \mid \ldots \\
B_{x_i} & \rightarrow x_i \approx \mu \mid \text{if}(\varphi)\{x_i \approx \mu\}\text{else}\{B_{x_i}\}
\end{align*}
\]

where \( \mu = \sum_{j=1}^{n} p_j \cdot \langle a_j \rangle \) with \( a_j \in \mathbb{Q} \) for \( 1 \leq j \leq n \).

**Theorem**

Every Bayesian network \( \mathcal{B} \) with observed nodes \( O \subseteq V \) can be translated into an equivalent BNL program:

\[
\text{wp}[\text{BNL}(\mathcal{B})] \left( \prod_{v \in V \setminus O} [x_v = v] \right),
\]

where \( v \) is a node, \( x_v \) a variable, and \( v \in \mathbb{Q} \) a fixed value.
The Bayesian Network Language (BNL)

\[
C \rightarrow \text{Seq} \mid \text{repeat \{Seq\} until}(\psi) \mid C; C
\]

\[
\text{Seq} \rightarrow \text{Seq; Seq} \mid B_{x_1} \mid B_{x_2} \mid \ldots
\]

\[
B_{x_i} \rightarrow x_i \triangleq \mu \mid \text{if}(\varphi)\{x_i \triangleq \mu\} \text{ else } \{B_{x_i}\}
\]

where \( \mu = \sum_{j=1}^{n} p_j \cdot \langle a_j \rangle \) with \( a_j \in \mathbb{Q} \) for \( 1 \leq j \leq n \).

**Theorem**

Every Bayesian network \( \mathcal{B} \) with observed nodes \( O \subseteq V \) can be translated into an equivalent BNL program:

\[
\text{wp} \left[ BNL(\mathcal{B}) \right] \left( \prod_{v \in V \setminus O} [x_v = v] \right) = \frac{\text{joint probability of } \mathcal{B}} {\text{prob. of observations}} = \frac{\Pr (\bigwedge_{v \in V} v = v)} {\Pr (\bigwedge_{o \in O} o = o)},
\]

where \( v \) is a node, \( x_v \) a variable, and \( v \in \mathbb{Q} \) a fixed value.
How long, O Bayesian network, will I sample thee?

<table>
<thead>
<tr>
<th></th>
<th>$S = 0$</th>
<th>$S = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R = 0$</td>
<td>$a$</td>
<td>$1 - a$</td>
</tr>
<tr>
<td>$R = 1$</td>
<td>$0.2$</td>
<td>$0.8$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$R = 0$</th>
<th>$R = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$1 - a$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$G = 0$</th>
<th>$G = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S = 0, R = 0$</td>
<td>$0.01$</td>
</tr>
<tr>
<td>$S = 0, R = 1$</td>
<td>$0.25$</td>
</tr>
<tr>
<td>$S = 1, R = 0$</td>
<td>$0.9$</td>
</tr>
<tr>
<td>$S = 1, R = 1$</td>
<td>$0.2$</td>
</tr>
</tbody>
</table>

Observe $G = 0$
How long, O Bayesian network, will I sample thee?

