Disjunctive Information Flow for Communicating Processes*

Ximeng Li[1], Flemming Nielson[1], Hanne Riis Nielson[1], and Xinyu Feng[2]

1 **Technical University of Denmark**
`{ximl, fnie, hrni}@dtu.dk`
2 **University of Science and Technology of China**
`xyfeng@ustc.edu.cn`

Abstract

The security validation of practical computer systems calls for the ability to specify and verify information flow policies that are dependent on data content. Such policies play an important role in concurrent, communicating systems: consider a scenario where messages are sent to different processes according to their tagging. We devise a security type system that enforces content-dependent information flow policies in the presence of communication and concurrency. The type system soundly guarantees a compositional noninterference property, which in turn guarantees a top-level security property for systems under the control of deterministic schedulers. The proofs of three main theoretical results out of all four are formalized [19] in the Coq proof assistant [8].

1998 ACM Subject Classification "F.3.1 Specifying and Verifying and Reasoning about Programs"

Keywords and phrases Information flow, Disjunctive Policies, Concurrency, Communication

Digital Object Identifier 10.4230/LIPIcs.xxx.yyy.p

1 Introduction

Language-based information flow control [27] aims to provide end-to-end guarantees against inadvertant information leakage in the execution of programs. The security enforcement is usually achieved by static type systems [30], dynamic monitoring [12], or a mixture of both [3]. The security guarantee is usually provided by noninterference properties (e.g., [26, 9]) requiring that the public parts of a system should stay invariant against variations in the confidential parts [7]. The area has gained practical impact in securing voting systems [6], cryptographic implementations such as RSA encryption (e.g., [27]), and in the end-to-end confidentiality enforcement in real-world programming languages like PHP [17] and JavaScript [14].

---

* This work was partially supported by SESAMO, a European Artemis project, and NordSecMob.

Conference title on which this volume is based on.
Editors: Billy Editor and Bill Editors; pp. 1–26
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In recent years, an emerging concern in information flow control is the enforcement of content-dependent flow policies [5, 1, 20]: different security classes are assigned to variables under different contents they hold. Content-dependent policies are useful in concurrent systems — consider processes exchanging messages whose destinations depend on their tagging. However, despite all the aforementioned existing work (all in a sequential setting), the interaction between content-dependent policies on the one hand, and concurrency and communication on the other, is largely unexplored so far.

In this paper, we enforce content-dependent information flow policies in a concurrent language, where processes make use of local variables and communicate with each other to share information. The selection of relevant policies at each program point is achieved with the help of a Hoare logic [2] component in our information flow type system. We consider synchronous communication much in the manner of a process calculus like CCS [23]. The types of communication channels act as bridges between the modular typing of the individual processes. The *presence* and *content* of communication behaviors are treated separately (e.g., [21, 18]), which leads to a more flexible confidentiality enforcement, and a more intuitive formulation of noninterference property (termed "communication-aware security") that is bisimulation-based, progress-sensitive [15], and *compositional*. This notion of noninterference further guarantees a scheduler-dependent noninterference property, under a class of deterministic schedulers.

This paper is structured as follows. A motivating example where a multiplexer and a demultiplexer communicate with each other, is introduced in Section 2. We then introduce the simple concurrent language used throughout our development in Section 3. Our security policies, termed *disjunctive policies*, are introduced in Section 4. This is followed by the presentation of our information flow type system in Section 5, and noninterference property in Section 6. The impact of deterministic schedulers is then studied in Section 7. The security of the motivating example is guaranteed by its well-typedness. We conclude and discuss certain elements of our development and related work in Section 8.

The subject reduction result of our type system (Theorem 7), the compositionality of communication-aware security (Theorem 11), and the soundness of the type system w.r.t. communication-aware security (Theorem 12), have been formally proved [19] in the Coq proof assistant [8]. This proof is also sketched in the appendix for the reader to get the intuition. The proof that communication-aware security guarantees security under deterministic scheduling (Theorem 19), is given in detail in the appendix (although not formalized in Coq).

2 Motivating Example

The MILS security architecture [25] aims to achieve controlled information flow between different system partitions that share certain resources. It has wide applications in domains such as avionics. A typical kind of shared component in MILS systems is a multiplexer (e.g., [11]) that directs confidential input from partition I to confidential output to partition II, and public input from partition III to public output to partition IV.

Consider a (simple-minded) concretization of this scenario with the processes in Figure 1. A multiplexer process (S_{M}) wraps up the source data in x_1 or x_2, along with tags 1 and 2 respectively, and forwards it over the dyadic channel c to a demultiplexer process (S_{D}). The demultiplexer will then unwrap the data and forward it to the sinks z_1 or z_2, depending on the tag value.

```
Multiplexer ($S_{\mathrm{M}}$) :  while true do          Demultiplexer ($S_{\mathrm{D}}$) :  while true do
                                    $c!(1, x_1)$;                                      $c?(y, z)$;
                                    $c!(2, x_2)$                                       if $y = 1$
                                                                                         then $z_1 := z$
                                                                                         else $z_2 := z$
```

Figure 1 The code for the multiplexer and the demultiplexer

Suppose the variable x_1 is confidential, whereas x_2 is public. The information flow analysis should then reveal that z_1 needs to be confidential, while z_2 can be public.

For modularity reasons, a *type-based analysis* needs to assign a confidentiality level to the channel c for S_{M} and S_{D} to be analyzed separately. In the demultiplexer process S_{D}, both z_1 and z_2 obtain data from c, depending on whether the tag is 1 or not. It would then be desirable for the type system to have the knowledge that *either* c is confidential and communicating $(1, \_)$, *or* c is public and communicating $(2, \_)$. This is precisely what our disjunctive policies aim to capture, in the setting of concurrent systems.

Moreover, when it is said that "c is confidential", what is actually meant is that it communicates confidential *content*. The observation of the mere *presence* of any communication action over c, without observing the content communicated, does not jeopardize the confidentiality of x_1. We thus distinguish between the presence and content of channel communication (e.g., [21, 18]), for a more fine-grained, permissive enforcement of our disjunctive policies.

3 The Language

We introduce the concurrent imperative language to be used, and specify its structural operational semantics.

Syntax: A system Σ consists of a fixed number of concurrent processes. All variables are local to their own processes, and information sharing is achieved by means of communication.

All processes are assumed to have identifiers in $\{1, 2, ...\}$. For a system Σ with the set $\mathit{Pid}(\Sigma)$ of process identifiers, its set of variables can thus be denoted by $\mathbf{Var}_\Sigma = \biguplus_{i \in \mathit{Pid}(\Sigma)} \mathbf{Var}_{\Sigma,i}$ ($\uplus$ for *disjoint union*), where the process with identifier i can only use variables from $\mathbf{Var}_{\Sigma,i}$. For communication, the set of polyadic channels is $\mathbf{PCh}$ and the set of atomic channels is $\mathbf{Ch} = \{c.1, \cdots, c.m \mid c \in \mathbf{PCh},\ \text{and } c \text{ has arity } m\}$.

We write x, y, z for variables, X for sets of variables, c for either a polyadic channel name or an atomic channel name (it will always be clear from the context which is the case), n for unspecified constants, op for unspecified arithmetic operators, rel for unspecified relational operators, and $\mathbf{tt}$ for the boolean constant denoting truth. The set of variables contained in an arithmetic expression a (resp. boolean expression b) is $\mathit{fv}(a)$ (resp. $\mathit{fv}(b)$).

