CS 4110

Probabilistic Programming

Probabilistic Programming

It's not about writing software.

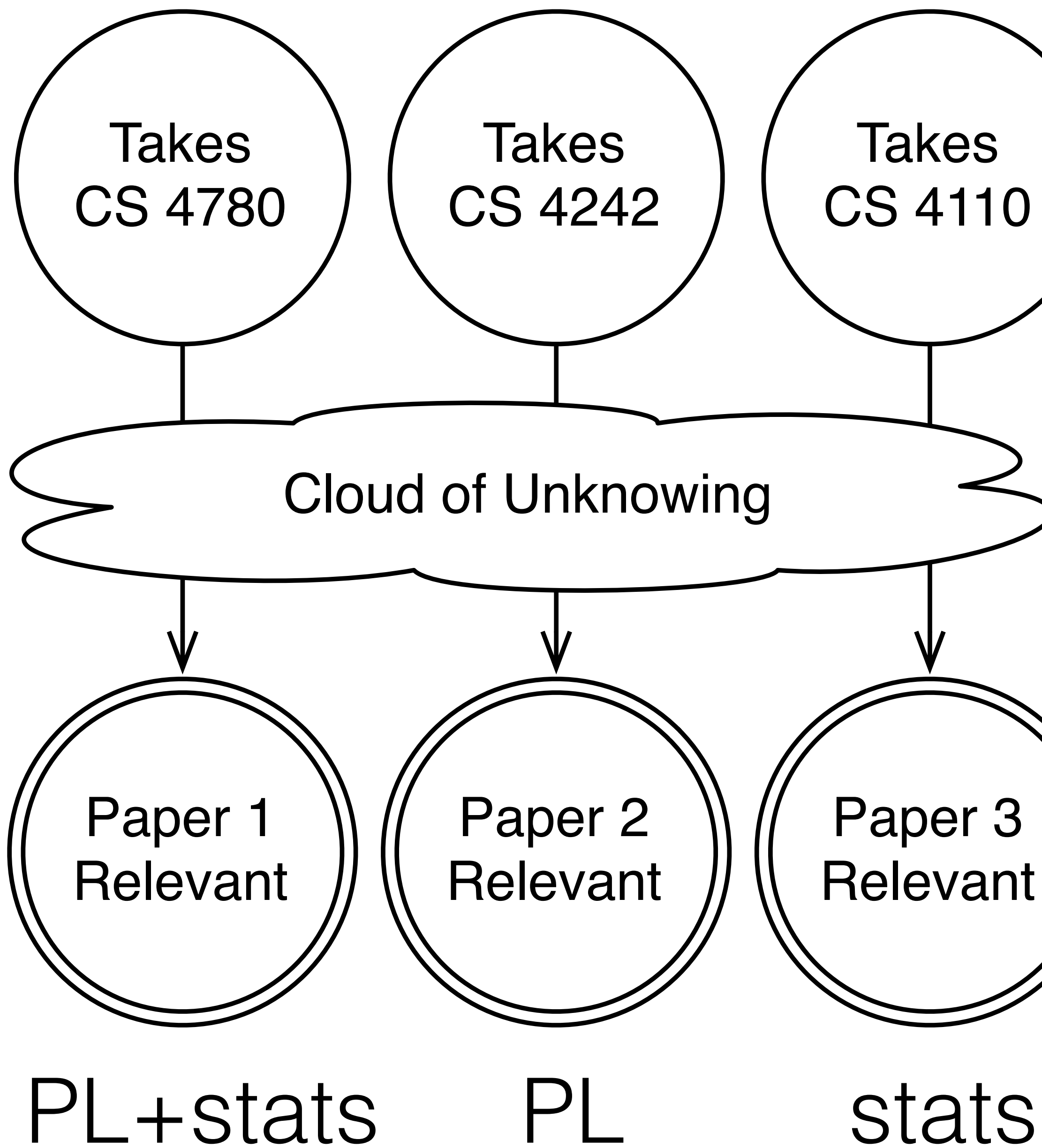
Probabilistic Programming

Probabilistic programming is
a tool for statistical modeling.

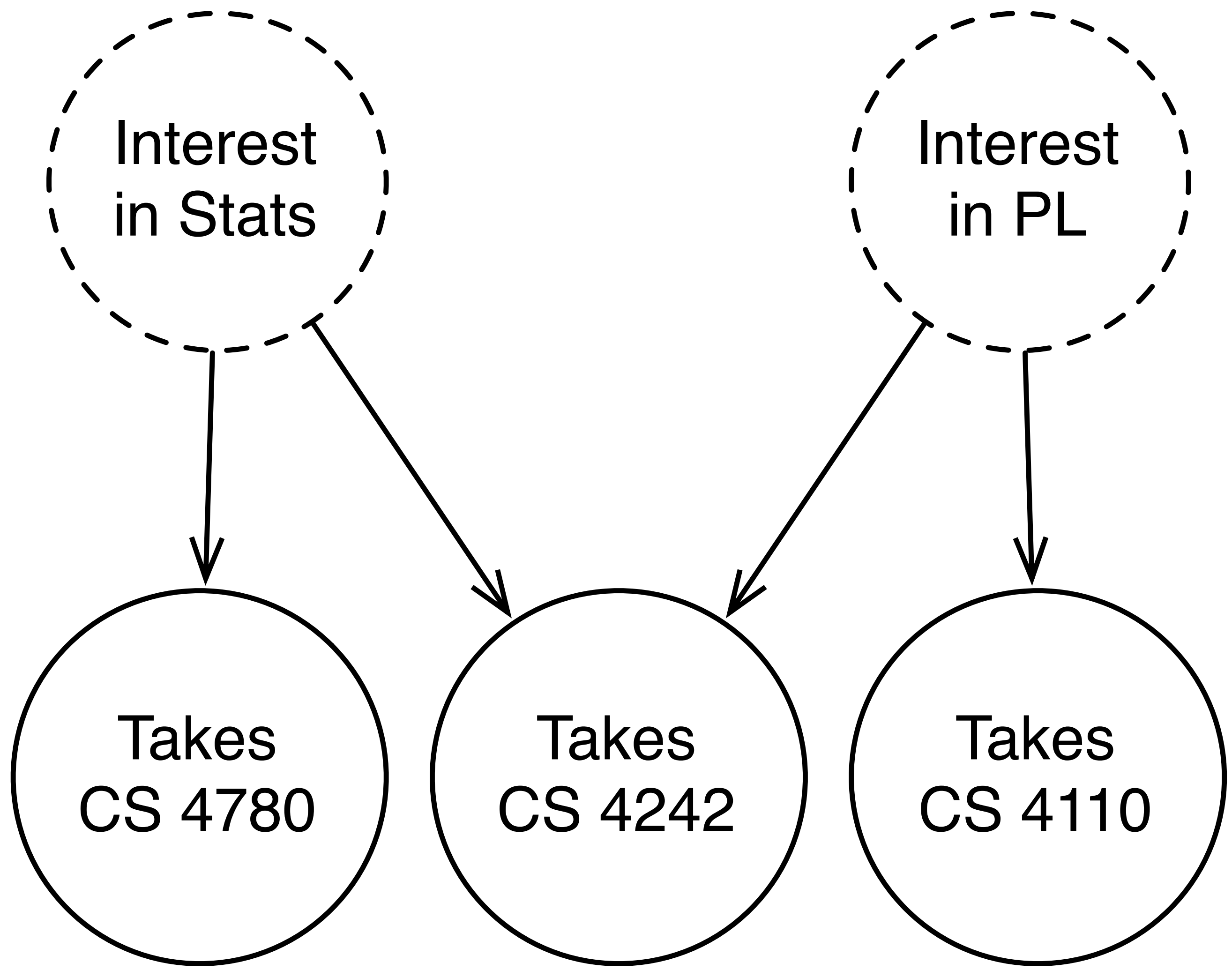
OR

A probabilistic programming language
is a plain old programming language
with `rand(3)` and a suite of fancy analysis tools
for understanding its probabilistic behavior.

