Available online at www.sciencedirect.com

ScienceDirect

Information and Computation 205 (2007) 1235–1273

Information and Computation

www.elsevier.com/locate/ic

Controlling information release in the π-calculus[☆]

Silvia Crafa [a,*], Sabina Rossi [b]

[a] Dipartimento di Matematica Pura e Applicata, Università di Padova, Italy
[b] Dipartimento di Informatica, Università Ca' Foscari di Venezia, Italy

Received 15 July 2005; revised 15 December 2006
Available online 21 January 2007

Abstract

We introduce a notion of *controlled information release* for a typed version of the π-calculus extended with declassification primitives; this property scales to noninterference when downgrading is not allowed. We provide various characterizations of controlled release, based on a typed behavioural equivalence relative to a security level σ, which captures the idea of external observers of level σ. First, we define our security property through a universal quantification over all the possible *active attackers*, i.e., malicious processes which interact with the system possibly leaking secret information. Then we characterize the controlled release property in terms of an unwinding condition, which deals with so-called *passive attackers* trying to infer confidential information just by observing the behaviour of the system. Furthermore, we express controlled information release in terms of partial equivalence relations (*per* models, for short) in the style of a stream of similar studies for imperative and multi-threaded languages. We show that the controlled release property is compositional with respect to most operators of the language leading to efficient proof techniques for the verification and the construction of (compositional) secure systems.

Keywords: Process Algebra; Noninterference; Downgrading

1. Introduction

A central challenge in information security of multilevel systems is the protection of sensitive data and resources from undesired access while also permitting information release whenever appropriate. Indeed, real-world applications often release bits of information as part of their intended behavior. 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. Another example is a password checking program: some information about the password is released even if a log-in attempt fails,

[☆] 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).
[*] Corresponding author.
E-mail addresses: crafa@math.unipd.it (S. Crafa), srossi@dsi.unive.it (S. Rossi).

since an attacker may learn whether the inserted sequence coincides or not with the password. (See Example 34 in Section 5 for a formalization of this example in the π-calculus framework.)

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. In the presence of downgrading, a satisfactory security policy should allow controlled forms of information release that weaken the rigid restrictions of classical noninterference properties. 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 and Sands presented in [1] for a road map of the main directions of the research on this topic.

In this paper, we study the downgrading in the π-calculus framework: we develop a general theory of a relaxed notion of noninterference for the π-calculus, which allows information to flow from a higher to a lower security level through a downgrader. In the π-calculus a piece of information is just the name of a communication channel, and such a name can be used to access that channel (in read or write mode) and to communicate it to other processes. According to this model, in the π-calculus the noninterference principle is centered around the behavior of processes, i.e., on their (dynamic) abilities of reading and writing through channels, rather than on the value of input or output variables as in the imperative framework.

There exists a quite extensive literature on type-based proof techniques for noninterference in the π-calculus (see, e.g., [2–6]), but none of these works deals with downgrading. Their main result is a soundness theorem stating that if a system is well-typed, then it is interference free. Moreover, a common feature of these works is that 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.

In the present paper, we follow a different approach where the security properties of a process are checked by reasoning on its operational semantics (i.e., its behavior), rather than by imposing typing constraints. We study the *controlled information release* property, which generalizes noninterference by allowing secure downgrading of information through an explicit declassification operation. As discussed above, even in the case of downgrading we part from the approaches followed by functional and imperative languages, where the objects of declassifications are expressions built from the value of high variables. Instead, given a π-calculus process, we declassify part of its behavior, that is, some of its read/write actions. Such a new property scales to noninterference when downgrading is not admitted.

More precisely, in order to deal with downgrading, 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 a complete lattice $\langle \Sigma, \preceq\rangle$ of security annotations. We call the new language the $\mathsf{Dec}\,\pi$-calculus. The intuition is that only programmers may enable the downgrading of secret information, while external entities cannot synchronize on declassified actions. If a is a channel of a security level σ with $\delta \preceq \sigma$ 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 $\overline{\mathsf{dec}_\delta\, a}\langle b\rangle$.

It is worth noticing that we could have used the dec constructor to declassify names rather than actions. However, since in the standard π-calculus a name represents a channel, a declassified name would be a channel over which only declassified actions are allowed. Hence, by declassifying actions instead of names, we obtain a more flexible, finer-grained, downgrading mechanism that allows programmers to interleave declassified and non declassified actions over the same channel (e.g., see Example 11 in Section 4, and Example 34 in Section 5).

In our theory the use of types is much lighter than in the previous works on the security for the π-calculus. 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 and that an action of a security level σ can be downgraded only at a lower level δ. Such a typing discipline ensures that information does not *directly* flow from high to low in an uncontrolled manner, however, intentional release is explicitly specified through the use of declassified actions. Being so light-weight, the type system does not deal with implicit flows and thus we do not use it as a proof technique for our security definition. On the contrary, we characterize the controlled release property in terms of the actions that typed processes may perform.

Our approach generalizes previous ideas, mainly developed for CCS (see [7]), 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 controlled information release for the π-calculus enriched with a downgrading primitive and (ii) we provide a number of sound and complete characterizations of secure processes, which can be exploited in the design of efficient verification techniques.

The controlled release property we are going to study is based on the notion of process behavior relative to a security level σ. In particular, we define a family of typed equivalences for the $\mathsf{Dec}\,\pi$-calculus indexed by an observation level σ, namely σ-reduction barbed congruences (see [8]). Two processes P and Q are σ-equivalent in the type environment Γ, written $\Gamma \models P \cong_\sigma Q$, if they exhibit the same σ-level behavior, 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, independently of downgrading. 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 labeled 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 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 $\mathsf{Dec}\,\pi$-calculus, we introduce the σ-controlled information release property $\mathcal{CR}(\cong_\sigma)$ for typed processes, which is inspired by the *DP_BNDC* property defined in [7] for CCS. As stated above, the controlled release property is built upon a declassification model that allows programmers to intentionally release sensitive information in a controlled way. In particular, only internal system components are allowed to downgrade sensitive information, therefore we model external attackers as high level processes which cannot enable the declassification, i.e., cannot perform downgraded actions. We say that a process P in a type environment Γ satisfies the property $\mathcal{CR}(\cong_\sigma)$, written $\Gamma \triangleright P \in \mathcal{CR}(\cong_\sigma)$, if for every configuration $\Gamma' \triangleright P'$ reachable from $\Gamma \triangleright P$ in the typed LTS, and for every σ-high level source H (that is a process which can perform only non downgraded actions over channels of level higher than σ) it holds

$$\Gamma' \triangleright P' \cong_\sigma \Gamma' \triangleright P'|H.$$

This definition involves a universal quantification over all the possible *active attackers*, 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 $\Gamma \triangleright P$ satisfies $\mathcal{CR}(\cong_\sigma)$ then also all the configurations reachable from it in the typed LTS satisfy $\mathcal{CR}(\cong_\sigma)$. As discussed in [9], 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.

Perhaps interestingly, we show that $\mathcal{CR}(\cong_\sigma)$ can be rephrased in terms of name restriction and the standard bisimilarity (i.e., bisimilarity on $\top$-actions,[1] $\approx_\top$), obtaining a definition of controlled release which is reminiscent of the noninterference property called *BNDC* and defined by Focardi and Gorrieri in [10] for CCS processes.

We provide a characterization of $\mathcal{CR}(\cong_\sigma)$ in terms of an *unwinding* condition in the style of [11]. An unwinding condition aims at specifying local constraints on process transitions which imply some global property. More precisely, our unwinding condition requires that whenever a configuration C performs a non declassified 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 all the non declassified actions over channels of level higher than σ are always simulated by internal computations, thus becoming invisible for the low level observers. Nevertheless, intentional information release is still admitted, since the unwinding condition does not prevent flows arising from high level declassified actions.

It is interesting to observe that the unwinding condition characterizes security with respect to the so-called *passive attackers*, which try to infer information about the classified behavior (σ-high actions) only by observing the σ-level behaviour of the system.

We also provide a quantifier-free characterization of controlled release in terms of partial equivalence relations (*per* models, for short). More precisely, we introduce a partial equivalence relation $\dot{\approx}_\sigma$ over configurations

[1] Where $\top$ is the top-element in the security lattice Σ.

Table 1
Syntax

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
$\mid\ \overline{\mathsf{dec}_\delta\, a}\langle b\rangle$	Declassified output		$\mid\ P \mid P$	Parallel
$\mid\ \mathsf{dec}_\delta\, a(x{:}T)$	Declassified input		$\mid\ (\boldsymbol{\nu} n : T)P$	Restriction
Types			$\mid\ !P$	Replication
$T ::= \sigma[\,] \mid \sigma[T]$			$\mid\ \mathbf{0}$	Inactive

and, inspired by the definitions in [12] 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.

Thanks to the last two characterizations one can immediately prove that the σ-controlled release property $\mathcal{CR}(\cong_\sigma)$ is decidable for the recursion-free calculus and even for the finite-control π-calculus where the number of parallel components in any process is bounded by a constant. Furthermore, we show that $\mathcal{CR}(\cong_\sigma)$ is compositional with respect to most operators of the $\mathsf{Dec}\,\pi$-calculus. In particular, if P and Q satisfy $\mathcal{CR}(\cong_\sigma)$ and they do not synchronize on declassified actions, then $P|Q$ satisfies $\mathcal{CR}(\cong_\sigma)$. Similarly, if P satisfies $\mathcal{CR}(\cong_\sigma)$ and it does not contain declassified actions, then $!P$ satisfies $\mathcal{CR}(\cong_\sigma)$.

The rest of the paper is organized as follows. In Section 2, we present the $\mathsf{Dec}\,\pi$-calculus, 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-controlled release and provide a number of characterizations based on typed actions. In Section 5, we illustrate the expressivity of our approach through a number of examples of secure systems. In Sections 6 and 7, we discuss some extension and related work. Section 8 concludes the paper. Two appendixes contain the proofs omitted from the main text.

2. The $\mathsf{Dec}\,\pi$-calculus

In this section, we introduce the $\mathsf{Dec}\,\pi$-calculus, which extends the typed π-calculus with a construct for declassifying actions. 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.[2] 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. The calculus is synchronous and monadic, and it includes the match/mismatch operator. The choice of a 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, polyadic calculus. For the sake of readability, we often write examples using channels that carry (possibly empty) tuples of names. Moreover, similarly to [8], the matching construct is essential for the coinductive characterization of the reduction barbed congruence[3] introduced in Section 3. More interestingly, the standard π-calculus is enriched 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 $\delta \in \Sigma$, which allow information to flow from a higher to a lower security level through a downgrader. The

[2] We distinguish between names and variables since in a calculus with communication it is natural to handle variables as place-holders for values. In our framework values are simply names and thus the distinction is not needed. However, we could extend the calculus by adding basic values such as integers and boolean; in this case it would be necessary to distinguish the two categories.

[3] The explanation of this point is inherently technical and outside the scope of this work. However, one can refer to Proposition 4.4 (Contextuality of LTS) in [8].

intuition is that only programmers may enable the downgrading of secret information, while external entities cannot synchronize on declassified actions. If a is a σ-level channel, then $a(x{:}T)$ is a σ-level read action which is intended to be observable by users of level higher than or equal to σ. On the contrary, $\mathsf{dec}_\delta\, a(x{:}T)$ 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 $\overline{\mathsf{dec}_\delta\, a}\langle b\rangle$.

It is worth noticing that we could equivalently rely on two distinct sets of names for "declassified" and "non-declassified" high channels. However, instead of partitioning names, we prefer to extend the syntax of the π-calculus with an explicit dec_δ constructor which enhances the clarity of programs and allows programmers to interleave declassified and non declassified actions over the same channel. Moreover, as discussed in Section 1, we use the dec_δ constructor to declassify actions instead of names; indeed, a process like $\overline{n}\langle \mathsf{dec}_\delta\, m\rangle.P$ is not admitted in the $\mathsf{Dec}\,\pi$-calculus.

The input constructs $a(x{:}T).P$ and $\mathsf{dec}_\delta\, a(x{:}T).P$ act as binders 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, which are defined as in the π-calculus, independently of the dec operator. 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. The theory developed in this paper is concerned with *closed* processes, that are processes containing no free occurrences of variables; in Section 7 we discuss how to extend our approach also to open terms.

Types 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 the corresponding level to types, that is $\Lambda(\sigma[\,]) = \sigma = \Lambda(\sigma[T])$.

2.1. Semantics

The operational semantics of the $\mathsf{Dec}\,\pi$-calculus is given in terms of a labelled transition system (LTS) defined over processes. The set of labels, or actions, consists of the usual input, (bound) output, internal τ actions, and additional actions for declassified input and (bound) output:

$$\begin{array}{lrl} \text{Actions} & \alpha ::= & n(m) \mid \mathsf{dec}_\delta\, n(m) \\ & \mid & \overline{n}\langle m\rangle \mid \overline{\mathsf{dec}_\delta\, n}\langle m\rangle \\ & \mid & (\boldsymbol{\nu} m{:}T)\, \overline{n}\langle m\rangle \mid (\boldsymbol{\nu} m{:}T)\, \overline{\mathsf{dec}_\delta\, n}\langle m\rangle \mid \tau \end{array}$$

We write $\mathrm{fn}(\alpha)$ and $\mathrm{bn}(\alpha)$ to denote the set of free and bound names occurring in the action α, where $\mathrm{bn}(\alpha) = \{m\}$ if $\alpha \in \{(\boldsymbol{\nu} m{:}T)\, \overline{n}\langle m\rangle, (\boldsymbol{\nu} m{:}T)\, \overline{\mathsf{dec}_\delta\, n}\langle m\rangle\}$, and $\mathrm{bn}(\alpha) = \emptyset$ otherwise. The LTS is defined by the rules in Tables 2 and 3 where we omitted the symmetric rules for (Sum), (Par), ((Dec)Comm) and ((Dec)Close) in which the role of the left and right components are swapped. As for the downgrading, we allow a declassified action over a channel n to synchronize only with the corresponding declassified co-action over n. For instance, the process $\overline{h}\langle \ell\rangle \mid \mathsf{dec}_\delta\, h(x).P$ has no τ-reductions, whereas $\mathsf{dec}_\delta\, \overline{h}\langle \ell\rangle \mid \mathsf{dec}_\delta\, h(x).P$ reduces to $P\{x := \ell\}$. In other words, we require that both users of a channel (the reader and the writer) agree to downgrade the communication to the same level.

2.2. Type system

The type system of $\mathsf{Dec}\,\pi$ extends the basic type system of the π-calculus (see, e.g., the Base-π typed calculus presented in [13]). 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. Furthermore, it guarantees that an action of a certain level can be downgraded only to a lower level. 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 4; they are based on the type formation rules (Empty Type) and (Channel Type), which prevent a channel of level δ from carrying values of level higher than δ. Type formation rules guarantee

Table 2
Labelled Transition System for Non Declassified Actions

(Out)
$$\frac{}{\overline{n}\langle m\rangle.P \xrightarrow{\overline{n}\langle m\rangle} P}$$

(In)
$$\frac{}{n(x:T).P \xrightarrow{n(m)} P\{x:=m\}}$$

(Mismatch)
$$\frac{}{\textsf{if}\ n=m\ \textsf{then}\ P\ \textsf{else}\ Q \xrightarrow{\tau} Q}\quad n\neq m$$

(Match)
$$\frac{}{\textsf{if}\ n=n\ \textsf{then}\ P\ \textsf{else}\ Q \xrightarrow{\tau} P}$$

(Par)
$$\frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}\quad \mathrm{bn}(\alpha)\cap\mathrm{fn}(Q)=\emptyset$$

(Comm)
$$\frac{P \xrightarrow{\overline{n}\langle m\rangle} P' \quad Q \xrightarrow{n(m)} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$$

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

(Rep-Act)
$$\frac{P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P'|!P}$$

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

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

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.

The following subject-reduction property expresses the consistency between the operational semantics and the typing rules (see [13]).

Proposition 1 (Subjectreduction)**.** *Let P be a process and Γ be a type environment such that $\Gamma \vdash P$. The following properties hold.*

- *If* $P \xrightarrow{\tau} P'$ *then* $\Gamma \vdash P'$.
- *If* $P \xrightarrow{n(m)} P'$ *then* $\Gamma \vdash n:\delta[T]$; *moreover if* $\Gamma \vdash m:T$ *then* $\Gamma \vdash P'$.
- *If* $P \xrightarrow{\mathsf{dec}_\delta\, n(m)} P'$ *then* $\Gamma \vdash n:\delta_1[T]$ *and* $\delta \prec \delta_1$; *moreover if* $\Gamma \vdash m:T$ *then* $\Gamma \vdash P'$.
- *If* $P \xrightarrow{\overline{n}\langle m\rangle} P'$ *then* $\Gamma \vdash n:\delta[T]$, $\Gamma \vdash m:T$ *and* $\Gamma \vdash P'$.
- *If* $P \xrightarrow{\overline{\mathsf{dec}_\delta\, n}\langle m\rangle} P'$ *then* $\Gamma \vdash n:\delta_1[T], \delta \prec \delta_1, \Gamma \vdash m:T$ *and* $\Gamma \vdash P'$.
- *If* $P \xrightarrow{(\nu m:T)\,\overline{n}\langle m\rangle} P'$ *then* $\Gamma \vdash n:\delta[T]$ *and* $\Gamma, m:T \vdash P'$.
- *If* $P \xrightarrow{(\nu m:T)\,\overline{\mathsf{dec}_\delta\, n}\langle m\rangle} P'$ *then* $\Gamma \vdash n:\delta_1[T]$, $\delta \prec \delta_1$ *and* $\Gamma, m:T \vdash P'$.

Table 3
Labelled transition system for declassified actions

(Dec Out)
$$\frac{}{\overline{\mathsf{dec}_\delta}\, n\langle m\rangle.P \xrightarrow{\overline{\mathsf{dec}_\delta}\, n\langle m\rangle} P}$$

(Dec In)
$$\frac{}{\mathsf{dec}_\delta\, n(x{:}T).P \xrightarrow{\mathsf{dec}_\delta\, n(m)} P\{x := m\}}$$

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

(Dec Comm)
$$\frac{P \xrightarrow{\overline{\mathsf{dec}_\delta}\, n\langle m\rangle} P' \quad Q \xrightarrow{\mathsf{dec}_\delta\, n(m)} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$$

(Dec Close)
$$\frac{P \xrightarrow{(\nu m:T)\,\overline{\mathsf{dec}_\delta}\, n\langle m\rangle} P' \quad Q \xrightarrow{\mathsf{dec}_\delta\, n(m)} Q'}{P|Q \xrightarrow{\tau} (\boldsymbol{\nu} m{:}T)(P'|Q')} \quad m \notin \mathrm{fn}(Q)$$

Table 4
Type system

(Empty Type)
$$\frac{}{\vdash \delta[\,]}$$

(Channel Type)
$$\frac{\vdash T}{\vdash \delta[T]} \quad \Lambda(T) \preceq \delta$$

(Empty)
$$\frac{}{\emptyset \vdash \diamond}$$

(Env a)
$$\frac{\Gamma \vdash \diamond \quad \vdash T}{\Gamma, a : T \vdash \diamond} \quad a \notin Dom\ (\Gamma)$$

(Project)
$$\frac{\Gamma, a : T \vdash \diamond}{\Gamma, a : T \vdash a : T}$$

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

(Input)
$$\frac{\Gamma \vdash a : \delta[T] \quad \Gamma, x{:}T \vdash P}{\Gamma \vdash a(x{:}T).P}$$

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

(Para)
$$\frac{\Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash P|Q}$$

(Res)
$$\frac{\Gamma, n : T \vdash P}{\Gamma \vdash (\boldsymbol{\nu} n : T)P}$$

(Repl)
$$\frac{\Gamma \vdash P}{\Gamma \vdash !P}$$

(Dead)
$$\frac{\Gamma \vdash \diamond}{\Gamma \vdash \mathbf{0}}$$

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

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

Proof. By induction on the depth of the derivation of $P \xrightarrow{\alpha} P'$ and a case analysis on α. □

3. Observation equivalences for the $\mathsf{Dec}\,\pi$-calculus

In this section, we introduce the notion of σ-level observation equivalence and we develop an equational theory for the $\mathsf{Dec}\,\pi$-calculus which is parametric on the security level (i.e., the observation power) of external observers. The declassification construct provided by the $\mathsf{Dec}\,\pi$-calculus has no significant impact on the equational theory developed in this section. Actually, our observation equivalences are concerned only with the level of channels, independently of downgrading. On the contrary, declassification is the core of the controlled information release property studied in the next section.

Our equivalences are reminiscent of the typed behavioral equivalences for the π-calculus [8,3,13], that 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 [8, 3] since we do not consider subtyping nor linearity/affinity.

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 well typed processes in Γ and they are related by $\mathcal{R}$.

To define our σ-level observation equivalences, 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 $\Gamma \vDash P\,\mathcal{R}\,Q$ 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 $\Gamma \vDash P \downarrow_n^\sigma$ 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 $\Gamma \vDash P' \downarrow_n^\sigma$. Note for instance that $\overline{\mathsf{dec}_\delta\, n}\langle m\rangle.P \not\downarrow_n^\sigma$ independently of the typing of n. However, when n is a low channel, the low action $\overline{\mathsf{dec}_\delta\, n}\langle m\rangle$ can be indirectly observed thanks to the fact that low-contexts can synchronize on low channels, even if declassified.
A type-indexed relation $\mathcal{R}$ over processes is *σ-barb preserving* if $\Gamma \vDash P\,\mathcal{R}\,Q$ 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. Note that variables and names that are free in P may become bound in $C[\cdot]$; thus we do not identify contexts up to renaming of bound variables and names. 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 environments 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 4 is extended with the following rule:

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

We are interested in σ-contexts that capture the idea of σ-level observers, which cannot interact with the observed system through actions over channels of level higher than σ, even if downgraded. Therefore we let a σ-context be an evaluation context which may interact with the process filling the hole just through channels of level at most σ.

Definition 2 (*σ-context*). Let $\sigma \in \Sigma$ and Γ, Γ' be two type environments. A (Γ'/Γ)-context $C[\cdot_\Gamma]$ is a σ-context if it is generated by the following grammar

Table 5
Typed LTS

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

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

(Weak)
$$\frac{\Gamma, m : T \rhd P \xrightarrow{n(m)}_{\delta} \Gamma' \rhd P'}{\Gamma \rhd P \xrightarrow{(\nu m:T)\, n(m)}_{\delta} \Gamma' \rhd P'}$$

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

(Red)
$$\frac{P \xrightarrow{\tau} P'}{\Gamma \rhd P \xrightarrow{\tau}_{\delta} \Gamma \rhd P'}$$

(Rep-Act)
$$\frac{\Gamma \rhd P \xrightarrow{\alpha}_{\delta} \Gamma' \rhd P'}{\Gamma \rhd !P \xrightarrow{\alpha}_{\delta} \Gamma' \rhd P' | !P}$$

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

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

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

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

Example 3. Consider the typing $h : \top[\bot[\,]]$, $\ell : \bot[\,]$ and $\sigma \prec \top$. The contexts $(\boldsymbol{\nu} h)(\overline{h}\langle\ell\rangle \,|\, [\cdot_\Gamma])$ and $\overline{\mathsf{dec}_\sigma h}\langle\ell\rangle \,|\, [\cdot_\Gamma]$ are not σ-contexts since the processes $\overline{h}\langle\ell\rangle$ and $\overline{\mathsf{dec}_\sigma h}\langle\ell\rangle$ in parallel with the hole have a free occurrence of the high name h. On the other hand, both $(\boldsymbol{\nu} h)(\overline{h}\langle\ell\rangle | h(x).\overline{x}\langle\rangle) | [\cdot_\Gamma]$ and $(\boldsymbol{\nu} h)(\overline{\mathsf{dec}_\sigma h}\langle\ell\rangle | \mathsf{dec}_\sigma h(x).\overline{x}\langle\rangle) | [\cdot_\Gamma]$ are σ-contexts. As another example of σ-context, let be $n : \sigma[\,]$ and $\delta \prec \sigma$, then $\overline{\mathsf{dec}_\delta\, n}\langle\rangle | [\cdot_\Gamma]$ is a σ-context since n is an observable channel, even without the downgrading. This example illustrates that the notion of observation deals only with the level of channels, independently of downgrading.

We say that a type-indexed relation $\mathcal{R}$ over processes is *σ-contextual* if $\Gamma \vDash P \;\mathcal{R}\; Q \;\; \Gamma' \vDash C[P] \;\mathcal{R}\; C[Q]$ for all (Γ'/Γ)-σ-contexts $C[\cdot_\Gamma]$.

Definition 4 (*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 5. *Let* $\sigma \in \Sigma, \Gamma$ *be a type environment and* P, Q *be two 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* $\sigma \in \Sigma$*.*

Table 6
Typed LTS

(Dec Out)

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

(Dec Open)

$$\frac{\Gamma, m{:}T \rhd P \xrightarrow{\overline{\mathsf{dec}_{\delta_1}n}\langle m\rangle}_{\delta} \Gamma' \rhd P'}{\Gamma \rhd (\nu m{:}T)P \xrightarrow{(\nu m:T)\,\overline{\mathsf{dec}_{\delta_1}n}\langle m\rangle}_{\delta} \Gamma' \rhd P'} \quad m \neq n$$

(Dec In)

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

(Dec Weak)

$$\frac{\Gamma, m : T \rhd P \xrightarrow{\mathsf{dec}_{\delta_1}n(m)}_{\delta} \Gamma' \rhd P'}{\Gamma \rhd P \xrightarrow{(\nu m:T)\,\mathsf{dec}_{\delta_1}n(m)}_{\delta} \Gamma' \rhd P'}$$

3.1. A bisimulation-based proof technique

In this section, we develop a proof technique for the σ-reduction barbed congruence $\cong_\sigma$. More precisely, following [14,2,8], we define an LTS of *typed actions* (a typed LTS) over configurations, that are pairs $\Gamma \rhd P$ where Γ is a type environment and P is a process such that $\Gamma \vdash P$.[4]

As in [2], actions are parameterized over security levels and take the form

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

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. Moreover, given a security level $\sigma \in \Sigma$, whenever $\Gamma \rhd P \xrightarrow{\alpha}_{\delta} \Gamma' \rhd P'$ with $\sigma \prec \delta$ (resp. $\delta \preceq \sigma$) we say that $\Gamma \rhd P$ has performed a σ-*high* (resp. σ-*low*) level action.

The rules of the typed LTS are obtained from those in Tables 2 and 3 by taking into account the type environment Γ, which records the security levels of the channels used by the process. Differently from [2], our typed actions are built around just a single type environment Γ constraining the observed process P. This differs from [2] 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 Tables 5 and 6; note that there are two additional input actions $(\nu m{:}T)\, n(m)$ and $(\nu m{:}T)\, \mathsf{dec}_\delta\, n(m)$, occurring when the process receives a new name m generated by the environment. Indeed, in order to take the type environment of the computing context into account, one must distinguish the case where a process inputs a name already known by the environment (for more details the reader is refered to [8]).

[4] $\Gamma \rhd P$ and $\Gamma \vdash P$ are indeed equivalent. Nevertheless, we prefer to keep the notation $\Gamma \rhd P$ of [8] stressing the fact that a typed action depends on both the process and its context.

As discussed above, σ-contexts cannot perform downgraded σ-high actions to observe sensitive information. Therefore, if a σ-high action $\overline{h}\langle n\rangle$ is declassified to an observable level δ, then by rule (Dec Out) the resulting action $\overline{\mathsf{dec}_\delta\, h\langle n\rangle}$ is still a σ-high action. In the next section we show that in our theory of controlled information release the actual impact of downgrading is on the admissible information flows.

