CHALMERS
UNIVERSITY OF TECHNOLOGY

A Quantale of Information

Citation for the original published paper (version of record):
Hunt, L., Sands, D. (2021). A Quantale of Information. Proceedings - IEEE Computer Security Foundations Symposium: 94-108. http://dx.doi.org/10.1109/CSF51468.2021.00031

(article starts on next page)

A Quantale of Information

Sebastian Hunt
City, University of London
London, UK
s.hunt@city.ac.uk

David Sands
Chalmers University of Technology
Gothenburg, Sweden
dave@chalmers.se

***Abstract*—Information flow properties are the semantic cornerstone of a wide range of program transformations, program analyses, and security properties. The variety of information that can be transmitted from inputs to outputs in a deterministic system can be captured by representing information as equivalence relations over the sets of possible values, using an equivalence relation on the input domain to model what may be learned, and an equivalence relation on the output to model what may be observed. The set of equivalence relations over a given set of values form a lattice, where the partial order models containment of information, and lattice join models the effect of combining information. This elegant and general structure is sometimes referred to as the *lattice of information*.**

In this paper we identify an abstraction of information flow which has not been studied previously, namely *disjunctive dependency* (depending on x *or* y, as distinct from depending on both x and y). We argue that this refines the space of semantic models for dependency in a way which is both interesting in its own right and which has applications in security settings of practical interest (in particular, where so-called "Chinese wall policies" are in effect).

To model disjunctive dependency we introduce a nontrivial generalisation of the lattice of information in the form of a richer structure, built on sets of equivalence relations closed under a novel condition called tiling-closure. This structure forms a *quantale* - a lattice equipped with a tensor operation - in which lattice join corresponds to disjunctive combination of information, and tensor corresponds to conjunctive combination. Using this we generalise the definition of information flow properties, and show that the definition has the key properties needed to support compositional reasoning about programs.

***Index Terms*—dependency analysis, information flow, noninterference, security policy**

I. Introduction

Consider any system with inputs and outputs. If you can place a bound on what information can be deduced about a certain input when observing a certain output, then you can derive many useful properties of the system - typically referred to as *information flow properties*. (A note on our use of the word *property*. There is potential for confusion in work related to formal verification, since the word is often used to refer to properties of *traces*, as in safety properties and liveness properties, but information flow requirements such as noninterference are neither saftey nor liveness properties. In the current paper we use the word to refer to properties of programs and systems, not of traces.)

For example, suppose our system has data sources (e.g. input channels) I_1, I_2 and I_3, and suppose we can show that $\{I_1, I_3\}$ is an upper bound on the set of sources about which you could potentially learn something by observing output channel O. The typical terminology is that there is *information flow from I_1 and I_3 to O*, or that *O depends on I_1 and I_3*. In this paper we will use the terms "information flows from A to B" and "B depends on A" interchangeably. Irrespective of *how* we compute the bound, we can instantiate this raw dependency information to derive a range of interesting properties of the program. For example we can check whether a given multi-level secure information-flow property is satisfied: suppose that the policy is specified by associating a security label with each data source and sink, and specifying a "may flow to" relation between labels, (for example, a simple two-level policy might specify ***public*** *may flow to* ***secret***, but not vice-versa). We require that the dependencies respect the flow policy, and this can be checked by ensuring that label(I_i) *may flow to* label(O) holds for both $i = 1$ and $i = 3$ [1].

In fact, information flow properties can be viewed as the basis of many other program analyses and transformations, such as for example *program slicing* [2]. This generality motivated Abadi et al. to develop a "core calculus" of dependency [3] with which to show that many different analyses had essentially the same dependency analysis at their core. However, from a soundness perspective the fundamental common factor is arguably the *semantics* of dependency itself, rather than any particular feature of a computable approximation to the dependency properties of programs. The semantics of information flow is the topic of this paper; the development of static analysis is left for future work.

A. The Lattice of Information

If information flow is about the possible information gained about a data source when observing a data sink, what exactly do we mean by information?

In a deterministic system, the information conveyed by an observation can be expressed at a variety of levels of granularity. In imperative code, one of the simplest but more widely used forms is to study the dependency between input parameters and outputs channels. More fine-grained notions of dependency might for example determine that a certain output only depends on some specific attributes of an input or inputs. Another form of dependency is *conditional dependency*; for example, one could have a security property in which data is labelled with security levels which might not be statically known. Here a conditional dependency property would be useful to (partly) determine security: *out.value* may only

depend on *in.value* when *out.level* (the security level of the output) is higher or equal to *in.level* (the security level of the input). One would also typically require that *out.level* does not depend on *in.value* in such a scenario.

Note that information flow properties are almost always about upper bounds on information. For example, we would say that $x * y$ depends on both x and y – even though y might be zero and thus $x * y$ does not depend on x in all cases.

In the context of deterministic computation (the focus of this paper) all of these varieties of information flow have a common semantic basis in a general structure which describes the possible varieties of information that one might be able to convey, a structure that has been studied abstractly since at least the 1920s [4], and used to understand various forms of dependence and independence in programs [5], [6], in security [7], and in the semantics of types in programming languages (e.g. [8]). This structure is the lattice of equivalence relations[1] ordered by refinement, dubbed the *lattice of information* by Landauer and Redmond [7].

We describe the lattice of information in more detail in Section II, but for now it suffices to say that the points in the lattice denote possible states of knowledge about an entity such as an input. The extreme states are: knowing everything about the value (modelled by the identity relation) at one extreme, and knowing nothing (modelled by the equivalence relation with a single big equivalence class) at the other. In between we can model varieties of information (e.g. knowing the checksum of a sequence of digits, modelled by the equivalence relation which relates any two values with the same checksum). These varieties of information form a lattice via a partial order relation which describes when one piece of knowledge subsumes another, and a least-upper-bound operation which describes the information obtained when combining information. Information flow properties of programs can be described uniformly in terms of elements of the lattice of information by exploring how the program behaves on each equivalence class. A typical static analysis of dependency, or static verification of an information flow security property, implicitly picks out a particular sublattice of the lattice of information, namely those points which describe information at the desired level of granularity.

This paper deals with a shortcoming of the lattice of information or any of its sublattices, namely an inability to express what we will call *disjunctive information flow properties*, and proposes a generalisation of the lattice of information, a *quantale of information*, which overcomes this limitation.

B. Motivation: Disjunctive Information Flow Policies

We motivate the development of a semantics for disjunctive information flow as a natural requirement when have policies which distinguish between alternative flows, an in particular when the choice of flow may be made at run time.

[1]Strictly speaking several of the above references use the more general lattice of *partial* equivalence relations, but we will not make use of the partial case in this work.

To build an intuitive understanding of the concept of disjunctive information flow policies, we consider some small examples of policies where disjunctive information flow is natural.

a) "Chinese Wall" Policies: The so-called "Chinese wall" policies are a standard concept from the business world relating to ethical handling of information. We will henceforth refer to these policies as *ethical wall* policies. The goal of such policies is to prevent conflict of interest situations. For example, a large law firm could represent both sides of a legal dispute. To handle this in an ethical way, there would be a strict separation between the respective legal teams – an ethical wall. We can view this as a disjunctive information flow policy: a single lawyer has, a priori, no constraint on which team they may work with. They can therefore access sensitive information from *either* party, but the key property is that they may not not learn sensitive information from both.

Such policies were studied formally in an access-control context by Brewer and Nash [9]. We will return to the Brewer-Nash model in detail in section IV.

b) Secret Sharing: An example based on the idea of *secret sharing* [10]. Suppose that a secret S has been split into two *shares* S_1 and S_2 to be distributed among agents A_1 and A_2 such that they each get one share. The idea in secret sharing is that neither A_1 nor A_2 can reconstruct S alone, but can do so by combining their shares. Suppose further that the assignment of respective shares to A_1 and A_2 is done on some basis outside of the control of the system (for example, first-come-first-served), or under the control of some other parameters of the system. What is the dependency property that we want from the system? The simplest property that provides the intended security is that data visible to A_i depends at most on S_1 *or* S_2. Existing semantics and analyses of dependency do not provide a way to represent this form of disjunction directly, and the meaning we intend for the informal "or" here is not just logical disjunction (the required property is *not* "A_i depends at most on S_1 or A_i depends at most on S_2"). We discuss this further below.

c) Bounding Information Release: Another example motivating disjunctive information flow policies relates to allowing a choice of what information is released but managing the amount which is released by providing a strict choice. Suppose that a system manages sensitive and non-sensitive data from a given user, and permits an analyst to make a query on the sensitive data under the assumption that the sensitive data has been protected using differential privacy [11] (a quantified notion of disclosure) by adding noise. Differential privacy has the nice compositional property that the disclosure quantity (the "epsilon" in the terminology of differential privacy) of two queries is the sum of the disclosure quantities of the individual queries.

Suppose that there are n possible queries that the system is willing to answer, each providing the same quantified amount of disclosure. The analyst may choose any answer (perhaps based on the non-sensitive data) – but cannot choose more

than one, since that would imply an unacceptable amount of disclosure.

C. The Semantics of Disjunctive Information Flow

In this paper we show how the lattice of information can be generalised to include all disjunctive combinations of information, and how this naturally captures the desired disjunctive dependency properties.

Disjunctive dependency looks intuitively reasonable but there is a major challenge if we are to formally analyse disjunctive dependency properties: *what do they mean?*

To see that disjunctive dependency is not just a trivial logical disjunction of standard dependency properties, consider again the computation over inputs x, y, and z in Listing 1. This is our canonical example of a disjunctive dependency. (For example, g and h may compute two secret shares and w may be the share released to A_1 in the above scenario, but the following discussion is not specific to this example.)

```
if p(x)
  w = g(y);
else
  w = h(z);
```

Fig. 1. A prototypical Disjunctive Dependency

As mentioned above, a conventional dependency analysis will say that `w` depends on `x`, `y` and `z` (the dependency on `x` arises indirectly via the control-flow). But note that, while `w` may depend on `x` and `y` in some runs, and on `x` and `z` in others, it will never depend on both `y` and `z` in the same run. Or, expressed more directly in terms of what you may learn by running this program and observing `w`: you may learn something about `x` and `y` (but nothing about `z`), *or* you may learn something about `x` and `z` (but nothing about `y`); there are no runs in which you simultaneously learn something about both `y` and `z`. This example suggests that there ought to be a general reasoning principle for conditionals, of the form:

$$\frac{E \textit{ depends-on } \mathbb{P} \quad F_i \textit{ depends-on } \mathbb{Q}_i \quad i = 1, 2;}{(\textbf{if } E \textbf{ then } F_1 \textbf{ else } F_2) \textit{ depends-on } \mathbb{P} \otimes (\mathbb{Q}_1 \vee \mathbb{Q}_2)}$$

where we read $\otimes$ as conjunction and $\vee$ as disjunction (here we anticipate the definitions and notation of Section III-D). In Section VI we prove the correctness of a formal (and more general) version of this rule (Lemma 6); establishing an appropriate semantic framework is the key contribution of this paper. But, at the outset, note that the semantics of $\vee$ cannot be a straightforward classical disjunction. Certainly it should combine with $\otimes$ in a reasonable way; in particular we require the equivalence $\mathbb{P} \otimes (\mathbb{Q}_1 \vee \mathbb{Q}_2) \equiv (\mathbb{P} \otimes \mathbb{Q}_1) \vee (\mathbb{P} \otimes \mathbb{Q}_2)$. But it cannot represent simple disjunction of dependencies: in fig. 1 it is *not* the case that "`w` depends only on `x` and `y`" or "`w` depends only on `x` and `z`", since neither of these statements is true for the program!