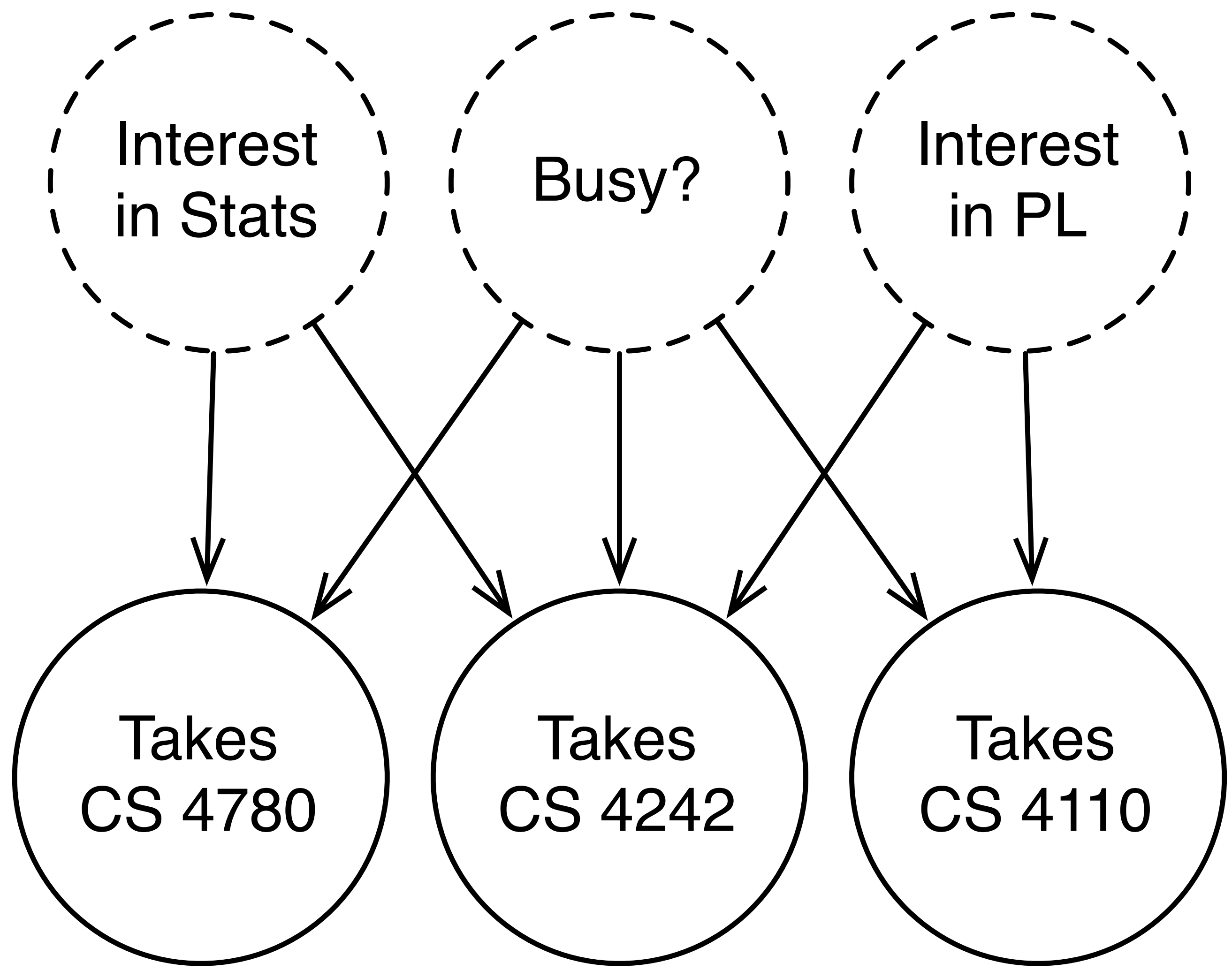
An Example Model



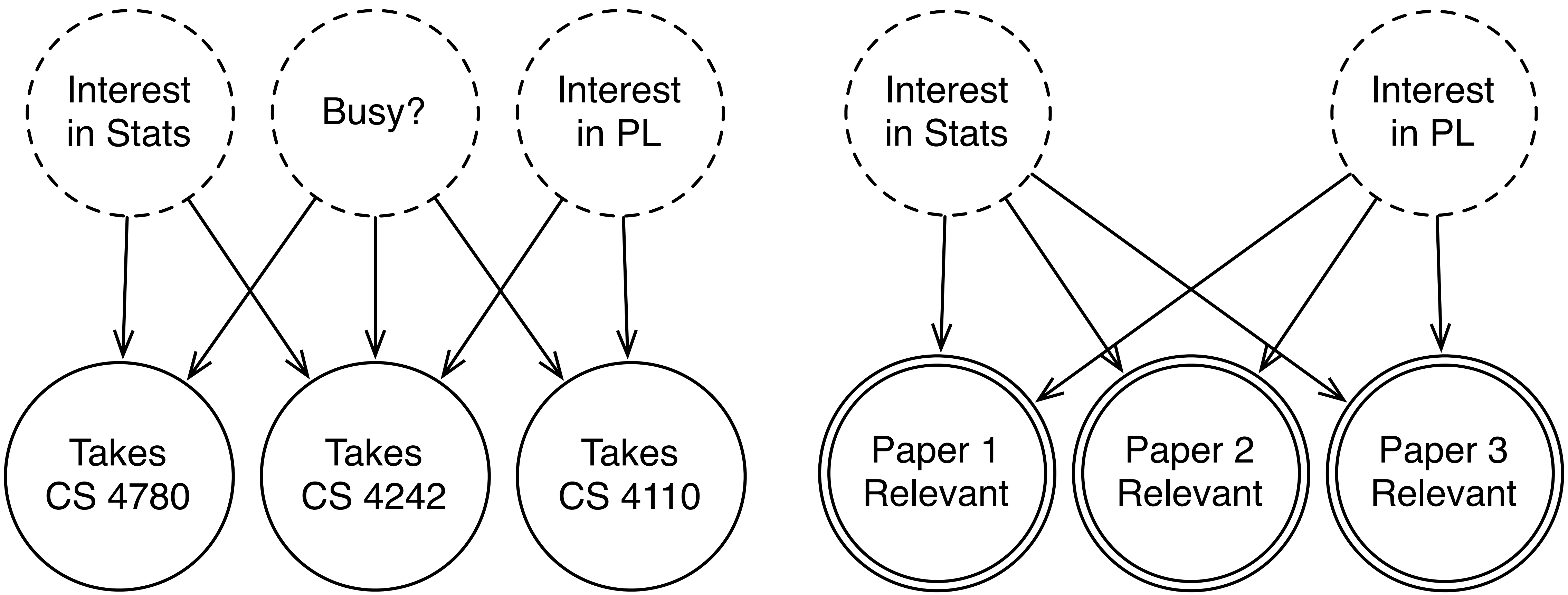
A Model for Humans



A Model for Humans



A Model for Humans



A Model for Humans

$$\Pr[A_{\mathrm{NIPS}} | I_{\mathrm{stats}} \wedge B] = 0.3$$
$$\Pr[A_{\mathrm{NIPS}} | I_{\mathrm{stats}} \wedge \neg B] = 0.8$$
$$\Pr[A_{\mathrm{NIPS}} | \neg I_{\mathrm{stats}}] = 0.1$$
$$\cdots$$
$$\Pr[A_{\mathrm{Dagstuhl}} | I_{\mathrm{stats}} \wedge I_{\mathrm{PL}}] = 0.3$$
$$\Pr[A_{\mathrm{Dagstuhl}} | I_{\mathrm{stats}} \wedge I_{\mathrm{PL}} \wedge \neg B] = 0.8$$
$$\Pr[A_{\mathrm{Dagstuhl}} | \neg(I_{\mathrm{stats}} \vee I_{\mathrm{PL}})] = 0.1$$
$$\cdots$$
$$R_1 \sim I_{\mathrm{PL}} \wedge I_{\mathrm{stats}}$$
$$R_2 \sim I_{\mathrm{PL}}$$
$$R_3 \sim I_{\mathrm{stats}}$$

Whither reuse?

Whither intermediate variables?

Whither abstraction?

Writing even this tiny model feels like **drudgery.**

(and we haven't even gotten to the hard part yet)

- What and Why
- **The Basics and Examples**
- Applications
- Current Problems

webppl.org


webppl.org
webppl
On Github
webppl is a small but feature-rich probabilistic programming language embedded in Javascript.
// Conference attendance.
var attendance = function(i_pl, i_stats, busy) {
var attendance = function (interest, busy, weight) {
if (interest) {

Our First Probabilistic Program

```
var b = flip(0.5);
b ? "yes" : "no"
```

Enumeration

```
var roll = function () {
  var die1 = randomInteger(6) + 1;
  var die2 = randomInteger(6) + 1;
  return die1 + die2;
}

Enumerate(roll)
```

Our Basic Model in webppl

model.wppl (~/science/ppl-intro/code) - VIM

```
// Class attendance model.
var attendance = function(i_pl, i_stats, busy) {
  var attendance = function (interest, busy) {
