Deductive Verification of Probabilistic Programs

with Caesar

Joost-Pieter Katoen

PROBABILISTIC PROGRAMMING

Stan

PYRO

Scenic

WebPPL

Every programming language has a probabilistic variant

The piranha problem [Tijms, 2004]

One fish is contained within the confines of an opaque fishbowl. The fish is equally likely to be a piranha or a goldfish. A sushi lover throws a piranha into the fish bowl alongside the other fish. Then, immediately, before either fish can devour the other, one of the fish is blindly removed from the fishbowl. The fish that has been removed from the bowl turns out to be a piranha. What is the probability that the fish that was originally in the bowl by itself was a piranha?

Probabilistic programs

Programs with random assignments and conditioning

```
f1 := gf [0.5] f1 := pir;
f2 := pir;
s := f1 [0.5] s := f2;
observe (s = pir)
```

$Pr\{f_1 = gf\} = \frac{1}{2}$ $Pr\{f_1 = pir\} = \frac{1}{2}$

ajaman

Probabilistic programs

Programs with random assignments and conditioning

```
f1 := gf [0.5] f1 := pir;
f2 := pir;
s := f1 [0.5] s := f2;
observe (s = pir)
```

Probabilistic programs

Programs with random assignments and conditioning

```
f1 := gf [0.5] f1 := pir;
f2 := pir;
s := f1 [0.5] s := f2;
observe (s = pir)
```

What is the probability that the original fish in the bowl was a piranha?

Probabilistic programs

Programs with random assignments and conditioning

```
f1 := gf [0.5] f1 := pir;
f2 := pir;
s := f1 [0.5] s := f2;
observe (s = pir)
```

p := unif[1..N]

They encode:

- randomised algorithms
- probabilistic graphical models beyond Bayes' networks
- controllers for autonomous systems
- security mechanisms
- . . . . . .

"Probabilistic programming aims to make probabilistic modeling and machine learning accessible to the programmer."

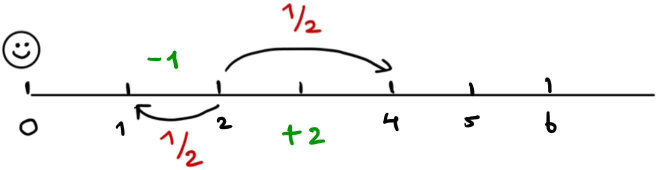
[Gordon, Henzinger, Nori and Rajamani, FOSE 2014]

Probabilistic programs are hard to grasp

Does this program almost surely terminate? That is, is it AST?

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

x+1



Probabilistic programs are hard to grasp

Does this program almost surely terminate? That is, is it AST?

$Pr\{\text{terminate}) = 1$

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

x+1

¬PAST

> If not, what is its probability to diverge?

Positive AST

```
int x := 1;
bool c := true;
while (c) {
    c := false [0.5] c := true;
    x := 2*x
}
```

Finite expected termination time?
aka: is this program positive AST?

Positive AST

```
int x := 1;
bool c := true;
while (c) {
    c := false [0.5] c := true;
    x := 2*x
}
```

Finite expected termination time?
aka: is this program positive AST?

```
while (x > 0) {
    x := x-1
}
```

Finite termination time!
PAST.

Positive AST

```
int x := 1;
bool c := true;
while (c) {
    c := false [0.5] c := true;
    x := 2*x
}
```

;

```
while (x > 0) {
    x := x-1
}
```

Finite expected termination time?
aka: is this program positive AST?

Finite termination time!
PAST.

Expected runtime of these programs in sequence?

Our objective

A powerful, simple proof calculus for probabilistic programs.

At the source code level.

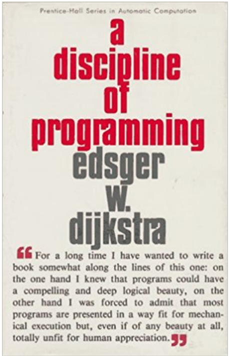
No “descend” into the underlying probabilistic model.

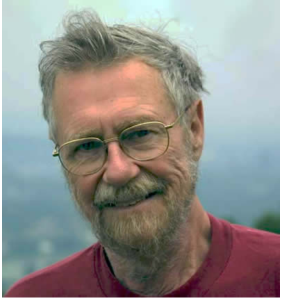
Push automation as much as we can.

This is a true challenge: undecidability!

Typically “more undecidable” than deterministic programs

WEAKEST PRECONDITIONS





Edsger Wybe Dijkstra

Predicate transformers

Let the set of states be:

$$\mathbb{S} \;=\; \{\, s \mid s : \mathit{Vars} \to \mathbb{Q}_{\geq 0} \,\}$$

e.g. $s(x) = 1$

$s(y) = 3$

$s(z) = 0$

Predicate transformers

Let the set of states be:

$$\mathbb{S} \;=\; \{\, s \mid s : \mathit{Vars} \to \mathbb{Q}_{\geq 0} \}$$

Let the set of predicates be:

$$\mathbb{P} \;=\; \left\{ F \mid F : \underbrace{\mathbb{S}}_{\text{states}} \to \underbrace{\{0,1\}}_{\mathbb{B}\text{(ooleans)}} \right\}$$

example: $F = x > 0 \wedge 0 \leq y < 10$

$s(x) = 4, \; s(y) = 3$ then $s \models F$

$s'(x) = 4, \; s'(y) = 10$ then $s' \not\models F$

Predicate transformers

Let the set of states be:

$$\mathbb{S} \;=\; \{\, s \mid s : Vars \to \mathbb{Q} \,\}$$

Let the set of predicates be:

$$\mathbb{P} \;=\; \left\{\, F \mid F : \underbrace{\mathbb{S}}_{\text{states}} \to \{\,0, 1\,\} \,\right\}$$

Predicate F is typically a first-order logic formula. It equals $\{\, s \in \mathbb{S} \mid s \models F \,\}$. Thus $\mathbb{P} = 2^{\mathbb{S}}$.

