Semantics for Interactive Sequential Systems and Non-Interference Properties

Matias Lee
Universidad Nacional de Córdoba, Fa.M.A.F. - CONICET,
Córdoba, Argentina,
lee@famaf.unc.edu.ar

and

Pedro R. D'Argenio
Universidad Nacional de Córdoba, Fa.M.A.F. - CONICET,
Córdoba, Argentina,
dargenio@famaf.unc.edu.ar

Abstract

An *interactive system* is a system that allows communication with the users. This communication is modeled through input and output actions. Input actions are controllable by a user of the system, while output actions are controllable by the system. Standard semantics for sequential system [1, 2] are not suitable in this context because they do not distinguish between the different kinds of actions. Applying a similar approach to the one used in [2] we define semantics for interactive systems. In this setting, a particular semantic is associated with a *notion of observability*. These notions of observability are used as parameters of a general definition of non-interference. We show that some previous versions of the non-interference property based on traces semantic, weak bisimulation and refinement, are actually instances of the observability-based non-interference property presented here. Moreover, this allows us to show some results in a general way and to provide a better understanding of the security properties.

Keywords: process theory, semantic, interactive systems, interface automata, non interference, secure information flow, refinement, composition.

1 Introduction

An *interactive system* is a system that allows communication with the users. Usually, to carry out this communication, the system provides an interface that is used by them. Through the interface, the user sends messages to the system and receives messages from it. *Interface Automata (IA)* [3, 4, 5] is a light-weight formalism that captures the temporal aspects of interactive system interfaces. In this formalism, the messages sent by the user are represented as *input actions*, while the received messages are represented as *output actions*.

Interface structure for security (ISS) [6] is a 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 [7]. In the setting of ISS, bisimulation based notion of non-interference has been considered, more precisely, the so called BSNNI and BNNI properties [8]. 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 [9] it was argued that the BSNNI/BNNI properties are not quite appropriate to formalize the concept of *secure interface*. To illustrate this point the following two examples are presented: 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. On the other hand, the second example (Figure 2) shows that weak bisimulation based security properties may fail to detect an information leakage through input transitions.

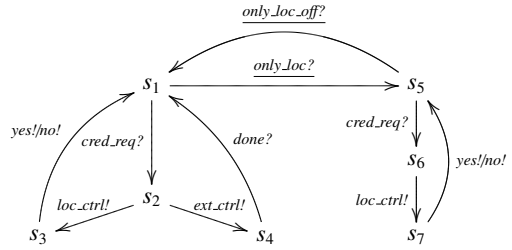

Figure 1: Credit approval process of an on-line banking service

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 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?*.

The Credit Request does not satisfy the BSNNI property (nor the BNNI property) and hence it is considered insecure in this setting. 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.

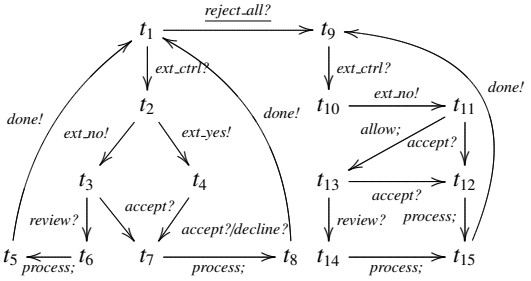

Figure 2: External Control Process in an on-line banking service

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.

These observations are based on the fact that input and output actions are conceptually very different. Input actions are controllable by the user while output actions are controllable by the system. Therefore, some behavior one would expect from input actions may be inappropriate for outputs and vice-versa. For instance, the assumption that *"wrong inputs will be rejected/ignored; otherwise, they will be accepted"* in the second example above, makes no sense if applied to outputs because the malicious user is interested in collecting all possible information rather than in rejecting it.

In [1] and [2], a deep study about semantic for sequential system is done, but they do not take in account systems where both kinds of actions coexist. In their setting all actions are controlled by one entity: the user or the system. For example, in *Fail Trace Semantic* a user executes (input) actions until one action is rejected by the system, in this case the user has the control of which action is executed. A different case is *Trace Semantic* where the system has the control of the actions and the user can only observe the executions of the system. Also in stronger semantics, for example with *global testing*, the control belongs to one entity. For instance, *Weak Bisimulation equivalence* is also called *observational equivalence* and its intuitive notion is "two system are observational equivalence if they cannot be distinguished by *an observer*", ie the user observes and the system executes (controls) the actions. Notice the subtlety in this case: global testing allows the user to force the system to execute all possible executions but, which actions can be executed in each state is controlled/defined by the system.

In this work we define semantics for systems where both coexist, actions controlled by the user (input actions) and actions controlled by the system (output actions). We have used an approach similar to the one used in [2]. First we define *types of observations*, an information record that can be performed by a user. Second, we define a *notion of observability* as a set of types of observations. Each notion of observability is a particular semantic. This approach is simple, elegant and allows to be exhaustive: when the types of observation and notion of observability are defined one has all the possible semantics that could be defined.

These new semantics are suitable to study secure information flow properties over ISS. Moreover, the definition of non-interference presented in this work has as parameter a notion of observability. This generalization through types of observations provides a framework to prove generic theorems that extends to families of security properties. In addition, the approach subsumes previous definitions of non-interference for ISS, in particular the one based on traces [9], the one based on weak bisimulation [6] and the one based on refinement [9].