    if (interest) {
      return busy ? flip(0.3) : flip(0.8);
    } else {
      return flip(0.1);
    }
  }
  var a_4110 = attendance(i_pl, busy);
  var a_4780 = attendance(i_stats, busy);
  var a_4242 = attendance(i_pl && i_stats, busy);

  return {cs4110: a_4110, cs4780: a_4780, cs4242: a_4242};
}

// Relevance of our three papers.
var relevance = function(i_pl, i_stats) {
  var rel1 = i_pl && i_stats;
  var rel2 = i_pl;
  var rel3 = i_stats;

  return {paper1: rel1, paper2: rel2, paper3: rel3};
}

// A combined model.
var model = function() {
  // Some even random priors for our "student profile."
  var i_pl = flip(0.5);
  var i_stats = flip(0.5);
  var busy = flip(0.5);

  return [relevance(i_pl, i_stats), attendance(i_pl, i_stats, busy)];
}

var dist = Enumerate(model);
viz.auto(dist);
~
~
~
~
~
```

Conditioning

```
var roll = function () {
  var die1 = randomInteger(6) + 1;
  var die2 = randomInteger(6) + 1;
  if (!(die1 === 4 || die2 === 4)) {
    factor(-Infinity);
  }
  return die1 + die2;
}

Enumerate(roll)
```

Conditioning on Observations

```
// Discard any executions that
// don't sum to 10.
var out = die1 + die2;
if (out !== 10) {
  factor(-Infinity);
}

// Return the values on the dice.
return [die1, die2];
```

Recommending Papers

```
// Require my class attendance.
var att = attendance(i_pl, i_stats,
                     busy);
require(att.cs4110 && att.cs4242
        && !att.cs4780);

return relevance(i_pl, i_stats);
```

Inference Algorithms

Enumerate is the simplest possible *inference* strategy.

- What and Why
- The Basics and Examples
- **Applications**
- Current Problems

TrueSkill

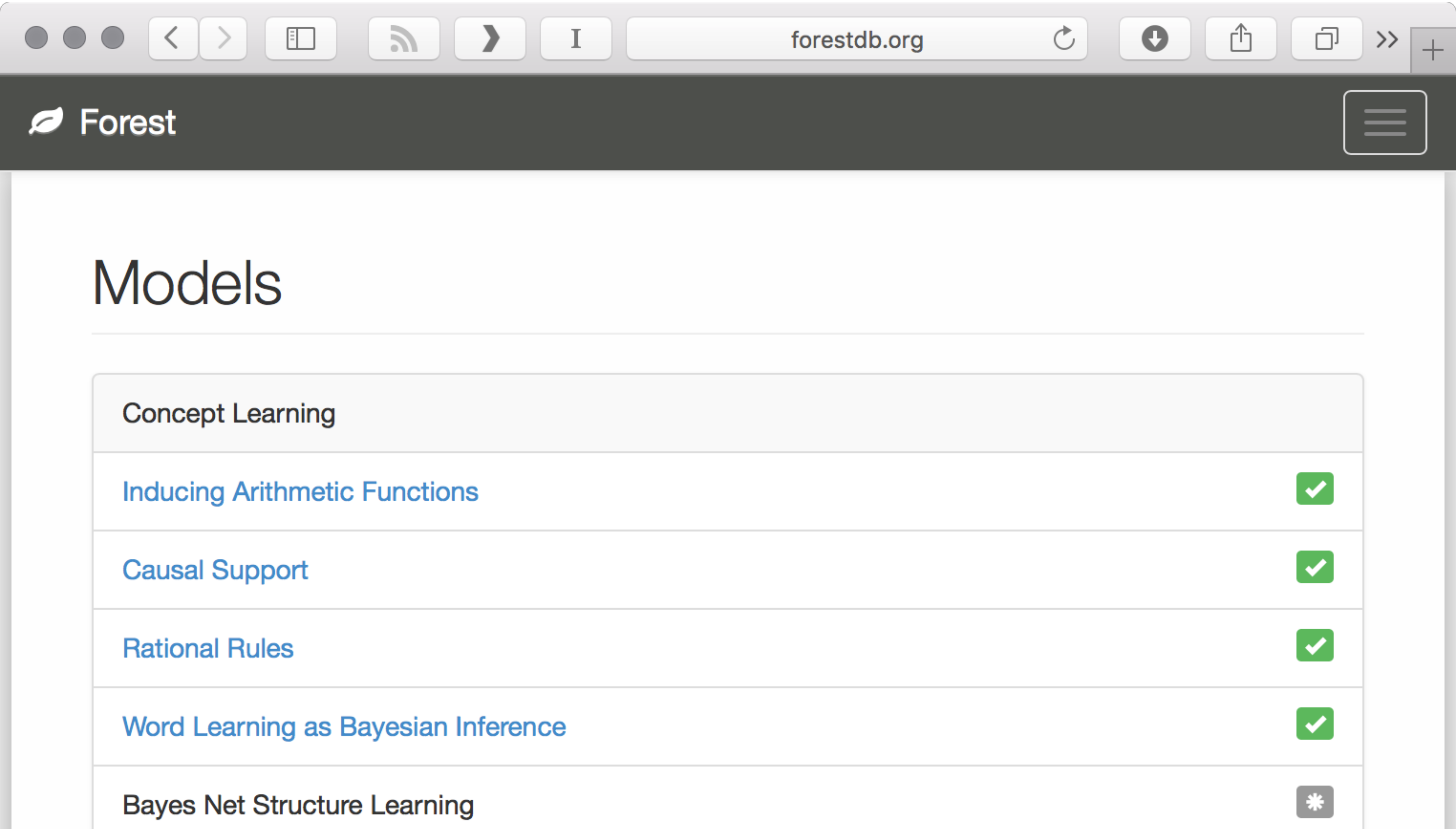
Measure Transformer Semantics for
Bayesian Machine Learning

