Proofs Methods for Bisimulation based Information Flow Security*

Riccardo Focardi, Carla Piazza, and Sabina Rossi

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

Abstract. *Persistent_BNDC* (*P_BNDC*, for short) is a security property for processes in dynamic contexts, i.e., contexts that can be reconfigured at runtime. We study how to efficiently decide if a process is *P_BNDC*. We exploit a characterization of *P_BNDC* through a suitable notion of *Weak Bisimulation up to high level actions*. In the case of finite-state processes, we study two methods for computing the largest weak bisimulation up to high level actions: (1) via *Characteristic Formulae* and Model Checking for μ-calculus and (2) via *Closure up to a set of actions* and *Strong Bisimulation*. This second method seems to be particularly appealing: it can be performed using already existing tools at a low time complexity.

1 Introduction

Systems are becoming more and more complex, and the security community has to face this by 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. 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.

Persistent_BNDC (*P_BNDC*, for short) [11, 12], is a security property based on the idea of Non-Interference [13] (formalized as *BNDC* [10]), which is suitable to analyze processes in dynamic 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* leads to some interesting results, as it can be equivalently defined as a *Weak Bisimulation up to high level actions*. This result, allowing to avoid both the universal quantification over all the possible attackers, present in *BNDC*, and the universal quantification over all possible reachable states, required by the definition of *P_BNDC*, naturally suggests the effective computability of *P_BNDC*.

* Partially supported by the MURST projects "Interpretazione astratta, type systems e analisi control-flow" and "Modelli formali per la sicurezza" and the EU Contract IST-2001-32617 "Models and Types for Security in Mobile Distributed Systems".

In this paper we consider the specific problem of automatically checking *P_BNDC*. In particular, we describe two methods for determining whether a system is *P_BNDC*. The first method is based on the derivation of *Characteristic Formulae* [21, 24] in the language of modal μ-calculus [16]. The characteristic formulae can be automatically verified using model checkers for μ-calculus, such as NCSU Concurrency Workbench [4]. The second method is in the spirit of [24]: it is based on the computation of a sort of transitive closure (*Closure up to high level actions*) of the system and on the verification of a *Strong Bisimulation*. This allows us to use existing tools as a large number of algorithms for computing the largest strong bisimulation between two processes have been proposed [22, 2, 17, 7] and are integrated in model checkers, such as NCSU Concurrency Workbench, XEVE [1], FDR2 [23]. In particular, this second approach improves on the polynomial time complexity of the Compositional Security Checker CoSeC presented in [9], since only one bisimulation test is necessary.

The paper is organized as follows. In Section 2 we recall the *Security Process Algebra* (SPA, for short) and the notions of Strong and Weak bisimulation. In Section 3 we introduce the *P_BNDC* property and we recall its characterization in terms of weak bisimulation up to high level actions. In Section 4 we propose two methods to prove the weak bisimulation up high level actions and we demonstrate some complexity results. Finally, in Section 5 we draw some conclusions.

2 Preliminaries

The *Security Process Algebra* (SPA, for short) [10] is a slight extension of Milner's CCS [20], 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 τ which models internal computations, i.e., not visible outside the system; a complementation function $\bar{\cdot} : \mathcal{L} \to \mathcal{L}$, such that $\bar{\bar{a}} = a$, for all $a \in \mathcal{L}$, and $\bar{\tau} = \tau$; $Act = \mathcal{L} \cup \{\tau\}$ is the set of all *actions*. The set of visible actions is partitioned into two sets, Act_H and Act_L, of high and low level actions such that $\overline{Act_H} = Act_H$ and $\overline{Act_L} = Act_L$, and $Act_H \cup Act_L = \mathcal{L}$ 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 executions are interleaved, possibly synchronized on complementary input/output actions, producing an internal action τ;

$$\text{Prefix} \quad \frac{}{a.E \stackrel{a}{\rightarrow} E}$$

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

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

$$\text{Restriction} \quad \frac{E \stackrel{a}{\rightarrow} E'}{E \setminus v \stackrel{a}{\rightarrow} E' \setminus v} \; \text{if } a \notin v$$

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

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

Fig. 1. The operational rules for SPA

$E \setminus v$ 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.

The operational semantics of SPA agents is given in terms of *Labelled Transition Systems*. A *Labelled Transition System* (LTS) is a triple $(S, A, \rightarrow)$ where S is a set of states, A is a set of labels (actions), $\rightarrow \subseteq S \times A \times S$ is a set of labelled transitions. The notation $(S_1, a, S_2) \in \rightarrow$ (or equivalently $S_1 \stackrel{a}{\rightarrow} S_2$) means that the system can move from the state S_1 to the state S_2 through the action a. The operational semantics of SPA is the LTS $(\mathcal{E}, Act, \rightarrow)$, where the states are the terms of the algebra and the transition relation $\rightarrow \subseteq \mathcal{E} \times Act \times \mathcal{E}$ is defined by structural induction as the least relation generated by the 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 and we refer to it as $LTS(E) = (S_E, Act, \rightarrow)$, where S_E is the set of processes reachable from E. A process E is said to be *finite-state* if S_E is finite.

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 $\mathcal{E}$, equating two processes when they are indistinguishable. In the following, we report the definitions of two observation equivalences called *strong bisimulation* and *weak bisimulation* [20].

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

Definition 1 (Strong Bisimulation). *A binary relation $\mathcal{R} \subseteq \mathcal{E} \times \mathcal{E}$ over agents 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 agents $E, F \in \mathcal{E}$ are strongly bisimilar*, denoted by $E \sim F$, if there exists a strong bisimulation $\mathcal{R}$ containing the pair (E, F).*

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 auxiliary notations. If $t = a_1 \cdots a_n \in Act^*$ and $E \xrightarrow{a_1} \cdots \xrightarrow{a_n} E'$, then we write $E \xrightarrow{t} E'$. We also write $E \stackrel{t}{\Longrightarrow} E'$ 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. Hence, $E \stackrel{\hat{a}}{\Longrightarrow} E'$ stands for $E \stackrel{a}{\Longrightarrow} E'$ if $a \in \mathcal{L}$, and for $E(\xrightarrow{\tau})^* E'$ if $a = \tau$.

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

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

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

In [20] it is proved that $\sim$ is the largest strong bisimulation, $\approx$ is the largest weak bisimulation and they are equivalence relations.

3 Security Properties

We recall the *Persistent_BNDC* (*P_BNDC*, for short) security property and its characterization in terms of weak bisimulation up to high level actions [11, 12].

We first give the definition of *Bisimulation-based Non Deducibility on Compositions* (*BNDC*, for short) [8, 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. Property *BNDC* is based on the idea of checking the system against all high level potential interactions, representing every possible high level malicious program. In particular, a system E is *BNDC* if for every high level process Π a low level user cannot distinguish E from $(E|\Pi) \setminus Act_H$, i.e., if Π cannot interfere [13] with the low level execution of the system E.

