*Math. Struct. in Comp. Science* (2015), *vol.* 25, *pp.* 404–428. © Cambridge University Press 2014
doi:10.1017/S0960129513000649 First published online 10 November 2014

# Algebraic foundations for quantitative information flow

PASQUALE MALACARIA

*School of Electronic Engineering and Computer Science,*
*Queen Mary University of London, London, Mile End Road, E1 4NS, United Kingdom*
*Email:* pm@eecs.qmul.ac.uk

*Received 31 December 2010; revised 18 June 2013*

Several mathematical ideas have been investigated for quantitative information flow. Information theory, probability, guessability are the main ideas in most proposals. They aim to quantify *how much information* is leaked, *how likely is to guess* the secret and *how long does it take* to guess the secret respectively. In this work, we investigate the relationship between these ideas in the context of the quantitative analysis of deterministic systems. We propose the lattice of information as a valuable foundation for these approaches; not only it provides an elegant algebraic framework for the ideas, but also to investigate their relationship. In particular, we will use this lattice to prove some results establishing order relation correspondences between the different quantitative approaches. The implications of these results w.r.t. recent work in the community is also investigated. While this work concentrates on the foundational importance of the lattice of information its practical relevance has been recently proven, notably with the quantitative analysis of Linux kernel vulnerabilities. Overall, we believe these works set the case for establishing the lattice of information as one of the main reference structure for quantitative information flow.

## 1. Introduction

Quantitative security analysis should be able to address confidentiality comparison questions like: 'given programs $P$ and $P'$ which one is more of a threat?' These comparison problems is related to the other fundamental question that a quantitative security analysis should be able to address: 'how much of a threat is program $P$?'

Quantitative analyses are based on some measure, usually a real number. This number may answer the comparison problems by reducing it to a numerical comparison and the second question by considering the magnitude of the number in relation to the size of the secret. In many of these measures the number 0 has been shown to characterize completely secure programs.

In recent years, a number of ideas have emerged as reasonable measures for quantitative information flow (abbreviated as QIF): information theory, probabilistic measures and guessability (Clark *et al.* 2007; Köpf and Basin 2007; Smith 2009). Further alternatives based on beliefs (Clarkson *et al.* 2009) have also been proposed. The information theoretical concepts of entropy, conditional entropy and mutual information have been used to answer questions like 'how much information can an attacker gain from observing the system?' whereas probabilities can be used to answer questions like 'how likely is that

https://doi.org/10.1017/S0960129513000649 Published online by Cambridge University Press

the attacker may guess the secret in $n$ tries after observing the system?' and 'what is the average number of guesses needed to guess the secret after the observations?'

There seems to be an intuitive connection between these questions, but the connection is not trivial; in fact some deep differences have been noticed in these approaches (Smith 2009). In the context of QIF the differences seems to relate mainly to the variety of attackers models and of what the scope of modelling should be.

In this work we relate the confidentiality comparison questions in probabilistic, guessability and information theoretical approaches for deterministic systems. We will do this by studying their relation to an algebraic structure: the lattice of information (abbreviated as `LoI`).

The `LoI`(Landauer and Redmond 1993) is the lattice of all equivalence relations on a set; by identifying observations over a system as the equivalence relation equating all (secret) states that cannot be distinguished by those observations we see `LoI` as the mathematical model for all observations generated by all possible deterministic systems over a set of (secret) states.

This allows for an elegant analysis decomposition of QIF into two steps, the first being an *algebraic interpretation*, the second being a *numerical evaluation*:

1. interpret the attacker view of the system as an *equivalence relation* identifying the states indistinguishable by the attacker through the observations,
2. *measure* the above equivalence relation. This measure should provide an indication of the leakage of confidential information (or vulnerability) of the system.

The usefulness of these equivalence relations has been demonstrated in several recent works (Clark *et al.* 2005; Giacobazzi and Mastroeni 2004; Köpf and Basin 2007; Malacaria 2007; Morgan 2009); we aim here to prove some results about their algebraic structure.

Given two systems $S, S'$ and the associated equivalence relations $\simeq_S, \simeq_{S'}$, we will show the following equivalences:

1. $\simeq_{S'}$ refines $\simeq_S$,
2. the entropy of $S$ is always less than the entropy of $S'$,
3. the expected probability of guessing the secret in $n$ tries according to $\simeq_S$ is always less than the expected probability of guessing the secret in $n$ tries according to $\simeq_{S'}$,
4. the expected numbers of guesses needed to guess the secret according to $\simeq_{S'}$ is always less than the expected numbers of guesses needed to guess the secret according to $\simeq_S$.

Moreover, these results are shown to be consistent with classical definitions of QIF based on the adversary gain through observations i.e. the difference in threat before and after observations are made (Yasuoka and Terauchi 2010a). In other terms, given two programs $P, P'$ to determine whether $P'$ refines $P$ (as observational equivalence relations) is the same as to determine whether is always the case that it is more likely to guess the secret using $P'$ instead of $P$. This is also the same as to determine whether the entropy of $P$ is always less than the entropy of $P'$.

These results hence provide a clear connection between the algebraic, probabilistic and information theoretical view of leakage in the context of deterministic programs.

The work also contributes to the foundations of QIF, in particular to the important work by Smith (2009), where the difference between the 'one guess' model and the

https://doi.org/10.1017/S0960129513000649 Published online by Cambridge University Press

information theoretical one were insightfully debated. What Smith noticed was that there exist programs such that, assuming a uniform distribution of the secret, their information theoretical measure is approximately the same but whose vulnerability to a one guess attack is very different. The argument relies on a specific, in this case uniform, distribution. What the above results show is that if we argue about the relative vulnerability to $n$ tries attack of two programs and the argument is not dependent on a specific distribution then their relative vulnerability is determined by their LoI order or equivalently by their entropy order.

While most works on QIF rightly concentrate on fixed distributions, mainly the uniform one, this work concentrates instead on the leakage properties inherent to the source code, i.e. the ones that hold for all possible distributions. The two views complement each other.

The LoI interpretation of programs is not just a pure academic exercise; in fact it has informed works integrating QIF with verification techniques (Backes *et al.* 2009; Heusser and Malacaria 2010; Köpf and Basin 2007) where model checkers and sat-solvers are used to build the equivalence $\simeq_S$ associated to a program. More recently these ideas have inspired the quantitative analysis of leakage of Linux kernel functions (Heusser and Malacaria 2010). These works make use of a basic relation between LoI and information theory, i.e. the fact that $\log(|\simeq_S|)$ is the channel capacity of the system $S$, i.e. the maximum amount that $S$ can leak.

### 1.1. *Plan of the paper*

After basics on LoI and information theory in Sections 2 and 3 will investigate the relationship between the refinement ordering in LoI, expected probability of guessing, expected number of guesses and entropy. Section 4 will relate the results from Section 3 to definitions of leakage from the literature and will prove ordering equivalence results between these different notions of leakage. Section 5 will illustrate the use of l.u.b.s in LoI in modelling issues related to multiple runs attacks and interpreting self-composition. Finally, Section 6 will illustrate the use of chains in LoI to analyse looping programs and will provide a short overview of the Linux kernel analysis.

## 2. Basics

### 2.1. *Observations and the lattice of information*

We see *observations* over a system as some *partial* information on the systems' states, in that an observation reveals some information about the (high) states of the system. Some systems may allow for observations revealing no information (all states are possible according to that system's observations) while other systems may allow for observations revealing complete information on the states of the system.

We make an important determinacy assumption, i.e. that a system's observations form a *partition* on the set of all possible states: a block in this partition is the set of states that are indistinguishable by that observation. This assumption is satisfied for example in the setting of deterministic languages when we take as observations the program outputs because the inverse image of a function form a partition on the domain of the function.

https://doi.org/10.1017/S0960129513000649 Published online by Cambridge University Press

In this work, we will use the terms partition or equivalence relation interchangeably. An equivalence relation can always be seen as the partition whose blocks are the equivalence classes and a partition can always be seen as the equivalence relation defined by two objects are related iff they are in the same block.

### 2.2. *Partitions and equivalence relations as lattice points*

Given a finite set $\Sigma$ the set of all possible equivalence relations over $\Sigma$ is a *complete lattice* (Landauer and Redmond 1993): the lattice of information. Order on equivalence relations is the refinement order.

Formally, let us define the set $\mathtt{LoI}$ as the set of all possible equivalence relations on a set $\Sigma$: we will often refer to elements of $\Sigma$ as *atoms*. For the purposes of this work, we will restrict ourselves to consider finite sets of atoms.

Given $\approx, \sim \in \mathtt{LoI}$ and $\sigma_1, \sigma_2 \in \Sigma$, the ordering of $\mathtt{LoI}$ is defined as

$$\approx \sqsubseteq \sim \leftrightarrow \ \forall \sigma_1, \sigma_2 \ (\sigma_1 \sim \sigma_2 \Rightarrow \sigma_1 \approx \sigma_2). \tag{1}$$

This is a *refinement* order: classes in $\sim$ refine (split) classes in $\approx$. Thus, higher elements in the lattice can distinguish more while lower elements in the lattice can distinguish less states. It easily follows from (1) that $\mathtt{LoI}$ is a complete lattice.

Alternatively, the lattice operations join $\sqcup$ and meet $\sqcap$ are defined as the intersection of relations and the transitive closure of the union of relations respectively.

In terms of partitions, a partition is above another if it is more informative, i.e. each block in the higher partition is included in a block in the partition below.