<table>
<thead>
<tr>
<th></th>
<th>$S = 0$</th>
<th>$S = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R = 0$</td>
<td>$a$</td>
<td>$1 - a$</td>
</tr>
<tr>
<td>$R = 1$</td>
<td>0.2</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$R = 0$</th>
<th>$R = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$1 - a$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$G = 0$</th>
<th>$G = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S = 0, R = 0$</td>
<td>0.01</td>
</tr>
<tr>
<td>$S = 0, R = 1$</td>
<td>0.25</td>
</tr>
<tr>
<td>$S = 1, R = 0$</td>
<td>0.9</td>
</tr>
<tr>
<td>$S = 1, R = 1$</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Observe $G = 0$

```
time
   300
   200
   100

   0.2  0.4  0.6  0.8  1

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

<table>
<thead>
<tr>
<th></th>
<th>$S = 0$</th>
<th>$S = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R = 0$</td>
<td>$a$</td>
<td>$1 - a$</td>
</tr>
<tr>
<td>$R = 1$</td>
<td>0.2</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$R = 0$</th>
<th>$R = 1$</th>
</tr>
</thead>
</table>
| $a$ | $1 - a$ | $a$ | $1 - a$

<table>
<thead>
<tr>
<th></th>
<th>$G = 0$</th>
<th>$G = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S = 0, R = 0$</td>
<td>0.01</td>
<td>0.99</td>
</tr>
<tr>
<td>$S = 0, R = 1$</td>
<td>0.25</td>
<td>0.75</td>
</tr>
<tr>
<td>$S = 1, R = 0$</td>
<td>0.9</td>
<td>0.1</td>
</tr>
<tr>
<td>$S = 1, R = 1$</td>
<td>0.2</td>
<td>0.8</td>
</tr>
</tbody>
</table>

Observe $G = 0$

\[
er\text{ert} \left[ BNL(B) \right] (0) = \frac{200a^2 - 40a - 460}{89a^2 - 69a - 21}
\]
How long, O Bayesian network, will I sample thee? (II)

Recall: Every Bayesian network is equivalent to a BNL program.
How long, O Bayesian network, will I sample thee? (II)

Recall: Every Bayesian network is equivalent to a BNL program.

**Theorem**

*Every loop in BNL is $f$–i.i.d. for every $f \in \mathbb{E}$.***
Recall: Every Bayesian network is equivalent to a BNL program.

**Theorem**

*Every loop in BNL is $f$–i.i.d. for every $f \in \mathbb{E}$.***

**Theorem**

*For every loop $\{C\}$ until $(\psi)$ in BNL and every $f \in \mathbb{E}$:

\[
\text{ert} \left[ \text{repeat} \{C\} \text{ until } (\psi) \right] (f) = \frac{1 + \text{ert} \left[ C \right] ([\psi] \cdot f)}{\text{wp} \left[ C \right] ([\psi])}.
\]
How long, O Bayesian network, will I sample thee? (II)

Recall: Every Bayesian network is equivalent to a BNL program.

**Theorem**

*Every loop in BNL is $f$–i.i.d. for every $f \in \mathbb{F}$.***

**Theorem**

*For every loop* \[ \text{repeat} \{C\} \text{ until } \psi \] *in BNL and every $f \in \mathbb{F}$:*

\[
\text{ert} \left[ \text{repeat} \{C\} \text{ until } \psi \right] (f) = \frac{1 + \text{ert} \left[ C \right] ([\psi] \cdot f)}{\text{wp} \left[ C \right] ([\psi])}.
\]

**Corollary**

*For every BNL program $C$ and (computable) $f \in \mathbb{F}$, a closed form of $\text{ert} \left[ C \right] (f)$ and $\text{wp} \left[ C \right] (f)$ can be computed.*
Summary
Summary

- Simple proof rules for exact runtimes and expected values of i.i.d. loops
Summary

- Simple proof rules for exact runtimes and expected values of i.i.d. loops
- Sound translation of Bayesian networks to BNL programs

Future work
- Consider extended graphical models
- Automate proof search
- Can we get more proof rules (for lower bounds)?
Summary

- Simple proof rules for exact runtimes and expected values of i.i.d. loops
- Sound translation of Bayesian networks to BNL programs
- Proof rules are applicable to all Bayesian networks
Summary

- **Simple** proof rules for *exact* runtimes and expected values of i.i.d. loops
- Sound translation of Bayesian networks to BNL programs
- Proof rules are applicable to all Bayesian networks

Future work
Summary

- **Simple** proof rules for exact runtimes and expected values of i.i.d. loops
- Sound translation of Bayesian networks to BNL programs
- Proof rules are applicable to all Bayesian networks

Future work

- Consider extended graphical models
Summary

- Simple proof rules for exact runtimes and expected values of i.i.d. loops
- Sound translation of Bayesian networks to BNL programs
- Proof rules are applicable to all Bayesian networks

Future work

- Consider extended graphical models
- Automate proof search
Summary

- **Simple** proof rules for exact runtimes and expected values of i.i.d. loops
- Sound translation of Bayesian networks to BNL programs
- Proof rules are applicable to all Bayesian networks

Future work

- Consider extended graphical models
- Automate proof search
- Can we get more proof rules?
Summary

- Simple proof rules for exact runtimes and expected values of i.i.d. loops
- Sound translation of Bayesian networks to BNL programs
- Proof rules are applicable to all Bayesian networks

Future work

- Consider extended graphical models
- Automate proof search
- Can we get more proof rules (for lower bounds)?
Summary

- Simple proof rules for exact runtimes and expected values of i.i.d. loops
- Sound translation of Bayesian networks to BNL programs
- Proof rules are applicable to all Bayesian networks

Future work

- Consider extended graphical models
- Automate proof search
- Can we get more proof rules (for lower bounds)? next talk
Summary

- **Simple** proof rules for **exact** runtimes and expected values of i.i.d. loops
- Sound translation of Bayesian networks to BNL programs
- Proof rules are applicable to all Bayesian networks

Future work

- Consider extended graphical models
- Automate proof search
- Can we get more proof rules *(for lower bounds)*? next talk

Thank you for your attention!