Table 1 Small-step semantics of processes and systems.

$$\vdash_i \langle \mathsf{skip}; \sigma\rangle \xrightarrow{\tau} \langle \mathsf{nil}; \sigma\rangle \qquad \vdash_i \langle x := a; \sigma\rangle \xrightarrow{\tau} \langle \mathsf{nil}; \sigma[x \mapsto \mathcal{A}[\![a]\!]\sigma]\rangle$$

$$\vdash_i \langle c!\vec{a}; \sigma\rangle \xrightarrow{c!\vec{v}} \langle \mathsf{nil}; \sigma\rangle \ \text{ if } \vec{v} = \mathcal{A}[\![\vec{a}]\!]\sigma \qquad \vdash_i \langle c?\vec{x}; \sigma\rangle \xrightarrow{c?\vec{v}} \langle \mathsf{nil}; \sigma[\vec{x} \mapsto \vec{v}]\rangle$$

$$\frac{\vdash_i \langle S_1; \sigma\rangle \xrightarrow{\alpha} \langle S_1'; \sigma'\rangle}{\vdash_i \langle S_1; S_2; \sigma\rangle \xrightarrow{\alpha} \langle S_1'; S_2; \sigma'\rangle} \ \text{ if } S_1' \neq \mathsf{nil} \qquad \frac{\vdash_i \langle S_1; \sigma\rangle \xrightarrow{\alpha} \langle \mathsf{nil}; \sigma'\rangle}{\vdash_i \langle S_1; S_2; \sigma\rangle \xrightarrow{\alpha} \langle S_2; \sigma'\rangle}$$

$$\vdash_i \langle \mathsf{if}\ b\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2; \sigma\rangle \xrightarrow{\tau} \langle S_1; \sigma\rangle \ \text{ if } \mathcal{B}[\![b]\!]\sigma = \mathbf{tt}$$

$$\vdash_i \langle \mathsf{if}\ b\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2; \sigma\rangle \xrightarrow{\tau} \langle S_2; \sigma\rangle \ \text{ if } \mathcal{B}[\![b]\!]\sigma = \mathbf{ff}$$

$$\vdash_i \langle \mathsf{while}\ b\ \mathsf{do}\ S; \sigma\rangle \xrightarrow{\tau} \langle (S; \mathsf{while}\ b\ \mathsf{do}\ S); \sigma\rangle \ \text{ if } \mathcal{B}[\![b]\!]\sigma = \mathbf{tt}$$

$$\vdash_i \langle \mathsf{while}\ b\ \mathsf{do}\ S; \sigma\rangle \xrightarrow{\tau} \langle \mathsf{nil}; \sigma\rangle \ \text{ if } \mathcal{B}[\![b]\!]\sigma = \mathbf{ff}$$

$$\frac{\vdash_i \langle S_i; \sigma\rangle \xrightarrow{\alpha} \langle S_i'; \sigma'\rangle}{\langle i : S_i; \sigma\rangle \xrightarrow{\alpha}_i \langle S_i'; \sigma'\rangle} \qquad \frac{\langle \Sigma_1; \sigma_1\rangle \xrightarrow{c\rho\vec{v}}_i \langle \Sigma_1'; \sigma_1'\rangle \ \langle \Sigma_2; \sigma_2\rangle \xrightarrow{\widetilde{c\rho\vec{v}}}_j \langle \Sigma_2'; \sigma_2'\rangle}{\langle \Sigma_1 || \Sigma_2; \sigma_1 \uplus \sigma_2\rangle \xrightarrow{\tau}_{i,j} \langle \Sigma_1' || \Sigma_2'; \sigma_1' \uplus \sigma_2'\rangle} \ \text{where } \rho \in \{!, ?\}$$

$$\frac{\langle \Sigma_1; \sigma_1\rangle \xrightarrow{\alpha}_\eta \langle \Sigma_1'; \sigma_1'\rangle}{\langle \Sigma_1 || \Sigma_2; \sigma_1 \uplus \sigma_2\rangle \xrightarrow{\alpha}_\eta \langle \Sigma_1' || \Sigma_2; \sigma_1' \uplus \sigma_2\rangle} \qquad \frac{\langle \Sigma_2; \sigma_2\rangle \xrightarrow{\alpha}_\eta \langle \Sigma_2'; \sigma_2'\rangle}{\langle \Sigma_1 || \Sigma_2; \sigma_1 \uplus \sigma_2\rangle \xrightarrow{\alpha}_\eta \langle \Sigma_1 || \Sigma_2'; \sigma_1 \uplus \sigma_2'\rangle}$$

$$\frac{\langle \Sigma; \sigma\rangle \xrightarrow{\alpha}_\eta \langle \Sigma'; \sigma'\rangle}{\langle \Sigma \setminus \Omega; \sigma\rangle \xrightarrow{\alpha}_\eta \langle \Sigma' \setminus \Omega; \sigma'\rangle} \ \text{ if } ch(\alpha) \notin \Omega$$

The syntax of our language is given by:

$$\begin{array}{lcl}
a & ::= & n \mid x \mid a_1\ op\ a_2 \\
b & ::= & \mathbf{tt} \mid a_1\ rel\ a_2 \mid b_1 \wedge b_2 \mid \neg b \\
S & ::= & \mathsf{nil} \mid \mathsf{skip} \mid x := a \mid S_1; S_2 \mid \mathsf{if}\ b\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2 \mid \mathsf{while}\ b\ \mathsf{do}\ S \mid c?\vec{x} \mid c!\vec{a} \\
\Sigma & ::= & i : S_i \mid \Sigma_1 || \Sigma_2 \mid \Sigma \setminus \Omega
\end{array}$$

Sequential processes S can contain communication binders: $c!\vec{a}$ for output of the vector $\vec{a}$ of arithmetic expressions over the polyadic channel c and $c?\vec{x}$ for input from the polyadic channel c into the vector $\vec{x}$ of variables. The process nil is an inert process that cannot perform any computation.

Systems are composed of concurrent, communicating processes. The construct $i : S_i$ represents a process running the statement S_i, with process identifier i. The construct $\Sigma_1 || \Sigma_2$ represents two systems running concurrently. We require $Pid(\Sigma_1) \cap Pid(\Sigma_2) = \emptyset$ for the wellformedness of $\Sigma_1 || \Sigma_2$. Finally the construct $\Sigma \setminus \Omega$ represents the system that can perform all the input/output actions of Σ provided that those actions are not over the polyadic channels in Ω. It allows to specify whether each channel used by a process is shared with another process or with the environment[1].

Semantics: The structural operational semantics of our language is presented in Table 1. In order to handle communication, the transitions are annotated with the *action* being performed; an action α takes one of three forms: $c!\vec{v}$ (for output over c), $c?\vec{v}$ (for input over c) or τ (for the remaining cases) where $\vec{v} \in \mathbf{Val}^\star$ is the vector of values being communicated

---

[1] Readers familiar with process calculi will find $\Sigma \setminus \Omega$ similar to the restriction operator of CCS [23].

over the channel. We tacitly assume that arities match without having explicitly to require this in the semantics. The evaluation of arithmetic and boolean expressions is specified using the functions $\mathcal{A}$ and $\mathcal{B}$, respectively.

The general form of the transitions for processes is $\vdash_i \langle S; \sigma\rangle \xrightarrow{\alpha} \langle S'; \sigma'\rangle$, where i is the identifier of the process being executed, and $\sigma, \sigma' \in \mathbf{Var}_{\Sigma,i} \to \mathbf{Val}$. The transition rules are fairly standard.

Lifting the semantics to systems, we have configurations of the form $\langle \Sigma; \sigma\rangle$. It is tacitly assumed that $\sigma \in \mathbf{St}_{\boldsymbol{\Sigma}}$, where $\mathbf{St}_{\boldsymbol{\Sigma}}$ is $\mathbf{Var}_{\Sigma} \to \mathbf{Val}$. The transitions are of the form $\langle \Sigma; \sigma\rangle \xrightarrow{\alpha}_{\eta} \langle \Sigma'; \sigma'\rangle$ where η is the list of identifiers for the processes executed. For a polarity ! or ?, we have $\widetilde{!} = ?$ and $\widetilde{?} = !$. For a mapping A, we write $\mathbf{D}_A$ for its domain. For two mappings A and B such that $\mathbf{D}_A \cap \mathbf{D}_B = \emptyset$, we denote by $A \uplus B$ the mapping with domain $\mathbf{D}_A \uplus \mathbf{D}_B$, such that $(A \uplus B)(i) = \begin{cases} A(i) & (\text{if } i \in \mathbf{D}_A) \\ B(i) & (\text{if } i \in \mathbf{D}_B) \end{cases}$. Then the transition rules for systems are mostly self-explanatory. In particular, the final rule says that $\langle \Sigma \setminus \Omega;\, \sigma\rangle$ can perform an action α of $\langle \Sigma; \sigma\rangle$ if the channel used by α is not in Ω.

▶ **Example 1.** The combination of the multiplexer and demultiplexer considered in Section 2 can be represented by the system $\Sigma_{\mathrm{MD}} = (1 : S_{\mathrm{M}} \,||\, 2 : S_{\mathrm{D}}) \setminus \{c\}$. ◀

4 Security Policies

Each confidentiality level is taken from the two-point confidentiality lattice $\mathbf{Lab}_{\mathbf{S}} = (\{L, H\}, \sqsubseteq)$ where $L \sqsubseteq H$, throughout our development. The level H (resp. L) represents high (resp. low) confidentiality. A generalization to arbitrary security lattices is straightforward but induces notational sophistication; hence we stay with $\mathbf{Lab}_{\mathbf{S}}$.