Here is an example of how these equivalence relations can be used in an information flow setting. Let us assume the set of states $\Sigma$ consists of a tuple $\langle l, h \rangle$ where $l$ is an observable, usually called *low*, variable and $h$ is a confidential variable, usually called *high*. One possible observer can be described by the equivalence relation:

$$\langle l_1, h_1 \rangle \approx \langle l_2, h_2 \rangle \leftrightarrow l_1 = l_2.$$

That is the observer can only distinguish two states whenever they disagree on the low variable part. Clearly, a more powerful attacker is the one who can distinguish any two states from one another, or

$$\langle l_1, h_1 \rangle \sim \langle l_2, h_2 \rangle \leftrightarrow l_1 = l_2 \wedge h_1 = h_2.$$

The $\sim$-observer gains more information than the $\approx$-observer by comparing states, therefore $\approx \sqsubseteq \sim$.

### 2.3. *Lattice of information as a lattice of random variables*

A discrete random variable (noted r.v.) is often defined as a map $X : D \to \mathbb{R}$, where $D$ is a countable set with a probability distribution and the set of reals $\mathbb{R}$ is the range of $X$. For each element $d \in D$, its probability will be denoted $\mu(d)$. For every element $x \in \mathbb{R}$ we write $\mu(X = x)$ (or often in short $\mu(x)$) to mean the probability that $X$ takes on the value $x$, i.e. $\mu(x) \stackrel{def}{=} \sum_{d \in X^{-1}(x)} \mu(d)$. In other words, what we observe by $X = x$ is that the input

https://doi.org/10.1017/S0960129513000649 Published online by Cambridge University Press

to $X$ in $D$ belongs to the set $X^{-1}(x)$. From that perspective, $X$ partitions the space $D$ into sets which are indistinguishable to an observer who sees the value that $X$ takes on, and blocks in the partition are the possible *outcomes* for the random variable.

For the purposes of this work the precise values $X$ can assume do not play any role, it is only $X^{-1}$ we are interested into, hence once the set of atoms of `LoI` are equipped with a probability distribution then we will identify elements of `LoI` as random variables.

Given two r.v. $X, Y$ in `LoI` we define the joint random variable $(X, Y)$ as their least upper bound in `LoI` i.e. $X \sqcup Y$. It is easy to verify that $X \sqcup Y$ is the partition obtained by all possible intersections of blocks of $X$ with blocks of $Y$.

### 2.4. *Basic concepts of information theory*

This section contains a very short review of some basic definitions of information theory: a standard textbook is Cover and Thomas (1991). Given a discrete probability space with probabilities $(\mu(x_i))_{i\in N}$ Shannon's entropy is defined as

$$-\sum_{i\in N} \mu(x_i) \log \mu(x_i). \tag{2}$$

It is usually said that this number measures the average information content of the set of events: if there is an event with probability 1 then the entropy will be 0 and if the distribution is uniform i.e. no event is more likely than any other the entropy is maximal, i.e. $\log |N|$. In the literature the terms information content and uncertainty in this context are often used interchangeably.

The entropy of a r.v. $X$ is just the entropy of its probability distribution i.e.

$$-\sum_{x\in X} \mu(X = x) \log \mu(X = x).$$

Given two random variables $X$ and $Y$, the joint entropy $H(X, Y)$ measures the uncertainty of the joint r.v. $(X, Y)$. It is defined as:

$$-\sum_{x\in X, y\in Y} \mu(X = x, Y = y) \log \mu(X = x, Y = y).$$

Conditional entropy $H(X|Y)$ measures the uncertainty about $X$ given knowledge of $Y$. It is defined as $H(X, Y) - H(Y)$. The higher $H(X|Y)$ is, the lower is the correlation between $X$ and $Y$. It is easy to see that if $X$ is a function of $Y$, then $H(X|Y) = 0$ (there is no uncertainty on $X$ knowing $Y$ if $X$ is a function of $Y$) and if $X$ and $Y$ are independent then $H(X|Y) = H(X)$ (knowledge of $Y$ doesn't change the uncertainty on $X$ if they are independent).

Mutual information $I(X; Y)$ is a measure of how much information $X$ and $Y$ share. It can be defined as

$$I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X).$$

Thus the information shared between $X$ and $Y$ is the information of $X$ (resp. $Y$) from which the information about $X$ given $Y$ has been deduced. This quantity measures the correlation between $X$ and $Y$. For example, $X$ and $Y$ are independent iff $I(X; Y) = 0$.

https://doi.org/10.1017/S0960129513000649 Published online by Cambridge University Press

Mutual information is a measure of binary *interaction*. Mutual and conditional mutual information, a form of ternary interaction will be used to quantify *leakage*. Conditional mutual information measures the correlation between two random variables conditioned on a third random variable; it is defined as

$$I(X;Y|Z) = H(X|Z) - H(X|Y,Z) = H(Y|Z) - H(Y|X,Z).$$

### 2.5. *Measures on the lattice of information*

Suppose we want attempt to quantify the amount of information provided by a point in the `LoI`.

We could for example associate to a partition $P$ the map $|P|$ = 'number of blocks in $P$'; this idea can be traced back to Lowe (2002). This map would be 1 for the least informative partition, its maximal value would be the number of atoms and would be reached by the top partition. It is also true that $A \sqsubseteq B$ implies $|A| \leqslant |B|$ so the measure reflects the order of the lattice. An important property of 'additivity' for measures is the inclusion–exclusion principle: this principle says that things should not be counted twice. In terms of sets, the inclusion–exclusion principle says that the number of elements in a union of sets is the sum of the number of elements of the two sets minus the number of elements in the intersection. The inclusion–exclusion principle is universal e.g. in propositional logic the truth value of $A \vee B$ is given by the truth value of $A$ plus the truth value of $B$ minus the truth value of $A \wedge B$.

In the case of the number of blocks the inclusion–exclusion principle is

$$|A \sqcup B| = |A| + |B| - |A \sqcap B|.$$

Unfortunately this property does not hold. As example, by taking

$$A = \{\{1,2\}\{3,4\}\},\ B = \{\{1,3\}\{2,4\}\}$$

as two partitions, then their join and meet will be

$$A \sqcup B = \{\{1\}\{2\}\{3\}\{4\}\},\ A \sqcap B = \{\{1,3,2,4\}\}.$$

hence, $|A \sqcup B| = 4 \neq 3 = |A| + |B| - |A \sqcap B|$.

Another problem with the map $|\ |$ is that when we consider `LoI` as a lattice of random variables the above measure may end up being too crude; in fact, all probabilities are disregarded[†] by $|\ |$. To address these problems more abstract lattice theoretic notions have been introduced in the literature (Birkhoff 1948).

A valuation on `LoI` is a real valued map $v$ : `LoI`$\to \mathbb{R}$, that satisfies the following properties:

$$v(X \sqcup Y) = v(X) + v(Y) - v(X \sqcap Y) \tag{3}$$

$$X \sqsubseteq Y \quad \text{implies} \quad v(X) \leqslant v(Y). \tag{4}$$

[†] Sections 4.5 will however relate $|\ |$ to information theory and channel capacity.

https://doi.org/10.1017/S0960129513000649 Published online by Cambridge University Press

A join-semivaluation is a weak valuation, i.e. a real valued map satisfying

$$v(X \sqcup Y) \leqslant v(X) + v(Y) - v(X \sqcap Y) \tag{5}$$

$$X \sqsubseteq Y \quad \text{implies} \quad v(X) \leqslant v(Y) \tag{6}$$

for every element $X$ and $Y$ in a lattice (Birkhoff 1948). The property (4) is order-preserving: a higher element in the lattice has a larger valuation than elements below itself. The first property (5) is a weakened inclusion–exclusion principle.

**Proposition 1.** Entropy is a join-semivaluation on LoI by defining

$$v(X \sqcup Y) = H(X, Y). \tag{7}$$

The result is proved in Nakamura (1970).

### 2.6. *Note: entropy as the best measure on* LoI

An important result proved in Nakamura (1970) gives a particular importance to Shannon entropy as a measure on LoI. Nakamura showed that the *only* probability-based join-semivaluation on the LoI is Shannon's entropy. It is easy to show that a valuation itself is not definable on this lattice, thus Shannon's entropy is the best approximation to a probability-based valuation on this lattice.

Nakamura starts by considering a family of function $(f_n)_{n\in N}$ such that $f_n$ is defined on a set of $n$ probabilities $p_1, \ldots, p_n$ and satisfies:

1. $f_n$ is continuous
2. $f_n$ is permutation invariant, i.e. $f_n(p_1, \ldots, p_n) = f_n(p_{\pi(1)}, \ldots, p_{\pi(n)})$ for any permutation $\pi$
3. $f_{n+1}(p_1, \ldots, p_n, 0) = f_n(p_1, \ldots, p_n)$

Such a family $(f_n)_{n\in N}$ induces a function $F$ on partitions with $n$ blocks $X = \{X_1, \ldots, X_n\}$ with block $X_i$ having probability $p_i$:

$$F(X) = f_n(p_1, \ldots, p_n).$$

Suppose now that

1. $F$ is a join-semivaluation on all lattices of partitions
2. If two partitions $X, Y$ are independent (in probability theory sense) then

$$F(X \sqcup Y) = F(X) + F(Y).$$

Nakamura's result is then that such a function $F$ is, up to a constant, Shannon's entropy function, i.e.

$$F(X) = f_n(p_1, \ldots, p_n) = -c \sum_{1\leqslant i\leqslant n} p_i \log(p_i).$$

https://doi.org/10.1017/S0960129513000649 Published online by Cambridge University Press

## 3. Lattice of information, expected probability of guessing, expected number of guesses and entropy

This Section will investigate the relationship in LoI between the refinement ordering, expected probability of guessing, expected number of guesses and entropy.

