

Quotient Automata for Non-Interference

Florian Lasinger, Alexander Knapp
Ludwig-Maximilians-Universität München
{lasinger, knapp}@pst.ifi.lmu.de

Abstract

We revisit Rushby's non-interference definitions and examine the different cases for a possibility to specify a secure automaton in terms of several smaller automata. These will be easier to specify and also easier to verify. The dimensions that we are interested in are deterministic vs. nondeterministic automata under transitive vs. intransitive security policies. While it is in most cases possible to define a secure automaton through quotient automata, we will see that the more complex the security property, the more remote the correlation between the automata's properties and the desired security property will be. In the end we get a bird's eye view on the different cases.

1 Introduction

Confidentiality of personal or classified data has been an ever growing concern in computing systems. As the pervasiveness of computers rises in private and business life, the problem of sensitive data is not going to disappear. The most prominent approaches include access control systems and cryptography, both of which allow to keep data secure in certain ways but lack the desired fine-grained control over where and in which ways data *flows*. Another approach therefore lies in the field of secure information flow control. The basic principle here is to track the confidentiality of data throughout computing systems (either dynamically or statically) and deny operations that would expose information to unauthorized parties. The central property is *noninterference* and requires secret actions to not be observable by anyone who is not explicitly entitled to notice them. In the past, a lot of effort went into defining variations of this basic property that would make it more suitable for one case or another, but little has been done to assist in actually specifying a noninterferent system. Therefore we wanted to examine Rushby's definition of noninterference [1] for the ability to define one large secure system through the use of several smaller systems, each representing one party's view of the system. The goal would be a system that is secure by construction if certain simpler properties hold for a set of smaller and easier to understand systems.

Sections 2 to 5 examine the different cases of Rushby's noninterference. Section 6 gives a bird's eye view on what we've learned from these sections. Section 7 sketches an algorithm we implemented to find quotient automata for a given secure system. At last, Section 8 concludes with future work.

2 Deterministic automaton + transitive policy

The first and simplest case we will examine is that of deterministic automata under a transitive security policy. In the following we will use the symbols NI , UP and QA to denote non-interference properties, unwinding properties or a particular class of quotient automata, respectively. Those will be annotated with e.g. dt for *deterministic transitive* and so on. So NI_{dt} denotes the non-interference property for the deterministic, transitive case.

2.1 NI_{dt}

Our definitions in this section are very close to Rushby's original ones. Given a set \mathcal{D} of security domains (observers) and a reflexive, transitive relation $\rightsquigarrow \subseteq \mathcal{D} \times \mathcal{D}$ over these. This interference relation specifies for every domain which others it is allowed to influence, or from which to which domains information is allowed to flow, respectively.

We then define an automaton to be a tuple $M = (\mathcal{A}, \mathcal{O}, \mathcal{S}, step, out)$, with

- \mathcal{A} a set of actions,
- \mathcal{O} a set of outputs,
- \mathcal{S} a set of states,
- $step : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ and
- $out : \mathcal{S} \times \mathcal{D} \rightarrow \mathcal{O}$.

To this belongs a function $dom : \mathcal{A} \rightarrow \mathcal{D}$ assigning actions to domains, and an initial state $s_0 \in \mathcal{S}$.

We proceed with a few small definitions that will help us state the first non-interference property. The function $run : \mathcal{S} \times \mathcal{A}^* \rightarrow \mathcal{S}$ just executes a series of actions from a given state and returns the state that was reached.

$$\begin{aligned} run(s, \lambda) &= s \\ run(s, a \circ \alpha) &= run(step(s, a), \alpha) \end{aligned}$$

Function $do : \mathcal{A}^* \rightarrow \mathcal{S}$ will execute a series of actions starting from the initial state.

$$do(\alpha) = run(s_0, \alpha)$$

Finally, $test : \mathcal{A}^* \times \mathcal{D} \rightarrow \mathcal{O}$ will run a series of actions and return the final state's output for any given domain u .

$$test(\alpha, u) = out(do(\alpha), u)$$

Further, we define a function $purge : \mathcal{A}^* \times \mathcal{D} \rightarrow \mathcal{A}^*$ that removes from an action sequence all actions whose domain does not directly influence a given domain u .

$$\begin{aligned} purge(\lambda, u) &= \lambda \\ purge(a \circ \alpha, u) &= \begin{cases} a \circ purge(\alpha, u) & \text{if } dom(a) \rightsquigarrow u \\ purge(\alpha, u) & \text{otherwise} \end{cases} \end{aligned}$$

The non-interference property for the transitive, deterministic case is then

$$NI_{dt} : test(\alpha, u) = test(purge(\alpha, u), u)$$

It states that for any domain u the observable output of any action sequence α needs to behave exactly as if all the actions that are not allowed to influence u had never been executed. That way, domains whose information is not allowed to flow to u will be completely invisible to u . We call an automaton that satisfies NI_{dt} non-interferent w.r.t. \rightsquigarrow .

2.2 UP_{sc}

This notion of security can be ensured by local, so called unwinding properties, as shown by Rushby. He shows that if relations $\sim_u \subseteq \mathcal{S} \times \mathcal{S}, u \in \mathcal{D}$ exist such that the three following properties are satisfied, then the automaton will be non-interferent w.r.t. \rightsquigarrow .

$$\begin{aligned} oc : s \sim_{dom(a)} t &\implies out(s, u) = out(t, u) && \text{output consistency} \\ sc : s \sim_u t &\implies step(s, a) \sim_u step(t, a) && \text{step consistency} \\ lr : dom(a) \not\rightsquigarrow u &\implies s \sim_u step(s, a) && \text{local respect} \end{aligned}$$

Rushby also showed that these unwinding properties are complete for NI_{dt} (that for every non-interferent automaton there exist unwinding relations satisfying these unwinding properties).