We also focus our attention in non-interference based on refinement. We give sufficient and simple conditions to ensure compositionality. 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. These results are relevant because they could be adapted to other instances of non interference based on notion of observability.

This paper is an extension of [9]. In [9] we introduce non-interference based on refinement to resolve some shortcomings in the non-interference based on weak bisimulation properties. The approach based on notions of observability shows that the shortcomings do not exist because the properties should be considered in different contexts. We explain this in the last section of the paper.

Organization of the paper. In section 2 we recall definitions of IA, composition and ISS. In section 3 we define the types of observations, notion of observability and the set of observable behaviors of an IA. In section 4 we present the notion of non-interference based on notion of observability. We show that the approach subsumes previous definition of non-interference for ISS and we proof some general properties of non-interference. In section 5 we review the definitions of non-interference based on refinement, and we show that these definitions also are subsumed by the new approach. We study compositionality in this setting and define two algorithms: one to check whether an interface

satisfies the property and the another to derive a secure interface from a given (non-secure) interface by controlling inputs actions. Section 6 concludes the paper.

2 Interfaces Automata and Interface Structure for Security

In the following, we define *Interface Automata* (IA) [3, 4] and *Interface Structure for Security* (ISS) [6], and introduce some notations.

2.1 Interfaces Automata

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.

Composition

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. *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* $\mathit{shared}(S, T) = (A^I_S \cap A^O_T) \cup (A^O_S \cap A^I_T)$*. 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 3. *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* $q^0_{S\otimes T} = (q^0_S, q^0_T)$*;*
- $A^I_{S\otimes T} = A^I_S \cup A^I_T - \mathit{shared}(S, T)$*,* $A^O_{S\otimes T} = A^O_S \cup A^O_T - \mathit{shared}(S, T)$*, 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 these are called *error states*.

Definition 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, 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 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 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 $S \otimes T$.*

2.2 Interface Structure For Security

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 7. *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 $m \in \{h, l\}$.

Extending the definition of composition of IA to ISS is straightforward.

Definition 8. *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 ISS. $\mathcal{S}$ and $\mathcal{T}$ are* composable *if S and T are composable. Given two composable ISS, $\mathcal{S}$ and $\mathcal{T}$, their* composition*, $\mathcal{S} \parallel \mathcal{T}$, is defined by the ISS $\langle S \parallel T, (A^h_S \cup A^h_T) - shared(S, T), (A^l_S \cup A^l_T) - shared(S, T)\rangle$.*

3 Observability

Semantic equivalences for sequential systems with silent moves are studied in [2]. Resulting in 155 notions of observability and a complete comparison between them. Unfortunately, these results cannot be applied straightforward to the IA context. For example, studied machines in [2] have not notions of input and output actions over the same machine. Moreover, in [2] there is not a notion of the internal structure of the analyzed machine. This situation have forced them to talk about *definite* and *hypothetical* behaviors of the machine. Despite these differences, we use [2] as a reference to define different semantics for IA. To avoid the distinction between definite and hypothetical behaviors, we use the transition relation of the IA to present the set of observable behaviors.

First we define *type of observation*, an information record that can be done by the user. Second, we define a *notion of observability* as a set of types of observations. Each notion of observability defines a particular semantic. Third, using the transition relation of the IA, we define the semantic of each type of observation and therefore a semantic for each possible notion of observability.

Given a system, a *type of observation* is an information that can be recorded by a user with respect to the interface. To define our types of observations we consider the following assumptions: input and output actions are observable when they are executed. Inputs are executed by a user, while outputs are executed by the interface. Then, input actions are controllable by the user and output actions are controllable by the interface. Internal transitions are controllable by the interface. In some cases, internal transitions can be detectable by the user but the user cannot distinguish between different internal actions. An user can observe how the interface interact with another user or he can be the one who interacts. If the user is interacting, the interface can behave in different ways as a result of some violation of its input assumptions: (*i*) it does not show any error and continues with the execution, (*ii*) it stops the execution and shows an error to the user, (*iii*) it shows an error to the user and continues with the execution; (*iv*) finally, an interface could provide a special service to inform which inputs are enabled in its current state. In this way, the user can avoid input assumption violations. Notice that cases (*i*), (*ii*) and (*iii*) determine, at the semantic level, a sort of input-enableness. In these cases we fix the behavior of input actions that are not defined in a particular state. The last four assumptions do not increase the expressiveness power of the model, as consequence they can be implemented in any IA. For example: let S be an IA, the assumption (*i*) can be implemented with self loops with action $a?$ for all state state $s \in Q_S$ and $a? \in A^I - I(s)$. Using the same reasoning, we assume an interface could provide a service to detect the end of an execution, where the end is reached when no more transitions are possible. In addition, a user can make copies of the interface with the objective of studying the interface in more detail. Finally, a user can do *global testing*. Under this assumption it is possible to say that a particular observation will not happen.

