Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 264 (2010) 107–123
www.elsevier.com/locate/entcs

Describing Secure Interfaces with Interface Automata[1]

Matias Lee and Pedro R. D'Argenio[2]

FaMAF, Universidad Nacional de Córdoba, Córdoba, Argentina
Email: lee@famaf.unc.edu.ar, dargenio@famaf.unc.edu.ar

Abstract

Interface automata are a model that allows for the representation of stateful interfaces. In this paper we introduce a variant of interface automata, which we call interface structure for security (ISS), that allows for the modelling of security. We focus on the property of non interference, more precisely in bisimulation-based non interference for reactive systems. We define the notion of compatible interfaces in this setting meaning that they can be composed so that a secure interface can be synthesized from the composition. In fact, we provide an algorithm that determines whether an ISS can be made secure by controlling (more specifically, pruning) some public input actions, and if so, synthesize the secure ISS. In addition, we also provide some sufficient conditions on the components ISS to ensure that their composition is secure (and hence no synthesis process is needed).

1 Introduction

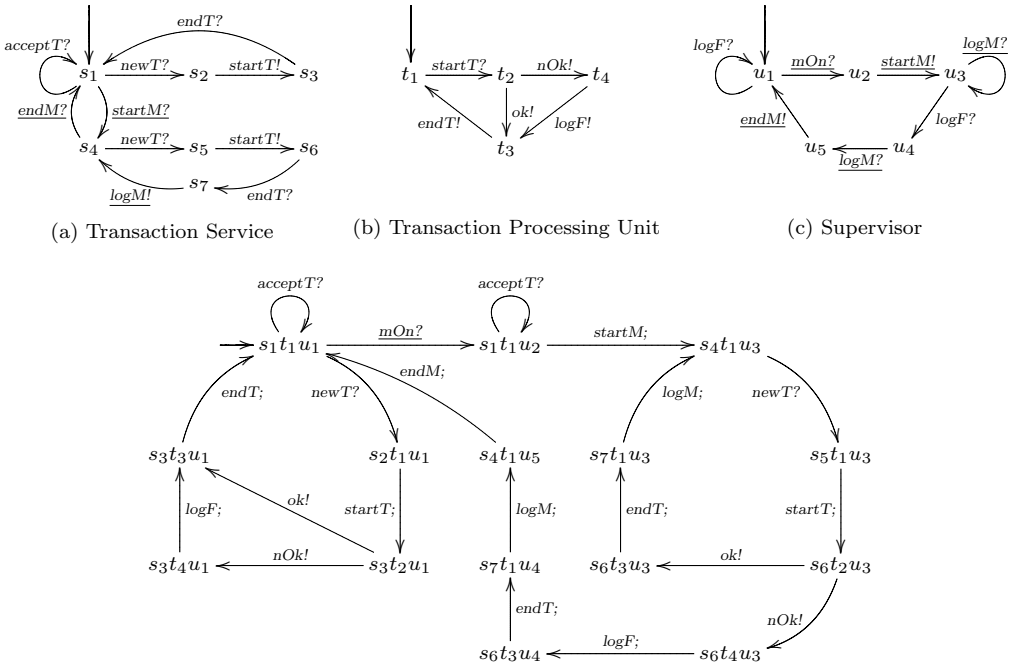
A system interface includes all methods and ways that a system uses with the aim to interact with its environment. Nowadays, it is natural to see two or more system interacting to carry out difficult or complex tasks, for example, a *web service* can use information provided by other web service to offer a more complex new service.

Good interface description should allow the analysis of the interaction between several systems. This way, we can predict if the composed system can satisfy our requirements. From this moment, we use the terms *"system"* and *"interface"* to refer to the description/abstraction of the real system interface.

According to the requirements to be enforced on the system, different tools to model interfaces have been devised. For all these definitions, two interfaces are *compatible* if, whenever they interact together, a new interface for the composition can be obtained so that it satisfies the desired requirement. *Interface automata* [6,5,7] is

[1] Supported by ANPCyT PICT 26135 and SeCyT-UNC.
[2] Also affiliated to CONICET

doi:10.1016/j.entcs.2010.07.008



(a) Transaction Service (b) Transaction Processing Unit (c) Supervisor

(d) The Composed System Interface

Figure 1. A distributed transaction processing

a light-weight formalism that captures the temporal aspects of software component interfaces. The leading requirement in this framework is that communication is properly carried out in the sense that for every output one interacting component produces, the other component is ready to receive it. In *Resource Interfaces* [3] (a variant of interface automata) the resources to complete a task are limited, then two interfaces are *not* compatible if they need more resources than the available ones. *Timed Interfaces* [4] is another variant of interface automata that allows expressing time requirements. In this setting two interfaces are compatible if there is a way to use them together so that the composition also satisfy the time requirements.

In this work we define a new variant of interface automata that we call *interface structure for security* (ISS). In this setting there are two different types of actions. One type carries *public* or *low confidential* information and the other carries *private* or *high confidential* information. For simplicity, we call them *low* and *high* actions, respectively. Following this criteria, low actions are intended to be accessed by any user while the high actions can only be accessed by those users having the appropriate permission. In this context, the desired requirement is the so-called *non-interference* property [12]. Informally, this property states that users with *no* appropriate permission cannot deduce any kind of confidential information or activity by only interacting through low actions. Therefore, we expect that compatible components, apart from communicating properly, can be composed in a way that the resulting interface is non interferent.

We illustrate this idea with an example that models a distributed transaction processing (see Fig. 1). The system has three components: a main server (Fig. 1a), a remote transaction processing unit (Fig. 1b), and a supervisor module (Fig. 1c). In its initial state the server is accepting transactions which can be known at the request of input action *acceptT?* (at this point an acknowledgement should be expected but we omit it for the sake of simplicity). As usual, outputs are suffixed by *!* and inputs by *?*. Each new transaction that arrives (*newT?*) is sent to be remotely processed (*startT!*). Then, the server waits for this processing to end (*endT?*) in which case a new transaction may be processed. When in the initial state, a supervisory system may order to start monitoring (*startM?*). At this moment transaction are processed as before, but after the remote processing ends, the result is logged (*logM!*). After this, the supervisor may end the monitoring (*endM?*). Notice that the actions involving monitoring are intended to be private. (All high actions are *underlined*.) The transaction processing unit receives transactions (*startT?*) and process them satisfactorily (*ok!*) or not (*nOk!*) in which case logs the failure (*logF!*). After this, it informs the result (*endT!*). All these actions are public. The supervisor is intended to monitor the different kind of activities. Failed transactions can always be logged (*logF?*). Besides, at an external requirement (*mOn?*), it may start (*startM!*) logging the activity of the server (*logM?*). At a possible failed transaction (*logF?* in state u_3) the activity of the server is logged (*logM?* at state u_4) and the monitoring is ended (*endM!*).

It is important that occasional monitoring of the server is not revealed to normal users (for example monitoring may be used to understand users behavior). That is why all actions involving monitoring are confidential, while actions involving transactions (which are in the interest of the user) are public. It can actually be shown that each of the components of the system satisfies the non interference property in the sense that a user that can only observe low actions cannot distinguish whether a high action has been performed or not in any execution. More specifically, they satisfy the (bisimulation based) non interference relations defined by Focardi and Gorrieri [10]. Roughly, these characterizations state that a system is non interferent whenever an instance of this system which does not perform any high action is weakly bisimilar [15] to another instance where high actions have been hidden.

Fig 1d represents the composition of the three component interfaces as defined in [7]. In the composition, synchronizing actions become hidden which is represented by a suffixed semicolon. Notoriously, this composition leads to a new interface that is not secure. We will explain precisely why after the formal concepts are introduced. By now, it is only important the question of whether a new composed interface that is non interferent can be synthesized from the composed interface automata of Fig 1d. If so, the component interfaces of Fig. 1 will be compatible.

