Modelling Downgrading in Information Flow Security*

Annalisa Bossi, Carla Piazza, Sabina Rossi
Dipartimento di Informatica - Università Ca' Foscari di Venezia
e-mail:{bossi,piazza,srossi}@dsi.unive.it

Abstract

Information flow security properties such as noninterference ensure the protection of confidential data by strongly limiting the flow of sensitive information. However, to deal with real applications, it is often necessary to admit mechanisms for downgrading or declassifying information.

In this paper we propose a general unwinding framework for formalizing different noninterference properties permitting downgrading, i.e., allowing information to flow from a higher to a lower security level through a downgrader. The framework is parametric with respect to the observation equivalence used to discriminate between different process behaviours. We prove general compositionality properties and provide conditions under which both horizontal and vertical refinements are preserved under all the security properties obtained as instances of the unwinding framework. Finally, we present a decision procedure to check our security properties and prove some complexity results.

1. Introduction

Since the seminal work by Goguen and Meseguer [7], noninterference plays a central role in the formalization of information flow security. It aims at characterizing the complete absence of any information flow or, indeed stronger, of any causal flow from high level entities to low level ones. As already noticed by many authors, see, e.g., [1, 12, 13, 17, 18, 19, 20, 21], this requirement is too strong. Absolute noninterference can hardly be achieved in real systems. In order to deal with real applications, it is often necessary to admit mechanisms for *downgrading* or *declassifying* information. For instance, even when two high level users communicate through an encrypted channel, a small amount of information can be leaked: indeed, a low level user may infer that a communication occurred, but this is not necessarily a sensitive information leak. As another example consider a simple device that allows information to flow from low to high but not viceversa. Such a device is feasible from a theoretical point of view only, in practice some flow from high to low is necessary to regulate the flow from low to high and avoid buffer overflows.

In order to permit systems to leak information by design, information flow controls often include some notion of downgrading whose invocation is limited to appropriately trusted subjects. The term downgrading is in fact used to refer to those situations in which trusted entities are permitted to move information from a higher to a lower security level. Thus the policy requirements may admit restricted/controlled information flows. For instance we have a downgrading when the classification of a previously sensitive file is turned to unclassified by a security officer.

The problem of detecting only uncontrolled information flows has first been considered by Goguen and Meseguer in [8]. They introduce the notion of *conditional noninterference* which admits flows from a high to a low level through a controlled channel. A more formal treatment of conditional noninterference is presented by Haigh and Young in [10]. Rushby in [19] develops a formal theory of downgrading for deterministic systems based on the notion of *intransitive noninterference*. Flows from the high level to a trusted part and flows from the trusted part to the low level are admissible since the trusted part takes care of controlling them, while a direct flow from high to low is not allowed. Pinsky in [17] unifies the concepts of standard and intransitive noninterference using *purge functions*. In [18] Roscoe and Goldsmith present a formalization of intransitive noninterference in the context of deterministic CSP.

The approaches mentioned above are limited to deterministic systems (except for [18] which can deal with a limited form of nondeterminism), and thus they are not applicable to distributed systems. In order to bridge this gap, in [13] Mantel proposes a generic security model for nondeterministic systems using *basic security predicates* which can cope with intransitive flow policies. In [1] Backes and Pfitzmann propose a notion of intransitive *probabilistic* noninterference for reactive systems. In [15] Mullins introduces a

* This work has been partially supported by the EU project IST-2001-32617 "Models and Types for Security in Mobile Distributed Systems" (MyThS) and the FIRB project RBAU018RCZ "Interpretazione astratta e model checking per la verifica di sistemi embedded".

property named *Admissible Interference* (*AI*) for processes expressed as terms of the CCS process algebra. This property is based on trace equivalence and can be understood as a generalization of the *Strong Deterministic NonInterference* (*SNNI*) property defined by Focardi and Gorrieri in [5].

All the properties discussed so far are based on traces and thus they do not allow to infer prohibited flows due to the possibility for a system component to block or unblock the system. Ryan and Schneider in [20] outline some generalizations of the notion of noninterference for CSP processes to handle *partial* and conditional information flows. Their approach is parametric with respect to an equivalence relation over processes. Lafrance and Mullins in [11] introduce the notion of *Bisimulation-based Non-deterministic Admissible Interference* (*BNAI*) which is a generalization of the *Bisimulation-based Non-Deducibility on Compositions* (*BNDC*) property presented by Focardi and Gorrieri in [5].

In this paper we propose a general unwinding framework for formalizing different noninterference properties permitting downgrading, i.e., allowing information to flow from a higher to a lower security level through a downgrader. The framework allows us to model both transitive and intransitive noninterference properties for distributed systems expressed as terms of the *Security Process Algebra* (SPA) language [5] extended with downgrading actions, and called here SPA^D. To give an intuition, a process E satisfies (an instance of) our unwinding framework if for each state F reachable from E, if F may perform a high level action reaching a state G then F may also perform a sequence of invisible actions reaching a state G' such that G and G' are indistinguishable for a low level user which is only able to observe low level actions.

The framework is parametric with respect to the observation equivalence used to distinguish between different process behaviours. In particular, we show how it can be instantiated by using trace equivalence and weak bisimilarity. Thus, differently from known proposals, we do not restrict ourselves to trace models. Indeed, as already noticed by Focardi and Gorrieri [5], there are applications in which trace equivalence is too weak while bisimilarity provides a more suitable notion of observation (see Section 3, Example 3.6).

Our general unwinding framework is obtained as a simple generalization of a previous unwinding schema for the definition of "strict" noninterference properties of SPA processes (a survey on our earlier work can be found in [2]). Analogously to the properties studied in [2], the security properties obtained as instances of the general framework presented here are all persistent, in the sense that if a process satisfies one of them then all its reachable states satisfy the same noninterference property. As discussed in Section 3 (see Examples 3.1, 3.2 and 3.4), persistence turns out to be fundamental in the treatment of downgrading. In fact, if we did not require persistence, we would not discover the uncontrolled information flows occurring after the first downgrading action.

We study how secure processes satisfying an instance of our general unwinding framework can be composed and refined while preserving the security property. In fact, in the stepwise development of complex systems it is important to consider security related issues from the very beginning in order to avoid the construction of poorly protected or, even worst, insecure processes. We first prove general compositionality results of our unwinding framework with respect to the SPA operators. Then we provide conditions under which refinement is preserved under all the security properties obtained as instances of the unwinding framework. In particular, we consider two forms of refinement, namely horizontal refinement and vertical refinement. Horizontal refinement is usually expressed in terms of preorder relations, such as trace inclusion, and aims at removing possible sources of nondeterminism, while vertical refinement consists in the replacement of abstract actions by processes which represent their implementation. Finally, we present a decision procedure to check our unwinding-based security properties, and prove some complexity results.

The paper is organized as follows. In Section 2 we recall the syntax and the semantics of the SPA language and report the definitions of some "strict" noninterference properties. In Section 3 we introduce the SPA^D language and present our general unwinding framework for the definition of noninterference properties permitting downgrading. We discuss different instances of our framework with both trace equivalence and weak bisimilarity. In Section 4 we prove various compositionality results of our class of properties with respect to the SPA operators. In Section 5 we provide conditions under which our properties are preserved under both horizontal and vertical refinement. In Section 6 we propose a decision procedure to check unwinding-based security properties permitting downgrading, and prove some complexity results. Finally, in Section 7 we draw some conclusions and discuss related work. All the proofs of the results presented in this paper are reported in the Appendix.

2. Preliminaries

We briefly recall the *Security Process Algebra* (SPA) language that we will use to model distributed systems. Moreover we report the definitions of some security properties whose aim is to completely avoid any flow of information from the high to the low level.

2.1. The SPA language

The SPA language [5] is a slight extension of Milner's CCS [14]. Analogously to CCS its syntax is based on: a set $\mathcal{L}$ of *visible* actions such that $\mathcal{L} = I \cup O$ where

$I = \{a, b, \ldots\}$ is a set of *input* actions and $O = \{\bar{a}, \bar{b}, \ldots\}$ is a set of *output* actions; a complement function $\bar{\cdot} : \mathcal{L} \to \mathcal{L}$, such that $\bar{\bar{a}} = a$, for all $a \in \mathcal{L}$; a special action τ which models internal computations, i.e., not visible outside the system. $Act = \mathcal{L} \cup \{\tau\}$ is the set of all *actions*. Function $\bar{\cdot}$ is extended to Act by defining $\bar{\tau} = \tau$. Differently from CCS, the set of visible actions is partitioned into high level actions and low level ones in order to specify multilevel systems. Thus we consider two sets, H and L, of high and low level actions which are closed with respect to $\bar{\cdot}$, i.e., $\overline{H} = H$ and $\overline{L} = L$; moreover they are disjoint and form a covering of $\mathcal{L}$, i.e., $H \cap L = \emptyset$ and $H \cup L = \mathcal{L}$.

The syntax of SPA *terms* (or *processes*) is as follows:

$$E ::= \mathbf{0} \mid a.E \mid E + E \mid E|E \mid E \setminus v \mid E[f] \mid Z$$

where $a \in Act$, $v \subseteq \mathcal{L}$, $f : Act \to Act$ with $f(\bar{\alpha}) = \overline{f(\alpha)}$, $f(\tau) = \tau$, $f(H) \subseteq H$, and $f(L) \subseteq L$. Moreover, Z is a constant that must be associated with a definition $Z \stackrel{\text{def}}{=} E$.

Let $\mathcal{E}$ be the set of SPA terms, ranged over by E and F. Let $\mathcal{L}(E)$ denote the *sort* of E, i.e., the set of the actions syntactically occurring in E. The set of high level processes is defined as $\mathcal{E}_H \stackrel{\text{def}}{=} \{E \in \mathcal{E} \mid \mathcal{L}(E) \subseteq H \cup \{\tau\}\}$.

The operational semantics of SPA processes is given in terms of *Labelled Transition Systems*. A *Labelled Transition System* (LTS) is a triple $(S, A, \to)$ where S is a set of states, A is a set of labels (actions), $\to \subseteq S \times A \times S$ is a set of labelled transitions. The notation $(S_1, a, S_2) \in \to$ (or equivalently $S_1 \stackrel{a}{\to} S_2$) means that the system can move from the state S_1 to the state S_2 through the action a. An LTS is *finite* if it has a finite number of states and transitions. The operational semantics of SPA is the LTS $(\mathcal{E}, Act, \to)$, where the states are the terms of the algebra and 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 Figure 1. The operational semantics of a process E is the subpart of the SPA LTS reachable from E. We write $E_1 \equiv E_2$ if the processes E_1 and E_2 have two isomorphic LTSs, i.e., they behave exactly in the same way.

The concept of *observation equivalence* between two processes is based on the idea that two systems have the same semantics if and only if they cannot be distinguished by an external observer. This is obtained by defining an equivalence relation over terms of the SPA LTS, equating two processes when they are indistinguishable. In the literature there are various equivalences of this kind. In this paper we consider the equivalence relations *weak bisimilarity*, $\approx_B$, and *trace equivalence*, $\approx_T$.