2.3 UP_{sc} and equivalence relations \sim_u

Although the relations \sim_u allowed by oc , lr and sc are usually arbitrary relations, we want to show that every such relation can be extended to an equivalence relation while preserving the unwinding properties. For this let \approx_u be the smallest equivalence relation containing \sim_u . That means the following hold:

- (1) $s \sim_u t \implies s \approx_u t$
- (2) $s \approx_u s$
- (3) $t \approx_u s \implies s \approx_u t$
- (4) $s \approx_u r \wedge r \approx_u t \implies s \approx_u t$

We can add the edges in $\approx_u - \sim_u$ to \sim_u one by one to generate a series $\sim_u = K_0^u \subset K_1^u \subset \dots \subset K_{n-1}^u \subset K_n^u = \approx_u$ of subsets of \approx_u . In each step $K_i^u \rightarrow K_{i+1}^u$ we choose an $s \approx_u t$ that is not in K_i^u but that is reflexive or whose premises according to 3 or 4 are in K_i^u , and let $K_{i+1}^u = K_i^u \cup \{(s, t)\}$. Such an edge always exists due to the minimality of \approx_u . Assuming that the unwinding properties hold for those edges of \approx_u that are already in K_i^u , we deduce the same for K_{i+1}^u . The base case is $K_0^u = \sim_u$.

For any edge $s \approx_u t$, the reason for its existence is one of 1,2,3 or 4. In case 1 we already know that the unwinding properties hold. Case 2 trivially satisfies oc and is self-fulfilling for sc . Also does it not interfere with the other cases. All the edges induced by local respect are already contained in \sim_u . Therefore what remains to be shown are oc and sc for cases 3 and 4.

For those edges $s \approx_u t$ introduced by 3 or 4 we take the biggest i such that $s K_{i+1}^u t$ but not $s K_i^u t$. If $s \approx_u t$ was added due to 3, we know there is a $t K_i^u s$ responsible for it. Thus

$$\begin{aligned} oc: s \approx_u t &\implies \exists i. t K_i^u s \implies_{oc} out(t, u) = out(s, u) \\ sc: s \approx_u t &\implies \exists i. t K_i^u s \implies_{sc} step(t, a) \approx_u step(s, a) \\ &\implies_3 step(s, a) \approx_u step(t, a) \end{aligned}$$

If $s \approx_u t$ was added due to 4, we have $s K_i^u r$ and $r K_i^u t$ and thus

$$\begin{aligned} oc: s \approx_u t &\implies \exists i. s K_i^u r \wedge r K_i^u t \implies_{oc} out(s) = out(r) = out(t) \\ sc: s \approx_u t &\implies \exists i. s K_i^u r \wedge r K_i^u t \\ &\implies_{sc} step(s, a) \approx_u step(r, a) \wedge step(r, a) \approx_u step(t, a) \\ &\implies_4 step(s, a) \approx_u step(t, a) \end{aligned}$$

So assuming that the edges in K_i^u satisfy the unwinding properties we have shown that those in K_{i+1}^u do. Therefore, if $\sim_u = K_u^0$ satisfies the unwinding properties then it can be extended to the equivalence relation \approx_u still satisfying them.

For the rest of this section we assume \sim_u to be equivalence relations without loss of generality.

2.4 QA_{sc}

In the next step we show that an automaton's characterization by the unwinding properties is equal to one using quotient automata.

Lemma 1. *Given a family of automata $(M_u)_{u \in \mathcal{D}} = (\mathcal{A}, \mathcal{O}, \mathcal{S}_u, out_u, step_u)$, $out_u : \mathcal{S}_u \times \mathcal{D} \rightarrow \mathcal{O}$, $step_u : \mathcal{S}_u \times \mathcal{A} \rightarrow \mathcal{S}_u$, such that*

$$A: dom(a) \not\rightarrow u \implies step_u(s_u, a) = s_u.$$

Then there are unwinding relations \sim_u for every automaton

$M = (\mathcal{A}, \mathcal{O}, \mathcal{S}, out, step)$ for which functions $f_u : \mathcal{S} \rightarrow \mathcal{S}_u$ can be defined, such that

$$1: out_u(f_u(s)) = out(s, u)$$

$$2a: step_u(f_u(s), a) = f_u(step(s, a)).$$

Proof. Relations \sim_u can be extracted from the M_u such that the unwinding properties hold. Let therefore $D: s \sim_u t \Leftrightarrow f_u(s) = f_u(t)$.

$$oc: s \sim_u t \implies out(s, u) = out(t, u) :$$

From $s \sim_u t$ follows $f_u(s) = f_u(t)$ by D and from that trivially $out_u(f_u(s)) = out_u(f_u(t))$. With 1 we have $out(s, u) = out(t, u)$.

$$lr: dom(a) \not\rightarrow u \implies s \sim_u step(s, a) :$$

Using A while choosing $s \in \mathcal{S}$ such that $s_u = f_u(s)$ we get $step_u(f_u(s), a) = f_u(s)$. Application of 2a gives $f_u(step(s, a)) = f_u(s)$ from where D gives the desired result $s \sim_u step(s, a)$.

$$sc: s \sim_u t \implies step(s, a) \sim_u step(t, a) :$$

This time we start with D to get $f_u(s) = f_u(t)$. Insertion into $step_u$ gives $step_u(f_u(s), a) = step_u(f_u(t), a)$. An application of 2a leads to $f_u(step(s, a)) = f_u(step(t, a))$, D gives $step(s, a) \sim_u step(t, a)$.

We now have an automaton M that is oc , lr and sc and is therefore non-interferent w.r.t. \rightsquigarrow . □

For the other direction we assume an automaton M and show that quotient automata $(M_u)_{u \in \mathcal{D}}$ can be derived if M is oc , sc and lr .