For systems Σ, we introduce policy environments $\mathcal{P}$ such that for each $i \in \mathit{Pid}(\Sigma)$, $\mathcal{P}(i)$ is a set of policies for the variables in $\mathbf{Var}_{\Sigma,i}$. Each variable policy $P \in \mathcal{P}(i)$ consists of two components, $P_{\mathsf{S}} : \mathbf{Var}_{\Sigma,i} \to \mathbf{Lab}_{\mathbf{S}}$ and $P_{\mathsf{V}} : \mathbf{Lab}_{\mathbf{F}}$, where P_{S} contains the confidentiality level of each variable in $\mathbf{Var}_{\Sigma,i}$ and P_{V} is a logical formula describing the possible values of these variables. Given a set $X \subseteq \mathbf{Var}_{\Sigma,i}$, we define $P_{\mathsf{S}}[X]$ as $\bigsqcup_{x \in X} P_{\mathsf{S}}(x)$. We denote by $P \mathrel{\mathsf{E}} \mathcal{P}$ the fact that $P \in \{(\biguplus_{i \in \mathbf{D}_{\mathcal{P}}} P_{i\mathsf{S}}, \bigwedge_{i \in \mathbf{D}_{\mathcal{P}}} P_{i\mathsf{V}}) \mid \forall i \in \mathbf{D}_{\mathcal{P}} : P_i \in \mathcal{P}(i)\}$.

We also allow to specify a (global) set $\mathcal{P}^{\mathrm{ch}}$ of channel policies. The set $\mathcal{P}^{\mathrm{ch}}$ has a distinguished member $P^{\circ} : \mathbf{PCh} \to \mathbf{Lab}_{\mathbf{S}}$ that gives the confidentiality level of the “presence” of communications over each polyadic channel. Apart from P°, there is at least one content policy $P^{\bullet} \in \mathcal{P}^{\mathrm{ch}}$. Each $P^{\bullet}$ has two components $P^{\bullet}_{\mathsf{S}} : \mathbf{Ch} \to \mathbf{Lab}_{\mathbf{S}}$ and $P^{\bullet}_{\mathsf{V}} : \mathbf{Ch} \to \mathbf{Lab}_{\mathbf{V}}$, where for each atomic channel c, $P^{\bullet}_{\mathsf{S}}(c)$ is the confidentiality level of the communication contents over c, and $P^{\bullet}_{\mathsf{V}}(c)$ is the set of values potentially communicated over c.

For a variable policy P, P_{V} can capture “relational constraints” between different variables. For instance, given an output $c!(x - y)$, it is valid to have $P_{\mathsf{V}} = (p(x) = p(y))$ where $p(-)$ is the *parity* function. Correspondingly, a channel policy $P^{\bullet}$ may come with *the set of even numbers* for $P^{\bullet}_{\mathsf{V}}(c.1)$.

Each $\mathcal{P}(i)$ $(i \in \mathit{Pid}(\Sigma))$ resembles a disjunctive formula of variable policies (where each policy is a conjunction over the confidentiality and content information provided by it). The same analogy is enjoyed by the set $\mathcal{P}^{\mathrm{ch}}_{\bullet} = \{P^{\bullet} \mid P^{\bullet} \in \mathcal{P}^{\mathrm{ch}}\}$ of content policies for channels. Hence we term our policies *disjunctive policies*. Hereafter, the parameterization on $\mathcal{P}^{\mathrm{ch}}$ in our formulations will often be elided since $\mathcal{P}^{\mathrm{ch}}$ is treated as a global constant. The distinguished presence policy $P^{\circ} \in \mathcal{P}^{\mathrm{ch}}$ will be left implicit for the same reason.

▶ **Example 2.** We will use the following policies for the multiplexer example presented in Section 2, where $\mathbb{Z}$ is the set of all integers.

$$\begin{aligned}
\mathcal{P}_{\mathrm{MD}}(1) &= \{P_{\mathrm{m}} = (x_1 : H;\ x_2 : L,\ \mathbf{tt})\}\\
\mathcal{P}_{\mathrm{MD}}(2) &= \{P_{\mathrm{d}}^{1} = (y : L;\ z : H;\ z_1 : H;\ z_2 : L,\ y = 1),\\
&\qquad P_{\mathrm{d}}^{2} = (y : L;\ z : L;\ z_1 : H;\ z_2 : L,\ y \neq 1)\}
\end{aligned}$$

$$\begin{aligned}
\mathcal{P}_{\mathrm{MD}}^{\mathrm{ch}} = \{&P^{\circ} = (c : L),\\
&P_1^{\bullet} = (c.1 : L;\ c.2 : H,\ c.1 : \{1\};\ c.2 : \mathbb{Z}),\\
&P_2^{\bullet} = (c.1 : L;\ c.2 : L,\ c.1 : \{2\};\ c.2 : \mathbb{Z})\}
\end{aligned}$$

For convenience of reference, the policies are named. Take the policy $P_{\mathrm{m}} \in \mathcal{P}(1)$ for example, we have $P_{\mathrm{m}\mathsf{V}} = \mathbf{tt}$ and $P_{\mathrm{m}\mathsf{S}} = [x_1 \mapsto H][x_2 \mapsto L]$. The syntax with colons and semi-colons is used for confidentiality policy components such as $P_{\mathrm{m}\mathsf{S}}$ for conciseness. ◀

We next define the satisfaction of variable policies by states ($\sigma \models P$), and the satisfaction of channel policies by actions ($\alpha \models_{\rho} P^{\bullet}$). Our concern here is what policies are relevant according to the memory content or communication content. For a vector $\vec{v}$, we write $|\vec{v}|$ for its total number of components, and v_j for the j-th one.

▶ **Definition 3** (Satisfaction).

$$\begin{aligned}
\sigma \models P &\ \triangleq\ \sigma \models P_{\mathsf{V}} \quad (\sigma \text{ is a model of the formula } P_{\mathsf{V}})\\
\alpha \models_{\rho} P^{\bullet} &\ \triangleq\ P^{\bullet} \in \mathcal{P}^{\mathrm{ch}} \wedge \forall c, \vec{v} : \alpha = c\rho\vec{v} \Rightarrow \forall j \ s.t.\ 1 \leq j \leq |\vec{v}| : v_j \in P^{\bullet}_{\mathsf{V}}(c.j)
\end{aligned}$$

For channel policies, the satisfaction relation $\models_{\rho}$ is parameterized with a polarity ρ. The intuition is that the check on content is turned on only when the polarity of α is ρ. In this case it is required that the j-th value communicated over the polyadic channel of α should indeed be described by the value component of $P^{\bullet}$ for the atomic channel $c.j$. If α does not have the polarity ρ, then nothing is required.

▶ Remark. The way our policies work is the easiest to understand when the content information of different policies is *disjoint* as in Example 2 (although this is not a restriction technically), In this case, the policy in effect at each point of execution will be uniquely determined by the state or the communicated vector of values, via the satisfaction relation.

5 The Type System

We specify a type system for ensuring that a system Σ respects the information flow policies given by $\mathcal{P}$ (such that $\mathbf{D}_{\mathcal{P}} = \mathit{Pid}(\Sigma)$) and $\mathcal{P}^{\mathrm{ch}}$. To deal with the value components P_{V} of policies P, the type system is integrated with a Hoare logic for reasoning about the values of variables [2]. The typing rules for processes and systems are specified in Table 2 in order.