In this article we provide an algorithm to determine whether two ISS are compatible and in this case we give an algorithm to synthesize the composed ISS satisfying non-interference. Actually, our algorithm only determines if an ISS can be made secure by controlling (more specifically, pruning) some *low* input actions, and if so, synthesize the secure ISS. We show that, though obtaining a secure ISS by control-

ling high input actions is possible in some occasion, there is no obvious pattern to determine this in general, let alone to synthesize a secure ISS.

In addition to this, we also provide some sufficient conditions on the components ISS to ensure that their composition is secure. Basically, these conditions require that the component ISS can be composed without missynchronizations and they are non-interferent under the same set of confidential actions as the composed ISS.

Related Work. To our knowledge, very little work has been developed in this same area. Cassez et al. [2] resolve a synthesis problem for trace-based non-interference (SNNI) [10]. They do not make difference between input and output action, and only confidential actions are controllable. [11] and [1] introduce synthesis algorithms for *timed* non-interference on timed automata. The setting is similar as before and in these cases, they study both trace-based and simulation-based timed non-interference. In [1], in particular they allow controllability of both public and confidential actions, but still they do not make distinction between controllable (input) and *uncontrollable* (output) actions.

Organization of the paper. Section 2 recalls interface automata. In Section 3 we define interface structures for security, the composition operation on them, and recall the bisimulation-based non interference properties of [10]. Section 4 introduces the different algorithms, and the sufficient conditions for compositionality are proved in Section 5. Section 6 concludes the paper. Omited proofs appear in [14].

2 Interface Automata

An *interface automaton* [6,5] is an automaton that models the interaction that a system component carries out with its environment. This interaction is performed by means of *input* and *output* actions. Input actions describe the behavior that the component expects (or *assumes*) from the environment. Output actions represent the behavior it communicates (or *guarantees*) to the environment. The automaton also describes the temporal ordering in which such interactions should take place. We also consider a third set of actions, the so called *hidden actions*, that are meant to represent internal non-determinism resulting from internal activity. In this section we recall the fundamentals of interface automata that we use in our paper.

Definition 2.1 An *Interface Automaton* (IA) is a tuple $S = \langle Q, q^0, A^I, A^O, A^H, \rightarrow \rangle$ where: *(i)* Q is a finite set of *states* with $q^0 \in Q$ being the *initial state*; *(ii)* A^I, A^O, and A^H are the (pairwise disjoint) finite sets of *input*, *output*, and *hidden actions*, respectively, with $A = A^I \cup A^O \cup A^H$; and *(iii)* $\rightarrow \subseteq Q \times A \times Q$ is the *transition relation* that is required to be *input deterministic* (i.e. $(q, a, q_1), (q, a, q_2) \in \delta$ implies $q_1 = q_2$ for all $a \in A^I$ and $q, q_1, q_2 \in Q$). In general, we denote Q_S, A^I_S, $\rightarrow_S$, etc. to indicate that they are the set of states, input actions, transitions, etc. of the IA S.

As usual, we denote $q \xrightarrow{a} q'$ whenever $(q, a, q') \in \rightarrow$, $q \xrightarrow{a}$ if there is q' s.t. $q \xrightarrow{a} q'$, and $q \not\xrightarrow{a}$ if this is not the case. An *execution* of S is a finite sequence $q_0\, a_0\, q_1\, a_1 \ldots q_n$ s.t. $q_i \in Q$, $a_i \in A$ and $q_i \xrightarrow{a_i} q_{i+1}$ for $0 \leq i < n$. An execution is *autonomous* if all

their actions are output or hidden (the execution does not need stimulus from the environment to run).

Parallel composition is central to the hierarchical construction of systems. Given two interface automata, their composition defines a new interface that represents the way this new composed system interacts with its environment, hiding the communication between the components. It is particularly important to impose new restrictions to the composed interface so that the composed system behaves correctly (i.e. *safely*).

Composition of two IA is only defined if their actions are disjoint except when input actions of one of the IA coincide with some of the output actions of the other. Such actions are intended to synchronize in a communication.

Definition 2.2 Let S and T be two IA, and let $\mathit{shared}(S,T) = (A_S \cap A_T)$ be the set of *shared actions*. We say that S and T are *composable* whenever $\mathit{shared}(S,T) = (A_S^I \cap A_T^O) \cup (A_S^O \cap A_T^I)$.

The product of two composable IA S and T is defined pretty much as CSP parallel composition: *(i)* the state space of the product is the product of the set of states of the components, *(ii)* shared actions can only synchronize, i.e., both component should perform a transition with the same synchronizing label (one input, and the other output), and *(iii)* transitions with non-shared actions are interleaved. Besides, shared actions are hidden in the product.

Definition 2.3 Let S and T be composable IA. The *product* $S \otimes T$ is the interface automata defined by:

- $Q_{S\otimes T} = Q_S \times Q_T$ with $q^0_{S\otimes T} = (q^0_S, q^0_T)$;
- $A^I_{S\otimes T} = A^I_S \cup A^I_T - \mathit{shared}(S,T)$, $A^O_{S\otimes T} = A^O_S \cup A^O_T - \mathit{shared}(S,T)$, and $A^H_{S\otimes T} = A^H_S \cup A^H_T \cup \mathit{shared}(S,T)$; and
- $(q_S, q_T) \xrightarrow{a}_{S\otimes T} (q'_S, q'_T)$ if any of the following holds:
  - $a \in A_S - \mathit{shared}(S,T)$, $q_S \xrightarrow{a}_S q'_S$, and $q_T = q'_T$;
  - $a \in A_T - \mathit{shared}(S,T)$, $q_T \xrightarrow{a}_S q'_T$, and $q_S = q'_S$;
  - $a \in \mathit{shared}(S,T)$, $q_S \xrightarrow{a}_S q'_S$, and $q_T \xrightarrow{a}_T q'_T$.

There may be reachable states on $S \otimes T$ for which one of the components, say S, may produce an output shared action that the other is not ready to accept (i.e., its corresponding input is not available at the current state). Then S violates the input assumption of T and this is not acceptable. States like this are called *error states*.

Definition 2.4 Let S and T be composable IA. A product state $(q_S, q_T) \in Q_{S\otimes T}$ is an *error state* if there is an action $a \in \mathit{shared}(S,T)$ s.t. either $a \in A^O_S$, $q_S \xrightarrow{a}_S$ and $q_T \not\xrightarrow{a}_T$, or $a \in A^O_T$, $q_T \xrightarrow{a}_T$ and $q_S \not\xrightarrow{a}_S$.

If the product $S \otimes T$ does not contain any reachable error state, each component satisfies the interface of the other (i.e., the input assumptions) and are thus compatible. Instead, the presence of a reachable error state is evidence that one component

is violating the interface of the other. This may not be a major problem as long as the environment is able to restrain of producing an output (an input to $S \otimes T$) that leads the product to the error state. Of course, it may be the case that $S \otimes T$ does not provide any possible input to the environment and reaches autonomously (i.e., via output or hidden actions) to an error state. In such a case we say that $S \otimes T$ is incompatible.

Definition 2.5 Let S and T be composable IA and let $S \otimes T$ be its product. A state $(q_S, q_T) \in Q_{S\otimes T}$ is an *incompatible state* if there is an error state reachable from (q_S, q_T) through an autonomous execution. If a state is not incompatible, it is *compatible*. If the initial state of $S \otimes T$ is compatible, then S and T are *compatible*.

Finally, if two IA are compatible, it is possible to define the interface for the resulting composition. Such interface is the result of pruning all input transitions of the product that lead to incompatible states i.e. states from which an error state can be autonomously reached.

Definition 2.6 Let S and T be compatible IA. The *composition* $S \parallel T$ is the IA that results from $S \otimes T$ by removing all transition $q \xrightarrow{a}_{S\otimes T} q'$ s.t. *(i)* q is a compatible state in $S \otimes T$, *(ii)* $a \in A^I_{S\otimes T}$, and *(iii)* q' is an incompatible state in $S \otimes T$.

