Information flow control revisited: Noninfluence = Noninterference + Nonleakage

David von Oheimb

Siemens CT IC Sec, Munich

Abstract. We revisit the classical notion of noninterference for state-based systems, as presented by Rushby in 1992. We strengthen his results in several ways, in particular clarifying the impact of transitive vs. intransitive policies on unwinding. Inspired partially by Mantel's observations on unwinding for event systems, we remove the restriction on the unwinding relation to be an equivalence and obtain new insights in the connection between unwinding relations and observational preorders.
Moreover, we make two major extensions. Firstly, we introduce the new notion of nonleakage, which complements noninterference by focusing not on the observability of actions but the information flow during system runs, and then combine it with noninterference, calling the result noninfluence. Secondly, we generalize all the results to (possibilistic) nondeterminism, introducing the notions of uniform step consistency and uniform local respect. Finally, we share our experience using nonleakage to analyze the confidentiality properties of the Infineon SLE66 chip.
Like Rushby's, our theory has been developed and checked using a theorem prover, so there is maximal confidence in its rigor and correctness.

1 Introduction

Noninterference, a very strong, abstract and mathematically elegant way of expressing secrecy, has been introduced more than two decades ago by Goguen and Meseguer [GM82,GM84]. Since then, a large body of work ([Sut86,Fol87,McC90], [Rya90,McL94,ZL97,Man00], etc.) has grown generalizing noninterference to nondeterministic systems, leading to a variety of definitions that partially coincide or exhibit subtle differences. A systematic overview is given by Mantel [Man03].

A further dimension of generalization is towards information flow policies that are not transitive, used for describing downgrading, information filters and channel control. The notion of *intransitive noninterference* [HY86,Rus92,Pin95] has caused some difficulties and debate. Also Roscoe and Goldsmith [RG99] motivate and explain intransitive noninterference, but on the other hand add new confusion: they criticize the classical definition via the *ipurge* function, giving examples of wrongly implemented downgraders and attributing the problem to limitations of the expressiveness of purging, yet the actual problem is at least in part due to the semantical discrepancies (i.e., the implementation errors) that they assume and not due to intransitivity. Nevertheless, we believe that intransitive noninterference is a notion both valid and practically useful, thus we make sure not to limit ourselves to the transitive case.

Many approaches focus on event systems and therefore typically use variants of process algebras like CSP as the underlying system model. Yet most systems that appear in practice are heavily state-oriented. Consequently, when it comes to their security, the flow of information contained in the system state often is the only thing that really matters or is at least as important as the visibility of actions or I/O events. Though state-oriented systems can be described in event-oriented formalisms, we feel that this is not very natural. Our aim is to have a theory of secure information flow as simple and abstract as possible that models state-oriented systems directly, hence we use plain state automata as the underlying system model. Furthermore, we require verification techniques like unwinding theorems implemented in a theorem proving system usable for practical security analysis. Moreover, we want to cover both deterministic and (non-total) nondeterministic systems and both transitive and intransitive policies. We are not aware of a theory that meets all these requirements — for instance, both McCullough's restrictiveness [McC90] and the contemporary language-based security [SM03] handle deterministic and nondeterministic state-based systems but not intransitive policies. So we decided to develop our own theory.

Rushby's work [Rus92], using simple automata and handling intransitive noninterference yet not nondeterminism, appeared as a good starting point. We have implemented a slight variant of his theory, using the state-of-the-art theorem prover Isabelle/HOL [NPW02], removing some unnecessary limitations, extending it as straightforwardly as possible to nondeterministic systems, and introducing a hierarchy of variants of noninterference that concentrate to various extents on information flow between domains. The complete theory sources including proofs are available online [Ohe04]. There is some closely related work by Mantel [Man01,Man03] that also handles nondeterminism and unwinding for intransitive noninterference. It deals with a large number of variants of noninterference in a systematic way, but is event-oriented and not implemented in a theorem prover. We give more comments on the similarities and differences to his approach in the course of presenting our development.

2 System model

We use two simple automaton models for describing systems as deterministic or nondeterministic state machines. Each action a of type *action* transforms the *state* via a total transition function $step(a)$, following Rushby [Rus92], or a (possibly partial and nonfunctional) transition relation $Step(a)$, respectively. [1]

$step : action \times state \to state$
$Step : action \to \wp(state \times state)$

Runs of a system are described by lifting steps over action sequences.

$run : action^* \times state \to state$
$Run : action^* \to \wp(state \times state)$

[1] Whenever there is a direct correspondence between the deterministic and the nondeterministic case, we use the same names except that the versions for the latter case are capitalized in part.

They are defined by straightforward primitive recursion:

def $run([], s) \equiv s$
def $run(a \frown \alpha, s) \equiv run(\alpha, step(a, s))$
def $Run([]) \equiv \{(s, s) |\ True\}$
def $Run(a \frown \alpha) \equiv \{(s, t) |\ \exists s'.\ (s, s') \in Step(a)\ \wedge\ (s', t) \in Run(\alpha)\}$

In the above definitions, '[]' stands for the empty sequence and '$a \frown \alpha$' denotes the action sequence α with the action a prepended to it.

The initial system state, used for defining classical noninterference even in the nondeterministic case, is denoted by s_0.

Each action is associated with a security *domain* used to describe both restrictions on its own visibility and the portion of the state it may read from.

$dom :\ action\ \rightarrow\ domain$

There is an output function defined on *state* yielding some value(s) of type *output*. Output is not associated with an action but with the state alone, i.e., we model systems as Moore automata. In order to express the observations possible for each security domain, the output function receives an extra parameter of type *domain*. In this respect, we deviate slightly from Rushby's system model which uses Mealy automata where output depends on the security domain indirectly via the domain associated with actions.

$output :\ domain\ \times\ state\ \rightarrow\ output$

We use *domain* instead of *action* as the extra parameter of *output* because this slightly simplifies some formulations and allows both a more direct interpretation of access control and an easier comparison with nonleakage.

3 Generic notions

3.1 Policies

Central to noninterference and its derivatives is the notion of a *policy*,

$\cdot \rightsquigarrow \cdot : \wp(domain\ \times\ domain)$

also called *interference relation*. It expresses that information is allowed to flow from the first domain to the second. The complementing relation, $\not\rightsquigarrow$, is called *noninterference relation*. Classical multi-level security polices induce a transitive interference relation, but others are intransitive: they allow indirect information flow via privileged channels like a censoring downgrader or an encryption engine while a direct (short-circuit) flow is prohibited.

As usual, we globally assume reflexivity of policies: $\forall u.\ u \rightsquigarrow u$. Other assumptions are of local nature. For instance, for part of our results, transitivity of policies is required (but only where explicitly stated).

