A Refinement Based Notion of Non-Interference for Interface Automata: Compositionality, Decidability and Synthesis

Matias Lee and Pedro R. D'Argenio
Fac. de Matemática, Astronomía y Física,
Univ. Nac. de Córdoba - CONICET
Córdoba, Argentina
{lee,dargenio}@famaf.unc.edu.ar

***Abstract*—Interface automata (IA) introduce a framework to model stateful interfaces. Interface structures for security (ISS) extend IA to cope with security properties. In this article, we argue that bisimulation-based non interference is not quite appropriate to characterize security on ISS. We instead introduce refinement-based variants of non-interference that fit better in this context. Moreover, we show that these new properties are not preserved by composition, but give sufficient conditions to ensure compositionality. We give two algorithms. The first one determines if an ISS satisfies the refinement-based non-interference property. The second one, determines if an ISS can be made secure by controlling some input actions and, if so, synthesizes the secure ISS.**

***Index Terms*—interface automata; non-interference; refinement; security**

I. Introduction

Nowadays, it is natural to see two or more system interacting in order to carry out difficult or complex tasks. For example, a *web service* can use information provided by other web services to offer a complex new service. To study how this interaction occurs the concept of *system interface* was introduced. A system interface includes all methods and ways that a system uses with the aim to interact with its environment.

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

Interface automata (IA) [1], [2] is a light-weight formalism that captures the temporal aspects of software component interfaces. In this context, interfaces interact with the environment using *input* and *output* actions (visible actions). Input actions represent the information that can be received by the interface, while output actions represent the information generated by it. There is a third sort of action, the *hidden* actions, which are used to represent internal transitions which cannot be observed by the environment. To enrich the description of interfaces, variants of IA have been devised. Each variant was enriched according to the requirements to be enforced on the system. In *Resource Interfaces* [3] the resources to complete a task are limited. *Timed Interfaces* [4] allow expressing time requirements.

Interface structure for security (ISS) [5] is yet another variant of IA, where there are two different types of visible 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. Low actions are intended to be accessed by any user while high actions can only be accessed by those users having the appropriate clearance. In this context the desired requirement is the so-called *non-interference* property [6]. In the setting of ISS we have considered bisimulation based notion of non-interference, more precisely, the so called BSNNI and BNNI properties [7]. Informally, these properties state that users with *no* appropriate permission cannot deduce any kind of confidential information or activity by only interacting through low actions. Since it is expected that a low-level user cannot distinguish the occurrence of high actions, the system has to behave the same when high actions are not performed or when high actions are considered as hidden actions. To formalize the idea of "behave the same", the concept of weak bisimulation is used. In this work, we argue that BSNNI/BNNI are not suitable properties to formalize the concept of *secure interface*.

To illustrate our point we present two examples: in the first one (Figure 1), we get that the system does not satisfy neither BNNI nor BSNNI but we show that it could be considered secure since no information is actually revealed to low users. The main problem is the way in which weak bisimulation relates output transitions. In addition, the second example (Figure 2) shows that weak bisimulation based security properties may fail to detect an information leakage through input transitions.

Figure 1 models a credit approval process of an on-line banking service using an ISS. As usual, outputs are suffixed by *!* and inputs by *?*. At the initial state s_1, a client can request a credit (*cred_req?*). The credit approval process can be carried on locally or by delegating it to an external component. This decision is modeled by a non deterministic choice. If it is locally processed (*loc_ctrl!*), an affirmative or negative response is given to the client (*yes!*/*no!*) and the

1522-4902/10 $26.00 © 2010 IEEE
DOI 10.1109/SCCC.2010.14

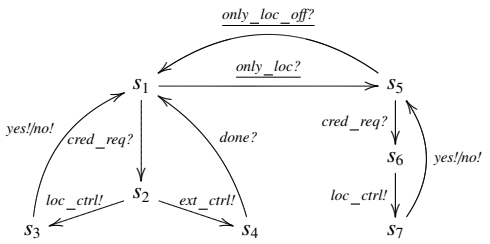

Figure 1: Credit approval process of an on-line banking service

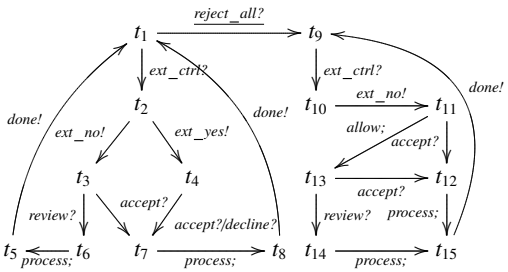

Figure 2: External Control Process in an on-line banking service

process returns to the initial state. On the other hand, if the decision is delegated (*ext_ctrl!*), the process waits until it receives a notification that the control is finished (*done?*), returning then to the initial state. Besides, in the initial state, an administrator can configure the system to do only local control (*only_loc?*). This action is high and is not visible for low users. (We underline private/high actions.) In state s_5, the administrator can configure the system to return to the original configuration using action *only_loc_off?*.

If we check the BSNNI (or BNNI) property on the the Credit Request Process, it results to be insecure. The system behaves differently depending on whether the private action *only_loc?* is performed or not. If *only_loc?* is not executed, after action *cred_req?*, it is possible to execute action *ext_ctrl!*. This behavior is not possible after the action *only_loc?*. Notice nevertheless that output actions are not visible for the user until they are executed. Then, from a low user perspective, the system behavior does not seem to change: the same input is accepted at states s_1 and s_5, and then, the low user cannot distinguish whether the observation of *loc_ctrl!* is a consequence of the unique option (at state s_6) or it is just an invariable decision of the Credit Request Process (at state s_2). Hence we expect the system to be classified as secure by the formalism.

We consider this example to be secure because a user does not know exactly what output action can be executed by an interface if he has no knowledge of the current state, he can observe the output actions only when they are executed.

On the other hand, a user may try to guess the behavior of the system by performing input actions: wrong inputs will be rejected/ignored; otherwise, they will be accepted. Based on this fact, the following example shows that weak bisimulation based non-interference may fail to detect an information leakage.

Figure 2 depicts the component that executes the external control. In the initial state, the interface waits for input *ext_ctrl?* from the Credit Request Process. After this stimulus, a response about the credit request is given. If the credit is denied (*ext_no!*), the client can either ask for a decision review (*review?*) or accept the decision (*accept?*). In both cases, the decision is processed by the component (*process;*). This action is internal and is not visible by users (hidden/internal action are suffixed by semicolon). The process finishes with action *done!* returning to the initial state. If the credit is approved (*ext_yes!*), the client can accept or decline the credit (*accept?/decline?*). The decision is processed, the component informs that the task is done and it returns to the initial state. As in the first example, the behavior of the component can be modified by an administrator, which can configure the interface to reject all credit requests (*reject_all?*). For this reason, if *reject_all?* is received at the initial state, after an input action *ext_ctrl?*, the process can only execute action *ext_no!*. At this point, clients are not allowed to ask for a decision review. Then, at state t_{11}, the interface accepts only input action *accept?*. However, based on the client records, the review may be enabled; this is represented with the internal transition $t_{11} \xrightarrow{allow;} t_{13}$, notice state t_{13} accepts both inputs actions *accept?* and *review?*. In any case, after the client response, the result is processed, the component informs that the task is done, and the process is restarted.

