Transforming processes to check and ensure Information Flow Security[⋆]

Annalisa Bossi, Riccardo Focardi, Carla Piazza, and Sabina Rossi

Dipartimento di Informatica, Università Ca' Foscari di Venezia
{bossi,focardi,piazza,srossi}@dsi.unive.it

Abstract. *Persistent_BNDC* (*P_BNDC* for short) is an information-flow security property for processes in dynamic contexts, i.e., contexts that can be reconfigured at runtime. We propose a method for transforming an arbitrary process into a process satisfying *P_BNDC* and show that the transformation preserves the "low level" observational semantics for a large class of processes. We also study how to efficiently verify *P_BNDC* by exploiting a characterization of it through a suitable notion of weak bisimulation *up to high level actions*. We define a second transformation over processes which allows us to reduce the problem of checking *P_BNDC* to the problem of testing a weak bisimulation between two processes. This approach is particularly appealing as it allows us to perform the *P_BNDC* check using already existing tools at a low time complexity.

1 Introduction

Systems are becoming more and more complex, and the security community has to face this by taking into account new threats and potentially dangerous situations. A significant example is the introduction of *process mobility* among different architectures and systems. A mobile process moving on the network collects information about the environments it crosses, and such information can influence it. A system or an application executing in a "secure way" inside one environment could find itself in a "insecure state" when moving to a different environment. In this setting, one can abstractly think that the environment is dynamically reconfigured at run-time, changing in unpredictable ways.

A number of formal definitions of security properties (see, for instance, [1, 3, 5, 7, 13–15, 17, 20–22]) has been proposed in the literature. *Persistent_BNDC* (*P_BNDC*, for short), proposed in [10], is a security property which is suitable to analyze processes in completely dynamic hostile environments, i.e., environments which can be dynamically reconfigured at run-time, changing in unpredictable ways. The notion of *P_BNDC* is based on the idea of Non-Interference [11, 19, 22] (formalized as *BNDC* [7]) and requires that every state which is reachable by the system still satisfies a basic Non-Interference property. If this holds, one is

[⋆] This work has been partially supported by the MURST project "Modelli formali per la sicurezza" and the EU Contract IST-2001-32617 "Models and Types for Security in Mobile Distributed Systems" (MyThS).

assured that even if the environment changes during the execution no malicious attacker will be able to compromise the system, as every possible reachable state is guaranteed to be secure. In [10] it is proved that the *P_BNDC* property is equivalent to an already proposed security property called *SBSNNI* and studied in [7]. In particular, the property *SBSNNI* is compared with different properties in the taxonomy of Non-Interference properties [11]. From the analysis presented in [7] two important problems emerge: how to verify the *P_BNDC* property and how to construct *P_BNDC* processes. The first problem has been considered in [10] where it has been shown to be decidable, and in [9] where efficiency issues have also been tackled. To the best of our knowledge the second problem has not been analyzed yet. In [7] there are many examples of processes which are not *P_BNDC* but can be modified in order to obtain a *P_BNDC* process. However each single example is treated in a different way by applying each time an "ad hoc" re-definition.

Our purpose here is to find a general method for rectifying non *P_BNDC* processes. It turns out that the method we suggest can be used both to rectify and to efficiently verify the *P_BNDC* property. We automatically transform a process E into a *P_BNDC* process E^{τ} and 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. Moreover, this also allows us to give an alternative characterization of *P_BNDC* through a suitable notion of weak bisimulation *up to high level actions* [10]. More precisely, we obtain that a process E is *P_BNDC* if and only if E and E^{τ} are weak bisimilar up to high level actions. The problem of verifying whether a process is *P_BNDC* is then reduced to the problem of checking whether E and E^{τ} are weak bisimilar up to high level actions. We show that this problem can be further simplified reducing it to the problem of checking the more usual notion of weak bisimulation between two processes. In particular we define a second transformation over processes such that the problem of checking whether a process is *P_BNDC* is reduced to the problem of testing a weak bisimulation relation. This approach seems to be particularly appealing as it allows us to perform the *P_BNDC* check using already existing tools at a low time complexity.

The paper is organized as follows. In Section 2 we present some basic notions on the *SPA* language, we introduce the *P_BNDC* property and we recall its characterization in terms of weak bisimulation up to high level actions. In Section 3 we define our first transformation and prove its main properties. In Section 4 we introduce a second transformation and show how to use both our transformations to check *P_BNDC*. In Section 5 we illustrate the usefulness of our transformations on a simple example. Finally, in Section 6 we draw some conclusions.

2 Basic Notions

In this section we report from [7] the syntax and semantics of the *Security Process Algebra* together with the definition of the Non-Interference property called

BNDC. We then report from [10] the definition of *Persistent_BNDC* together with the main result that we will exploit for the verification of such a property.