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

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

In [11, 12] it is shown that the *BNDC* property is not strong enough to analyse systems in dynamic execution environments. For example, if code mobility is allowed, a program could migrate to a different host in the middle of its computation. In this setting we have to guarantee that every reachable state of the process is secure. Another interesting example is the execution of an applet on a Java Card, where an attacker could try to bring the card in an unstable (insecure) state by powering off the card in the middle of applet computation.

To deal with these situations, in [11, 12] it has been introduced the security property named *P_BNDC*.

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

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

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

In [12] it has been proven that 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. In [11, 12] 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 Act_H. More in detail, it is defined an observation equivalence, named *weak bisimulation up to* Act_H, where actions from Act_H are allowed to be ignored, i.e., they are allowed to be matched by zero or more τ actions. To do this, it is used a transition relation which does not take care of both internal and high level actions.

We use the following notations. For an action $a \in Act$, we write $(\xrightarrow{a})^{\{0,1\}}$ to denote a sequence of zero or one a actions. The expression $E \overset{\hat{a}}{\Longrightarrow}_{\setminus Act_H} E'$ is a shorthand for $E \overset{\hat{a}}{\Longrightarrow} E'$ if $a \notin Act_H$, and for $E(\xrightarrow{\tau})^*(\xrightarrow{a})^{\{0,1\}}(\xrightarrow{\tau})^*E'$ if $a \in Act_H$. Notice that the relation $\overset{\hat{a}}{\Longrightarrow}_{\setminus Act_H}$ is a generalization of the relation $\overset{\hat{a}}{\Longrightarrow}$ used in the definition of weak bisimulation [20]. In fact, if $Act_H = \emptyset$, then for all $a \in Act$, $E \overset{\hat{a}}{\Longrightarrow}_{\setminus Act_H} E'$ coincides with $E \overset{\hat{a}}{\Longrightarrow} E'$.

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

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

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

The relation $\approx_{\setminus Act_H}$ is the largest weak bisimulation up to Act_H and it is an equivalence relation. In [12] it is proven that *P_BNDC* can be characterized in terms of $\approx_{\setminus Act_H}$ as follows. We will exploit this result for verifying *P_BNDC*.

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

4 Checking *P_BNDC*

In this section we present two methods to determine whether $E \approx_{\setminus Act_H} E \setminus Act_H$, in the case that E is a finite-state process. In particular, we tackle the problem of proving $E \approx_{\setminus Act_H} F$, when E and F are finite-state processes. The first method we propose consists in defining from a given process E a modal μ-calculus formula $\phi^{\approx_{\setminus Act_H}}(E)$ such that F satisfies $\phi^{\approx_{\setminus Act_H}}(E)$ if and only if $E \approx_{\setminus Act_H} F$. The second method consists in deriving from the LTS's of E and F two transformed LTS's that are strongly bisimilar if and only if $E \approx_{\setminus Act_H} F$.

4.1 Characteristic Formulae

The modal μ-calculus [16] is a small, yet expressive process logic. We consider modal μ-calculus formulae constructed according to the following grammar:

$$\phi ::= \mathbf{true} \mid \mathbf{false} \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \langle a \rangle \phi \mid [a]\phi \mid X \mid \mu X.\phi \mid \nu X.\phi$$

where X ranges over an infinite set of variables and a over a set of actions Act. The *fixpoint operators* μX and νX bind the respective variable X and we adopt the usual notion of closed formula. For a finite set M of formulae, we write $\bigwedge M$ and $\bigvee M$ for the conjunction and disjunction of the formulae in M.

Modal μ-calculus formulae are interpreted over processes, which are modelled by LTS's. Let E be a process and $LTS(E) = (S_E, Act_H, \rightarrow)$. The subset of states that satisfy a formula ϕ, denoted by $M_E(\phi)(\rho)$, is intuitively defined in Fig. 2. We use the notion of *environment* that is a partial mapping $\rho : Var \not\rightarrow 2^{S_E}$ which interprets at least the free variables of ϕ by subsets of S_E. For a set $x \subseteq S_E$ and a variable X, we write $\rho[X \mapsto x]$ for the environment that maps X to x and that is defined on a variable $Y \neq X$ iff ρ is defined on Y and maps Y then to $\rho(Y)$.

Intuitively, **true** and **false** hold for all resp. no states and $\wedge$ and $\vee$ are interpreted by conjunction and disjunction, $\langle a \rangle \phi$ holds for a state $E' \in S_E$ if there is a state E'' reachable from E' with an action a which satisfies ϕ, and $[a]\phi$

$$
\begin{aligned}
M_E(\mathbf{true})(\rho) &= S_E \\
M_E(\mathbf{false})(\rho) &= \emptyset \\
M_E(\phi_1 \wedge \phi_2)(\rho) &= M_E(\phi_1)(\rho) \cap M_E(\phi_2)(\rho) \\
M_E(\phi_1 \vee \phi_2)(\rho) &= M_E(\phi_1)(\rho) \cup M_E(\phi_2)(\rho) \\
M_E(\langle a \rangle \phi)(\rho) &= \{E' \mid \exists E'' :\ E' \xrightarrow{a} E'' \ \wedge\ E' \in M_E(\phi)(\rho)\} \\
M_E([a]\phi)(\rho) &= \{E' \mid \forall E'' :\ E' \xrightarrow{a} E'' \ \Rightarrow\ E'' \in M_E(\phi)(\rho)\} \\
M_E(X)(\rho) &= \rho(X) \\
M_E(\mu X.\phi)(\rho) &= \bigcap\{x \subseteq S_E \mid M_E(\phi)(\rho[X \mapsto x]) \subseteq x\} \\
M_E(\nu X.\phi)(\rho) &= \bigcup\{x \subseteq S_e \mid M_E(\phi)(\rho[X \mapsto x]) \supseteq x\}
\end{aligned}
$$

Fig. 2. Semantics of modal mu-calculus

holds for E' if all states E'' reachable from E' with an action a satisfy ϕ. The interpretation of a variable X is as prescribed by the environment. The formula $\mu X.\phi$, called *least fixpoint formula*, is interpreted by the smallest subset x of S_E that recurs when ϕ is interpreted with the substitution of x for X. Similarly, $\nu X.\phi$, called *greatest fixpoint formula*, is interpreted by the largest such set. Existence of such sets follow from the well-known Knaster-Tarski fixpoint theorem. As the meaning of a closed formula ϕ does not depend on the environment, we sometimes write $M_E(\phi)$ for $M_E(\phi)(\rho)$ where ρ is an arbitrary environment.

The set of processes *satisfying* a closed formula ϕ is $Proc(\phi) = \{F \mid F \in M_F(\phi)\}$. We also refer to (closed) *equation systems* of modal μ-calculus formulae,

$$Eqn : X_1 = \phi_1, \ldots, X_n = \phi_n$$

where $X_1, \ldots, X_n$ are mutually distinct variables and $\phi_1, \ldots, \phi_n$ are modal μ-calculus formulae having at most $X_1, \ldots, X_n$ as free variables.