Based on these assumptions, we introduce the following types of observations:

a	The execution of external actions $A^I \cup A^O$ are detectable.
$\varepsilon, \not\varepsilon$	The case of internal transitions are detectable is denoted with ε. Otherwise $\not\varepsilon$.
T	The session is terminated by the user. This is possible in any time. After this no more records are possible
$\not\rightleftarrows$, $\rightleftarrows$	If a user only observes the actions that are executed by an interface and cannot send stimuli to it, then there is no interaction. We denote this with $\not\rightleftarrows$. The case where the interaction is possible is denoted by $\rightleftarrows$.
F	The user interacts with system and the interface stops the execution whenever it receives an input action that is not enabled. In this case, the stop is observable.
FT	Suppose the previous type but now whenever the interface receives an input action that is not enabled, the error is informed to the user and the execution continues.
RT	To avoid the error of sending an input action that is not enabled, the interface can provide a method to check what input actions are enabled in its current state. In this case, the observation includes the set X of enabled inputs.
0	This type is used if it is detectable when an interface reachs a final state, i.e. no more activity is possible.
$\bigwedge$	Suppose the user has a machine to make arbitrary number of copies of the system. These copies reveal more information about the interface because one could observes different execution from the same interface. If the user makes N copies and in each copy executes ϕ_i for $i \in \{1, \ldots, N\}$, this observations is denoted with $\bigwedge_{i=1}^{n} \phi_i$.
$\neg$	It is possible to test the interface over all possible condition. This allows to ensure that a particular observation is not possible; then a user can do an observation $\neg\phi$ whenever ϕ is not possible execution of the system.

The types of observations studied here are not the studies in [2]. On one hand, we decided to skip some types for the sake of simplicity. For example we did not include η-replication nor *continuous copying*, which are different forms of make copies of the system. We did not include the notion of *stable state*, this avoids the inclusion of some variant of types of visibilities presented here. On the other hand, we have added new features. First, we differentiate between a user that interacts with the interface and a non-interacting user. Second, the knowledge of the internal structure of the interface allow us to know exactly when an internal action could be executed and define if the internal transitions are observable or not. This is a relevant feature in the context of security, because it could be used to represent covert channels.

A set of types of observations defines a *notion of observability*, see Definition 9. The notion of observability determines what information can be observed by a user. This has to be consistent, for example, types of observations "a user cannot interact with the interface" ($\not\rightleftarrows$) and "a user can detect that the input sent was not enabled" (F) cannot belong to same notion of observability. Note that the definition of notion of observability ensures consistency.

Definition 9. *A set V is a* notion of observability (for IA) *if $V \subseteq \{a, \varepsilon, \not\varepsilon, 0, \rightleftarrows, \not\rightleftarrows, T, F, FT, RT, \bigwedge, \neg\}$ and V satisfies the following conditions:*

1. $\{a, T\} \subseteq V$,
2. $|\{\varepsilon, \not\varepsilon\} \cap V| = 1$,
3. $|\{\rightleftarrows, \not\rightleftarrows, F, FT, RT\} \cap V| = 1$.

Condition (1) ensures that input and output actions are always visible and that the user can terminate the session when he wants. Condition (2) ensures that internal transitions are detectable or not. Condition (3) ensures that a user can interact with the interface ($\rightleftarrows, F, FT, RT$) or not ($\not\rightleftarrows$), and if he interacts, he will do in one particular way.

In [2] other kind of restrictions were added to simplify the study of which semantics make more differences: for example conditions as *"if $FT \in V$ then $F \in V$"* are added. This reflects the fact that if the interface stops when a disable input is received, all observations that one can do in this scenario, can be done in the same machine configured to continue when the error occurs. Since we are not interested in studying which semantics is coarser than others, we omit these conditions.

Semantic. First we define all possible observations as a set of logic formulas called *execution formulas*. Then the set of *observable behavior* of an IA is the set of execution formulas that are satisfied by the initial state of the interface.

Definition 10. *The set of* execution formulas $\mathcal{L}$ *for an IA $S = \langle Q, q^0, A^I, A^O, A^H, \rightarrow\rangle$ is the smallest set satisfying rules in Table 1.*

$$T \in \mathcal{L} \qquad 0 \in \mathcal{L} \qquad \not{a} \in \mathcal{L} \quad \forall a \in A^I \qquad \frac{\phi \in \mathcal{L} \quad a \in A^I \cup A^O \cup \{\varepsilon\}}{a\phi \in \mathcal{L}}$$

$$\frac{\phi \in \mathcal{L} \quad a \in A^I}{\not{a}\phi \in \mathcal{L}} \qquad \frac{\phi \in \mathcal{L} \quad X \subseteq A^I}{X\phi \in \mathcal{L}} \qquad \frac{\phi_i \in \mathcal{L} \quad i \in I}{\bigwedge_{i\in I} \phi_i \in \mathcal{L}} \qquad \frac{\phi \in \mathcal{L}}{\neg\phi \in \mathcal{L}}$$

Table 1: Recursive rules for definition of execution formulas.