The SPA Language. The *Security Process Algebra* (SPA, for short) [7] is a variation of Milner's CCS [16], where the set of visible actions is partitioned into high level actions and low level ones in order to specify multilevel systems. SPA syntax 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 $I = \{a, b, \dots\}$ is a set of *input* actions and $O = \{\bar{a}, \bar{b}, \dots\}$ 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 $a \in \mathcal{L}$, and $\bar{\tau} = \tau$; $Act = \mathcal{L} \cup \{\tau\}$ 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* (or *processes*) is defined as follows:

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

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

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 non-deterministic 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; $E \backslash v$ 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. For the definition of security properties it is also useful the *hiding* operator, $/$, of CSP which can be defined as a relabelling as follows: for a given set $v \subseteq \mathcal{L}$, $E/v \overset{\text{def}}{=} E[f_v]$ where $f_v(x) = x$ if $x \notin v$ and $f_v(x) = \tau$ if $x \in v$. In practice, E/v turns all actions in v into internal τ's.

Given a fixed language $\mathcal{L}$ 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 on $H \cup \{\tau\}$.

The operational semantics of SPA agents 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 $S_1 \xrightarrow{a} S_2$) 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 $(\mathcal{E}, Act, \to)$, 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 inference rules reported in Fig. 1. The operational semantics for an agent E is the subpart of the SPA LTS reachable from the initial state and we refer to it as $LTS(E) = (S_E, Act, \to)$. A process E is said to be *finite-state* if S_E is finite.

In [16] it is shown that a finite-state process E can always be defined through a system S of equations of the form

$$E_j = a_1.E_1 + \dots + a_n.E_n,$$

$$\text{Prefix} \quad \frac{}{a.E \xrightarrow{a} E}$$

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

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

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

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

$$\text{Constant} \quad \frac{E \xrightarrow{a} E'}{A \xrightarrow{a} E'} \quad \text{if } A \overset{\text{def}}{=} E$$

Fig. 1. The operational rules for SPA

such that $E_1, \ldots, E_n \in S_E$ and there is one equation in S for each $E_j \in S_E$.

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 $\mathcal{E}$. In the following, we report the definition of an observation equivalence called *weak bisimulation* [16]. Intuitively, weak bisimulation equates two processes if they are able to mutually simulate their behavior step by step. Weak bisimulation does not care about internal τ actions. So, when F simulates an action of E, it can also execute some τ actions before or after that action.

We will use the following auxiliary notations. 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 $E \overset{t}{\Longrightarrow} E'$ 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, $E \overset{\hat{a}}{\Longrightarrow} E'$ stands for $E \overset{a}{\Longrightarrow} E'$ if $a \in \mathcal{L}$, and for $E(\xrightarrow{\tau})^* E'$ if $a = \tau$ (note that $\overset{\tau}{\Longrightarrow}$ requires at least one τ labelled transition while $\overset{\hat{\tau}}{\Longrightarrow}$ means zero or more τ labelled transitions).

Definition 1 (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 $F \overset{\hat{a}}{\Longrightarrow} F'$ and $(E', F') \in \mathcal{R}$;*
- *if $F \xrightarrow{a} F'$, then there exists E' such that $E \overset{\hat{a}}{\Longrightarrow} E'$ and $(E', F') \in \mathcal{R}$.*

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

$\approx$ is the largest weak bisimulation and an equivalence relation (see [16]).

Security Properties. In this section, we recall from [10] the *Persistent_BNDC* (*P_BNDC*, for short) security property and its characterization in terms of weak bisimulation up to high level actions. We start by recalling the definition of *Bisimulation-based Non Deducibility on Compositions* (*BNDC*, for short) [7]. The *BNDC* security property aims at guaranteeing that no information flow from the high to the low level is possible, even in the presence of malicious processes. The main motivation is to protect a system also from internal attacks, which could be performed by the so called *Trojan Horse* programs, i.e., programs that pretend/appear to be honest but incorporate some malicious code.

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 [11] with the low level execution of the system E.

Definition 2 (BNDC). *Let $E \in \mathcal{E}$.*

$$E \in BNDC \;\; \textit{iff} \;\; \forall \; \Pi \in \mathcal{E}_H, \; E \backslash H \approx (E|\Pi)\backslash H.$$

In [10] it is shown that the *BNDC* property is not strong enough to analyse systems in dynamic execution environments. To deal with these situations, in [10] it has been introduced the security property named *P_BNDC*. Intuitively, a system E is *P_BNDC* if it never reaches insecure states.

Definition 3 (Persistent_BNDC). *Let $E \in \mathcal{E}$.*

$$E \in P\_BNDC \;\; \textit{iff} \;\; \forall \; E' \; \textit{reachable from} \; E, \; E' \in BNDC.$$

We give a simple example of a *P_BNDC* process. A more expressive example can be found in [10].

Example 1. Consider the process $E_1 = l.h.j.\mathbf{0} + l.(\tau.j.\mathbf{0} + \tau.\mathbf{0})$ where $l, j \in L$ and $h \in H$. E_1 can be proved to be *BNDC*. Indeed, the causality between h and j in the first branch of the process is "hidden" by the second branch $l.(\tau.j.\mathbf{0} + \tau.\mathbf{0})$, which may simulate all the possible interactions with a high level process. Suppose now that E_1 is moved in the middle of a computation. This might happen when it find itself in the state $h.j.\mathbf{0}$ (after the first l is executed). Now it is clear that this process is not secure, as a direct causality between h and j is present. In particular $h.j.\mathbf{0}$ is not *BNDC* and this gives evidence that E_1 is not *P_BNDC*. The process may be "repaired" as follows: $E_2 = l.(h.j.\mathbf{0}+\tau.j.\mathbf{0}+\tau.\mathbf{0})+l.(\tau.j.\mathbf{0}+\tau.\mathbf{0})$. It may be proved that E_2 is *P_BNDC*. Note that, from this example it follows that *P_BNDC* $\subset$ *BNDC*.