### 3.1. *Expected probability of guessing*

We define, given an equivalence relation, the average probability of guessing the secret in $n$ tries. This notion generalizes Smith's conditional Bayes vulnerability (Smith 2009).

Given a set $X$ where each element has associated a probability (w.l.g. we assume the probabilities being ordered decreasingly i.e. $\mu(x_i) \geqslant \mu(x_{i+1})$) define the probability of guessing the secret in $n$ tries as the following:

$$g_{n,\mu}(X) = \sum_{1 \leqslant i \leqslant m} \mu(x_i)$$

where $m = \min(|X|, n)$. Given a partition $X$ and a distribution $\mu$ the probability of guessing the secret in $n$ tries is

$$G_{n,\mu}(X) = \sum_{X_i \in X} g_{n,\mu}(X_i).$$

As an example consider the partition

$$\{\{x_1, \ldots, x_4\}\{x_5, x_6\}\}$$

where the first four atoms have probability $\frac{1}{16}$ each and $x_5, x_6$ have probability $\frac{3}{8}$ each.

Then the average probability of guessing the secret in 2 tries is $\frac{1}{8} + \frac{3}{4} = \frac{7}{8}$; indeed after the observations and two tries the probability of non-guessing the secret is $\frac{1}{8}$ corresponding to not having exhausted all possibilities from the first block.

Notice that the above definition is the same as having a probability distribution on each block, computing the probability of guessing the secret in each block and then taking the weighted average:

$$G_{n,\mu}(X) = \sum_{X_i \in X} g_{n,\mu}(X_i) = \sum_{X_i \in X} \mu(X_i) \sum_{1 \leqslant j \leqslant n, x_j \in X_i} \frac{\mu(x_j)}{\mu(X_i)}.$$

When clear from the context we will omit the subscript $\mu$ from $G$ and $g$.

**Proposition 2.**

$$X \sqsubseteq Y \Leftrightarrow \forall \mu, n.\ G_{n,\mu}(X) \leqslant G_{n,\mu}(Y)$$

*Proof.* Step 1:

$$X \sqsubseteq Y \Rightarrow \forall \mu, n.\ G_{n,\mu}(X) \leqslant G_{n,\mu}(Y)$$

w.l.g. it will be enough to consider a block $X_i$ in $X$ splitting into two blocks $Y_i, Y_j$ in $Y$; we then need to prove that

$$g_n(X_i) \leqslant g_n(Y_i) + g_n(Y_j).$$

https://doi.org/10.1017/S0960129513000649 Published online by Cambridge University Press

We can write $g_n(X_i) = \sum_{i \leqslant I} \mu(x_i) + \sum_{j \leqslant J} \mu(x_j)$ where the $x_i \in Y_i, x_j \in Y_j, I = |Y_i|, J = |Y_j|$. We can then write $g_n(Y_i)$ as $\sum_{i \leqslant I} \mu(x_i) + c_i$ where $c_i \geqslant 0$ is the sum of the elements in the sum $g_n(Y_i)$ which are not in the sum $g_n(X_i)$ and similarly $g_n(Y_j)$ can be written as $\sum_{j \leqslant J} \mu(x_j) + c_j$.

We have hence,

$$\begin{aligned} g_n(Y_i) + g_n(Y_j) &= \sum_{i \leqslant I} \mu(x_i) + c_i + \sum_{j \leqslant J} \mu(x_j) + c_j \\ &\geqslant \sum_{i \leqslant I} \mu(x_i) + \sum_{j \leqslant J} \mu(x_j) \\ &= g_n(X_i). \end{aligned}$$

Step 2:

$$(\forall \mu, n.\ \ G_n(X) \leqslant G_n(Y)) \Rightarrow X \sqsubseteq Y.$$

Suppose $X \not\sqsubseteq Y$, w.l.g. we can then find a block $Y_i \in Y$ included in two (or more) blocks in $X$; take then a distribution 0 everywhere apart from the elements in $Y_i$ and apply the previous reasoning (the inclusion here can be seen as a splitting because of the 0 probabilities), i.e. for $X_i, X_j$ splitting $Y_i$, $g_n(Y_i) < g_n(X_i) + g_n(X_j)$ (e.g. for $n = 1$), hence for this distribution $G_n(X) \not\leqslant G_n(Y)$. □

As an example consider the partitions

$$X = \{\{1,2\}\{3,4\}\}, \quad Y = \{\{1,3\}\{2,4\}\}.$$

$X$ and $Y$ are not order related because no block in $X$ is refined by a block in $Y$ and vice versa; hence following the proposition we can find distributions and number of guesses ordering them in any order: for $G(Y) < G(X)$ choose $n = 1$ and the distribution s.t. $\mu(1) = \mu(3) = \frac{1}{2}$ and is 0 elsewhere. Then

$$G_1(Y) = g_1(\{1,3\}) = \frac{1}{2} < \frac{1}{2} + \frac{1}{2} = g_1(\{1,2\}) + g_1(\{3,4\}) = G_1(X).$$

Likewise for $G(X) < G(Y)$ choose the distribution s.t. $\mu(1) = \mu(2) = \frac{1}{2}$ and is 0 elsewhere.

**Corollary 1.**

$$X \sqsubseteq Y \Leftrightarrow \forall \mu\ G_{1,\mu}(X) \leqslant G_{1,\mu}(Y)$$

### 3.2. *Expected number of guesses*

The expected probability of guessing should be related to the expected number of guesses.

Given a set $X$ where each element has associated a probability (we again assume the probabilities being ordered decreasingly) define the expected number of guesses as

$$ng_\mu(X) = \sum_{1 \leqslant i \leqslant n} i\mu(x_i).$$

https://doi.org/10.1017/S0960129513000649 Published online by Cambridge University Press

Given a partition $X$ and a distribution $\mu$ the expected number of guesses is (we abuse the notation):

$$\mathrm{NG}_{\mu}(X) = \sum_{X_i \in X} ng_{\mu}(X_i).$$

Intuitively the more is known of the secret the less guesses are needed, hence we should expect the NG order to reverse the `LoI` order; consider for example the set $\{a, b, c, d\}$ with probabilities $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}$ respectively; we have then

$$\mathrm{NG}\,(\{\{a, b, c, d\}\}) = \frac{15}{8} > \frac{10}{8} = \mathrm{NG}\,(\{\{a, d\}\{b, c\}\}).$$

We can now show that `LoI` order is the dual of the NG order:

**Proposition 3.**

$$X \sqsubseteq Y \Leftrightarrow \forall \mu, \quad \mathrm{NG}_{\mu}(Y) \leqslant \mathrm{NG}_{\mu}(X).$$

*Proof.*

$$X \sqsubseteq Y \Rightarrow \forall \mu, \quad \mathrm{NG}_{\mu}(Y) \leqslant \mathrm{NG}_{\mu}(X)$$

w.l.g. it will be enough to consider a block $X_i$ in $X$ splitting into two blocks $Y_i, Y_j$ in $Y$; consider any element $x \in X_i$; this element will appear as a term $j\mu(x)$ in the sum $ng_{\mu}(X_i)$. As the elements of $X_i$ are split in the two sets $Y_i, Y_j$ then the same $x$ will appear in $ng(Y_i)$ or in $ng(Y_j)$: in any case it will appear as a term $j'\mu(x)$ where $j' \leqslant j$ because $X_i$ is split in the two sets $Y_i, Y_j$ so the relative order of $x$ in $Y_i$ or $Y_j$ has to be less than the relative order of $x$ in $X_i$. Hence $\forall \mu, \quad \mathrm{NG}_{\mu}(Y) \leqslant \mathrm{NG}_{\mu}(X)$.

$$X \sqsubseteq Y \Leftarrow \forall \mu, \quad \mathrm{NG}_{\mu}(Y) \leqslant \mathrm{NG}_{\mu}(X).$$

Suppose $X \not\sqsubseteq Y$, w.l.g. we can then find a block $Y_i \in Y$ included into two (or more) blocks in $X$. Take then the distribution uniform on the elements in $Y_i$ and 0 everywhere else and apply the above reasoning: i.e. for $X_i, X_j$ splitting $Y_i$ any element $y \in Y_i$ (can be made to) appear in an earlier position in $X_i$ or $X_j$, hence $ng(Y_i) > ng(X_i) + ng(X_j)$, and so $\mathrm{NG}\,(Y) \not\leqslant \mathrm{NG}\,(X)$. □

### 3.3. *Entropy and* `LoI`

We now consider entropy; again we can relate entropy to the order in `LoI`. Two partitions are order related if and only if they are entropy related (in the same direction) for all possible distributions.