The typing of processes: The typing judgements for processes are of the form $X, l_1 \vdash_{\mathcal{K}} \{\phi\}\, S\, \{\phi'\} : Y, l_2$ where X is a set of variables that may incur implicit flows [10], Y is a set of variables whose information can be leaked through *progress*, $\mathcal{K}$ is the set of variable policies for the process S, and ϕ and ϕ' are the pre- and post-conditions of S in the form of logical formulae over the variables local to S. In addition, l_1 and l_2 are the levels of information that can be leaked through blocked communication attempts (due to inability of

Table 2 Information flow type system for processes and systems.

$$X, l \vdash_{\mathcal{K}} \{\phi\}\, \mathsf{nil}\, \{\phi\} : \emptyset, L \qquad\qquad X, l \vdash_{\mathcal{K}} \{\phi\}\, \mathsf{skip}\, \{\phi\} : \emptyset, L$$

$$\begin{array}{l} X, l \vdash_{\mathcal{K}} \{\phi[a/x]\}\, x := a\, \{\phi\} : \emptyset, L \\ \quad \text{if } \forall P \in \mathcal{K} : P_{\mathsf{V}} \wedge \phi[a/x] \Rightarrow \exists P' \in \mathcal{K} : P[x \mapsto l \sqcup P_{\mathsf{S}}[\mathit{fv}(a) \cup X]]_{\mathsf{S}} \preceq P'[a/x]_{\mathsf{V}} \end{array}$$

$$\frac{X, l \vdash_{\mathcal{K}} \{\phi\}\, S_1\, \{\rho\} : Y_1, l_1 \quad X \cup Y_1, l \sqcup l_1 \vdash_{\mathcal{K}} \{\rho\}\, S_2\, \{\psi\} : Y_2, l_2}{X, l \vdash_{\mathcal{K}} \{\phi\}\, S_1; S_2\, \{\psi\} : Y_1 \cup Y_2, l_1 \sqcup l_2}$$

$$\frac{X \cup \mathit{fv}(b), l \vdash_{\mathcal{K}} \{\phi \wedge b\}\, S_1\, \{\psi\} : Y_1, l_1 \quad X \cup \mathit{fv}(b), l \vdash_{\mathcal{K}} \{\phi \wedge \neg b\}\, S_2\, \{\psi\} : Y_2, l_2}{X, l \vdash_{\mathcal{K}} \{\phi\}\, \mathsf{if}\ b\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2\, \{\psi\} : Y_1 \cup Y_2 \cup \mathit{fv}(b), l_1 \sqcup l_2}$$

$$\frac{Y, l \vdash_{\mathcal{K}} \{\phi \wedge b\}\, S\, \{\phi\} : Y, l}{X, l \vdash_{\mathcal{K}} \{\phi\}\, \mathsf{while}\ b\ \mathsf{do}\, S\, \{\phi \wedge \neg b\} : Y, l} \quad \text{if } X \cup \mathit{fv}(b) \subseteq Y$$

$$\begin{array}{l} X, l \vdash_{\mathcal{K}} \{\phi\}\, c!\vec{a}\, \{\phi\} : \emptyset, l' \ \ \text{if } l \sqsubseteq P^{\circ}(c) \sqsubseteq l' \text{ and} \\ \ \forall P \in \mathcal{K} : P_{\mathsf{V}} \wedge \phi \Rightarrow (P_{\mathsf{S}}[X] \sqsubseteq P^{\circ}(c) \wedge\ \exists P' \in \mathcal{K}, P^{\bullet} \in \mathcal{P}^{\mathrm{ch}} : \\ \qquad\qquad P[(c.j \mapsto P_{\mathsf{S}}(a_j))_j]_{\mathsf{S}} \preceq (P'_S \uplus P^{\bullet}_{\mathsf{S}},\, P'_{\mathsf{V}} \wedge \bigwedge_j a_j \in P^{\bullet}_{\mathsf{V}}(c.j))) \end{array}$$

$$\begin{array}{l} X, l \vdash_{\mathcal{K}} \{\forall \vec{x} : \phi\}\, c?\vec{x}\, \{\phi\} : \emptyset, l' \ \ \text{if } l \sqsubseteq P^{\circ}(c) \sqsubseteq l' \text{ and} \\ \ \forall P \in \mathcal{K} : P_{\mathsf{V}} \wedge (\forall \vec{x} : \phi) \Rightarrow (P_{\mathsf{S}}[X] \sqsubseteq P^{\circ}(c) \wedge\ \forall P^{\bullet} \in \mathcal{P}^{\mathrm{ch}} : \forall \vec{v}\ s.t. \bigwedge_j v_j \in P^{\bullet}(c.j) : \\ \qquad\qquad \exists P' \in \mathcal{K} : P[(x_j \mapsto P^{\bullet}_{\mathsf{S}}(c.j) \sqcup P^{\circ}(c))_j]_{\mathsf{S}} \preceq P'[(\underline{v_j}/x_j)_j]_{\mathsf{V}}) \end{array}$$

$$\frac{X', l'_1 \vdash_{\mathcal{K}} \{\phi'\}\, S\, \{\psi'\} : Y', l'_2}{X, l_1 \vdash_{\mathcal{K}} \{\phi\}\, S\, \{\psi\} : Y, l_2} \quad \text{if} \quad \begin{array}{l} (\phi \Rightarrow \phi') \wedge (\psi' \Rightarrow \psi) \wedge \\ X \subseteq X' \wedge Y' \subseteq Y \wedge l_1 \sqsubseteq l'_1 \wedge l'_2 \sqsubseteq l_2 \end{array}$$

$$\frac{\emptyset, L \vdash_{\mathcal{K}} \{\phi\}\, S_i\, \{\psi\} : Y, l'}{[i \mapsto \mathcal{K}] \vdash \{[i \mapsto \phi]\}\, i : S_i\, \{[i \mapsto \psi]\}} \text{ if } \mathit{nip}(\mathcal{K}) \quad \frac{\mathcal{P} \vdash \{\Phi\}\, \Sigma\, \{\Psi\}}{\mathcal{P} \vdash \{\Phi\}\, \Sigma \setminus \Omega\, \{\Psi\}} \quad \frac{\mathcal{P}_1 \vdash \{\Phi_1\}\, \Sigma_1\, \{\Psi_1\} \ \ \mathcal{P}_2 \vdash \{\Phi_2\}\, \Sigma_2\, \{\Psi_2\}}{\mathcal{P}_1 \uplus \mathcal{P}_2 \vdash \{\Phi_1 \uplus \Phi_2\}\, \Sigma_1 || \Sigma_2\, \{\Psi_1 \uplus \Psi_2\}}$$

synchronization), before reaching S, and within S, respectively. The levels l_1 and l_2 become H when encountering communication channels whose presence levels are H.

In Table 2, $P \preceq P'$ represents $P_{\mathsf{S}} \sqsubseteq P'_{\mathsf{S}} \wedge P_{\mathsf{V}} \Rightarrow P'_{\mathsf{V}}$, where $P_{\mathsf{S}} \sqsubseteq P'_{\mathsf{S}}$ if and only if $\forall u \in \mathbf{D}_{P_{\mathsf{S}}} \cap \mathbf{D}_{P'_{\mathsf{S}}} : P_{\mathsf{S}}(u) \sqsubseteq P'_{\mathsf{S}}(u)$. We write $P[x \mapsto l]_{\mathsf{S}}$ for $(P_{\mathsf{S}}[x \mapsto l], P_{\mathsf{V}})$, which is an update if $x \in \mathbf{D}_{P_{\mathsf{S}}}$ and an extension otherwise, $P[u/x]_{\mathsf{V}}$ for $(P_{\mathsf{S}}, P_{\mathsf{V}}[u/x])$ where u is an arithmetic expression, $P \wedge f$ for $(P_{\mathsf{S}}, P_{\mathsf{V}} \wedge f)$ where f is a logical formula, and $\underline{v}$ for the numeral of the value v.

The Hoare logic part of the type system is fairly simple since all variables are local. Most typing rules strengthen a precondition ϕ to the formula $\phi \wedge P_{\mathsf{V}}$ that allows to select the relevant variable policies P using their content information P_{V}. We elaborate on the rules for assignment, output and input.

The typing rule for assignment requires the existence of a post-policy P' for each selected pre-policy P. This policy P' should satisfy $l \sqcup P_{\mathsf{S}}[\mathit{fv}(a) \cup X] \sqsubseteq P'_{\mathsf{S}}(x)$, and for all variables y different than x, $P_{\mathsf{S}}(y) \sqsubseteq P'_{\mathsf{S}}(y)$ should hold. Requiring $l \sqsubseteq P'_{\mathsf{S}}(x)$ and $P_{\mathsf{S}}[X] \sqsubseteq P'_{\mathsf{S}}(x)$ is to capture *implicit flows* [10]. On the other hand, under the pre-condition $P_{\mathsf{V}} \wedge \phi[a/x]$, it is required that $P_{\mathsf{V}} \Rightarrow P'_{\mathsf{V}}[a/x]$. In other words, $P_{\mathsf{V}} \wedge \phi[a/x] \Rightarrow P'_{\mathsf{V}}[a/x]$ should hold. This guarantees that for a local state σ satisfying P and the precondition $\phi[a/x]$, the post state derived from σ after the assignment satisfies the post policy P'.

