A Classification of Security Properties for Process Algebras

(JCS version)

R. Focardi and R. Gorrieri

November 30, 1994

Journal of Computer Security *draft printout, (November 30, 1994), 1-32*
IOS Press

A CLASSIFICATION OF SECURITY PROPERTIES FOR PROCESS ALGEBRAS[1]

Riccardo Focardi and Roberto Gorrieri[2]
Università di Bologna
Dipartimento di Matematica
Piazza di Porta San Donato 5
I – 40127 Bologna (Italy)

Abstract

Several information flow security definitions, proposed in the literature, are generalized and adapted to the model of labelled transition systems. This very general model has been widely used as a semantic domain for many process algebras, e.g. CCS. As a by-product, we provide a process algebra similar to CCS with a set of security notions, hence relating these two areas of concurrency research. A classification of these generalized security definitions is presented, taking into account also the additional property of input totality, which can influence this taxonomy. We also show that some of these security properties are composable w.r.t. the operators of parallelism and action restriction.

1. Introduction

In the literature, several *possibilistic* information flow security properties have been proposed [10, 17, 21, 25, 31, 32], in order to specify what properties a *multilevel* system should satisfy to be called secure. Unfortunately, most of them are based on rather dissimilar abstract machines and following also different intuitions, hence making difficult a formal comparison.

One feature, common to some of these properties, is the *synchrony* assumption [3, 32, 25]: a system is composed of several components which have to proceed together at every time instant. This assumption is incompatible

---

[1] Research partially supported by Italian CNR and MURST.
An extended abstract of this paper – entitled "A Taxonomy of Trace-based Security Properties for CCS" – appeared in Proc. Seventh IEEE *Computer Security Foundations Workshop* (CSFW'94), (Li Gong Ed.), IEEE Press, 126-136, Franconia (NH), June 1994.

[2] e_mail:{focardi,gorrieri}@cs.unibo.it

with the idea of describing *distributed systems*, where the geographical distance among the various components prevents the existence of a common global clock for the system. Hence, our first aim is to extend these definitions to models where the execution is asynchronous. Another common feature of some security models, at least in their original formulation, is the assumption of *determinism* [3, 10]: outputs are univocally determined by the history of the past inputs. Again, this assumption is unrealistic when modeling concurrent applications, where the actual behaviour depends on the relative speed of the various sequential subsystems. Indeed, most programming languages provide mechanism for controlling/specifying alternative choices in the flow of the computation, which can also be intrinsically nondeterministic. Hence, our second aim is to extend these definitions to models where nondeterminism can be expressed.

Asynchrony and nondeterminism are two basic features of several *process algebras*; perhaps, the most famous representative of this class is CCS, proposed by Milner in [26]. The main goal of the present paper is to import in this very active field of concurrency theory several of the information flow security properties and to compare their relative merits in this setting. This is achieved by studying them on a specific process algebra, called SPA (Security Process Algebra), which is a slight extension of CCS. For this reason, we have rephrased all of them (keeping as much as possible the original intuition) on a common, general model: *labelled transition systems* (lts, for short) [18], which is the semantic domain for the operational description of SPA. Moreover, because of the intrinsic algebraic nature of SPA, compositionality becomes a central issue; hence, we study which security properties are preserved by the central operators of parallel composition and restriction.

As all the security notions we investigate were defined with the assumption that the semantics of a system is the set of its execution traces, we also have taken this point of view. Indeed, the first part of the paper is essentially devoted to the classification of trace-based formulation of several existing information flow security properties. However, in the second part, we show that it may be the case that a finer notion of system behaviour, such as *weak bisimulation* [26], can be more appropriate in some cases. Then we study new definitions for four information flow properties based on weak bisimulation, showing that their taxonomy does not present some obvious relations holding for their trace variants. We also investigate the problem of considering systems which are *Input-Total*, i.e. systems always able to accept any input. We illustrate how the taxonomy changes when Input-Totality is assumed.

The paper is organized as follows. Section 2 introduces a few notions on execution traces and SPA. The first part of the paper consists of Section 3 and presents several information flow security notions in their lts formulation. These properties are: *Non-Deterministic Non Interference*, originated from [10]; *Two-Level Non Deducibility on Inputs*, inspired to [31]; *lts-Restrictiveness*, a version of [21]; *lts-Correctability*, from [17]; and *Non*

Deducibility on Composition, we introduced in [7] as a natural transposition on the lts model of *Non Deducibility on Strategies* [32]. The second part of the paper consists of Section 4, devoted to the study of weak bisimulation based security properties. Section 5 reports a few concluding remarks on the intuitively expected "correct" notion of security (together with a comment about computer-aided verification) and a short comparison with related literature.

In order to summarize all the results reported in this paper, the reader may find useful to look at Figure 3 (at the end of the first part of the paper, illustrating the inclusion diagram for the existing trace-based properties), Figure 4 (giving an account of the taxonomy for the properties studied with both trace and bisimulation semantics), Tables 2 and 3 (summarizing all the examples for the proper containment).

2. Labelled Transition Systems and Security Process Algebra

The model used to unify the definitions of security we are going to investigate is the labelled transition system model [18]. A labelled transition system is essentially an automaton which can have possibly infinite states. "A significant advantage of having the model based on a standard notion like the automaton is that extensive literature and well-developed intuition become immediately applicable to the problem domain" (quoted from the DoD-NCSC Integrity report [28]). In addition, labelled transition systems constitute the preferred semantic domain for the operational description of many concurrent languages.

Definition 2.1 *A* labelled transition system (lts) *is a triple* $(S, T, \rightarrow)$ *such that:*

- S *is a set of states*
- T *is a set of labels (actions)*
- $\rightarrow \subseteq S \times T \times S$ *is a set of labelled transitions*

$(S_1, \alpha, S_2) \in \rightarrow$ (or equivalently $S_1 \xrightarrow{\alpha} S_2$) means that the system can move from the state S_1 to the state S_2 through the action α.

In the following, systems will be specified using the *Security Process Algebra* (SPA for short), a slight extension of Milner's CCS [26]. SPA includes two more operators, namely the hiding operator of CSP [15] and the (new) input restriction operator, which are useful in characterizing some security properties in an algebraic style. Moreover the set of visible actions is partitioned into high level actions and low level ones in order to specify multilevel systems. [3]

SPA syntax is based on the following elements:

---

[3] Actually, only two-level systems can be specified; note that this is not a real limitation because it is always possible to deal with the multilevel case by grouping – in several ways – the various levels in two clusters.

- A set $I = \{a, b, \ldots\}$ of *input* actions, a set $O = \{\bar{a}, \bar{b}, \ldots\}$ of *output* actions, and a set $\mathcal{L} = I \cup O$ of *visible* actions, ranged over by α.
- A function $\bar{\cdot} : \mathcal{L} \to \mathcal{L}$ such that if $a \in I$ then $\bar{a} \in O$ and if $\bar{a} \in O$ then $\bar{\bar{a}} = a \in I$. If $L \subseteq \mathcal{L}$, then $\bar{L} \stackrel{\mathrm{def}}{=} \{\bar{a} : a \in L\}$.
- Two sets Act_H and Act_L of high and low level actions such that $\overline{Act_H} = Act_H$, $\overline{Act_L} = Act_L$, $Act_H \cup Act_L = \mathcal{L}$ and $Act_H \cap Act_L = \emptyset$.
- A set $Act = \mathcal{L} \cup \{\tau\}$ of actions (τ is the internal, invisible action), ranged over by μ.
- A set K of constants, ranged over by Z.

The syntax of SPA *agents* is defined as follows:

$$E ::= 0 \mid \mu.E \mid E + E \mid E|E \mid E \setminus L \mid E \setminus_I L \mid E/L \mid E[f] \mid Z$$

where $L \subseteq \mathcal{L}$ and $f : Act \to Act$ is such that $f(\bar{\alpha}) = \overline{f(\alpha)}$, $f(\tau) = \tau$. Moreover, for every constant Z there must be the corresponding definition: $Z \stackrel{\mathrm{def}}{=} E$. Intuitively, 0 is the empty process, which cannot do any action; $\mu.E$ can do an action μ and then behaves like E; $E_1 + E_2$ can alternatively choose [4] to behave like E_1 or E_2 ; $E_1|E_2$ is the parallel composition of E_1 and E_2, where the executions of the two systems are interleaved, possibly synchronized on complementary input/output actions, producing an internal τ. $E \setminus L$ can execute all the actions E is able to do, provided that they do not belong to $L \cup \bar{L}$, while $E \setminus_I L$ requires that the actions of E do not belong to $L \cap I$; E/L turns all the actions in L into internal τ's; if E can execute action μ, then $E[f]$ performs $f(\mu)$; finally, Z does what E does, if $Z \stackrel{\mathrm{def}}{=} E$.

Let $\mathcal{E}$ be the set of SPA agents, ranged over by E, F. Let $\mathcal{L}(E)$ denote the *sort* of E, i.e., the set of the (possibly executable) actions occurring syntactically in E. The sets of high level agents and low level ones are defined as $\mathcal{E}_H \stackrel{\mathrm{def}}{=} \{E \in \mathcal{E} \mid \mathcal{L}(E) \subseteq Act_H \cup \{\tau\}\}$ and $\mathcal{E}_L \stackrel{\mathrm{def}}{=} \{E \in \mathcal{E} \mid \mathcal{L}(E) \subseteq Act_L \cup \{\tau\}\}$, respectively.

The operational semantics of SPA is the lts $(\mathcal{E}, Act, \to)$ where the transition relation $\to \subseteq \mathcal{E} \times Act \times \mathcal{E}$ is defined by structural induction as the least relation generated by the axioms and inference rules reported in Table 1.

A widely used system composition operator requires that two systems are composed in parallel by forcing the synchronization on their common (complementary) actions. This can be implemented as an auxiliary operator by means of parallel composition (which permits synchronizations) and restriction (which forces the possible synchronizations).

---

[4] For notational convenience, we use sometimes the $\sum$ operator to represent a general n-ary (or even infinitary) sum operator.

Prefix	$\dfrac{-}{\mu.E \xrightarrow{\mu} E}$
Sum	$\dfrac{E_1 \xrightarrow{\mu} E_1'}{E_1 + E_2 \xrightarrow{\mu} E_1'} \qquad \dfrac{E_2 \xrightarrow{\mu} E_2'}{E_1 + E_2 \xrightarrow{\mu} E_2'}$
Parallel	$\dfrac{E_1 \xrightarrow{\mu} E_1'}{E_1 \vert E_2 \xrightarrow{\mu} E_1' \vert E_2} \qquad \dfrac{E_2 \xrightarrow{\mu} E_2'}{E_1 \vert E_2 \xrightarrow{\mu} E_1 \vert E_2'}$ $\dfrac{E_1 \xrightarrow{\alpha} E_1' \quad E_2 \xrightarrow{\bar{\alpha}} E_2'}{E_1 \vert E_2 \xrightarrow{\tau} E_1' \vert E_2'}$
Restriction	$\dfrac{E \xrightarrow{\mu} E'}{E \setminus L \xrightarrow{\mu} E' \setminus L}$ if $\mu \notin L \cup \bar{L}$
Input Restriction	$\dfrac{E \xrightarrow{\mu} E'}{E \setminus_I L \xrightarrow{\mu} E' \setminus_I L}$ if $\mu \notin L \cap I$
Hiding	$\dfrac{E \xrightarrow{\mu} E'}{E/L \xrightarrow{\mu} E'/L}$ if $\mu \notin L$ $\qquad \dfrac{E \xrightarrow{\mu} E'}{E/L \xrightarrow{\tau} E'/L}$ if $\mu \in L$
Relabelling	$\dfrac{E \xrightarrow{\mu} E'}{E[f] \xrightarrow{f(\mu)} E'[f]}$
Constant	$\dfrac{E \xrightarrow{\mu} E'}{A \xrightarrow{\mu} E'}$ if $A \overset{\text{def}}{=} E$

Table 1: The operational rules for SPA.