**Proposition 4.**

$$X \sqsubseteq Y \Leftrightarrow \forall \mu, \; H_{\mu}(X) \leqslant H_{\mu}(Y).$$

*Proof.* Step 1:

$$X \sqsubseteq Y \Rightarrow \forall \mu, \; H_{\mu}(X) \leqslant H_{\mu}(Y).$$

This is a well-known property of entropy (Cover and Thomas 1991): grouping probabilities reduces entropy and it is a consequence of the Jensen inequality.

https://doi.org/10.1017/S0960129513000649 Published online by Cambridge University Press

Step 2:

$$X \sqsubseteq Y \Leftarrow \forall \mu,\ H_\mu(X) \leqslant H_\mu(Y).$$

Suppose $X \not\sqsubseteq Y$, w.l.g. we can then find a block $Y_i \in Y$ included in two (or more) blocks in $X$ (say $X_1 \ldots X_n$); We then take a distribution 0 everywhere apart from the elements in $Y_i$; notice that for such a distribution $\mu(Y_i) = 1$ whereas in $X$ there are two or more blocks with non-zero probability: we have hence

$$H(X) = -\sum_{1 \leqslant i \leqslant n} \mu(X_i) \log(\mu(X_i)) > 0 = -\mu(Y_i) \log(\mu(Y_i)) = H(Y).$$

□

**Note:** the directions ($\Leftarrow$) of the proofs in these sections used distributions 0 everywhere apart some blocks. The arguments work if we were replacing 0 with very small probabilities. Consider for example $A = \{\{a,b\}\{c,d\}\{e\}\}$, $B = \{\{a,d\}\{b,c,e\}\}$. It is easy to see that if all atoms have equal probability $\frac{1}{5}$ then $B < A$ using $G, H$ and $A < B$ using NG. On the other hand using probabilities $a = 0.1, b = 0.1, c = 0.39, d = 0.4, e = 0.01$ then the opposite holds: $H(A) = 0.7994 < 1 = H(B)$, $G_1(A) = 0.51 < 0.79 = G_1(B)$, NG $(A) = 1.49 > 1.22 =$ NG $(B)$.

### 3.4. *Shannon's order of information*

A lattice of random variables was introduced in a little known note by Shannon (1953). His motivations was to characterize information, not just measuring it. As an example consider the processes 'flipping a fair coin' and 'presidential election between two candidates with equal chances': these two processes may well have the same entropy, but it is not the case that knowing one of the two gives information about the other one, so $H(X|Y) > 0$ for $X, Y$ being one of 'flipping a coin' or 'presidential election'.

Given random variables $X, Y$ Shannon's order is defined by

$$X \leqslant_d Y \Leftrightarrow H(X|Y) = 0.$$

The intuition here is that $Y$ provides complete information about $X$, i.e. $X$ is an abstraction of $Y$ (some information is forgotten).

Shannon also defined the related distance function:

$$d(X,Y) = H(X|Y) + H(Y|X).$$

The function $d$ and the relation $\leqslant_d$ are related as follows:

$$d(X,Y) = 0 \Leftrightarrow X \leqslant_d Y \wedge Y \leqslant_d X.$$

In fact suppose $d(X,Y) = 0$; then $H(X|Y) + H(Y|X) = 0$ so as conditional entropy is non-negative $X \leqslant_d Y \wedge Y \leqslant_d X$. On the other hand $X \leqslant_d Y \wedge Y \leqslant_d X$ implies $H(X|Y) = 0, H(Y|X) = 0$ so $d(X,Y) = 0$.

The equivalence classes of the order $\leqslant_d$, i.e. points s.t. $X \leqslant_d Y \wedge Y \leqslant_d X$ or equivalently the sets of points of distance 0, are the information theoretical characterization of

https://doi.org/10.1017/S0960129513000649 Published online by Cambridge University Press

information: all items in a class can be seen as objects having *the same* information, not just sharing the same measure.

Shannon's order and LoI order are the same.

**Proposition 5.**

$$X \sqsubseteq Y \Leftrightarrow \forall \mu.\ X \leqslant_d Y.$$

*Proof.* Direction $X \sqsubseteq Y \Rightarrow \forall \mu. X \leqslant_d Y$.

By definition of join in a lattice

$$X \sqsubseteq Y \Leftrightarrow X \sqcup Y = Y$$

hence for $X \sqsubseteq Y$ we have

$$H(X, Y) = H(X \sqcup Y) = H(Y)$$

and so

$$H(X|Y) = H(X, Y) - H(Y) = H(Y) - H(Y) = 0$$

which proves $\forall \mu.\ X \leqslant_d Y$.

For the other direction assuming $X \not\sqsubseteq Y$ then $X \sqsubset X \sqcup Y$ so we can find a distribution s.t. $H(X \sqcup Y) > H(X)$ and so

$$H(Y|X) = H(X \sqcup Y) - H(X) > H(X) - H(X) = 0$$

and we conclude $X \not\leqslant_d Y$. □

Shannon also noticed that $d$ defines a pseudometric and so the quotient space by the equivalence classes of points of distance 0 is a metric space.

## 4. Measuring leakage of programs

We now connect LoI and leakage of confidential information in programs.

### 4.1. LoI *interpretation of programs and basic properties*

Unless otherwise specified the term *program* will designate a program in a programming language whose denotational semantics is an input-output map: a simple example is the **while** programming language (Winskel 1993), that is a simple imperative language with assignments, sequencing, conditionals and loops. Syntax and semantics for the language are standard. The expressions of the language are arithmetic expression, with constants $0, 1, \ldots$ and boolean expressions with constants $\mathtt{tt}, \mathtt{ff}$. We will assume for the time being that the low variables are initialized in the code; we will discuss this assumption in Section 5.

4.1.1. *Observations over programs.* Observations over a program $P$ form an equivalence relation on the states of $P$. A particular equivalence class will be called an observable. Hence, an observable is a set of states indistinguishable by an attacker making that observation.

https://doi.org/10.1017/S0960129513000649 Published online by Cambridge University Press

This is a wide encompassing definition and general observational models e.g. equivalence relations on program traces have been studied (Malacaria and Chen 2008).

In this work, we will concentrate on a specific class of observations: the output observations (Clark *et al.* 2002; Malacaria 2007). For these observations the random variable associated to a program $P$ is the equivalence relation on any two states $\sigma, \sigma'$ from the universe of states $\Sigma$ defined by

$$\sigma \simeq \sigma' \iff [\![P]\!](\sigma) = [\![P]\!](\sigma') \tag{8}$$

where $[\![P]\!]$ represents the denotational semantics of $P$. Hence, the equivalence relation amounts to 'having the same observable output". We denote the interpretation of a program $P$ in LoI as defined by the equivalence relation (8) by $\mathtt{LoI}(P)$.

The equivalence relation $\mathtt{LoI}(P)$ is hence nothing else than the set-theoretical kernel of the denotational semantic of $P$. Assuming that the set of confidential inputs $h$ is equipped with a probability distribution $\mu$ we can see $\mathtt{LoI}_\mu(P)$ as a random variable. We will write simply $\mathtt{LoI}(P)$ unless we need to specify a specific distribution $\mu$.

**Note:** as $\mathtt{LoI}(-)$ is the set theoretical kernel of the denotational semantics of programs then the LoI interpretation is maximally expressive wrt output observations, i.e. it contains all information included in the denotational semantics of the program.

As a concrete example, let $P$ be the program

```
if (h==0) then x=0; else x=1;
```

where the variable h ranges over $\{0, 1, 2, 3\}$.

The partition $\mathtt{LoI}(P)$ associated to the above program is then

$$\mathtt{LoI}(P) = \{ \underbrace{\{0\}}_{\mathtt{x=0}} \underbrace{\{1,2,3\}}_{\mathtt{x=1}} \}$$

$\mathtt{LoI}(P)$ effectively partitions the domain of the variable $h$, where each disjoint subset represents an output. The partition reflects the idea of what an attacker can learn of secret inputs by *backwards* analysis of the program, from the outputs to the inputs.

The quantitative evaluation of the partition $\mathtt{LoI}(P)$ (Backes *et al.* 2009; Heusser and Malacaria 2009) measures such knowledge gains of an attacker, solely depending on the partition of states and the probability distribution of the input.

### 4.2. *Definition of leakage*

Let us start from the following intuition (Clark *et al.* 2002):

> The leakage of confidential information of a program is defined as the difference between an attacker's uncertainty about the secret before and after available observations about the program.

For a Shannon-based measure, the above intuition can be expressed in terms of mutual information. In fact, if we start by observing that the attacker uncertainty about the secret before observations is $H(h)$ and the attacker uncertainty about the secret after observations is $H(h|\mathtt{LoI}(P))$ then using the definition of mutual information we define leakage as

$$H(h) - H(h|\mathtt{LoI}(P)) = I(h; \mathtt{LoI}(P)).$$

https://doi.org/10.1017/S0960129513000649 Published online by Cambridge University Press

We can now simplify the above definition as follows:

$$\begin{aligned} I(\mathtt{LoI}(P);h) &= H(\mathtt{LoI}(P)) - H(\mathtt{LoI}(P)|h) \\ &=_A H(\mathtt{LoI}(P)) - 0 \\ &= H(\mathtt{LoI}(P)) \end{aligned} \tag{9}$$

where the equality $A$ holds because the program is deterministic. Thus, for such programs

**Leakage:** (Shannon-based) leakage of a program $P$ is defined as the (Shannon) entropy of the partition $\mathtt{LoI}(P)$.

In the light of the results from Section 3 in the context of programs, we deduce a correspondence between the refinement order of the observations, leakage, expected probability of guessing and expected number of guesses (we will return to this point in Theorem 1).

**Note:** in the more general case, where the restriction on the low inputs to be initialized in the code is lifted, leakage is defined in terms of conditional mutual information between the secret $h$ and the output of the program $\mathtt{LoI}(P)$ conditioned to the low input $l$ :

$$I(h;\mathtt{LoI}(P)|l) = H(h|l) - H(h|\mathtt{LoI}(P),l) = H(\mathtt{LoI}(P)|l).$$

### 4.3. *Relation with Yasuoka and Terauchi ordering results*

The results from Section 3 are related to some recent work by Yasuoka and Terauchi (2010a); they define quantitative analysis in terms of Shannon entropy, Smith's vulnerability and average number of guesses.

