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. The first ones are controllable by an user of the system, the second ones 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 semantic for interactive system. Thus, a particular semantic is associated with a notion of observability. These notions of observability are used as a parameter 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 properties in a general way and provides 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 the user. 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.

Conceptually, input and output actions are very different. Input actions are controllable by the user while output actions are controllable by the system. For example, suppose you have an arbitrary number of copies of a system that inform when an incorrect input is received. If someone asks if the input action $a$? is accepted by the system, one can take a copy of it, send $a$? and check if the error message is executed or not. On the other hand, if someone asks if the output action $b$! is executed by the system, no matter how many copies one takes, even if you never see a system executing $b$! is not enough to ensure that the system cannot execute $b$!.

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 an 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 see 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”, i.e. 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 an 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. In this context, within a system coexist input and output actions and these actions are classified as either low (or public) action or high (or confidential) action. Low actions are intended to be accessed by any user while high actions can only be accessed by those users having the appropriate clearance. We model this kind of system with Interface structure for security (ISS) [6], a variant of IA, where visible actions are classified as either high or low. Here, the goal is to ensure that a system is secure. A system is secure if it satisfies the property called non-interference that was introduced in [7]. Informally, the property establishes that a system is secure when 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 in the same way when high actions are not performed or when high actions are considered as hidden actions. To formalize the idea of “behave in the same way” we use different notions of observability. Moreover, the definition of non-interference presented in this work has as parameter a notion of observability. This generalization and the definition of notion of observability allow to show some general results. For example, for all possible definition of non-interference based on a notion of observability, the property is not preserved by composition. In addition, the approach subsumes previous definitions of non-interference for ISS, the one based on traces [8], the one based on weak bisimulation [6] and the one based on refinement [8].

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 [8]. In [8] 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 observation, 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 also review when these properties are preserved by interface composition and algorithms to check and to derive an interface that satisfies the property. 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 \( S = (Q, q^0, A^I, A^O, A^H, \rightarrow) \) 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. \( (q, a, q') \in \rightarrow \) implies \( q_1 = q_2 \) for all \( a \in A^I \) and \( q_1, q_2 \in Q \)). 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 \( (q, a, q') \in \rightarrow \), \( q \xrightarrow{a} q' \) if there is \( q' \) s.t. \( q \xrightarrow{a} q' \), and \( q^0 \xrightarrow{a} q' \) 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 \( q \xrightarrow{e} q' \). Notice this includes case \( q = q' \). We write \( q \xrightarrow{a} q' \) if there are \( q_1 \) and \( q_2 \) s.t. \( q \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{a} q' \). Moreover \( q \xrightarrow{a} q' \) denotes \( q \xrightarrow{a} q' \) or \( a \in A^H \) and \( q = q' \). We write \( q \xrightarrow{e,a} q' \) if there is \( q' \) s.t. \( q \xrightarrow{e} q' \) and \( q \xrightarrow{a} q' \). 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 \( q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \cdots \) is an execution. The set of traces of an IA \( S \), notation \( \text{Traces}(S) \), 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 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 \( \text{shared}(S, T) = (A_S \cap A_T) \) be the set of shared actions. We say that \( S \) and \( T \) are composable whenever \( \text{shared}(S, T) = (A^I_S \cap A^I_T) \). Two ISS \( S = (A^I_S, A^O_S) \) and \( T = (A^I_T, A^O_T) \) 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 \( S \times T \) is the interface automaton defined by:

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

There may be reachable states on \( S \times T \) 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 \( (q_S, q_T) \in Q_{S \times T} \) is an error state if there is an action \( a \in \text{shared}(S, T) \) s.t. either \( a \in A^O_S \), \( q_S \xrightarrow{a} q'_S \), and \( q_T \xrightarrow{a} q'_T \), or \( a \in A^O_T \), \( q_T \xrightarrow{a} q'_T \), and \( q_S \xrightarrow{a} q'_S \).

If the product \( S \times T \) 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 \( S \times T \)) that leads the product to the error state. Of course, it may be the case that \( S \times T \) 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 \( S \times T \) is incompatible.

**Definition 5.** Let \( S \) and \( T \) be composable IA and let \( S \times T \) be its product. A state \( (q_S, q_T) \in Q_{S \times T} \) 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 \( S \times T \) 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 \( S \times T \) by removing all transition \( q \xrightarrow{a}_{S \times T} q' \) s.t. (i) \( q \) is a compatible state in \( S \times T \), (ii) \( a \in A^I_S \), and (iii) \( q' \) is an incompatible state in \( S \times T \).

### 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 \( (S, A^h, A^l) \) where \( S = (Q, q^0, A^I, A^O, A^H, \rightarrow) \) is an IA and \( A^h \) and \( A^l \) are disjoint sets of actions s.t. \( A^h \cup A^l = A^O \cup A^I \).

CLEI ELECTRONIC JOURNAL, VOLUME 14, NUMBER 1, PAPER 1, APRIL 2011
If necessary, we will write $A^i_x$ and $A^j_y$ instead of $A^i$ and $A^j$, 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 $S = \langle S, A^i_x, A^j_y \rangle$ and $T = \langle T, A^k_z, A^l_w \rangle$ be two ISS. $S$ and $T$ are composable if $S$ and $T$ are composable. Given two composable ISS, $S$ and $T$, their composition, $S \parallel T$, is defined by the ISS $\langle S \parallel T, (A^i_x \cup A^k_z) - \text{shared}(S, T), (A^j_y \cup A^l_w) - \text{shared}(S, T) \rangle$.