▶ **Example 4.** We have $\{y\}, L \vdash_{\mathcal{P}_{\mathrm{MD}}(2)} \{y = 1\}\, z_1 := z\, \{y = 1\} : \emptyset, L$ for the assignment $z_1 := z$ in the demultiplexer process S_{D} of Figure 1. Essentially, it needs to be shown that no matter if P is instantiated with P^1_{d} or P^2_{d}, we can find an appropriate policy in $\mathcal{P}_{\mathrm{MD}}(2)$ for the instantiation of P', satisfying the side conditions of the typing rule for assignment.

First instantiate P with P^1_{d}. We still use P^1_{d} for P', and the side condition specializes to $y = 1 \Rightarrow P^1_{\mathsf{d}}[z_1 \mapsto L \sqcup P^1_{\mathsf{ds}}[\{y, z\}]] \preceq P^1_{\mathsf{d}}[z/z_1]_{\mathsf{V}}$. This condition further expands to the following, which holds.

$$\begin{aligned} y = 1 \quad \Rightarrow \quad & ((y : L; z : H; z_1 : H; z_2 : L)[z_1 \mapsto H],\ y = 1) \\ & \preceq ((y : L; z : H; z_1 : H; z_2 : L),\ (y = 1)[z/z_1]). \end{aligned}$$

Next instantiate P with P^2_{d}. The side condition specializes to $y = 1 \wedge y \neq 1 \Rightarrow ...$, which vacuously holds. ◀

The typing rule for output imposes the constraint $l \sqsubseteq P^\circ(c) \sqsubseteq l'$. Here $P^\circ(c) \sqsubseteq l'$ takes care of the possibility for the output to be blocked by the environment (in line with the use of synchronous communication, the treatment of output is "symmetric" to that of input; hence the possiblity of blocked output is also considered). In more detail, the presence/absence of the output can leak information if the subsequent computation is not kept confidential. This kind of leakage is in a sense analogous to the leakage created by looping. On the other hand, $l \sqsubseteq P^\circ(c)$ takes care of the possibility that a previously blocked communication can be revealed through the indirect blockage (absence) of the current output. Next, the constraint $P_{\mathsf{S}}[X] \sqsubseteq P^\circ(c)$ is concerned with the implicit flows from conditionals having variables in X to the *presence* of the output. Finally, the seemingly involved constraint $P[(c.j \mapsto P_{\mathsf{S}}(a_j))_j]_{\mathsf{S}} \preceq (P'_S \uplus P^\bullet_{\mathsf{S}}, P'_{\mathsf{V}} \wedge \bigwedge_j a_j \in P^\bullet_{\mathsf{V}}(c.j))$ can be understood by comparing the output $c!\vec{a}$ to the assignment $c := \vec{a}$.

The typing rule for input uses constraints about the presence label $P^\circ(c)$ of the channel c in a way similar to the rule for output does. Its last constraint $P[(x_j \mapsto P^\bullet_{\mathsf{S}}(c.j) \sqcup P^\circ(c))_j]_{\mathsf{S}} \preceq P'[(v_j/x_j)_j]_{\mathsf{V}}$, requires $P^\circ(c) \sqsubseteq P'_{\mathsf{S}}(x_j)$ for all $j \in \{1, 2, ..., |c|\}$, because the presence of the input leads to the modification of the variable x_j. The remaining part of this constraint can be understood by comparing the input $c?\vec{x}$ to the assignment $\vec{x} := c$.

We remark on the typing rule for if, where the set $fv(b)$ is unioned into the "progress set", resulting in $fv(b) \cup Y_1 \cup Y_2$. This guarantees a noninterference condition where two systems advance in a manner close to "lock-step" execution, thereby facilitating the articulation of the security guarantees under scheduling. Similar treatment of the "termination effects" of if can be found in [4, 28].

The typing of systems: The typing judgements for systems are of the form $\mathcal{P} \vdash \{\Phi\}\,\Sigma\,\{\Psi\}$. Here $\mathcal{P}$ is a policy environment for the system Σ, and for each $i \in Pid(\Sigma)$, $\Phi(i)$ and $\Psi(i)$ are the pre-condition and post-condition, respectively, for the process with identifier i in Σ. We also denote by $\mathbf{T}^\Sigma$ the mapping such that $\mathbf{D}_{\mathbf{T}^\Sigma} = Pid(\Sigma)$ and $\forall i \in \mathbf{D}_{\mathbf{T}^\Sigma} : \mathbf{T}^\Sigma(i) = \mathbf{tt}$. The typing rules follow patterns that are fairly straightforward.

The side condition $nip(\mathcal{K})$ of the rule for $i : S_i$ is a "healthiness" constraint saying that the choice of policies cannot be decided by confidential information. This is desirable since confidentiality levels have *access control* implications. A public observer would be able to deduce information about confidential variables based on whether it is allowed to access the values of certain variables, if confidential information had interference on the policies in use.

The predicate $nip(-)$ is expressed with the help of the notations $\sigma_1 \overset{\mathcal{K}}{=} \sigma_2$ and $c\rho'\vec{v}_1 \underset{\rho}{\overset{\mathcal{K}}{=}} c\rho'\vec{v}_2$. The notation $\sigma_1 \overset{\mathcal{K}}{=} \sigma_2$ represents that σ_1 and σ_2 have the same domain and map each variable that is *low with respect to every policy in* $\mathcal{K}$ to the same value. Similarly, $c\rho'\vec{v}_1 \underset{\rho}{\overset{\mathcal{K}}{=}} c\rho'\vec{v}_2$ says that if ρ is the same as ρ', then the atomic channels of c that are *low with respect to every* *policy in* $\mathcal{K}$ should communicate the same values.

▶ **Definition 5** (Low equivalence parameterized by sets of policies)**.**

$$\begin{aligned} \sigma_1 \overset{\mathcal{K}}{=} \sigma_2 \quad &\triangleq \quad \mathbf{D}_{\sigma_1} = \mathbf{D}_{\sigma_2} \wedge \forall x \in \mathbf{D}_{\sigma_1} : (\forall P \in \mathcal{K} : P_{\mathsf{S}}(x) = L) \Rightarrow \sigma_1(x) = \sigma_2(x) \\ c\rho'\vec{v}_1 \underset{\rho}{\overset{\mathcal{K}}{=}} c\rho'\vec{v}_2 \quad &\triangleq \quad \forall j : (\rho = \rho' \wedge \forall P \in \mathcal{K} : P_{\mathsf{S}}(c.j) = L) \Rightarrow v_{1j} = v_{2j} \end{aligned}$$

Formally, we have:

$$\begin{aligned} \mathit{nip}(\mathcal{K}) \quad \triangleq \quad &\forall \sigma_1, \sigma_2 : \sigma_1 \overset{\mathcal{K}}{=} \sigma_2 \Rightarrow (\forall P \in \mathcal{K} : \sigma_1 \models P \Leftrightarrow \sigma_2 \models P) \wedge \\ &\forall c, \vec{v}, \vec{v}' : c!\vec{v} \underset{!}{\overset{\mathcal{P}^{\mathrm{ch}}_{\bullet}}{=}} c!\vec{v}' \Rightarrow (\forall P^{\bullet} : c!\vec{v} \models_! P^{\bullet} \Leftrightarrow c!\vec{v}' \models_! P^{\bullet}). \end{aligned}$$

▶ **Example 6.** The system Σ_{MD} of Example 1 can be typed using the policies given in Example 2. It is not difficult to verify that $\mathit{nip}(\mathcal{P}_{\mathrm{MD}}(1))$ and $\mathit{nip}(\mathcal{P}_{\mathrm{MD}}(2))$ hold, and that $\mathcal{P}_{\mathrm{MD}} \vdash \{\mathbf{T}^{\Sigma_{\mathrm{MD}}}\}\, \Sigma_{\mathrm{MD}}\, \{\mathbf{T}^{\Sigma_{\mathrm{MD}}}\}$ can be established. ◀

Subject reduction: Subject reduction [24] ensures that a type system is technically safe, by asserting that well-typedness of a system is preserved under its execution. In Theorem 7, $\sigma \models \Phi$ represents $\forall i \in \mathbf{D}_{\Phi} : \sigma \models \Phi(i)$ and all un-quantified symbols are implicitly universally quantified. When an input is performed, the existence of channel policies describing the values received are relied on to ensure the satisfaction of the pre-condition Φ' of the derived system Σ', by the resulting state σ'.

