Information Flow Security in Dynamic Contexts*

Riccardo Focardi and Sabina Rossi

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

Abstract. We study a security property for processes in dynamic contexts, i.e., contexts that can be reconfigured at runtime. The security property that we propose, named *Persistent_BNDC*, is such that a process is "secure" when every state reachable from it satisfies a basic Non-Interference property. We define a suitable equivalence notion among processes, that allows us to express the new property as a single equivalence check, thus avoiding the universal quantifications over all the reachable states (required by *Persistent_BNDC*) and over all the possible hostile environments (implicit in the basic Non-Interference property we adopt). We also briefly discuss how *Persistent_BNDC* can be verified through model checking.

1 Introduction

In the recent years, systems are becoming more and more complex, and the security community has to face this by considering, e.g., issues like process mobility among different architectures and systems. A mobile process moving on the network can be influenced and reconfigured by the environments it crosses, possibly leading to new security breaches. As an example, consider a mobile agent that collects confidential data (e.g., marketing information) from different commercial hosts. It could be the case that one of the commercial hosts is malicious and tries to deduce some confidential information about the commercial hosts previously visited by the process. We should thus guarantee that the process is protected from the different visited hosts, that could be running different operating systems on different architectures.

In a complex setting like the one described above, we can abstractly think that the (possibly) hostile environment in which a system or an application is running can be dynamically reconfigured at run-time, changing in unpredictable ways. A program executing in a "secure way" inside one environment could find itself in a different setting (with different malicious attackers) at runtime, e.g., if the process decides to migrate during its execution like in the "collecting agent" example above. This could lead to unexpected dangerous situations as a program which is in a secure state for a certain environment might find itself with no protection if the environment itself suddenly changes.

* This work has been partially supported by MURST projects "Interpretazione astratta, type systems e analisi control-flow" and "Modelli Formali per la sicurezza".

A number of formal definitions of security properties (see, for instance, [1, 3, 4, 7, 8, 12, 14–16]) has been proposed in the literature.

In this paper we face the problem of defining a security property based on the idea of Non-Interference [9, 13, 16, 19] (formalized as *BNDC* [6]), which is suitable to analyze processes in completely dynamic hostile environments. The basic idea is to require that every state which is reachable by the system still satisfies a basic Non-Interference property. If this holds, we are assured that even if the system migrates during its execution no malicious attacker will be able to compromise it, as every possible reachable state is guaranteed to be secure.

This extension of *BNDC*, called *Persistent_BNDC* (*P_BNDC*, for short), leads also to some interesting results, as we are able to prove that it may be equivalently defined by considering the *BNDC* property with a different underlying equivalence notion, i.e., adopting a different discriminating power on processes. This result places this new property in the already studied taxonomy of Non-Interference properties [6] and gives a quite efficient way of verifying *P_BNDC*, as it allows us to avoid both the universal quantification over all the possible attackers, which is present in the *BNDC* basic definition, and the universal quantification over all possible reachable states, required by the definition of *P_BNDC* itself. Finally, as we show that *P_BNDC* is equivalent to an already proposed property called *SBSNNI* [6], this work also contributes in giving new verification techniques for such a property.

The paper is organized as follows. In Section 2, we define the *Security Process Algebra* (SPA) language and recall the notion of *weak bisimulation* over SPA terms. In Section 3, we give the definitions of *BNDC* and *P_BNDC*, we prove some results that relate the new property with already studied security properties and we characterize *P_BNDC* through a new definition of *weak bisimulation up to high level actions*. In Section 4, we report an example of a system which satisfies *P_BNDC* and, finally, in Section 5, we briefly discuss how *Persistent_BNDC* can be verified through model checking and give some concluding remarks.

2 The SPA language

The *Security Process Algebra* (SPA, for short) [6] is a slight extension of Milner's CCS [10], 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 $I = \{a, b, \ldots\}$ is a set of *input* actions and $O = \{\bar{a}, \bar{b}, \ldots\}$ is a set of *output* actions; a complementation function $\bar{\cdot} : \mathcal{L} \to \mathcal{L}$, such that $\bar{\bar{a}} = a$, for all $a \in \mathcal{L}$; a special action τ which models internal computations, i.e., not visible outside the system. $Act = \mathcal{L} \cup \{\tau\}$ is the set of all *actions*. Function $\bar{\cdot}$ is extended to Act by defining $\bar{\tau} = \tau$. In order to obtain a partition of the visible actions into two levels we consider also two sets, Act_H and Act_L, of high and low level actions which are closed with respect to function $\bar{\cdot}$, i.e., $\overline{Act_H} = Act_H$ and $\overline{Act_L} = Act_L$; moreover they form a covering of $\mathcal{L}$ and they are disjoint, i.e., $Act_H \cup Act_L = \mathcal{L}$ and $Act_H \cap Act_L = \emptyset$.

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

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

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 $Z \stackrel{\text{def}}{=} E$.

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 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 the executions of the two processes 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[1]; $E[f]$ 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 $v \subseteq \mathcal{L}$, $E/v \stackrel{\text{def}}{=} E[f_v]$ 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.

Let $\mathcal{E}$ be the set of SPA agents, ranged over by E and F. Let $\mathcal{L}(E)$ denote the *sort* of E, i.e., the set of the (possibly executable) actions occurring syntactically in E. The sets of high level agents and low level ones are defined as $\mathcal{E}_H \stackrel{\text{def}}{=} \{E \in \mathcal{E} \mid \mathcal{L}(E) \subseteq Act_H \cup \{\tau\}\}$ and $\mathcal{E}_L \stackrel{\text{def}}{=} \{E \in \mathcal{E} \mid \mathcal{L}(E) \subseteq Act_L \cup \{\tau\}\}$, respectively. Note that $\mathcal{E}_H \cup \mathcal{E}_L \subset \mathcal{E}$, i.e., there exist systems that execute both high and low level actions allowing communications between the two levels.

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 axioms and 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 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 states/terms of the SPA LTS, equating two processes when they are indistinguishable. In this paper we consider the *weak bisimulation* equivalence. The general notion of *bisimulation* [10] consists of a mutual step-by-step simulation, i.e., given two processes E and F, when E executes a certain action moving to E' then F must be able to simulate this single step by executing the same action and moving to an agent F' which is again bisimilar to E', and vice-versa.