In [10] it has been proven that the property *P_BNDC* is equivalent to the security property *SBSNNI* [6, 7], which is automatically checkable over finite-state processes.

However, this property still requires a universal quantification over all the possible states that are reachable from the initial process E. In [10] it has been shown that this can be avoided, by including the requirement of "being secure in every state" directly inside the bisimulation equivalence notion.

In particular, an observation equivalence, named *weak bisimulation up to* H, is defined in such a way that actions from H may be ignored, i.e., they may be matched with zero or more τ actions. This bisimulation notion is based on a suitable transition relation $\overset{\hat{a}}{\Longrightarrow}_{\backslash H}$ which does not take care of both internal actions and actions from H.

Definition 4. *Let $E, E' \in \mathcal{E}$ and $a \in Act$. We define the transition relation $\overset{\hat{a}}{\Longrightarrow}_{\backslash H}$ as follows:*

$$E \overset{\hat{a}}{\Longrightarrow}_{\backslash H} E' = \begin{cases} E \overset{\hat{a}}{\Longrightarrow} E' & \text{if } a \notin H \\ E \overset{a}{\Longrightarrow} E' \text{ or } E \overset{\hat{\tau}}{\Longrightarrow} E' & \text{if } a \in H \end{cases}$$

Note that the relation $\overset{\hat{a}}{\Longrightarrow}_{\backslash H}$ is a generalization of the relation $\overset{\hat{a}}{\Longrightarrow}$ used in the definition of weak bisimulation [16]. In fact, if $H = \emptyset$, then for all $a \in Act$, $E \overset{\hat{a}}{\Longrightarrow}_{\backslash H} E'$ coincides with $E \overset{\hat{a}}{\Longrightarrow} E'$.

The concept of weak bisimulation up to H is defined as follows.

Definition 5 (Weak Bisimulation up to H). *A binary relation $\mathcal{R} \subseteq \mathcal{E} \times \mathcal{E}$ over agents is a* weak bisimulation up to H *if $(E, F) \in \mathcal{R}$ implies, for all $a \in Act$,*

(1) *if $E \xrightarrow{a} E'$, then there exists F' such that $F \overset{\hat{a}}{\Longrightarrow}_{\backslash H} F'$ and $(E', F') \in \mathcal{R}$;*

(2) *if $F \xrightarrow{a} F'$, then there exists E' such that $E \overset{\hat{a}}{\Longrightarrow}_{\backslash H} E'$ and $(E', F') \in \mathcal{R}$.*

Two agents $E, F \in \mathcal{E}$ are weakly bisimilar up to H, *written $E \approx_{\backslash H} F$, if $(E, F) \in \mathcal{R}$ for some weak bisimulation $\mathcal{R}$ up to H.*

The relation $\approx_{\backslash H}$ is the largest weak bisimulation up to H and it is an equivalence relation.

In [10] it has been proved that *P_BNDC* can be characterized in terms of $\approx_{\backslash H}$ as follows.

Theorem 1. *Let $E \in \mathcal{E}$. Then, $E \in P\_BNDC$ iff $E \approx_{\backslash H} E \backslash H$.*

3 Defining *P_BNDC* Processes

In this section we define a transformation on processes which maps an arbitrary process into a *P_BNDC* one. Moreover, we show that a process is *P_BNDC* iff it is weak bisimilar up to H to its transformed version. In order to prove this second result we exploit some basic properties of weak bisimulation up to H which are introduced in Section 3.1.

3.1 Basic Properties of $\approx_{\backslash H}$

We start by proving some properties of the relation $\approx_{\backslash H}$. Actually the relation $\approx_{\backslash H}$ enjoys the majority of the properties of the standard weak bisimulation.

First of all $\approx_{\backslash H}$ is coarser than $\approx$.

Lemma 1. *If $E \approx F$, then $E \approx_{\backslash H} F$.*

Proof. Let $\mathcal{S} = \{(E, F) \mid E \approx F\}$. The binary relation $\mathcal{S}$ is a weak bisimulation up to H since for all processes E it holds that if $E \stackrel{\hat{a}}{\Longrightarrow} E'$, then $E \stackrel{\hat{a}}{\Longrightarrow}_{\backslash H} E'$. □

The relation $\approx_{\backslash H}$ is compositional with respect to the restriction on high level actions, as stated by the following lemma.

Lemma 2. *If $E \approx_{\backslash H} F$, then $E\backslash H \approx_{\backslash H} F\backslash H$.*

Proof. Let $\mathcal{S} = \{(E\backslash H, F\backslash H) \mid E \approx_{\backslash H} F\}$. It is easy to prove that $\mathcal{S}$ is a weak bisimulation up to H. □

The *P_BNDC* class of processes is closed with respect to the equivalence relation of $\approx_{\backslash H}$.

