Available online at www.sciencedirect.com
SCIENCE DIRECT
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 99 (2004) 127–154
www.elsevier.com/locate/entcs

Unwinding in Information Flow Security ⋆

A. Bossi[1] R. Focardi[2] D. Macedonio[3] C. Piazza[4] S. Rossi[5]

Dipartimento di Informatica, Università Ca' Foscari di Venezia
via Torino 155, 30172 Venezia, Italy

Abstract

We study information flow security properties which are *persistent*, in the sense that if a system is secure then all of its reachable states are secure too. We present a uniform characterization of these properties in terms of a general *unwinding* schema. This unwinding characterization allows us to prove several compositionality properties of the considered security classes. Moreover, we exploit the unwinding condition to dictate the form of the rules we can use to incrementally develop secure processes and to rectify insecure processes.

1 Introduction

Information flow security properties have been proposed as a means to ensure confidentiality of classified information in multilevel systems. These properties impose constraints on information flow among different groups of entities with different security levels. Often only two groups are considered and are labelled with the security levels *high* (H) and *low* (L). The condition is that no information should flow from H to L.

⋆ This work has been partially supported by MIUR project "Modelli formali per la sicurezza", the EU project MyThS (IST-2001-32617) and the FIRB project (RBAU018RCZ) "Interpretazione astratta e model checking per la verifica di sistemi embedded".

[1] Email: bossi@dsi.unive.it
[2] Email: focardi@dsi.unive.it
[3] Email: mace@dsi.unive.it
[4] Email: piazza@dsi.unive.it
[5] Email: srossi@dsi.unive.it

1571-0661/$ – see front matter
doi:10.1016/j.entcs.2004.02.006

The necessity of controlling information flow as a whole (both direct and indirect) motivated Goguen and Meseguer in introducing the notion of *Non-interference* [19,20]. Non-Interference formalizes the absence of information flow within deterministic systems. Given a system in which *confidential* (i.e., high level) and *public* (i.e., low level) information may coexist, *non-interference* requires that confidential inputs never affect the outputs on the public interface of the system, i.e., never interfere with the low level users. If such a property holds, one can conclude that no information flow is ever possible from high to low level.

Starting from Sutherland [36], many definitions extending the concept of non-interference to non-deterministic systems have been proposed in the literature. They are developed in different settings such as programming languages [4,33,34,35], trace models [25,26], process calculi [11,14,22,30,31,32], probabilistic models [2,12], timed models [15,21], cryptographic protocols [1,5,16].

In [13], Focardi and Gorrieri express the concept of non-interference in the *Security Process Algebra* (*SPA*) language, in terms of bisimulation semantics. In particular, inspired by [37], they introduce the notion of *Bisimulation-based non Deducibility on Compositions* (*BNDC*): a system E is *BNDC* if what a low level user sees of the system is not modified (in the sense of the bisimulation semantics) by composing any high level process Π with E. The main advantage of *BNDC* with respect to trace-based properties is that it is powerful enough to detect information flows due to the possibility, for a high level malicious process, to block or unblock a system. In particular, in [13,14], it is shown that a malicious process may build a channel from high to low, by suitably blocking and unblocking some system services accessible by low level users. The system used to build this covert channel turns out to be secure with respect to trace-based properties. This motivates the use of more discriminating equivalences such as bisimulation.

Although Martinelli [24] has shown that a class of *BNDC*-like properties is decidable over finite state processes, the problem of efficiently verifying *BNDC* is still open. Indeed, decidability of *BNDC* is an open problem. The main difficulty consists of getting rid of the universal quantification on high level processes Π. Another drawback of *BNDC* is that it is not compositional with respect to the main SPA operators, such as the parallel composition and the nondeterministic choice. Compositionality results are useful since they help in designing efficient verification algorithms and in defining *proof systems* which allow one to incrementally build systems which are secure by construction.

For these reasons many decidable and compositional sufficient conditions for *BNDC* have been studied in the literature. In [9] it has been proved that four of these sufficient conditions, namely *Persistent_BNDC* (*P_BNDC*),

Strong BNDC (*SBNDC*), *Compositional P_BNDC* (*CP_BNDC*), and *Progressing P_BNDC* (*PP_BNDC*), can be characterized in terms of *unwinding conditions*.

Unwinding conditions demand properties of individual actions: they aim at "distilling" the local effect of performing high level actions. As observed by many authors (see [31,29,23,32]) they are easier to handle and more amenable to automated proof with respect to global conditions.

In this paper we bridge the gap between unwinding conditions and compositionality. In particular, we introduce a parametric notion of unwinding which generalizes the unwinding characterizations considered in [9]. We exploit the parametric unwinding condition to formulate general compositionality results. Such results aim at establishing a link between the semantics of the operator with respect to which we want to ensure compositionality and the relations involved in the unwinding condition. The compositionality properties of *P_BNDC*, *SBNDC*, *CP_BNDC*, and *PP_BNDC* are just special instances of our general results. In the same spirit, we analize how to preserve unwinding conditions under *refinement* (see [8]). By exploiting the parametric unwinding condition and its general compositionality properties, we can also define *proof systems* (see [7]) which allow us to build processes which are secure by construction. Finally we suggest methods to *rectify* (see [6]) insecure processes in order to obtain processes which satisfy the unwinding conditions characterizing specific security properties.

The paper is organized as follows. In Section 2, we recall the syntax and the semantics of the *SPA* language. In Section 3 we introduce the security properties *BNDC* and *P_BNDC*. Moreover, in Section 3.1 we define a general unwinding schema and give a uniform presentation of the security properties *P_BNDC*, *SBNDC*, *CP_BNDC*, and *PP_BNDC* as different instances of the general schema. In Section 4 we analize the relationships between unwinding conditions and compositionality with respect to the SPA operators and refinement. We exploit these results to develop proof systems for properties characterized through unwinding. In Section 5 we exploit the general unwinding schema to present a method for rectifying insecure processes. Finally, in Section 6 we draw some conclusions.

This paper surveys previous work by the authors [6,7,8,9,10]. The above mentioned general framework is an original contribution which allows us to uniformly present our results and also to generalize some of them.

2 Preliminaries

In this section we report the syntax and semantics of the process algebra we consider. It is a variation of Milner's CCS [27], similar to the *Security Process Algebra* (SPA, for short) language [14], where the set of visible actions is partitioned into high level actions and low level ones in order to specify multilevel systems. In addition to constant definitions, we allow one to use the *replication* (!) operator for defining recursive systems.

The syntax of our process algebra is based on the same elements as CCS that is: 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 special action τ which models internal computations, i.e., not visible outside the system; a complementation function $\bar{\cdot} : \mathcal{L} \to \mathcal{L}$, such that $\bar{\bar{a}} = a$, for all $a \in \mathcal{L}$. $Act = \mathcal{L} \cup \{\tau\}$ is the set of all *actions*. The set of visible actions is partitioned into two sets, H and L, of high and low actions such that $\overline{H} = H$ and $\overline{L} = L$.

The syntax of SPA *terms*[6] (or *processes*) is defined as follows:

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

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

Intuitively, $\mathbf{0}$ is the empty process that does nothing; $a.E$ is a process that can perform an action a and then behaves as E; $E_1 + E_2$ represents the nondeterministic choice between the two processes E_1 and E_2; $E_1|E_2$ is the parallel composition of E_1 and E_2, where executions are interleaved, possibly synchronized on complementary input/output actions, producing an internal action τ; $E \setminus v$ is a process E prevented from performing actions in v; $E[f]$ is the process E whose actions are renamed *via* the relabelling function f; $!E$ (bang E) is the process $E|E|\ldots$, i.e., the parallel composition of as many copy as needed of the process E.

We say that a process E is *guarded* if it can be built by using the rule $a.E + a.E$ instead of $E + E$ in the syntax of SPA terms above.

We denote by $\mathcal{E}$ the set of all SPA processes and by $\mathcal{E}_H$ the set of all high level processes, i.e., those constructed only using actions in $H \cup \{\tau\}$.

The operational semantics of SPA agents is given in terms of *Labelled Transition Systems* (LTS, for short). A LTS is a triple $(S, A, \to)$ where S is a

[6] Actually, the SPA syntax does not include the ! operator. We maintain the name SPA for our language since adding ! does not increase the expressive power of the language.

set of states, A is a set of labels (actions), $\rightarrow \subseteq S \times A \times S$ is a set of labelled transitions. The notation $(S_1, a, S_2) \in \rightarrow$ (or equivalently $S_1 \xrightarrow{a} S_2$) means that the system can move from the state S_1 to the state S_2 through the action a. The operational semantics of SPA is the LTS $(\mathcal{E}, Act, \rightarrow)$, where the states are the terms of the algebra and the transition relation $\rightarrow \subseteq \mathcal{E} \times Act \times \mathcal{E}$ is defined by structural induction as the least relation generated by the inference rules depicted in Figure 1. We use also the notion of *rooted* labelled transition system which is a LTS augmented with a distinguish node, the root. Given a process E we denote by $LTS(E) = (S_E, E, Act, \rightarrow)$ the rooted LTS constituted of the subpart of the SPA LTS reachable from E. E is a *finite-state* process if $LTS(E)$ has a finite number of nodes, that is S_E is finite.

The concept of *observation equivalence* is used to establish equalities among processes and it 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 $\mathcal{E}$. The *strong bisimulation* relation [27] equates two processes if they are able to mutually simulate their behavior step by step.

