P-Congruences as Non-Interference for the Pi-calculus*

Silvia Crafa
Dipartimento di Matematica Pura ed Applicata
Università di Padova
via Trieste, 63
35131 Padova, Italy
crafa@math.unipd.it

Sabina Rossi
Dipartimento di Informatica
Università Ca' Foscari di Venezia
via Torino, 155
30172 Venezia, Italy
srossi@dsi.unive.it

ABSTRACT

We introduce a notion of noninterference for a typed version of the π-calculus where types are used to assign secrecy levels to channels. Noninterference is expressed in terms of a *partial congruence* (*p-congruence*, for short). We provide a proof technique in the form of a bisimulation-like partial equivalence relation that is a binary relation which is symmetric and transitive but not reflexive.

We show that the noninterference property is compositional with respect to most of the operators of the language leading to efficient proof techniques for the verification and the construction of (compositional) secure systems.

In order to allow downgrading of sensitive information, we extend the π-calculus with declassification primitives and we study a property which scales to noninterference when downgrading is not permitted.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs—*specification techniques, mechanical verification*; F.3.2 [**Logics and Meanings of Programs**]: Semantics of Programming Languages—*Algebraic approaches to semantics, Process models*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Security, Theory, Verification

*Work partially supported by the MIUR project "Fondamenti Logici dei Sistemi Distribuiti e Codice Mobile" (grant 2005015785) and the FIRB project "Interpretazione Astratta e Model Checking per la Verifica di Sistemi Embedded" (grant RBAU018RCZ).

FMSE'06, November 3, 2006, Alexandria, Virginia, USA.

Keywords

Process Calculi, Noninterference, Downgrading

1. INTRODUCTION

A number of formal definitions of security properties for different languages and models has been proposed and studied in the literature. One of the most successful approaches is information flow security which requires that there is no flow from private (secret) to public data. The concept of *noninterference* [6] has been introduced to formalize the absence of information flow in multilevel systems. It demands that public outputs are unchanged as secret inputs are varied or, more generally, that the low level observation of the system is independent from the behaviour of its high components. Useful surveys of research into language-based information flow security and noninterference in process languages can be found in [4, 16, 17].

Syntax-directed typechecking techniques are usually used to guarantee forms of noninterference for the π-calculus [8, 11, 13, 14, 15]. In these works type systems play a central role in the definition of noninterference, since both the observation of the system and the observed processes are constrained by types. A soundness theorem guarantees that well-typed processes are interference-free.

In our previous work [3] we propose a different approach to noninterference for the π-calculus, where the use of types is much lighter. We use a slight extension of the basic type system for the π-calculus [21] where security levels are associated with channels. As for security, the only typing constraint 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, but it does not deal with implicit flows. Instead, we characterize noninterference in terms of the actions that typed processes may perform.

The definition of noninterference presented in [3] is inspired by the *P_BNDC* (*Persistent BNDC*) property defined by Focardi and the second author in [5] for CCS. Roughly, a process P is interference-free if for every state P' reachable from P, and for every high level process H (that is a process which can only perform high level actions) the processes P' and $P' \mid H$ are indistinguishable for a low level observer. This definition involves a notion of reachability for typed processes which allows us to reason on all the possible states in which a process may evolve. The security definition is *persistent* in the sense that if a process satisfies noninterference then also all its reachable states do. As discussed in [5], persistence is technically useful since it allows one to apply inductive reasoning when proving security results (e.g., compositionality). Furthermore, in [3] persistence is exploited to give various quantifier-free characterizations of noninterference based on bisimulation-like relations leading to efficient methods for the verification and construction of (compositional) secure systems.

In this paper we follow a different approach: instead of requiring persistence we consider a more sophisticated notion of contextuality and show that this leads to more precise and efficient definitions. In order to give an intuition, consider the program $P = (\boldsymbol{\nu}\ell)(\overline{h}\langle\ell\rangle.\overline{\ell}\langle\rangle.P')$ with ℓ standing for a public channel and h for a private one. According to the operational semantics of the π-calculus, the process P may extrude the scope of the new name ℓ, evolving into a state where ℓ is a free name, and reaching the state P', i.e., $P \xrightarrow{(\nu l)\overline{h}\langle l\rangle} \overline{l}\langle\rangle.P' \xrightarrow{\overline{l}\langle\rangle} P'$. Thus, according to the definition of noninterference presented in [3], in order to prove that P is secure, we need to check that also P' satisfies the security definition. However, the name ℓ will never be extruded to a low level observer since it is communicated along a high level channel. Hence the low observer will never be able to observe the state P'. The process P is indeed secure independently from P', which should not be required to be secure. On the other hand, if $Q = (\boldsymbol{\nu}\ell_2)(\overline{\ell_1}\langle\ell_2\rangle.\overline{\ell_2}\langle\rangle.Q')$ where both ℓ_1 and ℓ_2 denote a public channel, then Q' is clearly observable from a low level point of view and thus it is correct to require that it satisfies the global security definition in order to ensure that Q is interference-free.

In this paper we study a more precise, contextual, definition of noninterference for the π-calculus expressed in terms of so-called *partial congruences* (p-congruences, for short) capturing the low level behaviour of processes whatever are their surrounding high level contexts, i.e., whatever are their high level behaviours.

One of the most natural observation equivalences for the π-calculus is reduction barbed congruence $\cong$ ([12, 9]), which is a congruence defined in terms of reduction and observability. More precisely, two processes P and Q exhibit the same behavior if they are are equivalent with respect to a relation $\mathcal{R}$ which is

(1) *reduction closed*, i.e., if $P \xrightarrow{\tau} P'$ then $Q \Longrightarrow Q'$ and $P'\mathcal{R}Q'$

(2) *barb preserving*, i.e., if $P \xrightarrow{\overline{n}\langle m\rangle}$ then there exists Q' such that $Q \Longrightarrow Q' \xrightarrow{\overline{n}\langle m\rangle}$

(3) *contextual*, i.e., $C[P]\ \mathcal{R}\ C[Q]$ for all contexts $C[\ ]$.

In multilevel systems, where processes and resources are associated with security clearances taken from a complete lattice $\langle\Sigma,\preceq\rangle$ of security annotations, the observation can be parameterized with respect to the security level σ of the observer. In [3] we define a σ-reduction barbed congruence capturing the σ-low behavior of processes. Formally, it is a relation $\mathcal{R}_\sigma$ which is

(1) *reduction closed*

(2) *σ-barb preserving*, i.e. if $P \xrightarrow{\overline{n}\langle m\rangle}$ where the security level of n is less or equal than σ then there exists Q' such that $Q \Longrightarrow Q' \xrightarrow{\overline{n}\langle m\rangle}$

(3) *contextual with respect to σ-low level contexts*, i.e., it holds $C_L[P]\ \mathcal{R}_\sigma\ C_L[Q]$ for all context $C_L[\ ]$ where $C_L[\ ]$ may interact with the process filling the hole just through channels of level at most σ.

In [3] we show how the *P_BNDC* property can be defined in the π-calculus using the abovementioned observation equivalence.

In this paper we generalize the σ-reduction barbed congruence in order to equate processes exhibiting the same σ-level behaviour whatever is the surrounding σ-high level context. Such a relation leads to a natural definition of noninterfering processes, that is processes whose observable behavior is independent from the surrounding high level context.

The new relation, denoted with $\cong_\sigma$, simply changes the notion of contextuality as follows:

$$P \cong_\sigma Q \text{ implies } C_L[C_H^1[P]] \cong_\sigma C_L[C_H^2[Q]].$$

for all σ-high contexts $C_H^1[\ ]$ and $C_H^2[\ ]$ which may interact with the process filling the hole just through channels of level not less than σ, and for all σ-low contexts $C_L[\ ]$.