[1] Notice that in CCS the operator $\setminus$ requires that the actions of $E \setminus v$ do not belong to $v \cup \bar{v}$.

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

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

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

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

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

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

Fig. 1. The operational rules for SPA

A weak bisimulation is a bisimulation which 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 notations.

If $t = a_1 \cdots a_n \in Act^*$, then we write $E \xrightarrow{t} E'$ if $E \xrightarrow{a_1} \cdots \xrightarrow{a_n} E'$. 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. Note, in particular, that $\widehat{\tau^n} = \epsilon$ (the empty sequence). 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 $\stackrel{\hat{\tau}}{\Longrightarrow} \equiv (\xrightarrow{\tau})^*$ means zero or more τ labelled transitions). We say that E' is reachable from E when there exists t such that $E \xrightarrow{t} E'$.

The notion of *weak bisimulation* is defined as follows.

Definition 1 (Weak Bisimulation). *A binary relation $\mathcal{S} \subseteq \mathcal{E} \times \mathcal{E}$ over agents is a* weak bisimulation *if $(E, F) \in \mathcal{S}$ implies, for all $a \in Act$,*

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

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

In [10] it is proved that $\approx$ is an equivalence relation.

3 Security Properties

In this section, we give some definitions that try to capture every possible information flow from a *classified* (*high*) level of confidentiality to an *untrusted* (*low*) one. A strong requirement of these definitions is that no information flow should be possible even in the presence of malicious processes that run at the classified level. 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 are apparently honest but hide inside some malicious code. This programs might be for example downloaded from the network or sent by e-mail, and executed by a high level user at the classified level. In the presence of mobility this becomes of course more and more crucial, as a Trojan Horse program could just enter the system through some transparent mechanism (like, e.g., the download of an applet), and the high level user could never be aware of being executing some downloaded (potentially malicious) code.

The definitions we are going to present are all based on the basic idea of Non-Interference [9]: "No information flow is possible from high to low if what is done at the high level *cannot interfere* in any way with the low level".

We start by reporting from [6] the definition of *Bisimulation-based Non Deducibility on Compositions* (*BNDC*, for short). Then, we extend it in order to deal with dynamic contexts, i.e., by considering the possibility for the environment (both high and low level) to change in an unpredictable way. This is what basically happens if we consider mobile processes. To make the system secure in this setting, we require that every single state of the system is secure by itself. Thus, even if the process is moved in the middle of a computation, this will never lead to security breaches.

3.1 Non Deducibility on Compositions

The security property called *Bisimulation-based Non Deducibility on Compositions* (*BNDC*, for short) is based on the idea of checking the system against all high level potential interactions, representing every possible high level malicious program. A system E is *BNDC* if for every high level process Π a low level user cannot distinguish E from $(E|\Pi) \setminus Act_H$ (see Fig. 2). 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 2 (BNDC). *Let $E \in \mathcal{E}$.*

$$E \in BNDC \text{ iff } \forall\, \Pi \in \mathcal{E}_H,\; E \setminus Act_H \approx (E|\Pi) \setminus Act_H .$$

We show how *BNDC* works through some simple examples.

Example 1. First, consider process $E_1 = l_1.h.\bar{l_2}.\mathbf{0}$, where h is the only high level action. It basically accepts the low level input l_1 and then gives $\bar{l_2}$ as output only if h is executed. Note that we have a direct causality between the high level input h and the low level output $\bar{l_2}$, representing a direct information flow. As expected,

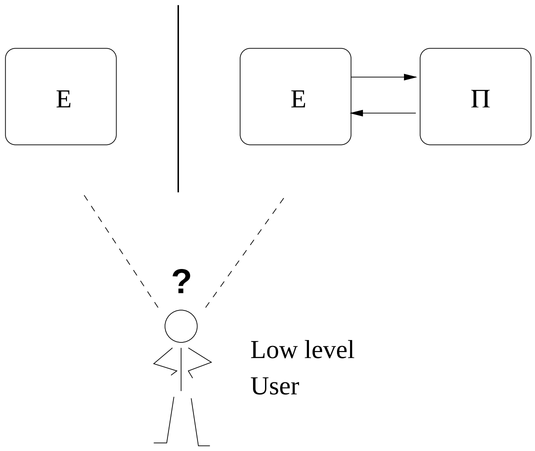

Fig. 2. The *BNDC* property

this system is not *BNDC*. It is sufficient to consider $\Pi = \bar{h}.\mathbf{0}$. In this case it is straightforward to see that $(E_1|\Pi) \setminus Act_H \approx l_1.\bar{l_2}.\mathbf{0}$ while $E_1 \setminus Act_H \approx l_1.\mathbf{0}$. Note that the latter can never execute $\bar{l_2}$, thus $E_1 \setminus Act_H \not\approx (E_1|\Pi) \setminus Act_H$.

Example 2. The next example aims at showing that *BNDC* is powerful enough to detect information flows due to the possibility for a high level malicious process to block or unblock a system. Let us just extend system E_1 as follows: $E_2 = l_1.h.\bar{l_2}.\mathbf{0} + l_1.\bar{l_2}.\mathbf{0}$. We have basically added the trace $l_1.\bar{l_2}$ in order to break the direct causality between h and $\bar{l_2}$. As a matter of fact, now $\bar{l_2}$ may be executed even without h has been previously performed. However, consider the same process $\Pi = \bar{h}.\mathbf{0}$ as before. We again have that $(E_2|\Pi) \setminus Act_H \approx l_1.\bar{l_2}.\mathbf{0}$, but now $E_2 \setminus Act_H \approx l_1.\mathbf{0} + l_1.\bar{l_2}.\mathbf{0}$. Note that the latter may (nondeterministically) block after the l_1 input. Having many instances of this process, a low level user could deduce if $\bar{h}$ is executed by observing whether the system always performs $\bar{l_2}$ or not. More discussion about how this can be exploited to actually implement a communication channel from high to low may be found in [6]. Process E_2 may be "repaired" and made *BNDC*, by including the possibility of choosing to execute $\bar{l_2}$ or not inside the process, thus completely masking high level activity. Indeed, process $E_3 = l_1.h.\bar{l_2}.\mathbf{0} + l_1.(\tau.\bar{l_2}.\mathbf{0} + \tau.\mathbf{0})$ can be proved to be *BNDC*.