Lemma 3. *Let $E, F \in \mathcal{E}$. If $E \approx_{\backslash H} F$ and $F \in P\_BNDC$, then $E \in P\_BNDC$.*

Proof. If $E \approx_{\backslash H} F$, then we obtain

$$\begin{aligned} E &\approx_{\backslash H} F \\ &\approx_{\backslash H} F\backslash H \quad \text{by Theorem 1} \\ &\approx_{\backslash H} E\backslash H \quad \text{by Lemma 2} \end{aligned}$$

Hence, since $E \approx_{\backslash H} E\backslash H$, by Theorem 1 we obtain that E is *P_BNDC*. □

By Lemma 1 and Lemma 3 it immediately follows that if $E \approx F$ and $F \in P\_BNDC$, then $E \in P\_BNDC$.

Another useful property of *P_BNDC* processes is that restriction and hiding with respect to high level actions yield weakly bisimilar processes.

Lemma 4. *If $E \in P\_BNDC$, then $E\backslash H \approx E/H$.*

Proof. This is a consequence of the fact that *P_BNDC* is equivalent to the *SBSNNI* property [10] and *SBSNNI* implies that $E\backslash H \approx E/H$ [6,7]. □

3.2 The τ completion of a process

Now we are ready to define our first transformation over finite-state processes which maps a process E into a *P_BNDC* process E^τ.

Definition 6 (τ completion of E). *Let $E \in \mathcal{E}$ be one of the processes defined by a system of equations S. The τ* completion of E *is the process E^τ defined by the system S^τ, where:*

– *if $F = \mathbf{0}$ is in S, then $F^\tau = \mathbf{0}$ is in S^τ;*
– *if $F = \sum_{i\in I} l_i.F_i + \sum_{j\in J} h_j.F_j$ is in S, with $l_i \in L \cup \{\tau\}$ and $h_j \in H$, then $F^\tau = \sum_{i\in I} l_i.F_i^\tau + \sum_{j\in J} h_j.F_j^\tau + \sum_{j\in J} \tau.F_j^\tau$ is in S^τ.*

In practice, the LTS associated to E^τ can be obtained from the LTS of E by simply adding a τ edge whenever there is a transition with a label in H.

Example 2. Consider the process E defined by the following system S:

$$\begin{cases} E = h.F + l_1.\mathbf{0} \\ F = l_2.E \end{cases}$$

Using the above definition we obtain the process E^τ defined by

$$\begin{cases} E^\tau = h.F^\tau + \tau.F^\tau + l_1.\mathbf{0} \\ F^\tau = l_2.E^\tau \end{cases}$$

The two LTS's for E and E^τ are depicted in Fig. 2. Different edges between the same nodes are represented in a compact way with a single arc labeled by a sequence of actions separated by commas.

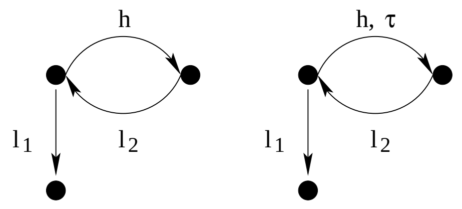

Fig. 2. The LTS's representing E and E^τ

The following lemma formalizes the relations between E and E^τ, which are at the basis of all the results in this section.

Lemma 5. *Let $E \in \mathcal{E}$.*

1. *if $E^\tau \xrightarrow{a} E'$ with $a \neq \tau$, then there exists E_1 such that $E' = E_1^\tau$ and $E \xrightarrow{a} E_1$;*
2. *if $E^\tau \xrightarrow{\tau} E'$, then there exists E_1 such that $E' = E_1^\tau$ and $E \xrightarrow{k} E_1$ with $k \in H \cup \{\tau\}$;*
3. *if $E \xrightarrow{a} E_1$ with $a \in H \cup L \cup \{\tau\}$, then $E^\tau \xrightarrow{a} E_1^\tau$;*
4. *if $E^\tau \xrightarrow{h} E'$ with $h \in H$, then $E^\tau \xrightarrow{\tau} E'$.*

Proof. It immediately follows by Definition 6. □

The difference between E and E^τ is that whenever E can perform a high action, E^τ can silently simulate the same reduction, thus hiding the high level actions to the low level user. We prove that E^τ so defined is *P_BNDC*.

Theorem 2. *For any process $E \in \mathcal{E}$, $E^\tau \in P\_BNDC$.*

Proof. Let $\mathcal{S} = \{(E^\tau, E^\tau\backslash H) \mid E \in \mathcal{E}\}$. It is easy to prove that $\mathcal{S}$ is a weak bisimulation up to H. The result follows by Theorem 1. $\square$

The following lemma shows that E and E^τ behave in the same way if we hide the high level actions.

Lemma 6. *For any process $E \in \mathcal{E}$ it holds that $E/H \approx E^\tau/H$.*

Proof. Let $\mathcal{S} = \{(E/H, E^\tau/H) \mid E \in \mathcal{E}\}$. By Lemma 5, it is easy to prove that $\mathcal{S}$ is a weak bisimulation. $\square$

The previous result is not sufficient to guarantee that the transformation preserves the semantics of the process, at least from the low level user point of view. In fact, what the low level user can observe are the restrictions $E\backslash H$ and $E^\tau\backslash H$. The following theorem identifies the class of processes for which the transformation preserves the low level semantics.