The resulting relation is no more an equivalence. Instead, it is a partial equivalence relation, that is symmetric and transitive but not reflexive. We call it *σ-reduction barbed p-congruence*, p-congruence for short; moreover, $\cong_\sigma$ is indexed on a typing environment Γ that associates security levels to channel names.

It is then natural to define noninterference as the reflexive closure of $\cong_\sigma$. Thus we say that a process P in a type environment Γ satisfies the *σ-noninterference property*, written $\Gamma \triangleright P \in \mathcal{NI}(\cong_\sigma)$, if

$$\Gamma \vDash P \cong_\sigma P.$$

In particular, by contextuality, if P is σ-noninterfering then

$$\Gamma' \vDash C_L[C_H^1[P]] \cong_\sigma C_L[C_H^2[P]]$$

for all σ-low contexts $C_L[\ ]$ and for all σ-high contexts $C_H^1[\ ]$ and $C_H^2[\ ]$.

Interestingly, a proof technique for $\cong_\sigma$ exists in the form of a partial equivalence relation (*per* model, see [18]) on typed labelled transition systems.

A typed LTS is built around typed actions of the form $\Gamma \triangleright P \xrightarrow{\alpha}_\delta \Gamma' \triangleright P'$ 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 define a partial equivalence relation $\dot{\approx}_\sigma$ in the spirit of the definitions in [18, 19] for imperative and multi-threaded languages, and we prove that $\dot{\approx}_\sigma$ provides a coinductive characterization of the p-congruence $\cong_\sigma$. We show that the partial equivalence relation $\dot{\approx}_\sigma$ is an efficient proof technique for the security property $\mathcal{NI}(\cong_\sigma)$, which is decidable over the set of finite state processes. 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.

It is well-known that strong noninterference can hardly be achieved in real systems. Indeed, real-world applications often release bits of information as part of their intended behaviour. For instance, when two high level users communicate through an encrypted channel, as in the case of a purchase protocol, secret information is revealed whenever a condition, such as "payment transferred", has been fulfilled. In order to permit systems to leak information by design, information flow controls often include some notion of downgrading, which allows trusted entities to declassify information from a higher to a lower security level.

A number of definitions and analysis for different kinds of information release policies over a variety of languages and calculi have been recently proposed. The reader is referred to the work of Sabelfeld *et al.* [20] for a road map of the main directions of the research on this topic.

In this paper we extend the noninterference framework for the π-calculus illustrated above by integrating a mechanism for the secure downgrading of information into the π-calculus. More precisely, following [2], we enrich the standard π-calculus with a family of *declassified actions* of the form $\mathsf{dec}_\delta\, a(x{:}T)$ and $\overline{\mathsf{dec}_\delta\, a}\langle b\rangle$ with δ belonging to the complete lattice of security annotations. We call $\mathsf{Dec}\,\pi$-calculus the new language. If a is a channel of a security level δ' with $\delta \preceq \delta'$ then $\mathsf{dec}_\delta\, a(x{:}T)$ is an action declassified to the lower level δ which can be used by the programmer to specify an "escape hatch" for information release, i.e., to allow information arising from this action to flow downwards up to level δ. The same holds for the write action $\overline{\mathsf{dec}_\delta\, a}\langle b\rangle$. In this approach the dec constructor is used to declassify actions instead of names, obtaining a flexible, finer-grained, downgrading mechanism that, for instance, allows programmers to interleave declassified and non declassified actions over the same channel.

We show that the noninterference property $\mathcal{NI}(\cong_\sigma)$ scales to the $\mathsf{Dec}\,\pi$-calculus, which also inherit the efficient proof technique based on the relation $\dot{\approx}_\sigma$.

The rest of the paper is organized as follows. In Section 2 we present the language, its semantics and the type system. P-congruences together with the corresponding proof technique are introduced in Section 3. In Section 4 we define σ-noninterference and show its compositionality properties. In Section 5 we extend the noninterference framework in order to deal with a declassification mechanism. Section 6 concludes the paper.

2. SYNTAX AND SEMANTICS OF TYPED PI-CALCULUS

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

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 synchronous, polyadic, calculus with the match/mismatch operator. The choice of the synchronous model is motivated by the fact that it gives rise to more interferences with respect to an asynchronous one. Nevertheless, our results can be adapted to the asynchronous case[1]. On the other hand, as explained in [9], the matching construct is essential for the coinductive characterization of the partial congruence shown in Section 3.

As usual, the input construct $a(x_1{:}T_1, \ldots, x_k{:}T_k).P$ acts as a binder for the variables $x_1, \ldots, x_k$ 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 $a(\tilde{x}{:}\tilde{T}), \overline{a}\langle\tilde{b}\rangle, (\boldsymbol{\nu}\tilde{p}{:}\tilde{T}), \tilde{T}$ as a shorthand for $a(x_1{:}T_1, ..., x_k{:}T_k).\mathbf{0}, \overline{a}\langle b_1, ..., b_k\rangle.\mathbf{0}, (\boldsymbol{\nu} p_1{:}T_1)...(\boldsymbol{\nu} p_k{:}T_k)$ and $T_1, \ldots, T_k$. In this paper we restrict to *closed* processes, that are processes containing no free occurrences of variables; for a discussion on how to extend our theory to open terms the reader is referred to [2].

Types assign security levels to channels. More precisely, if $\sigma \in \Sigma$, then $\sigma[T_1, \ldots, T_k]$ is the type of channels of level σ which carry k values of type $T_1, \ldots, T_k$. We consider a function Λ associating to types the corresponding level, that is $\Lambda(\sigma[T_1, \ldots, T_k]) = \sigma$.

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:

$$\begin{array}{rcllll} \alpha & ::= & \tau & & \text{internal action} \\ & | & n(\tilde{m}) & & \text{receive a tuple} \\ & | & (\boldsymbol{\nu}\tilde{p}{:}\tilde{T})\,\overline{n}\langle\tilde{m}\rangle & \tilde{p} \subseteq \tilde{m} & \text{send a tuple} \\ & & & & \text{of (fresh) names} \end{array}$$

We write $\mathrm{fn}(\alpha)$ and $\mathrm{bn}(\alpha)$ to denote the set of free and bound names occurring in the action α, where $\mathrm{bn}(\alpha) = \{\tilde{p}\}$ if $\alpha = (\boldsymbol{\nu}\tilde{p}{:}\tilde{T})\,\overline{n}\langle\tilde{m}\rangle$, 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. We simply write $P \xrightarrow{\alpha}$ when the process resulting from P after the action α does not matter.

Type System. Our type system corresponds to the basic type system for the π-calculus (see [21]). The main judgments 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, judgments 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 rule of type formation, which prevents a channel of security level δ from carrying values of level higher than δ.

$$\text{(Channel Type)}\quad \frac{\vdash T_i \quad \Lambda(T_i) \preceq \delta \quad i = 1, \ldots, k}{\vdash \delta[T_1, \ldots, T_k]}$$

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.

It is easy to prove that our type system enjoys the standard subject reduction property (see [21]), expressing the consistency between the operational semantics and the typing rules.

