# A Proof System for 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. Intuitively, *P_BNDC* requires that *high level* interactions never interfere with the *low level* behavior of the system, in every possible state. *P_BNDC* is verified by checking whether the system interacting with a high level component is bisimilar or not to the system in isolation. In this work we contribute to the verification of information-flow security in two respects: (*i*) we give an *unwinding* condition that allows us to express *P_BNDC* in terms of a local property on high level actions and (*ii*) we exploit this local property in order to define a proof system which provides a very efficient technique for the development and the verification of *P_BNDC* processes.

## 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, where an application running in a "secure way" inside one environment could enter an "insecure state" while moving to a different environment. In this setting, security properties should correctly deal with such a dynamic nature of executions.

A number of formal definitions of security properties (see, for instance, [1, 8, 10, 15, 18, 19, 23, 26–28]) has been proposed in the literature. *Persistent_BNDC* (*P_BNDC*, for short), proposed in [11], 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 [12, 25, 28] (formalized as *BNDC* [10]) and requires that every state which is reachable by the system still satisfies a basic Non-Interference property. If this holds, one is 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 [11] it has been proved that *P_BNDC* may

* 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).

be verified by checking whether the system interacting with a high level component is *behaviorally equivalent* or not to the system in isolation, where behavioral equivalence is defined in terms of a suitable notion of weak bisimulation [1]. Moreover, in [3] it has been shown that *P_BNDC* may also be verified by checking whether the system is weakly bisimilar to a rectification of the system itself, which makes it *P_BNDC*. Both of these techniques can be fully automatized if the *labelled transition system*, i.e., the automata representing the operational behavior of the considered system, is composed of a finite number of states. In particular, there exist efficient algorithms for checking bisimulation equivalences (see, e.g., [4, 14, 22, 7]) which are polynomial with respect to the number of states and transitions of the underlying transition system. However this kind of *behavioral verification* often suffers of the so-called state-explosion problem, i.e., the number of states increases exponentially with respect to the degree of parallelism inside the considered system. The reason is that every interleaving among parallel processes needs to be represented.

In this work we contribute to the verification of information-flow security in two respects: (*i*) we give an *unwinding* condition that allows us to express *P_BNDC* in terms of a local property of high level actions and (*ii*) we exploit this local property in order to define a proof system which provides a very efficient technique for the development and verification of *P_BNDC* processes.

The unwinding condition, similar to other already proposed in different settings (see, e.g., [16, 18, 20, 24]), requires that every high level event is "simulable" by a sequence of internal moves, i.e, that every time a high level event is performed moving the system to a state $E'$, a state $E''$ is also reachable (through internal computation) which is equivalent to $E'$ from a low level point of view, written $E' \setminus H \approx E'' \setminus H$. Intuitively, if this holds no high level event $h$ should be observable by a low level user, as there always exists a low-level equivalent state that the system may reach without performing $h$. We prove that this local property is a necessary and sufficient condition for *P_BNDC*.

As noticed in [16], unwinding conditions are useful for giving efficient proof techniques. Indeed, we use our local characterization to define a proof system which allows us to *statically prove* that a process is *P_BNDC*, i.e., by just inspecting its syntax. State-explosion is avoided by exploiting the compositionality of *P_BNDC* with respect to the parallel operator which is the source of the exponential growing of the number of states in a system. Moreover, the system offers a mean to build processes which are *P_BNDC* by construction in an incremental way. Our proof system extends the one given in [17] for finite processes, i.e., processes that may only perform finite sequences of actions. In particular, we are able to deal also with recursive processes which may perform unbounded sequences of actions. To illustrate the effectiveness of the new technique, we apply the proof system to the small, but non-trivial, example of an access monitor also considered in [10].

[1] In [10], it is shown that bisimulation-based properties are able to detect potential flows due to deadlocks caused by high level activity. Such flows are not revealed by simply observing traces, i.e., execution sequences.

The paper is organized as follows. In Section 2 we present some basic notions on the *SPA* language. In Section 3 we recall the *P_BNDC* property and we give the new *unwinding* condition. In Sections 4 and 5 we introduce our new proof system and in 6 we illustrate the usefulness of it on a simple example. Finally, in Section 7 we draw some conclusions. All proofs are collected in the Appendix.

## 2 Basic Notions: the SPA Language

In this section we report from [10] the syntax and semantics of the *Security Process Algebra*. The *Security Process Algebra* (SPA, for short) [10] is a variation of Milner's CCS [21], 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 special action $\tau$ 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 *processes* is defined by

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

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

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

Given a fixed language $\mathcal{L}$ we denote by $\mathcal{E}$ the set of all SPA processes, by $\mathcal{E}_H$ the set of all high level processes, i.e., those constructed over $H \cup \{\tau\}$, and by $\mathcal{E}_L$ the set of all low level processes, i.e., those constructed over $L \cup \{\tau\}$,

The operational semantics of SPA processes is given in terms of *Labelled Transition Systems* (LTS). A LTS is a triple $(S, A, \to)$ where $S$ is a set of states, $A$ is a set of labels (actions), $\to \subseteq S \times A \times S$ is a set of labelled transitions. The notation $(S_1, a, S_2) \in \to$ (or equivalently $S_1 \stackrel{a}{\to} S_2$) means that the system can move from the state $S_1$ to the state $S_2$ through the action $a$. 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 Figure 1. The operational semantics for a process $E$ is the subpart of the SPA LTS reachable from the initial state and we refer to it as $LTS(E) = (S_E, Act, \to)$.

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

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

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

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

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

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

**Fig. 1.** The operational rules for SPA

In the paper we use the following notations. If $t = a_1 \cdots a_n \in Act^*$ and $E \xrightarrow{a_1} \cdots \xrightarrow{a_n} E'$, then we say that $E'$ is reachable from $E$ and write $E \xrightarrow{t} E'$, or simply $E \rightsquigarrow E'$. 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 $\tau$ labelled transitions. If $t \in Act^*$, then $\hat{t} \in \mathcal{L}^*$ is the sequence gained by deleting all occurrences of $\tau$ from $t$. As a consequence, $E \stackrel{\hat{a}}{\Longrightarrow} E'$ stands for $E \stackrel{a}{\Longrightarrow} E'$ if $a \in \mathcal{L}$, and for $E(\xrightarrow{\tau})^* E'$ if $a = \tau$ (note that $\stackrel{\tau}{\Longrightarrow}$ requires at least one $\tau$ labelled transition while $\stackrel{\hat{\tau}}{\Longrightarrow}$ means zero or more $\tau$ labelled transitions). Moreover, we say that a process $E$ is *closed* if it does not contain constants. Given two processes $E, F$ we write $E \equiv F$ when $E$ and $F$ are syntactically equal.

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 $\mathcal{E}$. We report here the definition of two observational equivalences: *strong bisimulation* and *weak bisimulation* [21]. Intuitively, strong bisimulation equates two processes if they mutually simulate their behavior step by step.

**Definition 1 (Strong Bisimulation).** *A binary relation $\mathcal{R} \subseteq \mathcal{E} \times \mathcal{E}$ over processes is a* strong bisimulation *if $(E, F) \in \mathcal{R}$ implies, for all $a \in Act$,*

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

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

Relation $\sim$ is the largest strong bisimulation and is an equivalence relation [21].

Weak bisimulation is similar to strong bisimulation but it does not care about internal $\tau$ actions. So, when $P$ simulates an action of $Q$, it can also execute some $\tau$ actions before or after that action.

**Definition 2 (Weak Bisimulation).** *A binary relation $\mathcal{R} \subseteq \mathcal{E} \times \mathcal{E}$ over processes is a* weak bisimulation *if $(E, F) \in \mathcal{R}$ implies, for all $a \in Act$,*

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

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

Relation $\approx$ is the largest weak bisimulation and is an equivalence relation. Moreover, $\sim \subseteq \approx$ [21].