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

Proposition 6. *Let P be a process and Γ be a type environment such that $\Gamma \triangleright P$ is a configuration. Then*

- $\Gamma \triangleright P \xrightarrow{\tau}_\delta \Gamma \triangleright Q$ *iff* $P \xrightarrow{\tau} Q$.
- $\Gamma \triangleright P \xrightarrow{\alpha}_\delta \Gamma \triangleright P'$ *with* $\alpha \in \{\overline{n}\langle m\rangle, n(m)\}$ *iff* $P \xrightarrow{\alpha} P'$, $m \in Dom\ (\Gamma)$ *and* $\Lambda(\Gamma(n)) \preceq \delta$.
- $\Gamma \triangleright P \xrightarrow{\alpha}_\delta \Gamma \triangleright P'$ *with* $\alpha \in \{\overline{\mathsf{dec}_{\delta_1} n\langle m\rangle}, \mathsf{dec}_{\delta_1} n(m)\}$ *iff* $P \xrightarrow{\alpha} P'$, $m \in Dom\ (\Gamma)$ *and* $\delta_1 \prec \Lambda(\Gamma(n)) \preceq \delta$.
- $\Gamma \triangleright P \xrightarrow{(\nu m:T)\,\overline{n}\langle m\rangle}_\delta \Gamma' \triangleright P'$ *iff* $P \xrightarrow{(\nu m:T)\,\overline{n}\langle m\rangle} P'$ *with* $\Gamma(n) = \delta_1[T]$ *and* $\delta_1 \preceq \delta$.
- $\Gamma \triangleright P \xrightarrow{(\nu m:T)\,\overline{\mathsf{dec}_{\delta_2} n\langle m\rangle}}_\delta \Gamma' \triangleright P'$ *iff* $P \xrightarrow{(\nu m:T)\,\overline{\mathsf{dec}_{\delta_2} n\langle m\rangle}} P'$ *with* $\Gamma(n) = \delta_1[T]$ *and* $\delta_2 \prec \delta_1 \preceq \delta$.
- $\Gamma \triangleright P \xrightarrow{(\nu m:T)\, n(m)}_\delta \Gamma' \triangleright P'$ *iff* $P \xrightarrow{n(m)} P'$ *with* $\Gamma(n) = \delta_1[T], \delta_1 \preceq \delta$ *and* $m \notin Dom\ (\Gamma)$.
- $\Gamma \triangleright P \xrightarrow{(\nu m:T)\,\mathsf{dec}_{\delta_2} n(m)}_\delta \Gamma' \triangleright P'$ *iff* $P \xrightarrow{\mathsf{dec}_{\delta_2} n(m)} P'$ *with* $\Gamma(n) = \delta_1[T], \delta_2 \prec \delta_1 \preceq \delta$ *and* $m \notin Dom\ (\Gamma)$.

The next proposition shows how the type environment is modified after the execution of an action.

Proposition 7. *Let P be a process and Γ be a type environment such that $\Gamma \triangleright P$ is a configuration. Whenever* $\Gamma \triangleright P \xrightarrow{\alpha}_\delta \Gamma' \triangleright P'$:

- *if* $\alpha \in \{\tau,\ \overline{n}\langle m\rangle,\ n(m),\ \overline{\mathsf{dec}_\delta\, n\langle m\rangle},\ \mathsf{dec}_\delta\, n(m)\}$ *then* $\Gamma' = \Gamma$.
- *if* $\alpha \in \{(\nu m{:}T)\,\overline{n}\langle m\rangle, (\nu m{:}T)\, n(m), (\nu m{:}T)\,\overline{\mathsf{dec}_\delta\, n\langle m\rangle}, (\nu m{:}T)\,\mathsf{dec}_\delta\, n(m)\}$
then $\Gamma' = \Gamma, m{:}T$.

Proof. The proof follows by induction on the depth of the derivation of $\Gamma \triangleright P \xrightarrow{\alpha}_\delta \Gamma' \triangleright P'$. □

Relying on the typed LTS, we now introduce the *bisimilarity on σ-actions* which provides a coinductive characterization of the σ-reduction barbed congruence $\cong_\sigma$.

With an abuse of notation, we write $\Longrightarrow$ for the reflexive and transitive closure of $\xrightarrow{\tau}_\delta$ (regardless of the levels of the τ-transitions). 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.

Definition 8 (*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 $\Gamma \triangleright Q \overset{\hat{\alpha}}{\Longrightarrow}_\sigma \Gamma' \triangleright Q'$ and $(\Gamma' \triangleright Q') \approx_\sigma (\Gamma' \triangleright P')$.

According to the intuition about σ-level observers, we have that whenever $(\Gamma \triangleright P) \approx_\sigma (\Gamma \triangleright Q)$, only the σ-low actions of P are matched by corresponding actions of Q (and viceversa), whereas nothing is required about the σ-high actions, even if they are declassified to an observable level lower than or equal to σ. Hereafter, for a given relation $\mathcal{R}$ over configurations, we write

$$\Gamma \models P\, \mathcal{R}\, Q \quad \text{whenever} \quad (\Gamma \triangleright P)\, \mathcal{R}\, (\Gamma \triangleright Q).$$

The next theorem states that the bisimilarity on σ-actions coincides with the σ-reduction barbed congruence.

Theorem 9. *Let $\sigma \in \Sigma, \Gamma$ be a type environment and P, Q be two processes such that $\Gamma \vdash P, Q$. Then $\Gamma \models P \cong_\sigma Q$ iff $\Gamma \models P \approx_\sigma Q$.*

Proof. See Appendix A. □

4. Controlled information release

In this section, we introduce a notion of controlled information release for processes of the typed $\mathsf{Dec}\,\pi$-calculus which uses the σ-reduction barbed congruence $\cong_\sigma$ as observation equivalence. This property, called *σ-controlled release* (written $\mathcal{CR}(\cong_\sigma)$) is parametric with respect to the security level σ and it is inspired by the persistent security property *DP_BNDC* defined in [7] for CCS processes. In fact, as *DP_BNDC*, $\mathcal{CR}(\cong_\sigma)$ requires that no uncontrolled information flow occurs even in the presence of *active* malicious processes, e.g., Trojan Horse programs, that run at the classified (higher than σ) level. Moreover, $\mathcal{CR}(\cong_\sigma)$ is persistent in the sense that whenever a process is secure, then each state reachable from it is also secure.

The σ-controlled release property is built upon a declassification model that allows programmers to intentionally release sensitive information in a controlled way. In particular, we assume that only internal, trusted, system components can downgrade sensitive information, while external attackers cannot enable the declassification. This justifies the following formalization of attackers as *σ-high level source* processes. We use the following notations.

- We say that a configuration $\Gamma' \rhd P'$ is *reachable* from a configuration $\Gamma \rhd P$, written $\Gamma \rhd P \rightsquigarrow \Gamma' \rhd P'$, if there exist $n \geq 0, \alpha_1, \ldots, \alpha_n$ and $\sigma_1, \ldots, \sigma_n$ such that $\Gamma \rhd P \xrightarrow{\alpha_1}_{\sigma_1} \xrightarrow{\alpha_2}_{\sigma_2} \cdots \xrightarrow{\alpha_n}_{\sigma_n} \Gamma' \rhd P'$.
(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 $\Gamma \rhd P \not\xrightarrow{\alpha}_\delta$ (i.e., $\Gamma \rhd P$ does not perform any action) or if $\Gamma \rhd P \xrightarrow{\alpha}_\delta \Gamma' \rhd P'$ then $\alpha \in \{\overline{n}\langle m\rangle, n(m), (\boldsymbol{\nu} m{:}T)\,\overline{n}\langle m\rangle, (\boldsymbol{\nu} m{:}T)\,n(m)\}$ with $\sigma \prec \delta$ and P' is a σ-high level source in Γ'. In other words, P can only perform non declassified, σ-high level actions. 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 $\sigma \in \Sigma$, whenever $\Gamma \rhd P$ performs a σ-high level action $\Gamma \rhd P \xrightarrow{\alpha}_\delta \Gamma' \rhd P'$ with $\sigma \prec \delta$ we write $\Gamma \rhd P \xrightarrow{\alpha}{}^\sigma \Gamma' \rhd P'$ (with a superscript σ). We define $\overset{\hat{\alpha}}{\Longrightarrow}{}^\sigma$ accordingly.

A process P in a type environment Γ satisfies the property $\mathcal{CR}(\cong_\sigma)$ if for every configuration $\Gamma' \rhd P'$ reachable from $\Gamma \rhd P$ and for every σ-high level source H, a σ-context cannot distinguish, in the sense of $\cong_\sigma$, $\Gamma' \rhd P'$ from $\Gamma' \rhd P'|H$. The formal definition of $\mathcal{CR}(\cong_\sigma)$ is as follows.

Definition 10 (*σ-Controlled release*)**.** 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-controlled release property* in Γ, written $\Gamma \rhd P \in \mathcal{CR}(\cong_\sigma)$, if

$$\forall\, \Gamma' \rhd P' \text{ such that } \Gamma \rhd P \rightsquigarrow \Gamma' \rhd P' \text{ and } \forall\, H \in \mathcal{H}^\sigma_{\Gamma'},$$
$$\Gamma' \models P' \cong_\sigma P'|H.$$

Example 11. Let Γ be the type environment $h, k : \top[\,]$, $\ell : \bot[\,]$ and $\sigma = \bot$. In the following, when channels do not carry values, we simply write n and $\overline{n}$ instead of $n()$ and $\overline{n}\langle\rangle$.

- Let us first consider the following simple insecure process: $P_1 = \overline{h}.\ell | h$. To show that $\Gamma \rhd P_1 \notin \mathcal{CR}(\cong_\sigma)$ it is sufficient to consider the configuration $\Gamma \rhd P_1'$ with $P_1' = \overline{h}.\ell$ that is reachable from $\Gamma \rhd P_1$ after performing the input action h. The process P_1' is clearly insecure in the type environment Γ since the low level, observable, action ℓ directly depends on the high level output $\overline{h}$. Indeed, by choosing $H = h$ one can easily observe that $\Gamma \models P_1' \ncong_\sigma P_1'|H$.
- The previous example shows that the process $\Gamma \rhd \overline{h}.\ell$ is not secure. However, $\Gamma \rhd \overline{\mathsf{dec}_\sigma h}.\ell \in \mathcal{CR}(\cong_\sigma)$, i.e., $\Gamma \rhd \overline{\mathsf{dec}_\sigma h}.\ell$ is secure; indeed it can be proved that $\Gamma \models \overline{\mathsf{dec}_\sigma h}.\ell \cong_\sigma \mathbf{0}$. This captures the intuition that declassification can be enabled only by internal system components.

- Let $P = \overline{h}.\mathsf{dec}_\sigma k.\ell$. Since the action $\mathsf{dec}_\sigma k$ in P will never synchronize with an external context, it can be proved that P is secure, i.e., $\Gamma \triangleright P \in \mathcal{CR}(\cong_\sigma)$.
- As a further example, $P = \overline{\mathsf{dec}_\sigma h}|\mathsf{dec}_\sigma h.\ell$ can be easily shown to be a secure process such that $\Gamma \models P \cong_\sigma \ell$. On the other hand, $P_1 = \overline{k}.(\overline{\mathsf{dec}_\sigma h}|\mathsf{dec}_\sigma h.\ell)$ is not secure since the observable action ℓ depends on the execution of the high action $\overline{k}$.
- Finally, consider the following process which uses the high level channel h alternatively as a secret and a declassified channel:
  $P = \overline{\mathsf{dec}_\sigma h}.\, h.\, \overline{\mathsf{dec}_\sigma h}|\mathsf{dec}_\sigma h.\, \overline{\ell}.\, \overline{h}.\, \mathsf{dec}_\sigma h.$
  It can be proved that P is secure.

Further examples are going to be discussed in Section 5.

4.1. Controlled release through name restriction

Interestingly, the definition of σ-controlled release can be equivalently expressed in terms of bisimilarity on $\top$-actions over well-typed processes whose σ-high level names are restricted.

Definition 12. Let $\sigma \in \Sigma$, P be a process and Γ be a type environment such that $\Gamma \vdash P$. We denote by $(\boldsymbol{v}^\sigma)P$ the process $(\boldsymbol{v}m_1{:}T_1),\ldots,(\boldsymbol{v}m_k{:}T_k)P$ where $\{m_1,\ldots,m_k\} = \mathrm{fn}(P)$ and $\Gamma(m_i) = T_i$ with $\Lambda(T_i) \succ \sigma$.

The following proposition describes some useful behavioral properties of $(\boldsymbol{v}^\sigma)P$.

Proposition 13. *Let $\sigma \in \Sigma$, P be a process and Γ a type environment such that $\Gamma \vdash P$, then*

(1) $\Gamma \triangleright (\boldsymbol{v}^\sigma)P \overset{\alpha}{\not\longrightarrow}{}^\sigma$ *, that is $\Gamma \triangleright (\boldsymbol{v}^\sigma)P$ can only perform σ-low transitions* $\xrightarrow{\alpha}_\sigma$
(2) *for all $\Gamma' \triangleright Q$ such that $\Gamma \triangleright (\boldsymbol{v}^\sigma)P \rightsquigarrow \Gamma' \triangleright Q$, Q is of the form $(\boldsymbol{v}^\sigma)P'$ for some process P', and thus* $\Gamma' \triangleright (\boldsymbol{v}^\sigma)P' \overset{\alpha}{\not\longrightarrow}{}^\sigma$
(3) $\Gamma \triangleright (\boldsymbol{v}^\sigma)P \xrightarrow{\alpha}_\sigma \Gamma' \triangleright (\boldsymbol{v}^\sigma)P'$ *iff* $\Gamma \triangleright P \xrightarrow{\alpha}_\sigma \Gamma' \triangleright P'$.

Proof. The proof follows by induction on the length of the derivation $\Gamma \triangleright (\boldsymbol{v}^\sigma)P \rightsquigarrow \Gamma' \triangleright Q$ and the definition of the typed LTS given in Table 5. □

The following proposition shows that the restriction of σ-high free names does not affect the bisimilarity $\approx_\sigma$.

Proposition 14. *Let $\sigma \in \Sigma$, P, Q be two processes such that $\Gamma \vdash P, Q$. If $\Gamma \models (\boldsymbol{v}^\sigma)P \approx_\sigma (\boldsymbol{v}^\sigma)Q$ then $\Gamma \models P \approx_\sigma Q$.*

Proof. It is sufficient to prove that

$$\mathcal{S} = \{(\Gamma \triangleright P,\ \Gamma \triangleright Q) \mid \Gamma \models (\boldsymbol{v}^\sigma)P \approx_\sigma (\boldsymbol{v}^\sigma)Q\}$$

is a bisimulation on σ-actions, that is $\mathcal{S} \subseteq \approx_\sigma$, which is not difficult using Proposition 13.

The next proposition establishes a precise relation between the relations $\approx_\sigma$ and $\approx_\top$. □

Proposition 15. *Let $\sigma \in \Sigma$, P and Q be two processes and Γ be a type environment such that $\Gamma \vdash P, Q$.$\Gamma \models P \approx_\sigma Q$ iff $\Gamma \models (\boldsymbol{v}^\sigma)P \approx_\top (\boldsymbol{v}^\sigma)Q$.*

Proof. ($\Rightarrow$) By Theorem 9 we know that $\approx_\sigma$ is a congruence with respect to σ-contexts. Hence, from $\Gamma \models P \approx_\sigma Q$ we have $\Gamma \models (\boldsymbol{v}^\sigma)P \approx_\sigma (\boldsymbol{v}^\sigma)Q$, and we conclude observing that, by Proposition 13, $(\boldsymbol{v}^\sigma)P, (\boldsymbol{v}^\sigma)Q$ do never perform actions of level higher than σ.

($\Leftarrow$) From $\Gamma \models (\boldsymbol{v}^\sigma)P \approx_\top (\boldsymbol{v}^\sigma)Q$ we have $\Gamma \models (\boldsymbol{v}^\sigma)P \approx_\sigma (\boldsymbol{v}^\sigma)Q$ since, by Proposition 5 and Theorem 9, $\approx_\top \subseteq \approx_\sigma$, and we conclude by Proposition 14. □

The following corollary provides a characterization of $\mathcal{CR}(\cong_\sigma)$ in terms of bisimilarity on $\top$-actions and name restriction. Such a characterization of σ-controlled release is reminiscent of the noninterference *BNDC* property defined by Focardi and Gorrieri in [10] for CCS processes.

Corollary 16. *Let $\sigma \in \Sigma, P$ be a process and Γ be a type environment such that $\Gamma \vdash P$. It holds that $\Gamma \triangleright P \in \mathcal{CR}(\cong_\sigma)$ iff 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 $\Gamma' \models (\boldsymbol{\nu}^\sigma)P' \approx_\top (\boldsymbol{\nu}^\sigma)(P'|H)$.*

4.2. Controlled release through unwinding

Building on the ideas developed by Bossi et al. [11,7] for CCS processes, possibly dealing with downgrading, we provide a characterization of $\mathcal{CR}(\cong_\sigma)$ in terms of unwinding conditions. Intuitively, an unwinding condition specifies local constraints on the transitions of the system which imply some global security property. More precisely, our unwinding condition ensures that no σ-high action leading to a configuration C is observable by a σ-low context, as there always exists a configuration C', σ-equivalent to C, that the system may reach through internal transitions. In order to allow intentional information flows, the unwinding condition deals only with non declassified σ-high actions, requiring that only these actions are masked by internal transitions.

Definition 17 (*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 $\Gamma' \models P_2 \cong_\sigma P_3$;
- 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 $\Gamma' \models P_3 \cong_\sigma (\boldsymbol{\nu} m{:}T)P_2$.

This unwinding-based characterization captures the idea of security against *passive attacks* which try to infer information about the classified behavior of the system just by observing its σ-level behavior.

The following proposition states that both properties $\mathcal{CR}(\cong_\sigma)$ and $\mathcal{W}(\cong_\sigma)$ are persistent.

Proposition 18 (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 $\Gamma \triangleright P \in \mathcal{CR}(\cong_\sigma)$ then $\Gamma' \triangleright P' \in \mathcal{CR}(\cong_\sigma)$.*
- *if $\Gamma \triangleright P \in \mathcal{W}(\cong_\sigma)$ then $\Gamma' \triangleright P' \in \mathcal{W}(\cong_\sigma)$.*

Proof. Immediate. □

In Theorem 21, we prove that the properties $\mathcal{CR}(\cong_\sigma)$ and $\mathcal{W}(\cong_\sigma)$ are equivalent. The proof relies on the following notion of bisimulation up to σ-contexts and up to $\approx_\sigma$.

Definition 19 (*Bisimulation up to σ-contexts and up to $\approx_\sigma$*)**.** A symmetric relation $\mathcal{R}$ over configurations is a bisimulation up to σ-contexts and up to $\approx_\sigma$ if $(\Gamma \triangleright P)\ \mathcal{R}\ (\Gamma \triangleright Q)$ implies:

- if $\Gamma \triangleright P \xrightarrow{\alpha}_\sigma \Gamma' \triangleright P'$, then there exists Q' such that $\Gamma \triangleright Q \overset{\hat{\alpha}}{\Longrightarrow}_\sigma \Gamma' \triangleright Q'$ and there are two processes P'', Q'', a type environment Γ'' and a σ-context $C[\cdot_{\Gamma''}]$ with $\Gamma' \vdash C[\cdot_{\Gamma''}]$, such that $\Gamma' \models P' \approx_\sigma C[P'']$, $\Gamma' \models Q' \approx_\sigma C[Q'']$, and $(\Gamma'' \triangleright P'')\ \mathcal{R}\ (\Gamma'' \triangleright Q'')$.

Proposition 20. *If $\mathcal{R}$ is a bisimulation up to σ-contexts and up to $\approx_\sigma$, then $\mathcal{R} \subseteq \approx_\sigma$.*

Proof. See Appendix B. □

Theorem 21. *Let $\sigma \in \Sigma, P$ be a process and Γ be a type environment such that $\Gamma \vdash P. \Gamma \triangleright P \in \mathcal{CR}(\cong_\sigma)$ iff $\Gamma \triangleright P \in \mathcal{W}(\cong_\sigma)$.*

Proof. ($\Leftarrow$) Given $\Gamma \triangleright P \in \mathcal{W}(\cong_\sigma)$, we prove that $\forall \Gamma' \triangleright P'$ such that $\Gamma \triangleright P \rightsquigarrow \Gamma' \triangleright P'$ and $\forall H \in \mathcal{H}^\sigma_{\Gamma'}$, it holds $\Gamma' \models P' \cong_\sigma P'|H$. Let $\mathcal{S}$ be the symmetric closure of

$$\{(\Gamma \triangleright P,\ \Gamma \triangleright P|H) \mid \Gamma \triangleright P \in \mathcal{W}(\cong_\sigma) \text{ and } H \in \mathcal{H}^\sigma_\Gamma\}$$

We prove that $\mathcal{S}$ is a bisimulation on σ-actions up to σ-contexts and up to $\approx_\sigma$. The thesis follows by Proposition 20, the fact that $\approx_\sigma$ is the largest bisimulation on σ-actions, and Theorem 9.

First, let be $\Gamma \triangleright P \xrightarrow{\alpha}_\sigma \Gamma' \triangleright P'$, hence $\Gamma \triangleright P|H \xrightarrow{\alpha}_\sigma \Gamma' \triangleright P'|H$ by rule (Par) of LTS. Since $\Gamma \triangleright P \rightsquigarrow \Gamma' \triangleright P'$, by persistence we have $\Gamma' \triangleright P' \in \mathcal{W}(\cong_\sigma)$, then we conclude since $\Gamma' \models P' \approx_\sigma P'$, $\Gamma' \models P'|H \approx_\sigma P'|H$ and $(\Gamma' \triangleright C[P'])\ \mathcal{S}\ (\Gamma' \triangleright C[P'|H])$ with $C[\cdot_{\Gamma'}] = [\cdot_{\Gamma'}]$, which is a σ-context.

Assume now that $\Gamma \triangleright P|H \xrightarrow{\alpha}_\sigma \Gamma' \triangleright Q$, we distinguish the following cases:

- this comes by (Par) from $\Gamma \triangleright H \xrightarrow{\alpha}_\sigma \Gamma' \triangleright H'$ and $Q = P|H'$. In this case, since $H \in \mathcal{H}^\sigma_\Gamma$, we have that $\alpha = \tau$, $\Gamma' = \Gamma$ and $H' \in \mathcal{H}^\sigma_{\Gamma'}$, hence $(\Gamma \triangleright P,\ \Gamma \triangleright P|H') \in \mathcal{S}$, and we conclude by choosing the σ-context $[\cdot_\Gamma]$.
- this comes by (Par) from $\Gamma \triangleright P \xrightarrow{\alpha}_\sigma \Gamma' \triangleright P'$ and $Q = P'|H$. Since $\Gamma \triangleright P \rightsquigarrow \Gamma' \triangleright P'$, by persistence we have $\Gamma' \triangleright P' \in \mathcal{W}(\cong_\sigma)$, then $(\Gamma' \triangleright P',\ \Gamma' \triangleright P'|H) \in \mathcal{S}$ and we conclude by choosing the σ-context $[\cdot_{\Gamma'}]$.
- this comes by (Comm) from $\Gamma \triangleright P \xrightarrow{\overline{n}\langle m\rangle}_\delta \Gamma \triangleright P'$, $\Gamma \triangleright H \xrightarrow{n(m)}_\delta \Gamma \triangleright H'$ and $Q = P'|H'$. Since $H \in \mathcal{H}^\sigma_\Gamma$, we have that $\sigma \prec \delta$ and $H' \in \mathcal{H}^\sigma_\Gamma$. From the hypothesis $\Gamma \triangleright P \in \mathcal{W}(\cong_\sigma)$ we have that there exists P'' such that $\Gamma \triangleright P \Longrightarrow \Gamma \triangleright P''$ and $\Gamma \models P' \cong_\sigma P''$, then also $\Gamma \models P' \approx_\sigma P''$. Now, by persistence we have $\Gamma \triangleright P' \in \mathcal{W}(\cong_\sigma)$, then $(\Gamma \triangleright P',\ \Gamma \triangleright P'|H') \in \mathcal{S}$, hence $\Gamma \models P'' \approx_\sigma P'$, $\Gamma \models P'|H' \approx_\sigma P'|H'$ and we conclude by choosing the σ-context $[\cdot_\Gamma]$.
- this comes by (Comm) from $\Gamma \triangleright P \xrightarrow{n(m)}_\delta \Gamma \triangleright P'$, $\Gamma \triangleright H \xrightarrow{\overline{n}\langle m\rangle}_\delta \Gamma \triangleright H'$ and $Q = P'|H'$. This case is analogous to the previous one.
- this comes by (Dec Comm) from $P \xrightarrow{\mathsf{dec}_\delta\, n(m)} P'$, and $H \xrightarrow{\overline{\mathsf{dec}_\delta\, n}\langle m\rangle} H'$ or viceversa. This case is vacuous since $H \in \mathcal{H}^\sigma_\Gamma$, i.e., H cannot interact with P through downgraded actions.
- this comes by (Close) from $P \xrightarrow{(\nu m{:}T)\,\overline{n}\langle m\rangle} P'$, $H \xrightarrow{n(m)} H'$ and $Q = (\boldsymbol{\nu} m{:}T)(P'|H')$. By Propositions 6 and 7, $\Gamma \triangleright P \xrightarrow{(\nu m{:}T)\,\overline{n}\langle m\rangle}_\delta \Gamma, m{:}T \triangleright P'$, and $\Gamma \triangleright H \xrightarrow{(\nu m{:}T)\,n(m)}_\delta \Gamma, m{:}T \triangleright H'$. Since $H \in \mathcal{H}^\sigma_\Gamma$, we have $\sigma \prec \delta$. From the hypothesis that $\Gamma \triangleright P \in \mathcal{W}(\cong_\sigma)$ we have that there exists P'' such that $\Gamma \triangleright P \Longrightarrow \Gamma \triangleright P''$ and $\Gamma \models P'' \cong_\sigma (\boldsymbol{\nu} m{:}T)P'$, then also it holds that $\Gamma \models P'' \approx_\sigma (\boldsymbol{\nu} m{:}T)P'$. From $\Gamma \triangleright P \rightsquigarrow \Gamma, m{:}T \triangleright P'$, by persistence we have $\Gamma, m{:}T \triangleright P' \in \mathcal{W}(\cong_\sigma)$, then $(\Gamma, m{:}T \triangleright P')\ \mathcal{S}\ (\Gamma, m{:}T \triangleright P'|H')$. Summing up, let $C[\cdot_\Gamma]$ be the σ-context $(\boldsymbol{\nu} m{:}T)[\cdot_\Gamma]$, then we have that $\Gamma \models P'' \approx_\sigma C[P']$ and $\Gamma \models Q = (\boldsymbol{\nu} m{:}T)(P'|H') \approx_\sigma C[P'|H']$ that is what we need since $(\Gamma, m{:}T \triangleright P')\ \mathcal{S}\ (\Gamma, m{:}T \triangleright P'|H')$.
- this comes by (Close) from $P \xrightarrow{n(m)} P'$, $H \xrightarrow{(\nu m{:}T)\,\overline{n}\langle m\rangle} H'$ and $Q = (\boldsymbol{\nu} m{:}T)(P'|H')$. This case is similar to the previous one.
- by (Dec Close) from $P \xrightarrow{\mathsf{dec}_\delta\, n(m)} P'$, and $H \xrightarrow{(\nu m{:}T)\,\overline{\mathsf{dec}_\delta\, n}\langle m\rangle} H'$ or viceversa. This case is vacuous since $H \in \mathcal{H}^\sigma_\Gamma$, i.e., H cannot interact with P through downgraded actions.

($\Rightarrow$) Let be $\Gamma \triangleright P \in \mathcal{CR}(\cong_\sigma)$, we prove that $\forall \Gamma' \triangleright P'$ such that $\Gamma \triangleright P \rightsquigarrow \Gamma' \triangleright P'$, it holds $\Gamma' \triangleright P' \in \mathcal{W}(\cong_\sigma)$.

Let $\Gamma' \triangleright P' \xrightarrow{\alpha}{}^\sigma \Gamma' \triangleright P''$ with $\alpha \in \{\overline{n}\langle m\rangle, n(m)\}$. By persistence we have $\Gamma' \triangleright P' \cong_\sigma \Gamma' \triangleright P'|H$ for all $H \in \mathcal{H}^\sigma_{\Gamma'}$. Let H be the process $n(x{:}T).\mathbf{0}$ (resp. $\overline{n}\langle m\rangle.\mathbf{0}$) if $\alpha = \overline{n}\langle m\rangle$ (resp. $n(m)$); then $H \in \mathcal{H}^\sigma_{\Gamma'}$. Now, observe that $\Gamma' \triangleright P'|H \xrightarrow{\tau} \Gamma' \triangleright P''$, then from $\Gamma' \triangleright P' \cong_\sigma \Gamma' \triangleright P'|H$ we also have $\Gamma' \triangleright P' \Longrightarrow \Gamma' \triangleright P'''$ such that $\Gamma' \models P'' \cong_\sigma P'''$ as desired.

Let $\Gamma' \triangleright P' \xrightarrow{\alpha}{}^\sigma \Gamma', m{:}T \triangleright P''$ with $\alpha \in \{(\boldsymbol{\nu} m{:}T)\,\overline{n}\langle m\rangle, (\boldsymbol{\nu} m{:}T)\,n(m)\}$. By persistence we have $\Gamma' \triangleright P' \cong_\sigma \Gamma' \triangleright P'|H$ for all $H \in \mathcal{H}^\sigma_{\Gamma'}$. Let H be the process $n(x{:}T).\mathbf{0}$ (resp. $(\boldsymbol{\nu} m{:}T)\overline{n}\langle m\rangle.\mathbf{0}$) if $\alpha = (\boldsymbol{\nu} m{:}T)\,\overline{n}\langle m\rangle$ (resp. $(\boldsymbol{\nu} m{:}T)\,n(m)$); then $H \in \mathcal{H}^\sigma_{\Gamma'}$. Now, observe that $\Gamma' \triangleright P'|H \xrightarrow{\tau} \Gamma' \triangleright (\boldsymbol{\nu} m{:}T)P''$, then from $\Gamma' \triangleright P' \cong_\sigma \Gamma' \triangleright P'|H$ we also have $\Gamma' \triangleright P' \Longrightarrow \Gamma' \triangleright P'''$ such that $\Gamma' \models (\boldsymbol{\nu} m{:}T)P'' \cong_\sigma P'''$ as desired. □

4.3. Controlled release through partial equivalence relations

In [15,16] the security of sequential and multi-threaded programs is expressed in terms of partial equivalence relations (*per* models, for short) which capture the view of low-level observers. Intuitively, a configuration C, representing a program and the current state of the memory, is secure if $C \sim_\ell C$ where $\sim_\ell$ is a symmetric and transitive relation modeling the low-level observation of program executions. The relation $\sim_\ell$ is in general not reflexive, but it becomes reflexive on the set of secure configurations.