An environment $\rho : \{X_1, \ldots, X_n\} \to 2^{S_E}$ is a *solution* of an equation system Eqn, if $\rho(X_i) = M_E(\phi_i)(\rho)$. The fact that solutions always exist, is again a consequence of the Knaster-Tarski fixpoint theorem. In fact the set of environments that are candidates for solutions, $Env_E = \{\rho \mid \rho : \{X_1, \ldots, X_n\} \to 2^{S_E}\}$, together with the lifting $\sqsubseteq$ of the inclusion order on 2^{S_E}, defined by $\rho \sqsubseteq \rho'$ iff $\rho(X_i) \subseteq \rho'(X_i)$ for $i \in [1..n]$ forms a complete lattice. Now, we can define the *equation functional* $Func_E^{Eqn} : Env_E \to Env_E$ by $Func_E^{Eqn}(\rho)(X_i) = M_E(\phi_i)(\rho)$ for $i \in [1..n]$, the fixpoints of which are just the solutions of Eqn. $Func_E^{Eqn}$ is monotonic as $M_E(\phi_i)$ is monotonic. In particular, there is the largest solution $\nu Func_E^{Eqn}$ of Eqn (with respect to $\sqsubseteq$), which we denote by $M_E(Eqn)$. This definition interprets equation systems on the states of a given process E. We lift this to processes by agreeing that a process satisfies an equation system Eqn, if its initial state is in the largest solution of the first equation. Thus the set of processes satisfying the system Eqn is $Proc(Eqn) = \{F \mid F \in M_F(Eqn)(X_1)\}$.

The relation $\approx_{\backslash Act_H} \subseteq \mathcal{E} \times \mathcal{E}$ can be characterized as the greatest fixpoint $\nu Func_{\approx_{\backslash Act_H}}$ of the monotonic functional $Func_{\approx_{\backslash Act_H}}$ on the complete lattice of relations $\mathcal{R} \subseteq \mathcal{E} \times \mathcal{E}$ ordered by set inclusion, where $(E, F) \in Func_{\approx_{\backslash Act_H}}(\mathcal{R})$ if and only if points (1) and (2) of Definition 5 hold. Thus a relation $\mathcal{R}$ is a weak bisimulation up to Act_H if and only if $\mathcal{R} \subseteq \mathit{Func}_{\approx_{\setminus Act_H}}(\mathcal{R})$, i.e., $\mathcal{R}$ is a *post-fixpoint* of $\mathit{Func}_{\approx_{\setminus Act_H}}$. By the Knaster-Tarski fixpoint theorem, $\nu\mathit{Func}_{\approx_{\setminus Act_H}}$ is the union of all post-fixpoints of $\mathit{Func}_{\approx_{\setminus Act_H}}$, i.e., it is the largest weak bisimulation up to Act_H. If we restrict to the complete lattice of relations $\mathcal{R} \subseteq S_E \times S_F$ we obtain a monotonic functional $\mathit{Func}^{(E,F)}_{\approx_{\setminus Act_H}}$ whose greatest fixpoint is exactly $\nu\mathit{Func}_{\approx_{\setminus Act_H}} \cap (S_E \times S_F)$, and this is enough to determine if $E \approx_{\setminus Act_H} F$.

Let E be a finite-state process, $E_1, \ldots, E_n$ its $|S_E| = n$ states, and $E_1 = E$ its initial state. We construct a *characteristic equation system* [21]

$$\mathit{Eqn}_{\approx_{\setminus Act_H}} : X_{E_1} = \phi^{\approx_{\setminus Act_H}}_{E_1}, \ldots, X_{E_n} = \phi^{\approx_{\setminus Act_H}}_{E_n}$$

consisting of one equation for each state $E_1, \ldots, E_n \in S_E$. We define the formulae $\phi^{\approx_{\setminus Act_H}}_{E_i}$ such that the largest solution $M_F(\mathit{Eqn}_{\approx_{\setminus Act_H}})$ of $\mathit{Eqn}_{\approx_{\setminus Act_H}}$ on an arbitrary process F associates the variables $X_{E'}$ just with the states F' of F which are weakly bisimilar up to Act_H to E'. Theorem 2 is in the spirit of [21] and shows the exact form of such formulae. We use these notations:

$$\langle\!\langle a \rangle\!\rangle_{\setminus Act_H} \phi \stackrel{\text{def}}{=} \begin{cases} \langle\!\langle \tau \rangle\!\rangle \phi & \text{if } a = \tau \\ \langle\!\langle a \rangle\!\rangle \phi & \text{if } a \notin Act_H \text{ and } a \neq \tau \\ \langle\!\langle a \rangle\!\rangle \phi \vee \langle\!\langle \tau \rangle\!\rangle \phi & \text{if } a \in Act_H \text{ and } a \neq \tau \end{cases}$$

where $\langle\!\langle \tau \rangle\!\rangle \phi \stackrel{\text{def}}{=} \mu X.\phi \vee \langle \tau \rangle X$ and $\langle\!\langle a \rangle\!\rangle \phi \stackrel{\text{def}}{=} \langle\!\langle \tau \rangle\!\rangle \langle a \rangle \langle\!\langle \tau \rangle\!\rangle \phi$. Notice that $\langle\!\langle a \rangle\!\rangle_{\setminus Act_H}$, $\langle\!\langle \tau \rangle\!\rangle$ and $\langle\!\langle a \rangle\!\rangle$ correspond to $\stackrel{a}{\Longrightarrow}_{\setminus Act_H}$, $\stackrel{\hat{\tau}}{\Rightarrow}$ and $\stackrel{a}{\Rightarrow}$, respectively, since

$$\begin{aligned} M_E(\langle\!\langle a \rangle\!\rangle_{\setminus Act_H} \phi)(\rho) &= \{E' \mid \exists E'' \ : \ E' \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} E'' \wedge E'' \in M_E(\phi)(\rho)\}, \\ M_E(\langle\!\langle \tau \rangle\!\rangle \phi)(\rho) &= \{E' \mid \exists E'' \ : \ E' \stackrel{\hat{\tau}}{\Longrightarrow} E'' \wedge E'' \in M_E(\phi)(\rho)\}, \\ M_E(\langle\!\langle a \rangle\!\rangle \phi)(\rho) &= \{E' \mid \exists E'' \ : \ E' \stackrel{a}{\Longrightarrow} E'' \wedge E'' \in M_E(\phi)(\rho)\}. \end{aligned}$$

Theorem 2. *$M_F(\mathit{Eqn}_{\approx_{\setminus Act_H}})(X_{E'}) = \{F' \in S_F \mid E' \approx_{\setminus Act_H} F'\}$ when*

$$\begin{aligned} \phi^{\approx_{\setminus Act_H}}_{E'} \stackrel{\text{def}}{=} & \bigwedge \{ \bigwedge \{ \langle\!\langle \hat{a} \rangle\!\rangle_{\setminus Act_H} X_{E''} \mid E' \stackrel{a}{\to} E'' \} \mid a \in Act \} \wedge \\ & \bigwedge \{ [a] \bigvee \{ X_{E''} \mid E' \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} E'' \} \mid a \in Act \}. \end{aligned}$$