Definition 2.2 *The* system composition *of E and F is defined as* $E||F \stackrel{\text{def}}{=} (E|F) \setminus (\mathcal{L}(E) \cap \mathcal{L}(F))$.

Let us now introduce execution traces and trace equivalence on SPA process terms.

Definition 2.3 *The expression* $E \stackrel{\alpha}{\Longrightarrow} E'$ *is a shorthand for* $E(\stackrel{\tau}{\rightarrow})^* E_1 \stackrel{\alpha}{\rightarrow} E_2(\stackrel{\tau}{\rightarrow})^* E'$, *where* $(\stackrel{\tau}{\rightarrow})^*$ *denotes a (possibly empty) sequence of* τ *labelled transitions. Let* $\gamma = \alpha_1 \ldots \alpha_n \in \mathcal{L}^*$ *be a sequence of actions; then* $E \stackrel{\gamma}{\Longrightarrow} E'$ *if and only if there exist* $E_1, E_2, \ldots, E_n \in \mathcal{E}$ *such that* $E \stackrel{\alpha_1}{\Longrightarrow} E_1 \stackrel{\alpha_2}{\Longrightarrow} \cdots \stackrel{\alpha_n}{\Longrightarrow} E_n$ *and* $E_n = E'$. *When* $E \stackrel{\gamma}{\Longrightarrow} E'$ *we say that* E' *is* reachable *from* E.

Definition 2.4 *For any* $E \in \mathcal{E}$ *the set* $T(E)$ *of* traces associated with *E is defined as follows:* $T(E) = \{\gamma \in \mathcal{L}^* \mid \exists E' : E \stackrel{\gamma}{\Longrightarrow} E'\}$. *$E$ and F are* trace equivalent *(notation* $E \approx_T F$*) if and only if* $T(E) = T(F)$.

The next proposition, proved in [26], shows that trace equivalence is preserved by the parallel and restriction operators.

Proposition 2.5 *If* $E_1 \approx_T E_2$ *and* $F_1 \approx_T F_2$ *then:*

- *(i)* $E_1|F_1 \approx_T E_2|F_2$
- *(ii)* $E_1 \setminus L \approx_T E_2 \setminus L$

A few auxiliary definitions will be useful in the following. Let $\gamma, \gamma' \in \mathcal{L}^*$ be two sequences over $\mathcal{L}$. γ' is a *subsequence* of γ (denoted $\gamma' \prec \gamma$) if and only if γ and γ' are in the form:

$$\gamma = \alpha_1 \ldots \alpha_n, \ \gamma' = \alpha_{k_{\gamma',\gamma}(1)} \ldots \alpha_{k_{\gamma',\gamma}(m)} \quad \text{with } m \leq n$$

where $k_{\gamma',\gamma} : [1, m] \rightarrow [1, n]$, is an increasing monotone function. With abuse of notation, $\alpha_i \in \gamma$ stands for α_i *occurs in the sequence* γ.

The following functions will be used to define some security properties.

Definition 2.6 *The functions* low, highinput, lowinput *and* input *are defined as follows:*

- $low : \mathcal{L}^* \rightarrow Act_L^*$;
  $\gamma' = low(\gamma)$, $\gamma' \prec \gamma$ *is generated as follows:*

$$\forall \alpha_i \in \gamma, \ \alpha_i \in Act_L \Rightarrow \exists j : i = k_{\gamma',\gamma}(j)$$

- $highinput : \mathcal{L}^* \rightarrow (Act_H \cap I)^*$;
  $\gamma' = highinput(\gamma)$, $\gamma' \prec \gamma$ *is generated as follows:*

$$\forall \alpha_i \in \gamma, \ \alpha_i \in Act_H \cap I \Rightarrow \exists j : i = k_{\gamma',\gamma}(j)$$

- *lowinput* : $\mathcal{L}^* \rightarrow (Act_L \cap I)^*$;
  $\gamma' = lowinput(\gamma)$, $\gamma' \prec \gamma$ *is generated as follows:*

$$\forall \alpha_i \in \gamma,\ \alpha_i \in Act_L \cap I \Rightarrow \exists j : i = k_{\gamma',\gamma}(j)$$

- *input* : $\mathcal{L}^* \rightarrow I^*$;
  $\gamma' = input(\gamma)$, $\gamma' \prec \gamma$ *is generated as follows:*

$$\forall \alpha_i \in \gamma,\ \alpha_i \in I \Rightarrow \exists j : i = k_{\gamma',\gamma}(j)$$

They extract from a sequence γ the low, high input, low input and input subsequence, respectively.

Before starting our classification, we introduce a property called *Input-Totality* (IT for short) which has a relevant impact on the definition of security properties.

Definition 2.7 *A system $F \in \mathcal{E}$ is* Input-Total *if $\forall E \in \mathcal{E}$ reachable from F and $\forall i \in I$, $\exists E' \in \mathcal{E}$ such that $E \stackrel{i}{\Longrightarrow} E'$. $\mathcal{E}_{IT}$ denotes the set of all Input-Total systems.*

If a system is not IT, then a user could in principle know something about the state of the system through the inputs it accepts. For this reason, some of the existing properties assume IT.

3. Trace-based Security Properties

In this Section, we present the following information flow security properties: *Non-Deterministic Non Interference* (Section 3.1); *Two-Level Non Deducibility on Inputs* (Section 3.2); *lts-Restrictiveness* (Section 3.3); *lts-Correctability* (Section 3.4); and *Non Deducibility on Composition* (Section 3.5). Moreover, in Section 3.6, we study in detail the effects of the Input-Totality assumption, mainly over the *NDC* security property.

3.1. *Non-deterministic Non Interference*

The basic idea of *Non Interference* (*NI*, for short) [10] is the following: Let G and G' be two user groups. Given any input sequence γ, let γ' be its subsequence obtained by deleting all the actions of users in G. G is *non interfering with G'* iff for every input sequence γ, the users in G' obtain the same outputs after the execution of γ and γ'.

The original definition is based on a deterministic model where, for every input sequence, a user receives a unique output. There, an action is a pair (u, c) where u is a user and c is a command; in our approach, a command a executed by two different users u_1 and u_2 is represented by the actions $a_1 = (u_1, a)$ and $a_2 = (u_2, a)$. So, two user groups can be characterized by the sets Act_H and Act_L of actions they use.

The natural generalization to the lts model (assuming two user groups only) is as follows: The high level does not interfere with the low level if and only if for every trace γ, there exists a trace γ', containing no high level input actions, such that $low(\gamma') = low(\gamma)$.

Definition 3.1 *(NNI: Non-deterministic Non Interference)*
$E \in NNI$ if and only if $\forall \gamma \in T(E), \exists \delta \in T(E)$ such that

(*i*) $low(\gamma) = low(\delta)$

(*ii*) $highinput(\delta) = \langle\rangle$

where $\langle\rangle$ denotes the empty sequence.

The next proposition presents an algebraic characterization of *NNI*, based on SPA and trace semantics.

Proposition 3.2 $E \in NNI \iff (E \setminus_I Act_H)/Act_H \approx_T E/Act_H$.

Proof. ($\Rightarrow$) It is enough to show that if E is *NNI* then $T(E/Act_H) \subseteq T((E \setminus_I Act_H)/Act_H)$, because the opposite inclusion is trivial. Let $\gamma \in T(E/Act_H)$; then $\exists \gamma' \in T(E)$ such that $low(\gamma') = \gamma$. Since $E \in NNI$ then $\exists \delta \in T(E)$ such that $low(\gamma') = low(\delta)$ and $highinput(\delta) = \langle\rangle$. Hence $\delta \in T(E \setminus_I Act_H)$ and $\exists \delta' = low(\delta) \in T((E \setminus_I Act_H)/Act_H)$. Since $low(\gamma') = \gamma, low(\gamma') = low(\delta)$ and $low(\delta) = \delta'$, then $\gamma = \delta'$ and thus $\gamma \in T((E \setminus_I Act_H)/Act_H)$.
($\Leftarrow$) Let $\gamma \in T(E)$. Then $\exists \delta \in T(E \setminus_I Act_H)$ such that $low(\gamma) = low(\delta)$. Since $\delta \in T(E \setminus_I Act_H)$ then $highinput(\delta) = \langle\rangle$ and $\delta \in T(E)$. $\Box$

A more restrictive form of *NNI* requires that, for every trace γ, the sequence γ', obtained from γ deleting all the high level actions (inputs and outputs), is still a trace. The need for an explicit protection of high level outputs was argued in [13]. The particular property we are going to propose, called *Strong NNI* (*SNNI* for short), is similar to a number of generalizations [29, 1, 12] of *NI* over the CSP [15] process algebra. [5]

Definition 3.3 *(SNNI: Strong NNI).*

$$E \in SNNI \iff \forall \gamma \in T(E),\ low(\gamma) \in T(E).$$

The SPA version of *SNNI* is as follows:

Proposition 3.4 $E \in SNNI \iff E/Act_H \approx_T E \setminus Act_H$.

Proof. ($\Rightarrow$) It is sufficient to show that if E is *SNNI* then $T(E/Act_H) \subseteq T(E \setminus Act_H)$, because the opposite inclusion is trivial. Let $\gamma \in T(E/Act_H)$; then $\exists \gamma' \in T(E)$ such that $low(\gamma') = \gamma$. Since $E \in SNNI$ then $\gamma \in T(E)$. Hence $\gamma \in T(E \setminus Act_H)$ because $\forall \alpha \in \gamma, \alpha \notin Act_H$.
($\Leftarrow$) if $\gamma \in T(E)$ then $\exists \delta \in T(E/Act_H)$ such that $\delta = low(\gamma)$. Since $T(E \subseteq Act_H) = T(E/Act_H)$ then $\delta \in T(E \subseteq Act_H)$. Since $T(E \subseteq Act_H) \subseteq T(E)$ then $\delta \in T(E)$. $\Box$

---

[5] In CSP actions are not clearly distinguished as inputs or outputs; hence, *NNI* and *SNNI* are the same property in this case.

This notion, in the SPA approach, seems more appropriate than *NNI* since it directly protects high level outputs. In fact, it can detect deadlocks due to high level communications. As an example, consider the system $E = \bar{h}.\bar{l}.0$ which is *NNI* but not *SNNI*; if we compose E with a high level user $F = h'.h.0$ we obtain $E||F$, where E must wait for synchronization with F before it can execute $\bar{l}$; hence the low output $\bar{l}$ must wait for high level input h', generating an illegal information flow. Note that, in the previous example, we have considered a high level user as a process connected to the system as usually done in the process algebraic setting; we reconsider this point of view in Section 3.5.

Proposition 3.5 *SNNI* $\subset$ *NNI*.

Proof. If $E \in \mathit{SNNI}$ then $E/\mathit{Act}_H \approx_T E \setminus \mathit{Act}_H$. Since $T(E \setminus \mathit{Act}_H) \subseteq T((E \setminus_I \mathit{Act}_H)/\mathit{Act}_H)$ and $T((E \setminus_I \mathit{Act}_H)/\mathit{Act}_H) \subseteq T(E/\mathit{Act}_H)$ then $E \in \mathit{NNI}$. The inclusion is strict because $E = \bar{h}.l.0$ is *NNI* but not *SNNI*. □

The next proposition lists some properties of *NNI* and *SNNI*, whose proof requires the following simple lemma.

Lemma 3.6 $(E/\mathit{Act}_H)|(F/\mathit{Act}_H) \approx_T (E|F)/\mathit{Act}_H$.