### 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 has 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 *types of observations* is an information that can be recorded by an user with respect to the interface. To define our types of observation we consider the following assumption: input and output actions are observable when they are executed. Inputs are executed by an 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 see 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 stops the execution and continues with the execution; (iv) moreover, 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 $a' \in A' - I(s)$. 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, an user can make copies of the interface with the objective of studying the interface in more detail. Finally, an 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 observation:

- **a** The execution of external actions $A^I \cup A^O$ are detectable.
- **e, #** The case of internal transitions are detectable is denoted with e. Otherwise #.
- **T** The session is terminated by the user. This is possible in any time. After this no more records are possible.
- **z & , z#** If a user only sees the actions that are executed by an interface and cannot send stimuli to it, then there is no interaction. We denote this with $z &$. The case where the interaction is possible is denoted by $z #$.
- **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 reaches a final state, i.e. no more activity is possible.
The set of execution formulas 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 \( \eta \)-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 an 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 an user. This has to be consistent, for example, types of observations “an user cannot interact with the interface” \( (\neg \Phi) \) and “an 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 \( V \subseteq \{a, e, \varnothing, 0, \oplus, \ominus, T, F, FT, RT, \land, \neg\} \) and \( V \) satisfies the following conditions:

1. \( \{a, T\} \subseteq V, \)
2. \( ||e, \varnothing|| \cap V| = 1, \)
3. \( ||\ominus, \ominus, T, F, FT, RT|| \cap V| = 1. \)

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 an user can interact with the interface \( (\ominus, F, FT, RT) \) or not \( (\ominus) \), 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.

**Definition 11.** Given an IA \( S = \langle Q, q^0, A^i, A^o, A^H, \rightarrow \rangle \) 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 \( O_V(S) = \{ \phi \in \mathcal{L} : q^0 \models_V \phi \} \)

### Table 1: Recursive rules for definition of execution formulas.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T \in \mathcal{L} )</td>
<td>0 \in \mathcal{L}</td>
</tr>
<tr>
<td>( d \in \mathcal{L} )</td>
<td>( \forall a \in A^i \quad \phi \in \mathcal{L} \quad a \in A^i \cup A^o \cup {\varepsilon} )</td>
</tr>
<tr>
<td>( \phi \in \mathcal{L} )</td>
<td>( a \in A^i \quad d \in \mathcal{L} )</td>
</tr>
<tr>
<td>( \phi \in \mathcal{L} )</td>
<td>( X \subseteq A^i )</td>
</tr>
<tr>
<td>( X \phi \in \mathcal{L} )</td>
<td>( \phi \in \mathcal{L} \quad i \in I )</td>
</tr>
<tr>
<td>( \phi \in \mathcal{L} )</td>
<td>( i \in \cup )</td>
</tr>
<tr>
<td>( \neg \phi \in \mathcal{L} )</td>
<td>( \land \neg \phi \in \mathcal{L} )</td>
</tr>
</tbody>
</table>

\( a, d \) are observable or not. This is a relevant feature in the context of security, because it could be used to represent covert channels.
(T) \( q \models T \quad \forall q \in Q \)
(0) \( q \models 0 \) if \( q \rightarrow a \) for all \( a \in A \)
(a) \( q \models a \phi \) if \( a \in A^I \cup A^O \) and \( \exists q' \in Q : q \rightarrow a \) and \( q' \models \phi \)
(\( \phi \)) \( q \models \phi \) if \( a \in A^H \) and \( \exists q' \in Q : q \rightarrow a \) and \( q' \models \phi \)
(e) \( q \models e \phi \) if \( a \in A^H \) and \( \exists q' \in Q : q \rightarrow a \) and \( q' \models \phi \)
(\( \equiv \)) \( q \models a \phi \) if \( a \in A^I - I(q) \) and \( q \models \phi \)
(\( F \)) \( q \models d \phi \) if \( a \in A^I - I(q) \) and \( q \models \phi \)
(\( FT \)) \( q \models X \phi \) if \( X = I(s) \) and \( q \models \phi \)
(\( R \)) \( q \models X \phi \) if \( X = I(s) \) and \( q \models \phi \)
(\( \land \)) \( q \models \land_{a \in I} \phi_i \) 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

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 \( X \subseteq A^I \cup A^O \), define:

- the restriction of \( X \) in \( S \) by \( S \sm X = (Q_S, q^0_S, A^I_S, A^O_S, \rightarrow_{S \sm X}) \) where \( q \rightarrow_{S \sm X} q' \) if \( q \rightarrow q' \) and \( a \not\in X \).
- the hiding of \( X \) in \( S \) by \( S \sm X = (Q_S, q^0_S, A^I_S, A^O_S, H_X, \rightarrow_{S \sm X}) \).

Given an ISS \( S = (S, A^I, A^O) \) define the restriction of \( X \) in \( S \) by \( S \sm X = (S \sm X, A^I_S \sm X, A^O_S \sm X, H_X \sm X, \rightarrow_{S \sm X}) \).

Definition 13. Let \( S = (S, A^I, A^O) \) be an ISS and \( V \) a notion of observability, then:

- \( S \) is \( V \) strong non-deterministic non-interference (V-SNNI) if \( O_V(S / A^I) = O_V(S / A^O) \).
- \( S \) is \( V \) non-deterministic non-interference (V-NNI) if \( O_V(S / A^I) = O_V((S / A^H)^i) / A^H \).