3.2 Allowed source domains

In order to express the allowed information flow between domains for (in general) intransitive policies, we employ the auxiliary function *sources* [Rus92]. It takes

a sequence of actions α and a target domain u and yields the set of domains that are allowed to pass information to u immediately or indirectly via (a subsequence of) α. The following definition is equivalent to the classical one:

$$sources: \ action^* \ \times \ domain \ \rightarrow \ \wp(domain)$$
$$\textbf{def } sources([], u) \equiv \{u\}$$
$$\textbf{def } sources(a \frown \alpha, u) \equiv sources(\alpha, u) \cup \{w |\ \exists v.\ dom(a) = w \ \wedge \ w \rightsquigarrow v \ \wedge \ v \in sources(\alpha, u)\}$$

For example, a sufficient condition for $v \in sources(a_1 \frown a_2 \frown a_3 \frown a_4 \frown [], u)$ is $v = dom(a_2) \ \wedge \ dom(a_2) \ \rightsquigarrow \ dom(a_4) \ \wedge \ dom(a_4) \ \rightsquigarrow u$ (even if $v \not\rightsquigarrow u$).

For defining weak nonleakage in §5.3, we will use a second variant called *chain*. It can be derived from the above definition by leaving out the restriction $dom(a) = w$ on the chain elements. This variant yields all domains connected with the given domain via the relation $\{(u, u) |\ True\} \cup \rightsquigarrow \cup \ \rightsquigarrow^2 \ \cup \ldots \cup \ \rightsquigarrow^n$ where n is the length of the action sequence given as the first argument.

Obviously, *sources* yields a subset of the result of *chain*, i.e. $sources(\alpha, u) \subseteq chain(\alpha, u)$. Moreover, in the case of transitive policies, *chain* yields a subset of $\rightsquigarrow$, in the sense that $chain(\alpha, u) \subseteq \{w |\ w \rightsquigarrow u\}$.

3.3 Unwinding relations

Central for both the well-known unwinding results and our extensions is the *unwinding relation* on states, parameterized by the observing domain u:

$$\cdot \dot{\sim} \cdot : \ domain \ \rightarrow \ \wp(state \ \times \ state)$$

Classically, this relation is a *view-partitioning* equivalence [Rus92,ZM01] expressing indistinguishability of states from u's perspective. In most applications, it is simply a pointwise equation on the contents of those variables that u may observe. Zdancewic and Myers [ZM01] call this a *view* of a system S and use it for defining an observational equivalence, $S[\approx]$, on stuttering-equivalent traces. In order to maintain confidentiality of information contained in the system state, the unwinding relation is to be preserved locally by every computation step.

Regarding the unwinding relation to be an equivalence is intuitive and valid for most cases, yet Mantel pointed out that unwinding does not really require symmetry [Man00], neither reflexivity nor transitivity [Man03], and that in some applications the relation is e.g. intransitive[2]. Inspired partially by his results, we allow for arbitrary unwinding relations as long as they imply the observational equivalence (or preorder, respectively) induced by the *output* function. Only for the unwinding theorems of noninterference, it has to be assumed that for all observers the initial state is in unwinding relation with itself: $\forall u.\ s_0 \overset{u}{\sim} s_0$

The unwinding relation is lifted in the canonical way to sets of domains, inheriting any reflexivity, symmetry, and transitivity properties.

$$\cdot \overset{\cdot}{\approx} \cdot : \ \wp(domain) \ \rightarrow \wp(state \ \times \ state)$$
$$\textbf{def } s \overset{U}{\approx} t \equiv \forall u \in U.\ s \overset{u}{\sim} t$$

[2] This is not to be confused with intransitivity of the information flow policy.

4 Noninterference

In this section, we slightly improve the theory of noninterference as presented by Rushby in [Rus92]. Moreover, we extend the results to nondeterministic systems. Also in the two subsequent sections introducing nonleakage and noninfluence, we will handle both the deterministic and the nondeterministic case, displaying the inherent parallelism between the two cases as far as appropriate.

4.1 Purging

We define the classical purge function filtering out confidential events with respect to (generally) intransitive policies, as introduced in [Rus92]:

$$ipurge : domain \times action^* \to action^*$$
def $ipurge(u, []) \equiv []$
def $ipurge(u, a \frown \alpha) \equiv if\ dom(a) \in sources(a \frown \alpha, u)$
$$then\ a \frown ipurge(u, \alpha)\ else\ ipurge(u, \alpha)$$

For example, $ipurge(a_1 \frown a_2 \frown a_3 \frown a_4 \frown [], u) = a_2 \frown a_4 \frown []$ if a_1 and a_3 may not directly nor indirectly (i.e., via any of their successors in the given chain of actions) influence u, but if a_4 does so directly (i.e., $dom(a_4) \rightsquigarrow u$ holds) and a_2 indirectly, via $dom(a_2) \rightsquigarrow dom(a_4)$. Generally, *ipurge* enjoys properties like

lemma $sources\_ipurge : sources(ipurge(u, \alpha), u) = sources(\alpha, u)$
lemma $ipurge\_idempotent : ipurge(u, ipurge(u, \alpha)) = ipurge(u, \alpha)$

If we replace the condition $dom(a) \in sources(a \frown \alpha, u)$ by $dom(a) \rightsquigarrow u$, we obtain the simpler variant typically used for transitive policies, which we call *tpurge*. It can be shown that there is a very intuitive characterization of *tpurge*, namely: $tpurge(u, \alpha)$ removes from α all actions a with $dom(a) \not\rightsquigarrow u$. Moreover, as already stated by Rushby, *tpurge* coincides with *ipurge* in the case of transitive policies. Therefore, in the following we will use only *ipurge* because it covers both the transitive and the general (possibly intransitive) case.

4.2 The deterministic case

General version The essence of noninterference is that an observer cannot tell the difference between any system run and the variant of it obtained by removing ("purging") all events that he is not allowed to notice directly or indirectly. In order to formulate this notion and its derivatives in a concise way, we first define an *observational equivalence* relation on the state with an associated action sequence. The equivalence is parameterized by the observing domain and is induced by the *output* function applied to the final state after executing the respective action sequence.

$$\cdot \lhd \cdot \stackrel{\cdot}{\simeq} \cdot \lhd \cdot : domain \to \wp(state \times action^* \times state \times action^*)$$
def $s \lhd \alpha \stackrel{u}{\simeq} t \lhd \beta \equiv output(u, run(\alpha, s)) = output(u, run(\beta, t))$

Using this relation, classical noninterference can be written as
def $noninterference \equiv \forall \alpha\ u.\ s_0 \lhd \alpha \stackrel{u}{\simeq} s_0 \lhd ipurge(u, \alpha)$

Unwinding reduces this global security property to a set of local, step-wise properties, in particular the two complementing ones introduced in [Rus92]:

- *step consistency* : $s \overset{u}{\sim} t \longrightarrow step(a,s) \overset{u}{\sim} step(a,t)$, preserves the unwinding relation for each action a. Its meaning is that the effects of executing a on s and t as far as observable by the domain u, expressed by $step(a,s) \overset{u}{\sim} step(a,t)$, may depend on the previous values in the states s and t observable by u, expressed by $s \overset{u}{\sim} t$, but on nothing else. This property is used in the case that the domain of the action to be performed, $dom(a)$, is allowed to interfere with the observing domain u, i.e. $dom(a) \rightsquigarrow u$.
- *local respect* : $dom(a) \not\rightsquigarrow u \longrightarrow s \overset{u}{\sim} step(a,s)$ handles the opposite case.

