Squeeziness

A metric for avoiding fault masking in software testing

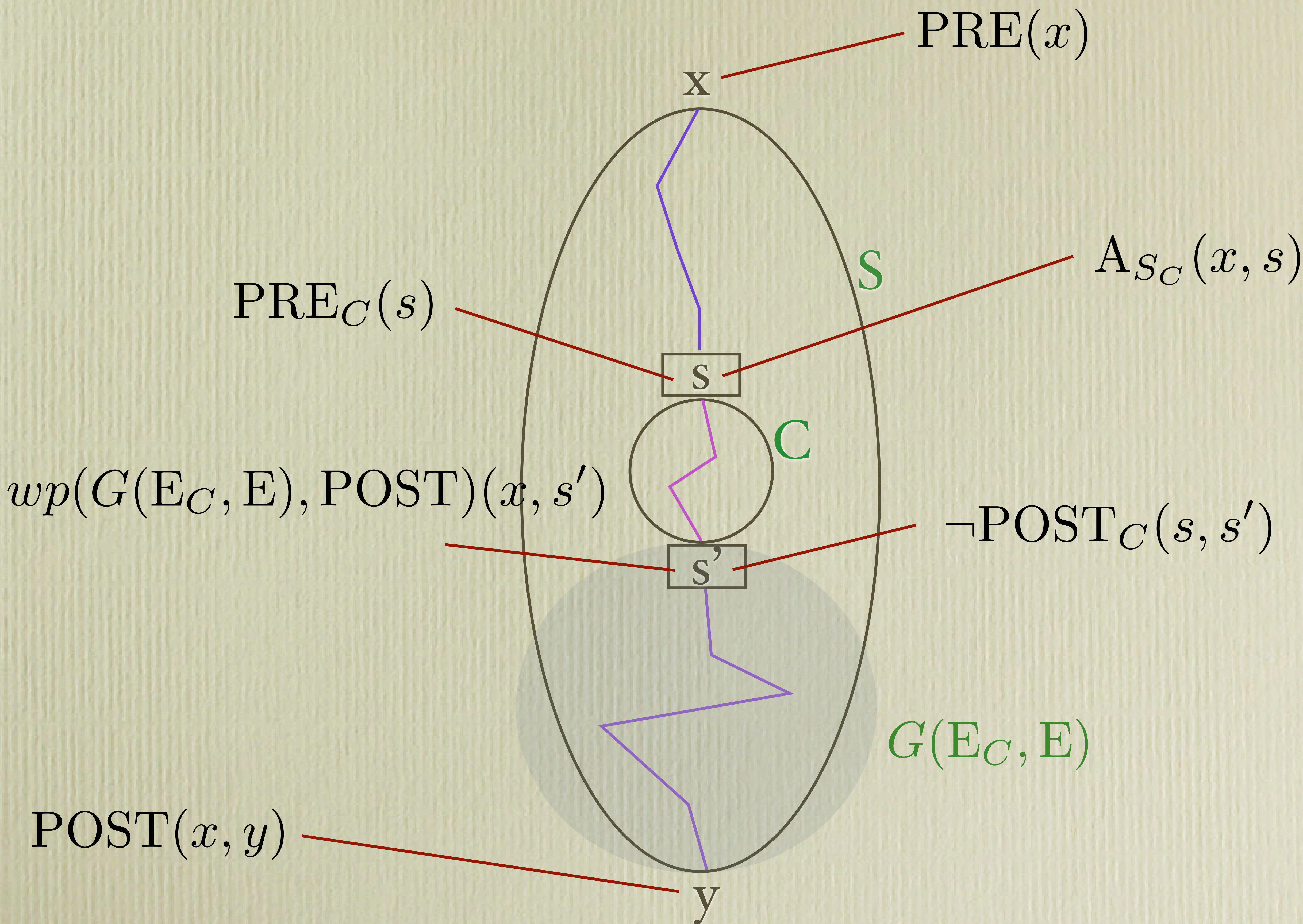
- Joint work with
- **Rob Hierons**
- **Haito Dan**
- **Matt Moroz**
- with help from Sebastian Hunt and Laurie Tratt

