A Theory of Noninterference for the π-calculus*

Silvia Crafa and Sabina Rossi

Dipartimento di Informatica, Università Ca' Foscari di Venezia
e-mail: {silvia,srossi}@dsi.unive.it

Abstract. We develop a theory of noninterference for a typed version of the π-calculus where types are used to assign secrecy levels to channels. We provide two equivalent characterizations of noninterference based on a typed behavioural equivalence relative to a security level σ, which captures the idea of external observers of level σ. The first characterization involves a universal quantification over all the possible *active attacks*, i.e., malicious processes which interact with the system possibly leaking secret information. The second definition of noninterference is expressed in terms of an unwinding condition, which deals with so-called *passive attacks* trying to infer confidential information just by observing the behaviour of the system. This unwinding-based characterization naturally leads to efficient methods for the verification and construction of (compositional) secure systems. Furthermore, we characterize noninterference in terms of bisimulation-like (partial) equivalence relations in the style of a stream of similar studies for other process calculi (e.g., CCS and CryptoSPA) and languages (e.g., imperative and multi-threaded languages).

1 Introduction

A central issue of multilevel security systems is the protection of sensitive data and resources from undesired access. Information flow security properties have been proposed as a means to provide strong guarantees of confidentiality of secret information. These properties impose constraints on information flow ensuring that no information can flow from a higher to a lower security level. Since Denning and Denning's work [6], information flow analysis has been studied for various programming languages, including imperative languages [6, 20, 23], functional languages [10, 18] and concurrent languages [4, 7, 14–17, 19, 22, 25].

One of the most successful approaches to information flow security relies on the notion of *Noninterference* [9]. The basic idea is that a system is interference free if the low level observation of the system is independent from the behaviour of its high components. Recently, various type-based proof techniques for the π-calculus have been proposed [11, 14–17]. In these works type systems are actually part of the definition of noninterference, in that both the observation of the system and the observed processes are constrained by types. A soundness theorem is then proved stating that if a system is well-typed, then no change in the behaviour of its high components can affect the low level view of the system.

* Supported by the EU-FET project IST-2001-32617 and the FIRB project RBAU018RCZ.

In this paper we wish to define a general theory of noninterference for the π-calculus, where the use of types is much lighter. In particular, the only typing constraint we impose is that values at a given security clearance cannot flow through channels with a lower security level. Such a typing discipline ensures that information does not explicitly flow from high to low. Instead, implicit flows are not dealt with the type system, and then we cannot use it as a proof technique for noninterference. On the contrary, we characterize noninterference in terms of the actions that typed processes may perform.

Our approach intends to generalize previous ideas, mainly developed for CCS, to the π-calculus, where new difficulties arise due to the presence of scope extrusion. The contribution of this paper is twofold: (i) we develop a rich and elegant theory of noninterference intrinsic of the π-calculus, almost independent of types, and (ii) we find a number of sound and complete characterizations of secure processes leading to efficient verification techniques.

The noninterference property we are going to study is based on the notion of process behaviour relative to a security level σ, taken from a complete lattice $\langle \Sigma, \preceq \rangle$ of security annotations. We define typed equivalences for the π-calculus relative to an observation level σ, namely σ-reduction barbed congruences (see [12]). Two processes P, Q are σ-equivalent in the type environment Γ, written $\Gamma \vDash P \cong_\sigma Q$, if they exhibit the same σ-level behaviour, i.e., they are indistinguishable for a σ-level observer.

A σ-level observer is formalized as a σ-context, i.e., a well typed context which can interact with the observed process only through channels of level at most σ. We require $\cong_\sigma$ to be a congruence for all σ-level contexts.

We also develop a proof technique for $\cong_\sigma$ in terms of a quite natural bisimilarity on σ-actions defined on typed labelled transition systems. A typed LTS is built around typed actions of the form $\Gamma \rhd P \xrightarrow{\alpha}_\delta \Gamma' \rhd P'$ indicating that in the type environment Γ, the process P performs the action α of level δ and evolves to P' in the possibly modified environment Γ'. We prove that two processes are σ-barbed congruent if and only if they are bisimilar on typed actions of level σ.

Relying on this equational theory for the π-calculus, we introduce the noninterference property $\mathcal{NI}(\cong_\sigma)$ for typed processes, which is inspired by the *P_BNDC* property defined in [8] for CCS. We say that a process P in a type environment Γ satisfies the property $\mathcal{NI}(\cong_\sigma)$, written $\Gamma \rhd P \in \mathcal{NI}(\cong_\sigma)$, if for every configuration $\Gamma' \rhd P'$ reachable from $\Gamma \rhd P$ in the typed LTS, and for every σ-high level source H (that is a process which can perform only actions at level higher than σ) it holds

$$\Gamma' \rhd P' \cong_\sigma \Gamma' \rhd P' \mid H.$$

This definition involves a universal quantification over all the possible *active attacks*, i.e., high level malicious processes H which interact with the system possibly leaking secret information. Moreover, it is *persistent* in the sense that if a configuration satisfies $\mathcal{NI}(\cong_\sigma)$ then also all the configurations reachable from it in the typed LTS satisfy $\mathcal{NI}(\cong_\sigma)$. As discussed in [8], persistence is technically useful since it allows us to apply inductive reasoning when proving security results (e.g., compositionality), but it is also intuitively motivated by the need for mobile processes to be secure at any computation step.

We provide a first characterization of $\mathcal{NI}(\cong_\sigma)$ in terms of an *unwinding* condition in the style of [2]. The unwinding condition aims at specifying local constraints on process transitions which imply the global security property. More precisely, we require that whenever a configuration C performs a typed action of level higher than σ moving to C', then a configuration C'' can also be reached through an internal computation such that C' and C'' are indistinguishable for a σ-level observer. In other words, the unwinding condition ensures that the σ-high actions are always simulated by internal computations, thus becoming invisible for the low level observers.

It is interesting to observe that the unwinding condition characterizes security with respect to the so-called *passive attacks*, which try to infer information about the classified behaviour (σ-high actions) just by observing the σ-level behaviour of the system. Thanks to this characterization, the noninterference property $\mathcal{NI}(\cong_\sigma)$ becomes decidable for finite state processes, i.e., processes whose typed LTS is finite. Furthermore, we show that $\mathcal{NI}(\cong_\sigma)$ is compositional with respect to most of the operators of the π-calculus. In particular, if P and Q satisfy $\mathcal{NI}(\cong_\sigma)$ then $P \mid Q$ and $!P$ also do.

We further develop two quantifier-free characterizations of noninterference based on bisimulation-like (partial) equivalence relations. More precisely, we first introduce a partial equivalence relation $\dot{\approx}_\sigma$ (*per* model) over configurations and, inspired by the definitions in [20] for imperative and multi-threaded languages, we prove that $\dot{\approx}_\sigma$ is reflexive only on the set of secure processes. Hence, we obtain that a typed process P is secure if and only if P is $\dot{\approx}_\sigma$-equivalent to itself. Then we investigate the impact of name restriction on noninterference. Let $(\boldsymbol{\nu}^\sigma)P$ be the process P where all its σ-high free names are restricted. We define the equivalence relation $\ddot{\approx}_\sigma$ and prove that a typed process P is secure if and only if P and $(\boldsymbol{\nu}^\sigma)P$ are $\ddot{\approx}_\sigma$-equivalent. Finally we show that two well typed processes P and Q are equivalent on σ-actions if and only if $(\boldsymbol{\nu}^\sigma)P$ and $(\boldsymbol{\nu}^\sigma)Q$ are equivalent on every action. This property allows us to precisely relate the standard bisimulation equivalence $\approx$ for the π-calculus with our bisimulation on σ-actions and also to express our noninterference property in terms of the equivalence relation $\approx$.