Johannes Borgström Andrew D. Gordon
Michael Greenberg James Margetson Jurgen Van Gael

```
// prior distributions, the hypothesis
let skill() = random (Gaussian(10.0,20.0))
let Alice,Bob,Cyd = skill(),skill(),skill()
// observe the evidence
let performance player = random (Gaussian(player,1.0))
observe (performance Alice > performance Bob) //Alice beats Bob
observe (performance Bob > performance Cyd) //Bob beats Cyd
observe (performance Alice > performance Cyd) //Alice beats Cyd
// return the skills
Alice,Bob,Cyd
```

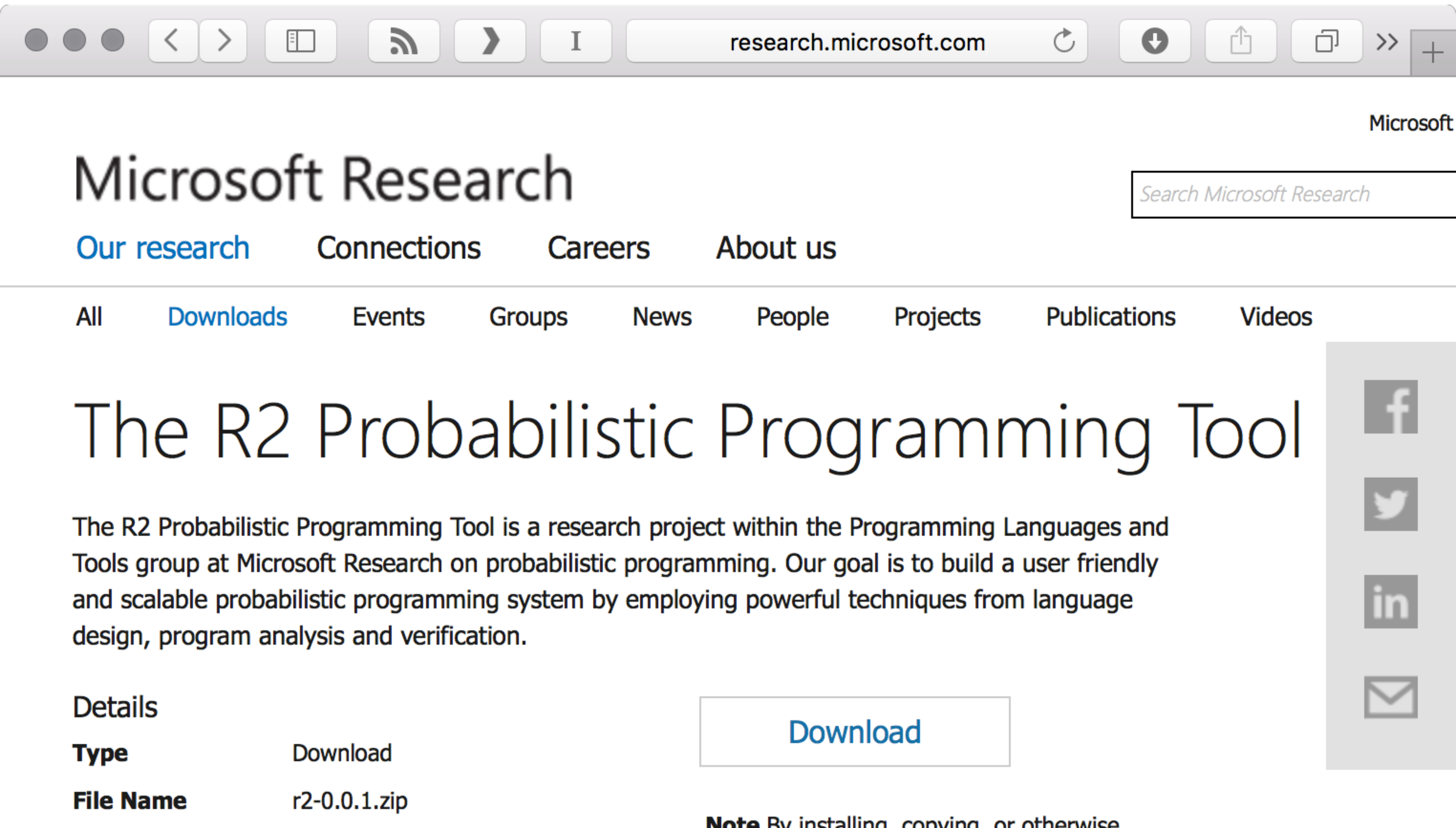
webppl Vision Demo

Forestdb.org



- What and Why
- The Basics and Examples
- Applications
- **Current Research**

R2



research.microsoft.com

Microsoft

Microsoft Research

Search Microsoft Research

Our research | Connections | Careers | About us

All | Downloads | Events | Groups | News | People | Projects | Publications | Videos

The R2 Probabilistic Programming Tool

The R2 Probabilistic Programming Tool is a research project within the Programming Languages and Tools group at Microsoft Research on probabilistic programming. Our goal is to build a user friendly and scalable probabilistic programming system by employing powerful techniques from language design, program analysis and verification.

Details

Type Download

File Name r2-0.0.1.zip

Download

Note By installing, copying, or otherwise

R2's weakest preconditions

```
var die1 = randomInteger(7) + 1;
var die2 = randomInteger(7) + 1;

// Discard any executions that
// don't sum to 10.
var out = die1 + die2;
require(out === 10);
```

wasted work!

R2’s weakest preconditions

```
var die1 = randomInteger(7) + 1;
var die2 = randomInteger(7) + 1;

require(
  (die1 == 3 && die2 == 7) || …);
var out = die1 + die2;
require(out === 10);
```

R2’s weakest preconditions

```
var die1 = randomInteger(7) + 1;
var die2 = randomInteger(7) + 1;

require(
  (die1 == 3 && die2 == 7) || …);
var out = die1 + die2;
```