[1]A discussion about the asynchronous π-calculus can be found in Section 6 where we compare our work with the security π-calculus of [8].

$$
\begin{array}{llll}
\textit{Prefixes} & \pi & ::= & \overline{a}\langle b_1,\ldots,b_k\rangle \;\mid\; a(x_1{:}T_1,\ldots,x_k{:}T_k) \quad \textit{where } k \geq 0 \\
\textit{Processes} & P & ::= & \pi.P \;\mid\; \mathsf{if}\ a = b\ \mathsf{then}\ P\ \mathsf{else}\ P \;\mid\; P \mid P \;\mid\; (\boldsymbol{\nu} n : T)P \;\mid\; !P \;\mid\; \mathbf{0} \\
\textit{Types} & T & ::= & \sigma[T_1,\ldots,T_k] \quad \textit{where } k \geq 0
\end{array}
$$

Table 1: Syntax

$$\textsc{(Par)}\quad \frac{P \xrightarrow{\alpha} P' \quad \mathrm{bn}(\alpha)\cap\mathrm{fn}(Q)=\emptyset}{P \mid Q \xrightarrow{\alpha} P' \mid Q}
\qquad
\textsc{(In)}\quad \frac{}{n(\tilde{x}{:}\tilde{T}).P \xrightarrow{n(\tilde{m})} P\{\tilde{x} := \tilde{m}\}}
\qquad
\textsc{(Out)}\quad \frac{}{\overline{n}\langle\tilde{m}\rangle.P \xrightarrow{\overline{n}\langle\tilde{m}\rangle} P}$$

$$\textsc{(Open)}\quad \frac{P \xrightarrow{(\boldsymbol{\nu}\tilde{p}:\tilde{T})\,\overline{n}\langle\tilde{m}\rangle} P' \quad q \neq n, \tilde{p} \quad q \in \tilde{m}}{(\boldsymbol{\nu} q{:}T)P \xrightarrow{(\boldsymbol{\nu} q:T)(\boldsymbol{\nu}\tilde{p}:\tilde{T})\,\overline{n}\langle\tilde{m}\rangle} P'}
\qquad
\textsc{(Rep-Act)}\quad \frac{P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P' \mid !P}
\qquad
\textsc{(Res)}\quad \frac{P \xrightarrow{\alpha} P' \quad n \notin \mathrm{fn}(\alpha)\cup\mathrm{bn}(\alpha)}{(\boldsymbol{\nu} n{:}T)P \xrightarrow{\alpha} (\boldsymbol{\nu} n{:}T)P'}$$

$$\textsc{(Close)}\quad \frac{P \xrightarrow{(\boldsymbol{\nu}\tilde{p}:\tilde{T})\,\overline{n}\langle\tilde{m}\rangle} P' \; Q \xrightarrow{n(\tilde{m})} Q' \; \tilde{p}\cap\mathrm{fn}(Q)=\emptyset}{P \mid Q \xrightarrow{\tau} (\boldsymbol{\nu}\tilde{p}{:}\tilde{T})(P' \mid Q')}
\qquad
\textsc{(Match)}\quad \frac{}{\mathsf{if}\ n = n\ \mathsf{then}\ P\ \mathsf{else}\ Q \xrightarrow{\tau} P}
\qquad
\textsc{(Mismatch)}\quad \frac{n \neq m}{\mathsf{if}\ n = m\ \mathsf{then}\ P\ \mathsf{else}\ Q \xrightarrow{\tau} Q}$$

Table 2: Labelled Transition System

3. OBSERVATION AS P-CONGRUENCES

In this section we define system observations in terms of p-congruences that are parametric with respect to the security level σ of external observers. P-congruences are type-indexed relations that equate the σ-low level, observable, behavior of processes interacting with whatever σ-high context.

We say that $\Gamma \triangleright P$ is a *configuration* if Γ is a type environment and P is a process such that $\Gamma \vdash P$. A type-indexed relation over processes is a family of binary relations between processes indexed by type environments. We write $\Gamma \vDash P \mathrel{\mathcal{R}} Q$ 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 relations, we will ask for the largest type-indexed relation over processes which satisfies the following properties.

Reduction Closure. A type-indexed relation $\mathcal{R}$ over processes is *reduction closed* if $\Gamma \vDash P \mathrel{\mathcal{R}} Q$ and $P \xrightarrow{\tau} P'$ imply that there exists Q' such that $Q \Longrightarrow Q'$ and $\Gamma \vDash P' \mathrel{\mathcal{R}} Q'$, 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 $\Gamma \vDash P \downarrow_n^\sigma$ if $P \xrightarrow{\overline{n}\langle m\rangle}$ with $\Lambda(\Gamma(n)) \preceq \sigma$. Furthermore we write $\Gamma \vDash P \Downarrow_n^\sigma$ if there exists some P' such that $P \Longrightarrow P'$ and $\Gamma \vDash P' \downarrow_n^\sigma$.

A type-indexed relation $\mathcal{R}$ over processes is *σ-barb preserving* if $\Gamma \vDash P \mathrel{\mathcal{R}} Q$ and $\Gamma \vDash P \downarrow_n^\sigma$ imply $\Gamma \vDash Q \Downarrow_n^\sigma$.

σ-Contextuality. Let a context be a process with at most one hole $[\cdot]$. If $C[\cdot]$ is a context and P is a process, then we write $C[P]$ for the process obtained by replacing the hole in $C[\cdot]$ by P.

A (Γ'/Γ)-context is a context $C[\cdot_\Gamma]$ such that, when filled with a process well typed in Γ, it becomes a process well typed in Γ'. More formally, if P is a process, Γ is a type environment such that $\Gamma \vdash P$ and $C[\cdot_\Gamma]$ is a (Γ'/Γ)-context, then $\Gamma' \vdash C[P]$.

In order to type contexts, the type system of Table 3 is extended with the following rule:

$$\textsc{(Ctx)}\quad \frac{}{\Gamma, \Gamma' \vdash [\cdot_\Gamma]}$$

We are interested in two classes of contexts, called σ-low and σ-high contexts. Intuitively, the σ-low contexts are used to observe the σ-low behaviour of processes, while the σ-high contexts are used to describe their possible σ-high interactions. More precisely, a σ-low (resp. σ-high) context is an evaluation context which may interact with the process filling the hole just through channels of level at most (resp. at least) σ. We first introduce the notions of σ-low and σ-high level sources.

Definition 1. (σ-low and σ-high level sources) Let P be a process and Γ be a type environment such that $\Gamma \vdash P$.

- We say that the process P is a *σ-low level source* in Γ, denoted $\Gamma \vdash_\sigma P$, if $\Gamma \vdash P$ and $\forall m \in \mathrm{fn}(P)$ it holds $\Lambda(\Gamma(m)) \preceq \sigma$.
- We say that the process P is a *σ-high level source* in Γ, denoted $\Gamma \vdash^\sigma P$, if for all names a used in P as a subject in an input or an output prefix, $\Lambda(\Gamma(a)) \not\preceq \sigma$. Notice that this definition does not prevent a σ-high level source from communicating σ-low values (along σ-high channels).

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

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

$$\frac{\Gamma \vdash a : T \quad \Gamma \vdash b : T \quad \Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash \mathsf{if}\ a = b\ \mathsf{then}\ P\ \mathsf{else}\ Q}\ (\textsc{Match}) \qquad \frac{\Gamma, n : T \vdash P}{\Gamma \vdash (\boldsymbol{\nu} n : T)P}\ (\textsc{Res}) \qquad \frac{\Gamma \vdash P}{\Gamma \vdash !P}\ (\textsc{Repl}) \qquad \frac{\Gamma \vdash \diamond}{\Gamma \vdash \mathbf{0}}\ (\textsc{Dead})$$

Table 3: Type System

The following definition provides a precise formalization of σ-low and σ-high contexts.

Definition 2. (σ-low and σ-high contexts) Let $\sigma \in \Sigma$. Consider the following grammar:

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

- A context $C[\cdot_\Gamma]$ is a σ-low context if it is a (Γ'/Γ)-context generated by the grammar above where $\Lambda(T) \preceq \sigma$ and $\Gamma' \vdash_\sigma P$.
- A context $C[\cdot_\Gamma]$ is a σ-high context if it is a (Γ'/Γ)-context generated by the grammar above where $\Lambda(T) \not\preceq \sigma$ and $\Gamma' \vdash^\sigma P$.