The rest of the paper is organized as follows. In Section 2 we present the language, its semantics and the type system. In Section 3 we study typed observation equivalences relative to a security level. In Section 4 we introduce the notion of σ-noninterference and provide a number of characterizations based on typed actions. Section 5 concludes the paper discussing some related work.

All the proofs of the results presented in this paper are available in [5].

2 The Language

In this section we introduce the language, its operational semantics and the type system with which we will be concerned.

We presuppose a countably-infinite set of names and a countably-infinite set of variables ranged over by $n, .., q$ and by $x, .., z$, respectively. We often use a, b, c to range over both names and variables. We also assume a complete lattice $\langle \Sigma, \preceq \rangle$ of security annotations, ranged over by σ, δ, where $\top$ and $\bot$ represent the top and the bottom elements of the lattice. The syntax of processes and types is shown in Table 1. It is a

Prefixes		*Processes*	
$\pi ::= \overline{a}\langle b\rangle$	output	$P ::= \pi.P$	prefix
$\mid \; a(x:T)$	input	$\mid$ if $a = b$ then P else P	matching
		$\mid \; P \mid P$	parallel
Types		$\mid \; (\boldsymbol{\nu} n:T)P$	restriction
$T ::= \sigma[\,]$		$\mid \; !P$	replication
$\mid \; \sigma[T]$		$\mid \; \mathbf{0}$	inactive

Table 1. Syntax

synchronous[1], monadic, calculus with the match/mismatch operator. As explained in [12], the matching construct is essential for the coinductive characterization of the reduction barbed congruence shown in Section 3.

As usual, the input construct $a(x : T).P$ acts as a binder for the variable x in P, while the restriction $(\boldsymbol{\nu} n : T)P$ acts as a binder for the name n in P. We identify processes up to α-conversion. We use $\mathrm{fn}(P)$ and $\mathrm{fv}(P)$ to denote the set of free names and free variables, respectively, in P. We write $P\{x := n\}$ to denote the substitution of all free occurrences of x in P with n, and we often write $a(x{:}T), \overline{a}\langle b\rangle$ omitting trailing $\mathbf{0}$'s. In this paper we restrict to *closed* processes, i.e., processes containing no free occurrences of variables; in Section 5 we discuss how to extend our theory to open terms.

Types are used to assign security levels to channels. More precisely, if $\sigma \in \Sigma$, then $\sigma[\,]$ is the type of channels of level σ which carry no values, while $\sigma[T]$ is the type of channels of level σ which carry values of type T. We consider the function Λ associating to types the corresponding level, that is $\Lambda(\sigma[\,]) = \sigma = \Lambda(\sigma[T])$.

Semantics. The operational semantics of our language is given in terms of a labelled transition system (LTS) defined over processes. The set of labels, or actions, is the following:

Actions	$\alpha ::= \overline{n}\langle m\rangle$	send a name
	$\mid \; (\boldsymbol{\nu} m{:}T)\,\overline{n}\langle m\rangle$	send a fresh name
	$\mid \; n(m)$	receive a name
	$\mid \; \tau$	internal action

We write $\mathrm{fn}(\alpha)$ and $\mathrm{bn}(\alpha)$ to denote the set of free and bound names occurring in the action α, where $\mathrm{bn}(\alpha) = \{m\}$ if $\alpha = (\boldsymbol{\nu} m{:}T)\,\overline{n}\langle m\rangle$, and $\mathrm{bn}(\alpha) = \emptyset$ otherwise. The LTS is defined in Table 2 and it is entirely standard; we just omitted the symmetric rules for (SUM), (PAR), (COMM) and (CLOSE) in which the role of the left and right components are swapped.

[1] We consider the synchronous calculus since it allows for more interferences. Nevertheless, our results can be adapted to the asynchronous, polyadic calculus.

(OUT) $\dfrac{}{\overline{n}\langle m\rangle.P \xrightarrow{\overline{n}\langle m\rangle} P}$

(IN) $\dfrac{}{n(x:T).P \xrightarrow{n(m)} P\{x := m\}}$

(MATCH) $\dfrac{}{\mathsf{if}\ n = n\ \mathsf{then}\ P\ \mathsf{else}\ Q \xrightarrow{\tau} P}$

(MISMATCH) $\dfrac{n \neq m}{\mathsf{if}\ n = m\ \mathsf{then}\ P\ \mathsf{else}\ Q \xrightarrow{\tau} Q}$

(PAR) $\dfrac{P \xrightarrow{\alpha} P' \quad \mathrm{bn}(\alpha) \cap \mathrm{fn}(Q) = \emptyset}{P \mid Q \xrightarrow{\alpha} P' \mid Q}$

(COMM) $\dfrac{P \xrightarrow{\overline{n}\langle m\rangle} P' \quad Q \xrightarrow{n(m)} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$

(CLOSE) $\dfrac{P \xrightarrow{(\boldsymbol{\nu} m{:}T)\,\overline{n}\langle m\rangle} P' \quad Q \xrightarrow{n(m)} Q' \quad m \notin \mathrm{fn}(Q)}{P \mid Q \xrightarrow{\tau} (\boldsymbol{\nu} m{:}T)(P' \mid Q')}$

(OPEN) $\dfrac{P \xrightarrow{\overline{n}\langle m\rangle} P' \quad m \neq n}{(\boldsymbol{\nu} m{:}T)P \xrightarrow{(\boldsymbol{\nu} m{:}T)\,\overline{n}\langle m\rangle} P'}$

(RES) $\dfrac{P \xrightarrow{\alpha} P' \quad n \notin \mathrm{fn}(\alpha) \cup \mathrm{bn}(\alpha)}{(\boldsymbol{\nu} n{:}T)P \xrightarrow{\alpha} (\boldsymbol{\nu} n{:}T)P'}$

(REP-ACT) $\dfrac{P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P' \mid !P}$

Table 2. Labelled Transition System

Type System. Our type system corresponds to the basic type system for the π-calculus (see [21]). The main judgements take the form $\Gamma \vdash P$, where Γ is a type environment, that is a finite mapping from names and variables to types. Intuitively, $\Gamma \vdash P$ means that the process P uses all channels as input/output devices in accordance with their types, as given in Γ. The other, auxiliary, judgements are $\Gamma \vdash a : T$ stating that the name/variable a has type T in Γ, and $\Gamma \vdash \diamond$ stating that the type environment Γ is well formed. The typing rules are collected in Table 3, and they are based on the following rules of type formation, which prevent a channel of level δ from carrying values of level higher than δ.