We will define a semantics for dependency that will include disjunctive dependencies (modelling for example "`x` *and* `y` ***or*** `x` *and* `z`") but is not restricted to talking about variable-to-variable dependencies; any degree of deterministic information flow can be represented.

Contributions

Our overarching contribution is the identification of a new property, disjunctive dependency and its semantics. More specifically, we make the following contributions:

1) We develop a semantic model for disjunctive dependency (§III), built on sets of equivalence relations closed under a novel condition called *tiling-closure*, and generalising the so-called *lattice of information*. We show that this structure forms a *quantale* [12], which is an enriched lattice where the partial order is the information ordering, but where lattice join models the *disjunctive* combination of information, and an additional tensor operation models the conjunctive combination of information.
2) We show (§IV) how we can give an information-flow interpretation to Brewer-Nash style ethical wall policies, and justify our interpretation with an epistemic analysis of the implications of such policies.
3) We show (§V) how tiling closure can be systematically used as an abstract domain refinement, enriching any abstraction based on the lattice of information with points representing the disjunctive combination of information.
4) We show (§VI) that key combinators for functions (composition, conditionals and pairing) enjoy compositionality properties with respect to their disjunctive information flows, paving the way for useful static program analyses.

II. Background: The Lattice of Information

In this section we review the *lattice of information* and how it can be used to specify information flow properties.

In this section we introduce a standard but general way to understand information flow properties of a function by considering its action on *equivalence relations* over its input space. This account is based on the common denominator of a variety of related models of information flow, including Landauer and Redmond's *lattice of information* [7], the *PER model of information flow* [6], [13], and *abstract noninterference* [14], [15], and the key ideas seem already present in an early note of Claude Shannon [16].

The core idea is to represent information about a given domain of values as an equivalence relation over that domain, where the equivalence classes represent sets of indistinguishable values. Before we present the lattice structure of equivalence relations, lets build some intuitions for the use of equivalence relations to model information taken from standard instances of this idea.

Consider a function f from some input space A to an output space B. To understand the information flow between A and B we will use equivalence relations over A and B respectively, where the equivalence relation on B represents the observer's view of B (i.e. the information about the output

that the observer can see), and the equivalence relation on A models what may be learned by said observation.

To illustrate this let us take a concrete example in a form often used to express confidentiality properties of imperative programs. Suppose that A and B are both spaces of finite mappings from a set of variables Var to some domain of values D. We will refer to such mappings as *stores*, $\mathrm{Sto} = \mathrm{Var} \to D$. Suppose further that Var is partitioned into two sets, *secret* and *public*. The intention is that variables in *secret* contain sensitive information, and that this should not leak to the public variables in the output store.

Now suppose that we want to give semantics to the statement "for function $f : \mathrm{Sto} \to \mathrm{Sto}$, there is no information flow from the secret part of the input to the public part of the output". In literature on language-based security this property is usually called *noninterference* [17].

The standard way to express this property is to require that for any two stores s and t which have equivalent values for all the public variables, $f(s)$ and $f(t)$ will have equivalent values for their respective public variables. Put another way, changing the values of secret input variables will have no effect on the final value of public variables.

Let us introduce some concise notation to make this statement precise, and to generalise this kind of property. For any subset $X \subseteq \mathrm{Var}$, we can characterise the observations available to an observer of X by the equivalence relation $\sim_X$ on Sto defined by: $s \sim_X t$ iff $\forall x \in X. s(x) = t(x)$. The essential idea is that, for an observer who can see only X, the stores s and t are indistinguishable iff $s \sim_X t$. In later examples we will write $\sim_x$ as shorthand for $\sim_{\{x\}}$, and $\sim_{xy}$ for $\sim_{\{x,y\}}$ etc.

Definition 1. *Let $f : A \to B$ and let P and Q be equivalence relations on A and B, respectively. Define:*

$$f : P \Rightarrow Q \;\text{ iff }\; a\ P\ a' \implies (fa)\ Q\ (fa')$$

In this notation[2], the noninterference statement that public outputs depend on at most the public inputs can be written as:

$$f : \sim_{\text{public}} \Rightarrow \sim_{\text{public}}$$

The key point here is that many varieties of information transmission between inputs and outputs can be modelled as properties of the form $f : P \Rightarrow Q$. Equivalence relation Q models the information that can be observed, and P models a bound on what can be learned.

We write Id for the finest equivalence relation (the identity relation) and All for the coarsest (the relation which lumps all elements together in a single equivalence class). Consider the role played by P when we characterise the dependency of f on its inputs using a statement of the form $f : P \Rightarrow Q$. The coarser P is, the *less* sensitive f must be to variations in its input. In the extreme case, if $f : \mathrm{All} \Rightarrow \mathrm{Id}$, then f must be a *constant* function, not depending on any part of its input. Conversely, a finer P means that f *may* depend strongly on its input. The lattice of information thus orders equivalence relations with finer relations above coarser ones:

$$P \sqsubseteq Q \stackrel{\text{def}}{=} Q \subseteq P$$

where in the right-hand-side we view a relation as a set of pairs (the pairs of elements in the relation). This means that if $P \sqsubseteq Q$ then the information represented by P is subsumed by the information represented by Q.

A key operation on equivalence relations for information flow analysis is *conjunction*, which we denote using $\sqcup$:

$$P \sqcup Q \stackrel{\text{def}}{=} P \cap Q$$

i.e. $a\ (P \sqcup Q)\ b$ iff $a\ P\ b$ and $a\ Q\ b$. With our chosen ordering, equivalence relations form a complete lattice with $\sqcup$ as the join, All as the bottom element, and Id as top [4]. Following [7], we will refer to this lattice as the *Lattice of Information*. We write $\mathrm{LoI}(A)$ to refer to the lattice of information over some set A.

A crucial consequence of these definitions is that $\Rightarrow$ enjoys the natural subtyping property with respect to the LoI lattice ordering:

Proposition 1. *Let $P \sqsubseteq P' \in \mathrm{LoI}(A)$ and let $Q' \sqsubseteq Q \in \mathrm{LoI}(B)$. Then:*

$$(f : P \Rightarrow Q) \implies (f : P' \Rightarrow Q')$$

for all $f : A \to B$.

A. Partitions

Another way of viewing an equivalence relations over a set A is as a *partition* of A. This view will be useful in the development that follows in this paper.

Given an equivalence relation P on a set A and an element $a \in A$, let $[a]_P$ denote the (necessarily unique) equivalence class of P which contains a. Let $[P]$ denote the set of all equivalence classes of P. Note that $[P]$ is a partition of A.

As an example, consider the domain $A = \{0, 1, 2, 3\}$. Figure 2 presents a Hasse diagramn of a sublattice of the lattice of information for A.

We will refer to the equivalence classes of a partition as *cells*. Note that one partition P is above another Q, precisely when each cell of Q can be "tiled" by (i.e. is the union of) some cells of P. For example the cell $\{1, 2, 3\}$ of the partition isZero can be tiled by the cells $\{\{1\}, \{2, 3\}\}$ of the partition above it in the figure. Tiling, as in the covering of a set using a collection of non-overlapping subsets, will play a key role in the semantics of disjunctive dependency.

III. The Semantics of Disjunctive Information

In this section we introduce our formal definition of disjunctive information.

[2]This notation is taken from Hunt's work on static analysis using partial equivalence relations [5], and later used by Sabelfeld and Sands to model information flow [13].

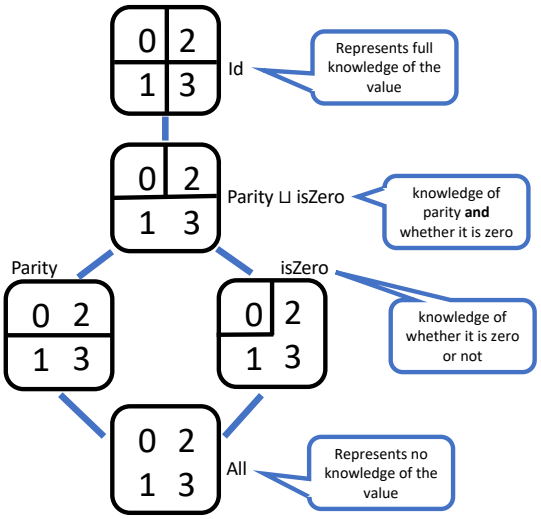

Fig. 2. An Example Sublattice of the lattice of Information over $\{0, 1, 2, 3\}$

A. Goals

We propose that any satisfactory definition should meet the following goals:

1) It should generalise the "standard" lattice of information definition reviewed in the previous section (Definition 1).
2) It should ensure that disjunctive dependencies allow us to specify more refined flow policies by "occupying the gap" between individual and conjunctive dependencies. That is, having dependency bounded by P_1 or P_2 should be a weaker requirement (satisfied by more functions) than having dependency P_i alone, but a stronger requirement than allowing both dependencies P_1 and P_2 simultaneously.
3) It should support reasoning about conditionals and, more generally, provide compositional reasoning principles, so that for example the dependency properties of a composition $f_2 \circ f_1$ can be computed compositionally from the properties of f_2 and f_1 respectively, as well as having strongest-postcondition and weakest-precondition principles.

B. Generalised Kernel

We recall that the *kernel* of a function $f : A \to B$ is the equivalence relation $\ker(f) \in \mathrm{LoI}(A)$ which relates all elements mapped by f to the same result: $a \ \ker(f) \ a'$ iff $fa = fa'$. Let us then focus on the equivalence classes of the kernel (the cells of its corresponding partition). Each $X \in [\ker(f)]$ is of the form $f^{-1}(\{b\})$ for some output b; an observer who sees b, knowing f, can infer only that the input belongs to X. Thus $[\ker(f)]$ can be understood as a bound on the information about inputs which is provided by the outputs of f.

This can usefully be generalised to provide an alternative characterisation of statements of the form $f : P \Rightarrow Q$.

Definition 2 (Generalised Kernel)**.** *Let $f : A \to B$ and let $Q \in \mathrm{LoI}(B)$. Let $f^{-1}(Q)$ denote the collection of non-empty preimages of the equivalence classes of Q under f:*

$$\{f^{-1}(Y) \mid Y \in [Q]\} \setminus \emptyset$$

It is a basic fact about functions that $f^{-1}(Q)$ defines a partition of the domain of f, thus we may view $f^{-1}(Q)$ itself as an equivalence relation.

