Accountability, Involvement, and Mediation for Information Flow

Elisavet Kozyri
Computer Science
UiT The Arctic University of Norway
Tromsø, Norway
elisavet.kozyri@uit.no

Fred B. Schneider
Computer Science
Cornell University
Ithaca, USA
fbs@cs.cornell.edu

Stephen Chong
SEAS
Harvard University
Boston, USA
chong@seas.harvard.edu

***Abstract*—Explainability and transparency of computations, as well as compliance with data-privacy requirements, presuppose an understanding of whether some input plays a non-trivial role in computing an output. However, difficulty in distinguishing between correlation and actual usage in computation, along with possible correlations between inputs, makes determination of such accountability challenging. A flow definition is presented that can make this distinction. The definition enables the construction of accountability evidence to establish how an input is computed based on an output, in terms of the intermediate variables used. Intermediate variables can also serve as mediation points that fully block information flow from an input to an output. A connection between accountability and mediation is established by showing the role of mediation points for constructing accountability evidence.**

***Index Terms*—information flow, accountability, mediation, flow through, involvement**

I. Introduction

Information flow security [15], [25] has primarily been concerned with end-to-end enforcement of confidentiality and integrity policies. Confidentiality policies restrict what variables can be affected by secret values; integrity policies restrict what variables can be affected by untrusted values. But prevention is not the only approach to enforcement, and it is also not the only reason to analyze information flows in programs. This paper shows how an information flow semantics can be used to produce evidence that a given variable X is *accountable* for the value stored in a variable Y, meaning X plays a non-trivial role in determining the value that Y stores: (i) information about X can be inferred by observing Y, and (ii) this inference is not due to correlations.

A way to produce evidence of accountability has various applications in building secure systems. Knowledge that evidence of accountability will be available can deter principals from engaging in attacks, for fear of being identified and then punished. Evidence of accountability also can facilitate verifying whether information stored by a service should be deleted when a user invokes her "right to be forgotten"[1]. Consider two users, Alice and Bob, that provide correlated input data to a service (e.g., sleeping measurements with similar patterns) and assume that Alice exercises her right to be forgotten. Then Alice's data and anything computed based on this data (e.g., recommendations for sleep quality improvement) should be deleted. But how can one decide whether a certain digital product is the result of computations applied to Alice's data, instead of only Bob's data? We are not aware of a rigorous definition that can enable this decision. And we will show that obtaining such a definition is subtle.

Information flow in computer security is typically substantiated by using a counterfactual argument that checks whether different values for one variable lead to different values for another. However, systems where variables are correlated cause difficulties when accountability is to be determined. To illustrate, consider a system having two variables X_1 and X_2 storing the same secret, so initial condition $X_1 = X_2$ holds. For the program

$$Y := X_2 \tag{1}$$

all should agree that X_2 is accountable for the final value of Y but X_1 is not. A counterfactual argument affirms that X_2 is accountable, since different initial values for X_2 lead to different final values for Y. That same counterfactual argument, though, can be used to affirm that X_1 is accountable for the final value of Y, since changing the initial value in X_1 also gives that same value to X_2 (because $X_1 = X_2$ holds initially), and different values of X_2 lead to different values of Y. That we could conclude X_1 is accountable for Y suggests that we were using a flawed test for establishing accountability.

The obvious repair to vary X_1 separately from X_2, which is equivalent to ignoring correlations among variables, is not always satisfactory, either. So we check whether changes to the initial values of X_1 (when X_2 is not also changed) cause executions of the above assignment statement to change Y, and we reach the desired conclusion: X_1 is not accountable for Y, but X_2 is accountable for Y. A second example program, though, shows that ignoring correlations does not always correctly identify accountability. The program

$$Y := X_2 - X_1 \tag{2}$$

always assigns 0 to Y when executed in the initial states satisfying $X_1 = X_2$ that are being assumed. Arguably, neither

Fred B. Schneider is supported in part by Air Force Office of Scientific Research under award number FA9550-23-1-0435, as well as by funding from MIT Lincoln Laboratory, Amazon, and Google.

[1]https://gdpr.eu/right-to-be-forgotten/

© 2025, Elisavet Kozyri. Under license to IEEE.
DOI 10.1109/CSF64896.2025.00013

X_1 nor X_2 is accountable for Y in executions of (2) that start in states where $X_1 = X_2$ holds, because (2) has the same effect as $Y := 0$. Yet just varying X_1 or just varying X_2 would change the final value of Y. We reach the wrong conclusion for accountability in (2) if we ignore correlations and just check all pairs of executions. So that test is flawed, too.

A definition of accountability must distinguish correlation from causality. As we saw with the two example programs above, straightforward application of counterfactual reasoning does not make those distinctions. Therefore, in section IV-A, we derive a flow definition that is sensitive to correlative flows between variables. This definition is derived from elementary information flow definitions presented in section III, and it serves as a building block for our approach to establishing accountability.

A definition of accountability must take into account implementation details. Consider the following program, executed in an environment where initially $X_1 = X_2$ holds:

$$W := X_2 - X_1; \quad Y := W + X_1 \tag{3}$$

Programs (1) and (3) differ in the details of how they implement the same input-output relation. However, X_2 is arguably not accountable for Y in (3), because W is 0 when assignment "$Y := W + X_1$" executes, and thus, the only variable that actually conveys information to Y is X_1. So whether X_2 is accountable for Y depends on whether (1) or (3) is implementing the input-output relation.

We should expect that whether X should be considered accountable for Y, would depend on intermediate variables that hold values derived from X and that also are involved in computing the value of Y. So, to be considered compelling, accountability evidence ought to give those intermediate variables and characterize the roles they play. Section IV-B formalizes this reasoning and presents a definition for accountability evidence where systems are modeled as cascades of deterministic channels (described in section II). A deterministic channel is a simple computational model that can abstract any function.

Accountability evidence for a cascade of deterministic channels can identify those intermediate variables that are *involved* in a flow. An involved variable can be the source of corruption or leaks. Consider for example, a program where a variable W stores the result of an untrusted function. An integrity policy could specify that if trusted inputs to the program flow-through W to the output—that is, if W is involved in the flow from inputs to outputs—then the output should be considered untrusted. To enforce this policy, one first needs to define what is an involved variable. Section IV-C gives a first formalization.

Certain intermediate variables have the potential to mediate the flow from some input to some output. In practice, such a variable can guide the placement of a sanitizer S, to ensure that all flows from untrusted inputs to trusted outputs are mediated by the variable the stores the result of S. Although mediation appears often in Computer Security, we are not aware of formalizations for this view. Section V formalizes and explores it. Alternative definitions for mediation, each exhibiting different properties, are suggested. By formalizing mediation and accountability, we are able to connect these two ideas in a rigorous way: mediation can be used to build accountability evidence.

II. Channels and Cascades

Our accountability and mediation framework is built on channels. We employ notational conventions for channels from Cuff *et al.* [10]. Uppercase letters denote variables that identify the inputs and outputs to a channel; lowercase letters denote the values of these variables. Each variable Z is associated with a domain $\mathcal{Z}$ of values it may store. A variable Z may itself be a tuple $\langle Z_1, Z_2, \ldots, Z_n \rangle$ of variables, in which case $\mathcal{Z}$ is $\mathcal{Z}_1 \times \mathcal{Z}_2 \times \cdots \times \mathcal{Z}_n$, and n-tuple $\langle z_1, z_2, \ldots, z_n \rangle \in \mathcal{Z}$ is a value for Z. If Z_i is not a tuple of variables, then it is called a *singleton* variable. If $I \subseteq \{1, \ldots, n\}$ holds and Z is a variable that is a tuple $\langle Z_1, Z_2, \ldots, Z_n \rangle$, then Z_I denotes *partial* variable $\langle Z_i, \ldots, Z_j \rangle$, with $\{i, \ldots, j\} = I$ and $i \leq \ldots \leq j$. Also z_I denotes *partial* value $\langle z_i, \ldots, z_j \rangle$. $\overline{I}$ denotes the complement of I with respect to the set $[1, n]$ of all indexes, and $z =_I z'$ denotes $z_I = z_I{}'$.

We write $C(X : \mathcal{X} \triangleright Y : \mathcal{Y})$ to denote a deterministic channel with *input variable* X and *output variable* Y; this channel maps every input value $x \in \mathcal{X}$ to an output value $y \in \mathcal{Y}$. When input and output variables and domains are understood from the context, we may just write C to identify a channel. And we write $C(x)$ to denote the value y of output variable Y for channel C when input variable X has value x. We posit that distinguished element $\perp$ is in $\mathcal{X}$ and $\mathcal{Y}$, and that $C(\perp) = \perp$ hold for all channels C.

A channel will be represented either as a table that maps input values to output values or as a sequence of assignment statements where the channel outputs are the target variables of these assignments and the inputs are the variables appearing in the right-hand-side expressions. Assignment $Z := Z$ is written to assert that a channel propagates an input Z to an output Z unmodified.

Channels may operate in an environment that limits the possible input values. The set of values that the environment allows for an input variable X will be characterized by using a *support predicate* σ that is defined on values in $\mathcal{X}$. So, $\sigma(x)$ holds iff $x \in \mathcal{X}$ is an input value allowed by the environment. *Supported set* $\mathcal{X}^\sigma$ is the subset $\{x \in \mathcal{X} \mid \sigma(x)\}$. We write $\sigma \Rightarrow \sigma'$ to denote $(\forall x \in \mathcal{X} : \ \sigma(x) \Rightarrow \sigma'(x))$ and therefore $\mathcal{X}^\sigma \subseteq \mathcal{X}^{\sigma'}$. As a convention, the support predicate will be given as a precondition before the sequence of assignments that define a channel. For example, the following indicates that channel C operates in an environment where support predicate $X_1 = X_2$ holds:

$$\{X_1 = X_2\}$$
$$C: \quad Y := X_1 \ xor \ X_2$$

A *cascade* $\mathbb{C} \triangleq C_1 C_2 \ \ldots \ C_n$ is formed when each output variable of channel C_i is an input variable to channel C_{i+1}. The variables that connect the channels are called the *intermediate* variables of the cascade. Each channel C_i is considered a

constituent channel of the cascade. When only the input variable X, output variable Y, and an intermediate variable Z between channels C_i and C_{i+1} need to be referenced, then the cascade will be denoted by

$$X\, C_1\, \ldots\, C_i\;\; Z\;\; C_{i+1}\, \ldots\, C_n\, Y\,.$$

We use the notation $W^0, W^1, \ldots, W^n$ to identify a series of partial variables in a cascade $\mathbb{C}$, W^0 is a partial variable of the input, W^n is a partial variable of the output, and W^i is a partial variable of the ith intermediate variable of $\mathbb{C}$ (i.e., W^i is an input to channel C_{i+1} within $\mathbb{C}$). If an output value of a channel C_i is not among the values for which C_{i+1} is defined, then we assume that C_{i+1} outputs special value $\perp$.

Every cascade $\mathbb{C}$ itself defines a channel, where the inputs of $\mathbb{C}$ are the inputs of channel C_1, the outputs of $\mathbb{C}$ are the outputs of channel C_n: $\mathbb{C}(x) \triangleq C_n(\,\cdots C_2(C_1(x)))$.

A cascade can model a system at various degrees of faithfulness; two ways that such a model can be refined will be of interest. If one of the channels C_i of a cascade $\mathbb{C}$ is replaced with a cascade $\mathbb{C}_i$ having the same input-output relation as C_i, then the resulting cascade $\mathbb{C}'$ is said to be a *channel-decomposition* of $\mathbb{C}$. A cascade that cannot be further channel-decomposed is constructed from *atomic channels*. An atomic channel may not be replaced by a cascade.

One reason to label a channel as atomic is because its internal structure is unknown, and thus, replacing such a channel with a cascade might lead to erroneous conclusions. So, whether a channel is considered atomic is a modeling choice—not a ground truth. For example, an atomic channel can model an indivisible atomic action.

A second form of decomposition is to replace a variable W of $\mathbb{C}$ with a tuple of variables, such that there is a certain isomorphism between the domain of W and that tuple. For example, a variable W that stores integer values can be replaced by a vector of variables that store the binary representation of W. The resulting cascade is said to be a *variable-decomposition* of $\mathbb{C}$.