(EMPTY TYPE) $\dfrac{}{\vdash \delta[\,]}$

(CHANNEL TYPE) $\dfrac{\vdash T \quad \Lambda(T) \preceq \delta}{\vdash \delta[T]}$

Notice that the type formation rules guarantee the absence of any explicit flow of information from a higher to a lower security level: for instance, the process $\overline{pub}\langle passwd\rangle.\mathbf{0}$ where a secret password is forwarded along a public channel, is not well-typed.

$$\frac{}{\emptyset \vdash \diamond}\ (\textsc{Empty}) \qquad \frac{\Gamma \vdash \diamond \quad \vdash T \quad a \notin Dom(\Gamma)}{\Gamma, a : T \vdash \diamond}\ (\textsc{Env}\ a) \qquad \frac{\Gamma, a : T \vdash \diamond}{\Gamma, a : T \vdash a : T}\ (\textsc{Project})$$

$$\frac{\Gamma \vdash a : \delta[T] \quad \Gamma \vdash b : T \quad \Gamma \vdash P}{\Gamma \vdash \overline{a}\langle b \rangle.P}\ (\textsc{Output}) \qquad \frac{\Gamma \vdash a : \delta[T] \quad \Gamma, x : T \vdash P}{\Gamma \vdash a(x : T).P}\ (\textsc{Input})$$

$$\frac{\Gamma \vdash a : \delta[T] \quad \Gamma \vdash b : \delta[T] \quad \Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash \mathsf{if}\ a = b\ \mathsf{then}\ P\ \mathsf{else}\ Q}\ (\textsc{Match}) \qquad \frac{\Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash P \mid Q}\ (\textsc{Para})$$

$$\frac{\Gamma, n : T \vdash P}{\Gamma \vdash (\boldsymbol{\nu} n : T)P}\ (\textsc{Res}) \qquad \frac{\Gamma \vdash P}{\Gamma \vdash\, !P}\ (\textsc{Repl}) \qquad \frac{\Gamma \vdash \diamond}{\Gamma \vdash \mathbf{0}}\ (\textsc{Dead})$$

Table 3. Type System

3 Observation Equivalences relative to a Security Level

In this section we introduce the notion of σ-level observation equivalence and we develop an equational theory for the π-calculus which is parametric on the security level (i.e., the observational power) of the observers.

Our equivalences are reminiscent of the typed behavioural equivalences for the π-calculus [1, 12, 14, 21]: they are equivalences indexed by a type environment Γ ensuring that both the observed process and the observer associate the same security levels to the same names. Our equivalences, however, are much simpler than those in the above mentioned works since we do not consider subtyping nor linearity/affinity.

Our type-indexed relations are based on the notion of configuration. We say that $\Gamma \triangleright P$ is a *configuration* if Γ is a type environment and P is a process such that $\Gamma \vdash P$ [2]. A type-indexed relation over processes is a family of binary relations between processes indexed by type environments. We write $\Gamma \vDash P\ \mathcal{R}\ Q$ to mean that P and Q are related by $\mathcal{R}$ at Γ and $\Gamma \triangleright P$ and $\Gamma \triangleright Q$ are configurations.

To define our σ-level observation equivalences, we will ask for the largest type-indexed relation over processes which satisfies the following properties.

[2] The two notations $\Gamma \triangleright P$ and $\Gamma \vdash P$ are essentially the same; however, we prefer to keep them distinct to make it uniform with the literature.

Reduction Closure. A type-indexed relation $\mathcal{R}$ over processes is *reduction closed* if $\Gamma \vDash P\,\mathcal{R}\,Q$ and $P \xrightarrow{\tau} P'$ imply that there exists Q' such that $Q \Longrightarrow Q'$ and $\Gamma \vDash P'\,\mathcal{R}\,Q'$, where $\Longrightarrow$ denotes the reflexive and transitive closure of $\xrightarrow{\tau}$.

σ-Barb Preservation. Let $\sigma \in \Sigma$, P be a process and Γ a type environment such that $\Gamma \vdash P$. We write $\Gamma \vDash P \downarrow^{\sigma}_{n}$ if $P \xrightarrow{\overline{n}\langle m\rangle}$ with $\Lambda(\Gamma(n)) \preceq \sigma$. We also write $\Gamma \vDash P \Downarrow^{\sigma}_{n}$ if there exists some P' such that $P \Longrightarrow P'$ and $\Gamma \vDash P' \downarrow^{\sigma}_{n}$. A type-indexed relation $\mathcal{R}$ over processes is *σ-barb preserving* if $\Gamma \vDash P\,\mathcal{R}\,Q$ and $\Gamma \vDash P \downarrow^{\sigma}_{n}$ imply $\Gamma \vDash Q \Downarrow^{\sigma}_{n}$.

σ-Contextuality. Let a typed context be a process with at most one typed hole $[\cdot_\Gamma]$. If $C[\cdot_\Gamma]$ is a typed context and P is a process such that $\Gamma \vdash P$, then we write $C[P]$ for the process obtained by replacing the hole in $C[\cdot_\Gamma]$ by P. In order to type contexts, the type system of Table 3 is extended with the following rule:

$$\text{(CTX)}\qquad \frac{}{\Gamma, \Gamma' \vdash [\cdot_\Gamma]}$$

Proposition 1. *Let $\Gamma \vdash P$ and $\Gamma, \Gamma' \vdash C[\cdot_\Gamma]$, then $\Gamma, \Gamma' \vdash C[P]$.*

We are interested in σ-contexts that capture the idea of σ-level observers. Intuitively, a σ-context is an evaluation context which may interact with the process filling the hole just through channels of level at most σ.

Definition 1 (σ-context). *Let $\sigma \in \Sigma$. A context $C[\cdot_\Gamma]$ is a σ-context if there exists a type environment Γ' such that $\Gamma, \Gamma' \vdash C[\cdot_\Gamma]$ and $C[\cdot_\Gamma]$ is generated by the following grammar*

$$C[\cdot_\Gamma] \;::=\; [\cdot_\Gamma] \;\mid\; (\boldsymbol{\nu} n{:}T)C[\cdot_\Gamma] \;\mid\; C[\cdot_\Gamma] \mid P \;\mid\; P \mid C[\cdot_\Gamma]$$

where P is a process such that $\forall n \in \mathrm{fn}(P)$ we have $\Lambda(\Gamma, \Gamma'(n)) \preceq \sigma$.

Example 1. Let Γ be the type environment $h : \top[\bot[\,]],\ \ell : \bot[\,]$ and $\sigma \prec \top$. The context $(\boldsymbol{\nu} h)(\overline{h}\langle\ell\rangle \mid [\cdot_\Gamma])$ is not a σ-context since the process $\overline{h}\langle\ell\rangle$ in parallel with the hole has a free occurrence of the high name h. This context does not represent a σ-level observer since it can interact with a process filling the hole through the high channel h. On the other hand, $(\boldsymbol{\nu} h)(\overline{h}\langle\ell\rangle) \mid [\cdot_\Gamma]$ is a σ-context.

We say that a type-indexed relation $\mathcal{R}$ over processes is *σ-contextual* if $\Gamma \vDash P\,\mathcal{R}\,Q$ and $\Gamma, \Gamma' \vdash C[\cdot_\Gamma]$ imply $\Gamma, \Gamma' \vDash C[P]\,\mathcal{R}\,C[Q]$ for all σ-contexts $C[\cdot_\Gamma]$.

Definition 2 (σ-Reduction Barbed Congruence $\cong_\sigma$). *Let $\sigma \in \Sigma$. The* σ-reduction barbed congruence, *denoted by $\cong_\sigma$, is the largest type-indexed relation over processes which is symmetric, σ-contextual, reduction closed and σ-barb preserving.*

The following proposition is immediate.

Proposition 2. *Let $\sigma \in \Sigma$, Γ be a type environment and P, Q be processes such that $\Gamma \vdash P, Q$. If $\Gamma \vDash P \cong_\sigma Q$ then $\Gamma \vDash P \cong_{\sigma'} Q$ for all $\sigma' \preceq \sigma$. In particular, $\Gamma \vDash P \cong_\top Q$ implies $\Gamma \vDash P \cong_\sigma Q$ for all $\sigma \in \Sigma$.*

3.1 A bisimulation-based proof technique

In this section we develop a proof technique for the equivalences $\cong_\sigma$ defined above. More precisely, following [1, 11, 12], we define a LTS of *typed actions* (called typed LTS) over configurations. As in [11], actions are parameterized over security levels and take the form

$$\Gamma \triangleright P \xrightarrow{\alpha}_\delta \Gamma' \triangleright P'$$

indicating that the process P in the type environment Γ can perform the action α to interact with some δ-level observer. In this case, we say that α is a δ-*level* action.