▶ **Theorem 7** (Subject Reduction)**.** *If* $\mathcal{P} \vdash \{\Phi\}\, \Sigma\, \{\Psi\}$, $\langle \Sigma; \sigma \rangle \xrightarrow{\alpha}_{\eta} \langle \Sigma'; \sigma' \rangle$, $\sigma \models \Phi$, $P \mathrel{\mathsf{E}} \mathcal{P}$ *and* $\sigma \models P$*, then*

1. $\exists P^{\bullet} : \alpha \models_! P^{\bullet}$*, and*
2. *if* $\exists P^{\bullet} : \alpha \models_? P^{\bullet}$*, then* $\exists P' \mathrel{\mathsf{E}} \mathcal{P}, \Phi' : \mathcal{P} \vdash \{\Phi'\}\, \Sigma'\, \{\Psi\} \wedge \sigma' \models \Phi' \wedge \sigma' \models P'$.

6 Noninterference

We introduce a bisimulation-based, compositional noninterference property that accounts for both the communications performed by a system and the modification of memory states. We prove that this noninterference property is enforced by the information flow type system presented in Section 5. Some auxiliary notations are first presented in Figure 2, where Definition 3 and Definition 5 are used.

We extend our transition labels α with *inaction* ϵ and *suspension* $\square$. We also introduce *action schemas* β where the inputs come with holes rather than data. The idea is that the data to be received is under the environment's control. We write $C \overset{\alpha}{\twoheadrightarrow}_{\eta} C'$ to represent that there *may be* a transition from configuration C to configuration C' over processes in η.

Low-equivalence of configurations is defined with respect to particular policies $P \mathrel{\mathsf{E}} \mathcal{P}$ in Figure 2. The main constraints are that P should be satisfied by the states of the two configurations, and that the values of variables that are low under P_{S} should be equal.

To be able to relate the communications performed in the two executions in our bisimulation-based property, we introduce a notion of low equivalence $(\underset{\rho}{\overset{P^{\bullet}}{\sim}})$ between actions α and actions/action schemas γ. The relation $\underset{\rho}{\overset{P^{\bullet}}{\sim}}$ is the smallest one satisfying the rules in Figure 2.

Actions and action schemas:

$$\begin{aligned}
\alpha &::= c!\vec{v} \mid c?\vec{v} \mid \tau \mid \epsilon \mid \Box \\
\beta &::= c!\vec{v} \mid c?[\,] \mid \tau \mid \epsilon \mid \Box \\
\gamma &::= \alpha \mid \beta
\end{aligned}$$

Decorated transitions:

$$\begin{aligned}
C \overset{\alpha}{\twoheadrightarrow}_{\eta} C' \;\triangleq\;\; & C \xrightarrow{\alpha}_{\eta} C' \;\vee \\
& \alpha = \epsilon \wedge C' = C \;\vee \\
& (\forall \alpha' : C \xrightarrow{\alpha'}_{\eta} \Rightarrow P^{\circ}(ch(\alpha')) = H) \wedge \alpha = \Box \wedge C' = C
\end{aligned}$$

Low equivalence of configurations:

$$\langle \Sigma_1; \sigma_1 \rangle \overset{P}{\underset{\mathcal{P}}{=}} \langle \Sigma_2; \sigma_2 \rangle \;\triangleq\; P \mathrel{\mathsf{E}} \mathcal{P} \,\wedge\, Pid(\Sigma_1) = Pid(\Sigma_2) = \mathbf{D}_{\mathcal{P}} \,\wedge\, \sigma_1 \models P \,\wedge\, \sigma_2 \models P \,\wedge\, \sigma_1 \overset{\{P\}}{=} \sigma_2$$

Low equivalence of actions and actions/action schemas ($\alpha \overset{P^{\bullet}}{\underset{\rho}{\sim}} \gamma$):

$$\begin{aligned}
& c\rho'\vec{v} \overset{P^{\bullet}}{\underset{\rho}{\sim}} \epsilon \;\text{ if } P^{\circ}(c) = H \wedge c\rho'\vec{v} \models_{\rho} P^{\bullet} \\
& c\rho'\vec{v}_1 \overset{P^{\bullet}}{\underset{\rho}{\sim}} c\rho'\vec{v}_2 \;\text{ if } P^{\circ}(c) = L \wedge c\rho'\vec{v}_1 \models_{\rho} P^{\bullet} \wedge c\rho'\vec{v}_2 \models_{\rho} P^{\bullet} \wedge c\rho'\vec{v}_1 \overset{\{P^{\bullet}\}}{\underset{\rho}{=}} c\rho'\vec{v}_2 \\
& \tau \overset{P^{\bullet}}{\underset{\rho}{\sim}} \tau \\
& \tau \overset{P^{\bullet}}{\underset{\rho}{\sim}} \Box
\end{aligned}$$

Hole filling ($\beta(\vec{v}')$):

$$c!\vec{v}(\vec{v}') = c!\vec{v} \qquad c?[\,](\vec{v}') = c?\vec{v}' \qquad \beta(\vec{v}') = \beta, \text{ if } \beta \in \{\tau, \epsilon, \Box\}$$

Figure 2 Auxiliary definitions for noninterference

Concerning the "presence" of communication, $\alpha \overset{P^{\bullet}}{\underset{\rho}{\sim}} \gamma$ requires that a communication with confidential presence should correspond to inaction (ϵ), which implies among others the absence of communication on the same channel[2]. It is worth pointing out that the $\vec{v}_2$ in the same definition can be the unary vector $[\,]$. Although Definition 3 has not been explicitly extended to take care of holes, the bisimulation of Definition 9 will use $\alpha \overset{P^{\bullet}}{\underset{\rho}{\sim}} \gamma$ in such a way that the check $[\,] \in P^{\bullet}_{\mathsf{V}}(c.1)$ can never be reached.

It is most ideal for a τ to correspond to (be simulated by) a τ. This is described by the third line in the definition of $\alpha \overset{P^{\bullet}}{\underset{\rho}{\sim}} \gamma$. However, not all intuitively secure programs adhere to this strict pattern. Consider degenerate policy environments $\mathcal{P}_0$ and $\mathcal{P}_0^{\mathrm{ch}}$ such that $\mathcal{P}_0(1) = \{(h : H,\ \mathbf{tt})\}$, $\mathcal{P}_0(2) = \{(h' : H,\ \mathbf{tt})\}$, and $\mathcal{P}_0^{\mathrm{ch}} = \{(c : H; c' : H,\ \mathbb{Z}),\ (c.1 : H; c'.1 : H,\ \mathbb{Z})\}$. The following system is intuitively secure; however, the τ action arising out of skip may correspond to the confidential communication $c!2$, rather than another τ.

$$1 : \mathsf{if}\ h > 1\ \mathsf{then}\ \mathsf{skip}\ \mathsf{else}\ c!2$$

Similarly, the following system is intuitively secure. However, the τ action produced by synchronization may correspond to suspension of execution, since $c'!2$ cannot synchronize

---

[2] This pattern is reminiscent of the "Weak bisimulation up to H" by Focardi and Rossi [13].

with any communication binder.

$$(1 : \mathsf{if}\ h > 1\ \mathsf{then}\ c!2\ \mathsf{else}\ c'!2 \mid\mid 2 : c?h') \setminus \{c, c'\}$$

The two examples above thus illustrate the need for $\tau \overset{P^\bullet}{\underset{\rho}{\sim}} \square$.

We are now in a position to define our noninterference property, termed *communication-aware security* (CA-security). In Definition 8, $- \overset{\text{com}}{\underset{\mathcal{P}}{\sim}} -$ is the union of all *communication-aware bisimulations* (CA-bisimulations) that are in turn characterized in Definition 9.

▶ **Definition 8** (CA-Security). $Sec_{\text{com}}(\Sigma, \mathcal{P})$ if and only if for all σ_1, σ_2, and P, if $\langle \Sigma; \sigma_1 \rangle \overset{P}{\underset{\mathcal{P}}{=}} \langle \Sigma; \sigma_2 \rangle$, then $(\langle \Sigma; \sigma_1 \rangle, P) \overset{\text{com}}{\underset{\mathcal{P}}{\sim}} (\langle \Sigma; \sigma_2 \rangle, P)$.

