Intransitive Noninterference in Nondeterministic Systems*

Kai Engelhardt
Computer Science and Engineering, The University of New South Wales, Sydney, NSW 2052, Australia
kaie@cse.unsw.edu.au

Ron van der Meyden
Computer Science and Engineering, The University of New South Wales, Sydney, NSW 2052, Australia
meyden@cse.unsw.edu.au

Chenyi Zhang†
Information Technology and Electrical Engineering, The University of Queensland, Brisbane, QLD 4072, Australia
chenyi@uq.edu.au

ABSTRACT

This paper addresses the question of how TA-security, a semantics for intransitive information-flow policies in deterministic systems, can be generalized to nondeterministic systems. Various definitions are proposed, including definitions that state that the system enforces as much of the policy as possible in the context of attacks in which groups of agents collude by sharing information through channels that lie outside the system. Relationships between the various definitions proposed are characterized, and an unwinding-based proof technique is developed. Finally, it is shown that on a specific class of systems, access control systems with local nondeterminism, the strongest definition can be verified by checking a simple static property.

Categories and Subject Descriptors

D.4.6 [**Security and Protection**]: Information flow controls

Keywords

access control, information-flow, nondeterminism, noninterference, security

1. INTRODUCTION

The theory of information flow security has been studied most extensively with respect to transitive security policies, motivated by the lattices associated with military multi-level security policies. It has long been recognised, however, that even in this setting, richer types of policies are required in order to deal with trusted components such as downgraders, which may violate a transitive policy. An example of this is given in Figure 1 which presents an abstract architecture for a system in which two multi-level secure machines M_1 and M_2 communicate across the internet.

*Research supported by Australian Research Council Discovery Grant DP1097203.

†Work of the third author conducted while employed at The University of New South Wales.

CCS'12, October 16–18, 2012, Raleigh, North Carolina, USA.

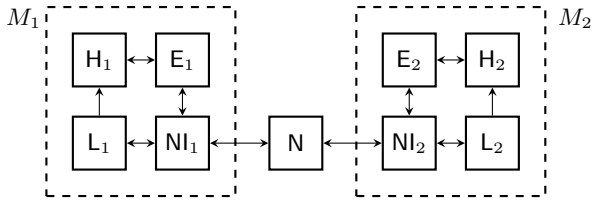


Figure 1: Architecture for a MILS system.

Each machine M_i contains a high-level domain H_i and a low-level domain L_i, with the usual policy $\mathsf{H}_i \not\rightsquigarrow \mathsf{L}_i$, intuitively requiring that no high-level information flow to the low level domain, enforced between the two. The internet is represented by the domain N. Additionally, there are two domains $\mathsf{E}_1, \mathsf{E}_2$ that represent downgraders that are responsible for encrypting and decrypting all communications between the domains $\mathsf{H}_1, \mathsf{H}_2$, as well as domains $\mathsf{NI}_1, \mathsf{NI}_2$ representing the network interface in each machine.

Globally, the security policy requires that there is no *direct* flow of information from the domains H_i to the domains $\mathsf{N}, \mathsf{NI}_j$ or L_j. All such flow of information must be mediated by the domains E_i. This does not provide a complete guarantee of all security properties that one might wish the system to satisfy, but it helps to focus proofs that high-level information remains secure onto the specific *trusted* components E_i. The intention of the architecture is to decompose the proof of the desired security property that high-level information does not flow to low domains (even if only the components E_i are trusted), to the proof that the components E_i properly encrypt all output (and possibly also, that they maintain a traffic stream to circumvent traffic analysis), and the fact that the architecture is enforced. We refer the reader to recent work on MILS [4] for a more detailed explanation of this idea.

In order to provide an account of formal security verification that captures the intuitions underlying such architectures, we first need a mathematically precise semantics for policies in the form of intransitive relations (note, e.g., that in Figure 1 we have $\mathsf{H}_1 \mapsto \mathsf{E}_1$ and $\mathsf{E}_1 \mapsto \mathsf{NI}_1$ but not $\mathsf{H}_1 \mapsto \mathsf{NI}_1$). Compared to classical information flow theory for transitive policies, this area is much less studied.

One of the landmarks in the area remains the work of Rushby [21], which clarified earlier work of Haigh and Young [10]. In particular, Rushby provides an proof method using *unwinding relations* that may be used to show security of a system, and moreover proves that secure systems can be concretely constructed by using an access control discipline satisfying a simple syntactic condition. This latter result is significant in that it can be understood as providing a more satisfactory basis for ideas from Bell and La Padula [2], addressing complaints about the well-foundedness of Bell and La

Padula's methods [14], by giving semantic meaning to the notions of *read* and *write*.

Recently, van der Meyden [24] pointed to weaknesses in Rushby's definitions and provided improvements, including a new definition of security called *TA-security*, that yield results similar to Rushby's but which make the unwinding proof method and access control discipline not just sound but also *complete* for security (showing that any secure system can be proved to be secure using the method, or *constructed* so that security is easily checkable.) This yields a pleasant theory in which the definition of security, proof methods and engineering discipline are tightly integrated.

This theory is limited to *deterministic* systems, however. There have been proposals for semantics of intransitive policies in nondeterministic systems [3, 19, 13, 1, 26], but none of these works deals with access control systems with nondeterminism. Our contribution in this paper is to develop the first such generalization, taking as our starting point van der Meyden's formulation of the deterministic case.

We first investigate how to generalize the notion of TA-security to the nondeterministic setting. After setting up the semantic framework in Section 2, in Section 3 we carefully tease out a number of dimensions that are relevant to the formulation of the semantics for intransitive policies in nondeterministic systems. Some definitions in the literature, we believe, have made inappropriate choices with respect to these ingredients, and in some, critical issues have been ignored. In particular, we argue that it is important in non-deterministic systems to base the definition of security on the effect of actions on agents' *history* of actions and observations, whereas some definitions in the literature have considered only their effect on single observations. We also recall from the literature on noninterference with respect to transitive policies the notion of *persistence*, which helps with an important distinction between deducibility and causality that emerges in nondeterministic systems. Finally, we present an example showing that while collusion attacks can be ignored in deterministic systems, they need to be taken into account in nondeterministic systems. Many of the definitions in the literature do not take such attacks into account.

We give our generalizations of TA-security to the nondeterministic setting in Section 4. We show that there are subtleties in the formulation of definitions that cover collusion attacks: it makes a difference, to what the attackers can deduce, whether they share information during a run of the system, or only at completion of the run. This leads us to a spectrum of definitions, depending on whether and, if so, how, collusion is treated, and via application of persistence or not, whether the definition is causal or deductive. We discuss some special cases of policies and systems in Section 5, where we show that our spectrum of definitions collapses to two well-known definitions of security in the case of deterministic systems or a two agent policy.

We then proceed, in Section 6, to develop an unwinding proof technique that is sound for all our definitions. In Section 7, we present a generalization of access control systems that covers nondeterminism, and identify conditions on such systems that imply that all our definitions of security hold. We discuss related work in Section 8 and conclude with a discussion of future directions for research in Section 9.

2. SYSTEMS MODEL

Goguen and Meseguer [7] developed a theory of information-flow that has formed a starting point for later research. They introduced the following policy model.

Definition 1. A *noninterference policy* for a set $\mathbb{U}$ of security domains is a reflexive binary relation $\mapsto$ on $\mathbb{U}$, representing the permitted interferences between domains.

Intuitively, $u \mapsto v$ represents that the policy permits information to flow from domain u to domain v. Another intuition for this is that actions of domain u are permitted to have causal effects on, or *interfere with*, domain v. Conversely, when $u \not\mapsto v$, domain u is not permitted to interfere with v. Reflexivity is assumed since, in general, nothing can be done to prevent flows of information from a domain to itself, so it is not sensible for the policy to prohibit this. A *machine* with domains $\mathbb{U}$ is a tuple $M = (S, s_0, A, \longrightarrow, obs, dom)$ with a set S of *states*, including a designated *initial state* $s_0 \in S$, a set A of actions, a (nondeterministic) transition relation $\longrightarrow \subseteq S \times A \times S$ an *observation function* $obs : \mathbb{U} \to (S \to O)$, for some set O, and a *domain function* $dom : A \to \mathbb{U}$. We write obs_u for the function $obs(u) : S \to O$, which represents the observation that domain u makes at each state of the machine. Following much of the security literature, we assume that the transition relation is *input-enabled*: for all states s and actions a, there exists a state t such that $s \xrightarrow{a} t$. This helps to prevent enabledness of actions being a source of information. A machine is *deterministic* if for all states s, t, t' and actions a, if $s \xrightarrow{a} t$ and $s \xrightarrow{a} t'$ then $t = t'$.

Notational and diagrammatic conventions.

Sequences play an important role in this paper. Most of the time, we denote a sequence such as $[a, b, c]$ by just abc, however, if the sequence elements have structure themselves, or if confusion is likely to arise, we tend to separate sequence elements with a dot: $a \cdot b \cdot c$.

A concise way to define A and dom is to list, for each $u \in \mathbb{U}$, the set $A_u = \{\ a \in A \mid dom(a) = u\ \}$ of actions of that domain.

Given a domain u, we write $^{\mapsto}u$ for $\{\ v \in \mathbb{U} \mid v \mapsto u\ \}$ and $^{\not\mapsto}u$ for $\mathbb{U} \setminus {}^{\mapsto}u = \{\ v \in \mathbb{U} \mid v \not\mapsto u\ \}$. We also write $u^{\mapsto}$ for $\{\ v \in \mathbb{U} \mid u \mapsto v\ \}$ and $u^{\not\mapsto}$ for $\mathbb{U} \setminus {}^{\mapsto}u$.

We use the following convention in diagrams representing machines. States are represented by circles labelled internally by the state name. The initial state is always named s_0. A transition $s \xrightarrow{a} t$ is represented by an edge from s to t labelled by a. To remove clutter from diagrams, we elide edges corresponding to self-loops $s \xrightarrow{a} s$ unless we wish to draw attention to them; thus, if there is no edge from a state labelled by an action a, then using the input-enabled assumption, we infer that the missing edge is a self-loop. (Note that if there does exist an edge labelled a from state s, we do *not* make this inference.) States are labelled externally by observations of some of the domains: the details of this depend on the example and are given with each diagram.