The rules of the typed LTS are obtained from those in Table 2 by taking into account the type environment Γ which records the security levels of the channels used by the process. Differently from [11], our typed actions are built around just a single type environment Γ constraining the observed process P. This differs from [11] where, due to the presence of subtyping, two distinct type environments are needed, one for the observer and the other for the observed process.

The rules of the typed LTS are reported in Table 4; note that there is an additional input action of the form $(\boldsymbol{\nu} m{:}T)\, n(m)$ occurring when the process receives a new name m generated by the environment.
Relying on the typed LTS, we now introduce the *bisimilarity on σ-actions* which provides a coinductive characterization of σ-reduction barbed congruence $\cong_\sigma$.

With an abuse of notation, we write $\Longrightarrow$ for the reflexive and transitive closure of $\xrightarrow{\tau}_\delta$. We also write $\stackrel{\alpha}{\Longrightarrow}_\delta$ for $\Longrightarrow \xrightarrow{\alpha}_\delta \Longrightarrow$, and $\stackrel{\hat{\alpha}}{\Longrightarrow}_\delta$ for $\Longrightarrow$ if $\alpha = \tau$ and $\stackrel{\alpha}{\Longrightarrow}_\delta$ otherwise.

Definition 3 (Bisimilarity on σ-actions $\approx_\sigma$). *Let $\sigma \in \Sigma$. Bisimilarity on σ-actions is the largest symmetric relation $\approx_\sigma$ over configurations, such that whenever $(\Gamma \triangleright P) \approx_\sigma (\Gamma \triangleright Q)$, if $\Gamma \triangleright P \xrightarrow{\alpha}_\sigma \Gamma' \triangleright P'$, then there exists Q' such that $\Gamma \triangleright Q \stackrel{\hat{\alpha}}{\Longrightarrow}_\sigma \Gamma' \triangleright Q'$ and $(\Gamma' \triangleright Q') \approx_\sigma (\Gamma' \triangleright P')$.*

In the following, for a given relation $\mathcal{R}$ over configurations, we write $\Gamma \vDash P \,\mathcal{R}\, Q$ whenever $(\Gamma \triangleright P)\, \mathcal{R}\, (\Gamma \triangleright Q)$.

Theorem 1. *Let $\sigma \in \Sigma$, Γ be a type environment and P, Q be processes such that $\Gamma \vdash P, Q$. $\Gamma \vDash P \cong_\sigma Q$ if and only if $\Gamma \vDash P \approx_\sigma Q$.*

4 Noninterference

In this section we introduce a notion of noninterference for processes of the typed π-calculus which uses the σ-reduction barbed congruence $\cong_\sigma$ as observation equivalence. This property, called $\mathcal{NI}(\cong_\sigma)$, is inspired by the *P_BNDC* property defined in [8] for CCS processes; it requires that no information flow should occur even in the presence of *active* malicious processes, e.g., Trojan Horse programs, that run at the classified (higher than σ) level.

We start by introducing the following notations:

(OUT)
$$\frac{\Gamma \vdash n : \delta_1[T] \quad \delta_1 \preceq \delta}{\Gamma \triangleright \overline{n}\langle m\rangle.P \xrightarrow{\overline{n}\langle m\rangle}_\delta \Gamma \triangleright P}$$

(IN)
$$\frac{\Gamma \vdash n : \delta_1[T] \quad \Gamma \vdash m : T \quad \delta_1 \preceq \delta}{\Gamma \triangleright n(x{:}T).P \xrightarrow{n(m)}_\delta \Gamma \triangleright P\{x := m\}}$$

(WEAK)
$$\frac{\Gamma, m : T \triangleright P \xrightarrow{n(m)}_\delta \Gamma' \triangleright P'}{\Gamma \triangleright P \xrightarrow{(\boldsymbol{\nu} m:T)\, n(m)}_\delta \Gamma' \triangleright P'}$$

(PAR)
$$\frac{\Gamma \triangleright P \xrightarrow{\alpha}_\delta \Gamma' \triangleright P' \quad \mathrm{bn}(\alpha) \cap \mathrm{fn}(Q) = \emptyset}{\Gamma \triangleright P \mid Q \xrightarrow{\alpha}_\delta \Gamma' \triangleright P' \mid Q}$$

(RED)
$$\frac{P \xrightarrow{\tau} P'}{\Gamma \triangleright P \xrightarrow{\tau}_\delta \Gamma \triangleright P'}$$

(OPEN)
$$\frac{\Gamma, m{:}T \triangleright P \xrightarrow{\overline{n}\langle m\rangle}_\delta \Gamma' \triangleright P' \quad m \neq n}{\Gamma \triangleright (\boldsymbol{\nu} m{:}T)P \xrightarrow{(\boldsymbol{\nu} m:T)\, \overline{n}\langle m\rangle}_\delta \Gamma' \triangleright P'}$$

(RES)
$$\frac{\Gamma, n{:}T \triangleright P \xrightarrow{\alpha}_\delta \Gamma', n{:}T \triangleright P' \quad n \notin \mathrm{fn}(\alpha) \cup \mathrm{bn}(\alpha)}{\Gamma \triangleright (\boldsymbol{\nu} n{:}T)P \xrightarrow{\alpha}_\delta \Gamma' \triangleright (\boldsymbol{\nu} n{:}T)P'}$$

(REP-ACT)
$$\frac{\Gamma \triangleright P \xrightarrow{\alpha}_\delta \Gamma' \triangleright P'}{\Gamma \triangleright !P \xrightarrow{\alpha}_\delta \Gamma' \triangleright P' \mid !P}$$

Table 4. Typed LTS for π-calculus

- We say that a configuration $\Gamma' \triangleright P'$ is *reachable* from a configuration $\Gamma \triangleright P$, written $\Gamma \triangleright P \rightsquigarrow \Gamma' \triangleright P'$, if there exist $n \geq 0$, $\alpha_1, \ldots, \alpha_n$ and $\sigma_1, \ldots, \sigma_n$ such that $\Gamma \triangleright P \xrightarrow{\alpha_1}_{\sigma_1} \xrightarrow{\alpha_2}_{\sigma_2} \cdots \xrightarrow{\alpha_n}_{\sigma_n} \Gamma' \triangleright P'$. (Notice that the concept of reachability is independent from the levels σ_i.)
- Given a type environment Γ, we say that a process P is a *σ-high level source* in Γ, written $P \in \mathcal{H}^\sigma_\Gamma$, if $\Gamma \vdash P$ and either $\Gamma \triangleright P \not\xrightarrow{\alpha}_\delta$ (i.e., $\Gamma \triangleright P$ does not perform any action) or if $\Gamma \triangleright P \xrightarrow{\alpha}_\delta \Gamma' \triangleright P'$ then $\sigma \prec \delta$ and $\Gamma' \triangleright P'$ is a σ-high level source. In other words, a σ-high level source can only perform δ-level actions with $\sigma \prec \delta$. Notice that this definition does not prevent a σ-high level source from communicating σ-low values (along σ-high channels).
- Given a security level $\sigma \in \Sigma$, we write $\Gamma \triangleright P \xrightarrow{\alpha}{}^\sigma \Gamma' \triangleright P'$ (with a superscript σ) if whenever $\Gamma \triangleright P \xrightarrow{\alpha}_\delta \Gamma' \triangleright P'$ then $\sigma \prec \delta$. In this case we say that $\Gamma \triangleright P$ has performed a *σ-high level action*. We define $\stackrel{\hat{\alpha}}{\Longrightarrow}{}^\sigma$ accordingly.