3.2 Persistent_BNDC

We define a security property which is stronger than *BNDC* but which allows us to deal with possibly dynamic attackers, i.e., high level processes which can be dynamically reconfigured. The novel security property is named *Persistent_BNDC* (*P_BNDC*, for short). The idea is that a system E is *P_BNDC* if for every high level process Π and for every state E' reachable from E a low level

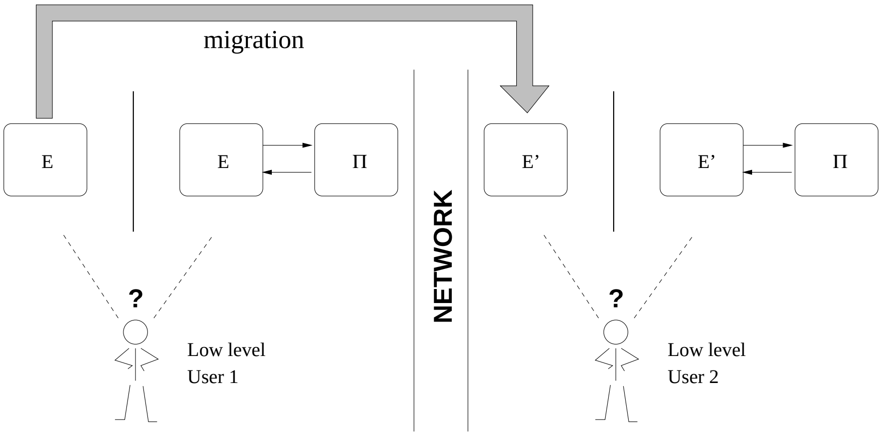

Fig. 3. The *P_BNDC* property

user cannot distinguish E' from $(E'|\Pi) \setminus Act_H$ (see Fig. 3). This is equivalent to say that E is *P_BNDC* if for every state E' reachable from E, E' is *BNDC*.

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

$$E \in P\_BNDC \;\; \textit{iff} \;\; \forall\, E' \textit{ reachable from } E \textit{ and } \forall\, \Pi \in \mathcal{E}_H,$$
$$E' \setminus Act_H \approx (E'|\Pi) \setminus Act_H., \;\; \textit{i.e.,} \;\; E' \in BNDC.$$

We show the idea of *P_BNDC* through a simple example.

Example 3. Consider again the *BNDC* process E_3 of Example 2, $E_3 = l_1.h.\bar{l_2}.\mathbf{0}+ l_1.(\tau.\bar{l_2}.\mathbf{0} + \tau.\mathbf{0})$. Suppose it may be moved in the middle of a computation. This might happen when it find itself in the state $h.\bar{l_2}.\mathbf{0}$ (after the first l_1 is executed). Now it is clear that this process is not secure, as a direct causality between h and $\bar{l_2}$ is present. In particular $h.\bar{l_2}.\mathbf{0}$ is not *BNDC* and this gives evidence that E_3 is not *P_BNDC*. The process may be "repaired" as follows: $E_4 = l_1.(h.\bar{l_2}.\mathbf{0} + \tau.\bar{l_2}.\mathbf{0} + \tau.\mathbf{0}) + l_1.(\tau.\bar{l_2}.\mathbf{0} + \tau.\mathbf{0})$. It may be proved that E_4 is *P_BNDC*. Note that, from this example it follows that *P_BNDC* is strictly stronger that *BNDC*, i.e., *P_BNDC* $\subset$ *BNDC*.

The next proposition shows that property *P_BNDC* is equivalent to the already proposed security property *SBSNNI* [5, 6] defined as follows.

Definition 4 (SBSNNI). *Let* $E \in \mathcal{E}$.

$$E \in SBSNNI \;\; \textit{iff} \;\; \forall\, E' \textit{ reachable from } E, \; E' \setminus Act_H \approx E'/Act_H.$$

This property was introduced to automatically check *BNDC*, i.e., to bypass the quantification over all the possible malicious high level processes. As it follows from the next proposition, *SBSNNI* is strictly stronger than *BNDC*, since, quite interestingly, it is equivalent to *P_BNDC*.

Proposition 1. $P\_BNDC = SBSNNI$.

Proof. $P\_BNDC \subseteq SBSNNI$ follows from the definitions of the two properties, while $SBSNNI \subseteq P\_BNDC$ follows from the proof of Proposition 6 in [6].

The next proposition follows from Theorem 3 in [6]: it states that property P_BNDC is *compositional*, in the sense that it is preserved by the parallel and restriction operators.

Proposition 2.

- *if* $E, F \in P\_BNDC$ *then* $(E|F) \in P\_BNDC$,
- *if* $E \in P\_BNDC$ *and* $v \subseteq \mathcal{L}$ *then* $E \setminus v \in P\_BNDC$.

We have seen that our new property P_BNDC is indeed the same as $SBSNNI$, a property which is automatically checkable over finite state processes and which has been previously studied in [5, 6]. As we already noticed, this solves the problem of quantifying over all possible high processes. However, we still have to verify a property (which is simpler than $BNDC$) over all the possible reachable states from the initial process E. In the following, we show that this can be avoided, by including the idea of "being secure in every state" directly inside the bisimulation equivalence notion. We do this in two steps: (i) we define a quite "natural" notion of bisimulation which implicitly contains a quantification over all possible malicious contexts; (ii) in order to overcome to the newly introduced quantification over contexts, we show that such an equivalence notion may be also defined in a simpler way, which do not require any universal quantification (neither on malicious contexts/processes nor on reachable states).

We start by showing that the universal quantification on high level processes can be moved from the definition of P_BNDC to the definition of a suitable equivalence relation, named $\approx_{hc}$, such that $E \in P\_BNDC$ if and only if E and $E \setminus Act_H$ are not distinguishable with respect to $\approx_{hc}$ for all hostile contexts, where the notion of hostile context is defined as follows.