Notice the difference between the two definitions. V-SNNI formulates 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 \( S \equiv_T T \), if \( \text{Traces}(S) = \text{Traces}(T) \). We say that two ISS \( S \) and \( T \) are trace equivalent, and write \( S \equiv_T T \), 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 \( S R 0 \) and, for all \( s \in Q_S \) and \( t \in Q_T \), \( s R t \) implies:

- for all \( a \in A_S \) and \( s' \in Q_S \), \( s \xrightarrow{a} s' \) implies that there exists \( t' \in Q_T \) s.t. \( t \xrightarrow{a} t' \) and \( s' R t' \); and
- for all \( a \in A_T \) and \( t' \in Q_T \), \( t \xrightarrow{a} t' \) implies that there exists \( s' \in Q_S \) s.t. \( s \xrightarrow{a} s' \) and \( s' R t' \).

We say that \( S \) and \( T \) are bisimilar, notation \( S \approx_T T \), if there is a bisimulation between \( S \) and \( T \). Moreover, we say that two ISS \( S \) and \( T \) are bisimilar, and write \( S \approx_T T \), whenever the underlying IA are bisimilar.
Definition 16. Let \( S = (S, A^h, A^l) \) be an ISS.

1. \( S \) satisfies strong non-deterministic non-interference (SNNI) if \( \forall s \in S \forall t \in T. O_S(s) \approx T O_S(t). \)

2. \( S \) satisfies non-deterministic non-interference (NNI) if \( \forall s \in S \forall t \in T. O_S(s) \approx T O_S(t). \)

3. \( S \) satisfies bisimulation-based strong non-deterministic non-interference (BSNNI) if \( \forall s \in S. O_S(s) \approx S O_S(s). \)

4. \( S \) satisfies bisimulation-based non-deterministic non-interference (BNNI) if \( \forall s \in S. O_S(s) \approx S O_S(s). \)

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

Theorem 1. Let \( S = (S, A^h, A^l) \) be an ISS then

1. \( S \) is S(SNNI) if \( S \) is V(3)(SNNI) with \( V = [a, T, \phi, \not\in]. \)

2. \( S \) is B(3)NNI if \( S \) is V(3)(SNNI) with \( V = [a, T, \phi, \not\in, \land, \neg]. \)

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 \( O_S(s) = O_T(t). \)

We define \( f \) as a function defined as:

\[
\begin{align*}
f(T) &= f(0) = f(\phi) = 0, \\
f(\phi) &= f(\phi) = f(X\phi) = f(\neg\phi) = f(\phi) + 1, \\
f(\land, \phi_i) &= max_i(f(\phi_i)) + 1 \quad (\ast)
\end{align*}
\]

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 \( T = T \) because \( V = [a, T, \phi, \not\in, \land, \neg] \) and since \( T \) is an observation for every state \( \phi \in O_T(t). \)

By induction suppose that if \( s \approx t \) then \( f(\phi) \leq k \) and \( \phi \in O_S(s) \) it holds \( \phi \in O_T(t). \)

To prove (1) we show that given two IA \( S \) and \( T \) it holds \( S \approx T \) \( T \) iff \( O_S(S) \approx O_T(T). \)

We reduce this to prove \( \phi \in \text{Traces}(S) \) iff \( \phi \in \text{Traces}(T). \) 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 \( S \) such that \( S \) is V-NNI and \( S \) is not V-SNNI.

Proof. Let \( S \) the following ISS \( s_0 \xrightarrow{H} s_1 \xrightarrow{a} s_2 \) with \( a \in A^l \cup A^h. \)

Notice \( S \) is always V-NNI. On the other hand \( S \) is not V-SNNI: if \( \not\in = \not\in \in V \) then \( aT \not\in O_T((S/A^h)) \) and \( aT \not\in O_T((S/A^h)) \).

This result is not novel. In [9], 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. 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 \( [0, \not\in] \cap \not\in = \emptyset. \) Let \( S' \) be an IA obtained by removing a set of internal transitions from \( S. \) Then \( O(S) \cup O(S'). \)

Proof. The proof is straightforward by induction in \( f(\phi) \) where \( \phi \in O(V(S')) \) and \( f \) is the function defined in (\ast). □

Theorem 3. Let \( S \) be an ISS and \( V \) a notion of observability such that \( [0, \not\in] \cap \not\in = \emptyset. \) If \( S \) is V-SNNI then \( S \) is V-NNI.

Proof. \( S \) is V-SNNI then \( O_T((S/A^h)) = O_T((S/A^h)). \)

Notice \( S/A^h \) is obtained by removing some hidden transitions from \( (S/A^h)/(A^h/0) \) then \( O_T((S/A^h)/(A^h/0)) = O_T((S/A^h)). \) Therefore as consequence of (\ast). □
In [8], we presented definitions of non interference based on refinement. The new versions of non-interference were obtained from the definition of BSNNI and BNNI. In this section we review the results obtained.

**Theorem 4.** For all notion of observability V such that \( V \cap [0, -] \neq \emptyset \) there is an ISS \( S \) such that \( S \) is V-SNNI and \( S \) is not V-NNI.

**Proof.** Define \( S \) as ISS in Figure 1 with \( a \in A^I \cup \tilde{A}^O \). Clearly \( S \) is V-SNNI for all \( V \). Suppose \( \notin V \): if \( \in V \) then \( a \cdots a \in O_V((S \backslash A^b)^I / \tilde{A}^b) \) while \( a \cdots a \notin O_V(S^b) \); if \( 0 \in V \) then \( a0 \in O_V((\tilde{S} \backslash A^b)^I / \tilde{A}^b) \) while \( a0 \notin O_V(\tilde{S}^b) \). Then \( S \) is not V-NNI for any \( V \) such that \( V \cap [0, -] \neq \emptyset \). The case \( \in V \) is analogous. \( \Box \)

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

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

**Proof.** Let \( S \) and \( T \) be ISS depicted in Figure 2. Both interfaces are V-(S)NNI for all notion of observability \( V \) but \( S \parallel T \) is not. If \( \notin V \) then \( b?T \in O_V((S \parallel T)^b) \) while if \( \in V \) then \( eb?T \in O_V((S \parallel T)^b) \). In any case, \( O_V((S \parallel T)^b) = O_V((S \parallel T) \backslash A^b) \) and \( b?T \notin O_V((S \parallel T)^b) \). Then \( S \parallel T \) is not V-(S)NNI. \( \Box \)

5 Non-interference based on refinement.

In [8], 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], i.e. BSNNI and BNNI. In this section we review the results obtained.