Suppose that the bank requires that the client cannot detect whether the external process is denying all credit request. Since a low user cannot see the output action until they are executed, he cannot differentiate between the executions $t_1 \xrightarrow{ext\_ctrl?} t_2 \xrightarrow{ext\_no!} t_3$ and $t_9 \xrightarrow{ext\_ctrl?} t_{10} \xrightarrow{ext\_no!} t_{11}$. If we compare states t_3 and t_{11} under weak bisimulation, both state can execute the same visible transitions and no security problem is detected. Notice that at state t_{11}, the process cannot respond immediately to a *review?* input, but it can execute $t_{11} \xrightarrow{allow;} t_{13} \xrightarrow{review?} t_{14}$ (recall *allow;* is an internal action). In fact, low users can distinguish state t_3 from t_{11}: testing the interface at state t_{11}, the low user can find out that input action *review?* is not enabled, while at t_3 it is. Hence, we consider that the interface is not secure.

In this work, we define refinement-based variants of non-interference that specifically considers this asymmetry between inputs and outputs. The idea of "behave the same" is relaxed: under the new approach, a system is secure if the system with high level activity "performs at most the same" as the system where no high action is performed. This ensures that a low user cannot observe new behavior in the system where high actions take place (but are not observable) w.r.t. the same system where no high action is executed. The new formalizations of security are obtained just like BSNNI and BNNI, but based on refinement instead of weak bisimulation.

Such refinement relation is based on the relation introduced in [2].

We show that the new non-interference property is not preserved by interface composition in general (just like it occurs with BSNNI and BNNI), but give sufficient and simple conditions to ensure compositionality. In addition, we compare it to existing notions of non-interference, and conclude that our refinement-based notion strictly includes trace-based non-interference, but it is incomparable to the bisimulation-based notion. We also provide two algorithms. The first one determines if an ISS satisfies the refinement-based non-interference property. The second one, determines if an ISS can be made secure by controlling some input actions, and if so, synthesizes the secure ISS. Both algorithms are polynomial in the number of states of the ISS under study.

Organization of the paper. In Section II we recall interface structure for security and the BSNNI and BNNI properties. Then, we present some examples where the BSNNI/BNNI properties are not suitable formalization of security. Finally, we introduce SIR-SNNI and SIR-NNI properties, our new variant of non-interference. In Section III we study how SIR-SNNI and SIR-NNI behave w.r.t. interface composition. Section IV presents the algorithms to check and to derive a secure interface. Section V concludes the paper.

II. Non-interference Properties for Interfaces

First, we define *Interface Automata* (IA) [1], [8] and *Interface Structure for Security* (ISS) [5], and introduce some notation.

Definition 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 finite and *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). If there is an autonomous execution from q to q' and all action are hidden, we write $q \overset{\varepsilon}{\Rightarrow} q'$. Notice this includes case $q = q'$. We write $q \overset{a}{\Rightarrow} q'$ if there are q_1 and q_2 s.t. $q \overset{\varepsilon}{\Rightarrow} q_1 \xrightarrow{a} q_2 \overset{\varepsilon}{\Rightarrow} q'$. Moreover $q \overset{\hat{a}}{\Rightarrow} q'$ denotes $q \overset{a}{\Rightarrow} q'$ or $a \in A^H$ and $q = q'$. We write $q \overset{\varepsilon}{\Rightarrow}\!\xrightarrow{a}$ if there is q' s.t. $q \overset{\varepsilon}{\Rightarrow} q'$ and $q' \xrightarrow{a}$. A *trace* from q_0 is a sequence of visible actions $a_0, a_1 \cdots$ such that there are states $q_1, q_2, \cdots$ such that $q_0 \overset{a_0}{\Rightarrow} q_1 \overset{a_1}{\Rightarrow} q_2 \cdots$ is an execution. The set of traces of an IA S, notation $Traces(S)$, is the set of all traces from the initial state of S.

An *Interface Structures for Security* is an IA, where visible actions are divided in two disjoint sets: the *high action* set and the *low action* set. Low actions can be observed and used for any user, while high actions are intended only for users with the appropriate clearance.

Definition 2: An *Interface Structure for Security (ISS)* is a tuple $\langle S, A^h, A^l \rangle$ where $S = \langle Q, q^0, A^I, A^O, A^H, \rightarrow \rangle$ 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 $m \in \{h, l\}$.

Non-interference Properties. It is expected that a low-level user cannot distinguish the occurrence of high actions. Therefore, we expect that a system behaves the same way if it does not perform any high action or if it just hides them to the view of the low users. Hence, restriction and hiding are central to our definitions of security.

Definition 3: 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 $\mathcal{S}/X = \langle S/X, A^h_S - X, A^l_S - X \rangle$.

A possible idea of "behaves in the same way" is encoded by the weak bisimulation.

Definition 4: 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. $q_T \overset{\hat{a}}{\Rightarrow}_T q'_T$ 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. $q_S \overset{\hat{a}}{\Rightarrow}_S q'_S$ 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 $\mathcal{T}$ are bisimilar, and write $\mathcal{S} \approx \mathcal{T}$, whenever the underlying IA are bisimilar.

The definitions of BSNNI and BNNI, which we recall in the following, were introduced in [7].

Definition 5: 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 formalizes the security property as we described 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 because less actions are restricted. Actually BNNI and BSNNI are incomparable [7]. In the definition of BNNI only high input actions are restricted

since the low-level user cannot provide this type of actions; instead high output actions are only hidden since they still can autonomously occur. The second notion is considered as it seems appropriate for IA where only input actions are controllable.

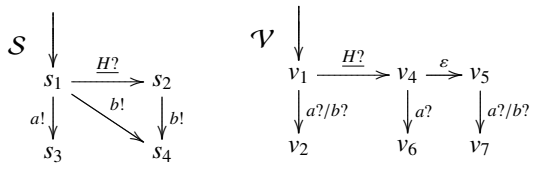

Figure 3: In these interfaces, BSNNI and BNNI are not appropriate properties to denote security.

Unfortunately, BSNNI and BNNI do not yield the expected security qualities in the context of interfaces as we have seen in the introduction. In the following we isolate the situations described before in two schematic examples. Consider the ISS $\mathcal{S}$ in Figure 3. Interface $\mathcal{S}$ does not satisfy BSNNI nor BNNI, but notice that under high level activity ($s_1 \xrightarrow{H?} s_2$) the interface does not change its behavior w.r.t. the low user view; the state s_2 just can execute $b!$ that is a possible execution before the execution of $s_1 \xrightarrow{H?} s_2$. Notice that the same reasoning holds if we replace $a!$ by a; or $b!$ by b;. On the other hand, interface $\mathcal{V}$ satisfies both BSNNI and BNNI, despite this, low users can see differences after high level activity: v_1 and v_4 are weak bisimilar, but v_1 is able to receive input actions $a?$ and $b?$ while v_4 is only able to receive $a?$. Examples $\mathcal{S}$ and $\mathcal{V}$ correspond respectively to the examples of Fig. 1 and 2. (Notice that the External Control Process is not BSNNI nor BNNI.)