Their definitions follow the pattern we discussed before:

The quantitative analysis of confidential information of a program is defined as the difference between an attacker's *capability* before and after available observations about the program.

By replacing the word 'capability' with: *(A) uncertainty about the secret, (B) probability of guessing the secret in one try, (C) expected number of guesses* we derive different quantitative analysis. Once formalized (A)(B)(C) as a function $F$ (and its conditional counterpart $F(-|-)$ ) on a probability space all these definitions will have the form:

$$F(h) - F(h|\mathtt{LoI}(P)).$$

Formally, the choices for $F, F(-|-)$ are:

A. for *uncertainty about the secret:* $F$ and $F(-|-)$ are Shannon entropy and conditional entropy
B. for *probability of guessing in one try:* (noted ME)

$$F(X) = -\log(\max_{x\in X} \mu(X = x)) \text{ and } F(X|Y) = -\log\left(\sum_{y\in Y} \mu(y)(\max_{x\in X} \mu(X = x|Y = y))\right)$$

C. for *the expected number of guesses:* (noted GE)

$$F(X) = \sum_{x_i\in X, i\geqslant 1} i\ \mu(X = x_i) \text{ and } F(X|Y) = \sum_{y\in Y} \mu(y)\left(\sum_{x_i\in X, i\geqslant 1} i\mu(X = x_i|Y = y)\right)$$

https://doi.org/10.1017/S0960129513000649 Published online by Cambridge University Press

Table 1. *Comparing measures.*

| | H | G | NG | ME | GE |
|---|---|---|---|---|---|
| $M_1$ | 0.8112 | 0.5 | 1.75 | 1 | 0.75 |
| $M_2$ | 2 | 1 | 1 | 2 | 1.5 |

(here we assume $i < j$ implies $\mu(X = x_i) \geqslant \mu(X = x_j)$).

Notice that is only for $F$ = Shannon's entropy that

$$F(h) - F(h|\mathtt{LoI}(P)) = F(\mathtt{LoI}(P))$$

i.e. only for Shannon's entropy the difference in capability is a 'measure' on LoI.

We now want to relate results from Section 3 with ME and GE definitions of leakage.

To appreciate the difference in the definitions let us consider the examples from Yasuoka and Terauchi (2010a); we consider the following programs:

1. $M_1 \equiv$ `if(h == 1) then o = 0;else o = 1;`
2. $M_2 \equiv$ `o = h`.

Table 1 shows the results of analyses of these programs for a 2 bits secret uniformly distributed. Columns H, G, NG corresponds to our definitions for Shannon entropy, the expected probability of guessing (in 1 guess) and the expected number of guesses on $\mathtt{LoI}(P)$, i.e. H, G, NG stands for $H(\mathtt{LoI}(P)), G(\mathtt{LoI}(P)), \mathrm{NG}(\mathtt{LoI}(P))$. ME and GE are the definitions given above.

The numbers in Table 1 can be connected in a uniform narrative. Take program $M_1$ : $G = 0.5$ means that after running the program an attacker has probability 0.5 of guessing the secret in one try. The chances of guessing the secret have doubled from 0.25 (= $G(h)$) before the program to 0.5 (= $G(\mathtt{LoI}(M_1))$) after the program, so the rate of increase is $2^{\mathrm{ME}(M_1)} = \frac{G(\mathtt{LoI}(M_1))}{G(h)} = 2^1$; the average number of questions needed (initially $\mathrm{NG}(h) = 2.5$) has been reduced by 0.75 (= $\mathrm{GE}(M_1)$) so that it will take now on average to guess it $\mathrm{NG}(\mathtt{LoI}(M_1)) = 1.75$ tries. The observations provide $H(\mathtt{LoI}(M_1)) = 0.8112$ bits of information about the secret.

Consider now the second row, i.e. program $M_2$: here $H = 2$ means that everything is leaked, i.e. the observations provide 2 bits of information about the secret. In this case the secret will be guessed in one try ($G(\mathtt{LoI}(M_2)) = \mathrm{NG}(\mathtt{LoI}(M_2)) = 1$) and the chances have hence increased 4 folds from the initial probabilities ($2^{\mathrm{ME}(M_2)} = \frac{G(\mathtt{LoI}(M_2))}{G(h))} = \frac{1}{0.25} = 2^2$); the average number of questions needed (initially $\mathrm{NG}(h) = 2.5$) has been reduced by 1.5 (= $\mathrm{GE}(M_2)$) to one ($\mathrm{NG}(\mathtt{LoI}(M_2)) = 1$).

The narrative can be strengthened formally: the following result clarifies how ME, NG relate to LoI; in the following we set $h$ in $G(h), \mathrm{NG}(h)$ to be the bottom partition in LoI (i.e. the equivalence relation relating all values of $h$).

**Proposition 6.** For a program $P$ :

1. $\forall \mu.\ \mathrm{ME}(P) = \log(G(\mathtt{LoI}(P))) - \log(G(h))$
2. $\forall \mu.\ \mathrm{GE}(P) = \mathrm{NG}(h) - \mathrm{NG}(\mathtt{LoI}(P))$.

https://doi.org/10.1017/S0960129513000649 Published online by Cambridge University Press

*Proof.*

1. We start by recalling Smith's definition of vulnerability:

$$\mathrm{ME}(P) = \log \frac{1}{\max_h \mu(h)} - \log \frac{1}{\sum_{o \in \mathtt{LoI}(P)} \max_h \mu(h|o)}.$$

We have then (we use $o$ for a block in the partition $\mathtt{LoI}(P)$): $\forall \mu$

$$\begin{aligned}\mathrm{ME}(P) &= \log(\sum_{o \in \mathtt{LoI}(P)} \max_h \mu(h|o)) - \log(\max_h \mu(h)) \\ &= \log(\sum_{o \in \mathtt{LoI}(P)} \max\{\mu(h) | h \in o\}) - \log(\max_h \mu(h)) \\ &= \log(G(\mathtt{LoI}(P))) - \log(G(h)).\end{aligned}$$

2. We can rewrite the definition of $\mathrm{GE}(P)$ from Yasuoka and Terauchi (2010a) as:

$$\mathrm{GE}(P) = \sum_{1 \leqslant i \leqslant n} i\mu(h_i) - \sum_{o \in \mathtt{LoI}(P)} \sum_{h_i \in o, 1 \leqslant i \leqslant m} i\mu(h_i).$$

Notice then that the first term is $\mathrm{NG}(h)$ and the second term is $\mathrm{NG}(\mathtt{LoI}(P))$.

□

The following results sum up the order result of this work in the context of programs:

**Theorem 1.** Given programs $P, P'$ the following are equivalent:

1. $\mathtt{LoI}(P) \sqsubseteq \mathtt{LoI}(P')$
2. $\forall \mu.\ \mathtt{LoI}(P) \leqslant_d \mathtt{LoI}(P')$
3. $\forall \mu.\ \ H_\mu(\mathtt{LoI}(P)) \leqslant H_\mu(\mathtt{LoI}(P'))$
4. $\forall n, \mu.\ \ G_{n,\mu}(\mathtt{LoI}(P)) \leqslant G_{n,\mu}(\mathtt{LoI}(P'))$
5. $\forall \mu.\ \ \mathrm{NG}_\mu(\mathtt{LoI}(P')) \leqslant \mathrm{NG}_\mu(\mathtt{LoI}(P))$
6. $\forall \mu.\ \ \mathrm{ME}_\mu(P) \leqslant \mathrm{ME}_\mu(P')$
7. $\forall \mu.\ \ \mathrm{GE}_\mu(P) \leqslant \mathrm{GE}_\mu(P')$.

*Proof.* Equivalence $1 \Leftrightarrow 3$ was first proved in Heusser and Malacaria (2009), equivalences $1 \Leftrightarrow 2, 4, 5$ proved in Section 3; equivalences $1 \Leftrightarrow 3, 6, 7$ are proven in Yasuoka and Terauchi (2010a). It may be however interesting to reprove the equivalences in Yasuoka and Terauchi (2010a) using the algebraic techniques and results from this paper. For example, we can prove $1 \Leftrightarrow 6$ as follows:

$$\begin{aligned}\mathtt{LoI}(P) \sqsubseteq \mathtt{LoI}(P') &\Leftrightarrow \forall \mu.\ G(\mathtt{LoI}(P)) \leqslant G(\mathtt{LoI}(P')) \\ &\Leftrightarrow \forall \mu.\ \log(G(\mathtt{LoI}(P))) - \log(G(h)) \leqslant \log(G(\mathtt{LoI}(P'))) - \log(G(h)) \\ &\Leftrightarrow \forall \mu.\ \mathrm{ME}(P) \leqslant \mathrm{ME}(P')\end{aligned}$$

where the first equivalence is Corollary 1 and the second is Proposition 6.

$1 \Leftrightarrow 7$ follows from Proposition 6(2) by rewriting $\mathrm{GE}(P)$ as $\mathrm{NG}(h) - \mathrm{NG}(\mathtt{LoI}(P))$. □

A consequence of these results is that programs that are not ordered in LoI can be ordered by particular distributions in any possible way. An important example is now discussed.

https://doi.org/10.1017/S0960129513000649 Published online by Cambridge University Press

4.4. *Discussion on Smith's argument on the foundations of quantitative information flow*

Consider the following two programs (Smith 2009) where $h$ is a secret of size $8k$ bits:

1. $P_1 \equiv$ `if (h%8 == 0) then o = h; else o = 1;`
2. $P_2 \equiv$ `o = h&` $0^{7k-1}1^{k+1}$.