Proof. We have to prove that $T((E/\mathit{Act}_H)|(F/\mathit{Act}_H)) = T((E|F)/\mathit{Act}_H)$. The inclusion $\subseteq$ is trivial. For the inclusion $\supseteq$, the only interesting case is the high synchronization $E|F \xrightarrow{\tau} E'|F'$ with $E \xrightarrow{h} E'$ and $F \xrightarrow{\bar{h}} F'$. It is sufficient to consider $E/\mathit{Act}_H \xrightarrow{\tau} E'/\mathit{Act}_H$ and $F/\mathit{Act}_H \xrightarrow{\tau} F'/\mathit{Act}_H$ to obtain $(E/\mathit{Act}_H)|(F/\mathit{Act}_H) \xRightarrow{\tau.\tau} (E'/\mathit{Act}_H)|(F'/\mathit{Act}_H)$. □

Proposition 3.7

- (i) $E, F \in \mathit{NNI} \Rightarrow E \mid F \in \mathit{NNI}$
- (ii) $E, F \in \mathit{SNNI} \Rightarrow E \mid F \in \mathit{SNNI}$
- (iii) $E \in \mathit{SNNI} \Rightarrow E \setminus L \in \mathit{SNNI}$*, where* $L \subseteq \mathcal{L}$
- (iv) $E, F \in \mathit{SNNI} \Rightarrow E||F \in \mathit{SNNI}$

Proof. (i) We have $(E|F)/\mathit{Act}_H \approx_T (E/\mathit{Act}_H)|(F/\mathit{Act}_H)$ (by Lemma 3.6) and $(E/\mathit{Act}_H)|(F/\mathit{Act}_H) \approx_T ((E \setminus_I \mathit{Act}_H)/\mathit{Act}_H)|((F \setminus_I \mathit{Act}_H)/\mathit{Act}_H)$, by hypothesis and Proposition 2.5.(i). By Lemma 3.6, we have that $((E \setminus_I \mathit{Act}_H)/\mathit{Act}_H)|((F \setminus_I \mathit{Act}_H)/\mathit{Act}_H) \approx_T ((E \setminus_I \mathit{Act}_H)|(F \setminus_I \mathit{Act}_H))/\mathit{Act}_H$. Since $T((E \setminus_I \mathit{Act}_H | F \setminus_I \mathit{Act}_H)/\mathit{Act}_H) \subseteq T(((E|F) \setminus_I \mathit{Act}_H)/\mathit{Act}_H)$, then we have $T((E|F)/\mathit{Act}_H) \subseteq T(((E|F) \setminus_I \mathit{Act}_H)/\mathit{Act}_H)$. Note that $T(((E|F) \setminus_I \mathit{Act}_H)/\mathit{Act}_H) \subseteq T((E|F)/\mathit{Act}_H)$ and so we have that $(E|F)/\mathit{Act}_H \approx_T ((E|F) \setminus_I \mathit{Act}_H)/\mathit{Act}_H$.
(ii) We have $(E|F)/\mathit{Act}_H \approx_T (E/\mathit{Act}_H)|(F/\mathit{Act}_H)$ by Lemma 3.6 and $(E/\mathit{Act}_H)|(F/\mathit{Act}_H) \approx_T (E \setminus \mathit{Act}_H)|(F \setminus \mathit{Act}_H)$, by hypothesis and Proposition 2.5.(i). Since $(E \setminus \mathit{Act}_H)|(F \setminus \mathit{Act}_H) \approx_T ((E \setminus \mathit{Act}_H)|(F \setminus \mathit{Act}_H)) \setminus$

Act_H and $T((E \setminus Act_H | F \setminus Act_H) \setminus Act_H) \subseteq T((E|F) \setminus Act_H)$, then we have $T((E|F)/Act_H) \subseteq T((E|F) \setminus Act_H)$. Note that $T((E|F) \setminus Act_H) \subseteq T((E|F)/Act_H)$ and so $(E|F) \setminus Act_H \approx_T (E|F)/Act_H$.
(*iii*) Observe that $T((E \setminus L)/Act_H) \subseteq T((E/Act_H) \setminus L)$; by hypothesis and Proposition 2.5.(*ii*) we have $(E/Act_H) \setminus L \approx_T (E \setminus Act_H) \setminus L$. Observe also that $(E \setminus Act_H) \setminus L \approx_T (E \setminus L) \setminus Act_H$. Hence $T((E \setminus L)/Act_H) \subseteq T((E \setminus L) \setminus Act_H)$. Hence, we have $(E \setminus L)/Act_H \approx_T (E \setminus L) \setminus Act_H$, as the opposite inclusion is obvious,
(*iv*) Trivial by (*ii*) and (*iii*). □

3.2. *Two-level Non Deducibility on Inputs*

Non Deducibility on Inputs (*NDI*, for short) [31] has been proposed as a non deterministic alternative to *NI*. The basic idea is simple and, abstractly, can be described independently of the system model. It assumes the existence of some *information functions* which represent the user views of the state of a system. Two information functions F_1 and F_2 are *independent* (or we could say that *information does not flow* from F_2 to F_1) if and only if the function $F_1 \times F_2 : W \rightarrow img(F_1) \times img(F_2)$ is onto, [6] where W is some representation of the behaviour of the system and $img(F_i)$ is the observable part for a certain user. Hence, a system is secure if and only if any possible information flow between two information functions is legal.

In [31], the authors of *NDI* define two particular information functions, *hidden* and *view*, which depend on the actual model they exploit. Here, the state of the system is determined by the executed input and output actions; so, the domain W of information functions is the set of traces; *hidden* and *view* simply correspond to *highinput* and *low* restricted on the set $T(E)$.

Definition 3.8 *(TNDI: Two-level Non Deducibility on Inputs)*
$E \in TNDI$ if and only if there is no information flow from hidden to view (i.e., hidden $\times$ view is onto).

In other words, for every $\gamma \in Act_L^*$ such that there exists a trace $t \in T(E)$, with $\gamma = low(t)$ and for every $\gamma' \in (Act_H \cap I)^*$ such that there exists a trace $t' \in T(E), with\, \gamma' = highinput(t')$, there must exist a trace which has γ as lowview and γ' as highinput view. Hence a low level user, who knows only the low level views of the system, can deduce nothing about the high level inputs.

Proposition 3.9 *$TNDI \subset NNI$.*

Proof. Let $E \in TNDI$, $\gamma \in T(E)$ and $\gamma' = low(\gamma)$. Since $\langle\rangle \in T(E)$ and $highinput(\langle\rangle) = \langle\rangle$, then, by hypothesis, $\exists \delta \in T(E)$ such that $low(\delta) = \gamma'$ and $highinput(\delta) = \langle\rangle$. Hence, $\forall \gamma \in T(E)$, $\exists \delta \in T(E)$ such that $low(\gamma) = low(\delta)$, $highinput(\delta) = \langle\rangle$. Hence *$TNDI \subseteq NNI$.*

---

[6] There is information flow from F_2 to F_1 when, for a fixed $w \in W$, there exists $z \in img(F_2)$ such that for any $w' \in W, F_1(w') = F_1(w)$ implies $F_2(w') \neq z$. Note that information flow is a *symmetric* concept.

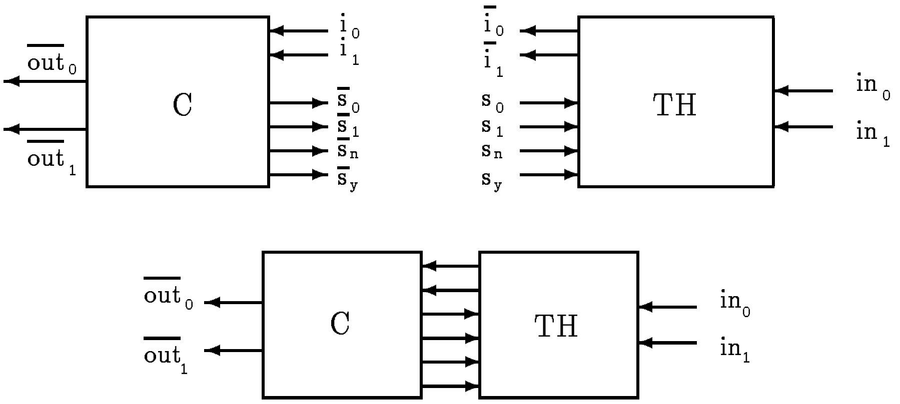

Figure 1: System Composition.

To show that *NNI* $\not\subseteq$ *TNDI*, consider $E = h.l.0 + l.0 + l'.0$ where $\{h\} \subseteq Act_H \cap I$ and $\{l, l'\} \subseteq Act_L \cap I$. E is *NNI* but not *TNDI*, as there is no trace containing both h and l'. □

In [31], exactly the opposite inclusion is proved: *NDI*, when transposed to the *NI* system model, contains *NI*. This is essentially due to the following two facts: (*i*) In the original definition, systems are considered Input-Total (if we require that *NNI* and *TNDI* systems are IT, then it can be proved that $NNI \cap \mathcal{E}_{IT} = TNDI \cap \mathcal{E}_{IT}$; see Corollary 3.29). (*ii*) We assume a partition in two security levels while, in *NI*, the two user groups which does (or does not) interfere are not a partition of the users.

Proposition 3.10 *SNNI* $\not\subseteq$ *TNDI.*

Proof. Agent $E = h.l.0 + l.0 + l'.0$ is *SNNI* (trivial) but not *TNDI* (there is no trace containing both h and l'). □

The next example – inspired to [25] – shows that *TNDI* is not closed with respect to system composition.

Example 3.11 Consider the sets $\{i_1, i_0, in_1, in_0, s_1, s_0, s_n, s_y\} \subseteq Act_H$ and $\{out_1, out_0\} \subseteq Act_L$. Let C_1 be the following SPA agent:

$$C_1 \stackrel{\text{def}}{=} \overline{s_1}.(i_1.C_1 + \overline{out_1}.C_1) + \overline{s_n}.i_1.\overline{out_1}.C_1$$

C_1 is an half-bit encryption machine, that is, depending on s_n and s_1 it emits or not a $\overline{out_1}$ signal corresponding to a i_1 signal. It is possible to construct a 1-bit encryption machine C by (roughly) adding a dual system C_0 to C_1:

$$C_0 \stackrel{\text{def}}{=} \overline{s_0}.(i_0.C_0 + \overline{out_0}.C_0) + \overline{s_n}.i_0.\overline{out_0}.C_0$$

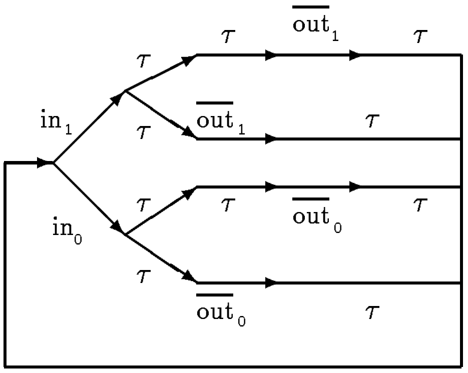

Figure 2: The lts of the Composed System.

Furthermore we introduce a synchronization signal $\overline{s_y}$ used to make simpler the composition with process *TH* below. Such signal $\overline{s_y}$ can be considered as part of the key. The resulting C is:

$$
\begin{aligned}
C &\stackrel{\mathrm{def}}{=} \overline{s_n}.(i_1.\overline{out_1}.C' + i_0.\overline{out_0}.C') + \overline{s_1}.(i_1.C' + \overline{out_1}.C') + \\
&\quad + \overline{s_0}.(i_0.C' + \overline{out_0}.C') \\
C' &\stackrel{\mathrm{def}}{=} \overline{s_y}.C
\end{aligned}
$$

In system C (Figure 1) a user who can see only low outputs (out_0 and out_1) can deduce nothing about high inputs (i_0 and i_1). So such a system is *TNDI* because any low sequence (encrypted message) is consistent with any high input sequence (original message), for some choice of internally generated keys. Let *TH* be the following SPA agent, to be composed with machine C:

$$
\begin{aligned}
TH &\stackrel{\mathrm{def}}{=} in_1.(s_n.\overline{i_1}.TH' + s_1.TH') + in_0.(s_n.\overline{i_0}.TH' + s_0.TH') \\
TH' &\stackrel{\mathrm{def}}{=} s_y.TH
\end{aligned}
$$

TH executes only high actions, so it is secure with respect to any definition; In particular, $TH \in TNDI$. Let $C||TH$ be the system composition of C and *TH*; its lts is reported in Figure 2. This process is not *TNDI* in fact it produces only traces where low output signals depend on the previous high inputs.