Let us first introduce the following auxiliary notations. If $t = a_1 \cdots a_n \in Act^*$ and $E \stackrel{a_1}{\to} \cdots \stackrel{a_n}{\to} E'$, then we write $E \stackrel{t}{\to} E'$ and we say that E' is *reachable* from E. We denote by $Reach(E)$ the set of all processes reachable from E. We also write $E \stackrel{t}{\Longrightarrow} E'$ if $E(\stackrel{\tau}{\to})^* \stackrel{a_1}{\to} (\stackrel{\tau}{\to})^* \cdots (\stackrel{\tau}{\to})^* \stackrel{a_n}{\to} (\stackrel{\tau}{\to})^* E'$ where $(\stackrel{\tau}{\to})^*$ denotes a (possibly empty) sequence of τ transitions. If $t \in Act^*$, then $\hat{t} \in \mathcal{L}^*$ is the sequence gained by deleting all occurrences of τ from t. As a consequence, $E \stackrel{\hat{a}}{\Longrightarrow} E'$ stands for $E \stackrel{a}{\Longrightarrow} E'$ if $a \in \mathcal{L}$, and for $E(\stackrel{\tau}{\to})^* E'$ if $a = \tau$ (note that $\stackrel{\tau}{\Longrightarrow}$ requires at least one τ transition while $\stackrel{\hat{\tau}}{\Longrightarrow}$ means zero or more τ transitions).

Weak Bisimilarity [14] equates two processes if they are able to mutually simulate their behavior step by step. Moreover, it does not care about internal τ actions.

Definition 2.1 (Weak Bisimulation) A symmetric binary relation $\mathcal{R}$ over processes is a *weak bisimulation* if $(E, F) \in \mathcal{R}$ implies, for all $a \in Act$, if $E \stackrel{a}{\to} E'$, then there exists F' such that $F \stackrel{\hat{a}}{\Longrightarrow} F'$ and $(E', F') \in \mathcal{R}$;

Two processes $E, F \in \mathcal{E}$ are *weakly bisimilar*, denoted by $E \approx_B F$, if there exists a weak bisimulation $\mathcal{R}$ containing the pair (E, F).

The relation $\approx_B$ (weak bisimilarity) is the largest weak bisimulation and it is an equivalence relation.

Trace equivalence equates two processes if they have the same sets of traces, again, without considering the τ actions.

Definition 2.2 (Trace Equivalence) The set of traces $Tr(E)$ associated with a process E is defined by: $Tr(E) = \{t \in \mathcal{L}^* \mid \exists E' : E \stackrel{t}{\Longrightarrow} E'\}$. Two processes E, F are *trace equivalent*, denoted by $E \approx_T F$, if $Tr(E) = Tr(F)$.

Trace equivalence is less demanding than weak bisimilarity, hence if two processes are weakly bisimilar, then they are also trace equivalent.

To define our security properties we need to consider low level observation equivalences, i.e., equivalences establishing which processes are indistinguishable from the low level point of view. This implicitly characterizes the power of possible attackers. Given an equivalence relation $\sim$ we can relativize it to the low level view in the following way.

Definition 2.3 (Equivalence on Low Actions) Let $\sim$ be an equivalence relation over processes. We say that two processes E and F are $\sim$*-equivalent on low actions*, denoted by $E \sim^l F$, if $E \setminus Comp(L) \sim F \setminus Comp(L)$ where $Comp(L)$ is the complementary set of L in $\mathcal{L}$, i.e., $Comp(L) = \mathcal{L} \setminus L$.

In particular, we will consider trace equivalence on low actions, $\approx_T^l$, and weak bisimilarity on low actions, $\approx_B^l$.

2.2. Total Non Interference

In this section we recall some security properties for SPA processes which aim at characterizing classes of processes having no information flows from high to low.

The *Non-Deducibility on Compositions* (*NDC*) and the *Bisimulation-based Non-Deducibility on Compositions* (*BNDC*) properties have been introduced by Fo-

$$\frac{-}{a.E \xrightarrow{a} E} \qquad \frac{E_1 \xrightarrow{a} E_1'}{E_1 + E_2 \xrightarrow{a} E_1'} \qquad \frac{E_2 \xrightarrow{a} E_2'}{E_1 + E_2 \xrightarrow{a} E_2'} \qquad \frac{E_1 \xrightarrow{a} E_1'}{E_1|E_2 \xrightarrow{a} E_1'|E_2} \qquad \frac{E_2 \xrightarrow{a} E_2'}{E_1|E_2 \xrightarrow{a} E_1|E_2'}$$

$$\frac{E_1 \xrightarrow{a} E_1' \;\; E_2 \xrightarrow{\bar{a}} E_2'}{E_1|E_2 \xrightarrow{\tau} E_1'|E_2'} a \in \mathcal{L} \qquad \frac{E \xrightarrow{a} E'}{E \setminus v \xrightarrow{a} E' \setminus v} \text{ if } a \notin v \qquad \frac{E \xrightarrow{a} E'}{E[f] \xrightarrow{f(a)} E'[f]} \qquad \frac{E \xrightarrow{a} E'}{Z \xrightarrow{a} E'} \text{ if } Z \stackrel{\text{def}}{=} E$$

Figure 1. The operational rules for SPA

cardi and Gorrieri [5] in order to capture every possible information flow from a *classified* (*high*) level of confidentiality to an *untrusted* (*low*) one. The definitions of *NDC* and *BNDC* are based on the basic idea of noninterference [7]: "No information flow is possible from high to low if what is done at the high level *cannot interfere* in any way with the low level". More precisely, a system E is *NDC* (*BNDC*) if what a low level user sees of the system is not modified by composing any high process Π to E. The two properties differ only on the low level observation equivalence they consider. *NDC* is based on trace equivalence on low actions, $\approx_T^l$, *BNDC* on weak bisimilarity on low actions, $\approx_B^l$.

We introduce the formal definitions of *NDC* and *BNDC* by exploiting a generalization of them, parametric with respect to the observation equivalence.

Definition 2.4 (NDC and BNDC) Let $\sim$ be an equivalence relation on processes and E be a SPA process. We say that $E \in NDC(\sim)$ if

$$\forall\, \Pi \in \mathcal{E}_H,\; E \sim^l (E|\Pi)$$

and use NDC for $NDC(\approx_T)$ and $BNDC$ for $NDC(\approx_B)$.

Notice that in this case, since $Comp(L) = H$, we have that $E \sim^l (E|\Pi)$ corresponds to $E\setminus H \sim (E|\Pi)\setminus H$. Moreover, since weak bisimilarity is stronger than trace equivalence, *BNDC* implies *NDC*.

Example 2.5 Let us consider an abstract specification M_x of a binary memory cell. M_x contains the binary value x and is accessible, by high and low users, through the four operations r_h, w_h, r_l, w_l representing a high read, a high write, a low read and a low write, respectively. Each operation is implemented through two different actions, one for each binary value. For example w_h_0 and w_h_1 indicate a high level user writing value 0 and 1, respectively. [1] The LTS of process M_x is depicted in Figure 2.

$$\begin{aligned} M\_x \;\stackrel{\text{def}}{=}\; & \overline{r_h\_x} \,.\, M\_x + w_h\_0 \,.\, M\_0 + w_h\_1 \,.\, M\_1 \\ + \;\; & \overline{r_l\_x} \,.\, M\_x + w_l\_0 \,.\, M\_0 + w_l\_1 \,.\, M\_1 \end{aligned}$$

1 The following expression for M_x is indeed a definition scheme: the actual processes M_0 and M_1 are obtained by replacing x with 0 and 1, respectively.

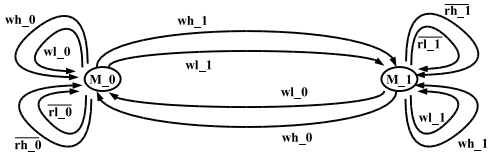

Figure 2. The LTS of the memory cell M_x.

Notice that M_0 and M_1 are totally insecure processes. As a matter of fact, a high level user may use the memory cell to directly send confidential information to the low level. Using both *NDC* and *BNDC* we detect that M_0 and M_1 are insecure. In fact, considering the high level process $\Pi \equiv \overline{w_h\_1}.\mathbf{0}$ we get that $(M_0|\Pi) \setminus H$ is neither weakly bisimilar nor trace equivalent to $M_0 \setminus H$, since in $(M_0|\Pi) \setminus H$ the low level user can read both 0 and 1, while in $M_0 \setminus H$ he can only read 0. □

In [5], Focardi and Gorrieri observe that properties *NDC* and *BNDC* are difficult to use in practice: *NDC* is not decidable in polynomial time, while the decidability of *BNDC* is still an open problem. In [6], Focardi and Rossi introduce the property *Persistent BNDC* (*P_BNDC*) which is a natural persistent extension of *BNDC* and it is a sufficient condition for *BNDC*. They show the decidability of *P_BNDC* over finite state processes by exploiting a bisimulation based characterization. The idea is that a system E is *P_BNDC* if for every high level process Π and for every state E' reachable from E a low level user cannot distinguish E' from $E'|\Pi$.

Other persistent security properties have been studied in the literature. We recall here the following: *Strong NDC* (*SNDC*) introduced in [3], *Strong BNDC* (*SBNDC*) introduced in [5], and *Compositional P_BNDC* (*CP_BNDC*) introduced in [3]. *SNDC* implies *NDC*, while the other properties imply *BNDC*.

All the *persistent* properties mentioned above can be defined as instances of a *generalized unwinding condition* introduced in [3]. The idea behind the notion of unwinding is to specify some constraints on the transitions of the system which imply some global properties. In particular, when an unwinding condition is used to define a noninterference

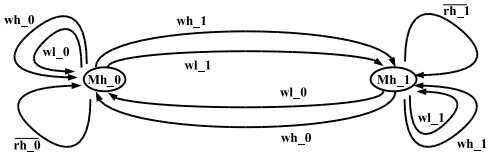

Figure 3. The LTS of the memory cell M^h_x.

property it usually requires that each high level action can be "simulated" in such a way that it is impossible for the low level user to infer which high level actions have been performed [18]. The generalized unwinding condition introduced in [3] is parametric with respect to two binary relations on processes: a low level equivalence relation, $\backsim^l$, which represents the low level view, and a transition relation, $\dashrightarrow$, which characterizes a local connectivity.

Definition 2.6 (Generalized Unwinding) Let $\sim$ be an equivalence relation and $\dashrightarrow$ be a binary relation on processes. The *unwinding class* $\mathcal{W}(\backsim^l, \dashrightarrow)$ is defined as

$$\mathcal{W}(\backsim^l, \dashrightarrow) \stackrel{\text{def}}{=} \{E \in \mathcal{E} \mid \forall\, F, G \in Reach(E) \text{ if } F \stackrel{h}{\rightarrow} G \text{ then } \exists G' \text{ such that } F \dashrightarrow G' \text{ and } G \backsim^l G'\}.$$

The notion of generalized unwinding on the SPA language entails a complete absence of information flow from H to L, since all the high level actions ($\stackrel{h}{\rightarrow}$) are required to be simulated ($\dashrightarrow$) in a way which is transparent to the low level users ($\backsim^l$).

We can say that the properties based on the generalized unwinding schema characterize *passive attacks*: the low level user observing the low level behavior of the system tries to infer the high level decisions. On the contrary, the *NDC*($\sim$) schema deals with *active attacks*: the process Π actively try to send down information to the low level user. As we will see, there is often a connection between properties characterizing passive attacks and properties involving active attackers.

The following theorem follows from the unwinding characterization of *P_BNDC* [2] and from the original definitions of *SBNDC*, *SNDC* and *CP_BNDC* [2, 3].

Theorem 2.7 (Unwinding) Let E be a SPA process.

- $E \in SNDC$ iff $E \in \mathcal{W}(\approx^l_T, \equiv)$;
- $E \in P\_BNDC$ iff $E \in \mathcal{W}(\approx^l_B, \stackrel{\hat{\tau}}{\Longrightarrow})$;
- $E \in SBNDC$ iff $E \in \mathcal{W}(\approx^l_B, \equiv)$;
- $E \in CP\_BNDC$ iff $E \in \mathcal{W}(\approx^l_B, \stackrel{\tau}{\Longrightarrow})$.