$(\mathbb{P}, \sqsubseteq)$ is a complete lattice where $F \sqsubseteq G$ if and only if $F \Rightarrow G$

Predicate transformers

Let the set of states be:

$$\mathbb{S} \ = \ \{ \, s \mid s : \mathit{Vars} \rightarrow \mathbb{Q} \, \}$$

Let the set of predicates be:

$$\mathbb{P} \ = \ \left\{ F \mid F : \underbrace{\mathbb{S}}_{\text{states}} \rightarrow \{ 0, 1 \} \right\}$$

Predicate F is typically a first-order logic formula. It equals $\{ \, s \in \mathbb{S} \mid s \models F \, \}$. Thus $\mathbb{P} = 2^{\mathbb{S}}$. Let partial order $\sqsubseteq$ equal $\subseteq$. Ergo:

$(\mathbb{P}, \sqsubseteq)$ is a complete lattice where $F \sqsubseteq G$ if and only if $F \Rightarrow G$

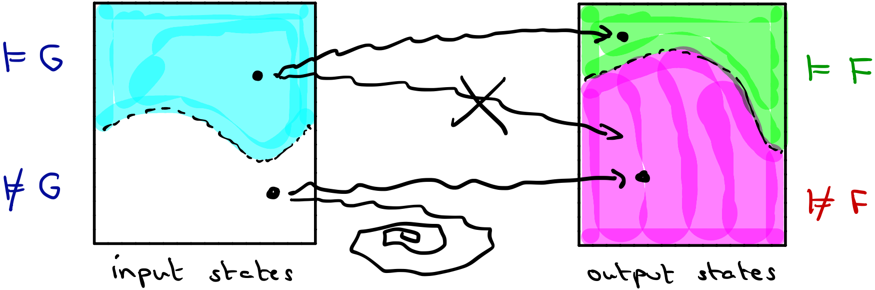
$\boxed{\text{Function } \Phi : \mathbb{P} \rightarrow \mathbb{P} \text{ is called a predicate transformer}}$

Weakest preconditions

For program P, let $wp[\![P]\!] : \mathbb{P} \to \mathbb{P}$ a predicate transformer.

$G = wp[\![P]\!](F)$ is P's weakest precondition w.r.t. postcondition F iff

- If P starts in a state $s \models G$, it terminates in a state $t \models F$.
- Otherwise, P either terminates in a state $t \not\models F$ or diverges



Weakest preconditions versus Hoare triples

Weakest preconditions are functional

- For each $F \in \mathbb{P}$ there is a unique $G \in \mathbb{P}$ such that

$$wp[\![P]\!](F) \;=\; G$$

- Weakest preconditions respect Hoare triples:

$$\boxed{\{\, wp[\![P]\!](F) \,\} \;\; P \;\; \{\, F \,\} \quad \text{is a valid statement}}$$

- For terminating[1] P:

$$\boxed{\{\, G \,\} \;\; P \;\; \{\, F \,\} \quad \text{is a valid statement, then} \quad \{\, G \,\} \;\Rightarrow\; wp[\![P]\!](F)}$$

[1] For diverging P, the statement $\{\,\text{true}\,\}\; P\; \{\, F\,\}$ is trivially true, but $wp[\![P]\!](F) = \text{false}$.

Weakest preconditions for GCL

Syntax program P	Weakest precondition $wp[\![P]\!](F)$
$\mathtt{skip}$	F
$x := E$	$F[x := E]$
$P; Q$	$wp[\![P]\!](wp[\![Q]\!](F))$
$\mathtt{if}\ (\varphi)\ P\ \mathtt{else}\ Q$	$(\varphi \wedge wp[\![P]\!](F)) \vee (\neg\varphi \wedge wp[\![Q]\!](F))$

Example

$$P = a{+}{+}\ ;\ b{-}{-}$$

$$F = a \cdot b = 0$$

What is $wp[\![P]\!](F)$?

Example

$\mathrm{wp}\,[\![P]\!]\,(\underbrace{y^2 > 0}_{F})$ where program P is:

$$\boxed{\begin{array}{l} \text{if } (y>0)\ \{\ x:=5\ \}\ \text{else}\ \{\ x:=2\ ;\ y{+}{+}\ \}\ ; \\[1em] y := x-3 \end{array}}$$

read this from bottom to top:

// $(y>0 \wedge \text{true}) \vee (\neg(y<0) \wedge \text{false})$ $\equiv y>0$

wp $[\![P]\!]\,(F)$

if $(y>0)$ {

// $(5-3)^2 > 2 \equiv$ true

$x:=5$

// $(x-3)^2 > 2$

} else {

// $(2-3)^2 > 2 \equiv$ false

$x:=2$;

// $(x-3)^2 > 2$

$y{+}{+}$

// $(x-3)^2 > 2$

};

// $(x-3)^2 > 2$

$y := x-3$

// $y^2 > 2$ $\longleftarrow$ post condition

Loops

$$\texttt{while}\ (\varphi)\ \{P\}$$

$$= \texttt{if}\ (\varphi)\ \{\ P\,;\ \texttt{while}(\varphi)\ \{P\}\ \}\ \texttt{else}\ \texttt{skip}$$

Possibly unbounded loops

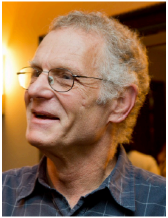
$$wp[\![\text{while } (\varphi)\{ P \}, F]\!] \;=\; \text{lfp}\, X.\, \underbrace{((\varphi \wedge wp[\![P]\!](X)) \vee (\neg\varphi \wedge F))}_{\text{loop characteristic function } \Phi_F(X)}$$

does exist!

- The function $\Phi_F : \mathbb{P} \to \mathbb{P}$ is Scott continuous on $(\mathbb{P}, \sqsubseteq)$.
- Kleene's fixed point theorem yields: $\text{lfp}\, \Phi_F = \sup_{n \in \mathbb{N}} \Phi_F^n(\text{false})$.
- $\Phi_F^n(\text{false})$ denotes the wp of running the loop n times starting from $\varnothing$, the empty set of states.

“Dijkstra’s weakest preconditions go random”

WEAKEST PRE-EXPECTATIONS



Dexter Kozen, Annabelle McIver, and Carroll Morgan

From predicates to quantities

- Let program P be:

```
x := x+5 [4/5] x := 10
```

The expected value of x on P's termination is:

initial value of x

$$\frac{4}{5} \cdot (x+5) + \frac{1}{5} \cdot 10 \;=\; \frac{4x}{5} + 6$$

initial value of x

From predicates to quantities

$$[\varphi](s) = \begin{cases} 1 & \text{if } s \models \varphi \\ 0 & \text{else} \end{cases}$$

$\uparrow$

$\in \mathbb{R}$

- Let program P be:

```
x := x+5 [4/5] x := 10
```

The expected value of x on P's termination is:

$$\frac{4}{5} \cdot (x+5) + \frac{1}{5} \cdot 10 \;=\; \frac{4x}{5} + 6$$

- The probability that x = 10 on P's termination is:

$$\frac{4}{5} \cdot \underbrace{[x+5 = 10]}_{\text{Iverson brackets}} + \frac{1}{5} \cdot 1 \;=\; \frac{4 \cdot [x = 5] + 1}{5}$$

Expectations

The set of expectations[2] (read: random variables):

$$\mathbb{E} \;=\; \left\{ \, {\color{red} f} \mid {\color{red} f} : \underbrace{\mathbb{S}}_{\text{states}} \rightarrow \mathbb{R}_{\geq 0} \cup \{ \, \infty \, \} \right\}$$

[2] $\neq$ expectations in probability theory.

Expectations

The set of expectations[2] (read: random variables):

$$\mathbb{E} \;=\; \left\{ f \mid f : \underbrace{\mathbb{S}}_{\text{states}} \rightarrow \mathbb{R}_{\geq 0} \cup \{\, \infty \,\} \right\}$$

Examples: $[x = 5]$ $\quad \frac{4x}{5} + 6 \quad \frac{4 \cdot [x=5]+1}{5} \quad \mathbf{1} \quad x^2 + \sqrt{(}y+1) \ldots$

[2] $\neq$ expectations in probability theory.