We use the notation $E[Y := X]$ to denote the process obtained by replacing in the process $E$ the constant $Y$ with the constant $X$. The following lemma provides us a syntactic way to determine when two constants are strong bisimilar.

**Lemma 1.** *Let $X, Y$ be two constants defined by $X \stackrel{\text{def}}{=} E$ and $Y \stackrel{\text{def}}{=} F$. If $E[Y := X] \equiv F[Y := X]$ then $E \sim F$.*

## 3 The *P_BNDC* Security Property

We first recall from [11] the *Persistent Bisimulation-based Non Deducibility on Compositions* (*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) [10]. 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 are apparently honest but hide inside some malicious code.

Property *BNDC* is based on the idea of checking a system against all high level potential interactions, representing all possible high malicious programs. A system $E$ is *BNDC* if for every high process $\Pi$ a low user cannot distinguish $E$ from $(E|\Pi)$, i.e., if $\Pi$ cannot interfere [12] with the low level execution of $E$.

**Definition 3 (BNDC).** *Let $E \in \mathcal{E}$ be a process.*

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

In [11] it is shown that *BNDC* is not strong enough for systems in dynamic execution environments. To deal with these situations, the property *P_BNDC* is introduced. Intuitively, a system $E$ is *P_BNDC* if it never reaches insecure states.

**Definition 4 (Persistent_BNDC).** *Let $E \in \mathcal{E}$ be a process.*

$$E \in P\_BNDC \ \ \textit{iff} \ \ E \rightsquigarrow E' \ \textit{implies} \ E' \in BNDC.$$

*Example 1.* Consider the process $E_1 \equiv l.h.j.\mathbf{0} + l.(\tau.j.\mathbf{0} + \tau.\mathbf{0})$ 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: $E_2 \equiv l.(h.j.\mathbf{0}+\tau.j.\mathbf{0}+\tau.\mathbf{0})+l.(\tau.j.\mathbf{0}+\tau.\mathbf{0})$. It may be proved that $E_2$ is $P\_BNDC$. Note that, from this example it follows that $P\_BNDC \subset BNDC$.

In [11] it has been proven that the property $P\_BNDC$ is equivalent to the security property $SBSNNI$ [9, 10] which is automatically checkable over finite-state processes. However, this property still requires a universal quantification over all the possible reachable states from the initial process $E$. In [11] it has been shown that this can be avoided, by including the idea of "being secure in every state" inside the bisimulation equivalence notion. This is done by defining an equivalence notion which just focus on observable actions not belonging to $H$.

In the following we propose another characterization of $P\_BNDC$ processes which allows us to express $P\_BNDC$ in terms of a local property of high level actions. This characterization recalls the unwinding conditions proposed in other settings (e.g., [16, 18, 20, 24]). In [16] it is shown how unwinding conditions can be used for the verification of security properties. Here we use our characterization to prove the correctness of the proof system defined in the next sections.

**Theorem 1.** *Let $E \in \mathcal{E}$ be a process.*

$$E \in P\_BNDC$$
$$\textit{iff}$$
$$\textit{if } E \rightsquigarrow E_i \xrightarrow{h} E_j, \textit{ then } E_i \stackrel{\hat{\tau}}{\Longrightarrow} E_k \textit{ and } E_j \setminus H \approx E_k \setminus H.$$

The class of $P\_BNDC$ processes enjoys the compositional properties below.

**Lemma 2 (Compositionality).** *The following properties hold:*

1. *if $E$ is a closed process in $\mathcal{E}_L$, then $E \in P\_BNDC$;*
2. *if $E$ is a closed process in $\mathcal{E}_H$, then $E \in P\_BNDC$;*
3. *if $E \in P\_BNDC$, then $E \setminus v \in P\_BNDC$;*
4. *if $E \in P\_BNDC$, then $E[f] \in P\_BNDC$;*
5. *if $E, F \in P\_BNDC$, then $E|F \in P\_BNDC$;*
6. *if $E_i, F_j \in P\_BNDC$, $i \in I$ and $j \in J$, then $\sum_{i\in I} a_i.E_i + \sum_{j\in J}(h_j.F_j + \tau.F_j) \in P\_BNDC$, where $a_i \in L$ and $h_j \in H$;*
7. *if $E \in P\_BNDC$ and $X \stackrel{\text{def}}{=} E$, then $X \in P\_BNDC$.*

## 4 Hypothetical *P_BNDC* Processes

In this section we develop a proof system which allows us to build $P\_BNDC$ processes in an incremental way. It is composed by a set of rules whose conclusion is in the form $E \in \mathcal{HP}[A]$, where $A$ is a set of constants. The intended meaning of the judgment is that $E$ is a $P\_BNDC$ process provided that all the constants in $A$ are $P\_BNDC$. The set $A$ plays the role of a set of assumptions: if it is empty then $E$ is $P\_BNDC$ otherwise we are still working on our construction under open hypothesis.

**Definition 5** ($\mathcal{HP}[A]$)**.** *Let $A$ be a set of constants and $E$ be a SPA process where some of the constants in $A$ may occur. We say that $E$ is $P\_BNDC$* under the hypothesis in $A$*, denoted by $E \in \mathcal{HP}[A]$, if $E \in P\_BNDC$ provided that all the constants in $A$ are $P\_BNDC$.*

*Example 2.* Let $a, b \in L$ and let $E \equiv a.X + b.Y$. It holds that $E \in \mathcal{HP}[\{X, Y\}]$, since if $X$ and $Y$ are $P\_BNDC$, then so is $a.X + b.Y$.

The rules in our proof system are suggested by the compositional properties of $P\_BNDC$ (see Lemma 2).

**Definition 6 (Core).** Core *is the proof system containing the following rules.*

$$\frac{}{P \in \mathcal{HP}[\emptyset]} \quad P \in \mathcal{E}_L,\ P \text{ is closed} \quad (Low)$$

$$\frac{}{P \in \mathcal{HP}[\emptyset]} \quad P \in \mathcal{E}_H,\ P \text{ is closed} \quad (High)$$

$$\frac{}{X \in \mathcal{HP}[\{X\}]} \quad X \text{ is a constant} \quad (Const)$$

$$\frac{E \in \mathcal{HP}[A]}{E \setminus v \in \mathcal{HP}[A]} \quad (Rest)$$

$$\frac{E \in \mathcal{HP}[A]}{E[f] \in \mathcal{HP}[A]} \quad (Label)$$

$$\frac{E \in \mathcal{HP}[A] \quad F \in \mathcal{HP}[B]}{E|F \in \mathcal{HP}[A \cup B]} \quad (Par)$$

$$\frac{E_i \in \mathcal{HP}[A_i] \quad F_j \in \mathcal{HP}[B_j]}{\sum_{i \in I} a_i.E_i + \sum_{j \in J}(h_j.F_j + \tau.F_j) \in \mathcal{HP}[\cup_{i \in I} A_i \cup \cup_{j \in J} B_j]} \quad a_i \in L \cup \{\tau\}, h_j \in H \quad (Choice)$$

$$\frac{E \in \mathcal{HP}[A]}{X \in \mathcal{HP}[A]} \quad X \stackrel{\text{def}}{=} E \quad (Def)$$

**Theorem 2 (Correctness).** *The system Core is correct, i.e., if there exists a proof in Core which ends with $E \in \mathcal{HP}[A]$, then $E$ is* P_BNDC *provided that all the constants in $A$ are* P_BNDC.