Lemma 2. *Given $M = (\mathcal{A}, \mathcal{O}, \mathcal{S}, step, out)$ and \sim_u we can define $M_u = (\mathcal{A}, \mathcal{O}, \mathcal{S}_u, step_u, out_u)$ such that A , 1 and 2a hold.*

Proof. Let \approx_u be the smallest equivalence relation containing \sim_u . We know that it will also satisfy UP_{sc} . Now define f_u such that $d: f_u(s) = f_u(t) \Leftrightarrow s \approx_u t$. Define $step_u$ and out_u by 1 and 2a. We then need to show that this is a legal definition and that A holds.

$$2a : step_u(f_u(s), a) = f_u(step(s, a)) :$$

In this case we show that $step_u$ maps to the same state both $f_u(s) = f_u(t)$. This equals $s \approx_u t$ and using sc leads to $step(s, a) \approx_u step(t, a)$. D gets us back to $f_u(step(s, a)) = f_u(step(t, a))$. The definition of $step_u$ completes this step.

$$1 : out_u(f_u(s)) = out(s, u) :$$

Again assume $f_u(s) = f_u(t)$, use D , oc and the definition of out_u .

$$A : dom(a) \not\rightsquigarrow u \implies step_u(s_u, a) = s_u :$$

lr gives $s \approx_u step(s, a)$, and with D follows $f_u(s) = f_u(step(s, a))$. Choosing $s_u \in \mathcal{S}_u$ such that $s_u = f_u(s)$ and using the definition of $step_u$, we arrive at $s_u = step_u(s_u, a)$. □

Remark: The fact that $step_u$ is deterministic hints to the circumstance that in a step consistent automaton there is no real information *flow*. Otherwise there would be non-deterministic transitions reflecting that the observable outcome depended on information that the observing domain did not have prior to the transition. The transition relation being deterministic implies that every domain has all its information right from the start. Generally speaking, transitivity turns \rightsquigarrow into a partial order where each domain can always see everything happening below it but never anything above (or beside) it.

3 Nondeterministic automaton + transitive policy

3.1 NI_{nt}

In this section we lift Rushby's step-consistent automata to non-deterministic step-consistent automata. Their definition $M = (\mathcal{A}, \mathcal{O}, \mathcal{S}, nstep, out)$ is similar, only this time the transition relation $nstep : \mathcal{S} \times \mathcal{A} \rightarrow 2^{\mathcal{S}}$ is non-deterministic.

Therefore we introduce counterpart helper functions to those defined for the deterministic case. All of them now accept sets of states as input. Generally, all functions and properties working on sets of states are named using upper-case letters by convention.

$$\begin{aligned}
Step &: 2^{\mathcal{S}} \times \mathcal{A} \rightarrow 2^{\mathcal{S}} \\
Step(S, a) &= \bigcup_{s \in S} nstep(s, a) \\
Out &: 2^{\mathcal{S}} \times \mathcal{D} \rightarrow 2^{\mathcal{O}} \\
Out(S, u) &= \{out(s, u) \mid s \in S\} \\
Run &: 2^{\mathcal{S}} \times \mathcal{A}^* \rightarrow 2^{\mathcal{S}} \\
Run(S, \lambda) &= S \\
Run(S, a \circ \alpha) &= Run(Step(S, a), \alpha) \\
Do &: \mathcal{A}^* \rightarrow 2^{\mathcal{S}} \\
Do(\alpha) &= Run(\{s_0\}, \alpha) \\
Test &: \mathcal{A}^* \times \mathcal{D} \rightarrow 2^{\mathcal{O}} \\
Test(\alpha, u) &= Out(Do(\alpha), u)
\end{aligned}$$

The non-interference property now uses *Test* and expresses that every domain may observe the same set of outputs whether actions invisible to it have taken place or not.

$$NI_{nt} : Test(\alpha, u) = Test(purge(\alpha, u), u)$$

3.2 $UP_{nSC}^{\approx} \iff NI_{nt}$

As *nstep* now produces several states in one transition, we need a new set of unwinding conditions and also a new unwinding relation, this time defined on sets of states. Let $\approx_u \subseteq 2^{\mathcal{S}} \times 2^{\mathcal{S}}$ be an equivalence relation in the following. Then our adapted unwinding conditions are

$$\begin{aligned}
OC^{\approx} & \quad S \approx_u T \implies Out(S, a) = Out(T, a), \\
nSC^{\approx} & \quad S \approx_u T \implies Step(S, a) \approx_u Step(T, a) \text{ and} \\
nLR^{\approx} & \quad dom(a) \not\prec u \implies S \approx_u Step(S, a).
\end{aligned}$$

We first adapt Rushby's proof for the deterministic case to make sure that these properties imply NI_{nt} .

Lemma 3. *Given an automaton M that is OC^{\approx} and such that $Do(\alpha) \approx_u Do(purge(\alpha, u))$. Then M is non-interferent.*

Proof. We proceed as follows:

$$\begin{array}{ccc}
Do(\alpha) & \approx_{dom(a)} & Do(purge(\alpha, u)) \\
& \Downarrow_{OC} & \\
Out(Do(\alpha), u) & = & Out(Do(purge(\alpha, u)), u) \\
& \Updownarrow_{Test} & \\
Test(\alpha, u) & = & Test(purge(\alpha, u), u),
\end{array}$$

which is the definition of non-interference. \square

Theorem 4. *Given a policy \rightsquigarrow , an automaton M that is OC^{\approx} , nSC^{\approx} and nLR^{\approx} . Then M is secure w.r.t. \rightsquigarrow .*

Proof. We use proof by induction on the length of α to establish

$$S \simeq_u T \implies \text{Run}(S, \alpha) \simeq_u \text{Run}(T, \text{purge}(\alpha, u)). \quad (1)$$

The base case is $\alpha = \lambda$ and is elementary. For the inductive step, we assume the inductive hypothesis for α and show it for $a \circ \alpha$. There are two cases to consider.