III. Modulation

One family of definitions for information flow in the literature (starting with Cohen [9]) is based on counterfactuals: there is an information flow from the input variables to the output variables, if different input values lead to different output values.

We are concerned here with information flows that arise for only those inputs that satisfy a support predicate σ. An information flow is *supported* iff there is a counterfactual argument that varies inputs over values satisfying a given σ. Since we will be interested in information flows from individual variables or from sets of variables, we require a counterfactual definition of information flow that handles a partial input X_I and partial output Y_J.

For a variable X and a set I of indexes, a straightforward counterfactual definition of information flow from partial variable X_I would check whether there are different outputs produced for inputs that only differ in the values of X_I—we do not vary other variables comprising X. However, varying partial input X_I while leaving $X_{\overline{I}}$ fixed, could constitute an input value x where $\sigma(x)$ does not hold. Arguably, the use of such unsupported inputs does not constitute a credible counterfactual argument for justifying supported information flows. So this straightforward definition for information flow is problematic.

A. Independent Modulation (IM-flow)

To be able to vary inputs as needed for a counterfactual argument that establishes a supported information flow from partial variable X_I to partial variable Y_J, the value of X_I must not constrain the value of $X_{\overline{I}}$, and vice versa. So, for any supported input values x and x', the value $x \parallel_I x'$ that comprises[2] x_I and $x'_{\overline{I}}$ should also be supported.[3] We conclude that a supported set $\mathcal{X}^\sigma$ can be used in counterfactual arguments for flows from X_I only if $\mathcal{X}^\sigma$ exhibits *full coverage for* X_I:

$$FC(I, \mathcal{X}^\sigma) \triangleq (\forall x, x' \in \mathcal{X}^\sigma :\; x \parallel_I x' \in \mathcal{X}^\sigma) \tag{4}$$

In fact, (4) signifies that X_I is *independent* of $X_{\overline{I}}$.

For supported sets $\mathcal{X}^\sigma$ that exhibit full coverage for X_I, a characterization of information flow based on counterfactuals might be formalized as follows.[4]

Definition: Independent Modulation (IM-flow).
Given a channel $C(X : \mathcal{X} \triangleright Y : \mathcal{Y})$, index sets I and J, and support predicate σ such that supported set $\mathcal{X}^\sigma$ exhibits full coverage for X_I, partial input X_I *IM-flows to* output Y_J iff

$$\begin{aligned} IM(C, I, J, \sigma) \;\triangleq\; (\exists\, x_1, x_2 \in \mathcal{X}^\sigma :& \\ & x_1 =_{\overline{I}} x_2 \;\wedge\; C(x_1) \neq_J C(x_2) \;\wedge \\ & C(x_1) \neq \perp \;\wedge\; C(x_2) \neq \perp) \end{aligned}$$ ■

Notice that output value $\perp$ is ignored when making a counterfactual argument for establishing IM-flow.

IM-flow instantiates a counterfactual argument, because some input values x_1 and x_2 that differ only in the values of variables in I, are required to produce different outputs. So, if X_I IM-flows to Y_J within an environment modeled by σ, then observing Y_J and knowing σ reveal something about the value of X_I. IM-flow will be used to construct other information flow definitions.

B. Variable Extension Modulation (VXM-flow)

To understand the implications of not requiring full coverage (4) for establishing an IM-flow, consider channel C depicted in Figure 1. Output Y reveals something about possible values for partial input X_1: if Y is 7, then X_1 must be 1 or 2; if Y is 8, then X_1 must be 3 or 4. Computing IM(C, I, J, σ) with I being $\{1\}$ (i.e., it represents X_1), J being $\{1\}$ (i.e., it

[2] If $x'' = x \parallel_I x'$, then $x'' =_I x$ and $x'' =_{\overline{I}} x'$ should hold.

[3] This requirement is similar to generalized noninterference (GNI) [22]. GNI specifies requirements for absence of flow, while the accountability theory that this paper develops captures presence of flow due to actual computation.

[4] This definition is essentially Cohen's *strong dependency* definition 2-10 [9]. But Cohen does not use the strong dependency definition to further capture accountability, involvement, and mediation. That is our contribution.

$\langle X_1, X_2 \rangle$	Y
$\langle 1, 1 \rangle$	7
$\langle 2, 1 \rangle$	7
$\langle 3, 2 \rangle$	8
$\langle 4, 2 \rangle$	8

Fig. 1. Channel C with all supported inputs depicted.

X_1	X_2	Y
0	0	7
0	1	7
1	0	8
1	1	8

Fig. 2. Channel C_1

represents Y), and σ supporting only the inputs depicted in Figure 1, and thus, not satisfying full coverage (4), would (erroneously) assert that X_1 does not flow to Y. This is because there are no input values that agree on X_2 but yield different outputs, when X_1 varies.

Yet, there will be cases where the IM-flow definition cannot be applied because variables in $X_{\overline{I}}$ are correlated with variables in X_I, making it impossible to vary only X_I (while keeping $X_{\overline{I}}$ fixed) within the supported set of inputs. One way to handle this case is to enlarge I and obtain a set $K \supset I$ of indexes, where partial input X_K is independent of $X_{\overline{K}}$, causing $\mathcal{X}^\sigma$ to exhibit full coverage for X_K. The limit case is to have X_K be variable X, so inputs from variables in X_I are still being included (since X includes all variables in X_I). The IM-flow definition now applies, but an IM-flow established by using X_K could over-approximate the set of variables from which there is information flow to Y.

Define $I \uparrow \sigma$ to be the intersection of all supersets of I for which full coverage holds:

$$I \uparrow \sigma \triangleq \bigcap_{K \supseteq I \wedge FC(K, \mathcal{X}^\sigma)} K \tag{5}$$

We prove the following theorem, which states that $\mathcal{X}^\sigma$ exhibits full coverage for $X_{I\uparrow\sigma}$.[5]

Theorem 1. *$FC(I \uparrow \sigma, \mathcal{X}^\sigma)$ holds.* ∎

So, by construction, $I \uparrow \sigma$ is the minimum superset of I for which full coverage holds.

We use $X_{I\uparrow\sigma}$ to extend the definition of IM-flow, resulting an information flow definition that handles cases where supported subset $\mathcal{X}^\sigma$ does not exhibit full coverage for X_I.

Definition: Variable Extension Modulation (VXM-flow). Given a channel $C(X : \mathcal{X} \triangleright Y : \mathcal{Y})$, index sets I and J, and support predicate σ, a partial input X_I *VXM-flows* to Y_J iff $X_{I\uparrow\sigma}$ IM-flows to Y_J.

$$\text{VXM}(C, I, J, \sigma) \triangleq \text{IM}(C, I \uparrow \sigma, J, \sigma)$$ ∎

If $\text{VXM}(C, I, J, \sigma)$ holds, then one can learn information about $X_{I\uparrow\sigma}$, by observing an output value Y_J. In addition, because σ creates a correlation between $X_{I\uparrow\sigma}$ and X_I, and σ is publicly known, when $\text{VXM}(C, I, J, \sigma)$ holds, one can also infer information about X_I by knowing $X_{I\uparrow\sigma}$ and σ.

[5]Proofs of theorems appear in the appendix.

C. Correlative Flows

VXM-flows are, by construction, supported information flows. However, some VXM-flows are artifacts of correlations. A VXM-flow from X_I to Y is *correlative* when this flow is due to X_I being correlated with $X_{I\uparrow\sigma}$, and the values in X_J with $J \subseteq I \uparrow \sigma - I$ (i.e., $J \notin I$) are being used to compute Y. If a VXM-flow disappears when the size of the supported domain is enlarged—which breaks correlations between the partial input variables—then this flow is due to correlations caused by the original environment. Alternatively if a VXM-flow persists as the supported domain is enlarged, then this flow is *non-correlative*.

Definition: Correlative versus Non-correlative Flows. If $\text{VXM}(C, I, J, \sigma)$ holds, then this flow is *correlative* iff there exists a "more permissive" support predicate σ' such that X_I does not VXM-flow in C to Y_J:

$$Cor(C, I, J, \sigma) \triangleq \text{VXM}(C, I, J, \sigma) \wedge (\exists \sigma' \colon \sigma \Rightarrow \sigma' \wedge \neg\text{VXM}(C, I, J, \sigma'))$$

If the second conjunct of $Cor(C, I, J, \sigma)$ is not satisfied, then the VXM-flow is *non-correlative*:

$$NCor(C, I, J, \sigma) \triangleq \text{VXM}(C, I, J, \sigma) \wedge (\forall \sigma' \colon (\sigma \Rightarrow \sigma') \Rightarrow \text{VXM}(C, I, J, \sigma'))$$

∎

As an example, consider the channel in Figure 2. Here, X_1 VXM-flows to Y when the environment satisfies $X_1 = X_2$. This VXM-flow is non-correlative, because it is preserved under any extension of the environment. We also have that X_2 VXM-flows to Y when the environment satisfies $X_1 = X_2$. But this VXM-flow is correlative, because X_2 does not VXM-flow to Y when, for instance, the environment is extended to the the entire domain.

There is a subtlety in how the environment should be extended when checking for correlative flows. We cannot just replace the second conjunct in $Cor(C, I, J, \sigma)$ with $\neg\text{VXM}(C, I, J, true)$ (and similarly the second conjunct in $NCor(C, I, J, \sigma)$ with $\text{VXM}(C, I, J, true)$), which would simply extend the original environment only to the entire domain. Rather, we must extend the environment, checking each intermediate set, too. This is because, $\text{VXM}(C, I, J, \sigma)$ and $\text{VXM}(C, I, J, true)$ might be both true, but there might be a σ' with $\sigma \Rightarrow \sigma'$ such that $\text{VXM}(C, I, J, \sigma')$ does not hold, indicating that the original VXM flow for σ is correlative. As the support predicate σ' supports more values, the extension $X_{I\uparrow\sigma'}$ tends to decrease, making it possible for

VXM flows to disappear and reappear again. The following example illustrates this:

$$\begin{aligned} &\{X = W = Z\} \\ C_2\colon\quad &Y := \langle X - W, Z\rangle \end{aligned}$$

Here, X VXM-flows to Y for the original environment (i.e., σ supports inputs with $\{X = W = Z\}$), because changing X would mean changing W and Z (here $X_{I\restriction\sigma}$ is $\langle X, W, Z\rangle$), which in turn implies that Y changes when the assignment statement executes (because Z changes). If the environment is extended to support the entire domain (i.e., $\sigma' = true$), then we still have that X VXM-flows to Y, because we are allowed to change X, keeping W and Z fixed (here $X_{I\restriction true}$ is X), with the result that Y changes, too. It is only when the environment is extended to inputs satisfying $X = W$, but not requiring equality with Z (i.e., σ' supports inputs with $\{X = W\}$), that the VXM-flow from X to Y disappears. Keeping Z fixed and changing X, means that W also changes to satisfy $X = W$ (here $X_{I\restriction\sigma'}$ is $\langle X, W\rangle$), and thus Y remains fixed (the first element of Y stays 0 and the second element equals the fixed value of Z). So, considering inputs that only satisfy $X = W$, rather than also satisfying $X = W = Z$, leads to a domain that is larger than the original but smaller than the entire domain—this exposes the VXM-flow as being a correlative flow. We conclude that it is not sufficient to extend the environment to the entire domain, if we want to check whether a flow is non-correlative.

IV. Accountability

Our goal is to derive evidence that the values of some specific partial input variable X_I to a given cascade operating in a certain environment are accountable for computing the values of certain partial output variable Y_J. To be considered compelling, we believe that such *accountability evidence* must satisfy the following criteria:

R1: Give a sequence of intermediate variables used to compute Y_J from X_I, where each variable plays a non-trivial role in computing every successor.

R2: If there exists accountability evidence for a cascade $\mathbb{C}$, then there should exist accountability evidence for any channel-decomposition $\mathbb{C}^+$ of $\mathbb{C}$.

R3: If there exists accountability evidence for a cascade $\mathbb{C}$, then there should exist accountability evidence for a variable-decomposition $\mathbb{C}^+$ of $\mathbb{C}$.

R1 is needed to ensure that accountability evidence is connected to the implementation details of the cascade. R2 and R3 are useful for viewing accountability evidence at various granularities. Some technical issues that arise are discussed at the end of Section IV-B.