Definition 5 (Context). *A* hostile context $C[\_]$ *denotes a term of the form* $(\_ \mid \Pi) \setminus Act_H$ *where* $\Pi \in \mathcal{E}_H$, *which can be regarded as a mapping from* $\mathcal{E}$ *to* $\mathcal{E}$ *that associates with each process* $E \in \mathcal{E}$ *the process* $C[E] \equiv (E|\Pi) \setminus Act_H$.

Observe that for any hostile context $C[\_]$ and process E, all the processes reachable from $C[E]$ have the form $C'[E']$ with $C'[\_]$ being a hostile context too.

We introduce the concept of *weak bisimulation on hostile contexts*: the idea is that, given two processes E and F, when a hostile context $C[\_]$ filled with E executes a certain action moving E to E' then the same context filled with F is able to simulate this step moving F to F' so that E' and F' are again weakly bisimilar on hostile contexts, and vice-versa. This must be true for every possible hostile context $C[\_]$. It is important to note that the quantification over all possible hostile contexts is re-iterated for E' and F'. This makes this equivalence suitable for dynamic settings in which the environment may change in the middle of system execution.

We use these notations. If $t \in Act^*$ and $C[\_]$ is a hostile context, then we write $E \xrightarrow{t}_C E'$ if $C[E] \xrightarrow{t} C'[E']$; we write $E \xRightarrow{t}_C E'$ if $C[E] \xRightarrow{t} C'[E']$. Thus, $E \xRightarrow{\hat{a}}_C E'$ stands for $C[E] \xRightarrow{a} C'[E']$ if $a \in \mathcal{L}$, and for $C[E](\xrightarrow{\tau})^* C'[E']$ if $a = \tau$.

Definition 6 (Weak Bisimulation on hostile contexts). *A binary relation $\mathcal{S} \subseteq \mathcal{E} \times \mathcal{E}$ over agents is a* weak bisimulation on hostile contexts *if $(E, F) \in \mathcal{S}$ implies, for all context $C[\_]$ and for all $a \in Act$,*

- *whenever $E \xrightarrow{a}_C E'$, then there exists F' such that $F \xRightarrow{\hat{a}}_C F'$ and $(E', F') \in \mathcal{S}$;*
- *whenever $F \xrightarrow{a}_C F'$, then there exists E' such that $E \xRightarrow{\hat{a}}_C E'$ and $(E', F') \in \mathcal{S}$.*

We say that two processes $E, F \in \mathcal{E}$ are *weakly bisimilar on hostile contexts*, written $E \approx_{hc} F$, if $(E, F) \in \mathcal{S}$ for some weak bisimulation on hostile contexts $\mathcal{S}$. This may be equivalently expressed as follows:

$$\approx_{hc} = \bigcup \{\mathcal{S} : \ \mathcal{S} \text{ is a weak bisimulation on hostile contexts}\}.$$

It is easy to prove that

- $\approx_{hc}$ is the largest weak bisimulation on hostile contexts
- $\approx_{hc}$ is an equivalence relation.

The next lemma is used for proving the subsequent theorem and easily follows from the definitions of hostile context and weak bisimulation on hostile contexts.

Lemma 1. *Let $E \in \mathcal{E}$ such that $E \approx_{hc} E \setminus Act_H$. Then for all E' reachable from E there exists $E'' \setminus Act_H$ reachable from $E \setminus Act_H$ such that $E' \approx_{hc} E'' \setminus Act_H$.*

Theorem below gives a characterization of P_BNDC processes in terms of $\approx_{hc}$.

Theorem 1. *Let $E \in \mathcal{E}$. Then, $E \in P\_BNDC$ iff $E \setminus Act_H \approx_{hc} E$.*

Proof. We first show that $E \setminus Act_H \approx_{hc} E$ implies $E \in P\_BNDC$. In order to do it we prove that

$$\mathcal{S} = \{(E_1 \setminus Act_H, (E_2|\Pi) \setminus Act_H) \mid \Pi \in \mathcal{E}_H \text{ and } E_1 \setminus Act_H \approx_{hc} E_2\}$$

is a bisimulation. This is sufficient to say that $E \in P\_BNDC$. In fact, by Lemma 1, for every state E' reachable from E there exists a state $E'' \setminus Act_H$ reachable from $E \setminus Act_H$ such that $E'' \setminus Act_H \approx_{hc} E'$. Hence, by definition of $\mathcal{S}$, we have that for all $\Pi \in \mathcal{E}_E$, $(E'' \setminus Act_H, (E'|\Pi) \setminus Act_H) \in \mathcal{S}$. Since $\mathcal{S}$ is a bisimulation, we have that for all $\Pi \in \mathcal{E}_E$, $E'' \setminus Act_H \approx (E'|\Pi) \setminus Act_H$ and, in particular, $E'' \setminus Act_H \approx E' \setminus Act_H$. Since $\approx$ is an equivalence relation, by symmetry and transivity, we have that for every E' reachable from E and for all $\Pi \in \mathcal{E}_E$, $E' \setminus Act_H \approx (E'|\Pi) \setminus Act_H$, i.e., $E \in P\_BNDC$.

The fact that $\mathcal{S}$ is a bisimulation follows from the following four cases. Let $(E_1 \setminus Act_H, (E_2|\Pi) \setminus Act_H) \in \mathcal{S}$.

Case 1. $E_1 \setminus Act_H \xrightarrow{a} E_1' \setminus Act_H$ with $a \notin Act_H$. Thus, for all context $C[\_]$, $E_1 \setminus Act_H \xrightarrow{a}_C E_1' \setminus Act_H$. By the hypothesis that $E_1 \setminus Act_H \approx_{hc} E_2$, for all context $C[\_]$ there exists E'_2 such that $E_2 \stackrel{\hat{a}}{\Longrightarrow}_C E'_2$ and $E'_1 \setminus Act_H \approx_{hc} E'_2$. Hence there exists E'_2 such that $(E_2|\Pi) \setminus Act_H \stackrel{\hat{a}}{\Longrightarrow} (E'_2|\Pi') \setminus Act_H$ and, by definition of $\mathcal{S}$, $(E'_1 \setminus Act_H, (E'_2|\Pi') \setminus Act_H) \in \mathcal{S}$.

