Algorithmic Verification of Noninterference Properties

Ron van der Meyden, Chenyi Zhang

School of Computer Science and Engineering,
University of New South Wales,
Sydney, Australia

National ICT Australia,
Locked Bag 6016, Sydney,
NSW 1466, Australia

Abstract

The paper discusses the problem of model checking a number of noninterference properties in finite state systems: noninterference, nondeducibility on inputs, generalized noninterference, forward correctability and restrictiveness. The complexity of these problems is characterized, and a number of possible heuristics for optimization of the model checking are discussed.

Keywords: Noninterference, Model Checking, Complexity

1 Introduction

The notion of 'noninterference' is a general term applied in the security literature to a number of causality-like notions intended to capture the intuition that information does not flow from high level users to lower level users, so that confidentiality of high level information is maintained. The main approach to verification that systems satisfy these properties has been proof theoretic methods using so-called 'unwinding conditions'. In this paper, we investigate the applicability of algorithmic verification techniques when the systems in question are finite state. We develop algorithms for model checking a number of different noninterference notions, and characterise the computational complexity of the associated verification problems. In particular, we deal with noninterference on deterministic systems [11,24], nondeducibility on inputs [25], generalized noninterference [17], forward correctability [14] and restrictiveness [17].

Noninterference has been studied under several distinct semantic models, including state based models[11,24], trace-set models[19,29] and process algebras [22,7].

⋆ National ICT Australia is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: `www.elsevier.nl/locate/entcs`

Only for the latter has there been a systematic study of algorithmic verification of these notions [7,8]. The process algebraic models are the most expressive, and definitions of noninterference notions on other models can be reduced to definitions of noninterference notions on a process algebraic model by means of natural mappings between the models [27]. However, state based system modelling approaches are more natural to many, are likely to be adequate for many applications, have a more extensive literature on algorithmic verification, and have a more highly developed set of verification tools. This modelling approach also remains the predominant approach in operating systems verification efforts [13], the area originally motivating the noninterference literature. It therefore makes sense to consider the algorithmic verification problem also on state based models. This is particularly so with respect to complexity bounds, where lower bounds proved for a more expressive semantic model may not apply on a more restrictive model. We therefore focus in this paper on a state-based modelling of systems, and (to make the verification problem decidable) restrict attention to finite state systems.

The contributions of the paper are as follows. First, we show that noninterference in deterministic systems can be reduced to a *safety* property, so it is expressible in both branching time and linear time temporal logics and verifiable in polynomial time by existing model checkers. Also in PTIME is the notion of restrictiveness on nondeterministic systems. We show that the remaining notions of noninterference on nondeterministic systems that we consider are PSPACE-complete. For some of these notions (restrictiveness and nondeducibility on inputs), these results are closely related to results of Focardi and Gorrieri [7,8] (but on a more restricted semantic model, hence not immediate consequences for the lower bounds). The results on generalised nondeducibility and forward correctability are new, as far as we know. Finally, we discuss heuristics that may be applied to the verification of noninterference notions, and give complexity arguments that suggest that this may sometimes lead to optimizations.

2 State-Observed Model

The state based system models in the literature on noninterference can be roughly classified into two distinct types, depending on whether observations are associated with states [20,3,23] or actions [11,24]. The system definitions are similar to those of finite state automata, with the distinction between the two types resembling the Mealy/Moore distinction. It can be shown [27] that there exist natural mappings between these two types of models that preserve all the security notions that we consider in this paper. Consequently, we consider only the state-observed modeling. The systems are input-enabled, in the sense that any action can be taken at any time. Most of the literature restricts attention to two agents High (H) and Low (L) and the security policy $L \leq H$. This policy permits information to flow from Low to High but not from High to Low. We also make this restriction here, and take the set of agents (also called *domains*) to be $D = \{L, H\}$.

A nondeterministic *state-observed* state machine is a tuple of the form $M = \langle S, s_0, next, obs, dom, A\rangle$ where S is a set of states; $s_0 \in S$ is the initial state; A is a set of actions; the function $next : S \times A \to \mathcal{P}(S) \setminus \{\emptyset\}$ is a transition

function, such that $next(s, a)$ defines the set of states to which it is possible to make a transition when action $a \in A$ is performed at a state $s \in S$; the function $dom : A \to D$ associates a security domain with each action, and the function $obs : S \times D \to O$ describes the observation made in each state by each security domain. For readability, we 'curry' the function obs by obs_u of type $S \to O$ if $u \in D$. Such a state-machine is *deterministic* if $next(s, a)$ is a singleton for all states s and actions a. In this case we may define a function $step : S \times A \to S$ by $next(s, a) = \{step(s, a)\}$. We write $\mathbb{M}_{ns}$ for the set of all nondeterministic state-observed machines, and $\mathbb{M}_s$ for the set of all deterministic state-observed machines.

A *run* of a state-observed system is a sequence $r = s_0 a_1 s_1 a_2 s_2 \dots a_n s_n \in S(AS)^*$ such that for all $1 \leq i \leq n$, $s_i \in next(s_{i-1}, a_i)$. Define $r(i) = s_i$ to be the i-th state on the run r and $r^a(j) = a_j$ to be the j-th action. We use two types of concatenation operation on sequences. We write $\alpha \cdot \beta$ for the usual notion of concatenation. A run can also be described as a *fusion* of two sequences: we write $r = r_1 \circ r_2$ if there exists m with $1 \leq m \leq n$ such that $r_1 = s_0 a_1 s_1 a_2 s_2 \dots a_m s_m$ and $r_2 = s_m a_{m+1} s_{m+1} a_{m+2} s_{m+2} \dots a_n s_n$.

3 State Based Security Definitions

In this section we recall from the literature a number of classical security definitions. Some of them are state based and some were originally defined as a trace-set property, in which case we give a corresponding definition in our system model.