The form of accountability evidence we propose in this section requires identifying a singleton intermediate variable between each atomic channel in the cascade, where that singleton variable is playing a non-trivial role in computing output Y_J from input X_I. The accountability evidence shows how X_I affects those singleton intermediate variables, in a way that propagates through the cascade to Y_J. So, R1 will be satisfied by accountability evidence that satisfies our definition. R2 and R3 will follow trivially because the cascades we consider have a particular form that does not admit channel or variable decompositions.

Our form of accountability evidence is defined by structural induction:

- *Base case*: an atomic channel. Input-output pairs and variations of VXM-flows between them constitute the accountability evidence.
- *Inductive case*: a cascade with multiple atomic channels. The accountability evidence is a sequence of intermediate singleton variables, where each plays a non-trivial role in the computation of all its successors.

A. Base Case: CD-flow

For the case of atomic channel C and support predicate σ, R1 requires partial input X_I to play a non-trivial role in computing partial output Y_J, if accountability from X_I to Y_J is to be established. The essence of playing "a non-trivial role in computing" is:

(i) Information about values of X_I can be inferred by observing values of Y_J.

(ii) The inference is not due to correlations.

If X_I VXM-flows to Y_J (i.e., VXM(C, I, J, σ) holds), then (i) is satisfied, since then knowledge of the value of Y_J, in conjunction with σ, informs what values are possible for input X_I.

The following example illustrates that if (i) does not hold, then there is no basis to argue for accountability. Consider atomic channel C:

$$\begin{aligned} &\{X_1 = X_2\} \\ C\colon\quad &Y := X_1 - X_2 \end{aligned} \tag{6}$$

Here, even though X_1 and X_2 are referenced in the computation, neither of them VXM-flows to Y when σ only supports initial states where $X_1 = X_2$ holds: Y is assigned 0 when the program is executed in states satisfying σ ($X_1 = X_2$ holds) and, thus, nothing new can be inferred about X_1 or X_2 from observing Y. So, arguably, neither X_1 nor X_2 should be considered accountable for Y within that environment. Thus, if there is no VXM flow, then there is no basis for accountability.

Inference alone, however, should not be considered sufficient for establishing accountability—(ii) must be satisfied, as well. To illustrate, consider

$$\begin{aligned} &\{X_1 = X_2\} \\ C'\colon\quad &Y := X_2 \end{aligned} \tag{7}$$

where again predicate σ supports only input states where $X_1 = X_2$ holds. Here, the value of Y can be used to infer the values of both X_1 and X_2. So, information flows from X_1 (and X_2) to Y. Indeed, X_1 VXM-flows to Y given σ, as does X_2. This is because the VXM-flow definition expands partial input X_1 to $\langle X_1, X_2\rangle$ (since the supported set does not exhibit full support for X_1) and $\langle X_1, X_2\rangle$ IM-flows to Y.

However, the VXM-flow from X_1 to Y is due to correlation caused by σ. Thus, X_1 should not be considered accountable for Y. Since there is a VXM-flow, we conclude that VXM-flows alone do not define accountability.[6] So to discharge requirement (ii), we must also check whether the definition of a non-correlative flow $NCor(C, I, J, \sigma)$ is satisfied: the VXM-flow must be preserved under all extensions of the original environment.

The obligations for establishing accountability with an atomic channel are thus characterized in the following.

Definition: Computational Dependency (CD-flow).
Let I and J be index sets and let $C(X : \mathcal{X} \triangleright Y : \mathcal{Y})$ be a channel with support σ. Partial variable X_I *CD-flows to* output Y_J, denoted $X_I \underset{C}{\overset{\sigma}{\Longrightarrow}} Y_J$, iff $NCor(C, I, J, \sigma)$ holds. ■

The CD-flow definition captures requirement (i) in first conjunct $\text{VXM}(C, I, J, \sigma)$ of $NCor(C, I, J, \sigma)$, and requirement (ii) by second conjunct $(\forall \sigma' : \ (\sigma \Rightarrow \sigma') \Rightarrow \text{VXM}(C, I, J, \sigma'))$ of $NCor(C, I, J, \sigma)$. So the CD-flow definition captures the requirements for accountability of atomic channels: a CD-flow from X_I to Y_J is accountability evidence for an atomic channel C that is executed in an environment modeled by σ.

Atomic channels (6) and (7) above illustrate how CD-flows serve as accountability evidence.

- Example (6): Neither X_1 nor X_2 CD-flows to Y, because neither X_1 nor X_2 VXM-flows to Y. So we used CD-flow to confirm our earlier contention that neither X_1 nor X_2 is accountable for Y
- Example (7): X_1 does not CD-flow to Y, because the VXM-flow from X_1 to Y is not preserved in an extended environment that supports the entire domain. The absence of a CD-flow here thus validates that X_1 is not accountable for the computation of Y in (7), since X_1 is never referenced during the computation of Y. But, X_2 CD-flows to Y, capturing a flow that is both supported and non-correlative.

B. Inductive Case: CD-trajectories

We now consider the construction of accountability evidence from a partial input variable W^1 to a partial output variable W^{n+1} in a cascade comprising a series of atomic channels.

1) CD-trajectories: A *trajectory* is defined to be an interleaved sequence of channels C_i and singleton variables W^i

$$\prec W^1 \succ \ C_1 \ \prec W^2 \succ \ C_2 \ \prec W^3 \succ \ \ldots \ C_n \ \prec W^{n+1} \succ \qquad (8)$$

where $\mathbb{C} \triangleq C_1 \ C_2 \ \ldots \ C_n$ is a cascade of atomic channels with an environment constrained by σ, and each W^i is among the output variables of C_{i-1} and is an input variable of C_i. If each W^i in trajectory (8) CD-flows to all its successor variables (i.e., $W^{i+1}, \ldots, W^{n+1}$) in that trajectory—and not only to its immediate successor (i.e., W^{i+1})—then we say the trajectory constitutes evidence that W^1 is accountable for W^{n+1} within environment σ.[7] Notice that, in this case, trajectory (8) satisfies requirements R1, R2, and R3 for accountability evidence. The obligation to show CD-flows between the W^i variables establishes that each is playing a non-trivial role in computing all its successors, including the cascade output. That satisfies R1. And the use of atomic channels and singleton variables means that there can be no decompositions, so R2 and R3 are trivially satisfied.

To check for the required CD-flows between intermediate variables in trajectory (8), we must determine the support predicate for every channel in the cascade $\mathbb{C}$. Predicate σ_i would support only values that reach C_i from initial inputs to $\mathbb{C}$ supported by σ. Equivalently, σ_i would support only those inputs that are outputs of cascade $C_1 \ldots C_{i-1}$ from inputs on C_1 supported by initial σ. We define this derived support for output Y of a given channel $C(X : \mathcal{X} \triangleright Y : \mathcal{Y})$ as follows

$$\sigma^C(y) \ \triangleq \ (\exists x \in \mathcal{X} : \ \sigma(x) \wedge C(x) = y).$$

Essentially, σ^C can be regarded as the strongest postcondition of σ under C. Since cascades are channels, we have that σ_i is $\sigma^{C_1 \ldots C_{i-1}}$.

We can now give a formal definition of CD-trajectories.

Definition: CD-trajectory.
Trajectory

$$\prec W^1 \succ \ C_1 \ \prec W^2 \succ \ C_2 \ \prec W^3 \succ \ \ldots \ C_n \ \prec W^{n+1} \succ$$

is a *CD-trajectory from* W^1 *to* W^{n+1}, for a cascade of atomic channels $\mathbb{C} \triangleq C_1 \ C_2 \ \ldots \ C_n$ with support σ on inputs to $\mathbb{C}$, and singleton variables W^i with $1 < i < n + 1$, iff

$$\forall i, j : \ 1 < i < j \leq n + 1 : \ W^i \underset{C_i \ldots C_j}{\overset{\sigma_i}{\Longrightarrow}} W^{j+1} \qquad (9)$$

holds, where σ_i is $\sigma^{C_1 \ldots C_{i-1}}$. ■

Notice that any segment

$$\prec W^i \succ \ C_i \ \ \prec W^{i+1} \succ \ \ldots \ C_{j-1} \ \prec W^j \succ$$

of CD-trajectory

$$\prec W^1 \succ \ C_1 \cdots \ \prec W^i \succ C_i \ \ \prec W^{i+1} \succ \ \ldots \\ C_{j-1} \ \prec W^j \succ \ \ \ldots \ C_n \ \prec W^{n+1} \succ$$

is also a CD-trajectory between intermediate singleton variables W^i and W^j.

To illustrate how CD-trajectories can be used to reason about accountability, consider a cascade constructed from channels C_1 and C_2:

$$\begin{array}{ll} & \{X_1 = X_2\} \\ C_1 : & W := X_2 - X_1; \quad Z := X_1 * 2 \\ C_2 : & Y := W + Z \end{array} \qquad (10)$$

[6] McLean [21] observes that a flow definition similar to VXM-flows is not able to distinguish between statistical correlations and causal relationships between input values and output values. He suggests that this problem "can be fixed by considering programs as inputs" to the flow definitions. With knowledge of the program, one can then deduce which variables contribute to the computation and which do not.

[7] The reason for requiring CD-flows to all successors, instead of only the immediate successor will become clear later in this section.

Here, $\prec X_1 \succ\ C_1\ \prec Z \succ\ C_2\ \prec Y \succ$ is a CD-trajectory. Therefore, X_1 is accountable for Y. There is no CD-trajectory that connects X_2 to Y, despite assignment $W := X_2 - X_1$ (i.e., description of output W in C_1). So, we have no evidence that X_2 is accountable for Y—as should be the case.

One alternative to using CD-trajectories as accountability evidence would be to have required CD-flows only between consecutive variables in the trajectory. This alternative definition does not satisfy R1, because the CD-flow relation is not transitive:

$$W^1 \underset{C_1}{\overset{\sigma_1}{\Longrightarrow}} W^2 \text{ and } W^2 \underset{C_2}{\overset{\sigma_2}{\Longrightarrow}} W^3 \tag{11}$$

does not imply that relation $W^1 \underset{C_1C_2}{\overset{\sigma_1}{\Longrightarrow}} W^3$ holds. An example is the following cascade:

$$\begin{array}{ll} C_1\colon & W := X_1 * 10 + (X_2 \ mod\ 10); \\ C_2\colon & Y := W \ mod\ 10; \end{array} \tag{12}$$

Here, we have $X_1 \underset{C_1}{\overset{\sigma}{\Longrightarrow}} W$ and $W \underset{C_2}{\overset{\sigma^{C_1}}{\Longrightarrow}} Y$ hold, where σ supports all initial states. However, relation

$$X_1 \underset{C_1;C_2}{\overset{\sigma}{\Longrightarrow}} Y$$

does not hold. Moreover, the value of X_1 is irrelevant for computing Y, so the alternative definition fails to satisfy R1. So, even though X_1 plays a non-trivial role in computing W and W plays a non-trivial role in computing Y, X_1 does not play a role in computing Y. Thus, the CD-flows in (11) are not sufficient for accountability evidence that satisfies R1.

2) Atomicity and Singleton Variables: Our definition assumed cascades of a restricted form—a composition of atomic channels and singleton variables. An obvious question is how this work generalizes to less restrictive forms. All the generalizations we considered turned out to violate requirements R2 and R3, though. Some examples illustrate.

A cascade that has non-atomic channels can be decomposed into cascades involving additional channels and, thus, additional intermediate variables. These decomposed cascades have CD-flow obligations that are not discharged by satisfying the obligations for a CD-flow with the original cascade.

We illustrate with the cascade below and σ that supports inputs where $W = Z$ holds:

$$\begin{array}{ll} & \{W = Z\} \\ C_1\colon & R := Z - W; \quad W := W; \\ C_2\colon & Y := W + R; \end{array} \tag{13}$$

Treating cascade $C_1; C_2$ as a (non-atomic) channel C, we have that Z CD-flows to Y. If we allowed CD-flows on non-atomic channels as accountability evidence, then we would deduce that Z is accountable for Y within channel C given in (13). Decompose channel C into cascade $\mathbb{C} \triangleq C_1C_2$. For Z to be accountable for Y, we would need to establish that either

$$\prec Z \succ\ C_1\ \prec W \succ\ C_2\ \prec Y \succ \tag{14}$$

or

$$\prec Z \succ\ C_1\ \prec R \succ\ C_2\ \prec Y \succ \tag{15}$$

is a CD-trajectory. However, (14) does not satisfy the obligations of a CD-trajectory, because Z does not CD-flow in C_1 to W, since Z does not VXM-flow in C_1 to W for extension $true$ of σ. Trajectory (15) does not satisfy the obligations of a CD-trajectory either, because Z does not CD-flow in C_1 to R, since Z does not VXM-flow in C_1 to R for σ. So, after refining non-atomic channel C into cascade $\mathbb{C}$, accountability evidence between Z and Y is absent. Accountability evidence as a CD-trajectory on non-atomic channels was not preserved under decomposition, and R2 is violated. In fact, there is no decomposition of cascade (13) such that Z becomes accountable for Y.