Probabilistic assertions: design goals

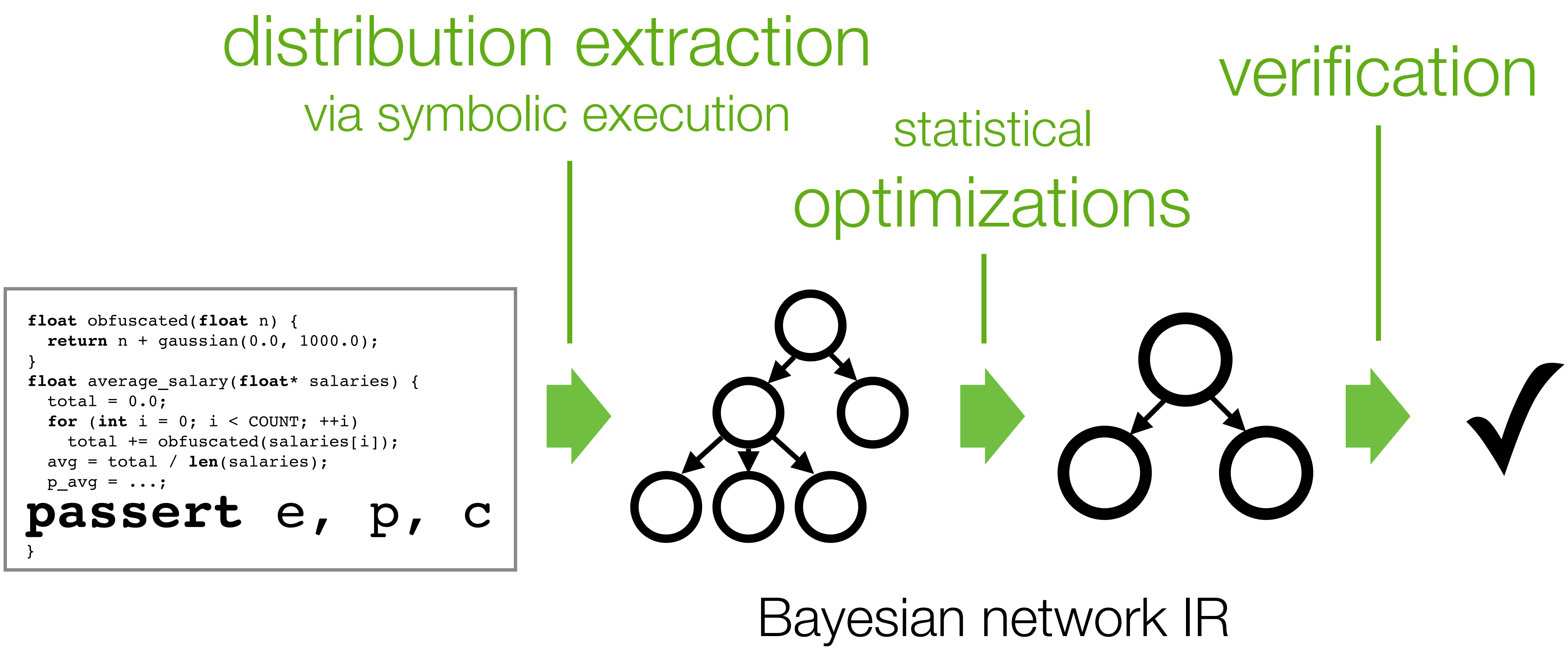
Work on a messy, mainstream language (C and C++)

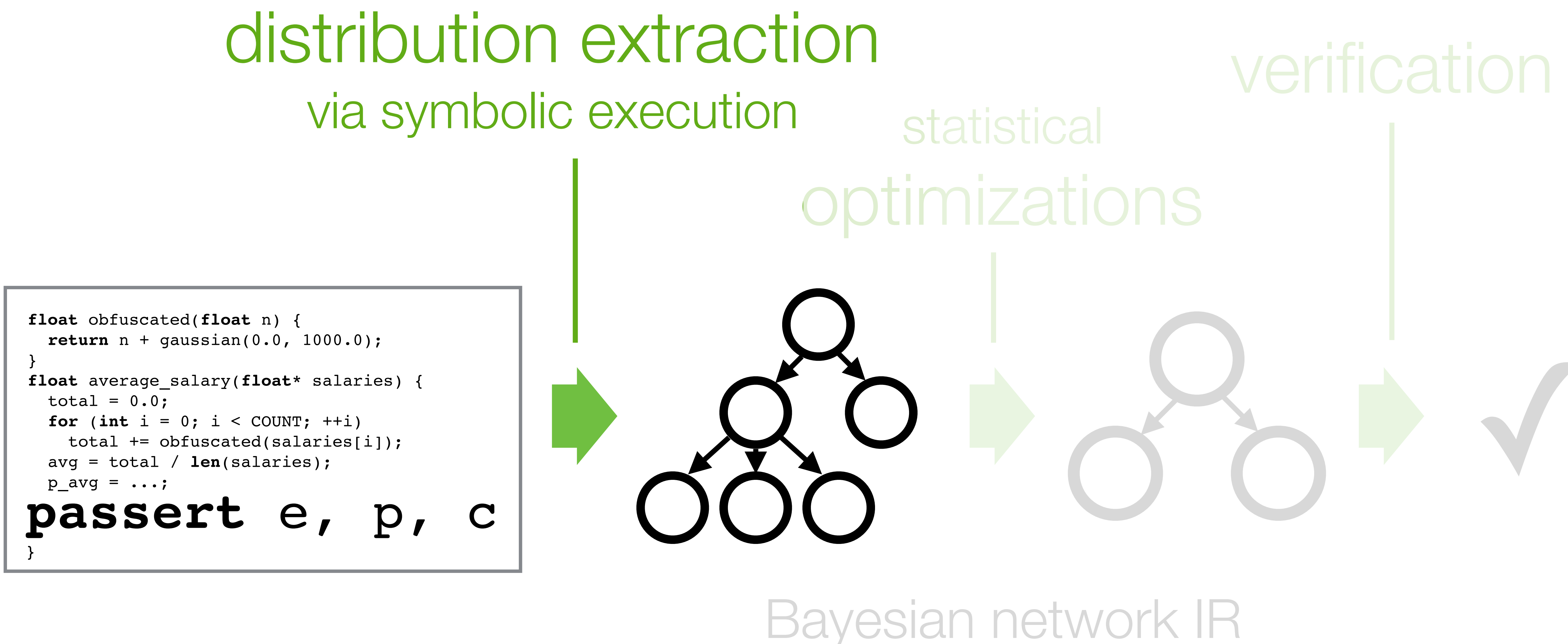
Efficiently check statistical properties of the output

We don't care about conditioning

`passert` *e*, *p*, *c*

e must hold with probability *p*
at confidence *c*


distribution extraction
via symbolic execution
statistical
optimizations
verification
float obfuscated(float n) {
  return n + gaussian(0.0, 1000.0);
}
float average_salary(float* salaries) {
  total = 0.0;
  for (int i = 0; i < COUNT; ++i)
    total += obfuscated(salaries[i]);
  avg = total / len(salaries);
  p_avg = ...;
passert e, p, c
}
✓
Bayesian network IR


distribution extraction
via symbolic execution
statistical
optimizations
verification
float obfuscated(float n) {
return n + gaussian(0.0, 1000.0);
}
float average_salary(float* salaries) {
total = 0.0;
for (int i = 0; i < COUNT; ++i)
total += obfuscated(salaries[i]);
avg = total / len(salaries);
p_avg = ...;
passert e, p, c
}
Bayesian network IR