We weaken (thus effectively generalize) Rushby's definition of step consistency:
def *weakly_step_consistent* $\equiv$ [3]
$\forall a\ u\ s\ t.\ dom(a) \rightsquigarrow u\ \wedge\ s \overset{dom(a)}{\sim} t\ \wedge\ s \overset{u}{\sim} t \longrightarrow step(a,s) \overset{u}{\sim} step(a,t)$
by adding two premises which make step consistency easier to establish in applications. Firstly, $dom(a) \rightsquigarrow u$ states that action a is allowed to interfere with the observing domain u. This is just an enhancement of convenience because in the other case, $dom(a) \not\rightsquigarrow u$, the property can be obtained from local respect.

Secondly, the premise $s \overset{dom(a)}{\sim} t$ is present not only in the intransitive case, as it was before. It allows that the effects of a depend also on the observables of $dom(a)$, the "official" input domain that the action a may read from. In Figure 1,

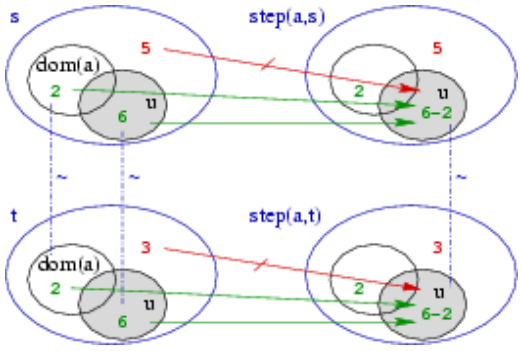

Fig. 1. data flow if $dom(a) \rightsquigarrow u$

the large ovals give an extensional view of all variables in the system state. The small ovals describe (possibly overlapping) subsets of them — the standard interpretation of security domains. The solid arrows depict allowed information flow into the domain u induced by a, while the dashed arrow depicts forbidden flow. Since the information flow from $dom(a)$ additionally allowed here is both very natural and important in applications, the "ordinary" step consistency is too strong. Adding the extra premise also for the transitive case is sound, as our main unwinding theorem confirms. Rushby does not state this sharpened result, although he compares the transitive and the intransitive case in detail.

[3] We adopt the convention that '$\wedge$' binds stronger than '$\longrightarrow$'

Like Mantel [Man00], we split Rushby's notion of local respect into a left-hand and right-hand variant, which allows us to remove the symmetry constraint on the unwinding relation. Moreover, we transform local respect such that it preserves rather than introduces the unwinding relation. This not only relieves us from requiring transitivity of the unwinding relation, but also yields a stronger observational preorder for the nondeterministic case than Mantel's (see §5.6).

def $local\_respect\_left \equiv \forall a\ u\ s\ t.\ dom(a)\ \not\rightsquigarrow\ u\ \wedge\ s \overset{u}{\sim} t \longrightarrow step(a,s) \overset{u}{\sim} t$

def $local\_respect\_right \equiv \forall a\ u\ s\ t.\ dom(a)\ \not\rightsquigarrow\ u\ \wedge\ s \overset{u}{\sim} t \longrightarrow s \overset{u}{\sim} step(a,t)$

def $local\_respect \equiv local\_respect\_left\ \wedge\ local\_respect\_right$

One can show easily that under the assumption that the unwinding relation is an equivalence, these definitions coincide with the classical one recalled above.

In the proof of the unwinding theorem for both noninterference and nonleakage, a consequence of local respect is used that is structurally very similar to step consistency, but handles the case $dom(a)\ \not\rightsquigarrow\ u$. We call it *step respect*.

def $step\_respect \equiv \forall a\ u\ s\ t.\ dom(a)\ \not\rightsquigarrow\ u\ \wedge\ s \overset{u}{\sim} t \longrightarrow step(a,s) \overset{u}{\sim} step(a,t)$

Obviously, the combination of the left-hand and right-hand variants of local respect implies step respect: $local\_respect \longrightarrow step\_respect$

Assuming $\forall u.\ s_0 \overset{u}{\sim} s_0$ and employing *output consistency*, which is defined as

def $output\_consistent \equiv \forall u\ s\ t.\ s \overset{u}{\sim} t \longrightarrow output(u,s) = output(u,t)$

we can prove the main unwinding theorem for noninterference:

theorem *noninterference* :
$weakly\_step\_consistent \wedge local\_respect \wedge output\_consistent \longrightarrow noninterference$

This theorem is essentially the same as [Rus92, Theorem 7] except that is not (unnecessarily) restricted to intransitive policies. Appendix A gives an abstract example of using its extra strength.

Strong version Strictly speaking, the classical notion of noninterference only states that an observer cannot deduce that the subsequence of all actions that he is not allowed to see has occurred. This is because purging *removes all* unsuitable actions from a given sequence, but not *part of* them, and neither *adds* any such actions. This shortcoming can be repaired by using the canonical strong version of noninterference (cf. e.g. [McC90,Rya90]) that handles arbitrary insertion and deletion of secret actions:

def $strong\_noninterference \equiv$
$$\forall \alpha\ u\ \beta.\ ipurge(u,\alpha) = ipurge(u,\beta) \longrightarrow\ s_0 \triangleleft \alpha \overset{u}{\simeq} s_0 \triangleleft \beta$$

Taking $\beta = ipurge(u,\alpha)$, it is easy to see from the idempotence of purging that this version implies the original version of noninterference. On the other hand, one can derive the strong version of noninterference from the standard right-hand one, exploiting symmetry and transitivity of the observational equivalence. Thus one can conclude that in the deterministic case, the strong version is not strictly stronger than the classical one.

The just mentioned proof scheme does not work for the nondeterministic case because there the observational preorder (see below) is not symmetric. Therefore, we prefer to prove the corresponding "strong" unwinding theorem directly.

theorem *strong_noninterference* : *weakly_step_consistent* $\wedge$
local_respect $\wedge$ *output_consistent* $\longrightarrow$ *strong_noninterference*

4.3 The nondeterministic case

After having elaborated classical noninterference and unwinding in the case of automata with total deterministic transitions, we now generalize these results to partial nondeterministic transitions, trying to parallel the above development as far as possible. This kind of transitions is the one that typically occurs in the abstract system models that we develop during our security analysis projects. It imposes two extra challenges to security: because of partiality, the enabledness of action sequences now plays a role in observations, and because of nonfunctionality, preservation of the unwinding relation becomes more difficult.