Example 2.8 The memory cell of Example 2.5 is neither *BNDC* nor *NDC*. In order to protect confidential data we can transform M_x into both a high level cell M^h_x (see Figure 3), by eliminating any low level read operation,

$$\begin{array}{rcl} M^h\_x & \stackrel{\text{def}}{=} & \overline{r_h\_x}.M^h\_x + w_h\_0.M^h\_0 + w_h\_1.M^h\_1 \\ & + & w_l\_0.M^h\_0 + w_l\_1.M^h\_1 \end{array}$$

and a low level cell M^l_x, by eliminating any high level write operation:

$$\begin{array}{rcl} M^l\_x & \stackrel{\text{def}}{=} & \overline{r_h\_x}.M^l\_x + \overline{r_l\_x}.M^l\_x \\ & + & w_l\_0.M^l\_0 + w_l\_1.M^l\_1 \end{array}$$

We can prove that both M^h_x and M^l_x are *P_BNDC*, *SNDC*, and *SBNDC*. They are not *CP_BNDC* since they cannot perform any τ transition. □

By exploiting our generalized unwinding framework we can also introduce the analogous of *P_BNDC* with trace equivalence as basic observation equivalence.

Definition 2.9 (P_NDC) Let E be a SPA process. We say that E satisfies the *Persistent Non-Deducibility on Composition* (*P_NDC*) property if $E \in \mathcal{W}(\approx^l_T, \stackrel{\hat{\tau}}{\Longrightarrow})$.

It is immediate to prove that $SNDC \subseteq P\_NDC$.

The following theorem establishes a connection between *NDC* and *BNDC*, which are based on the presence of an active attacker, and the properties introduced using the generalized unwinding which capture possible passive attacks.

Theorem 2.10 Let E be a SPA process. It holds:

- if $E \in P\_NDC$, then $E \in NDC$;
- if $E \in P\_BNDC$, then $E \in BNDC$.

As a consequence of the above theorem we also get that *SNDC* implies *NDC*, while *SBNDC* and *CP_BNDC* imply *BNDC*. This means that for our properties the existence of an active attack implies the existence of a passive one.

3. Downgrading via Unwinding

Many authors noticed that the notion of noninterference is too demanding when dealing with practical applications: indeed no real policy ever calls for total absence of information flow over any channel. In many practical applications confidential data can flow from high to low provided that the flow is not direct and it is controlled by the system, i.e., a trusted part of the system can control the downgrading of sensitive information. Consider for instance the case in which the high level user edits a file and sends it through a private channel to an encrypting protocol, the encrypting protocol encrypts the file and sends it through a public channel. Even if the high level data are sent through a public channel the encryption ensures that the low level users cannot read the data. Indeed, low level users can only observe that an encrypted file is passing on the public channel. In

this case the encrypting protocol represents the trusted part of the system which controls the flow from high to low.

In this section we show how our generalized unwinding can be instantiated in order to deal with processes admitting downgrading. This can be done just by extending the SPA language with a set of downgrading actions.

To model downgrading, we partition the set of visible actions $\mathcal{L}$ into the sets D (of downgrading actions), H, and L such that $\overline{D} = D$, $\overline{H} = H$ and $\overline{L} = L$. Moreover, we assume that for every relabelling function f, $f(\tau) = \tau$, $f(H) \subseteq H$, $f(L) \subseteq L$, and $f(D) \subseteq D$. We still denote by $\mathcal{E}_H$ the set of all high level processes. We denote by HD the set $H \cup D$ and use the notion SPA^D (SPA with Downgrading) to refer to this language.

Downgrading actions are used to model the behavior of a trusted component. It is reasonable to assume that an attacker cannot simulate the trusted part of the system, i.e., it cannot perform the actions in D. For instance, in the case of protocol analysis the attacker cannot distribute the encryption keys. Moreover, we can assume that the low level users cannot observe the actions performed by the trusted part.

If we consider *NDC*$(\sim)$ in SPA^D we get that a process E satisfies *NDC*$(\sim)$ if $E \setminus HD \sim (E|\Pi) \setminus HD$ for all $\Pi \in \mathcal{E}_H$, since now $comp(L) = H \cup D = HD$.

Example 3.1 Consider the case in which an encrypting protocol receives a confidential file on a private channel, encrypts it and sends the resulting file on a public channel. Let $file_h$ be the high level input representing the reception of the file on the private channel, enc_d be the downgrading action representing the encryption phase, $\overline{ok_h}$ be a confidential acknowledge to the high level user, and $\overline{file_l}$ be the low level output of the encrypted data. The encrypting protocol can be formalized as follows:

$$Enc = file_h.enc_d.\overline{ok_h}.\overline{file_l}.\mathbf{0}$$

If we consider any possible attacker $\Pi \in \mathcal{E}_H$ we get that

$$Enc \setminus HD \approx_B \mathbf{0} \approx_B (Enc|\Pi) \setminus HD$$

which means that Enc satisfies *BNDC* in SPA^D. □

Unfortunately imposing that a process E satisfies *BNDC*, more generally $NDC(\sim)$, over the language with downgrading is not enough to guarantee no information flow. In fact, all the (uncontrolled) flows which occur after the first downgrading are not revealed.

Example 3.2 In the process Enc above the high level action $file_h$ is downgraded through the action enc_d, while the high action $\overline{ok_h}$ is not downgraded. In particular, the action $\overline{ok_h}$ causes an uncontrolled information flow from high to low, since it can block or unblock the process. However, as illustrated in Example 3.1, this flow is not revealed by *BNDC*. This is due to the fact that it occurs after the downgrading action enc_d and that the *BNDC* property does not check all the reachable states. □

Let us analyze now the generalized unwinding in the SPA^D language. To avoid confusion we use the notation $\mathcal{W}^D(\sim^l, \dashrightarrow)$ to refer to the unwinding class defined by the relations $\sim$ and $\dashrightarrow$ in the language SPA^D. A process E of the SPA^D language is in the class $\mathcal{W}^D(\sim^l, \dashrightarrow)$ if

$$\forall F, G \in Reach(E), \text{ if } F \stackrel{h}{\rightarrow} G, \text{ then}$$
$$\exists G' \text{ such that } F \dashrightarrow G' \text{ and } G \sim^l G',$$

where $G \sim^l G'$ is equivalent to $G \setminus HD \sim G' \setminus HD$. On SPA^D our generalized unwinding does not entail a complete absence of information flow. Consider for instance the process $E = h.d.l.\mathbf{0}$. In E there is clearly a flow from H to D and from D to L. However, $E \in \mathcal{W}^D(\approx^l_B, \equiv)$, since $E \approx^l_B \mathbf{0} \approx^l_B d.l.\mathbf{0}$. In fact, the bisimilarity on low actions, which does not care about the actions in $H \cup D$, allows the flows from H to D. The fact that the unwinding imposes constraints only on the high level transitions ($\stackrel{h}{\rightarrow}$) implies that also the flows from D to L are allowed.

We can define the same security properties of Theorem 2.7 and Definition 2.9 also for SPA^D processes. To avoid confusion with the properties in pure SPA, we change their names by prefixing a D when we work on SPA^D. For instance, the DP_NDC property requires that if E reaches a process F which moves to G through a high level action, then F can also perform a sequence of silent actions ($\stackrel{\hat{\tau}}{\Longrightarrow}$ is the reachability relation in this case) to reach a process G' such that $G' \setminus HD$ and $G \setminus HD$ are trace equivalent ($\approx^l_T$ is the observation equivalence in this case). Hence both high and downgrading actions are not observed by the low level user. The other properties are obtained by using different reachability relations and observation equivalences.

Definition 3.3 Let E be a SPA^D process.

- $E \in DP\_NDC$ iff $E \in \mathcal{W}^D(\approx^l_T, \stackrel{\hat{\tau}}{\Longrightarrow})$;
- $E \in DSNDC$ iff $E \in \mathcal{W}^D(\approx^l_T, \equiv)$;
- $E \in DP\_BNDC$ iff $E \in \mathcal{W}^D(\approx^l_B, \stackrel{\hat{\tau}}{\Longrightarrow})$;
- $E \in DSBNDC$ iff $E \in \mathcal{W}^D(\approx^l_B, \equiv)$;
- $E \in DCP\_BNDC$ iff $E \in \mathcal{W}^D(\approx^l_B, \stackrel{\tau}{\Longrightarrow})$.

We can prove many relationships among the properties introduced above. For instance it is immediate to prove that $DSBNDC$ is included in both $DSNDC$ and DP_BNDC, while DCP_BNDC is included in DP_BNDC.

Example 3.4 Consider again the encrypting protocol of Example 3.1. It evolves into the process $E' = \overline{ok_h}.\overline{file_l}.\mathbf{0}$ and $E' \stackrel{\overline{ok_h}}{\rightarrow} E'' \equiv \overline{file_l}.\mathbf{0}$. However, there is no process reachable from E' through a sequence of τ actions and

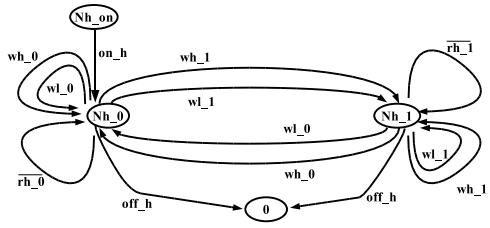

Figure 4. The LTS of the memory cell N^h_on.

weakly bisimilar or trace equivalent, on low actions, to E''. Indeed, Enc does not satisfies any of the properties of Definition 3.3. In fact, the low level user which observes the encrypted file passing on the public channel can infer that the high level user has received the acknowledge. We can avoid this kind of flow by adding a timeout to the protocol

$$Enc = file_h.enc_d.(\overline{ok_h}.\overline{file_l}.\mathbf{0} + \tau.\overline{file_l}.\mathbf{0}).$$

Now the process satisfies DP_NDC, DP_BNDC, and DCP_BNDC. □

Example 3.5 Consider the high memory cell of Example 2.8 and assume that the high level user has the possibility to 'turn the memory cell on and off', i.e., to start and stop the reading/writing operations. We also assume that the cell contains the value 0 when it is turned on.

Let on_h and off_h represent the high level actions which turn the cell on and off. The memory cell N^h_on (see Figure 4) is represented by the following system

$$\begin{array}{rcl} N^h\_\text{on} & \stackrel{\text{def}}{=} & \text{on}_h.N^h\_0 \\ N^h\_x & \stackrel{\text{def}}{=} & \overline{r_h\_x}.N^h\_x + w_h\_0.N^h\_0 + w_h\_1.N^h\_1 \\ & + & w_l\_0.N^h\_0 + w_l\_1.N^h\_1 + \text{off}_h.\mathbf{0} \end{array}$$

The cell N^h_on does not satisfy any of the properties defined in previous section: if we consider $\Pi \equiv \overline{\text{on}_h}.\mathbf{0}$ we have that $N^h\_\text{on} \setminus H \equiv \mathbf{0}$ is not trace equivalent to $(N^h\_\text{on}|\Pi) \setminus H$, i.e., it is not *NDC*. This represents the fact that the low level user which can write on the cell infers that the cell is off. Moreover, the states N^h_0 and N^h_1 satisfy only the *NDC* property. They are not *BNDC* because of the flow occurring when the cell is turned off (deadlock).