**Corollary 1.** *If there exists a proof of $E \in \mathcal{HP}[\emptyset]$ in Core, then $E$ is P_BNDC.*

Notice that the system Core is not complete. One reason is that the rule $(Choice)$ treats only some specific situations suggested by our characterization of Theorem 1 which can be detrmined by simple syntactic tests: for instance, the case that $E \xrightarrow{\tau} F_j$ holds whenever $E \xrightarrow{h} F_j$ holds. We could strengthen the rule by adding more complex tests based on bisimulation, but our purpose is to have a proof system whose rules are completely syntactic. Note that this is not so restrictive in the synthesis of $P\_BNDC$ processes, while in the case of verification it is not difficult to perform ad hoc modifications of rule $(Choice)$. A second source of incompleteness comes from the lack of rules for systems of definitions which are necessary to define recursive processes. We will treat this case in the next section.

In order to derive that a process is $P\_BNDC$ by using Core we have to use processes for which we are able to prove that they are in $\mathcal{HP}[A]$ and then provide $P\_BNDC$ definitions for the constants in $A$.

*Example 3.* Let $a, b \in L$ and $h \in H$. The following derivation in Core

$$\dfrac{\dfrac{\dfrac{\dfrac{}{b.\mathbf{0} \in \mathcal{HP}[\emptyset]}\,(Low)}{h.b.\mathbf{0} + \tau.b.\mathbf{0} \in \mathcal{HP}[\emptyset]}\,(Choice)}{a.(h.b.\mathbf{0} + \tau.b.\mathbf{0}) \in \mathcal{HP}[\emptyset]}\,(Choice) \qquad \dfrac{}{a.\mathbf{0} \in \mathcal{HP}[\emptyset]}\,(Low)}{a.(h.b.\mathbf{0} + \tau.b.\mathbf{0})|a.\mathbf{0} \in \mathcal{HP}[\emptyset]}\,(Par)$$

proves that $a.(h.b.\mathbf{0} + \tau.b.\mathbf{0})|a.\mathbf{0}$ is $P\_BNDC$. While the derivation below

$$\dfrac{\dfrac{\dfrac{}{X \in \mathcal{HP}[\{X\}]}\,(Var)}{a.X \in \mathcal{HP}[\{X\}]}\,(Choice) \qquad \dfrac{}{b.\mathbf{0} \in \mathcal{HP}[\emptyset]}\,(Low)}{a.X|b.\mathbf{0} \in \mathcal{HP}[\{X\}]}\,(Par)$$

proves that $E \equiv a.X|b.\mathbf{0}$ is in $\mathcal{HP}[\{X\}]$, which means that whenever we provide a proof of the fact that $X$ is $P\_BNDC$ we obtain that $E$ is $P\_BNDC$.

In Core there is no way to eliminate the hypothesis in the recursive definitions. If $X$ is a constant which has a definition $X \stackrel{\text{def}}{=} E$, and $X$ occurs in $E$, then we are only able to prove that $X \in \mathcal{HP}[X]$, i.e., $X$ is $P\_BNDC$ if $X$ is $P\_BNDC$. We will provide a more powerful system in the next section.

## 5 Systems of Definitions

It is possible to associate to a constant $X$ a definition $X \stackrel{\text{def}}{=} E$ where $E$ may possibly contain $X$ as well as other constants. When we have a set of definitions

$$\{Z_k \stackrel{\text{def}}{=} E_k \mid k \in K\}$$

which mutually depend on each other we call this set *system of definitions*. We consider only systems of definitions in which there is at most one definition for each constant occurring in the system. A system of definitions is *weakly guarded* if all the constants $Z_k$, $k \in K$, occur only within some subexpression of the form $a.F$. As an example, $Z = Z$ is not weakly guarded. In this paper we restrict to this class of systems of definitions since a weakly guarded system of definitions uniquely defines, up to strong bisimulation, a process (see [21]). Given a system of definitions $S = \{Z_k \stackrel{\text{def}}{=} E_k\}_{k \in K}$ we denote by $Const(S)$ the set $\{Z_k \mid k \in K\}$.

We have to pay attention to the transformations we apply to a system of definitions, in order to avoid indesiderable effects. For instance, if we substitute a subexpression of $E_k$ with a weakly bisimilar one we may not obtain a weakly bisimilar constant. Consider the system $\{X \stackrel{\text{def}}{=} a.X + \tau.Y; Y \stackrel{\text{def}}{=} b.Y + c.Y\}$ and replace $\tau.Y$ with $Y$ obtaining the transformed system $\{X \stackrel{\text{def}}{=} a.X + Y; Y \stackrel{\text{def}}{=} b.Y + c.Y\}$. In the first system $X$ can reach, by a $\tau$ move, a state which does not allows $a$ moves. This cannot be simulated (even weakly) by the constant $X$ in the second system. Hence the constants defined by the two systems are not bisimilar. Nevertheless, there are transformations which preserve weak bisimulation.

**Lemma 3.** *Let $X \stackrel{\text{def}}{=} \sum_{i \in I} a_i.E_i$ be a definition and $F \approx E_j$, for some $j \in I$. Let $Y \stackrel{\text{def}}{=} \sum_{i \in I} a_i.E'_i$ be a new definition where $E'_i \equiv E_i$ for all $i \in I, i \neq j$ and $E'_i \equiv F$ for $i = j$. Then $X \approx Y$.*

Next we introduce a transformation on processes which is at the basis of the syntactic conditions in the rule we are going to define on systems of definitions. In practice given a process $E$ our transformation maps $E$ into $E_{\backslash v}$, which is a sort of canonical form of $E \setminus v$, i.e., a process strong bisimilar to $E \setminus v$.

**Definition 7 ($E_{\backslash v}$).** *Let $v \subseteq \mathcal{L}$ and $E \in \mathcal{E}$. We define the process $E_{\backslash v}$ by induction on the structure of $E$.*

- *$E \equiv X$: $E_{\backslash v} \equiv X \setminus v$;*
- *$E \equiv a.E'$: If $a \in v$ then $E_{\backslash v} \equiv \mathbf{0}$ else $E_{\backslash v} \equiv a.E'_{\backslash v}$;*
- *$E \equiv E' + E''$: If $E'_{\backslash v} \equiv \mathbf{0}$ then $E_{\backslash v} \equiv E''_{\backslash v}$ else if $E''_{\backslash v} \equiv \mathbf{0}$ then $E_{\backslash v} \equiv E'_{\backslash v}$ else $E_{\backslash v} \equiv E'_{\backslash v} + E''_{\backslash v}$;*
- *$E \equiv E'|E''$: $E_{\backslash v} \equiv (E'|E'') \setminus v$;*
- *$E \equiv E' \setminus w$: $E_{\backslash v} \equiv (E'_{\backslash v}) \setminus w$;*
- *$E \equiv E'[f]$: $E_{\backslash v} \equiv (E'[f]) \setminus v$.*

**Lemma 4.** *Let $v \subseteq \mathcal{L}$ and $E \in \mathcal{E}$. It holds that $E_{\backslash v} \sim E \setminus v$.*

*Example 4.* Consider the expression $E \equiv a.X + h.b.Y + \tau.Y$. Let $v$ be such that $a, b \notin v$ and $h \in v$. We obtain $E_{\backslash v} \equiv a.(X \setminus v) + \tau.(Y \setminus v)$.

*Example 5.* Consider the system

$$\begin{cases} X \stackrel{\text{def}}{=} h.(h.X + \tau.X) + a.Y \\ Y \stackrel{\text{def}}{=} h.Y + \tau.Y \end{cases}$$

where $H = \{h\}$. The constant $X$ reaches with a high action $E \equiv h.X + \tau.X$. Moreover $E_{\backslash H} \equiv \tau.(X \setminus H)$, which implies $E \setminus H \approx \tau.X \approx X$. Since $X$ reaches $X$ with zero $\tau$ moves, the high transition which leads to $X$ cannot cause problems, hence we would like to prove that this system defines *P_BNDC* processes.

The following lemma allows us to syntactically determine when two constants are such that $X \setminus H \approx Y \setminus H$. In this case, if $X$ reaches $Y$ with a high transition, we do not have security problems since $X$ reaches with zero $\tau$ transition $X$.