A *run* is a sequence of the form $s_0 \xrightarrow{a_1} s_1 \ldots \xrightarrow{a_n} s_n$ in which $n \geq 0$, the $s_i \in S$ are states (the first being the initial state) and the $a_i \in A$ are actions, such that $(s_{i-1}, a_i, s_i) \in \longrightarrow$ for all $i = 1 \ldots n$. We write $\mathcal{R}(M)$ for the set of runs of machine M. The function *last* maps a nonempty sequence to its final element. A state is *reachable* if it is the final state of some run. The sequence of all actions in a run r is denoted $Act(r)$. With M implicit, we also write $\mathcal{R}(\alpha)$ for the set of $r \in \mathcal{R}(M)$ such that $Act(r) = \alpha$. For a domain u, we write $Act_u(r)$ for the subsequence of actions a in $Act(r)$ with $dom(a) = u$.

The *view* of a run obtained by a domain, or group of domains, records all the actions and observations of the domain or group during the run, except that stuttering observations are collapsed to a single copy to model that the agent/group operates asynchronously, so does not perceive the passing of time unless some event happens to it. Let $X \subseteq \mathbb{U}$ be a nonempty set of domains. We de-

fine the joint observation function of X as $obs_X : S \to O^X$ by $obs_X(s)(u) = obs_u(s)$ for $u \in X$. That is, $obs_X(s)$ is the tuple of observations made by the domains $u \in X$ on state s. The view function $\mathtt{view}_X : \mathcal{R}(M) \to (O^X)^+(A(O^X)^+)^*$ is defined inductively by $\mathtt{view}_X(s_0) = obs_X(s_0)$ and

$$\mathtt{view}_X(r \xrightarrow{a} q) = \begin{cases} \mathtt{view}_X(r) \cdot a \cdot obs_X(q) & \text{if } dom(a) \in X \\ \mathtt{view}_X(r) \mathbin{\hat{\circ}} obs_X(q) & \text{otherwise} \end{cases}$$

where "$\hat{\circ}$" denotes *absorptive* concatenation, that is,

$$\alpha \mathbin{\hat{\circ}} a = \begin{cases} \alpha & \text{if } last(\alpha) = a \\ \alpha \cdot a & \text{otherwise.} \end{cases}$$

We write the special case where $X = \{u\}$ is a singleton as $\mathtt{view}_u$. We note that $\mathtt{view}_X(r)$ may contain information about the order of actions from domains in X that cannot be deduced from the collection of views $(\mathtt{view}_u(r))_{u \in X}$. Intuitively, $\mathtt{view}_X(r)$ is the information that the group would obtain in r when the members of the group share their local information with the group at each step of the run, whereas the collection $(\mathtt{view}_u(r))_{u \in X}$ corresponds to the information that the group would have if the members shared their views only after r has completed.

3. BACKGROUND

In this section we review some of the literature on information-flow security and formulate versions of existing definitions within our system model. This review will motivate several of the ingredients we use in the new definitions we propose later. This section is largely a review of the existing literature, however, we do give new characterizations of some existing definitions, using a notion of relative information, that helps to clarify the relationship to the new definitions we introduce later on.

3.1 Noninterference, policy $\mathsf{H} \not\rightarrow \mathsf{L}$

Goguen and Meseguer's formal semantics for noninterference policies is restricted to *transitive* policies in *deterministic* machines. They define for each domain $u \in \mathbb{U}$ the purge function $\mathtt{purge}_u : A^* \to A^*$ that maps a sequence of actions to the subsequence of actions a with $dom(a) \mapsto u$.

Definition 2. A machine M satisfies *noninterference (NI)* w.r.t. $\mapsto$ if for all domains u and all runs r, r' of M with $\mathtt{purge}_u(Act(r)) = \mathtt{purge}_u(Act(r'))$, we have $obs_u(last(r)) = obs_u(last(r'))$.

It can be shown that, in deterministic machines, this is equivalent to the following: for all runs r, r' of M with $\mathtt{purge}_\mathsf{L}(Act(r)) = \mathtt{purge}_\mathsf{L}(Act(r'))$, we have $\mathtt{view}_\mathsf{L}(r) = \mathtt{view}_\mathsf{L}(r')$. This presentation makes it clear that the definition says that the information in L's view in a run depends only on the L actions in the run, and are independent of the H actions.

One of the main questions of study since the seminal work of Goguen and Meseguer on deterministic systems is how their definitions should be generalized to nondeterministic systems. A great deal of the literature on this topic has been concerned with the simple two domain policy (which we write as $\mathsf{H} \not\rightarrow \mathsf{L}$) given by the relation $\{(\mathsf{L},\mathsf{L}), (\mathsf{H},\mathsf{H}), (\mathsf{L},\mathsf{H})\}$ on the set $\mathbb{U} = \{\mathsf{L},\mathsf{H}\}$, comprised of the low-level (public) domain L and the high-level (classified) domain H. This is in part because even this simple policy presents many subtleties in the setting of nondeterministic systems, but also because there has been a view that it is possible to reduce arbitrary policies to this special case.

Numerous definitions have been proposed that generalize NI for the policy $\mathsf{H} \not\rightarrow \mathsf{L}$ to nondeterministic systems. We now discuss a number of points from this literature that highlight issues that are relevant to the novel definitions for the more general intransitive policies that we introduce later.

3.2 Nondeducibility and Relative Information

Sutherland [22] proposed to interpret the policy $\mathsf{H} \not\rightarrow \mathsf{L}$, as stating, informally, that L cannot deduce information about H, and gave a general formal account of deducibility using a relation on functions with domain the state space of the system. A generalization of (non)deducibility will be useful for what follows, to make explicit a common logical structure that underlies the definitions we propose. Definitions similar to the following have been considered by Halpern and O'Neill [11] and More et al [17].

Definition 3. Let f, g and h be functions, each with domain the set W of 'worlds'. We say that *in W, the function f contains no more information than g about h*, if for all $w, w' \in W$ such that $g(w) = g(w')$ there exists w'' such that $h(w'') = h(w')$ and $f(w'') = f(w)$.

As an application of Definition 3, consider the following definition of security:

Definition 4. A machine M for the policy $\mathsf{H} \not\rightarrow \mathsf{L}$ satisfies *correctability (COR)* if for all runs r and sequences $\alpha \in A^*$ with $Act_\mathsf{L}(r) = Act_\mathsf{L}(\alpha)$, there exists a run r' with $Act(r') = \alpha$ and $\mathtt{view}_\mathsf{L}(r') = \mathtt{view}_\mathsf{L}(r)$.

We call this notion correctability in view of its similarity to a notion of that name from [12]. It is easily seen that correctability may be given a clean characterization using the above notion of relative information as follows: M satisfies correctability iff in $\mathcal{R}(M)$, the function $\mathtt{view}_\mathsf{L}$ contains no more information about Act than $\mathtt{purge}_\mathsf{L} \circ Act$.

In a special case, we can furthermore formulate relative information in a more symmetric way. Using a notation reminiscent of probability theory, for functions f and g with domain W, and value v in the range of f, define $poss(g \mid f = v)$ to be the set $\{\, g(w) \mid w \in W \wedge f(w) = v \,\}$. Then we have the following result.

Proposition 1. Let f be a function with domain W, let $g : W \to V$ be surjective and let h be a function with domain V (so that $h \circ g$ also has domain W). Then in W, the function f contains no more information than $h \circ g$ about g iff for all $v, v' \in V$ with $h(v) = h(v')$ we have $poss(f \mid g = v) = poss(f \mid g = v')$.

Since we work with input-enabled machines, the function $Act : \mathcal{R}(M) \to A^*$ is surjective. By Proposition 1 and the relative information characterization of COR above, an equivalent formulation of COR is that for all $\alpha, \alpha' \in A^*$ with $\mathtt{purge}_\mathsf{L}(\alpha) = \mathtt{purge}_\mathsf{L}(\alpha')$ we have $poss(\mathtt{view}_\mathsf{L} \mid Act = \alpha) = poss(\mathtt{view}_\mathsf{L} \mid Act = \alpha')$. This states COR in a form that clarifies its relationship to the view-based formulation of NI, by showing that that COR is obtained by generalizing the single-valued view function of a deterministic machine used in NI by a set-valued view function in the nondeterministic setting.

3.3 Observation-based definitions are too weak in non-deterministic systems

For the policy $\mathsf{H} \not\rightarrow \mathsf{L}$, the definition of noninterference states that the *observation* of L in the final state of a run r should depend

only on $\texttt{purge}_{\mathsf{L}}(Act(r))$. As noted above, in deterministic systems, this is equivalent to the statement that the history $\texttt{view}_{\mathsf{L}}(r)$ of everything that is observable to L in the run should depend only on $\texttt{purge}_{\mathsf{L}}(Act(r))$. However, as the following example shows, a similar equivalence between definitions stated in terms of observations in the final state of a run and definitions stated in terms of the view on the run does not hold in nondeterministic systems.

Example 1. Consider the security policy $\mathsf{H} \not\rightarrow \mathsf{L}$ and the nondeterministic machine M depicted in Fig. 2, where states are labeled externally with L's observation, $A = \{\ell, h\}$, $dom(\ell) = \mathsf{L}$, and $dom(h) = \mathsf{H}$. Suppose we define an *observation*-based version of correctability: M is *obs-COR* if in $\mathcal{R}(M)$, the function $obs_{\mathsf{L}} \circ last$ contains no more information than $\texttt{purge}_{\mathsf{L}} \circ Act$ about Act. Equivalently, by Proposition 1, for all $\alpha, \alpha' \in A^*$, if $\texttt{purge}_{\mathsf{L}}(\alpha) = \texttt{purge}_{\mathsf{L}}(\alpha')$ then $poss(obs_{\mathsf{L}} \circ last \mid Act = \alpha) = poss(obs_{\mathsf{L}} \circ last \mid Act = \alpha')$. It can be seen that M satisfies obs-COR: the only H transition that can affect L observations is that from s_0, but if this is added or deleted from a run, there exists another run with the same subsequent sequence of actions ending in the same final observation for L. However, the possible views may differ, depending on whether h occurred: note that $\texttt{purge}_{\mathsf{L}}(h\ell\ell) = \ell\ell = \texttt{purge}_{\mathsf{L}}(\ell\ell)$, but $poss(\texttt{view}_{\mathsf{L}} \mid Act = \ell\ell) = \{0\ell1\ell2, 0\ell2\ell1\}$ whereas $poss(\texttt{view}_{\mathsf{L}} \mid Act = h\ell\ell) = \{0\ell1\ell1, 0\ell2\ell2\}$. Thus, this machine does not satisfy COR. Intuitively, it is insecure, since L can determine from its view whether the initial h action occurred.