Example 2. Consider the process E_1 of Example 1. For every state E' reachable from E', let $\psi_{E'}$ denote $\phi^{\approx_{\setminus Act_H}}_{E'}$. Then

$$\begin{aligned} \psi_{E_1} = {} & \langle\!\langle l \rangle\!\rangle_{\setminus Act_H} X_{h.j.\mathbf{0}} \wedge \langle\!\langle l \rangle\!\rangle_{\setminus Act_H} X_{\tau.j.\mathbf{0}+\tau.\mathbf{0}} \wedge \\ & [l](X_{h.j.\mathbf{0}} \vee X_{\tau.j.\mathbf{0}+\tau.\mathbf{0}} \vee X_{j.\mathbf{0}} \vee X_{\mathbf{0}}) \wedge [\tau] X_{E_1} \wedge [h] X_{E_1} \\ \psi_{\tau.j.\mathbf{0}+\tau.\mathbf{0}} = {} & \langle\!\langle \tau \rangle\!\rangle_{\setminus Act_H} X_{j.\mathbf{0}} \wedge \langle\!\langle \tau \rangle\!\rangle_{\setminus Act_H} X_{\mathbf{0}} \wedge \\ & [\tau](X_{\tau.j.\mathbf{0}+\tau.\mathbf{0}} \vee X_{\tau.j.\mathbf{0}} \vee X_{j.\mathbf{0}} \vee X_{\tau.\mathbf{0}} \vee X_{\mathbf{0}}) \wedge \\ & [h](X_{\tau.j.\mathbf{0}+\tau.\mathbf{0}} \vee X_{\tau.j.\mathbf{0}} \vee X_{j.\mathbf{0}} \vee X_{\tau.\mathbf{0}} \vee X_{\mathbf{0}}) \\ \psi_{\tau.j.\mathbf{0}} = {} & \langle\!\langle \tau \rangle\!\rangle_{\setminus Act_H} X_{j.\mathbf{0}} \wedge [\tau](X_{\tau.j.\mathbf{0}} \vee X_{j.\mathbf{0}}) \wedge [h](X_{\tau.j.\mathbf{0}} \vee X_{j.\mathbf{0}}) \\ \psi_{h.j.\mathbf{0}} = {} & \langle\!\langle h \rangle\!\rangle_{\setminus Act_H} X_{j.\mathbf{0}} \wedge [\tau] X_{h.j.\mathbf{0}} \wedge [h](X_{h.j.\mathbf{0}} \vee X_{j.\mathbf{0}}) \\ \psi_{j.\mathbf{0}} = {} & \langle\!\langle j \rangle\!\rangle_{\setminus Act_H} X_{\mathbf{0}} \wedge [h] X_{j.\mathbf{0}} \wedge [\tau] X_{j.\mathbf{0}} \wedge [j] X_{\mathbf{0}} \\ \psi_{\tau.\mathbf{0}} = {} & \langle\!\langle \tau \rangle\!\rangle_{\setminus Act_H} X_{\mathbf{0}} \wedge [\tau](X_{\tau.\mathbf{0}} \vee X_{\mathbf{0}}) \wedge [h](X_{\tau.\mathbf{0}} \vee X_{\mathbf{0}}) \\ \psi_{\mathbf{0}} = {} & [h] X_{\mathbf{0}} \wedge [\tau] X_{\mathbf{0}} \end{aligned}$$

Corollary 1. $Proc(Eqn_{\approx_{\setminus Act_H}}) = \{F \mid E \approx_{\setminus Act_H} F\}$.

This result holds for all processes F as $Eqn_{\approx_{\setminus Act_H}}$ does not depend on F.

Characteristic formulae, i.e., *single* formulae characterizing processes can be constructed by applying simple semantics-preserving transformation rules on equation systems as described in [21]. These rules are similar to the ones used by A. Mader in [19] as a mean of solving Boolean equation systems (with alternation) by Gauss elimination. Hence, since for any equation system Eqn there is a formula ϕ such that $Proc(Eqn) = Proc(\phi)$, we obtain that:

Theorem 3. *For all finite-state processes E there is a modal μ-calculus formulae $\phi^{\approx_{\setminus Act_H}}(E)$ such that $Proc(\phi^{\approx_{\setminus Act_H}}(E)) = \{F \mid E \approx_{\setminus Act_H} F\}$.*

Using this method we can for instance exploit the model checker NCSU Concurrency Workbench ([4]) to check whether $E \approx_{\setminus Act_H} F$. Unfortunately, in the μ-calculus formula we obtain for a process E there are both μ and ν operators (see [21]). In the worst case the number of μ and ν alternations in $\phi^{\approx_{\setminus Act_H}}(E)$ is $2|S_E|+1$ (when $LST(E)$ has a unique strongly connected component) and in that case the complexity of model checking $\phi^{\approx_{\setminus Act_H}}(E)$ on $LTS(F)$ is $O(|S_F|^{(2|S_E|+1)/2})$ (see [18,3]).

4.2 Strong Bisimulation

We show now how to reduce the problem of testing whether two processes are weakly bisimilar up to Act_H to a strong bisimulation problem. The next property follows from the definition of $\stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H}$.

Proposition 1. *A binary relation $\mathcal{R} \subseteq \mathcal{E} \times \mathcal{E}$ over agents is a* weak bisimulation *up to Act_H if and only if $(E, F) \in \mathcal{R}$ implies, for all $a \in Act$*

- *if $E \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} E'$, there is $F' \in \mathcal{E}$ such that $F \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} F'$ and $(E', F') \in \mathcal{R}$;*
- *if $F \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} F'$, there is $E' \in \mathcal{E}$ such that $E \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} E'$ and $(E', F') \in \mathcal{R}$.*

Proof. ($\Rightarrow$). We prove that if $\mathcal{R} \subseteq \mathcal{E} \times \mathcal{E}$ is a weak bisimulation up to Act_H, and $(E, F) \in \mathcal{R}$, then, for all $a \in Act$ we have

- if $E \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} E'$, there is $F' \in \mathcal{E}$ such that $F \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} F'$ and $(E', F') \in \mathcal{R}$;
- if $F \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} F'$, there is $E' \in \mathcal{E}$ such that $E \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} E'$ and $(E', F') \in \mathcal{R}$.

We distinguish three cases.