**Lemma 5.** *Let $X, Y$ be two constants defined by $X \stackrel{\text{def}}{=} E$ and $Y \stackrel{\text{def}}{=} F$. If $(E_{\backslash v})[Y := X] \equiv (F_{\backslash v})[Y := X]$ then $X \backslash v \sim Y \backslash v$.*

The novel characterization of *P_BNDC* stated in Theorem 1 together with Lemma 5 indicate to us some cases in which a high level transition out-coming from a variable $X$ does not compromise the security of the system. This cases are captured by the notion of $safe(X, S)$ introduced in the definition below. The intuitive meaning of the set $safe(X, S)$ is that if $F \in safe(X, S)$, then $F$ can be safely reached by $X$ with a high transition.

**Definition 8** ($safe(Z_k, S)$)**.** *Let $S = \{Z_k \stackrel{\text{def}}{=} E_k\}_{k \in K}$ be a system of definitions. For each $k \in K$ we define the set $safe(Z_k, S) = \cup_{i=1}^{4} safe_i(Z_k, S)$ where*

$$\begin{aligned} safe_1(Z_k, S) &= \{F \mid Z_k \stackrel{\hat{\tau}}{\Longrightarrow} F\} \\ safe_2(Z_k, S) &= \{F \mid F_{\backslash H} \equiv Z_{k \backslash H}\} \\ safe_3(Z_k, S) &= \{F \mid F_{\backslash H} \equiv \tau.Z_{k \backslash H}\} \\ safe_4(Z_k, S) &= \{Z_j \mid E_{j \backslash H}[Z_k := Z_j] \equiv E_{k \backslash H}[Z_k := Z_j]\}. \end{aligned}$$

*Example 6.* Let $a, b \in L$ and $h \in H$. Consider the system $S$:

$$\begin{cases} X = h.Y + \tau.Z \\ Y = a.Y \\ Z = \tau.Y \\ W = a.(h.b.X + \tau.b.X) + b.Y + h.W \\ V = a.V + h.Y \end{cases}$$

We have that $Y \in safe_1(X, S)$, $W \in safe_2(W, S)$ (and also $W \in safe_1(W, S)$), and $Y \in safe_4(V, S)$.

Consider now the system $S'$ of Example 5, in this case we have that $(h.X + \tau.X) \in safe_3(X, S')$.

The following lemma is useful to prove the main result of this section, i.e., to characterize *syntactically safe* systems.

**Lemma 6.** *Let $E \in \mathcal{HP}[\emptyset]$ be derived in Core. If $E \rightsquigarrow E' \xrightarrow{h} E''$, then $E' \stackrel{\hat{\tau}}{\Longrightarrow} E'''$ and $E'' \setminus H \approx E''' \setminus H$.*

*Let $E \in \mathcal{HP}[A]$ be derived in Core without applying the rule $(Par)$. If $E \rightsquigarrow E' \xrightarrow{h} E''$ without using the definitions of the constants in $A$, then $E' \stackrel{\hat{\tau}}{\Longrightarrow} E'''$ and $E'' \setminus H \approx E''' \setminus H$.*

**Definition 9 (Safe system).** *Let $S = \{Z_k \stackrel{\text{def}}{=} E_k\}_{k \in K}$ be a system of definitions of the form*

$$E_k \equiv \sum_{i_k \in I_k} a_{i_k}.E_{i_k} + \sum_{j_k \in J_k} h_{j_k}.F_{j_k}.$$

*The system $S$ is said to be* safe *if and only if for each $j_k \in J_k$ it holds that $F_{j_k} \in safe(Z_k, S)$ and for each $G$ in $SubEx(S) = \cup_{k \in K}(\cup_{i_k \in I_k}\{E_{i_k}\} \cup \cup_{j_k \in J_k}\{F_{j_k}\})$ one of the following properties holds:*

- *Core proves $G \in \mathcal{HP}[\emptyset]$, or*
- *Core proves $G \in \mathcal{HP}[A_G]$ without applying the rule $(Par)$, for some set $A_G$.*

*We call the set $A = \cup_{G \in SubEx(S)} A_G$* safety set *of $S$ (notation: $Safety(S)$).*

*Example 7.* The system $S$

$$\begin{cases} X \stackrel{\text{def}}{=} a.X + \tau.(\tau.Y + a.Y) + h.Y \\ Y \stackrel{\text{def}}{=} h.Y + \tau.Z + a.(\tau.Z + h.Z) \end{cases}$$

is safe and its safety set is $\{X, Y, Z\}$. In fact:

- $X \xrightarrow{h} Y$ and $X \stackrel{\hat{\tau}}{\Longrightarrow} Y$, hence $Y \in safe(X, S)$;
- $Y \xrightarrow{h} Y$ and $Y \in safe(Y, S)$;
- $X$ and $(\tau.Y + a.Y)$ and $Y$ can be derived to be $\mathcal{HP}[\{X\}]$ and $\mathcal{HP}[\{Y\}]$ respectively without applying $(Par)$;
- $Z$ and $(\tau.Z + h.Z)$ can be derived to be $\mathcal{HP}[\{Z\}]$ without applying $(Par)$.

**Theorem 3.** *Let $S = \{Z_k \stackrel{\text{def}}{=} E_k\}_{k \in K}$ be a safe system of definitions with safety set $A$. Then for all $k \in K$ the constant $Z_k$ is in $\mathcal{HP}[A \setminus \{Z_{k'} \mid k' \in K\}]$.*

*Example 8.* Consider again the system of Example 7. By Theorem 3, both $X$ and $Y$ belongs to $\mathcal{HP}[\{Z\}]$.

Suppose now that we want to extend the system of definitions of example above by adding the definition $Z \stackrel{\text{def}}{=} \tau.X + a.Y + b.Z$. Since we already discarded the assumptions $X$ and $Y$, we would like to be able to deduce $Z \in \mathcal{HP}[\emptyset]$. We can do it if we extend the notion of safe system by relaxing the request that all the proofs are performed in Core and allow them to be carried on in any correct proof system for the judgenment $E \in \mathcal{HP}[A]$ which extends Core and satisfies Lemma 6.

**Definition 10 (SafeSys).** *Let SafeSys be the system of rules obtained by adding to Core the following rule (Sys):*

$$\frac{G_1 \in \mathcal{HP}[A_{G_1}] \cdots G_n \in \mathcal{HP}[A_{G_n}]}{Z \in \mathcal{HP}[Safety(S) \setminus Const(S)]} \; S \text{ safe}, Z \in Const(S), \{G_1 \ldots G_n\} = SubEx(S)$$

*where a system $S$ is* safe *if and only if it satisfies all the conditions of Definition 9 with Core replaced by SafeSys in the two items.*

**Theorem 4.** *If there exists a proof in SafeSys which ends with $E \in \mathcal{HP}[A]$, then $E$ is P_BNDC provided that the constants in $A$ are P_BNDC.*

*Example 9.* In this example we illustrate a simple derivation in the full system SafeSys. Let $a, b \in L$ and $h \in H$. Consider the systems

$$S_X = \{X \stackrel{\text{def}}{=} a.X\}$$
$$S_Y = \{Y \stackrel{\text{def}}{=} h.Y\}$$
$$S_Z = \{Z \stackrel{\text{def}}{=} h.Z + b.(X|Y)\}$$

In order to prove that $Z$ is $P\_BNDC$ we have to use three times the rule $(Sys)$.

$$\dfrac{\dfrac{\dfrac{}{Z \in \mathcal{HP}[\{Z\}]}\,(Const) \qquad \dfrac{\dfrac{\dfrac{}{X \in \mathcal{HP}[\{X\}]}\,(Const)}{X \in \mathcal{HP}[\emptyset]}\,S_X(Sys) \qquad \dfrac{\dfrac{}{Y \in \mathcal{HP}[\{Y\}]}\,(Const)}{Y \in \mathcal{HP}[\emptyset]}\,S_Y(Sys)}{X|Y \in \mathcal{HP}[\emptyset]}\,(Par)}{Z \in \mathcal{HP}[\emptyset]}\,S_Z(Sys)$$

## 6 Example: a Process Monitor

We consider the process *Access_Monitor* which has been widely discussed in [10]. It is defined as a process which handles read and write requests from high and low level users on two binary objects: a high level variable and a low level one. To avoid information flows from high to low, two access control rules are imposed: $(i)$ *no read up*: low level users cannot read from high level object; $(ii)$ *no write down*: high level users cannot write into low level object. As a consequence, low level users are allowed to write into both objects and read only from the low one; conversely, high level users can read from both objects and write only into the high one. As the objects are binary, there are only two values to read or write: 0 and 1. When an object receives a read request it returns its actual value and resets itself in the same state; when it processes a write request it moves into the corresponding state.