Case 2. $(E_2|\Pi) \setminus Act_H \stackrel{a}{\to} (E'_2|\Pi) \setminus Act_H$ with $E_2 \setminus Act_H \stackrel{a}{\to} E'_2 \setminus Act_H$ and $a \notin Act_H$. Let $C[\_] \equiv (\_|\mathbf{0}) \setminus Act_H$. Hence $E_2 \stackrel{a}{\to}_C E'_2$. By the hypothesis that $E_1 \setminus Act_H \approx_{hc} E_2$, there exists E'_1 such that $E_1 \setminus Act_H \stackrel{\hat{a}}{\Longrightarrow}_C E'_1 \setminus Act_H$ and $E'_1 \setminus Act_H \approx_{hc} E'_2$. Since $C[E_1 \setminus Act_H]$ can only perform actions of $E_1 \setminus Act_H$, we have that $E_1 \setminus Act_H \stackrel{\hat{a}}{\Longrightarrow} E'_1 \setminus Act_H$ and $(E'_1 \setminus Act_H, (E'_2|\Pi) \setminus Act_H) \in \mathcal{S}$.

Case 3. $(E_2|\Pi) \setminus Act_H \stackrel{\tau}{\to} (E_2|\Pi') \setminus Act_H$ with $\Pi \stackrel{\tau}{\to} \Pi'$. By definition of $\mathcal{S}$, it follows trivially that $(E_1 \setminus Act_H, (E_2|\Pi') \setminus Act_H) \in \mathcal{S}$.

Case 4. $(E_2|\Pi) \setminus Act_H \stackrel{\tau}{\to} (E'_2|\Pi') \setminus Act_H$ where $E_2 \stackrel{a}{\to} E'_2$, $\Pi \stackrel{\bar{a}}{\to} \Pi'$ and $a \in Act_H$. Let $C[\_] \equiv (\_|\bar{a}) \setminus Act_H$. Hence $E_2 \stackrel{\tau}{\to}_C E'_2$. By the hypothesis that $E_1 \setminus Act_H \approx_{hc} E_2$, there exists E'_1 such that $E_1 \setminus Act_H \stackrel{\hat{\tau}}{\Longrightarrow}_C E'_1 \setminus Act_H$ and $E'_1 \setminus Act_H \approx_{hc} E'_2$. Since $C[E_1 \setminus Act_H]$ can only perform actions of $E_1 \setminus Act_H$, we have that $E_1 \setminus Act_H \stackrel{\hat{\tau}}{\Longrightarrow} E'_1 \setminus Act_H$ and $(E'_1 \setminus Act_H, (E'_2|\Pi) \setminus Act_H) \in \mathcal{S}$.

We now show that $E \in P\_BNDC$ implies $E \setminus Act_H \approx_{hc} E$. To this end it is sufficient to prove that

$$\mathcal{S} = \{(E_1 \setminus Act_H, E_2) \mid E_1 \setminus Act_H \approx E_2 \setminus Act_H \text{ and } E_2 \in P\_BNDC\}$$

is a weak bisimulation on hostile contexts. This follows from the following three cases. Let $C[\_]$ be an hostile context.

Case 1. $E_1 \setminus Act_H \stackrel{a}{\to}_C E'_1 \setminus Act_H$. Then $E_1 \setminus Act_H \stackrel{a}{\to} E'_1 \setminus Act_H$. From the fact that $E_1 \setminus Act_H \approx E_2 \setminus Act_H$, there exists E'_2 such that $E_2 \setminus Act_H \stackrel{\hat{a}}{\Longrightarrow} E'_2 \setminus Act_H$ and $E'_1 \setminus Act_H \approx E'_2 \setminus Act_H$. Moreover, since $E_2 \in P\_BNDC$ also $E'_2 \in P\_BNDC$. From the fact that $E_2 \setminus Act_H \stackrel{\hat{a}}{\Longrightarrow} E'_2 \setminus Act_H$ we have that $E_2 \stackrel{\hat{a}}{\Longrightarrow} E'_2$ and thus $E_2 \stackrel{\hat{a}}{\Longrightarrow}_C E'_2$ and, by definition of $\mathcal{S}$, $(E'_1 \setminus Act_H, E'_2) \in \mathcal{S}$.

Case 2. $E_2 \stackrel{a}{\to}_C E'_2$ with $E_2 \setminus Act_H \stackrel{a}{\to} E'_2 \setminus Act_H$. Since $E_1 \setminus Act_H \approx E_2 \setminus Act_H$, there exists E'_1 such that $E_1 \setminus Act_H \stackrel{\hat{a}}{\Longrightarrow} E'_1 \setminus Act_H$ and $E'_1 \setminus Act_H \approx E'_2 \setminus Act_H$. Moreover, since $E_2 \in P\_BNDC$ also $E'_2 \in P\_BNDC$. Hence $E_1 \setminus Act_H \stackrel{\hat{a}}{\Longrightarrow}_C E'_1 \setminus Act_H$ and, by definition of $\mathcal{S}$, $(E'_1 \setminus Act_H, E'_2) \in \mathcal{S}$.

Case 3. $E_2 \stackrel{\tau}{\to}_C E'_2$ with $E_2 \stackrel{a}{\to} E'_2$ and $a \in Act_H$. Then, $E_2/Act_H \stackrel{\tau}{\to} E'_2/Act_H$. From the fact that $E_1 \setminus Act_H \approx E_2 \setminus Act_H$ and $E_2 \in P\_BNDC$, we have that $E_1 \setminus Act_H \approx E_2/Act_H$ and thus there exists E'_1 such that $E_1 \setminus Act_H \stackrel{\hat{\tau}}{\Longrightarrow} E'_1 \setminus Act_H$ and $E'_1 \setminus Act_H \approx E'_2/Act_H$. Moreover, since $E_2 \in P\_BNDC$ also $E'_2 \in P\_BNDC$ and hence $E'_1 \setminus Act_H \approx E'_2 \setminus Act_H$. Thus $E_1 \setminus Act_H \stackrel{\hat{\tau}}{\Longrightarrow}_C E'_1 \setminus Act_H$ and, by definition of $\mathcal{S}$, $(E'_1 \setminus Act_H, E'_2) \in \mathcal{S}$. ∎

Theorem 1 provides a characterization for P_BNDC processes which is however still difficult to use in practice, because of the implicit quantification over all hostile contexts. The next step, is thus to show that the same equivalence notion may be expressed in a rather simpler way by just focusing on observable actions which do not belong to Act_H. More in detail, we define an observation equivalence where actions from Act_H *may* be ignored, i.e., they may be matched by zero or more τ actions. To this aim, we use a transition relation which does not take care of both internal actions and actions from Act_H.