We call this the generalised kernel (introduced, though not named, using the notation $f_{\#}(Q)$ in [7]) since the kernel of f corresponds to the special case given by $f^{-1}(\mathrm{Id})$. It is then easily seen that Definition 1 can be equivalently expressed as:

$$f : P \Rightarrow Q \ \text{ iff } \ f^{-1}(Q) \sqsubseteq P$$

With this perspective we may see $f : P \Rightarrow Q$ as stating that the information available to a Q-observer is bounded by $[P]$: for any actual input a, the observer learns $[a]_P$ or less (i.e. some superset of $[a]_P$). (This "epistemic" perspective is discussed in more detail in Section IV.)

Our definition of disjunctive information is essentially a generalisation of this view. Informally, we define $f : P \text{ or } R \Rightarrow Q$ to mean that that the information available to a Q-observer is bounded by the *union* of the cells of $[P]$ and $[R]$. To illustrate, we instantiate the secret-sharing idea as follows. For simplicity, we suppose that each share is a single bit and we only model one agent. Suppose that input y provides share S_1, input z provides share S_2 and input x is a flag determining which share agent A_1 receives (output variable w):

```
if(x) w = (1,y); else w = (2,z);
```

This defines a function $f(x, y, z)$ of three binary inputs. The output is independent of S_1 if it depends only on x and z; it is independent of S_2 if it depends only on x and y. Our policy for A_1 (the policy for A_2 would be the same) is therefore that it should depend (at most) on x and y *or* x and z:

$$f : \sim_{xy} \text{ or } \sim_{xz} \Rightarrow \sim_{w}$$

(To relate this to the earlier use of the $\sim$ notation, consider a triple as a simple kind of St_0 with just three components, each of which is a single bit. Thus $\sim_{xy}$ is the equivalence relation which relates just those (x, y, z) which agree on both x and y.) The three equivalence relations $f^{-1}(\sim_w)$ and $\sim_{xy}$ and $\sim_{xz}$ are represented in Figure 3 by their corresponding partitions. Note, for example, that each cell in the middle partition ($\sim_{xy}$) contains triples whose first two bits have the same value: 00 for the first cell, 01 for the second cell, and so on.
Note that $f^{-1}(\sim_w)$ cannot be represented in the same way as the other equivalence relations in this example; there is no subset $X \subseteq \{x, y, z\}$ such that $f^{-1}(\sim_w) = \sim_X$.

Given our informal definition we can immediately see that f satisfies the policy, since each cell of $f^{-1}(\sim_w)$ belongs to the union of the cells of the two disjuncts.

Our semantic model builds on this idea. We model a disjunction of equivalence relations by combining their equivalence classes in all possible ways to obtain a new *set* of equivalence

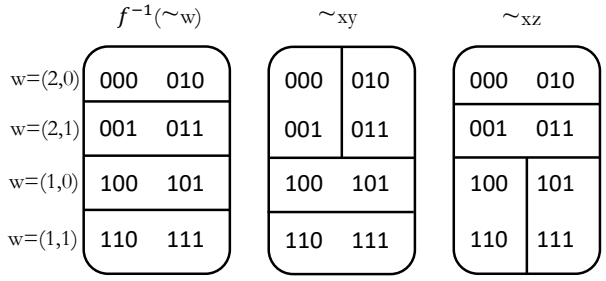

Fig. 3. Comparison of equivalence relations

relations, and we lift the definition of $\Rightarrow$ to sets of equivalence relations as follows:

Definition 3. *Let $f : A \to B$ and let $\mathbb{P}$ and $\mathbb{Q}$ be sets of equivalence relations on A and B, respectively. Define:*

$$f : \mathbb{P} \Rightarrow \mathbb{Q} \text{ iff } \forall Q \in \mathbb{Q}.\exists P \in \mathbb{P}.f : P \Rightarrow Q$$

(Overloading $\Rightarrow$ in this way should cause no confusion since $f : \{P\} \Rightarrow \{Q\}$ and $f : P \Rightarrow Q$ are clearly equivalent.)

C. Mixing and Tiling

The discussion above motivates the following definition:

Definition 4. *Let $\mathbb{P}$ be a subset of $\mathrm{LoI}(A)$ (i.e. a set of equivalence relations on some domain A). Define the set $\mathrm{mix}(\mathbb{P})$ to be the set of all equivalence relations that can be constructed by combining the cells of the members of $\mathbb{P}$. That is:*

$$\mathrm{mix}(\mathbb{P}) = \{R \in \mathrm{LoI}(A) | X \in [R] \implies \exists P \in \mathbb{P}.X \in [P]\}$$

Thus, in our secret-sharing example, the relevant disjunction is modelled by $\mathrm{mix}(\{\sim_{xy}, \sim_{xz}\})$, pictured in fig. 4. Note that

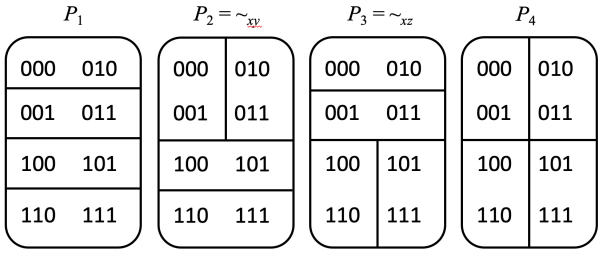

Fig. 4. $\mathrm{mix}(\{\sim_{xy}, \sim_{xz}\})$

as well as the original disjuncts P_2 and P_3 (which necessarily belong to the mix) the mix contains two new equivalence relations P_1 and P_4. It is clear that $f : \mathrm{mix}(\{\sim_{xy}, \sim_{xz}\}) \Rightarrow \{\sim_w\}$, since $f^{-1}(\sim_w) = P_1$. Any other f such that $f : P \Rightarrow \sim_w$ for some P in the mix will satisfy the same policy. For example:

f_1: w = (x,y);
f_2: w = (x,z);
f_3: if (x) w = (1,z); else w = (2,y);
f_4: if (x) w = z; else w = y;

Clearly, $f_1^{-1}(\sim_w) = \sim_{x,y} = P_2$ and $f_2^{-1}(\sim_w) = \sim_{x,z} = P_3$, so both f_1 and f_2 satisfy the policy. f_3 is the mirror image of the original f and it is easy to verify that it too satisfies the policy since $f_3^{-1}(\sim_w) = P_4$. f_4 is similar to f_3 except that it fails to tag its output. Intuitively, f_4 reveals *less* information than f_3, so we would expect it also to satisfy the policy. And indeed, $f_4^{-1}(\sim_w)$ can be obtained from P_4 by merging some cells, so $f_4^{-1}(\sim_w) \sqsubseteq P_4$, hence $f_4 : P_4 \Rightarrow \sim_w$.

While $\mathrm{mix}(\{\sim_{xy}, \sim_{xz}\})$ captures the intended disjunctive information, it is not unique in doing so. We take the view that $\mathbb{P}$ and $\mathbb{P}'$ represent *the same information* if they are interchangeable in all expressions of the form $f : \_ \Rightarrow \mathbb{Q}$ or $f : \mathbb{Q} \Rightarrow \_$. By Proposition 1, adding some additional P' makes no difference to the information represented by a set, as long as $P' \sqsubseteq P$ for some P already in the set. In particular, the following is an immediate corollary of Definition 3 and Proposition 1:

Proposition 2. *These are all equivalent (where $\downarrow$ denotes downwards closure):*

1) $f : \mathbb{P} \Rightarrow \mathbb{Q}$
2) $f : \downarrow\mathbb{P} \Rightarrow \mathbb{Q}$
3) $f : \mathbb{P} \Rightarrow \downarrow\mathbb{Q}$

To obtain canonical representations we therefore work with the downwards closures of our mix sets. (An alternative, and perhaps more fundamental, justification for downwards closure, is closure under post-processing. This is also discussed in more detail in Section IV.) For reasons that will soon be clear (Proposition 3) we call this combination of mixture and downwards closure, *tiling closure*:

Definition 5. *Let $\mathbb{P}$ be a set of equivalence relations. The tiling closure of $\mathbb{P}$, denoted $\mathrm{tc}(\mathbb{P})$, is defined by: $\mathrm{tc}(\mathbb{P}) = \downarrow\mathrm{mix}(\mathbb{P})$.*

It turns out that $\mathrm{tc}(\mathbb{P})$ can be defined directly in terms of a simple *tiling* relation. We say that a set of sets $\mathbb{X}$ tiles a set Y iff Y can be written as a disjoint union of sets chosen from $\mathbb{X}$. We lift this to equivalence relations in terms of the cells of their corresponding partitions: we say that a set of equivalence relations $\mathbb{P}$ tiles an equivalence relation S iff each $Y \in [S]$ can be tiled by the set of sets $\{X \in [P] | P \in \mathbb{P}\}$.

Proposition 3.

1) mix *and* tc *are both closure operators.*
2) $R \in \mathrm{tc}(\mathbb{P})$ *iff* $\mathbb{P}$ *tiles* R.

Proof:

1) *Evident consequences of the definitions.*
2) *Firstly, it is immediate from the definitions that $P \in \mathrm{mix}(\mathbb{P})$ implies that $\mathbb{P}$ tiles P. It is also clear that $R \sqsubseteq R'$ implies that R is tiled by R' (see the discussion at the end of Section II) so the set of all R tiled by $\mathbb{P}$ is downwards-closed. This establishes the implication from left to right.*
   Secondly, let $\chi \subseteq \bigcup_{P \in \mathbb{P}} [P]$ be a tiling of R by $\mathbb{P}$. Then χ is clearly a partition; let R_χ be the equivalence relation corresponding to this partition. Then $R_\chi \in \mathrm{mix}(\mathbb{P})$ and $R \sqsubseteq R_\chi$, hence $R \in \mathrm{tc}(\mathbb{P})$.

■

D. A Quantale of Information

To generalise the lattice of information we propose a structure which not only allows us to represent disjunction of information, in the way described above, but also to faithfully encode the same information as LoI. Our structure is a lattice, but it is not *only* a lattice: it has an additional operation, tensor, distinct from both join and meet. Moreover, while join provides conjunction in LoI, in this new lattice it provides disjunction; tensor provides conjunction.

Definition 6 (Quantale)**.**
A quantale [12] is a structure $\langle L, \sqsubseteq, \bigvee, \otimes, 1\rangle$ *such that:*

1) $\langle L, \sqsubseteq, \bigvee\rangle$ *is a complete join-semilattice.*
2) $\langle L, \otimes, 1\rangle$ *is a monoid:* $\otimes$ *is associative and* $x \otimes 1 = x = 1 \otimes x$ *for all* $x \in L$*, ;*
3) $\otimes$ *distributes over joins on both sides:* $x \otimes (\bigvee Y) = \bigvee\{x \otimes y | y \in Y\}$ *and* $(\bigvee Y) \otimes x = \bigvee\{y \otimes x | y \in Y\}$ *for all* $x \in L$ *and* $Y \subseteq L$*.*

A commutative *quantale is one in which* $\otimes$ *is commutative.*

For our purposes, the significance of the axioms (in particular for the commutative case) is that they ensure that disjunction (join) and conjunction (tensor) support algebraic reasoning conforming to some reasonably familiar logical principles: "and" distributes over "or", both are associative, both are commutative. However, as we will see, they are distinctly non-classical in some key respects.