However, if we assume that the actions on_h and off_h are controlled by a trusted entity we can downgrade them getting the following specification

$$\begin{array}{rcl} P^h\_\text{on} & \stackrel{\text{def}}{=} & \text{on}_h.\text{on}_d.P^h\_0 \\ P^h\_x & \stackrel{\text{def}}{=} & \overline{r_h\_x}.P^h\_x + w_h\_0.P^h\_0 + w_h\_1.P^h\_1 \\ & + & w_l\_0.P^h\_0 + w_l\_1.P^h\_1 + \tau.\text{off}_h.\text{off}_d.\mathbf{0} \end{array}$$

where $\text{on}_d, \text{off}_d \in D$ are the downgrading actions. Now, the cell is DP_NDC, $DSNDC$, DP_BNDC, and $DSBNDC$.

One can notice that in the definition of P^h_x the high level action off_h has been replaced by the sequence of actions $\tau.\text{off}_h.\text{off}_d$, thus adding not only a downgrading action off_d but also a leading τ action. The intuition for that is the necessity of imposing a nondeterministic choice between a read/write operation, on the one hand, and the decision of turning off the computer, on the other hand. □

As far as the choice between trace equivalence and bisimilarity is concerned, as already pointed out by Focardi and Gorrieri [5], trace equivalence cannot discriminate critical cases as shown by the next example.

Example 3.6 Let us consider a procedure in which a low level user sends an application for a grant. The application can follow either a normal path in which the low level user has to pass two examination phases or a short path in which the first examination is sufficient. The short path can be taken only if a high level user sponsors the low level user. However, also with a high level sponsor, sometimes the normal path is taken, in order to randomly check that the sponsors are honest. After the examinations there is a high level decision phase whose final result (accepted/refused) is downgraded to the low level user. We can model the abstract specification of this procedure as follows

$$\begin{array}{rcl} Gr & \equiv & \text{ask}_l.(\text{first\_ex}_l.\text{second\_ex}_l.\overline{\text{dec}_h}.\overline{\text{dec}_d}.\overline{\text{read}_l}.\mathbf{0} + \\ & & \text{spons}_h.(\text{first\_ex}_l.\overline{\text{dec}_h}.\overline{\text{dec}_d}.\overline{\text{read}_l}.\mathbf{0} + \\ & & \text{first\_ex}_l.\text{second\_ex}_l.\overline{\text{dec}_h}.\overline{\text{dec}_d}.\overline{\text{read}_l}.\mathbf{0})) \end{array}$$

where ask_l is the low level application for the grant, first_ex_l and second_ex_l are the two examination phases, spons_h is the high level sponsoring action, $\overline{\text{dec}_h}$ is the high level decision phase, and $\overline{\text{dec}_d}$ is the downgrading to the low level user of the decision. Thus the low level user reads the decision through the low level action $\overline{\text{read}_l}$.

Gr satisfies $DSNDC$, and hence also DP_NDC. In fact, if E_1 is the process reached by Gr after executing ask_l, then E_1 performs a high level action reaching a state E_2 such that $E_1 \setminus HD \equiv \text{first\_ex}_l.\text{second\_ex}_l.\mathbf{0} \approx_T \text{first\_ex}_l.\mathbf{0} + \text{first\_ex}_l.\text{second\_ex}_l.\mathbf{0} \equiv E_2 \setminus HD$.

However, when a short path is taken the low level user can infer that a high level user has sponsored his application. Indeed Gr does not satisfy neither $DSBNDC$ nor DP_BNDC, since $E_1 \setminus HD \not\approx_B E_2 \setminus HD$. In this case the use of bisimilarity allows us to capture the flow which occurs when the short path is taken. Notice that, the downgrading action $\overline{\text{dec}_d}$ takes care of downgrading only the final decision, while the undesired flow regarding the spons_h action is correctly captured using bisimulation. □

The next theorem relates unwinding properties for SPA^D and SPA processes. It allows us to check whether a SPA^D process E satisfies an unwinding condition $\mathcal{W}^D(\sim^l, \dashrightarrow)$ by testing whether all the SPA processes of the form $E' \setminus D$, with E' reachable from E, belong to $\mathcal{W}(\sim^l, \dashrightarrow)$.

Theorem 3.7 ($\mathcal{W}^D$ and $\mathcal{W}$) Let $\dashrightarrow$ be a binary relation on SPA^D such that for each process F it holds $F \dashrightarrow F'$ iff $F \setminus D \dashrightarrow F' \setminus D$. Let E be a SPA^D process. $E \in \mathcal{W}^D(\sim^l, \dashrightarrow)$ iff for each $E' \in Reach(E)$, $E' \setminus D \in \mathcal{W}(\sim^l, \dashrightarrow)$.

The above theorem provides a relation between the properties of Definition 3.3 and those of Theorem 2.7. For instance, by applying it to the unwinding condition defining the DP_NDC property we get that a process E is DP_NDC if and only if all the processes of the form $E' \setminus D$, with E' reachable from E, are P_NDC. More in general the following corollary holds.

Corollary 3.8 Let E be a SPA^D process and $X \in \{P\_NDC, SNDC, P\_BNDC, SBNDC, CP\_BNDC\}$. $E \in DX$ iff $E' \setminus D \in X, \forall E' \in Reach(E)$.

We already noticed that our unwinding security properties are concerned with the observation of passive attacks. However, the following theorem tells us that the absence of passive attacks ensures the absence of active ones. Hence the existence of a passive attack is a necessary condition for the existence of an active one.

Theorem 3.9 Let E be a SPA^D process.

- If $E \in DP\_NDC$, then $\forall E' \in Reach(E)$ and $\forall \Pi \in \mathcal{E}_H$, $E' \setminus HD \approx_T (E' \setminus D|\Pi) \setminus H$, i.e., $E' \setminus D \in NDC$.
- $E \in DP\_BNDC$ iff $\forall E' \in Reach(E)$ and $\forall \Pi \in \mathcal{E}_H$, $E' \setminus HD \approx_B (E' \setminus D|\Pi) \setminus H$, i.e., $E' \setminus D \in BNDC$.

Since E belongs to $Reach(E)$, if a process E is DP_NDC (DP_BNDC), then $E \setminus D$ is NDC ($BNDC$). Hence a high level malicious process cannot use the non-downgraded high level actions to reveal information to the low level user. Moreover, for all the processes E' reachable from E it holds $E' \setminus D \in NDC$ ($BNDC$), which means that also after the execution of some downgrading actions any high level malicious process cannot send information down to the low level.

4. Compositionality

In this section we study the compositionalily properties of our generalized unwinding. Compositionality is useful for both verification and synthesis: if a property is preserved when systems are composed, then the analysis may be performed on subsystems and, in case of success, the system as a whole will satisfy the desired property by construction. We establish some general compositionality results for the security properties obtained as instances of our generalized unwinding on SPA^D and show how they apply to the properties of Definition 3.3.

Given a class $\mathcal{W}^D(\sim^l, \dashrightarrow)$ and a partial function f from k-tuples of processes to processes, we say that $\mathcal{W}^D(\sim^l, \dashrightarrow)$ is *compositional* with respect to f if $E_1, \ldots, E_k \in \mathcal{W}^D(\sim^l, \dashrightarrow)$ implies that either $f(E_1, \ldots, E_k) \in \mathcal{W}^D(\sim^l, \dashrightarrow)$ or $f(E_1, \ldots, E_k)$ is not defined (denoted by $f(E_1, \ldots, E_k)\uparrow$).

The following notion of *preservation* of a relation with respect to a function is at the basis of our results.

Definition 4.1 (Preservation) Let f be a partial function from k-tuples of processes to processes and $\odot$ be a relation on processes. The function *f preserves* $\odot$ if the following condition holds. Let $E_1, \ldots, E_k$ and $E'_1, \ldots, E'_k$ be processes, and $I \uplus J$ be any partition of $\{1, \ldots, k\}$ with $I \neq \emptyset$. If $\forall i \in I\ (E_i \odot E'_i)$ and $\forall j \in J\ (E_j \equiv E'_j)$ then

$$f(E_1, \ldots, E_k) \odot f(E'_1, \ldots, E'_k) \text{ or } (f(E_1, \ldots, E_k)\uparrow \text{ and } f(E'_1, \ldots, E'_k)\uparrow)$$

In [2] we proved the following compositionality results which can be applied also to SPA^D.

Theorem 4.2 Let $\mathcal{W}^D(\sim^l, \dashrightarrow)$ be an unwinding class.

- Let $a \in L \cup \{\tau\}$ and $\mathrm{pre_a}$ be a function from $\mathcal{W}^D(\sim^l, \dashrightarrow)$ to processes such as $\mathrm{pre_a}(E) = a.E$. Then $\mathcal{W}^D(\sim^l, \dashrightarrow)$ is compositional with respect to the a-prefix operator, $\mathrm{pre_a}$;
- Let $v \subseteq \mathcal{L}$ and $\mathrm{rest_v}$ be a function from $\mathcal{W}^D(\sim^l, \dashrightarrow)$ to processes such as $\mathrm{rest_v}(E) = E \setminus v$. If $\mathrm{rest_v}$ preserves $\dashrightarrow$ and $\sim^l$, then $\mathcal{W}^D(\sim^l, \dashrightarrow)$ is compositional with respect to the v-restriction operator, $\mathrm{rest_v}$;
- Let g be a renaming and $\mathrm{ren_g}$ be a function from $\mathcal{W}^D(\sim^l, \dashrightarrow)$ to processes such as $\mathrm{ren_g}(E) = E[g]$. If $\mathrm{ren_g}$ preserves $\dashrightarrow$ and $\sim^l$, then $\mathcal{W}^D(\sim^l, \dashrightarrow)$ is compositional with respect to the g-renaming operator, $\mathrm{ren_g}$;
- Let par be a function from $(\mathcal{W}^D(\sim^l, \dashrightarrow))^2$ to processes such as $\mathrm{par}(E, F) = E|F$. If par preserves $\dashrightarrow$ and $\sim^l$, then $\mathcal{W}^D(\sim^l, \dashrightarrow)$ is compositional with respect to the parallel composition operator $|$;

As a consequence we get the following corollary.

Corollary 4.3 Let E be a SPA^D process and $X \in \{P\_NDC, SNDC, P\_BNDC, SBNDC, CP\_BNDC\}$. If $E \in DX$, then

- $a.E \in DX$, for all $a \in L \cup \{\tau\}$;
- $E \setminus v \in DX$, for all $v \subseteq \mathcal{L}$;
- $E[g] \in DX$, for all relabelling function g.

Unfortunately, our properties are not compositional with respect to the parallel operator.

Example 4.4 Consider the processes $E \equiv h.d.l.\mathbf{0}$ and $F \equiv \bar{d}$ with $h \in H$, $l \in L$ and $d, \bar{d} \in D$. Both E and F are DP_BNDC, however the process $(E|F)$ is not. Indeed the synchronization between the downgrading actions d and $\bar{d}$ produces a direct casuality between the high level action h and the low level action l. □

However, we can prove the compositionality with respect to the parallel operator provided that processes do not synchronize on downgrading actions.

Theorem 4.5 Let E, F be SPA^D processes and $X \in \{P\_NDC, SNDC, P\_BNDC, SBNDC, CP\_BNDC\}$. If $E, F \in DX$ and they cannot synchronize on downgrading actions, then $(E|F) \in DX$.

Example 4.6 Let P^h_on be the memory cell of Example 3.5. Since actions $\overline{\text{on}_d}$ and $\overline{\text{off}_d}$ are never used, by Theorem 4.5 we get that the parallel composition $(P^h\text{\_on}|P^h\text{\_on}|\ldots|P^h\text{\_on})$ of an arbitrary number of cells is still DP_NDC, $DSNDC$, DP_BNDC, $DSBNDC$. □

Moreover, as illustrated below, properties $DSNDC$, DP_BNDC, and $DSBNDC$ are not compositional with respect to the nondeterministic choice operator.