The program $P_1$ will return the value of $h$ when the last three bits of the secret are 0s and will return 1 otherwise; its `LoI` interpretation will hence be the partition of the form

$$X = \{\{h_1000\}, \ldots, \{h_m000\}, X_1\}$$

where the $h_i$ are arbitrary binary string of length $8k - 3$.

The program $P_2$ copies the last $k + 1$ bits of the secret in $o$ ( & the bitwise and). The partition associated has hence the shape

$$Y = \{Y_1, \ldots, Y_r\}$$

where each $Y_i$ is a set of string with the same $k + 1$ last bits.

Smith notices that under uniform distribution the two programs have a very similar entropy ($H(X) = k + 0.169 < H(Y) = k + 1$) but they have a very different guessing behaviour; in the case of the first program in fact with probability one eight the whole secret is revealed, while in the second program all attempts reveal the last $k + 1$ bits of the secret but give no indication of what the remaining bits are. Hence, in general it is much easier to guess the secret in one try after running the first program than it is to guess the secret after running the second one. These programs motivate the introduction of vulnerability in QIF.

Smith's analysis above assumes a uniform distribution on the secret: it is only one of the possible ones in the `LoI`. The partitions $X$ and $Y$ are unrelated in `LoI`, hence by the results from Section 3, we can find distributions and number of guesses that make one's expected guessing probability less than the other.

For $G_n(X) < G_n(Y)$ notice that $X_1$ splits in many blocks $Y_i$: hence take any distribution non-zero only on the atoms in $X_1$, e.g. let us consider the uniform distribution on the atoms in $X_1$ and take $n = |X_1| - 1$.

Then,

$$G_n(X) = g_n(X_1) = \frac{n}{n+1} < 1 = G_n(Y).$$

To make $G_n(X) > G_n(Y)$ pick any block $Y_i$ in $Y$ whose last three bits are 0s; then this block is split in many $X_i$s in $X$, again by taking the distribution uniform over the elements of $Y_i$ and 0 otherwise and taking $n = |Y_i| - 1$ we have

$$G_n(Y) = g_n(Y_1) = \frac{n}{n+1} < 1 = G_n(X).$$

In fact all distributions giving probability 0 to all values divisible by 8 will favour program $P_2$ even when we consider a single guess ($n = 1$ ), and things do not change when we take ME instead of $G_n$.

Similarly, we can find distributions that make the expected number of guesses of any of the two programs less than the expected number of guesses of the other program. In particular, while for the uniform distribution it is much easier to guess the secret in

https://doi.org/10.1017/S0960129513000649 Published online by Cambridge University Press

the case of the first program compared to the second (which is at the heart of Smith's argument), by choosing the distribution zero everywhere apart from the block $X_1$ it becomes easier to guess the secret using the second program. While such a distribution may be seen as pathological it still shows the possible problems in making code analysis dependent on particular distributions.

### 4.5. LoI, *maximum leakage and channel capacity*

The relation between LoI and channel capacity has been investigated in the literature (Köpf and Smith 2010; Malacaria and Chen 2008; Yasuoka and Terauchi 2010b). The channel capacity of a program is defined as the maximum possible leakage for that program. Intuitively this is the context most advantageous for the attacker. LoI provides an elementary characterization of channel capacity: in fact as the leakage is defined by $H(\mathtt{LoI}(P))$ using the well-known information theoretical fact that the maximal entropy over a system with $n$ probabilities is $\log(n)$, we deduce that the channel capacity is $\log(|\mathtt{LoI}(P)|)$.

We note by $\mathtt{CC}(P)$ for the channel capacity of the program $P$. We have then

**Proposition 7.**

$$\mathtt{LoI}(P) \sqsubseteq \mathtt{LoI}(P') \Rightarrow \mathtt{CC}(P) \leqslant \mathtt{CC}(P').$$

If $\mathtt{LoI}(P) \sqsubseteq \mathtt{LoI}(P')$ then all blocks of $\mathtt{LoI}(P)$ are refined by blocks of $\mathtt{LoI}(P')$,so the number of blocks of $\mathtt{LoI}(P)$ is $\leqslant$ than the number of blocks of $\mathtt{LoI}(P')$, but the channel capacity for programs is the log of the number of blocks interpretation, hence the result is proved.

The opposite direction of the implication doesn't hold: for example the partitions

$$\{\{a,b,c\},\{d\}\} \text{ and } \{\{a,b\},\{c,d\}\}$$

are not order related but have the same channel capacity 1.

#### 4.5.1. LoI *and min-entropy channel capacity.*

The relation between channel capacity of a program $P$ and $\log(|\mathtt{LoI}(P)|)$ is not confined to Shannon entropy. In fact Braun *et al.* (2009) have shown that even if we choose Smith's min-entropy quantitative analysis we get the same value, i.e. maximum vulnerability of a program $P$ according to Smith measure ME is $\log(|\mathtt{LoI}(P)|)$. We hence have the equalities

$$\mathtt{CC}(P) = \log(|\mathtt{LoI}(P)|) = \max_{\mu} H_{\mu}(\mathtt{LoI}(P)) = \max_{\mu} \mathrm{ME}_{\mu}(P).$$

## 5. Low inputs, multiple runs and l.u.b. in LoI

A major source of confusion in security analysis derives from poorly defined attacker models. In this Section, we discuss a few common modelling issues and how they can be dealt with in LoI.

https://doi.org/10.1017/S0960129513000649 Published online by Cambridge University Press

### 5.1. *Active and passive attackers*

The LoI allows for different attacker's models: the most common and possibly interesting is the one corresponding to an *active attacker*, i.e. an attacker who control the low inputs; a typical example would be a cash machine where an attacker is able to choose a pin number. An active attacker can be modelled as we did in the previous sections by assuming that the low variables are initialized in the code, the initialization values corresponding to the attacker choice.

We could however also model a *passive attacker*, an eavesdropper with no power to choose the low inputs. In this case, the lattice atoms are the pair of low and high inputs. Take for example the program

```
if (h == l) then  o= 1; else o=2;
```

where $h, l$ are 2 bits variables. The partition associated to the programs is:

$$\mathtt{LoI}(P) = \{\{(x,x)|0 \leqslant x \leqslant 3\}, \{(x,y)|0 \leqslant x, y \leqslant 3, x \neq y\}\}$$

and for the low input $\mathtt{LoI}(l) = \{\{(x,0)|0 \leqslant x \leqslant 3\}, \ldots, \{(x,3)|0 \leqslant x \leqslant 3\}\}$. Hence

$$\mathtt{LoI}(P) \sqcup \mathtt{LoI}(l) = \{\{(0,0)\}, \{(1,0),(2,0),(3,0)\}, \ldots, \{(3,3)\}, \{(0,3),(1,3),(2,3)\}\}$$

assuming uniform distribution on the low and high inputs we then compute leakage as

$$H(P|l) = H(P,l) - H(l) = H(\mathtt{LoI}(P) \sqcup \mathtt{LoI}(l)) - H(\mathtt{LoI}(l)) = 1.2075 - 1 = .2075\,.$$

In fact an active attacker is a particular case of this setting, where the distribution on the inputs is such that only one low input has probability non-zero. In that case the atoms of the lattice are, up to isomorphism, only the high inputs and $H(P|l) = H(\mathtt{LoI}(P))$.

5.1.1. *Hidden variables, probabilistic and non-deterministic systems.* By interpreting a probabilistic system as a deterministic system plus some *hidden variables* we can analyse such systems within LoI. As an example, consider the program

```
x= coin or h;
```

where coin is a random boolean and $h$ a one bit variable; then we can see this program as a function of two variables $P(\mathtt{coin},\mathtt{h})$. We have hence partitions: $\mathtt{LoI}(P) = \{\{(0,0)\},\{(0,1),(1,0),(1,1)\}\}, \mathtt{LoI}(\mathtt{coin}) = \{\{(x,y)|0 \leqslant x, y \leqslant 1\}$ and compute leakage as we did in section 5.1 by $H(\mathtt{LoI}(P) \sqcup \mathtt{LoI}(\mathtt{coin})) - H(\mathtt{LoI}(\mathtt{coin}))$. The development of such interpretation and its ability to model the subtleties of probabilistic and non-deterministic systems (Alvim *et al.* 2010b; McIver and Morgan 2003) is left for future work.

### 5.2. *Non-termination*

In this work, we have defined observations as the program output values. We could however include among the possible observations non-termination. This does not change the theory: non-termination is just an additional equivalence class: the class of all input states over which the program does not terminate; of course the usual decidability problems arise where we try to compute such a class.

https://doi.org/10.1017/S0960129513000649 Published online by Cambridge University Press

### 5.3. *Multiple runs*

Another aspect of an attacker model that has a natural algebraic interpretation in LoI is an attacker capability to run the system $n$ times: for example an attacker trying three pin numbers on a cash machine. Running a program several times with different low inputs may reveal more and more information about the secret; for example consider the password checking program $P$

```
if (h == l)  then o= 1; else o=2;
```

If we run it once assigning the value 5 to the low variable we gain the information whether the secret is 5 or not; by running it twice, assigning to the low variable the values 5 and 7, we will gain the information whether the secret is 5 or 7 or something else. Written in terms of partitions this is nothing else than the join operation in LoI

$$\{\{5\},\{\neq 5\}\} \sqcup \{\{7\},\{\neq 7\}\} = \{\{5\},\{7\},\{\neq 5,7\}\}.$$

Hence, the knowledge available to an attacker running the program who can choose the low inputs and run the program $m$ times is modelled by the partition

$$\mathtt{LoI}(P_1) \sqcup \cdots \sqcup \mathtt{LoI}(P_m)$$

where $\mathtt{LoI}(P_j)$ is the partition corresponding to the $i$th run of the program.