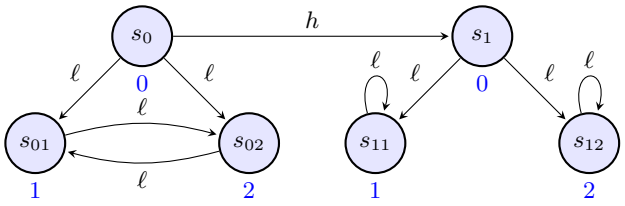


Figure 2: Deductions ought to be based on views rather than observations

This example points to the fact that in defining security in nondeterministic systems, we need to take the evidence from which an agent makes deductions to be its view, i.e., all that it could have observed to the present moment, rather than just its current observation, as in the definition of noninterference and some of its later generalizations (e.g., the definition of intransitive noninterference in deterministic systems [10, 21]). This point has sometimes been missed in the literature on nondeterministic systems, e.g., see the discussion of the work of von Oheimb [26] in Section 8 below.

3.4 Persistence

Two different intuitive interpretations of the notion of noninterference can be given: an epistemic interpretation which says that L is not able to *know*, or *deduce* anything about H activity, and a *causal* interpretation, which says that H actions may not have any causal effect on L observations. In deterministic systems, these interpretations may coincide, but this is no longer the case in nondeterministic systems, as the following example shows.

Example 2. Consider the machine depicted in Fig. 3 for the policy $\mathsf{H} \not\rightarrow \mathsf{L}$. States are labeled externally with L's observation, except for states on which L observes $\perp$, in which case we elide the observation. As in the previous example, $A = \{\ell, h\}$, $dom(\ell) = \mathsf{L}$, and $dom(h) = \mathsf{H}$. It can be seen that this machine satisfies the epistemic notion COR: domain L cannot make any deductions from its

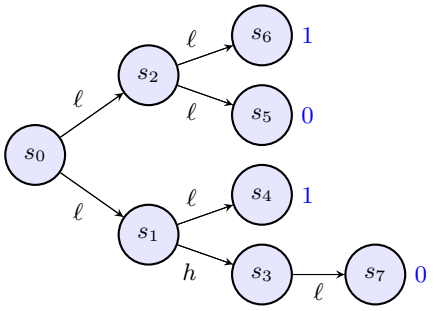


Figure 3: A machine that is COR but not P-COR.

view about H actions, or how these are interleaved with its own. (Recall that we elide self-loops.) However, it can reasonably be argued that this machine is not secure on a causal interpretation of security: note that by performing or not the action h from the state s_1, domain H is able to influence whether L subsequently observes 0 or 1.

Examples such as this can be addressed using the notion of *persistence*, which has been factored into a number of definitions in the literature [5, 6, 18].

Definition 5. For a security definition X, we say that machine $M = (S, s_0, A, \longrightarrow, obs, dom)$ *persistently satisfies* X *(P-*X*)* w.r.t. a policy $\mapsto$ if the machine $(S, s, A, \longrightarrow, obs, dom)$ satisfies X w.r.t. $\mapsto$, for all reachable states s of M.

Note that the machine in Example 2 is not P-COR.[1]

3.5 Collusion

For policies that generalize from the two-domain setting of the policy $\mathsf{H} \not\rightarrow \mathsf{L}$ in nondeterministic systems, the issue of collusion becomes of concern. As we illustrate in the present section, this is so even for transitive security policies.

The simplest type of policy for which this point can be made is the *separability* policy [20, 15] which says that no domain may interfere with any other. That is, for set of domains $\mathbb{U}$, separability is the policy $\Delta_{\mathbb{U}} = \{\ (u, u)\ \mid\ u \in \mathbb{U}\ \}$. In the case that $\mathbb{U}$ consists of two domains A and B, this seems to say that A may not interfere B, and *vice versa*, so one apparently reasonable interpretation of the policy is to apply a semantics for $\mathsf{H} \not\rightarrow \mathsf{L}$ for all domains. This idea suggests the following definition, when we apply the correctability semantics for $\mathsf{H} \not\rightarrow \mathsf{L}$ to each domain, and use our relative information formulation.

Definition 6. A machine M for a set $\mathbb{U}$ of domains satisfies *mutual correctability (MCOR)* w.r.t. $\mapsto$ if for all domains $u \in \mathbb{U}$, in $\mathcal{R}(M)$, $\texttt{view}_u$ contains no more information than $\texttt{purge}_u \circ Act$ about Act.

In the case of the policy $\mapsto\ = \Delta_{\mathbb{U}}$, this says that no domain is able to deduce from its view anything about what actions other

[1]We remark that in the case of COR, the notion P-COR is in the spirit of the notion *0-forward correctability* of Johnson and Thayer [12], which says that an addition or deletion from a trace of an H action requires only changes to *subsequent* H observations to obtain another trace that looks the same to L. However, P-COR is stronger, since it requires a correction from a given state, while 0-forward-correctability is a trace-based notion that allows the correction to pass through different states on the common prefix of events.

domains have performed, or how those actions were interleaved with its own.

It turns out that, in some circumstances, this is an insufficient guarantee. Suppose that we have a system with three separated domains, i.e., $\mathbb{U} = \{\mathsf{H}, \mathsf{L}_1, \mathsf{L}_2\}$ and the policy is $\Delta_{\mathbb{U}}$. However, L_1 and L_2 are corrupt, and collude by communicating via a channel that lies outside the system. Under these circumstances, there is nothing that can be done in practice to prevent information-flow between L_1 and L_2, even if the system is secure, so that it is not the *cause* of the information-flow. However, we expect that, since the system enforces the policy, H's information is still protected from leakage to L_1 and L_2. In fact, mutual correctability is too weak to provide such a guarantee, as is shown by the following example.

Example 3. Consider the machine depicted in Fig. 4 under the policy $\Delta_{\{\mathsf{H},\mathsf{L}_1,\mathsf{L}_2\}}$. The domain H observes $\perp$ at all states. The two low domains L_1 and L_2 have observations in the set $\{\perp, 0, 1\}$. These are indicated in Fig. 6 by labelling states to the right above and below by the observations made by L_1 and L_2, respectively. We omit the observation $\perp$ to reduce clutter. It can be verified that machine M satisfies MCOR. However, H's information is not secure against collusion by the coalition $\mathsf{L} = \{\mathsf{L}_1, \mathsf{L}_2\}$. We may represent the information held by the coalition L in a run r by $\mathtt{view}_{\mathsf{L}}(r)$, using the set-based view definition introduced above. Similarly, we may generalize the purge function to the coalition by writing $\mathtt{purge}_{\mathsf{L}}(\alpha)$ for the subsequence of actions a in $\alpha \in A^*$ with $dom(a) \in \mathsf{L}$. Let $\alpha = \ell_1\ell_2$ and $\beta = h\ell_1\ell_2$. Note that $\mathtt{purge}_{\mathsf{L}}(\alpha) = \alpha = \mathtt{purge}_{\mathsf{L}}(\beta)$. But

$$\begin{aligned} poss(\mathtt{view}_{\mathsf{L}} \mid Act = \alpha) &= \{\, {}^{\perp}_{\perp}\ell_1\,{}^{0}_{\perp}\ell_2\,{}^{0}_{0},\ {}^{\perp}_{\perp}\ell_1\,{}^{1}_{\perp}\ell_2\,{}^{1}_{1} \,\} \\ &\neq \{\, {}^{\perp}_{\perp}\ell_1\,{}^{0}_{\perp}\ell_2\,{}^{0}_{1},\ {}^{\perp}_{\perp}\ell_1\,{}^{1}_{\perp}\ell_2\,{}^{1}_{0} \,\} \\ &= poss(\mathtt{view}_{\mathsf{L}} \mid Act = \beta)\ . \end{aligned}$$

Intuitively, although the policy suggests that the coalition should not be able to distinguish between the sequences α and β, in fact they can, using the parity of their final observations.

There has been some informal recognition in the literature of the relevance of coalitions when dealing with policies beyond $\mathsf{H} \not\rightsquigarrow \mathsf{L}$, but there appear to have been very few formal studies of the issue. (We defer discussion of the related literature to Section 8.) One of our contributions in this paper is to pursue such a formal study in the general setting of intransitive policies.

3.6 Intransitive Noninterference

The final background we require concerns semantics for intransitive policies: this issue has been studied primarily in the setting of deterministic systems.

One of the main motivations for intransitive policies is to represent the role of *trusted* components within an architectural design of a system. A canonical example of this is a *downgrader*, a trusted component that manages declassification of high-level secrets to low-level domains. A policy for such a system is the policy HDL $= \{\mathsf{H}, \mathsf{D}, \mathsf{L}\}^2 \setminus \{(\mathsf{H}, \mathsf{L})\}$ depicted in Figure 5. Here D represents a trusted downgrader component.

Figure 5: Policy HDL.

Even in deterministic systems, NI is not an adequate definition of security for such intransitive policies, since it implies that L can learn nothing about H activity (not even when the downgrader permits it). To give a more adequate semantics, Haigh and Young [10] generalized the definition of the purge function to intransitive policies; we follow the formulation of Rushby [21]. Intuitively, the intransitive purge of a sequence of actions with respect to a domain u

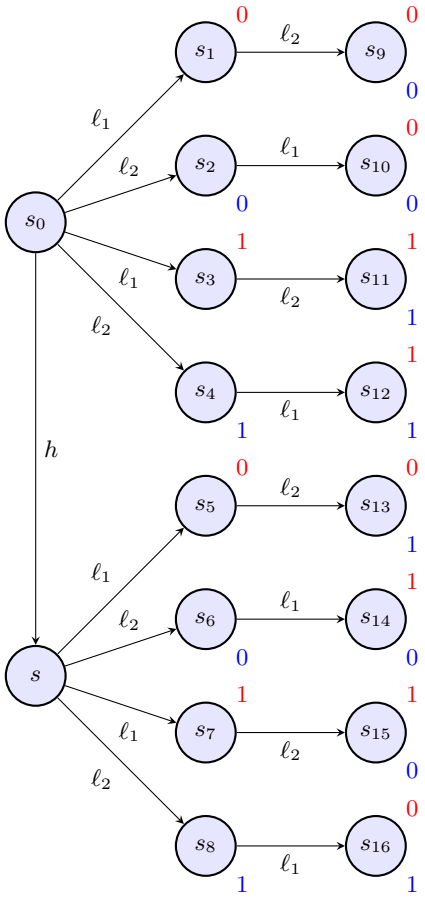


Figure 4: An insecure system that satisfies MCOR.

is the largest subsequence of actions that could form part of a causal chain of effects (permitted by the policy) ending with an effect on domain u. More formally, the definition makes use of a function $\mathtt{src} : \mathbb{U} \times A^* \to \mathcal{P}(\mathbb{U})$ defined inductively by $\mathtt{src}_u(\epsilon) = \{u\}$ and $\mathtt{src}_u(a\alpha) =$

$$\mathtt{src}_u(\alpha) \cup \{\ dom(a) \mid \exists v \in \mathtt{src}_u(\alpha)\,(dom(a) \mapsto v)\ \}$$

for $a \in A$ and $\alpha \in A^*$. Intuitively, $\mathtt{src}_u(\alpha)$ is the set of domains v such that there exists a sequence of permitted interferences from v to u within α. The *intransitive purge* function $\mathtt{ip}_u : A^* \to A^*$ for each domain $u \in \mathbb{U}$ is then defined inductively by $\mathtt{ip}_u(\epsilon) = \epsilon$ and, for $a \in A$ and $\alpha \in A^*$,

$$\mathtt{ip}_u(a\alpha) = \begin{cases} a \cdot \mathtt{ip}_u(\alpha) & \text{if } dom(a) \in \mathtt{src}_u(a\alpha) \\ \mathtt{ip}_u(\alpha) & \text{otherwise.} \end{cases}$$

Haigh and Young's definition of security uses the intransitive purge function in place of the purge function in Goguen and Meseguer's definition. Using our relative information formulation, the following is equivalent:

Definition 7. A deterministic machine M is *IP-secure* w.r.t. a (possibly intransitive) policy $\mapsto$ if for all $u \in \mathbb{U}$, in $\mathcal{R}(M)$, the function $obs_u \circ last$ contains no more information than $\mathtt{ip}_u \circ Act$ about Act.