For the reasons just given, we extend the observational equivalence of §4.2 with the preservation of enabledness, obtaining an *observational preorder*:

$$\cdot \lhd \cdot \stackrel{\cdot}{\underset{\sim}{}} \cdot \lhd \cdot : \ domain \ \rightarrow \wp(state \ \times \ action^* \ \times \ state \ \times \ action^*)$$

def $s \lhd \alpha \stackrel{u}{\underset{\sim}{}} t \lhd \beta \equiv \forall s'.\ (s,s') \in Run(\alpha) \ \longrightarrow$
$\exists t'.\ (t,t') \in Run(\beta) \ \wedge \ output(u,s') = output(u,t')$

Using this relation, we define noninterference for nondeterministic systems by the canonical generalization of strong noninterference (cf. [McC90, III]):

def $Noninterference \equiv \forall \alpha \ u \ \beta.\ ipurge(u,\alpha) = ipurge(u,\beta) \longrightarrow s_0 \lhd \alpha \stackrel{u}{\underset{\sim}{}} s_0 \lhd \beta$

Simple version The immediate generalization of step consistency, step respect, and local respect for nondeterministic systems is straightforward. [4] Here we define only "ordinary" rather than weak step consistency, for reasons given below. There is some similarity of these definitions with weak bisimulation, as explained e.g. in [Rya01, §10].

def $Step\_consistent \equiv \forall a \ u \ s \ s' \ t.\ dom(a) \rightsquigarrow u \ \wedge$
$(s,s') \in Step(a) \ \wedge \ s \stackrel{u}{\sim} t \longrightarrow (\exists t'.\ (t,t') \in Step(a) \ \wedge \ s' \stackrel{u}{\sim} t')$

def $Step\_respect \equiv \forall a \ u \ s \ s' \ t.\ dom(a) \not\rightsquigarrow u \ \wedge$
$(s,s') \in Step(a) \ \wedge \ s \stackrel{u}{\sim} t \longrightarrow (\exists t'.\ (t,t') \in Step(a) \ \wedge \ s' \stackrel{u}{\sim} t')$

def $Local\_respect\_left \equiv$
$\forall a \ u \ s \ s' \ t.\ dom(a) \not\rightsquigarrow u \ \wedge s \stackrel{u}{\sim} t \ \wedge \ (s,s') \in Step(a) \longrightarrow s' \stackrel{u}{\sim} t$

def $Local\_respect\_right \equiv$
$\forall a \ u \ s \ t.\ dom(a) \not\rightsquigarrow u \ \wedge s \stackrel{u}{\sim} t \longrightarrow (\exists t'.\ (t,t') \in Step(a) \ \wedge \ s \stackrel{u}{\sim} t')$

Obviously, *Local_respect_left* and *Local_respect_right* implies *Step_respect*.

[4] Actually, it would be sufficient to state them only for reachable states s and t. We refrain from doing so in order to avoid extra clutter in the presentation.

Note that for consistency with the preservation of enabledness, the left-hand variant assumes the transition between the two states while the right-hand variant requires it to be shown. If the unwinding relation is reflexive and transitive, our definitions of local respect essentially coincide with those given in [Man01]:

$lrf: \ \forall a\ u\ s\ s'\ t.\ dom(a)\ \not\rightsquigarrow\ u \longrightarrow (s,s') \in Step(a) \longrightarrow s' \overset{u}{\sim} s$ and
$lrb: \ \forall a\ u\ s\ t.\ dom(a)\ \not\rightsquigarrow\ u \longrightarrow \exists t'.\ (t,t') \in Step(a)\ \wedge\ t \overset{u}{\sim} t'$

Unwinding can be proved essentially as for the deterministic case.

theorem *simple_Noninterference* : $Step\_consistent\ \wedge\ Local\_respect\_left\ \wedge$
$Local\_respect\_right\ \wedge\ output\_consistent\ \longrightarrow\ Noninterference$

Uniform version Unfortunately, the classical unwinding results concerning weak step consistency cannot be directly transferred to the nondeterministic case. This is due to the two premises of weak step consistency, $s \overset{u}{\sim} t$ and $s \overset{dom(a)}{\sim} t$, which require an inductive argument that in each unwinding step the unwinding relation is preserved simultaneously for more than one domain: $(s,s') \in Step(a) \wedge s \overset{sources(a \frown \alpha, u)}{\approx} t \longrightarrow (\exists t'.\ (t,t') \in Step(a) \wedge s' \overset{sources(\alpha,u)}{\approx} t')$ Without uniformity, (weak) step consistency etc. only guarantee that for each $v \in sources(\alpha, u)$ there is a suitable t', but not necessarily a single t' for all v.

The problem can be circumvented by requiring that the relation $Step(a)$ is functional for all actions a, as done in [Man01]. This means that every transition for a with nondeterministic outcome has to be replaced by a set of transitions with distinguished actions a', a'', ..., where the choice between these actions is nondeterministic. We would like to avoid requiring such a transformation on system descriptions. This is possible, namely by resorting to the stronger notions of *uniform step consistency*, *uniform step respect*, etc. They generalize their counterparts by replacing the unwinding relation for single domains by the variant lifted over arbitrary sets of domains.

def $uni\_Step\_consistent \equiv \forall U\ a\ s\ s'\ t.\ (\exists u \in U.\ dom(a) \rightsquigarrow u)\ \wedge\ s \overset{dom(a)}{\sim} t\ \wedge$
$(s,s') \in Step(a)\ \wedge\ s \overset{U}{\approx} t \longrightarrow (\exists t'.\ (t,t') \in Step(a)\ \wedge\ s' \overset{U}{\approx} t')$

def $uni\_Step\_respect \equiv \forall U\ a\ s\ s'\ t.\ \neg(\exists u \in U.\ dom(a) \rightsquigarrow u)\ \wedge\ U \neq \emptyset\ \wedge$
$(s,s') \in Step(a)\ \wedge\ s \overset{U}{\approx} t \longrightarrow (\exists t'.\ (t,t') \in Step(a)\ \wedge\ s' \overset{U}{\approx} t')$

def $uni\_Local\_respect\_right \equiv \forall U\ a\ s\ t.\ \neg(\exists u \in U.\ dom(a) \rightsquigarrow u)\ \wedge\ U \neq \emptyset\ \wedge$
$s \overset{U}{\approx} t \longrightarrow (\exists t'.\ (t,t') \in Step(a)\ \wedge\ s \overset{U}{\approx} t')$

def $uni\_Local\_respect \equiv Local\_respect\_left\ \wedge\ uni\_Local\_respect\_right$

uni_Local_respect implies *uni_Step_respect*, as well as *uni_Local_respect_right* implies *Local_respect_right*. A uniform version of *Local_respect_left* is not required because one can show

lemma *uni_Local_respect_leftD*: $Local\_respect\_left\ \longrightarrow$
$(s,s') \in Step(a)\ \wedge\ \neg(\exists u \in U.\ dom(a) \rightsquigarrow u)\ \wedge\ s \overset{U}{\approx} t \longrightarrow s' \overset{U}{\approx} t$

With the help of the uniform variants of step consistency and step respect, the remaining notions and lemmas carry over from the deterministic variants, essentially retaining their structure, and we obtain

theorem *Noninterference* : $uni\_Step\_consistent \wedge$
$uni\_Local\_respect \wedge output\_consistent \longrightarrow Noninterference$

In applications of this theorem, in general it will take more effort to prove the uniform variants of step consistency and local respect. Yet in the important special case of a two-level hierarchy of domains $\{H, L\}$, to which every transitive policy may be reduced, the only non-trivial case is $\{L\}$, which happens to be the single standard case that has to be considered also for the non-uniform variants.

If in an application the relation $Step(a)$ is functional from the outset or has been transformed to be functional for every a, i.e. $\forall s\ t\ t'.\ (s,t) \in Step(a) \wedge (s,t') \in Step(a) \longrightarrow t = t'$, the original and the uniform variants of step consistency and step respect etc. coincide and therefore it is sufficient to prove only the simpler original versions.