Expectations

$$\boxed{\begin{array}{l} s(x) = 2 \\ s(y) = 3 \end{array}}$$

The set of expectations[2] (read: random variables):

$f: y^2+x$

$$\mathbb{E} \;=\; \left\{ f \mid f : \underbrace{\mathbb{S}}_{\text{states}} \rightarrow \mathbb{R}_{\geq 0} \cup \{ \infty \} \right\}$$

$f(s) = 3^2 + 2$

$= 11$

Examples: $[x = 5]$ $\quad \frac{4x}{5} + 6 \quad \frac{4 \cdot [x=5]+1}{5} \quad \mathbf{1} \quad x^2 + \sqrt{(}y+1) \ldots$

$(\mathbb{E}, \sqsubseteq)$ is a complete lattice where $f \sqsubseteq g$ if and only if $\forall s \in \mathbb{S}.\, f(s) \leq g(s)$

> expectations are the quantitative analogue of predicates

[2] $\neq$ expectations in probability theory.

Weakest pre-expectations

For program P, let $wp[\![P]\!] : \mathbb{E} \rightarrow \mathbb{E}$ an expectation transformer

$g = wp[\![P]\!](f)$ is P's weakest pre-expectation w.r.t. post-expectation f iff

the expected value of f after executing P on input s equals $g(s)$

Weakest pre-expectations

For program P, let $wp[\![P]\!] : \mathbb{E} \to \mathbb{E}$ an expectation transformer

$g = wp[\![P]\!](f)$ is P's weakest pre-expectation w.r.t. post-expectation f iff

the expected value of f after executing P on input s equals $g(s)$

Examples:

For P:: $\boxed{\texttt{x := x+5 [4/5] x := 10}}$ we have:

$$\underbrace{wp[\![P]\!](x) \;=\; \frac{4x}{5}+6}_{\text{expected value of } x} \quad \text{and} \quad \underbrace{wp[\![P]\!]([x=10]) \;=\; \frac{4\cdot[x=5]+1}{5}}_{\text{prob. } x=10}$$

Weakest pre-expectations

For program P, let $wp[\![P]\!] : \mathbb{E} \rightarrow \mathbb{E}$ an expectation transformer

$g = wp[\![P]\!](f)$ is P's weakest pre-expectation w.r.t. post-expectation f iff

the expected value of f after executing P on input s equals $g(s)$

Examples:

For P:: `x := x+5 [4/5] x := 10`, we have:

$$wp[\![P]\!](x) \;=\; \frac{4x}{5} + 6 \quad \text{and} \quad wp[\![P]\!]([x = 10]) \;=\; \frac{4 \cdot [x = 5] + 1}{5}$$

$wp[\![P]\!]([\varphi])$ is the probability of predicate φ on P's termination

$wp[\![P]\!](\mathbf{1})$ is P's termination probability

How to obtain wp for a program?

Syntax probabilistic program P	Semantics $wp[\![P]\!](f)$
skip	f
$x := E$	$f[x := E]$
$x :\approx \mu$	$\sum_{v} f(s[x := v]) \cdot \mu_s(v)$
$P; Q$	$\underbrace{wp[\![P]\!](wp[\![Q]\!](f))}_{\text{backwards!}}$

substitution

Example

program P : $\quad x :\approx \mathrm{unif}\,[y+x,\, 2\cdot y]$

post-expectation f : $\quad [x > 6]$

initial state s : $\quad s(x) = 0,\ s(y) = 4$

What is $\quad wp\,[\![P]\!](f)(s)$?

$$wp\,[\![P]\!](f)(s) = \sum_{v \in \mathbb{Q}} \mu_s(v) \cdot f(s[x := v])$$

$$\mu_s = \mathrm{unif}\,[4+0,\, 2\cdot 4] = \mathrm{unif}\,[4, 8]$$

$$\sum_{v \in \{4,8\}} \frac{1}{5} \cdot f(s\,[x := v])$$

How to obtain wp for a program?

Syntax probabilistic program P	Semantics $wp[\![P]\!](f)$
skip	f
$x := E$	$f[x := E]$
$x :\approx \mu$	$\sum_{v \in \mathbb{Q}} f(s[x := v]) \cdot \mu_s(v)$
$P; Q$	$wp[\![P]\!](wp[\![Q]\!](f))$
`if` (φ) P `else` Q	$[\varphi] \cdot wp[\![P]\!](f) + [\neg\varphi] \cdot wp[\![Q]\!](f)$
$P\,[p]\,Q$	$p \cdot wp[\![P]\!](f) + (1-p) \cdot wp[\![Q]\!](f)$

For $f \in \mathbb{E}$ and $c \in \mathbb{R}_{\geq 0}$, $(c \cdot f)(s) = c \cdot f(s)$

For $f, g \in \mathbb{E}$, let $(f + g)(s) = f(s) + g(s)$

Examples

1. Consider again program P:

```
x := x+5 [4/5] x := 10
```

For $f = x$, we have:

x+5

$$wp[\![P]\!](x) = \frac{4}{5}\cdot wp[\![x := 5]\!](x) + \frac{1}{5}\cdot wp[\![x := 10]\!](x)$$

$$= \frac{4}{5}\cdot(x+5) + \frac{1}{5}\cdot 10 \;=\; \boxed{\frac{4x}{5} + 6}$$

P

post f

$f : S \longrightarrow \mathbb{R}_{\geq 0} + \infty$

$wp[\![P]\!](f) =$ exp. value of f on P's termination

Examples

$wp[\![P]\!]([\varphi])$ = pr. that φ holds on P's termination

1. Consider again program P:

```
x := x+5 [4/5] x := 10
```

For $f = x$, we have:

$$wp[\![P]\!](x) = \frac{4}{5} \cdot wp[\![x := \overset{x+5}{5}]\!](x) + \frac{1}{5} \cdot wp[\![x := 10]\!](x)$$

$$= \frac{4}{5} \cdot (x+5) + \frac{1}{5} \cdot 10 \ = \ \boxed{\frac{4x}{5} + 6}$$

2. For program P (again) and $f = [x = 10]$, we have:

$$\begin{aligned} wp[\![P]\!]([x{=}10]) \quad &= \quad \frac{4}{5} \cdot wp[\![x := x+5]\!]([x{=}10]) + \frac{1}{5} \cdot wp[\![x := 10]\!]([x{=}10] \\ &= \quad \frac{4}{5} \cdot [x+5 = 10] + \frac{1}{5} \cdot [10 = 10] \\ &= \quad \boxed{\frac{4 \cdot [x = 5] + 1}{5}} \end{aligned}$$