As with NI, an equivalent statement is that for all $u \in \mathbb{U}$, in $\mathcal{R}(M)$, the function $\mathtt{view}_u$ contains no more information than $\mathtt{ip}_u \circ Act$

about Act. It can be seen that $\mathtt{ip}_u = \mathtt{purge}_u$ when $\mapsto$ is transitive, so IP-security is in fact a generalization of the definition of security for transitive policies.

It has been argued by van der Meyden [24] that IP-security misses some subtle flows of information relating to the ordering of events. A very simple example (from [25]) illustrating the problem is that there exists an IP-secure system for the policy $\mathsf{H} \mapsto \mathsf{D} \mapsto \mathsf{L}$ with actions h, d, ℓ in domains $\mathsf{H}, \mathsf{D}, \mathsf{L}$ respectively, in which L makes different observation after the sequence of actions $h\ell d$ than after the sequence of actions $\ell h d$. Since $\mathtt{ip}_{\mathsf{L}}(h\ell d) = h\ell d$ and $\mathtt{ip}_{\mathsf{L}}(\ell h d) = \ell h d$ are also distinct, this is not a violation of IP-security. However, it can be argued that such a system is not secure. Intuitively, L learns whether or not the h action came before the ℓ action. But, the system model is asynchronous, and according to the policy, the only way L is permitted to learn about the h action is via D. Since D is not permitted to know about the ℓ action, D cannot securely inform L how action h was ordered with respect to action ℓ.

In response to this sort of example, van der Meyden proposes an alternate semantics that is based on a function $\mathtt{ta}_u$ for each domain u, that is intended to capture the maximal information about actions that domain u may have, according to the policy. (The name of the function and the associated notion of security abbreviate "**t**ransmission of information about **a**ctions".) This function is inductively defined by

$$\mathtt{ta}_u(\epsilon) = \epsilon \text{ and}$$

$$\mathtt{ta}_u(\alpha a) = \begin{cases} (\mathtt{ta}_u(\alpha), \mathtt{ta}_{dom(a)}(\alpha), a) & \text{if } dom(a) \mapsto u \\ \mathtt{ta}_u(\alpha) & \text{otherwise.} \end{cases}$$

This definition resembles a *full-information protocol*, in which, when performing an action a after sequence α, domain $dom(a)$ sends everything that it knows (as represented by $\mathtt{ta}_{dom(a)}(\alpha)$), as well as the fact that it has performed action a, to every domain u to which it is permitted to transmit information. The recipient domain u adds this new information to its existing information $\mathtt{ta}_u(\alpha)$. Domains to which $dom(a)$ is not permitted to send information learn nothing when a happens. This function forms the basis for a definition of security which may be formulated using relative information as follows:

Definition 8. A deterministic machine M is *TA-secure* w.r.t. a policy $\mapsto$ if $obs_u \circ last$ contains no more information than $\mathtt{ta}_u \circ Act$ about Act.

Again, it proves to be equivalent to say that $\mathtt{view}_u$ contains no more information than $\mathtt{ta}_u \circ Act$ about Act. This definition is equivalent to NI in the case of transitive policies.

4. MAIN DEFINITIONS

We are now positioned to state our new definitions, which give meaning to intransitive noninterference policies in nondeterministic machines. As noted above, our focus in this paper is with definitions that place constraints on the flow of information concerning the actions that have been performed. We furthermore focus on generalizing the notion of TA-security from van der Meyden [24].

We characterized TA-security in deterministic machines as stating that for each domain u, the function $obs_u \circ last$ contains no more information than $\mathtt{ta}_u \circ Act$ about Act, and noted that it is equivalent to take $\mathtt{view}_u$ in place of $obs_u \circ last$ in this definition. However, in nondeterministic systems, an observation-based definition of security may be too weak, as shown in Section 3.3. This suggests the following as an appropriate generalization of the definition to nondeterministic systems.

Definition 9. A nondeterministic machine M is *nTA-secure w.r.t. a policy* $\mapsto$ if, for all $u \in \mathbb{U}$, the function $\mathtt{view}_u$ contains no more information than $\mathtt{ta}_u \circ Act$ about Act.

Intuitively, as in the deterministic case, this definition places, in each run r, an upper bound on the information about the action sequence $Act(r)$ that is permitted to be contained in each possible view $\mathtt{view}_u(r)$: it may be no more than the information contained in $\mathtt{ta}_u(Act(r))$. By Proposition 1, an equivalent statement is that for all α, β in A^*, if $\mathtt{ta}_u(\alpha) = \mathtt{ta}_u(\beta)$ then $poss(\mathtt{view}_u \mid Act = \alpha) = poss(\mathtt{view}_u \mid Act = \beta)$.

As we noted above in Section 3.5, a nondeterministic system may be secure with respect to a definition of security that constrains flow of information to individual domains, while allowing flows of information to groups of domains. If the system needs to be protected against collusion, this means that we require a stronger definition that takes into account the deductive capability of groups. One way to approach such a definition is to focus on what the group would know if agents in the group were, *after* the completion of a run of the system, to share what they have observed. We call this a *post-hoc* coalition. The state of information of such a coalition $X \subseteq \mathbb{U}$ may be represented by the function $\langle \mathtt{view}_u \rangle_{u \in X}$. We would like them to be able to deduce no more than they would be able to deduce if, after the run, they were to share their maximal permitted information, represented by the terms $\mathtt{ta}_u(Act(r))$ for u in the group. This leads to the following definition.

Definition 10. A nondeterministic machine M is *nTA-secure w.r.t. post-hoc coalitions (PCnTA)* for a policy $\mapsto$ if, for all nonempty sets $X \subseteq \mathbb{U}$, $\langle \mathtt{view}_u \rangle_{u \in X}$ contains no more information than $\langle \mathtt{ta}_u \rangle_{u \in X} \circ Act$ about Act.

By Proposition 1, an equivalent statement is that for all $X \subseteq \mathbb{U}$ and α, β in A^*, if $\forall u \in X\, (\mathtt{ta}_u(\alpha) = \mathtt{ta}_u(\beta))$ then

$$poss(\langle \mathtt{view}_u \rangle_{u \in X} \mid Act = \alpha) = poss(\langle \mathtt{view}_u \rangle_{u \in X} \mid Act = \beta)\ .$$

One situation in which this attack model is appropriate is a system in which multiple agents submit computations to a cloud server, and receive output only after the computation is complete. In a more interactive setting, the colluding coalition X may be able to do more than share information at the end of the run: they may be able to share information *at each step* of the run. We may call this a *runtime* coalition. Since our systems model is asynchronous, agents in X are typically not able to deduce a linear ordering on the actions performed by members of X from the set of views $\mathtt{view}_u(r)$ for $u \in X$. However, if they are able to communicate each time that an action is performed by one of them, then at the end of the run their information is the joint view $\mathtt{view}_X(r)$, which does contain the order of actions in domains X. This means that reasoning based on $\mathtt{view}_X$ represents a stronger attack model. The following definition attempts to state that the system is resilient to this stronger type of attack.

In order to formulate it, we need to characterize the maximal permitted information for a group X, if they are colluding by sharing information at each step. In the previous definition, the group's maximal permitted information in a run r is represented by the collection of terms $\mathtt{ta}_u(Act(r))$ for $u \in X$. In general, this will be too weak to allow deduction of the order of the actions performed in domains X, whereas as we just noted, the group can deduce this from its joint view $\mathtt{view}_X(r)$. This suggests that we need a stronger definition of the maximal permitted information on this attack model. We take the approach here that if the group cannot

be prevented from sharing information through channels lying outside the system, its maximal permitted information in the event of such collusion is represented by sharing of the maximal permitted information at each step of the computation. This leads us to generalize the definition of $\mathtt{ta}$ to take as a parameter, in place of a single agent u, a nonempty set $X \subseteq \mathbb{U}$. Inductively, we define $\mathtt{ta}_X$ by $\mathtt{ta}_X(\epsilon) = \epsilon$ and, for $\alpha \in A^*$ and $a \in A$,

$$\mathtt{ta}_X(\alpha a) = \begin{cases} (\mathtt{ta}_X(\alpha), \mathtt{ta}_{dom(a)}(\alpha), a) & \text{if } X \cap dom(a)^{\mapsto} \neq \emptyset \\ \mathtt{ta}_X(\alpha) & \text{otherwise} \end{cases}$$

It is easily seen that $\mathtt{ta}_{\{u\}}(\alpha) = \mathtt{ta}_u(\alpha)$, so this is in fact a generalization of the previous notion. Using this notion, we obtain the following definition.

Definition 11. A nondeterministic machine M is *nTA-secure w.r.t. runtime coalitions (RCnTA)* for a policy $\mapsto$ if for all nonempty sets $X \subseteq \mathbb{U}$, $\mathtt{view}_X$ contains no more information than $\mathtt{ta}_X \circ Act$ about Act.

By Proposition 1, an equivalent statement is that for all $X \subseteq \mathbb{U}$ and α, β in A^*, if $\mathtt{ta}_X(\alpha) = \mathtt{ta}_X(\beta)$ then $poss(\mathtt{view}_X \mid Act = \alpha) = poss(\mathtt{view}_X \mid Act = \beta)$.