To address these shortcomings, we introduce a new variation of non-interference. These variants are obtained from the definition of BSNNI and BNNI by replacing weak bisimulation by a new relation. Under this new relation, two states s and t are related if they are able to receive the same input actions; for every output transition that can execute t, the state s can execute zero or more hidden transitions before executing the same output; finally, all hidden transitions that can execute t can be "matched" by s with zero or more hidden transitions. In all cases, the reached states have to be also related. In this way state t does not reveal new visible behavior w.r.t. the state s. Formally:

Definition 6: Given two IA S and T, a relation $\succcurlyeq \subseteq Q_S \times Q_T$ is a *Strict Input Refinement (SIR)* of S by T if $q_S^0 \succcurlyeq q_T^0$ and for all $q_S \succcurlyeq q_T$ it holds:

(a) $\forall a \in A_S^I, q'_S \in Q_S$, if $q_S \xrightarrow{a}_S q'_S$ then $\exists q'_T \in Q_T : q_T \xrightarrow{a}_T q'_T$ and $q'_S \succcurlyeq q'_T$;

(b) $\forall a \in A_T^I, q'_T \in Q_T$, if $q_T \xrightarrow{a}_T q'_T$ then $\exists q'_S \in Q_S : q_S \xrightarrow{a}_S q'_S$ and $q'_S \succcurlyeq q'_T$;

(c) $\forall a \in A_T^O, q'_T \in Q_T$, if $q_T \xrightarrow{a}_T q'_T$ then $\exists q'_S \in Q_S : q_S \xRightarrow{\varepsilon}_S \xrightarrow{a}_S q'_S$ and $q'_S \succcurlyeq q'_T$;

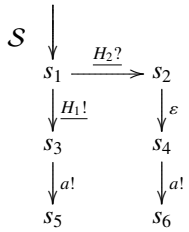

Figure 4: $\mathcal{S}$ is SIR-NNI and not SIR-SNNI

(d) $\forall a \in A_T^H, q'_T \in Q_T$, if $q_T \xrightarrow{a}_T q'_T$ then $\exists q'_S \in Q_S : q_S \xRightarrow{\varepsilon}_S q'_S$ and $q'_S \succcurlyeq q'_T$.

We say S is *refined (strictly on inputs) by* T, or, T *refines (strictly on inputs) to* S, notation $S \succcurlyeq T$, if there is a SIR $\succcurlyeq$ s.t. $S \succcurlyeq T$. Let $\mathcal{S}$ and $\mathcal{T}$ be two ISS, we write $\mathcal{S} \succcurlyeq \mathcal{T}$ if the underlying IA satisfy $S \succcurlyeq T$.

The definition of SIR is based on the definition of *refinement* of [2] only that restriction (b) is new in our relation. We now provide our SIR-based non-interference properties.

Definition 7: Let $\mathcal{S}$ be an ISS. *(i)* $\mathcal{S}$ is *SIR-based strong non-deterministic non-interference (SIR-SNNI)* if $\mathcal{S}\backslash A^h \succcurlyeq \mathcal{S}/A^h$ *(ii)* $\mathcal{S}$ is *SIR-based non-deterministic non-interference (SIR-NNI)* if $\mathcal{S}\backslash A^{I,h}/A^{O,h} \succcurlyeq \mathcal{S}/A^h$.

This new formalization of security ensures that under the presence of high level activity no new information is revealed to low users w.r.t. the system with only low activity, because the interface $\mathcal{S}\backslash A^h$ (resp. $\mathcal{S}\backslash A^{I,h}/A^{O,h}$) is refined by $\mathcal{S}/A^h$,

In Figure 3, interface $\mathcal{S}$ is SIR-NNI and SIR-SNNI but not BNNI or BSNNI; on the other hand, $\mathcal{V}$ is BNNI and BSNNI but not SIR-NNI or SIR-SNNI. Notice that these examples imply that SIR-NNI/SIR-SNNI and BNNI/BSNNI are incomparable properties.

In contrast to BNNI and BSNNI, SIR-NNI and SIR-SNNI are related. The following lemma establishes this relation.

Lemma 1: Let $\mathcal{S}$ be an ISS. If $\mathcal{S}$ is SIR-SNNI then $\mathcal{S}$ is SIR-NNI. The converse is not true.

Proof: As $\mathcal{S}$ is SIR-SNNI, then there is a SIR relation $\succcurlyeq$ s.t. $\mathcal{S}\backslash A^h \succcurlyeq \mathcal{S}/A^h$. Notice that all reachable states in $\mathcal{S}\backslash A^h$ are reachable in $\mathcal{S}\backslash A^{h,I}/A^{h,O}$. Let $s_r \in Q_{\mathcal{S}\backslash A^h}$ and $s_a \in Q_{\mathcal{S}/A^h}$ be two states s.t. $s_r \succcurlyeq s_a$. Define $\hat{s}_r$ as the state s_r in the interface $\mathcal{S}\backslash A^{h,I}/A^{h,O}$. The state $\hat{s}_r$ could have some new internal transition w.r.t the state s_r; these new transitions are results of hiding high output action. Since this kind of transitions are not taken into account by the definition of SIR, we have that $\hat{s}_r \succcurlyeq s_a$. Then $\mathcal{S}\backslash A^{I,h}/A^{O,h} \succcurlyeq \mathcal{S}/A^h$.

On the other hand, SIR-NNI does not imply SIR-SNNI. This is shown in Figure 4. ■

The coarsest sensible semantics on labeled transition system is trace semantic [9]. Then, it is desirable that given a definition of security and an interface satisfying that definition, the interface with hidden high level activity does not have more low traces than the interface with only low activity. SIR-SNNI and SIR-NNI satisfy this requirement. The following lemma

formalizes this, showing the inclusion of the traces of $\mathcal{S}/A^h$ in the traces of $\mathcal{S}\backslash A^h$ and $\mathcal{S}\backslash A^{I,h}/A^{O,h}$.

Lemma 2: Let $\mathcal{S} = \langle S, A^h, A^l\rangle$ be an ISS. *(i)* If $\mathcal{S}$ is SIR-NNI then $Traces(S\backslash A^{h,I}/A^{h,O}) \supseteq Traces(S/A^h)$. *(ii)* If $\mathcal{S}$ is SIR-SNNI then $Traces(S\backslash A^h) \supseteq Traces(S/A^h)$.

Proof: Notice $Traces(S\backslash A^h) \subseteq Traces(S\backslash A^{h,I}/A^{h,O})$, then we only have to prove (*ii*). The proof of (*ii*) is straightforward using the following inductive hypothesis on k: if $a_0 \cdots a_k \in Traces(S/A^h)$ is s.t. $q^0 \stackrel{a_0}{\Rightarrow} q_0 \stackrel{a_1}{\Rightarrow} \cdots \stackrel{a_k}{\Rightarrow} q_k$ then there are states $q'_0, \cdots, q'_k$ such that $q^0 \stackrel{a_0}{\Rightarrow} q'_0 \stackrel{a_1}{\Rightarrow} \cdots \stackrel{a_k}{\Rightarrow} q'_k$ is an execution of $S\backslash A^h$ and $q'_k \succcurlyeq q_k$. ∎

It is also possible to define trace-based non-interference properties following the same ideas as for BNNI and BSNNI [7]: an ISS $\mathcal{S}$ satisfies *non-deterministic non-interference (NNI)* (resp. *Strong NNI (SNNI)*) if $Traces(S\backslash A^{h,I}/A^{h,O}) = Traces(S/A^h)$ (resp. $Traces(S\backslash A^h) = Traces(S/A^h)$). From Lemma 2 we can straightforwardly conclude:

Corollary 1: Let $\mathcal{S} = \langle S, A^h, A^l\rangle$ be an ISS. *(i)* If $\mathcal{S}$ is SIR-NNI then $\mathcal{S}$ is NNI. *(ii)* If $\mathcal{S}$ is SIR-SNNI then $\mathcal{S}$ is SNNI.