We will use the following auxiliary notations. Act^* denotes the set of (possibly empty) sequences of actions, while Act^+ denotes the set of nonempty sequences of actions. If $t = a_1 \cdots a_n \in Act^*$ and $E \xrightarrow{a_1} \cdots \xrightarrow{a_n} E'$, then we write $E \xrightarrow{t} E'$. We also write $E \stackrel{t}{\Longrightarrow} E'$ if $E(\xrightarrow{\tau})^* \xrightarrow{a_1} (\xrightarrow{\tau})^* \cdots (\xrightarrow{\tau})^* \xrightarrow{a_n} (\xrightarrow{\tau})^* E'$ where $(\xrightarrow{\tau})^*$ denotes a (possibly empty) sequence of τ labelled 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 $E(\xrightarrow{\tau})^* E'$ if $a = \tau$ (note that $\stackrel{\tau}{\Longrightarrow}$ requires at least one τ labelled transition while $\stackrel{\hat{\tau}}{\Longrightarrow}$ means zero or more τ labelled transitions). We say that E' is reachable from E when there exists $t \in Act^*$ such that $E \xrightarrow{t} E'$. We denote by $Reach(E)$ the set of all sates reachable from E.

The notion of *strong bisimulation* can be defined through the *simulation* preorder as follows.

Definition 2.1 [Simulation] A binary relation $\mathcal{R} \subseteq \mathcal{E} \times \mathcal{E}$ over agents is a *simulation* if $(E, F) \in \mathcal{R}$ implies, for all $a \in Act$,

- if $E \xrightarrow{a} E'$, then there exists F' such that $F \xrightarrow{a} F'$ and $(E', F') \in \mathcal{R}$.

An agent E is *simulated by* another agent F, denoted by $E \leq F$, if there exists a simulation $\mathcal{R}$ containing the pair (E, F).

The relation $\leq$ is the largest simulation and it is a preorder relation, i.e., it is reflexive and transitive.

Prefix $$\frac{-}{a.E \xrightarrow{a} E}$$

Sum $$\frac{E_1 \xrightarrow{a} E_1'}{E_1 + E_2 \xrightarrow{a} E_1'} \qquad \frac{E_2 \xrightarrow{a} E_2'}{E_1 + E_2 \xrightarrow{a} E_2'}$$

Parallel $$\frac{E_1 \xrightarrow{a} E_1'}{E_1|E_2 \xrightarrow{a} E_1'|E_2} \qquad \frac{E_2 \xrightarrow{a} E_2'}{E_1|E_2 \xrightarrow{a} E_1|E_2'}$$

$$\frac{E_1 \xrightarrow{a} E_1' \quad E_2 \xrightarrow{\bar{a}} E_2'}{E_1|E_2 \xrightarrow{\tau} E_1'|E_2'}\ a \in \mathcal{L}$$

Restriction $$\frac{E \xrightarrow{a} E'}{E \setminus v \xrightarrow{a} E' \setminus v}\ \text{if } a \notin v$$

Relabelling $$\frac{E \xrightarrow{a} E'}{E[f] \xrightarrow{f(a)} E'[f]}$$

Constant $$\frac{E \xrightarrow{a} E'}{Z \xrightarrow{a} E'}\ \text{if } Z \stackrel{\text{def}}{=} E$$

Replication $$\frac{E \xrightarrow{a} E'}{!E \xrightarrow{a} E'|!E} \qquad \frac{E \xrightarrow{a} E' \quad E \xrightarrow{\bar{a}} E''}{!E \xrightarrow{\tau} E'|E''|!E}\ a \in \mathcal{L}$$

Fig. 1. The operational rules for SPA

Definition 2.2 [Strong Bisimulation] A binary relation $\mathcal{R} \subseteq \mathcal{E} \times \mathcal{E}$ over agents is a *strong bisimulation* if both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are simulations.
Two agents $E, F \in \mathcal{E}$ are *strongly bisimilar*, denoted by $E \sim_B F$, if there exists a strong bisimulation $\mathcal{R}$ containing the pair (E, F).

The relation $\sim_B$ is the largest strong bisimulation and it is an equivalence relation.

In many applications strong bisimulation is too demanding, i.e., it is too fine. In particular, the internal transitions are treated as all the other actions. The *weak bisimulation* relation is similar to strong bisimulation, but it does not care about internal τ actions.

Definition 2.3 [Weak Bisimulation] A binary relation $\mathcal{R} \subseteq \mathcal{E} \times \mathcal{E}$ over agents is a *weak bisimulation* if $(E, F) \in \mathcal{R}$ implies, for all $a \in Act$,

- if $E \xrightarrow{a} E'$, then there exists F' such that $F \stackrel{\hat{a}}{\Longrightarrow} F'$ and $(E', F') \in \mathcal{R}$;
- if $F \xrightarrow{a} F'$, then there exists E' such that $E \stackrel{\hat{a}}{\Longrightarrow} E'$ and $(E', F') \in \mathcal{R}$.

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

The relation $\approx_B$ is the largest weak bisimulation and it is an equivalence relation [27].

In our security properties we need the notions of *weak bisimulation on low actions*, which equates processes which are bisimilar from the low level user point of view, and *progressing bisimulation on low actions*, which also requires that each τ action is simulated by at least one τ.

Definition 2.4 [Weak Bisimulation on Low Actions] A binary relation $\mathcal{R} \subseteq \mathcal{E} \times \mathcal{E}$ over agents is a *weak bisimulation on low actions* if $(E, F) \in \mathcal{R}$ implies, for all $\ell \in L \cup \{\tau\}$,

- if $E \xrightarrow{\ell} E'$, then there exists F' such that $F \stackrel{\hat{\ell}}{\Longrightarrow} F'$ and $(E', F') \in \mathcal{R}$;
- if $F \xrightarrow{\ell} F'$, then there exists E' such that $E \stackrel{\hat{\ell}}{\Longrightarrow} E'$ and $(E', F') \in \mathcal{R}$.

Two agents $E, F \in \mathcal{E}$ are *weakly bisimilar on low actions*, denoted by $E \approx^l_B F$, if there exists a weak bisimulation $\mathcal{R}$ containing the pair (E, F).

It is immediate to prove that $E \approx^l_B F$ is equivalent to $E \setminus H \approx_B F \setminus F$. Progressing bisimulation on low actions is similar to weak bisimulation on low actions, but it is based on the notion of progressing bisimulation introduced in [28].

Definition 2.5 [Progressing Bisimulation on Low Actions] A binary relation $\mathcal{R} \subseteq \mathcal{E} \times \mathcal{E}$ over agents is a *progressing bisimulation on low actions* if $(E, F) \in \mathcal{R}$ implies, for all $\ell \in L \cup \{\tau\}$,

- if $E \xrightarrow{\ell} E'$, then there exists F' such that $F \stackrel{\ell}{\Longrightarrow} F'$ and $(E', F') \in \mathcal{R}$;
- if $F \xrightarrow{\ell} F'$, then there exists E' such that $E \stackrel{\ell}{\Longrightarrow} E'$ and $(E', F') \in \mathcal{R}$.

Two agents E, F are *progressing bisimilar on low actions*, denoted by $E \approx^l_P F$, if there exists a weak bisimulation $\mathcal{R}$ containing the pair (E, F).

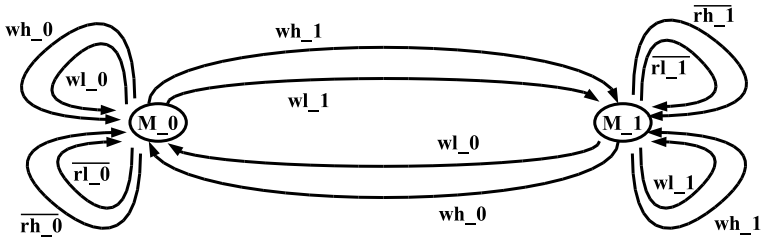

Fig. 2. The LTS of the memory cell M_x.

3 Bisimulation Based Security Properties

In [13], Focardi and Gorrieri express the concept of non-interference in terms of bisimulation semantics through the notion of *Bisimulation-based non Deducibility on Compositions* (*BNDC*). Property *BNDC* is based on the idea of checking the system against all high level potential interactions, representing every possible high level malicious program. In particular, a system E is *BNDC* if for every high level process Π a low level user cannot distinguish E from $(E|\Pi)$, i.e., if Π cannot interfere with the low level execution of the system E. In other words, a system E is *BNDC* if what a low level user sees of the system is not modified by composing any high level process Π to E.

Definition 3.1 [BNDC] Let $E \in \mathcal{E}$. $E \in BNDC$ if

$$\forall\ \Pi \in \mathcal{E}_H,\ E \approx^l_B (E|\Pi)$$

Example 3.2 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. [7] The LTS of process M_x is depicted in Figure 2.

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

Notice that read (write) operations are modelled as outputs (inputs). Process M_x can send the stored value x through the two output actions $\overline{r_h\_x}$ and $\overline{r_l\_x}$. Moreover, write operations are performed by accepting an input w_h_y

[7] 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.

and w_l_y (with $y \in \{0,1\}$) and moving to M_y, i.e., storing y into the memory cell.