Case 1. $a = \tau$. In this case $E \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} E'$ coincides with $E(\stackrel{\tau}{\to})^* E'$. The proof follows by induction on the number of τ actions in $E(\stackrel{\tau}{\to})^* E'$. The base case arises when zero τ actions are performed and it is trivial. For the induction step, let $E \stackrel{\tau}{\to} E''(\stackrel{\tau}{\to})^* E'$. Since, $(E, F) \in \mathcal{R}$, by Definition 5 there exists $F'' \in \mathcal{E}$ such that $F \stackrel{\hat{\tau}}{\Longrightarrow}_{\setminus Act_H} F''$, i.e., $F(\stackrel{\tau}{\to})^* F''$ and $(E'', F'') \in \mathcal{R}$. By the induction hypothesis, there exists $F' \in \mathcal{E}$ such that $F'' \stackrel{\hat{\tau}}{\Longrightarrow}_{\setminus Act_H} F'$, i.e., $F''(\stackrel{\tau}{\to})^* F'$ and $(E', F') \in \mathcal{R}$. This proves the thesis since $F(\stackrel{\tau}{\to})^* F''(\stackrel{\tau}{\to})^* F'$, i.e., $F \stackrel{\hat{\tau}}{\Longrightarrow}_{\setminus Act_H} F'$.

Case 2. $a \in \mathcal{L}$ and $a \notin Act_H$. In this case we have that $E \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} E'$ coincides with $E(\stackrel{\tau}{\rightarrow})^* E'' \stackrel{a}{\rightarrow} E'''(\stackrel{\tau}{\rightarrow})^* E'$. By Case 1 above, there exists $\bar{F}'' \in \mathcal{E}$ such that $F(\stackrel{\tau}{\rightarrow})^* \bar{F}''$ and $(E'', \bar{F}'') \in \mathcal{R}$. By Definition 5 there exists $\bar{F}''' \in \mathcal{E}$ such that $\bar{F}'' \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} \bar{F}'''$, i.e., $\bar{F}''(\stackrel{\tau}{\rightarrow})^* F'' \stackrel{a}{\rightarrow} F'''(\stackrel{\tau}{\rightarrow})^* \bar{F}'''$ and $(E''', \bar{F}''') \in \mathcal{R}$. Again, by Case 1 above, there exists $F' \in \mathcal{E}$ such that $\bar{F}'''(\stackrel{\tau}{\rightarrow})^* F'$ and $(E', F') \in \mathcal{R}$. This proves the thesis since $F(\stackrel{\tau}{\rightarrow})^* F'' \stackrel{a}{\rightarrow} F'''(\stackrel{\tau}{\rightarrow})^* F'$, i.e., $F \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} F'$.

Case 3. $a \in Act_H$. In this case $E \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} E'$ coincides either with $E(\stackrel{\tau}{\rightarrow})^* E'$ or with $E(\stackrel{\tau}{\rightarrow})^* E'' \stackrel{a}{\rightarrow} E'''(\stackrel{\tau}{\rightarrow})^* E'$. The proof follows by Case 1 and Case 2 above.

($\Leftarrow$). It is easy to prove that if $\mathcal{R} \subseteq \mathcal{E} \times \mathcal{E}$ is a binary relation over agents such that for all $(E, F) \in \mathcal{R}$, $a \in Act$ it holds

- if $E \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} E'$, there is $F' \in \mathcal{E}$ such that $F \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} F'$ and $(E', F') \in \mathcal{R}$;
- if $F \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} F'$, there is $E' \in \mathcal{E}$ such that $E \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} E'$ and $(E', F') \in \mathcal{R}$;

then $\mathcal{R}$ is a weak bisimulation up to Act_H. In particular, this follows from the fact that, by the definition of $\stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H}$, $E \stackrel{a}{\rightarrow} E'$ implies $E \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} E'$ for each $E, E' \in \mathcal{E}$ and $a \in Act$. ∎

A direct consequence of this theorem is that two systems E and F are weakly bisimilar up to Act_H if and only if they are strongly bisimilar when in place of the transition relation $\stackrel{a}{\rightarrow}$ we consider the set of labelled transitions $\stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H}$.

We can exploit this fact to determine whether $E \approx_{\setminus Act_H} F$ by: (i) translating the two labelled transition systems $LTS(E)$ and $LTS(F)$, into $LTS^H(E)$ and $LTS^H(F)$; (ii) computing the largest strong bisimulation $\sim$ between $LTS^H(E)$ and $LTS^H(F)$. More formally we define:

Definition 6 (Closure up to Act_H). *Let $E \in \mathcal{E}$ with $LTS(E) = (S_E, Act, \rightarrow)$. The* closure up to Act_H *of E is the labelled transition system $LTS^H(E) = (S_E, Act, \hookrightarrow)$, where $\stackrel{a}{\hookrightarrow}$ is defined as $\stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H}$, i.e.:*

$$E' \stackrel{a}{\hookrightarrow} E'' = \begin{cases} E'(\stackrel{\tau}{\rightarrow})^* E'' & \text{if } a = \tau \\ E'(\stackrel{\tau}{\rightarrow})^* F' \stackrel{a}{\rightarrow} F''(\stackrel{\tau}{\rightarrow})^* E'' & \text{if } a \notin Act_H \\ E'(\stackrel{\tau}{\rightarrow})^* F' \stackrel{a}{\rightarrow} F''(\stackrel{\tau}{\rightarrow})^* E'' \text{ or } E'(\stackrel{\tau}{\rightarrow})^* E'' & \text{if } a \in Act_H \end{cases}$$

Let us denote with E^H a process whose operational semantics is given by the transformed transition system $LTS^H(E)$, i.e., $LTS(E^H) = LTS^H(E)$. The next result is an immediate consequence of Proposition 1.

Corollary 2. *Let $E, F \in \mathcal{E}$. Then, $E \approx_{\setminus Act_H} F$ iff $E^H \sim F^H$.*

Now, our first problem is to compute $LTS^H(E)$ from $LTS(E)$, using Definition 6. This can be immediately obtained with the following algorithm:

Algorithm 1 Let $E \in \mathcal{E}$ with $LTS(E) = (S_E, Act, \rightarrow)$. The *closure up to Act_H* of E, $LTS^H(E) = (S_E, Act, \hookrightarrow)$, is computed as follows:

1. calculate $\overset{\tau}{\hookrightarrow}$ as $(\overset{\tau}{\rightarrow})^*$, i.e., as the reflexive and transitive closure of $\overset{\tau}{\rightarrow}$;
2. calculate $\overset{a}{\hookrightarrow}$ as the composition $\overset{\tau}{\hookrightarrow} \circ \overset{a}{\rightarrow} \circ \overset{\tau}{\hookrightarrow}$;
3. if $a \in Act_H$ then add $E \overset{a}{\hookrightarrow} F$, every time $E \overset{\tau}{\hookrightarrow} F$.