3.3. *lts-Restrictiveness*

Restrictiveness (*RES*, for short) [21, 20, 22] has been proposed as a compositional (w.r.t. system composition) specialization of *NDI*. The basic idea of *RES* is as follows: A system is secure if there exists an equivalence relation on the states such that a low level user cannot recognize state equivalence, i.e. low level states are equivalent classes of states. Therefore *RES* must satisfy the following rules: (i) a high level input moves the system from a state to an equivalent one (high inputs do not change low level states); (*ii*) a low level input gives no information about the state of the system and so it moves the system from equivalent states to equivalent states.

The model used in the original definition is similar to lts. They differ only in the composition of systems, as in such a model the result of a synchronization is a normal output. Since *RES* treats in a different way high and low level outputs — and consequently high and low level synchronizations — then we need a new invisible action τ_L, playing the role of τ in low level synchronizations, while τ is used for high level synchronizations.

In *RES* the Input-Total property is explicitly required: if absent, a low level user could distinguish two equivalent states through their different input acceptance sets. Moreover, it is a necessary condition for compositionality.

Definition 3.12 *A system E is* lts-Restrictive *(lts-RES) if and only if it is Input-Total and there exists an equivalence relation $\equiv\ \subseteq \mathcal{E} \times \mathcal{E}$ such that for all E_1 and E_2 reachable from E:*

1. $\forall e \in Act_H \cap I : E_1 \xrightarrow{e} E_1' \Rightarrow E_1 \equiv E_1'$
2. $\forall e \in Act_L \cap I : E_1 \xrightarrow{e} E_1',\ E_1 \equiv E_2 \Rightarrow \exists E_2' : E_2 \xrightarrow{e} E_2',\ E_1' \equiv E_2'$
3. $\forall e \in Act_H \cap O \cup \{\tau\} : E_1 \xrightarrow{e} E_1',\ E_1 \equiv E_2 \Longrightarrow \exists E_2',\ \exists \gamma' \in (Act_H \cap O)^* : E_2 \overset{\gamma'}{\Longrightarrow} E_2',\ E_1' \equiv E_2'$
4. $\forall e \in Act_L \cap O \cup \{\tau_L\},\ \forall \gamma, \delta \in (Act_H \cap O)^* : E_1 \overset{\gamma e \delta}{\Longrightarrow} E_1',\ E_1 \equiv E_2 \Longrightarrow \exists E_2',\ \exists \gamma', \delta' \in (Act_H \cap O \cup \{\tau\})^* :\ E_2 \xrightarrow{\gamma e \delta} E_2',\ E_1' \equiv E_2'$

Proposition 3.13

(*i*) $E, F \in$ *lts-RES* $\Rightarrow E|F \in$ *lts-RES.*

(*ii*) $E, F \in$ *lts-RES* $\Rightarrow E||F \in$ *lts-RES.*

Proof. (*i*) It is sufficient to define the relation $\equiv_{E|F}$ in this way:

$$E_1|E_2 \equiv_{E|F} E_1'|E_2' \Leftrightarrow E_1 \equiv_E E_1',\ E_2 \equiv_F E_2'$$

Obviously, system $E|F$ is still Input-Total. Let us prove the four conditions:

1. High input: Let us focus on the case in which the first process executes the input (the other case is symmetric). So $E_1|E_2 \xrightarrow{e} E_1'|E_2$. By hypothesis $E_1 \equiv_E E_1'$, then $E_1|E_2 \equiv_{E|F} E_1'|E_2$.
2. Low input: Let $P = E_1|E_2$ and $P' = E_1'|E_2$ where $P \xrightarrow{e} P'$. Let $P'' = E_1''|E_2''$ such that $P'' \equiv_{E|F} P$. It must be proved that there exists P''' such that $P'' \xrightarrow{e} P'''$ and $P''' \equiv_{E|F} P'$. It is sufficient to

construct P''' as follows: $P'' \equiv_{E|F} P \Rightarrow E_1 \equiv_E E_1''$ then $\exists E_1'''$ such that $E_1'' \xrightarrow{e} E_1'''$ and $E_1''' \equiv_E E_1'$. Let $P''' = E_1'''|E_2''$.

The proof scheme is the following:

$$\begin{tikzcd} E_1|E_2 \arrow[rr, "e"] & & E_1'|E_2 \\ \equiv & \Longrightarrow & \equiv \\ E_1''|E_2'' \arrow[rr, "e"] & & E_1'''|E_2'' \end{tikzcd}$$

The case in which the input signal is executed by the second process is symmetric.

3. High output or internal high signal: there are two subcases:

   (a) High output: let $P = E_1|E_2$ and $P' = E_1'|E_2$ where $P \xrightarrow{e} P'$. Let $P'' = E_1''|E_2''$ such that $P'' \equiv_{E|F} P$. It must be proved that there exist P''' and γ such that $P'' \stackrel{\gamma}{\Longrightarrow} P'''$ and $P''' \equiv_{E|F} P'$. It is sufficient to construct P''' as follows: $P'' \equiv_{E|F} P \Rightarrow E_1 \equiv_E E_1''$ then $\exists E_1'''$, $\exists \gamma$ such that $E_1'' \stackrel{\gamma}{\Longrightarrow} E_1'''$ and $E_1''' \equiv_E E_1'$. Let $P''' = E_1'''|E_2''$.

   The proof scheme is:

$$\begin{tikzcd} E_1|E_2 \arrow[rr, "e"] & & E_1'|E_2 \\ \equiv & \Longrightarrow & \equiv \\ E_1''|E_2'' \arrow[rr, Rightarrow, "\gamma"] & & E_1'''|E_2'' \end{tikzcd}$$

   (b) $e = \tau$: there are three subcases:

      i. τ is executed by the first process: $E_1|E_2 \xrightarrow{\tau} E_1'|E_2$. The proof is the same as in case 3(a).

      ii. τ is executed by the second process: $E_1|E_2 \xrightarrow{\tau} E_1|E_2'$. The proof is the same as in case 3(a).

      iii. τ is an high level synchronization: $E_1|E_2 \xrightarrow{\tau} E_1'|E_2'$, $E_1 \xrightarrow{i} E_1'$, $E_2 \xrightarrow{\bar{i}} E_2'$. The proof scheme is the following:

$$\begin{tikzcd} E_1|E_2 \arrow[rr, "\tau"] & & E_1'|E_2' \\ \equiv & \Longrightarrow & \equiv \\ E_1''|E_2'' \arrow[rr, Rightarrow, "\gamma"] & & E_1''|E_2''' \end{tikzcd}$$

      in fact $E_1'' \equiv E_1'$ because $E_1 \equiv E_1'$ and $E_2''' \equiv E_2'$ because $\bar{i}$ is an high level output (see case 3(a)).

4. Low output or internal low signal: the proof is very similar to case 3 and thus omitted.

(*ii*) The proof is an extension of (*i*). The only difference is in cases 3 and 4, where it is requested the existence of a particular sequence γ of signals. In fact such sequences could contain a signal $i \in \mathcal{L}(E) \cap \mathcal{L}(F)$ and so be rejected by restriction. However, since they are composed only of high level output signals then for every signal in $\mathcal{L}(E) \cap \mathcal{L}(F)$ the other system can execute the corresponding input (remember that we assume IT) and synchronize moving to an equivalent state (as it executes a high level input). □

3.4. *lts-Correctability*

Correctability [17] has been proposed as a compositional property which is in between *NDI* and *RES*. The basic idea of correctability is the following: A system is correctable if for every trace α, every perturbation α' (obtained from α adding or deleting high level inputs) can be corrected adding or deleting high level outputs in order to obtain a trace. A variant of this definition is *forward correctability* [17] where this correction can be done only after the perturbed input.

In the model (set of traces) used for the original definition [17] the *Input-Extendible* (IE, for short) property is assumed: $\forall E \in \mathcal{E}, \forall \gamma \in T(E), \forall i \in I$ the trace $\gamma.i$ must be a trace in $T(E)$. IE was introduced with the intention of modeling Input-Totality. As a matter of fact, IE and IT are the same property on traces, but the latter is strictly stronger than the former on lts. Here we prefer to assume IT, mainly for uniformity.

Given a trace $\gamma \in T(E)$, a *perturbation* γ' of γ is a sequence obtained adding or deleting high input actions $i \in Act_H \cap I$. Given $\gamma \in Act^*$, a *correction* γ' of γ is a sequence obtained adding or deleting high output actions $o \in Act_H \cap O$ such that $\gamma' \in T(E)$.

Definition 3.14 *A system $E \in \mathcal{E}_{IT}$ is* lts-correctable *if for all $\gamma \in T(E)$ and for all γ' perturbation of γ, there exists a correction γ'' of γ'.*

An occurrence of an action $\alpha \in \mathcal{L}$ in a sequence $\delta \in \mathcal{L}^*$ *closely precedes* a segment γ of δ if δ is of the form $\gamma'.\alpha.[i].\gamma$ where $[i]$ denotes the optional presence of a low level action $i \in Act_L \cap I$.

Definition 3.15 *A system E is* lts Forward Correctable *(lts-FC) if and only if $E \in \mathcal{E}_{IT}$ and for all $\delta \in T(E)$, for all δ' obtained inserting or deleting a high level input $i \in Act_H \cap I$ closely preceding a segment γ such that $highinput(\gamma) = \langle\rangle$, there exists a correction of δ' supported in (i.e., which involves only actions in) γ.*

The model used in [17] is very similar to ours, so it is possible to prove the following results using the same proof lines reported there.

Proposition 3.16

(*ii*) $E \in$ *lts-FC* $\Rightarrow E$ *lts-correctable and E lts-correctable* $\not\Rightarrow E \in$ *lts-FC.*

(*ii*) $E, F \in$ *lts-FC* $\Rightarrow E || F \in$ *lts-FC.*

Theorem 3.17

(*i*) *lts-RES* $\subset$ *lts-FC*.

(*ii*) E *lts-correctable* $\Rightarrow E \in$ *TNDI*.

(*iii*) *lts-RES* $\cap$ *SNNI* $\neq \emptyset$.

(*iv*) *lts-RES* $\not\subseteq$ *SNNI*

Proof. (*i*) Let $\delta = \gamma'.\alpha.i.\gamma$ be a trace of E, where $\alpha \in I \cap Act_H$, $i \in I \cap Act_L$ and $highinput(\gamma) = \langle\rangle$, and $\delta' = \gamma'.i.\gamma$ be the perturbation of δ obtained deleting the high input α. Then consider E' such that $E \stackrel{\gamma'}{\Longrightarrow} E'$ and E'' such that $E' \stackrel{\alpha}{\Longrightarrow} E''$. By item 1 of *lts-RES* definition we have $E' \equiv E''$. So, (since γ does not contain high level inputs) for items 2, 3 and 4 of *lts-RES* definition, there exists $\tilde{\gamma}$ which differs from γ only for high level outputs and such that $\gamma'.i.\tilde{\gamma}$ is a trace of E. The cases where the input is added and there is not i (which is optional in *lts-FC* definition) are analogously proved. The inclusion is proper because agent $Z = \sum_{i \in I} i.Z + \bar{l}.\Pi_0 + h.\Pi_0$ with $\Pi_0 = \sum_{i \in I} i.\Pi_0$ is *lts-FC* (because every perturbation of a trace is still a trace) but not *lts-RES* (because high input h can move system Z to Π_0 but Z, which can execute low level output $\bar{l}$, is not equivalent to Π_0).
(*ii*) Let $\gamma, \gamma' \in T(E)$, $\lambda = low(\gamma)$ and $\delta = highinput(\gamma')$. Consider a perturbation γ'' of γ obtained by deleting all high inputs $i \in Act_H \cap I$ and by adding all the actions of δ (only high input) in the same order. Let $\tilde{\gamma} \in T(E)$ be a correction of the obtained perturbation γ''. Trace $\tilde{\gamma}$ is such that $highinput(\tilde{\gamma}) = \delta$ and $low(\tilde{\gamma}) = \lambda$, hence E is *TNDI*.
(*iii*) It is sufficient to consider the Input-Total agent $\Pi_0 = \sum_{i \in I} i.\Pi_0$.
(*iv*) Let Z be defined as $Z = \sum_{i \in I} i.Z + \bar{h}.Z'$ and $Z' = \sum_{i \in I} i.Z' + \bar{l}.\Pi_0$ where Π_0 is defined in (*iii*). Z is *lts-RES* (there are three equivalence classes represented by Z,Z' and Π_0) but not *SNNI*. □

3.5. *Non Deducibility on Compositions*

In [7] we proposed *Non Deducibility on Compositions* (*NDC*, for short) as a transposition on the lts model of *Non Deducibility on Strategies* (*NDS*, for short) [32].