Example 4.7 Consider the processes $E \equiv h.d.\mathbf{0}$ and $F \equiv l.\mathbf{0}$. Both E and F are $DSBNDC$ (and hence they are also $DSNDC$ and DP_BNDC) but $E + F$ is neither $DSNDC$ nor DP_BNDC nor $DSBNDC$. The problem lies in the fact that while the high level action in E is safely simulated by a sequence of zero τ in $E \setminus HD$, the same high level action in $E+F$ is not safely simulated by a sequence of zero τ in $(E+F) \setminus HD$ due to the presence of the additional component F. This problem would not arise if h were simulated by at least one τ action. □

Since DCP_BNDC requires that each high level action is simulated by at least one τ action, it is compositional with respect to the nondeterministic choice operator.

Theorem 4.8 Let E, F be SPA^D processes. If $E, F \in DCP\_BNDC$, then $E + F \in DCP\_BNDC$.

5. Refinement

In the development of a complex system it is common practice to first describe it succinctly as a simple abstract specification and then refine it stepwise to a more concrete implementation. In the context of process algebra, this refinement methodology amounts to defining a mechanism for replacing abstract processes with more concrete ones. In the literature two kinds of refinement are distinguished: *horizontal* refinement and *vertical* refinement. Horizontal refinement is usually expressed in terms of preorders, such as trace inclusion, and aims at transforming the system into a more nearly executable one by, for instance, removing possible sources of nondeterminism. Vertical refinement instead consists in the replacement of abstract actions by more concrete processes which represent their implementation (see [9] for a survey on action refinement and its relationships with horizontal refinement).

A security-aware stepwise development requires that the security properties of interest are either preserved or gained during the development process, until a concrete specification is obtained. Following this approach the security properties are guaranteed, and thus verified, by construction.

Below we consider the problem of preserving our security properties under both horizontal and vertical refinement.

5.1. Horizontal Refinement

In [3] a general notion of horizontal refinement based on simulation is introduced.

Definition 5.1 (Simulation) A binary relation $\mathcal{S}$ over processes is a *simulation* if $(E, F) \in \mathcal{S}$ implies that, for all $a \in Act$, if $E \xrightarrow{a} E'$, then there exists F' such that $F \xrightarrow{a} F'$ and $(E', F') \in \mathcal{S}$.
We say that the process E *is simulated by* the process F, denoted by $E \leq F$, if there exists a simulation $\mathcal{S}$ containing the pair (E, F).

Definition 5.2 (Horizontal Refinement) A binary relation $\mathcal{R}$ over processes is a *horizontal refinement* if

- $\mathcal{R}^{-1}$, the inverse[2] of $\mathcal{R}$, is a simulation and
- $\mathcal{R}$ is a partial function from processes to processes [3].

We say that E *is a horizontal refinement of* F, denoted by $E \preceq F$, if there exists a horizontal refinement $\mathcal{R}$ such that $\mathcal{R}(F) = E$.

Notice that horizontal refinement is more restrictive than trace inclusion.

In [3] we have studied how to preserve unwinding-based security properties under refinement. The notion of preservation (Definition 4.1) is still at the basis of this result.

Theorem 5.3 (Unwinding and Horizontal Refinement) Let $\mathcal{W}^D(\sim^l, \dashrightarrow)$ be an unwinding class and $\mathcal{R}$ be a refinement. If $\mathcal{R}$ preserves $\dashrightarrow$ and $\sim^l$, then $\mathcal{W}^D(\sim^l, \dashrightarrow)$ is compositional with respect to $\mathcal{R}$, i.e., if $E \in \mathcal{W}^D(\sim^l, \dashrightarrow)$ and $\mathcal{R}(E) \downarrow$, then $\mathcal{R}(E) \in \mathcal{W}^D(\sim^l, \dashrightarrow)$.

By applying the above result to our properties we get the following corollary.

Corollary 5.4 Let $\mathcal{R}$ be a refinement.

- if $\mathcal{R}$ preserves $\approx^l_T$ and $\stackrel{\hat{\tau}}{\Longrightarrow}$, then DP_NDC is compositional with respect to $\mathcal{R}$;
- if $\mathcal{R}$ preserves $\approx^l_T$, then $DSNDC$ is compositional with respect to $\mathcal{R}$;
- if $\mathcal{R}$ preserves $\approx^l_B$ and $\stackrel{\hat{\tau}}{\Longrightarrow}$, then DP_BNDC is compositional with respect to $\mathcal{R}$;

2 $(E_1, E_2) \in \mathcal{R}^{-1} \iff (E_2, E_1) \in \mathcal{R}$.

3 Actually one could consider only total functions by using the process $\mathbf{0}$ to complete the partial ones.

- if $\mathcal{R}$ preserves $\approx_B^l$, then $DSBNDC$ is compositional with respect to $\mathcal{R}$;
- if $\mathcal{R}$ preserves $\approx_B^l$ and $\stackrel{\tau}{\Longrightarrow}$, then DCP_BNDC is compositional with respect to $\mathcal{R}$.

Example 5.5 Let us consider the memory cell P^h_on described in Example 3.5. The cell satisfies DP_BNDC. Let $\mathcal{R}$ be the refinement defined as $\mathcal{R}(E) = E \setminus \{w_l\_0\}$, for each process E. We have that $\mathcal{R}$ preserves both $\approx_B^l$ and $\stackrel{\hat{\tau}}{\Longrightarrow}$, hence by the above corollary $\mathcal{R}(P^h\_\text{on})$ still satisfies DP_BNDC. In fact, if we compute $\mathcal{R}(P^h\_\text{on}) \equiv Q^h\_\text{on}$ we get the following definition

$$\begin{array}{rcl} Q^h\_\text{on} & \stackrel{\text{def}}{=} & \text{on}_h.\text{on}_d.Q^h\_0 \\ Q^h\_x & \stackrel{\text{def}}{=} & \overline{r_h\_x}.Q^h\_x + w_h\_0.Q^h\_0 + w_h\_1.Q^h\_1 \\ & + & w_l\_1.Q^h\_1 + \tau.\text{off}_h.\text{off}_d.\mathbf{0} \end{array}$$

In Q^h_on the low level user may only write value 1. Since he cannot read the content of the cell, he may only infer that the cell is on, that is a downgraded information. □

5.2. Vertical Refinement

In [4] we formalized a notion of vertical refinement based on syntactic replacement and *context* composition.

A context C is nothing but a SPA term in which some of the constants may not be associated to a definition. The constants of C which are not associated to a definition are called *variables*. Note that when the term C contains a constant definition $Z \stackrel{\text{def}}{=} E$, then the set of variables occurring in C includes also the variables occurring in E. We will use the notation $C[X_1, \ldots, X_n]$ to denote a context whose variables are $X_1, \ldots, X_n$. Moreover, if $T_1, \ldots, T_n$ are processes, $C[T_1, \ldots, T_n]$ denotes the process obtained by simultaneously replacing X_i with T_i, for $i = 1, \ldots, n$ in the term C and in all its associated definitions. As an example, consider the term $T \equiv a.Z + b.\mathbf{0}$ where Z is a constant defined by $Z \stackrel{\text{def}}{=} c.Z + a.W$ and W is a variable. Then T can be written as $T[W]$ and $T[\mathbf{0}]$ is equal to $a.Z[\mathbf{0}] + b.\mathbf{0}$ where $Z[\mathbf{0}]$ is a new constant defined by $Z[\mathbf{0}] \stackrel{\text{def}}{=} c.Z[\mathbf{0}] + a.\mathbf{0}$.

Given a process F and a variable Y, we denote by F^Y or $F^Y[Y]$ the context obtained by replacing each occurrence of $\mathbf{0}$ in F with the variable Y. When F is a constant, or just it calls for constant definitions in its expression, the involved constants have to be renamed and redefined according to the same principle. As an example, consider the process $F \equiv a.Z + b.\mathbf{0}$ where Z is defined by $Z \stackrel{\text{def}}{=} c.Z + b.\mathbf{0}$. Then $F^Y \equiv a.Z^Y + b.Y$ where Z^Y is defined by $Z^Y \stackrel{\text{def}}{=} c.Z^Y + b.Y$.

To introduce our notion of vertical refinement we also need to define which are the *refinable* actions of a process.

Definition 5.6 (Free and Bound actions) Let T be a SPA^D term. The set of *bound* actions of T, denoted by $bound(T)$, is inductively defined as follows:

$$\begin{array}{l} bound(\mathbf{0}) = \emptyset \\ bound(Z) = \emptyset \text{ where } Z \text{ is a variable} \\ bound(a.T) = bound(T) \\ bound(T_1 + T_2) = bound(T_1) \cup bound(T_2) \\ bound(T_1|T_2) = bound(T_1) \cup bound(T_2) \\ bound(T \setminus v) = bound(T) \cup v \\ bound(T[f]) = bound(T) \cup \{a, f(a) \mid f(a) \neq a\} \\ bound(recZ.T) = bound(T) \end{array}$$

An action occurring in T is said to be *free* if it is not bound. We denote by $free(T)$ the set of free actions of T.

We do not want to refine an action r which occurs bound in E; moreover, in order to avoid problems with the synchronization, we require that $\bar{r}$ does not occur in E. As far as the process F which is intended to refine r is concerned, we do not want that either r or $\bar{r}$ occur in F otherwise we would enter into an infinite loop of refinements. We also require that the free actions of F are not bound in E, to avoid they become bounded in the refined process and viceversa.

Definition 5.7 (Refinable actions) Let E, F be SPA^D processes with distinct constants definitions and $r \in \mathcal{L}$. The action r is said to be *refinable* in E with F if for all subterm E' of E: (i) $r \notin bound(E')$ and $\bar{r} \notin bound(E') \cup free(E')$; (ii) $r, \bar{r} \notin bound(F) \cup free(F)$; (iii) $(bound(E') \cap free(F)) \cup (bound(F) \cap free(E')) = \emptyset$.

We introduce a syntactic and non-atomic notion of action refinement by structural induction on the syntax of the process to be refined.

Definition 5.8 (Vertical Refinement) Let E, E_1, E_2, F be SPA^D processes and r be an action refinable in E with F. The *refinement of r in E with F* is the process $Ref(r, E, F)$ inductively defined on the structure of E as follows:

1. $Ref(r, \mathbf{0}, F) \equiv \mathbf{0}$
2. $Ref(r, Z, F) \equiv Z$
3. $Ref(r, r.E_1, F) \equiv \tau.F^Y[Ref(r, E_1, F)]$
4. $Ref(r, a.E_1, F) \equiv a.Ref(r, E_1, F)$, if $a \neq r$
5. $Ref(r, E_1[f], F) \equiv Ref(r, E_1, F)[f]$
6. $Ref(r, E_1 \setminus v, F) \equiv Ref(r, E_1, F) \setminus v$
7. $Ref(r, E_1 + E_2, F) \equiv Ref(r, E_1, F) + Ref(r, E_2, F)$
8. $Ref(r, E_1|E_2, F) \equiv Ref(r, E_1, F)|Ref(r, E_2, F)$

where all the constant definitions of the form $Z_i \stackrel{\text{def}}{=} E_i$ have to be replaced by $Z_i \stackrel{\text{def}}{=} Ref(r, E_i, F)$.