We write $\Gamma' \vdash_\sigma C[\cdot_\Gamma]$ (resp. $\Gamma' \vdash^\sigma C[\cdot_\Gamma]$) to indicate that $C[\cdot_\Gamma]$ is a σ-low (resp. σ-high) (Γ'/Γ)-context.

Example 1. Let be $\Gamma = h{:}\top[\bot[\,]],\ \ell{:}\bot[\,]$, $\Gamma' = \ell{:}\bot[\,]$ and $\sigma \prec \top$.

- The context $(\boldsymbol{\nu} h)(\overline{h}\langle \ell \rangle \mid [\cdot_\Gamma])$ is a (Γ'/Γ)-σ-high context since the process $\overline{h}\langle \ell \rangle$ in parallel with the hole can only perform a σ-high action; it can then interact with a process filling the hole through the high channel h.
- On the other hand, the context $(\boldsymbol{\nu} h)(\overline{h}\langle \ell \rangle) \mid [\cdot_\Gamma]$ is a (Γ'/Γ')-σ-low context.

We say that a type-indexed relation $\mathcal{R}$ over processes is *σ-contextual* if for any σ-low context C_L and for all σ-high contexts C_H^1 and C_H^2 such that $\Gamma' \vdash^\sigma C_H^1[\cdot_\Gamma]$, $C_H^2[\cdot_\Gamma]$ and $\Gamma'' \vdash_\sigma C_L[\cdot_{\Gamma'}]$ it holds that $\Gamma \models P\ \mathcal{R}\ Q$ imply $\Gamma'' \models C_L[C_H^1[P]]\ \mathcal{R}\ C_L[C_H^2[Q]]$.

Definition 3. (P-Congruence $\cong_\sigma$) Let $\sigma \in \Sigma$. The *σ-reduction barbed partial 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 next proposition establishes a precise comparison between the σ-reduction barbed p-congruence $\cong_\sigma$ and the standard reduction barbed congruence $\cong$ [9] cited in the Introduction.

PROPOSITION 1. *Let $\sigma \in \Sigma$, P and Q be two processes and Γ be a type environment such that $\Gamma \vdash P, Q$.*

1. *$\Gamma \models P \cong Q \;\not\Longrightarrow\; \Gamma \models P \cong_\sigma Q$.*
2. *$\Gamma \models P \cong_\sigma Q \;\not\Longrightarrow\; \Gamma \models P \cong Q$.*
3. *If $\Gamma \models P \cong_\sigma P$ then for all Q such that $\Gamma \models P \cong Q$ it holds $\Gamma \models Q \cong_\sigma Q$ and $\Gamma \models P \cong_\sigma Q$.*

PROOF. *We give two counter-examples for the statements* 1 *and* 2. *Consider the simple processes $P_1 = h().\ell()$, $P_2 = \ell().h()$ and $P_3 = \ell().k()$. Moreover, consider the type environment $\Gamma = h{:}\top[\,], k{:}\top[\,], \ell{:}\bot[\,]$. It is easy to see that $\Gamma \models P_1 \cong P_1$ but $\Gamma \models P_1 \not\cong_\sigma P_1$, and $\Gamma \models P_2 \cong_\sigma P_3$ but $\Gamma \models P_2 \not\cong P_3$.*
The proof of 3 *is trivial.*

The previous counter-examples also show that $\cong_\sigma$ is in general not reflexive, e.g., $\Gamma \models P_1 \not\cong_\sigma P_1$.

3.1 A proof technique for P-congruences

In this section we develop a proof technique for the relations $\cong_\sigma$ defined above, which is decidable over the set of finite state processes. More precisely, following [8, 9], we define a LTS of *typed actions* (called typed LTS) over configurations. As in [8], 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 [8], our typed actions are built around just a single type environment Γ constraining the observed process P. This differs from [8] 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 the presence of an additional input action with the form $(\boldsymbol{\nu}\tilde{p}{:}\tilde{T})\,n(\tilde{m})$ occurring when the process receives the new names $\tilde{p}$ generated by the environment.

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

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

$$\text{(Rep-Act)}\quad \frac{\Gamma \triangleright P \xrightarrow{\alpha}_{\delta} \Gamma' \triangleright P'}{\Gamma \triangleright !P \xrightarrow{\alpha}_{\delta} \Gamma' \triangleright P' \mid !P}$$

$$\text{(Weak)}\quad \frac{\Gamma, q{:}T \triangleright P \xrightarrow{(\boldsymbol{\nu}\tilde{p}:\tilde{T})\, n(\tilde{m})}_{\delta} \Gamma' \triangleright P' \quad q \neq n, \tilde{p} \quad q \in \tilde{m}}{\Gamma \triangleright P \xrightarrow{(\boldsymbol{\nu} q:T)(\boldsymbol{\nu}\tilde{p}:\tilde{T})\, n(\tilde{m})}_{\delta} \Gamma' \triangleright P'}$$

$$\text{(Open)}\quad \frac{\Gamma, q{:}T \triangleright P \xrightarrow{(\boldsymbol{\nu}\tilde{p}:\tilde{T})\, \overline{n}\langle\tilde{m}\rangle}_{\delta} \Gamma' \triangleright P' \quad q \neq n, \tilde{p} \quad q \in \tilde{m}}{\Gamma \triangleright (\boldsymbol{\nu} q{:}T)P \xrightarrow{(\boldsymbol{\nu} q:T)(\boldsymbol{\nu}\tilde{p}:\tilde{T})\, \overline{n}\langle\tilde{m}\rangle}_{\delta} \Gamma' \triangleright P'}$$

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

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

$$\text{(Red)}\quad \frac{P \xrightarrow{\tau} P'}{\Gamma \triangleright P \xrightarrow{\tau}_{\delta} \Gamma \triangleright P'}$$

Table 4: Typed LTS for π-calculus

A precise relationship between the untyped actions and the typed ones is established in the following proposition, whose proof is immediate.

Proposition 2. *Let $\Gamma \triangleright P$ be a configuration. Then*

- $\Gamma \triangleright P \xrightarrow{\tau}_{\delta} \Gamma \triangleright Q$ *if and only if* $P \xrightarrow{\tau} Q$.
- $\Gamma \triangleright P \xrightarrow{(\boldsymbol{\nu}\tilde{p}:\tilde{T})\, \overline{n}\langle\tilde{m}\rangle}_{\delta} \Gamma' \triangleright P'$ *iff* $P \xrightarrow{(\boldsymbol{\nu}\tilde{p}:\tilde{T})\, \overline{n}\langle\tilde{m}\rangle} P'$ *with* $\Lambda(\Gamma(n)) = \delta_1$ *and* $\delta_1 \preceq \delta$.
- $\Gamma \triangleright P \xrightarrow{(\boldsymbol{\nu}\tilde{p}:\tilde{T})\, n(\tilde{m})}_{\delta} \Gamma' \triangleright P'$ *if and only if* $P \xrightarrow{n(\tilde{m})} P'$ *with* $\Lambda(\Gamma(n)) = \delta_1$, $\delta_1 \preceq \delta$ *and* $\tilde{p} \cap Dom(\Gamma) = \emptyset$.

The next proposition shows how the type environment is modified after the execution of an action. It can be easily proved by induction on the depth of the derivation of the judgment in the hypothesis.

Proposition 3. *Let be $\Gamma \triangleright P \xrightarrow{\alpha}_{\delta} \Gamma' \triangleright P'$, then*

- *if* $\alpha = \tau$ *then* $\Gamma' = \Gamma$.
- *if* $\alpha \in \{(\boldsymbol{\nu}\tilde{p}{:}\tilde{T})\, \overline{n}\langle\tilde{m}\rangle,\ (\boldsymbol{\nu}\tilde{p}{:}\tilde{T})\, n(\tilde{m})\}$ *then* $\Gamma' = \Gamma, \tilde{p}{:}\tilde{T}$.