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 *BNDC* we detect that M_0 and M_1 are insecure. In fact, considering the process $\Pi \equiv \overline{w_h\_1}.\mathbf{0}$ we get that $(M_0|\Pi)\backslash H \equiv \tau.M_1 \setminus H$ which is not weak bisimilar to $M_0 \setminus H$, since in $\tau.M_1 \setminus H$ the low level user reads 1, while in $M_0 \setminus H$ he reads 0.

In [14], Focardi and Gorrieri observe that the *BNDC* property is difficult to use in practice: its decidability is still an open problem. It would be desirable to have an alternative formulation of *BNDC* which avoids the universal quantification on high level processes and exploits local information only. One of the main difficulty in finding such an alternative characterization comes from the fact that *BNDC* is not persistent and thus the requirements on the processes reachable from a *BNDC* process E should be different from the requirements on E itself. In [17], it is introduced a security property called *Persistent_BNDC* (*P_BNDC*, for short), in which persistence is imposed by definition.

Definition 3.3 [P_BNDC] Let $E \in \mathcal{E}$. $E \in P\_BNDC$ if

$$\forall\ E' \in Reach(E),\ E' \in BNDC.$$

The decidability of *P_BNDC* over finite state processes has been proved in [17] by exploiting a bisimulation based characterization.

A standard way to protect confidential data is to apply the multilevel security model of [3]. First, we need to assign a security level to any information containers (called *objects*); then the following access control rules are imposed: (*i*) no low level user can read from high level objects; (*ii*) no high user can write into low level objects. Indeed, these are the only two (direct) ways for leaking confidential information. Sometimes they are sufficient to ensure security as described in the following example.

Example 3.4 The memory cell of Example 3.2 is neither *BNDC* nor *P_BNDC*.

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 (rule (*i*) above),

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

and a low level cell M^l_x, by eliminating any high level write operation (rule (*ii*) above):

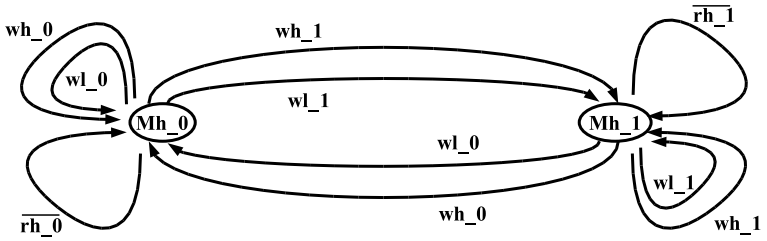

Fig. 3. The LTS of the memory cell M^h_x.

$$M^l\_x \stackrel{\text{def}}{=} \overline{r_h\_x} \;.\; M^l\_x + \overline{r_l\_x} \;.\; M^l\_x \\ + w_l\_0 \;.\; M^l\_0 + w_l\_1 \;.\; M^l\_1$$

We can prove that both M^h_x and M^l_x are *P_BNDC*.

Other bisimulation based persistent security properties have been studied in the literature. We recall here the following: *Strong BNDC* (*SBNDC*, for short), introduced in [13], *Compositional P_BNDC* (*CP_BNDC*, for short), introduced in [8], and *Progressing P_BNDC* (*PP_BNDC*, for short), introduced in [9]. All these properties are included in the *BNDC* class, i.e., if a process satisfies one of them, then it is *BNDC*. In the next subsection we introduce them through a uniform unwinding definition.

3.1 Unwinding Definitions

The idea behind the notion of unwinding is to introduce some constraints on the transitions of the system (see [32]) which imply some global properties. In particular, when an unwinding condition is used to define a non-interference 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 (see [29]).

In this section we give a uniform presentation of the security properties *P_BNDC*, *SBNDC*, *CP_BNDC*, and *PP_BNDC* by introducing a generalized unwinding condition. Our unwinding is parametric with respect to two binary relations on processes: an equivalence relation, $\backsim^l$, which represents the low level indistinguishability and a transition relation, $\dashrightarrow$, which characterizes the local connectivity required by the unwinding condition.

Definition 3.5 [Generalized Unwinding] Let $\backsim^l$ be a binary equivalence relation on $\mathcal{E}$ and $\dashrightarrow$ be a binary relation on $\mathcal{E}$. We define the *unwinding class*

$\mathcal{W}(\backsim^l, \dashrightarrow)$ as

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

The unwinding condition characterizing an unwinding class clearly implies persistence. Moreover, any process E which does not perform high level actions belongs to any unwinding class $\mathcal{W}(\backsim^l, \dashrightarrow)$, since the unwinding condition is trivially satisfied.

The following theorem follows from the unwinding characterizations of *P_BNDC* studied in [7] and of *PP_BNDC* studied in [9], and from the original definitions of *SBNDC* in [13] and of *CP_BNDC* in [8].

Theorem 3.6 (Unwinding) *Let $E \in \mathcal{E}$ be a process.*

- $E \in$ *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* $E \in \mathcal{W}(\approx^l_B, \stackrel{\tau}{\Longrightarrow})$;
- $E \in$ *PP_BNDC iff* $E \in \mathcal{W}(\approx^l_P, \stackrel{\tau}{\Longrightarrow})$;

where $\equiv$ is the syntactic equality between processes.

The above theorem helps us to understand the local meaning of our security properties. Let F be a process reachable from a *P_BNDC* process E. If F can perform a high level transition reaching a process G, then F can also simulate such a move reaching, through a (possible empty) sequence of silent transitions, a process G' which is undistinguishable from G from a low level view. In the case of *SBNDC* the sequence of silent transitions is replaced by no transitions, i.e., G' is F itself, while in the case of *CP_BNDC* and *PP_BNDC* the silent sequence cannot be empty. Moreover, in *PP_BNDC* weak bisimulation on low actions is replaced by progressing bisimulation on low actions.

Example 3.7 Consider the memory cells M^h_x and M^l_x described in Example 3.4. Exploiting the unwinding characterization of *P_BNDC* given in Theorem 3.6 it is easy to see that both M^h_0 and M^h_1 are *P_BNDC*. First, notice that $M^h\_0 \approx^l_B M^h\_1$, since there is no way for a low level user to distinguish between the two states. As a matter of fact, the only possible low level actions are the two write operations w_l_0, w_l_1 which, both in M^h_0 and in M^h_1, move the system into the same states. The fact that M^l_0 and M^l_1 are *P_BNDC* is even easier to prove: the only high level actions r_h_0, r_h_1 do not change the system state. Moreover, since neither M^h_x nor M^l_x perform

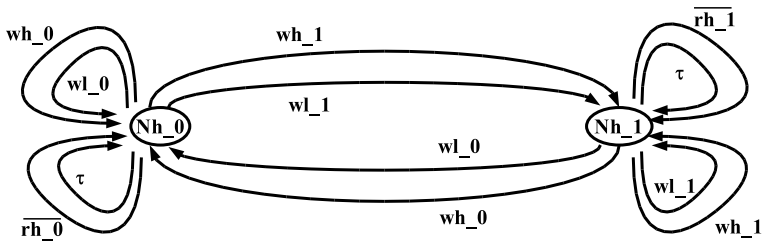

Fig. 4. The LTS of the memory cell N^h_x.

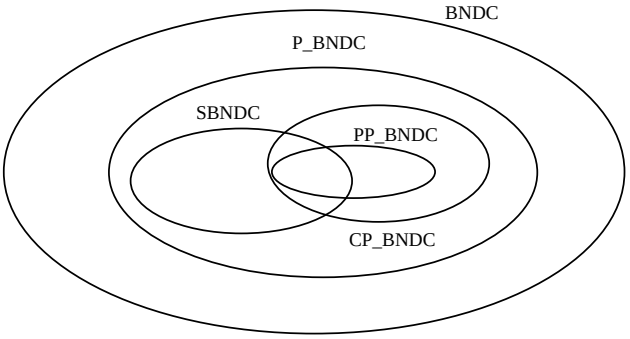

Fig. 5. Security Properties.

any τ transition, one can infer that they are also *SBNDC*. Finally, one can notice that M^h_x and M^l_x are neither *CP_BNDC* nor *PP_BNDC*, since there are not τ moves executable by the two processes.

Consider now the processes N^h_x (see Figure 4) and N^l_x obtained by adding a time-out realized by a τ-loop in the initial state of both M^h_x and M^l_x, i.e.,

$$N^h\_x \stackrel{\text{def}}{=} \overline{r_h\_x} \;.\; N^h\_x + w_h\_0 \;.\; N^h\_0 + w_h\_1 \;.\; N^h\_1 + w_l\_0 \;.\; N^h\_0 + w_l\_1 \;.\; N^h\_1 + \tau \;.\; N^h\_x$$

$$N^l\_x \stackrel{\text{def}}{=} \overline{r_h\_x} \;.\; N^l\_x + \overline{r_l\_x} \;.\; N^l\_x + w_l\_0 \;.\; N^l\_0 + w_l\_1 \;.\; N^l\_1 + \tau \;.\; N^l\_x.$$

The processes N^h_x and N^l_x are both *CP_BNDC* and *PP_BNDC*.

The unwinding characterizations allow us to easily prove that *PP_BNDC* $\subset$ *CP_BNDC* $\subset$ *P_BNDC*, *SBNDC* $\subset$ *P_BNDC*, and the processes containing only low level actions satisfy all of them. The situation is summarized in Figure 5.