Definition 7 (The Quantale of Information)**.**
$\mathrm{QoI}(A)$ *is* $\langle L, \sqsubseteq, \bigvee, \otimes, 1\rangle$ *where:*

- L *is the set of all tiling-closed subsets of* $\mathrm{LoI}(A)$*.*
- $\sqsubseteq = \subseteq$ *(hence* $\bot = \emptyset$*,* $\top = \mathrm{LoI}(A)$*)*
- $\bigvee_{i\in I} \mathbb{P}_i = \mathrm{tc}(\bigcup_{i\in I} \mathbb{P}_i)$
- $\mathbb{P} \otimes \mathbb{Q} = \mathrm{tc}(\{P \sqcup Q | P \in \mathbb{P}, Q \in \mathbb{Q}\})$
- $1 = \{\mathrm{All}\}$

Where the choice of A is not relevant to the discussion, we simply write QoI.

We have yet to establish that our definition actually respects the quantale axioms. First we need some basic lemmas; these essentially allow us to eliminate nested uses of tiling closure when reasoning about elements of QoI.

Definition 8. *For a closure operator* $\mathrm{cl} : A \to A$ *and a function* $F : A \to A$*, say that* F *weakly commutes with* cl *if* $F(\mathrm{cl}(X)) \subseteq \mathrm{cl}(F(X))$ *for all* $X \subseteq A$*.*

Lemma 1. *Let* $\mathrm{cl} : A \to A$ *be a closure operator and let* $X, Y \subseteq A$*. Suppose that* $F : A \to A$ *weakly commutes with* cl *and that* $G : A \times A \to A$ *weakly commutes with* cl *in each argument. Then:*

1) $\mathrm{cl}(F(\mathrm{cl}(X))) = \mathrm{cl}(F(X))$
2) $\mathrm{cl}(G(\mathrm{cl}(X) \times \mathrm{cl}(Y))) = \mathrm{cl}(G(X \times Y))$

Proof: Routine, using the properties of a closure operator. ■

Lemma 2.

1) *The join operator of* LoI *weakly commutes with* tc *in each argument.*
2) $\mathrm{tc}(X) \otimes \mathrm{tc}(Y) = \mathrm{tc}(\{P \sqcup Q | P \in X, Q \in Y\})$

Proof:

1) *Let* $P, Q \in \mathrm{LoI}$ *and let* $X \subseteq \mathrm{LoI}$*. It suffices to show that if* Q *is tiled by* X *then* $P \sqcup Q$ *is tiled by* $\{P \sqcup R | R \in X\}$*. This follows easily from the definition of tiling and the fact that* $[P \sqcup Q] = \{A \cap B | A \in [P], B \in [Q]\} \setminus \emptyset$*.*
2) *Follows from (1) by applying Lemma 1 to the definition of* $\otimes$*.*

■

Proposition 4. QoI *is a commutatative quantale.*
Proof:

1) *The join-semilattice conditions follow easily from the fact that* tc *is a closure operator.*
2) *To show associativity of* $\otimes$*: inline the definition of* $\otimes$ *in* $\mathbb{P} \otimes (\mathbb{Q} \otimes \mathbb{R})$ *and* $(\mathbb{P} \otimes \mathbb{Q}) \otimes \mathbb{R}$*; then use Lemma 2 to eliminate nested uses of* tc*; finally, flatten using associativity of* LoI $\sqcup$ *to obtain identical expressions. To show that* $\{\mathrm{All}\}$ *is a unit for* $\otimes$*, note that* All *is the bottom element of* LoI*; then* $\{\mathrm{All}\} \otimes \mathbb{P} = \mathrm{tc}(\{\mathrm{All} \sqcup P | P \in \mathbb{P}\}) = \mathrm{tc}(\mathbb{P}) = \mathbb{P}$*.*
3) *To establish distributivity we need to show that* $\mathbb{P} \otimes \bigvee_{i\in I} \mathbb{R}_i = \bigvee_{i\in I}(\mathbb{P} \otimes \mathbb{R}_i)$*: use Lemma 2 and basic properties of* $\bigcup$ *to reduce both sides to identical expressions. (Distribution on the right follows because* $\otimes$ *is commutative.)*

Commutativity of $\otimes$ *is inherited directly from commutativity of* LoI $\sqcup$*.* ■

Although lattice meet has no direct logical interpretation (note that it is not mentioned explicitly in the quantale axioms) every quantale is a complete lattice, so meets do exist in QoI. In fact, meet in QoI is simply set intersection (this is true for any lattice of sets in which joins are defined by applying a closure operator to unions).

E. QoI *Generalises* LoI

LoI can be embedded in QoI in such a way that every point of LoI has an exact representation in QoI, simply as its principal ideal (downward closure of a single point). Clearly, LoI cannot be a sublattice of QoI, since joins in QoI combine information in an essentially different way from joins in LoI. In fact, while the embedding does preserve meets, it maps join in LoI to *tensor* in QoI. Define $\mathrm{emb} : \mathrm{LoI} \to \mathrm{QoI}$ as the map: $P \mapsto \mathrm{tc}(\{P\}) = {\downarrow}P$. Then:

Proposition 5.

1) $P \sqsubseteq Q \implies \mathrm{emb}(P) \subseteq \mathrm{emb}(Q)$
2) $\mathrm{emb}(P \sqcup Q) = \mathrm{emb}(P) \otimes \mathrm{emb}(Q)$
3) $\mathrm{emb}(P \sqcap Q) = \mathrm{emb}(P) \cap \mathrm{emb}(Q)$

Proof: Routine. ■

F. Logical but Not Classical

As mentioned, while the quantale operations behave algebraically in a way which is broadly consistent with an

intuitive interpretation of $\otimes$ and $\vee$ as logical connectives for dependencies (conjunction and disjunction, respectively) they are not classical. We point out the key manifestations of this:

- As for classical disjunction, $\vee$ is idempotent. However $\otimes$ is only idempotent for the principal ideals of LoI. In general, $\mathbb{P} \otimes \mathbb{P} \neq \mathbb{P}$. For example, let $a = {\downarrow}P$, $b = {\downarrow}Q$. Then:

  $$(a \vee b) \otimes (a \vee b) = (a \otimes a) \vee (a \otimes b) \vee (b \otimes a) \vee (b \otimes b) = a \otimes b.$$

- The quantale lattice ordering does not correspond to classical entailment between propositions. In a classical entailment ordering, a and b would both be intermediate between $a \vee b$ and $a \wedge b$. However, in the quantale lattice we have $a, b \sqsubseteq a \vee b \sqsubseteq a \otimes b$.

Although we do not attempt to draw a formal connection here, the non-classical nature of these operators, together with the use of a quantale structure, strongly suggests connections with linear logic [18].

G. Goals Revisited

At the start of this section we listed three goals for a semantic definition of disjunctive dependency: (1) generalise LoI, (2) provide more fine-grained policies "occupying the gap" between individual and conjunctive dependencies, and (3) support compositional reasoning about conditionals (and more generally). For the first goal we showed quite explicitly how the LoI embeds into QoI, and together with Definition 1 it is clear that the standard definition of information flow between elements of LoI is subsumed by semantics of information flow in QoI. For the second goal, the following result shows that disjunction (join) in this specific quantale occupies that gap between conjunction (the tensor) and the individual dependencies. (But note that in general the tensor of a quantale need not dominate the join.)

Lemma 3. *Let $\mathbb{P}, \mathbb{Q} \in$ QoI. Then $\mathbb{P}, \mathbb{Q} \sqsubseteq \mathbb{P} \vee \mathbb{Q}$ and, if $\mathbb{P}, \mathbb{Q}$ are non-empty, then $\mathbb{P} \vee \mathbb{Q} \sqsubseteq \mathbb{P} \otimes \mathbb{Q}$.*

Proof: The first inequality holds simply because $\vee$ is lattice join. For the second inequality, consider that any non-empty element of QoI *contains* All*. It follows from the definitions that $\mathbb{P} \cup \mathbb{Q} \subseteq \mathbb{P} \otimes \mathbb{Q}$, since $R \sqcup \text{All} = R$, and hence (using the properties of closure operators) that $\mathbb{P} \vee \mathbb{Q} \subseteq \mathbb{P} \otimes \mathbb{Q}$.* ∎

Section VI addresses the third goal explicitly.

IV. Ethical Wall Policies

In this section we return to the question of ethical wall policies, and make precise: how ethical wall information flow policies can be modelled using disjunctive information flow, and in what sense our semantics captures the essence of an ethical wall policy in terms of what can be learned by an observer.

A. Modelling Policies

Following the formalisation proposed by Brewer and Nash [9], an ethical wall policy involves a set of companies C; the set of companies is partitioned into disjoint *conflict classes* $\{C_i\}_{i \in I}$ representing groups with mutual conflict of interest, for example a conflict class for all banks, another for all telecoms companies etc. We will write C_{ij} to denote the jth company in conflict class i, where j ranges over $\{1, \ldots, |C_i|\}$.

In the original Brewer-Nash model there is a data set associated with each company, consisting of a collection of objects. The access-control interpretation of a policy requires that a consultant (say) can access objects belonging to at most one company per conflict class.

In our information-flow interpretation of ethical wall policies we do not (need to) model how the company data is structured or represented.

We assume that the overall system has some input data D, and that D encompasses the data from all the companies, and possibly other data not pertaining to any company (e.g. public data, or data relating to the consultant or consultancy). To model the data from a specific company C_{ij} we assume an equivalence relation on D, RC_{ij}. Note that if we wanted to view the company data in terms of a projection function get$C_{ij} \in D \to D_{ij}$, then RC_i would simply be defined as the kernel of getC_{ij}.

At this point we might choose to insist that the information represented by the respective RC_i are independent from each other [3]. However, in our semantic analysis we will not require such an assumption, so our analysis will allow the case where some company data maybe shared between companies or even public.

Now consider the system as a function $f \in D \to E$. Our information-flow semantics for f satisfying the ethical wall policy is thus

$$f : \bigotimes_i \bigvee_j RC_{ij} \Rightarrow \text{Id}$$

Here and in the rest of this section we abuse notation slightly, writing an equivalence relation Q where we actually mean its embedding ${\downarrow}Q$ into QoI. (As noted previously, no ambiguity arises for statements of the form $f : P \Rightarrow Q$, since $f : P \Rightarrow Q$ iff $f : {\downarrow}P \Rightarrow {\downarrow}Q$.) Thus we are proposing a very direct encoding of the intended policy as a conjunction ($\otimes$) of disjunctions. Now we turn to a semantic justification for this choice in epistemic terms.