Distribution extraction: random draws are symbolic

symbolic heap

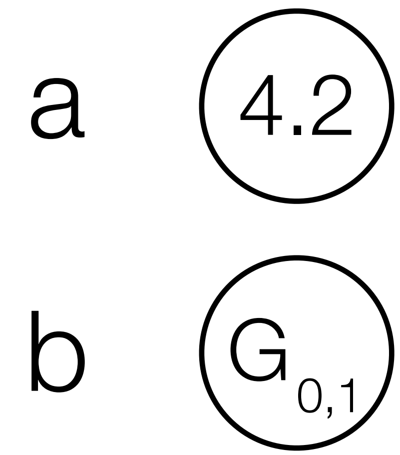
a	4.2

```
b = a + gaussian(0.0, 1.0)
```

a	4.2
b	$4.2 + G_{0,1}$

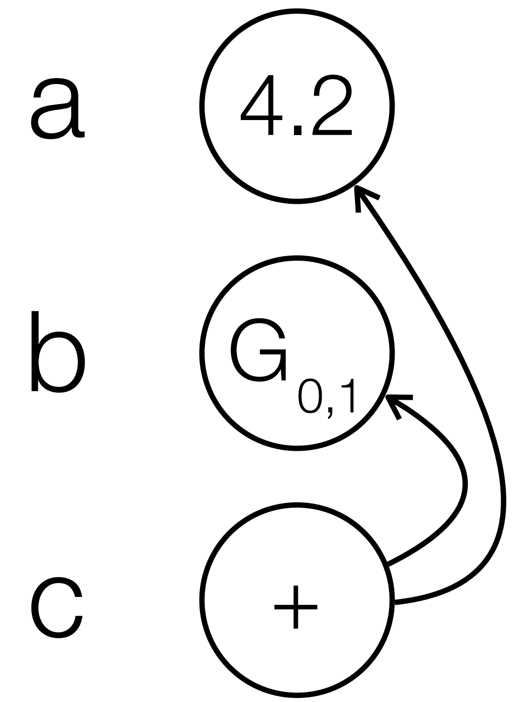
input: `a = 4.2`

```
b = gaussian(0.0, 1.0)
```



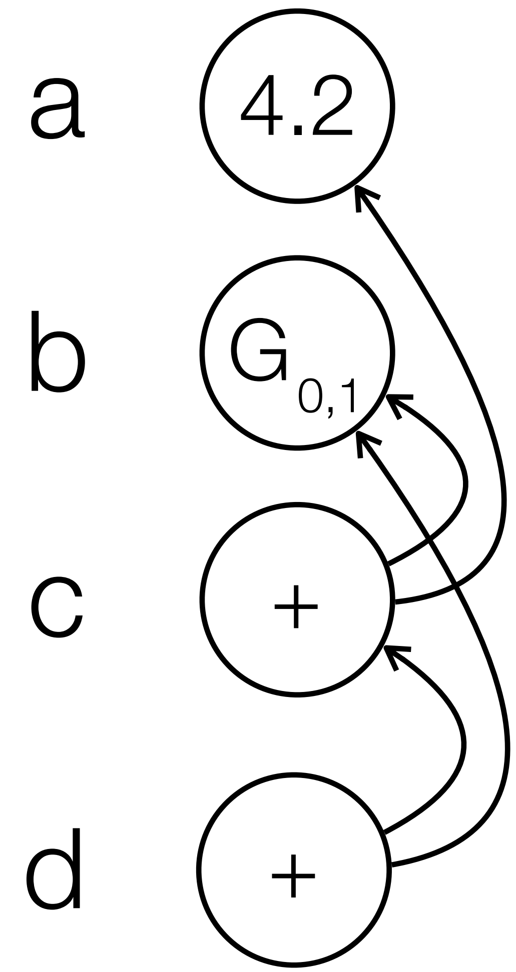
input: `a = 4.2`

```
b = gaussian(0.0, 1.0)
c = a + b
```



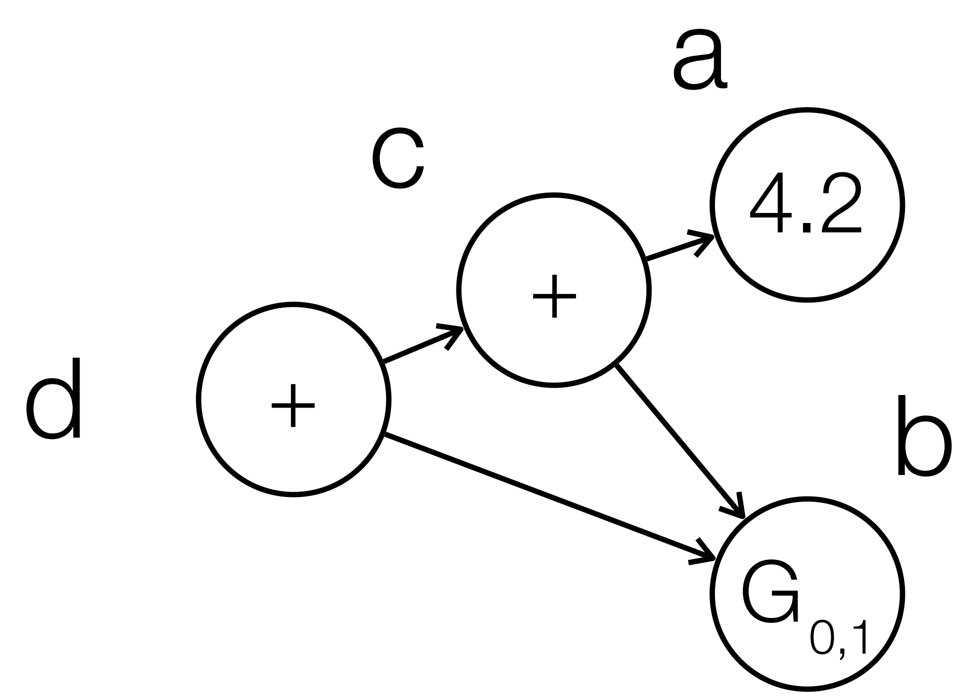
input: `a = 4.2`

```
b = gaussian(0.0, 1.0)
c = a + b
d = c + b
```