A process P in a type environment Γ satisfies the property $\mathcal{NI}(\cong_\sigma)$ if for every configuration $\Gamma' \triangleright P'$ reachable from $\Gamma \triangleright P$ and for every σ-high level source H, a σ-level user

cannot distinguish, in the sense of $\cong_\sigma$, $\Gamma' \triangleright P'$ from $\Gamma' \triangleright P' \mid H$. The formal definition of $\mathcal{NI}(\cong_\sigma)$ is as follows.

Definition 4 (σ-Noninterference). *Let $\sigma \in \Sigma$, P be a process and Γ be a type environment such that $\Gamma \vdash P$. The process P satisfies the* σ-noninterference property in Γ, *written $\Gamma \triangleright P \in \mathcal{NI}(\cong_\sigma)$, if for all $\Gamma' \triangleright P'$ such that $\Gamma \triangleright P \rightsquigarrow \Gamma' \triangleright P'$ and for all $H \in \mathcal{H}^\sigma_{\Gamma'}$ it holds $\Gamma' \vDash P' \cong_\sigma P' \mid H$.*

Example 2. In the following examples, we assume just two security levels: H and L with $\mathsf{L} \prec \mathsf{H}$; let also h be a high level channel and ℓ, ℓ_1, ℓ_2 be low level channels. Let Γ be the type environment $h : \mathsf{H}[\,],\ \ell : \mathsf{L}[\,],\ \ell_1 : \mathsf{L}[\,],\ \ell_2 : \mathsf{L}[\,]$ and $\sigma = \mathsf{L}$.

Let us first consider the following simple insecure process: $P_1 = h().\ell() \mid \overline{h}\langle\rangle$. To show that $\Gamma \triangleright P_1 \notin \mathcal{NI}(\cong_\sigma)$ it is sufficient to consider the configuration $\Gamma \triangleright P_1'$ with $P_1' = h().\ell()$ that is reachable from $\Gamma \triangleright P_1$ after performing the output action $\overline{h}\langle\rangle$. The process P_1' is clearly insecure in the type environment Γ since the low level, observable, action $\ell()$ directly depends on the high level input $h()$. Indeed, by choosing $H = \overline{h}\langle\rangle$ one can easily observe that $\Gamma \vDash P_1' \not\cong_\sigma P_1' \mid H$.

Let us consider a further classic example of insecure process, that is $P_2 = h(x : T).\mathsf{if}\ x = n\ \mathsf{then}\ \overline{\ell_1}\langle\rangle\ \mathsf{else}\ \overline{\ell_2}\langle\rangle$ in the type environment $\Gamma' = h : \mathsf{H}[T],\ \ell_i : \mathsf{L}[\,],\ n : T$ (here the security level of n is irrelevant). To show that $\Gamma' \triangleright P_2 \notin \mathcal{NI}(\cong_\sigma)$ one can choose $H = \overline{h}\langle n\rangle$, where $H \in \mathcal{H}^\sigma_{\Gamma'}$ independently on the level of n, and observe that $\Gamma' \vdash P_2 \not\cong_\sigma P_2 \mid H$. Intuitively, when n is a high level name, a low level observer may infer from P_2 the value of the high level variable x, which is clearly unsound.

Finally, consider the process $P_3 = P_2 \mid \overline{h}\langle n\rangle \mid \overline{h}\langle m\rangle$, where the variable x can be nondeterministically substituted either with n or m. P_3 is still an insecure process since an external attack can destroy the nondeterminism causing an interference: for instance, if $H = h(y).h(z).\overline{h}\langle n\rangle$, then $\Gamma' \vDash P_3 \not\cong_\sigma P_3 \mid H$.

Building on the ideas developed in [2] for a class of persistent noninterference properties for CCS processes, we provide a characterization of $\mathcal{NI}(\cong_\sigma)$ in terms of an unwinding condition. Intuitively, the unwinding condition specifies local constraints on the typed actions of the system which imply the global security property. More precisely, our unwinding condition ensures that no σ-high action α leading to a configuration C is observable by a σ-low user, as there always exists a configuration C', σ-equivalent to C, that the system may reach without performing α.

Definition 5 (σ-Unwinding Condition). *Let $\sigma \in \Sigma$, P be a process and Γ be a type environment such that $\Gamma \vdash P$. The process P satisfies the σ-unwinding condition in Γ, written $\Gamma \triangleright P \in \mathcal{W}(\cong_\sigma)$, if for all $\Gamma' \triangleright P_1$ such that $\Gamma \triangleright P \rightsquigarrow \Gamma' \triangleright P_1$*

- *if $\Gamma' \triangleright P_1 \xrightarrow{\alpha}{}^\sigma \Gamma' \triangleright P_2$ with $\alpha \in \{\overline{n}\langle m\rangle,\ n(m)\}$, then $\exists P_3$ such that $\Gamma' \triangleright P_1 \Longrightarrow \Gamma' \triangleright P_3$ and $\Gamma' \vDash P_2 \cong_\sigma P_3$;*
- *if $\Gamma' \triangleright P_1 \xrightarrow{\alpha}{}^\sigma \Gamma', m{:}T \triangleright P_2$ with $\alpha \in \{(\boldsymbol{\nu} m{:}T)\,\overline{n}\langle m\rangle,\ (\boldsymbol{\nu} m{:}T)\,n(m)\}$, then $\exists P_3$ such that $\Gamma' \triangleright P_1 \Longrightarrow \Gamma' \triangleright P_3$ and $\Gamma' \vDash P_3 \cong_\sigma (\boldsymbol{\nu} m{:}T)P_2$.*

This unwinding-based schema characterizes a notion of security with respect to all *passive attacks* which try to infer information about the classified behavior just by observing the σ-level behaviour of the system.

Both properties $\mathcal{NI}(\cong_\sigma)$ and $\mathcal{W}(\cong_\sigma)$ are persistent, as stated in the following proposition.

Proposition 3 (Persistence). *Let $\sigma \in \Sigma$, P be a process and Γ be a type environment such that $\Gamma \vdash P$. For all $\Gamma' \triangleright P'$ such that $\Gamma \triangleright P \rightsquigarrow \Gamma' \triangleright P'$ it holds*

- *if $\Gamma \triangleright P \in \mathcal{NI}(\cong_\sigma)$ then $\Gamma' \triangleright P' \in \mathcal{NI}(\cong_\sigma)$.*
- *if $\Gamma \triangleright P \in \mathcal{W}(\cong_\sigma)$ then $\Gamma' \triangleright P' \in \mathcal{W}(\cong_\sigma)$.*

The equivalence of properties $\mathcal{NI}(\cong_\sigma)$ and $\mathcal{W}(\cong_\sigma)$ is stated below.

Theorem 2. *Let $\sigma \in \Sigma$, P be a process and Γ be a type environment such that $\Gamma \vdash P$. $\Gamma \triangleright P \in \mathcal{NI}(\cong_\sigma)$ if and only if $\Gamma \triangleright P \in \mathcal{W}(\cong_\sigma)$.*

The unwinding-based characterization of σ-noninterfering processes provides a better understanding of the operational semantics of secure processes. Moreover, it allows one to define efficient proof techniques for σ-noninterference just by inspecting the typed LTS of processes. Notice that the σ-unwinding condition $\mathcal{W}(\cong_\sigma)$ is decidable over the class of finite state processes, i.e., processes whose typed LTS is finite. Moreover, by exploiting the following compositionality results, the unwinding condition $\mathcal{W}(\cong_\sigma)$ can be used to define methods, e.g., proof systems, both to check the security of complex systems and to incrementally build processes which are secure by construction.