4 How to Incrementally Build Secure Processes

Compositionality is useful for both verification and synthesis. On one hand, if a property is preserved when systems are composed, then the analysis process can be decomposed and applied to subsystems in order to prove that the system as a whole satisfies the desired property. On the other hand, in the synthesis of a system, compositionality makes it possible to deal with all the subcomponents in a uniform way. In this section we analyze the relations between the unwinding conditions and compositionality results. We show that all the security properties we considered are compositional with respect to the parallel operator, while not all of them are *fully* compositional. In particular, *P_BNDC* and *SBNDC* are not preserved by the nondeterministic choice operator. In general, when we build a system that may (nondeterministically) choose to behave as one of two secure subsystems, we could obtain an insecure system. As also observed in [18], this seems to be counterintuitive. On the contrary, *PP_BNDC* and *CP_BNDC* are fully compositional, i.e., they are compositional also with respect to the nondeterministic choice.

Besides standard algebra operators, we also consider refinement operators which are useful for the stepwise development of secure processes. Indeed, one usually starts from a very abstract specification of the desired system which is then refined and decomposed until one arrives at a concrete specification that can directly be implemented. If properties are preserved under each refinement step then those properties which have been already investigated in some phase need not to be re-investigated in later phases.

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

To study compositionality properties of unwinding classes we first introduce the following notions of *preservation* and *reflection*.

Definition 4.1 [Preservation and Reflection] Let $f : \mathcal{E}^k \longrightarrow \mathcal{E}$ be a partial function and $\odot \subseteq \mathcal{E} \times \mathcal{E}$ be a relation.
The function f *preserves* $\odot$ if the following condition holds. Let $I \uplus J$ be any partition of $\{1, \ldots, k\}$ with $I \neq \emptyset$. If $\forall i \in I\ (G_i \odot G'_i)$ and $\forall j \in J\ (G_j \equiv G'_j)$ then

$$f(G_1, \ldots, G_k) \odot f(G'_1, \ldots, G'_k) \text{ or } (f(G_1, \ldots, G_k)\uparrow \text{ and } f(G'_1, \ldots, G'_k)\uparrow)$$

The function f *reflects* $\odot$ if the following condition holds. If $f(G_1, \ldots, G_k) \odot$

M, then $\exists\ I, J,\ I \uplus J = \{1, \ldots, k\}$ and $I \neq \emptyset$ such that

$$\forall i \in I\ (G_i \odot G'_i) \text{ and } \forall j \in J\ (G_j \equiv G'_j) \text{ and } M \equiv f(G'_1, \ldots, G'_k)$$

The condition $I \neq \emptyset$ in the above definition has the aim of considering also non reflexive relations, e.g. the relation $\xrightarrow{h}$.

Example 4.2 Let $\odot$ be the weak bisimulation relation, i.e., $E \odot F$ if and only if $E \approx_B F$. It holds that the parallel composition operator preserves weak bisimulation [27]. On the other hand, the nondetermistic choice operator does not preserve weak bisimulation. In fact, $\mathbf{0} \approx_B \tau.\mathbf{0}$, but $a.\mathbf{0} + \mathbf{0} \not\approx_B a.\mathbf{0} + \tau.\mathbf{0}$.

Let $\odot$ be the reachability relation, i.e., $\{(E, F) \mid E \in \mathcal{E} \text{ and } F \in Reach(E)\}$. The parallel operator reflects $\odot$. In fact, if $G_1|G_2$ reaches M, i.e., $M \in Reach(G_1|G_2)$ then $M \equiv G'_1|G'_2$ with both $G'_1 \in Rach(G_1)$ and $G'_2 \in Rach(G_2)$.

Compositionality of an unwinding class can be proved by means of the following theorem.

Theorem 4.3 (Reflection-Preservation Composition) *Let $f : \mathcal{E}^k \longrightarrow \mathcal{E}$ be a partial function reflecting $\xrightarrow{h}$ and the reachability relation and preserving $\dashrightarrow$ and $\sim^l$. Then $\mathcal{W}(\sim^l, \dashrightarrow)$ is compositional with respect to f.*

Proof. It is not restrictive to assume $k = 2$.

Let $E, F \in \mathcal{W}(\sim^l, \dashrightarrow)$. We have to prove that $f(E, F) \in \mathcal{W}(\sim^l, \dashrightarrow)$. If $f(E, F)$ reaches M, then, since f reflects the reachability relation [8], there exist G, K (one of them possibly equal to E or F, respectively) such that E reaches G, F reaches K, and $M \equiv f(G, K)$. If $M \xrightarrow{h} M'$, then, since f reflects $\xrightarrow{h}$, three cases are possible:

- $G \xrightarrow{h} G'$ and $M' \equiv f(G', K)$;
- $K \xrightarrow{h} K'$ and $M' \equiv f(G, K')$;
- $G \xrightarrow{h} G'$, $K \xrightarrow{h} K'$, and $M' \equiv f(G', K')$.

In the first case $G \dashrightarrow G''$ and $G'' \sim^l G'$. Hence, since f preserves $\dashrightarrow$ and $\sim^l$ we have $M \dashrightarrow f(G'', K)$ and $f(G'', K) \sim^l f(G', K)$. The second and the third cases are similar. Hence, $f(E, F) \in \mathcal{W}(\sim^l, \dashrightarrow)$. □ □

As we will see in the next subsection, the hypotheses of the above theorem are satisfied when we deal with operators whose semantics is recursively defined on subprocesses (e.g., the parallel operator |). Other operators have a

[8] Note that the reachability relation is reflexive.

semantics which is a "union" of the semantics of subprocesses (e.g., the nondeterministic choice operator). To deal with such kind of operators we introduce the notions of *propagation* and *projection*.

Definition 4.4 [Propagation and Projection] Let $f : \mathcal{E}^k \longrightarrow \mathcal{E}$ be a partial function and $\odot \subseteq \mathcal{E} \times \mathcal{E}$ be a relation.
The function f *propagates* $\odot$ if the following condition holds. If $\exists i$ such that $(G_i \odot G_i')$, then $f(G_1, \ldots, G_k) \odot G_i'$ or $f(G_1, \ldots, G_k) \uparrow$.
The function f *projects* $\odot$ if the following condition holds. If $f(G_1, \ldots, G_k) \odot M$, then $\exists i$ such that $G_i \odot M$.

Example 4.5 Let $\odot$ be the relation $\stackrel{a}{\rightarrow}$ and f be the nondetermistic choice operator $+$. It holds that $+$ propagates $\stackrel{a}{\rightarrow}$. In fact, if $G_1 \stackrel{a}{\rightarrow} G_1'$ then $G_1 + G_2 \stackrel{a}{\rightarrow} G_1'$. Moreover, $+$ projects $\stackrel{a}{\rightarrow}$, since if $G_1 + G_2 \stackrel{a}{\rightarrow} M$ then either $G_1 \stackrel{a}{\rightarrow} M$ or $G_2 \stackrel{a}{\rightarrow} M$.

We say that a process E *positively reaches* a process E' if there exists a process E'' and an action a such that $E \stackrel{a}{\rightarrow} E''$ and E'' reaches E'.

Theorem 4.6 (Projection-Propagation Composition) *Let $f : \mathcal{E}^k \longrightarrow \mathcal{E}$ be a partial function projecting $\stackrel{h}{\rightarrow}$ and the positive reachability relation and propagating $\dashrightarrow$. Then $\mathcal{W}(\sim^l, \dashrightarrow)$ is compositional with respect to f.*

Proof. It is not restrictive to assume $k = 2$.

Let $E, F \in \mathcal{W}(\sim^l, \dashrightarrow)$. We have to prove that $f(E, F) \in \mathcal{W}(\sim^l, \dashrightarrow)$. If $f(E, F)$ reaches M, then, two cases are possible:

- $M \equiv f(E, F)$;
- $f(E, F)$ positively reaches M.

In the first case we have to prove that if $f(E, F) \stackrel{h}{\rightarrow} M'$, then $f(E, F) \dashrightarrow M''$ and $M'' \sim^l M'$. If $f(E, F) \stackrel{h}{\rightarrow} M'$, since f projects $\stackrel{h}{\rightarrow}$, it is not restrictive to assume that $E \stackrel{h}{\rightarrow} M'$. Since $E \in \mathcal{W}(\sim^l, \dashrightarrow)$, by definition, $E \dashrightarrow M''$ and $M'' \sim^l M'$. From the fact that f propagates $\dashrightarrow$ we get that $f(E, F) \dashrightarrow M''$, i.e., the thesis.

In the second case, since f projects the positive reachability relation, we can safely assume that E reaches M. Since $E \in \mathcal{W}(\sim^l, \dashrightarrow)$ and E reaches M, we immediately get the thesis. □
□

4.1 Compositionality with respect to the Algebra Operators

The following result is an immediate consequence of Theorem 4.3, since all the operators it deals with reflect $\stackrel{h}{\rightarrow}$ and the reachability relation.

Corollary 4.7 (Restriction, Renaming, Parallel, Definition) *Consider an unwinding class of processes* $\mathcal{W}(\sim^l, \dashrightarrow)$.