Let Sv, TP, and SU be the interfaces of Figs. 1a, 1b, and 1c, respectively. Sv and TP are compatible and $Sv \parallel TP = Sv \otimes TP$ since $Sv \otimes TP$ does not contain error states. $Sv \parallel TP$ and SU are also compatible but, for instance, state $s_2t_1u_2$ in $(Sv \parallel TP) \otimes SU$, which is reached through $s_1t_1u_2 \xrightarrow{newT?} s_2t_1u_2$, is an incompatible state since it reaches the error state $s_3t_4u_2$ through the autonomous execution $s_2t_1u_2 \xrightarrow{startT;} s_3t_3u_2 \xrightarrow{nOk!} s_3t_4u_2$. By removing transition $s_1t_1u_2 \xrightarrow{newT?} s_2t_1u_2$ and some other, interface $Sv \parallel TP \parallel SU$ is obtained (see Fig 1d).

3 Interface Structures for Security

In this section we extend IA to cope with security. For this purpose we separate visible actions in two classes: public or *low actions*, which can be observed and used by any user, and private or *high actions*, which are intended only for users with the appropriate clearance. According to this separation, we can characterize which components provide a secure manipulation of the information. We take the so-called non interference property [13] as such characterization. We first define interface structures for security (ISS for short) and later adapt the definition of non interference from [10] to our setting.

Definition 3.1 An *Interface Structure for Security (ISS)* is a tuple $\langle S, A^h, A^l\rangle$ where S is an IA and A^h and A^l are disjoint sets of actions s.t. $A^h \cup A^l = A^O \cup A^I$.

If necessary, we will write A^h_S and A^l_S instead of A^h and A^l, respectively, and write $A^{X,m}$ instead of $A^X \cap A^m$ with $X \in \{I, O\}$ and $m \in \{h, l\}$. We will also lift nomenclature of IA to ISS whenever it holds for the underlying IA. For instance,

we say that two ISS $\langle S, A^h_S, A^l_S\rangle$ and $\langle T, A^h_T, A^l_T\rangle$ are composable whenever S and T are composable.

Since it is expected that a low-level user cannot distinguish the occurrence of high actions, we expect that a system behaves the same when high actions are not performed or when high actions are considered as hidden actions. In this setting, by "behave the same" we mean that they are (weak) bisimilar. So, we recall the definition of the restriction and hiding operations and the notion of bisimulation.

Definition 3.2 Given an IA S and a set of actions $X \subseteq A^I_S \cup A^O_S$, define:

- the *restriction* of X in S by $S\backslash X = \langle Q_S, q^0_S, A^I_S - X, A^O_S - X, A^H_S, \rightarrow_{S\backslash X}\rangle$ where $q \xrightarrow{a}_{S\backslash X} q'$ iff $q \xrightarrow{a}_S q'$ and $a \notin X$.
- the *hiding* of X in S by $S/X = \langle Q_S, q^0_S, A^I_S - X, A^O_S - X, A^H_S \cup X, \rightarrow_S\rangle$.

Given an ISS $\mathcal{S} = \langle S, A^h_S, A^l_S\rangle$ define the *restriction* of X in $\mathcal{S}$ by $\mathcal{S}\backslash X = \langle S\backslash X, A^h_S - X, A^l_S - X\rangle$ and the *hiding* of X in $\mathcal{S}$ by $\mathcal{S}/X = \langle S/X, A^h_S - X, A^l_S - X\rangle$.

We write $q_1 \stackrel{\varepsilon}{\Rightarrow} q_{n+1}$ if there are actions $a_1, \cdots, a_n \in A^H$ and states $q_2, \cdots q_n \in Q$, $n \geq 1$, s.t. $q_i \xrightarrow{a_i} q_{i+1} \in \rightarrow$ for $1 \leq i \leq n$ or $q_1 = q_{n+1}$. We write $q \stackrel{a}{\Rightarrow} q'$ if there are q_1 and q_2 s.t. $q \stackrel{\varepsilon}{\Rightarrow} q_1 \xrightarrow{a} q_2 \stackrel{\varepsilon}{\Rightarrow} q'$, and $q \stackrel{\hat{a}}{\Rightarrow} q'$ if $q \stackrel{a}{\Rightarrow} q'$ or $a \in A^H$ and $q = q'$. Finally, we write $q \stackrel{\varepsilon}{\Rightarrow}\!\!\xrightarrow{a}$ if there is q' s.t. $q \stackrel{\varepsilon}{\Rightarrow} q'$ and $q' \xrightarrow{a}$.

Definition 3.3 Let S and T be IA. A relation $R \subseteq Q_S \times Q_T$ is a *(weak) bisimulation* between S and T if $q^0_S \; R \; q^0_T$ and, for all $q_S \in Q_S$ and $q_T \in Q_T$, $q_S \; R \; q_T$ implies:

- for all $a \in A_S$ and $q'_S \in Q_S$, $q_S \xrightarrow{a}_S q'_S$ implies that there exists $q'_T \in Q_T$ s.t. $q_T \stackrel{\hat{a}}{\Rightarrow}_T q'_T$ and $q'_S \; R \; q'_T$; and
- for all $a \in A_T$ and $q'_T \in Q_T$, $q_T \xrightarrow{a}_T q'_T$ implies that there exists $q'_S \in Q_S$ s.t. $q_S \stackrel{\hat{a}}{\Rightarrow}_S q'_S$ and $q'_S \; R \; q'_T$.

We say that S and T are *bisimilar*, notation $S \approx T$, if there is a bisimulation between S and T. Moreover, we say that two ISS $\mathcal{S}$ and $\mathcal{T}$, and write $\mathcal{S} \approx \mathcal{T}$, whenever the underlying IA are bisimilar.

The following definitions of non-interference are due to Focardi and Gorrieri [10].

Definition 3.4 Let $\mathcal{S}$ be an ISS. *(i)* $\mathcal{S}$ satisfies *bisimulation-based strong non-deterministic non-interference (BSNNI)* if $\mathcal{S}\backslash A^h \approx \mathcal{S}/A^h$. *(ii)* $\mathcal{S}$ satisfies *bisimulation-based non-deterministic non-interference (BNNI)* if $\mathcal{S}\backslash A^{h,I}/A^{h,O} \approx \mathcal{S}/A^h$.

Notice the difference between the two definitions. BSNNI represents the security property as we describe so far: a system satisfies BSNNI if a low-level user cannot distinguish (up to bisimulation) by means of low level actions (the only visible ones) whether the system performs high actions (so they are hidden) or not (high actions are restricted). BNNI is an apparently weaker notion. (Actually BNNI and BSNNI are incomparable [10].) In this case only input high actions are restricted since the low-level user cannot provide this type of actions; instead output high actions are only hidden since they still can autonomously occur. We also consider this second notion as it seems appropriate for IA where only input actions are controllable.

Parallel composition extends to ISS as follows.

Definition 3.5 Given two composable ISS, $\mathcal{S} = \langle S, A_S^h, A_S^l \rangle$ and $\mathcal{T} = \langle T, A_T^h, A_T^l \rangle$, their *composition*, $\mathcal{S} \parallel \mathcal{T}$, is defined by the ISS $\langle S \parallel T, (A_S^h \cup A_T^h) - shared(S,T), (A_S^l \cup A_T^l) - shared(S,T) \rangle$.

Returning to our running example, notice that the only high action in $Sv \parallel TP \parallel SU$ is $\underline{mOn?}$. We have already mention that $Sv \parallel TP \parallel SU$ does not satisfy non interference. In fact it does not satisfies BSNNI nor BNNI. (They agree in this example since $A^{h,O} = \emptyset$). In fact, there is no state in $(Sv \parallel TP \parallel SU)\backslash\{\underline{mOn?}\}$ that is bisimilar to state $s_6 t_2 u_3$ in $(Sv \parallel TP \parallel SU)/\{\underline{mOn?}\}$. So, parallel composition does not preserve BSNNI or BNNI in general.

4 Deriving Secure ISS

As we have just seen, the composition of secure interfaces may yield a new insecure interface. The question that arises then is: Is it possible to derive a secure interface out of the resulting composed interface? Notice that, in much the same spirit, a product of IA is adapted to obtain a safe composed interface whenever possible (see Def. 2.6). In this section we present an algorithm to derive an ISS satisfying BSNNI (or BNNI) out of a given ISS whenever possible. Since the method is similar in both cases, we focus on BSNNI.