5 Nonleakage

Classical noninterference is concerned with the visibility of *events*, or to be more precise, with the secrets that events introduce in the system state and that are possibly observed via outputs. While this is the adequate notion for some sorts of applications, there are many others where the concern is not to hide the fact that some secret event has (or has not) occurred but to prevent that initially present secret information leaks out of the domain(s) it is intended to be confined to. The most important of them apparently is language-based information-flow security where type systems give sufficient conditions for confidentiality; see [SM03] for an up-to-date survey. Its semantical core is that variations of high-level *data* (input) should not cause a variation of low-level data (output). In our notation, assuming that $s \overset{L}{\sim} t$ means that the low-level portions of the two states s and t are equal, this can be written as $s \overset{L}{\sim} t \longrightarrow step(a,s) \overset{L}{\sim} step(a,t)$ which is nothing but the common structure of step consistency and step respect.

Language-based security typically handles only the two-level domain hierarchy $\{H, L\}$ and in particular does not address intransitive policies. Inspired by our results on noninterference, we generalize it to arbitrary multi-domain policies. As already argued in §4.2 and depicted by Figure 1, it then becomes important to take into account the domain an action or transition (which in the automata-oriented setting is the analogon to atomic statements in the programming language setting) is allowed to read from. Therefore, the above formula has to be generalized to $(dom(a) \rightsquigarrow u \longrightarrow s \overset{dom(a)}{\sim} t) \wedge s \overset{u}{\sim} t \longrightarrow step(a,s) \overset{u}{\sim} step(a,t)$, which is nothing but the conjunction of weak step consistency and step respect. The conjunction of the two premises can equivalently be written as $s \overset{sources(a\frown[],u)}{\approx} t$, and further generalizing this idea from a single step to an arbitrary sequence of actions, we arrive at the new notion of *nonleakage*.

5.1 Notion

A system is said to be *nonleaking* iff for any pair of states s and t and observing domain u, the two states resulting from running any action sequence α on s and t are indistinguishable for u if s and t have been indistinguishable for all domains that may (directly or indirectly) interfere with u during the run of α:

def $nonleakage \equiv \forall \alpha\ s\ u\ t.\ s \overset{sources(\alpha,u)}{\approx} t \longrightarrow s \triangleleft \alpha \overset{u}{\doteq} t \triangleleft \alpha$

Or put in other words, for any sequence α of actions performed by the system, the outcome of u's observations is independent of variations in all domains except for those in $sources(\alpha, u)$, from which (direct or indirect) information flow in the course of α is allowed. Therefore, the relation in the premise is not simply the unwinding relation $s \overset{u}{\sim} t$ as for language-based security but the conjunction of all $s \overset{v}{\sim} t$ where $v \in sources(\alpha, u)$.
As motivated above, nonleakage can also be seen as the global criterion on allowed vs. actual information flow between domains that is induced by the local criteria of weak step consistency and step respect depicted by Figure 1.

Note that in comparison to the theory of noninterference, purging is not needed, and we do not relate the outcome of runs of different action sequences on the same initial state s_0 but of the same action sequence on two states suitably related at the beginning. Moreover, the unwinding relation is used not only as part of the proof technique, but also for specifying what a domain is allowed to observe. Indeed, it is a rather common approach (cf. [RG99]) to define noninterference in terms of an unwinding relations.

The above notion that initially present secrets do not leak can be simulated using noninterference: by prepending all system runs with a sequence of secret actions that produce the secrets in question and considering all subsequent actions non-secret such that they will not be purged. After the initial sequence, the two intermediate states reached in the purged and in the non-purged run are thus handled by noninterference in the same way as by nonleakage. Still we believe that it is worthwhile to have the simpler, independent notion of nonleakage that expresses the desired information flow property directly.

5.2 Unwinding

For nonleakage and its descendants introduced below, the unwinding techniques are analogous to those for noninterference. From the above introduction of nonleakage, it is easy to see that nonleakage is implied by weak step consistency, step respect, and output consistency, whereas local respect is not required:

theorem *nonleakage* :
$weakly\_step\_consistent \wedge step\_respect \wedge output\_consistent \longrightarrow nonleakage$

Apart from the fact that nonleakage can handle an arbitrary number of domains and arbitrary interference relations between them, the other major difference to language-based security is that, due to the more general automata-theory setting, we cannot give a type system that allows for static checks of confidentiality (provided the type system is sound). Instead, unwinding gives

local semantic conditions for confidentiality, which are harder to verify but on the other hand are (presumably) complete.

5.3 Weak nonleakage

The assignment of domains to actions and the resulting relation $\overset{sources(\alpha,u)}{\approx}$ gives a very fine-grained control over the information flow caused by action sequences α. Often this is not actually needed (or possible), namely if actions are not distinguished or else domains are not assigned to them, such that individual input domains of transitions cannot be specified. Replacing $sources(\alpha, u)$ with supersets not referring to particular actions, we obtain weaker variants of nonleakage.

Using $chain(\alpha, u)$, the set of all domains that may interfere with u via a chain of domains whose length is at most the length of α, we obtain *weak nonleakage*:

def $weak\_nonleakage \equiv \forall \alpha\ s\ u\ t.\ s \overset{chain(\alpha,u)}{\approx} t \longrightarrow s \triangleleft \alpha \overset{u}{\doteq} t \triangleleft \alpha$

For any action sequence α, any domain u may be influenced only by domains linked to u via $\rightsquigarrow$, up to a chain length given by α. One can understand this notion as an inductive, multi-domain generalization of language-based security, interpreting atomic statements as unclassified actions and attributing program variables a domain structure with a possibly intransitive interference relation.

As $sources(\alpha, u)$ is a subset of $chain(\alpha, u)$, *nonleakage* implies *weak_nonleakage*. Due to the use of *chain*, in unwinding proofs it suffices to use very weak versions of step consistency and step respect that allow one to assume that the unwinding relation holds for all domains that may influence the current domain u. Moreover, since the domains of actions do not longer play a role, step consistency and step respect collapse into a single notion:

def $weak\_step\_consistent\_respect \equiv$
$\forall s\ u\ t.\ s \overset{\{w|\ w \rightsquigarrow u\}}{\approx} t \longrightarrow \forall a.\ step(a, s) \overset{u}{\sim} step(a, t)$

This notion can also be seen as the direct generalization of language-based security to transitive multi-domain policies. Here we use it for unwinding, as follows:

theorem $weak\_nonleakage$:
$weak\_step\_consistent\_respect\ \wedge\ output\_consistent\ \longrightarrow\ weak\_nonleakage$

5.4 Transitive weak nonleakage

If actions do not have domains associated with them and additionally the length of a $\rightsquigarrow$ -chain is not of interest, for instance if the interference relation is transitive, we can further replace $chain(\alpha, u)$ with the even simpler $\{w|\ w \rightsquigarrow u\}$:

def $trans\_weak\_nonleakage \equiv \forall s\ u\ t.\ s \overset{\{w|\ w \rightsquigarrow u\}}{\approx} t \longrightarrow \forall \alpha.\ s \triangleleft \alpha \overset{u}{\doteq} t \triangleleft \alpha$

Transitive weak nonleakage expresses that if two states are initially indistinguishable for all domains that may influence u, then u cannot tell them apart after any sequence of actions. We share our experience using it (for the nondeterministic case) in §7. Actually, the application described there had motivated our research on noninterference and its variants presented in this article.