The need for *NDS* originated from the following observation: *NDI* is too weak because it does not take into account the feedback from the system. Indeed, *NDI* does not verify the independence between high processes – called high *strategies* by the authors (a strategy interacts with system and decide the next high input knowing the previous high inputs and outputs) – and low sequences of inputs and outputs. [7] The basic idea of *NDS* can be summarized as follows: A system is *NDS* if, when composed to any strategy Π, all of its low level views are still possible, i.e., if Π does not restrict the possible low executions.

---

[7] The encryption machine C in Example 3.11 is *NDI* but not *NDS* as its low executions can be affected by a high process *TH* interacting with it.

The original definition [32] is based on a very restrictive synchronous model in which, at every clock signal, the system takes an input from every input channel and emits an output to every output channel. In such a model, a strategy is simply a function which at time n computes a new high level input using the $n-1$ long sequence of previous high actions. In SPA, a strategy cannot be represented as a function. In fact, when a system emits an output, it must wait for a corresponding external input which could never arrive: a *deadlock* may arise on a system output, because of the hand-shake nature of SPA communication.

As a strategy is something that, looking at previous high inputs and outputs, decides the new high input, it may be seen as an active high level process interacting with the system. Fixing a certain strategy Π on a system E may be seen as considering the system $(E|\Pi) \setminus Act_H$. So a system E is *NDC* if for every high level process Π the low level view sets of E and $(E|\Pi) \setminus Act_H$ are the same. In other words, a system E is *NDC* if the set of its low level views cannot be modified by composing E with any high level process. To formally define *NDC* it is necessary a new function, *lowviews*, which returns the set of all low level views of a system E: $lowviews : \mathcal{E} \rightarrow \mathcal{P}(Act_L^*)$ is defined as

$$lowviews(E) \stackrel{\text{def}}{=} \{\gamma \in Act_L^* \mid \exists \gamma' \in T(E),\ \gamma = low(\gamma')\}$$

Definition 3.18 *(NDC: Non Deducible on Compositions)*
Let $H \subseteq Act_H$ and $E \in \mathcal{E}$. Then, E is Non Deducible on Compositions for actions in H *(NDC$_H$ for short) if* $\forall \Pi \in \mathcal{E}_H$*, we have that* $lowviews(E) = lowviews((E|\Pi) \setminus H)$ *holds.*
NDC is defined as NDC_H when $H = Act_H$.

Lemma 3.19 $E/Act_H \approx_T F/Act_H$ *iff* $lowviews(E) = lowviews(F)$.

Proof. It is enough to observe that $T(E/Act_H) = lowviews(E)$. □

The SPA characterization of NDC_H is as follows:

Proposition 3.20 *E is NDC_H iff $E/Act_H \approx_T ((E \mid \Pi) \setminus H)/Act_H$, for all $\Pi \in \mathcal{E}_H$.*

Proof. It follows from Lemma 3.19. □

In the theory of process algebra a system is observed by means of testing processes which are composed with it. The precise definition of what the testers can do and the modalities of the interaction define the *testing scenario*. In our opinion, *NDC* is the property that more closely respects this basic intuition: a system is secure if none of the potential testers can affect its low level behaviour. For this reason, we think that *NDC* is probably the most meaningful security property in a process algebraic setting.

The above definition of *NDC* is difficult to use in practice: it is not possible to verify the security of a system connecting to it all strategies and observing their effects on low computations (it is an exhaustive procedure which does not terminate, in general). We need an alternative formulation of *NDC*, which exploits local information, only. It is based on the following theorem, which asserts that the property of "transparence with respect to all strategies attacks" corresponds to a *static* property of an agent E, i.e. to a property that can be verified by examining the traces of the system E, without composing it with any other high process.

Theorem 3.21 *(Static characterization of* NDC_H*)*
Let $H \subseteq Act_H$. *Then,* $E \in NDC_H$ *if and only if* $E/Act_H \approx_T (E \setminus H)/Act_H$.

PROOF. ($\Rightarrow$) By hypothesis, $E/Act_H \approx_T ((E|0) \setminus H)/Act_H$ for the specific $\Pi = 0$. Since $((E|0) \setminus H)/Act_H \approx_T (E \setminus H)/Act_H$ then we have $E/Act_H \approx_T (E \setminus H)/Act_H$.

($\Leftarrow$) By hypothesis, $E/Act_H \approx_T (E \setminus H)/Act_H$. Since $T((E \setminus H)/Act_H) \subseteq T(((E|\Pi) \setminus H)/Act_H)$ then we have $T(E/Act_H) \subseteq T(((E|\Pi) \setminus H)/Act_H)$. Observe also that the reverse inclusion holds, in fact, if E and Π synchronize on a certain high action, then E/Act_H can always "hide" it. Hence $E/Act_H \approx_T ((E|\Pi) \setminus H)/Act_H$. □

We would like to mention that a similar proposal, at least in the intention, was reported in [9], where Foley defines a security property called *weak information flow*.

The following corollary, which is an immediate consequence of Theorem 3.21 and Proposition 3.4, is an important result concerning the formal classification of security definitions: *NDC* is equal to *SNNI*, despite of their very different underlying intuitions. We will see that this equality does no longer hold when considering weak bisimulation.

Corollary 3.22 *NDC = SNNI.* □

Since *SNNI* is equal to *NDC*, the latter inherits all the results of the former, namely compositionality (see Proposition 3.7). Furthermore it directly protects high level outputs, as well as *SNNI* does. It is interesting to verify if also NDC_H is closed with respect to system composition.

Theorem 3.23 *Let* $H \subseteq Act_H$.

(*i*) $E, F \in NDC_H \;\Rightarrow\; E|F \in NDC_H$

(*ii*) $L \subseteq H,\ E \in NDC_H \;\Rightarrow\; E \setminus L \in NDC_H$

(*iii*) $L \subseteq Act_L,\ E \in NDC_H \;\Rightarrow\; E \setminus L \in NDC_H$

(*iv*) $E,\ F \in NDC_H,\ (\mathcal{L}(E) \cap \mathcal{L}(F) \cap Act_H) \subseteq H \;\Longrightarrow\; E||F \in NDC_H$

PROOF. (*i*) We have $(E|F)/Act_H \approx_T (E/Act_H)|(F/Act_H)$, by applying Lemma 3.6, and $(E/Act_H)|(F/Act_H) \approx_T ((E \setminus H)/Act_H)|((F \setminus H)/Act_H)$, by applying hypothesis and Proposition 2.5.(*i*). By Lemma 3.6, we also have $((E \setminus H)/Act_H)|((F \setminus H)/Act_H) \approx_T ((E \setminus H)|(F \setminus H))/Act_H$. Now observe

that $T((E \setminus H | F \setminus H)/Act_H) \subseteq T(((E|F) \setminus H)/Act_H)$. Hence we have $T((E|F)/Act_H) \subseteq T(((E|F) \setminus H)/Act_H)$. Finally note that also $T(((E|F) \setminus H)/Act_H) \subseteq T((E|F)/Act_H)$ and so $(E|F)/Act_H \approx_T ((E|F) \setminus H)/Act_H$.
(*ii*) By hypothesis $E/Act_H \approx_T (E \setminus H)/Act_H$. Since $L \subseteq H$ we have that $E/Act_H \approx_T ((E \setminus L) \setminus H)/Act_H$, and so $T((E \setminus L)/Act_H) \subseteq T(((E \setminus L) \setminus H)/Act_H)$ because $T((E \setminus L)/Act_H) \subseteq T(E/Act_H)$. Observe that $T(((E \setminus L) \setminus H)/Act_H) \subseteq T((E \setminus L)/Act_H)$. Hence we have $((E \setminus L) \setminus H)/Act_H \approx_T (E \setminus L)/Act_H$.
(*iii*) By hypothesis and Proposition 2.5.(*ii*) we have $(E/Act_H) \setminus L \approx_T ((E \setminus H)/Act_H) \setminus L$. Since $L \cap Act_H = \emptyset$ then $(E \setminus L)/Act_H \approx_T ((E \setminus L) \setminus H)/Act_H$.
(*iv*) Trivial by (*i*), (*ii*) and (*iii*). □

Whilst *NDC* is always closed w.r.t. system composition, NDC_H is so if all the involved high actions are in H. This condition is not restrictive – rather it is what we expected – since NDC_H was introduced just to study security on a subset ($H \subseteq Act_H$) of the high actions.

Proposition 3.24

(*i*) $H, H' \subseteq Act_H : H' \subseteq H \Rightarrow NDC_H \subseteq NDC_{H'}$.

(*ii*) *Let* $H \subseteq Act_H$, $H \neq \emptyset$, *then lts-RES* $\not\subseteq NDC_H$.

(*iii*) *Let* $H \subseteq Act_H$, $H \cup \bar{H} \neq Act_H$, *then we have that* $NDC_H \not\subseteq NNI$.

Proof. (*i*) Let $E \in NDC_H$. Then $E/Act_H \approx_T (E \setminus H)/Act_H$. Since $T(E/Act_H) \subseteq T((E \setminus H)/Act_H) \subseteq T((E \setminus H')/Act_H)$ then $E/Act_H \approx_T (E \setminus H')/Act_H$.
(*ii*) It is sufficient to consider the SPA agent $Z = \sum_{i \in I} i.Z + \bar{h}.Z'$ and $Z' = \sum_{i \in I} i.Z' + \bar{l}.\Pi_0$ with $\Pi_0 = \sum_{i \in I} i.\Pi_0$ where $\bar{h} \in H \cup \bar{H}$ and $l \in Act_L$ ($H \neq \emptyset$ so there exists at least one $\bar{h} \in H \cup \bar{H}$). Such system is *lts-RES* but not NDC_H.
(*iii*) It is sufficient to consider the agent $h.l.0$ where $h \in Act_H, h \notin H \cup \bar{H}$, $l \in Act_L$. Such system is NDC_H but not *NNI*. □

3.6. *NDC vs NDS*

NDC may be seen as an alternative to the family of *TNDI*-included definitions: it has a non empty intersection with all definitions but it is not included in *TNDI*. Surprisingly, *NDS* (defined on the Synchronous State Machine model of [32], SSM for short) is between *FC* and *NDI* [25]. It can be shown that these inclusions are due to the restrictiveness of the SSM model which also requires Input-Totality. In fact, if we consider IT systems only, the set *NDC* is “shifted” in between *lts-FC* and *TNDI*.

Indeed, the introduction of the condition of Input-Totality on systems “shifts” set *NDC* rather than “restricting” it. This is because such a condition involves both the system E to be verified *NDC*-secure and high level test-systems Π. So Input-Totality decreases the number of candidates to be *NDC*-secure, but – decreasing the number of high level processes to be connected – also increases the percentage of candidate systems that are *NDC*-secure.

We need a new class of Input-Total systems to characterize the set of test-systems. A system $F \in \mathcal{E}_H$ is *High Input-Total* if $\forall i \in Act_H \cap I$, $\forall E \in \mathcal{E}_H$ reachable from F , $\exists E' \in \mathcal{E}$ such that $E \stackrel{i}{\Longrightarrow} E'$. $\mathcal{E}_{HIT}$ denotes the set of all High Input-Total systems.

Definition 3.25 *Let $H \subseteq Act_H$ and $E \in \mathcal{E}$. Then E is NDC_H Input-Total (NDC_HIT for short) if $E \in \mathcal{E}_{IT}$ and $E/Act_H \approx_T ((E|\Pi) \setminus H)/Act_H$ holds for all $\Pi \in \mathcal{E}_{HIT}$,*
$NDCIT$ is defined as NDC_HIT with $H = Act_H$.