(T)	$q \models T$	$\forall q \in Q$
(0)	$q \models 0$	if $q \overset{q}{\not\to}$ for all $a \in A$
(a)	$q \models a\phi$	if $a \in A^I \cup A^O$ and $\exists q' \in Q$: $q \xrightarrow{a} q'$ and $q' \models \phi$
$(\not{a})$	$q \models \phi$	if $a \in A^H$ and $\exists q' \in Q$: $q \xrightarrow{a} q'$ and $q' \models \phi$
(ε)	$q \models \varepsilon\phi$	if $a \in A^H$ and $\exists q' \in Q$: $q \xrightarrow{a} q'$ and $q' \models \phi$
$(\rightleftarrows)$	$q \models a\phi$	if $a \in A^I - I(q)$ and $q \models \phi$
(F)	$q \models \not{a}$	if $a \in A^I - I(q)$
(FT)	$q \models \not{a}\phi$	if $a \in A^I - I(q)$ and $q \models \phi$
(RT)	$q \models X\phi$	if $X = I(q)$ and $q \models \phi$
$(\bigwedge)$	$q \models \bigwedge_{i\in I} \phi_i$	if $q \models \phi_i$ for all $i \in I$
$(\neg)$	$q \models \neg\phi$	if $q \not\models \phi$

Table 2: Semantic of the observations

Definition 11. *Given an IA* $S = \langle Q, q^0, A^I, A^O, A^H, \to\rangle$ *and a notion of observability V, the* satisfaction relation $\models_V \subseteq Q \times \mathcal{L}$ *is defined for each type of observation in V by clauses in Table 2. The* observables behavior of an IA S with notion of observability V *is* $O_V(S) = \{\phi \in \mathcal{L} : q^0 \models_V \phi\}$

4 Non interference based on Notion of Observability.

First we introduce a general notion of non-interference. Informally, non-interference states 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. Hence, restriction and hiding are central to our definitions of security.

Definition 12. *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* $S\backslash X = \langle Q_S, q^0_S, A^I_S - X, A^O_S - X, A^H_S, \to_{S\backslash X}\rangle$ *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* $S/X = \langle Q_S, q^0_S, A^I_S - X, A^O_S - X, A^H_S \cup X, \to_S\rangle$.

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

Definition 13. *Let* $\mathcal{S} = \langle S, A^h, A^l\rangle$ *be an ISS and V a notion of observability, then:*

- $\mathcal{S}$ *is V strong non-deterministic non-interference (V-SNNI) if* $O_V(S/A^h) = O_V(S\backslash A^h)$.
- S *is V non-deterministic non-interference (V-NNI) if* $O_V(S/A^h) = O_V((S\backslash A^{h,I})/A^{h,O})$.

Notice the difference between the two definitions. V-SNNI formalizes the security property as we described so far: a system satisfies V-SNNI if a low-level user cannot distinguish (up to notion of observability V) 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). In the definition of V-NNI 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.

The approach of non-interference based on notion of observability generalizes other notion of non-interference for IA. For example *Non deterministic Non-Interference* (NNI), *Strong Non deterministic Non-Interference* (SNNI), both based on trace equivalence; *Bisimulation NNI* (BNNI) and *Bisimulation SNNI* (BSNNI) both based on bisimulation equivalence. To prove our statement, we recall the definitions of *trace equivalence, weak bisimulation* and non-interference properties.

Definition 14. *Let S and T be two IA. S and T are* trace equivalent*, notation $S \approx_T T$, if $Traces(S) = Traces(T)$. We say that two ISS $\mathcal{S}$ and $\mathcal{T}$ are trace equivalent, and write $\mathcal{S} \approx_T \mathcal{T}$, whenever the underlying IA are trace equivalent.*

Definition 15. *Let S and T be two IA. A relation $R \subseteq Q_S \times Q_T$ is a* (weak) bisimulation *between S and T if $s_0 \; R \; t_0$ and, for all $s \in Q_S$ and $t \in Q_T$, $s \; R \; t$ implies:*

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

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.*

Definition 16. *Let $\mathcal{S} = \langle S, A^h, A^l \rangle$ be an ISS.*

1. *$\mathcal{S}$ satisfies* strong non-deterministic non-interference (SNNI) *if $\mathcal{S}\backslash A^h \approx_T \mathcal{S}/A^h$.*
2. *$\mathcal{S}$ satisfies* non-deterministic non-interference (NNI) *if $\mathcal{S}\backslash A^{h,I}/A^{h,O} \approx_T \mathcal{S}/A^h$.*
3. *$\mathcal{S}$ satisfies* bisimulation-based strong non-deterministic non-interference (BSNNI) *if $\mathcal{S}\backslash A^h \approx \mathcal{S}/A^h$.*
4. *$\mathcal{S}$ satisfies* bisimulation-based non-deterministic non-interference (BNNI) *if $\mathcal{S}\backslash A^{h,I}/A^{h,O} \approx \mathcal{S}/A^h$.*

We prove how to represent these notions of security with notions of observability.

Theorem 1. *Let $\mathcal{S} = \langle S, A^h, A^l \rangle$ be an ISS then*

1. *$\mathcal{S}$ is (S)NNI iff $\mathcal{S}$ is V-(S)NNI with $V = \{a, T, \not\in, \not\in\!\!\!\!-\}$.*
2. *$\mathcal{S}$ is B(S)NNI iff $\mathcal{S}$ is V-(S)NNI with $V = \{a, T, \not\in, \not\in\!\!\!\!-\;, \bigwedge, \neg\}$.*

Proof. First we prove (2). For this, we have to show that for all states $s \in Q_S$ and $t \in Q_T$ it holds $s \approx t$ iff $O_V(s) = O_V(t)$. ($\Rightarrow$) Suppose $s \approx t$ and $\phi \in O_V(s)$. Let $f : \mathcal{L} \to \mathbb{N}$ a function defined as:

$$f(T) = f(0) = f(\not\phi) = 0 \qquad f(a\phi) = f(\not\phi\phi) = f(X\phi) = f(\neg\phi) = f(\phi) + 1 \qquad f(\textstyle\bigwedge_i \phi_i) = \max_i(f(\phi_i)) + 1 \qquad (*)$$