Several of the definitions are cast in terms of a notion of *view* capturing the information at an agent's disposal in a run. We take the view to be the maximal information that an agent can have in a nondeterministic asynchronous system: its sequence of actions and observations reduced modulo stuttering. Let $Cond : X^* \to X^*$ be the function which condenses a sequence of elements into a possibly shorter sequence by removing stuttering, such that for all $a, b \in X$, $\alpha \in X^*$, $Cond(\epsilon) = \epsilon$, $Cond(a) = a$,

$$\text{and } Cond(\alpha \cdot a \cdot b) = \begin{cases} Cond(\alpha \cdot a) \cdot b & \text{if } \ a \neq b, \\ Cond(\alpha \cdot a) & \text{otherwise.} \end{cases}$$

Definition 3.1 For $u \in D$, define the observation function $Obs_u : S(AS)^* \to O^+(AO^+)^*$ on a run by $Obs_u(s) = obs_u(s)$, and

$$Obs_u(\delta \cdot a \cdot s) = \begin{cases} Obs_u(\delta) \cdot a \cdot obs_u(s) & \text{if } \ dom(a) = u \\ Obs_u(\delta) \cdot obs_u(s) & \text{otherwise.} \end{cases}$$

Define the function $view_u : S(AS)^* \to O^+(AO^+)^*$ by $view_u(r) = Cond(Obs_u(r))$.

Note that an agent may make the same observation several times in a row, without an intervening action by that agent. This indicates that another agent has acted. To eliminate this timing-based reasoning, in order to make the definition compatible with the assumption of asynchrony, we apply the function $Cond$ in this definition.

3.1 Noninterference

Historically, one of the first information flow properties was (transitive) noninterference [11,12], defined with respect to deterministic machines. We base our discussion on the presentation of Rushby [24], which has been followed in many other works. As noted above, in state-observed deterministic systems, we have a function $step : S \times A \to S$ to represent the deterministic state evolution as a result of actions. To represent the result of executing a sequence of actions, define the operation $\bullet : S \times A^* \to S$, by $s \bullet \epsilon = s$; and $s \bullet (\alpha \cdot a) = step(s \bullet \alpha, a)$.

With respect to the simple policy $L \leq H$, the definition of noninterference can be described in terms of the operation $purge_L : A^* \to A_L^*$ on sequences of actions that restricts the sequence to the subsequence of actions of L. Intuitively, the purged H actions are not allowed to lead to any effects observable to L. This is formalised as follows in the definition of noninterference.

Definition 3.2 A (deterministic) system in $\mathbb{M}_s$ satisfies *Noninterference* if for all $\alpha \in A^*$, we have $obs_L(q_0 \bullet \alpha) = obs_L(q_0 \bullet purge_L(\alpha))$. We write NI_s for the set of such systems.

3.2 Nondeducibility on Inputs

One way of understanding the statement that H does not interfere with L in a deterministic system is as stating that every sequence of H actions is compatible with the actions and observations of L. This leads to the proposal to take a similar notion as the formulation of noninterference in nondeterministic systems: an approach known as nondeducibility [25]. Nondeducibility is defined in a quite general way, in terms of a pair of *views* of runs. We focus here on a commonly used special case: L's nondeducibility of H's actions.

To state the definition of nondeducibility, we also require a function to extract the sequence of actions performed by an agent. We write $Act_u(r)$ for the sequence of actions performed by agent u in run r, and $Act(r)$ the whole action sequence from all the agents in r.

Definition 3.3 [1] A system M satisfies *Nondeducibility on Inputs* if for every $\alpha \in A_H^*$, and every observation sequence β such that there exists a run r of M with $view_L(r) = \beta$, there exists a run r' of M with $Act_H(r') = \alpha$ and $view_L(r') = \beta$. Write NDI_s for the set of systems in $\mathbb{M}_{ns}$ satisfying Nondeducibility on Inputs.

3.3 Generalised Noninterference

Generalised Noninterference (GN) was proposed in [17] to generalise noninterference to nondeterministic systems. The original definition of GN is a trace based property with the intuition that the changes on high-level input must not alter the possible future sequences of low-level events (by modifying H outputs somewhere). Here we formulate it on our state based systems as:

Definition 3.4 A system M satisfies Generalised Noninterference (GN) if

[1] In [27] it has been shown that Nondeducibility on Inputs is equally strong as Nondeducibility on Strategies in *purely asynchronous* systems, though this is *not* true on synchronous machines due to [28].

(i) for all runs r of M with $Act(r) = \alpha \cdot \alpha'$, and for all $a \in A_H$, there exists another run r' such that $Act(r') = \alpha \cdot a \cdot \alpha'$ and $view_L(r) = view_L(r')$

(ii) for all runs r of M with $Act(r) = \alpha \cdot a \cdot \alpha'$ with $a \in A_H$, there exists another run r' such that $Act(r') = \alpha \cdot \alpha'$ and $view_L(r) = view_L(r')$.

Write GN_s for the set of systems in $\mathbb{M}_{ns}$ satisfying Generalised Noninterference.

Note that this definition implies that every possible L observation is consistent with every sequence of H Actions, so GN_s is at least as strong as NDI_s. However, GN_s is seemingly a stronger notion than NDI_s in that the latter allows L to rule out certain possible H/L action interleavings whereas GN_s requires that all interleavings are consistent.

3.4 Forward Correctability

Forward Correctability (FC) was first introduced in [14]. Similar to Generalised Noninterference, FC was defined as a property on traces. We formulate it as follows in state-observed systems. Define the relation $\equiv$ on runs by $r_1 \equiv r_2$ if r_1 and r_2 have the same length $n \in \mathbb{N}$, $r_1^a(j) = r_2^a(j)$ for all $1 \leq j \leq n$, and $obs_u(r_1(i)) = obs_u(r_2(i))$ for all $0 \leq i \leq n$ and $u \in D$.