As done for NDC_H, it is important to find a *static* characterization of NDC_HIT.

Theorem 3.26 *(Static characterization of NDC_HIT)*
Let $H \subseteq Act_H$. Then, $E \in NDC_HIT$ iff $E/Act_H \approx_T (E \setminus_I H)/Act_H$ and $E \in \mathcal{E}_{IT}$.

PROOF. ($\Rightarrow$) Let $\Pi_0 \stackrel{\mathrm{def}}{=} \sum_{i \in I \cap Act_H} i.\Pi_0$. Of course, Π_0 is the high Input-Total process which accepts any high input. Then it is easy to prove that $T(((E|\Pi_0) \setminus H)/Act_H) \subseteq T((E \setminus_I H)/Act_H)$ In fact $\forall \gamma \in T((E|\Pi_0) \setminus H)$, $\lambda = low(\gamma)$, if $\gamma_E \in T(E)$ is the trace of E which have part in generating γ, then γ_E has no element in the set $I \cap H$ (inputs in H are blocked by restriction). So $\gamma_E \in T(E \setminus_I H)$ with $low(\gamma_E) = \lambda$. By hypothesis $T(E/Act_H) \subseteq T((E \setminus_I H)/Act_H)$ and, since $T((E \setminus_I H)/Act_H) \subseteq T(E/Act_H)$, then $(E \setminus_I H)/Act_H \approx_T E/Act_H$.

($\Leftarrow$) Note that $T((E \setminus_I H)/Act_H) \subseteq T(((E \setminus_I H \mid \Pi) \setminus H)/Act_H)$ for any high level Input-Total Π. Note also that $T(((E \setminus_I H \mid \Pi) \setminus H)/Act_H) \subseteq T(((E|\Pi) \setminus H)/Act_H)$. Then, by hypothesis, we have $T(E/Act_H) = T((E \setminus_I H)/Act_H) \subseteq T(((E|\Pi) \setminus H)/Act_H)$. The thesis follows by observing that $T(((E|\Pi) \setminus H)/Act_H) \subseteq T(E/Act_H)$. □

The next theorem states that if we restrict *NNI* to Input-Total systems, then this property is equal to *NDCIT*. Hence, the Input-Total version of *NDC* does no longer protects high level outputs. A system E is *Non-deterministic Non Interference Input-Total* (*NNIIT* for short) if $E \in NNI$ and $E \in \mathcal{E}_{IT}$.

Theorem 3.27 *$NDCIT = NNIIT$*

PROOF. Trivial from Theorem 3.26 and Proposition 3.2. □

Proposition 3.28

(*i*) *E lts-correctable $\Rightarrow E \in NNIIT$ and $E \in NNIIT \not\Rightarrow E$ lts-correctable.*

(*ii*) *$E \in NNIIT \Rightarrow E \in TNDI$*

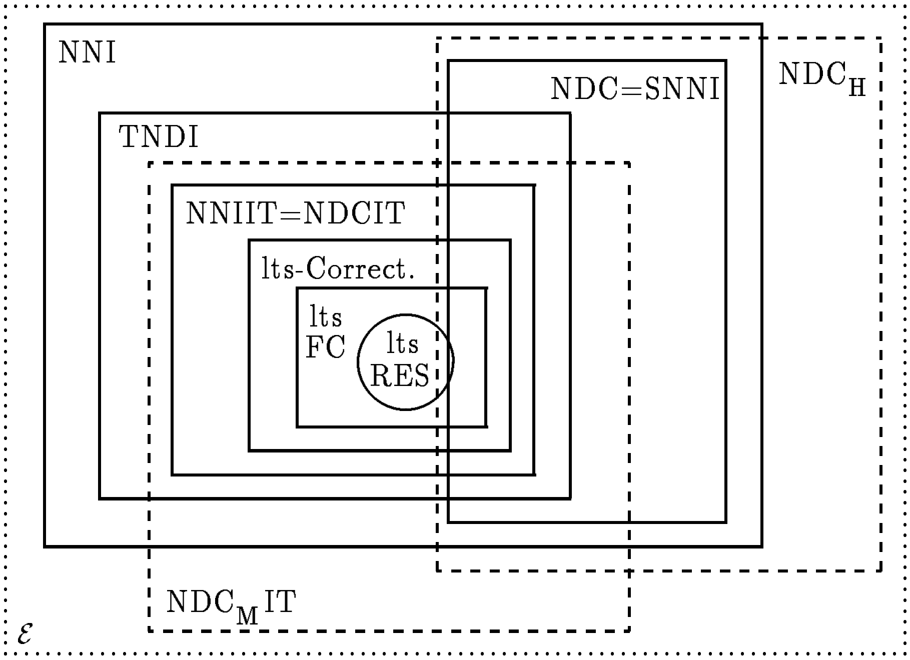

Figure 3: The inclusion diagram for trace-based security properties.

Proof. (*i*) For any $\gamma \in T(E)$ let us consider the perturbation obtained from γ deleting all the high level inputs $i \in I \cap Act_H$, then correct it in a trace $\gamma' \in T(E)$ such that $highinput(\gamma') = \langle\rangle$ and $low(\gamma') = low(\gamma)$. To prove that *NNIIT* does not implies *lts-correctable* consider the following agent: $Z = \sum_{i\in I} i.Z + \bar{h}.(\Pi_0 + \bar{l}.\Pi_0)$ with $\Pi_0 = \sum_{i\in I} i.\Pi_0$. Z is *NNIIT* but not *lts-correctable* in fact it is not possible to correct the perturbation $\bar{h}.h.\bar{l}$ of trace $\bar{h}.\bar{l}$.
(*ii*) Let $\lambda = low(\gamma)$ with $\gamma \in T(E)$ be a low level trace and let $\delta = highinput(\gamma')$ with $\gamma' \in T(E)$ be a high level input sequence. It has to be proved that there exists a trace consistent with these two sequences. By hypothesis there exists $\bar{\gamma} \in T(E)$ such that $low(\bar{\gamma}) = \lambda$ and $highinput(\bar{\gamma}) = \langle\rangle$. Define $\bar{\gamma}' = \bar{\gamma}.\delta$, which is a trace because $E \in \mathcal{E}_{IT}$. So we have found a trace $\bar{\gamma}' \in T(E)$ such that $low(\bar{\gamma}') = \lambda$ and $highinput(\bar{\gamma}') = \delta$. □

The next corollary is an immediate consequence of Propositions 3.9 and 3.28.

Corollary 3.29 *NNIIT* $=$ *TNDI* $\cap\, \mathcal{E}_{IT}$. □

Figure 3 reports the final diagram summarizing all the analyzed trace-based properties of information flow security. In the diagram, set H in NDC_H must be not empty and $H \cup \bar{H}$ must be different from Act_H; similarly, set M in $NDC_M IT$ is such that $M \cup \bar{M}$ is different from Act_H. Some further remarks permit to draw such diagram univocally.

	NNI	NDC SNNI	TNDI	lts-RES	lts-cor.	lts-FC	$\mathrm{NDC_H}$	NNIIT NDCIT	$\mathrm{NDC_M}$IT
NNI	$=$	**8**	2	2	2	2	9	10	10
NDC SNNI	$\subset$	$=$	**2**	2	2	2	$\subset$	**10**	10
TNDI	$\subset$	1	$=$	3	3	3	9	**10**	10
lts-RES	$\subset$	**1**	$\subset$	$=$	$\subset$	$\subset$	**9**	$\subset$	$\subset$
lts-cor.	$\subset$	1	$\subset$	6	$=$	6	9	$\subset$	$\subset$
lts-FC	$\subset$	1	$\subset$	**5**	$\subset$	$=$	9	$\subset$	$\subset$
$\mathrm{NDC_H}$	**7**	7	7	7	7	7	$=$	10	10
NNIIT NDCIT	$\subset$	1	$\subset$	3	**3**	3	9	$=$	$\subset$
$\mathrm{NDC_M}$IT	**4**	4	4	4	4	4	3	4	$=$

Let $T[x,y]$ be the table element contained in row x and column y. If $T[x,y] \in \{=, \subset\}$ then $xT[x,y]y$. If $T[x,y] = n$ then the agent n below is in x and is not in y. The emphasized numbers refer to agents mentioned in the proof of some theorem; other occurrences of the same number shows that the same counterexample is inherited for other cases. In the following Π_0 represents the Input-Total empty agent: $\Pi_0 = \sum_{i \in I} i.\Pi_0$.

1) $Z = \sum_{i \in I} i.Z + \bar{h}.Z'$ and $Z' = \sum_{i \in I} i.Z' + \bar{l}.\Pi_0$
2) $h.l.0 + l.0 + l'.0$
3) $Z = \sum_{i \in I} i.Z + \bar{h}.(\Pi_0 + \bar{l}.\Pi_0)$
4) $Z = \sum_{i \in I} i.Z + h.(\Pi_0 + \bar{l}.\Pi_0)$ with $h \notin M \cup \bar{M}$
5) $Z = \sum_{i \in I} i.Z + \bar{l}.\Pi_0 + h.\Pi_0$
6) $Z = \sum_{i \in I} i.Z + \bar{h}.Z' + \bar{h}'.Z''$ and $Z' = \Pi_0 + h.Z' + \bar{l}.\Pi_0$ and $Z'' = \sum_{i \in I \setminus \{h\}} i.Z'' + h.\Pi_0 + \bar{l}.\Pi_0$
7) $h.l.0$ with $h \notin H \cup \bar{H}$
8) $\bar{h}.l.0$
9) $Z = \sum_{i \in I} i.Z + \bar{h}.Z'$ and $Z' = \sum_{i \in I} i.Z' + \bar{l}.\Pi_0$ with $\bar{h} \in H \cup \bar{H}$
10) 0

Table 2: The inclusion table for trace-based security properties.

Proposition 3.30

(*i*) *If* $H, H' \subseteq Act_H$, $H' \subseteq H$, *then* $NDC_H IT \subseteq NDC_{H'} IT$

(*ii*) $SNNI \not\subseteq NDCIT$ *and* $TNDI \not\subseteq NDCIT$.

(*iii*) *If* $H \subseteq Act_H$ *and* $H \cup \bar{H} \neq Act_H$, *then* $NDC_H IT \not\subseteq NNI$.

Proof. (*i*) If $E \in NDC_H IT$ then $E/Act_H \approx_T (E \backslash_I H)/Act_H$. Since $T((E \backslash_I H)/Act_H) \subseteq T((E \backslash_I H')/Act_H)$, we have that $E/Act_H \approx_T (E \backslash_I H')/Act_H$ because by hypothesis $T(E/Act_H) \subseteq T((E \backslash_I)/Act_H)$ and, trivially, $T((E \backslash_I H')/Act_H) \subseteq T(E/Act_H)$.
(*ii*) Trivial, as there exists a *SNNI* and *TNDI* system which is not Input-Total (e.g. the empty process 0).
(*iii*) It is sufficient to consider the Input-Total agent $Z = \sum_{i \in I} i.Z + h.(\Pi_0 + \bar{l}.\Pi_0)$ with $\Pi_0 = \sum_{i \in I} i.\Pi_0$ where $h \in Act_H, l \in Act_L, h \notin H$. This system is $NDC_H IT$ but not *NNI*. □

Finally, Table 2 reports also all the needed counterexamples.

4. New Security Properties based on Bisimulation Semantics

All the security notions studied so far are based on the assumption that the semantics of a system is the set of its execution traces. In this section we show that some of these security properties – namely *NNI*, *SNNI* and *NDC* – can be naturally rephrased on the finer notion of system behaviour called *weak bisimulation* (or observational equivalence) [26].

This extension is badly needed as, unfortunately, trace semantics is rather weak, as it is unable to distinguish systems which give different observations to a user. For instance, systems $A = ab + a$ and $B = ab$ are trace equivalent but only the first can deadlock after action a. Hence, a low user could distinguish two low trace equivalent systems and so have information about high level users. As a simple example, let us consider the agent $C = \tau.l.0 + \tau.h.l.0$ which is *NDC* but can be deadlocked by a high level user who refuses to communicate with action $\bar{h}$.