Loops

$\in \mathbb{E}$

monotonic
continuous

$$wp[\![\text{while}\,(\varphi)\,\{\,P\,\}]\!](f) \;=\; \text{lfp}\,X.\ \underbrace{\big([\varphi]\cdot wp[\![P]\!](X) + [\neg\varphi]\cdot f\big)}_{\text{loop characteristic function } \Phi_f(X)}$$

$f = x$

$\text{while } (x>0)\ \{\ x{+}{+}\ [\tfrac{1}{2}]\ x:=0\ \}$

$(\mathbb{E}, \sqsubseteq)$: $\quad f \sqsubseteq g$ iff $\forall s \in \mathbb{S}\quad f(s) \le g(s)$

$\in \mathbb{R}_{\geq 0}^{+\infty}$

$f \sqsubseteq g \longrightarrow \Phi_h(f) \sqsubseteq \Phi_h(g)$

$\bigsqcup \Phi_h(f_i) = \Phi_h(\bigsqcup f_i)$

Loops

$$wp[\![\text{while}\,(\varphi)\,\{\,P\,\}]\!](f) \;=\; \text{lfp}\,X.\; \underbrace{\left([\varphi]\cdot wp[\![P]\!](X) \;+\; [\neg\varphi]\cdot f\right)}_{\text{loop characteristic function } \Phi_f(X)}$$

- Function $\Phi_f : \mathbb{E} \to \mathbb{E}$ is Scott continuous on $(\mathbb{E}, \sqsubseteq)$
- By Kleene's fixed point theorem: $\boxed{\text{lfp}\;\Phi_f \;=\; \sup_{n\in\mathbb{N}} \Phi_f^n(\mathbf{0})}$
- $\Phi_f^n(\mathbf{0})$ is f's expected value after n times running P, starting in $\mathbf{0}$

HOW TO TREAT LOOPS?

The good, the bad, and the ugly

A B

Duelling cowboys

$a, b \in [0,1]$

```
int cowboyDuel(float a, b) {
  int t := A
  bool c := true;
  while (c) {
    if (t = A) {
      (c := false [a] t := B);
    } else {
      (c := false [b] t := A);
    }
  }
  return t;
}
```

Duelling cowboys

```
int cowboyDuel(float a, b) { // 0 < a < 1, 0 < b < 1
   int t := A
   bool c := true;
   while (c) {
     if (t = A) {
         (c := false [a] t := B); // A shoots B with prob. a
     } else {
         (c := false [b] t := A); // B shoots A with prob. b
     }
   }
return t; // the survivor
}
```

Cowboy A wins the duel with probability ?

Computing survival probabilities

the probability that cowboy A wins the duel:

$$= \sum_{i=0}^{\infty} \left((1-a) \cdot (1-b) \right)^{i} \cdot a$$

$$= (* \text{ geometric series } *) \quad \frac{a}{a+b-ab}$$

Upper bounds by inductive invariants

Recall:

$$wp[\![\text{while } (\varphi)\,\{\, P \,\}]\!](f) \;=\; \text{lfp}\, X.\; \underbrace{\left([\varphi]\cdot wp[\![P]\!](X) \;+\; [\neg\varphi]\cdot f\right)}_{\Phi_f(X)}$$

By Park's lemma: for while$(\varphi)\{P\}$ and expectations f and I: $\in \mathbb{E}$

$$\boxed{\underbrace{\Phi_f(I) \sqsubseteq I}_{\text{"upper" invariant } I} \qquad \text{implies} \qquad \underbrace{wp[\![\text{while}(\varphi)\{P\}]\!](f)}_{\text{lfp}\,\Phi_f} \sqsubseteq I}$$

such I is an inductive invariant

Upper bounds

Recall:

$$wp[\![\text{while }(\varphi)\{\,P\,\}]\!](f) \;=\; \text{lfp}\, X.\ \underbrace{\left([\varphi]\cdot wp[\![P]\!](X) \;+\; [\neg\varphi]\cdot f\right)}_{\Phi_f(X)}$$

By Park's lemma: for while$(\varphi)\{P\}$ and expectations f and I:

$$\underbrace{\Phi_f(I) \sqsubseteq I}_{\text{“upper” invariant } I} \qquad \text{implies} \qquad \underbrace{wp[\![\text{while}(\varphi)\{P\}]\!](f)}_{\text{lfp}\,\Phi_f} \sqsubseteq I$$

exp. value of geom. distr

Example: `while(c = 0) { ~~x++~~ [p] c := 1 }`

$\underbrace{}_{x:=x+2}$ $\underbrace{\texttt{c := 1}}_{1-p}$

$I \;=\; x + [c=0]\cdot\dfrac{p}{1-p}\cdot 2$ is an “upper”-invariant w.r.t. $f = x$

$P::\quad \mathtt{while}\,(c=0)\,\{\ x{+}{+}\ [p]\ c:=1\ \}$

<u>claim</u> $\boxed{I = x + [c\neq 0]\cdot \frac{p}{1-p}}$ is a wp-superinvariant wrt. $f = x$

$\Phi_f(I) \sqsubseteq I$

@ invariant I

$$\Phi_f(I) = [c\neq 0]\cdot x + [c=0]\ \mathrm{wp}(x{+}{+}\ [p]\ c:=1,\ I)$$

$$= [c\neq 0]\cdot x + [c=0]\cdot\Big(\underbrace{p\Big((x+1) + [c=0]\frac{p}{1-p}\Big)}_{I[x{+}{+}]} + \underbrace{(1-p)\cdot x}_{I[c:=1]}\Big)$$

$$=$$

$$= x + [c=0]\underbrace{\Big(p + \frac{p^2}{1-p}\Big)}_{=\frac{p}{1-p}}$$

$$\Phi_f(I) = \underbrace{x + [c=0]\frac{p}{1-p}}_{=I} \implies$$

only equalities used, so $\Phi_f(I) = I$

i.e. I is a fixed point of Φ_f

Lower bounds by loop unrolling

$$wp[\![\text{while}\,(\varphi)\,\{\,P\,\}]\!](f) \;=\; \text{lfp}\,X.\,\Phi_f(I)$$

$$\boxed{\text{lfp}\;\Phi_f \;=\; \sup_{n\in\mathbb{N}} \Phi_f^n(\mathbf{0})}$$

so: $\underbrace{\Phi_f^0(0),\ \Phi_f^1(0),\ \Phi_f^2(0),\ \Phi_f^3(0)}_{\text{loop unrollings}}\ldots \sqsubseteq \underbrace{\text{lfp}\ \Phi_f}_{wp[\![\text{loop}]\!]}$

$$\Phi_f^k(0) = wp[\![\text{while}^k(e)\{P\}]\!](f)$$

Lower bounds : OST Proof Rule

$$\left(I \sqsubseteq \Phi_f(I) \;\wedge\; \text{side conditions}\right) \quad \textit{implies} \quad I \sqsubseteq \operatorname{lfp} \Phi_f$$

$$\Phi_f(I) \sqsubseteq I \longrightarrow wp[\![loop]\!](f) \sqsubseteq I$$

not: $I \sqsubseteq \Phi_f(I) \not\longrightarrow I \sqsubseteq wp[\![loop]\!](f)$

Dijkstra : wp wlp

$$I \sqsubseteq \Phi_f(I) \longrightarrow I \sqsubseteq \underline{wlp}[\![loop]\!](I)$$

Lower bounds : OST Proof Rule

$$\boxed{\left(I \sqsubseteq \Phi_f(I) \;\wedge\; \text{side conditions}\right) \quad \textit{implies} \quad I \sqsubseteq \operatorname{lfp} \Phi_f}$$

where the side conditions for the loop while(φ){P} are:

1. the loop is PAST, and

   highly undecidable, worse than UH !