Case 1: $\text{dom}(a) \rightsquigarrow u$

Since $S \simeq_u T$ and the automaton is nSC^\simeq , it follows that

$$\text{Step}(S, a) \simeq_u \text{Step}(T, a)$$

and the inductive hypothesis then gives

$$\begin{array}{ccc} \text{Run}(\text{Step}(S, a), \alpha) & \simeq_u & \text{Run}(\text{Step}(T, a), \text{purge}(\alpha, u)) \\ \parallel_{\text{Run}} & & \parallel_{\text{Run}} \\ \text{Run}(S, a \circ \alpha) & & \text{Run}(T, a \circ \text{purge}(\alpha, u)) \\ & & \parallel_{\text{purge}} \\ & & \text{Run}(T, \text{purge}(a \circ \alpha, u)) \end{array}$$

Case 2: $\text{dom}(a) \not\rightsquigarrow u$

Since \simeq_u is an equivalence relation, $S \simeq_u T$, $\text{dom}(a) \not\rightsquigarrow u$ and M is nLR^\simeq we have

$$S \simeq_u \text{Step}(S, a) \simeq_u T.$$

Using the inductive hypothesis we get

$$\begin{array}{ccc} \text{Run}(\text{Step}(S, a), \alpha) & \simeq_u & \text{Run}(T, \text{purge}(\alpha, u)) \\ \parallel_{\text{Run}} & & \parallel_{\text{purge}} \\ \text{Run}(S, a \circ \alpha) & & \text{Run}(T, \text{purge}(a \circ \alpha, u)) \end{array}$$

In order to complete the proof, we take $S = T = \{s_0\}$ in (1) to obtain

$$\text{Do}(\alpha) \simeq_u \text{Do}(\text{purge}(\alpha, u))$$

and then, since M is OC^\simeq , invoke Lemma 1 to complete the proof. \square

For the other direction we start with an automaton M satisfying NI_{nt} . We then show that OC^\simeq , nSC^\simeq and nLR^\simeq hold for all reachable sets of states if we define \simeq_u by

$$S \simeq_u T :\Leftrightarrow \forall \alpha \in A^*. \text{Test}(\alpha, u) = \text{Test}(\text{purge}(\alpha, u), u).$$

The proof is analogous to Rushby's proof for the deterministic case.

3.3 UP_{nSC}

The quotient automata up to this point were defined on equivalent states, i.e. $\mathcal{S}_u = [\mathcal{S}]_u$. UP_{nSC}^\simeq , however, is talking about equivalent sets of states. So while technically it should be possible to define automata over equivalence classes of sets of states, i.e. $\mathcal{S}_u = [2^{\mathcal{S}}]_u$, this seems neither practical nor intuitive due to the sheer size of these automata. Remember that \mathcal{S} is usually already exponential

in some number of variables. Instead we prefer a stronger set of properties that are based on equivalent states.

David von Oheimb [3] introduces one such set of properties. However, his properties are specifically engineered to allow for unwinding relations that are not equivalence relations, which don't translate to quotient automata. Therefore we stick with the simpler direct generalization of Rushby's unwinding properties.

As unwinding relation, instead of \succsim_u we are now using \sim_u again. On this we define an abbreviation $\simeq_u \subseteq 2^S \times 2^S$ relating sets of states as follows

$$S \simeq_u T :\Leftrightarrow \forall s \in S. \exists t \in T. s \sim_u t \wedge \forall t \in T. \exists s \in S. s \sim_u t.$$

In order to be compatible with the nondeterministic *nstep* relation, *sc* and *lr* are adopted to

$$\begin{aligned} nsc: & s \sim_u t \implies nstep(s, a) \simeq_u nstep(t, a) \text{ and} \\ nlr: & dom(a) \not\sim_u u \implies \{s\} \simeq_u nstep(s, a). \end{aligned}$$

However, a formulation using the same relation on both sides would be more convenient, as in

$$\begin{aligned} nSC: & S \simeq_u T \implies Step(S, a) \simeq_u Step(T, a) \text{ and} \\ nLR: & dom(a) \not\sim_u u \implies S \simeq_u Step(S, a). \end{aligned}$$

It can be easily shown that both formulations are equivalent, and we'll be using *nSC* and *nLR* in the following. The same holds for the replacement of *oc*,

$$OC: S \simeq_u T \implies Out(S, u) = Out(T, u).$$

3.4 $UP_{nSC} \implies NI_{nt}$

UP_{nSC}^{\succsim} follows easily from UP_{nsc} by taking $S \succsim_u T :\Leftrightarrow S \simeq_u T$. Therefore, we already have $UP_{nSC} \implies NI_{nt}$.

That the other direction doesn't hold can be seen from the simple example in Figure 1. (We assume for this and the following examples a total transition relation. Please mentally insert self-looping transitions where ever an action has no outgoing edge.) Let $s_0 = s$, $dom(b) = u$ and $dom(a) \not\sim_u u$. Then $s \simeq_u \{s, t\}$ follows from *nLR*, which contains $s \simeq_u t$. *nSC* then gives $\{s', t'\} \simeq_u t'$ and thus also $s' \simeq_u t'$. Now for $out(s', u) \neq out(t', u)$, *OC* does not hold, however for no sequence of actions can an occurrence of *a* be deduced from the final output.

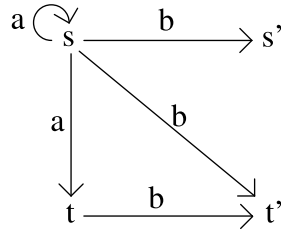


Figure 1: $NI_{nt} \not\Rightarrow UP_{nSC}$

3.5 UP_{nsc} and equivalence relations \sim_u

The proof that every relation \sim_u satisfying *oc*, *nsc* and *nlr* can be extended to an equivalence relation is completely analogous to the deterministic case in 2.3. We will therefore assume \sim_u to be equivalence relations throughout the rest of this section.

3.6 QA_{nsc}

We adapt the result from section 2.4 to the nondeterministic case.