Notice that this is an extensional definition, i.e. the way the attacker chooses this $m$ inputs is irrelevant. For example, adaptive attacks from Köpf and Basin (2007) can be modelled by $\sqcup\{\mathtt{LoI}(P_1) \sqcup \cdots \sqcup \mathtt{LoI}(P_m) | P_1 \ldots P_m \text{ a possible sequence}\}$.

5.3.1. *Does it leak the same information?* A related question is whether a program leaks always the same information for each run of the program; for example a program leaking the last bit of the secret always leaks the same information no matter how many times we run the program but a password check leaks different information when we run it choosing different low inputs. This question can also be addressed by using l.u.b.s: if the program $P$ leaks different information over different runs this means we can find two runs $P_i, P_j$ such that

$$\mathtt{LoI}(P_i) \sqcup \mathtt{LoI}(P_j) \sqsupset \mathtt{LoI}(P_i) \quad \text{or } \mathtt{LoI}(P_i) \sqcup \mathtt{LoI}(P_j) \sqsupset \mathtt{LoI}(P_j).$$

The interpretation of multiple runs in terms of l.u.b.s has also somehow a reverse implication, i.e. it is possible, given programs $P_1, P_2$ to build a program whose interpretation is their l.u.b. This result has a practical significance: when $P_1, P_2$ are different runs of the same program the l.u.b. is their self-composition (Barthe *et al.* 2004). Formally (Malacaria and Heusser 2010):

**Proposition 8.** Given programs $P_1, P_2$ there exists a program $P_{1\sqcup 2}$ such that

$$\mathtt{LoI}(P_{1\sqcup 2}) = \mathtt{LoI}(P_1) \sqcup \mathtt{LoI}(P_2).$$

Given $P_1, P_2$, define $P_{1\sqcup 2} = P_1'; P_2'$ where the primed programs $P_1', P_2'$ are $P_1, P_2$ with variables renamed so to have disjoint variable sets. If the two programs are syntactically

https://doi.org/10.1017/S0960129513000649 Published online by Cambridge University Press

equivalent, then this results in self-composition (Barthe *et al.* 2004). For example, consider the two programs:

$P_1 \equiv \texttt{if (h == 0) then x = 0; else x = 1;}$, $P_2 \equiv \texttt{if (h == 1) then x = 0; else x = 1;}$

with their partitions $\texttt{LoI}(P_1) = \{\{0\}, \{h \neq 0\}\}$ and $\texttt{LoI}(P_2) = \{\{1\}, \{h \neq 1\}\}$. The program $P_{1\sqcup 2}$ is the concatenation of the previous programs with variable renaming

$$\begin{aligned} P_{1\sqcup 2} \equiv\ & \texttt{h}' = \texttt{h}; \texttt{if (h}' == 0) \texttt{ x}' = 0; \texttt{ else x}' = 1; \\ & \texttt{h}'' = \texttt{h}; \texttt{if (h}'' == 1) \texttt{ x}'' = 0; \texttt{ else x}'' = 1 \end{aligned}$$

The corresponding lattice element is the join, i.e. intersection of blocks, of the individual programs $P_1, P_2$

$$\texttt{LoI}(P_{1\sqcup 2}) = \{\{0\}, \{1\}, \{h \neq 0, 1\} = \{\{0\}, \{h \neq 0\}\} \sqcup \{\{1\}, \{h \neq 1\}\}.$$

## 6. Further applications of LoI

We quickly review two applications of LoI beyond the foundational aspect:

### 6.1. *Loop analysis*

Loop constructs are difficult to analyse. However, they have a natural interpretation in the LoI. In informal terms the idea is that loops can be seen as l.u.b. of a chain in the LoI, where the chain is the interpretation of the different iterations of the loop. To understand the ideas let us consider the program

```
l=0;
while(l < h) {
 if (h==2) l=3 else l++
}
```

and let us now study the partitions it generates. The loop terminating in 0 iterations will reveal that h=0 i.e. the partition $W_0 = \{\{0\}\{1,2,3\}\}$, termination in 1 iteration will reveal h=1 if the output is 1 and h=2 if the output is 3 i.e. $W_1 = \{\{1\}\{2\}\{0,3\}\}$, the loop will never terminate in 2 iterations i.e. $W_2 = \{\{0,1,2,3\}\}$ and in 3 iterations will reveal that h=3 given the output 3, i.e. $W_3 = \{\{3\}\{0,1,2\}\}$. Let us define $W_{\leqslant n}$ as $\sqcup_{n \geqslant i \geqslant 0} W_i$; we have then the chain (trivial in this example)

$$W_{\leqslant 1} = W_{\leqslant 2} = W_{\leqslant 3} = \{\{0\}\{1\}\{2\}\{3\}\}.$$

We also introduce an additional partition $C$ to cater for the collisions in the loop: in the example above the collision partition is $C = \{\{0\}\{1\}\{2,3\}\}$ because for h=2 the loop terminates with output 3 in 1 iterations and for h=3 the loop terminates with the same output 3 in 3 iterations. We have then

$$\texttt{LoI}(P) = \sqcup_{n \geqslant 0} W_{\leqslant n} \sqcap C = \{\{0\}\{1\}\{2,3\}\}.$$

This setting is formalized in Malacaria and Heusser (2010): given a looping program $P$ define $W_{\leqslant n} = \sqcup_{n \geqslant i \geqslant 0} W_i$ as the equivalence relation corresponding to the output

https://doi.org/10.1017/S0960129513000649 Published online by Cambridge University Press

observations available for the loop terminating in $\leqslant n$ iterations and let the *collision equivalence* of a loop be the reflexive and transitive closure of the relation $\sigma \simeq_C \sigma'$ iff $\sigma, \sigma'$ generate the same output from different iterations.

The following is then true:

**Proposition 9.**

$$\mathtt{LoI}(P) = \sqcup_{n \geqslant 0} W_{\leqslant n} \sqcap C.$$

Hence, leakage $H(\mathtt{LoI}(P))$ for looping programs can be computed in terms of the chain $(W_{\leqslant n})_{n \geqslant 0}$ and the collision equivalence $C$. The equivalence of this technique with previous information theoretical analysis of loops (Malacaria 2010) is proved in Malacaria and Heusser (2010).

Notice that Propositions 6 and 9 can be used for an analysis of loops using Smith's vulnerability leakage and guessability leakage:

$$\mathrm{ME}(P) \;=\; \log(G(\mathtt{LoI}(P))) - \log(G(h)) = \log(G(\sqcup_{n \geqslant 0} W_{\leqslant n} \sqcap C)) - \log(G(h)) \tag{10}$$

$$\mathrm{GE}(P) \;=\; \mathrm{NG}(h) - \mathrm{NG}(\mathtt{LoI}(P)) = \mathrm{NG}(h) - \mathrm{NG}(\mathtt{LoI}(\sqcup_{n \geqslant 0} W_{\leqslant n} \sqcap C)) \tag{11}$$

### 6.2. *Analysis of C-code vulnerabilities*

Recent work inspired by the LoI interpretation of programs, has demonstrated the applicability of QIF to real world vulnerabilities. Previous attempts to implement a quantitative analysis had hit a major hurdle: in very simple terms since QIF is based on $\mathtt{LoI}(P)$ and $\mathtt{LoI}(P)$ is the set theoretical kernel of the denotational semantics of $P$ computing $\mathtt{LoI}(P)$ is computationally unfeasible. The approach followed in Heusser and Malacaria (2010) is to change the QIF question from computing $\mathtt{LoI}(P)$ to computing bounds on the channel capacity $\mathtt{CC}(P)$: their relation has been shown in Proposition 7.

In that work a C program is seen as a family of equivalence relations, one for each possible low input. Given a bound $k$ the aim is to determine whether does exists a member of this family of equivalence relations which has more than $k$ equivalence classes (we are hence modelling an active attacker in the sense of section 5.1). Using assume-guarantee reasoning these questions about bounds can be expressed in verification terms, an idea also proposed in Yasuoka and Terauchi (2010b). In particular by expressing them as drivers for the symbolic model checker CBMC (Clarke *et al.* 2004) several CVE reported vulnerabilities in the Linux kernel were quantitatively analysed in Heusser and Malacaria (2010), the secret $h$ here being the kernel memory. Moreover, the official patches for such vulnerabilities were formally verified as fixing the leak. From a verification point of view the problem is non-trivial: given a modest C structure consisting of five integers and a 4GB kernel memory there are $2^{160}$ equivalence relations over a space of $2^{32}$ atoms.

The need of considering all possible equivalence relations is demonstrated by vulnerability CVE-2007-2875 which is caused by an underflow, i.e. among all equivalence relations associated to the code only the ones associated to the input triggering the underflow had $> k$ equivalence classes. In general vulnerabilities were caused by programming bugs or wrong architecture: for example in CVE-2009-2847 kernel memory is not leaking when

https://doi.org/10.1017/S0960129513000649 Published online by Cambridge University Press

running on a 32 bits architecture but leaks when run on a 64 bits machine. The ability to analyse architecture leaks shows the flexibility of the approach.