An alternative accountability definition we explored considers non-singleton variables. In this case, R3 is violated. That is, if $\prec X_i \succ\ C_1 \cdots C_i\ \prec W_I \succ\ C_{i+1} \cdots C_n\ \prec Y_j \succ$ is a CD-trajectory, then there might not exist a partial variable W_K of W_I with $K \subset I$, such that $\prec X_i \succ\ C_1 \cdots C_i\ \prec W_K \succ\ C_{i+1} \cdots C_n\ \prec Y_j \succ$ is also a CD-trajectory. We illustrate using example (13). Consider cascade C_1C_2 and CD-trajectory

$$\prec Z \succ\ C_1\ \prec W, R \succ\ C_2\ \prec Y \succ\ .$$

Notice that replacing $\prec W, R \succ$ with any singleton W or R would no longer result in a CD-trajectory. So, had we allowed accountability evidence with CD-trajectories to involve coarse-grain tuples as variables, then accountability evidence could be lost when considering partial variables: accountability evidence of compound variables could not be substantiated with partial variables. R3 is thus violated.

A disadvantage of reasoning about accountability at the level of atomic channels and singleton variables is that designers usually reason about system behavior at a higher level of abstraction. But, as we saw above, staying at a higher level of abstraction might lead to conservative accountability claims with respect to our definition.

Putting it together, formulating a definition for accountability involved asking at least the following design questions:

(A) Whether accountability should be defined intensionally or extensionally.
(B) Whether accountability should be preserved under channel decomposition.
(C) Whether accountability should be preserved under variable decomposition.

In our work, accountability is defined intensionally, since it depends on the implementation of a system (i.e., a cascade)—it does not consider only the input-output relation of the system. This means that changes in the modeled details of the system might alter the accountability judgments. Such changes include channel decomposition, which is a form of implementation refinement where system components are modeled with additional implementation details. Replacing a command written in a high-level programming language (e.g., Python) with code written in a low-level programming language (e.g., assembly) that implements that command, is an example of implementation refinement.

Another form of implementation refinement is variable decomposition. Replacing a variable that stores an integer with

a vector of variables that store the binary representation of that integer is an example of variable decomposition. One would expect that accountability depends on the granularity at which a system is modeled. Our thesis is that accountability should not be altered by modeling more implementation details. Rather, accountability should be preserved under channel decomposition and variable decomposition. Once an entity is held accountable at higher level of abstraction, this accountability should not be lost when migrating to lower levels of abstraction. Equivalently, accountability evidence at a high level should be substantiated by some accountability evidence at low level. Accountability evidence is thus not an emergent property. This prompted us to constrain the definition of accountability to be in terms of atomic channels and singleton variables, where both cannot be further decomposed because implementation details are unknown. Future research might investigate alternative design choices for answering (A)–(C) when defining accountability.

C. From CD-trajectories to Involvement

Determining whether some system component affects the flow from the input to the output of that system can be useful when analyzing whether a system satisfies an integrity policy. Consider, the cascade with input X and output Y:

$$\begin{array}{ll} C_1\colon & W \coloneqq f(X);\ Z \coloneqq X; \\ C_2\colon & Y \coloneqq W + Z \end{array} \tag{16}$$

An integrity policy might specify that if result W produced by untrusted function f affects the flow of trusted input X to the cascade output Y, then Y should be considered untrusted. This integrity policy is a deprecation policy since the output's level of integrity should be decreased to untrusted, even though Y is computed based on trusted X.

To analyze whether program (16) satisfies the integrity policy above, the term "affects" needs to be formalized. An intermediate variable *affects* the flow of input values to a cascade output iff that variable has non-trivial involvement in the computation of the cascade output from a cascade input. We call such a variable an *involved variable*. CD-trajectories identify involved variables, as the following definition formalizes.

Definition: Involvement and Flow-Through.
Partial intermediate variable W^i of a cascade $\mathbb{C}$ is called an *involved variable* and the *flow-through relation* $W^1 \overset{W^i}{\rightsquigarrow} W^{n+1}$ is established iff there exists a CD-trajectory

$$<W^1>\ C_1\ <W^2>\ \ldots C_{i-1}\ <W^i>\ \ldots\ C_n\ <W^{n+1}>\ .\quad \blacksquare$$

By definition, if W^i is an involved variable for the flow between X and Y, then X is accountable for W^i and W^i is accountable for Y.

Returning to cascade $C_1\ C_2$ in (16), $X \overset{W}{\rightsquigarrow} Y$ holds, since there exists CD-trajectory $<X>\ \ C_1\ \ <W>\ \ C_2\ \ <Y>$. So, since X flows to Y through the output W of untrusted f, output values stored in Y should then be considered untrusted, based on the integrity policy specified above. Notice that the above definition of flow-through is sensitive to semantics—if, for example, command $W \coloneqq f(X)$ is replaced in (16) with $W \coloneqq f(X) * 0$, then X no longer flows to Y through W, and thus, Y is not considered untrusted by the integrity policy. This example shows that the notion of flow-through and the notion of involved variable are useful for expressing and enforcing policies where the sensitivity of values should change because a flow was partially intercepted by a given component in a system.

It might seem that the flow-through relation could be defined in terms of two CD-flow relations—from the input to the involved variable and from that involved variable to the output (instead of requiring all of the additional CD-flows needed to establish a CD-trajectory). Such a definition would be conservative, as the following example illustrates. Assuming σ supports all integer values, input Z CD-flows in C_1 to intermediate variable H_2 (i.e., $Z \overset{\sigma}{\underset{C_1}{\Longrightarrow}} H_2$), and H_2 CD-flows in $C_2\ C_3$ to output W (i.e., $H_2 \overset{\sigma^{C_1}}{\underset{C_2\ C_3}{\Longrightarrow}} W$).

$$\begin{array}{ll} C_1\colon & H_1 \coloneqq Z;\ H_2 \coloneqq Z; \\ C_2\colon & Y \coloneqq H_1 - H_2; \quad H_1 \coloneqq H_1; \\ C_3\colon & W \coloneqq \langle H_1, Y \rangle; \end{array} \tag{17}$$

Thus, the two CD-flows are present to satisfy the alternative definition. However, in (17) Z does not flow to W through H_2, since H_2 is only used to compute Y, which is always 0 during execution. Thus, H_2 does not contribute to output W. So, the proposed alternative definition conservatively would assert that $Z \overset{H_2}{\rightsquigarrow} W$ holds, even though H_2 does not contribute to output W. The flow-through definition given above did correctly conclude that H_2 is not an involved variable, because there is no CD-trajectory from Z to W that involves H_2.

The flow-through relation is not preserved when a cascade is channel-decomposed or variable-decomposed. When the implementation of channels in a cascade is decomposed, the CD-flow relations might change (recall the discussion of example (13)), and thus, the CD-trajectories that are formed might change, causing the involved variables to change, too. A variable characterized as involved for an abstract model of the system might not still be considered involved under a refined model. Conversely, an involved variable that has been identified at the most refined model might be hidden when working with a more abstract model. To avoid such changes, it suffices to establish involved variables for cascades built with atomic channels and singleton variables.

V. Mediation

A security engineer often will use mediation to limit the propagation of a secret or an untrusted value. One common way to implement mediation is for a trusted process to perform blocking or filtering. Viewed through this lens, a *downgrader* is a trusted process that uses filtering to convert secret information into public information or to convert untrusted information into trusted information. A general definition of mediation *per se* has not appeared in the literature, though several definitions are plausible.

In this section, we formally define mediation for systems modeled as cascades. Several candidate definitions are given; they differ in the kind of control they exert. So having the formal definitions allows rigorous comparison of different kinds of mediation, exposing conditions under which various form of mediation for blocking and mediation for filtering are related. Our formalizations also allow exploration of additional properties that mediation might satisfy: compositionality and support for accountability.

The starting point for our definitions of mediation is a statement of possible desired effects of mediation on the input-output relation of a cascade.

- *Blocking*: Because the mediation mechanism is present, the input to the cascade no longer IM-flows to the output of the cascade.
- *D-Filtering*: Because the mediation mechanism is present, the set of possible output values of the controller cascade is a strict subset D of the possible output values for the original cascade.

Blocking and D-filtering are achieved by (i) selecting a location within the cascade where the mediation mechanism will be inserted, and (ii) choosing appropriate semantics for that mediation mechanism. For (i), we designate a *mediation point* by giving the name of an intermediate variable between two channels C_i and C_{i+1} of a cascade. For (ii), a suitable *controller channel* is inserted between C_i and C_{i+1}. The controller channel blocks or filters some partial intermediate variables, which comprise the mediation point, in a way that blocks or filters the information flow from the input to the output of the cascade.

A. Controller Channels

To formalize the idea of a controller channel, notice first that every cascade

$$X\ C_1 \cdots C_i\ \ W\ \ C_{i+1} \cdots C_n\ Y$$

with two or more channels and an intermediate variable W can be modeled as a two-channel cascade

$$\mathbb{C} \triangleq\ X\ C_a\ W\ C_b\ Y \tag{18}$$

where $C_a \triangleq C_1 \cdots C_i$ and $C_b \triangleq C_{i+1} \cdots C_n$. Suppose $\mathbb{C}$ has a support predicate σ, such that X IM-flows to Y. We want to examine whether altering the value in a partial variable W_I would control the IM-flow from X to Y.

To specify that control, a new channel $C_{W_I}(W' : \mathcal{W} \triangleright W : \mathcal{W})$ is interposed between C_a and C_b to form cascade

$$\mathbb{C}_{W_I} \triangleq\ X\ C_a\ W'\ C_{W_I}\ W\ C_b\ Y \tag{19}$$

where W' is a fresh variable with the same number of singleton partial variables as W in $\mathbb{C}$, and taking values from the same domain $\mathcal{W}$. W' is said to be a *mirror* variable for W.

Mediation is achieved if C_{W_I} is *W_I-controller*, meaning that it propagates to its output all of its inputs, except for the values of intermediate variable W_I:

$$(\forall u \in \mathcal{W}:\ C_{W_I}(u) =_{\overline{I}} u)$$

Cascade $\mathbb{C}_{W_I}$ in (19) will be called a *W_I-controller* cascade, because the partial variable W_I that C_{W_I} outputs can control variation in the outputs of $\mathbb{C}_{W_I}$.

The choice of semantics for C_{W_I} allows the information flow from X to Y to be controlled in different ways. We illustrate those differences by considering the following cascade $\mathbb{C}$ in an environment σ that supports variables storing real numbers:

$$\mathbb{C}:\ \left[\begin{array}{ll} C_1: & Z := X; \\ C_2: & Q := 1;\ Z := Z; \\ C_3: & Y := Z * Q; \end{array}\right. \tag{20}$$

Here, X IM-flows to Y. Moreover, if we consider output variable Z of C_2 as a mediation point and we interpose between C_2 and C_3 a Z-controller channel

$$C_Z:\ Q := Q;\ Z := 3$$

(i.e., a channel that always assigns 3 to its output variable Z and leaves the values of other variables intact) at that point, then X no longer IM-flows to Y. So, intermediate partial variable Z blocks the IM-flow for the new cascade $\mathbb{C}_Z = C_1\ C_2\ C_Z\ C_3$.

The example illustrates that assigning a constant value to an intermediate variable W_I can prevent variation in the input to a cascade from causing variation in the output of that cascade. A W_I-controller channel C_{W_I} that outputs a constant $v \in \mathcal{W}_I$ in variable W_I and, therefore, is characterized by

$$(\forall u \in \mathcal{W}:\ C_{W_I}(u) =_{\overline{I}} u\ \wedge\ C_{W_I}(u) =_I v)$$

will be called *W_I/v-assignment* channel. So C_Z inserted into (20) is a $Z/3$-assignment channel.