The interface $\mathcal{V}$ in Fig. 3 shows that an ISS can satisfy SNNI/NNI but not SIR-SNNI/SIR-NNI.

III. Security Properties and Composition

In this section we study how the SIR-SNNI and SIR-NNI properties respond to interface composition. We first define interface composition and show that given two secure interfaces the composition is not secure in general. Then we present some conditions that guarantee that the composition of two interfaces is secure if the interfaces are secure.

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 8: Let S and T be two IA, and let $shared(S,T) = (A_S \cap A_T)$ be the set of *shared actions*. We say that S and T are *composable* whenever $shared(S,T) = (A^I_S \cap A^O_T) \cup (A^O_S \cap A^I_T)$. Two ISS $\mathcal{S} = \langle S, A^h_S, A^l_S\rangle$ and $\mathcal{T} = \langle T, A^h_T, A^l_T\rangle$ are composable if S and T are composable.

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)* only shared actions can 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 9: Let S and T be composable IA. The *product* $S \otimes T$ is the interface automaton defined by:

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

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 these are called *error states*.

Definition 10: 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 $a \in shared(S,T)$ s.t. either $a \in A^O_S$, $q_S \stackrel{a}{\rightarrow}_S$ and $q_T \stackrel{a}{\not\rightarrow}_T$, or $a \in A^O_T$, $q_T \stackrel{a}{\rightarrow}_T$ and $q_S \stackrel{a}{\not\rightarrow}_S$.

If the product $S \otimes T$ does not contain any reachable error state, then each component satisfies the interface of the other (i.e., the input assumptions) and thus are 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) an error state. In such a case we say that $S \otimes T$ is incompatible.

Definition 11: 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. Extending the definition of composition to ISS is straightforward.

Definition 12: 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 $q \stackrel{a}{\rightarrow}_{S\otimes T} q'$ 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 $S \otimes T$.

Given two composable ISS, $\mathcal{S} = \langle S, A^h_S, A^l_S\rangle$ and $\mathcal{T} = \langle T, A^h_T, A^l_T\rangle$, their *composition*, $\mathcal{S} \parallel \mathcal{T}$, is defined by the ISS $\langle S \parallel T, (A^h_S \cup A^h_T) - shared(S,T), (A^l_S \cup A^l_T) - shared(S,T)\rangle$.

It is known from [5] that neither BNNI nor BSNNI are preserved by interface parallel composition. The same happens for SIR-NNI and SIR-BNNI properties. An example is reported in Fig. 5. Despite this, 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 do not use confidential actions to synchronize. This is stated in the following theorem.

Theorem 1: Let $\mathcal{S} = \langle S, A^h_S, A^l_S\rangle$ and $\mathcal{T} = \langle T, A^h_T, A^l_T\rangle$ be two composable ISS such that $shared(S,T) \cap (A^h_S \cup A^h_T) = \emptyset$. If $\mathcal{S} \otimes \mathcal{T}$ has no reachable error states and $\mathcal{S}$ and $\mathcal{T}$ satisfy SIR-SNNI (resp. SIR-NNI) then $\mathcal{S} \parallel \mathcal{T}$ satisfies SIR-SNNI

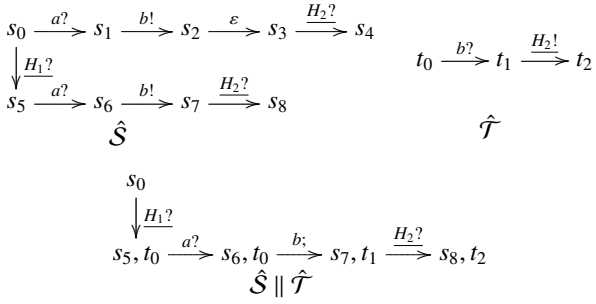

Figure 5: SIR-SNNI/SIR-NNI properties are not preserved by composition

(resp. SIR-NNI).

Proof: Define $\succcurlyeq$ by $(s_r, t_r) \succcurlyeq (s_a, t_a)$ iff $s_r \succcurlyeq_{\mathcal{S}} s_a$ and $t_r \succcurlyeq_{\mathcal{T}} t_a$ with $\succcurlyeq_{\mathcal{S}}$ being a SIR between $S\backslash A_S^h$ and S/A_S^h and similarly for $\succcurlyeq_{\mathcal{T}}$. We show that $\succcurlyeq$ is a SIR between $(\mathcal{S} \parallel \mathcal{T})\backslash A^h$ and $(\mathcal{S} \parallel \mathcal{T})/A^h$ where $A^h = (A_S^h \cup A_T^h) - \mathit{shared}(S,T) = A_S^h \cup A_T^h$.

Suppose $(s_r, t_r) \succcurlyeq (s_a, t_a)$. We proceed by case analysis on the different transfer properties on Def 6. For case (a) suppose $(s_r, t_r) \xrightarrow{a?} (s'_r, t_r)$ and $s_r \succcurlyeq_{\mathcal{S}} s_a$. Then there is s'_a such that $s_a \xrightarrow{a?} s'_a$ and $s'_r \succcurlyeq s'_a$. As a consequence of the absence of error state in the product, we can ensure $(s_a, t_a) \xrightarrow{a?} (s'_a, t_a)$ and $(s'_r, t_r) \succcurlyeq (s'_a, t_a)$. The case $(s_r, t_r) \xrightarrow{a?} (s_r, t'_r)$ is analogous. In the same way we prove that condition (b) holds. For condition (c), let $(s_a, t_a) \xrightarrow{a!} (s'_a, t_a)$ and $s_r \succcurlyeq_{\mathcal{S}} s_a$. Then there is s'_r such that $s_r \Rightarrow\!\xrightarrow{a!} s'_r$ and $s'_r \succcurlyeq_{\mathcal{S}} s'_a$. Let $\hat{s}$ be a state s.t. $s_r \Rightarrow \hat{s} \xrightarrow{a!} s'_r$. Notice that all internal transition used to reach $\hat{s}$ in $S\backslash A^h$ can be executed in $(\mathcal{S} \parallel \mathcal{T})\backslash A^h$. Then $(s_r, t_r) \Rightarrow (\hat{s}, t_r) \xrightarrow{a!} (s'_r, t_r)$ and $(s'_r, t_r) \succcurlyeq (s'_a, t_a)$. The case $(s_a, t_a) \xrightarrow{a!} (s_a, t'_a)$ is analogous. We finally prove that condition (d) holds. Cases $(s_a, t_a) \xrightarrow{\varepsilon} (s'_a, t_a)$ and $(s_a, t_a) \xrightarrow{\varepsilon} (s_a, t'_a)$ are similar to the previous one. Suppose now $(s_a, t_a) \xrightarrow{\varepsilon_c} (s'_a, t'_a)$ where ε_c is an internal action resulting from a synchronization between $\mathcal{S}$ and $\mathcal{T}$ on common action c. Notice $c \in A_S^l \cap A_T^l$. W.l.o.g suppose $s_a \xrightarrow{c?} s'_a$ and $t_a \xrightarrow{c!} t'_a$. Repeating previous reasoning, we can ensure there is state $\hat{t}$ such that $(s_r, t_r) \Rightarrow (s_r, \hat{t}) \xrightarrow{c;} (s'_r, t'_r)$ and $(s'_r, t'_r) \succcurlyeq (s'_a, t'_a)$. ■

This result is useful when we develop all the components of a complex system. As we have total control of each component design, it is possible to achieve full compatibility. In this way, to ensure that the composed system is secure, we only have to develop secure components s.t. every high action of the component is a high action of the final system. This result can also be used when we are not in control of all components, i.e. we want use components not developed by us. The idea is simple, given two ISS, define the high actions used in the communication process as low and check if the resulting ISS satisfies the hypothesis of Theorem 1.

