Quantifying Loop Leakage using a Lattice of Partitions

QA'09

Jonathan Heusser

Pasquale Malacaria

{jonathanh,pm}@dcs.qmul.ac.uk

Introduction

- What: QIF in an algebraic framework
- Why: improve integration with existing IF areas (e.g. declassification) and ease implementation/automation

Introduction

- What: QIF in an algebraic framework
- Why: improve integration with existing IF areas (e.g. declassification) and ease implementation/automation

In this talk, we show

- how to base Information Theory on a lattice of partitions
- how that enables us to reason about loops algebraically
- application: declassification and automation

Programs, Information, Partitions

$$P \equiv \texttt{if(pin==4) ok else ko}$$

Represent program P as partition of high inputs:

$$\{\underbrace{\{4\}}_{\text{ok}}, \underbrace{\{pin \mid pin \neq 4\}}_{\text{ko}}\}$$

Block in a partition = set of indistinguishable states

> Information **released** by program = partition where blocks are set of indistinguishable states according to the observations of the program

(idea been around for a while e.g. G.Lowe)

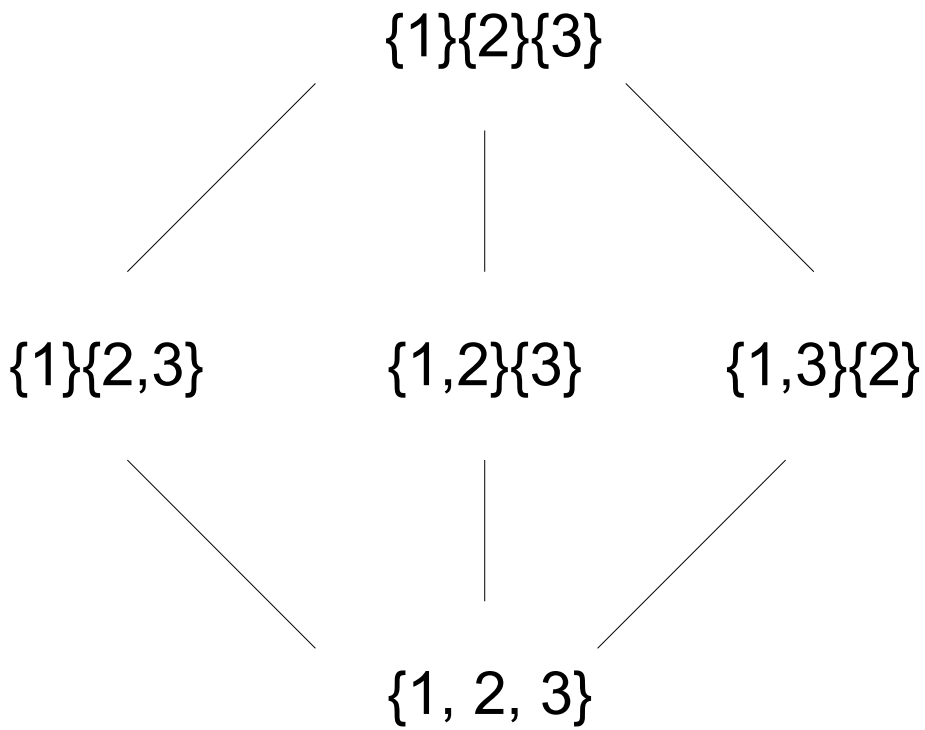
Lattice of Information

- Observations of a system can be seen as partitions: a block = a set of undistinguishable states according to the observation.
- Less informative observation: all states are undistinguishable $\{\{s_1, \ldots, s_n\}\}$
- Most informative observation: all states are distinguishable $\{\{s_1\}, \ldots, \{s_n\}\}$
- Partitions of states form a complete lattice. L.u.b. = intersection of partitions

$\longrightarrow$ Landauer and Redmond

Lattice of Information (LoI)

3 states: 1,2,3 and its lattice



$\top$ is identity relation, $\bot$ is singleton
$\sqcup$ is intersection, $\sqcap$ is (trans. clos.) union

Random Variables as LoI elements

$$X : D \rightarrow R(X)$$

- For an observation $X = x$ the probability is $p(x)$.
- If we observe $X = x$, input to X in D belongs to $X^{-1}(x)$.
- I.e., X partitions the space D into sets of indistinguishable elements to an observer who sees $X = x$.
- Equivalence relation: $d \sim d'$ iff $X(d) = X(d')$.
- Random variable can be seen as a LoI element.