Definition 3.5 A system M satisfies *Forward Correctability* (FC) if

(i) for all runs $r = r_1 \circ r_2$ of M such that $Act(r_1) = \alpha$ and $Act(r_2) = \alpha'$ with $\alpha' \in A_L^*$, for all $a \in A_H$, there exists a run $r' = r_1' \circ r_2'$ with $r_1 \equiv r_1'$, $Act(r_2') = a \cdot \alpha'$ and $view_L(r) = view_L(r')$

(ii) for all runs $r = r_1 \circ r_2$ of M such that $Act(r_1) = \alpha$ and $Act(r_2) = a \cdot \alpha'$ with $a \in A_H$ and $\alpha' \in A_L^*$, there exists a run $r' = r_1' \circ r_2'$ with $r_1 \equiv r_1'$, $Act(r_2') = \alpha'$ and $view_L(r) = view_L(r')$.

Write FC_s for the set of systems in $\mathbb{M}_{ns}$ satisfying Forward Correctability.

FC_s is seemingly stronger than GN_s because any 'perturbation' must be correctable in the future for FC_s but GN_s allows it to be correctable either in the past or in the future.

3.5 Restrictiveness

There are two versions of 'Restrictiveness' introduced in McCullough's early works. The former [16] is a trace-based definition, while the latter is essentially defined on labelled transition systems [17,18]. In [18] McCullough mentions both definitions and concludes that the one on labelled transition systems is a stronger notion. Here we follow his latter definition. This is close to the notion of unwinding relation which is a way of facilitating proofs of traditional noninterference on deterministic systems [24].

Definition 3.6 An *unwinding relation* for a system $M \in \mathbb{M}_s$ is an equivalence relation $\sim_L$ on the states of M satisfying the following conditions, for all states s, t and actions a:[2]

[2] We present a slight modification of the usual definition, which would have an equivalence relation $\sim_u$ for

- *Output Consistency*: if $s \sim_L t$ then $obs_L(s) = obs_L(t)$;
- *Locally Respects*: if $a \in A_H$ then $s \sim_L step(s, a)$;
- *Step Consistency*: if $a \in A_L$ and $s \sim_L t$ then $step(s, a) \sim_L step(t, a)$.

The relationship between unwinding conditions and noninterference is given by the following classical results:

Theorem 3.7 [12,24]

(i) *If there exists an unwinding relation for $M \in \mathbb{M}_s$, then $M \in NI_s$.*

(ii) *If $M \in NI_s$ then there exists an unwinding relation for M.*

The following is a natural generalization of Definition 3.6 to nondeterministic systems.

Definition 3.8 An unwinding relation for a system $M \in \mathbb{M}_{ns}$ is an equivalence relation satisfying

- OC: if $s \sim_L t$ then $obs_L(s) = obs_L(t)$.
- LR: if $a \in A_H$ and $t \in next(s, a)$ then $s \sim_L t$,
- SC: if $a \in A_L$ and $s \sim_L s'$ and $t \in step(s, a)$, then there exists $t' \in step(s', a)$ such that $t \sim_L t'$.

McCullough's [18] restrictiveness definition is similar in spirit to unwinding but distinguishes between inputs and outputs on actions. Since on state-observed systems, outputs are 'embedded' in states, the above is a somewhat simplified version of what was introduced by McCullough, and we have the following definition of restrictiveness on state-observed systems.

Definition 3.9 $M \in \mathbb{M}_{ns}$ satisfies *restrictiveness*, written $M \in RES_s$, if there exists an unwinding relation for M.

The property RES_s is inherently stronger than the other security notions we have introduced here, since intuitively every 'perturbation' from H always leads L to a state which is observationally bisimilar to its original state. The following summarizes the known relations between the definitions we have introduced above.

Proposition 3.10 *The following containments are strict: $RES_s \subset FC_s \subset GN_s \subset NDI_s$. On deterministic systems, the notions NI_s, NDI_s, GN_s, FC_s and RES_s are equivalent.*

We also note that observation on the whole history path seems no stronger than observation only on the last state given that the system is deterministic.

4 Verifying Noninterference Properties

We now turn to the main interest of this paper: verification methods and complexity results for the security properties introduced in the previous section. None of these

each agent u, satisfying a similar set of conditions for each u. For the policy $L \leq H$ we can take $\sim_H$ to be the universal relation, which automatically satisfies the necessary conditions.

properties is directly expressible in the traditional safety-liveness framework [1], as they express not a constraint on single system execution, but rather a constraint on the set of all possible executions. We therefore need to develop new techniques for their verification.

4.1 Unwinding Characterizable Properties

In this section we consider the property NI_s and its generalization RES_s, both of which can be characterized by an unwinding relation.

The following is a way to decide noninterference NI_s on deterministic state-observed systems by a doubling construction. Given a deterministic system $M = \langle S, s_0, step, obs, dom, A\rangle$, define $M^2 = \langle S^2, s_0^2, step^2, obs^2, dom, A\rangle$ to be the system with identical actions and domains, with states $S^2 = S \times S$, initial state $s_0^2 = (s_0, s_0)$, observation function $obs^2 : D \times S^2 \to (O \times O)$ given by $obs_u^2(s,t) = (obs_u(s), obs_u(t))$ for $s, t \in S$, and transition function $step^2 : S^2 \times A \to S^2$ given by $step^2((s_1, s_2), a) = (step(s_1, a), step(s_2, a))$ for $a \in A_L$ and $step^2((s_1, s_2), a) = (step(s_1, a), s_2)$ for $a \in A_H$.

Note that in every transition, $a \in A_H$ is applied only on the left part of each state pair. An easy induction shows that for every sequence of actions $\alpha \in A^*$, if $s_0^2 \bullet \alpha = (s, t)$ in M^2, then in M we have $s = s_0 \bullet \alpha$ and $t = s_0 \bullet purge_L(\alpha)$. We therefore obtain the following:

Proposition 4.1 *For $M \in \mathbb{M}_s$, we have $M \in NI_s$ iff in M^2, for all states (s,t) reachable from s_0^2, we have that $obs_L^2((s,t)) = (o, o')$ implies $o = o'$.*

Now NI_s is reduced to a safety property, which says M^2 will never reach a pair of states (s,t) on which L has a pair of different views. This enables noninterference to be checked using standard model checking technology, for both linear and branching time.