We define f in general for all $\mathcal{L}$ since we will make use of it again later. We proceed by complete induction. In the base case $f(\phi) = 0$ then $\phi = T$ because $V = \{a, T, \not\in, \not\in\!\!\!\!-\;\; \bigwedge, \neg\}$ and since T is an observation for every state $\phi \in O_V(t)$. By induction suppose that if $s \approx t$ then, if $f(\phi) \leq k$ and $\phi \in O_V(s)$ it holds $\phi \in O_V(t)$. Let $f(\phi) = k + 1$, we do case analysis according to the shape of the formula. Suppose $\phi = a\phi'$ with $a \in A^I \cup A^O$. $s \models a\phi$ implies $s \stackrel{a}{\Rightarrow} s'$ and $\phi \in O(s')$ (see (a) and ($\not\in$) in Table 2). Since $s \approx t$ there is state t' such that $s' \approx t'$. By induction $\phi' \in O_V(t')$, therefore $a\phi' \in O_V(t)$. Now let $\phi = \bigwedge_i \phi_i$. Since $f(\phi_i) \leq k$ for all i, by induction $\phi_i \in O_V(t)$. Therefore $\phi = \bigwedge_i \phi_i \in O_V(t)$. Now suppose $\phi = \neg\phi'$ then $f(\phi') = k$ and $s \not\models \phi'$, by induction $t \not\models \phi'$. Therefore $t \models \neg\phi'$, ie $\phi \in O_V(t)$. The other cases are outside of the observation defined by V. The symmetric case is analogous.

($\Leftarrow$) Let $O_V(s) = O_V(t)$ and $s \stackrel{a}{\to} s'$. We have to show that there is t' such that $t \stackrel{a}{\Rightarrow} t'$ and $O_V(s') = O_V(t')$. Since $O_V(s) = O_V(t)$ we have $t \stackrel{a}{\Rightarrow}$. Let Q be $\{t' : t \stackrel{a}{\Rightarrow} t'\}$. If for all $t' \in Q$ it holds $O_V(s') \neq O_V(t')$ then there is $\phi_{s'} \in O_V(s') - O_V(t')$ (as consequence of ($\neg$)). Then for any $t' \in Q$ it holds $\bigwedge_{q \in Q} \phi_q \in O_V(s') - O_V(t')$ (at least one ϕ_q fails). But then $a \bigwedge_{q \in Q} \phi_q \in O_V(s) - O_V(t)$ contradicting $O_V(s) = O_V(t)$. The symmetric case is analogous.

To prove (1) we show that given two IA S and T it holds $S \approx_T T$ iff $O_V(S) = O_V(T)$. We reduce this to prove $\phi \in Traces(S)$ iff $\phi T \in O_V(S)$. This proof is straightforward. □

The relation between V-SNNI and V-NNI depends on the notion observability V. In general, we only can ensure V-NNI is not stronger than V-SNNI for all V.

Theorem 2. *For all notion of observability V there is an ISS $\mathcal{S}$ such that $\mathcal{S}$ is V-NNI and $\mathcal{S}$ is not V-SNNI.*

Proof. Let $\mathcal{S}$ the following ISS $s_0 \xrightarrow{H!} s_1 \xrightarrow{a} s_2$ with $a \in A^I \cup A^O$. Notice $\mathcal{S}$ is always V-NNI. On the other hand $\mathcal{S}$ is not V-SNNI: if $\not\in \in V$ then $aT \in O_V((S/A^h))$ and $aT \notin O_V((S\backslash A^h))$; if $\varepsilon \in V$ then $\varepsilon aT \in O_V((S/A^h))$ and $\varepsilon aT \notin O_V((S\backslash A^h))$. □

This result is not novel. In [8], it is shown that SNNI is stronger than NNI. Therefore as trace semantic is the coarsest sensible semantic on labeled transition system, it is natural that the result holds for all other semantic. The Theorem 2 only formalize this fact for IA semantics.

The other relations depend on V and we state in the following two theorems. Previously an auxiliary lemma.

Lemma 1. *Let S be an IA and V a notion of observability such that $\{0, \neg\} \cap V = \emptyset$. Let S' be an IA obtained by removing a set of internal transitions from S. Then $O(S) \supseteq O(S')$.*

Proof. The proof is straightforward by induction in $f(\phi)$ where $\phi \in O_V(S')$ and f is the function defined in $(*)$. □

Theorem 3. *Let $\mathcal{S}$ be an ISS and V a notion of observability such that $\{0, \neg\} \cap V = \emptyset$. If $\mathcal{S}$ is V-SNNI then $\mathcal{S}$ is V-NNI.*

Proof. If $\mathcal{S}$ is V-SNNI then $O_V(S/A^H) = O_V(S \setminus A^H)$. Notice $S \setminus A^H$ is obtained by removing some hidden transitions from $(S \setminus A^{h,I})/A^{h,O}$, then $O_V((S \setminus A^{h,I})/A^{h,O}) \supseteq O_V(S \setminus A^H)$ by Lemma 1, and therefore $O_V((S \setminus A^{h,I})/A^{h,O}) \supseteq O_V(S/A^H)$. On the other hand $(S \setminus A^{h,I})/A^{h,O}$ is obtained by removing some hidden transitions from S/A^H then $O_V(S/A^H) \supseteq O_V((S \setminus A^{h,I})/A^{h,O})$ by Lemma 1. Both inclusions imply $O_V(S/A^H) = O_V((S \setminus A^{h,I})/A^{h,O})$. □