- *Let* $v \subseteq \mathcal{L}$. *If the function* $\mathrm{rest_v} : \mathcal{W}(\sim^l, \dashrightarrow) \longrightarrow \mathcal{E}$ *defined as* $\mathrm{rest_v}(E) = E \setminus v$ *preserves* $\dashrightarrow$ *and* $\sim^l$, *then* $\mathcal{W}(\sim^l, \dashrightarrow)$ *is compositional with respect to the* v*-restriction;*
- *Let* g *be a renaming. If the function* $\mathrm{ren_g} : \mathcal{W}(\sim^l, \dashrightarrow) \longrightarrow \mathcal{E}$ *defined as* $\mathrm{ren_g}(E) = E[g]$ *preserves* $\dashrightarrow$ *and* $\sim^l$, *then* $\mathcal{W}(\sim^l, \dashrightarrow)$ *is compositional with respect to the* g*-renaming;*
- *If the function* $\mathrm{par} : \mathcal{W}(\sim^l, \dashrightarrow)^2 \longrightarrow \mathcal{E}$ *defined as* $\mathrm{par}(E, F) = E|F$ *preserves* $\dashrightarrow$ *and* $\sim^l$, *then* $\mathcal{W}(\sim^l, \dashrightarrow)$ *is compositional with respect to the parallel composition* $|$*;*
- *If the function* $\mathrm{def} : \mathcal{W}(\sim^l, \dashrightarrow) \longrightarrow \mathcal{E}$ *defined as* $\mathrm{def}(E) = Z$, *with* $Z \stackrel{\mathrm{def}}{=} E$, *preserves* $\dashrightarrow$ *and* $\sim^l$, *then* $\mathcal{W}(\sim^l, \dashrightarrow)$ *is compositional with respect to the constant definition* $\stackrel{\mathrm{def}}{=}$.

The following result is a consequence of Theorem 4.6, since the nondeterministic choice operator projects $\stackrel{h}{\rightarrow}$ and the positive reachability relation.

Corollary 4.8 (Non Deterministic Choice) *Let* $\mathcal{W}(\sim^l, \dashrightarrow)$ *be an unwinding class of processes. If the function* $\mathrm{sum} : \mathcal{W}(\sim^l, \dashrightarrow)^2 \longrightarrow \mathcal{E}$ *defined as* $\mathrm{sum}(E, F) = E + F$ *propagates* $\dashrightarrow$, *then* $\mathcal{W}(\sim^l, \dashrightarrow)$ *is compositional with with respect to the nondeterministic choice operator* $+$.

Theorem 4.9 (Low Prefix) *Let* $\mathcal{W}(\sim^l, \dashrightarrow)$ *be an unwinding class of process. If* $l \in L$ *is a low level action, then* $\mathcal{W}(\sim^l, \dashrightarrow)$ *is compositional with respect to the low prefix operator which maps* E *into* $l.E$.

Proof. We have to prove that if $E \in \mathcal{W}(\sim^l, \dashrightarrow)$ and $l \in L$, then $l.E \in \mathcal{W}(\sim^l, \dashrightarrow)$. If $l.E$ reaches E', then two cases are possible:

- $E' \equiv l.E$;
- $E' \in Reach(E)$.

In the first case E' cannot perform any high level action, hence we have nothing to prove. In the second case by the hypothesis that $E \in \mathcal{W}(\sim^l, \dashrightarrow)$ we immediately get the thesis. □ □

The replication operator needs an ad-hoc theorem since it does not reflects $\stackrel{h}{\rightarrow}$ and the reachability relation. In fact, if $!E$ reaches E' this does not correspond to the fact that E reaches E'' and $E' \equiv !E''$. In particular, if $!E$ reaches E' we can prove that E' is of the form $E_1|\ldots|E_n|!E$ where all the E_i's are reached by E. The following theorem allows us to exploit this form

of 'reflection' of the reachability relation to obtain sufficient conditions for the compositionality with respect to the replication operator.

Theorem 4.10 (Replication) *Let $\mathcal{W}(\sim^l, \dashrightarrow)$ be an unwinding class of processes. If it holds that*

(1) *$\dashrightarrow$ is included in the reachability relation, i.e., if $E \dashrightarrow F$ then E reaches F,*

(2) *for each $F \in \mathcal{W}(\sim^l, \dashrightarrow)$ and $k \geq 0$ the function $f_k^F : \mathcal{W}(\sim^l, \dashrightarrow)^k \longrightarrow \mathcal{E}$ defined as $f_k^F(E_1, \ldots, E_k) = E_1|\ldots|E_k|!F$ preserves $\sim^l$,*

(3) *for each $k \geq 0$ the function $g_k : \mathcal{E}^k \longrightarrow \mathcal{E}$ such that $g_k(E_1, \ldots, E_k) = E_1|\ldots|E_k$ preserves $\dashrightarrow$,*

(4) *if $F \dashrightarrow F'$, then $!F \dashrightarrow F'|!F$,*

then $\mathcal{W}(\sim^l, \dashrightarrow)$ is compositional with respect to the replication operator $!$.

Proof. First we prove the following claim.
Claim 1. If $!F$ reaches F', then there exist $n \geq 0$ and $F_1, \ldots, F_n$ such that F reaches F_i, for $i = 1, \ldots, n$ and $F' \equiv F_1|F_2|\ldots|F_n|!F$.

Since $!F$ reaches F', there exists $t \in Act^*$ such that $!F \xrightarrow{t} F'$. We proceed by induction on the length ln of t.

If $ln = 0$, then $F' \equiv !F$, hence we have the thesis with $n = 0$.

Let us assume that we have proved the thesis for all the $ln \leq m$. Let $ln = m+1$. This means that there exists F'' such that $!F \xrightarrow{t'} F''$, t' has length m, and $F'' \xrightarrow{a} F'$. By inductive hypothesis there exist $n \geq 0$ and $F_1, \ldots, F_n$ such that F reaches F_i, for $i = 1, \ldots, n$ and $F'' \equiv F_1|F_2|\ldots|F_n|!F$. If the action a is performed by one of the F_i's, say F_1, we have the thesis, since F reaches F_1 and $F_1 \xrightarrow{a} F_1'$ and $F' \equiv F_1'|F_2|\ldots|F_n|!F$. Similarly we obtain the thesis if $a = \tau$ is a synchronization between two of the F_i's. If the action a is performed by $!F$ applying the first rule of Replication, then $F \xrightarrow{a} F_{n+1}$ and $F' \equiv F_1|F_2|\ldots|F_n|F_{n+1}|!F$. Similarly we obtain the thesis in the remaining two cases, i.e. if a is performed by $!F$ applying the second rule of Replication or if a is a synchronization between one of the F_i's and $!F$.

Now we have to prove that if $F \in \mathcal{W}(\sim^l, \dashrightarrow)$, then $!F \in \mathcal{W}(\sim^l, \dashrightarrow)$, i.e., if $!F$ reaches F' and $F' \xrightarrow{h} G$, then $F' \dashrightarrow G'$ with $G' \sim^l G$.

If $!F$ reaches F', by Claim 1, we have that F' is of the form $F_1|\ldots|F_n|!F$.

If $n = 0$, then $F' \equiv !F$. If $!F \xrightarrow{h} G$, then $F \xrightarrow{h} G''$ and $G \equiv G''|!F$. Since $F \in \mathcal{W}(\sim^l, \dashrightarrow)$, we have that $F \dashrightarrow K$ and $K \sim^l G''$. By hypothesis (4), $!F \dashrightarrow K|!F$. Moreover, by hypothesis (1), we have that $K \in \mathcal{W}(\sim^l, \dashrightarrow)$, hence since, by hypothesis (2), f_1^F preserves $\sim^l$, we get $K|!F \sim^l G''|!F$.

If $n > 0$, then $F' \equiv F_1|\ldots|F_n|!F$. If $F' \xrightarrow{h} G$, then two cases are possible:

- there exists i such that $F_i \xrightarrow{h} F_i'$ and $G \equiv F_1|\ldots|F_i'|\ldots|F_n|!F$;
- $F \xrightarrow{h} F''$ and $G \equiv F_1|\ldots|F_n|F''|!F$.

In the first case, since $F \in \mathcal{W}(\sim^l, \dashrightarrow)$ reaches F_i we have that $F_i \dashrightarrow K$ with $K \sim^l F_i'$. Since, by hypotheses (3) and (2), g_{n+1} preserves $\dashrightarrow$ and f_n^F preserves $\sim^l$ we get that $F' \dashrightarrow G' \equiv F_1|\ldots|K|\ldots|F_n|!F$ and $G' \sim^l G$.

In the second case, since $F \in \mathcal{W}(\sim^l, \dashrightarrow)$, $F \dashrightarrow K$ with $K \sim^l F''$. Hence, by hypothesis (4), we get that $!F \dashrightarrow K|!F$. Since, by hypothesis (3), g_{n+1} preserves $\dashrightarrow$ we have that $F' \dashrightarrow G' \equiv F_1|\ldots|F_n|K|!F$. By hypothesis (1) we obtain $K \in \mathcal{W}(\sim^l, \dashrightarrow)$, hence we can exploit the fact that f_{n+1}^F preserves $\sim^l$ to get $G' \sim^l G$. □
□

We are now ready to apply our general results to the security properties *P_BNDC*, *SBNDC*, *CP_BNDC*, and *PP_BNDC*.