Corollary 4.2 *For $M \in \mathbb{M}_s$, checking $M \in NI_s$ can be done in time $\mathcal{O}(|S|^2 \times |A|)$ and additional space $\mathcal{O}(|S|^2)$.*

Proof. The system M^2 has at most $|S|^2$ states. Performing a search algorithm to traverse every possibly reachable state by trying every possible action takes $|S|^2 \times |A|$. Marking states reached requires space $|S|^2$, in addition to the space needed to represent M. □

Barthe et al. [2] and Davas et al. [5] proposed a self-composition technique to reason about language based noninterference properties, which is somewhat similar to our method for NI_s. However, their definitions of noninterference are targeted at reasoning about programming languages and assume that a single input is given at the beginning and a single output observed at the end of the computation, whereas we deal with systems permitting an arbitrary sequence of actions to be performed by two distinct agents, and generating outputs throughout the computation.

The property RES_s can be regarded as a nondeterministic version of NI_s because they both are characterizable by the existence of an unwinding relation. This property can be characterized using fixpoints as follows. Define the operator $T_L : \mathcal{P}(S \times S) \to \mathcal{P}(S \times S)$ by $(p, q) \in T_L(X)$ iff

- $(p, q) \in X$ and $obs_L(p) = obs_L(q)$
- for all $p' \in step(p, a)$ and $a \in A_L$ there exists $q' \in step(q, a)$ such that $(p', q') \in X$
- for all $q' \in step(q, a)$ and $a \in A_L$ there exists $p' \in step(p, a)$ such that $(p', q') \in X$.

The operator T_L is *monotonic*, in the sense that $\sim_1 \subseteq \sim_2$ implies $T_L(\sim_1) \subseteq T_L(\sim_2)$. The set of binary relations on S and the subset relation ($\subseteq$) have the structure of a complete lattice. The Knaster-Tarski theorem [26] asserts the existence of a least and greatest fixed point operator on a complete lattice. We write $\nu X.T_L(X)$ for the greatest binary relation $\sim$ satisfying $\sim \subseteq T_L(\sim)$. The following result characterizes RES_s in terms of T_L and the property LR (in Definition 3.8).

Proposition 4.3

(i) *$M \in \mathbb{M}_{ns}$ satisfies RES_s iff there exists an equivalence relation $\sim \subseteq S \times S$ satisfying $\sim = T_L(\sim)$ and LR.*

(ii) *$M \in \mathbb{M}_{ns}$ satisfies RES_s iff $\nu X.T_L(X)$ satisfies LR.*

The understanding of the property RES_s in Proposition 4.3(ii) yields several algorithmic approaches to its verification. One approach is symbolic, noting that given a BDD encoding of M, the operation T_L is readily encoded as an operation on BDDs, and the computation of $\nu X.T_L(X)$ and the verification that it satisfies LR can be implemented using standard operations on BDDs.

We also obtain a bisimulation-based approach. From Definition 3.8, an unwinding relation is essentially a strong bisimulation relation with respect to A_L, and Proposition 4.3 requires the largest $\sim$. It is known that computing the largest bisimulation on a labelled transition system can be reduced to the problem of finding the coarsest partition, which is computable in $\mathcal{O}(|M| \times |S|)$ by Kanellakis and Smolka's algorithm [15] and in $\mathcal{O}(|M| \times log_2(|S|))$ by Paige and Tarjan's algorithm [21], where $|M|$ is the number of transitions. In our case $|M| = |S|^2 \times |A_L|$, and we start from an initial partition corresponding to the equivalence relation $\approx$ defined by $s \approx t$ iff $obs_L(s) = obs_L(t)$. To verify LR we need to check every H transition, which takes $|S|^2 \times |A_H|$. The space requirement is comparable to the size of the system itself in [21].

Theorem 4.4 *Given $M \in \mathbb{M}_{ns}$, M in RES_s is verifiable in $\mathcal{O}(|S|^2 log_2(|S|) \times |A_L| + |S|^2 \times |A_H|)$ time and space $\mathcal{O}(|M|)$.*

If M is deterministic, the size of the transition relation $|M|$ becomes $|S| \times |A|$ and the time complexity of the bisimulation algorithm in [15] reduces to $\mathcal{O}(|S|^2 \times |A_L|)$. Thus, the complexity for the whole procedure becomes $\mathcal{O}(|S|^2 \times |A_L| + |S| \times |A_H|)$ time and and space linear in $|M|$. This is marginally better than than the result in Proposition 4.2 on time and better on space. However, the reduction to a classical model checking problem in Proposition 4.2 permits various optimization techniques to be used (e.g., partial order reductions) so it is unclear which technique will perform better in practice.

We note that another way to approach these results is by a reduction to results of Bossi and Focardi et al. [4,9] who defined a property P_BNDC as a bisimulation based properties on labelled transition systems (LTS) and proved a polynomial time complexity result for it. We may define a linear time translation $F_{sl} : \mathbb{M}_{ns} \to \mathbb{L}^{IO}$

from state observed system into τ-free and input-enabled LTS as follows. Assuming $O = O_H \dot\cup O_L$, for $M = \langle S, s_0, step, obs, dom, A\rangle \in \mathbb{M}_{ns}$, define $F_{sl}(M) = \langle P, p_0, \rightarrow, \mathcal{L}\rangle$ where

(i) $P = S$, $p_0 = s_0$

(ii) $\mathcal{L} = A \cup O$

(iii) $\rightarrow = \{(s,a,t) | \exists a \in A : t \in step(s,a)\} \cup \{(s,o,s) | \exists o \in O, u \in D : o = obs_u(s)\}$.

The following relates RES_s to a property on labelled transition systems.

Proposition 4.5 *$M \in \mathbb{M}_{ns}$ in RES_s iff $F_{sl}(M)$ is in P_BNDC.*

It follows that checking RES_s has a polynomial time upper bound, from [9]. In particular, [9]'s algorithm for P_BNDC reduces a weak bisimulation problem into checking strong bisimulation with an additional step of transitive closure on A_H. Their algorithm works in $\mathcal{O}(|\rightarrow| \times log_2(|S|))$ in general, where $|\rightarrow|$ is comparable to $|S|^2 \times |A|$ in our approach. The complexities of our direct approach and this approach by reduction are therefore essentially equivalent.