Theorem 3. *Let $E \in \mathcal{E}$. $E\backslash H \approx E^\tau\backslash H$ iff $E\backslash H \approx E/H$.*

Proof. If $E\backslash H \approx E^\tau\backslash H$, then

$$\begin{aligned} E\backslash H &\approx E^\tau\backslash H && \text{by hypothesis} \\ &\approx E^\tau/H && \text{by Theorem 2 and Lemma 4} \\ &\approx E/H && \text{by Lemma 6.} \end{aligned}$$

If $E\backslash H \approx E/H$, then

$$\begin{aligned} E\backslash H &\approx E/H && \text{by hypothesis} \\ &\approx E^\tau/H && \text{by Lemma 6} \\ &\approx E^\tau\backslash H && \text{by Theorem 2 and Lemma 4.} \end{aligned}$$

$\square$

The processes satisfying $E\backslash H \approx E/H$ are studied in [7] and form the class of *BSNNI* processes. In [7] it is also shown that the class of *P_BNDC* processes (there called *SBSNNI*) is properly included in the class of *BSNNI* processes.

In a certain sense we have the feeling that E^τ is a straight completion of E in order to obtain a *P_BNDC* process, and that if E is *P_BNDC* then E must be *not too far from* (in strong connection with) E^τ. In the rest of this section we study which is this connection. First, we show that *P_BNDC* properly includes the class of processes which are weakly bisimilar to their τ completion and it is properly included in the class of processes whose H restriction is weak bisimilar to the restriction of their completion.

Proposition 1. *Let $E \in \mathcal{E}$. The following properties hold:*

(1) *if $E \approx E^\tau$ then $E \in P\_BNDC$;*
(2) *if $E \in P\_BNDC$ then $E\backslash H \approx E^\tau\backslash H$.*

Proof. (1) By Theorem 2 we have that E^τ is *P_BNDC*, hence by Lemma 3 we have the thesis. (2) This is a corollary of Theorem 3 and of Lemma 4. $\square$

Note that $E \in P\_BNDC$ does not imply $E \approx E^\tau$. As an example consider the process $E = h.\mathbf{0}$ which is *P_BNDC* but it is not weak bisimilar to $E^\tau = h.\mathbf{0}+\tau.\mathbf{0}$. Moreover, $E\backslash H \approx E^\tau\backslash H$ does not imply $E \in P\_BNDC$. This is a consequence of the fact that, by Theorem 3, $E\backslash H \approx E^\tau\backslash H$ is equivalent to the *BSNNI* property which, as already said, is weaker than *P_BNDC* (see [7]).

The following theorem shows that if we use the relation $\approx_{\backslash H}$ instead of $\approx$ we obtain the desired characterization of *P_BNDC* processes.

Theorem 4. *Let $E \in \mathcal{E}$. Then, $E \in P\_BNDC$ iff $E \approx_{\backslash H} E^\tau$.*

Proof. ($\Rightarrow$)

$$\begin{array}{rlll} E & \approx_{\backslash H} & E\backslash H & \text{by Theorem 1} \\ & \approx_{\backslash H} & E/H & \text{by Lemma 4} \\ & \approx_{\backslash H} & E^\tau/H & \text{by Lemma 6} \\ & \approx_{\backslash H} & E^\tau\backslash H & \text{by Lemma 4} \\ & \approx_{\backslash H} & E^\tau & \text{by Theorem 1.} \end{array}$$

($\Leftarrow$) By Theorem 2, we have that $E^\tau \in P\_BNDC$, hence, by Lemma 3, we obtain the thesis. $\square$

4 Checking *P_BNDC*

By Theorem 4, it follows that in order to check whether a process E is *P_BNDC* we can equivalently check whether $E \approx_{\backslash H} E^\tau$. The first question should be whether this test is decidable or not; then it is necessary to study the complexity of a decision algorithm. Instead of defining an ad hoc algorithm we prefer here to prove that it is possible to reduce the test of weak bisimilarity up to H to a test of weak bisimilarity[1]. Hence, we define a second transformation over processes which maps a process E into a process E^H in such a way that $E \approx_{\backslash H} F$ iff $E^H \approx F^H$. Since the transformation can be performed in linear time we obtain that the test of weak bisimilarity up to H is in the same complexity class of the test of weak bisimilarity.

Definition 7. *Let $E \in \mathcal{E}$ be one of the processes defined by a system of equations S. E^H is the process defined by the system S^H, where:*

- *if $F = \mathbf{0}$ is in S, then $F^H = \sum_{h\in H} h.F^H$ is in S^H;*
- *if $F = \sum_{i\in I} a_i.F_i + \sum_{j\in J} \tau.F_j$ is in S, with $a_i \neq \tau$, then $F^H = \sum_{i\in I} a_i.F_i^H + \sum_{j\in J} \tau.F_j^H + \sum_{j\in J, h\in H} h.F_j^H + \sum_{h\in H} h.F^H$ is in S^H.*

The LTS associated to E^H can be obtained from the LTS of E by adding all the possible H transitions to any τ transition and all the possible H self-loops.