Corollary 4.11 P_BNDC, SBNDC, CP_BNDC, PP_BNDC *are compositional with respect to the following operators:*

- *the l-prefix operator, for each $l \in L$;*
- *the v-restriction operator, for each $v \subseteq \mathcal{L}$;*
- *the g-renaming operator, for each renaming g;*
- *the parallel composition $|$;*
- *the constant definition $\stackrel{\text{def}}{=}$;*
- *the replication operator $!$.*

CP_BNDC, PP_BNDC *are compositional with respect to the nondeterministic choice operator $+$.*

Proof. As far as the first 5 operators are concerned, the compositionality can be proved by observing that the hypothesis of Theorem 4.9 and Corollary 4.7 hold.

To prove the compositionality with respect to the replication operator we need to prove that the hypothesis of Theorem 4.10 hold.

(1) The relations $(\xrightarrow{\tau})^*, (\xrightarrow{\tau})^0, (\xrightarrow{\tau})^+$ are included in the reachability.

(2) We prove that the f_k^F's preserve $\sim^l$. In the case of *P_BNDC* the fact that each f_k^F preserves $\approx_B^l$ can be proved by proving that

$$\mathcal{R} = \{(E_1|\ldots|E_k|!F, E_1'|\ldots|E_k'|!F) \mid E_i, E_i', F \in P\_BNDC \text{ and } E_i \approx_B^l E_i'\}$$

is a weak bisimulation on low actions (see Lemma 5 of [10]). In the case of *SBNDC* and *CP_BNDC* the thesis follows from the case of *P_BNDC*, since

they are both included in *P_BNDC* and they use $\approx^l_B$, as *P_BNDC* does. In the case of *PP_BNDC* the proof can be obtained similarly by proving that

$$\mathcal{R} = \{(E_1|\ldots|E_k|!F, E'_1|\ldots|E'_k|!F) \mid E_i, E'_i, F \in PP\_BNDC \text{ and } E_i \approx^l_P E'_i\}$$

is a progressing bisimulation on low actions.

(3) The fact that each g_k preserves $(\xrightarrow{\tau})^*, (\xrightarrow{\tau})^0, (\xrightarrow{\tau})^+$ is a consequence of the semantics of the parallel operator.

(4) Also the last hypothesis, i.e., $F \dashrightarrow F'$ implies $!F \dashrightarrow F'|!F$, can be easily proved for our security properties (modulo the use of structural congruence in the case of *SBNDC* and *P_BNDC*).

To prove that *CP_BNDC* and *PP_BNDC* are compositional with respect to the nondeterministic choice operator we can apply Corollary 4.8.

□

□

Example 4.12 Consider the parallel composition of the high and low memory cells M^h_x and M^l_x defined in Example 3.4, i.e.,

$$M^{h|l}\_x \stackrel{\text{def}}{=} M^h\_x|M^l\_x.$$

Since both M^h_0 and M^l_0 are *P_BNDC*, by Corollary 4.11, $M^{h|l}\_0$ is *P_BNDC* too. Similarly an unbounded number of high memory cells defined as

$$M^{!h}\_x \stackrel{\text{def}}{=} !M^h\_x.$$

is *P_BNDC*.

Consider now the non-deterministic composition of M^h_x and M^l_x. In particular, consider the memory cell M^{h+l}_x that behaves as either M^h_x or M^l_x, i.e.,

$$M^{h+l}\_x \stackrel{\text{def}}{=} M^h\_x + M^l\_x.$$

We know that M^h_x and M^l_x are *P_BNDC*, however their non-deterministic composition, i.e., M^{h+l}_x, is not. Indeed, consider the execution of a high level write action w_h_0. This moves the whole M^{h+l}_0 system to M^h_0 (notice that M^l_0 does not accept the high level input w_h_0). The problem is that a low level user can observe this move by trying to write some value into the memory cell. As a matter of fact, since M^h_0 does not accept low level inputs, the low level user can deduce that some high level action has been performed. This indirect information flow can be exploited to build a so called *covert-channel* (see, e.g., [14] for more detail). Formally, we can prove that M^{h+l}_0 is neither *P_BNDC*, *SBNDC*, *CP_BNDC* nor *PP_BNDC* by observing that the move $M^{h+l}\_0 \stackrel{w_h\_0}{\rightarrow} M^h\_0$ cannot be simulated by M^{h+l}_0.

Consider now the the memory cell N^{h+l}_x obtained as non-deterministic composition of the cells N^h_x and N^l_x of Example 3.7, i.e.,

$$N^{h+l}\_x \stackrel{\text{def}}{=} N^h\_x + N^l\_x.$$

Since we have already showed that N^h_x and N^l_x are both *CP_BNDC* and *PP_BNDC*, by compositionality results we obtain that N^{h+l}_x is both *CP_BNDC* and *PP_BNDC*. Notice that the problem of simulating the move $N^{h+l}\_0 \stackrel{w_h}{\rightarrow} N^h\_0$ is now solved by performing the τ of the added $\tau \,.\, N^h\_0$ branch in the definition of N^h_0. In particular we have that $N^{h+l}\_0 \stackrel{\tau}{\rightarrow} N^h\_0$.

4.2 Compositionality with respect to Refinement

In [8] we introduced a new notion of refinement for SPA processes. Intuitively, an abstract specification (given here as a SPA system) defines the set of possible (allowed) behaviors of a system. Refining a specification corresponds to choosing among these allowed behaviors, the ones that will be actually implemented. The idea is that a refined specification should never show behaviors that were not foreseen in the initial specification. To formalize this idea, we require that (*i*) each state of the abstract specification is refined to, at most, one state of the more concrete (i.e., refined) specification; (*ii*) the behavior of the refined states is simulated by the abstract states, i.e., it should always be possible to simulate an action performed by a refined state by the corresponding abstract state, and the two reached states should be still one the refinement of the other.

Refinement is formalized as a partial function from processes.

Definition 4.13 (Refinement) A binary relation $\mathcal{R} \subseteq \mathcal{E} \times \mathcal{E}$ over processes is a *refinement* if

- $\mathcal{R}^{-1}$ is a simulation and
- $\mathcal{R}$ is a partial function from $\mathcal{E}$ to $\mathcal{E}$.

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

The following theorem has been proved in [8] but it is easy to see that it is also a consequence of Theorem 4.3. Just note that any refinement $\mathcal{R}$ reflects $\stackrel{h}{\rightarrow}$ and the reachability relation since, by definition, $\mathcal{R}^{-1}$ is a simulation.

Theorem 4.14 *Let $\mathcal{W}(\sim^l, \dashrightarrow)$ be an unwinding class of processes and $\mathcal{R}$ be a refinement. If $\mathcal{R}$ preserves $\dashrightarrow$ and $\sim^l$, then $\mathcal{W}(\sim^l, \dashrightarrow)$ is compositional with respect to $\mathcal{R}$.*

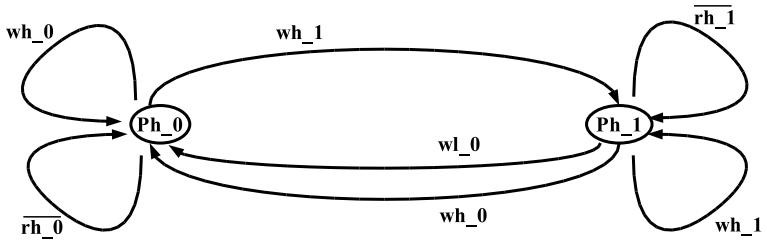

Fig. 6. The LTS of the memory cell P^h_x.

Example 4.15 Consider the processes M^h_x and M^l_x introduced in Example 3.4. We have seen that they are both *P_BNDC*. It is now interesting to study how this property is preserved by further refining the processes. To this aim we apply Theorem 4.14. Notice that neither M^h_0 nor M^l_0 perform any τ transitions, thus the only condition that we should care about is that the refinement preserves $\approx^l_B$. As a consequence, removing high level actions does not affect the security of the two systems. For example, if we allow the high level user to only reset the cell value to 0 (by removing the $w_h\_1 \,.\, M^h\_1$ branch), the resulting process is still *P_BNDC*.

On the other hand, modifications of low behavior should be performed coherently in all equivalent states. For example, the refinement

$$P^h\_0 \stackrel{\text{def}}{=} \overline{r_h\_0} \,.\, P^h\_0 + w_h\_0 \,.\, P^h\_0 + w_h\_1 \,.\, P^h\_1$$
$$P^h\_1 \stackrel{\text{def}}{=} \overline{r_h\_1} \,.\, P^h\_1 + w_h\_0 \,.\, P^h\_0 + w_h\_1 \,.\, P^h\_1 + w_l\_0 \,.\, P^h\_0$$

in which the low level user can reset to 0 the high level cell, only when the cell contains value 1 (notice that in P^h_0 no low level write operations are allowed) is not preserving $\approx^l_B$. The LTS of P^h_0 is depicted in Figure 6.

It is easy to see that $P^h\_0 \notin P\_BNDC$. The fact that P^h_0 is not *P_BNDC* reveals a slightly subtle information flow due to the fact that a low level user may track the content of the high level cell by trying to reset it: every time the reset succeeds the low level user can conclude that the cell contains value 1. A correct refinement achieving the same low level reset behavior described above, should include the branch $w_l\_0 \,.\, P^h\_0$ also in P^h_0.