▶ **Definition 9** (CA-Bisimulation).
A CA-bisimulation $R_{\mathcal{P}}$ is a symmetric relation such that
$(C_1, P)\ R_{\mathcal{P}}\ (C_2, P)$ implies $C_1 \overset{P}{\underset{\mathcal{P}}{=}} C_2$ and the following:

$$\begin{aligned}
&\forall \alpha, \eta, C_1'\ s.t.\ C_1 \xrightarrow{\alpha}_\eta C_1' : \\
&\quad \exists P_!^\bullet, \beta : \alpha \overset{P_!^\bullet}{\underset{!}{\sim}} \beta\ \wedge \\
&\qquad \forall P_?^\bullet, \vec{v}\ s.t.\ \alpha \overset{P_?^\bullet}{\underset{?}{\sim}} \beta(\vec{v}) : \\
&\qquad\quad \exists C_2', P' : C_2 \overset{\beta(\vec{v})}{\twoheadrightarrow}_\eta C_2'\ \wedge\ (C_1', P')\, R_{\mathcal{P}}\, (C_2', P').
\end{aligned}$$

In words, a symmetric relation $R_{\mathcal{P}}$ qualifies as a CA-bisimulation if for a pair (C_1, P) and (C_2, P) related by $R_{\mathcal{P}}$, and a transition performing action α from C_1, involving processes in η, there exists a policy $P_!^\bullet$ and an action schema β low-equivalent to α concerning output, and for all value vectors $\vec{v}$ and policies $P_?^\bullet$ such that $\beta(\vec{v})$ is low-equivalent to α concerning input, there exists a simulation of $\xrightarrow{\alpha}_\eta$ by $\overset{\beta(\vec{v})}{\twoheadrightarrow}_\eta$ from C_2, and a policy P' whose pairings with the configurations reached are still related under $R_{\mathcal{P}}$.

▶ **Example 10.** To aid the reader's intuition, we provide a partial unfolding of a CA-bisimulation for the system $2 : S_{\text{D}}$. Note that a proof of the CA-security of $2 : S_{\text{D}}$ is *not* the aim here. We represent by $\sigma_{v_1 v_2 v_3 v_4}$ the local state $[y \mapsto v_1][z \mapsto v_2][z_1 \mapsto v_3][z_2 \mapsto v_4]$, and by $\mathcal{P}_{\text{D}}$ the policy environmnet $[2 \mapsto \mathcal{P}_{\text{MD}}(2)]$.

We have $\langle 2 : S_{\text{D}}; \sigma_{2070} \rangle \overset{P_{\text{d}}^2}{\underset{\mathcal{P}_{\text{D}}}{=}} \langle 2 : S_{\text{D}}; \sigma_{2080} \rangle$. Hence $Sec_{\text{com}}(2 : S_{\text{D}}, \mathcal{P}_{\text{D}})$ calls for (among other things)

$$(\langle 2 : S_{\text{D}}; \sigma_{2070} \rangle, P_{\text{d}}^2)\ R_\star\ (\langle 2 : S_{\text{D}}; \sigma_{2080} \rangle, P_{\text{d}}^2),$$

where $R_\star$ is a CA-bisimulation.

Suppose

$$\langle 2 : S_{\text{D}}; \sigma_{2070} \rangle \xrightarrow{\tau}_2 \langle 2 : c?(y, z); \underline{if}; \underline{wh}; \sigma_{2070} \rangle. \tag{1}$$

There exist $P_1^\bullet$ and τ, such that $\tau \overset{P_1^\bullet}{\underset{!}{\sim}} \tau$. Pick particular $P_?^\bullet = P_1^\bullet$ and $\vec{v} = (0, 0)$, for which $\tau \overset{P_1^\bullet}{\underset{?}{\sim}} \tau(0, 0)$. Simulation of (1) is required with the action $\tau(0, 0) = \tau$. The only possibility is $\langle 2 : S_{\text{D}}; \sigma_{2080} \rangle \xrightarrow{\tau}_2 \langle 2 : c?(y, z); \underline{if}; \underline{wh}; \sigma_{2080} \rangle$. And the following is required

$$(\langle 2 : c?(y, z); \underline{if}; \underline{wh}; \sigma_{2070} \rangle, P_{\text{d}}^2)\ R_\star\ (\langle 2 : c?(y, z); \underline{if}; \underline{wh}; \sigma_{2080} \rangle, P_{\text{d}}^2).$$

Suppose

$$\langle 2 : c?(y,z); \underline{if}; \underline{wh}; \sigma_{2070}\rangle \xrightarrow{c?(1,k_1)}_2 \langle 2 : \underline{if}; \underline{wh}; \sigma_{1k_170}\rangle, \tag{2}$$

where k_1 is an integer. There should exist some $P^\bullet_!$ and β such that $c?(1,k_1) \overset{P^\bullet_!}{\underset{!}{\sim}} \beta$. Since $P^\circ(c) = L$, β must be $c?[\,]$. Pick $P^\bullet_? = P^\bullet_1$, and $\vec{v}$, $c?(1,k_1) \overset{P^\bullet_1}{\underset{?}{\sim}} c?[\,](\vec{v})$ implies $v_1 = 1$ since $P^\bullet_1(c.1) = L$. Hence a simulation of (2) with action $c?[\,](1,k_2)$ is required for all $k_2 \in \mathbb{Z}$ (note that $P^\bullet_1(c.2) = H$). It can only be of the form $\langle 2 : c?(y,z); \underline{if}; \underline{wh}; \sigma_{2080}\rangle \xrightarrow{c?(1,k_2)}_2 \langle 2 : \underline{if}; \underline{wh}; \sigma_{1k_280}\rangle$. And the following is required

$$(\langle 2 : \underline{if}; \underline{wh}; \sigma_{1k_170}\rangle, P^1_{\mathrm{d}})\ R_\star\ (\langle 2 : \underline{if}; \underline{wh}; \sigma_{1k_280}\rangle, P^1_{\mathrm{d}}).$$

This further requires $\langle 2 : \underline{if}; \underline{wh}; \sigma_{1k_170}\rangle \overset{P^1_{\mathrm{d}}}{\underset{\mathcal{P}_{\mathrm{D}}}{=}} \langle 2 : \underline{if}; \underline{wh}; \sigma_{1k_280}\rangle$, which holds. Going through two more "lock steps", the following is required.

$$(\langle 2 : z_1 := z; \underline{wh}; \sigma_{1k_170}\rangle, P^1_{\mathrm{d}})\ R_\star\ (\langle 2 : z_1 := z; \underline{wh}; \sigma_{1k_280}\rangle, P^1_{\mathrm{d}}) \tag{3}$$

$$(\langle 2 : S_{\mathrm{D}}; \sigma_{1k_1k_10}\rangle, P^1_{\mathrm{d}})\ R_\star\ (\langle 2 : S_{\mathrm{D}}; \sigma_{1k_2k_20}\rangle, P^1_{\mathrm{d}}) \tag{4}$$

And we still have $\langle 2 : S_{\mathrm{D}}; \sigma_{1k_1k_10}\rangle \overset{P^1_{\mathrm{d}}}{\underset{\mathcal{P}_{\mathrm{D}}}{=}} \langle 2 : S_{\mathrm{D}}; \sigma_{1k_2k_20}\rangle$ as required by the last relation... ◀

In Example 10, the systems are always the same on both sides of $R_\star$, which is a special case due partly to $P^\circ(c) = L$ in $\mathcal{P}^{\mathrm{ch}}_{\mathrm{MD}}$.

CA-security is *compositional*: it is preserved under $\setminus$ and $||$. In particular, its preservation under parallel composition is enabled by the rely-guarantee [16] pattern in the treatment of output and input, in CA-bisimulation.

▶ **Theorem 11** (Compositionality). *For Σ_1 with policy environment $\mathcal{P}_1$, and Σ_2 with policy environment $\mathcal{P}_2$, such that $Pid(\Sigma_1) \cap Pid(\Sigma_2) = \emptyset$,*

1. $Sec_{\mathrm{com}}(\Sigma_1, \mathcal{P}_1) \implies \forall \Omega \subseteq \mathbf{PCh} : Sec_{\mathrm{com}}(\Sigma_1 \setminus \Omega, \mathcal{P}_1)$*, and*
2. $Sec_{\mathrm{com}}(\Sigma_1, \mathcal{P}_1) \wedge Sec_{\mathrm{com}}(\Sigma_2, \mathcal{P}_2) \implies Sec_{\mathrm{com}}(\Sigma_1 || \Sigma_2, \mathcal{P}_1 \uplus \mathcal{P}_2)$*.*

The most important result of this section, that well-typedness guarantees CA-security, is formalized in Theorem 12. This soundness result means that our motivating example is noninterfering (Example 13).