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

![Graphical representation](image-url)
Definition 17. Given two IA $S$ and $T$, a relation $\trianglerighteq \subseteq Q_S \times Q_T$ is a Strict Input Refinement (SIR) of $S$ by $T$ if $q^S_S \trianglerighteq q^T_T$ and for all $q^S_S \trianglerighteq q^T_T$ it holds:

(a) $\forall a \in A^S_S, q^S_S \in Q_S$, if $q^S_S \xrightarrow{a} q^S_S$ then $\exists q^T_T \in Q_T : q^S_S \xrightarrow{a} q^T_T$ and $q^S_S \trianglerighteq q^T_T$;

(b) $\forall a \in A^T_T, q^T_T \in Q_T$, if $q^T_T \xrightarrow{a} q^T_T$ then $\exists q^S_S \in Q_S : q^S_S \xrightarrow{a} q^S_S$ and $q^S_S \trianglerighteq q^T_T$;

(c) $\forall a \in A^S_S, q^T_T \in Q_T$, if $q^T_T \xrightarrow{a} q^T_T$ then $\exists q^S_S \in Q_S : q^S_S \xrightarrow{a} q^S_S$ and $q^S_S \trianglerighteq q^T_T$;

(d) $\forall a \in A^T_T, q^S_S \in Q_S$, if $q^S_S \xrightarrow{a} q^S_S$ then $\exists q^T_T \in Q_T : q^S_S \xrightarrow{a} q^S_S$ and $q^S_S \trianglerighteq q^T_T$.

We say $S$ is refined (strictly on inputs) by $T$, or, $T$ refines (strictly on inputs) to $S$, notation $S \trianglerighteq T$. Let $S$ and $T$ be two ISS, we write $S \trianglerighteq T$ if the underlying IA satisfy $S \trianglerighteq T$.

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 $S$ be an ISS, $(i)$ $S$ is SIR-based strong non-deterministic non-interference (SIR-SNNI) if $S \setminus A^S_S \trianglerighteq S \setminus A^T_T$ $(ii)$ $S$ is SIR-based non-deterministic non-interference (SIR-NNI) if $S \setminus A^{T,T} / A^{S,S} \trianglerighteq S \setminus A^T_T$.

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 $S \setminus A^S_S$ (resp. $S \setminus A^{T,T} / A^{S,S}$) is refined by $S \setminus A^T_T$.

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 $S \trianglerighteq T$ iff $O_V(S) \supseteq O_V(T)$ with $V = \{a, T, \emptyset, RT, \emptyset\}$.

Proof. For this, we have to show that for all states $s \in Q_S$ and $t \in Q_T$ it holds $s \trianglerighteq t$ iff $O_V(s) \supseteq O_V(t)$.

$(\Rightarrow)$ Suppose $s \trianglerighteq t$ and $\phi \in O_V(t)$. Let $f : \mathcal{L} \to \mathbb{N}$ the function defined in $s$. We proceed by complete induction. In the base case $f(\emptyset) = 0$ then $\phi = T$ because $V = \{a, T, \emptyset, RT, \emptyset\}$ and since $T$ is an observation for every state, then $\phi \in O_V(s)$.

Inductive case. By induction suppose that if $s \trianglerighteq t$ then, if $f(\emptyset) \leq k$ and $\phi \in O_V(t)$ it holds $\phi \in O_V(s)$. Let $f(\emptyset) = k + 1$, we do case analysis according to the shape of the formula. Suppose $\phi = X\emptyset'$. Since $t \models X\emptyset'$ then $t \models \emptyset'$. Moreover, $s \trianglerighteq t$ implies $(s) = (t)$ and therefore $s \models X\emptyset'$ using induction. Cases $a\emptyset'$ and $\emptyset \wedge \phi_i$ 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 $s \xrightarrow{a?} t$ then $t \models (t)$ and therefore $O_V(s) \supseteq O_V(t)$ because $t \models (t)$ and $s \not\models (t)$. 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 $\emptyset' \in O_V(t') - O_V(s')$. This implies $a?\emptyset' \in O_V(t) - O_V(s)$ and we get a contradiction. In the case $s \xrightarrow{a?} s'$, we have to show there is $t'$ such that $s \xrightarrow{a?} t'$ and $O_V(s') \supseteq O_V(t')$, this proof is similar to the previous one.

Now let $t \xrightarrow{a?} t'$, we have to show there is $s'$ such that $s \not\xrightarrow{a?} s'$ and $O_V(s') \supseteq O_V(t')$. Let $Q$ be $\{s' \mid t \models (s')\}$. If for all $s' \in Q$ it holds $O_V(s') \not\supseteq O_V(t')$ then there is $\emptyset_i \in O_V(t') - O_V(s')$. Then for any $s' \in Q$ it holds $\emptyset \wedge Q \emptyset_i \in O_V(t') - O_V(s')$ (at least one $\emptyset_i$ fails). But then $a \wedge Q \emptyset_i \in O_V(t) - O_V(s)$ contradicting $O_V(s) \supseteq O_V(t)$. Case $t \xrightarrow{a?} t'$ is analogous. \hfill\$\Box$

Now we are able show the statement.