We have the following straightforward implications between these notions:

Proposition 2. RCnTA implies nTA; PCnTA implies nTA.

These containment relationships are strict, as is shown by the following two examples.

Example 4. For a machine that is RCnTA but not PCnTA, consider M as given in Fig. 6 under the security policy $\Delta_{\{\mathsf{H}_1,\mathsf{H}_2\}}$. The two domains have observations in the set $\{\bot, 0, 1\}$. Each domain H_i has one action, h_i. Below each state s of M there is a label of the form $\begin{smallmatrix} obs_{\mathsf{H}_1}(s) \\ obs_{\mathsf{H}_2}(s) \end{smallmatrix}$ to indicate the observations made by the two domains in s.

Machine M is not only nTA but also RCnTA for $\Delta_{\{\mathsf{H}_1,\mathsf{H}_2\}}$. For instance, for $\alpha = h_1h_2$, $\beta = h_2h_1$, and the coalition $\mathsf{H} = \{\mathsf{H}_1, \mathsf{H}_2\}$ we have that $\mathtt{ta}_{\mathsf{H}}(\alpha) = (\mathtt{ta}_{\mathsf{H}}(h_1), \mathtt{ta}_{\mathsf{H}_2}(h_1), h_2) = ((\epsilon, \epsilon, h_1), \epsilon, h_2) \neq ((\epsilon, \epsilon, h_2), \epsilon, h_1) = \mathtt{ta}_{\mathsf{H}}(\beta)$ and thus it is considered secure that

$$\begin{aligned} &poss(\mathtt{view}_{\mathsf{H}} \mid Act = \alpha) \\ &\quad = \{ \begin{smallmatrix}\bot\\\bot\end{smallmatrix} h_1 \begin{smallmatrix}0\\\bot\end{smallmatrix} h_2 \begin{smallmatrix}0\\0\end{smallmatrix}, \begin{smallmatrix}\bot\\\bot\end{smallmatrix} h_1 \begin{smallmatrix}1\\\bot\end{smallmatrix} h_2 \begin{smallmatrix}1\\0\end{smallmatrix}, \begin{smallmatrix}\bot\\\bot\end{smallmatrix} h_1 \begin{smallmatrix}1\\\bot\end{smallmatrix} h_2 \begin{smallmatrix}1\\1\end{smallmatrix} \} \\ &\quad \neq \{ \begin{smallmatrix}\bot\\\bot\end{smallmatrix} h_2 \begin{smallmatrix}\bot\\0\end{smallmatrix} h_1 \begin{smallmatrix}0\\0\end{smallmatrix}, \begin{smallmatrix}\bot\\\bot\end{smallmatrix} h_2 \begin{smallmatrix}\bot\\1\end{smallmatrix} h_1 \begin{smallmatrix}0\\1\end{smallmatrix}, \begin{smallmatrix}\bot\\\bot\end{smallmatrix} h_2 \begin{smallmatrix}\bot\\1\end{smallmatrix} h_1 \begin{smallmatrix}1\\1\end{smallmatrix} \} \\ &\quad = poss(\mathtt{view}_{\mathsf{H}} \mid Act = \beta) \ . \end{aligned}$$

On the other hand, we have that M is not PCnTA. The individual $\mathtt{ta}_{\mathsf{H}_i}(\alpha) = \mathtt{ta}_{\mathsf{H}_i}(h_i) = \mathtt{ta}_{\mathsf{H}_i}(\beta)$, for $i = 1, 2$. But $r = s_0 \xrightarrow{h_1} s_3 \xrightarrow{h_2} s_6 \in \mathcal{R}(\alpha)$ gives the pair $\langle \mathtt{view}_{\mathsf{H}_1}(r), \mathtt{view}_{\mathsf{H}_2}(r) \rangle = \langle \bot h_1 1, \bot h_2 0 \rangle$ of views. There is no run in $\mathcal{R}(\beta)$ with this pair of views.

We next show that PCnTA does not imply RCnTA.

Example 5. Consider the security policy HNLL consisting of a High domain H and two Low domains L_1 and L_2 who are all allowed to interfere with each other, except that H is not permitted to interfere with any Low domain. (See Fig. 7b.) Let M be the machine depicted in Fig. 7a where the only actions are ℓ_2 in domain L_2 and h in domain H. Only L_1 has non-trivial observations, in $\{0, 1\}$, as indicated by the external labels below states. Machine M is not only nTA but also PCnTA for HNLL. That it is not RCnTA for HNLL can be seen by comparing $\alpha = \ell_2 h$ and $\beta = \ell_2$

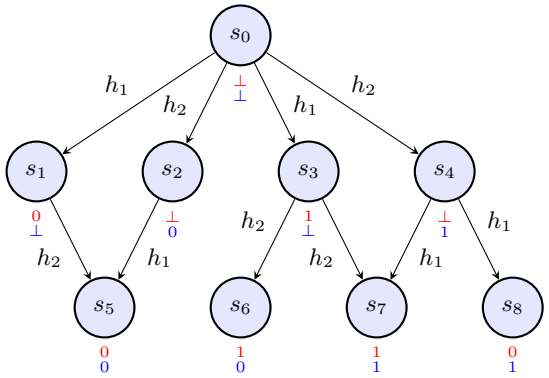
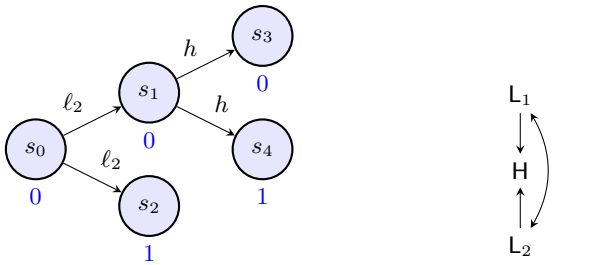


Figure 6: A system that is RCnTA but not PCnTA.



(a) A system that is PCnTA but not RCnTA for HNLL.

(b) Policy HNLL

Figure 7: Example 5.

for the coalition $X = \{\mathsf{L}_1, \mathsf{L}_2\}$. We have $\mathtt{ta}_X(\alpha) = \mathtt{ta}_X(\beta)$, by definition, but

$$\begin{aligned} &poss(\mathtt{view}_X \mid Act = \alpha) \\ &\quad = \{ \begin{smallmatrix}0\\\bot\end{smallmatrix} \ell_2 \begin{smallmatrix}0\\\bot\end{smallmatrix}, \begin{smallmatrix}0\\\bot\end{smallmatrix} \ell_2 \begin{smallmatrix}1\\\bot\end{smallmatrix}, \begin{smallmatrix}0\\\bot\end{smallmatrix} \ell_2 \begin{smallmatrix}0\\\bot\end{smallmatrix} \begin{smallmatrix}1\\\bot\end{smallmatrix} \} \\ &\quad \neq \{ \begin{smallmatrix}0\\\bot\end{smallmatrix} \ell_2 \begin{smallmatrix}0\\\bot\end{smallmatrix}, \begin{smallmatrix}0\\\bot\end{smallmatrix} \ell_2 \begin{smallmatrix}1\\\bot\end{smallmatrix} \} \\ &\quad = poss(\mathtt{view}_X \mid Act = \beta) \ . \end{aligned}$$

Note that this problem does not show up in the views considered separately:

$$\begin{aligned} &poss(\langle \mathtt{view}_u \rangle_{u \in X} \mid Act = \alpha) \\ &\quad = \{\langle 0, \bot \ell_2 \bot \rangle, \langle 01, \bot \ell_2 \bot \rangle\} \\ &\quad = poss(\langle \mathtt{view}_u \rangle_{u \in X} \mid Act = \beta) \end{aligned}$$

Checking the other cases, it can be verified that machine M is PCnTA for HNLL.

Evidently, like the machine of Example 2, the machine of Example 5 should not be considered to be secure on a *causal* interpretation of security, since at state s_1, the occurrence of the H action h causes a change to the L_1 observation, whereas H is not supposed to interfere with L_1.

To obtain notions of security that avoid this difficulty, we can apply the persistence construct of Definition 5. This yields several further notions of security: P-nTA, P-PCnTA and P-RCnTA. Plainly the latter two imply the former, and each P-X implies the source notion X from which it is derived. We now present some examples that show that these notions are all distinct.

The first such example demonstrates that PCnTA and RCnTA but neither of their persistent variants.

Example 6. Reconsider the machine of Example 5, but with an additional edge $\mathsf{H} \mapsto \mathsf{L}_2$ in the security policy. That is, we have an instance of the well-known downgrader policy where D happens to be called L_2. Making the policy more liberal plainly cannot turn a secure system to an insecure system (for any of our definitions) so M is still PCnTA. Moreover, now the counter-example to this from Example 5 no longer works because now we have that $\mathtt{ta}_X(\alpha) = (\mathtt{ta}_X(\ell_2), \mathtt{ta}_{\mathsf{H}}(\ell_2), h) = ((\epsilon, \epsilon, \ell_2), (\epsilon, \epsilon, \ell_2), h)$ differs from $\mathtt{ta}_X(\beta) = (\epsilon, \epsilon, \ell_2)$. It can be verified that M now satisfies RCnTA. However, it is immediate that this example does not satisfy any of the persistent notions of security (in particular, it does not satisfy the weakest of these, P-nTA) because the H action h changes the L_1 observation from state s_1.

The next examples shows that P-nTA does not imply P-RCnTA, and at the same time that P-PCnTA also does not imply P-RCnTA.

Example 7. Recall the policy HNLL (see Fig. 7b) and let M be as depicted in Fig. 8. Only L_1 has non-trivial observations in $\{0, 1\}$, indicated by labels below states. The other domains observe $\perp$ at all states. Actions ℓ_2 and h belong to domains L_2 and H, respectively. Machine M is P-nTA for HNLL, and also P-PCnTA.

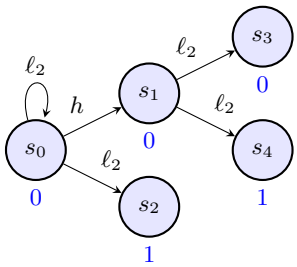


Figure 8: A system that is P-nTA and P-PCnTA but not P-RCnTA.