In [10], the authors develop different definitions for the process Monitor. The aim is finding a process which is *BNDC* and for which this property is easy to check. Here we show how it is easy to synthesize a *P_BNDC* Monitor in SafeSys.

Let $access\_read(u,x)$ and $access\_write(u,x,y)$ be the access requests of the user $u$ ($u = 0$ low, $u = 1$ high) for the object $x$ ($x = 0$ low, $x = 1$ high) and the value $y$, and $\overline{val(u,y)}$ defines the values returned to the user $u$, where $y \in \{0,1,err\}$. All the actions that involve high level users, i.e., the ones with $u = 1$ are considered high level ones.

In order to develop *Access_Monitor* we associate to each object $x$ a private monitor $Monitor(x)$ which handles the requests to the object $x$ in a secure way. As it is shown in [10], if we are able to build two *P_BNDC* processes realizing the two private monitors, we can then easily construct (by $(Par)$ and $(Rest)$ rules) a *P_BNDC* process realizing *Access_Monitor*.

Since each object has two possible values we have to define four processes: *M00* and *M01* defining $Monitor(0)$, *M10* and *M11* defining $Monitor(1)$. For sake of simplicity we indicate them by *Mxy*. To develop their (recursive) definitions, we first assume that all of them are *P_BNDC* and then we construct a safe system of definitions whose safety set contains exactly these assumptions.

We start by considering $Monitor(0)$ which handles the accesses to the low level object. For both of its components, there are six different possible requests, two *access_read*: $access\_read(u,0)$, $u \in \{0,1\}$, and four *access_write*: $access\_write(u,0,y)$, $u,y \in \{0,1\}$.

First we consider the requests from the low level users ($u = 0$). Since both read and write on the same level are allowed, the reaction of *M0y* are the natural ones: on a read request it returns the correct value ($y$) and on a write request it moves into the right state. In Core there are the derivations:

$$\frac{}{Mxy \in \mathcal{HP}[\{Mxy\}]}\ (Const) \qquad \frac{\dfrac{}{M0y \in \mathcal{HP}[\{M0y\}]}\ (Const)}{\overline{val(0,y)}.M0y \in \mathcal{HP}[\{M0y\}]}\ (Choice)$$

The requests from the high level user (high actions), need more care. Since high users cannot write down, the only possible reaction to the high requests $access\_write(1,0,z)$, $z \in \{0,1\}$ is a reset of the actual state. As regards the request $access\_read(1,0)$, a problem arises since the action $(\overline{val(1,y)})$ returning the value $y$ to the high level user is a high action and we cannot derive in Core the judgement $access\_read(1,0).\overline{val(1,y)}.M0y \in \mathcal{HP}[\{M0y\}]$, $y \in \{0,1\}$. Note that process $access\_read(1,0).\overline{val(1,y)}.M0y$ is potentially dangerous as a deadlock could be caused since no high level user is accepting the output action $\overline{val(1,y)}$ (see [10] for more detail on how this could be exploited to obtain an information flow from high to low). A possible solution is suggested in [3] where a lossy channel is introduced. Intuitively, the low level object sends the right value but its answer might be lost. This is represented by process $\overline{val(1,y)}.M0y + \tau.M0y$. Note that now no deadlock may be caused by high activity as it is always possible to reach *M0y* through an internal action. Now, in Core we can derive:

$$\frac{\dfrac{}{M0y \in \mathcal{HP}[\{M0y\}]}\ (Const)}{\overline{(val(1,y)}.M0y + \tau.M0y) \in \mathcal{HP}[\{M0y\}]}\ (Choice)$$

Summing up, to define $Monitor(0)$ we can introduce the system of definitions:

$$\begin{aligned}
M00 \stackrel{\text{def}}{=}\ & access_read(0,0).\overline{val(0,0)}.M00 \\
& + access_read(1,0).(\overline{val(1,0)}.M00 + \tau.M00) \\
& + access_write(0,0,0).M00 \\
& + access_write(0,0,1).M01 \\
& + access_write(1,0,0).M00 \\
& + access_write(1,0,1).M00
\end{aligned}$$

$$\begin{aligned}
M01 \stackrel{\text{def}}{=}\ & access_read(0,0).\overline{val(0,1)}.M01 \\
& + access_read(1,0).(\overline{val(1,1)}.M01 + \tau.M01) \\
& + access_write(0,0,0).M00 \\
& + access_write(0,0,1).M01 \\
& + access_write(1,0,0).M01 \\
& + access_write(1,0,1).M01
\end{aligned}$$

where each $G \in SubEx(Monitor(0))$ is derivable in Core without using $(Par)$.

In order to apply the rule $(Sys)$ we have to prove also that the system $Monitor(0)$ is safe. To this aim, we have to prove that $safe(M0y, Monitor(0))$ contains $M0y$ and $(\overline{val(1,0)}.M0y + \tau.M0y)$. Both statements holds since $M0y \stackrel{\hat{\tau}}{\Longrightarrow} M0y$ $(safe_1)$ and $(\overline{val(1,0)}.M0y + \tau.M0y)_{\setminus H} \equiv \tau.M0y$ $(safe_3)$. Hence, we can apply the rule $(Sys)$ to derive that both $M00$ and $M01$ are $P\_BNDC$.

The construction of the monitor for the high level object is similar to the one used to derive the system of definitions for $Monitor(0)$. It is easy to see that each subexpression in the right sides of the following system defining $Monitor(1)$ is derivable in Core without using $(Par)$.

$$\begin{aligned}
M10 \stackrel{\text{def}}{=}\ & access_read(1,1).(\overline{val(1,0)}.M10 + \tau.M10) \\
& + access_read(0,1).\overline{val(0,err)}.M10 \\
& + access_write(0,1,0).M10 \\
& + access_write(0,1,1).M11 \\
& + access_write(1,1,0).M10 \\
& + access_write(1,1,1).M11
\end{aligned}$$

$$\begin{aligned}
M11 \stackrel{\text{def}}{=}\ & access_read(1,1).(\overline{val(1,1)}.M11 + \tau.M11) \\
& + access_read(0,1).\overline{val(0,err)}.M11 \\
& + access_write(0,1,0).M10 \\
& + access_write(0,1,1).M11 \\
& + access_write(1,1,0).M10 \\
& + access_write(1,1,1).M11
\end{aligned}$$

As in the previous case we have to prove that the system is safe. To this aim we have to prove that: $safe(M1y, Monitor(1))$ contains $(\overline{val(1,0)}.M1y + \tau.M1y)$, $M1y$ and $M1z$, where $z = 1 - y$. The first two conditions can be treated exactly as in the previous case. To prove the third one we need to observe that if we

substitute $M11$ by $M10$ in both right sides of the two definitions and apply the $\backslash_H$ transformation we obtain in both sides the same term:

$$\begin{array}{l} access\_read(0,1).\overline{val(0,err)}.M10 \\ +access\_write(0,1,0).M10 \\ +access\_write(0,1,1).M10\,. \end{array}$$

Hence $M1z \in safe_4(M1y, Monitor(1))$, thus by $(Sys)$ we can derive that both $M10$ and $M11$ are $P\_BNDC$.