2. for any $s \models \varphi$, $\quad wp[\![P]\!](|I(s) - I|)(s) \;\leq\; c \quad$ for some $c \in \mathbb{R}_{\geq 0}$

Lower bounds : OST Proof Rule $\text{lfp} \sqsubseteq I$

PCF

$$\Big(I \sqsubseteq \Phi_f(I) \;\wedge\; \text{side conditions} \Big) \quad \textit{implies} \quad I \sqsubseteq \text{lfp}\, \Phi_f$$

1978 Iranian mathematician

where the side conditions for the loop while$(\varphi)\{P\}$ are:

positive almost sure termination

$\underbrace{\text{almost sure termination}}_{\Pr[\Diamond\, \text{terminate}] = 1}$

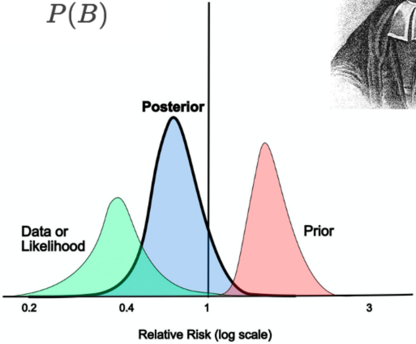
1. the loop is PAST, and
2. for any $s \models \varphi$, $\quad \underbrace{wp[\![P]\!](\overbrace{|I(s) - I|}^{\Delta I})(s) \;\leq\; c}_{\text{“conditional difference boundedness”}}$ $\quad$ for some $c \in \mathbb{R}_{\geq 0}$

Example. Program: `while`$(c = 0)\{\, x{+}{+}\,[p]\, c := 1\,\}$ satisfies the conditions.

$$I \;=\; x + [c = 0] \cdot \frac{p}{1-p} \quad \text{is a “lower”-invariant w.r.t. } f = x$$

BAYESIAN INFERENCE

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$



Bayesian learning by example

```
{ w := 0 } [5/7] { w := 1 };
if (w = 0) { c := poisson(6) }
else { c := poisson(2) };
observe (c = 5)
```

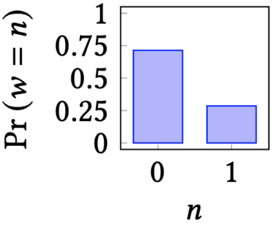
prior

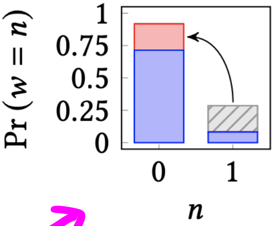
[w=1] ← posterior distr.

What is the posterior distribution of w (weekend or not)?

Bayesian learning by example

```
{ w := 0 } [5/7] { w := 1 };
if (w = 0) { c := poisson(6) }
else { c := poisson(2) };
observe (c = 5)
```





prior

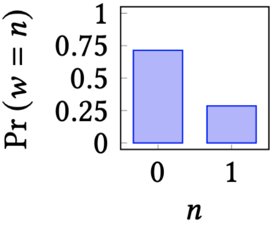
posterior

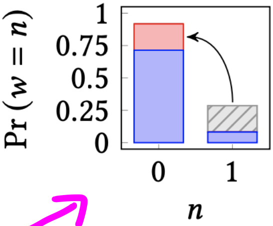
Bayesian learning by example

```
{ w := 0 } [5/7] { w := 1 };
if (w = 0) { c := poisson(6) }
else { c := poisson(2) };
observe (c = 5)
```

prior

posterior





Probability mass is normalised by the probability of feasible runs

Learning

The probability of feasible program runs:

$$wp[\![P]\!](\mathbf{1}) \;=\; 1 - Pr\{\, P \text{ violates an observation} \,\}$$

Learning

The probability of feasible program runs:

$$wp[\![P]\!](\mathbf{1}) \; = \; 1 - Pr\{ P \text{ violates an observation} \}$$

Weakest pre-expectation of observations:

$$wp[\![\text{observe } \varphi]\!](f) \; = \; [\varphi] \cdot \; f$$

Learning

The probability of feasible program runs:

$$wp[\![P]\!](\mathbf{1}) \;=\; 1 - Pr\{\,P \text{ violates an observation}\,\}$$

Weakest pre-expectation of observations:

$$wp[\![\text{observe } \varphi]\!](f) \;=\; [\varphi] \cdot f$$

Normalisation:

$$\frac{wp[\![P]\!](f)}{wp[\![P]\!](\mathbf{1})}$$

Learning

The probability of feasible program runs:

$$wp[\![P]\!](\mathbf{1}) \;=\; 1 - Pr\{\, P \text{ violates an observation} \,\}$$

Weakest pre-expectation of observations:

$$wp[\![\text{observe } \varphi]\!](f) \;=\; [\varphi] \cdot f$$

Nori et al. AAAI 2018/19 if P is AST then $\forall f$

Normalisation:

$$\frac{wp[\![P]\!](f)}{wp[\![P]\!](\mathbf{1})}$$

$wp[\![P]\!](f) =$

$wlp[\![P]\!](f)$

Fine point: under possible program divergence: $\dfrac{wp[\![P]\!](f)}{wlp[\![P]\!](\mathbf{1})}$

The piranha problem [Tijms, 2004]

One fish is contained within the confines of an opaque fishbowl. The fish is equally likely to be a piranha or a goldfish. A sushi lover throws a piranha into the fish bowl alongside the other fish. Then, immediately, before either fish can devour the other, one of the fish is blindly removed from the fishbowl. The fish that has been removed from the bowl turns out to be a piranha. What is the probability that the fish that was originally in the bowl by itself was a piranha?