4.3 Proof Systems for Unwinding classes

Unwinding conditions are also useful for giving efficient proof techniques. Indeed, we used them to define proof systems which allow us to *statically prove* that a process is secure, i.e., by just inspecting its syntax [7,10]. These systems offer a means to build processes which are secure by construction, in an incremental way. They extend the one given in [24] for finite processes, i.e.,

processes that may only perform finite sequences of actions. In particular, we are able to deal also with recursive processes which may perform unbounded sequences of actions. Here we provide a general scheme for the construction of correct proof rules for unwinding classes of processes which generalize the proof rules proposed in [7,10].

Theorem 4.16 *Let $\mathcal{W}(\sim^l, \dashrightarrow)$ be an unwinding class of processes. Let Sys be a proof system whose rules are of the following form*

$$\frac{E_1, \ldots, E_k \in \mathcal{W}(\sim^l, \dashrightarrow)}{f(E_1, \ldots, E_k) \in \mathcal{W}(\sim^l, \dashrightarrow)}$$

where $\mathcal{W}(\sim^l, \dashrightarrow)$ is compositional with respect to f, or the rule

$$\frac{E \in \mathcal{E}_L}{E \in \mathcal{W}(\sim^l, \dashrightarrow)}$$

Then Sys is correct, i.e., if there exists a derivation of $E \in \mathcal{W}(\sim^l, \dashrightarrow)$ in Sys, then $E \in \mathcal{W}(\sim^l, \dashrightarrow)$.

By Theorem 4.16 and Corollary 4.11 we get for instance the following rule

$$\frac{E_1, E_2 \in P\_BNDC}{E_1 | E_2 \in P\_BNDC}$$

However, by considering the proof system obtained exploiting only to the operators in Corollary 4.11 we can only prove that the processes in $\mathcal{E}_L$ are $P\_BNDC, SBNDC, CP\_BNDC, PP\_BNDC$. In fact, we have no way to introduce high level actions. In the case of P_BNDC we have that P_BNDC is compositional with respect to the functions of the form $f : \mathcal{E}^{p+q} \longrightarrow \mathcal{E}$ defined as

$$f(F_1, \ldots, F_p, G_1, \ldots, G_q) = \sum_{1 \le i \le p} l_i.F_i + \sum_{1 \le j \le q} (h_j.G_j + \tau.G_j)$$

where $l_i \in L$ for all $i = 1, \ldots, p$ and $h_j \in H$ for all $j = 1, \ldots, q$ (see also Theorem 5.2). Hence we can add to the proof system the rules of the form

$$\frac{F_1, \ldots, F_k, G_1, \ldots, G_h \in P\_BNDC}{\sum_{1 \le i \le p} l_i.F_i + \sum_{1 \le j \le q} (h_j.G_j + \tau.G_j) \in P\_BNDC}$$

which allows use to build secure processes not in $\mathcal{E}_L$. These rules can be used also in the cases of CP_BNDC and PP_BNDC, while in the case of $SBNDC$

we can prove the correctness of the rule

$$\frac{E \in SBNDC}{E + h.E \in SBNDC}$$

5 How to Rectify Insecure Processes

In [6] we propose a general method for rectifying non *P_BNDC* processes. The idea is to automatically transform a process E into a *P_BNDC* process E^τ and to identify a large class of processes for which the transformation preserves the low level observational semantics, i.e., for the low level user E and E^τ are not distinguishable. This transformation can be used to construct "secure" processes from a first possibly "insecure" definition. Here we generalize the transformation presented in [6] to deal with any unwinding class of processes and sequences s of actions. Given a process E and a sequence of actions $s = s_1 \dots s_n \in Act^+$, we denote by $s.E$ the process $s_1.\dots.s_n.E$.

Definition 5.1 [E^s] Let $\mathcal{W}(\sim^l, \dashrightarrow)$ be an unwinding class of processes compositional with respect to the v-restriction, for each $v \subseteq \mathcal{L}^*$, the g-renaming, for each renaming g, the parallel composition operator $|$, the constant definition $\stackrel{\text{def}}{=}$, the replication operator !. Given a guarded process E and $s \in Act^+$ with $n > 0$ we inductively define E^s as follows:

$$\mathbf{0}^s = \mathbf{0} \qquad (E \setminus v)^s = E^s \setminus v \quad (E[g])^s = E^s[g]$$

$$(E_1|E_2)^s = E_1^s|E_2^s \qquad Z^s \stackrel{\text{def}}{=} F^s \qquad !E^s = !(E^s)$$

$$(\textstyle\sum_i l_i.F_i + \sum_j h_j.G_j)^s = \sum_i l_i.F_i^s + \sum_j (h_j.G_j^s + s.G_j^s)$$

where $l_i \in L \cup \{\tau\}$, $h_j \in H$, and Z was associated to $Z \stackrel{\text{def}}{=} F$.

Theorem 5.2 (Rectification^s) *Let $\mathcal{W}(\sim^l, \dashrightarrow)$ be an unwinding class of processes compositional with respect to the v-restriction, for each $v \subseteq \mathcal{L}^*$, the g-renaming, for each renaming g, the parallel composition operator $|$, the constant definition $\stackrel{\text{def}}{=}$, the replication operator !. Let $E \in \mathcal{E}$ be a guarded process. If $s \in Act^+$ is a sequence of actions and such that $E \stackrel{s}{\rightarrow} F$ implies $E \dashrightarrow F$, then*

$$E^s \in \mathcal{W}(\sim^l, \dashrightarrow).$$

Proof. By induction on E. If $E \equiv \mathbf{0}$, then $E^s \equiv \mathbf{0} \in \mathcal{W}(\sim^l, \dashrightarrow)$.

If $E \equiv E_1 \setminus v$, then by inductive hypothesis on E_1, $E_1^s \in \mathcal{W}(\sim^l, \dashrightarrow)$, hence, since $\mathcal{W}(\sim^l, \dashrightarrow)$ is compositional with respect to the v-restriction, we get the thesis. The cases of renaming, parallel composition, constant definition, and replication are similar.

If $E \equiv \sum_{l_i \in L \cup \{\tau\}} l_i.F_i + \sum_{h_j \in H} h_j.G_j$ and E^s reaches E' two cases are possible:

- E' is E^s;
- one of the F_i^s, G_j^s's reaches E'.

In the first case if $E^s \xrightarrow{h} E''$ we have that there exists j such that $E'' \equiv G_j$. Hence, $E^s \dashrightarrow G_j$ and $G_j \sim^l G_j$, since $\sim^l$ is an equivalence relation.
In the second case the thesis follows by inductive hypothesis on the F_i^s, G_j^s's.□
□

Corollary 5.3 *Let $E \in \mathcal{E}$ be a guarded process.*

$$E^\tau \in P\_BNDC, CP\_BNDC, PP\_BNDC.$$

Example 5.4 The memory cell M_x presented in Example 3.2 was not secure. We transformed it into two memory cells, a high level one and a low level one. Since the low level user cannot read from the high memory cell and the high level user cannot write on the low memory cell we obtain that the two memory cell are secure. Imagine now that we want to model the low level memory cell in such a way that each value can be read at most once. At the beginning the cell Q^l_e is empty, when a low level user writes a value x the cell is moved in the state Q^l_x in which it remains until either a high or a low level user read the value. After a reading the cell is reset in the state Q^l_e.

$$Q^l\_e \stackrel{\text{def}}{=} w_l\_x \;.\; Q^l\_x$$
$$Q^l\_x \stackrel{\text{def}}{=} \overline{r_l\_x} \;.\; Q^l\_e + \overline{r_h\_x} \;.\; Q^l\_e$$

In particular, with this implementation each value is read exactly once. However, Q^l_e is not *P_BNDC*. In fact, if any user reads the value the low level user cannot write a new value, i.e., the system is blocked. Applying to Q^l_e our rectification we get

$$Q^l\_e^\tau \stackrel{\text{def}}{=} w_l\_x \;.\; Q^l\_x^\tau$$
$$Q^l\_x^\tau \stackrel{\text{def}}{=} \overline{r_l\_x} \;.\; Q^l\_e^\tau + \overline{r_h\_x} \;.\; Q^l\_e^\tau + \tau.Q^l\_e^\tau$$

In this case the rectification corresponds to the modelling of a timeout: if the value is not read within a certain amount of time, the system reset the cell. Now each value is read at most once.

The LTS's of $Q^l\_e^\tau$ is depicted in Figure 7.

The above theorem does not requires the compositionality with respect to the non-deterministic choice operator. As a consequence the correction can be applied only to guarded processes. In the case we deal with a fully compositional unwinding class we can extend the correction to non guarded

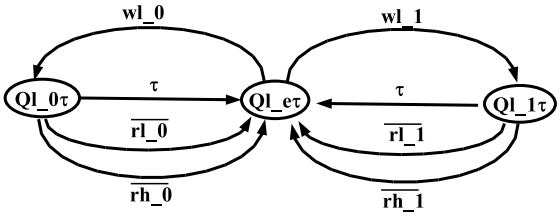

Fig. 7. The LTS of the memory cell $Q^l\_e^\tau$.

processes.