Lemma 5. *Given a family of automata $(M_u)_{u \in \mathcal{D}} = (\mathcal{A}, \mathcal{O}, \mathcal{S}_u, out_u, step_u)$, $out_u : \mathcal{S}_u \times \mathcal{D} \rightarrow \mathcal{O}$, $step_u : \mathcal{S}_u \times \mathcal{A} \rightarrow 2^{\mathcal{S}_u}$, and $\rightsquigarrow \subseteq \mathcal{D} \times \mathcal{D}$ such that*

$$A: dom(a) \not\rightsquigarrow u \implies step_u(s_u, a) = \{s_u\}.$$

*Then unwinding relations \sim_u satisfying *oc*, *nsc* and *nlr* exist for any automaton $M = (\mathcal{A}, \mathcal{O}, \mathcal{S}, out, step)$, $step : \mathcal{S} \times \mathcal{A} \rightarrow 2^{\mathcal{S}}$ for which there are $f_u : \mathcal{S} \rightarrow \mathcal{S}_u$, such that*

- 1: $out_u(f_u(s)) = out(s, u)$
- 2: $step_u(f_u(s), a) = F_u(step(s, a))$.

where $\Gamma : F_u(\mathcal{S}) = \{f_u(s) \mid s \in \mathcal{S}\}$.

Proof. By unwinding. The M_u can be transformed into unwinding relations \sim_u that satisfy the unwinding conditions *nlr*, *nsc* and *oc*. Let therefore $D : s \sim_u t \Leftrightarrow f_u(s) = f_u(t)$. From this follows $D' : S \simeq_u T \Leftrightarrow F_u(S) = F_u(T)$.

$$oc : s \sim_u t \implies out(s, u) = out(t, u)$$

Analog to 2.4.

$$nlr : dom(a) \not\rightsquigarrow u \implies \{s\} \simeq_u step(s, a)$$

Starting from $dom(a) \not\rightsquigarrow u$ we get $step_u(s_u, a) = \{s_u\}$ by means of *A*. Choosing an s such that $f_u(s) = s_u$ gives $step_u(f_u(s), a) = \{f_u(s)\}$. Application of 2 on the left side and Γ on the right side gives $F_u(step(s, a)) = F_u(\{s\})$. With D' we get $\{s\} \simeq_u step(s, a)$.

$$nsc : s \sim_u t \implies step(s, a) \simeq_u step(t, a)$$

Analog to 2.4. □

Lemma 6. *Assume an automaton $M = (\mathcal{A}, \mathcal{O}, \mathcal{S}, out, step)$, $step : \mathcal{S} \times \mathcal{A} \rightarrow 2^{\mathcal{S}}$ and equivalence relations \sim_u satisfying *oc*, *nlr*, and *nsc*. Then we can define automata M_u satisfying *A*, 1 and 2.*

Proof. Let \approx_u be the smallest equivalence relation containing \sim_u . Define f_u such that $f_u(s) = f_u(t) :\Leftrightarrow s \approx_u t$ and let out_u and $step_u$ be defined by 1 and 2. We then need to show that out_u and $step_u$ are uniquely defined and that *A* holds. This works largely along the lines of the corresponding proof for the deterministic case. □

4 Deterministic automaton + intransitive policy

As we have seen, in the transitive case we only have direct information flow or, looking at it from another perspective, no information *flow* at all. One generalization to this is that of *intransitive noninterference*, where information is allowed to be passed on from domain to domain.

4.1 NI_{di}

In order to allow intransitive information flow, the function *purge* is replaced with *ipurge* that allows observations to be made through a sequence of actions whose domains form a chain in the interference policy. Given an action sequence α and a domain u , the function $src : \mathcal{A}^* \times \mathcal{D} \rightarrow 2^{\mathcal{D}}$ computes the set of domains that will be observable to u after execution of α .

$$\begin{aligned} src(\lambda, u) &= \{u\} \\ src(a \circ \alpha, u) &= \begin{cases} src(\alpha, u) \cup \{dom(a)\} & \text{if } \exists v \in src(\alpha, u) \wedge dom(a) \rightsquigarrow v \\ src(\alpha, u) & \text{otherwise} \end{cases} \end{aligned}$$

ipurge then eliminates all actions that are not allowed to affect u neither directly nor through an appropriate sequence of actions α .

$$\begin{aligned} ipurge : \mathcal{A}^* \times \mathcal{D} &\rightarrow \mathcal{A}^* \\ ipurge(\lambda, u) &= \lambda \\ ipurge(a \circ \alpha, u) &= \begin{cases} a \circ ipurge(\alpha, u) & \text{if } dom(a) \in src(a \circ \alpha, u) \\ ipurge(\alpha, u) & \text{otherwise} \end{cases} \end{aligned}$$

Non-interference for the intransitive case is then defined as

$$NI_{di} : test(\alpha, u) = test(ipurge(\alpha, u), u)$$

4.2 UP_{wsc}

For the deterministic intransitive case, Rushby defined a weakened version of the unwinding condition *sc*:

$$wsc: s \sim_u t \wedge s \sim_{dom(a)} t \implies step(s, a) \sim_u step(t, a) \quad \text{weak sc}$$

He proves that $UP_{wsc} \implies NI_{di}$, but leaves the other direction open. That other direction is covered by Ron van der Meyden [2] who shows that UP_{wsc} is actually equal to the slightly different notion of *TA-security*, which is strictly stronger than NI_{di} .

4.3 UP_{wsc} and equivalence relations \sim_u

That unwinding relations satisfying UP_{wsc} can not necessarily be extended to equivalence relations can be shown by an example. Consider Figure 2 where $dom(a) = v \rightsquigarrow u$, $s \sim_u t$ and $t \sim_v s$. This can be enforced by further domains m and n that aren't allowed to influence u or v respectively. Right now *wsc* only implies $t' \sim_v s'$, but if we made \sim_u and \sim_v symmetric, $s' \sim_u t'$ would need to be added. This can be used to break *oc*.

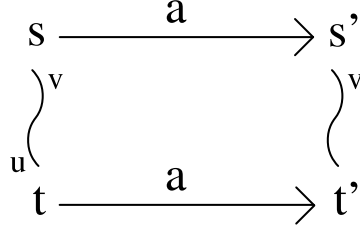


Figure 2: Symmetry of \sim_u and \sim_v would break *oc*

4.4 QA_{wsc}

The result of section 2.4 can be partly adapted to the case of an intransitive policy.