## 7 Related Works and Conclusion

In this paper we have proposed a new local characterization of $P\_BNDC$ and a proof system that allows us to efficiently construct and verify $P\_BNDC$ processes. We have shown the effectiveness of the new technique through the example of the Access Monitor.

It is worthwhile noticing that there are many other approaches to the verification of information flow properties. For instance, there are verification techniques for information flow security which are based on types (see, e.g., [28, 25, 13, 5]) and control flow analysis (see, e.g., [2, 6]). However, most of them are concerned with different models, e.g., trace semantics [15, 16, 18, 19].

In this paper we follow the approach of Focardi and Gorrieri [10] and focus on bisimulation based information flow properties. To the best of our knowledge, there is only another example of a proof system for security proposed by Martinelli in [17]. However, Martinelli's system deals only with finite processes. Our proof system extends [17] to the case of recursively defined processes. We avoid the state explosion problem by exploiting the compositionality results of $P\_BNDC$. Indeed, if a property is preserved when secure systems are composed, then the analysis may be performed on subsystems and, in case of success, the system as a whole can be proved to be secure (see also [8, 9, 19]).

## References


1. M. Abadi. Secrecy by Typing in Security Protocols. *Journal of the ACM*, 46(5):749–786, 1999.
2. C. Bodei, P. Degano, F. Nielson, and H. Nielson. Static analysis for the pi-calculus with applications to security. *Information and Computation*, 168(1):68–92, 2001.
3. A. Bossi, R. Focardi, C. Piazza, and S. Rossi. Transforming Processes to Check and Ensure Information Flow Security. In *Proc. of Int. Conf. on Algebraic Methodology and Software Technology*, LNCS. Springer, 2002. To appear.
4. A. Bouali and R. de Simone. Symbolic Bisimulation Minimization. In *Proc. of Computer Aided Verification*, volume 663 of *LNCS*, pages 96–108. Springer, 1992.
5. G. Boudol and I. Castellani. Non-Interference for Concurrent Programs. In *Proc. of Int. Colloquium on Automata, Languages and Programming*, volume 2076 of *LNCS*, pages 382–395. Springer, 2001.

6. C. Braghin, A. Cortesi, and R. Focardi. Control Flow Analysis of Mobile Ambients with Security Boundaries. In *Proc. of IFIPM Int. Conf. on Formal Methods for Open Object-Based Distributed Systems*, pages 197–212. Kluwer, 2002.
7. A. Dovier, C. Piazza, and A. Policriti. A Fast Bisimulation Algorithm. In *Proc. of Computer Aided Verification*, volume 2102 of *LNCS*, pages 79–90. Springer, 2001.
8. N. A. Durgin, J. C. Mitchell, and D. Pavlovic. A Compositional Logic for Protocol Correctness. In *Proc. of Computer Security Foundations Workshop*. IEEE, 2001.
9. R. Focardi and R. Gorrieri. The Compositional Security Checker: A Tool for the Verification of Information Flow Security Properties. *IEEE Transactions on Software Engineering*, 23(9):550–571, 1997.
10. 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.
11. R. Focardi and S. Rossi. Information Flow Security in Dynamic Contexts. In *Proc. of 15th Computer Security Foundations Workshop*, pages 307–319. IEEE, 2002.
12. J. A. Goguen and J. Meseguer. Security Policies and Security Models. In *Proc. of the IEEE Symp. on Security and Privacy*, pages 11–20. IEEE, 1982.
13. M. Hennessy and J. Riely. Information Flow vs. Resource Access in the Asynchronous Pi-Calculus. In *Proc. of Int. Colloquium on Automata, Languages and Programming (ICALP'00)*, volume 1853 of *LNCS*, pages 415–427. Springer, 2000.
14. D. Lee and M. Yannakakis. Online Minimization of Transition Systems. In *Proc. of 24th Symp. on Theory of Computing*, pages 264–274. ACM, 1992.
15. H. Mantel. Possibilistic Definitions of Security - An Assebly Kit -. In *Proc. of the IEEE Symp. on Security and Privacy*, pages 185–199. IEEE, 2000.
16. H. Mantel. Unwinding Possibilistic Security Properties. In *Proc. of European Symp. on Research in Computer Security*, volume 2895 of *LNCS*. Springer, 2000.
17. F. Martinelli. Partial Model Checking and Theorem Proving for Ensuring Security Properties. In *Proc. Computer Security Foundations Workshop*. IEEE, 1998.
18. D. McCullough. A Hookup Theorem for Multilevel Security. *IEEE Transactions on Software Engineering*, pages 563–568, June 1990.
19. J. McLean. A General Theory of Composition for Trace Sets Closed under Selective Interleaving Functions. In *Proc. Symp. on Security and Privacy*, pages 79–93, 1994.
20. J. K. Millen. Unwinding Forward Correctability. In *Proc. of 7th Computer Security Foundations Workshop*, pages 2–10. IEEE, 1994.
21. R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
22. R. Paige and R. E. Tarjan. Three Partition Refinement Algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.
23. L. C. Paulson. Proving Properties of Security Protocols by Induction. In *Proc. of 10th Computer Security Foundations Workshop*, pages 70–83. IEEE, 1997.
24. J. Rushby. Noninterference, Transitivity, and Channel-Control Security Policies. Technical Report Technical Report CSL-92-02, SRI International, December 1992.
25. A. Sabelfeld and D. Sands. Probabilistic Noninterference for Multi-threaded Programs. In *Proc. of Computer Security Foundations Workshop*. IEEE, 2000.
26. S. Schneider. Verifying Authentication Protocols in CSP. *IEEE Transactions on Software Engineering*, 24(9), 1998.
27. V. Shmatikov and J. C. Mitchell. Analysis of a Fair Exchange Protocol. In *Proc. of 7th Annual Symp. on Network and Distributed System Security*, pages 119–128. Internet Society, 2000.
28. G. Smith and D. M. Volpano. Secure Information Flow in a Multi-threaded Imperative Language. In *Proc. of 25th Symp. on Principles of Programming Languages*, pages 355–364. ACM, 1998.

## Appendix

This Appendix contains all proofs of the results presented in the paper.

*Proof of Lemma 1.* The proof of this lemma is done by transition induction, i.e., by induction on the length of the derivation of $E \stackrel{a}{\rightarrow} E'$ using the rules in Figure 1 (see [21]). Let

$$\mathcal{S} = \{(E', F') \mid E'[Y := X] \equiv F'[Y := X], \\ \text{where } X \stackrel{\text{def}}{=} E, Y \stackrel{\text{def}}{=} F \text{ and } E[Y := X] \equiv F[Y := X]\}.$$

We prove that $\mathcal{S}$ is a strong bisimulation.

Let $(E', F') \in \mathcal{S}$ and $E' \stackrel{a}{\rightarrow} E''$ we have to prove that there exists $F''$ such that $F' \stackrel{a}{\rightarrow} F''$ and $(E'', F'') \in \mathcal{S}$. The proof follows by transition induction on the inference $E' \stackrel{a}{\rightarrow} E''$.