4.2 Trace Set Properties

Verifying the remaining properties NDI_s, GN_s and FC_s proves to be more complex than NI_s and RES_s. In this section we prove the following:

Theorem 4.6 *For $\mathcal{P}$ any of NDI_s, GN_s or FC_s, the problem of deciding $M \in \mathcal{P}$ is PSPACE-complete.*

For the lower bound part of this result, we use the following polynomial time reduction to convert the classical problem of deciding if the language accepted by a nondeterministic finite state automaton is equal to Σ^* into the problem of verifying any of the following: NDI_s, GN_s and FC_s. Let $A = \langle S, \rightarrow, \Sigma, s_0, \mathcal{F}\rangle$ be a nondeterministic finite state automaton (without ϵ-transitions) which does not accept ε, where S is the set of states, Σ the alphabet, $\rightarrow$ the transition relation, s_0 the initial state and $\mathcal{F}$ the set of final states. Define $M(A) = \langle S^m, s_0, obs, next, dom, A^m\rangle$ to be the system with

- $S^m = S \dot\cup S^\Sigma$, where $S^\Sigma = \{s'_0, s'_1, s'_2\}$
- $A^m = \Sigma \dot\cup \{h\}$ with $dom(a) = L$ for all $a \in \Sigma$ and $dom(h) = H$
- $obs : D \times S^m \rightarrow \{0,1\}$ with $obs_H(s) = 0$ for all $s \in S^m$ and $obs_L(s) = 0$ for all $s \in S \cup \{s'_0, s'_2\}$, and $obs_L(s) = 1$ if $s = s'_1$
- $next : S^m \times A \rightarrow \mathcal{P}(S^m)$ defined as follows
  - For $a = h$: $next(s_0, h) = \{s'_0\}$, $next(s, h) = \{s\}$ if $s \neq s_0$
  - For $a \in \Sigma$: $next(s'_0, a) = S^\Sigma$, $next(s'_1, a) = \{s'_1\}$ and $next(s'_2, a) = \{s'_2\}$. For $s \in S$, $next(s,a) = \{s'_2\}$ if there does not exist t such that $s \xrightarrow{a} t$. Otherwise, $next(s,a) = \{t \in S | s \xrightarrow{a} t\} \cup \{s'_1\}$ if $\{t \in \mathcal{F} | s \xrightarrow{a} t\} \neq \emptyset$, and $next(s,a) = \{t \in S | s \xrightarrow{a} t\}$ if $\{t \in \mathcal{F} | s \xrightarrow{a} t\} = \emptyset$.

The construction of $M(A)$ from A can be done in polynomial time.

Proposition 4.7 *Let $\mathcal{P}$ be any of the properties NDI_s, GN_s, FC_s. Then $\mathcal{L}(A) = \Sigma^* \backslash \{\varepsilon\}$ iff $M(A) \in \mathcal{P}$*

Deciding if the language accepted by a nondeterministic finite state automaton equals $\Sigma^*\backslash\{\varepsilon\}$ with $|\Sigma| \geq 2$ is known to be a PSPACE-complete problem [10], so all the above security properties are PSPACE hard. Next we will show each is solvable in polynomial space.

Lemma 4.8 *$M \in \mathbb{M}_{ns}$ in NDI_s iff for every possible low observation $\beta \in O^+(AO^+)^*$, there exists a run $r \in S(AS)^*$ such that $view_L(r) = \beta$ and $Act_H(r) = \epsilon$.*

Lemma 4.9 *If $M \in \mathbb{M}_{ns}$ in NDI_s then for every reachable state s and $t \in next(s, a)$ with $a \in A_H$, we have $obs_L(s) = obs_L(t)$.*

Lemma 4.8 shows a system is in NDI_s iff H's actions do not cause more observations to L than if H does nothing. The following definitions sketch a reduction from NDI_s into a regular language equivalence problem.

Definition 4.10 The *H-Condenser* is the function $Cond^H : \mathbb{M}_{ns} \to \mathbb{M}_{ns}$ defined as follows on a machine $M \in \mathbb{M}_{ns}$. For $s \in S$ let $[s] = \{t | \exists \alpha \in A_H^* :$ t is reachable from s by $\alpha\}$. Define $Cond^H(M) = \langle S^c, s_0, next^c, obs^c, A_L\rangle$, where

- $S^c = \{[s] \mid s \in S\}$.
- $next^c : S^c \times A_L \to \mathcal{P}(S^c)$ such that $[t] \in next^c([s], a)$ if there exists $s' \in [s]$, such that $t \in next(s', a)$.
- $obs^c : S^c \to \mathcal{P}(O)$ such that $obs^c([s]) = \{o \in O | \exists s' \in [s] : obs_L(s') = o\}$.

Definition 4.11 The *H-Restrictor* is the function $Rest^H : \mathbb{M}_{ns} \to \mathbb{M}_{ns}$ such that for $M \in \mathbb{M}_{ns}$, $Rest^H(M) = \langle S^r, s_0, next^r, obs_L, A_L\rangle$, where $S^r \subseteq S$ is the set of states reachable from s_0 by actions in A_L only, and $next^r : S^r \times A_L \to S^r$ is the restriction of the '*next*' function to $S^r \times A_L$.

The systems $Cond^H(M)$ and $Rest^H(M)$ can be regarded as Moore machines with the same input set A_L and output set O. If all the values $obs^c(s)$ are singletons and we shift the outputs on states to their incoming transitions, we get two finite automata sharing the same alphabet $A_L \times O$.

Proposition 4.12 *$M \in NDI_s$ iff in $Cond^H(M)$ for all $s \in S^c$, the set $obs^c(s)$ is a singleton, and $Cond^H(M)$ and $Rest^H(M)$ are language equivalent on $A_L \times O$.*

Every state in $Cond^H(M)$ having single observation is a necessary condition by Lemma 4.9, and this is linearly checkable. The language equivalence between the two regular language is a PSPACE-complete problem [10]. Both $Cond^H(M)$ and $Rest^H(M)$ have state space at most S, so generating them can be done in polynomial time. Thus, NDI_s is in PSPACE.