## 7. Related works

The relation between number of observables and leakage was first stressed by Lowe (2002). The random variable interpretation of partitions we presented originated from Clark *et al.* (2005) and Köpf and Basin (2007) who also showed its relevance to side channels analysis. The lattice structure here presented was first outlined in Heusser and Malacaria (2009). The use of partitions in implementing QIF using verification techniques was first demonstrated by Backes *et al.* (2009).

The work (Yasuoka and Terauchi 2010a) has been very inspiring in many respects: equivalences in Theorem 1 are also considered in that paper from a verification perspective.

There is a wealth of work in QIF whose relation to `LoI` deserves further study. One thread is QIF beyond non-interactive deterministic systems; examples are Chatzikokolakis *et al.* (2008) and Chen and Malacaria (2009) on anonymity protocols, Alvim *et al.* (2010a) on information hiding systems, McIver and Morgan (2003) and Morgan (2009) on probabilistic systems. Another thread is about QIF approaches not based on an *objective probability* distribution, a high profile example being the work on beliefs (Clarkson *et al.* 2009).

## 8. Conclusions

We investigated the importance of the `LoI` for QIF. This lattice allows for an algebraic treatment of information leakage and clarifies the relationship between the information theoretical, probabilistic and guessability measures by relating them to the refinement order in `LoI`.

We have seen how these results fit and contribute to recent work in the community, especially the ones by Yasuoka and Terauchi (2010a) and by Smith (2009). It is a matter for future research to determine whether the `LoI` can also provide a unifying foundation for probabilistic systems and beliefs based approaches.

## Acknowledgements

I am very grateful to Jonathan Heusser with whom work reported in Sections 2 and 6 was carried on and to the anonymous referees for the helpful comments.

## References

Alvim, M., Andrés, M. and Palamidessi, C. (2010a) Information flow in interactive systems. In: *Proceedings CONCUR 2010* 102–116.

Alvim, M., Andrés, M. and Palamidessi, C. (2010b) Probabilistic information flow. In: *Proceedings LICS 2010* 314–321.

Backes, M., Köpf, B. and Rybalchenko, P. (2009) Automatic discovery and quantification of information leaks. In: *Proceeding of the 30th IEEE Symposium on Security and Privacy* 141–153.

https://doi.org/10.1017/S0960129513000649 Published online by Cambridge University Press

Barthe, G., D'Argenio, P. and Rezk, T. (2004) Secure information flow by self-composition. In: *CSFW'04: Proceedings of the 17th IEEE workshop on Computer Security Foundations* 100–114.

Birkhoff, G. (1948) *Lattice Theory, American Mathematical Society Colloquium Publications*, volume 25, revised edition, American Mathematical Society, New York.

Braun, C., Chatzikokolakis, K. and Palamidessi, C. (2009) Quantitative notions of leakage for one-try attacks. In: Proceeding of the 25th Conference on Mathematical Foundations of Programming Semantics (MFPS 2009). *Elsevier Electronic Notes in Theoretical Computer Science* **249** 7591.

Chatzikokolakis, K., Palamidessi, C. and Panangaden, P. (2008) Anonymity protocols as noisy channels. *Information and Computation* **206** (2-4) 378–401.

Chen, H. and Malacaria, P. (2007) Quantitative analysis of leakage for multi-threaded programs. In: *Proceedings ACM Workshop on Programming Languages and Analysis for Security (PLAS'07)* 31–40.

Chen, H. and Malacaria, P. (2009) Quantifying maximal loss of anonymity in protocols. In: *Proceedings ACM Symposium on Information, Computer and Communication Security (ASIACCS)* 206–217.

Clark, D., Hunt, S. and Malacaria, P. (2002) Quantitative analysis of the leakage of confidential data. In: QAPL'01, Quantitative Aspects of Programming Languages. *Electronic Notes in Theoretical Computer Science* **59** (3) 238–251.

Clark, D., Hunt, S. and Malacaria, P. (2005) Quantitative information flow, relations and polymorphic types. *Journal of Logic and Computation* **18** (2) 181–199.

Clark, D., Hunt, S. and Malacaria, P. (2007) A static analysis for quantifying information flow in a simple imperative language. *Journal of Computer Security* **15** (3) 321–371.

Clarke, E., Kroening, D. and Lerda, F. (2004) A tool for checking ANSI-C programs. *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2004)* Springer **988** 168–176.

Clarkson, M., Myers, A. and Schneider, F. (2009) Quantifying information flow with beliefs. *Journal of Computer and Security* **17** (5) 655–701.

Cover, T. and Thomas, J. (1991) *Elements of Information Theory*, John Wiley.

Denning, D. (1976) A lattice model of secure information flow. *Communication of the ACM* **19** (5) 236–243 ACM, New York, NY, USA.

Giacobazzi, R. and Mastroeni, I. (2004) Abstract non-interference: Parameterizing non-interference by abstract interpretation. In: *31st Annual Symposium on Principles of Programming Languages (POPL'04)*, ACM, Venice, Italy 186–197.

Gray, J. W. III (1991) Toward a mathematical foundation for information flow security. In: *Proceeding of the 1991 IEEE Symposium on Security and Privacy Oakland, CA* 2134.

Heusser, J. and Malacaria, P. (2009) Quantifying loop leakage using a lattice of partitions. In: *Proceedings of the 1st Workshop on Quantitative Analysis of Software (QA'09)* 17–26.

Heusser, J. and Malacaria, P. (2010) Quantifying information leaks in software. In: *Proceedings ACM Annual Computer Security Applications Conference, ACSAC 2010*, Austin, Texas. ACM 216–219.

Köpf, B. and Basin, D. (2007) An information-theoretic model for adaptive side-channel attacks. In: *Proceedings ACM conference on Computer and Communications Security* 286–296.

Köpf, B. and Smith, G. (2010) Vulnerability bounds and leakage resilience of blinded cryptography under timing attacks. *CSF 2010* 44–56.

Landauer, J. and Redmond, T. (1993) A lattice of information. In: *Proceeding of the IEEE Computer Security Foundations Workshop*, IEEE Computer Society Press 65–70.

Lowe, G. (2002) Quantifying information flow. In: *15th IEEE Computer Security Foundations Workshop (CSFW 2002), Nova Scotia Canada*, IEEE Computer Society 18–31.

Malacaria, P. (2007) Assessing security threats of looping constructs. In: *Proceeding of the ACM Symposium on Principles of Programming Language (POPL'07)* 225–235.

https://doi.org/10.1017/S0960129513000649 Published online by Cambridge University Press

Malacaria, P. (2010) Risk assessment of security threats for looping constructs. *Journal Of Computer Security* **18** (2) 191–228.

Malacaria, P. and Chen, H. (2008) Lagrange multipliers and maximum information leakage in different observational models. *ACM Workshop on Programming Languages and Analysis for Security* 135–146.

Malacaria, P. and Heusser, J. (2010) Information theory and security: Quantitative information flow. In: 10th International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM 2010. *Springer Lecture Notes in Computer Science* Bertinoro **6154** 87–134.

Massey, J. (1994) Guessing and entropy. In: *Proceedings of the 1994 IEEE International Symposium on Information Theory* 204.

Mclean, J. (1990) Security models and information flow. In: Proceedings of the IEEE Symposium on Security and Privacy, IEEE Computer Society Press 180–187.

McIver, A. and Morgan, C. (2003) A probabilistic approach to information hiding. *Programming Methodology*, Springer, New York, NY, USA 441–460.

Millen, J. K. (1987) Covert channel capacity. *IEEE Symposium on Security and Privacy* **600**, 1540–7993 IEEE Computer Society, Los Alamitos, CA, USA.

Morgan, C. C. (2009) The shadow knows: Refinement of ignorance in sequential programs. *Science of Computer Programming* Elsevier **74** (8) 629–653.

Mu, C. and Clark, D. (2009) An abstraction quantifying information flow over probabilistic semantics. In: Workshop on Quantitative Aspects of Programming Languages (QAPL). *ENTCS* **253** (3) 119–141.

Nakamura, Y. (1970) Entropy and semivaluations on semilattices. *Kodai Mathematical Seminar Reports* **22** 443–468.

Rényi, A. (1960) On measures of information and entropy. In: *Proceedings of the 4th Berkeley Symposium on Mathematics, Statistics and Probability* 547–561.

Shannon, C. (1948) A mathematical theory of communication. *Bell Systems Technical Journal* **27** (3) 379–423.

Shannon, C. (1953) The lattice theory of information. *IEEE Transactions on Information Theory* **1** 105–107.

Smith, G. (2009) On the foundations of quantitative information flow. In: Proceeding of the FOSSACS 2009: 12th International Conference on Foundations of Software Science and Computation Structures. *Lecture Notes in Computer Science* **5504** 288–302 York, UK.

Terauchi, T. and Aiken, A. (2005) Secure information flow as a safety problem. In: SAS. *Lecture Notes in Computer Science* **3672** 352–367.

Yasuoka, H. and Terauchi, T. (2010a) Quantitative information flow - verification hardness and possibilities. In: *Proceedings CSF* 15–27.

Yasuoka, H. and Terauchi, T. (2010b) On bounding problems of quantitative information flow. In: *Proceedings ESORICS* 357–372.

Winskel, G. (1993) *The Formal Semantics of Programming Languages*, The MIT Press.

https://doi.org/10.1017/S0960129513000649 Published online by Cambridge University Press