The piranha program

```
f1 := gf [0.5] f1 := pir;
f2 := pir;
s := f1 [0.5] s := f2;
observe (s = pir)
```

prior

$Pr\{f_1 == pir\} = \frac{1}{2}$

What is the probability that the original fish in the bowl was a piranha?

$$\frac{wp[\![P]\!](f)}{wp[\![P]\!](\mathbf{1})}$$

$f = [f_1 = pir]$

```
f1 := gf [0.5] f1 := pir;
f2 := pir;
s := f1 [0.5] s := f2;
observe (s = pir)
```

$\Big\}\ P$

$f = [f_1 = pir]$

$wp\,[\![P]\!](f)$

$wp\,[\![P]\!](1)$

$[f_1 = pir]$ $\quad 1$

observe $(s = pir)$

$[s = pir] \cdot [f_1 = pir]$ $\quad [s = pir]$

$s := f_1 \ [1/2] \ s := f_2$

$\frac{1}{2}\,\cancel{[f_1 = pir]}\,[f_1 = pir] + \frac{1}{2}\,\underbrace{[f_2 = pir]}_{=1}\,[f_1 = pir]$

$\frac{1}{2}[f_1 = pir] + \frac{1}{2}[f_2 = pir]$

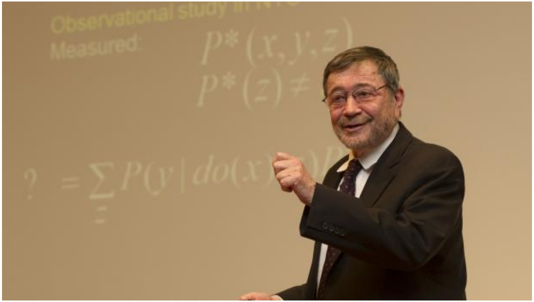
$f_2 := pir$

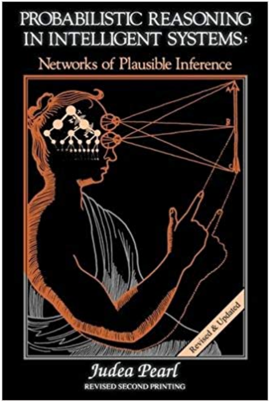
$\frac{1}{2}[f_1 = pir] + \frac{1}{2}[f_1 = pir]$ $\quad \frac{1}{2}[f_1 = pir] + \frac{1}{2}$

$\frac{2}{3}$

$= \frac{1}{2} \Big/ \frac{3}{4}$

Judea Pearl: The father of Bayesian networks

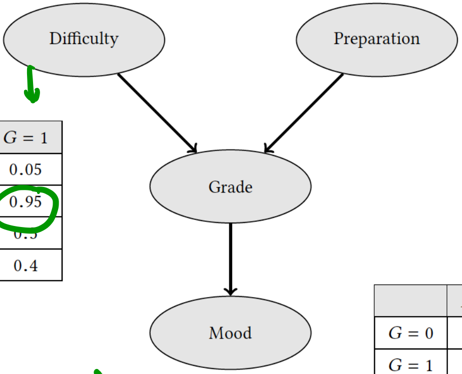




Turing Award 2011: "for fundamental contributions to AI through the development of a calculus for probabilistic and causal reasoning".

Example

$D = 0$	$D = 1$
0.6	0.4



$P = 0$	$P = 1$
0.7	0.3

	$G = 0$	$G = 1$
$D = 0, P = 0$	0.95	0.05
$D = 1, P = 1$	0.05	0.95
$D = 0, P = 1$	0.5	0.5
$D = 1, P = 0$	0.6	0.4

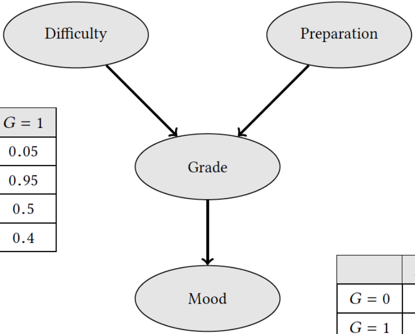
	$M = 0$	$M = 1$
$G = 0$	0.9	0.1
$G = 1$	0.3	0.7

inference :

How likely does a student end up with a bad mood after getting a bad grade for an easy exam, **given that** she is well prepared?

Example

$D=0$	$D=1$
0.6	0.4



$P=0$	$P=1$
0.7	0.3

	$G=0$	$G=1$
$D=0, P=0$	0.95	0.05
$D=1, P=1$	0.05	0.95
$D=0, P=1$	0.5	0.5
$D=1, P=0$	0.6	0.4

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

$$Pr(D=0, G=0, M=0 \mid P=1) = \frac{Pr(D=0, G=0, M=0, P=1)}{Pr(P=1)}$$

$$= \frac{0.6 \cdot 0.5 \cdot 0.9 \cdot 0.3}{0.3} = \mathbf{0.27}$$

The benefits of Bayesian networks

Bayesian networks provide a compact representation of joint distribution functions

if the dependencies between the random variables are sparse.

Another advantage of BNs is

the explicit representation of conditional independencies.

Probabilistic inference

The probabilistic inference problem:

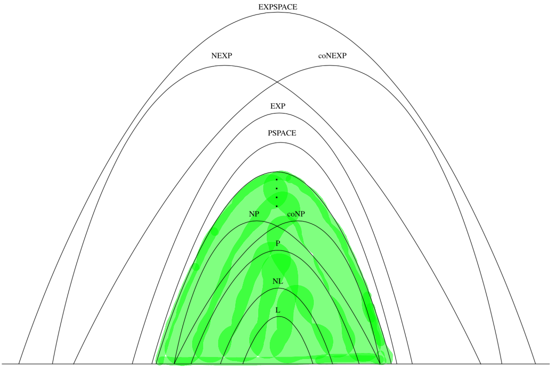
Let BN B with vertex set V, the evidence $\mathbf{E} \subseteq V$ and the questions $\mathbf{Q} \subseteq V$.

Let $\mathbf{e}$ be the value of $\mathbf{E}$, and $\mathbf{q}$ be the value of $\mathbf{Q}$.

(Exact) probabilistic inference is to determine the conditional probability

$$\boxed{Pr(\mathbf{Q} = \mathbf{q} \mid \mathbf{E} = \mathbf{e}) \;=\; \frac{Pr(\mathbf{Q} = \mathbf{q} \wedge \mathbf{E} = \mathbf{e})}{Pr(\mathbf{E} = \mathbf{e})}.}$$

The complexity class PP



NP $\subseteq$ PP (as SAT lies in PP) and coNP $\subseteq$ PP (as PP is closed under complement). PP is contained in PSPACE (as there is a polynomial-space algorithm for MAJSAT).

PP is comparable to the class #P — the counting variant of NP — the class of function problems "compute $f(x)$" where f is the number of accepting runs of an NTM running in polynomial time.

Complexity of probabilistic inference

Decision variants of probabilistic inference. For probability $p \in \mathbb{Q} \cap [0,1)$:

- does $Pr(\mathbf{Q} = \mathbf{q} \mid \mathbf{E} = \mathbf{e}) \; > \; p$? TI
- special case: $Pr(\mathbf{E} = \mathbf{e}) \; > \; p$? STI

Complexity of probabilistic inference **[Cooper, 1990]**

The decision problems TI and STI are PP-complete.

Proof.