This result is related to work of Focardi Gorrieri et al. [8], who studied the complexity problem of the information flow properties NNI, $SNNI$, NDC, $BNNI$ and $SBNNI$ in a process algebraic framework. For input-enabled systems, $NNI, SNNI$ and NDC are equivalent.

Proposition 4.13 *For $M \in \mathbb{M}_{ns}$, we have $M \in NDI_s$ iff $F_{sl}(M)$ is in $SNNI$.*

Focardi et al. give an exponential time subset-construction based algorithm for the property $SNNI$. Our result for NDI_s states the complexity more precisely, but also yields exponential time in practice, pending advances in complexity theory.

The security property GN_s requires arbitrary H action interleavings to be consistent with L views, and it is seemingly more complicated than NDI_s. However, the following analysis yields an in-place exhaustive solution to refute GN_s.

Define $View_L : A^* \to \mathcal{P}(O^+(AO^+)^*)$ such that an observation $\beta \in View_L(\alpha)$ if there exists a run r with $view_L(r) = \beta$ and $Act(r) = \alpha$. Intuitively $View_L(\alpha)$ is the set of L observations compatible with α.

Lemma 4.14 *$M \in \mathbb{M}_{ns}$ is in GN_s iff for all $\alpha \in A^*$, $View_L(\alpha) = View_L(\alpha|L)$.*

Proof. For the 'only if' part, suppose $M \in GN_s$. We need to show $View_L(\alpha) = View_L(\alpha|L)$. For a particular $\beta \in View_L(\alpha)$, there exists a run r with $Act(r) = \alpha$ and $view_L(r) = \beta$. From $M \in GN_s$ we can delete actions in A_H from r to get a new run r' such that $view_L(r') = \beta$. If all actions in A_H are deleted, $Act(r) = \alpha|L$, so $\beta \in View_L(\alpha|L)$. So $View_L(\alpha) \subseteq View_L(\alpha|L)$. $View_L(\alpha|L) \subseteq View_L(\alpha)$ can be proved similarly by inserting actions in A_H into a run compatible with L's observation in $View_L(\alpha|L)$. For the 'if' part, suppose $View_L(\alpha) = View_L(\alpha|L)$ for all $\alpha \in A^*$. It follows that $View_L(\alpha') = View_L(\alpha)$ for any $\alpha, \alpha' \in A^*$ with $\alpha'|L = \alpha|L$. For a particular run r with $Act(r) = \alpha$, an arbitrary insertion or deletion of actions in A_H into α generates a new action sequence α' with $\alpha'|L = \alpha|L$. From $View_L(\alpha') = View_L(\alpha)$, we have $view_L(r) \in View_L(\alpha')$, so that there exists a run r' with $Act(r') = \alpha'$ and $view_L(r') = view_L(r)$. So M in GN_s. □

From Lemma 4.14, we have $M \notin GN_s$ iff there exists some action sequence $\alpha \in A^*$ and some L observation $\beta \in O^+(AO^+)^*$, such that $\beta \in View_L(\alpha) \setminus View_L(\alpha|L)$ or $\beta \in View_L(\alpha|L) \setminus View_L(\alpha)$. This motivates the following algorithm $DecGN$, which, given a system M, nondeterministically guesses an action sequence α and a low observation β and checks consistency with α and $\alpha|L$.

$DecGN(M)$:

(i) Place a red marker and a blue marker on the initial state;

(ii) Repeat the next step $2^{2\times|S|}$ times, where $|S|$ is the number of states in M;

(iii) Nondeterministically select $a \in A$;

- If $a \in A_H$, then for every s with a blue marker, erase the old marker on s and place new blue markers on all $t \in step(s, a)$. If there exists any states s, t marked either red or blue with $obs_L(s) \neq obs_L(t)$ return $true$, else proceed.
- If $a \in A_L$, then nondeterministically choose $o \in O$. For every s with a red marker, erase the old red marker on s and place new markers on every $t \in step(s, a)$ with $obs_L(t) = o$. Then do the same on blue markers. After that, if there is only one colour remaining, return $true$, if no colour remaining, return $false$, otherwise, proceed.

(iv) Return $false$ in the end.

Intuitively, the blue markers are tracing the executions of a possible input action sequence α with respect to a particular L view, the red markers are tracing the executions of α restricted to L with respect to the same Low view. Whenever any H action changes L's local view or only one of the sets of executions can follow a step in the L view, GN_s is detected to be false. The number $2^{2\times|S|}$ covers all possible pairs of sets of marked states. Formally, we claim the following statement.

Proposition 4.15 *$M \notin GN_s$ iff there exists a computation of $DecGN(M)$ which returns true.*

Proof. We justify this by following the definition of $DecGN(M)$. Suppose an action sequence α and an L observation β are chosen, such that $\alpha|L = \beta|A_L$. Then the algorithm marks the sets $S_{blue} = \{s \in S | \exists r : Act(r) = \alpha \wedge view_L(r) = \beta \wedge s = lstate(r)\}$ and $S_{red} = \{s \in S | \exists r : Act(r) = (\alpha|L) \wedge view_L(r) = \beta \wedge s = lstate(r)\}$, where $lstate(r)$ denotes the last state in the run r.

If $M \notin GN_s$, suppose β is a minimal length L observation such that there exists an action sequence α satisfying $\alpha|L = \beta|A_L$ and β is only consistent with one of the α and $\alpha|L$. Consider a run of $DecGN$ where the sequence of actions selected is α and the sequence of L actions and observations is β. By minimality, $DecGN$ has not terminated before this point. Then there are two possibilities:

(i) $\beta \in View_L(\alpha) \setminus View_L(\alpha|L)$.

(ii) $\beta \in View_L(\alpha|L) \setminus View_L(\alpha)$.