Lemma 2. An IA $S$ is SIR-(S)NNI if $S$ is $\{a, T, \emptyset, RT, \emptyset\}$-(S)NNI.

Proof. If $S$ is SIR-SNNI then $S \setminus A^S_S \trianglerighteq S \setminus A^T_T$. By Theorem 6 we have $O_V(S \setminus A^S_S) \subseteq O_V(S \setminus A^T_T)$. On the other hand, by Lemma 1 we have $O_V((S \setminus A^S_S)) \supseteq O_V((S \setminus A^T_T))$. Finally $O_V((S \setminus A^S_S)) = O_V((S \setminus A^T_T))$. The case $S$ is SIR-NNI is analogous. \hfill\$\Box$

Two properties about SIR-NNI and SIR-SNNI were introduced in [8]. 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 $\{a, T, \emptyset, RT, \emptyset\}$-(S)NNI and $\{a, T, \emptyset, \emptyset\}$-(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.
\textbf{Theorem 7.} Let $S = \langle S, A_S^b, A_S^c \rangle$ and $T = \langle T, A_T^b, A_T^c \rangle$ be two composable ISS such that shared($S, T$) $\cap (A_S^b \cup A_T^b) = \emptyset$. If $S \parallel T$ has no reachable error states and $S$ and $T$ satisfy SIR-NNI (resp. SIR-NNI) then $S \parallel T$ satisfies SIR-SNNI (resp. SIR-NNI).

\textit{Proof.} Define $\equiv$ by $(s_v, t_v) \equiv (s_{v'}, t_{v'})$ iff $s_v \equiv_S s_{v'}$ and $t_v \equiv_T t_{v'}$ with $\equiv_S$ being a SIR between $S \setminus A_S^b$ and $S \setminus A_S^b$ and similarly for $\equiv_T$. We show that $(v, t_v)$ is a SIR between $(S \parallel T)\lfloor A^b$ and $(S \parallel T)\lfloor A^b$ where $A^b = (A_S^b \cup A_T^b) \setminus$ shared($S, T$) = $A_S^b \cup A_T^b$.

Suppose $(s_v, t_v) \equiv (s_{v'}, t_{v'})$. We proceed by case analysis on the different transfer properties on Def 17. For case (a) suppose $(s_v, t_v) \xrightarrow{c} (s_v', t_v')$ and $s_v \equiv_S s_{v'}$. Then there is $s_v''$ such that $s_v \xrightarrow{a} s_v''$ and $s_v'' \equiv_S s_v'$. As a consequence of the absence of error state in the product, we can ensure $(s_v, t_v) \xrightarrow{a} (s_v', t_v')$. The case $(s_v, t_v) \xrightarrow{V} (s_v', t_v')$ is analogous. In the same way we prove that condition (b) holds. For condition (c), let $(s_v, t_v) \xrightarrow{a} (s_v', t_v')$ and $s_v \equiv_S s_{v'}$. Then there is $s_v''$ such that $s_v \equiv_S s_v''$ and $s_v'' \equiv_S s_v'$.

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.

\textbf{Corollary 1.} Let $S = \langle S, A_S^b, A_S^c \rangle$ and $T = \langle T, A_T^b, A_T^c \rangle$ be two composable ISS. Let $S' = \langle S, A_S^b, \text{shared}(S, T), A_S^c \cup \text{shared}(S, T) \rangle$ and $T' = \langle T, A_T^b, \text{shared}(S, T), A_T^c \cup \text{shared}(S, T) \rangle$. If $S \parallel T$ has no reachable error states and $S'$ and $T'$ satisfy SIR-SNNI (resp. SIR-NNI) then $S \parallel T$ 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.

\subsection{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 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. 6); i.e. we restrict some input transitions in order the components are already available but they were designed independently and they were not meant to interact. The semi-saturation of a 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. We then obtain 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 algorithms 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 $\Rightarrow a^{-1}$; (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 $\mathcal{V} \not\equiv \mathcal{V}'$; if there is always a way to avoid reaching a fail state, then $\mathcal{V} \equiv \mathcal{V}'$. We later define properly semi-saturation, semi-synchronous product and what means inevitably reaching a fail state. In this way, given an ISS $S$, we can check if $\mathcal{S}A^b \gg S/A^b$, if the check succeeds, then $S$ satisfies SIR-SNNI (see Theorem 8). If it does not succeed, then we provide an algorithm to decide whether $S$ 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).
Definition 19. Let $S$ be an IA and $B \subseteq A^H_S$. Define $S$ marking $B$ or marking $B$ in $S$ as the IA $S_B = \langle Q_S, q^0_S, A^I_S, A^O_S, \{e, e'\}, \rightarrow_{S_B} \rangle$ where $\rightarrow_{S_B}$ is the least relation satisfying following rules:

$$\begin{align*}
q & \rightarrow_{S_B} q' & q & \stackrel{\alpha}{\rightarrow_{S_B}} q' & q & \stackrel{\alpha'}{\rightarrow_{S_B}} q', & a & \in A^I_S \\
q & \rightarrow_{S_B} q' & q & \stackrel{\alpha}{\rightarrow_{S_B}} q' & q & \stackrel{\alpha}{\rightarrow_{S_B}} q', & a & \in A^O_S - B \\
q & \rightarrow_{S_B} q' & q & \stackrel{\alpha}{\rightarrow_{S_B}} q'
\end{align*}$$

Given an ISS $S$, the marking $B$ in $S$, notation $S_B$, 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 $q \rightarrow a q'$ to the model for each weak transition $q \overset{\alpha}{\rightarrow} q'$, 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 $q \rightarrow a q'$ whenever $q \overset{\alpha}{\rightarrow} q'$ with $a$ an output action. We call this process semi-saturation.