Our algorithm is based on the algorithm of Fernandez and Mounier to check bisimulation on the fly [8,9]. Roughly, our algorithm works as follows: *(i)* IA are saturated adding all weak transitions $\stackrel{\hat{a}}{\Rightarrow}$; *(ii)* a full synchronous product is constructed where transitions synchronize whenever they have the same label; *(iii)* whenever there is a mismatching transition, a new transition is added on the product leading to a special *fail* state; *(iv)* if reaching a fail state is inevitable (we later define this properly) the IA are not bisimilar; if there is always a way to avoid reaching a fail state, the IA are bisimilar.

The original algorithm stops as soon as it encounters an evidence that the IA are not bisimilar. We basically follow the same algorithm and check if $\mathcal{S}\backslash A^h \approx \mathcal{S}/A^h$, but construct the complete synchronous product regardless whether bisimulation fails. If the bisimulation check succeeds, then $\mathcal{S}$ satisfies BSNNI (see Theorem 4.5). If it does not succeed, then we provide an algorithm to decide whether $\mathcal{S}$ can be transformed into a secure ISS by pruning low input transitions. This decision mechanism categorizes non-bisimilar pair of states in three different classes: one in which the pair of states can surely be transformed into bisimilar (*May*), a second class that definitely cannot be transformed into bisimilar (*Fail*), and a third class of undetermined pairs of states (*Undet*). If the pair of the initial state of the synchronous product fall into the first class, the ISS can be transformed into a non-interferent interface by pruning low input transitions. Otherwise, it cannot.

The algorithm to synthesize the secure ISS (once it is decided that it is possible) selects low input transitions to prune, prune them, and checks whether the resulting ISS is secure. If it is not, it is certainly the case that the initial state of the product

Table 1
Transitions for the saturation of S marking B (with $\alpha \in (A_S^I \cup A_S^O \cup \overline{A_S^I} \cup \overline{A_S^O})$ and $x \in \{\varepsilon, \varepsilon'\}$)

$$q \xrightarrow{\varepsilon}_{\overline{S}_B} q \qquad q \xrightarrow{\varepsilon'}_{\overline{S}_B} q \qquad \frac{q \xrightarrow{a}_S q' \quad a \in A_S^I \cup A_S^O}{q \xrightarrow{a}_{\overline{S}_B} q'} \qquad \frac{q \xrightarrow{a}_S q' \quad a \in B}{q \xrightarrow{\varepsilon'}_{\overline{S}_B} q'} \qquad \frac{q \xrightarrow{a}_S q' \quad a \in A_S^H - B}{q \xrightarrow{\varepsilon}_{\overline{S}_B} q'}$$

$$\frac{q \xrightarrow{x}_{\overline{S}_B} q' \quad q' \xrightarrow{\alpha}_{\overline{S}_B} q''}{q \xrightarrow{\overline{\alpha}}_{\overline{S}_B} q''} \qquad \frac{q \xrightarrow{a}_{\overline{S}_B} q' \quad a \in A_S^I \cup A_S^O \quad q' \xrightarrow{x}_{\overline{S}_B} q''}{q \xrightarrow{a}_{\overline{S}_B} q''}$$

ISS is still in the *May* class and we can proceed by pruning again in the previously pruned ISS. This process is shown to terminate (see Theorem 4.10).

We first adapt Fernandez and Mounier's algorithm to check bisimulation, then provide the algorithm to decide whether the ISS can be transformed into a secure one, and finally present the synthesis algorithm.

4.1 Checking bisimulation-based non-interference

A natural way to check weak bisimulation is to *saturate* the transition system, i.e., to add a new transition $q \xrightarrow{a} q'$ to the model for each weak transition $q \overset{a}{\Rightarrow} q'$, and then check strong bisimulation on the saturated transition system. This is the starting point of Fernandez and Mounier's algorithm.

In our case, we will add some extra decoration to the actions on the saturated ISS in order to identify better the different categories of states. These decorations are nonetheless ignored to check bisimilarity. First, we map all hidden actions in A^H to ε and ε' (we call this two actions *silent steps*). This difference will eventually be so that the abstracted high input actions (to calculate BNNI or BSNNI) are mapped into ε' while all other hidden actions are mapped into ε which will be crucial to calculate the *Undet* class. Besides, we also mark some actions with $\bar{\cdot}$ to indicate that a transition $q \xrightarrow{\overline{a}} q$ is derived by saturating with one or more ε or ε' in front of it (and maybe some behind it). That is, if $q \xrightarrow{\overline{a}}$ is valid in the saturation of the ISS $\mathcal{S}$, then $q \overset{\varepsilon}{\Rightarrow} \xrightarrow{a}$ is valid in $\mathcal{S}$. Notice, hence, that $q \xrightarrow{a}$ is valid in the saturation of the ISS $\mathcal{S}$ if and only if $q \xrightarrow{a}$ is valid in $\mathcal{S}$. We consider that $\bar{\cdot}$ is idempotent ($\overline{\overline{a}} = \overline{a}$) and for all action $a \in A$, it creates a new action $\overline{a}$. We say that $\overline{a}$ *is marked* and a *is not marked* otherwise, i.e., if $a \neq \overline{a}$. For a set X of actions, let $\overline{B} = \{\overline{a} \mid a \in B\}$.

Definition 4.1 Let S be an IA and $B \subseteq A_S^H$. The *saturation of S marking B* is the IA $\overline{S}_B = \langle Q_S, q_S^0, A_S^I \cup \overline{A_S^I}, A_S^O \cup \overline{A_S^O}, \{\varepsilon, \varepsilon'\}, \rightarrow_{\overline{S}_B} \rangle$ where $\rightarrow_{\overline{S}_B}$ is the smallest relation satisfying rules in Table 1. Given an ISS $\mathcal{S}$, its *saturation marking B*, $\overline{\mathcal{S}}_B$, is the ISS obtained by saturating the underlying IA.

The set B in previous definition is the set of hidden actions that we want to distinguish as ε'. This can be observed in the two rightmost upper rules in Table 1. Notice that the leftmost lower rule marks the action when it is preceded by an execution of a hidden action. Besides we do not consider saturation of silent steps with the exception of the self loop transitions (the two leftmost upper rules) since it is not needed. Apart from this exceptions, saturation works as usual. We remark

Table 2
Transitions for the synchronous product ($a \in A^I \cup A^O$)

$$\frac{q_S \xrightarrow{a}_S q'_S \quad q_T \xrightarrow{a}_T q'_T}{(q_S,q_T) \xrightarrow{a}_{S\times T} (q'_S,q'_T)} \qquad \frac{q_S \xrightarrow{a}_S q'_S \quad q_T \xrightarrow{\overline{a}}_T q'_T}{(q_S,q_T) \xrightarrow{\overline{a}}_{S\times T} (q'_S,q'_T)} \qquad \frac{q_S \xrightarrow{\overline{a}}_S q'_S \quad q_T \xrightarrow{a}_T q'_T}{(q_S,q_T) \xrightarrow{\overline{a}}_{S\times T} (q'_S,q'_T)}$$

$$\frac{q_S \xrightarrow{\varepsilon'}_S q'_S \quad q_T \xrightarrow{\varepsilon}_T q'_T}{(q_S,q_T) \xrightarrow{\varepsilon'}_{S\times T} (q'_S,q'_T)} \qquad \frac{q_S \xrightarrow{\varepsilon}_S q'_S \quad q_T \xrightarrow{\varepsilon'}_T q'_T}{(q_S,q_T) \xrightarrow{\varepsilon'}_{S\times T} (q'_S,q'_T)} \qquad \frac{q_S \xrightarrow{a}_S \quad q_T \not\xrightarrow{\overline{a}}_T}{(q_S,q_T) \xrightarrow{a}_{S\times T} \mathit{fail}} \qquad \frac{q_S \not\xrightarrow{\overline{a}}_S \quad q_T \xrightarrow{a}_T}{(q_S,q_T) \xrightarrow{a}_{S\times T} \mathit{fail}}$$

that $(q \xrightarrow{\overline{a}}_{\overline{\mathcal{S}}_B} q'$ iff $q \overset{a}{\Rightarrow}_S q')$ and $(q \xrightarrow{a}_{\overline{\mathcal{S}}_B} q'$ iff $q \overset{a}{\longrightarrow\!\!\!\!\Rightarrow}_S q')$

The second part of the algorithm is to construct a *synchronous product* attending the missynchronization which leads to a distinguished state *fail*. Moreover, we should also take into account the different markings (namely ε' and $\bar{\cdot}$).