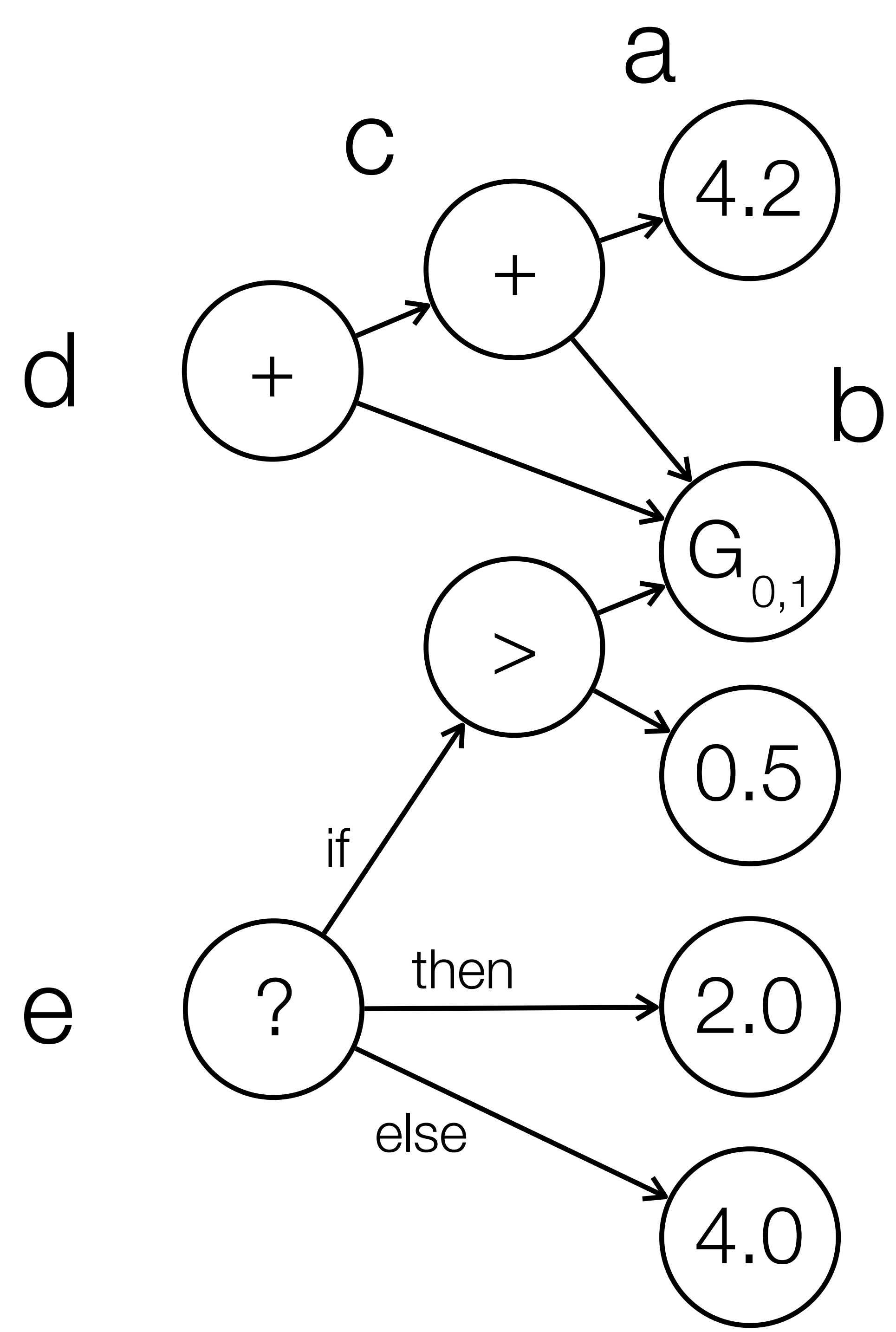
input: `a = 4.2`

```
b = gaussian(0.0, 1.0)
c = a + b
d = c + b
```