Below we show how this approach can be adapted to the Dec π-calculus to characterize the class of processes that satisfy the σ-controlled release property. We first introduce the following notion of partial bisimilarity up to σ-high actions, denoted by $\dot{\approx}_\sigma$. Intuitively, $\dot{\approx}_\sigma$ requires that σ-high actions are simulated by internal transitions, while on σ-low actions it behaves as $\approx_\sigma$. Additional constraints on declassified actions make the bisimilarity persistent, which is essential in order to characterize the $\mathcal{CR}(\cong_\sigma)$ property.

Definition 22 (*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 $\Gamma \triangleright P \xrightarrow{\alpha}_\sigma \Gamma' \triangleright P'$, 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 $\alpha \in \{\overline{n}\langle m\rangle, n(m)\}$, then there exists Q' such that $\Gamma \triangleright Q \Longrightarrow \Gamma \triangleright Q'$ with $\Gamma \vDash Q' \dot{\approx}_\sigma P'$.
- 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 $\Gamma \triangleright Q \Longrightarrow \Gamma \triangleright Q'$ with $\Gamma \vDash Q' \dot{\approx}_\sigma (\boldsymbol{\nu} m : T)P'$ and $\Gamma, m : T \vDash P' \dot{\approx}_\sigma P'$.
- if $\Gamma \triangleright P \xrightarrow{\alpha}^\sigma \Gamma' \triangleright P'$ where α is a declassified action, i.e., $\alpha \in \{\mathsf{dec}_\delta\, n(m), (\boldsymbol{\nu} m{:}T)\, \mathsf{dec}_\delta\, n(m), \overline{\mathsf{dec}_\delta\, n}\langle m\rangle, (\boldsymbol{\nu} m{:}T)\, \overline{\mathsf{dec}_\delta\, n}\langle m\rangle\}$, then $\Gamma' \vDash P' \dot{\approx}_\sigma P'$.

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

The next theorem states that $\dot{\approx}_\sigma$ is reflexive on the set of well typed processes which satisfy the σ-controlled release property. The proof exploits the persistence property of $\dot{\approx}_\sigma$ described by the following lemma.

Proposition 23 (Persistence of $\dot{\approx}_\sigma$)**.** *Let $\sigma \in \Sigma, P$ be a process and Γ be a type environment such that $\Gamma \vdash P$. If $\Gamma \vDash P \dot{\approx}_\sigma P$, then for all $\Gamma' \triangleright P'$ such that $\Gamma \triangleright P \rightsquigarrow \Gamma' \triangleright P'$, it holds $\Gamma' \vDash P' \dot{\approx}_\sigma P'$.*

Proof. See Appendix B. □

The next lemma states that partial bisimilarity up to σ-high actions implies bisimilarity on σ-actions, i.e., $\dot{\approx}_\sigma \subseteq \approx_\sigma$. The proof is immediate.

Lemma 24. *Let $\sigma \in \Sigma, P, Q$ be two processes and Γ be a type environment such that $\Gamma \vdash P, Q$. If $\Gamma \vDash P \dot{\approx}_\sigma Q$ then $\Gamma \vDash P \approx_\sigma Q$.*

The next lemma states that $\mathcal{CR}(\cong_\sigma)$ is preserved under restriction.

Lemma 25. *Let $\sigma \in \Sigma, P$ be a process and $\Gamma, m : T$ be a type environment such that $\Gamma, m : T \vdash P$. If $\Gamma, m : T \triangleright P \in \mathcal{CR}(\cong_\sigma)$ then $\Gamma \triangleright (\boldsymbol{\nu} m : T)P \in \mathcal{CR}(\cong_\sigma)$.*

Proof. See Appendix B. □

We are now in position to prove that a process P in a type environment Γ is secure if and only if $\Gamma \vDash P \dot{\approx}_\sigma P$.

Theorem 26. *Let $\sigma \in \Sigma, P$ be a process and Γ be a type environment such that $\Gamma \vdash P$. $\Gamma \triangleright P \in \mathcal{CR}(\cong_\sigma)$ iff $\Gamma \vDash P \dot{\approx}_\sigma P$.*

Proof. By Theorem 21 it is sufficient to prove that $\Gamma \triangleright P \in \mathcal{W}(\cong_\sigma)$ iff $\Gamma \vDash P \dot{\approx}_\sigma P$.
($\Leftarrow$) From $\Gamma \vDash P \dot{\approx}_\sigma P$, by Proposition 23, we have that $\forall\ \Gamma' \triangleright P'$ such that $\Gamma \triangleright P \rightsquigarrow \Gamma' \triangleright P'$, $\Gamma' \vDash P' \dot{\approx}_\sigma P'$. Let then be $\Gamma \triangleright P \rightsquigarrow \Gamma' \triangleright P'$, we distinguish two cases that correspond to the definition of $\Gamma \triangleright P \in \mathcal{W}(\cong_\sigma)$:

- $\Gamma' \triangleright P' \xrightarrow{\alpha}{}^{\sigma} \Gamma' \triangleright P_2$ with $\alpha \in \{\overline{n}\langle m\rangle, n(m)\}$. Then by definition of $\dot{\approx}_\sigma$ there exists P_3 such that $\Gamma' \triangleright P' \Longrightarrow \Gamma' \triangleright P_3$ and $\Gamma' \vDash P_3 \dot{\approx}_\sigma P_2$. By Lemma 24 and Theorem 9, we conclude $\Gamma' \vDash P_3 \cong_\sigma P_2$.
- $\Gamma' \triangleright P' \xrightarrow{\alpha}{}^{\sigma} \Gamma', m{:}T \triangleright P_2$ with $\alpha \in \{(\nu m{:}T)\,\overline{n}\langle m\rangle, (\nu m{:}T)\,n(m)\}$. Then by definition of $\dot{\approx}_\sigma$, there exists a process P_3 such that $\Gamma' \triangleright P' \Longrightarrow \Gamma' \triangleright P_3$ and $\Gamma' \vDash P_3 \dot{\approx}_\sigma (\nu m{:}T)P_2$. By Lemma 24 and Theorem 9, we conclude $\Gamma' \vDash P_3 \cong_\sigma (\nu m{:}T)P_2$.

($\Rightarrow$) Consider the following binary relation:

$$\mathcal{S} = \{(\Gamma \triangleright P,\ \Gamma \triangleright Q) \mid \Gamma \triangleright P \in \mathcal{W}(\cong_\sigma),\ \Gamma \triangleright Q \in \mathcal{W}(\cong_\sigma) \text{ and } \Gamma \vDash P\ \dot{\approx}_\sigma\ Q\}$$

by Theorem 21 and the fact that $\dot{\approx}_\sigma$ is reflexive, it is sufficient to prove that $\mathcal{S}$ is a partial bisimulation up to σ-high actions. Let us distinguish the following cases:

- $\Gamma \triangleright P \xrightarrow{\alpha}{}_{\sigma} \Gamma' \triangleright P'$. From the hypothesis $\Gamma \vDash P\ \dot{\approx}_\sigma\ Q$ we have that there exists Q' such that $\Gamma \triangleright Q \overset{\hat{\alpha}}{\Longrightarrow}_\sigma \triangleright \Gamma' Q'$ with $\Gamma' \vDash P'\ \dot{\approx}_\sigma\ Q'$. By Proposition 18 we have $\Gamma' \triangleright P' \in \mathcal{W}(\cong_\sigma)$ and $\Gamma' \triangleright Q' \in \mathcal{W}(\cong_\sigma)$, hence, by definition of $\mathcal{S}$, $(\Gamma' \triangleright P',\ \Gamma' \triangleright Q') \in \mathcal{S}$ as desired.
- $\Gamma \triangleright P \xrightarrow{\alpha}{}^{\sigma} \Gamma \triangleright P'$ with $\alpha \in \{\overline{n}\langle m\rangle, n(m)\}$. From the hypothesis $\Gamma \triangleright P \in \mathcal{W}(\cong_\sigma)$, we have that there exists P'' such that $\Gamma \triangleright P \Longrightarrow \Gamma \triangleright P''$ and $\Gamma \vDash P' \cong_\sigma P''$, hence $\Gamma \vDash P'\ \dot{\approx}_\sigma\ P''$ by Theorem 9. Now, from $\Gamma \vDash P\ \dot{\approx}_\sigma\ Q$, we have that there exists Q' such that $\Gamma \triangleright Q \Longrightarrow \Gamma \triangleright Q'$ and $\Gamma \vDash Q'\ \dot{\approx}_\sigma\ P''$, then also $\Gamma \vDash P'\ \dot{\approx}_\sigma\ Q'$. By Proposition 18 we have $\Gamma \triangleright P' \in \mathcal{W}(\cong_\sigma)$ and $\Gamma \triangleright Q' \in \mathcal{W}(\cong_\sigma)$, hence, by definition of $\mathcal{S}$, $(\Gamma \triangleright P',\ \Gamma \triangleright Q') \in \mathcal{S}$ as desired.
- $\Gamma \triangleright P \xrightarrow{\alpha}{}^{\sigma} \Gamma, m{:}T \triangleright P'$ with $\alpha \in \{(\nu m{:}T)\,\overline{n}\langle m\rangle, (\nu m{:}T)\,n(m)\}$. From the hypothesis $\Gamma \triangleright P \in \mathcal{W}(\cong_\sigma)$, we have that there exists P'' such that $\Gamma \triangleright P \Longrightarrow \Gamma \triangleright P''$ and $\Gamma \vDash (\nu m{:}T)P' \cong_\sigma P''$, hence $\Gamma \vDash (\nu m{:}T)P'\ \dot{\approx}_\sigma\ P''$ by Theorem 9. Now, from $\Gamma \vDash P\ \dot{\approx}_\sigma\ Q$, we have that there exists Q' such that $\Gamma \triangleright Q \Longrightarrow \Gamma \triangleright Q'$ and $\Gamma \vDash Q'\ \dot{\approx}_\sigma\ P''$, then also $\Gamma \vDash (\nu m{:}T)P'\ \dot{\approx}_\sigma\ Q'$. By Proposition 18, Theorem 21 and Lemma 25, we have $\Gamma \triangleright (\nu m{:}T)P' \in \mathcal{W}(\cong_\sigma)$ and $\Gamma \triangleright Q' \in \mathcal{W}(\cong_\sigma)$, hence, by definition of $\mathcal{S}$, $(\Gamma \triangleright (\nu m{:}T)P',\ \Gamma \triangleright Q') \in \mathcal{S}$. To conclude we also need $(\Gamma, m{:}T \triangleright P',\ \Gamma, m{:}T \triangleright P') \in \mathcal{S}$, which comes from $\Gamma, m{:}T \triangleright P' \in \mathcal{W}(\cong_\sigma)$ and $\Gamma, m{:}T \vDash P'\ \dot{\approx}_\sigma\ P'$.
- $\Gamma \triangleright P \xrightarrow{\alpha}{}^{\sigma} \Gamma' \triangleright P'$ where α is a declassified action.From the hypothesis $\Gamma \triangleright P \in \mathcal{W}(\cong_\sigma)$, by Proposition 18 we have $\Gamma' \triangleright P' \in \mathcal{W}(\cong_\sigma)$. Now, since $\Gamma' \vdash P'\ \dot{\approx}_\sigma\ P'$, we conclude $(\Gamma' \triangleright P',\ \Gamma' \triangleright P') \in \mathcal{S}$ as desired. □

Corollary 27. *Let $\sigma \in \Sigma$, P be a process and Γ be a type environment such that $\Gamma \vdash P$ and $\forall n \in \text{fn}(P)$, $\Lambda(\Gamma(n)) \preceq \sigma$ (i.e., P has no free σ-high level names). Then $\Gamma \triangleright P \in \mathcal{CR}(\cong_\sigma)$.*

Proof. The fact that the process P has no free σ-high level names implies that $\Gamma \triangleright P \overset{\alpha}{\not\longrightarrow}{}^{\sigma}$. From this fact, together with $\Gamma \vDash P\ \dot{\approx}_\sigma\ P$, we have that $\Gamma \vDash P\ \dot{\approx}_\sigma\ P$, and we conclude by Theorem 26. □

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 $\Gamma \triangleright P \notin \mathcal{CR}(\cong_\sigma)$: in fact, by choosing $H = \overline{h}\langle\ell\rangle$, we have $\Gamma \vDash P \ncong_\sigma P|H$, that is P is insecure.

The characterizations of σ-controlled release presented above provide a better understanding of the operational semantics of secure processes. Moreover, they allow one to define efficient proof techniques for σ-controlled release just by inspecting the typed LTS of processes.

Notice that, as in the case of standard bisimilarity, even if the LTS's are not finite, our property is decidable for the recursion-free calculus and even for the finite-control π-calculus where the number of parallel components in any process is bounded by a constant.

4.4. Compositionality of controlled information release

In this section, we prove some compositionality results of the controlled release property defined above. Such results are useful 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.

The next technical lemma is useful to reason on the derivatives of $\Gamma \triangleright P|Q$ and $\Gamma \triangleright P$.

Lemma 28. *Let P, Q, R be processes and Γ and Γ' be type environments.*

(1) *Let $\Gamma \triangleright P|Q \rightsquigarrow \Gamma' \triangleright R$, then $R = (\nu n_1 : T_1, \ldots, n_k : T_k)(P'|Q')$ for some k such that $k \geq 0$, $\Gamma \triangleright P \rightsquigarrow \Gamma_1 \triangleright P'$ and $\Gamma \triangleright Q \rightsquigarrow \Gamma_2 \triangleright Q'$ with $\Gamma_i \subseteq \Gamma', n_1 : T_1, \ldots, n_k : T_k$ for $i = 1, 2$.*
(2) *Let $\Gamma \triangleright !P \rightsquigarrow \Gamma' \triangleright R$, then $R = (\nu n_1 : T_1, \ldots, n_k : T_k)(P_1 | \cdots | P_s | !P)$ for some k, s such that $k, s \geq 0$ and $\Gamma \triangleright P \rightsquigarrow \Gamma_i \triangleright P_i$ with $\Gamma_i \subseteq \Gamma', n_1 : T_1, \ldots, n_k : T_k$ for $i = 1, \ldots, s$.*

Proof. See Appendix B. □

We say that two processes P and Q *do not synchronize on declassified actions* if for all R such that $\Gamma \triangleright P|Q \rightsquigarrow \Gamma' \triangleright R$ and $R = (\nu n_1{:}T_1, \ldots, n_k{:}T_k)(P'|Q')$ it holds that $\Gamma' \triangleright R \overset{\tau}{\not\longrightarrow}$ using the rules (Dec Comm) and (Dec Close) of Table 3 applied to P' and Q'. Notice that in this case also P' and Q' do not synchronize on declassified actions, i.e., this property is preserved on derivatives.

Theorem 29 (Compositionality of $\mathcal{CR}(\cong_\sigma)$)**.** *Let $\sigma \in \Sigma, P$ and Q be two processes and Γ be a type environment such that $\Gamma \vdash P, Q$. If $\Gamma \triangleright P \in \mathcal{CR}(\cong_\sigma)$ and $\Gamma \triangleright Q \in \mathcal{CR}(\cong_\sigma)$ then*

(1) $\Gamma' \triangleright \overline{a}\langle b \rangle.P \in \mathcal{CR}(\cong_\sigma)$ *where* $\Gamma' = \Gamma \cup \{a : \delta[T]\} \cup \{b : T\}$ *and* $\delta \preceq \sigma$;
(2) $\Gamma' \triangleright a(x{:}T).P \in \mathcal{CR}(\cong_\sigma)$ *where* $\Gamma' = \Gamma \cup \{a : \delta[T]\}$ *and* $\delta \preceq \sigma$;
(3) $\Gamma' \triangleright \mathsf{if}\ a = b\ \mathsf{then}\ P\ \mathsf{else}\ Q \in \mathcal{CR}(\cong_\sigma)$ *where* $\Gamma' = \Gamma \cup \{a : T\} \cup \{b : T\}$;
(4) $\Gamma \triangleright P|Q \in \mathcal{CR}(\cong_\sigma)$ *whenever P and Q do not synchronize on declassified actions.*
(5) $\Gamma' \triangleright (\nu n : T)P \in \mathcal{CR}(\cong_\sigma)$ *where* $\Gamma = \Gamma', n : T$;
(6) $\Gamma \triangleright\ !P \in \mathcal{CR}(\cong_\sigma)$ *whenever P does not syntactically contain declassified actions.*

Proof. See Appendix B. □

Example 30. Let P and Q be finite state processes and Γ be a type environment such that $\Gamma \vdash P, Q$. Even if $R = !P|Q$ might be an infinite state process, we can easily check whether $\Gamma \triangleright R \in \mathcal{CR}(\cong_\sigma)$ just by exploiting the decidability of $\Gamma \triangleright P \in \mathcal{CR}(\cong_\sigma)$ and $\Gamma \triangleright Q \in \mathcal{CR}(\cong_\sigma)$ and the compositionality of $\mathcal{CR}(\cong_\sigma)$ with respect to the parallel composition and replication operators.

5. Examples

In this section, we show a couple of examples that illustrate the expressiveness of our approach. In the following, we use a CCS-style for channels that do not carry values, writing simply n and $\overline{n}$ instead of $n()$ and $\overline{n}\langle\rangle$.

Example 31. Consider the process $P = (\nu h{:}\mathsf{H}[\,])(\overline{h}|!\,h.k.\overline{h})|\overline{k}.\ell$ in the type environment $\Gamma = k : \mathsf{H}[\,], \ell : \mathsf{L}[\,]$. Even if in P the low action ℓ depends on the high action k, we can prove that P is secure by showing that $\Gamma \models P \approx_\sigma P$. Indeed, let $\mathcal{S}$ be the symmetric closure of the following relation:

$$\{\ (P,P),\ (P_1,P_1),\ (P_2,P_2),\ (P_3,P_3),\ (P_4,P_4),\ (P,P_1),\ (P_3,P_4),\ (P_2,P_5)\ \}$$

where

$$\begin{array}{ll} P_1 = (\nu h)(k.\overline{h}|!\,h.k.\overline{h})|\overline{k}.\ell & P_4 = (\nu h)(k.\overline{h}|!\,h.k.\overline{h}) \\ P_2 = (\nu h)(\overline{h}|!\,h.k.\overline{h})|\ell & P_5 = (\nu h)(k.\overline{h}|!\,h.k.\overline{h})|\ell \\ P_3 = (\nu h)(\overline{h}|!\,h.k.\overline{h}) & \end{array}$$

It is straightforward to prove that $\mathcal{S}$ is a bisimulation up to high actions, i.e., $\mathcal{S} \subseteq \dot{\approx}_\sigma$.

Example 32. Consider the insecure process

$$P = h(x{:}T).\,\mathsf{if}\, x = n \,\mathsf{then}\, \overline{\ell_1}\langle\rangle \,\mathsf{else}\, \overline{\ell_2}\langle\rangle \mid \overline{h}\langle n\rangle \mid \overline{h}\langle m\rangle$$

in the type environment $\Gamma = h : \mathsf{H}[T],\ n : T,\ \ell_i : \mathsf{L}[\,]$ for $i = 1, 2$ (here the security level of n is irrelevant), where the variable x can be nondeterministically substituted either with n or not. P is an insecure process since an external attacker can destroy the nondeterminism causing an interference: to prove that $\Gamma \triangleright P \notin \mathcal{CR}(\cong_\sigma)$, let $H = h(y).h(z).\overline{h}\langle n\rangle$, then $\Gamma \vDash P \ \ncong_\sigma P|H$.

Anyway, it might be the case that a programmer wants to allow the flow of information resulting from a test over the value communicated along a secret channel. In this case he simply has to declassify the communication on the channel h, obtaining for instance the following process, which can be proved to be secure:

$$P' = \mathsf{dec}_\sigma h(x{:}T).\,\mathsf{if}\, x = n \,\mathsf{then}\, \overline{\ell_1}\langle\rangle \,\mathsf{else}\, \overline{\ell_2}\langle\rangle \mid \overline{\mathsf{dec}_\sigma h}\langle n\rangle \mid \overline{\mathsf{dec}_\sigma h}\langle m\rangle$$

Notice that this does not prevent the channel h to be later used without declassification. For instance, let *passwd* be a sensitive value which should not be tested at low level, then the process P' can safely run in parallel with a thread $\overline{h}\langle passwd\rangle|h(x{:}T).Q$, which will not be involved in the declassification.

Example 33 (*Job scheduler*)**.** Assume that there are n jobs $P_1, \ldots, P_n$ whose execution must be scheduled. We implement the scheduler as the parallel composition of two threads: the first one produces a numbered token, assigns it to the next job and increments the counter. The second thread consumes a token from a job, checking if it corresponds to the next scheduled number. In case of matching, the consumer acknowledges the job to start its execution, waiting for its end to increment the internal counter. Let be

$$\text{Scheduler} = \text{Producer}|\text{Consumer}$$

where, with an abuse of notation, we use natural numbers as channels:

$$\begin{aligned}
\text{Producer} &= (\boldsymbol{\nu} p{:}T)(\overline{p}\langle 1\rangle|!\,p(x{:}T').\text{enqueue}(y{:}T'').(\overline{y}\langle x\rangle|\overline{p}\langle x+1\rangle))\\
\text{Consumer} &= (\boldsymbol{\nu} c{:}T)(\overline{c}\langle 1\rangle|!\,c(x{:}T').\text{check}(y{:}T').\\
&\qquad\qquad \mathsf{if}\, x = y \,\mathsf{then}\, (\overline{y}\langle \text{ok}\rangle|\text{ack}.\overline{\text{c}}\langle \text{x}+1\rangle) \,\mathsf{else}\, (\overline{\text{y}}\langle \text{no}\rangle|\overline{\text{c}}\langle \text{x}\rangle))
\end{aligned}$$

where the channels enqueue and check are used by jobs respectively to get a token and to exhibit it to the scheduler. Jobs are then written as follows[5] :

$$\begin{aligned}
\text{Job}_i = (\boldsymbol{\nu} j{:}T'')(&\overline{\text{enqueue}}\langle j\rangle.j(x{:}T').\\
&(\boldsymbol{\nu} l{:}\mathsf{L}[\,])(\overline{l}|!\,l.\overline{\text{check}}\langle x\rangle.x(y{:}\mathsf{L}[\,]).\mathsf{if}\, y = \text{ok} \,\mathsf{then}\, P_i.\overline{\text{ack}} \,\mathsf{else}\, \overline{l}))
\end{aligned}$$

First, a job asks for a token and waits for it along the private channel j. The job then starts a loop where it repeatedly exhibits the token to the scheduler, waiting for its turn to be executed. The loop ends when the job receives the ok message, so that it can run the process P, and signal its end using the ack channel.

The system scheduler, $|\text{Job}_1|, \cdots |\text{Job}_n$ can be proved to be secure if we rely on the following type assignment, where the two private channels c and p are high-level, while tokens are low level value (of suitable arity), and the channels enqueue and check are low level as well.

$$\begin{array}{lllll}
c, p : T, & \text{enqueue} : \mathsf{L}[T''], & \text{check} : T'', & 1, 2, \ldots : T', & \text{ok}, \text{no}, \text{ack} : \mathsf{L}[\,]\\
\text{where} & T = \mathsf{H}[T'] & T'' = \mathsf{L}[T'] & T' = \mathsf{L}[\mathsf{L}[\,]] &
\end{array}$$

The fact that the system is secure comes easily by Corollary 27 since there are no free high level names.

[5] With an abuse of notation we write $P_i.\overline{\text{ack}}$; this can be rewritten using the correct syntax of the π-calculus assuming that every job signals its termination.

Example 34. A final example shows the use of downgrading to enhance the flexibility of secure programs. Consider the following lattice of security levels:

$$\begin{array}{c} \mathsf{H} = \text{Top Secret} \\ | \\ \sigma = \text{Protected} \\ | \\ \mathsf{L} = \text{Public} \end{array}$$

where $\mathsf{L} \prec \sigma \prec \mathsf{H}$. We write a simple protocol where a server checks the client's password. The protocol is built upon the following channels, where we assume that Bool is the type of a public boolean value:

check : $\sigma[\sigma[\,],\sigma[\,]]$	carries the client's id together with the inserted password
ack : L[Bool]	acknowledges the result of the check
trans : H[Bool]	forwards the result of the check

As detailed below, a client simply sends his id together with a password and waits for an acknowledgment. The server spawns an instance of the (replicated) checker process, which matches the received password y against the expected password which is stored in the system database as the image $I(x)$ of client's id. The result of this matching is transmitted twice along a secret channel trans. The first, declassified, communication along trans causes the sending of the acknowledgment back to the client, then the second secret communication along trans controls the reconfiguration of the server which either restarts or enters a new, not specified, high-level state Alert.

$$\begin{aligned} \text{Client} &= \overline{\text{check}}\langle \text{id}, \text{pwd}\rangle.\text{ack}(\text{x}).\text{P} \\ \text{Checker} &= (\boldsymbol{\nu} n{:}\sigma[\,])\ (\ \overline{n}\langle\rangle \\ &\qquad |!n().\text{check}(x,y).\overline{\mathsf{dec}_\sigma \text{trans}}\langle y = I(x)\rangle.\overline{\text{trans}}\langle y = I(x)\rangle \\ &\qquad |!\mathsf{dec}_\sigma\text{trans}(z).\overline{\text{ack}}\langle z\rangle \\ &\qquad |!\text{trans}(z).\mathsf{if}\ z\ \mathsf{then}\ \overline{n}\langle\rangle\ \mathsf{else}\ \text{Alert}\ ) \end{aligned}$$

We can prove that the process Client|Checker satisfies the controlled information release property with respect to the observable level σ.

6. Discussion

The theory developed in this paper applies to closed processes and does not explicitly consider recursive types. In this section we discuss a couple of extensions, dealing with open terms and recursive types.

6.1. Open terms

When modeling reactive systems, that is, concurrent systems with interacting subsystems, it is useful to reason on open terms, that is, processes which are only partially specified. In this paper we considered closed processes only, however, our theory scales to open terms as described below.

- First introduce 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[6] Γ with Γ', and then
- say that a term T satisfies the *σ-controlled release property in* Γ, written $\Gamma \rhd T \in \mathcal{CR}(\cong^o_\sigma)$, if for all closing substitution ρ which respects Γ with Γ', $\Gamma' \rhd T\rho \in \mathcal{CR}(\cong_\sigma)$.

In this way, we obtain that if $\Gamma \rhd T \in \mathcal{CR}(\cong^o_\sigma)$ then for all $H \in \mathcal{H}^\sigma_{\Gamma'}$, it holds $\Gamma \vDash T \cong^o_\sigma T|H$.

[6] 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 $i = 1, \ldots, n$.

6.2. Recursive types

Recursive types are ubiquitous in modern programming languages, as they occur when typing constructs such as datatype definitions.

Our theory smoothly scales to the π-calculus extended with recursive types, the main changes being in the type system's rules for type and environment formation. In the following we present the new rules, together with the generalization of the function Λ which assigns a level to each type. Moreover, we illustrate that none of the results and theorems presented in the paper are affected by the presence of recursive types. This is due to the fact that we assume type equality up to unfolding of recursive types and we assume that the types assigned to names have no free type variables. We show that these two assumptions are sufficient to state that all the results presented in the paper remain unchanged also in the presence of recursive types.

In the Dec$\,\pi$-calculus with recursive types the syntax of types becomes the following:

$$T \;::=\; X \mid \mu X.\delta[T] \mid \delta[] \mid \delta[T]$$

where X is a type variable and μ is the recursion operator. As an example, in this calculus the process $\overline{a}\langle a\rangle$ can be typed assuming for the name a the recursive type $\mu X.\delta[X]$.

As stated above, in the following

($*$) we assume type equality up to renaming of bound type variables and up to unfolding of recursive types (i.e., $\mu X.\delta[T] = \delta[T\{X := \mu X.\delta[T]\}]$), and

($**$) we assume that in processes and type judgments the types assigned to names have no free (type) variables, i.e., they are closed types.

We are going to illustrate that, thanks to these assumptions, the results of Section 3 and 4 smoothly scale to the case of recursive types.