Relying on the typed LTS, we now introduce a partial bisimilarity on σ-low actions, denoted with $\dot{\approx}_\sigma$, which provides a coinductive characterization of the σ-reduction barbed p-congruence $\cong_\sigma$. Intuitively, the relation $\dot{\approx}_\sigma$ observes the σ-low actions, while simulating the σ-high actions by internal transitions.

We use the following notation: given a security level $\sigma \in \Sigma$, we write $\Gamma \triangleright P \xrightarrow{\alpha}{}^{\sigma} \Gamma' \triangleright P'$ 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*.

With an abuse of notation, we write $\Longrightarrow$ for the reflexive and transitive closure of $\xrightarrow{\tau}_{\delta}$. We also write $\overset{\alpha}{\Longrightarrow}_{\delta}$ for $\Longrightarrow \xrightarrow{\alpha}_{\delta} \Longrightarrow$, and $\overset{\hat{\alpha}}{\Longrightarrow}_{\delta}$ for $\Longrightarrow$ if $\alpha = \tau$ and $\overset{\alpha}{\Longrightarrow}_{\delta}$ otherwise. Moreover, for a given relation $\mathcal{R}$ over configurations, we write $\Gamma \vDash P \,\mathcal{R}\, Q$ whenever $(\Gamma \triangleright P)\ \mathcal{R}\ (\Gamma \triangleright Q)$. When $\tilde{T} = T_1, \ldots, T_k$ we denote by $\Lambda(\tilde{T})$ the least upper bound of levels $\Lambda(T_1), \ldots, \Lambda(T_k)$.

Definition 4. (Partial Bisimilarity on σ-low actions $\dot{\approx}_\sigma$) Let $\sigma \in \Sigma$. Partial bisimilarity on σ-low actions is the largest symmetric relation $\dot{\approx}_\sigma$ over configurations, such that whenever $\Gamma \vDash P \dot{\approx}_\sigma Q$

- if $\Gamma \triangleright P \xrightarrow{\alpha}_{\sigma} \Gamma' \triangleright P'$, then there exists Q' such that $\Gamma \triangleright Q \overset{\hat{\alpha}}{\Longrightarrow} \Gamma' \triangleright Q'$ with $\Gamma' \vDash Q' \dot{\approx}_\sigma P'$.
- if $\Gamma \triangleright P \xrightarrow{\alpha}{}^{\sigma} \Gamma' \triangleright P'$ with α belonging to $\{(\boldsymbol{\nu}\tilde{p}{:}\tilde{T})\, \overline{n}\langle\tilde{m}\rangle,\ (\boldsymbol{\nu}\tilde{p}{:}\tilde{T})\, n(\tilde{m})\}$ where $\tilde{p} : \tilde{T} = \tilde{p}_1{:}\tilde{T}_1,\ \tilde{p}_2{:}\tilde{T}_2$ such that $\Lambda(\tilde{T}_1) \not\preceq \sigma$, and $\Lambda(\tilde{T}_2) \preceq \sigma$, then there exists Q' such that $\Gamma \triangleright Q \Longrightarrow \Gamma \triangleright Q'$ with $\Gamma, \tilde{p}_1{:}\tilde{T}_1 \vDash Q' \dot{\approx}_\sigma (\boldsymbol{\nu}\tilde{p}_2{:}\tilde{T}_2)P'$.

To give an intuition of the second item in which restricted low and high names are handled differently, consider the processes $P = (\boldsymbol{\nu}\ell)(\overline{h}\langle\ell\rangle.\overline{\ell}\langle\rangle.R)$ and $Q = (\boldsymbol{\nu}k)(\overline{h}\langle k\rangle.\overline{k}\langle\rangle.R)$ in the type environment $\Gamma = h{:}\top[\,], k{:}\top[\,], \ell{:}\bot[\,]$. In both processes a name is extruded along a high level channel, which means that only high level contexts can receive that name and use it to synchronize on the second action. However, when the extruded name is low the high context cannot read from it, hence no context will ever interact with R. On the other hand, when the extruded name is high the high context can synchronize on the second action and possibly interact with the process R.

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 $\Gamma \vDash P \not\dot{\approx}_\sigma P$ when $\sigma = \bot$.

Theorem 1. *Let $\sigma \in \Sigma$, P and Q be processes and Γ be a type environment such that $\Gamma \vdash P, Q$. It holds:*

$$\Gamma \vDash P \cong_\sigma Q \textit{ if and only if } \Gamma \vDash P \dot{\approx}_\sigma Q.$$

4. INFORMATION FLOW

We are now ready to define noninterference in terms of P-congruences in the spitit of [18]. This property is called $\mathcal{NI}(\cong_\sigma)$ and ensures that no information flow occurs even in the presence of malicious processes, e.g., Trojan Horse programs, that run at the classified (higher than σ) level.

A process P in a type environment Γ satisfies the property $\mathcal{NI}(\cong_\sigma)$ if the configuration $\Gamma \triangleright P$ belongs to the reflexivity closure of $\cong_\sigma$. The formal definition of $\mathcal{NI}(\cong_\sigma)$ is as follows.

Definition 5. (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

$$\Gamma \vDash P \cong_\sigma P.$$

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

- Let us first consider the following simple insecure process: $P_1 = h().\ell() \mid \overline{h}\langle\rangle$. 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 $C^1_H[\cdot_\Gamma] = h() \mid [\cdot_\Gamma]$ and $C_L[\cdot_\Gamma] = C^2_H[\cdot_\Gamma] = [\cdot_\Gamma]$, one can easily observe that $\Gamma \vDash C_L[C^1_H[P_1]] \not\cong_\sigma C_L[C^2_H[P_1]]$.

- 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 $\Gamma' = h : \top[T],\ \ell_i : \bot[],\ n : T$ (here the security level of n is irrelevant). To show that $\Gamma' \triangleright P_2 \notin \mathcal{NI}(\cong_\sigma)$ one can choose $C^1_H[\cdot_{\Gamma'}] = \overline{h}\langle n\rangle \mid [\cdot_{\Gamma'}], C_L[\cdot_{\Gamma'}] = C^2_H[\cdot_{\Gamma'}] = [\cdot_{\Gamma'}]$, and observe that $\Gamma' \vDash C_L[C^1_H[P_2]] \not\cong_\sigma C_L[C^2_H[P_2]]$. 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 $C^1_H[\cdot] = h(y).h(z).\overline{h}\langle n\rangle \mid [\cdot]$ and $C_L[\cdot] = C^2_H[\cdot] = [\cdot]$, then $C_L[C^1_H[P_3]] \not\cong_\sigma C_L[C^2_H[P_3]]$.

The characterization of p-congruences in terms of partial bisimilarity on σ-low actions 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 partial bisimilarity on σ-low actions is decidable in the case of finite state processes, i.e., processes whose typed LTS is finite. Moreover, by exploiting the following compositionality results, the partial bisimilarity on σ-low actions can be used to define methods, e.g., a proof system, both to check the security of complex systems and to incrementally build processes which are secure by construction.