1. Hardness: by a reduction of MAJSAT to STI (since STI is a special case of TI, MAJSAT is reducible to TI).
2. Membership: To show TI $\in$ PP, a polynomial-time algorithm is given that guesses a solution to TI while ensuring that the guess is correct with probability $> 1/2$. $\square$

$X = \{x_1, x_2, x_3\}$ $\qquad v: X \to \{tt, ff\}$

① $\alpha_1 = (x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3)$

- is satisfiable, e.g. $v(x_1) = v(x_2) = tt$
- satisfying assignments:

x_1	x_2	x_3
1	1	0
1	0	0
0	0	0
1	1	1

$\notin$ MAJSAT

② $\alpha_2 = (x_1 \vee \neg x_2) \wedge (x_1 \vee \neg x_3)$

- all four assignments with $v(x_1) = 1$ are satisfying $\in$ MAJSAT

  + the assignment $v(x_1) = 0, v(x_2) = 0, v(x_3) = 0$

Rejection sampling

P=1

E

For a given Bayesian network and some evidence:

1. Sample from the joint distribution described by the BN
2. If the sample complies with the evidence, accept the sample and halt
3. If not, repeat sampling (that is: go back to step 1.)

If this procedure is applied N times, N iid-samples result.

Removal of conditioning = rejection sampling

```
x := 0 [p] x := 1;
y := 0 [p] y := 1;
observe(x != y)
```

⇝

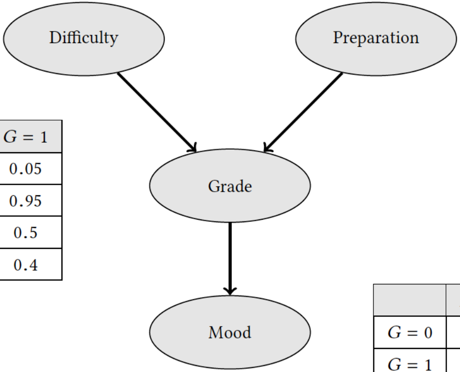
```
sx, sy := x, y; flag := true;
while(flag) {
  x, y := sx, sy; flag := false;
  x := 0 [p] x := 1;
  y := 0 [p] y := 1;
  flag := (x = y)
}
```

This program transformation replaces observe-statements by loops.

The resulting loopy programs represent rejection sampling.

Student exam's mood example

$D = 0$	$D = 1$
0.6	0.4



$P = 0$	$P = 1$
0.7	0.3

	$G = 0$	$G = 1$
$D = 0, P = 0$	0.95	0.05
$D = 1, P = 1$	0.05	0.95
$D = 0, P = 1$	0.5	0.5
$D = 1, P = 0$	0.6	0.4

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

How likely does a student end up with a bad mood after getting a bad grade for an easy exam, **given that** she is well prepared?

Bayesian networks as programs

- Take a topological sort of the BN's vertices, e.g., D; P; G; M
- Map each conditional probability table (aka: node) to a program, e.g.:

```
if (xD = 0 && xP = 0) {
   xG := 0 [0.95] xG := 1
   } else if (xD = 1 && xP = 1) {
   xG := 0 [0.05] xG := 1
   } else if (xD = 0 && xP = 1) {
   xG := 0 [0.5] xG := 1
   } else if (xD = 1 && xP = 0) {
   xG := 0 [0.6] xG := 1
}
```

	$G = 0$	$G = 1$
$D = 0, P = 0$	0.95	0.05
$D = 1, P = 1$	0.05	0.95
$D = 0, P = 1$	0.5	0.5
$D = 1, P = 0$	0.6	0.4

- Condition on the evidence, e.g., for $P = 1$ ("well-prepared"):

```
progD ; progP; progG ; progM ; observe (P=1)
```

Soundness

For BN B with evidence $E \subseteq V$ and value $\underline{v}$ for vertex v:

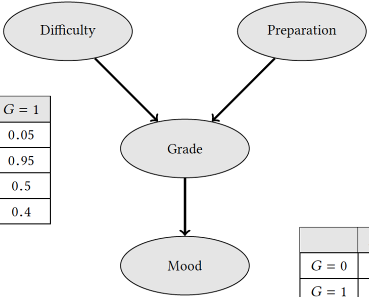
$$\underbrace{wp[\![\mathtt{prog}(B, \mathbf{e})]\!]\left(\bigwedge_{v \in V \setminus E} x_v = \underline{v}\right)}_{\text{wp of the BN program of } B} \quad = \quad \underbrace{Pr\left(\bigwedge_{v \in V \setminus E} v = \underline{v} \;\Big|\; \bigwedge_{e \in E} e = \underline{e}\right)}_{\text{joint distribution of BN } B}$$

where $\mathtt{prog}(B, \mathbf{e})$ equals $progB$; observe$\left(\bigwedge_{e \in E} x_e = \underline{e}\right)$.

Thus: $\boxed{\text{inference of BNs can be done using wp-reasoning}}$

Inference by wp-reasoning

$D=0$	$D=1$
0.6	0.4



$P=0$	$P=1$
0.7	0.3

	$G=0$	$G=1$
$D=0, P=0$	0.95	0.05
$D=1, P=1$	0.05	0.95
$D=0, P=1$	0.5	0.5
$D=1, P=0$	0.6	0.4

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

Ergo: exact Bayesian inference can be done by wp-reasoning, e.g.,

$$wp[\![P_{mood}]\!]([x_D = 0 \wedge x_G = 0 \wedge x_M = 0]) \;=\; \frac{Pr(D=0, G=0, M=0, P=1)}{Pr(P=1)} \;=\; \mathbf{0.27}$$

"How long does your program take on average?"

EXPECTED RUNTIMES

The runtime of a probabilistic program

The runtime of a probabilistic program depends

on the input and

on the internal randomness of the program.

Expected runtimes

Expected run-time of program P on input s:

$$\sum_{i=1}^{\infty} i \cdot Pr\left(\begin{array}{c} \text{“}P \text{ terminates after} \\ i \text{ loops on input } s\text{”} \end{array} \right)$$

$ert[\![P]\!](t)(s)$ = expected runtime of P on s where t is runtime after P

Counting loops is incorrect

expected runtime ∞

P:

```
while(true) { skip; x++ }
```

$wp [\![P]\!] (x)$

- Post: x, as seemingly x counts #loop iterations
- Characteristic function: $\Phi_x(Y) \;=\; Y(x \mapsto x+1)$
- Candidate upper bound: $I \;=\; \mathbf{0}$
- Induction: $\Phi_x(I) \;=\; \mathbf{0}(x \mapsto x+1) \;=\; \mathbf{0} \;=\; I \;\sqsubseteq\; I$
- By Park induction: $\Phi_x(I) \;\sqsubseteq\; I$ implies $wp[\![\text{loop}]\!](x) \;\sqsubseteq\; I$

We — wrongly — get runtime $\mathbf{0}$. wp is unsound for expected runtimes.

Expected run-time transformer

solution: slightly adapt definition of wp

$$ert[\![\text{while}(\varphi)\{P\}]\!](t) =$$

$$\text{lfp}\ X.\ \left(1 + [\varphi] \cdot ert[\![P]\!](X) + [\neg\varphi] \cdot t \right)$$

"counts" the # loop iterations

That's all!