The type system with recursive types First, we have to extend type environments to let them list both (free) type variables and type assignments to names. Type environments are now generated by the following grammar:

$$\Gamma \;::=\; \emptyset \mid \Gamma, X \mid \Gamma, a : T$$

We also generalize to recursive types the level function Λ that associates to types the corresponding level. In order to deal with type variables, the level function Λ must depend on an environment Δ, that is a function from type variables to security levels in Σ:

$$\Lambda_\Delta(\delta[]) = \Lambda_\Delta(\delta[T]) = \Lambda_\Delta(\mu X.\delta[T]) = \delta$$
$$\Lambda_\Delta(X) = \Delta(X)$$

The environment Δ is needed in the rules for type formation only as illustrated below.

The type judgments used in the type system are of the following forms:

$\Gamma \vdash_\Delta T$	the type T is well formed in Γ
$\Gamma \vdash \diamond$	the type environment Γ is well formed
$\Gamma \vdash a : T$	the name/variable a has type T in Γ
$\Gamma \vdash P$	the process P is well typed in Γ

Notice that only the judgment for type formation depends on the function Δ. This comes from the fact that well formed types ensure that a channel of level δ carries values of level lower or equal than δ, hence the rules of type formation depend on the (generalized) level function Λ_Δ defined above. The rules for well formed types are in the following:

(WF TYPE VAR)
$$\frac{\Gamma \vdash \diamond}{\Gamma \vdash_\Delta X} \quad X \in \Gamma$$

(WF EMPTY CH TYPE)
$$\frac{\Gamma \vdash \diamond}{\Gamma \vdash_\Delta \delta[\,]}$$

(WF CH TYPE)
$$\frac{\Gamma \vdash_\Delta T}{\Gamma \vdash_\Delta \delta[T]} \quad \Lambda_\Delta(T) \preceq \delta$$

(WF CH REC TYPE)
$$\frac{\Gamma, X \vdash_{\Delta \cup \{X \to \delta\}} T}{\Gamma \vdash_\Delta \mu X.\delta[T]} \quad \Lambda_{\Delta \cup \{X \to \delta\}}(T) \preceq \delta$$

Type formation rules are used in the following rules for well formed type environments. In particular, the second premise of rule (NAME VAR) states that well formed environments associate (well formed) *closed types* to names and variables. As a consequence, in every derivable judgment of the form $\Gamma \vdash a : T$, the type T is closed, according to our assumption $(**)$.

(EMPTY)
$$\frac{}{\emptyset \vdash \diamond}$$

(TYPE VAR)
$$\frac{\Gamma \vdash \diamond}{\Gamma, X \vdash \diamond} \quad X \notin Dom\,(\Gamma)$$

(NAME VAR)
$$\frac{\Gamma \vdash \diamond \quad \emptyset \vdash_\emptyset T}{\Gamma, a : T \vdash \diamond} \quad a \notin Dom\,(\Gamma)$$

The remaining rules of the type system, that is the rules for the judgments $\Gamma \vdash a : T$ and $\Gamma \vdash P$, are identical to those in Section 2 (Table 4). It is worth noticing that the presence of recursive types does not affect the typing rules. For instance, if the name a has a recursive type $\mu X.\delta[T]$ in Γ, then, whenever a judgment of the form $\Gamma \vdash a : \delta[T]$ is needed in the premise of a rule (e.g., in rule (OUTPUT)), such a judgment is derivable by the type equality up to unfolding that we assumed above (*), i.e., by the assumption that $\mu X.\delta[T] = \delta[T\{X := \mu X.\delta[T]\}]$.

For the same reason, the Subject Reduction property is the same as in Section 2, and the typed LTS for the calculus with recursive types is identical to that in Section 3. Furthermore, the notation $\Lambda(\Gamma(n))$ we used throughout the paper can be safely thought of as $\Lambda_\emptyset(\Gamma(n))$ thanks to the assumption (**) about closed channel types. For these reasons the results about observation equivalence and controlled release developed in Section 3 and Section 4 need not to be modified by the presence of recursive types.

6.3. Encoding of Decπ *into π-calculus*

It is worth noticing that the Decπ-calculus is a smooth extension of the π-calculus. Indeed, it can be encoded into the standard π-calculus by simulating a declassified communication along h with a synchronization along a fresh (session) channel h', as exemplified by the following:

$$\{\!|\, \overline{\mathsf{dec}\, h}\langle m\rangle.P | \mathsf{dec}\, h(x).Q \,|\!\} = (\nu h')(\overline{\mathsf{D}}\langle h'\rangle.\overline{h'}\langle m\rangle.\{\!|\, P \,|\!\})|\mathsf{D}(w).w(x).\{\!|\, Q \,|\!\}$$

where a specific name D is used as a port to transmit the name of the private session-channel used for the declassified communication.

A $\cong_\sigma$-preserving encoding of the Decπ into the π-calculus can then be defined along the lines of the previous intuition, taking special care in handling the typing of encoded processes and the level of the downgrading constructs. The notion of Noninterference for the π-calculus has been studied in [17], together with the corresponding proof techniques based on name restriction, unwinding and per-models; it is then interesting to rely on the previous encoding to compare Controlled Release and Noninterference.

Let P be a Decπ-process such that $P \in \mathcal{CR}(\cong_\sigma)$, then a declassified action performed by P is encoded into a high action performed by $\{\!|\, P \,|\!\}$ along the port D. According to [17], in the π-calculus a noninterferent process is such that every high action is matched by a number of unobservable-τ-steps, which is not in general the case of $\{\!|\, P \,|\!\}$. In other words, if $P \in \mathcal{CR}(\cong_\sigma)$ then in general $\{\!|\, P \,|\!\}$ is not interference-free. This fact shows that in order to implement a downgrading mechanism in the π-calculus, it is not sufficient to communicate using private names, but there must be also a mean to specify a set of admissible information flows. The Decπ-calculus we propose uses the new construct dec to identify the actions that determine an admissible flow, and studies a security property that precisely allows only the intended information flows.

7. Related work

In the context of process calculi, the problem of detecting only uncontrolled information flows has been studied for CSP and CCS, see, e.g., [7,18–21]. The work which is most related to ours is [7] by Bossi, Piazza and the second author. They propose a general unwinding framework for formalizing different noninterference 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.

As for the π-calculus, the only work we are aware of dealing with a form of downgrading is a recent work by Gordon and Jeffrey about conditional secrecy [22]. 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. On the other hand, there exist a number of works about noninterference in the π-calculus. Hennessy and Riely [23,2] 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. Honda, Yoshida and Vasconcelos [3,4] consider advanced type systems for processes of the linear/affine π-calculus where each action type is associated to a secrecy level. Their noninterference results are expressed in terms of typed bisimulation equivalences. In [6] Pottier develops a type theory which is, roughly, as expressive as the one of Hennessy and Riely [23] and proves a noninterference result based on bisimulation equivalence. Kobayashi in [5] proposes a refinement of a previous type system for deadlock/livelock-freedom and shows that well-typed processes enjoy a bisimulation-based noninterference property.

In all these works types play an essential role in the proof of noninterference: the security of processes is ensured by the strong constraints imposed by the type systems. On the contrary, our approach relies on a much simpler typing discipline. Indeed, our type system does not deal with implicit information flow. Instead, we characterize security in terms of the actions that may be performed by typed processes.

In order to illustrate the difference between our approach and those discussed above, consider the process $P = (\boldsymbol{\nu} h{:}\mathsf{H}[\,])(\overline{h}|!\,(h.(\overline{k}|\overline{h}))|k.\overline{\ell})$. We can prove that P satisfies the controlled release property. 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. One might think that our approach is more general than the others, but it is not the case. For instance, the process $!\ h(x{:}T).\overline{\ell}\langle x\rangle$ is always insecure in our framework, whereas using the type system of [4] one can find a linear/affine typing which deems it secure.

The type-based proof techniques for noninterference appearing in the literature are often associated with subtyping. Subtyping is typically used to increase the flexibility by allowing more system interactions. In this paper we do not deal with subtyping. However, we could extend our approach with a form of subtyping to safely increase the secure upward information flows permitted in the $\mathsf{Dec}\,\pi$-calculus. This could be done by introducing a subtyping relation that allows a channel type $\delta_1[T]$ (resp. $\delta_1[\,]$) to be suptype of $\delta_2[T]$ (resp. $\delta_2[\,]$) when $\delta_1 \preceq \delta_2$. This form of subtyping would allow a low level channel to be used in a place where a high level channel is expected, e.g., $h(x : \top[]).P|\overline{h}\langle\ell\rangle$ where $h : \top[\top[]], \ell : \bot[]$. Clearly, adding this subtyping relation would complicate the observation equivalence of the $\mathsf{Dec}\,\pi$-calculus since, as explained in [8], typed actions would require reasoning about two different type environments, one to type the observer and another one to type the observed process. We argue that our approach could be extended with subtyping along the lines of [8]. We plan to investigate this topic for future work.

8. Conclusions

In this paper, we develop a theory of controlled information release for processes of the Decπ-calculus, which is an extension of the π-calculus with a construct intended to be used by programmers for declassifying information from a higher to a lower security level.

In [1] Sabelfeld and Sands suggest some principles which are intended to guide the definition of satisfactory security policies for systems admitting declassification mechanisms. We conclude the paper with a discussion showing that our notion of controlled information release satisfies the principles of semantic consistency, conservativity, monotonicity of release, and non-occlusion.

The *semantic consistency* principle is useful for modular design of secure complex systems. It allows one to replace part of a system with a semantically equivalent process provided that it does not perform any declassification operation. More precisely, the semantic consistency principle states that "the (in)security of a program is invariant under semantics-preserving transformations of declassification-free subprograms". In the case of controlled information release, we can prove that if P, Q and R are processes and Γ is a type environment such that $\Gamma \vdash P, Q, R$ and neither Q or R contain any declassified action then the following holds: whenever $\Gamma \models Q \cong_{\top} R$ and $\Gamma \triangleright P|Q \in \mathcal{CR}(\cong_\sigma)$ then also $\Gamma \triangleright P|R \in \mathcal{CR}(\cong_\sigma)$.

The *conservativity* principle deals with the natural intuition that a notion of security in the presence of downgrading should be a conservative extension of a security definition for a language without downgrading. It simply states that "the security for programs with no declassification is equivalent to noninterference". In other words, it requires that processes without declassified actions satisfy a strong noninterference property that forbids any secret leaks. It is straightforward to show that our notion of controlled information release satisfies this principle. Indeed, we can immediately prove that if P is a Decπ-process and Γ is a type environment such that $\Gamma \vdash P$ and P does not syntactically contain any declassified action then $\Gamma \triangleright P \in \mathcal{CR}(\cong_\sigma)$ if and only if $\Gamma \triangleright P \in \mathcal{NI}(\cong_\sigma)$ where $\mathcal{NI}(\cong_\sigma)$ is the strong noninterference property for processes of the π-calculus presented by the authors in [17].

The principle named *monotonicity of release* says that "adding further declassifications to a secure program cannot render it insecure, or, equivalently, an insecure program cannot be made secure by removing declassification annotations". Also this principle is easy to check for controlled information release just by looking at its unwinding characterization. Monotonicity comes from the fact that adding a declassification reduces the number of high actions and then the checks required by the unwinding condition.

Finally, the *non-occlusion* principle is intended to prevent the risk of laundering secrets not intended for declassification. It formally states that "the presence of a declassification operation cannot mask other covert information leaks". In our framework, where we do not declassify expressions and an information is just the fact that an action has occurred, this principle simply requires that a low level observer cannot infer any non declassified sensitive information, which is exactly our controlled information release property.

Acknowledgments

Thanks are due to Nobuko Yoshida for the useful discussions on noninterference in the linear/affine π-calculus.

Appendix

A. Proofs omitted from Section 3

In the following we write $\Gamma \vdash_\sigma P$ to state that $\Gamma \vdash P$ and P is a process of level at most σ in Γ, i.e., $\forall n \in \text{fn}(P)$, $\Lambda(\Gamma(n)) \preceq \sigma$. Similarly, we write $\Gamma' \vdash_\sigma C[\cdot_\Gamma]$ to state that $C[\cdot_{\Gamma'}]$ is a (Γ'/Γ)-σ-context.

We start with Propositions 36, 37, 38 proving that $\approx_\sigma$ is an equivalence relation such that $\approx_\sigma \subseteq \cong_\sigma$. The next lemma is used in the proof of Proposition 36.

Lemma 35. *Let $\sigma \in \Sigma$, R be a process and Γ be a type environment such that $\Gamma \vdash_\sigma R$ and $\Gamma \triangleright R \xrightarrow{\alpha}_\sigma \Gamma' \triangleright R'$. Then $\Gamma' \vdash_\sigma R'$.*

Proof. Immediate. □

Proposition 36. *Let $\sigma \in \Sigma$. $\approx_\sigma$ is a congruence with respect to σ-contexts.*

Proof. The proof that $\approx_\sigma$ is an equivalence relation is standard.

We prove that $\approx_\sigma$ is preserved by all σ-contexts. We consider all the constructs simultaneously: let $\mathcal{S}$ be a binary relation such that

- $\approx_\sigma \subseteq \mathcal{S}$
- $\Gamma \models P \;\mathcal{S}\; Q$ implies $\Gamma, \Gamma' \models (P|R) \;\mathcal{S}\; (Q|R)$ for every process R such that $\Gamma, \Gamma' \vdash_\sigma R$.
- $\Gamma \models P \;\mathcal{S}\; Q$ implies $\Gamma' \models (\boldsymbol{\nu} n{:}T)P \;\mathcal{S}\; (\boldsymbol{\nu} n{:}T)Q$ where $\Gamma = \Gamma', n : T$.

We show that $\mathcal{S}$ is a bisimulation on σ-actions, hence $\mathcal{S} \subseteq \approx_\sigma$. The proof is by induction on the formation of $\mathcal{S}$.

- The case where $\Gamma \models P \;\mathcal{S}\; Q$ comes from $\Gamma \models P \approx_\sigma Q$ is trivial.
- Let $\Gamma, \Gamma' \models (P|R) \;\mathcal{S}\; (Q|R)$ with $\Gamma \models P \;\mathcal{S}\; Q$ and $\Gamma, \Gamma' \vdash_\sigma R$. Assume $\Gamma, \Gamma' \triangleright P|R \xrightarrow{\alpha}_\sigma \Gamma'' \triangleright P'$, this comes from one of following cases:

  . $\Gamma, \Gamma' \triangleright P \xrightarrow{\alpha}_\sigma \Gamma'' \triangleright P''$, $\mathrm{bn}(\alpha) \cap \mathrm{fn}(R) = \emptyset$ and $P' = P''|R$. From $\Gamma \models P \;\mathcal{S}\; Q$, by inductive hypothesis we have that $\exists Q'$ such that $\Gamma, \Gamma' \triangleright Q \overset{\hat{\alpha}}{\Longrightarrow}_\sigma \Gamma'' \triangleright Q'$ and $\Gamma'' \models P'' \;\mathcal{S}\; Q'$. Then by rule (Par) of the LTS, $\Gamma, \Gamma' \triangleright Q|R \overset{\hat{\alpha}}{\Longrightarrow}_\sigma \Gamma'' \triangleright Q'|R$. Now, from $\Gamma'' \models P'' \;\mathcal{S}\; Q'$ by definition of $\mathcal{S}$ and the fact that $\Gamma'' \vdash_\sigma R$ we conclude $\Gamma'' \models (P''|R) \;\mathcal{S}\; (Q'|R)$ as desired.

  . $\Gamma, \Gamma' \triangleright R \xrightarrow{\alpha}_\sigma \Gamma'' \triangleright R'$ and $P' = P|R'$. By (Par) we have $\Gamma, \Gamma' \triangleright Q|R \xrightarrow{\alpha}_\sigma \Gamma'' \triangleright Q|R'$. By Lemma 35 we know that $\Gamma'' \vdash_\sigma R'$, then from $\Gamma \models P \;\mathcal{S}\; Q$ we conclude $\Gamma'' \models P|R' \;\mathcal{S}\; Q|R'$ as desired.

  . $\alpha = \tau$, $P \xrightarrow{\overline{n}\langle m\rangle} P''$, $R \xrightarrow{n(m)} R'$ and $P' = P''|R'$. By Proposition 6 we have $\Gamma, \Gamma' \triangleright P \xrightarrow{\overline{n}\langle m\rangle}_\delta \Gamma, \Gamma' \triangleright P''$, and $\Gamma, \Gamma' \triangleright R \xrightarrow{n(m)}_\delta \Gamma, \Gamma' \triangleright R'$ with $\Lambda(\Gamma, \Gamma'(n)) \preceq \delta$. From $\Gamma, \Gamma' \vdash_\sigma R$, we have $\Lambda(\Gamma, \Gamma'(n)) \preceq \sigma$, then we can choose $\delta = \sigma$. Now, from $\Gamma \models P \;\mathcal{S}\; Q$, by inductive hypothesis we have that $\Gamma, \Gamma' \triangleright Q \overset{\overline{n}\langle m\rangle}{\Longrightarrow}_\sigma \Gamma, \Gamma' \triangleright Q'$ with $\Gamma, \Gamma' \models P'' \;\mathcal{S}\; Q'$. By rules (Comm) and (Par) of the LTS we have $\Gamma, \Gamma' \triangleright Q|R \Longrightarrow \Gamma, \Gamma' \triangleright Q'|R'$. From $\Gamma, \Gamma' \models P'' \;\mathcal{S}\; Q'$, by definition of $\mathcal{S}$ and Lemma 35 we conclude $\Gamma, \Gamma' \models (P''|R') \;\mathcal{S}\; (Q'|R')$.

  . $\alpha = \tau$, $R \xrightarrow{\overline{n}\langle m\rangle} R''$, $P \xrightarrow{n(m)} P''$. This case is analogous to the previous one.

  . $\alpha = \tau$, $R \xrightarrow{\overline{\mathsf{dec}_{\delta_1} n}\langle m\rangle} R''$, $P \xrightarrow{\mathsf{dec}_{\delta_1} n(m)} P''$. By Proposition 6 we have that $\Gamma, \Gamma' \triangleright R \xrightarrow{\overline{\mathsf{dec}_{\delta_1} n}\langle m\rangle}_\delta \Gamma, \Gamma' \triangleright R''$, and $\Gamma, \Gamma' \triangleright P \xrightarrow{\mathsf{dec}_{\delta_1} n(m)}_\delta \Gamma, \Gamma' \triangleright P''$ with $\delta_1 \prec \Lambda(\Gamma, \Gamma'(n)) \preceq \delta$. From $\Gamma, \Gamma' \vdash_\sigma R$, we have $\Lambda(\Gamma, \Gamma'(n)) \preceq \sigma$, then we can choose $\delta = \sigma$ and we conclude similarly to the previous case. The symmetric case is similar.

  . $\alpha = \tau$, $\Gamma, \Gamma' \triangleright P|R \xrightarrow{\tau}_\sigma \Gamma, \Gamma' \triangleright (\boldsymbol{\nu} m{:}T)(P'|R')$ since $P \xrightarrow{(\nu m:T)\,\overline{n}\langle m\rangle} P'$ and $R \xrightarrow{n(m)} R'$ with $m \neq \mathrm{fn}(R)$. By Propositions 6 and 7 we have $\Gamma, \Gamma' \triangleright P \xrightarrow{(\nu m:T)\,\overline{n}\langle m\rangle}_\delta \Gamma, \Gamma', m{:}T \triangleright P'$ and $\Gamma, \Gamma' \triangleright R \xrightarrow{(\nu m:T)\, n(m)}_\delta \Gamma, \Gamma', m{:}T \triangleright R'$ with $\Gamma, \Gamma'(n) = \sigma_1[T]$ and $\sigma_1 \preceq \delta$. From $\Gamma, \Gamma' \vdash_\sigma R$, we have $\Lambda(\Gamma, \Gamma'(n)) \preceq \sigma$, then we can choose $\delta = \sigma$. Now, from $\Gamma \models P \;\mathcal{S}\; Q$, by inductive hypothesis we have that $\Gamma, \Gamma' \triangleright Q \Longrightarrow \Gamma, \Gamma' \triangleright Q_1 \xrightarrow{(\nu m:T)\,\overline{n}\langle m\rangle}_\sigma \Gamma, \Gamma', m{:}T \triangleright Q_2 \Longrightarrow \Gamma, \Gamma', m{:}T \triangleright Q'$ with $\Gamma, \Gamma', m{:}T \models P' \;\mathcal{S}\; Q'$. Then we also have $\Gamma, \Gamma' \triangleright Q|R \Longrightarrow \Gamma, \Gamma' \triangleright Q_1|R \xrightarrow{\tau} \Gamma, \Gamma' \triangleright (\boldsymbol{\nu} m{:}T)(Q_2|R') \Longrightarrow \Gamma, \Gamma' \triangleright (\boldsymbol{\nu} m{:}T)(Q'|R')$. From $\Gamma, \Gamma', m{:}T \models P' \;\mathcal{S}\; Q'$, by definition of $\mathcal{S}$ and Lemma 35 we conclude $\Gamma, \Gamma' \models (\boldsymbol{\nu} m{:}T)(P'|R') \;\mathcal{S}\; (\boldsymbol{\nu} m{:}T)(Q'|R')$. The symmetric case is similar.

- $\alpha = \tau, \Gamma, \Gamma' \rhd P|R \xrightarrow{\tau}_\sigma \Gamma, \Gamma' \rhd (\nu m{:}T)(P'|R')$ since $P \xrightarrow{(\nu m:T)\,\overline{\mathsf{dec}_{\delta_1} n}\langle m\rangle} P'$ and $R \xrightarrow{\mathsf{dec}_{\delta_1} n(m)} R'$ with $m \neq \mathrm{fn}(R)$. By Propositions 6 and 7 we have $\Gamma, \Gamma' \rhd P \xrightarrow{(\nu m:T)\,\overline{\mathsf{dec}_{\delta_1} n}\langle m\rangle}_\delta \Gamma, \Gamma', m{:}T \rhd P'$ and $\Gamma, \Gamma' \rhd R \xrightarrow{(\nu m:T)\,\mathsf{dec}_{\delta_1} n(m)}_\delta \Gamma, \Gamma', m{:}T \rhd R'$ with $\Gamma, \Gamma'(n) = \sigma_1[T]$ and $\delta_1 \prec \sigma_1 \preceq \delta$. From $\Gamma, \Gamma' \vdash_\sigma R$, we have $\Lambda(\Gamma, \Gamma'(n)) \preceq \sigma$, then we can choose $\delta = \sigma$ and we conclude as in the previous case. The symmetric case is similar.

- Let $\Gamma' \vDash (\nu n{:}T)P \mathrel{\mathcal{S}} (\nu n{:}T)Q$ with $\Gamma \vDash P \mathrel{\mathcal{S}} Q$ and $\Gamma = \Gamma', n{:}T$. Assume $\Gamma' \rhd (\nu n{:}T)P \xrightarrow{\alpha}_\sigma \Gamma'' \rhd P'$, this comes from one of the following cases:

  - $\alpha = (\nu n{:}T)\,\overline{p}\langle n\rangle$ and $\Gamma' \rhd (\nu n{:}T)P \xrightarrow{(\nu n:T)\,\overline{p}\langle n\rangle}_\sigma \Gamma', n{:}T \rhd P'$, with $\Gamma', n{:}T \rhd P \xrightarrow{\overline{p}\langle n\rangle}_\sigma \Gamma', n{:}T \rhd P'$. By inductive hypothesis we have $\Gamma', n{:}T \rhd Q \xRightarrow{\overline{p}\langle n\rangle}_\sigma \Gamma', n{:}T \rhd Q'$ with $\Gamma', n{:}T \vDash P' \mathrel{\mathcal{S}} Q'$. We conclude $\Gamma' \rhd (\nu n{:}T)Q \xRightarrow{(\nu n:T)\,\overline{p}\langle n\rangle}_\sigma \Gamma', n{:}T \rhd Q'$ by an application of the rule (OPEN).
  - $\alpha = (\nu n{:}T)\,\overline{\mathsf{dec}_\delta\, p}\langle n\rangle$ and $\Gamma' \rhd (\nu n{:}T)P \xrightarrow{\alpha}_\sigma \Gamma', n{:}T \rhd P'$, which comes from $\Gamma', n{:}T \rhd P \xrightarrow{\overline{\mathsf{dec}_\delta\, p}\langle n\rangle}_\sigma \Gamma', n{:}T \rhd P'$. From the hypothesis that $\Gamma' \vdash (\nu n{:}T)P$ we have that $\delta \prec \sigma$. By inductive hypothesis we have that $\Gamma', n{:}T \rhd Q \xRightarrow{\overline{\mathsf{dec}_\delta\, p}\langle n\rangle}_\sigma \Gamma', n{:}T \rhd Q'$ with $\Gamma', n{:}T \vDash P' \mathrel{\mathcal{S}} Q'$. We can then conclude $\Gamma' \rhd (\nu n{:}T)Q \xRightarrow{(\nu n:T)\,\overline{\mathsf{dec}_\delta\, p}\langle n\rangle}_\sigma \Gamma', n{:}T \rhd Q'$ by an application of the rule (DEC OPEN).
  - $\Gamma' \rhd (\nu n{:}T)P \xrightarrow{\alpha}_\sigma \Gamma'' \rhd (\nu n{:}T)P'$, with $\Gamma', n{:}T \rhd P \xrightarrow{\alpha}_\sigma \Gamma'', n{:}T \rhd P'$ and $n \notin \mathrm{fn}(\alpha) \cup \mathrm{bn}(\alpha)$. By inductive hypothesis $\Gamma', n{:}T \rhd Q \xRightarrow{\hat{\alpha}}_\sigma \Gamma'', n{:}T \rhd Q'$ with $\Gamma'', n{:}T \vDash P' \mathrel{\mathcal{S}} Q'$. Then also $\Gamma' \rhd (\nu n{:}T)Q \xRightarrow{\hat{\alpha}}_\sigma \Gamma'' \rhd (\nu n{:}T)Q'$ by an application of the rule (RES) of the LTS, and we can conclude $\Gamma'' \vDash (\nu n{:}T)P' \mathrel{\mathcal{S}} (\nu n{:}T)\, Q'$ by definition of $\mathcal{S}$ and by $\Gamma'', n{:}T \vDash P' \mathrel{\mathcal{S}} Q'$. □

Proposition 37. *Let $\sigma \in \Sigma$. $\approx_\sigma$ is reduction closed.*

Proof. Let P, Q be processes such that $\Gamma \vDash P \approx_\sigma Q$ and $P \xrightarrow{\tau} P'$. Then by rule (RED) of the LTS, $\Gamma \rhd P \xrightarrow{\tau}_\sigma \Gamma \rhd P'$. By definition of $\approx_\sigma$, $\Gamma \rhd Q \Longrightarrow \Gamma \rhd Q'$ and $\Gamma \vDash P' \approx_\sigma Q'$ as desired. □

Proposition 38. *Let $\sigma \in \Sigma$. $\approx_\sigma$ is σ-barb preserving.*

Proof. Let P, Q be processes such that $\Gamma \vDash P \approx_\sigma Q$ and $\Gamma \vDash P \downarrow_n^\sigma$, that is $P \xrightarrow{\overline{n}\langle m\rangle} P'$. In this case, by Proposition 6 and the hypothesis $\Lambda(\Gamma(n)) \preceq \sigma$, we have $\Gamma \rhd P \xrightarrow{\overline{n}\langle m\rangle}_\sigma \Gamma \rhd P'$. Now, by definition of $\approx_\sigma$ we also have $\Gamma \rhd Q \xRightarrow{\overline{n}\langle m\rangle}_\sigma \Gamma \rhd Q'$, then $\Gamma \vDash Q \Downarrow_n^\sigma$ as desired. □

Proposition 41 proves that $\cong_\sigma \subseteq \approx_\sigma$, using the following Lemma 39 and 40.