B. A knowledge-based Interpretation of Ethical Wall Information Flow

We now want to show how our semantics supports the intuitive interpretation of what an ethical wall policy should satisfy. But what is the intuitive meaning? Suppose that our consultant observes the outcome of applying f. Intuitively we might say that this observer should learn something about at most one company in each conflict class. But since there may

[3] A suitable notion of independence in the lattice of information is provided by Landauer and Redmond [7].

be mutual information across the companies in a conflict class, this would not be reasonable since what we learn could be part of the common knowledge between conflicting companies. Instead we can say that whatever *is* learned, could have been learned from at most one company per conflict class. To make this a bit more precise it is useful to take an example from which the general case should be self-evident: suppose that there are just two conflict classes A and B, each containing two companies $A = \{A_1, A_2\}$, $B = \{B_1, B_2\}$. Then we expect that, if f satisfies the ethical wall policy, our observer of f's output learns something which could have been learned from one of (i) A_1 and B_1, (ii) A_1 and B_2, (iii) A_2 and B_1, (iv) A_2 and B_2.

Now we turn to our semantic interpretation. The quantale properties allow us to rearrange the term $\bigotimes_i \bigvee_j RC_{ij}$ into a disjunction of conjunctions. In the case of the above example, the ethical wall policy is

$$(A_1 \vee A_2) \otimes (B_1 \vee B_2) \Rightarrow \mathrm{Id}$$

which, by the quantale properties, can be written equivalently as

$$(A_1 \otimes B_1) \vee (A_1 \otimes B_2) \vee (A_2 \otimes B_1) \vee (A_2 \otimes B_2) \Rightarrow \mathrm{Id}$$

Suppose that an observer sees the output of a function satisfying such a policy. What we would like to show, in some formal sense, is that what the observer can deduce about the input is something that could be deduced from a single "disjunct". As noted above, each A_i in these terms actually denotes the embedding $\downarrow A_i$ of a single equivalence relation into QoI. And by Proposition 5, $\downarrow A_i \otimes \downarrow B_j = \downarrow P$, where $P = A_i \sqcup B_j$ is a single equivalence relation. Thus the tensors in these terms have no special relevance. So what we will study is simply policies of the form $P_1 \vee \ldots \vee P_n \Rightarrow \mathrm{Id}$, and show that an observation of an output of a function satisfying this property allows us to learn something which is consistent with one of the P_i.

Towards this goal, let us introduce some terminology to capture what is learned by an observation – sometimes referred to as a "knowledge-based" view of information flow [19].

Definition 9 (Knowledge Set)**.** *Suppose $f \in D \to E$. We say that $K \subseteq D$ is a* knowledge set *for f, if K is non-empty and $K \in \{f^{-1}\{e\} \mid e \in E\}$.*

Each knowledge set of a function represents what an observer of a single output might be able to deduce about the input. Note that $f^{-1}\{e\}$ is empty iff $e \in E$ is not in the range of f, so e is not a possible observation. We exclude the empty set from our definition because impossible observations have no relevance in what follows. Note also that the set of all knowledge sets of f is just the (partition corresponding to) the kernel of f.

When we use an equivalence relation to model everything that might be known about an input we use it as an upper bound on what might be learned. For example, when we use a "low equivalence" to model knowledge of some low security inputs, we do not actually require that these can be learned from the output – the observer might not learn them at all on some or all runs. So what is the relation between equivalence relations used to model upper bounds on knowledge, and knowledge sets? This is captured in the following:

Definition 10 (Consistency)**.** *Given $R \in \mathrm{LoI}(D)$, we say that K is* consistent with R *iff R tiles K.*

Proposition 6. *$f : R \Rightarrow \mathrm{Id}$ if and only if every knowledge set of f is consistent with R.*

This is just a reformulated specialisation of the observation about generalised kernels from Section III, using the terminology above together with the fact that tiling captures the ordering relation in LoI.

Now we are in a position to formally state the informal property that we were aiming for. Suppose $f : \bigvee_{i \in I} P_i \Rightarrow \mathrm{Id}$. We want to show, simply, that each knowledge set of f is consistent with at least one of the disjuncts P_i. In fact we will prove something slightly weaker, but not weaker in any significant sense: we will show that there exists a function f' satisfying this property, and that f can be obtained from f' by post processing, i.e. there exists a function p such that $f = p \circ f'$.

It is worth saying a few words about post processing. In any notion of information or information flow which is based on providing upper bounds on information then it is reasonable to expect that if a given system satisfies the policy (in this case we are talking about the ethical wall policy), then post-processing the results of the system should also satisfy the same policy. This closure under post-processing has been articulated, in [20], as an axiom for any reasonable formal definition of privacy.

The intuition here is that post-processing can at best preserve information, and in general may throw information away. So with this in mind, we will show that the intended property is satisfied by some function f', and that f can be obtained from f' by post-processing.

We begin with a general factorisation property:

Lemma 4 (Factorisation)**.** *Given $f \in A \to B$, if $f : P \Rightarrow \mathrm{Id}$ and $P \sqsubseteq Q$ then f factors into a pair $f' : A \to C$ and $p : C \to B$ such that $f = p \circ f'$ and Q is the kernel of f'.*

Proof: Let $C = [Q]$ and let $f' = q : A \to [Q]$ be the quotient map $a \mapsto [a]_Q$. First, note that Q is the kernel of q, since $aQa' \iff [a]_Q = [a']_Q$. Then define $p : [Q] \to B$ by $p([a]_Q) = f(a)$. This is well-defined because: if aQa' then aPa' (since $P \sqsubseteq Q$) hence $f(a) = f(a')$ (since $f : P \Rightarrow \mathrm{Id}$). Let $a \in A$. Then $p(q(a)) = p([a]_Q) = f(a)$. ∎

Theorem 1. *If $f : \bigvee_{i \in I} P_i \Rightarrow \mathrm{Id}$ then $f = p \circ f'$ for some f' such that every knowledge set of f' is consistent with at least one P_i.*

Proof: Let $\mathbb{P} = \{P_i\}_{i \in I}$. From $f : \bigvee \mathbb{P} \Rightarrow \mathrm{Id}$ we have $f : Q \Rightarrow \mathrm{Id}$ for some $Q \in \mathrm{tc}(\mathbb{P}) = \downarrow \mathrm{mix}(\mathbb{P})$. So there is some $Q' \in \mathrm{mix}(\mathbb{P})$ such that $Q \sqsubseteq Q'$, and by the lemma f factors into $p \circ f'$ such that Q' is the kernel of f'. Since Q' is the

kernel of f', the knowledge sets of f' are just the elements of $[Q']$, and since $Q' \in \mathrm{mix}(\mathbb{P})$, every element of $[Q']$ is an element of at least one $[P_i]$. Thus every knowledge set of f' is an equivalence class of (and so is consistent with) at least one P_i. ∎

V. Tiling-Closure as an Abstract Domain Refinement

As remarked in the Introduction, many program analysis and transformation systems have dependency analysis at their core, and a common approach for imperative languages is to represent dependencies by sets of program variables[4].

In this section we show how the domains of simple dependency analyses can be enriched in a systematic way by adding in all disjunctions. It should, however, be noted up front that this is only one possible route to the verification of disjunctive policies; it is likely that more sophisticated forms of dependency analysis, specifically those which are *path sensitive*, and thus allow dependency properties to be conditional on the control-flow path [27], would be able to verify disjunctive policies directly. How this alternative approach plays out is left to further work.

Semantically, each such set of variables X can be understood to denote $\sim_X$, an element of the lattice of information, with larger sets denoting finer equivalence relations (higher points in the lattice, with the usual LoI ordering). More precisely, the usual interpretation of X is $\downarrow\sim_X$ - the downwards closure of $\sim_X$ - since in most applications we use the points of LoI to model upper bounds on information[5]. Figure 5 illustrates some examples, depicting the lattice of information over stores with four of the interior points and their respective downwards closures.

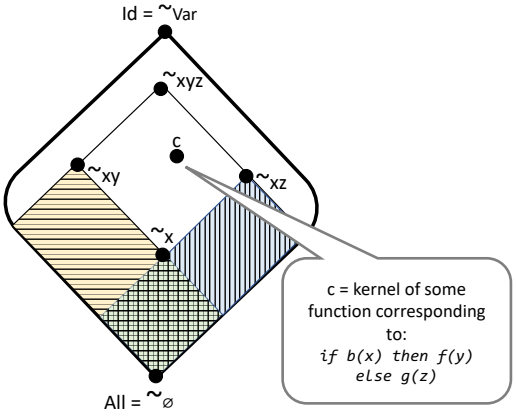

Fig. 5. Visualisation of some elements of LoI(Sto)

[4]For example, the "universal domain" of Hunt and Sands, [21], shown to be dual to the *Independ* domain and analysis of Amtoft and Banerjee [2], [22]. The *Independ* domain was also shown [23] to be isomorphic to Genaim, Giacobazzi and Mastroeni's *IF* domain for information flows [24]. Less formal variants of similar ideas arise in earlier works on dependency analysis, e.g. [25], [26].

[5]There are very few exceptions, for example Chong's work on required information release [28] and Foley's work on "separation" policies [29]

In the terminology of Abstract Interpretation[6] [30] we might say that such approaches use $\wp(\mathrm{Var})$ as an *abstract domain* together with a *concretisation function*

$$\gamma_L : \wp(\mathrm{Var}) \to \wp(\mathrm{LoI})$$

where $\gamma_L(X) = \downarrow\sim_X$. This abstraction has many advantages: it is intuitive and it is easy to implement in practical tools. More technically, it has the very pleasant property that it corresponds to a *sublattice* of LoI:

$$\begin{array}{rcl} \gamma_L(X \cup Y) & = & \downarrow(\sim_X \sqcup \sim_Y) \\ \gamma_L(X \cap Y) & = & \downarrow(\sim_X \sqcap \sim_Y) \end{array}$$

However, as we have argued, such an abstraction is unable to capture disjunctive dependency properties. This is shown in fig. 5: the best available abstraction of c - the kernel of a conditional function - is $\sim_{xyz}$. Note also that it would not help to use the obvious "disjunctive completion" of this abstraction; that would give us an abstract domain point corresponding to the union $\downarrow\sim_{xy} \cup \downarrow\sim_{xz}$, but this is no improvement since c belongs to neither set. Instead, we use tiling closure to lift the $\wp(\mathrm{Var})$ abstraction of LoI to an abstraction of QoI, thus lifting a purely conjunctive model of dependency to one that can also capture disjunctive dependencies. (The same construction could actually be used to lift *any* LoI abstraction to QoI.)

The lifting construction has two parts:

1) Elements of the new abstract domain are sets of elements from the original. So, for the sets-of-variables abstraction, we lift to *sets of sets* of variables. Given such a set of sets, for example $\{X, Y, Z\}$, our new concretisation function uses tiling closure to combine the equivalence relations corresponding to X, Y and Z:

$$\gamma_Q : \wp(\wp(\mathrm{Var})) \to \mathrm{QoI}$$

where $\gamma_Q(\mathbb{X}) = \mathrm{tc}(\bigcup_{X \in \mathbb{X}} \gamma_L(X))$.