[1] Note that weak bisimilarity is usually tested through strong bisimilation on transformed processes (see [4]).

Example 3. Let $H = \{h_1, h_2\}$ and consider the process E defined by the following system

$$\begin{cases} E = \tau.F \\ F = h_1.E \end{cases}$$

We have that E^H is process defined by the following system

$$\begin{cases} E^H = \tau.F^H + h_1.F^H + h_2.F^H + h_1.E^H + h_2.E^H \\ F^H = h_1.E^H + h_1.F^H + h_2.F^H \end{cases}$$

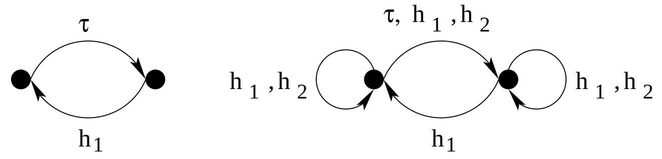

Fig. 3. The LTS's associated to E and E^H

Similarly to Lemma 5 in the previous section, the following lemma formalizes the relations between E and E^H. Its proof follows by Definitions 4 and 7.

Lemma 7. *Let $E \in \mathcal{E}$.*

(1) *if $E^H \xrightarrow{a} E'$ with $a \notin H$, then there exists E_1 such that $E' = E_1^H$ and $E \xrightarrow{a} E_1$;*
(2) *if $E^H \xrightarrow{h} E'$ with $h \in H$ then there exists E_1 such that $E' = E_1^H$ and $E \xrightarrow{k} E_1$ with $k \in \{h\} \cup \{\tau\}$;*
(3) *if $E^H \xrightarrow{\tau} E'$ then $E^H \xrightarrow{h} E'$ for all $h \in H$;*
(4) *if $E \xrightarrow{a} E_1$ then $E^H \xrightarrow{a} E_1^H$ for any action a;*
(5) *if $E \xrightarrow{\tau} E_1$ then $E^H \xrightarrow{h} E_1^H$ for all $h \in H$;*
(6) *if $E \overset{\hat{a}}{\Longrightarrow}_{\setminus H} E_1$ then $E^H \overset{\hat{a}}{\Longrightarrow} E_1^H$ for any action a;*
(7) *if $E \overset{\hat{\tau}}{\Longrightarrow}_{\setminus H} E_1$ then $E^H \overset{\hat{h}}{\Longrightarrow} E_1^H$ for all $h \in H$;*
(8) *if $E^H \overset{\hat{a}}{\Longrightarrow} E_1^H$ then $E \overset{\hat{a}}{\Longrightarrow}_{\setminus H} E_1$ for any action a.*

We are now ready to prove the main result of this section which shows that we can reduce the test of $\approx_{\setminus H}$ to a test of $\approx$.

Theorem 5. *Let $E, F \in \mathcal{E}$. Then, $E \approx_{\setminus H} F$ iff $E^H \approx F^H$.*

Proof. ($\Rightarrow$) Let $\mathcal{S} = \{(E^H, F^H) \mid E \approx_{\setminus H} F\}$. By Lemma 7, it is easy to prove that $\mathcal{S}$ is a weak bisimulation. ($\Leftarrow$) Let $\mathcal{S} = \{(E, F) \mid E^H \approx F^H\}$. By Lemma 7, it is easy to prove that $\mathcal{S}$ is a weak bisimulation up to H. $\square$

The results presented so far show that the τ completion of a process E is a *P_BNDC* process which can be used both to check whether E is *P_BNDC* and in case it is not to *rectify* it. Moreover, both the construction of E^τ and the *P_BNDC* test performed using E^τ have a low time complexity, as stated by the following result.

Theorem 6. *Let $T(n_1, n_2, m_1, m_2)$ be the time complexity of an algorithm to test $F_1 \approx F_2$ where $n_{1,2}$ is the number of nodes in $LTS(F_{1,2})$ and $m_{1,2}$ is the number of edges in $LTS(F_{1,2})$. It is possible to check if $E \in P\_BNDC$ in time $T(n, n, m^H, m^{\tau H}) + O(n + m^\tau)$, where n is the number of nodes in $LTS(E)$, m^H is the number of edges in $LTS(E^H)$, m^τ is the number of edges in $LTS(E^\tau)$, and $m^{\tau H}$ is the number of edges in $LTS((E^\tau)^H)$.*

Proof. This is an immediate consequence of the following facts:

- $LTS(E^H)$ and $LTS(E^\tau)$ have the same number of nodes of $LTS(E)$;
- $LTS(E^H)$ can be computed through a visit of $LTS(E)$;
- $LTS(E^\tau)$ can be computed through a visit of $LTS(E)$;
- $LTS((E^\tau)^H)$ can be computed through a visit of $LTS(E^\tau)$;
- the number of edges in $LTS(E^\tau)$ is greater than the number of edges in $LTS(E)$, hence $O(n+m) + O(n+m^\tau) = O(n+m^\tau)$, where m is the number of edges in $LTS(E)$.