Lemma 39. *If $\Gamma, \omega{:}\sigma[\,] \vDash (\overline{\omega}\langle\rangle \mid P) \cong_\sigma (\overline{\omega}\langle\rangle \mid Q)$ with ω fresh in P, Q, then $\Gamma \vDash P \cong_\sigma Q$.*

Proof. It is sufficient to prove that the following relation

$$\mathcal{R} = \{(\Gamma \rhd P,\ \Gamma \rhd Q) \mid \Gamma, \omega{:}\sigma[\,] \vDash P|\overline{\omega}\langle\rangle \cong_\sigma Q|\overline{\omega}\langle\rangle \quad \omega \notin \mathrm{fn}(P) \cup \mathrm{fn}(Q)\}$$

is reduction closed, σ-barb preserving and σ-contextual. See [8] for details. □

Lemma 40. *If $\Gamma, \omega{:}\sigma[T] \vDash (\nu m{:}T)(P|\overline{\omega}\langle m\rangle) \cong_\sigma (\nu m{:}T)(Q|\overline{\omega}\langle m\rangle)$ with ω fresh in P, Q, then $\Gamma, m{:}T \vDash P \cong_\sigma Q$.*

Proof. Let $\mathcal{R}$ be the following relation:

$$\begin{aligned}&\{(\Gamma, m{:}T \rhd P,\ \Gamma, m{:}T \rhd Q) \mid \\ &\qquad \Gamma, \omega{:}\sigma[T] \vDash (\nu m{:}T)(P|\overline{\omega}\langle m\rangle) \cong_\sigma (\nu m{:}T)(Q|\overline{\omega}\langle m\rangle) \\ &\qquad \omega \notin \mathrm{fn}(P) \cup \mathrm{fn}(Q)\ \}\end{aligned}$$

It is sufficient to prove that $\mathcal{R}$ is reduction closed, σ-barb preserving and σ-contextual. See [8] for details. □

Proposition 41. *For any process P, Q, if $\Gamma \models P \cong_\sigma Q$ then $\Gamma \models P \approx_\sigma Q$.*

Proof. Let $\mathcal{S}$ be the following relation

$$\mathcal{S} = \{(\Gamma \rhd P,\ \Gamma \rhd Q) | \Gamma \models P \cong_\sigma Q\}.$$

We prove that $\mathcal{S}$ is a bisimulation on σ actions; then $\cong_\sigma \subseteq \approx_\sigma$ follows by the fact that $\approx_\sigma$ is the largest bisimulation on σ-actions.
Assume $\Gamma \rhd P \xrightarrow{\alpha}_\sigma \Gamma' \rhd P'$, we distinguish the following cases:

- $\Gamma \rhd P \xrightarrow{\tau}_\sigma \Gamma \rhd P'$. From the hypothesis $\Gamma \models P \cong_\sigma Q$, we have that $\Gamma \rhd Q \Longrightarrow \Gamma \rhd Q'$ with $\Gamma \models P' \cong_\sigma Q'$, and we conclude $(\Gamma \rhd P',\ \Gamma \rhd Q') \in \mathcal{S}$ as desired.
- $\Gamma \rhd P \xrightarrow{n(m)}_\sigma \Gamma \rhd P'$ with $\Gamma \vdash n : \sigma_1[T]$, $\Gamma \vdash m : T$ and $\sigma_1 \preceq \sigma$. Now, let be

$$C_\omega[\cdot_\Gamma] = \overline{n}\langle m\rangle.\overline{\omega}\langle\rangle \| [\cdot_\Gamma]$$

where ω is fresh and $\Gamma, \omega{:}\sigma[\,] \vdash_\sigma C_\omega[\cdot_\Gamma]$. Let $\Gamma' = \Gamma, \omega{:}\sigma[\,]$. Then $C_\omega[\cdot_\Gamma]$ is a (Γ'/Γ)-σ-context and $\Gamma' \rhd C_\omega[P] \xrightarrow{\tau} \Gamma' \rhd \overline{\omega}\langle\rangle | P'$ with $\Gamma' \models (\overline{\omega}\langle\rangle \,|\, P') \downarrow^\sigma_\omega$. Now, from $\Gamma \models P \cong_\sigma Q$ and the fact that $\cong_\sigma$ is σ-contextual, we have that there exists Q'' such that $\Gamma' \rhd C_\omega[Q] \Longrightarrow \Gamma' \rhd Q''$ with $\Gamma' \models \overline{\omega}\langle\rangle | P' \cong_\sigma Q''$ and $\Gamma' \models Q'' \downarrow^\sigma_\omega$. Then $Q'' = \overline{\omega}\langle\rangle | Q'$, and by Lemma 39, $\Gamma \models P' \cong_\sigma Q'$. Since $\Gamma' \models C_\omega[Q] \Downarrow^\sigma_\omega$, we have that $\Gamma \rhd Q \xRightarrow{n(m)}_\sigma \Gamma \rhd Q'$ and we conclude $(\Gamma \rhd P',\ \Gamma \rhd Q') \in \mathcal{S}$ as desired.
- $\Gamma \rhd P \xrightarrow{\mathsf{dec}_\delta\, n(m)}_\sigma \Gamma \rhd P'$ with $\Gamma \vdash n : \sigma_1[T]$, $\Gamma \vdash m : T$ and $\delta \prec \sigma_1 \preceq \sigma$. This case comes as the previous one using $C_\omega[\cdot_\Gamma] = \overline{\mathsf{dec}_\delta\, n}\langle m\rangle.\overline{\omega}\langle\rangle \| [\cdot_\Gamma]$, which still is a σ-context since $\Lambda(\Gamma(n)) \preceq \sigma$.
- $\Gamma \rhd P \xrightarrow{\overline{n}\langle m\rangle}_\sigma \Gamma \rhd P'$ with $\Gamma \vdash n : \sigma_1[T]$, $\Gamma \vdash m : T$ and $\sigma_1 \preceq \sigma$. Now, let be

$$C_\omega[\cdot_\Gamma] = [\cdot_\Gamma] | n(x{:}T).\mathsf{if}\, x = m\, \mathsf{then}\, \overline{\omega_1}\langle\rangle\, \mathsf{else}\, \overline{\omega_2}\langle\rangle$$

where ω_1, ω_2 are fresh names and $\Gamma, \omega_1 : \sigma[\,], \omega_2 : \sigma[\,] \vdash_\sigma C_\omega[\cdot_\Gamma]$. Let us consider $\Gamma' = \Gamma, \omega_1{:}\sigma[\,], \omega_2{:}\sigma[\,]$. Then $C_\omega[\cdot_\Gamma]$ is a (Γ'/Γ)-σ-context and we have $\Gamma' \rhd C_\omega[P] \Longrightarrow \Gamma' \rhd \overline{\omega_1}\langle\rangle | P'$ with $\Gamma' \models (\overline{\omega_1}\langle\rangle | P') \downarrow^\sigma_{\omega_1}$. Now, from $\Gamma \models P \cong_\sigma Q$ and the fact that $\cong_\sigma$ is σ-contextual, we have that there exists Q'' such that $\Gamma' \rhd C_\omega[Q] \Longrightarrow \Gamma' \rhd Q''$ with $\Gamma' \models \overline{\omega_1}\langle\rangle | P' \cong_\sigma Q''$ and $\Gamma' \models Q'' \downarrow^\sigma_{\omega_1}$. Then $Q'' = \overline{\omega_1}\langle\rangle | Q'$, and by Lemma 39, $\Gamma \models P' \cong_\sigma Q'$. Since $\Gamma' \models C_\omega[Q] \Downarrow^\sigma_{\omega_1}$, we have that $\Gamma \rhd Q \xRightarrow{\overline{n}\langle m\rangle}_\sigma \Gamma \rhd Q'$ and we conclude $(\Gamma \rhd P',\ \Gamma \rhd Q') \in \mathcal{S}$ as desired.
- $\Gamma \rhd P \xrightarrow{\overline{\mathsf{dec}_\delta\, n}\langle m\rangle}_\sigma \Gamma \rhd P'$ with $\Gamma \vdash n : \sigma_1[T]$, $\Gamma \vdash m : T$ and $\delta \prec \sigma_1 \preceq \sigma$. This case comes as the previous one using $C_\omega[\cdot_\Gamma] = [\cdot_\Gamma] | \mathsf{dec}_\delta\, n(x{:}T).\mathsf{if}\, x = m\, \mathsf{then}\, \overline{\omega_1}\langle\rangle\, \mathsf{else}\, \overline{\omega_2}\langle\rangle$, which still is a σ-context since $\Lambda(\Gamma(n)) \preceq \sigma$.
- $\Gamma \rhd P \xrightarrow{(\nu m{:}T)\,\overline{n}\langle m\rangle}_\sigma \Gamma, m{:}T \rhd P'$ comes from $P = (\nu m{:}T)P_1$, $\Gamma \vdash n : \sigma_1[T]$ with $\sigma_1 \preceq \sigma$, and $\Gamma, m{:}T \rhd P_1 \xrightarrow{\overline{n}\langle m\rangle}_\sigma \Gamma, m{:}T \rhd P'$. Let $\{p_1, \ldots, p_k\} = \{p | p \in \mathrm{fn}(P) \cup \mathrm{fn}(Q)$ and $\Gamma \vdash p : T\}$, and let

$$\begin{aligned} C_\omega[\cdot_\Gamma] = [\cdot_\Gamma] | n(x{:}T).\ \mathsf{if}\, x = p_1\, &\mathsf{then}\, \overline{\omega_1}\langle\rangle\, \mathsf{else} \\ &\mathsf{if}\, x = p_2\, \mathsf{then}\, \overline{\omega_1}\langle\rangle\, \mathsf{else} \\ &\quad\ldots \\ &\quad\mathsf{if}\, x = p_k\, \mathsf{then}\, \overline{\omega_1}\langle\rangle\, \mathsf{else}\, \overline{\omega_2}\langle x\rangle \end{aligned}$$

with ω_1, ω_2 fresh and $\Gamma, \omega_1{:}\sigma[\,], \omega_2{:}\sigma[T] \vdash_\sigma C_\omega[\cdot_\Gamma]$. Let $\Gamma' = \Gamma, \omega_1{:}\sigma[\,], \omega_2{:}\sigma[T]$. Then $C_\omega[\cdot_\Gamma]$ is a (Γ'/Γ)-σ-context and $\Gamma' \rhd C_\omega[P] \Longrightarrow \Gamma' \rhd (\nu m{:}T)(\overline{\omega_2}\langle m\rangle | P')$ with $\Gamma' \models (\nu m{:}T)(\overline{\omega_2}\langle m\rangle | P') \downarrow^\sigma_{\omega_2}$. Now, from $\Gamma \models P \cong_\sigma Q$ and the fact that $\cong_\sigma$ is σ-contextual, we have that there exists Q'' such that $\Gamma' \rhd C_\omega[Q] \Longrightarrow \Gamma' \rhd Q''$ with $\Gamma' \models (\nu m{:}T)(\overline{\omega_2}\langle m\rangle | P') \cong_\sigma Q''$ and $\Gamma' \models Q'' \downarrow^\sigma_{\omega_2}$. Since $\Gamma' \models C_\omega[Q] \Downarrow^\sigma_{\omega_2}$, we have that $\Gamma \rhd Q \xRightarrow{(\nu m{:}T)\,\overline{n}\langle m\rangle}_\sigma \Gamma, m{:}T \rhd Q'$ where $Q'' = (\nu m{:}T)(\overline{\omega_2}\langle m\rangle | Q')$. Since $\Gamma' \models (\nu m{:}T)(\overline{\omega_2}\langle m\rangle | P') \cong_\sigma (\nu m{:}T)(\overline{\omega_2}\langle m\rangle | Q')$, by Lemma 40 we have $\Gamma, m{:}T \models P' \cong_\sigma Q'$ and we can conclude that $(\Gamma, m{:}T \rhd P',\ \Gamma, m{:}T \rhd Q') \in \mathcal{S}$ as desired.

- $\Gamma \triangleright P \xrightarrow{(\nu m:T)\, n(m)}_{\sigma} \Gamma, m{:}T \triangleright P'$ where $\Gamma \vdash n : \sigma_1[T]$ and $\sigma_1 \preceq \sigma$. Now, let

$$C_\omega[\cdot_\Gamma] = [\cdot_\Gamma] | (\nu m : T)\ \overline{n}\langle m\rangle.\overline{\omega}\langle m\rangle$$

  where ω is fresh and $\Gamma, \omega{:}\sigma[T] \vdash_\sigma C_\omega[\cdot_\Gamma]$. Let $\Gamma' = \Gamma, \omega{:}\sigma[T]$. Then $C_\omega[\cdot_\Gamma]$ is a (Γ'/Γ)-σ-context and $\Gamma' \triangleright C_\omega[P] \xrightarrow{\tau} \Gamma' \triangleright (\nu m{:}T)(\overline{\omega}\langle m\rangle | P')$ with $\Gamma' \models (\nu m{:}T)(\overline{\omega}\langle m\rangle | P') \downarrow^{\sigma}_{\omega}$. Now, from $\Gamma \models P \cong_\sigma Q$ and the fact that $\cong_\sigma$ is σ-contextual, we have that there exists Q'' such that $\Gamma' \triangleright C_\omega[Q] \Longrightarrow \Gamma' \triangleright Q''$ with $\Gamma' \models (\nu m{:}T)(\overline{\omega}\langle m\rangle | P') \cong_\sigma Q''$ and $\Gamma' \models Q'' \downarrow^{\sigma}_{\omega}$. Since $\Gamma' \models C_\omega[Q] \Downarrow^{\sigma}_{\omega}$, we have that $\Gamma \triangleright Q \overset{(\nu m:T)\, n(m)}{\Longrightarrow}_{\sigma} \Gamma, m{:}T \triangleright Q'$ where $Q'' = (\nu m{:}T)(Q' | \overline{\omega}\langle m\rangle)$. Since $\Gamma' \models (\nu m{:}T)(\overline{\omega}\langle m\rangle | P') \cong_\sigma (\nu m{:}T)(\overline{\omega}\langle m\rangle | Q')$, by Lemma 40 we have $\Gamma, m{:}T \models P' \cong_\sigma Q'$ and we conclude $(\Gamma, m{:}T \triangleright P',\ \Gamma, m{:}T \triangleright Q') \in \mathcal{S}$ as desired.
- The cases where $\Gamma \triangleright P$ performs a declassified bound action are similar to the previous two cases; they rely on corresponding contexts which use a declassified communication over the channel n. □

Proof of Theorem 9 Let $\sigma \in \Sigma$, Γ be a type environment and P, Q be processes such that $\Gamma \vdash P, Q$. Then $\Gamma \models P \cong_\sigma Q$ iff $\Gamma \models P \approx_\sigma Q$.

Proof. The proof of $\approx_\sigma \subseteq \cong_\sigma$ comes by Propositions 36, 37 and 38. The converse comes by Proposition 41. □

B. Proofs omitted from Section 4

Proof of Proposition 20 If $\mathcal{R}$ is a bisimulation up to σ-contexts and up to $\approx_\sigma$, then $\mathcal{R} \subseteq \approx_\sigma$.

Proof. We define the relation $\mathcal{S}$ as the smallest relation such that:

(1) $\Gamma \models P\ \mathcal{R}\ Q$ implies $\Gamma \models P\ \mathcal{S}\ Q$.
(2) $\Gamma \models P \approx_\sigma R$, $\Gamma \models R\ \mathcal{S}\ U$ and $\Gamma \models U \approx_\sigma Q$ imply $\Gamma \models P\ \mathcal{S}\ Q$.
(3) $\Gamma \models P\ \mathcal{S}\ Q$ implies $\Gamma' \models C[P]\ \mathcal{S}\ C[Q]$ for every (Γ'/Γ)-σ-context $C[\cdot_\Gamma]$.

We prove by induction on its definition that $\mathcal{S}$ is a bisimulation on σ-actions. This will assure the soundness of $\mathcal{R}$ since $\mathcal{R} \subseteq \mathcal{S} \subseteq \approx_\sigma$. The symmetry of $\mathcal{S}$ comes by induction and the symmetry of $\mathcal{R}$.

- Let $\Gamma \models P\,\mathcal{S}\,Q$ because $\Gamma \models P\,\mathcal{R}\,Q$. Assume $\Gamma \triangleright P \xrightarrow{\alpha}_\sigma \Gamma' \triangleright P'$, then from $\Gamma \models P\,\mathcal{R}\,Q$ we know that $\exists Q', Q'', P'', C[\cdot_{\Gamma''}]$ such that $\Gamma' \vdash_\sigma C[\cdot_{\Gamma''}]$ and $\Gamma \triangleright Q \overset{\hat{\alpha}}{\Longrightarrow}_\sigma \Gamma' \triangleright Q'$ with $\Gamma' \models P' \approx_\sigma C[P'']$, $\Gamma' \models Q' \approx_\sigma C[Q'']$, and $\Gamma'' \models P''\ \mathcal{R}\ Q''$. Now, by definition of $\mathcal{S}$ (1) we have $\Gamma'' \models P''\ \mathcal{S}\ Q''$, then by (3) $\Gamma' \models C[P'']\ \mathcal{S}\ C[Q'']$, and by (2) we conclude $\Gamma' \models P'\ \mathcal{S}\ Q'$.
- Let $\Gamma \models P\,\mathcal{S}\,Q$ because $\Gamma \models P \approx_\sigma R$, $\Gamma \models R\,\mathcal{S}\,U$ and $\Gamma \models U \approx_\sigma Q$. Assume $\Gamma \triangleright P \xrightarrow{\alpha}_\sigma \Gamma' \triangleright P'$, then from $\Gamma \models P \approx_\sigma R$ we know that $\exists R'$ such that $\Gamma \triangleright R \overset{\hat{\alpha}}{\Longrightarrow}_\sigma \Gamma' \triangleright R'$ with $\Gamma' \models P' \approx_\sigma R'$. Now, from $\Gamma \models R\,\mathcal{S}\,U$, by induction we have that $\Gamma \triangleright U \overset{\hat{\alpha}}{\Longrightarrow}_\sigma \Gamma' \triangleright U'$ with $\Gamma' \models R'\,\mathcal{S}\,U'$, and from $\Gamma \models U \approx_\sigma Q$ we have $\Gamma \triangleright Q \overset{\hat{\alpha}}{\Longrightarrow}_\sigma \Gamma' \triangleright Q'$ with $\Gamma' \models U' \approx_\sigma Q'$. Then, by definition of $\mathcal{S}$ (2) we conclude $\Gamma' \models P'\,\mathcal{S}\,Q'$.
- Let $\Gamma' \models C[P]\ \mathcal{S}\ C[Q]$ because $\Gamma' \vdash_\sigma C[\cdot_\Gamma]$ and $\Gamma \models P\,\mathcal{S}\,Q$. Assume that $\Gamma' \triangleright C[P] \xrightarrow{\alpha}_\sigma \Gamma'' \triangleright U$, we have to show that $\Gamma' \triangleright C[Q] \overset{\hat{\alpha}}{\Longrightarrow}_\sigma \Gamma'' \triangleright V$ with $\Gamma'' \models U\,\mathcal{S}\,V$. The proof proceeds by induction on the definition of σ-contexts (Definition 2).

  . $C[\cdot_\Gamma] = [\cdot_\Gamma]$, with $\Gamma' = \Gamma$ and $\Gamma \triangleright P \xrightarrow{\alpha}_\sigma \Gamma'' \triangleright U$. Then from $\Gamma \models P\,\mathcal{S}\,Q$, by induction hypothesis on the definition of $\mathcal{S}$, we have $\Gamma \triangleright Q \overset{\hat{\alpha}}{\Longrightarrow}_\sigma \Gamma'' \triangleright V$ and $\Gamma'' \models U\,\mathcal{S}\,V$ as desired.
  . $C[\cdot_\Gamma] = (\nu n{:}T)C'[\cdot_\Gamma]$. Note that from $\Gamma \models P\,\mathcal{S}\,Q$, by definition of $\mathcal{S}$ (3) we have $\Gamma', n{:}T \models C'[P]\ \mathcal{S}\ C'[Q]$. The inductive hypothesis on σ-contexts states that if $\Gamma', n{:}T \triangleright C'[P] \xrightarrow{\alpha}_\sigma \Gamma'', n{:}T \triangleright U'$ then $\Gamma', n{:}T \triangleright$

$C'[Q] \overset{\hat\alpha}{\Longrightarrow}_\sigma \Gamma'', n{:}T \rhd V'$ with $\Gamma'', n{:}T \vDash U' \mathcal{S} V'$.

Assume $\Gamma' \rhd (\boldsymbol{\nu} n{:}T)C'[P] \xrightarrow{\alpha}_\sigma \Gamma'' \rhd U$, this must have been derived in one of the following ways:

by (Res) from $\Gamma', n{:}T \rhd C'[P] \xrightarrow{\alpha}_\sigma \Gamma'', n{:}T \rhd U'$ with $U = (\boldsymbol{\nu} n{:}T)U'$ since $n \notin \mathrm{fn}(\alpha) \cup \mathrm{bn}(\alpha)$. Now, by induction on σ-contexts we have that $\Gamma', n{:}T \rhd C'[Q] \overset{\hat\alpha}{\Longrightarrow}_\sigma \Gamma'', n{:}T \rhd V'$ with $\Gamma'', n{:}T \vDash U' \mathcal{S} V'$. Then by (Res) we also have $\Gamma' \rhd (\boldsymbol{\nu} n{:}T)C'[Q] \overset{\hat\alpha}{\Longrightarrow}_\sigma \Gamma'' \rhd (\boldsymbol{\nu} n{:}T)V'$ and by definition of $\mathcal{S}$ (3), from $\Gamma'', n{:}T \vDash U' \mathcal{S} V'$ we can conclude that $\Gamma'' \vDash (\boldsymbol{\nu} n{:}T)U' \mathcal{S} (\boldsymbol{\nu} n{:}T)V'$ as desired.

by (Open) from $\Gamma', n{:}T \rhd C'[P] \xrightarrow{\overline{m}\langle n\rangle}_\sigma \Gamma', n{:}T \rhd U$ with $\Gamma'' = \Gamma', n{:}T, \alpha = (\boldsymbol{\nu} n{:}T)\,\overline{m}\langle n\rangle$. Now, by induction on σ-contexts we have that $\Gamma', n{:}T \rhd C'[Q] \overset{\overline{m}\langle n\rangle}{\Longrightarrow}_\sigma \Gamma', n{:}T \rhd V$ with $\Gamma', n{:}T \vDash U \mathcal{S} V$, and we conclude $\Gamma' \rhd (\boldsymbol{\nu} n{:}T)C'[Q] \overset{\alpha}{\Longrightarrow}_\sigma \Gamma', n{:}T \rhd V$ by (Open).

by (Dec Open) from $\Gamma', n{:}T \rhd C'[P] \xrightarrow{\overline{\mathsf{dec}_\delta\, m\langle n\rangle}}_\sigma \Gamma', n{:}T \rhd U$ where $\Gamma'' = \Gamma', n{:}T$ and $\alpha = (\boldsymbol{\nu} n{:}T)\,\overline{\mathsf{dec}_\delta\, m\langle n\rangle}$. Now, by induction on σ-contexts we have that $\Gamma', n{:}T \rhd C'[Q] \overset{\overline{\mathsf{dec}_\delta\, m\langle n\rangle}}{\Longrightarrow}_\sigma \Gamma', n{:}T \rhd V$ with $\Gamma', n{:}T \vDash U \mathcal{S} V$. We conclude $\Gamma' \rhd (\boldsymbol{\nu} n{:}T)C'[Q] \overset{\alpha}{\Longrightarrow}_\sigma \Gamma', n{:}T \rhd U$ by (Dec Open).

- $C[\cdot_\Gamma] = C'[\cdot_\Gamma]|R$ or $C[\cdot_\Gamma] = R|C'[\cdot_\Gamma]$ where $\Gamma' \vdash_\sigma R$. The proof of the two cases is similar, therefore we show only the first one. Note that from $\Gamma \vDash P \mathcal{S} Q$, by definition of $\mathcal{S}$ (3) we have $\Gamma' \vDash C'[P] \mathcal{S} C'[Q]$. The inductive hypothesis on σ-contexts states that if $\Gamma' \rhd C'[P] \xrightarrow{\alpha}_\sigma \Gamma'' \rhd U'$ then $\Gamma' \rhd C'[Q] \overset{\hat\alpha}{\Longrightarrow}_\sigma \Gamma'' \rhd V'$ with $\Gamma'' \vDash U' \mathcal{S} V'$. Let $\Gamma' \rhd C[P] \xrightarrow{\alpha}_\sigma \Gamma'' \rhd U$, this must have been derived in one of the following ways:

  . $\alpha \neq \tau$ In this case, the hypothesis must come by (Par) from either

  $\Gamma' \rhd C'[P] \xrightarrow{\alpha}_\sigma \Gamma'' \rhd U'$ with $U = U'|R$. By induction on σ-contexts we have $\Gamma' \rhd C'[Q] \overset{\hat\alpha}{\Longrightarrow}_\sigma \Gamma'' \rhd V'$ with $\Gamma'' \vDash U' \mathcal{S} V'$. By an application of the rule (Par) we have $\Gamma' \rhd C[Q] \overset{\hat\alpha}{\Longrightarrow}_\sigma \rhd \Gamma'' V'|R$. Now, from $\Gamma'' \vDash U' \mathcal{S} V'$ and $\Gamma'' \vdash_\sigma R$, which comes from $\Gamma' \vdash_\sigma R$ and $\Gamma' \subseteq \Gamma''$, we conclude $\Gamma'' \vDash (U'|R)\ \mathcal{S}\ (V'|R)$ by definition of $\mathcal{S}$ (3).
  $\Gamma' \rhd R \xrightarrow{\alpha}_\sigma \Gamma'' \rhd R'$ with $U = C'[P]|R'$. By Lemma 35 we have $\Gamma'' \vdash_\sigma R'$. By an application of the rule (Par) we also have $\Gamma' \rhd R|C'[Q] \xrightarrow{\alpha}_\sigma \Gamma'' \rhd R'|C'[Q]$. Now, from $\Gamma'' \vDash C'[P] \mathcal{S} C'[Q]$ we conclude $\Gamma'' \vDash (C'[P]|R')\ \mathcal{S}\ (C'[Q]|R')$ by definition of $\mathcal{S}$ (3).

  . $\alpha = \tau$. In this case, the hypothesis must have been derived in one of the following ways:

  by (Par). This case follows similarly to the previous one.
  by (Comm) from $C'[P] \xrightarrow{\overline{n}\langle m\rangle} U'$ and $R \xrightarrow{n(m)} R'$ (or viceversa, which follows similarly) with $U = U'|R'$. From $\Gamma' \vdash_\sigma R$ we have $\Lambda(\Gamma'(n)) \preceq \sigma$, hence by Propositions 6 and 7 $\Gamma' \rhd C'[P] \xrightarrow{\overline{n}\langle m\rangle}_\sigma \Gamma' \rhd U'$ and $\Gamma' \rhd R \xrightarrow{n(m)}_\sigma \Gamma' \rhd U''$. By induction on σ-contexts we have that $\Gamma' \rhd C'[Q] \overset{\overline{n}\langle m\rangle}{\Longrightarrow}_\sigma \Gamma' \rhd V'$ with $\Gamma' \vDash U' \mathcal{S} V'$, hence $\Gamma' \rhd C[Q] \Longrightarrow \Gamma' \rhd V'|R'$. From $\Gamma' \vDash U' \mathcal{S} V'$, since $\Gamma' \vdash_\sigma R'$, by definition of $\mathcal{S}$ (3) we conclude $\Gamma' \vDash (U'|R')\ \mathcal{S}\ (V'|R')$.
  by (Close) from $C'[P] \xrightarrow{(\nu m:T)\,\overline{n}\langle m\rangle} U'$ and $R \xrightarrow{n(m)} R'$ (or viceversa, which follows similarly) with $U = (\boldsymbol{\nu} m{:}T)(U'|R')$. From the fact that $\Gamma' \vdash_\sigma R$ we have $\Lambda(\Gamma'(n)) \preceq \sigma$ and by Propositions 6 and 7, we obtain $\Gamma' \rhd C'[P] \xrightarrow{(\nu m:T)\,\overline{n}\langle m\rangle}_\sigma \Gamma', m{:}T \rhd U'$ and $\Gamma' \rhd R \xrightarrow{(\nu m:T)\,n(m)}_\sigma \Gamma', m{:}T \rhd R'$. By induction on σ-contexts we obtain that it holds $\Gamma' \rhd C'[Q] \overset{(\nu m:T)\,\overline{n}\langle m\rangle}{\Longrightarrow}_\sigma \Gamma', m{:}T \rhd V'$ with $\Gamma', m{:}T \vDash U' \mathcal{S} V'$, hence $\Gamma' \rhd C[Q] \Longrightarrow \Gamma' \rhd (\boldsymbol{\nu} m{:}T)(V'|R')$. From $\Gamma' m{:}T \vDash U' \mathcal{S} V'$, since $\Gamma' \vdash_\sigma R'$, by definition of $\mathcal{S}$ (3) we conclude $\Gamma' \vDash (\boldsymbol{\nu} m{:}T)(U'|R')\ \mathcal{S}\ (\boldsymbol{\nu} m{:}T)(V'|R')$.

by (Dec Comm) from $C'[P] \xrightarrow{\overline{\mathsf{dec}_\delta\, n\langle m\rangle}} U'$ and $R \xrightarrow{\mathsf{dec}_\delta\, n(m)} R'$ (or viceversa, that follows similarly) with $U = U'|R'$. From $\Gamma' \vdash_\sigma R$ we have $\Lambda(\Gamma'(n)) \preceq \sigma$, hence by Propositions 6 and 7, $\Gamma' \rhd C'[P] \xrightarrow{\overline{\mathsf{dec}_\delta\, n\langle m\rangle}}_\sigma \Gamma' \rhd U'$ and $\Gamma' \rhd R \xrightarrow{\mathsf{dec}_\delta\, n(m)}_\sigma \Gamma' \rhd R'$. The proof then follows as above.
by (Dec Close) from $C'[P] \xrightarrow{(\nu m{:}T)\,\overline{\mathsf{dec}_\delta\, n\langle m\rangle}} U'$ and $R \xrightarrow{\mathsf{dec}_\delta\, n(m)} R'$ (or viceversa, which follows similarly) with $U = (\boldsymbol{\nu} m{:}T)(U'|R')$. From $\Gamma' \vdash_\sigma R$ we have $\Lambda(\Gamma'(n)) \preceq \sigma$ and by Propositions 6 and 7, $\Gamma' \rhd C'[P] \xrightarrow{(\nu m{:}T)\,\overline{\mathsf{dec}_\delta\, n\langle m\rangle}}_\sigma \Gamma', m{:}T \rhd U'$ and $\Gamma' \rhd R \xrightarrow{(\nu m{:}T)\,\mathsf{dec}_\delta\, n(m)}_\sigma \Gamma', m{:}T \rhd R'$. The proof then follows as above. □

Proof of Proposition 23 If $\Gamma \models P \dot{\approx}_\sigma P$, then for all $\Gamma' \rhd P'$ such that $\Gamma \rhd P \rightsquigarrow \Gamma' \rhd P'$, it holds $\Gamma' \models P' \dot{\approx}_\sigma P'$.