THEOREM 2. *(Compositionality of σ-Noninterference) Let $\sigma \in \Sigma$, P and Q be two processes and Γ be a type environment such that $\Gamma \vdash P, Q$. If $\Gamma \triangleright P \in \mathcal{NI}(\cong_\sigma)$ and $\Gamma \triangleright Q \in \mathcal{NI}(\cong_\sigma)$ then*

1. $\Gamma' \triangleright \overline{a}\langle\tilde{b}\rangle.P \in \mathcal{NI}(\cong_\sigma)$ *where* $\Gamma' = \Gamma \cup \{a : \delta[\tilde{T}]\} \cup \{\tilde{b} : \tilde{T}\}$ *and* $\delta \preceq \sigma$*;*
2. $\Gamma' \triangleright a(\tilde{x} : \tilde{T}).P \in \mathcal{NI}(\cong_\sigma)$ *where* $\Gamma' = \Gamma \cup \{a : \delta[\tilde{T}]\}$ *and* $\delta \preceq \sigma$*;*
3. $\Gamma' \triangleright \mathsf{if}\ a = b\ \mathsf{then}\ P\ \mathsf{else}\ Q \in \mathcal{NI}(\cong_\sigma)$ *where* $\Gamma' = \Gamma \cup \{a : T\} \cup \{b : T\}$*;*
4. $\Gamma \triangleright P \mid Q \in \mathcal{NI}(\cong_\sigma)$*;*
5. $\Gamma' \triangleright (\boldsymbol{\nu} n{:}T)\, P \in \mathcal{NI}(\cong_\sigma)$ *where* $\Gamma = \Gamma', n : T$*;*
6. $\Gamma \triangleright !P \in \mathcal{NI}(\cong_\sigma)$.

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

5. EXTENSION WITH DOWNGRADING

In this section we extend the π-calculus with a declassification mechanism that allows a programmer to control information release from higher to lower levels.

The new calculus, called $\mathsf{Dec}\,\pi$-calculus, is obtained by enriching the syntax of processes with a family of declassified actions of the form $\overline{\mathsf{dec}_\delta\, n}\langle\tilde{m}\rangle$ and $\mathsf{dec}_\delta\, n(\tilde{x}{:}\tilde{T})$. Whenever n is a channel of level higher than δ, the declassified read/write actions $\mathsf{dec}_\delta\, n(\tilde{x}{:}\tilde{T})$ and $\overline{\mathsf{dec}_\delta\, n}\langle\tilde{m}\rangle$ can be used by the programmer to specify an "escape hatch" for information release, that is to allow information arising from these actions to flow downwards up to the level δ. Notice that the declassification has a visible impact only when a σ-high action is declassified to an observable level δ (i.e., when $\delta \preceq \sigma$). However, following the lines of the previous sections, we prefer to introduce a downgrading mechanism which is parametric on the security levels.

According to the literature, we assume a declassification model where only programmers may enable the downgrading of secret information to an observable level, while external entities can only synchronize on those declassified actions that do not allow the flow of information to cross the observation level σ.

We refer to [2] for a detailed discussion of motivations and a formal study of $\mathsf{Dec}\,\pi$-calculus. In this section we show how p-congruences and the related noninterference propreties can be extended to deal with downgrading.

The type system of the $\mathsf{Dec}\,\pi$-calculus ensures that actions can be downgraded only to lower levels. It can be obtained by adding the following rules to those in Table 3.

(DEC OUTPUT)

$$\frac{\Gamma \vdash \overline{a}\langle\tilde{b}\rangle.P \quad \Gamma \vdash a : \delta_1[\tilde{T}]}{\Gamma \vdash \overline{\mathsf{dec}_\delta\, a}\langle\tilde{b}\rangle.P}\ \delta \prec \delta_1$$

(DEC INPUT)

$$\frac{\Gamma \vdash a(\tilde{x} : \tilde{T}).P \quad \Gamma \vdash a : \delta_1[\tilde{T}]}{\Gamma \vdash \mathsf{dec}_\delta\, a(\tilde{x} : \tilde{T}).P}\ \delta \prec \delta_1$$

The operational semantics of the $\mathsf{Dec}\,\pi$-calculus is obtained from that of the π-calculus by adding the rules in Table 5, that are built around the new actions $\mathsf{dec}_\delta\, n(\tilde{m})$, $\overline{\mathsf{dec}_\delta\, n}\langle\tilde{m}\rangle$ and $(\boldsymbol{\nu}\tilde{p}{:}\tilde{T})\,\overline{\mathsf{dec}_\delta\, n}\langle\tilde{m}\rangle$. Notice that we allow a declassified action to synchronize only with the corresponding declassified co-action. In other words, we require that both users of a channel (the reader and the writer) agree to downgrade the communication.

$$\frac{}{\overline{\mathsf{dec}_\delta\, n}\langle\tilde{m}\rangle.P \xrightarrow{\overline{\mathsf{dec}_\delta\, n}\langle\tilde{m}\rangle} P} \ \text{(Dec Out)} \qquad \frac{}{\mathsf{dec}_\delta\, n(\tilde{x}{:}\tilde{T}).P \xrightarrow{\mathsf{dec}_\delta\, n(\tilde{m})} P\{\tilde{x} := \tilde{m}\}} \ \text{(Dec In)}$$

$$\frac{P \xrightarrow{(\boldsymbol{\nu}\tilde{p}:\tilde{T})\,\overline{\mathsf{dec}_\delta\, n}\langle\tilde{m}\rangle} P' \quad q \neq n \quad q \in \tilde{m}}{(\boldsymbol{\nu}q{:}T)P \xrightarrow{(\boldsymbol{\nu}q:T)(\boldsymbol{\nu}\tilde{p}:\tilde{T})\,\overline{\mathsf{dec}_\delta\, n}\langle\tilde{m}\rangle} P'} \ \text{(Dec Open)}$$

$$\frac{P \xrightarrow{(\boldsymbol{\nu}\tilde{p}:\tilde{T})\,\overline{\mathsf{dec}_\delta\, n}\langle\tilde{m}\rangle} P' \quad Q \xrightarrow{\mathsf{dec}_\delta\, n(\tilde{m})} Q' \quad \tilde{p} \cap \mathrm{fn}(Q) = \emptyset}{P \mid Q \xrightarrow{\tau} (\boldsymbol{\nu}\tilde{p} : \tilde{T})(P' \mid Q')} \ \text{(Dec Close)}$$

Table 5: Labeled Transition System for Declassified Actions

The theory of p-congruences developed in the previous sections scales to the $\mathsf{Dec}\,\pi$-calculus by adapting the definitions of σ-low and σ-high contexts.

As we said before, the downgrading must be controlled by programmers, whereas neither external observers, nor attackers can fire a declassified communication that allows the flow of information to cross the observation level σ. This results in defining σ-low and σ-high level sources as follows, where the only allowed declassifications do not let the flow of information to cross the level σ.

Definition 6. (σ-low and σ-high level sources) Let $\Gamma \vdash P$.

- We say that the process P is a *σ-low level source* in Γ, denoted $\Gamma \vdash_\sigma P$, if $\Gamma \vdash P$ and $\forall m \in \mathrm{fn}(P)$ it holds $\Lambda(\Gamma(m)) \preceq \sigma$.
- We say that the process P is a *σ-high level source* in Γ, denoted $\Gamma \vdash^\sigma P$, if every subject of an input or an output prefix is of the form a with $\Lambda(\Gamma(a)) \not\preceq \sigma$ or $\mathsf{dec}_\delta\, a$ with $\delta \not\preceq \sigma$.

Now σ-low and σ-high contexts can be defined in the $\mathsf{Dec}\,\pi$-calculus exactly as in Definition 2 where σ-low and σ-high level sources are defined according to definition 6.

As for σ-barb preservation, the $\mathsf{Dec}\,\pi$-calculus inherits the definition of σ-barbs form Section 3. Notice that these barbs are enough to also observe those declassified actions that use low level channels.

The p-congruences and the noninterference property for the $\mathsf{Dec}\,\pi$-calculus are then defined as follows:

Definition 7. (P-Congruence $\cong_\sigma^{\mathsf{dec}}$) Let $\sigma \in \Sigma$. The *σ-reduction barbed partial congruence*, denoted by $\cong_\sigma^{\mathsf{dec}}$, is the largest type-indexed relation over processes which is symmetric, σ-contextual, reduction closed and σ-barb preserving (according to the above definitions adapted to the $\mathsf{Dec}\,\pi$-calculus).