Theorem 3 (Compositionality of $\mathcal{W}(\cong_\sigma)$). *Let $\sigma \in \Sigma$, P and Q be processes and Γ be a type environment such that $\Gamma \vdash P, Q$. If $\Gamma \triangleright P \in \mathcal{W}(\cong_\sigma)$ and $\Gamma \triangleright Q \in \mathcal{W}(\cong_\sigma)$ then*

- *$\Gamma, \Gamma' \triangleright \overline{a}\langle b\rangle.P \in \mathcal{W}(\cong_\sigma)$ where $\Gamma, \Gamma' \vdash a : \delta[T]$, $\Gamma, \Gamma' \vdash b : T$ and $\delta \preceq \sigma$;*
- *$\Gamma, \Gamma' \triangleright a(x:T).P \in \mathcal{W}(\cong_\sigma)$ where $\Gamma, \Gamma' \vdash a : \delta[T]$ and $\delta \preceq \sigma$;*
- *$\Gamma, \Gamma' \triangleright \mathsf{if}\ a = b\ \mathsf{then}\ P\ \mathsf{else}\ Q \in \mathcal{W}(\cong_\sigma)$ where $\Gamma, \Gamma' \vdash a : T$ and $\Gamma, \Gamma' \vdash b : T$;*
- *$\Gamma \triangleright P \mid Q \in \mathcal{W}(\cong_\sigma)$;*
- *$\Gamma' \triangleright (\boldsymbol{\nu} n : T)P \in \mathcal{W}(\cong_\sigma)$ where $\Gamma = \Gamma', n : T$;*
- *$\Gamma \triangleright !P \in \mathcal{W}(\cong_\sigma)$.*

Example 3. Let P and Q be finite state processes and Γ be a type environment such that $\Gamma \vdash P, Q$. Although $R = !P \mid Q$ might be an infinite state process, one can easily check whether $\Gamma \triangleright R \in \mathcal{NI}(\cong_\sigma)$ just by exploiting the decidability of $\Gamma \triangleright P \in \mathcal{W}(\cong_\sigma)$ and $\Gamma \triangleright Q \in \mathcal{W}(\cong_\sigma)$ and the compositionality of $\mathcal{NI}(\cong_\sigma)$ with respect to the parallel composition and replication operators.

4.1 Noninterference through a Partial Equivalence Relation

In [20, 19] the notion of noninterference for sequential and multithreaded programs is expressed in terms of a partial equivalence relation (*per* model) which captures the view of a σ-level observer. Intuitively, a configuration C, representing a program and the current state of the memory, is secure if $C \sim_\sigma C$ where $\sim_\sigma$ is a symmetric and transitive relation modeling the σ-level observation of program executions. The relation $\sim_\sigma$ is in general not reflexive, but it becomes reflexive over the set of secure configurations.

Below we show how this approach can be adapted to the π-calculus to characterize the class of σ-noninterfering processes. We first introduce the following notion of partial bisimilarity up to σ-high actions, $\dot{\approx}_\sigma$. Intuitively, $\dot{\approx}_\sigma$ requires that σ-high actions are simulated by internal transitions, while on the remaining actions it behaves as $\approx_\sigma$.

Definition 6 (Partial Bisimilarity up to σ-high actions $\dot{\approx}_\sigma$). *Let $\sigma \in \Sigma$. Partial bisimilarity up to σ-high actions is the largest symmetric relation $\dot{\approx}_\sigma$ over configurations, such that whenever $\Gamma \vDash P \,\dot{\approx}_\sigma\, Q$*

- *if $\Gamma \rhd P \xrightarrow{\alpha}_\sigma \Gamma' \rhd P'$, then there exists Q' such that $\Gamma \rhd Q \stackrel{\hat{\alpha}}{\Longrightarrow}_\sigma \Gamma' \rhd Q'$ with $\Gamma' \vDash Q' \,\dot{\approx}_\sigma\, P'$.*
- *if $\Gamma \rhd P \xrightarrow{\alpha}{}^\sigma \Gamma \rhd P'$ with $\alpha \in \{\overline{n}\langle m\rangle, n(m)\}$, then there exists Q' such that $\Gamma \rhd Q \Longrightarrow \Gamma \rhd Q'$ with $\Gamma \vDash Q' \,\dot{\approx}_\sigma\, P'$.*
- *if $\Gamma \rhd P \xrightarrow{\alpha}{}^\sigma \Gamma, m : T \rhd P'$ with $\alpha \in \{(\boldsymbol{\nu} m{:}T)\,\overline{n}\langle m\rangle, (\boldsymbol{\nu} m{:}T)\,n(m)\}$, then there exists Q' such that $\Gamma \rhd Q \Longrightarrow \Gamma \rhd Q'$ with $\Gamma \vDash Q' \,\dot{\approx}_\sigma\, (\boldsymbol{\nu} m : T)P'$ and $\Gamma, m : T \vDash P' \,\dot{\approx}_\sigma\, P'$.*

The relation $\dot{\approx}_\sigma$ is a partial equivalence relation, i.e., it is not reflexive. In fact, if we consider the process $P = \overline{h}\langle\rangle.\overline{\ell}\langle\rangle.\mathbf{0}$ and the type environment $\Gamma = h : \top[\,], \ell : \bot[\,]$ we get $\Gamma \vDash P \,\dot{\not\approx}_\sigma\, P$ when $\sigma = \bot$.

The next theorem states that relation $\dot{\approx}_\sigma$ is reflexive on the set of well typed non-interfering processes. The proof exploits a sort of persistence property of $\dot{\approx}_\sigma$, that is: if $\Gamma \vDash P \,\dot{\approx}_\sigma\, P$, then for all $\Gamma' \rhd P'$ such that $\Gamma \rhd P \rightsquigarrow \Gamma' \rhd P'$, it holds $\Gamma' \vDash P' \,\dot{\approx}_\sigma\, P'$.

Theorem 4. *Let $\sigma \in \Sigma$, P be a process and Γ be a type environment such that $\Gamma \vdash P$. $\Gamma \rhd P \in \mathcal{NI}(\cong_\sigma)$ if and only if $\Gamma \vDash P \,\dot{\approx}_\sigma\, P$.*

4.2 Noninterference through Name Restriction

In [8] the *P_BNDC* property for CCS processes is characterized in terms of a single bisimulation-like equivalence check. We show that the same idea can be applied to the π-calculus. Let us first introduce the following definition.

Definition 7. *Let $\sigma \in \Sigma$, P be a process and Γ be a type environment such that $\Gamma \vdash P$. We denote by $(\boldsymbol{\nu}^\sigma)P$ the process $(\boldsymbol{\nu} m_1{:}T_1)\ldots(\boldsymbol{\nu} m_k{:}T_k)P$ where $m_1, \ldots m_k$ are all the free names occurring in P such that $\Gamma(m_i) = T_i$ and $\Lambda(T_i) \succ \sigma$.*

Definition 6 of partial bisimilarity up to σ-high actions can be modified as follows in order to obtain an equivalence relation.