Corollary 2: 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 SIR-SNNI (resp. SIR-NNI) then $\mathcal{S} \parallel \mathcal{T}$ satisfies SIR-SNNI (resp. SIR-NNI).

This result is based on the fact that actions used in the synchronization become hidden in the composition, then it is not important the confidential level of the actions.

IV. Deriving Secure Interfaces

As we have just seen, the composition of secure interfaces may yield a new insecure interface. This may happen when the components are already available but they where designed independently and they were not meant to interact. The question that arises then is if there is a way to derive a secure interface out of an insecure one. To derive the secure interface, we adapt the idea used to define ISS composition (see Def. 12); i.e. we restrict some input transitions in order to avoid insecure behavior. We then obtained a composed system that offers less services than the original one but is secure. In this section we present an algorithm to derive an ISS satisfying SIR-SNNI (or SIR-NNI) from a given ISS whenever possible. Since the method is similar in both cases, we focus on SIR-SNNI.

This algorithm is based on the algorithm presented in [5] to derive interfaces that satisfy BSNNI/BNNI, which in turn is based on the algorithm for bisimulation checking of [10]. The differences between both algorithm are consequence of the definition of SIR but the idea behind the procedure is the same. The new algorithm works as follows: given two interfaces $\mathcal{V}$ and $\mathcal{V}'$, the second without high actions, *(i)* $\mathcal{V}$ is *semi-saturated* adding all weak transitions $\Rightarrow\!\xrightarrow{a}$; *(ii)* a *semi-synchronous product* of $\mathcal{V}$ and $\mathcal{V}'$ is constructed where transitions synchronize whenever they have the same label and satisfy some particular conditions; *(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* then $\mathcal{V} \not\succcurlyeq \mathcal{V}'$; if there is always a way to avoid reaching a fail state, then $\mathcal{V} \succcurlyeq \mathcal{V}'$. We later define properly *semi-saturation*, *semi-synchronous product* and what means *inevitably reaching a fail state*. In this way, given an ISS $\mathcal{S}$, we can check if $\mathcal{S}\backslash A^h \succcurlyeq \mathcal{S}/A^h$, if the check succeeds, then $\mathcal{S}$ satisfies SIR-SNNI (see Theorem 2). If it does not succeed, then we provide an algorithm to decide whether $\mathcal{S}$ can be transformed into a secure ISS by controlling (i.e. pruning) input transitions. This decision mechanism categorizes insecure interfaces in two different classes: the class of interfaces that can surely be transformed into secure one and the class in which this is not possible.

The algorithm to synthesize the secure ISS (once it is decided that it is possible) selects an input transition to prune, prune it, and checks whether the resulting ISS is secure. If it is not, a new input transition is selected and pruned. The process is repeated until it gets a secure interface. This process is shown to terminate (see Theorem 3).

Checking Strict Inputs Refinement. Different labels for internal actions do not play any role in a SIR relation. Then, to

simplify, we replace all labels of internal action for two new ones: ε and ε'. The label ε' is used to represent an internal transition that can be removed; in our context, an internal action can be removed because it is a high input action that was hidden in order to check for security. Label ε is used to identify internal action that cannot be removed. This is formalized in the following definition, which includes self-loops with ε and ε' for future simplifications.

Definition 13: Let S be an IA and $B \subseteq A_S^H$. Define S *marking* B or *marking* B *in* S as the IA $S_B = \langle Q_S, q_S^0, A_S^I, A_S^O, \{\varepsilon, \varepsilon'\}, \rightarrow_{S_B}\rangle$ where $\rightarrow_{S_B}$ is the least relation satisfying following rules:

$$q \xrightarrow{\varepsilon}_{S_B} q \qquad q \xrightarrow{\varepsilon'}_{S_B} q \qquad \frac{q \xrightarrow{a}_S q' \quad a \in A_S^I \cup A_S^O}{q \xrightarrow{a}_{S_B} q'}$$

$$\frac{q \xrightarrow{a}_S q' \quad a \in B}{q \xrightarrow{\varepsilon'}_{S_B} q'} \qquad \frac{q \xrightarrow{a}_S q' \quad a \in A_S^H - B}{q \xrightarrow{\varepsilon}_{S_B} q'}$$

Given an ISS $\mathcal{S}$, the *marking* B *in* $\mathcal{S}$, notation $\mathcal{S}_B$, is the ISS obtained after marking B in the underlying IA.

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 checking strong bisimulation on the saturated transition system. Applying a similar idea we can check if there is a SIR relation. We add a transition $q \xrightarrow{a} q'$ whenever $q \Rightarrow\!\xrightarrow{a} q'$ with a an output action. We call this process *semi-saturation*.

Definition 14: Let S be an IA such that $A_S^H = \{\varepsilon, \varepsilon'\}$. The *semi-saturation of* S is the IA $\overline{S} = \langle Q_S, q_S^0, A_S^I, A_S^O, \{\varepsilon, \varepsilon'\}, \rightarrow_{\overline{S}}\rangle$ where $\rightarrow_{\overline{S}}$ is the smallest relation satisfying the following rules:

$$\frac{q \xrightarrow{a}_S q'}{q \xrightarrow{a}_{\overline{S}} q'} \qquad \frac{q \xrightarrow{\varepsilon}_{\overline{S}} q' \quad q' \xrightarrow{a}_{\overline{S}} q'' \quad a \in A_S^O}{q \xrightarrow{a}_{\overline{S}} q''}$$

Given an ISS $\mathcal{S}$, its *semi-saturation*, $\overline{\mathcal{S}}$, is the ISS obtained by saturating the underlying IA.

The last definition ensure that: if $a \in A^O$ then $q \Rightarrow\!\xrightarrow{a}_S q'$ iff $q \xrightarrow{a}_{\overline{S}} q'$.

Following [5] and [10], the definition of the synchronous products follows from the conditions of the relation being checked, in this case SIR. First, we recapitulate these conditions and then we present the formal definition. If $S \Rightarrow T$ then for two states $s \in Q_S$ and $t \in Q_T$ s.t. $s \Rightarrow t$, every output/hidden action that t can execute has to be simulated by s (probably using internal action); on the other hand, t is not forced to simulate output/hidden actions from s. Finally, both states have to simulate all input action that can be executed by the other one without performing previously any internal action. All these restrictions become evident from the definition of SIR. When a condition is not satisfied, a transition to a special state *fail* is created. Taking this into account we define the *semi-synchronized product*.