The above definition deserves some explanations, mainly on items 3, 5, 8. In item 3 we consider a process of the form $r.E_1$ and we replace the refinable action r by $\tau.F^Y$ instead of the more intuitive F^Y. This choice allows us to keep under control the non deterministic behaviour of the process. In item 5 we consider a process of the form $E_1[f]$

where f is a given renaming. In the definition of refinable actions we impose that the actions involved in the renaming are not refinable. This guarantees that refinement and renaming can commute, and the correctness of item 5. Finally, item 8 points out the fact that our refinement is not atomic. Hence for instance if $E \equiv r.\mathbf{0}|a.\mathbf{0}$ and $F \equiv \bar{a}.b.\mathbf{0}$ we get $Ref(r, E, F) \equiv \tau.\bar{a}.b.\mathbf{0}|a.\mathbf{0}$ in which a and $\bar{a}$ can synchronize. On the contrary by atomic refinement this synchronization would not be allowed.

We are interested in the definition of classes of processes satisfying an instance of $\mathcal{W}^D(\sim^l, \dashrightarrow)$ and preserving such a property under vertical refinement.

Given a sequence $s = s_1, s_2, \ldots, s_n$ of actions, we denote by $s.E$ the process $s_1.s_2.\ldots.s_n.E$.

Definition 5.9 ($\mathcal{C}(s)$) Let $\mathcal{W}^D(\sim^l, \dashrightarrow)$ be an unwinding condition compositional with respect to restriction and renaming. Let $s \in (L \cup \{\tau\})^*$ be a sequence of low and silent actions such that if $E \xrightarrow{s} E'$, then $E \dashrightarrow E'$. The class $\mathcal{C}(s)$ contains all processes defined by the following productions:

$$T ::= \mathbf{0} \mid \textstyle\sum_{i\in I} l_i.T_i + \sum_{j\in J}(h_j.T_j + s.(T_j \setminus HD)) \mid T \setminus v \mid T[f] \mid Z$$

where $l_i \in L \cup \{\tau\}$ for $i \in I$, $h_j \in H$ for $j \in J$, and Z is associated to a definition of the form $Z \stackrel{\text{def}}{=} T$.

Theorem 5.10 Let $E, F \in \mathcal{C}(s)$ be two SPA^D processes. Let r be an action not occurring in s.

$$Ref(r, E, F) \in \mathcal{W}^D(\sim^l, \dashrightarrow).$$

We can apply the above result to DP_NDC, DP_BNDC and DCP_BNDC using the sequence $s = \tau$.

More interestingly, we can identify some situations in which action refinement can be used to rectify a process which is not in the class $\mathcal{W}^D(\sim^l, \dashrightarrow)$ in order to get a process which is in $\mathcal{W}^D(\sim^l, \dashrightarrow)$. First we assume that $\mathcal{W}^D(\sim^l, \dashrightarrow)$ is such that for each process F it holds $F \dashrightarrow F$. Then we consider a process E which is not in $\mathcal{W}^D(\sim^l, \dashrightarrow)$ because there exists an action $k \in H$ which occurs in E and a process G' reachable from E such that $G' \xrightarrow{k} G''$ and for each G''' such that $G' \stackrel{\hat{\tau}}{\dashrightarrow} G'''$ we have $G'' \not\sim^l G'''$ (i.e., this is the only point in which the unwinding condition is violated and k does not occur elsewhere in E). In this case we say that *E is not in $\mathcal{W}^D(\sim^l, \dashrightarrow)$ because of k*. As an example consider the process $E \equiv \tau.(k.\mathbf{0} + l.\mathbf{0}) + h.l.\mathbf{0}$ which is not in DP_BNDC because of the action k. The violation of the unwinding condition in correspondence of the action k represents the fact that k causes an uncontrolled information flow. If we refine k using $k.d.\mathbf{0}$, with $d \in D$, we have that the flow is controlled by the downgrader. Unfortunately, this is generally not sufficient to get a process which belongs to $\mathcal{W}^D(\sim^l, \dashrightarrow)$. For instance the process $E \equiv \tau.(k.\mathbf{0} + l.\mathbf{0}) + h.l.\mathbf{0}$ would be transformed into $\tau.(\tau.k.d.\mathbf{0} + l.\mathbf{0}) + h.l.\mathbf{0}$ which now is not DP_BNDC because of the action h. Hence, it could be necessary to iterate the process of refinement over other high level actions. However, we can identify cases in which we can ensure that the downgrading of k gives us a process belonging to $\mathcal{W}^D(\sim^l, \dashrightarrow)$.

Definition 5.11 Let $\mathcal{W}^D(\sim^l, \dashrightarrow)$ be an unwinding class and E be a SPA^D process. $Low(E, \sim^l, \dashrightarrow)$ is the set

$$Low(E, \sim^l, \dashrightarrow) = \{E'', E''' \mid \exists E' \in Reach(E) \text{ such that } E' \xrightarrow{h} E'' \text{ and } E' \dashrightarrow E''' \text{ and } E'' \sim^l E'''\}.$$

Theorem 5.12 Let $\mathcal{W}^D(\sim^l, \dashrightarrow)$ be such that for each process F it holds $F \dashrightarrow F$ and $\dashrightarrow$ is preserved by the refinements of the form $Ref(r, F, r.d.\mathbf{0})$, i.e., if $F \dashrightarrow G$ then $Ref(r, F, r.d.\mathbf{0}) \dashrightarrow Ref(r, G, r.d.\mathbf{0})$. Let E be a SPA^D process, $k \in H$, $d \in D$. If E is not in $\mathcal{W}^D(\sim^l, \dashrightarrow)$ because of k and k does not occur in the processes of $Low(E, \sim^l, \dashrightarrow)$, then $Ref(k, E, k.d.\mathbf{0}) \in \mathcal{W}^D(\sim^l, \dashrightarrow)$.

Corollary 5.13 Let E be a SPA^D process, $k \in H$, $d \in D$.

- If E is not in DP_NDC because of k and k does not occur in the processes of $Low(E, \approx^l_T, \stackrel{\hat{\tau}}{\Longrightarrow})$, then $Ref(k, E, k.d.\mathbf{0})$ is in DP_NDC.
- If E is not in $DSNDC$ because of k and k does not occur in the processes of $Low(E, \approx^l_T, \equiv)$, then $Ref(k, E, k.d.\mathbf{0})$ is in $DSNDC$.
- If E is not in DP_BNDC because of k and k does not occur in the processes of $Low(E, \approx^l_B, \stackrel{\hat{\tau}}{\Longrightarrow})$, then $Ref(k, E, k.d.\mathbf{0})$ is in DP_BNDC.
- If E is not in $DSBNDC$ because of k and k does not occur in the processes of $Low(E, \approx^l_B, \equiv)$, then $Ref(k, E, k.d.\mathbf{0})$ is in $DSBNDC$.

Example 5.14 Let us consider again the high level memory cell $N^h_$on of Example 3.5. We already noticed that $N^h_$on is not DP_BNDC. In particular, its subprocess N^h_0 is not DP_BNDC because of off_h. If we compute $Ref(\text{off}_h, N^h\_0, \text{off}_h.\text{off}_d.\mathbf{0})$ we get the process S^h_0 which is defined by the following equations

$$\begin{aligned} S^h\_x \stackrel{\text{def}}{=} & \ \overline{r_h\_x}.S^h\_x + w_h\_0.S^h\_0 + w_h\_1.S^h\_1 \\ + & \ w_l\_0.S^h\_0 + w_l\_1.S^h\_1 + \tau.\text{off}_h.\text{off}_d.\mathbf{0} \end{aligned}$$

S^h_0 is DP_BNDC. Now the process $S^h_$on defined as $\text{on}_h.S^h\_0$ is not DP_BNDC because of on_h. By Corollary 5.13 we get that $Ref(\text{on}_h, S^h\_\text{on}, \text{on}_h.\text{on}_d.\mathbf{0})$ is DP_BNDC. Note that $Ref(\text{on}_h, S^h\_\text{on}, \text{on}_h.\text{on}_d.\mathbf{0})$ is weakly bisimilar to the process $P^h_$on of Example 3.5: they only differ on the initial τ added by the refinement of on_h. □

6. Decidability and Complexity

In this section we show how to decide whether a process E belongs or not to an unwinding class $\mathcal{W}^D(\sim^l, \dashrightarrow)$.

Let us assume that we have an algorithm $A_{\mathcal{W}}$ which decides whether a process F of the SPA language belongs to $\mathcal{W}(\sim^l, \dashrightarrow)$ or not. Moreover, let Time$(A_{\mathcal{W}})$ and Space$(A_{\mathcal{W}})$ be the time and space complexities of $A_{\mathcal{W}}$. The following theorem, which is an immediate consequence of Theorem 3.7, allows us to exploit the algorithm $A_{\mathcal{W}}$ also to decide whether a process E of the SPA^D language belongs to the class $\mathcal{W}^D(\sim^l, \dashrightarrow)$.

Theorem 6.1 Let $\mathcal{W}^D(\sim^l, \dashrightarrow)$ be an unwinding condition such that for each process F it holds $F \dashrightarrow F'$ iff $F \setminus D \dashrightarrow F' \setminus D$. Let E be a SPA^D process. Let $\tilde{E} = \Sigma_{E' \in Reach(E)} \tau.E' \setminus D$. It holds

$$E \in \mathcal{W}^D(\sim^l, \dashrightarrow) \quad \text{iff} \quad \tilde{E} \in \mathcal{W}(\sim^l, \dashrightarrow).$$

Given a process E to decide whether $E \in \mathcal{W}^D(\sim^l, \dashrightarrow)$ or not we need to:

- compute $\tilde{E}$;
- compute $A_{\mathcal{W}}$ on $\tilde{E}$.

As far as time and space complexities are concerned, we have that the LTS associated to $\tilde{E}$ can be build in linear time from the LTS associated to E without increasing its size. In fact, to get the LTS associated to $\tilde{E}$ it is sufficient to: (i) consider the LTS associated to E; (ii) add a new node named $\tilde{E}$; (iii) for each $E' \in Reach(E)$ add the edge $\tilde{E} \xrightarrow{\tau} E'$; (iv) delete each edge $E' \xrightarrow{d} E''$, with $d \in D$. Hence, if the LTS associated to E has n nodes and m edges, the LTS associated to $\tilde{E}$ has at most $n+1$ nodes and $m+n$ edges. Then, we can decide if $E \in \mathcal{W}^D(\sim^l, \dashrightarrow)$ or not in time Time$(A_{\mathcal{W}}) + O(n+m)$ and space Space$(A_{\mathcal{W}}) + O(n+m)$.

In particular, in the case of DP_BNDC and $DSBNDC$, we can exploit the polynomial algorithms for *P_BNDC* and *SBNDC* implemented in the CoPS tool [16], getting the following complexity results.

Corollary 6.2 Let E be a SPA^D process. It is possible to decide $E \in DP\_BNDC$ and $E \in DSBNDC$ in time $O(n^3)$ and space $O(n^2)$, where n is the number of states of the LTS associated to E.

7. Conclusion and Related Works

In this paper we present a general unwinding framework for formalizing different noninterference properties of SPA processes admitting downgrading, i.e., allowing information to flow from a higher to a lower security level through a downgrader. The framework is parametric with respect to the observation equivalence used to discriminate between different process behaviours. In particular we discuss instances of the unwinding framework with trace equivalence and weak bisimilarity.

Here we compare our approach with some related works.

Admissible Interference. In [15] Mullins introduced the security property named *Admissible Interference* (*AI*) as a trace based generalization of *NDC* [5] to deal with nondeterministic processes permitting downgrading. Like in our approach, his model is a variant of CCS and thus we can easy compare his definition with our unwinding-based ones.

The notion of AI is defined as follows. We denote by F/H (F *hiding* H) the process obtained by replacing all the high level actions in F with τ actions.