Correctness of algorithm above is trivially obtained by observing that (by Definition 6): $\overset{\tau}{\hookrightarrow}$ is equivalent to $(\overset{\tau}{\rightarrow})^*$; $\overset{a}{\hookrightarrow}$ with $a \in \mathcal{L} \setminus Act_H$ is equivalent to $(\overset{\tau}{\rightarrow})^* \circ \overset{a}{\rightarrow} \circ (\overset{\tau}{\rightarrow})^*$, i.e., to $\overset{\tau}{\hookrightarrow} \circ \overset{a}{\rightarrow} \circ \overset{\tau}{\hookrightarrow}$; $\overset{a}{\hookrightarrow}$ with $a \in Act_H$ is equivalent to the union of $(\overset{\tau}{\rightarrow})^* \circ \overset{a}{\rightarrow} \circ (\overset{\tau}{\rightarrow})^*$ (calculated in step 2 above) and $(\overset{\tau}{\rightarrow})^*$ (calculated in step 3 above). As far as time and space complexities are concerned, we notice that they depend on the algorithms used for computing the reflexive and transitive closure and the composition of relations. We start by fixing some notations. Let $n = |S_E|$ be the number of states in $LTS(E)$, for each $a \in Act$, let m_a be the number of $\overset{a}{\rightarrow}$ transitions in $LTS(E)$, and $m = \sum_{a \in Act} m_a$. Similarly, let $\hat{m}_a$ be the number of $\overset{a}{\hookrightarrow}$ transitions in $LTS^H(E)$, and $\hat{m} = \sum_{a \in Act} \hat{m}_a$.

The next theorem shows that $E \approx_{\setminus Act_H} F$ can be checked in polynomial time with respect to the number of states of the system.

Theorem 4. *Algorithm 1 can be executed in time $O(n\hat{m}_\tau + n^w)$ and space $O(n^2)$, where w denotes the exponent in the running time of the matrix multiplication algorithm used.*[2] *If $\hat{m} \leq n$, then it is possible to work in time $O(n\hat{m})$ and space $O(n)$.*

Proof. First of all we have to determine the transitive closure of $\overset{\tau}{\rightarrow}$. The algorithm proposed in [14] computes the transitive closure of a graph represented with adjacency-lists in time $O(m_\tau + ne)$, where e is the number of edges in the transitive closure of the graph of the strongly connected components. Since $m_\tau, e \leq \hat{m}_\tau$, an upper bound to the cost of the computation of $(\overset{\tau}{\rightarrow})^*$ is $O(n\hat{m}_\tau)$.

Let us consider the computation of the composition $(\overset{\tau}{\rightarrow})^* \circ \overset{a}{\rightarrow} \circ (\overset{\tau}{\rightarrow})^*$. Given two transition relations $\rightarrow_1$ and $\rightarrow_2$ on a set of n nodes, the problem of determining the composition $\rightarrow_1 \circ \rightarrow_2$ is known to be equivalent to the $n \times n$ Boolean matrix multiplication problem (see [6]). In particular, if A_i is the adjacency-matrix defined by $\rightarrow_i$, for $i = 1, 2$, then the adjacency-matrix of $\rightarrow_1 \circ \rightarrow_2$ is the matrix $A_1 \cdot A_2$. Hence, in our case, we have to: (i) determine the adjacency-matrixes $A_{\tau*}$ and A_a associated to $(\overset{\tau}{\rightarrow})^*$ and $\overset{a}{\rightarrow}$ respectively; (ii) compute the product $(A_{\tau*} \cdot A_a) \cdot A_{\tau*}$; (iii) rebuild the adjacency-list representation (in the computation of the strong bisimulation it is important to use the adjacency-list representation). Starting from the adjacency-list representations of $(\overset{\tau}{\rightarrow})^*$ and $\overset{a}{\rightarrow}$ in time $O(n^2)$ we obtain their adjacency-matrix representations $A_{\tau*}$ and A_a. The matrix product $(A_{\tau*} \cdot A_a) \cdot A_{\tau*}$ can be determined in time $O(n^{2.376})$ using twice the algorithm in [5]. Then, again in time $O(n^2)$, we rebuild the adjacency-list representation. So, the global cost of the computation of $(\overset{\tau}{\rightarrow})^* \circ \overset{a}{\rightarrow} \circ (\overset{\tau}{\rightarrow})^*$ is $O(n^{2.376})$. We have to perform this step once for each $a \in \mathcal{L}$, assuming that $|\mathcal{L}|$ is

[2] In the algorithm in [5], which is at the moment the fastest in literature, we have that $w = 2.376$.

a constant wrt. n. Notice that we could work using only 2 matrix multiplications, instead of $2|\mathcal{L}|$ matrix multiplications, but in this case we would have to use matrixes in which each element is an array of length $\mathcal{L}$ of bits, hence also in this way it is not possible to drop the assumption that $|\mathcal{L}|$ is a constant wrt. n.

Hence, we have described a procedure which maps E into $LTS^H(E)$ in time $O(n\hat{m}_\tau + n^w)$ and space $O(n^2)$, where w is the exponent in the running time of the matrix multiplication algorithm used ($w = 2.376$ using [5]).

In the procedure just described we use the adjacency-matrix representation to compute $\stackrel{a}{\rightarrow} \circ (\stackrel{\tau}{\rightarrow})^*$. If we know that $\hat{m} \leq n$, then using the adjacency-list representation and a naïve algorithm (two iterations of the naïve algorithm for the transitive closure [6]) we can perform this step in time $O(n\hat{m})$. Thus, when $\hat{m} \leq n$, we determine $LTS^H(E)$ in time $O(n\hat{m})$ and space $O(n + \hat{m}) = O(n)$. ■

The theorem above is applicable to the general case $E \approx_{\setminus Act_H} F$. However, since in our case $F = E \setminus Act_H$, we can interleave the computation of $LTS^H(E)$ and $LTS^H(E \setminus Act_H)$, lowering the constant involved in the time complexity. To do so, we need the notion of Act_H-Completion defined as follows:

Definition 7 (Act_H-Completion). *Let $E \in \mathcal{E}$ with $LTS(E) = (S_E, Act, \rightarrow)$. The Act_H-Completion of E, $LTS_\mathcal{C}(E) = (S_E, Act, \hookrightarrow)$, is defined as follows: we have $E \stackrel{a}{\hookrightarrow} E'$ every time $E \stackrel{a}{\rightarrow} E'$. Moreover, every time $E \stackrel{\tau}{\rightarrow} E'$ we have $E \stackrel{a}{\hookrightarrow} E'$ for all $a \in Act_H$.*

Intuitively, the Act_H-completion extends a given LTS by adding an edge $\stackrel{a}{\hookrightarrow}$, with $a \in Act_H$, each time that there is an edge $\stackrel{\tau}{\rightarrow}$ in the original LTS.

Let us denote with $E^\emptyset$ a process whose operational semantics is given by the closure up to $\emptyset$ of $LTS(E)$. Note that this amounts to saying that $LTS(E^\emptyset) = (S_E, Act, \stackrel{\hat{a}}{\Longrightarrow})$. In fact, recall that if $Act_H = \emptyset$, then $E \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} E'$ coincides with $E \stackrel{\hat{a}}{\Longrightarrow} E'$ for all $a \in Act$. The following holds:

Proposition 2. *Let $E \in \mathcal{E}$ be a process.*

(i) $LTS^H(E) = LTS_\mathcal{C}(E^\emptyset)$
(ii) $LTS^H(E \setminus Act_H) = LTS_\mathcal{C}(E^\emptyset \setminus Act_H)$

Proof. The first equation follows immediately from the definitions and states that the Act_H-Completion of $E^\emptyset$ is the closure up to high level actions of E.