A controller channel not only can suppress values but it also can filter values. For example, suppose we want to ensure that cascade (20) only produces output values Y that are integers. We might incorporate a *sanitizer* into $\mathbb{C}$ by interposing Z-controller channel

$$C'_Z:\ Q := Q;\ Z := Int(Z) \tag{21}$$

between C_2 and C_3. Here Int is a function from input values in Z to integers. In general, we might employ a W_I-controller channel that outputs $f(W_I)$ for some function $f : \mathcal{W}_I \to \mathcal{W}_I$, instead of outputting a specific constant:

$$(\forall u \in \mathcal{W}:\ C_{W_I}(u) =_{\overline{I}} u\ \wedge\ C_{W_I}(u) =_I f(u_I)).$$

C_{W_I} will be called *W_I/f-filtering channel*. For example, C'_Z of (21) is a Z/Int-filtering channel. If a W_I/f-filtering channel is inserted between two channels where W_I is an intermediate variable, we may simply say that "f is placed at W_I". Notice that f has a local effect to the specific intermediate variable W_I, and thus, it does not introduce flows between W_I and its complement $W_{\overline{I}}$.

There is good reason for a W_I-controller channel C_{W_I} within a cascade to output values that are among the inputs of C_{W_I}, since C_{W_I} then forwards values that its predecessor C_a could have produced during execution of the original cascade $\mathbb{C}$ within environment σ. A channel C_{W_I} is *support-conserving*

for $\mathbb{C}$ and σ provided that $\sigma^{C_a C_{W_I}} \Rightarrow \sigma^{C_a}$ holds. C_Z that we inserted into (20) is support-conserving, since the output that $C_1C_2C_Z$ produces (i.e., $\langle Q = 1, Z = 3\rangle$) is one of the possible outputs that C_1C_2 produces. If W_I can be assigned v when cascade $\mathbb{C}$ runs on some input supported by σ, then we say that this value v is *W_I-reachable* for $\mathbb{C}$ and σ. So, if a W_I/v-assignment channel C_{W_I} is support-conserving for $\mathbb{C}$ and σ, then, by definition, v is W_I-reachable for $\mathbb{C}$ and σ. Returning to (20) and C_Z, we see that 3 is Z-reachable.

B. Plausible Definitions for Mediation

To achieve blocking—and thus, prevent IM-flows from the input to the output of a cascade $\mathbb{C}$—an obvious approach is to identify an intermediate variable in $\mathbb{C}$ that should be set to a fixed value.

Definition: $\exists^*$-blocking point.
Given is cascade $\mathbb{C} \triangleq X\ C_a\ W\ C_b\ Y$ and support predicate σ. Partial intermediate variable W_I is an *$\exists^*$-blocking point* iff

(i) X IM-flows to Y in cascade $\mathbb{C}$,
(ii) there exists W_I/v-assignment channel C_{W_I} such that X does not IM-flow to Y in $\mathbb{C}_{W_I} \triangleq X\ C_a\ W'\ C_{W_I}\ W\ C_b\ Y$, where W' is a mirror variable for W. ■

Treating (20) as a two-channel cascade, where $C_a = C_1C_2$ and $C_b = C_3$, we see that Q is an $\exists^*$-blocking point: if Q is given fixed value 0, then X no longer IM-flows to Y. Here, we add superscript $*$ to existential $\exists$, because the witness value v that satisfies the existential can be any value from the domain (e.g., v does not have to be reachable).

To achieve D-filtering—and thus, restrict the set of possible cascade outputs to a subset D—we might devise a sanitizing-function San that maps vectors of singleton values that comprise some selected intermediate partial variable W_I into vectors of values in D and insert a W_I/San-filtering channel C_{San} between C_a and C_b in cascade (18). To preserve and not destroy the results of filtering, such an approach presupposes that channel C_b is *D-preserving*: singleton inputs values in D cause the channel (i.e., C_b) to output (possibly different) singleton values in D. Inputs that are not in D may produce arbitrary outputs.

Definition: San-filtering point.
Given is cascade $\mathbb{C} \triangleq X\ C_a\ W\ C_b\ Y$, support predicate σ, and function San that maps a tuple of singleton values to a tuple of values in D. Partial intermediate variable W_I is a *San-filtering point* iff

(i) X IM-flows to Y for $\mathbb{C}$ and σ,
(ii) for some W_I/San-filtering channel C_{San} and mirror variable W' for W, each output value of Y of $\mathbb{C}_{San} \triangleq X\ C_a\ W'\ C_{San}\ W\ C_b\ Y$ is in D:

$$(\forall x \in \mathcal{X}^\sigma\colon\ \mathbb{C}_{San}(x) \in D)$$ ■

For example, returning to (20) and (21), we have that intermediate variable Z is an Int-filtering point. Notice that if the definition of San is extended to be a function that takes as input all the variables of a channel and returns a constant value, then San creates a blocking point.

An intermediate variable W_I that is an $\exists^*$-blocking point is not necessarily a San-filtering point. Example (20) illustrates. Here we examine cascade $X\ C_1C_2\ \langle Q, Z\rangle\ C_3\ Y$. Assume the input values to the cascade are the real numbers. Consider a sanitizer Int that converts all real numbers into integers. We have that Q is a $\exists^*$-blocking point, because when Q is set to 0, Y remains 0 and thus there is no flow from X to Y. But Q is not an Int-filtering point, because if we place the sanitizer at Q, then Y is not always sanitized: when $X = 2.1$, we have $Z = 2.1$, and ultimately $Y = 2.1$.

A San-filtering point will not necessarily be an $\exists^*$-blocking point, either. Consider the following example:

$$\mathbb{C}\colon \left[\begin{array}{ll} C_1\colon & Z := X;\ X := X \\ C_2\colon & Z := Z;\ Q := Int(X) \\ C_3\colon & Y := Z + Q; \end{array}\right. \tag{22}$$

Here, Z (either between C_1 and C_2, or between C_2 and C_3) is an Int-filtering point, because if we place Int at Z, then all outputs of cascade (22) will be integers. However, Z is not an $\exists^*$-blocking point, since no fixed value for Z blocks the information flow from input X to output Y.

$\exists^*$-blocking points allow a form of out-of-band signaling, since an arbitrary value is used to suppress variation of the output of the cascade. A form of in-band signaling occurs if variation in the output is suppressed by a value of W_I that could have been produced by an input to cascade $\mathbb{C}$. A formal definition of such mediation is obtained from the definition of $\exists^*$-blocking by restricting the scope of the existential in (ii) and requiring v to be W_I-reachable.

Definition: $\exists$-blocking point.
Given is cascade $\mathbb{C} \triangleq X\ C_a\ W\ C_b\ Y$ and support predicate σ. Partial intermediate variable W_I is an *$\exists$-blocking point* iff,

(i) X IM-flows to Y for cascade $\mathbb{C}$,
(ii) there exists W_I/v-assignment channel C_{W_I}, where v is a W_I-reachable constant for $\mathbb{C}$ and σ, such that X does not IM-flow to Y in

$$\mathbb{C}_{W_I} \triangleq X\ C_a\ W'\ C_{W_I}\ W\ C_b\ Y,$$

where W' is a mirror variable for W. ■

To illustrate $\exists$-blocking, consider the following example, where σ supports real input values:

$$\mathbb{C}\colon \left[\begin{array}{ll} C_1\colon & Z := X;\ X := X \\ C_2\colon & Z := Z;\ Q := X \\ C_3\colon & Y := Z * Q; \end{array}\right. \tag{23}$$

Here, X IM-flows to Y. We see that Q is an $\exists$-blocking point: if Q is set to reachable value 0, then the IM-flow is blocked. Still, Q is not a Int-filtering point. So, $\exists$-blocking points are not necessarily San-filtering points.

Some blocking points also are San-filtering points. These are intermediate variables where, for *any* W_I-reachable constant value they might store, the flow from the input to the

output of the cascade is blocked. The following definition formalizes this situation.

Definition: $\forall$-blocking point.
Given is a cascade $\mathbb{C} \triangleq X\ C_a\ W\ C_b\ Y$ and support predicate σ. W_I is a $\forall$-*blocking point* iff

(i) X IM-flows to Y for $\mathbb{C}$ and σ, and
(ii) for any W_I/v-assignment channel C_{W_I}, where v is a W_I-reachable constant for $\mathbb{C}$ and σ, C_{W_I} is support-conserving for $\mathbb{C}$ and σ, and X does not IM-flow to Y for σ and $\mathbb{C}_{W_I} \triangleq X\ C_a\ W'\ C_{W_I}\ W\ C_b\ Y$, where W' is a mirror variable for W. ■

For example, considering (20), where C_a is instantiated with C_1C_2 and C_b is instantiated with C_3, we have that Z is a $\forall$-blocking point. We use relation $X \xrightarrow{W_I}\!\!\!\!\!\!| \; Y$ to denote that W_I is a $\forall$-blocking point for the flow from X to Y. Notice that $X \xrightarrow{W_I}\!\!\!\!\!\!| \; Y$ implies that

$$W_{\overline{I}} \text{ does not IM-flow to } Y. \tag{24}$$

Notice also that the support-conserving requirement imposed for any reachable constant v assigned to W_I implies that

$$W_I \text{ is independent of } W_{\overline{I}}. \tag{25}$$

From (24) and (25), we can conclude that W_I, being a $\forall$-blocking, functions as a "choke point" for the information flow from X to Y.

Relating mediation points: The blocking point definitions presented in this section form a hierarchy:

Theorem 2. *Given cascade $\mathbb{C} \triangleq X\ C_a\ W\ C_b\ Y$ and support predicate σ:*

(i) If W_I is a $\forall$-blocking point, then W_I is an $\exists$-blocking point.
(ii) If W_I is an $\exists$-blocking point, then W_I is an $\exists^*$-blocking point. ■

The converses of the statements given in theorem 2 do not hold.

Using theorem 2, one can easily deduce that a San-filtering point does not satisfy any of the blocking point definitions of this section. Recall our earlier conclusion that if a variable W is a San-filtering point, then W is not necessarily an $\exists^*$-blocking point (see discussion of (22)). And if W is not an $\exists^*$-blocking point, then, using the contrapositive of the implications in theorem 2, we also get that W is neither an $\exists$-blocking point nor a $\forall$-blocking point.

If some of the supported values of $W_{\overline{I}}$ are in D (i.e., $D \cap \mathcal{W}_{\overline{I}}^{\sigma^{C_1}} \neq \varnothing$), then a $\forall$-blocking point is also a San-filtering point:

Theorem 3. *Consider support predicate σ and cascade $\mathbb{C} \triangleq X\ C_1\ W\ C_2\ Y$, where X, W_I, $W_{\overline{I}}$, and Y are singleton variables. If W_I is a $\forall$-blocking point, channel C_2 is D-preserving, and $D \cap \mathcal{W}_{\overline{I}}^{\sigma^{C_1}} \neq \varnothing$ holds, then W_I is a San-filtering point, for any function San that maps singleton values to D and makes the W_I/San-filtering channel support-conserving for $\mathbb{C}$ and σ.* ■

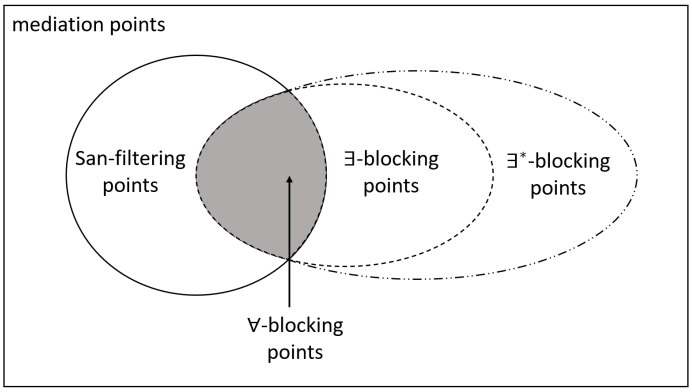

Fig. 3. Sets of mediation points according to Theorems 2 and 3.

This theorem connects blocking with filtering. It implies that $\forall$-blocking points can guide the correct placement of filters, such as downgraders, within a cascade. If W_I is a $\forall$-blocking point in a cascade, and if the downgrader is inserted between W_I and the next channel in the cascade, then any input to the cascade will be downgraded before reaching the output of that cascade.