Definition 20. Let $S$ be an IA such that $A^O_S = \{e, e'\}$. The semi-saturation of $S$ is the IA $\overline{S} = \langle Q_S, q^0_S, A^I_S, A^O_S, \{e, e'\}, \rightarrow_{\overline{S}} \rangle$ where $\rightarrow_{\overline{S}}$ is the smallest relation satisfying the following rules:

$$\begin{align*}
q & \rightarrow_{\overline{S}} q' & q & \overset{\alpha}{\rightarrow_{\overline{S}}} q' & q & \overset{\alpha'}{\rightarrow_{\overline{S}}} q', & a & \in A^O_S \\
q & \rightarrow_{\overline{S}} q' & q & \overset{\alpha}{\rightarrow_{\overline{S}}} q' & q & \overset{\alpha}{\rightarrow_{\overline{S}}} q'
\end{align*}$$

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

The last definition ensure that: if $a \in A^O$ then $q \overset{\alpha}{\rightarrow} q'$ iff $q \overset{\alpha}{\rightarrow_{\overline{S}}} q'$.

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 $S \overset{\alpha}{\rightarrow} T$ then for two states $s \in Q_S$ and $t \in Q_T$ s.t. $s \overset{\alpha}{\rightarrow} t$, every output/hide 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/hide 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 $A^S_X = A^T_Y = A^X_Y$ for $X \in \{I, O\}$ and $A^O_S = A^O_T = \{e, e'\}$. The semi-synchronous product of $S$ and $T$ is the IA $S \times T = \langle (Q_S \times Q_T) \cup \{fail\}, (q^0_S, q^0_T), A^I, A^O, \{e, e'\}, \rightarrow_{S \times T} \rangle$ where $\rightarrow_{S \times T}$ is the smallest relation satisfying following rules:

$$\begin{align*}
q_S & \overset{\alpha}{\rightarrow} q'_S & q_T & \overset{\alpha}{\rightarrow} q'_T & q_S & \overset{\alpha}{\rightarrow} q'_T & q_T & \overset{\alpha}{\rightarrow} q'_T & (q_S, q_T) & \overset{\alpha}{\rightarrow_{S \times T}} (q'_S, q'_T) \\
q_S & \overset{\alpha}{\rightarrow} q'_S & q_T & \overset{\alpha}{\rightarrow} q'_T & q_S & \overset{\alpha}{\rightarrow} q'_T & q_T & \overset{\alpha}{\rightarrow} q'_T & (q_S, q_T) & \overset{\alpha}{\rightarrow_{S \times T}} (q'_S, q'_T) \\
q_S & \overset{\alpha}{\rightarrow} q'_S & q_T & \overset{\alpha}{\rightarrow} q'_T & q_S & \overset{\alpha}{\rightarrow} q'_T & q_T & \overset{\alpha}{\rightarrow} q'_T & (q_S, q_T) & \overset{\alpha}{\rightarrow_{S \times T}} (q'_S, q'_T) \\
q_S & \overset{\alpha}{\rightarrow} q'_S & q_T & \overset{\alpha}{\rightarrow} q'_T & q_S & \overset{\alpha}{\rightarrow} q'_T & q_T & \overset{\alpha}{\rightarrow} q'_T & (q_S, q_T) & \overset{\alpha}{\rightarrow_{S \times T}} (q'_S, q'_T)
\end{align*}$$

Given $S = \langle S, A^S_S, A^O_S \rangle$ and $T = \langle T, A^S_T, A^O_T \rangle$ with $S$ and $T$ satisfying conditions above and $A^S_m = A^O_m = A^m$ for $m \in \{I, O\}$, then the semi-synchronous product of $S$ and $T$ is defined by the ISS $S \times T = \langle S \times T, A^B, A^A \rangle$.

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 $(q_S, q_T) \overset{\alpha}{\rightarrow_{S \times T}} fail$ then it is evident that $q_S \neq q_T$. Moreover, suppose the synchronous product only has states $(q_S, q_T)$ and $fail$ and the transition $(q_S, q_T) \overset{\alpha}{\rightarrow_{S \times T}} fail$. 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 $q_S \neq q_T$. 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 $(q_S, q_T) \overset{\alpha}{\rightarrow_{S \times T}} fail$ in the synchronous product and we get two states such that $q_S \not\geq q_T$. 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.

\[
\text{Fail}^A = \{(q_s, q_T) \mid (q_s, q_T) \xrightarrow{a \in A^I} \text{ fail, } a \notin A^F \} \cup \{\text{fail}\}
\]

\[
\text{Fail}^{k+1} = \text{Fail}^k \cup \{(q_s, q_T) \mid a \in A^G \cup A, q_T \xrightarrow{a} q_T', (\forall q'_S : (q_s, q_T) \xrightarrow{a} (q'_S, q'_T) : (q'_S, q'_T) \in \text{Fail}^k)\}
\]

Table 3: The \(Fail\) set.

\[
\text{May}^0 = \bigcup \frac{a}{q'_{S} \leftarrow q, \gamma \leftarrow T}
\]

\[
\text{May}^k = \bigcup \frac{a}{q'_{S} \leftarrow q, \gamma \leftarrow T}
\]

\[
\text{May}^{k+1} = \bigcup \frac{a}{q'_{S} \leftarrow q, \gamma \leftarrow T}
\]

\[
\text{May}^{k+1}_{q, q_T \leftarrow q} = \{(q_s, q_T) \notin \text{Fail} : a \in A, q_s \xrightarrow{a} q'_S, (\forall q'_T : (q_s, q_T) \xrightarrow{a} (q'_S, q'_T) : (q'_S, q'_T) \in \text{Fail} \cup \text{May}^k)\}
\]

\[
\text{May}^{k+1}_{q, q_T \leftarrow q} = \{(q_s, q_T) \notin \text{Fail} : a \in A, q_s \xrightarrow{a} q'_S, (\forall q'_T : (q_s, q_T) \xrightarrow{a} (q'_S, q'_T) : (q'_S, q'_T) \in \text{Fail} \cup \text{May}^k)\}
\]

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:

\(\text{Fail} = \cup_{(q, q') \in \text{Fail}}\) where \(\text{Fail}^k\) is defined in Table 3. If \(q \in \text{Fail}\), we say that the pair \(q\) fails the SIR relation test.