2) We impose a quantale structure on the new abstract domain, thus:

$$\begin{array}{rcl} \mathbb{X} \vee \mathbb{Y} & = & \mathbb{X} \cup \mathbb{Y} \\ \mathbb{X} \otimes \mathbb{Y} & = & \{X \cup Y | X \in \mathbb{X}, Y \in \mathbb{Y}\} \end{array}$$

It is easy to verify that this does indeed have the structure of a quantale (in particular that $\otimes$ distributes over $\cup$) and that the quantale structure is preserved by γ_Q.

Actually, using arbitrary sets of sets entails some redundancy (some distinct abstract domain points represent the same property). For example, $\gamma_Q(\{\{x, y\}, \{x\}\}) = \gamma_Q(\{x, y\})$, since $\sim_x \sqsubseteq \sim_{xy}$. To obtain an irredundant abstraction we can either restrict to downwards closed sets of sets (with respect to subset inclusion of the element sets) or to sets of sets which are irredundant with respect to subset inclusion ($Y \in \mathbb{X} \wedge Z \in \mathbb{X} \implies Y \not\subset Z$). These are equivalent: the former can be more convenient in proofs; the latter is

[6]The analogy is not perfect. In some sense $\wp(\mathrm{LoI})$ here plays the role of the "collecting interpretation", which would usually be a lifting to sets of the concrete semantics. We are effectively taking the kernel of a function to be its concrete semantics.

more useful for implementation purposes and for readable presentation of examples.

In fig. 6 we illustrate the tiling closure of $\{\sim_{xy}, \sim_{xz}\}$, which is the γ_Q concretisation of the abstract domain point $\{\{x, y\}, \{x, z\}\}$. Note how the tiling closure includes the kernel c while lying strictly *between* $\downarrow\sim_{xyz}$ and $(\downarrow\sim_{xy})\cup(\downarrow\sim_{xz})$.

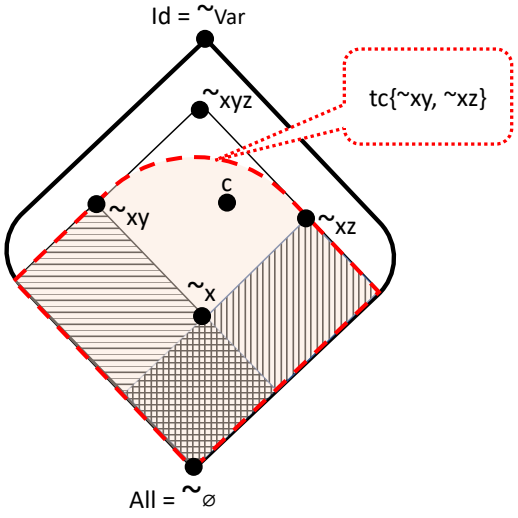

Fig. 6. Visualisation of some elements of QoI(Sto)

Example correspondences between the two abstractions are summarised in fig. 7, noting that for $R \in$ LoI we have $\mathrm{tc}\{R\} = \downarrow R$, and for $\mathbb{R} \subseteq$ LoI we have $\mathrm{tc}(\downarrow\mathbb{R}) = \mathrm{tc}(\mathbb{R})$.

VI. Towards Proving Properties of Programs

In this section we establish some basic lemmas which support compositional reasoning about program properties involving QoI. We address sequential composition, conditionals and pairing. We also establish the existence of weakest precondition and strongest post conditions, providing semantic support for Hoare-style logical reasoning.

For simplicity we assume a semantics in which programs denote functions in $D \to D$ for some domain D. We assume that $0, 1 \in D$ and that these are interpreted as Boolean values in conditionals (in fact, we assume that conditionals treat 0 as false, and all other values as true).

A. Sequential Composition

Reasoning about sequential compositions $f_1; f_2$ (that is, function composition $f_2 \circ f_1$) is supported by a standard "chain rule", whereby the postcondition of f_1 implies the precondition of f_2:

Lemma 5. *The following inference is valid:*

$$\frac{f_1 : \mathbb{P} \Rightarrow \mathbb{Q} \quad \mathbb{Q} \sqsupseteq \mathbb{Q}' \quad f_2 : \mathbb{Q}' \Rightarrow \mathbb{R}}{f_1; f_2 : \mathbb{P} \Rightarrow \mathbb{R}}$$

Proof: Use $f_1 : P \Rightarrow Q' \ \wedge \ f_2 : Q' \Rightarrow R$ *implies* $(f_2 \circ f_1) : P \Rightarrow R$ *and a routine unwinding of* $\forall\exists$ *in the definitions, working from right to left. Note that* $\mathbb{Q} \sqsupseteq \mathbb{Q}'$ *is just* $\mathbb{Q} \supseteq \mathbb{Q}'$*, so* $Q' \in \mathbb{Q}' \implies Q' \in \mathbb{Q}$. ■

B. Conditionals

For simplicity, we restrict attention to the case that the output property is the embedding $\mathrm{emb}(Q)$ of a single $Q \in$ LoI. To avoid notational clutter in what follows we abuse notation (as previously in Section IV) and simply write Q to mean $\downarrow Q$.

Let $f, g_1, g_2 : D \to D$. Define $\mathrm{cond}(f, g_1, g_2) : D \to D$ by:

$$\mathrm{cond}(f, g_1, g_2)(d) = \begin{cases} g_2(d) & \text{if } f(d) = 0 \\ g_1(d) & \text{otherwise} \end{cases}$$

Let $\mathrm{is0} \in \mathrm{LoI}(D)$ be the equivalence relation which distinguishes 0 from all other values, i.e. is0 has just two equivalence classes: $\{(0, 0)\}$ and $(D \times D) \setminus \{(0, 0)\}$. Then:

Lemma 6. *The following inference is valid:*

$$\frac{f : \mathbb{P} \Rightarrow \mathrm{is0} \quad g_1 : \mathbb{R}_1 \Rightarrow Q \quad g_2 : \mathbb{R}_2 \Rightarrow Q}{\mathrm{cond}(f, g_1, g_2) : \mathbb{P} \otimes (\mathbb{R}_1 \vee \mathbb{R}_2) \Rightarrow Q}$$

Proof: (Sketch) Extend Q *to* $\widehat{Q} \in \mathrm{LoI}(\{0, 1\} \times D)$ *where* $(i, d)\ \widehat{Q}\ (j, d')$ *iff* $i = j \wedge d\ Q\ d'$*. Note that the map* $\mathrm{untag} = (i, d) \mapsto d$ *satisfies* $\mathrm{untag} : \widehat{Q} \Rightarrow Q$*.*

Define a "tagged conditional" $\mathrm{tcond}(f, g_1, g_2) : D \to \{0, 1\} \times D$ *which pairs its output with* 0 *if* f *returns* 0*, or with* 1 *otherwise. Note that* $\mathrm{cond}(f, g_1, g_2) = \mathrm{tcond}(f, g_1, g_2); \mathrm{untag}$*. Since* $\mathrm{untag} : \widehat{Q} \Rightarrow Q$*, it will suffice to show* $\mathrm{tcond}(f, g_1, g_2) : \mathbb{P} \otimes (\mathbb{R}_1 \vee \mathbb{R}_2) \Rightarrow \widehat{Q}$ *and apply Lemma 5.*

Let $h = \mathrm{tcond}(f, g_1, g_2)$*. By the assumptions, there exist* $R_i \in \mathbb{R}_i$ *such that* $g_i : R_i \Rightarrow Q$*. Then by the definitions of* QoI $\otimes$ *and* $\vee$ *it will suffice to show that there exists* $P \in \mathbb{P}$ *such that* $h^{-1}(\widehat{Q})$ *is tiled by* $\{P \sqcup R_1, P \sqcup R_2\}$*. This follows with an easy argument by cases on* $X \in [\widehat{Q}]$*, according to whether* $X = \{0\} \times Y$ *or* $X = \{1\} \times Y$*, with* $Y \in [Q]$ *(note that* $X = \{0\} \times Y$ *entails* $f(d) = 0$ *for all* $d \in h^{-1}(X)$*, and* $X = \{1\} \times Y$ *entails* $f(d) \neq 0$ *for all* $d \in h^{-1}(X)$*).* ■

Together with Lemma 5, Lemma 6 is sufficient to establish the correctness of the examples considered in Section III. Note: in the examples of Section III, the functions corresponding to the f_i, g_i in the above lemmas are actually projections of a store onto a variable, but there is nothing in the lemmas which restricts their application to functions of this specific form.

C. Pairing

By pairing we mean pairing of functions in the sense of a categorical product. Our QoI inference rule for pairing turns out to be essentially just a lifting of the corresponding LoI rule to sets of equivalence relations. This is perhaps unsurprising since pairing is inherently conjunctive; LoI already provides a satisfactory representation of conjunctive information and this is inherited by QoI.

First we present an LoI inference rule. For $P_1 \in \mathrm{LoI}(A)$, $P_2 \in \mathrm{LoI}(B)$, define $P_1 \times P_2 \in \mathrm{LoI}(A \times B)$ by:

$$(a, b)\ (P_1 \times P_2)\ (a', b') \text{ iff } (a\ P_1\ a') \wedge (b\ P_2\ b')$$

Dependency Domain $\wp(\mathrm{Var})$		**Disjunctive Dependency Domain** $\wp(\wp(\mathrm{Var}))$	
Abstract value	Concrete property	Abstract value	Concrete property
$\{x, y\}$	$\gamma_L\{x, y\} = \downarrow\sim_{xy}$	$\{\{x, y\}\}$	$\gamma_Q\{\{x, y\}\}$ $= \mathrm{tc}(\gamma_L\{x, y\})$ $= \downarrow\sim_{xy}$
$\{x, y\} \cup \{x, z\}$ $= \{x, y, z\}$	$\gamma_L\{x, y, z\} = \downarrow\sim_{xyz}$	$\{\{x, y\}\} \otimes \{\{x, z\}\}$ $= \{\{x, y, z\}\}$	$\gamma_Q\{\{x, y, z\}\}$ $= \mathrm{tc}(\gamma_L\{x, y, z\})$ $= \downarrow\sim_{xyz}$
-	-	$\{\{x, y\}\} \vee \{\{x, z\}\}$ $= \{\{x, y\}, \{x, z\}\}$	$\gamma_Q\{\{x, y\}, \{x, z\}\}$ $= \mathrm{tc}((\gamma_L\{x, y\}) \cup (\gamma_L\{x, z\}))$ $= \mathrm{tc}\{\sim_{xy}, \sim_{xz}\}$

Fig. 7. Comparison of Dependency Abstractions

Validity of the following rule follows easily from the definitions:

$$\frac{f_1 : P_1 \Rightarrow Q_1 \quad f_2 : P_2 \Rightarrow Q_2}{\langle f_1, f_2\rangle : P_1 \sqcup P_2 \Rightarrow Q_1 \times Q_2}$$

We then lift _ × _ to sets of equivalence relations in the obvious way:

$$\mathbb{P}_1 \hat{\times} \mathbb{P}_2 = \{P_1 \times P_2 | P_1 \in \mathbb{P}_1, P_2 \in \mathbb{P}_2\}$$

(Note: so far this is just an operator on unstructured sets of equivalence relations, not on QoI.)