In the first case S_{blue} is nonempty since β is compatible with α but S_{red} is empty since β is incompatible with $\alpha|L$. So the blue marker vanishes after the last action $a \in A_L$ in α is selected but red marker remains, and *true* is returned in this case. In the second case S_{red} is empty but S_{blue} is nonempty by similar reasoning. In both cases $DecGN(M)$ produces *true* within $(2^{|S|})^2$ steps, because S is finite, the state space for sets of states is $2^{|S|}$, and the space for a pair of sets is $(2^{|S|})^2$, which is the longest path to search. If $t \in next(s, a)$ with a blue marker on s, $a \in A_H$ and $obs_L(s) \neq obs_L(t)$, choosing a will lead s to state t which has a different L view from s, and we identify it by comparing this with states bearing red markers. From Lemma 4.9 we have $M \notin NDI_s$, so $M \notin GN_s$ from $GN_s \subset NDI_s$. $DecGN(M)$ returns *true* also.

If $M \in GN_s$, then from Lemma 4.14 for any possible L observation β and $\alpha \in A^*$, β is either consistent with both α and $\alpha|L$ or inconsistent with both. In both cases $DecGN(M)$ returns $false$ after $(2^{|S|})^2$ steps.

□

Since the algorithm is nondeterministic, $M \notin GN_s$ is decidable in NPSPACE. Savitch's theorem states PSPACE = NPSPACE, so deciding GN_s is in PSPACE. FC_s can be shown in NPSPACE in a similar procedure as $DecGN$ by fixing not only L's observations, but also H's observations.

5 Heuristics

We now consider some heuristic approaches which may optimize the verification of the properties we have considered, which work by reducing the problem of verifying a property on a system to a verification on an "equivalent" system.

In particular, we define a relation capturing equivalence on states with respect to L's actions, and and consider the use of this to compress the state space of the system. Define an L-bisimulation on a system M to be an equivalence relation $\sim_L \subseteq S \times S$ such that $s_1 \sim_L s_2$ iff

(i) $obs_L(s_1) = obs_L(s_2)$

(ii) for all $a \in A$ and $s'_1 \in next(s_1, a)$ there exists $s'_2 \in next(s_2, a)$ such that $s'_1 \sim_L s'_2$.

Write $[s]$ for the equivalence class of s with respect to $\sim_L$.

Let min be a function $\mathbb{M}_{ns} \to \mathbb{M}_{ns}$ such that $min(M) = \langle S^m, s_0^m, next^m, obs^m, dom, A\rangle$, where, with $\sim_L$ the maximal L-bisimulation on M,

(i) $S^m = S/\sim_L$ and $s_0^m = [s_0]$

(ii) For $[s], [t] \in S^m$, $a \in A$, $[t] \in next([s], a)$ if there exists $s' \in [s]$ and $t' \in [t]$ such that $t' \in next(s', a)$

(iii) For $[s] \in S^m$, $obs_L^m([s]) = o$ if $obs_L(s) = o$, and $obs_H^m([s]) = \bot$ for all $[s] \in S^m$, where $\bot$ is any observation on the range of obs.

Theorem 5.1 *For $M \in \mathbb{M}_{ns}$, $M \in \mathcal{P}$ iff $min(M) \in \mathcal{P}$, where $\mathcal{P}$ is any of the properties NDI_s, GN_s, FC_s and RES_s. In particular, $M \in RES_s$ iff for all $s \in S_{min(M)}$, $a \in A_H$, $next(s, a) = \{s\}$.*

This result may produce optimizations since the size of $min(M)$ may be significantly smaller than M. In general, 'bisimulation minimization' is not a viable approach for the verification of invariance properties since the partition based bisimulation usually takes more resources than it saves in the subsequent model checking [6]. However, for properties such as NDI_s, GN_s and FC_s, which seem to unavoidably take exponential time, spending polynomial time on minimization may benefit the verification significantly. Also, on deterministic systems, as described in Proposition 4.1, we reduce the NI_s problem into a safety problem which is verifiable by running a model checker on a larger state space (precisely, from $|S|$ to $|S|^2$). This extra cost may mean that a prior minimization step is beneficial.

In this case, another consideration may result in further reductions. In state based systems, a state is usually represented as an assignment to a set of variables. Let $s \in S$ be represented as a function $s : V \to U$ where V is a set of variables and U is a universe of values. Let the L observation function be represented so that $obs_L(s)$ is the restriction of s to a set $V_L \subseteq V$. For $v \in V_L$, define the function $obs_L^v(s) = s(v)$, and let M^v be the system in which obs_L is replaced by obs_L^v. Then we have the following:

Proposition 5.2 *For $M \in \mathbb{M}_s$, we have $M \in NI_s$ iff $M^v \in NI_s$ for all $v \in V_L$.*

This result suggests an approach where we apply the bisimulation minimization approach to each M^v before applying the doubling construction. Each of these systems may be significantly smaller than $min(M)$. Approximately, the bisimulation algorithm takes $\mathcal{O}(|A_L| \times |S| log_2(|S|))$ to get a partition and the further exhaustive search from Theorem 4.2 takes $\mathcal{O}(|S_r|^2 \times |A|)$ where S_r is the maximal size quotient state space obtained for the M^v. So the whole time complexity is $|V_L| \times \mathcal{O}(|A_L| \times |S| log_2(|S|) + |S_r|^2 \times |A|)$. Since $|S| \approx 2^{|V|}$ if all variables are binary, $|V_L|$ is far less than $|S|$. Since observations in M^v are based on a single variable, it seems likely that $|S_r|$ is far less than $|S|$, and $|S_r|^2 \times |A|$ may be far less than $|S|^2 \times |A|$. Also $|A_L| \times |S| log_2(|S|)$ is far less than $|S|^2 \times |A|$. So we conclude it is very likely that $|V_L| \times (|A_L| \times |S| log_2(|S|) + |S_r|^2 \times |A|)$ is far less than $|S|^2 \times |A|$. For the space complexity, we are using additional space $\mathcal{O}(|S_r|^2)$ repeatedly in the

model checking phase instead of $\mathcal{O}(|S|^2)$. Since space costs are often the critical factor in model checking, this gain may be significant.