$ert[\![P]\!](t)(s)$ = expected runtime of P on s where t is runtime after P

Positive almost-sure termination

For every pGCL program P and input state s:

$$\underbrace{\mathit{ert}[\![P]\!](\mathbf{0})(s) \; < \; \infty}_{\text{positive a.s-termination on } s} \qquad \text{implies} \qquad \underbrace{\mathit{wp}[\![P]\!](\mathbf{1})(s) \; = \; \mathbf{1}}_{\text{a.s.–termination on } s}$$

Moreover:

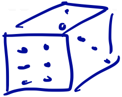
$$\underbrace{\mathit{ert}[\![P]\!](\mathbf{0}) \; < \; \infty}_{\text{universal positive a.s–termination}} \qquad \text{implies} \qquad \underbrace{\mathit{wp}[\![P]\!](\mathbf{1}) \; = \; \mathbf{1}}_{\text{universal a.s.–termination}}$$

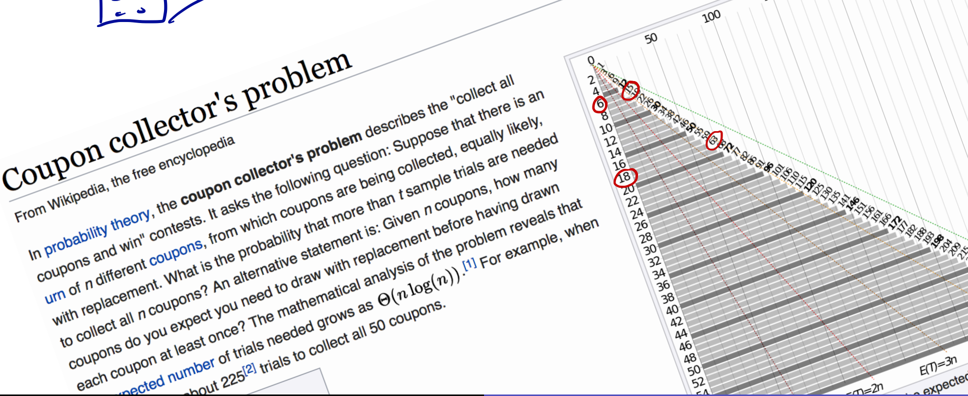
Coupon collector's problem

ON A CLASSICAL PROBLEM OF PROBABILITY THEORY

by

P. ERDŐS and A. RÉNYI




Coupon collector's problem

From Wikipedia, the free encyclopedia

In probability theory, the **coupon collector's problem** describes the "collect all coupons and win" contests. It asks the following question: Suppose that there is an urn of n different coupons, from which coupons are being collected, equally likely, with replacement. What is the probability that more than t sample trials are needed to collect all n coupons? An alternative statement is: Given n coupons, how many coupons do you expect you need to draw with replacement before having drawn each coupon at least once? The mathematical analysis of the problem reveals that the expected number of trials needed grows as $\Theta(n\log(n))$.[1] For example, when ...about 225[2] trials to collect all 50 coupons.

Coupon collector's problem

N

```
cp := [0,...,0]; // no coupons yet
i := 1; // coupon to be collected next
x := 0: // number of coupons collected
while (x < N) {
   while (cp[i] != 0) {
       i := uniform(1..N) // next coupon
   }
   cp[i] := 1;  // coupon i obtained
   x++;  // one coupon less to go
}
```

The expected runtime of this program is in $\Theta(N \cdot \log N)$.

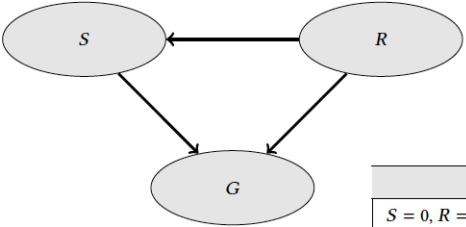
How long to sample a BN?

[Gordon, Nori, Henzinger, Rajamani, 2014]

“the main challenge in this setting [sampling-based approaches] is that many samples that are generated during execution are ultimately rejected for not satisfying the observations."

Sampling time of a toy Bayesian network

	$S = 0$	$S = 1$
$R = 0$	a	$1 - a$
$R = 1$	0.2	0.8



$R = 0$	$R = 1$
a	$1 - a$

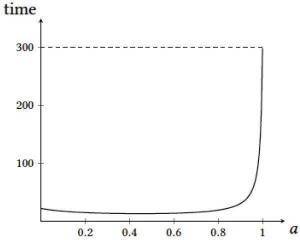
	$G = 0$	$G = 1$
$S = 0, R = 0$	0.01	0.99
$S = 0, R = 1$	0.25	0.75
$S = 1, R = 0$	0.9	0.1
$S = 1, R = 1$	0.2	0.8

This BN is parametric (in a)

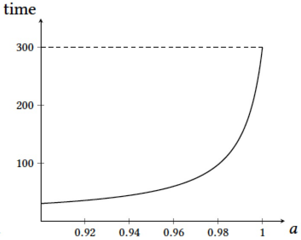
How many samples are needed on average for a **single** iid-sample for evidence $G = 0$?

Sampling time for example BN

Rejection sampling for $G = 0$ requires $\dfrac{200a^2 - 40a - 460}{89a^2 - 69a - 21}$ samples:



zooming in



For $a \in [0.1, 0.78]$, < 18 samples; for $a \geq 0.98$, 100 samples are needed

For real-life BNs, one may exceed 10^{15} (or more) samples

How Long to Simulate a Bayes Network?

\# observed variables

expected simulation time

verification time

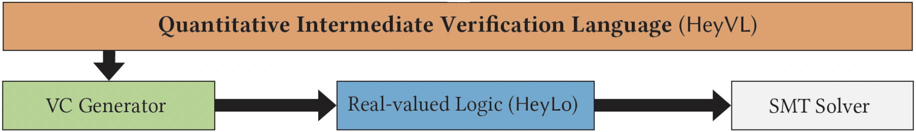
Benchmark BNs from `www.bnlearn.com`

BN	$\lvert V \rvert$	$\lvert E \rvert$	aMB	$\lvert O \rvert$	EST	time (s)
`hailfinder`	56	66	3.54	5	$5\;10^5$	0.63
`hepar2`	70	123	4.51	1	$1.5\;10^2$	1.84
`win95pts`	76	112	5.92	3	$4.3\;10^5$	0.36
`pathfinder`	135	200	3.04	7	∞	5.44
`andes`	223	338	5.61	3	$5.2\;10^3$	1.66
`pigs`	441	592	3.92	1	$2.9\;10^3$	0.74
`munin`	1041	1397	3.54	5	∞	1.43

aMB = *average Markov Blanket*, a measure of independence in BNs

Caesar

A Deductive Verifier for Probabilistic Programs



Caesar: A verification infrastructure for probabilistic programs

`caesarverifier.org`