Proof. By induction on the length of the derivation $\Gamma \rhd P \rightsquigarrow \Gamma' \rhd P'$. The base case, where the length is 0 is immediate. For the inductive case, assume $\Gamma \rhd P \rightsquigarrow \Gamma_1 \rhd P_1 \xrightarrow{\alpha}_\delta \Gamma' \rhd P'$; we distinguish the following cases:

- $\Gamma_1 \rhd P_1 \xrightarrow{\alpha}_\sigma \Gamma' \rhd P'$. By induction we know $\Gamma_1 \models P_1 \dot{\approx}_\sigma P_1$, then by definition of $\dot{\approx}_\sigma$ we have $\Gamma_1 \rhd P_1 \overset{\alpha}{\Longrightarrow}_\sigma \Gamma' \rhd P_2$ and $\Gamma' \models P' \dot{\approx}_\sigma P_2$. Then by symmetry of $\dot{\approx}_\sigma$ we also have $\Gamma' \models P_2 \dot{\approx}_\sigma P'$, and by transitivity $\Gamma' \models P' \dot{\approx}_\sigma P'$.
- $\Gamma_1 \rhd P_1 \xrightarrow{\alpha}{}^\sigma \Gamma' \rhd P'$ with $\alpha \in \{n(m), \overline{n}\langle m\rangle\}$ and $\Gamma' = \Gamma_1$. By induction we know $\Gamma_1 \models P_1 \dot{\approx}_\sigma P_1$, then by definition of $\dot{\approx}_\sigma$ we have $\Gamma_1 \rhd P_1 \Longrightarrow \Gamma_1 \rhd P_2$ and $\Gamma_1 \models P' \dot{\approx}_\sigma P_2$. Then by symmetry of $\dot{\approx}_\sigma$ we also have $\Gamma_1 \models P_2 \dot{\approx}_\sigma P'$, and by transitivity $\Gamma_1 \models P' \dot{\approx}_\sigma P'$.
- $\Gamma_1 \rhd P_1 \xrightarrow{\alpha}{}^\sigma \Gamma' \rhd P'$ with $\alpha \in \{(\boldsymbol{\nu} m{:}T)\, n(m), (\boldsymbol{\nu} m{:}T)\, \overline{n}\langle m\rangle\}$ and $\Gamma' = \Gamma, m{:}T$. By induction we know $\Gamma_1 \models P_1 \dot{\approx}_\sigma P_1$, then by definition of $\dot{\approx}_\sigma$ we have $\Gamma_1 \rhd P_1 \Longrightarrow \Gamma_1 \rhd P_2$, $\Gamma_1 \models P_2 \dot{\approx}_\sigma (\boldsymbol{\nu} m{:}T)P'$ and $\Gamma' \models P' \dot{\approx}_\sigma P'$ as desired.
- $\Gamma_1 \rhd P_1 \xrightarrow{\alpha}{}^\sigma \Gamma' \rhd P'$ where α is a declassified action. By induction we know $\Gamma_1 \models P_1 \dot{\approx}_\sigma P_1$, then by definition of $\dot{\approx}_\sigma$ we have $\Gamma' \models P' \dot{\approx}_\sigma P'$ as desired. □

The next lemma is used in the proof of of Lemma 25.

Lemma 42. *Let P be a process and Γ be a type environment. Let $\Gamma \rhd (\boldsymbol{\nu} m{:}T)P \rightsquigarrow \Gamma' \rhd P'$, then one of the following two cases hold:*

(1) $P' = (\boldsymbol{\nu} m{:}T)P''$ *and* $\Gamma, m{:}T \rhd P \rightsquigarrow \Gamma', m{:}T \rhd P''$
(2) $\Gamma' = \Gamma'', m{:}T$ *and* $\Gamma, m{:}T \rhd P \rightsquigarrow \Gamma'', m{:}T \rhd P'$

Proof. By induction on the length of $\Gamma \rhd (\boldsymbol{\nu} m{:}T)P \rightsquigarrow \Gamma' \rhd P'$. If the length is one, then $\Gamma \rhd (\boldsymbol{\nu} m{:}T)P \xrightarrow{\alpha}_\sigma \Gamma' \rhd P'$ must have been derived in one of the following two ways:

- by rule (Res) from $\Gamma, m{:}T \rhd P \xrightarrow{\alpha}_\sigma \Gamma', m{:}T \rhd P''$ where $P' = (\boldsymbol{\nu} m{:}T)P''$, which is case 1.
- by rule (Open) from $\Gamma, m{:}T \rhd P \xrightarrow{\overline{n}\langle m\rangle}_\sigma \Gamma, m{:}T \rhd P'$ where $\Gamma' = \Gamma, m{:}T$ and $\alpha = (\boldsymbol{\nu} m{:}T)\, \overline{n}\langle m\rangle$, which is case 2.

Let now consider the inductive case where $\Gamma \rhd (\boldsymbol{\nu} m{:}T)P \rightsquigarrow \Gamma_1 \rhd P_1 \xrightarrow{\alpha}_\sigma \Gamma' \rhd P'$. By induction we have two cases:

- $P_1 = (\boldsymbol{\nu} m{:}T)P_2$ and $\Gamma, m{:}T \rhd P \rightsquigarrow \Gamma_1, m{:}T \rhd P_2$. We distinguish two subcases depending on the rule that gave $\Gamma_1 \rhd P_1 \xrightarrow{\alpha}_\sigma \Gamma' \rhd P'$:

  . it comes by (Res) from $\Gamma_1, m{:}T \rhd P_2 \xrightarrow{\alpha}_\sigma \Gamma', m{:}T \rhd P''$ and $P' = (\boldsymbol{\nu} m{:}T)P''$. Hence $\Gamma, m{:}T \rhd P \rightsquigarrow \rhd \Gamma', m{:}TP''$ which is case 1.

. it comes by (Open) from $\Gamma_1, m{:}T \triangleright P_2 \xrightarrow{\overline{n}\langle m\rangle}_\sigma \Gamma_1, m{:}T \triangleright P'$ with $\alpha = (\nu m{:}T)\,\overline{n}\langle m\rangle$ and $\Gamma' = \Gamma_1, m{:}T$. Hence $\Gamma, m{:}T \triangleright P \rightsquigarrow \Gamma_1, m{:}T \triangleright P'$ which is case 2.

- $\Gamma_1 = \Gamma_2, m{:}T$ and $\Gamma, m{:}T \triangleright P \rightsquigarrow \Gamma_2, m{:}T \triangleright P_1$. In this case we have $\Gamma, m{:}T \triangleright P \rightsquigarrow \xrightarrow{\alpha}_\sigma \Gamma' \triangleright P'$ with $\Gamma' = \Gamma'', m{:}T$ for some Γ'', which is case 2. □

Proof of Lemma 25 Let $\sigma \in \Sigma$, P be a process and $\Gamma, m : T$ be a type environment such that $\Gamma, m : T \vdash P$. If $\Gamma, m : T \triangleright P \in \mathcal{CR}(\cong_\sigma)$ then $\Gamma \triangleright (\nu m : T)P \in \mathcal{CR}(\cong_\sigma)$.

Proof. By Theorem 21 it is sufficient to prove that if $\Gamma, m : T \triangleright P \in \mathcal{W}(\cong_\sigma)$ then $\Gamma \triangleright (\nu m : T)P \in \mathcal{W}(\cong_\sigma)$.
Let $\Gamma' \triangleright P'$ be a configuration such that $\Gamma \triangleright (\nu m{:}T)P \rightsquigarrow \Gamma' \triangleright P'$, then we prove the following two items:

(1) if $\Gamma' \triangleright P' \xrightarrow{\alpha}^\sigma \Gamma' \triangleright P_2$ with $\alpha \in \{\overline{p}\langle q\rangle,\ p(q)\}$, then $\exists P_3$ s.t. $\Gamma' \triangleright P' \Longrightarrow \Gamma' \triangleright P_3$ and $\Gamma' \models P_2 \cong_\sigma P_3$.
(2) if $\Gamma' \triangleright P' \xrightarrow{\alpha}^\sigma \Gamma', q{:}T' \triangleright P_2$ with $\alpha \in \{(\nu q{:}T')\,\overline{p}\langle q\rangle,\ (\nu q{:}T')\,p(q)\}$, then $\exists P_3$ s.t. $\Gamma' \triangleright P' \Longrightarrow \Gamma' \triangleright P_3$ and $\Gamma' \models P_3 \cong_\sigma (\nu q{:}T')P_2$.

By Lemma 42 we distinguish two cases:

- $P' = (\nu m{:}T)P''$ and $\Gamma, m{:}T \triangleright P \rightsquigarrow \Gamma', m{:}T \triangleright P''$. Let us prove the two items in the definition of $\mathcal{W}(\cong_\sigma)$:

  (1) $\Gamma' \triangleright P' \xrightarrow{\alpha}^\sigma \Gamma' \triangleright P_2$ with $\alpha \in \{\overline{p}\langle q\rangle, p(q)\}$. Since $P' = (\nu m{:}T)P''$, the action comes by (Res) from $\Gamma', m{:}T \triangleright P'' \xrightarrow{\alpha}^\sigma \Gamma', m{:}T \triangleright P'''$ where $P_2 = (\nu m{:}T)P'''$ and $m \notin \mathrm{fn}(\alpha) \cup \mathrm{bn}(\alpha)$. From $\Gamma, m{:}T \triangleright P \rightsquigarrow \Gamma', m{:}T \triangleright P''$ and $\Gamma, m{:}T \triangleright P \in \mathcal{W}(\cong_\sigma)$, there exists P^* such that $\Gamma', m{:}T \triangleright P'' \Longrightarrow \Gamma', m{:}T \triangleright P^*$ with $\Gamma', m{:}T \models P^* \cong_\sigma P'''$. Since $\cong_\sigma$ is a congruence, we also have $\Gamma' \models (\nu m{:}T)P^* \cong_\sigma (\nu m{:}T)P'''$, and we conclude observing that by (Res) we have $\Gamma' \triangleright P' \Longrightarrow \Gamma' \triangleright (\nu m{:}T)P^*$.
  (2) $\Gamma' \triangleright P' \xrightarrow{\alpha}^\sigma \Gamma', q{:}T' \triangleright P_2$ with $\alpha \in \{(\nu q{:}T')\,\overline{p}\langle q\rangle, (\nu q{:}T')\,p(q)\}$. We distinguish two subcases:

  . $q : T' = m : T, \alpha = (\nu m{:}T)\,\overline{p}\langle m\rangle$ and $\Gamma', m{:}T \triangleright P'' \xrightarrow{\overline{p}\langle m\rangle}^\sigma \Gamma', m{:}T \triangleright P_2$. Since $\Gamma, m{:}T \triangleright P \rightsquigarrow \Gamma', m{:}T \triangleright P''$ and $\Gamma, m{:}T \triangleright P \in \mathcal{W}(\cong_\sigma)$, we have that there exists P^* such that $\Gamma', m{:}T \triangleright P'' \Longrightarrow \Gamma', m{:}T \triangleright P^*$ and $\Gamma', m{:}T \models P^* \cong_\sigma P_2$. Then by rule (Res) we also have $\Gamma' \triangleright P' \Longrightarrow \Gamma' \triangleright (\nu m{:}T)P^*$ and $\Gamma' \models (\nu m{:}T)P^* \cong_\sigma (\nu m{:}T)P_2$ since $\cong_\sigma$ is a congruence.
  . $q : T' \neq m : T$, then $\Gamma' \triangleright P' \xrightarrow{\alpha}^\sigma \Gamma', q{:}T' \triangleright P_2$ comes by (Res) from $\Gamma', m{:}T \triangleright P'' \xrightarrow{\alpha}^\sigma \Gamma', q{:}T', m{:}T \triangleright P'''$ with $P_2 = (\nu m{:}T)P'''$. Now from $\Gamma, m{:}T \triangleright P \rightsquigarrow \Gamma', m{:}T \triangleright P''$ and $\Gamma, m{:}T \triangleright P \in \mathcal{W}(\cong_\sigma)$ we have that there exists P^* such that $\Gamma', m{:}T \triangleright P'' \Longrightarrow \Gamma', m{:}T \triangleright P^*$ and $\Gamma', m{:}T \models P^* \cong_\sigma (\nu q{:}T')P'''$. Then by rule (Res) we also have $\Gamma' \triangleright (\nu m{:}T)P'' \Longrightarrow \Gamma' \triangleright (\nu m{:}T)P^*$ and $\Gamma' \models (\nu m{:}T)P^* \cong_\sigma (\nu m{:}T)(\nu q{:}T')P''' = (\nu q{:}T')P_2$ since $\cong_\sigma$ is a congruence.

- $\Gamma' = \Gamma'', m{:}T$ and $\Gamma, m{:}T \triangleright P \rightsquigarrow \Gamma' \triangleright P'$. In this case we conclude using the hypothesis $\Gamma, m{:}T \triangleright P \in \mathcal{W}(\cong_\sigma)$. □

B.1. Proofs of compositionality of controlled information release

Lemma 43 (*Strengthening*). *Let be $\Gamma \triangleright P \rightsquigarrow \Gamma' \triangleright P'$ and $\Gamma_1 \subseteq \Gamma$ such that $\Gamma_1 \vdash P$. Then $\Gamma_1 \triangleright P \rightsquigarrow \Gamma'_1 \triangleright P'$ with $\Gamma'_1 \subseteq \Gamma'$.*

Proof. The proof proceeds by induction on the lenght of $\Gamma \triangleright P \rightsquigarrow \Gamma' \triangleright P'$, and it is not difficult. □

Lemma 44 (*Weakening*). *Let P, Q be two processes and Γ and Γ' type environments.*

(1) $\Gamma \triangleright P \rightsquigarrow \Gamma_1 \triangleright P'$, $\Gamma \subseteq \Gamma'$ *imply* $\Gamma' \triangleright P \rightsquigarrow \Gamma'_1 \triangleright P'$ *with* $\Gamma_1 \subseteq \Gamma'_1$.
(2) $\Gamma \models P \approx_\sigma Q$ *and* $\Gamma \subseteq \Gamma'$ *imply* $\Gamma' \models P \approx_\sigma Q$.
(3) $\Gamma \triangleright P \in \mathcal{W}(\cong_\sigma)$ *and* $\Gamma \subseteq \Gamma'$ *imply* $\Gamma' \triangleright P \in \mathcal{W}(\cong_\sigma)$.

Proof. Proof of (1) proceeds by induction on the derivation of $\Gamma \triangleright P \leadsto \Gamma_1 \triangleright P'$. Proof of (2). It is sufficient to show that the relation

$$\mathcal{S} = \{(\Gamma' \triangleright P,\ \Gamma' \triangleright Q) \mid \Gamma \subseteq \Gamma' \text{ such that } \Gamma \models P \approx_\sigma Q\}$$

is a bisimulation on σ-actions, which is not difficult using (1) and Lemma 43. Proof of (3) is not difficult, using (1), (2) and Lemma 43.

Let $\Gamma \triangleright P \leadsto_{0/1} \Gamma' \triangleright P'$ mean that the configuration $\Gamma' \triangleright P'$ is reachable from $\Gamma \triangleright P$ in zero or one step. □

Lemma 45. *Let P, Q be processes and let Γ be a type environment. If $\Gamma \triangleright P|Q \xrightarrow{\alpha}_\delta \Gamma' \triangleright R$, then one of the following cases holds*:

(1) $\alpha = \tau, R = P'|Q'$ *with* $\Gamma \triangleright P \leadsto_{0/1} \Gamma \triangleright P'$ *and* $\Gamma \triangleright Q \leadsto_{0/1} \Gamma \triangleright Q'$.
(2) $\alpha = \tau, R = (\nu m{:}T)(P'|Q')$ *with* $\Gamma \triangleright P \leadsto_{0/1} \Gamma, m{:}T \triangleright P'$ *and* $\Gamma \triangleright Q \leadsto_{0/1} \Gamma, m{:}T \triangleright Q'$.
(3) $\alpha \neq \tau, \Gamma \triangleright P \xrightarrow{\alpha}_\delta \Gamma' \triangleright P'$ *and* $R = P'|Q$.
(4) $\alpha \neq \tau, \Gamma \triangleright Q \xrightarrow{\alpha}_\delta \Gamma' \triangleright Q'$ *and* $R = P|Q'$.

Proof. We distinguish the cases in which the hypothesis $\Gamma \triangleright P|Q \xrightarrow{\alpha}_\delta \Gamma' \triangleright R$ must have been derived.

- $\alpha = \tau$ and the hypothesis comes by an application of the rule (Comm) from $P \xrightarrow{\overline{n}\langle m\rangle} P'$ and $Q \xrightarrow{n(m)} Q'$ (or viceversa, which follows similarly). By Propositions 6 and 7 we have $\Gamma \triangleright P \xrightarrow{\overline{n}\langle m\rangle}_{\delta_1} \Gamma \triangleright P'$ and $\Gamma \triangleright Q \xrightarrow{n(m)}_{\delta_1} \Gamma \triangleright Q'$ for some δ_1, which corresponds to case 1.
- $\alpha = \tau$ and the hypothesis comes by an application of the rule (Dec Comm). This case is similar to the previous one.
- $\alpha = \tau$ and the hypothesis comes by an application of the rule (Close) from $P \xrightarrow{(\nu m:T)\,\overline{n}\langle m\rangle} P'$ and $Q \xrightarrow{n(m)} Q'$ (or viceversa, which follows similarly). By Propositions 6 and 7 we have $\Gamma \triangleright P \xrightarrow{(\nu m:T)\,\overline{n}\langle m\rangle}_{\delta_1} \Gamma, m : T \triangleright P'$ and $\Gamma \triangleright Q \xrightarrow{(\nu m:T)\,n(m)}_{\delta_1} \Gamma, m : T \triangleright Q'$, for some δ_1, which corresponds to case 2.
- $\alpha = \tau$ and the hypothesis comes by an application of the rule (Dec Close). This case is similar to the previous one.
- $\alpha \neq \tau$ and the hypothesis comes by an application of the rule (Par). Then one of the case 3. or 4. applies. □

Lemma 46. *Let P, Q be processes and Γ be a type environment.*
If $\Gamma \triangleright (\nu n_1 : T_1, \ldots, n_k : T_k)(P|Q) \xrightarrow{\alpha}_\delta \Gamma' \triangleright R$, then one of the following cases holds, where $\Delta = n_1 : T_1, \ldots, n_k : T_k$.

(*a*) $\alpha = \tau$ *and* $R = (\nu n_1 : T_1, \ldots, n_k : T_k)(P'|Q')$ *with* $\Gamma, \Delta \triangleright P \leadsto_{0/1} \Gamma', \Delta \triangleright P'$ *and* $\Gamma, \Delta \triangleright Q \leadsto_{0/1} \Gamma', \Delta \triangleright Q'$.
(*b*) $\alpha = \tau$ *and* $R = (\nu n_1 : T_1, \ldots, n_k : T_k)(\nu n_{k+1} : T_{k+1})(P'|Q')$ *with* $\Gamma, \Delta \triangleright P \leadsto_{0/1} \Gamma', \Delta, n_{k+1}{:}T_{k+1} \triangleright P'$ *and* $\Gamma, \Delta \triangleright Q \leadsto_{0/1} \Gamma', \Delta, n_{k+1}{:}T_{k+1} \triangleright Q'$.
(*c*) $\alpha \neq \tau, \mathrm{bn}(\alpha) \cap \{n_1, \ldots, n_k\} = \emptyset, \Gamma, \Delta \triangleright P \xrightarrow{\alpha}_\delta \Gamma', \Delta \triangleright P'$ *and* $R = (\nu n_1{:}T_1, \ldots, n_k{:}T_k)(P'|Q)$.
(*d*) $\alpha \neq \tau$, $\mathrm{bn}(\alpha) \cap \{n_1, \ldots, n_k\} = \emptyset, \Gamma, \Delta \triangleright Q \xrightarrow{\alpha}_\delta \Gamma', \Delta \triangleright Q'$ *and* $R = (\nu n_1{:}T_1, \ldots, n_k{:}T_k)(P|Q')$.
(*e*) $\alpha = (\nu n_j{:}T_j)\,\overline{m}\langle n_j\rangle$ *with* $n_j \in \{n_1, \ldots, n_k\}$, $\Gamma' = \Gamma, n_j : T_j$ *and* $\Gamma, \Delta \triangleright P \xrightarrow{\overline{m}\langle\rangle n_j}_\delta \Gamma, \Delta \triangleright P'$, $R = (\nu n_1 : T_1, \ldots, n_{j-1} : T_{j-1},\ n_{j+1} : T_{j+1}, \ldots, n_k : T_k)(P'|Q)$.
(*f*) $\alpha = (\nu n_j{:}T_j)\,\overline{m}\langle n_j\rangle$ *with* $n_j \in \{n_1, \ldots, n_k\}$, $\Gamma' = \Gamma, n_j : T_j$ *and* $\Gamma, \Delta \triangleright Q \xrightarrow{\overline{m}\langle\rangle n_j}_\delta \Gamma, \Delta \triangleright Q'$, $R = (\nu n_1 : T_1, \ldots, n_{j-1} : T_{j-1},\ n_{j+1} : T_{j+1}, \ldots, n_k : T_k)(P|Q')$.

Proof. The proof proceeds by induction on the number of restricted names k. When $k = 0$, the proof followed by Lemma 45. Let be $k > 0$, then the hypothesis $\Gamma \triangleright (\nu n_1 : T_1, \dots, n_k : T_k)(P|Q) \xrightarrow{\alpha}_\delta \Gamma' \triangleright R$ must have been derived using either the rule (Res) or the rule (Open). Let distinguish the two cases:

- by (Res) from $\Gamma, n_1 : T_1 \triangleright (\nu n_2 : T_2, \dots, n_k : T_k)(P|Q) \xrightarrow{\alpha}_\delta \Gamma', n_1 : T_1 \triangleright R_1$, with $R = (\nu n_1 : T_1)R_1$ and $n_1 \notin \mathrm{fn}(\alpha) \cup \mathrm{bn}(\alpha)$. By induction we have the following cases, where $\Delta = n_1 : T_1, \dots, n_k : T_k$:
  - (*a*) $\alpha = \tau$ and $R_1 = (\nu n_2 : T_2, \dots, n_k : T_k)(P'|Q')$ with $\Gamma, \Delta \triangleright P \rightsquigarrow_{0/1} \Gamma', \Delta \triangleright P'$ and $\Gamma, \Delta \triangleright Q \rightsquigarrow_{0/1} \Gamma', \Delta \triangleright Q'$. Then case (a) applies since $R = (\nu n_1 : T_1)R_1$.
  - (*b*) $\alpha = \tau$ and $R_1 = (\nu n_2 : T_2, \dots, n_k : T_k)(\nu n_{k+1} : T_{k+1})(P'|Q')$ with $\Gamma, \Delta \triangleright P \rightsquigarrow_{0/1} \Gamma', \Delta, n_{k+1}{:}T_{k+1} \triangleright P'$ and $\Gamma, \Delta \triangleright Q \rightsquigarrow_{0/1} \Gamma', \Delta, n_{k+1}{:}T_{k+1} \triangleright Q'$. Then case (b) applies since $R = (\nu n_1 : T_1)R_1$.
  - (*c*) $\alpha \neq \tau$, $\mathrm{bn}(\alpha) \cap \{n_1, \dots, n_k\} = \emptyset$ with $\Gamma, \Delta \triangleright P \xrightarrow{\alpha}_\delta \Gamma', \Delta \triangleright P'$ and $R_1 = (\nu n_2{:}T_2, \dots, n_k{:}T_k)(P'|Q)$. Then case (c) applies since $R = (\nu n_1 : T_1)R_1$.
  - (*d*) This case is similar to the previous one.
  - (*e*) $\alpha = (\nu n_j{:}T_j)\,\overline{m}\langle n_j \rangle$ with $n_j \in \{n_2, \dots, n_k\}$ and $\Gamma', n_1 : T = \Gamma, n_1 : T_1, n_j : T_j$. Moreover we have $\Gamma, \Delta \triangleright P \xrightarrow{\overline{m}\langle n_j \rangle}_\delta \Gamma, \Delta \triangleright P'$ and $R_1 = (\nu n_2 : T_2, \dots, n_{j-1} : T_{j-1},\ n_{j+1} : T_{j+1}, \dots, n_k : T_k)(P'|Q)$. Then case (e) applies since $R = (\nu n_1 : T_1)R_1$ and $\Gamma' = \Gamma, n_j : T_j$.
  - (*f*) This case is similar to the previous one.
- by (Open) from $\alpha = (\nu n_1{:}T_1)\,\overline{a}\langle n_1 \rangle$ and $\Gamma, n_1 : T_1 \triangleright (\nu n_2 : T_2, \dots, n_k : T_k)(P|Q) \xrightarrow{\overline{a}\langle n_1 \rangle}_\delta \Gamma, n_1 : T_1 \triangleright R$. By induction we have a number of cases, most of which are vacuous due to the form of the action $\overline{a}\langle n_1 \rangle$. The two remaining cases are:
  - (*c*) from $\Gamma, \Delta \triangleright P \xrightarrow{\overline{a}\langle n_1 \rangle}_\delta \Gamma, \Delta \triangleright P'$ with $R = (\nu n_2 : T_2, \dots, n_k : T_k)(P'|Q)$. Then case (e) applies with $n_j = n_1$ since $\Gamma' = \Gamma, n_j : T_j$ and $R = (\nu n_{j+1} : T_{j+1}, \dots, n_k : T_k)(P'|Q)$.
  - (*d*) This case is similar to the previous one. □

Proof of Lemma 28

Proof. Let P, Q, R be processes and Γ and Γ' be type environments.

(1) Let $\Gamma \triangleright P|Q \rightsquigarrow \Gamma' \triangleright R$, then $R = (\nu n_1 : T_1, \dots, n_k : T_k)(P'|Q')$ for some k such that $k \geq 0$, $\Gamma \triangleright P \rightsquigarrow \Gamma_1 \triangleright P'$ and $\Gamma \triangleright Q \rightsquigarrow \Gamma_2 \triangleright Q'$ with $\Gamma_i \subseteq \Gamma', n_1 : T_1, \dots, n_k : T_k$ for $i = 1, 2$.

(2) Let $\Gamma \triangleright !P \rightsquigarrow \Gamma' \triangleright R$, then $R = (\nu n_1 : T_1, \dots, n_k : T_k)(P_1| \cdots |P_s|!P)$ for some k, s such that $k, s \geq 0$ and $\Gamma \triangleright P \rightsquigarrow \Gamma_i \triangleright P_i$ with $\Gamma_i \subseteq \Gamma', n_1 : T_1, \dots, n_k : T_k$ for $i = 1, \dots, s$.