For transitive policies, *weak_nonleakage* implies *trans_weak_nonleakage*, hence we obtain

theorem *trans_weak_nonleakage* :
weak_step_consistent_respect $\wedge$ *output_consistent* $\longrightarrow$ *trans_weak_nonleakage*

Note the strong similarities between the global property of transitive weak nonleakage and the associated local criterion *weak_step_consistent_respect*.

5.5 The nondeterministic case

All results on nonleakage generalize to the nondeterministic case in analogy with noninterference. We give the formal details without further ado in Appendix B.

5.6 Observation and unwinding relations

The notion of nonleakage and its connection with the proof of unwinding properties gives rise to some new insights on observational equivalence (or preorders) and their connection with unwinding relations.

The observational equivalence $s \triangleleft \alpha \stackrel{u}{\simeq} t \triangleleft \alpha$ can be seen as equal outcome of tests on s and t: an attacker belonging to domain u tries to distinguish the two states on their secret contents by attending and/or executing actions α. This is closely related to the passive and active attacks defined by Zdancewic and Myers [ZM01].

Recall that in the deterministic case, the observed outcome of tests is solely $output(u, run(\alpha, s))$ and $output(u, run(\alpha, t))$, respectively. In the nondeterministic case, where we use the observational preorder $s \triangleleft \alpha \stackrel{u}{\precsim} t \triangleleft \alpha$, the outcome is additionally the enabledness of the event sequence α. If *output* respects the unwinding relation $\stackrel{u}{\sim}$, our observational equivalence seems to be equivalent to $S[\approx]$ defined in [ZM01]. Since any output may be encoded by the enabledness of certain "probing" actions, our observational preorder is in fact also very similar to the one implicitly used by Mantel [Man03, Remark 5.2.2], namely, preservation of enabledness for every sequence α of visible events. The only difference is that this preorder is weaker because it restricts α to non-secret events (from u's perspective), i.e. there is no chance for "higher-level" events accidentally helping u to distinguish the two states. One can — and we do — allow for an arbitrary mixture of secret and non-secret events because the unwinding relation is preserved not only by (weak) step consistency dealing with the visible actions, but also by step respect dealing with the invisible ones.

Mantel states that preservation of enabledness (for sequences of visible events) is implied by the unwinding relation, which in the above sense is analogous to output consistency requiring that equality of outputs is implied by unwinding.

As already mentioned, neither Mantel's nor our theory requires that the unwinding relation is reflexive, symmetric, or transitive, while the observational equivalence (or preorder) is weaker and necessarily reflexive and transitive. One could regard the latter as the reflexive and transitive closure of the former, and for many applications, both relations even coincide.

6 Noninfluence

As Mantel and Sabelfeld [MS01] point out, it is important to combine language-based security (of information flow in local computations) and the secrecy of (inter-process communication) events. They achieve this by translating a multi-threaded while-language with variables labeled either *high* or *low* to state-event systems with deterministic transitions and prove the two corresponding notions of security equivalent. We can also deal with both worlds (even for intransitive policies and unrestrictedly nondeterministic systems) by combining noninterference and nonleakage, obtaining a new security notion that we call *noninfluence*:

def $noninfluence \equiv \forall \alpha\ s\ u\ t.\ s \overset{sources(\alpha,u)}{\approx} t \longrightarrow s \triangleleft \alpha \overset{u}{\doteq} t \triangleleft ipurge(\alpha, u)$

Note that here no translation between system descriptions is needed.
Noninfluence is the adequate security notion for state-oriented systems if both

- the occurrence of certain events, which may introduce new secrets, should not be observed, as with classical noninterference, and
- initially present secret data should not be leaked. Allowing for any two indistinguishable initial states s and t rather than the same (and fixed) initial states s_0 gives the extra strength of noninfluence over noninterference.

One could also define a variant of noninfluence resembling the stronger formulation of noninterference allowing arbitrary insertion and deletion of actions.

It is interesting to observe that the proof of the main unwinding theorem for noninterference (cf. §4.2) uses a lemma which already states essentially the above formula (apart from applying the *output* function on the result of run), namely $s \overset{sources(\alpha,u)}{\approx} t \longrightarrow run(\alpha, s) = run(ipurge(u, \alpha), t)$. Rushby uses it too [Rus92, Lemma 5], yet he does not attribute to it an independent value. Its extra strength is needed to get through the main induction in the proof, though later it is specialized to $s = t = s_0$ to deduce noninterference. Thus, noninfluence implies noninterference if $\forall u.\ s_0 \overset{u}{\sim} s_0$ holds.

In the light of the new notion of noninfluence, both noninterference and nonleakage are just special cases where either leakage of initial secrets or the visibility of secret events does not play a role. Nevertheless, it makes sense to keep both of them because they are simpler than noninfluence, and nonleakage does not require local respect.

From the observation in the last paragraph it will be clear that the unwinding theorem for noninfluence requires only those preconditions already used for for noninterference, even though the conclusion is stronger — it simply makes the strength already contained in [Rus92, Lemma 5] available as a security property in its own right. Recall that technically it is even slightly stronger because unwinding may be an arbitrary (even nonreflexive) relation rather than an equivalence.

theorem *noninfluence* :
$weakly\_step\_consistent \wedge local\_respect \wedge output\_consistent \longrightarrow noninfluence$

The generalization to the nondeterministic case is given in Appendix B.

7 Security of the Infineon SLE66

As a concrete example of applying the purely state-based notion of nonleakage, we sketch an extended security analysis of the Infineon SLE66 smart card processor. The main security objective for this device is that part of the chip-internal security mechanisms and secret data like the master encryption key is not leaked to any non-authorized observer or manipulator, called "spy".

In the course of the evaluation of the chip according to ITSEC and Common Criteria, an abstract formal security model, the so-called "LKW model" [LKW00] has been developed and later machine-checked [OL02] using the ISM approach and toolset [ON02]. This model takes a rather naive view of information leakage: a secret value is revealed to the spy if and only if its representation appears on the external interface of the chip. This interpretation does not account for partial or indirect leakage of information that may be used to deduce at least certain properties of the secret values.

We have improved the security analysis using transitive weak nonleakage (for nondeterministic systems). This is the adequate notion for the SLE66 because the observability of actions does not play a role, but the information flow between domains. Only two security domains, *Sec* and *NSec*, are distinguished, so the interference relation is trivially transitive and we have to show *weak_uni_Step_consistent_respect* only for $U = \{NSec\}$, as explained at the end of §4.3. Since encryption is not explicit in the model, we do not have to deal with bogus "interference" of secrets with encrypted outputs, though this would be possible by stating an intransitive information flow via the encryption unit.