Definition 7.1 (AI) Let E be a SPA^D process. E satisfies *AI* if $\forall\, E' \in Reach(E),\ (E' \setminus D)/H \approx_T E' \setminus HD$.

We show that DP_NDC implies *AI*.

Theorem 7.2 Let E be a SPA^D process. If $E \in DP\_NDC$ then $E \in AI$.

However, the converse of Theorem 7.2 does not hold.

Example 7.3 Consider the process $E \equiv h.l_1.\mathbf{0} + l_1.\mathbf{0} + l_2.\mathbf{0}$. E is *AI* (and also *NDC*) since $E \setminus H \approx_T E/H$ (notice that there are no downgrading actions). However E is neither P_NDC nor DP_NDC since $E \xrightarrow{h} l_1.\mathbf{0}$ but there is no state E' reachable from E through a possibly empty sequence of τ actions and such that $E' \setminus H \approx_T l_1.\mathbf{0}$. □

In [15] Mullins shows that, in the deterministics case, *AI* implies the purging-based definitions of conditional noninterference for deterministic systems proposed by Haigh and Young in [10], by Rushby in [19], and by Pinsky in [17]. Thus also our DP_NDC property implies them.

In [11] Lafrance and Mullins propose a variant of the notion of Admissible Interference by using weak bisimilarity instead of trace equivalence. The new property is named *Bisimulation-based Non-deterministic Admissible Interference* (*BNAI*) and is defined as follows.

Definition 7.4 (BNAI) Let E be SPA^D process. E is *BNAI* if $\forall\, E' \in Reach(E),\ (E' \setminus D)/H \approx_B E' \setminus HD$.

We can prove that DP_BNDC is equivalent to *BNAI*.

Theorem 7.5 Let E be a SPA^D process. $E \in DP\_BNDC$ iff $E \in BNAI$.

Robust Declassification. In [21], Zdancewic and Myers introduce the notion of *robust system* which contains some intentional flows of confidential information obtained by declassification. This notion is parametric with respect to both an equivalence relation and a class of active attacks. First they define a parametric security property $\mathcal{SP}(\approx)$ where $\approx$ is an equivalence relation. This property is satisfied by a system if an observer with view $\approx$ cannot learn anything

by watching the system run. Since, in this case, information cannot be lost or destroyed, they say that the system is secure with respect to *passive attacks*. A system S is *robust* with respect to $\mathcal{SP}(\approx)$ and a class $\mathcal{B}$ of *active attacks* if for each $A \in \mathcal{B}$, the composition of S with A still satisfies the property $\mathcal{SP}(\approx)$. Zdancewic and Myers prove that when the class $\mathcal{B}$ of active attacks coincides with the systems satisfying $\mathcal{SP}(\approx)$, then a system S which is secure with respect to passive attacks is also secure with respect to the active attacks in $\mathcal{B}$, i.e., it is robust with respect to $\mathcal{B}$.

We can find similarities with our approach. In fact, Theorems 2.10 and 3.9 show that P_NDC (P_BNDC, DP_NDC, DP_BNDC) processes are robust with respect to $\approx_T$ ($\approx_B$, $\approx_T$, $\approx_B$) and the class of active attacks of the form $\Pi \in \mathcal{E}_H$ composed through the parallel operator.

Intransitive Basic Security Predicates. In [13] Mantel proposes a generic security model for information flow control of nondeterministic systems with two or more security levels. He shows how to define basic security predicates (BPSs) which can cope with intransitive flows. In particular he defines the security predicates *intransitive backwards strict deletion of confidential events* ($IBSD$) and *intransitive backwards strict insertion of confidential events* ($IBSIA$). These predicates are parametric with respect to a set of *security domains* $\mathcal{D}$ and an *extension set* X_D, where $D \in \mathcal{D}$, which possibly extends the view of D. As far as trace models are concerned, Mantel's framework is more general than our.

Partial Information Flow. In [20] the relationships between various definitions of noninterference and notions of process equivalence are analyzed and some generalizations to handle *partial* and conditional information flows are outlined. The authors provide a general definition of noninterference and discuss how such a generalization could be appropriate to deal with realistic practical situations, e.g., with policies that allow for automatic downgrading of certain statistical information from a database. Their definition is parametric with respect to an equivalence process relation and a set of constraints describing the high level behaviours for which it is intended to restrict the flow of information. The general definition of noninterference presented in [20] deals with active attacks only. Moreover, the authors neither provide an unwinding theorem nor discuss the verification problem for the properties that can be obtained as instances of their general definition.

References

[1] M. Backes and B. Pfitzmann. Intransitive Non-Interference for cryptographic purposes. In *Proc. of the IEEE Symposium on Security and Privacy (SSP'03)*, pages 140–152. IEEE Computer Society Press, 2003.

[2] A. Bossi, R. Focardi, D. Macedonio, C. Piazza, and S. Rossi. Unwinding in Information Flow Security. *Electronic Notes in Theoretical Computer Science*, 2004. To appear. Available at `http://www.dsiunive.it/~srossi/entcs04.ps`.

[3] A. Bossi, R. Focardi, C. Piazza, and S. Rossi. Refinement Operators and Information Flow Security. In *Proc. of the 1st IEEE Int. Conference on Software Engineering and Formal Methods (SEFM'03)*, pages 44–53. IEEE Computer Society Press, 2003.

[4] A. Bossi, D. Macedonio, C. Piazza, and S. Rosssi. Compositional Action Refinement and Information Flow Security. Technical Report CS-2003-13, Dipartimento di Informatica, Università Ca' Foscari di Venezia, Italy, 2003.

[5] R. Focardi and R. Gorrieri. Classification of Security Properties (Part I: Information Flow). In R. Focardi and R. Gorrieri, editors, *Proc. of Foundations of Security Analysis and Design (FOSAD'01)*, volume 2171 of *LNCS*, pages 331–396. Springer-Verlag, 2001.

[6] R. Focardi and S. Rossi. Information Flow Security in Dynamic Contexts. In *Proc. of the IEEE Computer Security Foundations Workshop (CSFW'02)*, pages 307–319. IEEE Computer Society Press, 2002.

[7] J. A. Goguen and J. Meseguer. Security Policies and Security Models. In *Proc. of the IEEE Symposium on Security and Privacy (SSP'82)*, pages 11–20. IEEE Computer Society Press, 1982.

[8] J. A. Goguen and J. Meseguer. Unwinding and Inference Control. In *Proc. of the IEEE Symposium on Security and Privacy (SSP'84)*, pages 75–86. IEEE Computer Society Press, 1984.

[9] R. Gorrieri and A. Rensink. Action Refinement. Technical Report UBLCS-99-09, University of Bologna (Italy), 1999.

[10] J. T. Haigh and W. D. Young. Extending the noninterference version of mls for sat. *IEEE Transactions on Software Engineering*, 13(2):141–150, 1987.

[11] S. Lafrance and J. Mullins. Bisimulation-based Nondeterministic Admissible Interference and its Application to the Analysis of Cryptographic Protocols. *Electronic Notes in Theoretical Computer Science*, 61:1–24, 2002.

[12] G. Lowe. Quantifying Information Flow. In *Proc. of the IEEE Computer Security Foundations Workshop (CSFW'02)*, pages 18–31. IEEE Computer Society Press, 2002.

[13] H. Mantel. Information Flow Control and Applications - Bridging a Gap. In *Proc. of the nternational Symposium of Formal Methods Europe (FME'01)*, LNCS, pages 153–172. Springer-Verlag, 2001.

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

[15] J. Mullins. Nondeterministic Admissible Interference. *Journal of Universal Computer Science*, 11:1054–1070, 2000.

[16] C. Piazza, E. Pivato, and S. Rossi. Cops - Checker of Persistent Security. In *Proc. of International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04)*, LNCS. Springer-Verlag, 2004. To appear.

[17] S. Pinsky. Absorbing Covers and Intransitive Noninterference. In *Proc. of the IEEE Symposium on Security and Privacy (SSP'95)*, pages 102–113. IEEE Computer Society Press, 1995.

[18] A. W. Roscoe and M. H. Goldsmith. What is intransitive noninterference? In *Proc. of the IEEE Computer Security Foundations Workshop (CSFW'99)*, pages 228–238. IEEE Computer Society Press, 1999.

[19] J. Rushby. Noninterference, transitivity, and channel-control security policies. Technical Report CSL-92-02, SRI International, December 1992.

[20] P. Ryan and S. Schneider. Process Algebra and Non-Interference. *Journal of Computer Security*, 9(1/2):75–103, 2001.

[21] S. Zdancewic and A. C. Myers. Robust Declassification. In *Proc. of the IEEE Computer Security Foundations Workshop (CSFW'01)*, pages 15–23. IEEE Computer Society Press, 2001.

A. Proofs

Proof of Theorem 2.10. The second item has been proved in [6]. We prove the first item.

Let E be a process such that for each $F \in Reach(E)$ if $F \xrightarrow{h} G$, then $F \stackrel{\hat{\tau}}{\Longrightarrow} G'$ with $G \setminus H \approx_T G' \setminus H$. We have to prove that for each $\Pi \in \mathcal{E}_H$ it holds $Tr((E|\Pi) \setminus H) = Tr(E \setminus H)$. The inclusion $Tr((E|\Pi) \setminus H) \supseteq Tr(E \setminus H)$ trivially holds, hence we only have to prove $Tr((E|\Pi) \setminus H) \subseteq Tr(E \setminus H)$.

Let $\gamma \in Tr((E|\Pi) \setminus H)$. There exists a trace γ' including τ actions such that γ is obtained from γ' by removing the τ actions. We proceed by induction on the number of τ's in γ'. If γ' has no τ actions, then $\gamma = \gamma'$ is a trace of $E \setminus H$, since E and Π never synchronize. Let γ' be of the form $(E|\Pi)\setminus H \xrightarrow{\gamma_1} (E'|\Pi')\setminus H \xrightarrow{\tau} (E''|\Pi'')\setminus H \stackrel{\gamma_2}{(} E'''|\Pi''')\setminus H$ in which we point out the last occurrence of τ. If τ is not a synchronization between E and Π we immediately get the thesis by inductive hypothesis on γ_1. If τ is a synchronization between E and Π, then we have that $E' \xrightarrow{h} E''$, hence $E' \stackrel{\hat{\tau}}{\Longrightarrow} F$ with $E'' \setminus H \approx_T F \setminus H$. Hence, by inductive hypothesis $\gamma_2 \in Tr(F \setminus H)$ and $\gamma_1\gamma_2 \in Tr(E \setminus H)$, which is equivalent to $\gamma \in Tr(E \setminus H)$. □

Proof of Theorem 3.7. $\Rightarrow$) Let $E \in \mathcal{W}^D(\sim^l, \dashrightarrow)$. We have to prove that for each $E' \in Reach(E)$, for each $E'' \in Reach(E' \setminus D)$ if $E'' \xrightarrow{h} G$, then $E'' \dashrightarrow G'$ with $G\setminus H \sim G'\setminus H$. From the fact that $E'' \in Reach(E'\setminus D)$ we have that $E'' \equiv E''' \setminus D$ with $E''' \in Reach(E')$ and $G \equiv G''' \setminus D$ with $E''' \xrightarrow{h} G'''$. Hence, since $E' \in Reach(E)$, we get $E''' \in Reach(E)$. From the hypothesis that $E \in \mathcal{W}^D(\sim^l, \dashrightarrow)$, since $E''' \xrightarrow{h} G'''$, we have that $E''' \dashrightarrow K$ with $G'''\setminus HD \sim K \setminus HD$. Hence $E'' \equiv E'''\setminus D \dashrightarrow K\setminus D$ with $G \setminus H \equiv G''' \setminus HD \sim K \setminus HD \equiv (K \setminus D) \setminus H$, i.e., the thesis.