Definition 15: Let S be a semi-saturated IA and T be an IA such that $A_S^X = A_T^X = A^X$ for $X \in \{I, O\}$ and $A_S^H = A_T^H = \{\varepsilon, \varepsilon'\}$. The *semi-synchronous product* of S and T is the IA $S \times T = \langle (Q_S \times Q_T) \cup \{fail\}, (q_S^0, q_T^0), A^I, A^O, \{\varepsilon, \varepsilon'\}, \rightarrow_{S\times T}\rangle$ where $\rightarrow_{S\times T}$ is the smallest relation satisfying following rules:

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

$$\frac{q_S \xrightarrow{\varepsilon}_S q_S' \quad q_T \xrightarrow{\varepsilon'}_T q_T'}{(q_S, q_T) \xrightarrow{\varepsilon'}_{S\times T} (q_S', q_T')} \qquad \frac{q_S \xrightarrow{a}_S \quad q_T \not\xrightarrow{a}_T \quad a \in A^I}{(q_S, q_T) \xrightarrow{a}_{S\times T} fail} \qquad \frac{q_S \not\xrightarrow{a}_S \quad q_T \xrightarrow{a}_T}{(q_S, q_T) \xrightarrow{a}_{S\times T} fail}$$

Given $\mathcal{S} = \langle S, A_S^h, A_S^l\rangle$ and $\mathcal{T} = \langle T, A_T^h, A_T^l\rangle$ with S and T satisfying conditions above and $A_S^m = A_T^m = A^m$ for $m \in \{l, h\}$, then the *semi-synchronous product* of $\mathcal{S}$ and $\mathcal{T}$ is defined by the ISS $\mathcal{S} \times \mathcal{T} = \langle S \times T, A^h, A^l\rangle$.

Let us show how we can use synchronous product to check and derive, whenever it is possible, a SIR relation. If there is a state (q_S, q_T) such that $(q_S, q_T) \xrightarrow{a}_{S\times T} fail$ then it is evident that $q_S \not\Rightarrow q_T$. Moreover, suppose the synchronous product only has states (q_S, q_T) and *fail* and the transition $(q_S, q_T) \xrightarrow{a}_{S\times T} fail$. If $a \in A^O$, as the progress from (q_S, q_T) is autonomous, there is no way to control the execution of a! and hence there is no way to avoid $q_S \not\Rightarrow q_T$. Then, we say that (q_S, q_T) *fails* the SIR-relation test. On the other hand, if $a \in A^I$, a state offers a service that the other does not. In this case, removing the input transition a (the interface offers less services), we avoid transition $(q_S, q_T) \xrightarrow{a}_{S\times T} fail$ in the synchronous product and we get two states such that $q_S \Rightarrow q_T$, moreover, we get two interfaces related by a SIR relation. In this case, we say that (q_S, q_T) *may pass* the SIR relation test. In a more complex synchronous product, the "failure" in the state (q_S, q_T) has to be propagated backwards appropriately to identify pairs of states that cannot be related. This propagation is done by the definitions of two different sets: *Fail* and *May*. The set *Fail* contains those pairs that are not related by a refinement and there is no set of input transitions to prune so that the pair may become related by the refinement. On the other hand, *May* contains pairs of states that are not related but will be related if some transition is pruned. States not in *Fail* ∪ *May*, belong to the set *Pass*. All pairs in *Pass* are related by a SIR relation.

Definition 16: Let $S \times T$ be a synchronous product. We define the sets $Fail, May, Pass \subseteq Q_{S\times T}$ respectively by:

- $Fail = \cup_{i=0}^{\infty} Fail^i$ where $Fail^i$ is defined in Table I. If $q \in Fail$, we say that the pair *q fails the SIR relation test*.
- $May = \cup_{i=0}^{\infty} May^i$ where May^i is defined in Table II. If $q \in May$, we say that the pair *q may pass the SIR relation test*.
- $Pass = Q_{S\times T} - (May \cup Fail)$. If $q \in Pass$, we say that the pair *q passes the SIR relation test*

If the initial state of the underlying IA of an ISS $\mathcal{S}\times\mathcal{T}$ passes (may pass, fails) the SIR relation test, we say that $\mathcal{S}\times\mathcal{T}$ passes (may pass, fails) the SIR relation test.

The proof of the following lemma is based on the proof of the algorithm to check bisimulation in [10], for this reason we only present a proof sketch. Our proof deviates a little from the original as a consequence of not all mismatching transitions are problematic.

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

$$Fail^{k+1} = Fail^k \cup \{(q_S, q_T) : a \in A^O \cup A, q_T \xrightarrow{a} q'_T, (\forall q'_S : (q_S, q_T) \xrightarrow{a} (q'_S, q'_T) : (q'_S, q'_T) \in Fail^k)\}$$

Table I: The *Fail* set.

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

$$May^0_{q \xrightarrow{a} q'} = \{(q_S, q_T) : (q = q_S \vee q = q_T), a \in A^I, (q_S, q_T) \xrightarrow{a}_{S\times T} fail\}$$

$$May^{k+1}_{q_S \xrightarrow{a} q'_S} = \{(q_S, q_T) \notin Fail : a \in A, q_S \xrightarrow{a} q'_S, (\forall q'_T : (q_S, q_T) \xrightarrow{a} (q'_S, q'_T) : (q'_S, q'_T) \in Fail \cup May^k)\}$$

$$May^{k+1}_{q_T \xrightarrow{a} q'_T} = \{(q_S, q_T) \notin Fail : a \in A, q_T \xrightarrow{a} q'_T, (\forall q'_S : (q_S, q_T) \xrightarrow{a} (q'_S, q'_T) : (q'_S, q'_T) \in Fail \cup May^k)\}$$

Table II: The *May* set definition.

Lemma 3: A semi-synchronized product $S \times T$ passes the SIR relation test iff $S \geqslant T$.

Proof sketch: Since $(May \cup Fail) \cap Pass = \emptyset$, we only have to prove that *(i)* $(q_S, q_T) \in May \cup Fail$ implies $q_S \not\geqslant q_T$ and *(ii)* if $(q_S, q_T) \in Pass$ then $q_S \geqslant q_T$. The proof of (i) is by induction on k in May^k and $Fail^k$. The proof of (ii) is straightforward after showing that, given a state $(s, t) \in Q_{S\times T} \cap Pass$, then:

1) if $s \xrightarrow{a} s'$ and $a \in A^I$ then there is a state t' s.t. there is a transition $(s, t) \xrightarrow{a} (s', t')$ and $(s', t') \in Pass$.
2) if $t \xrightarrow{a} t'$ then there is a state s' s.t. there is a transition $(s, t) \xrightarrow{a} (s', t')$ and $(s', t') \in Pass$.

The proof of both statements is by case analysis on a obtaining always a contradiction. ∎

Using this lemma, we can verify if an interface is SIR-SNNI, since $\mathcal{S}$ is SIR-SNNI if $S\backslash A^h$ is refined by S/A^h. Notice that we cannot use $S\backslash A^h$ and S/A^h to create a semi-synchronized product; in general, $S\backslash A^h$ does not satisfy $A^H = \{\varepsilon, \varepsilon'\}$ and it is not semi-saturated. This can be solved marking $\emptyset$ in $S\backslash A^h$ and then semi-saturating the interface, i.e. we work with $\overline{(S\backslash A^h)_\emptyset}$ instead of $S\backslash A^h$. Similarly, S/A^h does not satisfy $A^H = \{\varepsilon, \varepsilon'\}$. Since ε' is used to represent the internal action that can be removed, we solve this problem marking $A^{h,I}$ in S/A^h, i.e. we replace S/A^h by $(S/A^h)_{A^{h,I}}$. Therefore, verifying that $\mathcal{S}$ satisfies SIR-SNNI amounts to checking whether $P_{\mathcal{S}} = \overline{S\backslash A^h}_\emptyset \times (S/A^h)_{A^{h,I}}$ passes the refinement test. Applying a similar reasoning, if we are interested on verifying SIR-NNI, we can check if $\overline{((S\backslash A^{h,I})/A^{h,O})}_\emptyset \times (S/A^h)_{A^{h,I}}$ passes the SIR-relation test. Then we have a decision algorithm to check whether an ISS satisfies SIR-SNNI or SIR-NNI. We state it in the following theorem.