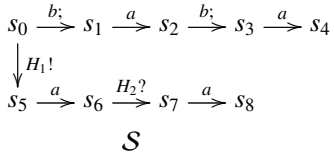

Figure 3: $\mathcal{S}$ is V-SNNI does not imply $\mathcal{S}$ is V-NNI if $V \cap \{0, \neg\} \neq \emptyset$

Theorem 4. *For all notion of observability V such that $V \cap \{0, \neg\} \neq \emptyset$ there is an ISS $\mathcal{S}$ such that $\mathcal{S}$ is V-SNNI and $\mathcal{S}$ is not V-NNI.*

Proof. Define $\mathcal{S}$ as ISS in Figure 3 with $a \in A^I \cup A^O$. Clearly $\mathcal{S}$ is V-SNNI for all V. Suppose $\not\in V$: if $\neg \in V$ then $a\neg a \in O_V((S \setminus A^{h,I})/A^{h,O})$ while $a\neg a \notin O_V(S \setminus A^H)$; if $0 \in V$ then $a0 \in O_V((S \setminus A^{h,I})/A^{h,O})$ while $a0 \notin O_V(S \setminus A^H)$. Then $\mathcal{S}$ is not V-NNI for any V such that $V \cap \{0, \neg\} \neq \emptyset$. The case $\varepsilon \in V$ is analogous. □

The approach based on notion of observability also allows to show that security properties are not preserved by composition.

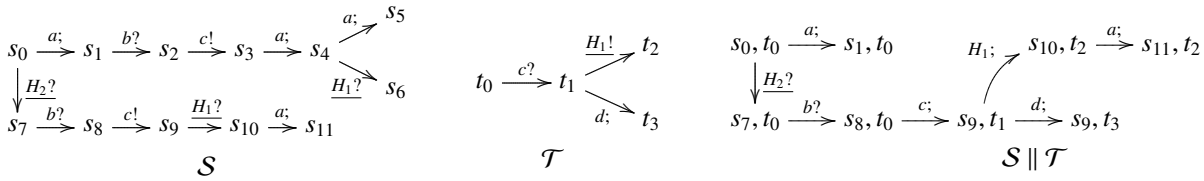

Figure 4: V-SNNI and V-NNI properties are not preserved by composition.

Theorem 5. *For all notion of observability V there are ISS $\mathcal{S}$ and $\mathcal{T}$ such that $\mathcal{S}$ and $\mathcal{T}$ are V-(S)NNI and composable, and the composition $\mathcal{S} \parallel \mathcal{T}$ is not V-(S)NNI.*

Proof. Let $\mathcal{S}$ and $\mathcal{T}$ be ISS depicted in Figure 4. Both interfaces are V-(S)NNI for all notion of observability V but $\mathcal{S} \parallel \mathcal{T}$ is not. If $\not\in V$ then $b?T \in O_V((\mathcal{S} \parallel \mathcal{T})/A^h)$ while if $\varepsilon \in V$ then $\varepsilon b?T \in O_V((\mathcal{S} \parallel \mathcal{T})/A^h)$. In any case, $O_V((\mathcal{S} \parallel \mathcal{T})\setminus A^h) = O_V(((\mathcal{S} \parallel \mathcal{T})\setminus A^{h,I})/A^{h,O})$ and $b?T, \varepsilon b?T \notin O_V((\mathcal{S} \parallel \mathcal{T})\setminus A^h)$. Then $\mathcal{S} \parallel \mathcal{T}$ is not V-(S)NNI. □

5 Non-interference based on refinement.

In [9], we presented definitions of non interference based on refinement. The new versions of non-interference were introduced to solve some shortcomings detected in the definitions of non interference based on bisimulation of [6], ie BSNNI and BNNI. In this section we review the results obtained.

To address the shortcomings detected in B(S)NNI properties, a variation of non-interference based on refinement was introduced. 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; in addition, 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:

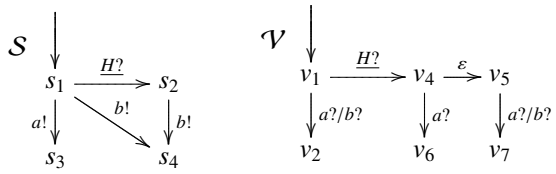

Figure 5: In these interfaces, BSNNI and BNNI are not appropriate properties to denote security.

Definition 17. *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 $\exists q_S' \in Q_S : q_S \overset{\varepsilon}{\Rightarrow}_S \xrightarrow{a}_S q_S'$ and $q_S' \succcurlyeq q_T'$;
(d) $\forall a \in A_T^H, q_T' \in Q_T$, if $q_T \xrightarrow{a}_T q_T'$ then $\exists q_S' \in Q_S : q_S \overset{\varepsilon}{\Rightarrow}_S q_S'$ 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 [5] only that restriction (b) is new with respect to the original version. Based on this relation are defined non-interference properties based on refinement. They are called SIR-NNI and SIR-SNNI.