Trace equivalence does not detect this kind of "attacks" and so it is necessary to substitute it with a stronger one. The coarsest equivalence which detects deadlocks is based on the *testing* idea [14] (or, equivalently, on the *failure* idea [15]). However, this equivalence can be too weak for other purposes. For instance, if one wants to extend SPA with the priority operator of the process algebra ACP [2], we are forced to consider the finer *ready-trace* semantics (see [2]) for a satisfactory treatment. Since there is not a clear limit to the features that one could want to include in the language, it is good practice to consider a very fine equivalence on action-labeled transition systems, which implies most of the other ones and all those mentioned above. A natural possibility is *observational equivalence* [26], based on weak bisimulation, which is the standard semantics for CCS. The nice consequence is that security properties based on other semantics coarser than observational equivalence (e.g., failure equivalence) would be coarser than those studied in this section. We will come back to this point in the concluding remarks.

Here we recall the definition of weak bisimulation [26] over SPA agents. In the following $E \overset{\hat{\mu}}{\Longrightarrow} E'$ stands for $E \overset{\mu}{\Longrightarrow} E'$ if $\mu \in \mathcal{L}$, and for $E\ (\overset{\tau}{\rightarrow})^*\ E'$ if $\mu = \tau$ (note that $(\overset{\tau}{\rightarrow})^*$ means "zero or more τ labelled transitions" while $\overset{\tau}{\Longrightarrow}$ requires at least one τ labelled transition).

Definition 4.1 *A relation $R \subseteq \mathcal{E} \times \mathcal{E}$ is a* weak bisimulation *if it satisfies:*

- *whenever $(E, F) \in R$ and $E \xrightarrow{\mu} E'$, then there exists $F' \in \mathcal{E}$ such that $F \overset{\hat{\mu}}{\Longrightarrow} F'$ and $(E', F') \in R$;*
- *conversely, whenever $(E, F) \in R$ and $F \xrightarrow{\mu} F'$, then there exists $E' \in \mathcal{E}$ such that $E \overset{\hat{\mu}}{\Longrightarrow} E'$ and $(E', F') \in R$.*

Two SPA agents $E, F \in \mathcal{E}$ are observational equivalent, notation $E \approx_B F$, if there exists a weak bisimulation containing the pair (E, F). Note that $\approx_B$ is an equivalence relation.

The next proposition, proved in [26], shows that observation equivalence is stronger than trace equivalence.

Proposition 4.2 $E \approx_B F \Longrightarrow E \approx_T F$

We can now introduce the bisimulation-based security properties *BNNI*, *BSNNI* and *BNDC*, by substituting $\approx_B$ for $\approx_T$ in their algebraic SPA-based characterizations.

Definition 4.3 *(Bisimulation NNI, SNNI, NDC)*

(i) $E \in BNNI \iff (E \setminus_I Act_H)/Act_H \approx_B E/Act_H$

(ii) $E \in BSNNI \iff E \setminus Act_H \approx_B E/Act_H$

(iii) $E \in BNDC \iff E/Act_H \approx_B (E \mid \Pi) \setminus Act_H$, *for all* $\Pi \in \mathcal{E}_H$.

As expected, these new properties are finer than their trace-based corresponding ones.

Proposition 4.4 *The following hold:*

(i) $BNNI \subset NNI$,

(ii) $BSNNI \subset SNNI$,

(iii) $BNDC \subset NDC$.

Proof. It immediately follows from Proposition 4.2. The inclusions are proper because $E = \tau.l.0 + \tau.h.l.0$ is such that $E \in NNI, SNNI$ and $E \notin BNNI, BSNNI, BNDC$. □

The next theorem shows that the bisimulation-based properties are related in a different way with respect to the corresponding ones.

Theorem 4.5

(*i*) *BNNI* $\not\subseteq$ *BSNNI and BSNNI* $\not\subseteq$ *BNNI.*

(*ii*) *BNDC* $\subseteq$ *BSNNI and BNDC* $\subseteq$ *BNNI.*

(*iii*) *BSNNI* $\not\subseteq$ *BNDC and BNNI* $\not\subseteq$ *BNDC.*

PROOF. (*i*) Let us consider the following agent $E = l.\overline{h}.l.h.l.0 + l.l.l.0 + l.l.0$; we have that $E \in BNNI$ and $E \notin BSNNI$. Let us now consider the agent $F = l.\overline{h}.l.h.l.0 + l.l.l.0 + l.0$; we have that $F \in BSNNI$ and $F \notin BNNI$.
(*ii*) To show that $BNDC \subseteq BSNNI$, consider $\Pi' = 0$ (the empty process). Then, by *BNDC* definition, we have that $E/Act_H \approx_B (E \mid 0) \setminus Act_H$ and so $E/Act_H \approx_B E \setminus Act_H$.
To show that $BNDC \subseteq BNNI$, consider $\Pi'' = \sum_{i \in Act_H \cap I} i.\Pi''$ (the process which accepts all the high inputs) then, by *BNDC* definition, we have that $E/Act_H \approx_B (E \mid \Pi'') \setminus Act_H$. Now it is sufficient to note that $(E \mid \Pi'') \setminus Act_H \approx_B (E \setminus_I Act_H)/Act_H$.
(*iii*) It trivially follows from (*i*) and (*ii*) above. □

A static characterization of *BNDC* – which does not involve composition with every processes Π – is not immediate. In Theorem 4.5 we have proved that *BSNNI* (the natural candidate after Corollary 3.22) is not the required static *BNDC*. As a matter of fact, this problem is still open. However, we propose below a new property which is static and compositional, but strictly stronger than *BNDC*.

Definition 4.6 *(SBNDC: Strong BNDC)*
A system $E \in SBNDC$ *if and only if* $\forall E'$ *reachable from* E *and* $\forall E''$ *such that* $E' \xrightarrow{h} E''$ *for* $h \in Act_H$, *then* $E' \setminus Act_H \approx_B E'' \setminus Act_H$

We need the following Lemma

Lemma 4.7 $E \in BNDC \Leftrightarrow E \setminus Act_H \approx_B (E|\Pi) \setminus Act_H$ *for all* $\Pi \in \mathcal{E}_H$.

PROOF. It follows immediately from Theorem 4.5.(*ii*) and Definition 4.3. □

In the following proofs, $E \Rightarrow E'$ denotes that E' is reachable from E.

Proposition 4.8 *SBNDC* $\subset$ *BNDC.*

PROOF. Let E be a system and Π a high level process. Let R be a relation defined as follows: $(E' \setminus Act_H, (E' \mid \Pi') \setminus Act_H) \in R$, for all E', Π' such that $E \Rightarrow E'$ and $\Pi \Rightarrow \Pi'$. We want to prove that if E is *SBNDC* then R is a bisimulation up to $\approx_B$ (see [26]). There is only one interesting case: $(E' \mid \Pi') \setminus Act_H \xrightarrow{\tau} (E'' \mid \Pi'') \setminus Act_H$. As there exists $h \in Act_H$ such that $E' \xrightarrow{h} E''$,

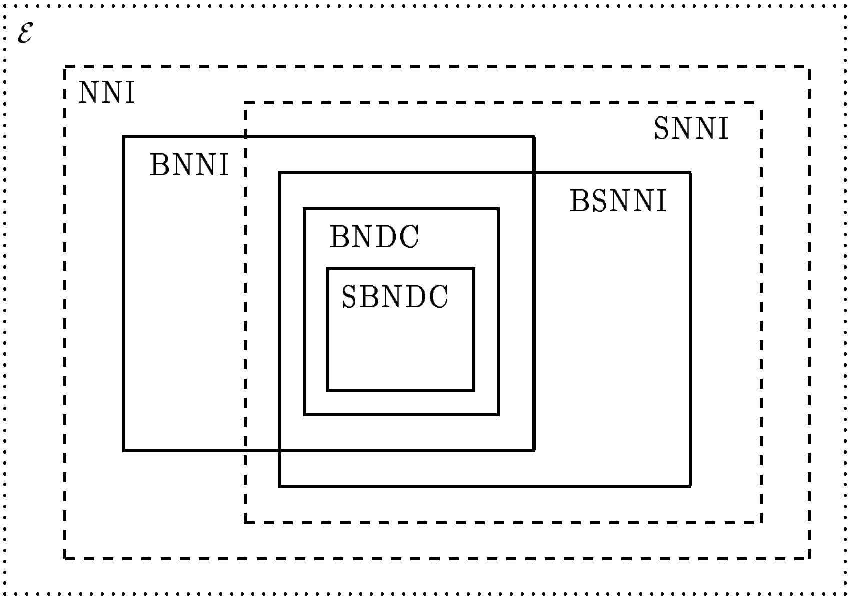

Figure 4: The inclusion diagram.

then $E'' \setminus Act_H \approx_B E' \setminus Act_H$; since $(E'' \setminus Act_H, (E'' \mid \Pi'') \setminus Act_H) \in R$, then $(E'' \mid \Pi'') \setminus Act_H \approx_B E' \setminus Act_H$. So $E \setminus Act_H \approx_B (E \mid \Pi) \setminus Act_H$ and the thesis follows by Lemma 4.7. The inclusion is strict because agent $E = l.h.l.0 + l.0 + l.l.0$ is *BNDC* but not *SBNDC*. □

Theorem 4.9 *The following hold:*

(*i*) $E, F \in SBNDC \Longrightarrow (E|F) \in SBNDC$,

(*ii*) $E \in SBNDC \Rightarrow E \setminus S \in SBNDC$, *if* $S \subseteq \mathcal{L}$.

(*iii*) $E, F \in SBNDC \Longrightarrow (E||F) \in SBNDC$

PROOF. (*i*) It must be proved that $\forall E', F' : E \Rightarrow E', F \Rightarrow F', \forall E'', F'' : (E' \mid F') \xrightarrow{h} (E'' \mid F'')$ with $h \in Act_H$, the following holds: $(E' \mid F') \setminus Act_h \approx_B (E'' \mid F'') \setminus Act_H$. Let R be the relation defined as follows: $((\tilde{E} \mid \tilde{F}) \setminus Act_H, (\tilde{E}' \mid \tilde{F}') \setminus Act_H) \in R, \forall \tilde{E}, \tilde{E}', \tilde{F}, \tilde{F}'$ such that $E \Rightarrow \tilde{E}, E \Rightarrow \tilde{E}', F \Rightarrow \tilde{F}, F \Rightarrow \tilde{F}'$ and $\tilde{E} \setminus Act_H \approx_B \tilde{E}' \setminus Act_H, \tilde{F} \setminus Act_h \approx_B \tilde{F}' \setminus Act_H$. It can be easily proved that R is a weak bisimulation (under the hypothesis $E, F \in SBNDC$); there is only one interesting case: $(\tilde{E} \mid \tilde{F}) \setminus Act_H \xrightarrow{\tau} (\tilde{E}'' \mid \tilde{F}'') \setminus Act_H$. Thus we have that $\tilde{E} \setminus Act_H \approx_B \tilde{E}'' \setminus Act_H$ and $\tilde{F} \setminus Act_H \approx_B \tilde{F}'' \setminus Act_H$ and so $((\tilde{E}'' \mid \tilde{F}'') \setminus Act_H, (\tilde{E}' \mid \tilde{F}') \setminus Act_H) \in R$.
(*ii*) Observe that $(E' \setminus S) \setminus Act_H \approx_B (E'' \setminus S) \setminus Act_H$ if and only if $(E' \setminus Act_H) \setminus S \approx_B (E'' \setminus Act_H) \setminus S$. As $\approx_B$ is a congruence for restriction [26] the thesis follows trivially.

	BNNI	BSNNI	BNDC	NNI	SNNI NDC	SBNDC
BNNI	$=$	**1**	1	$\subset$	**4**	5
BSNNI	**2**	$=$	2	$\subset$	$\subset$	5
BNDC	$\subset$	$\subset$	$=$	$\subset$	$\subset$	**5**
NNI	**3**	3	3	$=$	4	5
SNNI NDC	**3**	**3**	**3**	$\subset$	$=$	5
SBNDC	$\subset$	$\subset$	$\subset$	$\subset$	$\subset$	$=$

The legend is as for Table 2.