Requiring at least one execution of the cascade in σ to yield a value for $W_{\overline{I}}$ in D (i.e., $D \cap \mathcal{W}_{\overline{I}}^{\sigma^{C_1}} \neq \varnothing$) is crucial for this theorem to hold, and thus, for blocking to imply filtering. If $W_{\overline{I}}$ never takes a value in D, then the hypothesis of C_2 being D-preserving is never exercised, and thus, a $\forall$-blocking point is not necessarily a San-filtering point. To illustrate, consider a cascade $X\ C_1\ W\ C_2\ Y$ and support predicate σ such that the $W_{\overline{I}}$ input of C_2 is always a constant value in $\overline{D}$. Assume also that channel C_2 outputs (possibly different) values in $\overline{D}$ when at least one of its inputs is in $\overline{D}$. Thus, Y always takes values in $\overline{D}$. Here, W_I can be a $\forall$-blocking point for the flow from X to Y. But placing a San filter at W_I will not result in filtering Y. This is because for the cascade $X\ C_1\ W'\ C_{San}\ W\ C_2\ Y$, $W_{\overline{I}}$ will still be taking a constant value from $\overline{D}$, and thus C_2 will still output a value in $\overline{D}$.

We earlier saw that the other blocking definitions ($\exists$-blocking and $\exists^*$-blocking) do not indicate the correct placement of sanitizer San in the cascade. And since a downgrader is an instance of a sanitizer, these other blocking definition are not appropriate mitigation points. Involved variables are not appropriate for guiding the placement of sanitizers, either. This is because involved variables intercept part—not the entire—flow of information from an input to an output.

Summarizing, we have shown that mediation points for filtering in a system are not necessarily mediation points for blocking, and vice versa. So, in general, mediation points that enable filtering cannot be repurposed to enable blocking, and vice versa. But there is an exception: a $\forall$-blocking point can be a mediation point for filtering, too. Thus, the semantics of a $\forall$-blocking point can be used to justify the correct placement of a filter within a system. Figure V-B gives a Venn diagram that relates the sets of mediation points according to theorems 3 and 2.

Structural properties of ∀-blocking: A ∀-blocking point satisfies two interesting structural properties. The following theorem shows the first—that ∀-blocking points satisfy a form of compositionality: the ∀-blocking point of a ∀-blocking point is a ∀-blocking point.

Theorem 4. *For a cascade* $\mathbb{C} \triangleq X\ C_1\ W\ C_2\ Z\ C_3\ Y$ *and support predicate* σ*, if* $X \xrightarrow{Z_I}\!\!\!\!\blacktriangleright\!\!| Y$ *holds for* $\mathbb{C}$*, and* $X \xrightarrow{W_J}\!\!\!\!\blacktriangleright\!\!| Z_I$ *holds for* C_1C_2*, then* $X \xrightarrow{W_J}\!\!\!\!\blacktriangleright\!\!| Y$ *holds for* $\mathbb{C}$*.* ∎

The other definitions of blocking points do not exhibit this kind of compositionality.

The second structural property is that singleton variables that are ∀-blocking points can be used to construct CD-trajectories, thus making these singleton variables involved variables, too. Other blocking points do not necessarily satisfy this property.

Theorem 5. *Consider support predicate* σ *and a cascade*

$$X\ C_1\ Z^1\ C_2\ Z^2 \cdots Z^{n-1}\ C_n\ Y\ ,$$

where X *IM-flows to* Y *and* W^j *is a singleton partial variable of* Z^j*, and thus, an input to channel* C_{j+1}*. If* $X \xrightarrow{W^{n-1}}\!\!\!\!\blacktriangleright\!\!| Y$*,* $X \xrightarrow{W^{n-2}}\!\!\!\!\blacktriangleright\!\!| W^{n-1}$*, ..., and* $X \xrightarrow{W^1}\!\!\!\!\blacktriangleright\!\!| W^2$ *hold in cascade* $\mathbb{C}$*, then*

$$\begin{aligned} &\prec X \succ\ C_1\ \prec W^1 \succ\ C_2\ \prec W^2 \succ\ \dots \\ &\qquad \prec W^{n-2} \succ\ C_{n-1}\ \prec W^{n-1} \succ\ C_n\ \prec Y \succ \end{aligned}$$

is a CD-trajectory. ∎

A question that remains open is whether any variable—rather than singleton variables—that is ∀-blocking point can serve as an involved variable.

VI. Related Work

We appear to be the first to give information flow accounts for accountability, involvement, and mediation. Our work, however, builds on prior work both within and outside Computer Science.

A. Accountability

Accountability is connected to causality. And researchers in many disciplines have explored counterfactual arguments to capture causality (e.g., [1], [27], [18], [20]). Claiming that standard causal models cannot support constraints on variables, Beckers *et al.* [4] extend standard causal models to support causality claims where initial constraints create dependencies between variables. The resulting causality relations are analogous to our supported flows. Similar to VXM, the causality relation Beckers *et al.* propose reports correlative relations that are induced by the constraints. These correlative relations are labelled "non-causal" in that work, but this term is not formalized there.

Causality reasoning has been employed to solve accountability problems in systems. Datta *et al.* [11] investigate the problem of identifying the events that were the *actual cause* of a fault in a system. Tschantz *et al.* [29] employ counterfactual arguments to estimate whether web data has been used by certain applications and, thus, whether this data is accountable for the values that these applications compute. Both approaches do not consider correlations induced by the environment.

Audit logs are often employed to achieve deterrence through accountability. Intrusion detection schemes (e.g, [12]) employ anomaly detection methods on audit logs to find user actions that deviate from patterns previously deemed normal. These audit logs then function as evidence for holding some users accountable for identifying intrusion. Causality analysis has also been applied to audit logs for identifying causal paths that point to the root cause of an attack. For example, Kwon *et al.* [17] use counterfactuals to establish causal relations between events recorded in an audit log. Complications that arise from correlations of data and the non-transitivity of causal relations were not discussed in that work.

Correlations between data has been extensively studied in connection with relational databases. Here, a *functional dependency* (e.g., [19], [14]) captures how the values of an attribute set uniquely determine the value of another attribute. For example, the country attribute uniquely determines the currency attribute. Functional dependencies are similar to our notion of a support predicate that captures correlations among initial values of variables. In databases, methods have been proposed to discover functional dependencies in order to improve schema normalization and database redesign. In our work, the correlations are given—not discovered. Accountability is then defined based on these correlations and all possible relaxations.

B. Mediation

Mediation has mostly been studied in the literature of causality. Soloviev *et al.* [26] use "mediated causation" to indicate total control between a cause and an effect. A *cause* is an assignment of a value to a variable, and an *effect* is an event in the system. So, a variable W mediates causation if by fixing W to a reachable value then the effect is no longer seen. This control of W is closely related to our ∃-blocking. However, ∃-blocking intercepts the flow of information between variables—it does not intercept the influence that a specific value assignment has to a system event. Soloviev *et al.* [26] also discuss causal graphs, where paths conservatively approximate causal relations between variables. Those paths are conservative approximations of our CD-trajectories.

Pear [23] asserts that mediation analysis (e.g., [23], [28], [24]) "aims to uncover causal pathways along which changes are transmitted from causes to effects". Knowledge of such pathways enhance an understanding about how subsequent variables contribute to building a causal claim. So, these causal pathways are addressing the same idea as CD-trajectories and, thus, the variables referenced in those causal pathways are similar in spirit to involved variables. Pear [23] does not identify variables that are interconnected with non-correlative relations, whereas CD-trajectories do identify these variables. Also, the mediation analysis of Pear [23] does not handle correlations on variables and the effects of these correlations on the formation of causal pathways.

C. Information Flow Definitions

Most of the literature in information flow control concerns the absence of flow [15]—what it means for an input not to flow to an output of a program. Our work is concerned with the presence of flow. In approaches that do propose definitions for existence of flow, correlations between inputs have not been adequately addressed (if at all). Here we describe such approaches.

Our theory of accountability and mediation uses modulation to create counterfactual arguments. Modulation is employed by Cohen [9] to characterize what he terms *strong dependency*. This information flow definition incorporates constraints imposed on inputs. Similar to VXM-flows, Cohen extends a partial input into one that is independent from the rest of the input. He uses the terms "clumps" and "pseudo-objects" to refer to these extensions, but formalizations are not provided. Our VXM-flow definition formalizes such extensions and requires them to be minimal. As with VXM-flow, strong dependency might report correlative flows.

IM-flow, VXM-flow, CD-flow, and strong dependency address a qualitative question: whether there is an information flow from one entity to another. Researchers in quantitative information flow (QIF) are concerned with a different question: how much information flows from one entity to another. Alvim *et al.* [2] survey research advances in QIF, analyzing different QIF definitions, including Shannon's theory. These definitions are not based on modulation; rather, they are based on differences in the probability distributions for a secret before and after observing an output. This formalism usually considers a channel with one input and one output; initial correlations across multiple inputs has not been a concern for QIF researchers. However, Alvim *et al.* [2] does discuss the *Dalenius perspective* [6], where a secret outside the modeled system might be correlated with the actual input to the corresponding channel. Thus one can calculate the amount of information that is leaked from that outside secret to the output of the channel. The Dalenius perspective is an example where information about an outside secret S can be inferred by observing an output O. Yet, based on the understanding that our paper builds, S is not accountable for O since S, being outside the system, has not been used in computing O.

Extensive research in Computer Science has concerned analyzing programs for identifying information flows. A number of tools have been proposed to approximate the quantity of information that a program might leak about its secret inputs (e.g., [3], [5], [13], [8], [7]). LDX [16] use alternative executions to build counterfactual arguments for establishing causality relations between variables at different program points. However, LDX does not handle cases where inputs are correlated.

ACKNOWLEDGEMENTS

The content of this paper has been greatly benefited by extensive discussions with Marius Ingebrigtsen. We would also like to thank the reviewers for giving insightful comments that improved our formalism and presentation.

APPENDIX

A. Proofs of Theorems

Theorem 1. *$FC(I{\uparrow}\sigma, \mathcal{X}^\sigma)$ holds.* ■

Proof. Consider $J, K \supseteq I$ and $L = J \cap K$. It suffices to show that if $FC(J, \mathcal{X}^\sigma)$ and $FC(K, \mathcal{X}^\sigma)$ hold, then $FC(L, \mathcal{X}^\sigma)$ holds, too. Let $x, x' \in \mathcal{X}^\sigma$. By the definition of $FC(J, \mathcal{X}^\sigma)$, we then have $x \parallel_J x' \in \mathcal{X}^\sigma$. By the definition of $FC(K, \mathcal{X}^\sigma)$, we then have $(x \parallel_J x') \parallel_K x' \in \mathcal{X}^\sigma$. Because $(x \parallel_J x') \parallel_K x' = x \parallel_L x'$, we then get $x \parallel_L x' \in \mathcal{X}^\sigma$. Thus $FC(L, \mathcal{X}^\sigma)$ holds. □

Theorem 2. *Given cascade $\mathbb{C} \triangleq X\ C_1\ W\ C_2\ Y$ and support predicate σ, the following logical implications hold:*

(i) If W_I is a $\forall$-blocking point, then W_I is an $\exists$-blocking point.

(ii) If W_I is an $\exists$-blocking point, then W_I is an $\exists^*$-blocking point. ■

Proof. We first prove (i). Let W_I be a $\forall$-blocking point for cascade $\mathbb{C}$ with support predicate σ. This means that

(a) X IM-flows to Y for $\mathbb{C}$ and σ, and

(b) for any W_I/v-assignment channel C_{W_I}, where v is a W_I-reachable constant for $\mathbb{C}$ and σ, C_{W_I} is support-conserving for $\mathbb{C}$ and σ, and X does not IM-flow to Y for σ and $\mathbb{C}_{W_I} \triangleq X\ C_1\ W'\ C_{W_I}\ W\ C_2\ Y$.

If $\mathbb{C}$ is executed, then there will be a value for intermediate variable W for some supported input to the cascade. Therefore, there will be a W_I-reachable value stored in partial variable W_I. Let this value be v'. Based on v', we can construct W_I/v'-assignment channel C'_{W_I}. By instantiating in (b), C_{W_I} with C'_{W_I}, we get that X does not IM-flow to Y for σ and $\mathbb{C}'_{W_I}$. From (a), we then get that W_I is an $\exists$-blocking point.

