# CMSC 631

# Program Analysis and Understanding

## Spring 2013

---

# Probabilistic computation

# Estimated Glomular Filtration Rate

```
1   real estimateLogEGFR( real logScr, int age,
2                         bool isFemale, bool isAA){
3     var k,alpha: real;
4     var f: real;
5     f= 4.94;
6     if (isFemale) {
7        k = -0.357;
8        alpha = -0.329;
9     } else {
10       k = -0.105;
11       alpha = -0.411;
12    }
13
14    if ( logScr < k ) {
15       f = alpha * (logScr - k);
16    } else {
17       f = -1.209 * (logScr - k);
18    }
19    f = f - 0.007 *  age;
20
21    if (isFemale)  f = f + 0.017;
22    if (isAA) f = f + 0.148;
23    return f;
24  }
```

# Estimating the possible error

```
1   void compareWithNoise(real logScr, real age,
2                         bool isFemale, bool isAA) {
3     f1 = estimateLogEGFR(logScr, age, isFemale,isAA);
4     logScr = logScr + uniformRandom(-0.1, 0.1);
5     age = age + uniformRandomInt(-1,1);
6     if (flip(0.01))
7       isFemale = not( isFemale );
8     if (flip(0.01))
9       isAA = not( isAA );
10    f2 = estimateLogEGFR(logScr, age, isFemale,isAA);
11    estimateProbability (f1 - f2 <= 0.1);
12    estimateProbability (f2-f1 <= 0.1);
13  }
```

# Bayesian learning by observing

```
let mystery () =
  let aliceDunnit = flip(0.3) in
  let withGun =
    if aliceDunnit then flip(0.03)
    else flip(0.80)
  (aliceDunnit, withGun)

let PipeFoundAtScene () =
  let aliceDunnit, withGun = mystery () in
  observe(withGun = false)
  aliceDunnit, withGun
```

Increases chances Alice dunnit to 69%

# Security applications too!

- Observations add information
  - The amount of information can be quantified according to the change in distributions [Clarkson 2008]

- Can even define security policies aimed to limit leaking information [Mardziel et al, 2011]

# Many programming languages

- Church
- Fun (with Infer.NET)
- IBAL
- Probabilistic Scheme
- BUGS
- HANSEI
- Factorie
- ...

# Probabilistic language

a ::= n | X | a0+a1 | a0-a1 | a0×a1

b ::= bv | a0=a1 | a0≤a1 | ¬b | b0∧b1 | b0∨b1

c ::= skip | X:=a | c0;c1 | if b then c0 else c1 | while b do c

| pif $n_1/n_2$ then c0 else c1

- Same as the language we looked at for the operational semantics, but we add a probabilistic choice operator
  - Read this as: $n_1$ out of $n_2$ executions of the pif will execute c0, the rest c1

# Distributions

- Our semantics for the language uses *distributions*

- Model distributions discretely as functions from values to real numbers
  - $\delta_D : D \longrightarrow \mathbb{R}$ for some $D$
  - With no subscript, we assume the domain is states $\sigma$
  - These are *frequency distributions* (mass not sum to 1)

- Standard operations
  - Sum: $\delta_1 + \delta_2 = \lambda x.\ \delta_1(x) + \delta_2(x)$
  - Scaling: $p \cdot \delta = \lambda x.\ p \cdot \delta(x)$
  - Update: $\delta[d \mapsto r] = \lambda x.\ \text{if } x = d \text{ then } r \text{ else } \delta(x)$

# Semantics, take 1

- Semantics could map states to (discrete) distributions
  - $\langle a, \sigma \rangle \rightarrow \delta_{\mathbb{N}}$
  - $\langle b, \sigma \rangle \rightarrow \delta_{bv}$
  - $\langle c, \sigma \rangle \rightarrow \delta'$
- Probabilistic choice scales distributions

$$\frac{p = n_1/n_2 \quad \langle c_1, \sigma \rangle \rightarrow \delta_1 \quad \langle c_2, \sigma \rangle \rightarrow \delta_2}{\langle \text{pif } n_1/n_2 \text{ then } c_1 \text{ else } c_2, \sigma \rangle \rightarrow p \cdot \delta_1 + (1\text{-}p) \cdot \delta_2}$$