Theorem 2: Let $\mathcal{S} = \langle S, A^h, A^I\rangle$ be an ISS.

1) $\mathcal{S}$ satisfies SIR-SNNI iff $\overline{(\mathcal{S}\backslash A^h)}_\emptyset \times (\mathcal{S}/A^h)_{A^{h,I}}$ passes the SIR-relation test.
2) $\mathcal{S}$ satisfies SIR-NNI iff $\overline{((S\backslash A^{h,I})/A^{h,O})}_\emptyset \times (S/A^h)_{A^{h,I}}$ passes the SIR-relation test.

Synthesizing Secure ISS. In the following, we show that if a synchronized product $P_{\mathcal{S}}$ may pass the SIR relation test then there is a set of input transition that can be pruned so that the resulting interface is secure. First, we need to select which are the candidate input actions to be removed. So, if $\mathcal{S}$ is an ISS such that $P_{\mathcal{S}}$ may pass the SIR-relation test, the set $EC(\mathcal{S}) \subseteq \rightarrow \cap\, Q \times A^I \times Q$ (see Table III) is the set of *eliminable candidates*.

All transitions in $EC(\mathcal{S})$ are involved in a synchronization that connects a source pair that may pass the SIR-relation test and a failing target. This can happen in four different situations. The first one is the basic case, in which one of the components of the pair can perform a low input transition that cannot be matched by the other. The following two cases are symmetric and consider the case in which both sides can perform an equally low input transition but end up in a failing state. The last case includes high input actions that are hidden in the synchronized product and always reach a pair that fails. Notice that if $P_{\mathcal{S}}$ may pass the bisimulation test then $EC(\mathcal{S}) \neq \emptyset$.

An important result is that no new failing pair of states is introduced by removing eliminable candidates. Moreover, if a pair of states fails in the synchronous product of the original ISS and it is also present in the synchronous product of the reduced ISS, then it also fails in this ISS. This ensures that a synchronous product that may pass the SIR-relation test, will not fail after pruning. In a sense, Lemma 4 below states that the sets *Fail* and $Pass \cup May$ remain invariant.

Lemma 4: Let $\mathcal{S}$ be an ISS s.t. $P_{\mathcal{S}}$ may pass the SIR-relation 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 it holds that: *(i)* $Fail_{P_{\mathcal{S}'}} = Fail_{P_{\mathcal{S}}} \cap Q_{P_{\mathcal{S}'}}$; *(ii)* $(Pass_{P_{\mathcal{S}}} \cup May_{P_{\mathcal{S}}}) \cap Q_{P_{\mathcal{S}'}} = Pass_{P_{\mathcal{S}'}} \cup May_{P_{\mathcal{S}'}}$ [1].

[1]Subindices in $Fail_{P_{\mathcal{S}}}$, $May_{P_{\mathcal{S}}}$, etc. indicate that these sets were obtained from the synchronous product $P_{\mathcal{S}}$

$$EC(P_{\mathcal{S}}) = \{q \xrightarrow{a} q' : (\exists \hat{q} : (q,\hat{q}) \in May^0_{q \xrightarrow{a} q'} \vee (\hat{q},q) \in May^0_{q \xrightarrow{a} q'})\} \cup \quad (1)$$

$$\{q \xrightarrow{a} q' : (\exists \hat{q} : (q,\hat{q}) \in May^1_{q \xrightarrow{a} q'}, (\forall \hat{q}' : (q,\hat{q}) \xrightarrow{a} (q',\hat{q}') : (q',\hat{q}') \in Fail))\} \cup \quad (2)$$

$$\{q \xrightarrow{a} q' : (\exists \hat{q} : (\hat{q},q) \in May^1_{q \xrightarrow{a} q'}, (\forall \hat{q}' : (\hat{q},q) \xrightarrow{a} (\hat{q}',q') : (\hat{q}',q') \in Fail))\} \cup \quad (3)$$

$$\{q \xrightarrow{a} q' : a \in A^{h,I}, q \xrightarrow{a} q', (\exists \hat{q} : (\hat{q},q) \in May^1_{q \xrightarrow{\varepsilon'} q'}, (\forall \hat{q}' : (\hat{q},q) \xrightarrow{\varepsilon'} (\hat{q}',q') : (\hat{q}',q') \in Fail))\} \quad (4)$$

Table III: Set of eliminable candidates.

Proof: We only show (i). (ii) is an immediate consequence of (i).

(*Case* $\subseteq$). Clearly $Q_{P_{S'}} \subseteq Q_{P_S}$. Suppose $q \xrightarrow{b?} q' \in EC(\mathcal{S})$ is the transition that is removed. By induction on k we show $Fail^k_{q \xrightarrow{a} q', P_{S'}} \subseteq Fail^k_{q \xrightarrow{a} q', P_S}$ for all k. This implies $Fail^k_{P_{S'}} \subseteq Fail^k_{P_S}$ and then $Fail_{P_{S'}} \subseteq Fail_{P_S}$. Suppose $(q_r,q_a) \in Fail^0_{q_a \xrightarrow{a} q'_a, P_{S'}}$. By definition, action $a \notin A^I \cup \{\varepsilon'\}$ and $(q_r,q_a) \xrightarrow{a} fail$. Then $a \neq b?$ and therefore $(q_r,q_a) \xrightarrow{a} fail$ belongs to P_S. Then $(q_r,q_a) \in Fail^0_{q_a \xrightarrow{a} q'_a, P_S}$. Suppose now $(q_r,q_a) \in Fail^{k+1}_{q_a \xrightarrow{a} q'_a, P_{S'}}$. Then $a \notin A^I \cup \{\varepsilon'\}$ and $(\forall q'_r : (q_r,q_a) \xrightarrow{a} (q'_r,q'_a) : (q'_r,q'_a) \in Fail^k_{P_{S'}})$. Notice that $\{(q'_r,q'_a) : (q_r,q_a) \xrightarrow{a}_{P_S} (q'_r,q'_a)\} = \{(q'_r,q'_a) : (q_r,q_a) \xrightarrow{a}_{P_{S'}} (q'_r,q'_a)\}$ as consequence of $b? \in A^I \cup \{\varepsilon'\}$. By induction hypothesis $Fail^k_{P_{S'}} \subseteq Fail^k_{P_S}$, then $\forall q'_a : (q_r,q_a) \xrightarrow{a} (q'_r,q'_a) : (q'_r,q'_a) \in Fail^k_{P_S}$ and we get $(q_r,q_a) \in Fail^{k+1}_{q_a \xrightarrow{a} q'_a, P_S}$ and $(q_r,q_a) \in Fail^{k+1}_{P_S}$.