Definition 8 (Bisimilarity up to σ-high actions $\ddot{\approx}_\sigma$). *Let $\sigma \in \Sigma$. Bisimilarity up to σ-high actions is the largest symmetric relation $\ddot{\approx}_\sigma$ over configurations, such that whenever $\Gamma \vDash P \,\ddot{\approx}_\sigma\, Q$*

- *if $\Gamma \rhd P \xrightarrow{\alpha}_\sigma \Gamma' \rhd P'$, then there exists Q' such that $\Gamma \rhd Q \stackrel{\hat{\alpha}}{\Longrightarrow}_\sigma \Gamma' \rhd Q'$ with $\Gamma' \vDash Q' \,\ddot{\approx}_\sigma\, P'$.*

– *if* $\Gamma \triangleright P \xrightarrow{\alpha}{}^{\sigma} \Gamma \triangleright P'$ *with* $\alpha \in \{\overline{n}\langle m\rangle, n(m)\}$, *then there exists* Q' *such that either* $\Gamma \triangleright Q \stackrel{\hat{\alpha}}{\Longrightarrow}{}^{\sigma} \Gamma \triangleright Q'$ *with* $\Gamma \vDash Q' \ddot{\approx}_{\sigma} P'$ *or* $\Gamma \triangleright Q \Longrightarrow \Gamma \triangleright Q'$ *with* $\Gamma \vDash Q' \ddot{\approx}_{\sigma} P'$.

– *if* $\Gamma \triangleright P \xrightarrow{\alpha}{}^{\sigma} \Gamma, m : T \triangleright P'$ *with* $\alpha \in \{(\boldsymbol{\nu} m{:}T)\,\overline{n}\langle m\rangle, (\boldsymbol{\nu} m{:}T)\,n(m)\}$, *then there exists* Q' *such that either* $\Gamma \triangleright Q \stackrel{\hat{\alpha}}{\Longrightarrow}{}^{\sigma} \Gamma, m : T \triangleright Q'$ *with* $\Gamma, m : T \vDash Q' \ddot{\approx}_{\sigma} P'$ *or* $\Gamma \triangleright Q \Longrightarrow \Gamma \triangleright Q'$ *with* $\Gamma \vDash Q' \ddot{\approx}_{\sigma} (\boldsymbol{\nu} m : T)P'$ *and* $\Gamma, m : T \vDash P' \ddot{\approx}_{\sigma} (\boldsymbol{\nu}^{\sigma})P'$.

We can now characterize $\mathcal{NI}(\cong_{\sigma})$ in terms of a single equivalence check between P and $(\boldsymbol{\nu}^{\sigma})P$ through $\ddot{\approx}_{\sigma}$. The proof of the next theorem exploits the fact that if $\Gamma \vDash P \ddot{\approx}_{\sigma} (\boldsymbol{\nu}^{\sigma})P$, then for all $\Gamma' \triangleright P'$ such that $\Gamma \triangleright P \rightsquigarrow \Gamma' \triangleright P'$, it holds $\Gamma' \vDash P' \ddot{\approx}_{\sigma} (\boldsymbol{\nu}^{\sigma})P'$.

Theorem 5. *Let* $\sigma \in \Sigma$, P *be a program and* Γ *be a type environment such that* $\Gamma \vdash P$. $\Gamma \triangleright P \in \mathcal{NI}(\cong_{\sigma})$ *if and only if* $\Gamma \vDash P \ddot{\approx}_{\sigma} (\boldsymbol{\nu}^{\sigma})P$.

Corollary 1. *Let* $\sigma \in \Sigma$, P *be a process and* Γ *be a type environment such that* $\Gamma \vdash P$ *and* $\forall n \in \mathrm{fn}(P)$, $\Lambda(\Gamma(n)) \preceq \sigma$ *(i.e.,* P *has no free* σ*-high level names). Then* $\Gamma \triangleright P \in \mathcal{NI}(\cong_{\sigma})$.

Example 4. Let us consider the processes $P_1 = h().\ell() \mid \overline{h}\langle\rangle$ and $P_3 = h(x{:}T).\mathsf{if}\ x = n\ \mathsf{then}\ \overline{\ell_1}\langle\rangle\ \mathsf{else}\ \overline{\ell_2}\langle\rangle \mid \overline{h}\langle n\rangle \mid \overline{h}\langle m\rangle$ and the type environments Γ and Γ' of Example 2. We have seen that $\Gamma \triangleright P_1 \notin \mathcal{NI}(\cong_{\sigma})$ and $\Gamma' \triangleright P_3 \notin \mathcal{NI}(\cong_{\sigma})$. Now, by Corollary 1, we can immediately state that both $\Gamma \triangleright (\boldsymbol{\nu} h)P_1 \in \mathcal{NI}(\cong_{\sigma})$ and $\Gamma' \triangleright (\boldsymbol{\nu} h)P_3 \in \mathcal{NI}(\cong_{\sigma})$.

Notice that a process whose free names have a security level higher than σ is, in general, not secure. For instance, let Γ be the type environment $h : \top[\bot[\,]],\ \ell : \bot[\,]$ and P be the process $h(x{:}\bot[\,]).\overline{x}\langle\rangle$. Assuming that $\sigma \prec \top$, we have that the only free name h occurring in P has a security level higher than σ. It is easy to see that $\Gamma \triangleright P \notin \mathcal{NI}(\cong_{\sigma})$: in fact, by choosing $H = \overline{h}\langle\ell\rangle$, we have $\Gamma \vDash P \not\approx_{\sigma} P \mid H$, that is P is insecure.

We conclude this section observing that, as in [7] for CCS, the definition of σ-noninterference can be also expressed in terms of bisimilarity on $\top$-actions over well-typed processes whose σ-high level names are restricted. This comes as a corollary of the following property.

Proposition 4. *Let* $\sigma \in \Sigma$, P *and* Q *be two processes and* Γ *be a type environment such that* $\Gamma \vdash P, Q$. $\Gamma \vDash P \approx_{\sigma} Q$ *if and only if* $\Gamma \vDash (\boldsymbol{\nu}^{\sigma})P \approx_{\top} (\boldsymbol{\nu}^{\sigma})Q$.

Corollary 2. *Let* $\sigma \in \Sigma$, P *be a process and* Γ *be a type environment such that* $\Gamma \vdash P$. $\Gamma \triangleright P \in \mathcal{NI}(\cong_{\sigma})$ *if and only if for all* $\Gamma' \triangleright P'$ *such that* $\Gamma \triangleright P \rightsquigarrow \Gamma' \triangleright P'$ *and for all* $H \in \mathcal{H}^{\sigma}_{\Gamma'}$, *it holds* $\Gamma' \vDash (\boldsymbol{\nu}^{\sigma})P' \approx_{\top} (\boldsymbol{\nu}^{\sigma})(P' \mid H)$.

5 Conclusion and Related Work

In this paper we develop a theory of noninterference for processes of the typed π-calculus. In the literature there are a number of works which study type-based techniques for noninterference. A few of them are discussed in the following.

Hennessy and Riely [13, 11] consider a typed version of the asynchronous π-calculus where types associate read/write capabilities to channels as well as security clearances. They study noninterference properties based on may and must equivalences. A similar study is conducted by Pottier [17] relying on the synchronous π-calculus and bisimulation equivalence. Honda, Yoshida, Vasconcelos and Berger [14, 15, 24] consider advanced type systems for the linear/affine π-calculus and express noninterference in terms of typed bisimulation equivalences. Their type systems guarantee that every communication on a linear channel must eventually succeed, and so its success alone does not carry any information. For instance, the process $h().\overline{\ell}\langle\rangle$, which waits an input on the secret channel h and then performs the low-level output $\overline{\ell}\langle\rangle$, is considered secure as long as h is a linear channel. Similarly, Zdancewic and Myers [25] propose a type system dealing with linear channels in a concurrent language with (a restricted form of) join-patterns as synchronization primitives. Furthermore, their type system controls the temporal ordering of communications on linear channels. Kobayashi [16] presents an even more flexible type system which can deal with arbitrary usage of channels, so that programs using various concurrency primitives (including locks) can be encoded into the π-calculus and analyzed.