- $E' \equiv a.E''$. Since $a.E''[Y := X] \equiv F'[Y := X]$, also $F'$ admits an $a$ transition $F' \stackrel{a}{\rightarrow} F''$ where $E''[Y := X] \equiv F''[Y := X]$. Then, $(E'', F'') \in \mathcal{S}$.
- $E' \equiv E'_1 + E'_2$. Assume $E'_1 \stackrel{a}{\rightarrow} E''$. Since $(E'_1 + E'_2)[Y := X] \equiv F'[Y := X]$, there exists $F'_1$ and $F'_2$ such that $F' = F'_1 + F'_2$ and $E'_1[Y := X] \equiv F'_1[Y := X]$. Then, by inductive hypothesis, there exists $F''$ such that $F'_1 \stackrel{a}{\rightarrow} F''$ and $(E'', F'') \in \mathcal{S}$.
- $E' \equiv E'_1 | E'_2$. As in the previous case, there exist $F'_1$ and $F'_2$ such that $F' = F'_1 | F'_2$, $E'_1[Y := X] \equiv F'_1[Y := X]$ and $E'_2[Y := X] \equiv F'_2[Y := X]$. We consider the case of synchronization, the other cases are similar and simpler. Assume $a = \tau$, $E'_1 \stackrel{b}{\rightarrow} E''_1$, $E'_2 \stackrel{\bar{b}}{\rightarrow} E''_2$ and $E'' \equiv E''_1 | E''_2$,. Then, by inductive hypothesis, there exist $F''_1, F''_2$ such that $F'_1 \stackrel{b}{\rightarrow} F''_1$, $F'_2 \stackrel{\bar{b}}{\rightarrow} F''_2$, $(E''_1, F''_1) \in \mathcal{S}$, $(E''_2, F''_2) \in \mathcal{S}$. Hence there exists $F'' \equiv F''_1 | F''_2$, such that $F' \stackrel{\tau}{\rightarrow} F''$ and $(E'', F'') \in \mathcal{S}$.
- $E' \equiv E_1 \backslash v$. Similar to the previous cases.
- $E' \equiv E_1[f]$. Similar to the previous cases.
- $E' \equiv Z$ where $Z$ is a constant. There are only two symmetric non trivial cases: $E' \equiv Z \equiv X$ and $F' \equiv Y$ or $E' \equiv Z \equiv Y$ and $F' \equiv X$. In both cases, $E \stackrel{a}{\rightarrow} E''$ by a shorter inference, and since $E[Y := X] \equiv F[Y := X]$, by inductive hypothesis, there exists $F''$ such that $F \stackrel{a}{\rightarrow} F''$ and $(E'', F'') \in \mathcal{S}$.

□

*Proof of Theorem 1.* $\Leftarrow$) Let $E$ be a process such that for all $E_1$ reachable from $E$, if $E_1 \stackrel{h}{\rightarrow} E_2$ then $E_1 \stackrel{\hat{\tau}}{\Longrightarrow} E_3$ and $E_2 \setminus H \approx E_3 \setminus H$. Let

$$\mathcal{S} = \{(E_i \setminus H, (E_i | \Pi) \setminus H) \mid \Pi \in \mathcal{E}_H \text{ is a process and } E \rightsquigarrow E_i\}.$$

We prove that $\mathcal{S}$ is a weak bisimulation up to $\approx$. We have to consider the following cases:

- $(E_i | \Pi) \setminus H \stackrel{\tau}{\rightarrow} (E_i | \Pi_1) \setminus H$. Since $E_i \setminus H \stackrel{\hat{\tau}}{\Longrightarrow} E_i \setminus H$, by definition of $\mathcal{S}$ we have $(E_i \setminus H, (E_i | \Pi_1) \setminus H) \in \mathcal{S}$.

- $(E_i|\Pi) \setminus H \xrightarrow{l} (E_j|\Pi) \setminus H$, with $l \in L \cup \{\tau\}$. Hence $E_i \setminus H \xrightarrow{l} E_j \setminus H$ and, by definition of $\mathcal{S}$, $(E_j \setminus H, (E_j|\Pi) \setminus H) \in \mathcal{S}$.
- $(E_i|\Pi) \setminus H \xrightarrow{\tau} (E_j|\Pi_1) \setminus H$ where $E_i \xrightarrow{h} E_j$. By hypothesis $E_i \setminus H \stackrel{\hat{\tau}}{\Longrightarrow} E_k \setminus H$ and $E_j \setminus H \approx E_k \setminus H$. Hence, $E_k \setminus H \approx E_j \setminus H\ \mathcal{S}\ (E_j|\Pi_1) \setminus H)$.
- $E_i \setminus H \xrightarrow{a} E_j \setminus H$. Then, $(E_i|\Pi) \setminus H \xrightarrow{a} (E_j|\Pi) \setminus H$ and $(E_j \setminus H, (E_j|\Pi) \setminus H) \in \mathcal{S}$.

$\Rightarrow$) Let $E$ be $P\_BNDC$. Then, for all $E_i$ reachable from $E$, $E_i \in P\_BNDC$. In particular, for all $E_i$ reachable from $E$ and for all $\Pi \in \mathcal{E}_H$, $E_i \setminus H \approx (E_i|\Pi) \setminus H$. Suppose that $E_i \xrightarrow{h} E_j$. Let $\Pi \equiv \bar{h}$. Then $(E_i|\Pi) \setminus H \xrightarrow{\tau} E_j \setminus H$. Since $E_i \setminus H \approx (E_i|\Pi) \setminus H$, $E_i \setminus H \stackrel{\hat{\tau}}{\Longrightarrow} E_k \setminus H$ and $E_j \setminus H \approx E_k \setminus H$. □

*Proof of Lemma 2.* (1) is an immediate consequence of Theorem 1.

(2) follows from Theorem 1, since if $E \rightsquigarrow E'$ and $E' \xrightarrow{h} E''$, then $E' \stackrel{\hat{\tau}}{\Longrightarrow} E'$ and $E' \setminus H \approx \mathbf{0} \approx E'' \setminus H$.

In order to prove (3) and (4) it is sufficient to observe that if $E$ is $P\_BNDC$, then so are $E \setminus v$ and $E[f]$, since the first operation does not add high level transitions, while the second does not exchange low and high actions.

As far as (5) is concerned, it is known that if $E, F \in P\_BNDC$, then $E|F$ is $P\_BNDC$ (see [10]).

We now prove (6) by using Theorem 1. Let $E_i, F_j \in P\_BNDC$, with $i \in I, j \in J$. Consider $R \equiv (\sum_{i \in I} a_i.E_i + \sum_{j \in J}(h_j.F_j + \tau.F_j))$. If $R$ reaches $R'$ with at least one transition, then either there exists $i \in I$ such that $E_i$ reaches $R'$ or there exists $j \in J$ such that $F_j$ reaches $R'$, hence $R'$ is $P\_BNDC$. If $R$ reaches $E'$ with no transitions, then $R' \equiv R$, hence if $R' \xrightarrow{h} R''$, then there exists $j \in J$ such that $R'' \equiv F_j$, and $R' \xrightarrow{\tau} F_j$, so we have the thesis.

(7) immediately follows from the operational semantics of SPA terms. □

*Proof of Theorem 2.* We prove that all the rules in Core are correct.

The correctness of rules $(Low)$ and $(High)$ directly follows from Lemma 2.

Rule $(Const)$ is trivially correct.

From Lemma 2 we have the correctness of rules $(Rest)$, $(Label)$, $(Par)$, $(Choice)$, and $(Def)$ in the case in which $A = \emptyset$. The general case follows immediately by the definition of $\mathcal{HP}[A]$. □

*Proof of Lemma 3.* Immediate since if $X \xrightarrow{a_i} E_i$ with $i \neq j$, then $Y \xrightarrow{a_i} E_i$ and if $X \xrightarrow{a_j} E_j$ then $Y \xrightarrow{a_j} F$ with $F \approx E_j$. □

*Proof of Lemma 4.* By induction on the structure of $E$.