The key to showing that the LoI rule lifts to QoI is the following:

Lemma 7. *The $\hat{\times}$ operator preserves mix-closure, i.e. if $\mathbb{P}_1$ and $\mathbb{P}_2$ are both mix-closed, then so is $\mathbb{P}_1 \hat{\times} \mathbb{P}_2$.*

Proof: (Sketch) In essence, $P \times Q$ only combines the equivalence classes of P and Q in an inherently reversible way: every equivalence class of $P \times Q$ is a full Cartesian product $A \times B$ where $A \in [P]$ and $B \in [Q]$. As a consequence, $\mathbb{P}_1 \hat{\times} \mathbb{P}_2$ doesn't create any new opportunities for mixing. ∎

Lemma 7 ensures that the following is well-defined (note the use of downward-closure: no need to mix).

Definition 11. *Let $\mathbb{P}_1 \in \mathrm{QoI}(A)$ and let $\mathbb{P}_2 \in \mathrm{QoI}(B)$. Then $\mathbb{P}_1 \times \mathbb{P}_2 \in \mathrm{QoI}(A \times B)$ is defined to be $\downarrow(\mathbb{P}_1 \hat{\times} \mathbb{P}_2)$.*

Finally:

Lemma 8. *The following inference rule is valid for QoI:*

$$\frac{f_1 : \mathbb{P}_1 \Rightarrow \mathbb{Q}_1 \quad f_2 : \mathbb{P}_2 \Rightarrow \mathbb{Q}_2}{\langle f_1, f_2\rangle : \mathbb{P}_1 \otimes \mathbb{P}_2 \Rightarrow \mathbb{Q}_1 \times \mathbb{Q}_2}$$

Proof: By the definitions of $\otimes$ and $\times$, together with Proposition 2 and the fact that mix *is a closure operator, it suffices to show that the hypotheses of the rule ensure:*

$$\langle f_1, f_2\rangle : \{P_1 \sqcup P_2 | P_i \in \mathbb{P}_i\} \Rightarrow \{Q_1 \times Q_2 | Q_i \in \mathbb{Q}_i\}$$

So let $Q_i \in \mathbb{Q}_i$. We must show that there exist $P_i \in \mathbb{P}_i$ such that $\langle f_1, f_2\rangle : P_1 \sqcup P_2 \Rightarrow Q_1 \times Q_2$. For each choice of Q_i, the hypotheses of the QoI pairing rule imply the existence of $P_i \in \mathbb{P}_i$ such that $f_i : P_i \Rightarrow Q_i$, so the result follows by the LoI pairing rule. ∎

D. Weakest Preconditions and Strongest Postconditions

We conclude this section by showing the existence of weakest preconditions and strongest postconditions.

Proposition 7. *Let $f : A \to B$. Let $\mathbb{P} \in \mathrm{QoI}(A)$ and let $\mathbb{Q} \in \mathrm{QoI}(B)$. Then there exist:*

1) *A smallest $\mathbb{P}_* \in \mathrm{QoI}(A)$ such that $f : \mathbb{P}_* \Rightarrow \mathbb{Q}$*
2) *A greatest $\mathbb{Q}^* \in \mathrm{QoI}(B)$ such that $f : \mathbb{P} \Rightarrow \mathbb{Q}^*$*

Proof:

1) *Let $P_* = \mathrm{tc}(\{f^{-1}(Q) | Q \in \mathbb{Q}\})$. All elements of* QoI *are tiling-closed, hence downwards closed: by the definition of $\Rightarrow$ it then follows that we require a tiling-closed set which contains all the inverse images $f^{-1}(Q)$. P_* as defined is the smallest such set because* tc *is a closure operator.*
2) *Let $Q^* = \{Q \in \mathrm{LoI}(B) | f^{-1}(Q) \in \mathbb{P}\}$. Clearly, this is the largest set for which $f : \mathbb{P} \Rightarrow \mathbb{Q}^*$. It remains to show that it belongs to $\mathrm{QoI}(B)$, i.e. that it is tiling-closed. Let $Q \in \mathrm{LoI}(B)$ such that Q is tiled by $\mathbb{Q}^*$. Then all the tiles covering Q belong to some Q' such that $f^{-1}(Q') \in \mathbb{P}$. But then $f^{-1}(Q)$ can also be tiled by $\mathbb{P}$, hence $f^{-1}(Q) \in \mathbb{P}$ (since $\mathbb{P}$ is tiling-closed), hence $Q \in Q^*$.*

∎

VII. Related Work

The Brewer-Nash model for ethical wall policies is an access-control model [9]. As such, in contrast with the information-flow semantics that we propose in the current paper, it has nothing to say about how information flows once it has been accessed (legitimately or otherwise). Of more relevance is the later work of Foley: [29] proposes a policy framework that aims, in part, to allow the expression of ethical wall policies, while [31] proposes a semantics for this policy framework, using the non-interference (NI) model of Goguen and Meseguer [17]. In addition to policies which constrain what information flows are permitted, the framework of [29] attempts also to allow the expression of policies which *require* certain flows (Foley refers to the latter as "separation" policies); we model only permitted flows, so we do not consider this aspect of the work further. A direct formal

comparison between Foley's semantics and the QoI semantics is difficult, though our basic semantic setting – the lattice of information – is certainly more general than the specific deterministic system model and trace-purge NI definitions of [17] (a key benefit of this generality is the ability to talk about information flows at arbitrarily fine levels of granularity). For a given trace t and a given "class" a of output observer, Foley's NI semantics is based on a definition[7] of the set of all those classes of input providers which can interfere with the output (i.e. purging actions attributed to those classes from trace t would result in a different output observation at class a). We might try to encode this definition in our setting, by something like:

$$\bigcup\{X | \exists s \in [t]_{\sim_{\overline{X}}}. f(s) \notin [f(t)]_{\sim_a}\}$$

where $\overline{X}$ is the set of all observer classes *not* in X. However, this is rather convoluted and bears no obvious similarity to the QoI semantics. Of course, better encodings may be possible, but in any case it is not immediately clear how to develop a fully formal comparison of the two approaches. On the other hand, it is possible to construct informal analogues of our prototypical "if x then y else z" example in Foley's model. The essential idea is to define a system such that the observable state is insensitive to either the effects of "y-actions" or "z-actions" prior to a "write-once x-action", after which the observable state becomes sensitive either only to y-actions or only to z-actions. The corresponding policy in Foley's framework would include permitted flows $\{x, y\} \mapsto a$ and $\{x, z\} \mapsto a$, but not $\{x, y, z\} \mapsto a$. In this simple case, the NI semantics of [31] does confirm that the expected disjunctive flow policy is respected. Our intuition is that, like ours, Foley's model is able to capture such disjunctive flows because it builds on a per-run definition (the choice of trace t above) which gives it an implicitly epistemic character (c.f. Section IV).

In a setting of database queries [32], [33] define a notion of disclosure lattice built from sets of database "views", which is said to be a strict generalisation of LoI. It is not explained exactly what form of generalisation is achieved, but join in a disclosure lattice still appears to combine information in a purely conjunctive manner (the information content of a set of views is all the queries which can be answered by combining the views). The earlier paper [32] does propose a form of ethical wall policy – as a cut across the disclosure lattice – but this is a conventional logical disjunction; principals must choose a query which satisfies a specific disjunct, thus prohibiting the use of queries which make a choice between disjuncts dynamically, based on the state of the database.

We are aware of just one other line of work which builds a related semantics for disjunctive dependency. Morgan [34] observes that the lattice of information arises from considering information flow in deterministic systems, and seeks to find a corresponding structure for nondeterministic systems (rather than by viewing nondeterminism in terms of set-valued functions [13]). Nondeterminism can be thought of as a form of disjunction, and Morgan's proposed model eliminates the equivalence relation structure altogether, using sets of sets of values with no constraints on overlap (indeed, overlap corresponds in some sense to the sources of nondeterminism). The "demonic" lattice of information is built by taking these sets of sets and closing them under all unions, ordered by superset inclusion. Our tiling-closure can be thought of as a union-closure, but only applied to disjoint sets. The fundamental difference, however, is that we are building a richer set of abstractions than those provided by the lattice of information, but for the same underlying computation model, whereas Morgan's Demonic lattice is built as an analogue of the lattice of information for nondeterministic systems. It is not obvious how to combine Morgan's model with our model of disjunctive dependency.

In classical abstract interpretation the idea of lifting an analysis to include disjunctive properties (a process called *disjunctive completion*) is well known [35], going back to Cousot and Cousot's foundational work showing how disjunctive completion of an abstract domain gives a representation of meet-over-all-paths [36]. However these works deal with abstract domain elements which represent simple sets of concrete values, and thus the semantics of disjunction amounts to a logical disjunction of properties in the concrete world. This is not appropriate to model disjunction in a relational analysis like information flow, where the meanings of abstract domain points are relations on the concrete values, not just sets of values. Cousot and Cousot introduced *comportment analysis* [37] as an arguably more direct formulation of relational program properties as abstract interpretation, by working with a less constrained space of concrete objects, just sets of sets of values. Spoto shows that certain abstract lattices for dependency analysis, (namely Amtoft and Banerjee's lattice of independence properties *Independ* [2], [22], and Genaim, Giacobazzi and Mastroeni's *IF* domain for information flows [24]) are isomorphic and moreover already disjunctively complete.

Cousot and Cousot note some limitations in expressive power of equivalence-relation-based models, in particular when it comes to certain disjunctive combinations of properties, although these are likely more related to the disjunctive combination of information flow properties with other kinds of properties (such as strictness). In recent work, Cousot [38] defines a general semantics for dependency between inputs and program points, and shows how it can be systematically abstracted using abstract interpretation without resorting to a new collecting semantics.

Giacobazzi and Mastroeni [14] introduced a framework for information flow properties based on the idea of using abstract interpretations as a means to specify both the data "released" and the power of the observer. It should be noted that since any abstraction is a function, it induces an equivalence relation via its kernel, so the semantic content of the approach is similar to the use of LoI in specifying properties [15], but the approach accrues benefits from the abstract interpretation framework

[7]The definition of δ in [31], Section 3.

[39]. Abstract noninterference allows one to ask questions within a particular subset of the lattice of information which is identified by an abstract interpretation, such as what is the strongest observer (within that family) that we can allow for a given amount of information we wish to release on the input, or for a given observer, what is the most information (among the family) that we can release.

Quantale structures arise in the work of Giacobazzi et al [40] when refining abstract domains to achieve certain completness properties. In their terminology, tiling-closure may be seen as a complete abstraction with respect to the tensor of the "free" powerset quantale obtained by lifting LoI join to sets. However, there appears to be no way to *derive* tiling-closure using the techniques described in that paper. In particular, there is no evident "prototype" abstraction such that tiling-closure is the best (i.e., most abstract) complete refinement of the prototype. Modelling disjunctive dependencies appears to require a "eureka step" such as our definition of tiling-closure.