We now prove (ii). By definition, for a variable to be an $\exists$-blocking point, it should satisfy all requirements imposed by an $\exists^*$-blocking point and the requirement of the witness value to be reachable. So, if a variable is an $\exists$-blocking point, then it is also an $\exists^*$-blocking point.

□

Theorem 3. *Given is support predicate σ and cascade $\mathbb{C} \triangleq X\ C_1\ W\ C_2\ Y$, where X, W_I, $W_{\overline{I}}$, and Y are singleton variables. If W_I is a $\forall$-blocking point, channel C_2 is D-preserving, and $D \cap \mathcal{W}_{\overline{I}}^{\sigma\,C_1} \neq \varnothing$ holds, then W_I is a San-filtering point, for any function San that maps singleton values to D and makes the W_I/San-filtering channel support-conserving for $\mathbb{C}$ and σ.* ■

Proof. To prove that W_I is a San-filtering point, we prove that:

(i) X IM-flows to Y for $\mathbb{C}$ and σ,

(ii) for W_I/San-filtering channel C_{San}, each output value of Y of $\mathbb{C}_{San} \triangleq X\ C_1\ W'\ C_{San}\ W\ C_2\ Y$ is in D:

$$(\forall x \in \mathcal{X}^\sigma \colon \mathbb{C}_{San}(x) \in D)$$

We get (i) from the hypothesis that W_I is a $\forall$-blocking point.

We now prove (ii). Assume for contradiction that for some input $x \in \mathcal{X}^\sigma$, we have

$$\mathbb{C}_{San}(x) \notin D. \tag{26}$$

Let $w' = C_1(x)$ and

$$w^a = C_{San}(w'). \tag{27}$$

From (26), we then have

$$C_2(w^a) \notin D. \tag{28}$$

Because C_2 is D-preserving, and because $w^a_I \in D$, by the construction of C_{San}, (26) also gives $w^a{}_{\overline{I}} \notin D$.

Let $\sigma_1 = \sigma^{C_1}$ and $w'' \in \mathcal{W}^{\sigma_1}$ where $w''_{\overline{I}} \in D$. Such w'' exists, due to hypothesis $D \cap \mathcal{W}^{\sigma_1}_{\overline{I}} \neq \varnothing$. Let also

$$w^b = C_{San}(w''). \tag{29}$$

We show that $w^b{}_{\overline{I}} \in D$ holds. By the definition of C_{San} and because C_2 is D-preserving and $w''_{\overline{I}} \in D$, we then have that

$$C_2(w^b) \in D \tag{30}$$

and

$$w^b{}_I \in D \ \wedge \ w^b{}_{\overline{I}} \in D. \tag{31}$$

By construction, we have that w' and w'' are supported by σ^{C_1}. From property (ii) of the $\forall$-blocking point definition, the hypothesis that W_I is a $\forall$-blocking point implies that for all W_I-reachable constants v, the corresponding W_I/v-assignment channel C_{W_I} is support-conserving. This implies that W_I is independent (see (4)) of $W_{\overline{I}}$ for the set of values supported by σ^{C_1}. We then get that there exists w''' supported by σ^{C_1} such that

$$w''' =_I w' \tag{32}$$

and

$$w''' =_{\overline{I}} w''. \tag{33}$$

Let

$$w^c = C_{San}(w'''). \tag{34}$$

Given that W_I is a $\forall$-blocking point, we expect that the original flow from X to Y involves only W_I and not its compliment $W_{\overline{I}}$. In particular, we prove the stronger statement that no information flows from $W_{\overline{I}}$ to Y:

$$W_{\overline{I}} \text{ does not IM-flow to } Y \text{ for } \sigma^{C_1} \text{ and } C_2. \tag{35}$$

This holds because whenever $w_1, w_2 \in \sigma^{C_1}$ with $w_1 =_I w_2$, we will prove that $C_2(w_1) = C_2(w_2)$. Otherwise, X would IM-flow to Y for σ and $\mathbb{C}_{W_I}$, where C_{W_I} is an W_I/v-assignment channel with $v =_I w_1$, where

$$\mathbb{C}_{W_I} \triangleq \ X\ C_1\ W'\ C_{W_I}\ W\ C_2\ Y\ .$$

This flow would contradict property (ii) of the $\forall$-blocking point definition applied to W_I.

We prove $w^c =_I w^a$ by starting from (32) and using the definition of C_{San}, (34), and (27):

$$w''' =_I w' \Rightarrow C_{San}(w''') =_I C_{San}(w')$$
$$\Rightarrow w^c =_I C_{San}(w') \Rightarrow w^c =_I w^a$$

So, we have

$$w^c =_I w^a. \tag{36}$$

Similarly, from (33), (34), and (29), we get

$$w^c =_{\overline{I}} w^b. \tag{37}$$

From (37), (34), and (31), we get that $w^c{}_I, w^c{}_{\overline{I}} \in D$. Because C_2 is D-preserving, we then get $C_2(w^c) \in D$. Because C_{San} is support-conserving for $\mathbb{C}$ and σ, we get that w^a and w^c are reachable values when $\mathbb{C}$ is executed on inputs supported by σ. So, w^a and w^c are supported by σ^{C_1}. From (36), (28), and $C_2(w^c) \in D$, we then get that $W_{\overline{I}}$ IM-flows to Y for C_2 and σ^{C_1}, which contradicts (35). So, we proved (ii). □

Theorem 4. *For a cascade $\mathbb{C} \triangleq \ X\ C_1\ W\ C_2\ Z\ C_3\ Y$ and support predicate σ, if $X \xrightarrow{Z_I}\!\!\!\!\!\!| \ Y$ holds for $\mathbb{C}$, and $X \xrightarrow{W_J}\!\!\!\!\!\!| \ Z_I$ holds for C_1C_2, then $X \xrightarrow{W_J}\!\!\!\!\!\!| \ Y$ holds for $\mathbb{C}$.* ■

Proof. In what follows, let $C^{W_I/v}$ denote a W_I/v-assignment channel C for a given v.

By hypothesis we have:

$$X \xrightarrow{Z_I}\!\!\!\!\!\!| \ Y \text{ for } \sigma \text{ and } \mathbb{C}, \tag{38}$$

$$X \xrightarrow{W_J}\!\!\!\!\!\!| \ Z_I \text{ for } \sigma \text{ and } C_1C_2. \tag{39}$$

We prove that

$$X \xrightarrow{W_J}\!\!\!\!\!\!| \ Y \text{ for } \sigma \text{ and } \mathbb{C}. \tag{40}$$

Let v be a W_J-reachable constant for $\mathbb{C}$ and σ. From property (ii) of $\forall$-blocking point definition applied to (39), we get that

$$C^{W_J/v} \text{ is support-conserving for } C_1C_2 \text{ and } \sigma \tag{41}$$

and, from property (i), we get that X does not IM-flow to Z_I for σ and

$$\mathbb{C}^{W_J/v}_a \triangleq X\ C_1\ W'\ C^{W_J/v}\ W\ C_2\ Z \tag{42}$$

So,

$$Z_I \text{ has a constant value } z_0 \text{ when } \mathbb{C}^{W_J/v}_a \text{ is executed in } \sigma. \tag{43}$$

Consider $\mathbb{C}^{W_J/v} \triangleq X\ C_1\ W'\ C^{W_J/v}\ W\ C_2\ Z\ C_3\ Y$. When $\mathbb{C}^{W_J/v}$ is executed in σ, then Z_I has always a constant value z_0, because of (43) and because $\mathbb{C}^{W_J/v} = \mathbb{C}^{W_J/v}_a\ C_3\ Y$. From (41) and the construction of $\mathbb{C}^{W_J/v}_a$, we get that

$$z_0 \text{ is } Z_I\text{-reachable for } \mathbb{C} \text{ and } \sigma. \tag{44}$$

From property (ii) of $\forall$-blocking point definition and (38), we get that for any Z_I-reachable constant u for $\mathbb{C}$ and σ, $C^{Z_I/u}$ is input conserving for $\mathbb{C}$ and σ, and X does not IM-flow to Y for $\mathbb{C}^{Z_I/u}$ and σ, where

$$\mathbb{C}^{Z_I/u} \triangleq X\ C_1\ W\ C_2\ Z'\ C^{Z_I/u}\ Z\ C_3\ Y. \tag{45}$$

We instantiate u with z_0 to have

$$\mathbb{C}^{Z_I/z_0} \triangleq X\ C_1\ W\ C_2\ Z'\ C^{Z_I/z_0}\ Z\ C_3\ Y. \tag{46}$$

From (44) we then get C^{Z_I/z_0} is input conserving for $\mathbb{C}$ and σ, and

$$X \text{ does not IM-flow to } Y \text{ for } \mathbb{C}^{Z_I/z_0} \text{ and } \sigma. \tag{47}$$

We have that $\mathbb{C}^{W_J/v}$ has the same channel matrix for σ as

$$\mathbb{C}^{W_J/v,Z_I/z_0} \triangleq X\ C_1\ W'\ C^{W_J/v}\ W\ C_2\ Z'\ C^{Z_I/z_0}\ Z\ C_3\ Y.$$

Because of (41), we get that the set of values that Z' takes at $\mathbb{C}^{W_J/v,Z_I/z_0}$ is a subset of the set of values that Z' takes at $\mathbb{C}^{Z_I/z_0}$. From (47), we then get that X does not IM-flow to Y for $\mathbb{C}^{W_J/v,Z_I/z_0}$, either. Because $\mathbb{C}^{W_J/v}$ has the same channel matrix for σ as $\mathbb{C}^{W_J/v,Z_I/z_0}$, we then get that X does not IM-flow to X for $\mathbb{C}^{W_J/v}$. Thus $X \xrightarrow{W_J}\!\!\!\!\!\!| \; Y$ holds. □

Lemma 1. *For $n > 2$, support predicate σ, and trajectory*

$$\begin{aligned} &\prec W^0 \succ C_1\ \prec W^1 \succ\ C_2\ \prec W^2 \succ\ \cdots \\ &\qquad C_i\ \prec W^i \succ\ \cdots C_n\ \prec W^n \succ \end{aligned} \tag{48}$$

if $W^0 \xrightarrow{W^{n-1}}\!\!\!\!\!\!| \; W^n$, $W^0 \xrightarrow{W^{n-2}}\!\!\!\!\!\!| \; W^{n-1}$, …, $W^0 \xrightarrow{W^1}\!\!\!\!\!\!| \; W^2$, then $W^0 \xrightarrow{W^1}\!\!\!\!\!\!| \; W^n$ holds. ■

Proof. Induction on n, where the induction hypothesis is: if $W^0 \xrightarrow{W^{n-2}}\!\!\!\!\!\!| \; W^{n-1}$, …, $W^0 \xrightarrow{W^1}\!\!\!\!\!\!| \; W^2$, then $W^0 \xrightarrow{W^1}\!\!\!\!\!\!| \; W^{n-1}$ holds.

- Base case $n = 3$: By the hypothesis of the lemma, we have $W^0 \xrightarrow{W^2}\!\!\!\!\!\!| \; W^3$ and $W^0 \xrightarrow{W^1}\!\!\!\!\!\!| \; W^2$. Theorem 4 then gives $W^0 \xrightarrow{W^1}\!\!\!\!\!\!| \; W^3$, where X in the statement of theorem 4 is instantiated as W^0, Z_I as W^2, Y as W^3, and W_J as W^1.
- Inductive case $n > 3$: By the hypothesis of the lemma, we get $W^0 \xrightarrow{W^{n-2}}\!\!\!\!\!\!| \; W^{n-1}$, …, $W^0 \xrightarrow{W^1}\!\!\!\!\!\!| \; W^2$. By the induction hypothesis, we then get $W^0 \xrightarrow{W^1}\!\!\!\!\!\!| \; W^{n-1}$. By the hypothesis of the lemma, we also have $W^0 \xrightarrow{W^{n-1}}\!\!\!\!\!\!| \; W^n$. From $W^0 \xrightarrow{W^1}\!\!\!\!\!\!| \; W^{n-1}$ and theorem 4, we then get $W^0 \xrightarrow{W^1}\!\!\!\!\!\!| \; W^n$ (where X in the statement of theorem 4 is instantiated as W^0, Z_I as W^{n-1}, Y as W^n, and W_J as W^1.).