The typing constraints imposed by the type systems discussed above allow one to reason only on a limited class of processes and contexts. For instance the process $!x(y).P|!x(y).Q$ is rejected by the type system of, e.g., [15] and thus it is considered insecure independently of the security level of its channels. As another example, when h is a nonlinear channel, the process $(\boldsymbol{\nu} h)(h().\ell() \mid \overline{h}\langle\rangle)$ is never typed in most of the mentioned type systems. However, this process does not leak any secret information, as shown in Example 4.

Our approach relies on a much simpler typing discipline which does not deal with implicit information flow. Instead, we characterize secure processes in terms of the actions they perform. The use of a lighter type system leads to stronger noninterference properties, that check the security of processes against a bigger class of attackers. Compared with the literature discussed so far, such properties could be considered too restrictive. Nevertheless, they are more suitable in contexts with partial trust, where it would be not realistic to assume that attackers are well typed in a strong way. Interestingly, we can increase the flexibility of our approach by admitting mechanisms for *downgrading* or *declassifying* information as done in [3] for CCS. This would allow the process $h().\overline{\ell}\langle\rangle$ to be deemed secure by declassifying the high action $h()$.

Another difference with respect to previous works is that they deal with open terms, while our theory applies to closed processes. However, the results presented in this paper scale to open terms by: (1) introducing the open extension of $\cong_\sigma$ as the type-indexed relation $\cong^o_\sigma$ over terms such that $\Gamma \vDash T \ \cong^o_\sigma \ U$ if and only if $\Gamma' \vDash T\rho \ \cong_\sigma \ U\rho$ for all closing substitution ρ which respects[3] Γ with Γ', and (2) saying that a term T satisfies the *σ-noninterference property in Γ* if for all closing substitution ρ which respects Γ with Γ', it holds $\Gamma' \triangleright T\rho \in \mathcal{NI}(\cong_\sigma)$.

[3] We say that $\rho = \{x_1 := m_1, \ldots, x_n := m_n\}$ *is a substitution which respects* Γ *with* Γ' if $\Gamma = \Delta, x_1{:}T_1, \ldots, x_n{:}T_n$ and there exists Δ' such that $\Gamma' = \Delta, \Delta'$ and $\Gamma' \vdash m_i : T_i$ for $i = 1, \ldots, n$.

References

1. M. Boreale and D. Sangiorgi. Bisimulation in Name-Passing Calculi without Matching. In *Proc. of 13th IEEE Symposium on Logic in Computer Science (LICS'98)*, pages 165–175. IEEE Computer Society Press, 1998.
2. A. Bossi, R. Focardi, C. Piazza, and S. Rossi. Verifying Persistent Security Properties. *Computer Languages, Systems and Structures*, 30(3-4):231–258, 2004.
3. A. Bossi, C. Piazza, and S. Rossi. Modelling Downgrading in Information Flow Security. In *Proc. of the 17th IEEE Computer Security Foundations Workshop (CSFW'04)*, pages 187–201. IEEE Computer Society Press, 2004.
4. S. Crafa, M. Bugliesi, and G. Castagna. Information Flow Security for Boxed Ambients. *ENTCS*, 66(3), 2002.
5. S. Crafa and S. Rossi. A Theory of Noninterference for the π-calculus. Technical Report CS-2004-8, Dipartimento di Informatica, Università Ca' Foscari di Venezia, Italy, 2004. `http://www.dsi.unive.it/~silvia/CS-2004-8.ps.gz`.
6. D.E. Denning and P.J. Denning. Certification of programs for secure information flow. *Communications of the ACM*, 20:504–513, 1977.
7. R. Focardi and R. Gorrieri. Classification of Security Properties (Part I: Information Flow). In R. Focardi and R. Gorrieri, editors, *Proc. of Foundations of Security Analysis and Design (FOSAD'01)*, volume 2171 of *LNCS*, pages 331–396. Springer-Verlag, 2001.
8. R. Focardi and S. Rossi. Information Flow Security in Dynamic Contexts. In *Proc. of the IEEE Computer Security Foundations Workshop (CSFW'02)*, pages 307–319. IEEE Computer Society Press, 2002.
9. J. A. Goguen and J. Meseguer. Security Policies and Security Models. In *Proc. of the IEEE Symposium on Security and Privacy (SSP'82)*, pages 11–20. IEEE Computer Society Press, 1982.
10. N. Heintze and J. G. Riecke. The SLam Calculus: Programming with Secrecy and Integrity. In *Proc. of ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'98)*, pages 365–377. ACM Press, 1998.
11. M. Hennessy. The security picalculus and non-interference. *Journal of Logic and Algebraic Programming*, 2004. To Appear.
12. M. Hennessy and J. Rathke. Typed Behavioural Equivalences for Processes in the Presence of Subtyping. *Mathematical Structures in Computer Science*, 14(5):651–684, 2004.
13. M. Hennessy and J. Riely. Information Flow vs. Resource Access in the Asynchronous Pi-calculus. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 24(5):566–591, 2002.
14. K. Honda, V.T. Vasconcelos, and N. Yoshida. Secure Information Flow as Typed Process Behaviour. In *Proc. of European Symposium on Programming (ESOP'00)*, volume 1782 of *LNCS*, pages 180–199. Springer-Verlag, 2000.
15. K. Honda and N. Yoshida. A Uniform Type Structure for Secure Information Flow. In *Proc. of ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'02)*, pages 81–92. ACM Press, 2002.
16. N. Kobayashi. Type-Based Information Flow Analysis for the Pi-Calculus. Technical Report TR03-0007, Dept. of Computer Science, Tokyo Institute of Technology, 2003.
17. F. Pottier. A simple view of type-secure information flow in the π-calculus. In *Proc. of the 15th IEEE Computer Security Foundations Work shop*, pages 320–330, 2002.
18. F. Pottier and V. Simonet. Information Flow Inference for ML. In *Proc. of ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'02)*, pages 319–330. ACM Press, 2002.

19. A. Sabelfeld and H. Mantel. Static Confidentiality Enforcement for Distributed Programs. In *Proc. of Int. Static Analysis Symposium (SAS'02)*, volume 2477 of *LNCS*, pages 376–394. Springer-Verlag, 2002.
20. A. Sabelfeld and D. Sands. Probabilistic Noninterference for Multi-threaded Programs. In *Proc. of the IEEE Computer Security Foundations Workshop (CSFW'00)*, pages 200–215. IEEE Computer Society Press, 2000.
21. D. Sangiorgi and D. Walker. *The pi calculus: A theory of mobile processes*. Cambridge, 2001.
22. G. Smith and D. Volpano. Secure Information Flow in a Multi-threaded Imperative Language. In *Proc. of ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'98)*, pages 355–364. ACM Press, 1998.
23. D. Volpano, G. Smith, and C. Irvine. A Sound Type System for Secure Flow Analysis. *Journal of Computer Security*, 4(3):167–187, 1996.
24. N. Yoshida, K. Honda, and M. Berger. Linearity and Bisimulation. In *Proc. of the International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'02)*, volume 2303 of *LNCS*, pages 417–434. Springer-Verlag, 2002.
25. S. Zdancewic and A. C. Myers. Observational Determinism for Concurrent Program Security. In *Proceedings of the 16th IEEE Computer Security Foundations Work shop*, pages 29–45, 2003.