Random Variables as LoI elements

$$X : D \rightarrow R(X)$$

- For an observation $X = x$ the probability is $p(x)$.
- If we observe $X = x$, input to X in D belongs to $X^{-1}(x)$.
- I.e., X partitions the space D into sets of indistinguishable elements to an observer who sees $X = x$.
- Equivalence relation: $d \sim d'$ iff $X(d) = X(d')$.
- Random variable can be seen as a LoI element.

Programs $\sim$ Random variables $\sim$ LoI element

Partitions for Looping programs

```
l=0;
while(l < h) {
  l = l + 1;
}
```

Partitions for Looping programs

```
l=0;
while(l < h) {
  l = l + 1;
}
```

Partition generated by n iteration:
$P_n = \{\{0\}, \{1\}, \ldots, \{n-1\}\} \bigcup \{\{x \mid x \geq n\}\}$

Total information is the l.u.b of this chain in LoI:
$P = \bigsqcup P_n = \{\{s\} \mid 0 \leq s < 2^k\}\}$

Everything about the input is revealed.

Measure on LoI

- Measure on Lattices (valuation: Birkhoff) is given by a monotone map $\nu : L \rightarrow \mathcal{R}$ s.t.
  $\nu(X \sqcup Y) = \nu(X) + \nu(Y) - \nu(X \sqcap Y)$ (incl-excl)
- Consider the weakened monotone map s.t
  $\nu(X \sqcup Y) \leq \nu(X) + \nu(Y) - \nu(X \sqcap Y)$

Measure on LoI

- Measure on Lattices (valuation: Birkhoff) is given by a monotone map $\nu : L \to \mathcal{R}$ s.t. $\nu(X \sqcup Y) = \nu(X) + \nu(Y) - \nu(X \sqcap Y)$ (incl-excl)
- Consider the weakened monotone map s.t $\nu(X \sqcup Y) \leq \nu(X) + \nu(Y) - \nu(X \sqcap Y)$

Theorem (Nakamura 1971): *The only semivaluation on LoI is Shannon Entropy*

In general, no valuation is definable on LoI

Shannon Entropy is the best measure on LoI

Measuring Leakage

Program P from before: $P = \bigsqcup P_n = \{\{s\} \mid 0 \leq s < 2^k\}\}$

Measuring Leakage

Program P from before: $P = \bigsqcup P_n = \{\{s\} \mid 0 \leq s < 2^k\}\}$

Measuring Leakage: $\nu(P) = \nu(\sqcup_{n \geq 0} P_n)$

Measuring Leakage

Program P from before: $P = \bigsqcup P_n = \{\{s\} \mid 0 \leq s < 2^k\}\}$

Measuring Leakage: $\nu(P) = \nu(\sqcup_{n \geq 0} P_n)$

$\rightarrow$ POPL leakage $lim_{n \rightarrow \infty} W(e, M)_n$ is equal to $\nu(\sqcup_{n \geq 0} P_n)$

$$W(e, M)_n = H(\mu(e^{<0>}), \cdots, \mu(e^{<n>})) + \sum_{1 \leq i \leq n} \mu(e^{<i>}) H(M^i | e^{<i>})$$

Measuring Leakage

Program P from before: $P = \bigsqcup P_n = \{\{s\} \mid 0 \leq s < 2^k\}\}$

Measuring Leakage: $\nu(P) = \nu(\sqcup_{n \geq 0} P_n)$

$\rightarrow$ POPL leakage $lim_{n \rightarrow \infty} W(e, M)_n$ is equal to $\nu(\sqcup_{n \geq 0} P_n)$

$$W(e, M)_n = H(\mu(e^{<0>}), \cdots, \mu(e^{<n>})) + \sum_{1 \leq i \leq n} \mu(e^{<i>}) H(M^i | e^{<i>})$$

$\rightarrow$ Leakage of loops = semivaluation of least fixpoint of P_n

Nice algebraic interpretation of previous results.
Potentially simpler for implementation and approximation.