□

Lemma 2. *Given $\mathbb{C} \triangleq X\ C_a\ W\ C_b\ Y$ and support predicate σ, if $X \xrightarrow{W_I}\!\!\!\!\!\!| \; Y$ holds, then W_I CD-flows to Y for C_b and σ^{C_a} (i.e., $W_I \xRightarrow[C_b]{\sigma^{C_a}} Y$).* ■

Proof. From property (ii) from the definition of ∀-blocking point, $X \xrightarrow{W_I}\!\!\!\!\!\!| \; Y$, we get that for all W_I-reachable constants v, the corresponding W_I/v-assignment channel C_{W_I} is support conserving. This means that any pairing of reachable values of W_I and reachable values of $W_{\overline{I}}$ is supported by σ^{C_a}. This implies that W_I and $W_{\overline{I}}$ are independent for the set of values supported by σ^{C_a}, so to prove that W_I CD-flows to Y for C_b and σ^{C_a}, it suffices to show:

$$W_I \text{ IM-flows to } Y \text{ for } C_b \text{ and } \sigma^{C_a}. \tag{49}$$

since if W_I IM-flows to Y, then we can expand σ^{C_a} to support more values and the flow will be maintained (since W_I and $W_{\overline{I}}$ are independent)—W_I CD-flows to Y.

From property (i) of the definition of ∀-blocking point, $X \xrightarrow{W_I}\!\!\!\!\!\!| \; Y$, we get that X IM-flows to Y for $\mathbb{C}$ and σ. So, there will be supported inputs x_1 and x_2 such that $w_1 = C_a(x_1)$, $w_2 = C_a(x_2)$, $w_1 \neq w_2$, and

$$C_b(w_1) \neq C_b(w_2). \tag{50}$$

Let $y_1 = C_b(w_1)$ and $y_2 = C_b(w_2)$. Let v be a W_I-reachable constant for $\mathbb{C}$ and σ. By construction of W_I/v-assignment channel C_{W_I}, we get that

$$(\exists w_a, w_b \in \mathcal{X}^{\sigma^{C_a C_{W_I}}} :\ w_a =_{\overline{I}} w_1 \wedge w_b =_{\overline{I}} w_2 \wedge w_a =_I w_b =_I v). \tag{51}$$

$C_b(w_a) = C_b(w_b)$ should hold. The proof is by contradiction. Otherwise, we would get that there exist $x_a, x_b \in \mathcal{X}^{\sigma}$ such that $w_a = C_{W_I}(C_a(x_a))$, $w_b = C_{W_I}(C_b(x_b))$ (because $w_a, w_b \in \mathcal{X}^{\sigma^{C_a C_{W_I}}}$) and $C_b(C_{W_I}(C_a(x_a))) \neq C_b(C_{W_I}(C_b(x_b)))$, meaning $\mathbb{C}_{W_I}(x_a) \neq \mathbb{C}_{W_I}(x_b)$ would hold. So, X IM-flows to Y for $\mathbb{C}_{W_I}$ and σ, which contradicts property (ii) of the definition of ∀-blocking point. So, we established that $C_b(w_a) = C_b(w_b)$ holds.

Let y' be $C_b(w_a)$, or equivalently $C_b(w_b)$. Value y' cannot both equal to y_1 and to y_2, otherwise (50) would not hold. W.l.o.g. say that $y' \neq y_1$. So, we have that $C_b(w_a) \neq C_b(w_1)$, where $w_a =_{\overline{I}} w_1$ (due to (51)), w_1 is supported by σ^{C_a} (because $w_1 = C_a(x_1)$ and x_1 is supported by σ) and w_a is supported by σ^{C_a}, since w_a is supported by $\sigma^{C_a C_{W_I}}$ and C_{W_I} is input conserving. So, we have shown the existence of two values w_a and w_b that agree of their I part and lead to different out put values y' and y_1. Thus, (49) holds.

□

Theorem 5. *Consider support predicate σ and a cascade*

$$X\ C_1\ Z^1\ C_2\ Z^2 \cdots Z^{n-1}\ C_n\ Y\ ,$$

where X IM-flows to Y and for each Z^j there is a singleton partial variable W^j, for $1 \leq j \leq n-1$. If $X \xrightarrow{W^{n-1}}\!\!\!\!\!\!| \; Y$, $X \xrightarrow{W^{n-2}}\!\!\!\!\!\!| \; W^{n-1}$, …, and $X \xrightarrow{W^1}\!\!\!\!\!\!| \; W^2$ hold, then

$$\begin{aligned} &\prec X \succ\ C_1\ \prec W^1 \succ\ C_2\ \prec W^2 \succ\ \cdots \\ &\qquad \prec W^{n-2} \succ\ C_{n-1}\ \prec W^{n-1} \succ\ C_n\ \prec Y \succ \end{aligned} \tag{52}$$

is a CD-trajectory. ■

Proof. To prove that (52) satisfies the definition of CD-trajectory, we must show that the CD-flow relation $W^i \xRightarrow[C_{i+1}\ldots C_j]{\sigma_{i+1}} W^j$ holds, for $\sigma_{i+1} = \sigma^{C_1 \ldots C_i}$ and $0 \leq i < j \leq n$, where $W^0 \triangleq X$ and $W^n \triangleq Y$. By hypothesis, we have that $X \xrightarrow{W^{n-1}}\!\!\!\!\!\!| \; Y$, $X \xrightarrow{W^{n-2}}\!\!\!\!\!\!| \; W^{n-1}$, …, and $X \xrightarrow{W^1}\!\!\!\!\!\!| \; W^2$. We consider two cases.

- Case $j = i + 1$: From hypothesis $X \xrightarrow{W^i}\!\!\!\!\!\!| \; W^{i+1}$, we then get $X \xrightarrow{W^i}\!\!\!\!\!\!| \; W^j$.
- Case $j > i + 1$: Let $C' = C_1 \ldots C_i$, then from (52) we derive trajectory

$$\begin{aligned} &\prec X \succ\ C'\ \prec W^i \succ\ C_{i+1}\ \prec W^{i+1} \succ\ \cdots \\ &\qquad \prec W^{j-1} \succ\ C_j\ \prec W^j \succ \end{aligned} \tag{53}$$

Applying Lemma 1 with $n = j - i + 1$ (i.e., $n > 2$) on (53) and $X \xrightarrow{W^{j-1}}\!\!\!\!\blacktriangleright W^j$, ..., $X \xrightarrow{W^i}\!\!\!\!\blacktriangleright W^{i+1}$, we have that $X \xrightarrow{W^i}\!\!\!\!\blacktriangleright W^j$.

Instantiating Lemma 2 with a cascade where $C_a = C_1 \dots C_i$ and $C_b = C_{i+1} \dots C_j$ (if $j = i + 1$ holds, then $C_b = C_{i+1}$), support predicate σ, and $X \xrightarrow{W^i}\!\!\!\!\blacktriangleright W^j$, we get that $W^i \xRightarrow[C_{i+1} \dots C_j]{\sigma_{i+1}} W^j$ holds (if $j = i + 1$ holds, then $C_{i+1} \dots C_j$ is C_{i+1}). □

REFERENCES

[1] *Counterfactuals*. Harvard University Press, Cambridge, Mass., 1973.

[2] M. S. Alvim, K. Chatzikokolakis, A. McIver, C. Morgan, C. Palamidessi, and G. Smith. *The Science of Quantitative Information Flow*. Springer, 2020.

[3] M. Backes, B. Köpf, and A. Rybalchenko. Automatic discovery and quantification of information leaks. In *2009 30th IEEE Symposium on Security and Privacy*, pages 141–153, May 2009.

[4] S. Beckers, J. Y. Halpern, and C. Hitchcock. Causal models with constraints. In *Proceedings of the 2nd Conference on Causal Learning and Reasoning*. 2023.

[5] F. Biondi, M. A. Enescu, A. Heuser, A. Legay, K. S. Meel, and J. Quilbeuf. Scalable approximation of quantitative information flow in programs. In I. Dillig and J. Palsberg, editors, *Verification, Model Checking, and Abstract Interpretation*, pages 71–93, Cham, 2018. Springer International Publishing.

[6] N. E. Bordenabe and G. Smith. Correlated secrets in quantitative information flow. In *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, pages 93–104, June 2016.

[7] T. Chothia, Y. Kawamoto, C. Novakovic, and D. Parker. Probabilistic point-to-point information leakage. In *2013 IEEE 26th Computer Security Foundations Symposium (CSF)*, page 193–205, USA, 2013. IEEE Computer Society.

[8] D. Clark, S. Hunt, and P. Malacaria. A static analysis for quantifying information flow in a simple imperative language. *J. Comput. Secur.*, 15(3):321–371, Aug. 2007.

[9] E. S. Cohen. Strong dependency: A formalism for describing information transmission in computational systems. Technical report, Carnegie Mellon University, 1976.

[10] P. Cuff and L. Yu. Differential privacy as a mutual information constraint. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 43–54, New York, NY, USA, 2016. Association for Computing Machinery.

[11] A. Datta, D. Garg, D. Kaynar, D. Sharma, and A. Sinha. Program actions as actual causes: A building block for accountability. In *2015 IEEE 28th Computer Security Foundations Symposium (CSF)*, pages 261–275, July 2015.

[12] D. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, SE-13(2):222–232, 1987.

[13] J. Heusser and P. Malacaria. Quantifying information leaks in software. In *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC '10, page 261–269, New York, NY, USA, 2010. Association for Computing Machinery.

[14] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen. Tane: An efficient algorithm for discovering functional and approximate dependencies. *The Computer Journal*, 42(2):100–111, 1999.

[15] E. Kozyri, S. Chong, and A. C. Myers. Expressing information flow properties. *Foundations and Trends in Privacy and Security*, 3(1):1–102, 2022.

[16] Y. Kwon, D. Kim, W. N. Sumner, K. Kim, B. Saltaformaggio, X. Zhang, and D. Xu. LDX: Causality inference by lightweight dual execution. *SIGARCH Comput. Archit. News*, 44(2):503–515, March 2016.

[17] Y. Kwon, F. Wang, W. Wang, K. H. Lee, W.-C. Lee, S. Ma, X. Zhang, D. Xu, S. Jha, G. F. Cretu-Ciocarlie, A. Gehani, and V. Yegneswaran. MCI: Modeling-based causality inference in audit logging for attack investigation. In *Network and Distributed System Security Symposium*, 2018.

[18] X. S. Liang. Information flow and causality as rigorous notions *ab initio*. *Phys. Rev. E*, 94:052201, Nov 2016.

[19] J. Liu, J. Li, C. Liu, and Y. Chen. Discover dependencies from data—A review. *IEEE Transactions on Knowledge and Data Engineering*, 24(2):251–264, 2012.

[20] J. Lizier and M. Prokopenko. Differentiating information transfer and causal effect. *The European Physical Journal B*, 73, 12 2008.

[21] J. McLean. Security models and information flow. In *Proceedings. 1990 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 180–187, 1990.

[22] J. McLean. A general theory of composition for trace sets closed under selective interleaving functions. In *1994 IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, CA, USA, May 16-18, 1994*, pages 79–93. IEEE Computer Society, 1994.

[23] J. Pearl. Interpretation and identification of causal mediation. *Psychological methods*, 19, 06 2014.

[24] L. Röth. *Pathway Analysis, Causal Mediation, and the Identification of Causal Mechanisms*, pages 123–151. 02 2023.

[25] A. Sabelfeld and A. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, 2003.

[26] M. Soloviev. *Rational Inattention and a Causal Account of Program Security*. PhD thesis, Cornell University, USA, 2021.

[27] R. C. Stalnaker. *A Theory of Conditionals*, pages 41–55. Springer Netherlands, Dordrecht, 1981.

[28] D. Tingley, H. Teppei, Y. Mit, L. Keele, P. State, and K. Imai. Mediation: R package for causal mediation analysis. *Journal of Statistical Software*, 59, 10 2014.

[29] M. C. Tschantz, A. Datta, A. Datta, and J. M. Wing. A methodology for information flow experiments. In *2015 IEEE 28th Computer Security Foundations Symposium (CSF)*, pages 554–568, July 2015.