That it is not P-RCnTA can be seen by comparing $\alpha = h\ell_2\ell_2$ and $\beta = \ell_2\ell_2$ for the coalition $X = \{\mathsf{L}_1, \mathsf{L}_2\}$. We have $\mathtt{ta}_X(\alpha) = ((\epsilon, \epsilon, \ell_2), (\epsilon, \epsilon, \ell_2), \ell_2) = \mathtt{ta}_X(\beta)$ but

$$\begin{aligned} &\mathit{poss}(\mathtt{view}_X \mid \mathit{Act} = \alpha) \\ &\quad = \{\, {}^{0}_{\perp}\ell_2\,{}^{0}_{\perp}\ell_2\,{}^{0}_{\perp},\ {}^{0}_{\perp}\ell_2\,{}^{1}_{\perp}\ell_2\,{}^{1}_{\perp} \,\} \\ &\quad \neq \{\, {}^{0}_{\perp}\ell_2\,{}^{0}_{\perp}\ell_2\,{}^{0}_{\perp},\ {}^{0}_{\perp}\ell_2\,{}^{0}_{\perp}\ell_2\,{}^{1}_{\perp},\ {}^{0}_{\perp}\ell_2\,{}^{1}_{\perp}\ell_2\,{}^{1}_{\perp} \,\} \\ &\quad = \mathit{poss}(\mathtt{view}_X \mid \mathit{Act} = \beta)\ . \end{aligned}$$

This problem vanishes when considering the views separately: both $\mathit{poss}(\langle \mathtt{view}_u \rangle_{u \in X} \mid \mathit{Act} = \alpha)$ and $\mathit{poss}(\langle \mathtt{view}_u \rangle_{u \in X} \mid \mathit{Act} = \beta)$ are equal to $\{\langle 0, \perp\ell_2\perp\ell_2\perp\rangle, \langle 01, \perp\ell_2\perp\ell_2\perp\rangle\}$.

Finally, P-RCnTA implies neither PC-PCnTA nor PCnTA. For this, note that the machine in Example 4 (Fig. 6) is P-RCnTA. We have already argued that it is RCnTA, so the verification of this claim amounts to checking RCnTA on each of the very simple machines that result from choosing any state other than $s0$ as the initial state. This is easily done by inspection. We have already noted above that this machine is not PCnTA, so, a fortiori, it is also not PC-PCnTA.

5. SPECIAL CASES

In this section we reconsider some simple special cases of policies and systems and show that our new definitions of security reduce to some of the existing notions of security discussed above. This lends credibility to the definitions and helps clarify their positioning with respect to the existing literature.

We have started with the aim of generalizing the notion of TA-security on deterministic systems, and introduced a variety of distinct definitions in the more general setting of nondeterministic systems. The following result shows that all of these notions collapse to TA-security in the deterministic setting.

Proposition 3. Given a security policy, the classes of deterministic machines that are nTA-, PCnTA-, or RCnTA-secure are all equal to the class of TA-secure machines. Moreover, the persistent versions of these notions in deterministic machines are also all identical to TA-security.

We have built our definitions of security for intransitive policies using ingredients from prior work on nondeterministic systems with respect to the simple policy $\mathsf{H} \not\mapsto \mathsf{L}$. The following result characterizes our definitions in the special case of this policy (in nondeterministic systems) as collapsing to variants of the notion of Correctability.

Proposition 4. For the policy $\mathsf{H} \not\mapsto \mathsf{L}$, we have that nTA = PC-nTA = RCnTA = COR and P-nTA = P-PCnTA = P-RCnTA = P-COR.

We showed in Example 2 that COR and its persistent version P-COR are distinct. Thus, we have a collapse to two distinct notions in the case of this policy.

6. UNWINDING

Unwinding is a useful technique for security proofs, which reduces the verification of security to checking the existence of binary relations on states. In deterministic systems, unwinding is a sound and complete technique for the proof of NI, and the existence of a weak form of unwinding relations is a sound technique for IP-security and TA-security [8, 21, 24]. We define a generalization of this method that is sound for our definitions of security in nondeterministic systems.

Definition 12. Let $M = (S, s_0, A, \longrightarrow, \mathit{obs}, \mathit{dom})$ be a machine for a security policy $(\mathbb{U}, \mapsto)$; let $\mathcal{S} = (\sim_u)_{u \in \mathbb{U}}$ be a family of equivalence relations on S. For $X \subseteq \mathbb{U}$ define $\sim_X = \bigcap_{u \in X} \sim_u$.

We say that $\mathcal{S}$ is a *generalized weak unwinding on M with respect to* $\mapsto$ (or *GWU*) if the following conditions are met for all $s, s', t \in S$, $u \in \mathbb{U}$, nonempty $X \subseteq \mathbb{U}$, and $a \in A$:

$$s \sim_u t \Rightarrow \mathit{obs}_u(s) = \mathit{obs}_u(t) \qquad \textbf{OC}$$

$$s \xrightarrow{a} t \Rightarrow s \sim_{\mathit{dom}(a) \not\mapsto} t \qquad \textbf{LR}$$

$$s \sim_X t \wedge s \sim_{\mathit{dom}(a)} t \wedge s \xrightarrow{a} s' \Rightarrow \exists t' \left(t \xrightarrow{a} t' \wedge s' \sim_X t' \right) \qquad \textbf{GWSC}$$

Condition **OC** (for "**o**utput **c**onsistency") implies that coalition X can distinguish states that some member can tell apart based on its observation. Condition **LR** (for "**l**ocal **r**espect") implies that actions that are not allowed to interfere with anyone in the coalition cannot change a state to another state that is distinguishable to the coalition. According to **GWSC** ("**g**eneralized **w**eak **s**tep **c**onsistency"), if two states s and t can be distinguished neither by

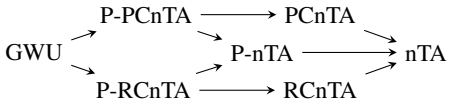


Figure 9: Implications between notions defined in this paper.

coalition X nor by the domain of action a, then each state reachable from s by performing a is indistinguishable to X from some state reachable via a from t. This condition is the main point of difference with the notion of weak unwinding from [21]: it adds to the condition WSC the universal quantification over X (which is just a single domain u in WSC) and the existential quantification over t' (this corresponds to the nondeterminism of the system, whereas WSC is designed for deterministic systems.)

THEOREM 1. *If there is a GWU on M w.r.t. $\mapsto$, then M is PC-nTA and RCnTA w.r.t. $\mapsto$.*

We may note that the definition of GWU is insensitive to the initial state of the machine. Thus, if the existence of a GWU is sound evidence for X-security, then it is also sound evidence for P-X-security. Thus, we also have the following.

Corollary 1. If there is a GWU on M w.r.t. $\mapsto$, then M is both P-PCnTA-secure and P-RCnTA-secure w.r.t. $\mapsto$.

Thus, by the corollary above and our previous results, the existence of a GWU implies every single one of the security notions defined in this paper.

Fig. 9 summarizes the relationships we have found to hold between the various security definitions discussed above.

7. ACCESS CONTROL SYSTEMS

In the preceding sections, we developed definitions of security, and showed that the notion of generalized weak unwinding provides a sound proof technique for showing that these definitions hold in a system. Neither the definitions nor this proof technique provide much concrete guidance for the engineer seeking to construct a secure system, however. A result by Rushby [21] (and refined in [24]) shows that, in deterministic systems, intransitive noninterference is guaranteed to hold in a system built from a collection of objects subject to an access control discipline that satisfies a simple static check that is essentially Bell and La Padula's [2] information flow condition. In the present section, we show that it is possible to generalize this result to nondeterministic systems.

We first recall the notion of a *machine with structured state* [21]. This is a machine M (with states S) together with (we rename some of the components) a set $\texttt{Obj}$ of *objects*, a set V of *values*, and functions $\texttt{contents} : S \times \texttt{Obj} \to V$, with $\texttt{contents}(s, n)$ interpreted as the value of object n in state s, $\texttt{observe}, \texttt{alter} : \mathbb{U} \to \mathcal{P}(\texttt{Obj})$, with $\texttt{observe}(u)$ and $\texttt{alter}(u)$ interpreted as the set of objects that domain u can observe and alter, respectively. For brevity, we write $s(x)$ for $\texttt{contents}(s, x)$. We call the pair $(\texttt{observe}, \texttt{alter})$ the *access control* table of the machine.

Rushby introduced *reference monitor conditions* on such machines in order to capture formally the intuitions associated with the pair $(\texttt{observe}, \texttt{alter})$ being an access control table that restricts the ability of the actions to "read" and "write" the objects $\texttt{Obj}$. In order to formulate a generalization of these conditions in nondeterministic systems, we first need to introduce some further structure that constrains the nondeterminism in the system.

Define a *choose-resolve characterization of nondeterminism* in a machine M with states S and actions A to be a tuple (N, n, r), where N is a set, $n : A \times S \to \mathcal{P}(N)$ and $r : A \times S \times N \to S$, such that the following properties hold:

$$\forall s \in S, a \in A, c \in n(a, s)\left(s \xrightarrow{a} r(a, s, c)\right) \qquad \textbf{CR1}$$

$$\forall s, t \in S, a \in A\left(s \xrightarrow{a} t \Rightarrow \exists c \in n(a, s)\,(t = r(a, s, c))\right) \qquad \textbf{CR2}$$

Intuitively, N represents the set of all possible nondeterministic choices that can be made in the process of executing an action, n restricts those choices as a function of the particular state and action being executed, and r deterministically resolves the transition, as a function of the state, action and nondeterministic choice made in that context. Condition **CR1** says that the resolution of every possible nondeterministic choice, allowed in the context of a given state and action, is in fact a state to which a transition is possible. Conversely, **CR2** says that every transition can be obtained by resolving some allowed nondeterministic choice.

Trivially, every nondeterministic machine has a choose-resolve characterization of nondeterminism. (For, we can simply take $N = S$, and define $n(a, s)$ as $\{\, t \mid s \xrightarrow{a} t \,\}$, and $r(a, s, t) = t$, and **CR1** and **CR2** are then satisfied.) The following restrictions on the structure of the characterization make this notion more interesting.

Suppose the machine M has structured state, with objects $\texttt{Obj}$ and access control table $(\texttt{observe}, \texttt{alter})$. Write $\texttt{Val}(x)$ for the set of all possible values of $x \in \texttt{Obj}$. Say that the choose-resolve characterization of nondeterminism (N, n, r) has *local choices* if for each $x \in \texttt{Obj}$ there exists a set N_x and a function $n_x : A \times \texttt{Val}(x) \to \mathcal{P}(N_x)$ such that $N = \Pi_{x \in \texttt{Obj}} N_x$, and $n(a, s)$ is the set of $c \in N$ such that for all $x \in \texttt{Obj}$, we have $c_x \in n_x(a, s(x))$. That is, a choice of nondeterminism on performing action a in state s is obtained by independently making a choice of nondeterminism at each of the objects $x \in \texttt{Obj}$. Say that a machine has *local nondeterminism* if it is a machine with structured state that has a choose-resolve characterization with local choices.