□

Notice that if m is the number of edges in $LTS(E)$, then $m^\tau \leq m + m_H$, where m_H is the number of edges in $LTS(E)$ labelled with a high action. Moreover, $m^H \leq m + H * m_\tau$, where m_τ is the number of edges in $LTS(E)$ labelled with a τ action. Hence, $m^{\tau H} \leq m + +H * (m_\tau + m_H)$. However, in order to check $E^H \approx (E^\tau)^H$ it is not really necessary to explicitly compute E^H and $(E^\tau)^H$, since it is sufficient to build a simple interface for the bisimulation algorithm which reinterprets the labels of the transitions. More precisely, for instance, the set of processes which can evolve into E^H with a h action is equal to the union of the set of processes which can evolve into E with a τ or a h action.

5 An Example

In this section we illustrate through an example how the τ completion can be used to rectify a process which is neither P_BNDC nor *BSNNI*.

Note that, as the system is not *BSNNI*, then it is neither $BNDC$, i.e., it is insecure even with respect to non-dynamic environments. Moreover, the fact that the system is not *BSNNI* implies (by Theorem 3) that the low level semantics is not preserved by the τ completion. However, we will see that the way low level semantics is changed is very reasonable and only affects some deadlock states caused by high level activity.

Consider the process C described through a value-passing extension of SPA by

$$C = in(x).\overline{out}(x).C$$

C is a channel which may accept a value x at the left-hand port, labelled in. When it holds a value, it may deliver it at the right-hand port, labelled $\overline{out}$. If the domain of x is $\{0, 1\}$, then the channel C can be translated into SPA in a standard way by following [16] as:

$$C = in_0.\overline{out}_0.C + in_1.\overline{out}_1.C$$

Let us assume that C is used as communication channel from low to high level. This can be expressed as $in_0, in_1 \in L$ and $\overline{out}_0, \overline{out}_1 \in H$.

Note that such a channel should be secure as it provides a "legal" information flow from low to high. However, we show that this is not the case. If we compute C^τ we obtain

$$C^\tau = in_0.(\overline{out}_0.C^\tau + \tau.C^\tau) + in_1.(\overline{out}_1.C^\tau + \tau.C^\tau)$$

Moreover, C^H and $(C^\tau)^H$ are respectively

$$\begin{aligned} C^H \quad &= in_0.\overline{out}_0.C^H + in_1.\overline{out}_1.C^H + \overline{out}_0.C^H + \overline{out}_1.C^H \\ (C^\tau)^H &= in_0.(\overline{out}_0.(C^\tau)^H + \tau.(C^\tau)^H + \overline{out}_1.(C^\tau)^H) + \\ &\quad\; in_1.(\overline{out}_1.(C^\tau)^H + \tau.(C^\tau)^H + \overline{out}_0.(C^\tau)^H) + \\ &\quad\; \overline{out}_0.(C^\tau)^H + \overline{out}_1.(C^\tau)^H \end{aligned}$$

It is immediate to see that they are not weak bisimilar, since $(C^\tau)^H$ can silently reset itself after every input action, while C^H must always execute the corresponding output. Hence C is not P_BNDC. Intuitively, a high level user can indefinitely block the process C after each low level input by just refusing to accept the corresponding output (remind that communication is synchronous). The potential high level deadlocks could be exploited to transmit information as shown, e.g., in [7].

Now, by Theorem 2 we can replace C by C^τ thus obtaining a P_BNDC process. Intuitively C^τ is P_BNDC as the presence of "resetting" τ transitions avoids the high level deadlocks mentioned above.

These τ's basically makes the channel a lossy one, as high level outputs may be non-deterministically lost. However, note that non-determinism is used to abstract away implementation details. For example, such τ's could correspond, at implementation time, to time-outs for the high output actions, i.e., events that empty the channel and allow a new low level input, whenever high outputs are not accepted within a certain amount of time. In this respect, it would be quite interesting to rephrase our theory to models enriched with time or probability as [2, 12, 8], in order to study how the τ completion instantiate to more concrete settings. Even if the resulting process behaves differently from the low level point of view (C is not *BSNNI*), we think that C^τ can be reasonably proposed as a secure rectifying of C.

Indeed, note that the only difference, from a low level perspective, is the absence in $C^\tau \setminus H$ of deadlock states after the low level input actions. Such states of $C \setminus H$ are exactly the cause of potential information flows in process C, as they provide a causality between high level activity (i.e., accepting or not high outputs) and low level one.

In general when we define E^τ from a given process E and we add a τ transition relative to a high level output, this can always be seen as the insertion of a time-out. While in the case we add a τ transition relative to a high level input this corresponds to generating a non-deterministic high input. This latter case is clearly less reasonable. Hence, our transformation seems to be appropriate for fixing flows related to high level outputs. A rectifying strategy could be to add only the τ transitions relative to the outputs and check whether this is sufficient to have a *P_BNDC* process.

6 Conclusions

In the recent years, issues concerning security have received an increasing attention due to the augmented possibilities of interconnections and information exchanges. A number of formal definitions of security properties has been proposed in the literature.