Lemma 7. *Given a family $(M_u)_{u \in \mathcal{D}}$ of automata $M_u = (\mathcal{A}, \mathcal{S}_u, \mathcal{O}, out_u, step_u)$, $out_u : \mathcal{S}_u \rightarrow \mathcal{O}$, $step_u : \mathcal{S}_u \times \mathcal{A} \rightarrow 2^{\mathcal{S}_u}$, such that*

$$A: dom(a) \not\rightsquigarrow u \implies step_u(s_u, a) = \{s_u\}.$$

Further, given a family $(M_{vu})_{v, u \in \mathcal{D}}$ of automata $M_{vu} = (\mathcal{A}, \mathcal{S}_{vu} = \mathcal{S}_v \times \mathcal{S}_u, step_{vu})$, $step_{vu} : \mathcal{S}_{vu} \times \mathcal{A} \rightarrow 2^{\mathcal{S}_{vu}}$, such that for all $v, u \in \mathcal{D}$ with $v \rightsquigarrow u$ we have

$$\alpha: dom(a) = v \wedge \{t_{vu}, t'_{vu}\} \subseteq step_{vu}(s_{vu}, a) \implies \pi_2(t_{vu}) = \pi_2(t'_{vu})$$

*Then relations \sim_u satisfying *oc*, *wsc* and *lr* exist for every automaton $M = (\mathcal{A}, \mathcal{S}, \mathcal{O}, out, step)$, $out : \mathcal{S} \times \mathcal{D} \rightarrow \mathcal{O}$, $step : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ for which functions $f_u : \mathcal{S} \rightarrow \mathcal{S}_u$ and $f_{vu} : \mathcal{S} \rightarrow \mathcal{S}_v \times \mathcal{S}_u$ exist, such that*

- 1: $out_u(f_u(s)) = out(s, u)$
- 2: $step_u(f_u(s), a) = \{f_u(step(s', a)) \mid f_u(s') = f_u(s)\}$
- 3: $step_{vu}(f_{vu}(s), a) = \{f_{vu}(step(s', a)) \mid f_{vu}(s') = f_{vu}(s)\}$
- 4: $f_{vu}(s) = (f_v(s), f_u(s))$

Proof. Let $s \sim_u t \Leftrightarrow f_u(s) = f_u(t)$ and show that *oc*, *wsc* and *lr* hold.

$$oc: s \sim_u t \implies out(s, u) = out(t, u) :$$

Analog to 2.4.

$$lr: dom(a) \not\rightsquigarrow u \implies s \sim_u step(s, a) :$$

From $dom(a) \not\rightsquigarrow u$ and *A* follows $\forall s_u. step_u(s_u, a) = \{s_u\}$. We choose $f_u(s)$ for s_u and get $step_u(f_u(s), a) = \{f_u(s)\} = \{f_u(step(s', a)) \mid f_u(s') = f_u(s)\}$ (using 2). For $s' = s$ we then have $f_u(step(s, a)) \in \{f_u(s)\}$ and since the set contains only one element: $f_u(s) = f_u(step(s, a))$. Use *D* to arrive at $s \sim_u step(s, a)$.

$$wsc: dom(a) \rightsquigarrow u \wedge s \sim_{dom(a)} t \wedge s \sim_u t \implies step(s, a) \sim_u step(t, a) :$$

Let $dom(a) = v$. From $s \sim_v t$ and $s \sim_u t$ and *D* follows that $f_v(s) = f_v(t)$ and $f_u(s) = f_u(t)$. Then also $(f_v(s), f_u(s)) = (f_v(t), f_u(t))$ and with 4: $f_{vu}(s) = f_{vu}(t)$. According to 3, $f_{vu}(step(s, a))$ and $f_{vu}(step(t, a)) \in step_{vu}(f_{vu}(s, a))$.

From α now follows $\pi_2(f_{vu}(step(s, a))) = \pi_2(f_{vu}(step(t, a)))$, from 4 then $f_u(step(s, a)) = f_u(step(t, a))$, and finally $step(s, a) \sim_u step(t, a)$ using D . \square

As unwinding relations satisfying UP_{wsc} are not always equivalence relations, the reverse direction to the above lemma does not hold.

Remark: The proof of the above lemma also works with 2 and 3 replaced by

$$\begin{aligned} 2': f_u(step(s, a)) &\in step_u(f_u(s), a) \\ 3': f_{vu}(step(s, a)) &\in step_{vu}(f_{vu}(s), a). \end{aligned}$$

Seeing it that way, the M_u and M_{vu} only offer a selection of secure transitions. Any automaton using a subset of these will be secure.

Remark: Determinism for each domain's own actions, i.e.

$$B : dom(a) = u \implies step_u(s_u, a) = \{t_u\}$$

follows from α and the reflexivity of \rightsquigarrow as follows:

$$\begin{aligned} step_u(f_u(s), a) &=_{2} \{f_u(step(s', a)) \mid f_u(s') = f_u(s)\} \\ &=_{4} \{\pi_2(f_{uu}(step(s', a))) \mid f_{uu}(s') = f_{uu}(s)\} \\ &=_{\bullet} \pi_2(\{f_{uu}(step(s', a)) \mid f_{uu}(s') = f_{uu}(s)\}) \\ &=_{3} \pi_2(step_{uu}(f_{uu}(s), a)) \\ &=_{\times} \{t_u = f_u(step(s, a))\} \end{aligned}$$

(For $=_{\times}$ we use α , $dom(a) = u$ and the reflexivity of \rightsquigarrow .)