Definition 4.2 Let S and T be two saturated IA with $A^X_S = A^X_T = A^X$ for $X \in \{I, O\}$ and $A^H_S = A^H_T = \{\varepsilon, \varepsilon'\}$. The *synchronous product* of S and T is the IA $S \times T = \langle (Q_S \times Q_T) \cup \{\mathit{fail}\}, (q^0_S, q^0_T), A^I, A^O, \{\varepsilon, \varepsilon'\}, \rightarrow_{S\times T}\rangle$ where $\rightarrow_{S\times T}$ is the smallest relation satisfying rules in Table 2. Given $\mathcal{S} = \langle S, A^h_{\mathcal{S}}, A^l_{\mathcal{S}}\rangle$ and $\mathcal{T} = \langle T, A^h_{\mathcal{T}}, A^l_{\mathcal{T}}\rangle$ with S and T satisfying conditions above and $A^m_{\mathcal{S}} = A^m_{\mathcal{T}} = A^m$ for $m \in \{l, h\}$, then the *synchronous product* of $\mathcal{S}$ and $\mathcal{T}$ is defined by the ISS $\mathcal{S} \times \mathcal{T} = \langle S \times T, A^h, A^l\rangle$.

With the exception of the two rightmost lower rules in Table 2, all other rules state that the two states can perform some equally labeled action, possibly after many silent steps (recall that the interfaces are saturated). The last two rules say that if an interface can do a transition, and this transition cannot be simulated by the other one, even by using silent steps, then the system makes a transition to a special state called *fail*.

Notice that, unlike the product of Def. 2.3, the synchronous product synchronizes transition with the same name *and* the same type (input, output, or silent step). Moreover, transitions *do not* interleave. In fact, if one transition does not have a matching in the other component, the synchronization leads to a distinguished *fail* state. Since saturated IA have silent steps self-loops in all states, there is never a silent step leading to *fail*.

A state of the synchronous product that has a transition to *fail* represents a pair of states that are not bisimilar. This is evident since such transition is defined whenever a state has a transition that the other cannot simulate. The last step of Fernandez and Mounier's algorithm is to propagate this failure backwards appropriately to identify non bisimilar pairs of states.

Definition 4.3 Let $S \times T$ be a synchronous product. We define the set $\mathit{NoPass} \subseteq Q_{S\times T}$ as $\mathit{NoPass} = \cup_{i=0}^{\infty} \mathit{NoPass}^i$ where NoPass^i is defined in Table 3. If $q \in \mathit{NoPass}$, we say that the pair q *does not pass the bisimulation test*. Besides, we define the set $\mathit{Pass} \subseteq Q_{S\times T}$ as $\mathit{Pass} = \mathit{NoPass}^c$. If $q \in \mathit{Pass}$, we say that the pair q *pass the bisimulation test*. If the initial state of an IA underlying an ISS $\mathcal{S} \times \mathcal{T}$ passes (does not pass) the bisimulation test, we say that $\mathcal{S} \times \mathcal{T}$ passes (does not pass) the bisimulation test.

Table 3
Auxiliary definitions for *NoPass* ($q \xrightarrow{a,\overline{a}} q'$ is a shorthand for "$q \xrightarrow{a} q'$ or $q \xrightarrow{\overline{a}} q'$".)

$$NoPass^0 = \{(q_S, q_T) : (q_S, q_T) \xrightarrow{a}_{S\times T} fail, a \in A\} \cup \{fail\}$$

$$NoPass^{k+1} = NoPass^k \cup \bigcup_{q \xrightarrow{a} q' \in (\rightarrow_S \cup \rightarrow_T)} NoPass^{k+1}_{q \xrightarrow{a} q'}$$

$$NoPass^{k+1}_{q_S \xrightarrow{a} q'_S} = \{(q_S, q_T) : \forall q'_T : (q_S, q_T) \xrightarrow{a,\overline{a}} (q'_S, q'_T) \Rightarrow (q'_S, q'_T) \in NoPass^k\}$$

$$NoPass^{k+1}_{q_T \xrightarrow{a} q'_T} = \{(q_S, q_T) : \forall q'_S : (q_S, q_T) \xrightarrow{a,\overline{a}} (q'_S, q'_T) \Rightarrow (q'_S, q'_T) \in NoPass^k\}$$

The proof of the following lemma is a variation of Fernandez and Mounier's taking into account our modifications. For this reason we omit it.

Lemma 4.4 *Let $\mathcal{S}$ and $\mathcal{T}$ be two ISS. Let B and B' be any two subsets of the hidden actions of $\mathcal{S}$ and $\mathcal{T}$ respectively. Then $\overline{\mathcal{S}}_B \times \overline{\mathcal{T}}_{B'}$ passes the bisimulation test if and only if $S \approx T$.*

As an immediate corollary, there is a decision algorithm to check whether an ISS satisfies BSNNI (or BNNI). We state it in the following theorem.

Theorem 4.5 *Let $\mathcal{S} = \langle S, A^h, A^l \rangle$ be an ISS.*

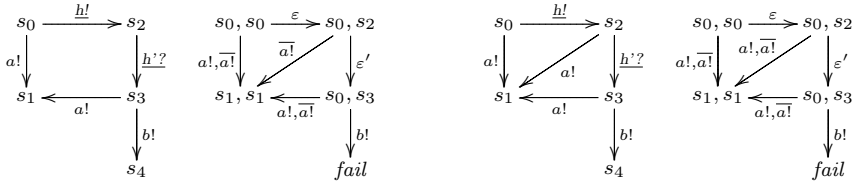
(i) *$\mathcal{S}$ satisfies BSNNI iff $\overline{(\mathcal{S}\backslash A^h)}_\emptyset \times \overline{(\mathcal{S}/A^h)}_{A^{h,I}}$ passes the bisimulation test.*

(ii) *$\mathcal{S}$ satisfies BNNI iff $\overline{((S\backslash A^{h,I})/A^{h,O})}_\emptyset \times \overline{(S/A^h)}_{A^{h,I}}$ passes the bisimulation test.*

4.2 Synthesizing Secure ISS

Given an ISS $\mathcal{S}$, suppose its synchronous product $P_\mathcal{S} = \overline{(\mathcal{S}\backslash A^h)}_\emptyset \times \overline{(\mathcal{S}/A^h)}_{A^{h,I}_S}$ does not pass the simulation test. Notice that the behavior of an interface very much depends on the stimulus that it receives from its environment. Then one could ask if there is a set of stimulus (namely, inputs) such that, if they are forbidden, a secure interface is obtained. This idea repeats the spirit of the idea in the definition of the composition where we prune the set of input transitions that leads to the miscommunication (which is the "undesirable situation" in this case). In this section we provide a first method to find out whether an ISS $\mathcal{S}$ that is not secure (i.e., $P_\mathcal{S}$ did not pass the bisimulation test) can be modified by pruning input transitions so that the result is. Then we use this method to define a new one that, provided $\mathcal{S}$ can be modified, so it synthesizes the resulting secure ISS by successively pruning input transitions.

In the previous section we defined the set *NoPass* containing states of the synchronous product $P_\mathcal{S}$. This set contains pair of states that are not bisimilar. This set can in turn be devided in three disjoint subsets. The first subset (*May*) contains pairs of states of $P_\mathcal{S}$ such that if some low input transitions of $\mathcal{S}$ are pruned this pairs become bisimilar in the new product. Consider, for instance, that the product contains the only transition $(q_S, q_T) \xrightarrow{a?} fail$ where a is low, and suppose w.l.o.g., that $q_T \xrightarrow{a?}\!\!\!\!\!\!/$. By removing $q_S \xrightarrow{a?}$ the pair (q_S, q_T) would become bisimilar.