Definition 7. *For an action $a \in Act$, we write $(\stackrel{a}{\to})^{\{0,1\}}$ to denote a sequence of zero or one a actions. The expression $E \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} E'$ is a shorthand for $E \stackrel{\hat{a}}{\Longrightarrow} E'$ if $a \notin Act_H$, and for $E(\stackrel{\tau}{\longrightarrow})^*(\stackrel{a}{\longrightarrow})^{\{0,1\}}(\stackrel{\tau}{\longrightarrow})^*E'$ if $a \in Act_H$.*

Notice that the relation $\stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H}$ is a generalization of the relation $\stackrel{\hat{a}}{\Longrightarrow}$ used in the definition of weak bisimulation [10]. In fact, if $Act_H = \emptyset$ then for all $a \in Act$, $E \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} E'$ coincides with $E \stackrel{\hat{a}}{\Longrightarrow} E'$.

We define the concept of weak bisimulation up to Act_H.

Definition 8 (Weak Bisimulation up to Act_H). *A binary relation $\mathcal{S} \subseteq \mathcal{E} \times \mathcal{E}$ over agents is a* weak bisimulation up to Act_H *if $(E, F) \in \mathcal{S}$ implies, for all $a \in Act$,*

- *whenever $E \stackrel{a}{\to} E'$, then there exists F' such that $F \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} F'$ and $(E', F') \in \mathcal{S}$;*
- *whenever $F \stackrel{a}{\to} F'$, then there exists E' such that $E \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} E'$ and $(E', F') \in \mathcal{S}$.*

We say that two agents $E, F \in \mathcal{E}$ are *weakly bisimilar up to Act_H*, written $E \approx_{\setminus Act_H} F$, if $(E, F) \in \mathcal{S}$ for some weak bisimulation $\mathcal{S}$ up to Act_H. It is easy to prove that

- $\approx_{\setminus Act_H}$ is the largest weak bisimulation up to Act_H
- $\approx_{\setminus Act_H}$ is an equivalence relation.

The next theorem shows that the relations $\approx_{hc}$ and $\approx_{\setminus Act_H}$ are equivalent.

Theorem 2. *Let $E, F \in \mathcal{E}$.*

$$E \approx_{hc} F \text{ iff } E \approx_{\setminus Act_H} F.$$

Proof. We first show that $E \approx_{hc} F$ implies $E \approx_{\setminus Act_H} F$. To this end it is sufficient to prove that

$$\mathcal{S} = \{(E, F) \mid E \approx_{hc} F\}$$

is a weak bisimulation up to Act_H. This follows from the following two cases.

Case 1. $E \stackrel{a}{\to} E'$ with $a \notin Act_H$. Let $C \equiv (\_|\mathbf{0}) \setminus Act_H$ be a hostile context. Then $E \stackrel{a}{\to}_C E'$. From the fact that $E \approx_{hc} F$ it follows that there exists F' such that $F \stackrel{\hat{a}}{\Longrightarrow}_C F'$ and $E' \approx_{hc} F'$. By the choice of C, we also have that $F \stackrel{\hat{a}}{\Longrightarrow} F'$ and, since $a \notin Act_H$, $F \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} F'$. Moreover, by definition of $\mathcal{S}$, $(E', F') \in \mathcal{S}$.

Case 2. $E \stackrel{a}{\to} E'$ with $a \in Act_H$. Let $C \equiv (\_|\bar{a}) \setminus Act_H$ be a hostile context. Then $E \stackrel{\tau}{\to}_C E'$. From the fact that $E \approx_{hc} F$ it follows that there exists F'

such that $F \stackrel{\hat{\tau}}{\Longrightarrow}_C F'$ and $E' \approx_{hc} F'$. By the choice of C, we also have that $F \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} F'$ and, by definition of $\mathcal{S}$, $(E', F') \in \mathcal{S}$.

We now show that $E \approx_{\setminus Act_H} F$ implies $E \approx_{hc} F$. To this end it is sufficient to prove that

$$\mathcal{S} = \{(E, F) \mid E \approx_{\setminus Act_H} F\}$$

is a weak bisimulation on hostile contexts. This follows from the following two cases. Let $C[\_]$ be a hostile context.

Case 1. $E \stackrel{a}{\rightarrow}_C E'$ with $E \stackrel{a}{\rightarrow} E'$ and $a \notin Act_H$. Since $E \approx_{\setminus Act_H} F$, there exists F' such that $F \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} F'$ and $E' \approx_{\setminus Act_H} F'$. Since $a \notin Act_H$, we also have $F \stackrel{\hat{a}}{\Longrightarrow} F'$. Thus $F \stackrel{\hat{a}}{\Longrightarrow}_C F'$ and, by definition of $\mathcal{S}$, $(E', F') \in \mathcal{S}$.

Case 2. $E \stackrel{\tau}{\rightarrow}_C E'$ with $E \stackrel{a}{\rightarrow} E'$ and $a \in Act_H$. Since $E \approx_{\setminus Act_H} F$, there exists F' such that $F \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} F'$ and $E' \approx_{\setminus Act_H} F'$. Thus either $F \stackrel{\hat{\tau}}{\Longrightarrow} F'$ or $F \stackrel{\hat{a}}{\Longrightarrow} F'$. Since the hostile context $C[\_]$ may synchronize on a by performing the complementary action $\bar{a}$, we have that $F \stackrel{\hat{\tau}}{\Longrightarrow}_C F'$ and $(E', F') \in \mathcal{S}$. ∎

Theorem 2 allows us to identify a local property of processes (with no quantification on the states and on the hostile contexts) which is a necessary and sufficient condition for *P_BNDC*. This is stated by the following corollary.

Corollary 1. *Let* $E \in \mathcal{E}$. $E \in P\_BNDC$ *iff* $E \setminus Act_H \approx_{\setminus Act_H} E$.

In practice, we have proven that a process is *P_BNDC* if and only if it is equivalent – with respect to a particular bisimulation-like equivalence relation – to the same process prevented from performing high level actions.