“Convential” Loop Leakage

```
while e M
```

- Loop iterations seen as family of functions $f_0 \oplus \cdots \oplus f_n$.
- f_i corresponds to the loop ending in i iterations
- Disjoint co/domain for deterministic program
- Domain of f_i given by all states σ where $f_j(\sigma)(e)$ is true for $0 \leq j \leq i$ and false otherwise
- We denote this set *event* $e^{<i>}$

“Convential” Loop Leakage

```
while e M
```

- Loop iterations seen as family of functions $f_0 \oplus \cdots \oplus f_n$.
- f_i corresponds to the loop ending in i iterations
- Disjoint co/domain for deterministic program
- Domain of f_i given by all states σ where $f_j(\sigma)(e)$ is true for $0 \leq j \leq i$ and false otherwise
- We denote this set *event* $e^{<i>}$
- Leakage of loop = Entropy of probabilities $e^{<0>}, \ldots, e^{<n>}$ + entropy of body M iterated i times

Properties of Semivaluation

- They often induce metrics: entropy does.
- Metric induced on LoI by entropy:

$$\begin{aligned} d(X,Y) &= H(X,Y) - I(X;Y) \\ &= H(X|Y) + H(Y|X) \end{aligned}$$

  $H(X,Y) - I(X;Y) =$ sum info of X and Y - info shared by X and Y
  $H(X|Y) + H(Y|X) =$ info about X given Y + info about Y given X
- Other metrics possible, if not as elegant.

Application 1: Metric & Declassification

- Use metric d as ingredient to quantitative declassification policy
- Idea of Declassification: *establish what information can be released via a policy*.
- Example policy: `pin==guess` check to be released, translates to information available $\{\{\text{pin}\}, \{\text{all other values}\}\}$

Application 1: Declassification inflexible

Let $h_1, \ldots, h_n$ be secret data. We allow their average $avg = (h_1 + \cdots + h_n)/n$ to be released. What about the following code:

$P' \equiv h_n = h_1; h_{n-1} = h_1; \ldots; h_2 = h_1; avg$

This code leaks all the secret h_1, so declassification would disallow it.

Application 1: Declassification inflexible

Let $h_1, \ldots, h_n$ be secret data. We allow their average $avg = (h_1 + \cdots + h_n)/n$ to be released. What about the following code:

$P' \equiv h_n = h_1; h_{n-1} = h_1; \ldots; h_2 = h_1; avg$
This code leaks all the secret h_1, so declassification would disallow it.

$P'' \equiv h_2 = h_1; avg$
This code leaks a little more than the average wrt to h_1. Declassification still disallows this.

However for large n the leakage is very small ..

Algebraic algorithm - combining QIF and Declassification

1. Take $X =$ the point in LoI associated to the program $P'' \equiv h_2 = h_1; avg$
2. Take $Y =$ the point in LoI associated to the declassification policy avg
3. **If** $X \leq Y$ *accept* P'' **else** compute $\epsilon = d(X, Y)$
4. **If** $\epsilon < threshold$ *accept* P'' **else** *reject* P''

Where $threshold$ defines how flexible your policy should be.

Application 1: Remarks

- $d(X, Y)$ expresses how much information is leaked by allowing X and Y together
- The precise value depends on the probability distribution on the secret, in fact
- *For any* $\epsilon > 0$ *there exists a probability distribution s.t. for all* X *in LoI* $\nu(X) < \epsilon$
  (The more is known in advance the less there is to leak)

Application 2: Automation using (all)SAT

- Given a program, LoI point can be calculated using SAT and allSAT solvers
- Step 1: find *some* input for each output using SAT based fixed point computation, "*find another input which does not lead to the same output*"
- Step 2: for each of those inputs, count # of satisfying models, "*Given that input, how many other inputs lead to the same output*"

Application 2: Automation using (all)SAT

- Given a program, LoI point can be calculated using SAT and allSAT solvers
- Step 1: find *some* input for each output using SAT based fixed point computation, "*find another input which does not lead to the same output*"
- Step 2: for each of those inputs, count # of satisfying models, "*Given that input, how many other inputs lead to the same output*"
- 1. gives number of equivalence classes, 2. gives sizes of equivalence classes, together they are the LoI point
- Quantification through semivaluation

→ similar to program by Backes, Köpf, Rybalchenko

Conclusions

- Nice, unifying algebraic framework
- Leakage of loops are semivaluation of fixpoints
- Metric
- Applications: declassification, and SAT-based automation