We prove the second equation. By definition, $LTS^H(E \setminus Act_H)$ is the LTS obtained by substituting $\stackrel{a}{\rightarrow}$ with $\stackrel{\hat{a}}{\Longrightarrow}$ in $LTS(E \setminus Act_H)$, as $E \setminus Act_H$ cannot execute high level actions. Thus, if E' is a state in $LTS^H(E \setminus Act_H)$, then E' is also a state in $LTS(E \setminus Act_H)$, i.e., there is a path from E to E' which does not involve actions of Act_H. This implies that E' is a state of $LTS(E^\emptyset \setminus Act_H)$, and hence it belongs also to $LTS_\mathcal{C}(E^\emptyset \setminus Act_H)$. Similarly we can prove that if E' is a state in $LTS_\mathcal{C}(E^\emptyset \setminus Act_H)$, then E' is a state in $LTS^H(E \setminus Act_H)$.

Now, we prove that $E' \stackrel{a}{\hookrightarrow} E''$ in $LTS^H(E \setminus Act_H)$ if and only if $E' \stackrel{a}{\hookrightarrow} E''$ in $LTS_\mathcal{C}(E^\emptyset \setminus Act_H)$. We distinguish three cases.

Case 1. $a = \tau$. Since operation $\backslash Act_H$ has no effects on τ transitions in both cases the τ transitions are exactly those in the transitive closure $(\xrightarrow{\tau})^*$ of E.

Case 2. $a \in \mathcal{L}$ and $a \notin Act_H$. Again, since operation $\backslash Act_H$ has no effects on the a transitions in both cases the a transitions are exactly the transitions in $(\xrightarrow{\tau})^* \circ \xrightarrow{a} \circ (\xrightarrow{\tau})^*$ computed on E.

Case 3. $a \in Act_H$. The a transitions which are in $LTS^H(E \setminus Act_H)$ are exactly the transitions in $(\xrightarrow{\tau})^*$ computed on E and also the a transitions which are in $LTS_{\mathcal{C}}(E^{\emptyset} \setminus Act_H)$ are exactly the transitions in $(\xrightarrow{\tau})^*$ computed on E. ∎

Hence we can determine $LTS^H(E)$ and $LTS^H(E \setminus Act_H)$ as follows:

Algorithm 2 Let $E \in \mathcal{E}$. We calculate $LTS^H(E)$ and $LTS^H(E \setminus Act_H)$ through the following steps:

1. compute $E^{\emptyset}$;
2. compute and give as output $LTS_{\mathcal{C}}(E^{\emptyset})$;
3. compute $E^{\emptyset} \setminus Act_H$;
4. compute and give as output $LTS_{\mathcal{C}}(E^{\emptyset} \setminus Act_H)$.

The correctness of the algorithm is given by Proposition 2 which proves that $LTS_{\mathcal{C}}(E^{\emptyset}) = LTS^H(E)$ (step 2 above) and $LTS_{\mathcal{C}}(E^{\emptyset} \setminus Act_H) = LTS^H(E \setminus Act_H)$ (step 4 above). The time and space complexity of the algorithm are the ones in Theorem 4, since steps 2, 3, and 4 can be performed using three visits.

Once we have the LTS's $LTS^H(E)$ and $LTS^H(E \setminus Act_H)$ there are many algorithms which can be used to decide whether $E^H \sim (E \setminus Act_H)^H$ (e.g., [22, 15, 17, 2, 7]). Some of these algorithms are integrated in model checkers [1, 4, 23]. The worst case time complexity of the algorithms in [22, 7] to decide $E^H \sim (E \setminus Act_H)^H$ is $O(\hat{m} \log n)$, assuming that the LTS's are represented using adjacency-lists. Using these complexity results together with Theorem 4 we obtain that:

Corollary 3. *It is possible to decide $E \approx_{\backslash Act_H} E \setminus Act_H$ in time $O(n\hat{m}_\tau + n^w + \hat{m} \log n)$ and space $O(n^2)$, where w denotes the exponent in the running time of the matrix multiplication algorithm used. If $\hat{m} \leq n$, then it is possible to work in time $O(n\hat{m})$ and space $O(n)$.*

Notice that using this approach in many practical cases there are a large number of states which occur both in $LTS^H(E)$ and in $LTS^H(E \setminus Act_H)$. We can avoid to replicate these states, share them among the two LTS's, and test whether the two roots are bisimilar. In particular, this can be done in the following way: after the computation of $E^{\emptyset}$, using a backward visit, mark all the nodes of $E^{\emptyset}$ which do not reach a transition whose label is in Act_H; while computing $LTS_{\mathcal{C}}(E^{\emptyset} \setminus Act_H)$ with a breath-first visit consider that if E' is a marked node, then E' is also a node in $LTS_{\mathcal{C}}(E^{\emptyset})$, hence share E' with $LTS_{\mathcal{C}}(E^{\emptyset})$ and do not call the breath-first visit on E'. In this way we lower again the constants involved in the effective time and space complexities: if we mark n' nodes, then in steps 3. and 4. of Algorithm 2 we have to visit only $n - n'$ nodes, and the total space required to store the nodes is $2n - n'$ instead of $2n$.

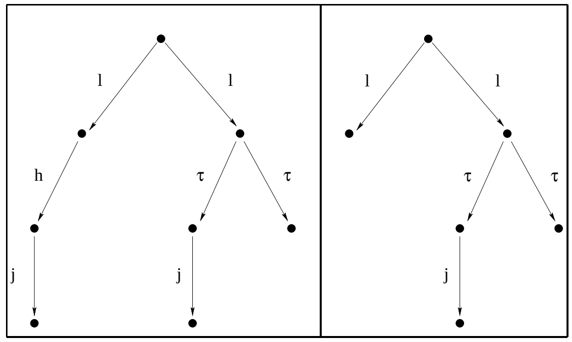

Fig. 3. The labelled transition systems of E_1 and $E_1 \setminus Act_H$.

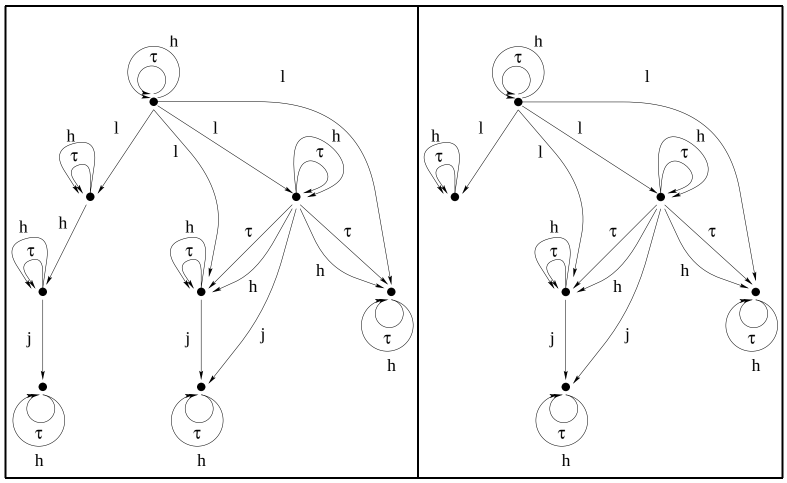

Fig. 4. The labelled transition systems $LTS^H(E_1)$ and $LTS^H(E_1 \setminus Act_H)$.