Definition 18. *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 $\mathcal{S}/A^h$.

Now we show there is a notion of observability V such that V-(S)NNI is equivalent to SIR-(S)NNI. To prove the result we need the following theorem:

Theorem 6. *Given two IA S and T, S is refined strictly on inputs by T, ie $S \succcurlyeq T$ iff $O_V(S) \supseteq O_V(T)$ with $V = \{a, T, \not\in, RT, \bigwedge\}$.*

Proof. For this, we have to show that for all states $s \in Q_S$ and $t \in Q_T$ it holds $s \succcurlyeq t$ iff $O_V(s) \supseteq O_V(t)$. ($\Rightarrow$) Suppose $s \succcurlyeq t$ and $\phi \in O_V(t)$. Let $f : \mathcal{L} \to \mathbb{N}$ the function defined in ($*$). We proceed by complete induction. In the base case $f(\phi) = 0$ then $\phi = T$ because $V = \{a, T, \not\in, RT, \bigwedge\}$ and since T is an observation for every state, then $\phi \in O_V(s)$. *Inductive case*. By induction suppose that if $s \succcurlyeq t$ then, if $f(\phi) \leq k$ and $\phi \in O_V(t)$ it holds $\phi \in O_V(s)$. Let $f(\phi) = k + 1$, we do case analysis according to the shape of the formula. Suppose $\phi = X\phi'$. Since $t \models X\phi'$ then $t \models \phi'$. Moreover, $s \succcurlyeq t$ implies $I(s) = I(t)$ and therefore $s \models X\phi'$ using induction. Cases $a\phi'$ and $\bigwedge_i \phi_i$ are like this respective case in proof of Theorem 1.

($\Leftarrow$) Let $O_V(s) \supseteq O_V(t)$. Case $t \xrightarrow{a?} t'$: we have to show there is s' such that $s \xrightarrow{a?} s'$ and $O_V(s') \supseteq O_V(t')$. If $s \overset{a?}{\not\to}$ then $I(s) \neq I(t)$ and therefore $O_V(s) \not\supseteq O_V(t)$ because $t \models I(t)T$ and $s \not\models I(t)T$. Let s' such that $s \xrightarrow{a?} s'$, notice s' is unique because IA are input deterministic. If $O_V(s') \not\supseteq O_V(t')$ there is $\phi' \in O_V(t') - O_V(s')$. This implies $a?\phi' \in O_V(t) - O_V(s)$ and we get a contradiction. In the case $s \xrightarrow{a?} s'$, we have to show there is t' such that $t \xrightarrow{a?} t'$ and $O_V(s') \supseteq O_V(t')$, this proof is similar to the previous one. Let now $t \xrightarrow{a!} t'$, we have to show there is s' such that $s \Rightarrow\xrightarrow{a!} s'$ and $O_V(s') \supseteq O_V(t')$. Let Q be $\{s' : t \Rightarrow\xrightarrow{a} s'\}$. If for all $s' \in Q$ it holds $O_V(s') \not\supseteq O_V(t')$ then there is $\phi_{s'} \in O_V(t') - O_V(s')$. Then for any $s' \in Q$ it holds $\bigwedge_{q \in Q} \phi_q \in O_V(t') - O_V(s')$ (at least one ϕ_q fails). But then $a \bigwedge_{q \in Q} \phi_q \in O_V(t) - O_V(s)$ contradicting $O_V(s) \supseteq O_V(t)$. Case $t \xrightarrow{a;} t'$ is analogous. □

Now we are able show the statement.

Lemma 2. *An IA $\mathcal{S}$ is SIR-(S)NNI iff $\mathcal{S}$ is $\{a, T, \not\in, RT, \bigwedge\}$-(S)NNI.*

Proof. If $\mathcal{S}$ is SIR-SNNI then $S\backslash A^h \succcurlyeq S/A^h$. By Theorem 6 we have $O_V(S\backslash A^h) \supseteq O_V(S/A^h)$. On the other hand, by Lemma 1 we have $O_V((S/A^h)) \supseteq O_V((S\backslash A^h))$. Finally $O_V((S/A^h)) = O_V((S\backslash A^h))$. The case $\mathcal{S}$ is SIR-NNI is analogous. □

Two properties about SIR-NNI and SIR-SNNI were introduced in [9]. The first one, if an ISS is SIR-(S)NNI then it is (S)NNI. This is straightforward using their respective equivalent definition with notion of observability, ie $\{a, T, \not\in, RT, \bigwedge\}$-(S)NNI and $\{a, T, \not\in, \not\in\!\!\!\!-\}$-(S)NNI. The second one, if an ISS is SIR-SNNI then it is SIR-NNI. This is a particular case of Theorem 3.

5.1 Composition

Theorem 5 shows that non-interference properties are not preserved for all notion of observation V. This implies SIR-SNNI and SIR-NNI properties are not preserved by the composition.

Despite this, we give sufficient conditions to ensure that the composition of ISS results in a non-interferent ISS (always with respect to SIR-SNNI and SIR-NNI). 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 7. *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 such that $\mathit{shared}(S,T) \cap (A_S^h \cup A_T^h) = \emptyset$. 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 (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 17. 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 7.

Corollary 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 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.