- Many other rules obvious liftings

$$\frac{}{\langle n, \sigma \rangle \rightarrow \lambda x.\ \text{if } x = n \text{ then } 1.0 \text{ else } 0.0} \qquad \frac{}{\langle X, \sigma \rangle \rightarrow \lambda x.\ \text{if } x = \sigma(X) \text{ then } 1.0 \text{ else } 0.0}$$

# But composition rules don't work

- Sequencing

$$\frac{\langle c0, \sigma\rangle \rightarrow \delta \quad \langle c1, ?\rangle \rightarrow ??}{\langle c0; c1, \sigma\rangle \rightarrow ??}$$

- How to compose output distribution with input state?

# Final semantics

- Judgments map distributions to distributions
  - $\langle a, \delta \rangle \to \delta_{\mathbb{N}} \quad \langle b, \delta \rangle \to \delta_{bv} \quad \langle c, \delta \rangle \to \delta'$
- Liftings, probabilistic choice roughly the same

$$\frac{p = n_1/n_2 \quad \langle c_1, \delta \rangle \to \delta_1 \quad \langle c_2, \delta \rangle \to \delta_2}{\langle \text{pif } n_1/n_2 \text{ then } c_1 \text{ else } c_2, \delta \rangle \to p \cdot \delta_1 + (1-p) \cdot \delta_2}$$

$$\frac{}{\langle n, \delta \rangle \to \lambda x.\ \text{if } x = n \text{ then } 1.0 \text{ else } 0.0}$$

- Sequencing

$$\frac{\langle c_0, \delta \rangle \to \delta_1 \quad \langle c_1, \delta_1 \rangle \to \delta_2}{\langle c_0; c_1, \delta \rangle \to \delta_2}$$

# Interesting rules

- Conditionals
  - Define $\delta|b = \lambda\sigma.$ if $\langle b, \sigma\rangle \rightarrow$ true then $\delta(\sigma)$ else 0
    - (Uses standard operational semantics as subroutine)

$$\frac{\langle c_1, \delta|b\rangle \rightarrow \delta_1 \qquad \langle c_2, \delta|\neg b\rangle \rightarrow \delta_2}{\langle \text{if } b \text{ then } c_1 \text{ else } c_2, \delta\rangle \rightarrow \delta_1 + \delta_2}$$

- Assignments to variables
  - $\delta[X \mapsto a] = \lambda\sigma. \sum_{\sigma' \in D(a,\sigma)} \delta(\sigma')$
    - with $D(a,\sigma) = \{ \sigma' \mid \langle a,\sigma'\rangle \rightarrow n \wedge \sigma'[X \mapsto n] = \sigma \}$
    - In sum: we total the probability mass of all states that could have produced n, the final value substituted

# Loops

- Like collecting semantics, looks to find a fixpoint
  - $\lambda x.0$ is $0_{dist}$

$$\frac{\delta|b = 0_{dist}}{\langle \text{while b do c}, \delta\rangle \rightarrow \delta|\neg b}$$

$$\frac{\delta|b \neq 0_{dist} \quad \langle c, \delta|b\rangle \rightarrow \delta_1 \quad \langle \text{while b do c}, \delta_1\rangle \rightarrow \delta_2}{\langle \text{while b do c}, \delta\rangle \rightarrow \delta_2 + \delta|\neg b}$$

# Discussion

- As usual, we might like to statically analyze a probabilistic program
  - We can set up an abstract interpretation; see other slide deck

- We might also like richer programs
  - E.g., involving strings, real numbers, etc.
  - Can extend semantics to include these