$\Leftarrow$) Let E be such that for each $E' \in Reach(E)$ it holds $E' \setminus D \in \mathcal{W}(\sim^l, \dashrightarrow)$. We have to prove that for each $E' \in Reach(E)$ if $E' \xrightarrow{h} G$, then $E' \dashrightarrow G'$ with $G \setminus HD \sim G' \setminus HD$. If $E' \xrightarrow{h} G$, then $E' \setminus D \xrightarrow{h} G \setminus D$, hence by hypothesis, $E' \setminus D \dashrightarrow G' \setminus D$ with $(G \setminus D) \setminus H \sim (G' \setminus D) \setminus H$. Hence, $E' \dashrightarrow G'$ with $G \setminus HD \sim G' \setminus HD$, i.e., the thesis. □

Proof of Theorem 3.9. As far as the first item is concerned, if $E \in DP\_NDC$ by Corollary 3.8 we have that for each $E' \in Reach(E)$ it holds $E' \setminus D \in P\_NDC$, hence by Theorem 2.10 $E' \setminus D \in NDC$.

As far as the second item is concerned, if $E \in DP\_NDC$ by Corollary 3.8 we have that for each $E' \in Reach(E)$ it holds $E' \setminus D \in P\_NDC$, hence by Theorem 2.10 $E' \setminus D \in NDC$. On the other hand, if $\forall E' \in Reach(E)$ it holds that $E' \setminus D \in BNDC$, then $\forall E'' \in Reach(E')$ it holds $E'' \setminus D \in BNDC$, which implies by using a characterization of P_BNDC given in [6], that $E' \setminus D \in P\_BNDC$, hence, by Corollary 3.8, $E \in DP\_BNDC$. □

Proof of Theorem 4.5. Let us consider the case of $\mathcal{D} = DSNDC$. Applying Corollary 3.8 we have to prove that for each $E'|F' \in Reach(E|F)$ it holds that $(E'|F') \setminus D \in SNDC$. Since E and F cannot synchronize on downgrading actions $(E'|F')\setminus D$ is trace equivalent (and also weakly bisimilar) to $E' \setminus D|F' \setminus D$. Moreover, $E' \in Reach(E)$ and $F' \in Reach(F)$. From the hypothesis that $E, F \in DSNDC$ applying Corollary 3.8 we get that for each $E' \in Reach(E)$ and for each $F' \in Reach(F)$, $E'\setminus D$ and $F'\setminus D$ are in $SNDC$, hence, since $SNDC$ is compositional with respect to the parallel composition, $E'\setminus D|F'\setminus D$ is in $SNDC$.

The other cases are similar. □

Proof of Theorem 5.10. Notice that $\mathcal{C}(s) \subseteq \mathcal{W}^D(\sim^l, \dashrightarrow)$, hence if we prove $Ref(r, E, F) \in \mathcal{C}(s)$ we get the thesis.

In our proof we exploit the following claim, which can be proved by structural induction.

Claim 1. If $P, F \in \mathcal{C}(s)$, then $F^Y[P] \in \mathcal{C}(s)$.

We now prove that $Ref(r, E, F) \in \mathcal{C}(s)$ proceeding by structural induction on E and exploiting the cases of Definition 5.8.

The cases 1. and 2. are trivial.

Case 3. follows by inductive hypothesis and Claim 1.

Case 4. can occur only with $a \in L \cup \{\tau\}$, hence it follows by inductive hypothesis.

Cases 5. and 6. follow by inductive hypothesis.

Case 7. can occur only with sums of the form $l.E_1 + h.E_2 + s.(E_2 \setminus HD)$. We can distinguish three subcases

- $r = l$. We get the thesis by inductive hypothesis;

- $r = h$. We get that $Ref(r, E, F) = l.Ref(r, E_1, F) + \tau.F^Y[Ref(r, E_2, F)] + s.(Ref(r, E_2, F) \setminus HD)$ which is in $\mathcal{C}(s)$ by Claim 1 and inductive hypothesis;
- $r \neq h$. We get that $Ref(r, E, F) = l.Ref(r, E_1, F) + h.Ref(r, E_2, F) + s.(Ref(r, E_2, F) \setminus HD)$ which is in $\mathcal{C}(s)$ by inductive hypothesis.

Case 8. never occurs because $E \in \mathcal{C}(s)$ and the parallel operator is not in the syntax of $\mathcal{C}(s)$. □

Proof of Theorem 5.12. We have that for each $h \neq k$ if $E' \in Reach(E)$ is such that $E' \xrightarrow{h} G$ and $E' \dashrightarrow G'$ with $G \sim^l G'$, then $Ref(k, E', k.d.\mathbf{0}) \in Reach(Ref(k, E, k.d.\mathbf{0})$ and $Ref(k, E', k.d.\mathbf{0}) \xrightarrow{h} Ref(k, G, k.d.\mathbf{0})$. Since k does not occur in $Low(E, \sim^l, \dashrightarrow)$ we have that $Ref(k, G, k.d.\mathbf{0}) \equiv G$ and $Ref(k, G', k.d.\mathbf{0}) \equiv G'$ Moreover, since this refinement preserves $\dashrightarrow$ we get that $Ref(k, E', k.d.\mathbf{0}) \dashrightarrow Ref(k, G', k.d.\mathbf{0})$, hence the unwinding condition on $h \neq k$ is still satisfied.

As far as k is concerned we have that if $E' \in Reach(E)$ is such that $E' \dashrightarrow k{\rightarrow}G$, then $Ref(k, E', k.d.\mathbf{0}) \xrightarrow{\tau} k.d.Ref(k, G, k.d.\mathbf{0}) \dashrightarrow k{\rightarrow}d.Ref(k, G, k.d.\mathbf{0})$. Since $k.d.Ref(k, G, k.d.\mathbf{0}) \setminus HD \equiv \mathbf{0} \equiv d.Ref(k, G, k.d.\mathbf{0})$ and $k.d.Ref(k, G, k.d.\mathbf{0}) \dashrightarrow k.d.Ref(k, G, k.d.\mathbf{0})$ we have that also the unwinding condition on k is satisfied. □

Proof of Theorem 7.2. We prove that $E \in DP\_NDC$ implies $E \in AI$.

$\Rightarrow$ Let $E \in DP\_NDC$, i.e., for all $E' \in Reach(E)$, if $E' \xrightarrow{h} E''$ with $h \in H$ then $E' \stackrel{\hat{\tau}}{\Longrightarrow} E'''$ and $E'' \setminus HD \approx_T E''' \setminus HD$. Let $E' \in Reach(E)$ and $\gamma \in Tr((E' \setminus D)/H)$. Then there exists $\gamma' \in Tr(E' \setminus D)$ such that $\gamma = \gamma'/H$ where γ'/H is obtained from γ by deleting all high level actions occurring in it. We show that $\gamma \in Tr(E' \setminus HD)$. This is proved by induction on the number of high level actions occurring in γ'. Indeed if γ' does not contain any high level action then $\gamma = \gamma'$ and it trivially belongs to $Tr(E' \setminus HD)$. Otherwise let $\gamma' = t_1, h_1, t_2$ with $t_1 \in L^*$, $h \in H$ and $t_2 \in Act^*$. Hence $E' \setminus D \stackrel{t_1}{\Longrightarrow} E'' \setminus D \xrightarrow{h_1} E''' \setminus D$ and $t_2 \in Tr(E''' \setminus D)$. By the induction hypothesis $t_2/H \in Tr(E''' \setminus HD)$. By the hypothesis that $E \in DP\_NDC$ we have that $E'' \setminus D \stackrel{\hat{\tau}}{\Longrightarrow} \tilde{E}''' \setminus D$ with $E''' \setminus HD \approx_T \tilde{E} \setminus HD$. Hence $E' \setminus HD \stackrel{t_1}{\Longrightarrow} E'' \setminus HD \stackrel{\hat{\tau}}{\Longrightarrow} \tilde{E}''' \setminus HD$ with $t_2/H \in Tr(E''' \setminus HD)$. Thus $\gamma = \gamma'/H = t_1, t_2/H \in E' \setminus HD$. Moreover, if $\gamma \in Tr(E' \setminus HD)$ then it trivially holds that $\gamma \in Tr((E' \setminus D)/H)$. This proves that if $E \in DP\_NDC$ then for all $E' \in Reach(E)$, $Tr((E' \setminus D)/H) = Tr(E' \setminus HD)$, i.e, $(E' \setminus D)/H \approx_T E' \setminus HD$. □

Proof of Theorem 7.5. We prove that $E \in DP\_BNDC$ iff $E \in BNAI$.

$\Rightarrow$ Let E be a process such that for all $E' \in Reach(E)$, if $E' \xrightarrow{h} E''$ with $h \in H$ then $E' \stackrel{\hat{\tau}}{\Longrightarrow} E'''$ and $E'' \setminus HD \approx E''' \setminus HD$. We show that

$$\mathcal{S} = \{((E' \setminus D)/H, E' \setminus HD) \mid E' \text{ is reachable from } E\}$$

is a weak bisimulation up to $\approx_B$. Hence for all E' reachable from E, $(E' \setminus D)/H) \approx_B E' \setminus HD$, i.e., $E \in$ *BNAI*.

To show that $\mathcal{S}$ is weak bisimulation up to $\approx_B$ we have to consider the following cases:

- $(E' \setminus D)/H \xrightarrow{a} (E'' \setminus D)/H$ with $a \in L \cup \{\tau\}$ and $E' \xrightarrow{a} E''$. Hence $E' \setminus HD \xrightarrow{a} E'' \setminus HD$ and, by definition of $\mathcal{S}$, $((E'' \setminus D)/H, E'' \setminus HD) \in \mathcal{S}$.
- $(E' \setminus D)/H \xrightarrow{\tau} (E'' \setminus D)/H$ where $E' \xrightarrow{h} E''$ and $h \in H$. By hypothesis $E' \stackrel{\hat{\tau}}{\Longrightarrow} E'''$ and $E'' \setminus HD \approx_B E''' \setminus HD$. Hence, $E' \setminus HD \stackrel{\hat{\tau}}{\Longrightarrow} E''' \setminus HD$ with $E''' \setminus HD \approx_B E'' \setminus HD$ and $((E'' \setminus D)/H, E'' \setminus HD) \in \mathcal{S}$.
- $E' \setminus HD \xrightarrow{a} E'' \setminus HD$ with $a \in L \cup \{\tau\}$ and $E' \xrightarrow{a} E''$. Then, $(E' \setminus D)/H \xrightarrow{a} (E'' \setminus D)/H$ and by definition of $\mathcal{S}$, $((E'' \setminus D)/H, E'' \setminus HD) \in \mathcal{S}$. □

$\Leftarrow$ Let E be $BNAI$. Let $E' \in Reach(E)$ such that $E' \xrightarrow{h} E''$ with $h \in H$. Then $(E' \setminus D)/H \xrightarrow{\tau} (E'' \setminus D)/H$. Since, by definition of *BNAI*, $(E' \setminus D)/H \approx_B E' \setminus HD$ for all $E' \in Reach(E)$, we have that $E' \setminus HD \stackrel{\hat{\tau}}{\Longrightarrow} E''' \setminus HD$ and $(E'' \setminus D)/H \approx_B E'' \setminus HD \approx_B E''' \setminus HD$. Thus $E' \stackrel{\hat{\tau}}{\Longrightarrow} E'''$ and $E'' \setminus HD \approx_B E''' \setminus HD$. □