(a) $\mathcal{S}_1$ and the synchronous product $P_{\mathcal{S}_1}$ (b) $\mathcal{S}_2$ and the synchronous product $P_{\mathcal{S}_2}$

Figure 2.

In this case we say that (q_S, q_T) *may pass* the bisimulation test. The second set (*Fail*) contains pairs of states that cannot be turned into bisimilar by pruning low input transitions. Consider the same example as before but now with an output: $(q_S, q_T) \xrightarrow{a!} fail$. Since the progress from (q_S, q_T) is autonomous, there is no way to control the execution of $a!$ and hence (q_S, q_T) will remain non-bisimilar. We say then that (q_S, q_T) *fails* the bisimulation test. More generally, *Fail* contains those pairs that have an autonomous sequence to a *fail* state. The last set (*Undet*) contains undetermined pair of states in the sense that they may become bisimilar if a high input transitions is removed.

The way to remove high input transitions is far from obvious. Consider example in Fig. 2a. On the left-hand side of Fig. 2a we depict the ISS $\mathcal{S}_1$, on the right-hand side, the synchronous product $P_{\mathcal{S}_1}$ (self ε and ε' loops are omitted). $\mathcal{S}_1$ does not satisfy BSNNI. It is easy to verify that $s_0 s_0 \in NoPass$. To avoid the backward propagation of the failure induced by the missynchronization of transition $s_3 \xrightarrow{b} s_4$, the only point of control is high input $\underline{h'?}$ in transition $s_2 \xrightarrow{\underline{h'?}} s_3$. However, removing this transition in $\mathcal{S}_1$ has no impact at all as the new reduced ISS (having only states s_0, s_1, s_2 and the remaining transitions that connect them) does not satisfy BSNNI either. Of course, it is also possible that removing a high input transition does change the model to a secure one. This is precisely the case of the example in Fig. 2b which presents a slight variation of $\mathcal{S}_1$. If transition $s_2 \xrightarrow{\underline{h'?}} s_3$ is removed in $\mathcal{S}_2$, the reduced model satisfies BSNNI.

The difficulty on eliminating high input transitions lies on the fact that they are only present in one of the ISS of the synchronous product $P_{\mathcal{S}}$, namely, in $\overline{(\mathcal{S}/A^h)}_{A_S^{h,I}}$. This is not the case of a low input transition since the candidate to be removed come from an effective synchronization between $\overline{(\mathcal{S}\backslash A^h)}_{\emptyset}$ and $\overline{(\mathcal{S}/A^h)}_{A_S^{h,I}}$. This can be better seen in the formal definition of these sets.

Definition 4.6 Let $S \times T$ be a synchronous product. We define the sets $Fail, Undet, May \subseteq Q_{S\times T}$ respectively by:

- $Fail = \cup_{i=0}^{\infty} Fail^i$ where $Fail^i$ is defined in the Table 5. If $q \in Fail$, we say that the pair q *fail the bisimulation test.*
- $Undet = \cup_{i=0}^{\infty} Undet^i$ where $Undet^i$ is defined in the Table 6. If $q \in Undet$, we say that the pair q is an *undetermined state w.r.t the bisimulation test.*
- $May = \cup_{i=0}^{\infty} May^i$ where May^i is defined in the Table 7. If $q \in May$, we say that the pair q *may pass the bisimulation test.*

Table 4
Ω is auxiliary to the definitions of $Fail^{k+1}$, $Undet^{k+1}$ and May^{k+1}

$$\Omega(X,\Lambda,\Omega') = \Omega' \cup \bigcup_{q \xrightarrow{a} q' \in (\rightarrow_S \cup \rightarrow_T)} \Omega_{q \xrightarrow{a} q'}(X,\Lambda,\Omega')$$

$$\Omega_{q_S \xrightarrow{a} q'_S}(X,\Lambda,\Omega') = \{(q_S,q_T) \notin X : a \in \Lambda, q_S \xrightarrow{a} q'_S, (\forall q'_T : (q_S,q_T) \xrightarrow{a,\overline{a}} (q'_S,q'_T) \Rightarrow (q'_S,q'_T) \in X \cup \Omega')\}$$

$$\Omega_{q_T \xrightarrow{a} q'_T}(X,\Lambda,\Omega') = \{(q_S,q_T) \notin X : a \in \Lambda, q_T \xrightarrow{a} q'_T, (\forall q'_S : (q_S,q_T) \xrightarrow{a,\overline{a}} (q'_S,q'_T) \Rightarrow (q'_S,q'_T) \in X \cup \Omega')\}$$

Table 5
Auxiliary definitions for *Fail*

$$Fail^0 = \{(q_S,q_T) : (q_S,q_T) \xrightarrow{a}_{S\times T} fail, a \notin A^I \cup \{\varepsilon'\}\} \cup \{fail\} \qquad Fail^{k+1} = \Omega(\emptyset, A^O \cup \{\varepsilon\}, Fail^k)$$

Table 6
Auxiliary definitions for *Undet*

$$Undet^{\,0} = \bigcup_{q \xrightarrow{a} q' \in (\rightarrow_S \cup \rightarrow_T)} Undet^{\,0}_{q \xrightarrow{a} q'} \qquad Undet^{k+1} = \Omega(Fail, A^O \cup \{\varepsilon,\varepsilon'\}, Undet^k)$$

$$Undet^{\,0}_{q_S \xrightarrow{\varepsilon'} q'_S} = \{(q_S,q_T) \notin Fail : q_S \xrightarrow{\varepsilon'} q'_S, (\forall q'_T : (q_S,q_T) \xrightarrow{\varepsilon'} (q'_S,q'_T) \Rightarrow (q'_S,q'_T) \in Fail)\}$$

$$Undet^{\,0}_{q_T \xrightarrow{\varepsilon'} q'_T} = \{(q_S,q_T) \notin Fail : q_T \xrightarrow{\varepsilon'} q'_T, (\forall q'_S : (q_S,q_T) \xrightarrow{\varepsilon'} (q'_S,q'_T) \Rightarrow (q'_S,q'_T) \in Fail)\}$$

Table 7
Auxiliary definitions for *May*

$$May^0 = \bigcup_{q \xrightarrow{a} q' \in (\rightarrow_S \cup \rightarrow_T)} May^0_{q \xrightarrow{a} q'} \qquad May^{k+1} = \Omega(Fail \cup Undet, A, May^k)$$

$$May^0_{q_S \xrightarrow{a} q'_S} = \{(q_S,q_T) \notin Fail \cup Undet : q_S \xrightarrow{a} q'_S, (q_S,q_T) \xrightarrow{a}_{S\times T} fail\}$$

$$May^0_{q_T \xrightarrow{a} q'_T} = \{(q_S,q_T) \notin Fail \cup Undet : q_T \xrightarrow{a} q'_T, (q_S,q_T) \xrightarrow{a}_{S\times T} fail\}$$

In general, if the initial state of the underlying IA of an ISS $\mathcal{S} \times \mathcal{T}$ may pass (fail, is undetermined w.r.t.) the bisimulation test, we say that $\mathcal{S} \times \mathcal{T}$ may pass (fail, is undetermined w.r.t.) the bisimulation test.

Sets *Fail*, *Undet*, and *May* form a partition of *NoPass*. The fact that they are disjoint is immediate from their definitions. The inclusion $Fail \cup Undet \cup May \subseteq NoPass$ can be proved by routine induction.

Lemma 4.7 *For any synchronous product $P_{\mathcal{S}}$, $Fail \cup Undet \cup May = NoPass$*

Therefore, all pairs in $Fail \cup Undet \cup May$ are not bisimilar. In the following we show that an ISS $\mathcal{S}$ that is not non-interferent, but whose synchronous product $P_{\mathcal{S}}$ may pass the bisimulation test, can be controlled (in the sense that some low input transitions will be forbidden) so that it results in a new reduced ISS $\mathcal{S}$ that is non-interferent.