4 An example

In this section we report from [5, 6] a non trivial example of a system which is *SBSNNI* and thus *P_BNDC*. Our aim is to give evidence that the proposed property is not too restrictive. Moreover, in [5, 6], *SBSNNI* was used to prove that the system was *BNDC* (as a sufficient condition). Following the intuition of *P_BNDC* we can now state that such a system is secure even when its execution environment is dynamic and changes at runtime. The system itself could be seen as a very simple mobile "colelcting agent" (as the one described in the introduction) that may be accessed in different hosts.

More precisely, the example reported here is an access monitor which handles read and write requests on two binary variables enforcing the Multilevel Security Policy [2], a particular access control policy which has the aim of ensuring that no information flow is possible from high level to low one. The policy is based on two access rules: *no read up*, i.e., no subject can read from an object with a higher level; *no write down*, i.e., no subject can write to an object with a lower level. In particular, the access monitor process handles read and write requests from high and low level users on two binary objects: a high level variable and a low

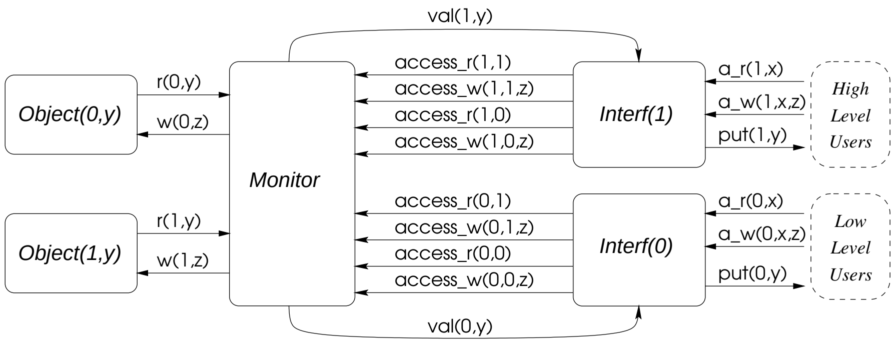

Fig. 4. The *P_BNDC Access_Monitor*

level one. It achieves *no read up* and *no write down* access control rules allowing a high level user to read from both objects and write only on the high one; conversely, a low level user is allowed to write on both objects and read only from the low one.

The access monitor system [2] is reported in Fig. 5 (see also Fig. 4). In such a system we have that $k \in \{0, 1, err\}$, $L = \{r, w\}$, $N = \{val,\ access\_r,\ access\_w\}$ and $a\_r(1, x)$, $a\_w(1, x)$, $put(1, y) \in Act_H$ $\forall x \in \{0, 1\}$ and $\forall y \in \{0, 1, err\}$, while the same actions with 0 as first parameter belong to Act_L. Users interact with the monitor through the following access actions:

- $a\_r(l, x)$, a read request from level l to object x;
- $a\_w(l, x, z)$, a write request from level l to object x of value z;
- $put(l, y)$, the response to level l for a previous read request; y is the returned (read) value.

As an example, consider $a\_r(0, 1)$ which represents a low level user ($l = 0$) read request from the high level object ($x = 1$), and $a\_w(1, 0, 0)$ which represents a high level user ($l = 1$) write request of value 0 ($z = 0$) on the low object ($x = 0$). Read results are returned to users through the output actions $put(l, y)$. This can be also an error in case of a read-up request. Note that if a high level user tries to write on the low object – through $access\_w(1, 0, z)$ – such a request is not executed and no error message is returned.

The *Access_Monitor* is the parallel composition of the actual monitor AM and an interface for each level which temporarily stores the output value of the monitor (passing it later to the users and thus making communication asynchronous) and that guarantees mutual exclusion within the same level. This interface is crucial to guarantee *P_BNDC* property as without it an high level user could block the monitor process indefinitely, by never accepting the response of a read request (output value) leading to an indirect information flow.

[2] Note that the system is specified using a value-passing extension of SPA. We will briefly explain its translation to the "pure" calculus in the following.

$$
\begin{aligned}
Access\_Monitor &= (AM \mid Interf) \setminus N \\
AM &= (Monitor \mid Object(1,0) \mid Object(0,0)) \setminus L \\
Monitor &= access\_r(l,x). \\
&\quad (\ \textbf{if}\ x \leq l\ \textbf{then}\ r(x,y).\overline{val}(l,y).Monitor \\
&\quad\ \ \textbf{else}\ \overline{val}(l, err).Monitor) \\
&\quad + \\
&\quad access\_w(l,x,z). \\
&\quad (\ \textbf{if}\ x \geq l\ \textbf{then}\ \overline{w}(x,z).Monitor \\
&\quad\ \ \textbf{else}\ Monitor) \\
Object(x,y) &= \overline{r}(x,y).Object(x,y) + w(x,z).Object(x,z) \\
Interf &= Interf(0) \mid Interf(1) \\
Interf(l) &= a\_r(l,x).\overline{access\_r}(l,x).val(l,k).\overline{put}(l,k).Interf(l) \\
&\quad + \\
&\quad a\_w(l,x,z).\overline{access\_w}(l,x,z).Interf(l)
\end{aligned}
$$

Fig. 5. The *Access_Monitor* System.

In order to understand how the system works, let us consider the following transitions sequence representing the writing of value 1 in the low level object, performed by the low level user:

$$
\begin{aligned}
&(AM \mid Interf(0) \mid Interf(1)) \setminus N \\
\xrightarrow{a\_w(0,0,1)}\ &(AM \mid \overline{access\_w}(0,0,1).Interf(0) \mid Interf(1)) \setminus N \\
\xrightarrow{\tau}\ &((\overline{w}(0,1).Monitor \mid Object(1,0) \mid Object(0,0)) \setminus L \mid Interf) \setminus N \\
\xrightarrow{\tau}\ &((Monitor \mid Object(1,0) \mid Object(0,1)) \setminus L \mid Interf) \setminus N
\end{aligned}
$$

The trace corresponding to this sequence of transitions is

$$a\_w(0,0,1)$$

and so we can write:

$$
\begin{aligned}
&(AM \mid Interf(0) \mid Interf(1)) \setminus N \\
\stackrel{a\_w(0,0,1)}{\Longrightarrow}\ &((Monitor \mid Object(1,0) \mid Object(0,1)) \setminus L \mid Interf) \setminus N
\end{aligned}
$$

Note that, after the execution of the trace, the low level object contains value 1.