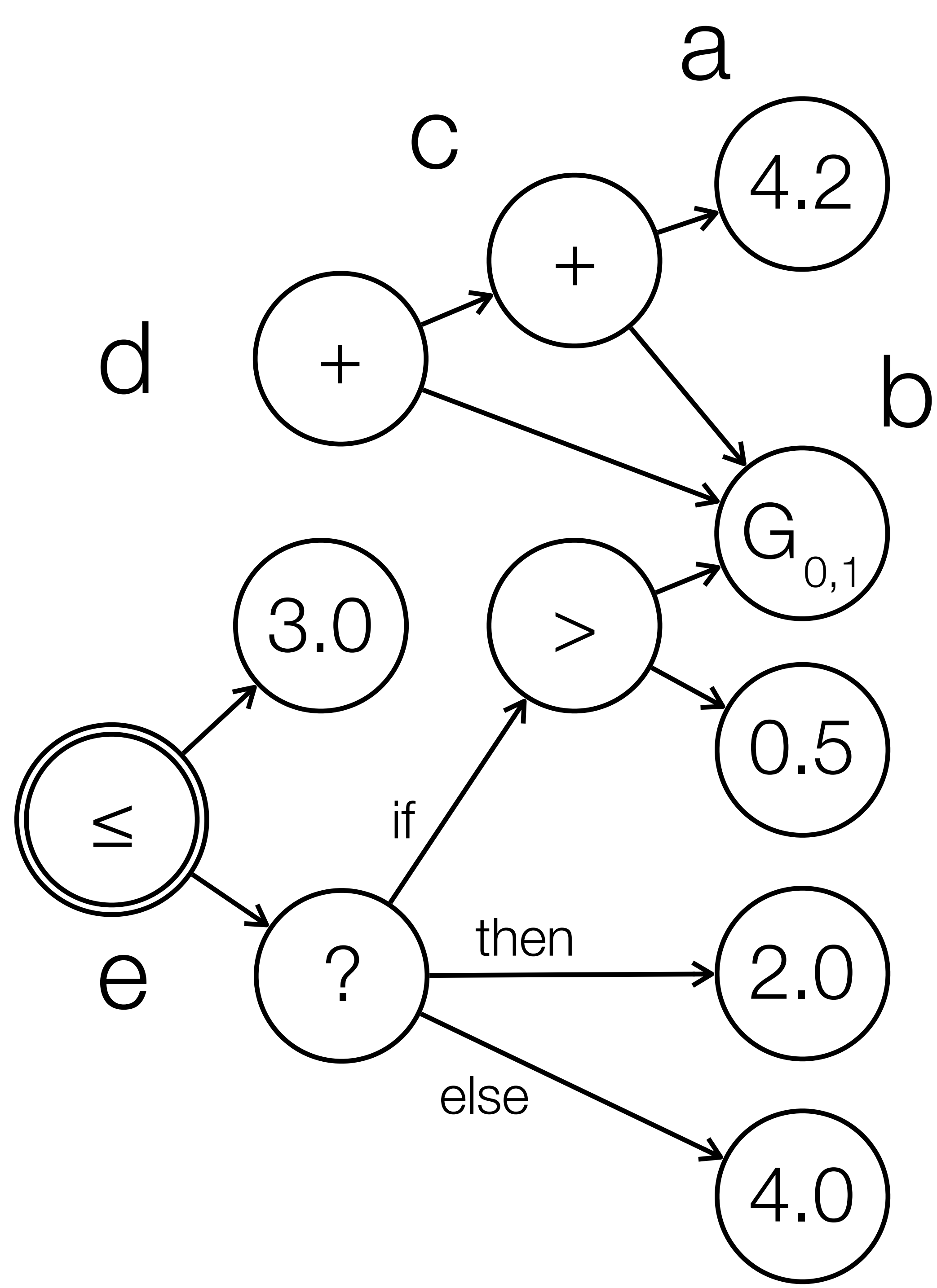
input: `a = 4.2`

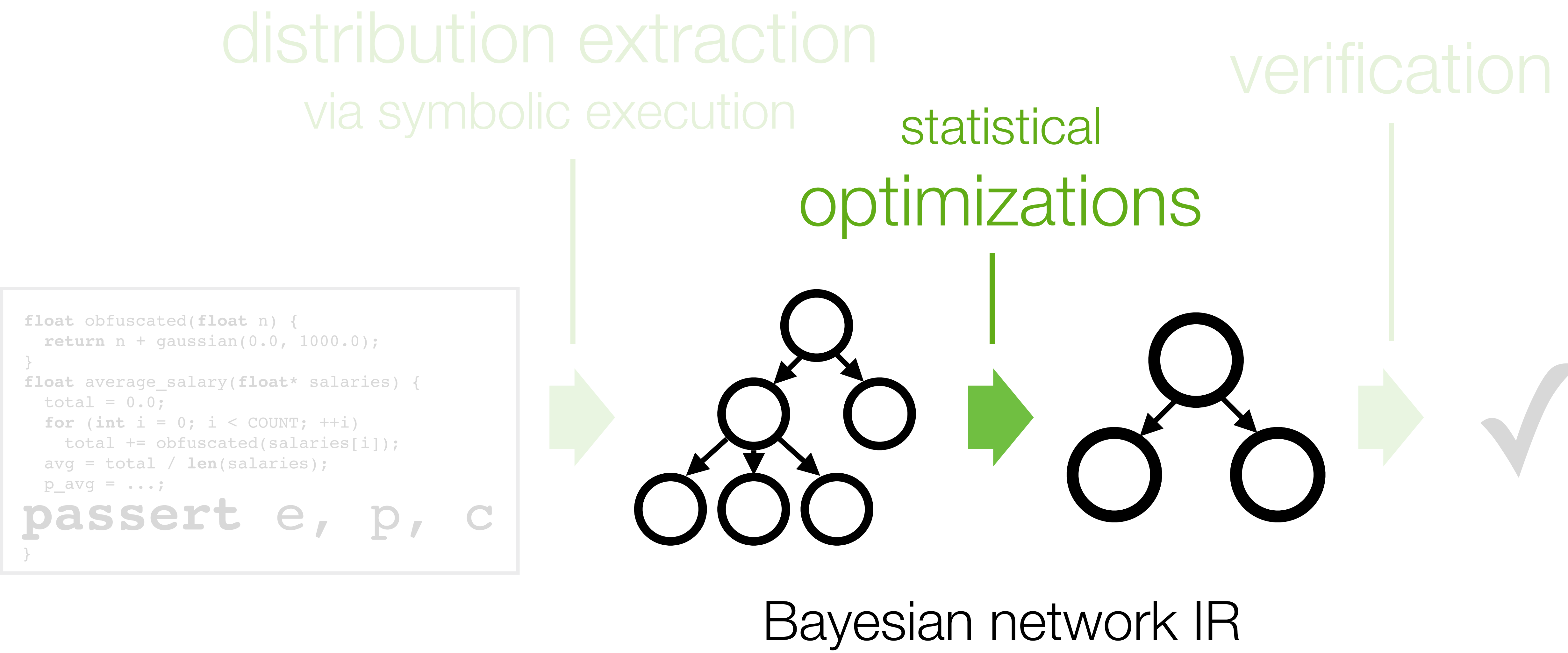
```
b = gaussian(0.0, 1.0)
c = a + b
d = c + b
if b > 0.5
  e = 2.0
else
  e = 4.0
```



input: `a = 4.2`

```
b = gaussian(0.0, 1.0)
c = a + b
d = c + b
if b > 0.5
  e = 2.0
else
  e = 4.0
passert e <= 3.0,
         0.9, 0.9
```




distribution extraction
via symbolic execution
statistical
optimizations
verification
float obfuscated(float n) {
return n + gaussian(0.0, 1000.0);
}
float average_salary(float* salaries) {
total = 0.0;
for (int i = 0; i < COUNT; ++i)
total += obfuscated(salaries[i]);
avg = total / len(salaries);
p_avg = ...;
passert e, p, c
}
Bayesian network IR

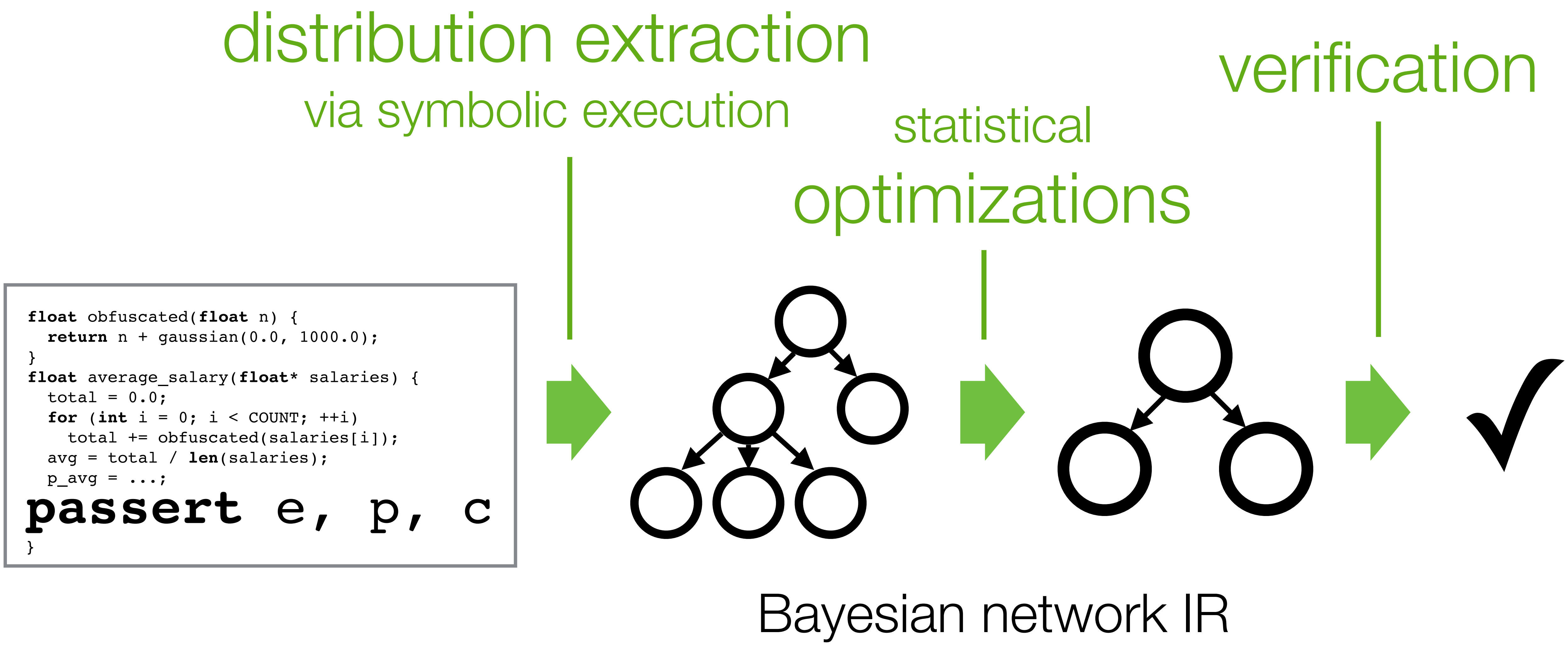
$$X \sim G(\mu_X, \sigma_X^2)$$
$$Y \sim G(\mu_Y, \sigma_Y^2)$$
$$Z = X + Y$$
$$\Rightarrow Z \sim G(\mu_X + \mu_Y, \sigma_X^2 + \sigma_Y^2)$$

$$X \sim U(a, b)$$
$$Y = cX$$
$$\Rightarrow Y \sim U(ca, cb)$$

statistical property → passert verifier optimization

$$X \sim U(a, b)$$
$$Y \sim X \leq c$$
$$a \leq c \leq b$$
$$\Rightarrow Y \sim B\left(\frac{c-a}{b-a}\right)$$

$$X_1, X_2, \ldots, X_n \sim D$$
$$Y = \sum_i X_i$$
$$\Rightarrow Y \sim G(n\mu_D, n\sigma_D^2)$$


distribution extraction
via symbolic execution
statistical
optimizations
verification
float obfuscated(float n) {
  return n + gaussian(0.0, 1000.0);
}
float average_salary(float* salaries) {
  total = 0.0;
  for (int i = 0; i < COUNT; ++i)
    total += obfuscated(salaries[i]);
  avg = total / len(salaries);
  p_avg = ...;
passert e, p, c
}
Bayesian network IR