(*Case* $\supseteq$).) We show by induction on k that $Fail^k_{P_{S'}} \supseteq Fail^k_{P_S} \cap Q_{P_{S'}}$ for all k. Let $(q_r,q_a) \in Fail^0_{P_S} \cap Q_{P_{S'}}$. Moreover, w.l.o.g. suppose $(q_r,q_a) \in Fail^0_{q_a \xrightarrow{a} q'_a, P_S}$. Since $a \notin A^I$, the transition $q_r \xrightarrow{a} q'_r$ cannot be removed and since $q_r \not\xrightarrow{a}$, then it holds that $(q_r,q_a) \in Fail^0_{q_r \xrightarrow{a} q'_r, P_{S'}} \subseteq Fail^0_{P_{S'}}$. For the induction case, suppose w.l.o.g. $(q_r,q_a) \in Fail^{k+1}_{q_a \xrightarrow{a} q'_a, P_S} \cap Q_{P_{S'}}$. Then $(\forall q'_r : (q_r,q_a) \xrightarrow{a} (q'_r,q'_a) : (q'_r,q'_a) \in Fail^k_{P_S})$. Since (q_r,q_a) is reachable in $\mathcal{S}'$ and $a \notin A^I$, all pair (q'_r,q'_a) is reachable in $\mathcal{S}'$. By induction hypothesis, $(q'_r,q'_a) \in Fail^k_{P_{S'}}$ and then $(q_r,q_a) \in Fail^{k+1}_{q_a \xrightarrow{a} q'_a, P_S} \subseteq Fail^{k+1}_{P_S}$. ∎

The following theorem is the main result of this section. Notice that its proof defines the algorithm to prune input actions and obtain a secure interface. A similar result holds for SIR-NNI.

Theorem 3: Let $\mathcal{S}$ be an ISS such that $P_{\mathcal{S}}$ may pass the SIR relation test. Then there is an input transition set $\rightarrow_\chi$ such that, if $\mathcal{S}'$ is the ISS obtained from $\mathcal{S}$ by removing all transitions in $\rightarrow_\chi$, $\mathcal{S}'$ is SIR-SNNI.

Proof: We only report a proof sketch. The complete proof follows in the same way as the proof of Theorem 4.10 in [5]. Let $\mathcal{S}'$ be an ISS obtained from $\mathcal{S}$ by removing one transition from the set $EC(\mathcal{S})$. Lemma 4 ensures that $\mathcal{S}'$ may pass or passes the SIR relation test. If $\mathcal{S}'$ passes the SIR relation test, we stop. If $\mathcal{S}'$ may pass the SIR relation test, we repeat the process until we obtain an ISS that passes the test. Since the transition set is finite, in the worst case, we will continue with the process until obtaining an ISS with an empty set of eliminable candidates. If this ISS may pass the SIR-relation test we get a contradiction with the fact that the set of eliminable candidates is empty, then this ISS has to pass the test. Finally, $\rightarrow_\chi$ is composed by the set of transitions removed along the way. ∎

V. Concluding remarks

Our contribution. We have presented a framework to manipulate stateful interfaces for security and their composition. In this context, an ISS is secure if it satisfies the SIR-SNNI (or SIR-NNI) property. SIR-SNNI and SIR-NNI properties have been introduced in this paper and we showed that are suitable definitions of non interference for interfaces; moreover, SIR-SNNI and SIR-NNI imply (traced-based) SNNI and NNI, respectively. We have studied how the properties behaves w.r.t composition and we have presented sufficient conditions ensuring the preservation of the properties under composition. We also provided a synthesis algorithm that decides whether an ISS can be turned into a secure ISS by controlling input transitions.

Related work. Little work has been developed to synthesis secure system. Cassez et al. [11] resolve a synthesis problem for SNNI in a framework similar to ours. In [12] and [13] the results are extended for *timed* non-interference on timed automata. Similar results presented in this paper can be found in [5] for BSNNI and BNNI. It is worthwhile to note that the BSNNI/BNNI based algorithm presented in [5] can only synthesize a secure interface through the control of low inputs. It maybe the case that a secure interface can be synthesized by controlling some high input, in which case the algorithm will only announce that it cannot determine whether it is possible to synthesize a secure interface. This is not the case on the SIR-SNNI/SIR-NNI based algorithm presented in this paper, which can always synthesize a secure interface if this is indeed possible.

Applications. The presented results can be useful in the context of web services. For example, in [14], an IA is constructed based on behavioral descriptions of a web service, which are expressed in OWL-S language. The IA constructed is used to find web services s.t. when they are composed, the resulting

interface satisfies the behavioral description. [15] addresses the problem of adapting the behavior of an interface to carry out a successful communication. Using this kind of techniques, we can remove input action to get a secure interface. Finally, [16] proposes a model to characterize security properties of a software component and show how to use this information to create *compositional security contracts* to ensure secure web service composition. In this context, SIR-SNNI and SIR-NNI can be new security attributes. Moreover, the result of Corollary 2 can be useful to create compositional security contracts for the new attributes.

Acknowledgment

We would like to thank Miguel Pagano for his useful comments on the draft version of this article.

References

[1] L. de Alfaro and T. A. Henzinger, "Interface theories for component-based design," in *EMSOFT*, ser. LNCS, T. A. Henzinger and C. M. Kirsch, Eds., vol. 2211. Springer, 2001.

[2] L. de Alfaro and T. H. Henzinger, "Interface-based design," in *Engineering Theories of Software-Intensive Systems*, ser. Nato Science Series, M. B. et al., Ed. Springer, 2005, pp. 83–104.

[3] A. Chakrabarti, L. de Alfaro, T. Henzinger, and M. Stoelinga, "Resource interfaces," in *EMSOFT*. LNCS, Springer, January 2003, pp. 117–133.

[4] L. de Alfaro, T. Henzinger, and M. Stoelinga, "Timed interfaces," in *Procs. of EMSOFT*. LNCS, Springer, January 2002, pp. 108–122.

[5] M. Lee and P. R. D'Argenio, "Describing secure interfaces with interface automata," *Electron. Notes Theor. Comput. Sci.*, vol. 264, no. 1, pp. 107–123, 2010.

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

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

[8] L. de Alfaro and T. A. Henzinger, "Interface automata," in *ESEC / SIGSOFT FSE*. ACM Press, 2001, pp. 109–120.

[9] R. J. V. Glabbeek, "The linear time - branching time spectrum i. the semantics of concrete, sequential processes," in *In Handbook of Process Algebra*. Elsevier, 2001, pp. 3–99.

[10] J.-C. Fernandez and L. Mounier, ""On the fly" verification of behavioural equivalences and preorders," in *Procs. of CAV '91*, ser. LNCS, vol. 575. Springer, 1991, pp. 181–191.

[11] F. Cassez, J. Mullins, and O. H. Roux, "Synthesis of non-interferent systems," in *MMM-ACNS'07*, ser. Comm. in Comp. and Inform. Sc., vol. 1. Springer, 2007, pp. 307–321.

[12] G. Gardey, J. Mullins, and O. H. Roux, "Non-interference control synthesis for security timed automata," *Electr. Notes Theor. Comput. Sci.*, vol. 180, no. 1, pp. 35–53, 2007.

[13] G. Benattar, F. Cassez, D. Lime, and O. H. Roux, "Synthesis of non-interferent timed systems," in *FORMATS*, ser. Lecture Notes in Computer Science, J. Ouaknine and F. W. Vaandrager, Eds., vol. 5813. Springer, 2009, pp. 28–42.

[14] S. Hashemian and F. Mavaddat, "A graph-based approach to web services composition," in *Applications and the Internet, 2005. Proceedings. The 2005 Symposium on*. IEEE, 2005, pp. 183–189.

[15] M. Dumas, M. Spork, and K. Wang, Eds., *Adapt or Perish: Algebra and Visual Notation for Service Interface Adaptation*, ser. LNCS, vol. 4102. Springer, 2006.

[16] K. Khan, J. Han, and Y. Zheng, "A framework for an active interface to characterise compositional security contracts of software components," in *Procs of ASWEC '01*. Washington, DC, USA: IEEE Computer Society, 2001, p. 117.