Definition 8. (σ-Noninterference) Let $\sigma \in \Sigma$, P be a $\mathsf{Dec}\,\pi$-process and Γ be a type environment such that $\Gamma \vdash P$. The process P satisfies the *σ-noninterference property in* Γ, written $\Gamma \triangleright P \in \mathcal{NI}(\cong_\sigma^{\mathsf{dec}})$, if

$$\Gamma \vDash P \cong_\sigma^{\mathsf{dec}} P.$$

Example 4. Consider the processes $P_1 = h().\ell() \mid \overline{h}\langle\rangle$ and $P_2 = \mathsf{dec}_\delta\, h().\ell() \mid \overline{\mathsf{dec}_\delta\, h}\langle\rangle$ in the type environment $\Gamma = h{:}\top[\,], \ell{:}\bot[\,]$ and let δ be an observable level under σ. We can prove that P_1 is not secure since $\Gamma \vDash P_1 \not\cong_\sigma P_1$, whereas $\Gamma \vDash P_2 \cong_\sigma^{\mathsf{dec}} \ell()$, hence P_2 is a secure process. The difference between the to processes comes from the fact that σ-high contexts can interfere with the plain communication along the channel h, but not with the declassified communication. On the other hand, the process $P_3 = h_1().\mathsf{dec}_\delta\, h().\ell() \mid \overline{\mathsf{dec}_\delta\, h}\langle\rangle$ with h_1 being a high channel is insecure since observing the low action $\ell()$ reveals the occurrence of the high actions $h_1()$ and $h()$, but only the second one has been downgraded by the programmer.

In order to define a proof technique for noninterference, the partial bisimilarity studied in Section 3 can be adapted to the $\mathsf{Dec}\,\pi$-calculus by extending the typed LTS of Table 4 with the declassified actions collected in Table 6. The new rules state that if a σ-high action, e.g., $\overline{h}\langle n\rangle$ is declassified to an observable level δ, then the resulting action $\overline{\mathsf{dec}_\delta\, h}\langle n\rangle$ is still a σ-high action. This is justified by the fact that σ-low contexts cannot synchronize on (i.e., observe) declassified actions, thus setting to δ the level of the action $\overline{\mathsf{dec}_\delta\, h}\langle n\rangle$ would be wrong. Even if the downgrading does not affect the level of a typed action, the examples above show the actual impact of declassification on the admissible information flows. The following definition of partial bisimilarity gives a further account of the impact of declassification on security.

Definition 9. (Partial Bisimilarity on σ-low actions $\dot{\approx}_\sigma^{\mathsf{dec}}$) Let $\sigma \in \Sigma$. Partial bisimilarity on σ-low actions is the largest symmetric relation $\dot{\approx}_\sigma^{\mathsf{dec}}$ over configurations, such that whenever $\Gamma \vDash P \dot{\approx}_\sigma^{\mathsf{dec}} Q$

- if $\Gamma \triangleright P \xrightarrow{\alpha}_\sigma \Gamma' \triangleright P'$, then there exists Q' such that $\Gamma \triangleright Q \stackrel{\hat{\alpha}}{\Longrightarrow} \Gamma' \triangleright Q'$ with $\Gamma' \vDash Q' \dot{\approx}_\sigma P'$.
- if $\Gamma \triangleright P \xrightarrow{\alpha}{}^\sigma \Gamma' \triangleright P'$ with α belonging to $\{(\boldsymbol{\nu}\tilde{p}{:}\tilde{T})\,\overline{n}\langle\tilde{m}\rangle,$ $(\boldsymbol{\nu}\tilde{p}{:}\tilde{T})\, n(\tilde{m})\}$ where $\tilde{p} : \tilde{T} = \tilde{p}_1{:}\tilde{T}_1,\ \tilde{p}_2{:}\tilde{T}_2$ such that $\Lambda(\tilde{T}_1) \not\preceq \sigma$, and $\Lambda(\tilde{T}_2) \preceq \sigma$, then there exists Q' such that $\Gamma \triangleright Q \Longrightarrow \Gamma \triangleright Q'$ with $\Gamma, \tilde{p}_1{:}\tilde{T}_1 \vDash Q' \dot{\approx}_\sigma (\boldsymbol{\nu}\tilde{p}_2{:}\tilde{T}_2)P'$.

Even if the definition of $\dot{\approx}_\sigma^{\mathsf{dec}}$ appears identical to Definition 4, its second clause implies that σ-high actions that have been declassified by P to an observable level, need not to be matched by τ-steps of Q, thus implementing the fact that they represent an (explicitly allowed) information flow.

In the $\mathsf{Dec}\,\pi$-calculus the following theorem holds by a straightforward extention of the proof of Theorem 1.

THEOREM 3. *Let $\sigma \in \Sigma$, P and Q be processes and Γ be a type environment such that $\Gamma \vdash P, Q$. It holds:*

$$\Gamma \vDash P \cong_\sigma^{\mathsf{dec}} Q \text{ if and only if } \Gamma \vDash P \dot{\approx}_\sigma^{\mathsf{dec}} Q.$$

(DEC OUT)

$$\frac{\Gamma \vdash n : \delta_1[\tilde{T}]}{\Gamma \triangleright \overline{\mathsf{dec}_{\delta_2} n}\langle \tilde{m} \rangle.P \xrightarrow{\overline{\mathsf{dec}_{\delta_2} n}\langle \tilde{m} \rangle}_{\delta} \Gamma \triangleright P} \; \delta_1 \preceq \delta$$

(DEC IN)

$$\frac{\Gamma \vdash n : \delta_1[\tilde{T}] \quad \Gamma \vdash \tilde{m} : \tilde{T}}{\Gamma \triangleright \mathsf{dec}_{\delta_2} n(\tilde{x}{:}\tilde{T}).P \xrightarrow{\mathsf{dec}_{\delta_2} n(\tilde{m})}_{\delta} \Gamma \triangleright P\{\tilde{x} := \tilde{m}\}} \; \delta_1 \preceq \delta$$

(DEC WEAK)

$$\frac{\Gamma, q{:}T \triangleright P \xrightarrow{(\boldsymbol{\nu} \tilde{p}{:}\tilde{T})\, \mathsf{dec}_{\delta_1} n(\tilde{m})}_{\delta} \Gamma' \triangleright P' \quad q \neq n, \tilde{p} \quad q \in \tilde{m}}{\Gamma \triangleright P \xrightarrow{(\boldsymbol{\nu} q{:}T)(\boldsymbol{\nu} \tilde{p}{:}\tilde{T})\, \mathsf{dec}_{\delta_1} n(\tilde{m})}_{\delta} \Gamma' \triangleright P'}$$

(DEC OPEN)

$$\frac{\Gamma, q{:}T \triangleright P \xrightarrow{(\boldsymbol{\nu} \tilde{p}{:}\tilde{T})\, \overline{\mathsf{dec}_{\delta_1} n}\langle \tilde{m} \rangle}_{\delta} \Gamma' \triangleright P' \quad q \neq n, \tilde{p} \quad q \in \tilde{m}}{\Gamma \triangleright (\boldsymbol{\nu} q{:}T)P \xrightarrow{(\boldsymbol{\nu} q{:}T)(\boldsymbol{\nu} \tilde{p}{:}\tilde{T})\, \overline{\mathsf{dec}_{\delta_1} n}\langle \tilde{m} \rangle}_{\delta} \Gamma' \triangleright P'}$$

Table 6: Typed LTS for Dec π-calculus

6. CONCLUSION AND RELATED WORK

In this paper we introduce the concept of *p-congruences* to model noninterference for a typed version of the π-calculus. We first consider a simple typing discipline where types are used to assign secrecy levels to channels. We then extend our approach by allowing the secure downgrading of information through explicit declassification operations.