First, we need to select which are the candidate input actions to be removed on

Table 8
Set of eliminable candidates.

$$EC(\mathcal{S}) = \{q \xrightarrow{a} q' : (\exists \hat{q} : (q,\hat{q}) \in May^0_{q \xrightarrow{a} q'} \vee (\hat{q},q) \in May^0_{q \xrightarrow{a} q'})\} \cup$$
$$\{q \xrightarrow{a} q' : (\exists \hat{q} : (q,\hat{q}) \in May^1_{q \xrightarrow{a} q'}, (\forall \hat{q}' : (q,\hat{q}) \xrightarrow{a,\overline{a}} (q',\hat{q}') \Rightarrow (q',\hat{q}') \in Fail \cup Undet)\} \cup$$
$$\{q \xrightarrow{a} q' : (\exists \hat{q} : (\hat{q},q) \in May^1_{q \xrightarrow{a} q'}, (\forall \hat{q}' : (\hat{q},q) \xrightarrow{a,\overline{a}} (\hat{q}',q') \Rightarrow (\hat{q}',q') \in Fail \cup Undet)\}$$

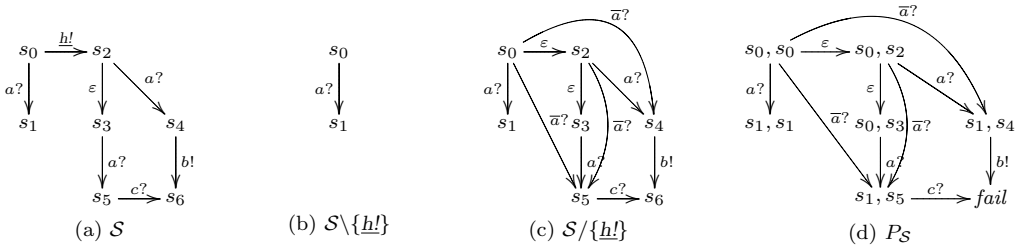


Figure 3.

these kind of ISS so that they becomes secure. So, if $\mathcal{S}$ is an ISS such that $P_{\mathcal{S}}$ may pass the bisimulation test, the set $EC(\mathcal{S}) \subseteq \rightarrow \cap\, Q \times A^{I,I} \times Q$, defined in Table 8, is the set of *eliminable candidates*.

Notice that all transitions in $EC(\mathcal{S})$ are indeed labeled with *low input actions*. Moreover, all these transitions are involved in a synchronization that connects a source pair that may pass the bisimulation test and a target pair that fails the bisimulation test. This can happen in three different manners. The first case is the basic case, in which one of the components of the pair can perform the transition but not the other. The other two are symmetric and consider the case in which both sides can perform an equally labeled transition but end up in a pair that fails or is undetermined.

As an example consider the ISS $\mathcal{S}$ in Fig. 3a. It only contains one private action, namely $\underline{h!}$. Fig. 3b, 3c, and 3d, represents the different stages until the construction of the synchronous product $P_{\mathcal{S}}$. It is easy to see that $\mathcal{S}$ is not secure: $\mathcal{S}/\{\underline{h!}\}$ can perform actions $b!$ or $c?$ after $a?$ and some silents steps, while $\mathcal{S}\backslash\{\underline{h!}\}$ cannot. Applying rules in Tables 3 to 7, we obtain that $Pass = \{(s_1,s_1)\}$, $Fail = \{(s_1,s_4), fail\}$ and May contains all remaining states in $P_{\mathcal{S}}$. It is also possible to calculate that $EC(\mathcal{S}) = \{s_5 \xrightarrow{c?} s_6, s_2 \xrightarrow{a?} s_4\}$. Notice that, if both transitions are removed, the resulting ISS is indeed non-interferent (in fact, it suffices to remove only $s_2 \xrightarrow{a?} s_4$).

An important result is that no new fail or undetermined pair of states is introduced by removing eliminable candidates. Moreover, if a pair of states is fail or undetermined in the synchronous product of the original ISS and it is also present in the synchronous product of the reduced ISS, then it is also fail or undetermined in this ISS. By the counter positive, this ensures that a synchronous product that passes or may pass the bisimulation test, will not fail or be undetermined after pruning. In a sense, Lemma 4.8 below states that the sets $Fail \cup Undet$ and $Pass \cup May$

remain invariant.

Lemma 4.8 *Let $\mathcal{S}$ be an ISS s.t. $P_{\mathcal{S}}$ may pass the bisimulation test. Let $\mathcal{S}'$ be an ISS obtained by removing one transition in $EC(\mathcal{S})$ from $\mathcal{S}$ (i.e. $\rightarrow_{\mathcal{S}'} = \rightarrow_{\mathcal{S}} - \{q \xrightarrow{a} q'\}$, provided $q \xrightarrow{a} q' \in EC(\mathcal{S})$, and unreachable states are removed form $\mathcal{S}'$). Then:* (i) $Fail_{P_{\mathcal{S}'}} \cup Undet_{P_{\mathcal{S}'}} = (Fail_{P_{\mathcal{S}}} \cup Undet_{P_{\mathcal{S}}}) \cap Q_{P_{\mathcal{S}'}}$; *and* (ii) $(Pass_{P_{\mathcal{S}}} \cup May_{P_{\mathcal{S}}}) \cap Q_{P_{\mathcal{S}'}} = Pass_{P_{\mathcal{S}'}} \cup May_{P_{\mathcal{S}'}}$. *(Subindices in $Fail_{P_{\mathcal{S}}}$, $Undet_{P_{\mathcal{S}}}$, etc. indicate that these sets were obtained from the synchronous product $P_{\mathcal{S}}$.)*

In addition, we have that all eliminable candidate transition in the original ISS that have not been removed somehow (either by explicit elimination or because they are not reachable anymore), are also eliminable candidates of the reduced ISS. This is stated in the following theorem.

Lemma 4.9 *Let $\mathcal{S}$ be an ISS s.t. $P_{\mathcal{S}}$ may pass the bisimulation test. Let $\mathcal{S}'$ be the ISS obtained by removing one transition in $EC(\mathcal{S})$ from $\mathcal{S}$. Then $EC(\mathcal{S}) \cap \rightarrow_{\mathcal{S}'} \subseteq EC(\mathcal{S}')$.*

The next theorem states the main result of this section. (As we mention in the introduction of this section, similar proofs will lead to a similar algorithm to synthesize interfaces satisfying BNNI whenever possible.)

Theorem 4.10 *Let $\mathcal{S}$ be an ISS s.t. $P_{\mathcal{S}}$ may pass the bisimulation test. Then there exists a set $\rightarrow_{\chi}$ of low input transitions such that, if $\mathcal{S}'$ is the ISS obtained from $\mathcal{S}$ by removing all transitions in $\rightarrow_{\chi}$, $\mathcal{S}'$ satisfies BSNNI.*

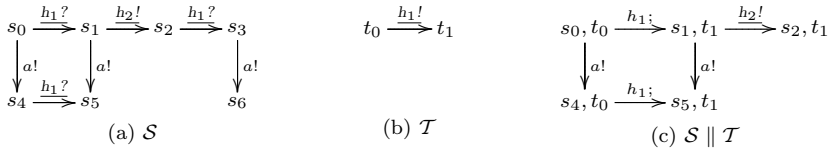
Proof Let $\mathcal{S}^0 = \mathcal{S}$. For all k such that $P_{\mathcal{S}^k}$ does not pass the bisimulation test, $\mathcal{S}^{k+1}$ be the ISS obtained by removing one transition $t_k \in EC(\mathcal{S}^k)$ from $\mathcal{S}^k$.

If this algorithm terminates, say at iteration K, then $\mathcal{S}^K$ satisfies BSNNI (by Theorem 4.5 and Lemma 4.7) and $\rightarrow_{\chi} = \{t_k \in EC(\mathcal{S}^k) \mid 0 \leq k < K\}$.