For states s, t and $u \in \mathbb{U}$ define $s \sim_u^{\texttt{oc}} t$ to hold if $s(x) = t(x)$ for all $x \in \texttt{observe}(u)$. Intuitively, $s \sim_u^{\texttt{oc}} t$ says that u could not distinguish s from t if all it knew were the content of objects that it is permitted to observe. Similarly, for choices $c, c' \in \Pi_{x \in \texttt{Obj}} N_x$, and $u \in \mathbb{U}$ we write $c \sim_u^{\texttt{oc}} c'$ if $c_x = c'_x$ for all $x \in \texttt{observe}(u)$. The following conditions generalize the reference monitor conditions to machines with local nondeterminism.

$$s \sim_u^{\texttt{oc}} t \Rightarrow obs_u(s) = obs_u(t) \qquad \textbf{LC-RM1}$$

$$\left(\begin{array}{l} c \in n(a, s) \wedge c' \in n(a, t) \wedge \\ s \sim_{dom(a)}^{\texttt{oc}} t \wedge c \sim_{dom(a)}^{\texttt{oc}} c' \wedge \\ s(x) = t(x) \wedge c_x = c'_x \end{array}\right) \Rightarrow r(a, s, c)(x) = r(a, t, c')(x)$$

LC-RM2

$$s \xrightarrow{a} t \wedge x \notin \texttt{alter}(dom(a)) \Rightarrow t(x) = s(x) \qquad \textbf{LC-RM3}$$

Intuitively, **LC-RM1** says that the observations of domain u depend only on the values of objects u is permitted to observe. Condition **LC-RM2** says that the value of object x after performing action a depends only on the previous values of objects observable to $dom(a)$, nondeterministic choices made locally at those objects, the previous value of x, and the nondeterministic choice made at x. Condition **LC-RM3** says that, for an action a, if $dom(a)$ is not permitted to alter the value of object x, then the value of x is in fact unaffected when a is performed.

We also have a condition that relates the policy to the access control structure, essentially that identified by Bell and La Padula [2].

$$\texttt{alter}(u) \cap \texttt{observe}(v) \neq \emptyset \Rightarrow u \mapsto v \qquad \textbf{AOI}$$

Intuitively, if there is an object x that can be altered by u and observed by v, then x provides a channel through which information can flow from u to v. Thus, for the system to be secure, this flow must be permitted by the policy. The following result shows that access control provides a design discipline that suffices to enforce our definitions of security.

THEOREM 2. *If M is a machine with local nondeterminism satisfying **LC-RM1**-**LC-RM3** and **AOI** w.r.t. $\mapsto$ then there exists a GWU on M w.r.t. $\mapsto$.*

It follows that such a machine M satisfies all the security properties identified in this paper. The following example illustrates the two theorems.

Example 8. Consider a system comprised of three domains, for a sender $\mathtt{S}$, a receiver $\mathtt{R}$, communicating through an unreliable channel $\mathtt{B}$. We have four objects $\mathtt{Obj} = \{\mathtt{xS}, \mathtt{xI}, \mathtt{xO}, \mathtt{xR}\}$, representing the state of the sender memory, buffer input queue, buffer output queue, and receiver memory, respectively. That the channel is modeled by two queues is motivated by the desire to separate the access rights to the head of the queue from those for the tail. The functionality of the channel is thus limited to the transportation within the buffer, shipping messages from the input queue to the output queue.

Letting $\mathtt{Msg}$ be a set of messages, we may suppose that $\mathtt{Val}(\mathtt{xS}) = \mathtt{Val}(\mathtt{xR}) = \mathtt{Msg}$, so that the memory of the sender or receiver consists of a single message, and $\mathtt{Val}(\mathtt{xI}) = \mathtt{Val}(\mathtt{xO}) = \mathtt{Msg}^*$, so that the states of the buffer queues are sequences of messages. A state of the system as a whole may be represented as a tuple $(m_\mathtt{S}, w, v, m_\mathtt{R}) \in \mathtt{Msg} \times \mathtt{Msg}^* \times \mathtt{Msg}^* \times \mathtt{Msg}$. We suppose that the sender has actions $\mathtt{set}(m)$ (setting its memory to message m) and $\mathtt{put}$ (enqueue the current memory value to the buffer input queue), the buffer has an action $\mathtt{trans}$ (dequeue a message from the input queue and enqueue that message on the output queue), and the receiver has an action $\mathtt{get}$ (dequeue a message from the buffer output queue and store it in the receiver's memory).

We suppose that only the channel is unreliable. It may drop messages in transit from the head (modelled as $\mathtt{xI}$) to the tail ($\mathtt{xO}$). To that end, we take $N_\mathtt{xO} = \{\mathtt{ok}, \mathtt{drop}\}$ consisting of two choices representing normal performance of the transmission action and dropping off the associated message. This is the only nondetermism we need, so we set $N_\mathtt{xI} = N_\mathtt{xS} = N_\mathtt{xR} = \{0\}$. Furthermore, we make all the $n_x(a, s) = \{0\}$ except for $n_\mathtt{xO}(\mathtt{trans}, s) = \{\mathtt{ok}, \mathtt{drop}\}$. The only interesting case of the resolve function is $r(\mathtt{trans}, (m_\mathtt{S}, w \cdot m, v, m_\mathtt{R}), (0, 0, f, 0))$, which is defined as the value $(m_\mathtt{S}, w, m \cdot v, m_\mathtt{R})$ if $f = \mathtt{ok}$, and as $(m_\mathtt{S}, w, v, m_\mathtt{R})$ if $f = \mathtt{drop}$.

We define observations of the domains by $obs_\mathtt{S}((m_\mathtt{S}, w, v, m_\mathtt{R})) = m_\mathtt{S}$, $obs_\mathtt{B}((m_\mathtt{S}, w, v, m_\mathtt{R})) = (w, v)$, and $obs_\mathtt{R}((m_\mathtt{S}, w, v, m_\mathtt{R})) = (v, m_\mathtt{R})$. (We omit the discussion of minor issues such as no effects will take place if $\mathtt{R}$ performs $\mathtt{get}$ in case the output queue is empty.)

Let the access control structure for the system be given by the following table

	observe	alter
S	xS	xS, xI
B	xI, xO	xI, xO
R	xO, xR	xO, xR

which is derived from the descriptions of the actions. Then we have $(m_\mathtt{S}, w, v, m_\mathtt{R}) \sim^{\mathtt{oc}}_\mathtt{R} (m'_\mathtt{S}, w', v', m'_\mathtt{R})$ just when $(v, m_\mathtt{R}) = (v', m'_\mathtt{R})$, from which it follows that $obs_\mathtt{R}((m_\mathtt{S}, w, v, m_\mathtt{R})) = (v, m_\mathtt{R}) = (v', m'_\mathtt{R}) = obs_\mathtt{R}((m'_\mathtt{S}, w', v', m'_\mathtt{R}))$, confirming part of **LC-RM1**. The reader may verify that the remainder of **LC-RM1** holds in this system, as do conditions **LC-RM2** and **LC-RM3**. It is clear from the access control structure that the system also satisfies condition **AOI** for the policy $\mapsto$ defined by $\mathtt{S} \mapsto \mathtt{B} \leftrightarrow \mathtt{R}$.

Thus, by Theorems 2 and 1 the system is P-RCnTA-secure and P-PCnTA-secure w.r.t. $\mapsto$.

We now give a sketch of how our results might apply to the example of the introduction (Figure 1).

Example 9. Within each machine M_i, access control (monitored, e.g., by a security kernel and by hardware access control features) can be used to enforce the local portion of the policy. At this lower, access control level, the system might contain objects $\mathtt{chan}_{u,v}$ representing communication channels for each of the pairs (u, v) in $\{(\mathsf{L}_i, \mathsf{H}_i), (\mathsf{H}_i, \mathsf{E}_i), (\mathsf{E}_i, \mathsf{NI}_i), (\mathsf{L}_i, \mathsf{NI}_i), (\mathsf{NI}_i, \mathsf{N})\}$.

The details of the channels might be reliable or resemble the lossy buffer of Example 8. (For example, one expects that E_1 and L_1 will place their output in a buffer read by the network interface NI_1, and that domain N consists of a network of lossy buffers.) Some of the operations by a domain u on its data components $\mathtt{D}_u$ may also involve nondeterminism, e.g., E_1 may draw on a local source of nondeterminism to generate random nonces to be used as padding in the encrypted messages it constructs.

Suitable access control settings on such nondeterministic components that reside within each machine M_i suffice to enforce the policy. In particular, consider the access control setting for M_1 depicted in Figure 10.

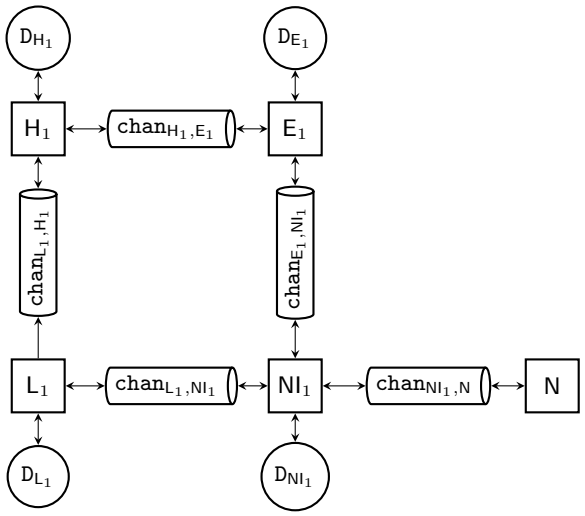


Figure 10: Access Control within machine M_1. A bidirectional arrow $u \leftrightarrow d$ between a domain u and a data object d denotes $d \in \mathtt{observe}(u) \cap \mathtt{alter}(u)$ whereas a unidirectional arrow $u \rightarrow d$ denotes $d \in \mathtt{alter}(u)$.

Note that all edges indicated are bidirectional, except the edge from domain L_1 to object $\mathtt{chan}_{\mathsf{L}_1,\mathsf{H}_1}$. This ensures that L_1 may pass information to H_1, but not vice versa. It is straighforward to check that this access control policy ensures satisfaction of AOI with respect to the policy in Figure 1.

8. RELATED WORK

The original work on semantics of intransitive noninterference [10, 21] was confined to transitive policies and deterministic systems. We have already noted, in Section 3, some of the main points of connection between our contributions in this paper and the voluminous literature on the transitive policy $\mathsf{H} \not\mapsto \mathsf{L}$ in nondeterministic

systems. We focus here on comparing our work to that of some others who have considered semantics for intransitive policies specifically in the setting of nondeterministic systems.