For the unwinding relation we choose an equivalence that states equality for all parts of the chip state that is allowed to be observed from outside, namely the phases of the chip life-cycle, the availability of chip functionality (see below), and all non-secret data values. Once the right relation has been chosen, the proof is rather schematic and in this case not very difficult. The only complication is that the property holds only for reachable states for which suitable invariants hold, but we can re-use the invariants that we had already shown during the previous analysis.

Conducting the proof, we obtained the following results.

- It is crucial that chip functions do not internally leak secret data or give them away via the chip interface. Since chip functionality is heavily underspecified, this auxiliary property cannot be proved but has to be provided as an axiom.
- The possibility that at most one (encrypted) secret data value or the encryption key itself gets leaked is explicitly allowed by the chip designers because there is no practical means to prevent this. This leakage is actually found during the proof. Yet it does not do harm because immediately thereafter the chip gets completely locked such that no more values can be obtained and thus not both some encrypted data and the key are known to the spy.
- The chip cannot avoid leaking the availability even of those functions that are considered to be secret (whereas their actual code is not leaked).
- Apart from the exceptions just given, no secret information, not even any partial information about secret data, can be leaked.

8 Conclusion

We have refined Rushby's work on noninterference by rectifying its minor shortcomings concerning transitive vs. intransitive policies and the requirements on the unwinding relation. This opens a wider range of applications and enables stronger results like in Appendix A. We have significantly extended Rushby's theory to handle nondeterministic systems, introducing uniform step consistency.

We have introduced notions of information flow security that have not been considered before but naturally arise from re-interpreting Rushby's unwinding lemmas, and gained new insights in the nature of unwinding and observability. Nonleakage has a high application potential because it generalizes language-based security to arbitrary policies and state-transforming systems. Noninfluence combines this with classical event-oriented noninterference.

It should be worthwhile conducting further theoretical investigations, e.g. on completeness of the unwinding conditions and the comparison with related notions like (robust) declassification and bisimulation.

Our theory has been implemented in the interactive theorem proving system Isabelle/HOL. As witnessed by the example of the SLE66, it is ready to be applied in the formal security analysis of state-oriented systems.

Acknowledgments We thank John Rushby, Heiko Mantel, Peter Ryan, Thomas Bleher, Volkmar Lotz, Stephan Merz, Tamara Rezk, and several anonymous referees for their encouragement and feedback on earlier versions of this paper.

References

Fol87. Simon N. Foley. A universal theory of information flow. In *IEEE Symposium on Security and Privacy*, pages 116–122, 1987.

GM82. J. A. Goguen and J. Meseguer. Security policies and security models. In *Symposium on Security and Privacy*. IEEE Computer Society Press, 1982.

GM84. J. A. Goguen and J. Meseguer. Unwinding the inference control. In *Symposium on Security and Privacy*. IEEE Computer Society Press, 1984.

HY86. J. Haigh and W. Young. Extending the non-interference version of MLS for SAT. In *Proc. of the Symposium on Security and Privacy*, pages 232–239. IEEE Computer Society Press, 1986.

LKW00. Volkmar Lotz, Volker Kessler, and Georg Walter. A Formal Security Model for Microprocessor Hardware. In *IEEE Transactions on Software Engineering*, volume 26, pages 702–712, August 2000.

Man00. Heiko Mantel. Unwinding possibilistic security properties. In *Proc. of ESORICS*, volume 1895 of *LNCS*, pages 238 – 254. Springer, 2000.

Man01. Heiko Mantel. Information Flow and Applications — Bridging a Gap. In *Proc. of FME 2001: Formal Methods for Increasig Software Productivity*, volume 2021 of *LNCS*, pages 153 – 172. Springer, 2001.

Man03. H. Mantel. *A Uniform Framework for the Formal Specification and Verification of Information Flow Security*. PhD thesis, Univ. d. Saarlandes, 2003.

McC90. Darly McCullough. A hookup theorem for multilevel security. In *IEEE Transactions on Software Engineering*, pages 563–568, 1990.

McL94. John McLean. A general theory of composition for trace sets closed under selective interleaving functions. In *IEEE Symposium on Security and Privacy*, pages 79–93, 1994.
MS01. Heiko Mantel and Andrei Sabelfeld. A Generic Approach to the Security of Multi-threaded Programs. In *Proc. of 14th CSFW*, pages 126–142, Cape Breton, Nova Scotia, Canada, 2001. IEEE Computer Society.
NPW02. Tobias Nipkow, Lawrence Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002. See also http://isabelle.in.tum.de/docs.html.
Ohe04. David von Oheimb. *Isabelle theory sources: Noninfluence = Noninterference + Nonleakage*, 2004. http://ddvo.net/HOL/NI/.
OL02. David von Oheimb and Volkmar Lotz. Formal Security Analysis with Interacting State Machines. In Dieter Gollmann, Günter Karjoth, and Michael Waidner, editors, *Proc. of the 7th European Symposium on Research in Computer Security (ESORICS)*, volume 2502 of *LNCS*, pages 212–228. Springer, 2002. http://ddvo.net/papers/FSA_ISM.html. An extended version has appeared as NICTA technical report 0401005T-1.
ON02. David von Oheimb and Sebastian Nanz. *ISM Homepage: Documentation, sources and distribution*, 2002. http://ddvo.net/ISM/.
Pin95. Sylvan Pinsky. Absorbing covers and intransitive non-interference. In *IEEE Symposium on Security and Privacy*, pages 102–113, 1995.
RG99. A.W. Roscoe and M.H. Goldsmith. What is intransitive noninterference? In *12th Computer Security Foundations Workshop*, pages 228–238. IEEE Computer Society Press, 1999.
Rus92. John Rushby. Noninterference, Transitivity, and Channel-Control Security Policies. Technical Report CS-92-02, SRI International, 1992.
Rya90. Peter Ryan. A CSP formulation of non-interference and unwinding. In *Proc. of IEEE CSFW-3*. Cipher, 1990.
Rya01. Peter Y. A. Ryan. Mathematical models of computer security. In Riccardo Focardi and Roberto Gorrieri, editors, *Foundations of Security Analysis and Design: Tutorial Lectures*, volume 2171 of *LNCS*. Springer, 2001.
SM03. A. Sabelfeld and A. C. Myers. Language-based information-flow security. *IEEE J. on Selected Areas in Communications*, 21(1):5–19, January 2003.
Sut86. D. Sutherland. A model of information. In *Proc. National Computer Security Conference*, pages 175–183, 1986.
ZL97. Aris Zakinthinos and E. Stewart Lee. A general theory of security properties. In *Computer Society Symposium on Research in Security and Privacy*, 1997.
ZM01. Steve Zdancewic and Andrew C. Myers. Robust Declassification. In *14th IEEE Computer Security Foundations Workshop (CSFW)*, 2001.

A Access control interpretation

As a (rather abstract) application of our strengthened noninterference theory, we give an improvement of Rushby's access control interpretation [Rus92, §2.1].

Employing our unwinding theorem which works for both transitive and intransitive interference relations, we only have to show weak step consistency. Doing so, we have stronger preconditions and thus can dispense with the monotonicity condition $u \rightsquigarrow v \longrightarrow observe(u) \subseteq observe(v)$, which, according to Rushby, had forced the transitive completion of the policy. In effect, we generalize his access control interpretation to the general (possibly intransitive) case.