Definition 5.5 [E_s] Let $\mathcal{W}(\sim^l, \dashrightarrow)$ be an unwinding class of processes compositional with respect to the v-restriction, for each $v \subseteq \mathcal{L}^*$, the g-renaming, for each renaming g, the parallel composition operator $|$, the constant definition $\stackrel{\text{def}}{=}$, the replication operator ! and the nondeterministic choice operator $+$. Given a process E and $s \in Act^+$ we inductively define E_s as follows:

$$\mathbf{0}_s = \mathbf{0} \qquad (l.E)_s = l.E_s \qquad (h.E)_s = h.E_s + s.E_s$$
$$(E \setminus v)_s = E_s \setminus v \quad (E[g])_s = E_s[g] \qquad (E_1|E_2)_s = (E_1)_s|(E_2)_s$$
$$Z_s \stackrel{\text{def}}{=} F_s \qquad !E_s = !(E_s) \qquad (E_1 + E_2)_s = (E_1)_s + (E_2)_s$$

where $l \in L \cup \{\tau\}$, $h \in H$, and Z was associated to $Z \stackrel{\text{def}}{=} F$.

Theorem 5.6 (Rectification$_s$) *Let $\mathcal{W}(\sim^l, \dashrightarrow)$ be an unwinding class of processes compositional with respect to the v-restriction, for each $v \subseteq \mathcal{L}^*$, the g-renaming, for each renaming g, the parallel composition operator $|$, the constant definition $\stackrel{\text{def}}{=}$, the replication operator !, and the nondeterministic choice operator $+$. Let $E \in \mathcal{E}$ be a process. If $s \in Act^+$ is a sequence of actions and such that $E \stackrel{s}{\rightarrow} F$ implies $E \dashrightarrow F$, then*

$$E_s \in \mathcal{W}(\sim^l, \dashrightarrow).$$

Proof. The result can be proved by induction on E exploiting the fact that all the unwinding classes are compositional with respect to the low prefix operator. □
□

Corollary 5.7 *Let $E \in \mathcal{E}$ be a process.*

$$E_\tau \in CP\_BNDC, PP\_BNDC.$$

6 Conclusions

In this paper we consider information flow security properties of SPA processes expressed in terms of unwinding conditions. The aim of the present work is to bridge the gap between unwinding conditions and compositionality results. This is done by exploiting a generalized unwinding condition $\mathcal{W}(\sim^l, \dashrightarrow)$, parametric with respect to a low level behavioral equivalence $\sim^l$ and a transition relation $\dashrightarrow$. To prove the compositionality of a class of secure processes, expressed as an instance of $\mathcal{W}(\sim^l, \dashrightarrow)$, with respect to an operator f we need to establish connections between the semantics of f and the relations $\sim^l$ and $\dashrightarrow$. By instantiating f as one of the algebra operators we rediscover already proved compositionality results (e.g., the compositionality of *P_BNDC* with respect to the parallel operator). Moreover, by instantiating f as a *refinement* operator, which solves the non-deterministic choices, we obtain results concerning the preservation of the security properties under refinement. Unwinding conditions can be also exploited for defining proof systems which provide efficient techniques for the verification and the development of secure processes. Proof systems allow us to verify whether a process is secure just by inspecting its syntax, and thus avoiding the state-explosion problem. Moreover, they also allow us to build processes which are secure by construction in an incremental way. Finally compositionality of unwinding conditions can be easily exploited to rectify insecure processes.

References

[1] Martín Abadi. Secrecy by typing in security protocols. *Journal of the ACM*, 46(5):749–786, 1999.

[2] A. Aldini, M. Bravetti, and R. Gorrieri. A Process-algebraic Approach for the Analysis of Probabilistic Non-interference. *Journal of Computer Security*, 2004. To appear.

[3] D. E. Bell and L. J. La Padula. Secure computer systems: Unified exposition and multics interpretation. Technical Report ESD-TR-75-306, MITRE MTR-2997, 1975.

[4] V. Benzaken, M. Burelle, and G. Castagna. Information flow security for xml transformations. In *Proc. of Asian Computing Science Conference (ASIAN'03)*, LNCS. Springer-Verlag, 2003. To appear.

[5] C. Bodei, P. Degano, F. Nielson, and H. Riis Nielson. Static analysis for secrecy and non-interference in networks of processes. In Victor E. Malyshkin, editor, *Proc. of International Conference on Parallel Computing Technologies*, volume 2127 of *Lecture Notes in Computer Science*, pages 27–41. Springer-Verlag, 2001.

[6] A. Bossi, R. Focardi, C. Piazza, and S. Rossi. Transforming Processes to Ensure and Check Information Flow Security. In H. Kirchner and C. Ringeissen, editors, *Int. Conference on Algebraic Methodology and Software Technology (AMAST'02)*, volume 2422 of *LNCS*, pages 271–286. Springer-Verlag, 2002.

[7] A. Bossi, R. Focardi, C. Piazza, and S. Rossi. A Proof System for Information Flow Security. In M. Leuschel, editor, *Logic Based Program Development and Transformation*, volume 2664 of *LNCS*, pages 199–218. Springer-Verlag, 2003.

[8] 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.

[9] A. Bossi, R. Focardi, C. Piazza, and S. Rossi. Verifying Persistent Security Properties. *Computer Languages, Systems and Structures*, 2003. To appear. Available at `http://www.dsi.unive.it/~srossi/cl03.ps.gz`.

[10] A. Bossi, D. Macedonio, C. Piazza, and S. Rossi. Information Flow Security and Recursive Systems. In *Proc. of the Italian Conference on Theoretical Computer Science (ICTCS'03)*, volume 2841 of *LNCS*, pages 369–382. Springer-Verlag, 2003.

[11] M. Bugliesi, S. Crafa, M. Merro, and V. Sassone. Communication Interference in Mobile Boxed Ambients. In M. Agrawal and A. Seth, editors, *Proc. of Int. Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'02)*, volume 2556 of *LNCS*, pages 71–84. Springer-Verlag, 2002.

[12] A. Di Pierro, C. Hankin, and H.Wiklicky. Approximate Non-Interference. In *Proc. of the IEEE Computer Security Foundations Workshop (CSFW'02)*, pages 3–17. IEEE Computer Society Press, 2002.

[13] R. Focardi and R. Gorrieri. A Classification of Security Properties for Process Algebras. *Journal of Computer Security*, 3(1):5–33, 1994/1995.

[14] 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.

[15] R. Focardi, R. Gorrieri, and F. Martinelli. Information flow analysis in a discrete-time process algebra. In *Proc. of the IEEE Computer Security Foundations Workshop (CSFW'00)*, pages 170–184. IEEE Computer Society Press, 2000.

[16] R. Focardi, R. Gorrieri, and F. Martinelli. Non Interference for the Analysis of Cryptographic Protocols. In U. Montanari, J. D. P. Rolim, and E. Welzl, editors, *Proc. of Int. Colloquium on Automata, Languages and Programming (ICALP'00)*, volume 1853 of *LNCS*, pages 744–755. Springer-Verlag, 2000.

[17] 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.

[18] R. Forster. *Non-Interference Properties for Nondeterministic Processes*. PhD thesis, Oxford University Computing Laboratory, 1999.

[19] 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.

[20] 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.

[21] R. Gorrieri, E. Locatelli, and F. Martinelli. A simple language for real-time cryptographic protocol analysis. In P. Degano, editor, *Proc. of European Symposium on Programming (ESOP'03)*, volume 2618 of *LNCS*, pages 114–128. Springer-Verlag, 2003.

[22] M. Hennessy and J. Riely. Information Flow vs. Resource Access in the Asynchronous Pi-calculus. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 24(5):566–591, 2002.

[23] H. Mantel. Unwinding Possibilistic Security Properties. In *Proc. of the European Symposium on Research in Computer Security (ESoRiCS'00)*, volume 2895 of *LNCS*, pages 238–254. Springer-Verlag, 2000.

[24] F. Martinelli. Partial Model Checking and Theorem Proving for Ensuring Security Properties. In *Proc. of the IEEE Computer Security Foundations Workshop (CSFW'98)*, pages 44–52. IEEE Computer Society Press, 1998.

[25] J. McLean. Security Models and Information Flow. In *Proc. of the IEEE Symposium on Security and Privacy (SSP'90)*, pages 180–187. IEEE Computer Society Press, 1990.

[26] J. McLean. Security Models. *Encyclopedia of Software Engineering*, 1994.

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

[28] U. Montanari and V. Sassone. CCS Dynamic Bisimulation is Progressing. In *Proc. of the Int. Symposium on Mathematical Foundations of Computer Science (MFCS'91)*, volume 520 of *LNCS*, pages 346–356. Springer-Verlag, 1991.

[29] 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.

[30] A. W. Roscoe, J. C. P. Woodcock, and L. Wulf. Non-Interference through Determinism. *Journal of Computer Security*, 4(1), 1996.

[31] P. Y. A. Ryan. A CSP Formulation of Non-Interference and Unwinding. *Cipher*, pages 19–27, 1991.

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

[33] A. Sabelfeld and A. C. Myers. Language-Based Information-Flow Security. *IEEE Journal on Selected Areas in Communication*, 21(1):5–19, 2003.

[34] A. Sabelfeld and D. Sands. Probabilistic Noninterference for Multi-threaded Programs. In *Proc. of the IEEE Computer Security Foundations Workshop (CSFW'00)*, pages 200–215. IEEE Computer Society Press, 2000.

[35] G. Smith and D. M. Volpano. Secure Information Flow in a Multi-threaded Imperative Language. In *Proc. of ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'98)*, pages 355–364. ACM Press, 1998.

[36] D. Sutherland. A Model of Information. In *Proc. of the 9th National Computer Security Conference*, pages 175–183, 1986.

[37] 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.