The earliest work in this vein appears to be that of Bevier and Young [3]. Unlike most of the subsequent work, Bevier and Young are alert to the issue of collusion: they present an example to justify a component of their definition as necessary in order to deal with inferences by groups of agents. Moreover, their definition implies a type of persistence, in that it places requirements on future views from all pairs of states related by an equivalence relation (hence also from all individual states compared to themselves.) The equivalence relations used are the concrete relations defined by $s \sim_X t$ iff $obs_u(s) = obs_u(t)$ for all $u \in X$, where X is a set of domains. Additionally, their definition refers to versions of the intransitive purge functions $\mathtt{ip}_u$ and the equivalence relation and $\mathtt{view}_u(r) = \mathtt{view}_u(r')$ that generalize the single domain u to a *list* of domains. Unfortunately, a definition of neither generalization is provided. Our distinction between post-hoc and run-time coalitions shows that there are in fact several plausible candidates for such a generalization — it is unclear which approach was intended by Bevier and Young. They also give a corresponding definition of unwinding, which is also based on the specific concrete equivalence relation of observation identity (although they discuss the fact that the analyst might tailor the definition of observation). Thus, although this work is close ours in the mix of concerns factored into the definition of security, it differs in the use of a specific concrete equivalence relation and its basis in `ip` rather than `ta`, and failed to notice subtleties concerning the treatment of coalitions.

Roscoe and Goldsmith [19] considered the application of the notion of *local determinism* within the setting of the process algebra CSP to give semantics to intransitive policies: they give two versions, one using a *lazy* abstraction operator, the other using a *mixed* operator that gives special treatment to *signal events*. It is shown in [23] that, in the special case of deterministic systems, the lazy abstraction approach corresponds to the application of the purge function of NI to nondeterministic systems, and the mixed abstraction approach corresponds to a notion of security van der Meyden calls ITO-security: this differs from TA security in that it uses functions that are like $\mathtt{ta}_u$, but which track information about observations as well as information about actions. Thus, both definitions differ from the TA-based definitions we have presented. The issue of collusion is not considered in [19], however, semantics of a policy is given by checking for flows of information to each individual domain u from the noninterfering domains $\not\mapsto u$, considered jointly.

Mantel [13] enriches the type of security policies we have considered (by distinguishing between observable, deducible and noninterfering events) and studies the extension of his (trace-based) Basic Security Property compositional approach to definitions of security for $\mathsf{H} \not\mapsto \mathsf{L}$ to this richer policy setting. His definitions make use of a restricted quantification over sets of domains. However, the intended effect of this quantification is not to capture collusion attacks, but to spread the local purge-like effect of the basic security properties to simulate `ip`-like noninterference conditions. He also proves soundness of an unwinding proof technique.

Another definition of intransitive noninterference in nondeterministic systems is given by von Oheimb [26]. His approach is also based on the `ip` function, but relates this to final observations in a run. As we noted in Section 3.3, this may be inappropriate in nondeterministic systems. A notion of unwinding is also presented that is similar to ours, in that it quantifies over sets of domains. However, the motivation for this appears to have been to obtain a soundness result for unwinding, rather than an overt recognition of the possibility of collusion attacks, since these are not acknowledged in the definition of security.

Bismulation-based definitions of security (an approach that is closely related to unwinding) are considered in the context of the very specific intransitive policy HDL in [5]. This work deals with the persistence dimension, but it is not clear how to generalize the definitions to policies richer than HDL. Gorrieri and Vernali [9] define similar notions of noninterference for the $\mathsf{H} \not\mapsto \mathsf{L}$ and HDL policies on labelled transition systems and elementary Petri nets: the emphasis here is on the particularities of these semantic models rather than essential novelty in the semantics of security, where they essentially follow [5].

Finally, Backes and Pfitzman [1] go beyond the nondeterministic setting to consider intransitive policies in systems with probabilistic transitions and cryptographic notions of information-flow. They give a number of definitions that take a rather different approach from the rest of the literature, focussing on whether information about an initial bit can be transmitted from one party to another, in some sense "only via particular intermediaries". It would take some research to clarify relationships to any of the definitions in the literature, but, prima facie, it would seem that these definitions do not constrain information flow after the first transmission by the intermediaries, as do all the other definitions.

Of the works discussed above, only von Oheimb applies his results to a class of Access Control systems, but these are just Rushby's *deterministic* Access Control systems, whereas we have introduced a more general non-deterministic class of such systems.

9. CONCLUSION

Our focus in this paper has been to present a set of generalizations of the TA-security semantics for intransitive noninterference in nondeterministic systems. We have teased out a number of parameters of such generalizations, leading to a range of definitions. We characterize the relationships between these definitions and provide an unwinding proof technique that is sound for all.

Several issues are left open by this work. The TA style of semantics concentrates of the possibility for attackers to make deductions about the actions that have been performed. In a nondeterministic setting, the observations nondeterministically resulting from these actions may also need to be protected against inference attacks (e.g., when these observations are randomly generated keys). Some TA-like definitions (TO-security and ITO-security) that take observations into account have been presented by van der Meyden [24]. It remains to work out how these definitions should be generalized to the nondeterministic setting.

Another concern is that whereas unwinding proof techniques should preferably be complete as well as sound (and some in the literature achieve both) our technique is just sound. It would be of interest to have sound and complete proof techniques for each of our definitions, in order to cover examples that lie outside of their intersection. Given the connection we establish between P-nTA and 0-forward correctability, ideas from Millen's sound and complete unwinding for forward correctability [16] may be appropriate in this case.

10. REFERENCES

[1] M. Backes and B. Pfitzmann. Intransitive non-interference for cryptographic purposes. In *Proc. IEEE Symp. on Security and Privacy*, pages 140–152, 2003.

[2] D. Bell and L. L. Padula. Secure computer system: unified exposition and multics interpretation. Technical Report ESD-TR-75-306, Mitre Corporation, Bedford, M.A., Mar. 1976.

[3] W. Bevier and W. Young. A state-based approach to noninterference. *Journal of Computer Security*, 3(1):55–70, 1995. (an earlier version appears in CSFW'94).

[4] C. Boettcher, R. DeLong, J. Rushby, and W. Sifre. The MILS component integration approach to secure information sharing. In *Proc. 27th IEEE/AIAA Digital Avionics Systems Conference*, pages 1.C.2–1–1.C.2–14, Oct. 2008.

[5] A. Bossi, C. Piazza, and S. Rossi. Modelling downgrading in information flow security. In *Proc. IEEE Computer Security Foundations Workshop*, pages 187–201, 2004.

[6] R. Focardi and S. Rossi. Information flow security in dynamic contexts. In *Proc. IEEE Computer Security Foundations Workshop*, pages 307–319, 2002.

[7] J. Goguen and J. Meseguer. Security policies and security models. In *Proc. IEEE Symp. on Security and Privacy*, pages 11–20, 1982.

[8] J. Goguen and J. Meseguer. Unwinding and inference control. In *IEEE Symp. on Security and Privacy*, pages 75–87, 1984.

[9] R. Gorrieri and M. Vernali. On intransitive non-interference in some models of concurrency. In *Foundations of Security Analysis and Design VI - FOSAD Tutorial Lectures*, volume 6858 of *LNCS*, pages 125–151. Springer-Verlag, 2011.

[10] J. Haigh and W. Young. Extending the noninterference version of MLS for SAT. *IEEE Trans. Softw. Eng.*, SE-13(2):141–150, Feb. 1987.

[11] J. Halpern and K. O'Neill. Secrecy in multiagent systems. In *Proc. IEEE Computer Security Foundations Workshop*, page 32, Los Alamitos, CA, USA, 2002. IEEE Computer Society.

[12] D. M. Johnson and F. J. Thayer. Security and the composition of machines. In *Proc. IEEE Computer Security Foundations Workshop*, pages 72–89, 1988.

[13] H. Mantel. Information flow control and applications - bridging a gap. In J. N. Oliveira and P. Zave, editors, *FME*, volume 2021 of *LNCS*, pages 153–172. Springer-Verlag, 2001.

[14] J. McLean. Reasoning about security models. In *Proc. IEEE Conf. on Security and Privacy*, pages 123–131, 1987.

[15] J. McLean. A general theory of composition for trace sets closed under selective interleaving functions. In *Proc. IEEE Symp. on Security and Privacy*, pages 79–93, May 1994.

[16] J. K. Millen. Unwinding forward correctability. In *Proc. IEEE Computer Security Foundations Workshop*, pages 2–10, 1994.

[17] S. M. More, P. Naumov, B. Nicholls, and A. Yang. A ternary knowledge relation on secrets. In *Proc. Conf. on Theoretical Aspects of Rationality and Knowledge (TARK-2011), Groningen, The Netherlands, July 12-14, 2011*, pages 46–54, 2011.

[18] J. Mullins. Nondeterministic admissible interference. *Journal of Universal Computer Science*, 6(11):1054–1070, 2000.

[19] A. Roscoe and M. Goldsmith. What is intransitive noninterference? In *Proc. IEEE Computer Security Foundations Workshop*, pages 228–238, 1999.

[20] J. Rushby. Design and verification of secure systems. In *Proc. 8th Symposium on Operating Systems Principles*, pages 12–21, Asilomar CA, Dec. 1981. (ACM Operating Systems Review, Vol 15, No. 1).

[21] J. Rushby. Noninterference, transitivity, and channel-control security policies. Technical Report CSL-92-02, SRI International, Dec. 1992.

[22] D. Sutherland. A model of information. In *Proc. 9th National Computer Security Conf.*, pages 175–183, 1986.

[23] R. van der Meyden. A comparison of semantic models for intransitive noninterference. unpublished manuscript, available at `http://www.cse.unsw.edu.au/~meyden`, Dec. 2007.

[24] R. van der Meyden. What, indeed, is intransitive noninterference? In J. Biskup and J. Lopez, editors, *Proc. European Symposium On Research In Computer Security (ESORICS)*, volume 4734 of *LNCS*, pages 235–250. Springer-Verlag, 2007.

[25] R. van der Meyden. On notions of causality and distributed knowledge. In *Proc. 11th Int. Conf. on Principles of Knowledge Representation and Reasoning*, pages 209–219, 2008.

[26] D. von Oheimb. Information flow control revisited: Noninfluence = Noninterference + Nonleakage. In *Proc. European Symposium On Research In Computer Security (ESORICS)*, volume 3193 of *LNCS*, pages 225–243. Springer-Verlag, 2004.