Example 3. Consider again process $E_1 = l.h.j.\mathbf{0} + l.(\tau.j.\mathbf{0} + \tau.\mathbf{0})$ of Example 1. In Fig. 3 we show $LTS(E_1)$ and $LTS(E_1 \setminus Act_H)$. By performing the closure up to Act_H (Algorithm 1) we obtain the transformed labelled transition systems $LTS^H(E_1)$ and $LTS^H(E_1 \setminus Act_H)$ reported in Fig. 4. In particular, the first step just adds the τ-loops in every state; the second one, adds two transitions labelled with l corresponding to $l.\tau$ and one transition labelled with j corresponding to $\tau.j$; finally, step 3 adds a h-labelled transition every time there is a τ transition. The two transformed transition systems are not strongly bisimilar: the leftmost node after l in $LTS^H(E_1)$ is not bisimilar to any node in $LTS^H(E_1 \setminus Act_H)$, since in $LTS^H(E_1 \setminus Act_H)$ all the nodes are either "sink-nodes" (which only executes τ and h loops) or they have at least one outgoing edge with label j or l. Indeed, that node in $LTS^H(E_1)$ may execute only h and τ actions and could thus be simulated only by sink-nodes in $LTS^H(E_1 \setminus Act_H)$. However, differently from sink-nodes, after one h, it is also able to execute a j. This proves that

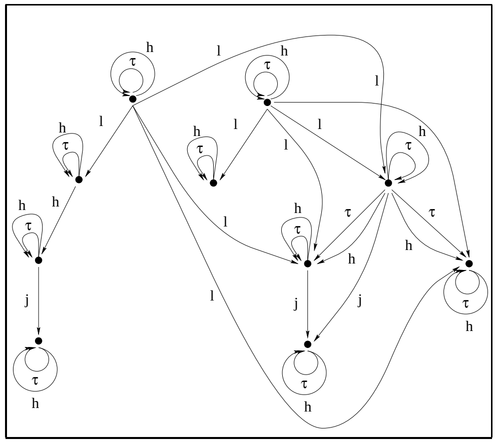

Fig. 5. The labelled transition systems $LTS^H(E_1)$ and $LTS^H(E_1 \setminus Act_H)$ with sharing.

$E_1^H \not\approx (E_1 \setminus Act_H)^H$, thus, by Corollary 2, $E_1 \notin P\_BNDC$. In Fig. 5 we show again $LTS^H(E_1)$ and $LTS^H(E_1 \setminus Act_H)$, now sharing the common states, i.e., we avoid to repeat the states (and the sub-LTS's) which do not reach an action h.

5 Conclusions

We consider the security property *P_BNDC* and we present two methods to prove it. While the first method exploit model checkers for the μ-calculus, the second one is based on the use of bisimulation algorithms. We show that this second approach can perform the *P_BNDC*-check in polynomial time with respect to the number of states of the system and improves on the polynomial time complexity of the Compositional Security Checker CoSeC presented in [9].

References

1. A. Bouali. XEVE, an ESTEREL verification environment. In A. J. Hu and M. Y. Vardi, editors, *Proc. of Int. Conference on Computer Aided Verification (CAV'98)*, volume 1427 of *LNCS*, pages 500–504. Springer, 1998.
2. A. Bouali and R. de Simone. Symbolic bisimulation minimization. In G. von Bochmann and D. K. Probst, editors, *Proc. of Int. Conference on Computer Aided Verification (CAV'92)*, volume 663 of *LNCS*, pages 96–108. Springer, 1992.
3. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model checking*. The MIT Press, 1999.
4. R. Cleaveland and S. Sims. The NCSU concurrency workbench. In R. Alur and T. Henzinger, editors, *Proc. of Int. Conference on Computer Aided Verification (CAV'96)*, volume 1102 of *LNCS*, pages 394–397. Springer, 1996.
5. D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progression. In *Proc. of the 19th Symposium on Theory of Computing*, pages 1–6, 1987.

6. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, 1990.
7. A. Dovier, C. Piazza, and A. Policriti. A fast bisimulation algorithm. In G. Berry, H. Comon, and A. Finkel, editors, *Proc. of Int. Conference on Computer Aided Verification (CAV'01)*, volume 2102 of *LNCS*, pages 79–90. Springer, 2001.
8. R. Focardi and R. Gorrieri. A Classification of Security Properties for Process Algebras. *Journal of Computer Security*, 3(1):5–33, 1994/1995.
9. 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.
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. A Security Property for Processes in Dynamic Contexts. In *Proc. of Workshop on Issues in the Theory of Security (WITS '02)*.To appear.
12. R. Focardi and S. Rossi. Information Flow Security in Dynamic Contexts. Technical Report CS-2001-16, Dipartimento di Informatica, Università Ca' Foscari di Venezia, Italy, 2001.
13. J. A. Goguen and J. Meseguer. Security Policy and Security Models. In *Proc. of the 1982 Symposium on Security and Privacy*, pages 11–20. IEEE Computer Society Press, 1982.
14. A. Goralcikova and V. Koubek. A reduct and closure algorithm for graphs. In *Proc. of Mathematical Foundations of Computer Science (MFCS'79)*, volume 74 of *LNCS*, pages 301–307. Springer, 1979.
15. P. C. Kannellakis and S. A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Information and Computation*, 86(1):43–68, 1990.
16. D. Kozen. Results on the Propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.
17. D. Lee and M. Yannakakis. Online minimization of transition systems. In *Proc. of 24th ACM Symposium on Theory of Computing (STOC'92)*, pages 264–274. ACM Press, 1992.
18. D. Long, A. Browne, E. Clarke, S. Jha, and W. Marrero. An improved Algorithm for the Evaluation of Fixpoint expressions. In D. L. Dill, editor, *Proc. of Int. Conference on Computer Aided Verification (CAV'94)*, volume 818 of *LNCS*, pages 338–350. Springer, 1994.
19. A. Mader. Modal μ-calculus, Model Checking, and Gauss elimination. In E. Brinksma, R. Cleaveland, K.G. T. Margaria Larsen, and B. Steffen, editors, *Proc. of Int. Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS'95)*, volume 1019 of *LNCS*, pages 72–88. Springer, 1995.
20. R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
21. M. Müller-Olm. Derivation of Characteristic Formulae. *Electronic Notes in Theoretical Computer Science*, 18, 1998.
22. R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.
23. A. W. Roscoe. *The Theory and Practice of Concurrency*. Series in Computer Science. Prentice Hall, 1998.
24. B. Steffen and A. Ingòlfsdòttir. Characteristic Formulae for Processes with Divergence. *Information and Computation*, 110(1):149–163, 1994.