\(\text{May} = \cup_{(q, q') \in \text{May}}\) where \(\text{May}^k\) is defined in Table 4. If \(q \in \text{May}\), we say that the pair \(q\) may pass the SIR relation test.

\(\text{Pass} = Q_{S \times T} - (\text{May} \cup \text{Fail})\). If \(q \in \text{Pass}\), we say that the pair \(q\) passes the SIR relation test.

If the initial state of the underlying IA of an ISS \(S \times T\) passes (may pass, fails) the SIR relation test, we say that \(S \times T\) 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 \(S \not\succ T\).

**Proof sketch.** Since \((\text{May} \cup \text{Fail}) \cap \text{Pass} = \emptyset\), we only have to prove that (i) \((q_s, q_T) \in \text{May} \cup \text{Fail}\) implies \(q_s \not\succ q_T\) and (ii) if \((q_s, q_T) \in \text{Pass}\) then \(q_s \not\succ q_T\). 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 \((s, t) \in Q_{S \times T} \cap \text{Pass}\), 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 \((s', t') \in \text{Pass}\).
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 \((s', t') \in \text{Pass}\).

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 \(S\) is SIR-SNNI if \(S \setminus A^h\) is refined by \(S / A^h\). Notice that we cannot use \(S \setminus A^h\) and \(S / A^h\) to create a semi-synchronized product; in general, \(S \setminus A^h\) does not satisfy \(A^h = \{e, e'\}\) and it is not semi-saturated. This can be solved marking \(\emptyset\) in \(S \setminus A^h\) and then semi-saturating the interface, i.e. we work with \((S \setminus A^h)_{\emptyset}\) instead of \(S \setminus A^h\). Similarly, \(S / A^h\) does not satisfy \(A^h = \{e, e'\}\). Since \(e'\) is used to represent the internal action that can be removed, we solve this problem marking \(A^h\) in \(S / A^h\), i.e. we replace \(S / A^h\) by \((S / A^h)_{\emptyset}\). Therefore, verifying that \(S\) satisfies SIR-SNNI amounts to checking whether \(P_S = \overline{(S \setminus A^h)_{\emptyset}} \times (S / A^h)_{\emptyset}\) passes the refinement test. Applying a similar reasoning, if we are interested on verifying SIR-NNI, we can check if \((\overline{(S \setminus A^h)_{\emptyset}} \times (S / A^h)_{\emptyset})_{\emptyset}\) 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 \( S = (S, A^t, A^r) \) be an ISS.

1. \( S \) satisfies SIR-SNNI iff \((S \backslash A^t) \times (S \backslash A^r) \) passes the SIR-relation test.

2. \( S \) satisfies SIR-NNI iff \((\langle S \backslash A^t \rangle) \times (\langle S \backslash A^r \rangle) \) passes the SIR-relation test.

**Synthesizing Secure ISS.** In the following, we show that if a synchronized product \( P_S \) 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 \( S \) is an ISS such that \( P_S \) may pass the SIR-relation test, the set \( EC(S) \subseteq Q \times A^I \times Q \) (see Table 5) is the set of eliminable candidates.

All transitions in \( EC(S) \) are involved in a synchronization that connects a source pair that may pass the SIR-relation test and a failing tail. 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 \( P_S \) may pass the bisimulation test then \( EC(S) \neq \emptyset \).

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 synchronized product of the original ISS and it is also present in the synchronized 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 \( \text{Fail} \) and \( \text{Pass} \cup \cup \text{May} \) remain invariant.

**Lemma 4.** Let \( S \) be an ISS s.t. \( P_S \) may pass the SIR-relation test. Let \( S' \) be an ISS obtained by removing one transition in \( EC(S) \) from \( S \) (i.e., \( r \rightarrow q \rightarrow a \rightarrow q' \rightarrow \text{Fail} \)). Then it holds that: (i) \( \text{Fail}_{P_S} = \text{Fail}_{P_S} \cap Q_{P_S} \cup \text{May}_{P_S} \cap Q_{P_S} = \text{Pass}_{P_S} \cup \text{May}_{P_S} \cup Q_{P_S} \).