Suppose this algorithm does not terminate. Since $\mathcal{S}$ is finite, there must be a K' such that $EC(\mathcal{S}^{K'}) = \emptyset$ and $P_{\mathcal{S}^{K'}}$ may pass the bisimulation test. (This is guaranteed by Lemma 4.8.ii.) Therefore $May_{P_{\mathcal{S}^{K'}}} \neq \emptyset$ and hence $May^0_{P_{\mathcal{S}^{K'}}} \cup May^1_{P_{\mathcal{S}^{K'}}} \neq \emptyset$. If $May^0_{P_{\mathcal{S}^{K'}}} \neq \emptyset$, it is immediate that $EC(\mathcal{S}^{K'}) \neq \emptyset$ which is a contradiction. So, suppose $May^0_{P_{\mathcal{S}^{K'}}} = \emptyset$ and $May^1_{P_{\mathcal{S}^{K'}}} \neq \emptyset$. Then, there must be a transition $q \xrightarrow{a} q'$ and a state $\hat{q}$ such that either $(q, \hat{q})$ or $(\hat{q}, q)$ is in $May^1_{q \xrightarrow{a} q', P_{\mathcal{S}^{K'}}}$. W.l.o.g. suppose it is $(q, \hat{q})$. Then $(q, \hat{q}) \notin Fail_{P_{\mathcal{S}^{K'}}} \cup Undet_{P_{\mathcal{S}^{K'}}}$ and $\forall \hat{q}' : (q, \hat{q}) \xrightarrow{a, \overline{a}} (q', \hat{q}') : (q', \hat{q}') \notin Fail_{P_{\mathcal{S}^{K'}}} \cup Undet_{P_{\mathcal{S}^{K'}}} \cup May^0_{P_{\mathcal{S}^{K'}}} = Fail_{P_{\mathcal{S}^{K'}}} \cup Undet_{P_{\mathcal{S}^{K'}}}$, since $May^0_{P_{\mathcal{S}^{K'}}} = \emptyset$. But then, necessarily $a \in A^{l,I}$ (otherwise $(q', \hat{q}') \in Fail_{P_{\mathcal{S}^{K'}}} \cup Undet_{P_{\mathcal{S}^{K'}}}$). Hence $q \xrightarrow{a} q' \in EC(\mathcal{S}^{K'})$ which, again is a contradiction. □

5 Compositional construction of secure interfaces

It is known from [10] that neither BNNI nor BSNNI are preserved by CSP-like parallel composition. This also extends to the composition of interface automata.



(a) $\mathcal{S}$ (b) $\mathcal{T}$ (c) $\mathcal{S} \parallel \mathcal{T}$

Figure 4.

An example is reported in Fig. 4. In this brief section we report a small but yet important result: we give sufficient conditions to ensure that the composition of ISS results in a non-interferent ISS. Basically, these conditions require that (i) the component ISS are *fully compatible*, i.e. no error state is reached in the composition (in any way, not only autonomously), and (ii) they satisfy the non-interference property under the same set of confidential actions as the composed ISS. This is stated in the following theorem.

Theorem 5.1 *Let* $\mathcal{S} = \langle S, A_S^h, A_S^l \rangle$ *and* $\mathcal{T} = \langle T, A_T^h, A_T^l \rangle$ *be two composable ISS. Let* $\mathcal{S}' = \langle S, A_S^h - \mathit{shared}(S,T), A_S^l \cup \mathit{shared}(S,T) \rangle$ *and* $\mathcal{T}' = \langle T, A_T^h - \mathit{shared}(S,T), A_T^l \cup \mathit{shared}(S,T) \rangle$*. If* $\mathcal{S} \otimes \mathcal{T}$ *has no reachable error states and* $\mathcal{S}'$ *and* $\mathcal{T}'$ *satisfy BSNNI (resp. BNNI) then* $\mathcal{S} \parallel \mathcal{T}$ *satisfies BSNNI (resp. BNNI).*

In particular, notice that the $\mathcal{S} \otimes \mathcal{T}$ in Fig 4 has no reachable error states (and hence it agrees with $\mathcal{S} \parallel \mathcal{T}$) but $\mathcal{S}$ does not satisfy BSNNI nor BNNI if only $\mathit{shared}(S,T) = \{\underline{h_1}\}$ is consider confidential (hence being out of the hypothesis of the theorem).

6 Concluding remarks

We have presented a framework to manipulate stateful interfaces for security and their composition. We also provided a synthesis algorithm that decides whether an ISS can be turned into a secure ISS (namely, satisfying BSNNI or BNNI) by controlling low input transitions. In particular, this can be seen in the example of the introduction (Fig. 1d) where the distributed transaction processing can be turned into secure by eliminating action *acceptT?*. (This can be derived with the algorithm of Theorem 4.10.) However, as we saw extending controllability to all input actions (including confidential actions) is much more difficult. This is something we are considering in our future work.

We also have shown that in certain non trivial cases, compositional construction of secure interfaces is possible. Yet, the requirements are still a little too restricting since ISS are suppose to be composed so that no error state (i.e. miscommunication) is reached. This will be always possible in input enabled interfaces, but the setting of interface automata is intended to drop this condition. A further study to relax this conditions is also worthwhile.

Finally, we are currently busy extending the idea of refinement of interface automata to ISS and studying its relation to BSNNI and BNNI.

Acknowledgements: We thank the anonymous referees for their valuable remarks.

References

[1] Benattar, G., F. Cassez, D. Lime and O. H. Roux, *Synthesis of non-interferent timed systems*, in: J. Ouaknine and F. W. Vaandrager, editors, *FORMATS*, Lecture Notes in Computer Science **5813** (2009), pp. 28–42.

[2] Cassez, F., J. Mullins and O. H. Roux, *Synthesis of non-interferent systems*, in: *4th Int. Conf. on Mathematical Methods, Models and Architectures for Computer Network Security (MMM-ACNS'07)*, Communications in Computer and Inform. Science **1** (2007), pp. 307–321.

[3] Chakrabarti, A., L. de Alfaro, T. Henzinger and M. Stoelinga, *Resource interfaces*, in: *Proceedings of the Third International Conference on Embedded Software (EMSOFT)*, LNCS, Springer, 2003, pp. 117–133.

[4] de Alfaro, L., T. Henzinger and M. Stoelinga, *Timed interfaces*, in: *Proceedings of the Second International Workshop on Embedded Software*, LNCS, Springer, 2002, pp. 108–122.

[5] de Alfaro, L. and T. A. Henzinger, *Interface automata*, in: *ESEC / SIGSOFT FSE* (2001), pp. 109–120.

[6] de Alfaro, L. and T. A. Henzinger, *Interface theories for component-based design*, in: T. A. Henzinger and C. M. Kirsch, editors, *EMSOFT*, Lecture Notes in Computer Science **2211** (2001), pp. 148–165.

[7] de Alfaro, L. and T. H. Henzinger, *Interface-based design*, in: M. B. et al., editor, *Engineering Theories of Software-Intensive Systems*, Nato Science Series (2005), pp. 83–104.

[8] Fernandez, J.-C. and L. Mounier, *"on the fly" verification of behavioural equivalences and preorders*, in: *Procs. of CAV '91*, LNCS **575** (1991), pp. 181–191.

[9] Fernandez, J.-C., L. Mounier, C. Jard and T. Jéron, *On-the-fly verification of finite transition systems*, Formal Methods in System Design **1** (1992), pp. 251–273.

[10] Focardi, R. and R. Gorrieri, *Classification of security properties (part i: Information flow)*, in: *Procs. of FOSAD 2000*, LNCS **2171** (2001), pp. 331–396.

[11] Gardey, G., J. Mullins and O. H. Roux, *Non-interference control synthesis for security timed automata*, Electr. Notes Theor. Comput. Sci. **180** (2007), pp. 35–53.

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

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

[14] Lee, M. D. and P. R. D'Argenio, *Estructura de interfaces para sistemas con seguridad multi-nivel.*, Tech. Rep. Serie A 2009/3, FaMAF, Universidad Nacional de Córdoba (2009).

[15] Milner, R., "Communication and Concurrency," Prentice Hall International, 1989.