VIII. Conclusions and Further Work

We have described a novel generalisation of the lattice of information that captures a disjunctive form of information flow. The resulting structure, QoI, is a quantale – a lattice equipped with some additional structure – and the corresponding definition of information flow is well behaved in the sense that it has useful compositional properties. We have demonstrated a practical application of QoI by giving a semantics to ethical wall policies. These are clearly just first steps. What we need to consider next is how to compute sound disjunctive information flow properties, and where they can be further applied.

A. Static Analysis of Disjunctive Information Flow

The semantic framework developed in this paper lays the foundation for the development of provably sound static program analyses which can be applied in the automatic verification and enforcement of disjunctive policies.

A direct approach would be driven by the properties of the quantale of information in section III-D, which provide the core ingredients for describing approximations to disjunctive information flow in a compositional way. A promising strategy in this direction would be to adapt the flow-sensitive dependency analysis of Hunt and Sands [21] to include disjunctive dependencies, using the abstract lattice described in Section V as the basis of the language of types, and the properties of Section VI as key ingredients in a semantic soundness proof. An interesting aspect of this approach is that the addition of disjunctive dependencies creates a new dimension in precision which is not present in standard dependency analysis, namely the distinction between an *independent attribute* analysis (where the dependency of each output is described individually) and a relational analysis where one can say, for example, "outputs a and b depend on input x or y". In standard analysis of dependency there is no extra expressivity obtained from such relational abstractions, as shown in [23]. However, in the presence of disjunctive information this is no longer true, a there is a hierarchy of relational (i.e. non independent-attribute) variants of dependency analysis to explore.

Another approach, as mentioned in Section V, would be to build instead on existing forms of path-sensitive static analysis (e.g. [27]). The intuition here is that a path-sensitive analysis should be able to establish a disjunctive flow property by virtue of the fact that it can describe the precise path conditions under which the disjuncts arise.

To see a potential advantage of this approach, suppose we have a sequence of two conditionals, each of which gives rise to a disjunctive dependency on input variables $\{x, y\} \vee \{x, z\}$. Even supposing that neither conditional modifies the input variables, a naive compositional approach could never assume that the sequence as a whole satisfies the same disjunctive dependency, since, for a given value of x, the first conditional may choose a branch which depends on y while the second chooses a branch which depends on z. A path-sensitive approach, on the other hand, might also be able to establish that in any given run, both conditionals always branch the same way, and thereby establish that the program as a whole satisfies a disjunctive policy.

It remains to establish precisely how a path-sensitive analysis can be used for disjunctive dependency policies, and to compare and contrast the relative merits of these two approaches.

Acknowledgements

This work was partially supported by the Swedish Science Council and The Swedish Foundation for Strategic Research. The authors would like to thanks the anonymous referees for constructive suggestions.

References

[1] D. E. Denning, "A lattice model of secure information flow," *Comm. of the ACM*, vol. 19, no. 5, pp. 236–243, May 1976.

[2] T. Amtoft and A. Banerjee, "A logic for information flow analysis with an application to forward slicing of simple imperative programs," *Science of Computer Programming*, vol. 64, no. 1, pp. 3–28, 2007.

[3] M. Abadi, A. Banerjee, N. Heintze, and J. Riecke, "A core calculus of dependency," in *Proc. ACM Symp. on Principles of Programming Languages*, Jan. 1999, pp. 147–160.

[4] O. Ore, "Theory of equivalence relations," *Duke Math. J.*, vol. 9, no. 3, pp. 573–627, 09 1942. [Online]. Available: https://doi.org/10.1215/S0012-7094-42-00942-6

[5] S. Hunt, "Abstract interpretation of functional languages: from theory to practice," Ph.D. dissertation, Imperial College London, UK, 1991.

[6] S. Hunt and D. Sands, "Binding time analysis: A new perspective," in *In Proceedings of the ACM Symposium on Partial Evaluation and Semantics-Based Program Manipulation (PEPM'91*. ACM Press, 1991, pp. 154–164.

[7] J. Landauer and T. Redmond, "A lattice of information," in *CSFW*, 1993.

[8] M. Coppo and M. Zacchi, "Type inference and logical relations," in *LICS*, 1986.

[9] D. F. Brewer and M. J. Nash, "The chinese wall security policy," in *1989 IEEE Symposium on Security and Privacy*, 1989, pp. 206–214.

[10] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, p. 612–613, Nov. 1979.

[11] C. Dwork, "Differential privacy," in *Automata, languages and programming*. Springer, 2006, pp. 1–12.

[12] Springer Verlag GmbH, European Mathematical Society, "Encyclopedia of Mathematics," Website, uRL: https://encyclopediaofmath.org/index.php?title=Quantale\&oldid=42430. Accessed on 2020-10-15.

[13] A. Sabelfeld and D. Sands, "A per model of secure information flow in sequential programs," *Journal of Higher-Order and Symbolic Computation*, vol. 14, no. 1, pp. 59–91, Mar. 2001.

[14] R. Giacobazzi and I. Mastroeni, "Abstract non-interference: Parameterizing non-interference by abstract interpretation," in *POPL*, 2004.

[15] S. Hunt and I. Mastroeni, "The per model of abstract non-interference," in *Static Analysis*, C. Hankin and I. Siveroni, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 171–185.

[16] C. Shannon, "The lattice theory of information," *Transactions of the IRE Professional Group on Information Theory*, vol. 1, no. 1, pp. 105–107, 1953.

[17] J. A. Goguen and J. Meseguer, "Security policies and security models," in *IEEE Symposium on Security and Privacy*, 1982.

[18] D. N. Yetter, "Quantales and (noncommutative) linear logic," *The Journal of Symbolic Logic*, vol. 55, no. 1, pp. 41–64, 1990. [Online]. Available: http://www.jstor.org/stable/2274953

[19] A. Askarov and A. Sabelfeld, "Gradual release: Unifying declassification, encryption and key release policies," in *Proc. IEEE Symp. on Security and Privacy*, May 2007, pp. 207–221.

[20] D. Kifer and B.-R. Lin, "Towards an axiomatization of statistical privacy and utility," in *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ser. PODS '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 147–158. [Online]. Available: https://doi.org/10.1145/1807085.1807106

[21] S. Hunt and D. Sands, "On flow-sensitive security types," in *POPL'06, Proceedings of the 33rd Annual. ACM SIGPLAN - SIGACT. Symposium. on Principles of Programming Languages*, January 2006.

[22] T. Amtoft and A. Banerjee, "Information flow analysis in logical form," in *SAS 2004 (11th Static Analysis Symposium), Verona, Italy, August 2004*, ser. LNCS, vol. 3148. Springer-Verlag, 2004, pp. 100–115.

[23] F. Spoto, "Optimality and condensing of information flow through linear refinement," *Theor. Comput. Sci.*, vol. 388, no. 1–3, p. 53–82, Dec. 2007. [Online]. Available: https://doi.org/10.1016/j.tcs.2007.05.004

[24] S. Genaim, R. Giacobazzi, and I. Mastroeni, "Modeling Secure Information Flow with Boolean Functions," in *IFIP WG 1.7, ACM SIGPLAN and GI FoMSESS Workshop on Issues in the Theory of Security (WITS'04)*, 2004, pp. 55–66.

[25] J.-F. Bergeretti and B. Carré, "Information-flow and data-flow analysis of while-programs," *ACM TOPLAS*, vol. 7, no. 1, pp. 37–61, 1985.

[26] G. R. Andrews and R. P. Reitman, "An axiomatic approach to information flow in programs," *ACM TOPLAS*, vol. 2, no. 1, pp. 56–75, Jan. 1980.

[27] P. Li and D. Zhang, "Towards a flow- and path-sensitive information flow analysis," in *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, 2017, pp. 53–67.

[28] S. Chong, "Required information release," *J. Comput. Secur.*, vol. 20, no. 6, pp. 637–676, 2012.

[29] S. N. Foley, "A taxonomy for information flow policies and models," in *Proceedings of the 1991 IEEE Symposium on Security and Privacy, Oakland, California, USA, May 20-22, 1991*. IEEE Computer Society, 1991, pp. 98–109. [Online]. Available: https://doi.org/10.1109/RISP.1991.130778

[30] P. Cousot and R. Cousot, "Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints," in *Proceedings 4th Annual ACM Symposium on Principles of Programming Languages*, 1977, pp. 238–252.

[31] S. N. Foley, "Aggregation and separation as noninterference properties," *J. Comput. Secur.*, vol. 1, no. 2, pp. 159–188, 1992. [Online]. Available: https://doi.org/10.3233/JCS-1992-1203

[32] G. Bender, L. Kot, J. Gehrke, and C. Koch, "Fine-grained disclosure control for app ecosystems," in *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, K. A. Ross, D. Srivastava, and D. Papadias, Eds. ACM, 2013, pp. 869–880. [Online]. Available: https://doi.org/10.1145/2463676.2467798

[33] G. Bender, L. Kot, and J. Gehrke, "Explainable security for relational databases," in *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, C. E. Dyreson, F. Li, and M. T. Özsu, Eds. ACM, 2014, pp. 1411–1422. [Online]. Available: https://doi.org/10.1145/2588555.2593663

[34] C. Morgan, "A demonic lattice of information," in *Concurrency, Security, and Puzzles*, ser. Lecture Notes in Computer Science, T. Gibson-Robinson, P. J. Hopcroft, and R. Lazic, Eds., vol. 10160. Springer, 2017, pp. 203–222.

[35] G. Filé and F. Ranzato, "The powerset operator on abstract interpretations," *Theor. Comput. Sci.*, vol. 222, no. 1–2, p. 77–111, Jul. 1999. [Online]. Available: https://doi.org/10.1016/S0304-3975(98)00007-3

[36] P. Cousot and R. Cousot, "Systematic design of program analysis frameworks," in *Proceedings of the 6th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, ser. POPL '79. New York, NY, USA: ACM, 1979, pp. 269–282. [Online]. Available: http://doi.acm.org/10.1145/567752.567778

[37] ——, "Higher-order abstract interpretation (and application to comportment analysis generalizing strictness, termination, projection and PER analysis of functional languages)," in *Proceedings of the 1994 International Conference on Computer Languages, ICCL'94*. Toulouse, France: IEEE Computer Society Press, May 1994, pp. 95–112. [Online]. Available: http://www.ens.fr/~cousot/publications.www/CousotCousot-ICCL-94-IEEE-p95--112-1994.ps.gz

[38] P. Cousot, "Abstract semantic dependency," in *Static Analysis*, B.-Y. E. Chang, Ed. Cham: Springer International Publishing, 2019, pp. 389–410.

[39] R. Giacobazzi and I. Mastroeni, "Abstract non-interference: A unifying framework for weakening information-flow," *ACM Trans. Priv. Secur.*, vol. 21, no. 2, Feb. 2018. [Online]. Available: https://doi.org/10.1145/3175660

[40] R. Giacobazzi, F. Ranzato, and F. Scozzari, "Making abstract domains condensing," *ACM Trans. Comput. Logic*, vol. 6, no. 1, p. 33–60, Jan. 2005. [Online]. Available: https://doi.org/10.1145/1042038.1042040