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

*(Case \( \subseteq \)).* Clearly \( Q_{P_S} \subseteq Q_{P_S} \). Suppose \( q \rightarrow a \rightarrow q' \in EC(S) \) is the transition that is removed. By induction on \( k \) we show \( \text{Fail}^k_{q\rightarrow a \rightarrow q', P_S} \subseteq \text{Fail}^k_{q\rightarrow a \rightarrow q', P_S} \) for all \( k \). This implies \( \text{Fail}^k_{q\rightarrow a \rightarrow q', P_S} \subseteq \text{Fail}^k_{q\rightarrow a \rightarrow q', P_S} \) and then \( \text{Fail}_{P_S} \subseteq \text{Fail}_{P_S} \). Suppose \( (q_t, q_a) \in \text{Fail}^0_{q\rightarrow a \rightarrow q', P_S} \). By definition, action \( a \notin A^I \cup \{\varepsilon\} \) and \( (q_t, q_a) \rightarrow \text{Fail} \). Then \( a \neq \emptyset \) and therefore \( (q_t, q_a) \rightarrow \text{Fail} \) belongs to \( P_S \). Then \( (q_t, q_a) \in \text{Fail}^0_{q\rightarrow a \rightarrow q', P_S} \). Suppose now \( (q_t, q_a) \in \text{Fail}^1_{q\rightarrow a \rightarrow q', P_S} \). Then \( a \notin A^I \cup \{\varepsilon\} \) and \( (q_t, q_a) \rightarrow \text{Fail}^1_{q\rightarrow a \rightarrow q', P_S} \). Notice that \( (q_t, q_a) \rightarrow \text{Fail}^1_{q\rightarrow a \rightarrow q', P_S} \) as consequence of \( b \notin A^I \cup \{\varepsilon\} \). By induction hypothesis \( \text{Fail}^k_{P_S} \subseteq \text{Fail}^k_{P_S} \), then \( (q_t, q_a) \rightarrow \text{Fail}^1_{q\rightarrow a \rightarrow q', P_S} \).

*(Case \( \supseteq \)).* We show by induction on \( k \) that \( \text{Fail}^k_{P_S} \subseteq \text{Fail}^k_{P_S} \cap Q_{P_S} \) for all \( k \). Let \( (q_t, q_a) \in \text{Fail}^k_{P_S} \cap Q_{P_S} \). Moreover, w.l.o.g. suppose \( (q_t, q_a) \in \text{Fail}^k_{q\rightarrow a \rightarrow q', P_S} \). Since \( a \notin A^I \), the transition \( q_t \rightarrow q' \) cannot be removed and since \( q_t, q_a \), then it holds that \( (q_t, q_a) \in \text{Fail}^k_{q\rightarrow a \rightarrow q', P_S} \). For the induction case, suppose w.l.o.g. \( (q_t, q_a) \in \text{Fail}^k_{q\rightarrow a \rightarrow q', P_S} \cap Q_{P_S} \).

Then \( (q_t, q_a) \rightarrow \text{Fail}^k_{q\rightarrow a \rightarrow q', P_S} \). Since \( (q_t, q_a) \) is reachable in \( S' \) and \( a \notin A^I \), all pair \( (q_t, q_a) \) is reachable in \( S' \). By induction hypothesis, \( (q_t, q_a) \rightarrow \text{Fail}^k_{q\rightarrow a \rightarrow q', P_S} \) and then \( (q_t, q_a) \in \text{Fail}^k_{q\rightarrow a \rightarrow q', P_S} \).

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.

**Table 5: Set of eliminable candidates.**

<table>
<thead>
<tr>
<th>q → a → q'</th>
<th>q → a → q'</th>
<th>q → a → q'</th>
</tr>
</thead>
</table>
| Table 5: Set of eliminable candidates.

Subindices in \( \text{Fail}_{P_S} \), \( \text{May}_{P_S} \), etc. indicate that these sets were obtained from the synchronous product \( P_S \).
Theorem 9. Let $S$ be an ISS such that $P_S$ may pass the SIR relation test. Then there is an input transition set $\rightarrow_c$ such that, if $S'$ is the ISS obtained from $S$ by removing all transitions in $\rightarrow_c$, $S'$ 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 $S'$ be an ISS obtained from $S$ by removing one transition from the set $EC(S)$. Lemma 4 ensures that $S'$ may pass or passes the SIR relation test. If $S'$ passes the SIR relation test, we stop. If $S'$ 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, $\rightarrow_c$ is composed by the set of transitions removed along the way. □

6 Concluding remarks

In this work, we have presented semantics for interactive sequential system. 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 these. Notice that, to avoid the avoided types of observation introduced in [2] should not be be problematic, but probably, Definition 9 has to be redefined.

The approach to define the semantic is based on both types of observation and notion of observability (Definition 9). This approach gives an important insight about the model and its possible semantics. To define which types of observation have sense for the model is not an easy task, one has to have a clear idea about what is possible in the model and how the model behaves. For example, type $\varepsilon$ has sense because IA has information about internal transition; type $FT$ has sense because an interface could be interested in continue the execution after a violation of an input assumption. On the other hand, to define the notion of observability it is necessary to know how the types of observation are related between them. As a result, one has the complete range of possible semantics of the model (with respect to the set of types of observations).

We also presented a definition of non-interference based on notion of observability. This new definition generalize 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 [8], 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 soul be considered in a context where an attacker can only see how the system behaves. On the other hand, SIR-(S)NNI soul be considered in a context where the attacker can know what inputs are enabled in the interface. This is obvious when we see the notion of observability used to represent each property: $\{a, T, \&, \&; \&, \&\} \cup \{a\}$ for B(S)NNI and $\{a, T, \&; \&, \&, \&\}$ for SIR-(S)NNI. Notice that B(S)NNI has the no interaction type ($\&$) while in SIR-(S)NNI the interaction is explicit due to the type ($RT$).

In addition, types of observations allow an easy refinement of the notion of security. For example notice interface $S$ in Figure 2. One can argue that still there is still an information leakage, because the execution of action $a$! is evidence that the high user has not interacted with the interface. If this information is sensitive and the attacker interacts with the interface, one can use notion of observability $V = \{a, T, \&; \&, \&, \&\}$ 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. In first place, the types of observations presented in [2] that have been omitted, have to be added, and a deep study comparing the different semantics should be carried out to get a better understanding of them. Second, a study about 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 is needed.

References