software fault masking

- also called **error masking**
- reduces test set effectiveness
- Error masking condition:

$$\exists x, s, s', y \,.\, \mathrm{PRE}(x) \;\wedge\; \mathrm{A}_{S_C}(x,s) \;\wedge\; \mathrm{PRE}_C(s)$$
$$\wedge\, \neg\mathrm{POST}_C(s,s') \;\wedge\; wp(\mathrm{G}(\mathrm{E}_C,\mathrm{E}),\mathrm{POST})(x,s')$$
$$\wedge\, \mathrm{POST}(x,y)$$

Laski et al. ’95


$\mathrm{PRE}(x)$
x
S
$\mathrm{A}_{S_C}(x, s)$
$\mathrm{PRE}_C(s)$
s
C
$wp(G(\mathrm{E}_C, \mathrm{E}), \mathrm{POST})(x, s')$
$\neg\mathrm{POST}_C(s, s')$
s'
$G(\mathrm{E}_C, \mathrm{E})$
$\mathrm{POST}(x, y)$
y

example

Intended

```
x=x+2;
if(x>0)
  x=x%4;
  else x=x;
```

Unintended

```
x=3*x;
if(x>0)
  x=x%4;
  else x=x;
```

"oracle"

output (Intended)

```
t1:x==1
t2:x==-3
```

input

```
t1:x==3
t2:x==-5
```

output (Unintended)

```
t1:x==1
t2:x==-15
```

collisions and state abstraction

- `(x>0)== true;x% 4`: **collisions**
- **also:** oracle may examine only part of the state
- execution path plus oracle identify good and bad states

Domain to Range Ratio

- collisions *necessary*, not *sufficient,* for fault masking
- [Woodward and al-Khanjari (2000)] observed fault masking associated with domain to range ratio
- "loss of information measure" $|D|/|R|$

information theoretic view

Treat the input space and the output space for a program as random variables: I and O