5 Nondeterministic automaton + intransitive policy

5.1 NI_{ni}

The nondeterministic noninterference property for intransitive policies is just the combination of the above nondeterministic ($Test$) and intransitive ($ipurge$) cases:

$$NI_{ni} : Test(\alpha, u) = Test(ipurge(\alpha, u), u)$$

5.2 $UP_{nWSC}^{\rightsquigarrow}$

The problem with defining unwinding properties for the nondeterministic intransitive case lies in the conservation of weak step consistency. There are two immediate ways of dealing with that. Von Oheimb [3] generalizes Rushby's step consistency to uniform step consistency that takes into account all possible sets of domains. The other possibility is to stick with Rushby's weak step consistency but use equivalence relations \simeq_u over sets of states, as in section 3.2. This approach leads to

$$nWSC^{\rightsquigarrow} : S \simeq_u T \wedge S \simeq_{dom(a)} T \implies Step(S, a) \simeq_u Step(T, a)$$

For the following proofs, \simeq_u is expanded to sets of domains:

$$S \simeq_C T :\Leftrightarrow \forall u \in C. S \simeq_u T$$

Further, we assume \simeq_u to be equivalence relations.

Lemma 8. *Given an automaton M that is OC^\times and such that $Do(\alpha) \simeq_u Do(ipurge(\alpha, u))$. Then M is non-interferent.*

Proof. Analogous to Lemma 3. □

Lemma 9. *Given an automaton M that is $nWSC^\times$ and nLR^\times , then*

$$S \simeq_{src(a \circ \alpha, u)} T \implies Step(S, a) \simeq_{src(\alpha, u)} Step(T, a).$$

Proof. Let $v \in src(\alpha, u)$, then we need to show $Step(S, a) \simeq_v Step(T, a)$. From the definition of src follows that also $v \in src(a \circ \alpha, u)$. Using this and assuming $S \simeq_{src(a \circ \alpha, u)} T$, we get $S \simeq_v T$. We now consider two cases.

Case 1: $dom(a) \rightsquigarrow v$

From $dom(a) \rightsquigarrow v$ and $v \in src(\alpha, u)$ we know that $dom(a) \in src(a \circ \alpha, u)$, and thus we can derive $S \simeq_{dom(a)} T$ from $S \simeq_{src(a \circ \alpha, u)} T$.

Having $S \simeq_{dom(a)} T$ and $S \simeq_v T$ we get $Step(S, a) \simeq_v Step(T, a)$ by application of $nWSC^\times$.

Case 2: $dom(a) \not\rightsquigarrow v$

Using nLR^\times we get $S \simeq_v Step(S, a)$ and $T \simeq_v Step(T, a)$. Together with $S \simeq_v T$ from above and knowing that \simeq_v is an equivalence relation, we arrive at $Step(S, a) \simeq_v Step(T, a)$. □

Lemma 10. *Given an automaton M that is nLR^\times , then*

$$dom(a) \not\rightsquigarrow src(a \circ \alpha, u) \implies S \simeq_{src(\alpha, u)} Step(S, a).$$

Proof. Let $v \in src(\alpha, u)$. From the definition of src and the assumption follows that $dom(a) \not\rightsquigarrow v$. Using nLR^\times we arrive at $S \simeq_v Step(S, a)$. □

Lemma 11. *Given an automaton M that is $nWSC^\times$ and nLR^\times , then*

$$S \simeq_{src(\alpha, u)} T \implies Run(S, \alpha) \simeq_u Run(T, ipurge(\alpha, u)).$$

Proof. By induction over the length of α . The base case is $\alpha = \lambda$ and is easily shown by application of definitions. In the inductive step we assume the hypothesis for α and need to show

$$S \simeq_{src(a \circ \alpha, u)} T \implies Run(S, a \circ \alpha) \simeq_u Run(T, ipurge(a \circ \alpha, u)).$$

We consider two cases.

Case 1: $dom(a) \in src(a \circ \alpha, u)$, and thus $ipurge(a \circ \alpha, u) = a \circ ipurge(\alpha, u)$ and we proceed as follows:

$$\begin{array}{ccc}
S & \simeq_{src(a \circ \alpha, u)} & T \\
& \Downarrow \text{Lemma 9} & \\
Step(S, a) & \simeq_{src(\alpha, u)} & Step(T, a) \\
& \Downarrow \text{i.h.} & \\
Run(Step(S, a), \alpha) & \simeq_u & Run(Step(T, a), ipurge(\alpha, u)) \\
& \Downarrow Run & \\
Run(S, a \circ \alpha) & \simeq_u & Run(T, a \circ ipurge(\alpha, u)) \\
& \Downarrow ipurge & \\
Run(S, a \circ \alpha) & \simeq_u & Run(T, ipurge(a \circ \alpha, u))
\end{array}$$

Case 2: $dom(a) \notin src(a \circ \alpha, u)$, then $ipurge(a \circ \alpha, u) = ipurge(\alpha, u)$.

From $S \simeq_{src(a \circ \alpha, u)} T$ directly follows $S \simeq_{src(\alpha, u)} T$.

Application of Lemma 10 on $dom(a) \notin src(a \circ \alpha, u)$ gives $S \simeq_{src(\alpha, u)} Step(S, a)$.

Because $\simeq_{src(\alpha, u)}$ is an equivalence relation, we also have

$$Step(S, a) \simeq_{src(\alpha, u)} T.$$

Plugging this into the inductive hypothesis gives

$$Run(Step(S, a), \alpha) \simeq_u Run(T, ipurge(\alpha, u)),$$

which can be transformed to $Run(S, a \circ \alpha) \simeq_u Run(T, ipurge(a \circ \alpha, u))$ in a similar way as in the first case. \square

Theorem 12. *Given an automaton M that is OC^\simeq , nLR^\simeq and $nWSC^\simeq$. Then M is secure.*

Proof. Let $S = T = \{s_0\}$ in Lemma 11. Then we arrive at

$$Do(\alpha) \simeq_u Do(ipurge(\alpha, u)),$$

which by Lemma 8 makes M secure. \square

For the other direction we can say that UP_{nWSC}^\simeq are incomplete for NI_{ni} for the same reason UP_{wsc} are incomplete for NI_{di} .