- $E \equiv X$ is a constant. It is immediate, since by definition $E_{\setminus v}$ is $X \setminus v$.
- $E \equiv a.E'$. By inductive hypothesis on $E'$ we have the thesis.
- $E \equiv E' + E''$. We have $E \setminus v \sim E' \setminus v + E'' \setminus v \sim E'_{\setminus v} + E''_{\setminus v}$. If $E'_{\setminus v} \equiv \mathbf{0}$, then $E'_{\setminus v} + E''_{\setminus v} \sim E''_{\setminus v}$, hence we have the thesis. Similarly we obtain the thesis if $E''_{\setminus v} \sim \mathbf{0}$. In the third case we already have the thesis.

– $E \equiv E'|E''$. It is trivial.
– $E \equiv E' \setminus w$. We have $E \setminus v \sim E' \setminus w \setminus v \sim E' \setminus v \setminus w \sim E'_{\setminus v} \setminus w$.
– $E \equiv E'[f]$. It is trivial.

□

*Proof of Lemma 5.*

$$\begin{array}{rll} X\backslash v \sim E\backslash v & & \text{since } X \stackrel{\text{def}}{=} E; \\ \sim E_{\backslash v} & & \text{by Lemma 4;} \\ \sim F_{\backslash v} & & \text{by Lemma 1;} \\ \sim F\backslash v & & \text{by Lemma 4;} \\ \sim Y\backslash v & & \text{since } Y \stackrel{\text{def}}{=} F; \end{array}$$

□

*Proof of Lemma 6.* The first part of the lemma immediately follows from Theorem 1, since if $E$ has been proved to be $\mathcal{HP}[\emptyset]$ in Core, then it is $P\_BNDC$.

The second part follows by induction on the length of the proof $E \in \mathcal{HP}[A]$ in Core.

If $E \equiv P$ and $P$ is a closed process and $P \in \mathcal{E}_L$ then, since $P$ is $P\_BNDC$, by Theorem 1 we have the thesis.

If $E \equiv P$ and $P$ is a closed process and $P \in \mathcal{E}_H$ then, since $P$ is $P\_BNDC$, by Theorem 1 we have the thesis.

If $E \equiv X$ and $X \in A$, then we immediately get the thesis, since $X$ reaches only $X$ and $X$ does not perform high actions without using its definition.

If $E \equiv X$ and $X \notin A$, then $X \stackrel{\text{def}}{=} E_1$ and Core proves that $E_1 \in \mathcal{HP}[A]$, with a shorter proof. Since $X \rightsquigarrow E' \stackrel{h}{\rightarrow} E''$ if and only if $E_1 \rightsquigarrow E' \stackrel{h}{\rightarrow} E''$ by inductive hypothesis on $E_1$ we have the thesis.

If $E \equiv E_1 \setminus v$, then if $E_1 \setminus v \rightsquigarrow E' \setminus v \stackrel{h}{\rightarrow} E'' \setminus v$ by inductive hypothesis $E' \stackrel{\hat{\tau}}{\Longrightarrow} E'''$ with $E''\backslash H \approx E'''\backslash H$, hence $E' \stackrel{\hat{\tau}}{\Longrightarrow} E'''\backslash v$ with $E''\backslash v\backslash H \approx E'''\backslash v\backslash H$

If $E \equiv E_1[f]$, as in the previous case we obtain the thesis by inductive hypothesis.

If $E \equiv \sum_{i\in I} a_i.E_i + \sum_{j\in J}(h_j.F_j + \tau.F_j)$, then if $E' \equiv E$ we immediately get the thesis, otherwise we obtain it by inductive hypothesis. □

*Proof of Theorem 3.* By using Theorem 1 we have to prove that if $Z_k \rightsquigarrow P' \stackrel{h}{\rightarrow} P''$ without applying the definitions of the constants in $A \setminus \{Z_{k'} \mid k' \in K\}$, then $P' \stackrel{\hat{\tau}}{\Longrightarrow} P'''$ without applying the definitions of the constants in $A\setminus\{Z_{k'} \mid k' \in K\}$, with $P'' \setminus H \approx P''' \setminus H$.

We proceed by induction of the number of applications of the (Definition) rule in the semantic derivation of $Z_k \rightsquigarrow P'$.

If the rule has never been applied, then $P' \equiv Z_k$. If $Z_k \stackrel{h}{\rightarrow} P''$, then there exists $j_k$ such that $P'' \equiv F_{j_k}$ and $F_{j_k} \in \mathit{safe}(Z_k, S)$. Hence four cases are possible: (1) $Z_k \stackrel{\tau}{\Longrightarrow} F_{j_k}$; (2) $F_{j_k\backslash H} \equiv Z_{k\backslash H}$; (3) $F_{j_k\backslash H} \equiv \tau.Z_{k\backslash H}$; (4) $F_{j_k} \equiv Z_j$ and

$E_{j\setminus H}[Z_k := Z_j] \equiv E_{k\setminus H}[Z_k := Z_j]$. In case (1) we obtain the thesis since $F_{j_k} \setminus H \approx F_{j_k} \setminus H$ no matter which is the definition for the constants which occur in it. In case (2) we have that $Z_k$ reaches with zero $\tau$ actions $Z_k$ and $Z_k \setminus H \approx Z_{k\setminus H} \equiv F_{j_k\setminus H} \approx F_{j_k} \setminus H$. Similarly we obtain the thesis in case (3). In case (4) we obtain the thesis since $Z_k$ reaches with zero $\tau$ actions $Z_k$ and from Lemma 5 $Z_k \setminus H \approx Z_j \setminus H$.

If the rule has been applied exactly once, then the rule has been applied to $Z_k$ in the first step, i.e. $Z_k \stackrel{a_{i_k}}{\rightarrow} E_{i_k} \rightsquigarrow P' \stackrel{h}{\rightarrow} P''$ (or $Z_k \stackrel{h_{j_k}}{\rightarrow} F_{j_k} \ldots$) and in the derivation of $E_{i_k} \rightsquigarrow P' \stackrel{h}{\rightarrow} P''$ the (Definition) rule has never been applied. This means that $E_{i_k} \rightsquigarrow P' \stackrel{h}{\rightarrow} P''$ without using the definitions of the $Z$'s. From Lemma 6 we have that $P' \stackrel{\hat{\tau}}{\Longrightarrow} P'''$ without using the definition of the $Z$'s and $P''' \setminus H \approx P'' \setminus H$. This implies that $P' \stackrel{\hat{\tau}}{\Longrightarrow} P'''$ with $P''' \setminus H \approx P'' \setminus H$ no matter which is the definition of the $Z$'s.

Let us assume that we have proved that for each $k \in K$ if $Z_k \rightsquigarrow P' \stackrel{h}{\rightarrow} P''$ with $n$ applications of the (Definition) rule, then $P' \stackrel{\hat{\tau}}{\Longrightarrow} P'''$ with $P'' \setminus H \approx P''' \setminus H$. Let $Z_k \rightsquigarrow P' \stackrel{h}{\rightarrow} P''$ with $n+1$ applications of the (Definition) rule. This means that $Z_k \stackrel{a_{i_k}}{\rightarrow} E_{i_k} \rightsquigarrow Z_r \rightsquigarrow P' \stackrel{h}{\rightarrow} P''$ or $Z_k \stackrel{h_{j_k}}{\rightarrow} F_{j_k} \rightsquigarrow Z_r \rightsquigarrow P' \stackrel{h}{\rightarrow} P''$, and since the (Definition) rule has been applied once in the first step we have that in $Z_r \rightsquigarrow P' \stackrel{h}{\rightarrow} P''$ the (Definition) rule is applied at most $n$ times. Hence by inductive hypothesis we have the thesis. □

*Proof of Theorem 4.* By Theorem 3 we have that if the rule $(Sys)$ is applied once, then the proof is correct.

If the rule $(Sys)$ is applied more than once, then we obtain the thesis since Lemma 6 holds also if $G$ has been proved to be in $\mathcal{HP}[A]$ by applying the rule $(Sys)$. This last can be proved by induction on the number of application of the rule $(Sys)$. □