Proof of 1. We proceed by induction on the length of $\Gamma \triangleright P|Q \rightsquigarrow \Gamma' \triangleright R$. When the length is 0, the proof is immediate. Let instead be $\Gamma \triangleright P|Q \rightsquigarrow \Gamma' \triangleright R' \xrightarrow{\alpha}_\delta \Gamma'' \triangleright R''$. By induction we know that $R' = (\nu n_1 : T_1, \dots, n_k : T_k)(P'|Q')$ with $\Gamma \triangleright P \rightsquigarrow \Gamma_1 \triangleright P'$, $\Gamma \triangleright Q \rightsquigarrow \Gamma_2 \triangleright Q'$ and $\Gamma_i \subseteq \Gamma', n_1 : T_1, \dots, n_k : T_k$ for $i = 1, 2$. Now, by Lemma 46 we have to distinguish the following cases where $\Delta = n_1 : T_1, \dots, n_k : T_k$:

(*a*) $\alpha = \tau$ and $R'' = (\nu n_1 : T_1, \dots, n_k : T_k)(P''|Q'')$ with $\Gamma', \Delta \triangleright P' \rightsquigarrow_{0/1} \Gamma'', \Delta \triangleright P''$ and $\Gamma', \Delta \triangleright Q' \rightsquigarrow_{0/1} \Gamma'', \Delta \triangleright Q''$. Then by Lemma 43, since $\Gamma_i \subseteq \Gamma', \Delta$ for $i = 1, 2$, we have $\Gamma_1 \triangleright P' \rightsquigarrow_{0/1} \Gamma''_1 \triangleright P''$ and $\Gamma_2 \triangleright Q' \rightsquigarrow_{0/1} \Gamma''_2 \triangleright Q''$ with $\Gamma''_i \subseteq \Gamma'', \Delta$ for $i = 1, 2$. Hence $\Gamma \triangleright P \rightsquigarrow \Gamma''_1 \triangleright P''$ and $\Gamma \triangleright Q \rightsquigarrow \Gamma''_2 \triangleright Q''$.

(*b*) $\alpha = \tau$ and $R'' = (\nu n_1 : T_1, \dots, n_k : T_k)(\nu n_{k+1} : T_{k+1})(P''|Q'')$ with $\Gamma', \Delta \triangleright P \rightsquigarrow_{0/1} \Gamma'', \Delta, n_{k+1}{:}T_{k+1} \triangleright P''$ and $\Gamma', \Delta \triangleright Q \rightsquigarrow_{0/1} \Gamma'', \Delta, n_{k+1}{:}T_{k+1} \triangleright Q''$. By Lemma 43, since $\Gamma_i \subseteq \Gamma', \Delta$ for $i = 1, 2$, we have $\Gamma_1 \triangleright P' \rightsquigarrow_{0/1} \Gamma''_1 \triangleright P''$ and $\Gamma_2 \triangleright Q' \rightsquigarrow_{0/1} \Gamma''_2 \triangleright Q''$ with $\Gamma''_i \subseteq \Gamma'', \Delta$ for $i = 1, 2$. Hence $\Gamma \triangleright P \rightsquigarrow \Gamma''_1 \triangleright P''$ and $\Gamma \triangleright Q \rightsquigarrow \Gamma''_2 \triangleright Q''$.

(*c*) $\alpha \neq \tau$, $\mathrm{bn}(\alpha) \cap \{n_1, \dots, n_k\} = \emptyset$, $\Gamma', \Delta \triangleright P' \xrightarrow{\alpha}_\delta \Gamma'', \Delta \triangleright P''$ and $R'' = (\nu n_1 : T_1, \dots, n_k : T_k)(P''|Q')$. By Lemma 43, since $\Gamma_1 \subseteq \Gamma', \Delta$, we have $\Gamma_1 \triangleright P' \rightsquigarrow_{0/1} \Gamma''_1 \triangleright P''$ with $\Gamma''_1 \subseteq \Gamma'', \Delta$. Hence $\Gamma \triangleright P \rightsquigarrow \Gamma''_1 \triangleright P''$ and $\Gamma \triangleright Q \rightsquigarrow \Gamma_2 \triangleright Q'$.

(*d*) This case is similar to the previous one.

(*e*) $\alpha = (\boldsymbol{\nu} n_j{:}T_j)\,\overline{m}\langle n_j\rangle$ with $n_j \in \{n_1, \ldots, n_k\}$, $\Gamma'' = \Gamma', n_j : T_j$, $\Gamma', \Delta \rhd P' \xrightarrow{\overline{m}\langle n_j\rangle}_{\delta} \Gamma', \Delta \rhd P''$ and $R'' = (\boldsymbol{\nu} n_1 : T_1, \ldots, n_{j-1} : T_{j-1}, n_{j+1} : T_{j+1}, \ldots, n_k : T_k)(P''|Q')$. By Lemma 43, since $\Gamma_1 \subseteq \Gamma', \Delta$, we have $\Gamma_1 \rhd P' \xrightarrow{\overline{m}\langle n_j\rangle}_{\delta} \Gamma_1'' \rhd P''$ with $\Gamma_1'' \subseteq \Gamma', \Delta$. Then $\Gamma \rhd P \rightsquigarrow \Gamma_1'' \rhd P''$ and we conclude since $\Gamma_1'' \subseteq \Gamma'', n_{j-1} : T_{j-1}, n_{j+1} : T_{j+1}, \ldots, n_k : T_k = \Gamma', \Delta$.

(*f*) This case is similar to the previous one.

Proof of 2. We proceed by induction on the length of $\Gamma \rhd !P \rightsquigarrow \Gamma' \rhd R$. When the length is 0, the proof is immediate. Let instead be $\Gamma \rhd !P \rightsquigarrow \Gamma' \rhd R' \xrightarrow{\alpha}_{\delta} \Gamma'' \rhd R''$. By induction we know that $R' = (\boldsymbol{\nu} n_1 : T_1, \ldots, n_k : T_k)(P_1| \cdots |P_s|!P)$ with $\Gamma \rhd P \rightsquigarrow \Gamma_i \rhd P_i$ and $\Gamma_i \subseteq \Gamma', n_1 : T_1, \ldots, n_k : T_k$ for $i = 1, \ldots, s$. The proof follows similarly to the previous case, by Lemma 46 and Lemma 43. □

Lemma 47. *Let $\sigma \in \Sigma, P_1, P_2, Q_1, Q_2$ be processes and Γ a type environment such that $\Gamma \vdash P_i$ and $\Gamma \vdash Q_i$ for $i = 1, 2$. Let be $\Gamma \models P_1 \cong_\sigma P_2$, $\Gamma \models Q_1 \cong_\sigma Q_2$, $\Gamma \rhd P_i \in \mathcal{W}(\cong_\sigma)$ and $\Gamma \rhd Q_i \in \mathcal{W}(\cong_\sigma)$ for $i = 1, 2$, and let P_i and Q_i for $i = 1, 2$ do not synchronize on declassified actions. Therefore $\Gamma \models P_1|Q_1 \cong_\sigma P_2|Q_2$.*

Proof. Consider the following relation $\mathcal{S}$:

$$\begin{aligned}\mathcal{S} = \{(\Gamma \rhd P_1|Q_1,\ \Gamma \rhd P_2|Q_2) \mid\ & \Gamma \models Q_1 \cong_\sigma Q_2 \quad \Gamma \models P_1 \cong_\sigma P_2 \\ & \Gamma \rhd P_i \in \mathcal{W}(\cong_\sigma) \text{ and } \Gamma \rhd Q_i \in \mathcal{W}(\cong_\sigma)\ i = 1, 2 \\ & P_i \text{ and } Q_i \text{ do not synchronize on declassified actions } i = 1, 2\}\end{aligned}$$

it is sufficient to show that $\mathcal{S}$ is a bisimulation up to σ-contexts and up-to $\approx_\sigma$.

Assume $\Gamma \rhd P_1|Q_1 \xrightarrow{\alpha}_{\sigma} \Gamma' \rhd U$, we show that $\Gamma \rhd P_2|Q_2 \overset{\hat{\alpha}}{\Longrightarrow}_{\sigma} \Gamma' \rhd V$ with $\Gamma' \models U \approx_\sigma C[U']$ and $\Gamma' \models V \approx_\sigma C[V']$ such that $\Gamma' \vdash_\sigma C[\cdot_{\Gamma''}]$, i.e. it is a σ-context, and $(\Gamma'' \rhd U')\,\mathcal{S}\,(\Gamma'' \rhd V')$. The assumption $\Gamma \rhd P_1|Q_1 \xrightarrow{\alpha}_{\sigma} \Gamma' \rhd U$ must have been derived in one of the following cases:

- by (Par) from $\Gamma \rhd P_1 \xrightarrow{\alpha}_{\sigma} \Gamma' \rhd P_1'$ with $U = P_1'|Q_1$. Now, from $\Gamma \models P_1 \approx_\sigma P_2$ we have $\Gamma \rhd P_2 \overset{\hat{\alpha}}{\Longrightarrow}_{\sigma} \Gamma' \rhd P_2'$ with $\Gamma' \models P_1' \approx_\sigma P_2'$. Then we also have $\Gamma \rhd P_2|Q_2 \overset{\hat{\alpha}}{\Longrightarrow}_{\sigma} \Gamma' \rhd P_2'|Q_2$ The fact that $\Gamma' \models P_1' \approx_\sigma P_2'$, together with $\Gamma' \rhd P_i' \in \mathcal{W}(\cong_\sigma)$ for $i = 1, 2$, which follow by persistence, and the fact that P_i' and Q_i do not synchronize on declassified actions being derivatives of $P_i|Q_i$ for $i = 1, 2$, imply $(\Gamma' \rhd P_1'|Q_1)\ \mathcal{S}\ (\Gamma' \rhd P_2'|Q_2)$, which is sufficient to conclude the proof since we can choose $C[\cdot_{\Gamma''}] = [\cdot_{\Gamma'}]$.
- by (Par) from $\Gamma \rhd Q_1 \xrightarrow{\alpha}_{\sigma} \Gamma' \rhd Q_1'$ with $U = P_1|Q_1'$. This case follows as the previous one.
- by (Comm) from $P_1 \xrightarrow{\overline{n}\langle m\rangle} P_1'$ and $Q_1 \xrightarrow{n(m)} Q_1'$ with $U = P_1'|Q_1'$ (or viceversa, which follows similarly). By Propositions 6 and 7 we have $\Gamma \rhd P_1 \xrightarrow{\overline{n}\langle m\rangle}_{\delta} \Gamma' \rhd P_1'$ and $\Gamma \rhd Q_1 \xrightarrow{n(m)}_{\delta} \Gamma' \rhd Q_1'$ for some δ. Let distinguish two cases:
  - $\delta \preceq \sigma$. In this case, from $\Gamma \models P_1 \approx_\sigma P_2$ we have $\Gamma \rhd P_2 \overset{\overline{n}\langle m\rangle}{\Longrightarrow}_{\delta} \Gamma' \rhd P_2'$ with $\Gamma' \models P_1' \approx_\sigma P_2'$. Moreover, from $\Gamma \models Q_1 \approx_\sigma Q_2$ we have $\Gamma \rhd Q_2 \overset{n(m)}{\Longrightarrow}_{\delta} \Gamma' \rhd Q_2'$ with $\Gamma' \models Q_1' \approx_\sigma Q_2'$. Then $\Gamma \rhd P_2|Q_2 \Longrightarrow \Gamma' \rhd P_2'|Q_2'$. In order to conclude the proof it is sufficient to choose $C[\cdot_{\Gamma''}] = [\cdot_{\Gamma'}]$ and to show that $(\Gamma' \rhd P_1'|Q_1')\ \mathcal{S}\ (\Gamma' \rhd P_2'|Q_2')$. The last fact comes from (i) the fact that P_i' and Q_i', for $i = 1, 2$, do not synchronize on declassified actions being derivatives of $P_i|Q_i$ for $i = 1, 2$ and , (ii) $\Gamma' \rhd P_i' \in \mathcal{W}(\cong_\sigma)$ and $\Gamma' \rhd Q_i' \in \mathcal{W}(\cong_\sigma)$ which follow by persistence and (iii) $\Gamma' \models P_1' \approx_\sigma P_2'$ and $\Gamma' \models Q_1' \approx_\sigma Q_2'$.
  - $\sigma \prec \delta$. From $\Gamma \rhd Q_1 \xrightarrow{n(m)}_{\delta} \Gamma \rhd Q_1'$ and $\Gamma \rhd Q_1 \in \mathcal{W}(\cong_\sigma)$ we have $\Gamma \rhd Q_1 \Longrightarrow \Gamma \rhd Q_1''$ with $\Gamma \models Q_1' \approx_\sigma Q_1''$. From $\Gamma \models Q_1 \approx_\sigma Q_2$ we have $\Gamma \rhd Q_2 \Longrightarrow \Gamma \rhd Q_2'$ with $\Gamma \models Q_2' \approx_\sigma Q_1''$, hence $\Gamma \models Q_2' \approx_\sigma Q_1'$, by transitivity. Moreover, from $\Gamma \rhd P_1 \xrightarrow{\overline{n}\langle m\rangle}_{\delta} \Gamma \rhd P_1'$ and $\Gamma \rhd P_1 \in \mathcal{W}(\cong_\sigma)$ we have $\Gamma \rhd P_1 \Longrightarrow \Gamma \rhd P_1''$ with $\Gamma \models P_1' \approx_\sigma P_1''$. From $\Gamma \models P_1 \approx_\sigma P_2$ we have $\Gamma \rhd P_2 \Longrightarrow \Gamma \rhd P_2'$ with $\Gamma \models P_2' \approx_\sigma P_1''$, hence, by transitivity, $\Gamma \models P_2' \approx_\sigma P_1'$. By (Par) we have $\Gamma \rhd P_2|Q_2 \Longrightarrow \Gamma \rhd P_2'|Q_2'$. In order to conclude the proof it is sufficient to choose $C[\cdot_{\Gamma''}] = [\cdot_{\Gamma}]$ and to show that $(\Gamma \rhd P_1'|Q_1')\ \mathcal{S}\ (\Gamma \rhd P_2'|Q_2')$. The last fact comes

from (i) the fact that P'_i and Q'_i, for $i = 1, 2$, do not synchronize on declassified actions being derivatives of $P_i|Q_i$ for $i = 1, 2$ and , (ii) $\Gamma' \triangleright P'_i \in \mathcal{W}(\cong_\sigma)$ and $\Gamma' \triangleright Q'_i \in \mathcal{W}(\cong_\sigma)$ which follow by persistence and (iii) $\Gamma \models P'_2 \approx_\sigma P'_1$ and $\Gamma \models Q'_2 \approx_\sigma Q'_1$.

- by (Dec Comm). This case is vacuous since, by the hypothesis, P_1 and Q_1 do not synchronize on declassified actions.
- by (Close) from $P_1 \xrightarrow{(\nu m:T)\,\overline{n}\langle m\rangle} P'_1$ and $Q_1 \xrightarrow{n(m)} Q'_1$ with $U = (\nu m : T)(P'_1|Q'_1)$ (or viceversa, which follows similarly). By Propositions 6 and 7 we have $\Gamma \triangleright P_1 \xrightarrow{(\nu m:T)\,\overline{n}\langle m\rangle}_\delta \Gamma, m : T \triangleright P'_1$ and $\Gamma \triangleright Q_1 \xrightarrow{(\nu m:T)\,n(m)}_\delta \Gamma, m : T \triangleright Q'_1$ for some δ. Let distinguish two cases, where $\Gamma' = \Gamma, m : T$:

  . $\delta \preceq \sigma$. In this case, from $\Gamma \models P_1 \approx_\sigma P_2$ we have $\Gamma \triangleright P_2 \overset{(\nu m:T)\,\overline{n}\langle m\rangle}{\Longrightarrow}_\delta \Gamma' \triangleright P'_2$ with $\Gamma' \models P'_1 \approx_\sigma P'_2$. Moreover, from $\Gamma \models Q_1 \approx_\sigma Q_2$ we have $\Gamma \triangleright Q_2 \overset{(\nu m:T)\,n(m)}{\Longrightarrow}_\delta \Gamma' \triangleright Q'_2$ with $\Gamma' \models Q'_1 \approx_\sigma Q'_2$. Then $\Gamma \triangleright P_2|Q_2 \Longrightarrow \Gamma' \triangleright (\nu m : T)(P'_2|Q'_2)$. In order to conclude the proof it is sufficient to choose $C[\cdot_{\Gamma''}] = (\nu m : T)[\cdot_{\Gamma'}]$ and to show that $(\Gamma' \triangleright P'_1|Q'_1)\ \mathcal{S}\ (\Gamma' \triangleright P'_2|Q'_2)$. The last fact comes from (i) the fact that P'_i and Q'_i, for $i = 1, 2$, do not synchronize on declassified actions being derivatives of $P_i|Q_i$ for $i = 1, 2$ and (ii) $\Gamma' \triangleright P'_i \in \mathcal{W}(\cong_\sigma)$ and $\Gamma' \triangleright Q'_i \in \mathcal{W}(\cong_\sigma)$ which follow by persistence and (iii) $\Gamma' \models P'_1 \approx_\sigma P'_2$ and $\Gamma' \models Q'_1 \approx_\sigma Q'_2$.

  . $\sigma \prec \delta$. From $\Gamma \triangleright Q_1 \xrightarrow{(\nu m:T)\,n(m)}_\delta \Gamma' \triangleright Q'_1$, we know that $\Gamma, m : T \triangleright Q_1 \xrightarrow{n(m)}_\delta \Gamma' \triangleright Q'_1$ and, similarly, we know $\Gamma, m : T \triangleright P_1 \xrightarrow{\overline{n}\langle m\rangle}_\delta \Gamma' \triangleright P'_1$. From $\Gamma \triangleright Q_1 \in \mathcal{W}(\cong_\sigma)$ and $\Gamma \triangleright P_1 \in \mathcal{W}(\cong_\sigma)$, by the weakening Lemma 44 we have $\Gamma' \triangleright Q_1 \in \mathcal{W}(\cong_\sigma)$ and $\Gamma' \triangleright P_1 \in \mathcal{W}(\cong_\sigma)$, then $\Gamma, m : T \triangleright Q_1 \Longrightarrow \Gamma, m : T \triangleright Q''_1$ with $\Gamma, m : T \models Q'_1 \approx_\sigma Q''_1$ and $\Gamma, m : T \triangleright P_1 \Longrightarrow \Gamma, m : T \triangleright P''_1$ with $\Gamma, m : T \models P'_1 \approx_\sigma P''_1$. From $\Gamma \models P_1 \approx_\sigma P_2$, by the weakening Lemma 44 we have $\Gamma, m : T \models P_1 \approx_\sigma P_2$, hence $\Gamma, m : T \triangleright P_2 \Longrightarrow \Gamma, m : T \triangleright P''_2$ with $\Gamma, m : T \models P''_2 \approx_\sigma P''_1 \approx_\sigma P'_1$. Similarly, from $\Gamma \models Q_1 \approx_\sigma Q_2$, $\Gamma, m : T \triangleright Q_2 \Longrightarrow \Gamma, m : T \triangleright Q''_2$ with $\Gamma, m : T \models Q''_2 \approx_\sigma Q''_1 \approx_\sigma Q'_1$. By (Par) we have $\Gamma, m : T \triangleright P_2|Q_2 \Longrightarrow \Gamma, m : T \triangleright P''_2|Q''_2$, and by (Res) $\Gamma \triangleright (\nu m : T)(P_2|Q_2) \Longrightarrow \Gamma \triangleright (\nu m : T)(P''_2|Q''_2)$. Notice that $\Gamma \models (\nu m : T)(P_2|Q_2) \approx_\sigma (P_2|Q_2)$ since m is not free in $P_2|Q_2$ being $\Gamma \vdash P_2|Q_2$ the hypothesis, and $m \notin Dom\ \Gamma$. In order to conclude the proof it is sufficient to choose $C[\cdot_{\Gamma''}] = (\nu m : T)[\cdot_{\Gamma'}]$ and to show that $(\Gamma' \triangleright P'_1|Q'_1)\ \mathcal{S}\ (\Gamma' \triangleright P''_2|Q''_2)$. The last fact comes from (i) the fact that P'_1 and Q'_1, as well as P''_2 and Q''_2 do not synchronize on declassified actions being derivatives of $P_i|Q_i$ for $i = 1, 2$ and (ii) $\Gamma' \triangleright P'_1 \in \mathcal{W}(\cong_\sigma)$, $\Gamma' \triangleright P''_2 \in \mathcal{W}(\cong_\sigma)$, $\Gamma' \triangleright Q'_1 \in \mathcal{W}(\cong_\sigma)$ and $\Gamma' \triangleright Q''_2 \in \mathcal{W}(\cong_\sigma)$ which follow by persistence and (iii) $\Gamma' \models P'_1 \approx_\sigma P''_2$ and $\Gamma' \models Q'_1 \approx_\sigma Q''_2$.
- by (Dec Close). This case is vacuous since, by the hypothesis, P_1 and Q_1 do not synchronize on declassified actions. □

Proposition 48. *Let $\sigma \in \Sigma$, P and Q be two processes that do not synchronize on declassified actions. Let 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 $\Gamma \triangleright P|Q \in \mathcal{W}(\cong_\sigma)$.*

Proof. Let be $\Gamma \triangleright P|Q \rightsquigarrow \Gamma' \triangleright R$, then by Lemma 28 (1) we have that $R = (\nu n_1{:}T_1, \ldots, n_k{:}T_k)(P'|Q')$ where $k \geq 0$, $\Gamma \triangleright P \rightsquigarrow \Gamma_1 \triangleright P'$ and $\Gamma \triangleright Q \rightsquigarrow \Gamma_2 \triangleright Q'$ with $\Gamma_i \subseteq \Gamma', n_1 : T_1, \ldots, n_k : T_k$ for $i = 1, 2$.

- Assume $\Gamma' \triangleright R \xrightarrow{\alpha}{}^\sigma \Gamma'' \triangleright R''$ with $\alpha \in \{\overline{a}\langle b\rangle, a(b)\}$. This must come either by case (c) or by case (d) of Lemma 46. Let assume the case (c), the other one being similar. Let be $\Delta = n_1{:}T_1, \ldots, n_k{:}T_k$, in this case we know that $\Gamma', \Delta \triangleright P' \xrightarrow{\alpha}{}^\sigma \Gamma'', \Delta \triangleright P''$ with $\Gamma'' = \Gamma'$ and $R'' = (\nu n_1{:}T_1, \ldots, n_k{:}T_k)(P''|Q')$. From $\Gamma \triangleright P \in \mathcal{W}(\cong_\sigma)$ and $\Gamma \triangleright P \rightsquigarrow \Gamma_1 \triangleright P'$, by persistence we have $\Gamma_1 \triangleright P' \in \mathcal{W}(\cong_\sigma)$. Moreover, from $\Gamma_1 \subseteq \Gamma', \Delta$, by the weakening Lemma 44 we also have $\Gamma', \Delta \triangleright P' \in \mathcal{W}(\cong_\sigma)$, which implies $\Gamma', \Delta \triangleright P' \Longrightarrow \Gamma', \Delta \triangleright P'''$ and $\Gamma', \Delta \models P''' \approx_\sigma P''$, hence $\Gamma' \triangleright R \Longrightarrow \Gamma' \triangleright (\nu n_1{:}T_1, \ldots, n_k{:}T_k)(P'''|Q')$. We can now apply Lemma 47 since $\Gamma', \Delta \models P''' \approx_\sigma P''$, both P'' and Q', and P''' and Q' do not synchronize on declassified actions being derivatives of P and Q, and $\Gamma', \Delta \triangleright P'' \in \mathcal{W}(\cong_\sigma)$, $\Gamma', \Delta \triangleright P''' \in \mathcal{W}(\cong_\sigma)$ and $\Gamma', \Delta \triangleright Q' \in \mathcal{W}(\cong_\sigma)$ which follow by persistence and the weakening Lemma 44. By Lemma 47 we then obtain $\Gamma', \Delta \models P'''|Q' \approx_\sigma P''|Q'$, and we can conclude $\Gamma' \models (\boldsymbol{\nu} n_1{:}T_1, \dots, n_k{:}T_k)(P'''|Q') \approx_\sigma (\boldsymbol{\nu} n_1{:}T_1, \dots, n_k{:}T_k)(P''|Q')$ by contextuality of $\approx_\sigma$, i.e. Proposition 36.

- Assume $\Gamma' \triangleright R \xrightarrow{\alpha}{}^{\sigma} \Gamma', m{:}T \triangleright R''$ with $\alpha \in \{(\boldsymbol{\nu} m{:}T)\,\overline{a}\langle m\rangle, (\boldsymbol{\nu} m{:}T)\, a(m)\}$. Let distinguish two subcases:

  . $m \notin \{n_1, \dots, n_k\}$. In this case the hypothesis comes either by case (c) or by case (d) of Lemma 46, and follows similarly to the case shown above.
  . $m = n_j$ with $n_j \in \{n_1, \dots, n_k\}$, then the case (e) or (f) of Lemma 46 applies. Let assume the case (e), the other one being similar. In this case we have
  $R'' = (\boldsymbol{\nu} n_1{:}T_1, \dots, n_{j-1}{:}T_{j-1},\ n_{j+1}{:}T_{j+1}, \dots, n_k{:}T_k)(P''|Q')$,
  and $\Gamma', \Delta \triangleright P' \xrightarrow{\alpha_1}{}^{\sigma} \Gamma', \Delta \triangleright P''$ with $\alpha_1 \in \{\overline{a}\langle m\rangle, a(m)\}$. From $\Gamma \triangleright P \in \mathcal{W}(\cong_\sigma)$ and $\Gamma \triangleright P \rightsquigarrow \Gamma_1 \triangleright P'$, by persistence we have $\Gamma_1 \triangleright P' \in \mathcal{W}(\cong_\sigma)$. Moreover, from $\Gamma_1 \subseteq \Gamma', \Delta$, by the weakening Lemma 44 we also have $\Gamma', \Delta \triangleright P' \in \mathcal{W}(\cong_\sigma)$, which implies $\Gamma', \Delta \triangleright P' \Longrightarrow \Gamma', \Delta \triangleright P'''$, with $\Gamma', \Delta \models P'' \approx_\sigma P'''$, hence $\Gamma' \triangleright R \Longrightarrow \Gamma' \triangleright (\boldsymbol{\nu} n_1{:}T_1, \dots, n_k{:}T_k)(P'''|Q')$ The fact that $\Gamma', \Delta \models P'' \approx_\sigma P'''$, together with $\Gamma', \Delta \triangleright P'' \in \mathcal{W}(\cong_\sigma)$, $\Gamma', \Delta \triangleright P''' \in \mathcal{W}(\cong_\sigma)$, $\Gamma', \Delta \triangleright Q' \in \mathcal{W}(\cong_\sigma)$, which follow by persistence and the weakening Lemma 44, and the fact that both P'' and Q' and P''' and Q' do not synchronize on declassified actions being derivatives of P and Q, by Lemma 47 we have $\Gamma', \Delta \models P'''|Q' \approx_\sigma P''|Q'$. By contextuality of $\approx_\sigma$ we conclude $\Gamma' \models (\boldsymbol{\nu} n_1{:}T_1, \dots, n_k{:}T_k)(P'''|Q') \approx_\sigma (\boldsymbol{\nu} n_j{:}T_j)R''$ as desired. $\square$

Lemma 49. *Let $\sigma \in \Sigma, P, P_1, P_2$ be processes and Γ a type environment such that $\Gamma \vdash P$ and $\Gamma \vdash P_i$ for $i = 1, 2$. Let be $\Gamma \models P_1 \cong_\sigma P_2$, $\Gamma \triangleright P \in \mathcal{W}(\cong_\sigma)$ and $\Gamma \triangleright P_i \in \mathcal{W}(\cong_\sigma)$ for $i = 1, 2$, and let P do not contain declassified actions. Then $\Gamma \models P_1|!P \cong_\sigma P_2|!P$.*

Proof. Consider the following relation $\mathcal{S}$:

$$\begin{aligned}\mathcal{S} = \{(\Gamma \triangleright P_1|!P,\ \Gamma \triangleright P_2|!P) \mid\ & \Gamma \models P_1 \cong_\sigma P_2 \\ & \Gamma \triangleright P \in \mathcal{W}(\cong_\sigma) \text{ and } \Gamma \triangleright P_i \in \mathcal{W}(\cong_\sigma)\ i = 1, 2 \\ & P \text{ does not contain declassified actions}\}\end{aligned}$$

it is sufficient to show that $\mathcal{S}$ is a bisimulation up to σ-contexts and up-to $\approx_\sigma$.
Assume $\Gamma \triangleright P_1|!P \xrightarrow{\alpha}_\sigma \Gamma' \triangleright P'$, we show that $\Gamma \triangleright P_2|!P \overset{\hat{\alpha}}{\Longrightarrow}_\sigma \Gamma' \triangleright Q'$ with $\Gamma' \models P' \approx_\sigma C[P'']$ and $\Gamma' \models Q' \approx_\sigma C[Q'']$ such that $\Gamma' \vdash_\sigma C[\cdot_{\Gamma''}]$, i.e. it is a σ-context, and $(\Gamma'' \triangleright P'')\, \mathcal{S}\, (\Gamma'' \triangleright Q'')$. The assumption $\Gamma \triangleright P_1|!P \xrightarrow{\alpha}_\sigma \Gamma' \triangleright P'$ must have been derived in one of the following cases:

- by (Par) from $\Gamma \triangleright P_1 \xrightarrow{\alpha}_\sigma \Gamma' \triangleright P_1'$ with $P' = P_1'|!P$. Now, from $\Gamma \models P_1 \approx_\sigma P_2$ we have $\Gamma \triangleright P_2 \overset{\hat{\alpha}}{\Longrightarrow}_\sigma \Gamma' \triangleright P_2'$ with $\Gamma' \models P_1' \approx_\sigma P_2'$. Then we also have $\Gamma \triangleright P_2|!P \overset{\hat{\alpha}}{\Longrightarrow}_\sigma \Gamma' \triangleright P_2'|!P$ The fact that $\Gamma' \models P_1' \approx_\sigma P_2'$, together with $\Gamma' \triangleright P_i' \in \mathcal{W}(\cong_\sigma)$ for $i = 1, 2$, which follow by persistence, implies $(\Gamma' \triangleright P_1'|!P)\ \mathcal{S}\ (\Gamma' \triangleright P_2'|!P)$, which is sufficient to conclude the proof since we can choose $C[\cdot_{\Gamma''}] = [\cdot_{\Gamma'}]$.
- by (Par) and (Rep-Act) from $\Gamma \triangleright P \xrightarrow{\alpha}_\sigma \Gamma' \triangleright P''$ with $P' = P_1|P''|!P$. Then $\Gamma \triangleright P_2|!P \xrightarrow{\alpha}_\sigma \Gamma' \triangleright P_2|P''|!P$. In order to conclude the proof it is sufficient to choose $C[\cdot_{\Gamma''}] = [\cdot_{\Gamma'}]$ and to show that $(\Gamma' \triangleright P_1|P''|!P)\ \mathcal{S}\ (\Gamma' \triangleright P_2|P''|!P)$. The last fact comes from (i) the fact that P does not contain declassified actions, (ii) $\Gamma' \triangleright P_1|P'' \in \mathcal{W}(\cong_\sigma)$ and $\Gamma' \triangleright P_2|P'' \in \mathcal{W}(\cong_\sigma)$ which follow by persistence, the weakening Lemma 44 (since $\Gamma \subseteq \Gamma'$) and the compositionality property of $\mathcal{W}(\cong_\sigma)$ (Proposition 48), and (iii) $\Gamma' \models P_1|P'' \approx_\sigma P_2|P''$ which comes by Lemma 47.
- by (Comm) and (Rep-Act) from $P_1 \xrightarrow{\overline{n}\langle m\rangle} P_1'$ and $P \xrightarrow{n(m)} P''$ with $P' = P_1'|P''|!P$ (or viceversa, which follows similarly). By Propositions 6 and 7 we have $\Gamma \triangleright P_1 \xrightarrow{\overline{n}\langle m\rangle}_\delta \Gamma' \triangleright P_1'$ and $\Gamma \triangleright P \xrightarrow{n(m)}_\delta \Gamma' \triangleright P''$ for some δ. Let distinguish two cases:

- $\delta \preceq \sigma$. In this case, from $\Gamma \models P_1 \approx_\sigma P_2$ we have $\Gamma \rhd P_2 \overset{\overline{n}\langle m\rangle}{\Longrightarrow}_\delta \Gamma' \rhd P_2'$ with $\Gamma' \models P_1' \approx_\sigma P_2'$. Then $\Gamma \rhd P_2|!P \Longrightarrow \Gamma' \rhd P_2'|P''|!P$. In order to conclude the proof it is sufficient to choose $C[\cdot_{\Gamma''}] = [\cdot_{\Gamma'}]$ and to show that $(\Gamma' \rhd P_1'|P''|!P)\ \mathcal{S}\ (\Gamma' \rhd P_2'|P''|!P)$. The last fact comes from (i) the fact that P does not contain declassified actions, (ii) $\Gamma' \rhd P_1'|P'' \in \mathcal{W}(\cong_\sigma)$ and $\Gamma' \rhd P_2'|P'' \in \mathcal{W}(\cong_\sigma)$ which follow by persistence and the compositionality property of $\mathcal{W}(\cong_\sigma)$ (Proposition 48) and (iii) $\Gamma' \models P_1'|P'' \approx_\sigma P_2'|P''$ which comes by Lemma 47.
- $\sigma \prec \delta$. From $\Gamma \rhd P \xrightarrow{n(m)}_\delta \Gamma \rhd P''$ and $\Gamma \rhd P \in \mathcal{W}(\cong_\sigma)$ we have $\Gamma \rhd P \Longrightarrow \Gamma \rhd P'''$ with $\Gamma \models P'' \approx_\sigma P'''$. Hence also $\Gamma \rhd !P \Longrightarrow \Gamma \rhd P'''|!P$. Moreover, from $\Gamma \rhd P_1 \xrightarrow{\overline{n}\langle m\rangle}_\delta \Gamma \rhd P_1'$ and $\Gamma \rhd P_1 \in \mathcal{W}(\cong_\sigma)$ we have $\Gamma \rhd P_1 \Longrightarrow \Gamma \rhd P_1''$ with $\Gamma \models P_1' \approx_\sigma P_1''$. From $\Gamma \models P_1 \approx_\sigma P_2$ we have $\Gamma \rhd P_2 \Longrightarrow \Gamma \rhd P_2'$ with $\Gamma \models P_2' \approx_\sigma P_1''$, hence, by transitivity, $\Gamma \models P_2' \approx_\sigma P_1'$. By (Par) we have $\Gamma \rhd P_2|!P \Longrightarrow \Gamma \rhd P_2'|P'''|!P$. In order to conclude the proof it is sufficient to choose $C[\cdot_{\Gamma''}] = [\cdot_\Gamma]$ and to show that $(\Gamma \rhd P_1'|P''|!P)\ \mathcal{S}\ (\Gamma \rhd P_2'|P''|!P)$. The last fact comes from (i) the fact that P does not contain declassified actions, (ii) $\Gamma \rhd P_1'|P'' \in \mathcal{W}(\cong_\sigma)$ and $\Gamma \rhd P_2'|P'' \in \mathcal{W}(\cong_\sigma)$ which follow by persistence and the compositionality property of $\mathcal{W}(\cong_\sigma)$ (Proposition 48) and (iii) $\Gamma \models P_1'|P'' \approx_\sigma P_2'|P''$ which comes by Lemma 47.

- by (Dec Comm). This case is vacuous since P does not contain declassified actions.
- by (Close) and (Rep-Act) from $P_1 \xrightarrow{(\nu m:T)\,\overline{n}\langle m\rangle} P_1'$ and $P \xrightarrow{n(m)} P''$ with $P' = (\boldsymbol{\nu} m : T)(P_1'|P''|!P)$ (or vice-versa, which follows similarly). By Propositions 6 and 7 we have $\Gamma \rhd P_1 \xrightarrow{(\nu m:T)\,\overline{n}\langle m\rangle}_\delta \Gamma, m : T \rhd P_1'$ and $\Gamma \rhd P \xrightarrow{(\nu m:T)\,n(m)}_\delta \Gamma, m : T \rhd P''$, hence $\Gamma \rhd !P \xrightarrow{(\nu m:T)\,n(m)}_\delta \Gamma, m : T \rhd P''|!P$. Let distinguish two cases, where $\Gamma' = \Gamma, m : T$:

  - $\delta \preceq \sigma$. In this case, from $\Gamma \models P_1 \approx_\sigma P_2$ we have $\Gamma \rhd P_2 \overset{(\nu m:T)\,\overline{n}\langle m\rangle}{\Longrightarrow}_\delta \Gamma' \rhd P_2'$ with $\Gamma' \models P_1' \approx_\sigma P_2'$. Then $\Gamma \rhd P_2|!P \Longrightarrow \Gamma' \rhd (\boldsymbol{\nu} m : T)(P_2'|P''|!P)$. In order to conclude the proof it is sufficient to choose $C[\cdot_{\Gamma''}] = (\boldsymbol{\nu} m : T)[\cdot_{\Gamma'}]$ and to show that $(\Gamma' \rhd P_1'|P''|!P)\ \mathcal{S}\ (\Gamma' \rhd P_2'|P''|!P)$. The last fact comes from (i) the fact that P does not contain declassified actions, (ii) $\Gamma' \rhd P_1'|P'' \in \mathcal{W}(\cong_\sigma)$ and $\Gamma' \rhd P_2'|P'' \in \mathcal{W}(\cong_\sigma)$ which follow by persistence and the compositionality property of $\mathcal{W}(\cong_\sigma)$ (Proposition 48) and (iii) $\Gamma' \models P_1'|P'' \approx_\sigma P_2'|P''$ which comes by Lemma 47.
  - $\sigma \prec \delta$. From $\Gamma \rhd P \xrightarrow{(\nu m:T)\,n(m)}_\delta \Gamma' \rhd P''$, we know that $\Gamma, m : T \rhd P \xrightarrow{n(m)}_\delta \Gamma' \rhd P''$ and, similarly, we know $\Gamma, m : T \rhd P_1 \xrightarrow{\overline{n}\langle m\rangle}_\delta \Gamma' \rhd P_1'$. From $\Gamma \rhd P \in \mathcal{W}(\cong_\sigma)$ and $\Gamma \rhd P_1 \in \mathcal{W}(\cong_\sigma)$, by the weakening Lemma 44 we have $\Gamma' \rhd P \in \mathcal{W}(\cong_\sigma)$ and $\Gamma' \rhd P_1 \in \mathcal{W}(\cong_\sigma)$, then $\Gamma, m : T \rhd P \Longrightarrow \Gamma, m : T \rhd P'''$ with $\Gamma, m : T \models P'' \approx_\sigma P'''$ and $\Gamma, m : T \rhd P_1 \Longrightarrow \Gamma, m : T \rhd P_1''$ with $\Gamma, m : T \models P_1' \approx_\sigma P_1''$. By (Rep-Act) we have $\Gamma, m : T \rhd !P \Longrightarrow \Gamma, m : T \rhd P'''|!P$ From $\Gamma \models P_1 \approx_\sigma P_2$, by the weakening Lemma 44 we have $\Gamma, m : T \models P_1 \approx_\sigma P_2$, hence $\Gamma, m : T \rhd P_2 \Longrightarrow \Gamma, m : T \rhd P_2''$ with $\Gamma, m : T \models P_2'' \approx_\sigma P_1'' \approx_\sigma P_1'$. By (Par) we have $\Gamma, m : T \rhd P_2|!P \Longrightarrow \Gamma, m : T \rhd P_2''|P'''|!P$, and by (Res) $\Gamma \rhd (\boldsymbol{\nu} m : T)(P_2|!P) \Longrightarrow \Gamma \rhd (\boldsymbol{\nu} m : T)$ $(P_2''|P'''|!P)$. Notice that $(\boldsymbol{\nu} m : T)(P_2|!P) \approx_\sigma (P_2|!P)$ since m is not free in $P_2|!P$. In order to conclude the proof it is sufficient to choose $C[\cdot_{\Gamma''}] = (\boldsymbol{\nu} m : T)[\cdot_{\Gamma'}]$ and to show that $(\Gamma' \rhd P_1'|P''|!P)\ \mathcal{S}\ (\Gamma' \rhd P_2''|P'''|!P)$. The last fact comes from (i) the fact that P does not contain declassified actions, (ii) $\Gamma' \rhd P_1'|P'' \in \mathcal{W}(\cong_\sigma)$ and $\Gamma' \rhd P_2''|P''' \in \mathcal{W}(\cong_\sigma)$ which follow by persistence and the compositionality property of $\mathcal{W}(\cong_\sigma)$ (Proposition 48) and (iii) $\Gamma' \models P_1'|P'' \approx_\sigma P_2''|P'''$ which comes by Lemma 47.

- by (Dec Close). This case is vacuous since P does not contain declassified actions. □

Proof of Theorem 29

Proof. Let $\sigma \in \Sigma$, P and Q be two processes and Γ be a type environment such that $\Gamma \vdash P, Q$. If $\Gamma \rhd P \in \mathcal{CR}(\cong_\sigma)$ and $\Gamma \rhd Q \in \mathcal{CR}(\cong_\sigma)$ then

(1) $\Gamma' \triangleright \overline{a}\langle b\rangle.P \in \mathcal{CR}(\cong_\sigma)$ where $\Gamma' = \Gamma \cup \{a : \delta[T]\} \cup \{b : T\}$ and $\delta \preceq \sigma$;
(2) $\Gamma' \triangleright a(x{:}T).P \in \mathcal{CR}(\cong_\sigma)$ where $\Gamma' = \Gamma \cup \{a : \delta[T]\}$ and $\delta \preceq \sigma$;
(3) $\Gamma' \triangleright \text{if } a = b \text{ then } P \text{ else } Q \in \mathcal{CR}(\cong_\sigma)$ where $\Gamma' = \Gamma \cup \{a : T\} \cup \{b : T\}$;
(4) $\Gamma \triangleright P|Q \in \mathcal{CR}(\cong_\sigma)$ whenever P and Q do not synchronize on declassified actions.
(5) $\Gamma' \triangleright (\boldsymbol{\nu} n : T)P \in \mathcal{CR}(\cong_\sigma)$ where $\Gamma = \Gamma', n : T$;
(6) $\Gamma \triangleright\, !P \in \mathcal{CR}(\cong_\sigma)$ whenever P does not contain declassified actions.

The cases (1), (2), (3) are immediate.
Proof of (4). By Theorem 21 it is sufficient to prove that, given $\Gamma \triangleright P$ and $\Gamma \triangleright Q$ such that $\Gamma \triangleright P \in \mathcal{W}(\cong_\sigma)$ and $\Gamma \triangleright Q \in \mathcal{W}(\cong_\sigma)$, it holds $\Gamma \triangleright P|Q \in \mathcal{W}(\cong_\sigma)$, which comes by Proposition 48.
Proof of (5) comes by Lemma 25.
Proof of (6). By Theorem 21 it is sufficient to prove that if $\Gamma \triangleright P \in \mathcal{W}(\cong_\sigma)$ then $\Gamma \triangleright !P \in \mathcal{W}(\cong_\sigma)$. Let be $\Gamma \triangleright !P \rightsquigarrow \Gamma' \triangleright R$, then by Lemma 28 (2) $R = (\boldsymbol{\nu} n_1{:}T_1, \dots, n_k{:}T_k)(P_1| \cdots |P_s|!P)$ where $k, s \geq 0$ and $\Gamma \triangleright P \rightsquigarrow \Gamma_i \triangleright P_i$ with $\Gamma_i \subseteq \Gamma', n_1 : T_1, \dots, n_k : T_k$ for $i = 1, \dots, s$.

- Assume $\Gamma' \triangleright R \xrightarrow{\alpha}{}^{\sigma} \Gamma' \triangleright R'$ with $\alpha \in \{\overline{a}\langle b\rangle, a(b)\}$. By Lemma 46, this must come from one of the following cases, where we assume $\Delta = n_1 : T_1, \dots, n_k : T_k$:
  - there exists $i \in \{1,...,s\}$ such that $\Gamma', \Delta \triangleright P_i \xrightarrow{\alpha}{}^{\sigma} \Gamma', \Delta \triangleright P_i'$ and $R' = (\boldsymbol{\nu} n_1{:}T_1, \dots, n_k{:}T_k)(P_1| \cdots |P_i'| \cdots |P_s|!P)$. From $\Gamma \triangleright P \rightsquigarrow \Gamma_i' \triangleright P_i$, by persistence we have $\Gamma_i' \triangleright P_i \in \mathcal{W}(\cong_\sigma)$ and by the weakening lemma 44 $\Gamma', \Delta \triangleright P_i \in \mathcal{W}(\cong_\sigma)$. Then we have $\Gamma', \Delta \triangleright P_i \Longrightarrow \Gamma', \Delta \triangleright P_i''$, and $\Gamma', \Delta \vDash P_i'' \approx_\sigma P_i'$. Then $\Gamma' \triangleright R \Longrightarrow \Gamma' \triangleright (\boldsymbol{\nu} n_1{:}T_1, \dots, n_k{:}T_k)(P_1| \cdots |P_i''| \cdots |P_s|!P)$. Now, from $\Gamma', \Delta \vDash P_i'' \approx_\sigma P_i', \Gamma', \Delta \triangleright P_i' \in \mathcal{W}(\cong_\sigma), \Gamma', \Delta \triangleright P_i'' \in \mathcal{W}(\cong_\sigma)$ and $\Gamma', \Delta \triangleright P_j \in \mathcal{W}(\cong_\sigma)$ for $j \in \{1,..,i-1,i+1,..,s\}$, which follow by persistence and the weakening lemma 44, we can apply Lemma 47 (note that by hypothesis P does not contain declassified actions) obtaining $\Gamma', \Delta \vDash P_1| \cdots |P_i''| \cdots |P_s \approx_\sigma P_1| \cdots |P_i'| \cdots |P_s$. Then by Lemma 49 we also have $\Gamma', \Delta \vDash P_1| \cdots |P_i''| \cdots |P_s|!P \approx_\sigma P_1| \cdots |P_i'| \cdots |P_s|!P$, and by contextuality of $\approx_\sigma$, i.e. Proposition 36, we conclude $\Gamma' \vDash (\boldsymbol{\nu} n_1{:}T_1, \dots, n_k{:}T_k)(P_1| \cdots |P_i''| \cdots |P_s|!P) \approx_\sigma R'$.
  - Assume $\Gamma', \Delta \triangleright !P \xrightarrow{\alpha}{}^{\sigma} \Gamma', \Delta \triangleright P'|!P$ since $\Gamma', \Delta \triangleright P \xrightarrow{\alpha}{}^{\sigma} \Gamma', \Delta \triangleright P'$, and $R' = (\boldsymbol{\nu} n_1{:}T_1, \dots, n_k{:}T_k)(P_1| \cdots |P_s|P'|!P)$. By the weakening lemma 44, since $\Gamma \subseteq \Gamma', \Delta$, we have $\Gamma', \Delta \triangleright P \in \mathcal{W}(\cong_\sigma)$. Then we also have $\Gamma', \Delta \triangleright P \Longrightarrow \Gamma', \Delta \triangleright P''$, and $\Gamma', \Delta \vDash P' \approx_\sigma P''$. Then $\Gamma' \triangleright R \Longrightarrow \Gamma' \triangleright (\boldsymbol{\nu} n_1{:}T_1, \dots, n_k{:}T_k)(P_1| \cdots |P_s|P''|!P)$. We can then conclude by Lemma 47 (note that by hypothesis P does not contain declassified actions) and contextuality of $\approx_\sigma$, i.e. Proposition 36, $\Gamma' \vDash (\boldsymbol{\nu} n_1{:}T_1, \dots, n_k{:}T_k)(P_1| \cdots |P_s|P''|!P) \approx_\sigma R'$.
- $\Gamma' \triangleright R \xrightarrow{\alpha}{}^{\sigma} \Gamma', m{:}T \triangleright R'$ with $\alpha \in \{(\boldsymbol{\nu} m{:}T)\, \overline{a}\langle m\rangle, (\boldsymbol{\nu} m{:}T)\, a(m)\}$. Let distinguish two subcases:
  - $m \notin \{n_1, \dots, n_k\}$. This case follows similarly to the case shown above.
  - $m = n_j$ with $n_j \in \{n_1, \dots, n_k\}$. By Lemma 46, this must come from one of the following cases, where we assume $\Delta = n_1 : T_1, \dots, n_k : T_k$:
    (1) $\exists i \in \{1,...,s\}$ such that $\Gamma', \Delta \triangleright P_i \xrightarrow{\alpha_1}{}^{\sigma} \Gamma', \Delta \triangleright P_i'$ with $\alpha_1 \in \{\overline{a}\langle n_j\rangle, a(n_j)\}$, and $R' = (\boldsymbol{\nu} n_1{:}T_1, \dots, n_{j-1}{:}T_{j-1}, n_{j+1}{:}T_{j+1}, \dots, n_k{:}T_k)\ (P_1| \cdots |P_i'| \cdots |P_s|!P)$. From $\Gamma \triangleright P \rightsquigarrow \Gamma_i' \triangleright P_i$, by persistence we have $\Gamma_i' \triangleright P_i \in \mathcal{W}(\cong_\sigma)$ and by the weakening lemma 44 $\Gamma', \Delta \triangleright P_i \in \mathcal{W}(\cong_\sigma)$. Then $\Gamma', \Delta \triangleright P_i \Longrightarrow \Gamma', \Delta \triangleright P_i''$, and $\Gamma', \Delta \vDash P_i'' \approx_\sigma P_i'$, hence $\Gamma' \triangleright R \Longrightarrow \Gamma' \triangleright (\boldsymbol{\nu} n_1{:}T_1, \dots, n_k{:}T_k)(P_1| \cdots |P_i''| \cdots |P_s|!P)$. Now, from $\Gamma', \Delta \vDash P_i'' \approx_\sigma P_i', \Gamma', \Delta \triangleright P_i' \in \mathcal{W}(\cong_\sigma), \Gamma', \Delta \triangleright P_i'' \in \mathcal{W}(\cong_\sigma)$ and $\Gamma', \Delta \triangleright P_l \in \mathcal{W}(\cong_\sigma)$ for $l \in \{1,..,i-1,i+1,..,s\}$, which follow by persistence and the weakening lemma 44, we can apply Lemma 47 (note that by hypothesis P does not contain declassified actions) obtaining $\Gamma', \Delta \vDash P_1| \cdots |P_i''| \cdots |P_s \approx_\sigma P_1| \cdots |P_i'| \cdots |P_s$. Then by Lemma 49 $\Gamma', \Delta \vDash P_1| \cdots |P_i''| \cdots |P_s|!P \approx_\sigma P_1| \cdots |P_i'| \cdots |P_s|!P$, and by contextuality of $\approx_\sigma$, i.e. Proposition 36, we conclude $\Gamma' \vDash (\boldsymbol{\nu} n_1{:}T_1, \dots, n_k{:}T_k)(P_1| \cdots |P_i''| \cdots |P_s|!P) \approx_\sigma (\boldsymbol{\nu} n_j{:}T_j)R'$ as desired.
    (2) $R' = (\boldsymbol{\nu} n_1{:}T_1, .., n_{j-1}{:}T_{j-1}, n_{j+1}{:}T_{j+1}, .., n_k{:}T_k)(P_1| \cdots |P_s|P_{s+1}|!P)$ since $\Gamma', \Delta \triangleright !P \xrightarrow{\alpha_1}{}^{\sigma} \Gamma', \Delta \triangleright !P|P_{s+1}$ with $\alpha_1 \in \{\overline{a}\langle n_j\rangle, a(n_j)\}$, that comes from $\Gamma', \Delta \triangleright P \xrightarrow{\alpha_1}{}^{\sigma} \Gamma', \Delta \triangleright P_{s+1}$. From $\Gamma \triangleright P \in \mathcal{W}(\cong_\sigma)$,

together with $\Gamma \subseteq \Gamma', \Delta$, by the weakening lemma 44 we have that $\Gamma', \Delta \rhd P \in \mathcal{W}(\cong_\sigma)$, thus $\Gamma', \Delta \rhd P \Longrightarrow \Gamma', \Delta \rhd P''$ and $\Gamma', \Delta \vDash P'' \approx_\sigma P_{s+1}$, hence $\Gamma' \rhd R \Longrightarrow \Gamma' \rhd (\boldsymbol{\nu}\, n_1{:}T_1, \ldots, n_k{:}T_k)(P_1| \cdots |P_s|P''|!P)$. Now, from $\Gamma', \Delta \vDash P'' \approx_\sigma P_{s+1}$, $\Gamma', \Delta \rhd P'' \in \mathcal{W}(\cong_\sigma)$, $\Gamma', \Delta \rhd P_{s+1} \in \mathcal{W}(\cong_\sigma)$ and $\Gamma', \Delta \rhd P_l \in \mathcal{W}(\cong_\sigma)$ for $l \in \{1, \ldots, s\}$, which follow by persistence and the weakening lemma 44, we can apply Lemma 47 (note that by hypothesis P does not contain declassified actions) obtaining $\Gamma', \Delta \vDash P_1| \cdots |P_s|P'' \approx_\sigma P_1| \cdots |P_s|P_{s+1}$. Then by Lemma 49 $\Gamma', \Delta \vDash P_1| \cdots |P_s|P''|!P \approx_\sigma P_1| \cdots |P_s|P_{s+1}|!P$, and by contextuality of $\approx_\sigma$, i.e. Proposition 36, we can conclude that $\Gamma' \vDash (\boldsymbol{\nu}\, n_1{:}T_1, \ldots, n_k{:}T_k)(P_1| \cdots |P_s|P''|!P) \approx_\sigma (\boldsymbol{\nu} n_j{:}T_j)R'$ as desired. $\square$

References

[1] A. Sabelfeld, D. Sands, Dimensions and principles of declassification, in: Proc. IEEE Computer Security Foundations Workshop (CSFW'05), IEEE Computer Society Press, 2005, pp. 255–269.

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

[3] K. Honda, V. Vasconcelos, N. Yoshida, Secure information flow as typed process behaviour, in: Proc. of European Symposium on Programming (ESOP'00), Lecture Notes in Computer Science, vol. 1782, Springer-Verlag, 2000, pp. 180–199.

[4] K. Honda, N. Yoshida, A uniform type structure for secure information flow, in: Proc. ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'02), ACM Press, 2002, pp. 81–92.

[5] N. Kobayashi, Type-based information flow analysis for the pi-calculus, Tech. Rep. TR03-0007, Dept. Computer Science, Tokyo Institute of Technology (2003).

[6] F. Pottier, A simple view of type-secure information flow in the π-calculus, in: Proc. the 15th IEEE Computer Security Foundations Work shop, 2002, pp. 320–330.

[7] A. Bossi, C. Piazza, S. Rossi, Modelling downgrading in information flow security, in: Proc. 17th IEEE Computer Security Foundations Workshop (CSFW'04), IEEE Computer Society Press, 2004, pp. 187–201.

[8] M. Hennessy, J. Rathke, Typed behavioural equivalences for processes in the presence of subtyping, Mathematical Structures in Computer Science 14 (5) (2004) 651–684.

[9] R. Focardi, S. Rossi, Information flow security in dynamic contexts, in: Proc. IEEE Computer Security Foundations Workshop (CSFW'02), IEEE Computer Society Press, 2002, pp. 307–319.

[10] R. Focardi, R. Gorrieri, Classification of security properties (part i: information flow), in: Proc. Foundations of Security Analysis and Design (FOSAD'01), in: R. Focardi, R. Gorrieri (Eds.), Lecture Notes in Computer Science, vol. 2171, Springer-Verlag, 2001, pp. 331–396.

[11] A. Bossi, R. Focardi, C. Piazza, S. Rossi, Verifying persistent security properties, Computer Languages, Systems and Structures 30 (3–4) (2004) 231–258.

[12] A. Sabelfeld, D. Sands, Probabilistic noninterference for multi-threaded programs, in: Proc. IEEE Computer Security Foundations Workshop (CSFW'00), IEEE Computer Society Press, 2000, pp. 200–215.

[13] D. Sangiorgi, D. Walker, The pi calculus: a theory of mobile processes, Cambridge, 2001.

[14] M. Boreale, D. Sangiorgi, Bisimulation in name-passing calculi without matching, in: Proc. 13th IEEE Symposium on Logic in Computer Science (LICS'98), IEEE Computer Society Press, 1998, pp. 165–175.

[15] A. Sabelfeld, H. Mantel, Static confidentiality enforcement for distributed programs, in: Proc. of Int. Static Analysis Symposium (SAS'02), LNCS, vol. 2477, Springer-Verlag, 2002, pp. 376–394.

[16] A. Sabelfeld, D. Sands, A per model of secure iinformation flow in sequential programs, in: Proc. European Symposium on Programming (ESOP'99), Lecture Notes in Computer Science, vol. 1576, Springer-Verlag, 1999, pp. 40–58.

[17] S. Crafa, S. Rossi, A theory of noninterference for the π-calculus, in: Symposium on Trustworthy Global Computing, TGC'05, Lecture Notes in Computer Science, vol. 3705, Springer-Verlag, 2005, pp. 2–18.

[18] S. Lafrance, J. Mullins, Bisimulation-based non-deterministic admissible interference and its application to the analysis of cryptographic protocols, Electronic Notes in Theoretical Computer Science 61 (2002) 1–24.

[19] J. Mullins, Nondeterministic admissible interference, Journal of Universal Computer Science 11 (2000) 1054–1070.

[20] A.W. Roscoe, M.H. Goldsmith, What is intransitive noninterference?, in: Proc. IEEE Computer Security Foundations Workshop (CSFW'99), IEEE Computer Society Press, 1999, pp. 228–238.

[21] P. Ryan, S. Schneider, Process algebra and non-interference, Journal of Computer Security 9 (1/2) (2001) 75–103.

[22] A.D. Gordon, A.S.A. Jeffrey, Secrecy despite compromise: types, cryptography, and the pi-calculus, in: Proc. 16th International Conference on Concurrency Theory, (CONCUR'05), Lecture Notes in Computer Science, 3653, Springer-Verlag, 2005, pp. 186–201.

[23] M. Hennessy, J. Riely, Information flow vs. resource access in the asynchronous pi-calculus, ACM Transactions on Programming Languages and Systems (TOPLAS) 24 (5) (2002) 566–591.