With respect to our previous works [3, 2], in this paper we provide a more precise and efficient definition of noninterference. As discussed in the introduction, the security definitions presented in [3, 2] are persistent in the sense that if a process is secure then also all the states reachable from it in the typed LTS do. Although persistence allows us to apply inductive reasoning when proving security results, it turns out to be too strong when modeling noninterference.

In this paper we show that persistence may be replaced by a more sophisticated notion of contextuality which leads to more precise and efficient definitions in terms of a generalized version of reduction barbed congruence. We precisely compare this new relation with the standard barbed congruence for the π-calculus, and we develop a coinductive characterization that provides efficient methods for the verification and construction of (compositional) secure systems.

A number of type-based techniques ensuring forms of noninterference have been developed for the π-calculus. Between them, the security π-calculus of Hennessy and Riely [10, 8] is the closest one. It consists of a typed version of the asynchronous π-calculus where types associate read/write capabilities to channels as well as security clearances. Noninterference properties based on may and must equivalences are achieved by means of typing constraints forcing a no-write-down policy. In particular, the noninterference results presented by Hennessy in [8] can be stated succintly as follows. If a well typed process P only performs input actions of level at most σ, then for all processes H whose outputs are not observable (i.e., H writes at levels not lower that σ), P and $P \mid H$ are indistinguishable by any testing process running at security level at most σ. The noninterference theorem comes from the fact that in the security π-calculus communication between processes of different levels is precisely constrained. Let P and H be two communicating processes, then either (1) P sends an output to H, but this case does not produce any information leak due to the fact that the output is asynchronous, or (2) P reads a message from H, which is instead prevented by the typing rules which forbid write-downs.

Our type system can be embedded into that of [8] using the following type translation:

$$[\![\, \delta[T] \,]\!] = \{\mathsf{r}_\delta([\![T]\!]),\ \mathsf{w}_\delta([\![T]\!])\}$$

where a channel of level δ corresponds to a read-write channel where both capabilities have security level δ. Such a translation helps in understanding the difference between the two approaches. Splitting the read and write capabilites and using an associated subtyping relation increases the flexibility of typing, however, consider the process $P = \ell().h()$, which is clearly a secure process. Besides being well typed, P does not match the hypothesis of the noninterference theorem of [8] since it contains an high level input, hence nothing can be said about its security properties, whereas it can be easily proved that $P \in \mathcal{NI}(\cong_\sigma)$.

Furthermore, in the security π-calculus no observation congruences are studied. We think that proving contextual noninterference results using flexible type systems of i/o-types is not straightforward, and we keep it for future work.

Honda, Yoshida, Vasconcelos and Berger [11, 13, 22] consider advanced type systems for processes of the linear/affine π-calculus where each action type is associated to a secrecy level. They 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 for 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 [23] 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 [14] 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, consider the process $!x(y).P|!x(y).Q$. It is rejected by the type system of, e.g., [13] and thus it is not considered secure 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 even if this process does not leak any secret information. As another example consider the process $P = (\boldsymbol{\nu} h)(\overline{h} \mid !(h.(\overline{k} \mid \overline{h})) \mid k.\overline{\ell})$ with h and k being high channels and l being a low one. We can prove that P satisfies noninterference. However it cannot be deemed secure by using the type systems in the above mentioned works. The problem comes from the insecure subterm $k.\overline{\ell}$ where an observable action depends on a high one.

Our use of a lighter type system leads to stronger noninterference properties, that check the security of processes against a bigger class of attackers. Interestingly, as shown in Section 5, we can increase the flexibility of our approach by admitting a form of *downgrading* which allows trusted entities to declassify information from a higher to a lower security level. This is done following the ideas previously developped in [1, 2]. Thus, for instance, the process $h().\overline{\ell}\langle\rangle$ can be deemed secure by declassifying the high level action $h()$.

As for the downgrading, the work which is most related to our approach is [1] by Bossi, Piazza and the second author. They propose a general unwinding framework for formalizing different nointerference properties of CCS processes permitting downgrading. Their calculus is not extended with any particular declassification operator but instead a distinct set D of downgrading actions is considered. The main advantage of explicitly introducing a declassification construct, as the dec operator used in this paper, is that, in the same process, high level names can be used both as secret channels and as downgraded ones. This is clearly not possible in [1].

The only work we are aware of dealing with a form of downgrading for the π-calculus is a recent work by Gordon and Jeffrey about conditional secrecy [7]. They propose a system of secrecy types for the π-calculus which supports multiple, dynamically-generated security levels, together with the controlled downgrading of security levels. Differently from our approach, their system downgrades names instead of actions and is based on trace semantics. Furthermore, their security notion deals with direct flows only and does not address implicit flows nor noninterference.

7. ACKNOWLEDGMENTS

We would like to acknowledge the anonymous referees for their useful and stimulating comments which helped us in improving the presentation.

8. REFERENCES

[1] 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.

[2] S. Crafa and S. Rossi. Controlling Information Release in the π-calculus. Technical Report 14, Dipartimento di Matematica Pura e Applicata, Università DI Padova, Italy, 2005. http://www.math.unipd.it/~crafa/TR-14-05. ps.gz.

[3] S. Crafa and S. Rossi. A Theory of Noninterference for the π-calculus. In *Proceedings of the International Symposium on Trustworthy Global Computing (TGC'05)*, volume 3705 of *LNCS*, pages 2–18. Springer-Verlag, 2005.

[4] R. Focardi and R. Gorrieri. A Classification of Security Properties for Process Algebras. *Journal of Computer Security*, 3(1):5–33, 1994/1995.

[5] 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.

[6] 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.

[7] A. D. Gordon and A. S. A. Jeffrey. Secrecy despite compromise: Types, cryptography, and the pi-calculus. In *Proc. of the 16th International Conference on Concurrency Theory, (CONCUR'05)*, volume 3653 of *LNCS*, pages 186–201. Springer-Verlag, 2005.

[8] M. Hennessy. The security picalculus and non-interference. *Journal of Logic and Algebraic Programming*, 63(1):3–34, 2004.

[9] 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.

[10] 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.

[11] K. Honda, V. 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.

[12] K. Honda and N. Yoshida. On Reduction-based Process Semantics. *Theoretical Computer Science*, 152(2):437–486, 1995.

[13] 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.

[14] N. Kobayashi. Type-Based Information Flow Analysis for the Pi-Calculus. *Acta Informatica*, 42(4–5):291–347, 2005.

[15] F. Pottier. A simple view of type-secure information flow in the π-calculus. In *Proc. of the IEEE Computer Security Foundations Workshop (CSFW'02)*, pages 320–330. IEEE Computer Society Press, 2002.

[16] P. Ryan and S. Schneider. Process Algebra and Non-Interference. *Journal of Computer Security*, 9(1/2):75–103, 2001.

[17] A. Sabelfeld and A. C. Myers. Language-Based Information-Flow Security. *IEEE Journal on Selected Areas in Communication*, 21(1):5–19, 2003.

[18] A. Sabelfeld and D. Sands. A Per Model of Secure Iinformation Flow in Sequential Programs. In *Proc. of European Symposium on Programming (ESOP'99)*, volume 1576 of *LNCS*, pages 40–58. Springer-Verlag, 1999.

[19] 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.

[20] A. Sabelfeld and D. Sands. Dimensions and principles of declassification. In *Proc. of the IEEE Computer Security Foundations Workshop (CSFW'05)*, pages 255–269. IEEE Computer Society Press, 2005.

[21] D. Sangiorgi and D. Walker. *The pi calculus: A theory of mobile processes.* Cambridge, 2001.

[22] 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.

[23] S. Zdancewic and A. C. Myers. Observational Determinism for Concurrent Program Security. In *Proc. of the IEEE Computer Security Foundations Workshop (CSFW'03)*, pages 29–45. IEEE Computer Society Press, 2003.