Oracle's Observation of Output

Information in a random variable

$$\mathcal{H}(X) = -\sum_{x \in X} p(x) log_2 p(x)$$

Loss of information from running program **P**

$$\mathcal{H}(I) - \mathcal{H}(O)$$

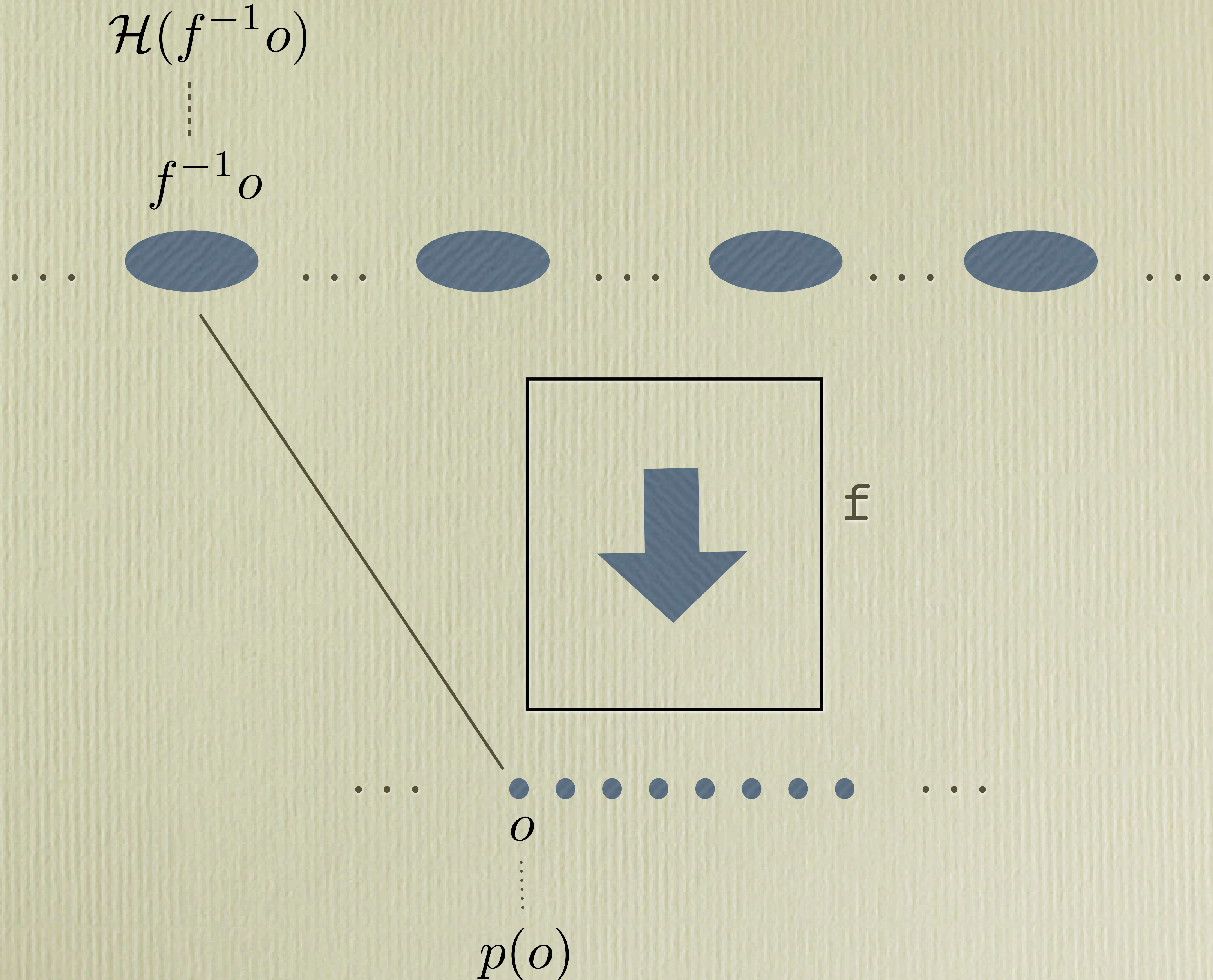
"*Simple!*"

where $[\![P]\!] I = O$

We call this quantity **Squeeziness**.

$$Sq(f) = \mathcal{H}(I) - \mathcal{H}(O) = \sum_{o \in O} p(o)\ \mathcal{H}(f^{-1}o)$$

via the partition property



$\mathcal{H}(f^{-1}o)$
$f^{-1}o$
f
o
$p(o)$

it’s not DRR

- Squeeziness is not a refinement of DRR (and vice versa).
- DRR is a cruder measure than Squeeziness and makes fewer distinctions.
- orderings they produce on (f, I) pairs are inconsistent.

the likelihood of collisions

assume uniform distribution on I

$|f^{-1}o_i|$

$$PColl(f) = \sum_{i=1}^{n} \frac{m_i * (m_i - 1)}{d * (d-1)}$$

$|I|$

Relationship between Squeeziness and PColl not monotonic

Pearson coefficient

Domain size	Max sub	Corr with Sq	Corr with DRR
1.00E+05	200	0.981	0.849
1.00E+05	200	0.986	0.889
1.00E+05	2000	0.981	0.849
1.00E+05	2000	0.986	0.889
1.00E+06	200	0.971	0.748
1.00E+06	200	0.964	0.686
1.00E+07	200	0.968	0.645
1.00E+07	200	0.975	0.606
1.00E+08	200	0.978	0.584
1.00E+08	200	0.975	0.668

what can we do with Squeeziness?

- (1) Measure how much Software Under Test is inclined to fault masking (not so helpful . . .)
- (2) Improve test set selection?