Access_Monitor is a value passing specification of an access monitor. Its translation into pure SPA is reported and described in detail in [5, 6]. The idea is to translate each possible combination of the values into different SPA actions and different SPA processes. As an example, here we provide the translation of $Object(x, y)$ into the pure calculus by means of the following four constant definitions:

$$\begin{aligned}
Object_{00} &= \overline{r}_{00}.Object_{00} + w_{00}.Object_{00} + w_{01}.Object_{01}\\
Object_{01} &= \overline{r}_{01}.Object_{01} + w_{00}.Object_{00} + w_{01}.Object_{01}\\
Object_{10} &= \overline{r}_{10}.Object_{10} + w_{10}.Object_{10} + w_{11}.Object_{11}\\
Object_{11} &= \overline{r}_{11}.Object_{11} + w_{10}.Object_{10} + w_{11}.Object_{11}
\end{aligned}$$

Note that we have, for every possible value of the pair (x, y), one different process $Object_{xy}$ and two different actions $\overline{r}_{xy}$ and w_{xy}.

5 Conclusion

In this paper we studied a security property, named *Persistent_BNDC*, which is based on the idea of Non-Interference and it is suitable to deal with processes in dynamic environments. We characterized *P_BNDC* through a local property (with no quantification on the states and on the hostile contexts) by using a new definition of *weak bisimulation up to high level actions*, denoted by $\approx_{\setminus Act_H}$.

This result is interesting since it allows us to apply the model-checking technique for verifying *P_BNDC*. In fact, using the well-known greatest fixpoint characterization of bisimulation-like relations [11], one can easily derive modal mu-calculus formulae characterizing finite-state processes up to the equivalence relation $\approx_{\setminus Act_H}$. This allows us to employ model checkers as *P_BNDC* checkers. Indeed, if $\phi^{\approx_{\setminus Act_H}}$ is a characteristic formulae for a finite state process E up to $\approx_{\setminus Act_H}$, then $E \in P\_BNDC$ if and only if $E \setminus Act_H \models \phi^{\approx_{\setminus Act_H}}$ [17, 18].

Actually, the compositional security checker described in [5] provides an automatic tool for verifying *P_BNDC* over finite state processes: this is done by checking *SBSNNI* that requires to verify a bisimulation property over all the possible reachable states. The results presented in this paper show us how we may improve the *P_BNDC* checker by extending our compositional checker in order to deal with the novel bisimulation-like equivalence relation $\approx_{\setminus Act_H}$. We plan to work on this subject in the next future.

Finally, we observe that *P_BNDC* is a property suitable for reasoning on security of processes in precence of mobility. However the language upon which we study such a property is not exactly a language for mobility. Thus the next step will be to study *P_BNDC* (or similar properties) when more suitable languages for dealing with mobility are considered.

Acknowledgements. We would like to thank Fabio Martinelli for the helpful discussions on the *BNDC* property.

References

1. M. Abadi. Secrecy by Typing in Security Protocols. *Journal of ACM*, 46(5):749–786, 1999.

2. D. E. Bell and L. J. La Padula. 'Secure Computer Systems: Unified Exposition and Multics Interpretation". *ESD-TR-75-306, MITRE MTR-2997*, March 1976.
3. C. Bodei, P. Degano, R. Focardi, and C. Priami. Primitives for Authentication in Process Algebras. *Theoretical Computer Science*, 2001. To appear.
4. N. A. Durgin, J. C. Mitchell, and D. Pavlovic. A Compositional Logic for Protocol Correctness. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, June 2001.
5. 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.
6. 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 *Lecture Notes in Computer Science*. Springer-Verlag, 2001.
7. R. Focardi, R. Gorrieri, and F. Martinelli. Non Interference for the Analysis of Cryptographic Protocols. In *Proceedings of ICALP'00*, pages 744–755. LNCS 1853, July 2000.
8. R. Focardi and F. Martinelli. A Uniform Approach for the Definition of Security Properties. In *Proceedings of World Congress on Formal Methods (FM'99)*, pages 794–813. Springer, LNCS 1708, 1999.
9. J. A. Goguen and J. Meseguer. Security Policy and Security Models. In *Proceedings of the 1982 Symposium on Security and Privacy*, pages 11–20. IEEE Computer Society Press, April 1982.
10. R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
11. M. Müller-Olm. Derivation of Characteristic Formulae. *Electronic Notes in Theoretical Computer Science*, 18, 1998.
12. L. C. Paulson. Proving Properties of Security Protocols by Induction. In *Proceedings of the 10th Computer Security Foundations Workshop*, pages 70–83. IEEE Computer Society Press, 1997.
13. A. Sabelfeld and D. Sands. Probabilistic Noninterference for Multi-threaded Programs. In *Proceedings of the IEEE Computer Security Foundations Workshop*, pages 200–215. IEEE Computer Society Press, July 2000.
14. S. Schneider. Verifying authentication protocols in CSP. *IEEE Transactions on Software Engineering*, 24(9), September 1998.
15. V. Shmatikov and J. C. Mitchell. Analysis of a Fair Exchange Protocol. In *Seventh Annual Symposium on Network and Distributed System Security (NDSS 2000)*, pages 119–128. Internet Society, 2000.
16. G. Smith and D. M. Volpano. Secure Information Flow in a Multi-Threaded Imperative Language. In *Proc. of the 25th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 355–364. ACM Press, 1998.
17. B. Steffen and A. Ingòlfsdòttir. Characteristic Formulae for Processes with Divergence. *Information and Computation*, 110(1):149–163, 1994.
18. C. Stirling. Modal and Temporal Logics for Processes. In E. Brinksma, R. Cleaveland, K. G. Larsen, T. Margaria, and B. Steffen, editors, *Logics for Concurrency: Structures versus Automata*, volume 1043 of *Lecture Notes in Computer Science*, pages 149–237. Springer-Verlag, 1996.
19. D. M. Volpano and G. Smith. Probabilistic Noninterference in a Concurrent Language. *Journal of Computer Security*, 7(1), 1999.