6 Conclusion

We have considered a number of security properties on state-observed systems and have studied the complexity of the verification problems for all these properties. The unwinding characterisable properties NI_s and GN_s are tractable, based on the result of deciding bisimulation on finite states. Both symbolic and explicit state methods are applicable to these properties. The trace based properties (NDI_s, GN_s, FC_s) are PSPACE-complete. Furthermore, we have proposed some heuristics based on bisimulation minimization and argued that they may be effective. We leave the work of implementation and empirical evaluation of this claim for future work. It will be interesting, in particular, to compare the performance of BDD and explicit state model checking approaches with the compositional approaches to noninterference verification of [8], which are based on process algebraic modelling.

References

[1] Alpern, B. and F. B. Schneider, *Defining liveness*, in: *Information Processing Letters, 21(4):181–185*, 1985.

[2] Barthe, G., P. R. D'Argenio and T. Rezk, *Secure information flow by self-composition*, in: *17th IEEE Computer Security Foundation Workshop* (2004), pp. 100–114.

[3] Bevier, W. R. and W. D. Young, *A state-based approach to noninterference*, in: *Proc. 7th Computer Security Foundations Workshop*, 1994, pp. 11–21.

[4] Bossi, A., R. Focardi, C. Piazza and S. Rossi, *Bisimulation and unwinding for verifying possibilistic security properties*, in: *Proc. of Int. Conference on Verication, Model Checking, and Abstract Interpretation (VMCAI'03)*, 2003.

[5] Darvas, A., R. Hähnle and D. Sands, *A theorem proving approach to analysis of secure information flow*, in: *Workshop on Issues in the Theory of Security, (WITS'03)*, 2003.

[6] Fisler, K. and M. Y. Vardi, *Bisimulation and model checking*, in: *Conference on Correct Hardware Design and Verification Methods (CHARME'99)*, 1999, pp. 338–341.
URL citeseer.ist.psu.edu/fisler99bisimulation.html

[7] Focardi, R. and R. Gorrieri, *A classification of security properties for process algebras*, in: *Journal of Computer Security*, 1, IOS Press, 1995 pp. 5–33.

[8] Focardi, R. and R. Gorrieri, *The compositional security checker: A tool for the verification of information flow security properties*, Technical Report UBLCS-96-14, Università di Bologna (1996).

[9] Focardi, R., C. Piazza and S. Rossi, *Proofs methods for bisimulation based information flow security*, in: *Proc. of Int. Conference on Verication, Model Checking, and Abstract Interpretation (VMCAI'02)*, 2002, pp. 16–31.

[10] Garey, M. R. and D. S. Johnson, "Computers and Intractability - A Guide to the Theory of NP-Completeness," W.H. Freeman and Company, 1979.

[11] Goguen, J. and J. Meseguer, *Security policies and security models*, in: *IEEE Symp. on Security and Privacy*, 1982, pp. 11–20.

[12] Goguen, J. and J. Meseguer, *Unwinding and inference control*, in: *IEEE Symp. on Security and Privacy*, 1984.

[13] Greve, D., M. Wilding and W. van Fleet, *A separation kernel formal security policy*, in: *ACL2 Workshop*, 2003.

[14] Johnson, D. M. and F. J. Thayer, *Security and the composition of machines*, in: *Proc. IEEE Computer Security Foundations Workshop*, 1988, pp. 72–89.

[15] Kanellakis, P. C. and S. A. Smolka, *CCS expressions, finite state processes, and three problems of equivalence*, in: *Proc. 2nd Annual ACM Symposium on Principles of Distributed Computing*, New York, NY, 1983, pp. 228–240.

[16] McCullough, D., *Specifications for multi-level security and a hook-up property*, in: *Proc. IEEE Symp. on Security and Privacy*, 1987, pp. 161–166.

[17] McCullough, D., *Noninterference and the composability of security properties*, in: *Proc. IEEE Symp. on Security and Privacy*, 1988, pp. 177–186.

[18] McCullough, D., *A hookup theorem for multi-level security*, IEEE Transactions on Software Engineering **16** (1990), pp. 563–568.

[19] McLean, J., *A general theory of composition for trace sets closed under selective interleaving functions*, in: *Proc. IEEE Symp. on Security and Privacy*, 1994, pp. 79–93.

[20] Oheimb, D. v., *Information flow control revisited: Noninfluence = Noninterference + Nonleakage*, in: *Computer Security – ESORICS 2004*, LNCS **3193** (2004), pp. 225–243.

[21] Paige, R. and R. Tarjan, *Three partition refinement algorithms*, in: *SIAM Journal of Computing*, 16, 1987, pp. 973–989.

[22] Roscoe, A., *CSP and determinism in security modelling*, in: *Proc. IEEE Symp. on Security and Privacy*, 1995, pp. 114–221.

[23] Rushby, J., *Proof of separability a verification technique for a class of security kernels*, in: *Proc. 5th International Symposium on Programming, Turin, Italy*, 1982, pp. 352–367.

[24] Rushby, J., *Noninterference, transitivity, and channel-control security policies*, Technical report, SRI international (1992).
URL http://www.csl.sri.com/papers/csl-92-2/

[25] Sutherland, D., *A model of information*, in: *Proc. 9th National Computer Security Conference*, 1986, pp. 175–183.

[26] Tarski, A., *A lattice-theoretical fixpoint theorem and its applications*, in: *Pacific J. Math*, 1955 pp. 285–309.

[27] van der Meyden, R. and C. Zhang, *A comparison of semantic models for noninterference* (2006), to appear in FAST'06, available at http://www.cse.unsw.edu.au/~czhang/fast.ps.

[28] Wittbold, J. T. and D. M. Johnson, *Information flow in nondeterministic systems*, in: *Proc. IEEE Symp. on Security and Privacy*, 1990, pp. 144–161.

[29] Zakinthinos, A. and E. Lee, *A general theory of security properties*, in: *Proc. IEEE Symp. on Security and Privacy*, 1997, pp. 94–102.