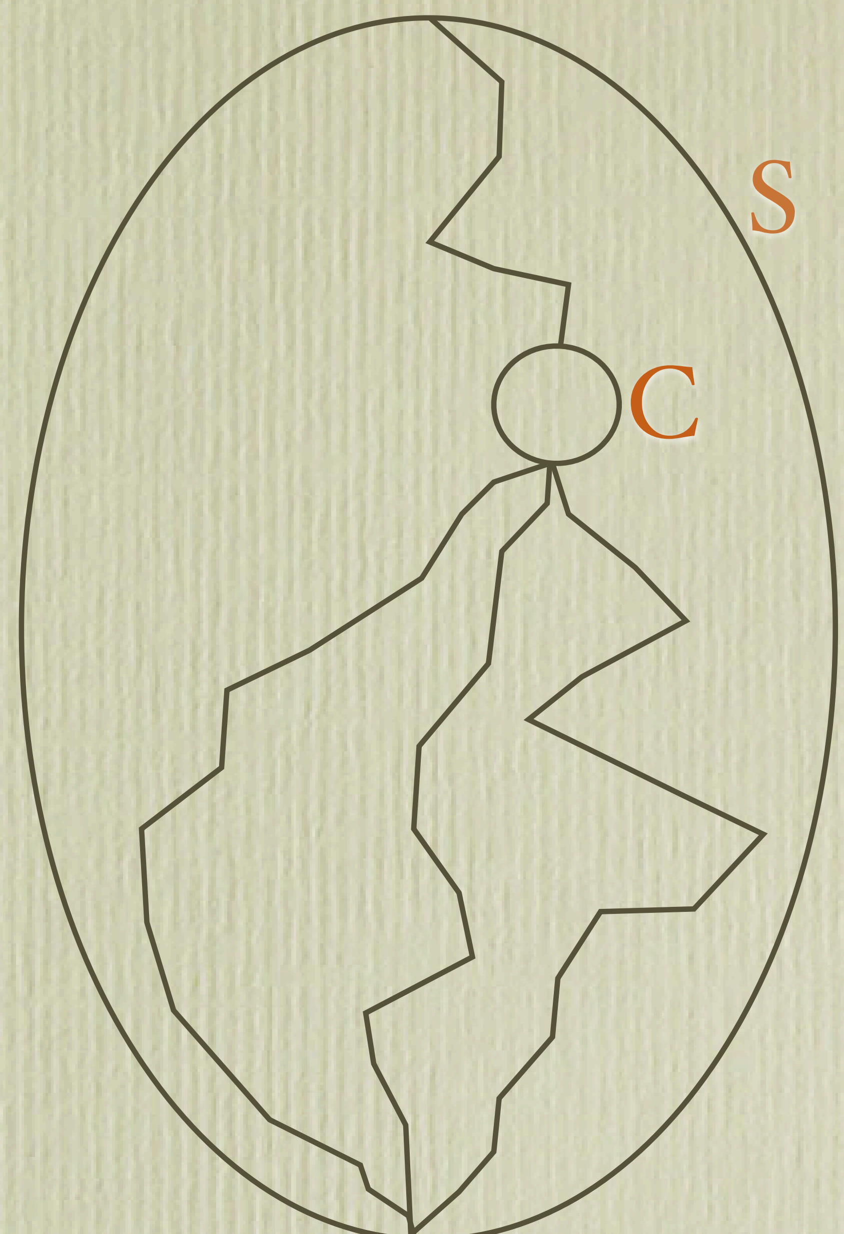
test suite selection

- current “standard” for white box testing is structural coverage: statements, branches, etc.
- limited relationship between coverage and test suite effectiveness, e.g. [Cai and Lyu. A-MOST 2005] plus other papers

Use covering paths
to generate tests

Pick a less
Squeezy path

Reduce possible
fault masking



probability distributions

How can developers know the random variable in inputs?

(1) Maximum Entropy Principle (= Uniform distribution)

$$Sq(f) = \frac{1}{|I|} \sum_{o \in O} |f^{-1}o| \; log_2(|f^{-1}o|)$$

(2) Maximum Squeeziness: $Sq(f) = log_2|f^{-1}o'|$

(3) WesWeimar: estimating path execution frequency statically

current research

- experimental validation of post structural element path selection using a mutation testing approach
- theory of probabilistic testing
- program analyses to estimate Squeeziness
- relationship to mutation testing, SBT
- position paper: Clark and Hierons. Squeeziness: An Information Theoretic Measure for Avoiding Fault Masking. Accepted for publication in Information Processing Letters

Questions?