5.2 Deriving Secure Interfaces

As we have seen, the composition of secure interfaces may yield a new insecure interface. This may happen when the components are already available but they were 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. 6); 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 [6] 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 $\stackrel{a}{\Rightarrow\!\!\rightarrow}$; *(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 8). 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 9).

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 19. *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:*

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

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

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 $q \stackrel{a}{\rightarrow} q'$ to the model for each weak transition $q \stackrel{a}{\Rightarrow} q'$, 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 $q \stackrel{a}{\rightarrow} q'$ whenever $q \stackrel{a}{\Rightarrow\!\!\rightarrow} q'$ with a an output action. We call this process *semi-saturation*.

Definition 20. *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:*

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

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 $q \stackrel{a}{\Rightarrow\!\!\rightarrow}_S q'$ iff $q \stackrel{a}{\rightarrow}_{\overline{S}} q'$.

Following [6] 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 $S \succcurlyeq T$ then for two states $s \in Q_S$ and $t \in Q_T$ s.t. $s \succcurlyeq t$, 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 21. *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:*

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

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

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* 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 $q_S \not\succcurlyeq q_T$. 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 $q_S \not\succcurlyeq q_T$. 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 $q_S \succcurlyeq q_T$, 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* $\cup$ *May*, belong to the set *Pass*. All pairs in *Pass* are related by a SIR relation.

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

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

Table 3: 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 4: The definition of *May* set .

Definition 22. *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 3. If* $q \in Fail$*, we say that the pair* q fails the SIR relation test.
- $May = \cup_{i=0}^{\infty} May^j$ *where* May^i *is defined in Table 4. 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.

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

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 $q_S \not\succcurlyeq q_T$ and *(ii)* if $(q_S, q_T) \in Pass$ then $q_S \succcurlyeq q_T$. 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 $(s', t') \in Pass$.

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

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

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

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

Table 5: Set of eliminable candidates.

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 8. *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 $EC(\mathcal{S}) \subseteq \to \cap\, Q \times A^I \times Q$ (see Table 5) 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* ∪ *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. $\to_{\mathcal{S}'} = \to_{\mathcal{S}} - \{q \xrightarrow{a} q'\}$, 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].

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

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

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

(*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 $q_r \not\xrightarrow{q}$, 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 9. *Let $\mathcal{S}$ be an ISS such that P_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 [6]. 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. □

6 Concluding remarks

In this work, we have presented semantics for interactive sequential systems. In this way we have extended the work of [1] and [2] to models where the control of the actions is shared by the user and the system. To reduce complexity, we did not include all types of observations presented in [2], thus limiting ourselves to work with a subset of them. We do not foresee major problems in extending our theory to the types of observations we left out.

The approach to define non-interference security properties through types of observations gives important insight about the security model, in particular about the characteristics of the attacker. For instance, if the attacker can make use of the covert channels, then the type of observation ε should be chosen. Another example is the type of observation F which can be interpreted as the system detecting an attack and aborting the execution. In this way, the types of observations define a catalogue to characterize the attackers that could be considered.

This general definition encloses previous definitions of non-interference for ISS. We found notions of observability to represent (S)NNI, B(S)NNI and SIR-(S)NNI (Theorem 1 and Lemma 2). This approach also provides a better understanding of the security properties. In [9], SIR-(S)NNI is introduced to resolve some shortcomings found in B(S)NNI, but in fact, these shortcomings do not exist because the properties should be considered in a different context. B(S)NNI should be considered in a context where an attacker can only observes how the system behaves. On the other hand, SIR-(S)NNI should be considered in a context where the attacker can interact through the interface. This is obvious when we see the notions of observability used to represent each property: $\{a, T, \not\in, \not\rightleftharpoons, \bigwedge, \neg\}$ for B(S)NNI and $\{a, T, \not\in, RT, \bigwedge\}$ for SIR-(S)NNI. Notice that B(S)NNI has the no interaction type ($\not\rightleftharpoons$) while in SIR-(S)NNI the interaction is explicit due to the type (RT).

In addition, the different types of observations provide a simple way to chose the appropriate notion of security. For example notice interface $\mathcal{S}$ in Figure 5. One could argue that still there is an information leakage, because the execution of action $a!$ is an evidence that the high user has not interacted with the interface. If this information is sensitive and the attacker interacts with the interface, one could use the notion of observability $V = \{a, T, \not\in, RT, \bigwedge, \neg\}$ to detect this kind of problem. Notice this notion of observability is stronger than the notion used for B(S)NNI.

Future Works. We have identified two different research lines to continue this work. At first place, the types of observations presented in [2] that have been omitted, have to be addressed, and a deep study comparing the different semantics should be carried out to get a better understanding of them. Second, we also plan to study how the new semantics for interactive systems affect the different models with both input/controllable and output/uncontrollable actions and the results obtained for them.

References

[1] 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.

[2] R. van Glabbeek, "The linear time-branching time spectrum II: The semantics of sequential processes with silent moves," in *Proceedings CONCUR*, vol. 93, pp. 66–81.

[3] 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.

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

[5] L. de Alfaro and T. A. 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.

[6] 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.

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

[8] 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.

[9] M. Lee and P. R. D'Argenio, "A refinement based notion of non-interference for interface automata: Compositionality, decidability and synthesis," in *SCCC*, 2010, pp. 280–289.

[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.