In this paper we consider the security property *P_BNDC* based on the idea of Non-Interference [11, 19, 22] which has been deeply studied in [10] and showed to be suitable to guarantee security in a dynamically reconfigurable context. We present a method to automatically construct a *P_BNDC* process by a transformational approach. We show that the transformation preserves the low level observational semantics of *BSNNI* processes. Moreover we illustrate on an example how the transformation produces reasonable corrections also for non *BSNNI* processes where a modification of the low level semantics is necessary in order to ensure security. We show that our transformation can be used also to efficiently check the *P_BNDC* property, exploiting existing tools for bisimulation.

We are presently trying to apply our techniques to more significant examples in order to establish their effectiveness in producing secure systems from insecure ones. Moreover, it would be interesting to have a measure of what security damage would be in case *P_BNDC* does not hold. In [18], it is proposed a way of classifying information flow properties depending on which kind of channels from high to low level are implementable from systems that do not satisfy such properties. For example, obtaining a “perfect” channel represents a damage worse than, e.g., obtaining a noisy one. It could be interesting to measure the strenghtness of *P_BNDC* with respect to this kind of classification.

References

1. M. Abadi. Secrecy by Typing in Security Protocols. *Journal of the ACM*, 46(5):749–786, 1999.
2. A. Aldini. Probabilistic information flow in a process algebra. In *Proc. of the 12th International Conference on Concurrency Theory (CONCUR'01)*, pages 152–168. Springer LNCS 2154, August 2001.
3. C. Bodei, P. Degano, R. Focardi, and C. Priami. Primitives for Authentication in Process Algebras. *Theoretical Computer Science*, 2001. To appear.
4. T. Bolognesi and S. A. Smolka. Fundamental results for the verification of observational equivalence: A survey. In H. Rudin and C. H. West, editors, *Prooc. of Int'l Conference on Protocol Specification, Testing and Verification (PSTV'87)*, pages 165–179. North-Holland, 1987.
5. N. A. Durgin, J. C. Mitchell, and D. Pavlovic. A Compositional Logic for Protocol Correctness. In *Proc. of of the 14th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, 2001.

6. R. Focardi and R. Gorrieri. The Compositional Security Checker: A Tool for the Verification if Information Flow Security Properties. *IEEE Transactions on Software Engineering*, 23(9):550–571, 1997.
7. R. Focardi and R. Gorrieri. Classification of Security Properties (Part I: Information Flow). In R. Focardi and R. Gorrieri, editors, *Foundations of Security Analysis and Design*, volume 2171 of *LNCS*. Springer, 2001.
8. R. Focardi, R. Gorrieri, and F.Martinelli. Information-flow analysis in a discrete-time process algebra. In *Proc. of of the 13th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, 2001.
9. R. Focardi, C. Piazza, and S. Rossi. Proof methods for bisimulation based information flow security. In A. Cortesi, editor, *Proc. of Int. Workshop on Verification, Model Checking and Abstract Interpretation*, LNCS. Springer, 2002.
10. R. Focardi and S. Rossi. Information flow security in dynamic contexts. In *Proc. of of the 15th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, 2002.
11. J. A. Goguen and J. Meseguer. Security Policy and Security Models. In *Proc. of the 1982 Symposium on Security and Privacy*, pages 11–20. IEEE Computer Society Press, 1982.
12. Jan Jürjens. Secure information flow for concurrent processes. In *Proc. of International Conference on Concurrency Theory (Concur 2000)*. LNCS 1877, Springer-Verlag, August 2000.
13. Heiko Mantel and Andrei Sabelfeld. A generic approach to the security of multi-threaded programs. In *Proc. of of the 14th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, 2001.
14. D. McCullough. “A Hookup Theorem for Multilevel Security”. *IEEE Transactions on Software Engineering*, pages 563–568, June 1990.
15. J. McLean. “A General Theory of Composition for Trace Sets Closed Under Selective Interleaving Functions”. In *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*. IEEE Computer Society Press, 1994.
16. R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
17. L. C. Paulson. Proving Properties of Security Protocols by Induction. In *Proc. of the IEEE Computer Security Foundations Workshop*, pages 70–83. IEEE Computer Society Press, 1997.
18. Roberto Gorrieri Riccardo Focardi and Roberto Segala. A new definition of multilevel security. In *Proc. of Workshop on Issues in the Theory of Security (WITS '00), (P. Degano, ed.)*, July 2000.
19. A. Sabelfeld and D. Sands. Probabilistic Noninterference for Multi-threaded Programs. In *Proc. of the IEEE Computer Security Foundations Workshop*, pages 200–215. IEEE Computer Society Press, 2000.
20. S. Schneider. Verifying Authentication Protocols in CSP. *IEEE Transactions on Software Engineering*, 24(9), 1998.
21. V. Shmatikov and J. C. Mitchell. Analysis of a Fair Exchange Protocol. In *Proc. of 7th Annual Symposium on Network and Distributed System Security (NDSS 2000)*, pages 119–128. Internet Society, 2000.
22. G. Smith and D. M. Volpano. Secure Information Flow in a Multi-threaded Imperative Language. In *Proc. of 25th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL98)*, pages 355–364. ACM Press, 1998.