1) $E = l.\bar{h}.l.h.l.0 + l.l.l.0 + l.l.0$
2) $E = l.\bar{h}.l.h.l.0 + l.l.l.0 + l.0$
3) $E = \tau.l.0 + \tau.h.l.0$
4) $E = \bar{h}.l.0$
5) $E = l.h.l.0 + l.0 + l.l.0$

Table 3: The inclusion table.

(*iii*) Trivial from (*i*) and (*ii*). □

Figure 4 reports the diagram summarizing the relations among the main security properties we analyzed in both the trace-based and bisimulation-based approach. We need some further remarks:

Proposition 4.10 *The following hold:*

(*i*) *SNNI* $\not\subseteq$ *BNNI and*

(*ii*) *BNNI* $\not\subseteq$ *SNNI.*

Proof. (*i*) holds because $E = \tau.l.0 + \tau.h.l.0$ is such that $E \in \mathit{SNNI}$ and $E \notin \mathit{BNNI}$. To prove case (*ii*) it is sufficient to consider agent $E = \bar{h}.l.0$ which is *BNNI* but not *SNNI*. □

Finally, Table 3 reports also all the needed counterexamples.

5. Conclusions

In this paper we have studied a number of existing security properties by rephrasing them over the general model of lts. This uniform presentation allowed us to compare them formally, leading to a rather complete taxonomy. We have also investigated the effect over this classification of restricting our attention to Input-Total systems. Finally, we have introduced four new security properties based on observational equivalence.

In the following we would like to focus our attention on three specific aspects: First, a remark on the consequences of our results on ongoing research about automatic tools for checking the security of system specifications against some of the properties we have investigated here. Second, the influence that more accurate notions of equivalence can have on the definition of (more accurate) security properties. Third, a short report on related work about other taxonomies, other process algebraic studies and other properties we have ignored in this paper.

5.1. *Computer-Aided Verification*

We want to stress that our work is not only a mere exercise of translation of known properties in yet another model, namely lts and process algebras. As a matter of fact, process algebras possess a very strong mathematical theory which begins to show its usefulness in the definition of automatic tools for computer-aided verification; one significant example is the Concurrency Workbench [5]. As the final purpose is the identification of a security property which guarantees multilevel security and which can be verified effectively, we claim that this goal of automatic verification can be more easily achieved through our translation; indeed, we have implemented a tool for checking SPA specifications against various security properties. This tool, called *Security Checker* (SC for short),[8] accepts SPA specifications which generate finite state [9] lts and verify (most of) the security properties we have classified here. A detailed description of the tool is reported in [8], where also a significative example is completely verified automatically. We hope that SC can help in evaluating the merits of the various security properties, by testing them over several examples. Indeed, our work has only better revealed the relationships among the various notions of security which have been defined so far, but has not contributed to give evidence to which of these most tightly captures the intuitive, expected definition of security. Even if our work seems to suggest *BNDC* as the "right" security policy, we think that only an extensive experimentation on a large number of (finite state) case studies can certify our expectation.

---

[8] SC is based on the Concurrency Workbench [5] (version 6.1), and exploits some of the routines of CW such as the one for checking observational equivalence. SC is written in standard ML (New Jersey, version 0.93) and is available on request from the authors.

[9] It can also accept specifications which can generate an infinite state space, but the verification can only be partially performed on finite-state approximations.

We are presently designing another tool — to be integrated with SC — that exploits the compositionality of the investigated properties. This would be very beneficial for keeping under control the well-known state explosion problem.

5.2. *Security Properties and Behavioural Equivalences*

Our investigation showed that the notion of security can be deeply affected by the choice of the behavioural equivalence for systems. In fact, the security notions based on trace semantics are not deadlock sensitive, hence "insecure" by definition. However, there are other features of system behaviour that could be critical for security. For instance, *partial deadlock*, i.e., the deadlock of some subsystems (which anyway does not imply the deadlock of the system as a whole). Another example can be provided by *timing* information, because a real-time constraint unsatisfied by a late response can produce an information flow.

It is well-known that in order to include more accurate information in the semantic description, it is essential to move towards *non-interleaving* equivalences, such as *location* equivalence [4], *causal* equivalence [6] or *performance* equivalence [11]. We plan to study in the future the effect of rephrasing our *NDC* security property over these very fine behavioural equivalences.

5.3. *Related Work*

In the literature, there are two relevant papers that discuss classification of possibilistic information flow security properties. One is [3], where the authors studied a few security properties over a synchronous model in a trace-based approach. They discovered the relations among these properties when considering additionally a combination of the following conditions: Input-Totality, determinism and bipartition into two levels only. However, (*i*) in our work no synchrony assumption is taken, (*ii*) in their work no language is considered, hence (*iii*) compositionality issues are neglected and, finally, (*iv*) only trace-semantics based is investigated. Recently, another paper [24] approaches a similar task in a different perspective; a few security properties are uniformly presented in a trace-based model by means of a class of functions on traces, called *selective interleaving functions*. Moreover, a large part of the paper is devoted to a compositionality study of these properties w.r.t. several operations on sets of traces (e.g., product, union and intersection). Also in this case, (*i*) a synchrony assumption is taken, (*ii*) no language is considered and (*iii*) finer semantics are neglected.

Common to these two interesting papers is the discussion of one property, called *causality* in the former and *separability* in the latter,[10] which we have ignored. (On the contrary, we have studied several properties not considered in these two papers.) Among the other information flow security

---

[10] These two properties are close, as they both derives from the same intuitive idea of separability, reported in [30].

properties not studied here, we mention those in [27] (which is based on a classic automata model) and [19].

There are also two papers dealing with security properties in a process algebraic setting. One is [12] where Non Interference is investigated in CSP [15]; there, the author proposes a set of algebraic laws which, essentially, characterize the compositionality on *NI* w.r.t. the CSP combinators. The other one is [1], where Non Interference and Non Deducibility are compared.

A strong criticism against the *possibilistic* approach to information flow security we have investigated here is reported in [16], where the author claims that the properties in this class can only offer necessary conditions to security. His claim is based on the observation, reported in [32], that nondeterministic systems may exhibit probabilistic covert channels that are not ruled out by standard security models. Hence, Gray proposes a *probabilistic* version of Millen's synchronous state machine [25] on which two security properties are rephrased and compared. Even if in principle we agree with this criticism, we think that (*i*) the analysis of information flow security on nondeterministic systems is somehow complementary and (*ii*) it is very useful to check the security of a specification with respect to an automatically testable property, which could discover most of the design mistakes from the very beginning.

A reader interested in a nice introductory survey on the subject is invited to consult [23].

Acknowledgements

We would like to thank Simon Foley and Josè Meseguer for helpful comments and encouragement. We also thank an anonymous referee for fruitful suggestions.

References

[1] P. G. Allen. "A Comparison of Non-Interference and Non-Deducibility using CSP". In *Proceedings of the Fourth Workshop on the Foundations of Computer Security*, pages 43–54, Franconia, New Hampshire, June 1991.

[2] J. C. M. Baeten and W.P. Weijland. *Process Algebra.* Cambrige University Press, 1985. Cambrige Tracts in Theoretical Computer Science 18.

[3] P. Bieber and F. Cuppens. "A Logical View of Secure Dependencies". *Journal of Computer Security*, 1(1):99–129, 1992.

[4] G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn. "A Theory of Processes with Localities". In *Proceedings Concur 92*, pages 108–122. Springer-Verlag LNCS 630, 1992.

[5] R. Cleaveland, J. Parrow, and B. Steffen. "The Concurrency Workbench: a Semantics Based Tool for the Verification of Concurrent Systems". *ACM Transactions on Programming Languages and Systems*, Vol. 15 No. 1:36–72, January 1993.

[6] P. Darondeau and P. Degano. "Causal trees: interleaving + causality". In *Proceedings LITP Spring School on Semantics of Systems of Concurrent Processes*, pages 239–255. Lecture Notes in Computer Science, Vol. 469. Springer, Berlin, 1990.

[7] R. Focardi and R. Gorrieri. "An Information Flow Security Property for CCS ". In *Proceedings of the Second North American Process Algebra Workshop (NAPAW '93)*, TR 93-1369, Cornell (Ithaca), August 1993.

[8] R. Focardi, R. Gorrieri, and V. Panini. "The Security Checker: a Semantics Based Tool for the Verification of Security Properties". Forthcoming.

[9] S. N. Foley. "Understanding Information Flow in Systems". Unpublished manuscript, December 1989.

[10] J. A. Goguen and J. Meseguer. "Security Policy and Security Models". In *Proceedings of the 1982 Symposium on Security and Privacy*, pages 11–20. IEEE Computer Society Press, April 1982.

[11] R. Gorrieri, M. Roccetti, and E. Stancampiano. "A Theory of Processes with Durational Actions". *Theoretical Computer Science*, 140, April 1995.

[12] J. Graham-Cumming. "Laws of Non-Interference in CSP". *Journal of Computer Security*, 2:37–52, 1993.

[13] J. Guttman and M. Nadel. "What Needs Securing?". In *Proc. Computer Security Foundations Workshop*, pages 34–57, Franconia, New Hampshire, June 1988.

[14] M. Hennessy. *Algebraic Theory of Processes.* The MIT Press, 1988.

[15] C. A. R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1985.

[16] J. W. Gray III. "Toward a Mathematical Foundation for Information Flow Security". *Journal of Computer Security*, 1:255–294, 1992.

[17] D. M. Johnson and F. J. Thayer. "Security and the Composition of Machines". In *Proceedings of the Computer Security Foundations Workshop*, pages 72–89, June 1988.

[18] R. Keller. "Formal Verification of Parallel Programs". *Communications of the ACM*, 19 (7):561–572, 1976.

[19] L. G. Marcus. "Proving Varieties of Information Flow Security". Technical Report ATR-90(5778)-3, The Aerospace Corporation, El Segundo, California, 1990.

[20] D. McCullough. "Covert Channels and Degrees of Insecurity". In *Proceedings of the First Computer Security Foundations Workshop*, pages 1–33, June 1988.

[21] D. McCullough. "Noninterference and the Composability of Security Properties". In *Proceedings, 1988 IEEE Symposium on Security and Privacy*, pages 177–186. IEEE Computer Society Press, April 1988.

[22] D. McCullough. "A Hookup Theorem for Multilevel Security". *IEEE Transactions on Software Engineering*, pages 563–568, June 1990.

[23] J. McLean. "Security Models". in the *Encyclopedia of Software Engineering*, Wiley, in press.

[24] J. McLean. "A General Theory of Composition for Trace Sets Closed Under Selective Interleaving Functions". In *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy.* IEEE Computer Society Press, 1994.

[25] J. K. Millen. "Hookup Security for Synchronous Machines". In *Proceedings of the Third Computer Security Foundation Workshop III.* IEEE Computer Society Press, 1990.

[26] R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.

[27] I. S. Moskowitz and O. L. Costich. "A Classical Automata Approach to Noninterference Type Problems". In *Proceedings of the Fifth Workshop on the Foundations of Computer Security*, pages 2–8, Franconia, New Hampshire, June 1992.

[28] Department of Defense. "Integrity in Automated Information Systems". Technical Report C 79-91, National Computer Security Center, September 1991.

[29] C. O'Halloran. "A Calculus of Information Flow". In *Proceedings ESORICS*, 1990.

[30] J. Rushby. "Design and Verification of Secure Systems". In *Proceedings of the Eighth Symposium on Operating Systems Principles.* ACM, 1981.

[31] D. Sutherland. "A Model of Information". In *Proceedings of the 9th National Computer Security Conference*, pages 175–183. National Bureau of Standards and National Computer Security Center, September 1986.

[32] J. T. Wittbold and D. M. Johnson. "Information Flow in Nondeterministic Systems". In *Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy*, pages 144–161. IEEE Computer Society Press, 1990.