Formally, based on both UP_{nWSC}^\simeq or uniform step consistency, it should be possible to define properties over quotient automata that guarantee the complete automaton's security. The problem in both cases is that the resulting set of automata is likely too large and unintuitive to be of practical use as a specification tool. In the case of UP_{nWSC}^\simeq we have the same problem as with UP_{nSC}^\simeq , but with the added complexity of the product automata (M_{uv}). In the case of uniform step consistency we would need to define $2^{|\mathcal{D}'|}$ automata, one for every subset $\mathcal{D}' \subseteq \mathcal{D}$.

We leave a final decision on the practicality of this approach to future work.

5.3 UP_{nwsc}

Again, we try to use a stronger property employing a relation over states. By making the same restriction on $nWSC^\simeq$ as we did in the case of nSC^\simeq we get the lifted variant of wsc :

$$nwsc: s \sim_u t \wedge s \sim_{dom(a)} t \implies step(s, a) \simeq_u step(t, a)$$

However, $nwsc$ does not imply $nWSC^\simeq$ and thus the above proof is not applicable. It is best shown by an example that an automaton satisfying $nwsc$ is not necessarily secure. Consider figure 3. Let $\mathcal{D} = \{u, v, x\}$, $v \rightsquigarrow u$, $x \not\rightsquigarrow \{u, v\}$. The only purpose of x is to make sure that $s \sim_{\{u, v\}} t$. All other transitions belong to v . With $nwsc$ follows $\{e, h\} \simeq_{\{u, v\}} \{g, f\}$, which can be forced into the given relations by setting out_u and out_v appropriately. $nWSC^\simeq$ would now imply that o and p had the same output for u . $nwsc$ however is not triggered and thus by assigning different output to o and p we can observe whether transition x was taken or not.

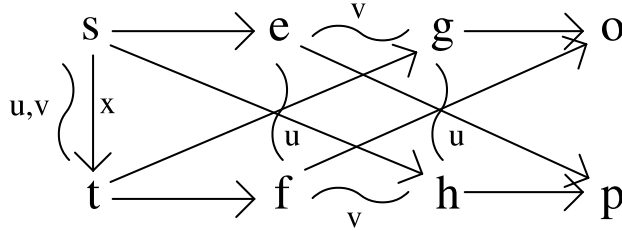


Figure 3: $UP_{nwsc} \not\Rightarrow NI_{ni}$

5.4 QA_{nwsc}

For the sake of completeness it should be noted that it is possible to define quotient automata that enforce UP_{nwsc} . They have no real relevance though, as their use does not produce a secure automaton.

6 Overview

Having all four cases explored we can now give a diagrammatic overview of the situation in Figure 4. As can be seen, the 'farther down' we get (i.e. the more complex the security property), the more remote the correlation between the quotient automata and the non-interference property will be.

QA_{sc}	\iff	UP_{sc}	\iff	NI_{dt}		
QA_{nsc}	\iff	UP_{nsc}	\implies	UP_{nSC}^\simeq	\iff	NI_{nt}
QA_{wsc}	\implies	UP_{wsc}	\iff	TA	\implies	NI_{di}
QA_{nwsc}	\implies	UP_{nwsc}	\times	UP_{nWSC}^\simeq	\implies	NI_{ni}

Figure 4: Decreasing correlation of properties

7 Finding quotient automata

We also implemented an algorithm that finds quotient automata for a given system. It works on the explicit state space by successively building state regions of equivalent states for every domain. The procedure depends on whether the system is deterministic and whether the policy is transitive or not. In any way, it is based on simple observations about the unwinding properties:

- lr can not be broken by making more states equivalent
- oc can not be broken by making more states distinct

Therefore these two properties define upper and lower bounds on the size of regions of equivalent states. The oc -bound would make all states equivalent that have the same output. Any additional equivalence between states breaks oc . The lr -bound would make all states equivalent that are connected by actions invisible to u . Any further distinction between these breaks lr .

We consider the case UP_{dt} first. Starting from the lr -bound, sc will tell us exactly which states (and thereby state regions) need to be equivalent. We proceed by successively merging these regions until either sc is satisfied or oc is violated. In the first case we end up with a secure quotient automaton, in the second we know that the given system is not secure. We could also start from the oc -bound and split regions until either sc is satisfied or lr is violated. The final regions need not be the same in both cases.

In the case of a nondeterministic automaton starting from the lr -bound is difficult because nsc does not tell us which regions to merge. However, starting from the oc -bound still leads to a result in a deterministic fashion.

The opposite is the case for a deterministic automaton under an intransitive policy. wsc makes it difficult to go *backwards* (from the oc -bound) because one cannot tell how to split regions, while going forward is still fully determined.

In the last case of $nwsc$ neither going forward nor backward leads to a deterministic algorithm. We do, however, already know that there is no relation between quotient automata and non-interference in this case, anyway.

8 Conclusion and future work

We have laid a formal basis for the construction of secure systems from a set of behavioral abstractions.

An open problem is still how to actually construct a complete system from a set of quotient automata. Generally, each set of quotient automata corresponds not with a single secure automaton but with a set of secure automata. Which of these is desired lies in the eyes of the specifier.

Therefore a future question is to what extent the secure automaton can be automatically constructed. Maybe there are only a few degrees of freedom that distinguish the possible automata. Ideally the specifier would make a few simple choices and have the rest of the automaton automatically generated. Whether this is possible needs to be explored.

References

- [1] John Rushby. Noninterference, transitivity and channel-control security policies.
- [2] Ron van der Meyden. What, indeed, is intransitive noninterference?, 2007.
- [3] David von Oheimb. Information flow control revisited: Noninfluence = Noninterference + Nonleakage. In P. Samarati, P. Ryan, D. Gollmann, and R. Molva, editors, *Computer Security – ESORICS 2004*, volume 3193 of *LNCS*, pages 225–243. Springer, 2004.