In order to express access control, the system model is extended by adding structure to the state, which now becomes a function that maps names to values:

$$\mathit{contents} : \mathit{state} \times \mathit{name} \rightarrow \mathit{value}$$

Policies are refined accordingly, associating each domain with a set of names of objects that it is allowed to read and write, respectively:

$$\begin{aligned} \mathit{observe} &: \mathit{domain} \rightarrow \wp(\mathit{name}) \\ \mathit{alter} &: \mathit{domain} \rightarrow \wp(\mathit{name}) \end{aligned}$$

Following Rushby, for the application of the unwinding theorem, we use the canonical unwinding relation induced by *contents* and *observe*,

$$\textbf{def}\ s \overset{u}{\sim} t \equiv \forall n \in \mathit{observe}(u).\ \mathit{contents}(s,n) = \mathit{contents}(t,n)$$

which happens to be an equivalence, though we do not take advantage of this.

Rushby introduces three *reference monitor assumptions*. The first of them is the already introduced output consistency:

$$\textbf{def}\ \mathit{RMA}_1 \equiv \mathit{output\_consistent}$$

As a matter of fact, if the output function yields all values observable for the given domain, i.e. $\mathit{output}(u,s) \equiv \{(n, \mathit{contents}(s,n)) \mid n \in \mathit{observe}\ u\}$, output consistency is fulfilled immediately.

Rushby's second reference monitor assumption states that if an action a changes the value at some location n, the new value depends only on $\mathit{dom}(a)$. Due to our observation that weak step consistency is sufficient in any case, we can use a weaker variant of it, offering the extra premises $\mathit{dom}(a) \rightsquigarrow u$, $s \overset{u}{\sim} t$, and $n \in \mathit{observe}\ u$ which allow information flow into n from any domain u that is allowed to observe n and that may be influenced by the input domain of a. In other words, if action a changes the contents of variable n observable by domain u and if $\mathit{dom}(a)$ may influence u, the new value depends only on values observable by $\mathit{dom}(a)$ and u:

$$\begin{aligned} \textbf{def}\ \mathit{RMA}_2 \equiv \forall a\ u\ s\ t\ n.\ s \overset{\mathit{dom}(a)}{\sim} t\ \wedge\ \mathit{dom}(a) \rightsquigarrow u\ \wedge\ s \overset{u}{\sim} t\ \wedge\ n \in \mathit{observe}\ u\ \wedge \\ (\mathit{contents}(\mathit{step}(a,s),n) \neq \mathit{contents}(s,n)\ \vee \\ \mathit{contents}(\mathit{step}(a,t),n) \neq \mathit{contents}(t,n)) \longrightarrow \\ \mathit{contents}(\mathit{step}(a,s),n) = \mathit{contents}(\mathit{step}(a,t),n) \end{aligned}$$

Interestingly, weak step consistency can now be derived from the second reference monitor assumption alone: $\mathit{RMA}_2 \longrightarrow \mathit{weakly\_step\_consistent}$

The third assumption, stating that any changes must be granted by *alter*,

$$\textbf{def}\ \mathit{RMA}_3 \equiv \forall a\ s\ n.\ \mathit{contents}(\mathit{step}(a,s),n) \neq \mathit{contents}(s,n) \longrightarrow n \in \mathit{alter}(\mathit{dom}(a))$$

in conjunction with the remaining condition of Rushby's Theorem 2,

$$\textbf{def}\ \mathit{AC\_policy\_consistent} \equiv \forall u\ v.\ \mathit{alter}(u) \cap \mathit{observe}(v) \neq \emptyset \longrightarrow u \rightsquigarrow v$$

implies local respect: $\mathit{RMA}_3 \wedge \mathit{AC\_policy\_consistent} \longrightarrow \mathit{local\_respect}$

Hence, we can prove enforcement of access control

$$\begin{aligned} &\textbf{theorem}\ \mathit{access\_control\_secure} : \\ &\mathit{RMA}_1 \wedge \mathit{RMA}_2 \wedge \mathit{RMA}_3 \wedge \mathit{AC\_policy\_consistent} \longrightarrow \mathit{noninterference} \end{aligned}$$

under weaker assumptions than Rushby does: we do *not* require that

- *observe* and *alter* induce a transitive information flow policy,
- granted information flow induces a hierarchy of observable locations, nor
- information flow into a location n from any domain that is allowed to observe n and that may be influenced by the input domain of the current action does not occur.

B Nondeterministic nonleakage and noninfluence

Nonleakage

def $Nonleakage \equiv \forall \alpha\ s\ u\ t.\ s \overset{sources(\alpha,u)}{\approx} t \longrightarrow s \triangleleft \alpha \overset{u}{\leadsto} t \triangleleft \alpha$

theorem *Nonleakage* :
$uni\_Step\_consistent \wedge uni\_Step\_respect \wedge output\_consistent \longrightarrow Nonleakage$

Weak nonleakage

def $weak\_Nonleakage \equiv \forall \alpha\ s\ u\ t.\ s \overset{chain(\alpha,u)}{\approx} t \longrightarrow s \triangleleft \alpha \overset{u}{\leadsto} t \triangleleft \alpha$

As above, $Nonleakage \longrightarrow weak\_Nonleakage$.

def $weak\_uni\_Step\_consistent\_respect \equiv \forall U\ a\ s\ s'\ t.\ U \neq \emptyset \wedge$
$(s, s') \in Step(a) \wedge (\forall u \in U.\ s \overset{\{w|\ w \rightsquigarrow u\}}{\approx} t) \longrightarrow (\exists t'.\ (t, t') \in Step(a) \wedge s' \overset{U}{\approx} t')$

theorem *weak_Nonleakage* :
$weak\_uni\_Step\_consistent\_respect \wedge output\_consistent \longrightarrow weak\_Nonleakage$

Transitive weak nonleakage

def $trans\_weak\_Nonleakage \equiv \forall s\ u\ t.\ s \overset{\{w|\ w \rightsquigarrow u\}}{\approx} t \longrightarrow \forall \alpha.\ s \triangleleft \alpha \overset{u}{\leadsto} t \triangleleft \alpha$

For transitive policies, $weak\_Nonleakage \longrightarrow trans\_weak\_Nonleakage$, hence

theorem *trans_weak_Nonleakage* :
$weak\_uni\_Step\_consistent\_respect \wedge output\_consistent \longrightarrow$
$trans\_weak\_Nonleakage$

Noninfluence

Paralleling the deterministic case as far as possible, we define

def *Noninfluence* $\equiv$
$\forall \alpha\ \beta\ s\ u\ t.\ s \overset{sources(\alpha,u)}{\approx} t \wedge ipurge(u, \alpha) = ipurge(u, \beta) \longrightarrow s \triangleleft \alpha \overset{u}{\leadsto} t \triangleleft \beta$

and can prove

theorem *Noninfluence* :
$uni\_Step\_consistent \wedge uni\_Local\_respect \wedge output\_consistent \longrightarrow Noninfluence$