▶ **Theorem 12** (Soundness). *For all systems Σ with policy environments $\mathcal{P}$, if $\mathcal{P} \vdash \{\mathbf{T}^\Sigma\}\,\Sigma\,\{\mathbf{T}^\Sigma\}$, then $Sec_{\mathrm{com}}(\Sigma, \mathcal{P})$.*

▶ **Example 13.** Going back to the multiplexer example, by Theorem 12, the well-typedness of the system Σ_{MD} of Example 1 guarantees $Sec_{\mathrm{com}}(\Sigma_{\mathrm{MD}}, \mathcal{P}_{\mathrm{MD}})$. ◀

7 Noninterference under Deterministic Schedulers

In this section, we formulate a noninterference property for systems executing under the control of deterministic schedulers. We then show that CA-security implies security under every such scheduler whose behavior is only influenced by its observation of the *public* memory (and its internal state).

Table 3 Transition rules for systems under scheduling

$$\frac{\langle \Sigma; \sigma \rangle \xrightarrow{\alpha}_{\eta} \langle \Sigma'; \sigma' \rangle}{\langle \Sigma; \sigma; q \rangle \xrightarrow{\alpha}_{\eta} \langle \Sigma'; \sigma'; \delta(q, \sigma) \rangle} \text{ if } o(q, \sigma) = \{\eta\} \qquad \frac{\neg(\exists \alpha : \langle \Sigma; \sigma \rangle \xrightarrow{\alpha}_{\eta})}{\langle \Sigma; \sigma; q \rangle \xrightarrow{\Box}_{\eta} \langle \Sigma; \sigma; \delta(q, \sigma) \rangle} \text{ if } o(q, \sigma) = \{\eta\}$$

We formalize deterministic schedulers Δ for systems Σ as Mealy automata $(Q, q_0, \mathbf{St}_{\mathbf{\Sigma}}, 2^{Pid}, \delta, o)$ where Q is its set of states, q_0 the initial state, $\mathbf{St}_{\mathbf{\Sigma}}$ (the memory states of Σ) is the alphabet, 2^{Pid} is the alphabet used for output, $\delta : Q \times \mathbf{St}_{\mathbf{\Sigma}} \to Q$ is the transition function and $o : Q \times \mathbf{St}_{\mathbf{\Sigma}} \to 2^{Pid}$ is the output function. For each state q and memory σ, $o(q, \sigma)$ is a set of one or two process identifiers, signaling the process(es) scheduled for the next step.

The semantics at the system level is adapted to execute the process(es) picked by the scheduler, the transition rules are given in Table 3. The extended configurations $\widehat{C} = \langle \Sigma; \sigma; q \rangle$ now contain the current state q of the scheduler.

It is desirable to constrain the observational power of the scheduler to the parts of the states that are low with respect to all policies. This is achieved with the concept of H-obliviousness (e.g., [29]).

▶ **Definition 14** (H-Oblivious Schedulers)**.** For $\Delta = (Q, q_0, \mathbf{St}_{\mathbf{\Sigma}}, 2^{Pid}, \delta, o)$, we have $Obl_{\mathcal{P}}(\Delta)$ if $q_0 \overset{\Delta}{\underset{\mathcal{P}}{\approx}} q_0$ holds, where $\overset{\Delta}{\underset{\mathcal{P}}{\approx}}$ is the largest relation s.t. $q_1 \overset{\Delta}{\underset{\mathcal{P}}{\approx}} q_2$ implies

$$\forall \sigma_1, \sigma_2 : \sigma_1 \overset{\{P | P \,\mathsf{E}\, \mathcal{P}\}}{=} \sigma_2 \quad \Rightarrow \quad o(q_1, \sigma_1) = o(q_2, \sigma_2) \wedge \delta(q_1, \sigma_1) \overset{\Delta}{\underset{\mathcal{P}}{\approx}} \delta(q_2, \sigma_2).$$

It is not difficult to come up with sensible H-oblivious schedulers (simple examples can be ones whose decisions do not depend on the memory at all) for the system Σ_{MD} considered in our motivating example.

In the following definition, we extend the notion of "low equivalence" ($\overset{P}{\underset{\mathcal{P}}{=}}$) introduced in the last section to a relation $\overset{P,\Delta}{\underset{\mathcal{P}}{=}}$ on our new type of configurations. In fact, an alternative interpretation of this new relation is $\overset{P,\Delta}{\underset{\mathcal{P}}{=}} = \overset{P}{\underset{\mathcal{P}}{=}} \times \overset{\Delta}{\underset{\mathcal{P}}{\approx}}$.

▶ **Definition 15** (Low Equivalence of Extended Configurations)**.**
$\langle \Sigma_1; \sigma_1; q_1 \rangle \overset{P,\Delta}{\underset{\mathcal{P}}{=}} \langle \Sigma_2; \sigma_2; q_2 \rangle$ if and only if $\langle \Sigma_1; \sigma_1 \rangle \overset{P}{\underset{\mathcal{P}}{=}} \langle \Sigma_2; \sigma_2 \rangle$ and $q_1 \overset{\Delta}{\underset{\mathcal{P}}{\approx}} q_2$.

We are in a position to state the security criterion for scheduled systems. In Definition 16, $- \overset{\Delta}{\underset{\mathcal{P}}{\sim}} -$ is the union of all bisimulations on scheduled systems characterized in Definition 17, where we omit all transition labels by abbreviating $\widehat{C} \xrightarrow{\alpha}_{\eta} \widehat{C}'$ as $\widehat{C} \longrightarrow \widehat{C}'$.

▶ **Definition 16** (Security of Scheduled Systems)**.** A system Σ is secure under the scheduling of Δ, denoted $Sec^{\Delta}(\Sigma, \mathcal{P})$, if and only if for all σ_1, σ_2, and P, if $\langle \Sigma; \sigma_1; q_0 \rangle \overset{P}{\underset{\mathcal{P}}{=}} \langle \Sigma; \sigma_2; q_0 \rangle$, then $(\langle \Sigma; \sigma_1; q_0 \rangle, P) \overset{\Delta}{\underset{\mathcal{P}}{\sim}} (\langle \Sigma; \sigma_2; q_0 \rangle, P)$.

▶ **Definition 17** (Bisimulation for Scheduled Systems)**.** A bisimulation $R^{\Delta}_{\mathcal{P}}$ for scheduled systems is a symmetric relation such that $(\widehat{C}_1, P)\ R^{\Delta}_{\mathcal{P}}\ (\widehat{C}_2, P)$ implies $\widehat{C}_1 \overset{P}{\underset{\mathcal{P},\Delta}{=}} \widehat{C}_2$ and the following:

$$\forall \widehat{C}'_1 \ s.t.\ \widehat{C}_1 \longrightarrow \widehat{C}'_1 : \exists \widehat{C}'_2 : \widehat{C}_2 \longrightarrow \widehat{C}'_2 \wedge \exists P' : (\widehat{C}'_1, P')\, R^{\Delta}_{\mathcal{P}}\, (\widehat{C}'_2, P').$$

Our final result (Theorem 19) is that the CA-security of a system guarantees its security under the control of any H-oblivious scheduler. However, the bisimulation for scheduled systems defined above dispensed with reliance upon certain channel policies for an input to be simulated; hence we need to impose the following condition of *input completeness* on the set $\mathcal{P}^{\mathrm{ch}}$ of channel policies for the link between the two noninterference conditions to be finally established.

▶ **Definition 18** (Input Completeness). $\mathcal{P}^{\mathrm{ch}}$ is input complete for Σ, denoted $IC(\mathcal{P}^{\mathrm{ch}}, \Sigma)$, if $\forall \sigma, \Sigma', \sigma', c, \vec{v} : \langle \Sigma; \sigma \rangle \rightarrow^{*} \langle \Sigma'; \sigma' \rangle \xrightarrow{c?\vec{v}} \Rightarrow\ \exists P^{\bullet} \in \mathcal{P}^{\mathrm{ch}} : c?\vec{v} \models_{?} P^{\bullet}$.

▶ **Theorem 19.** *For all systems Σ, policy environments $\mathcal{P}$ for variables, and schedulers Δ, if $Sec_{\mathrm{com}}(\Sigma, \mathcal{P})$, $IC(\mathcal{P}^{\mathrm{ch}}, \Sigma)$ and $Obl_{\mathcal{P}}(\Delta)$, then $Sec^{\Delta}(\Sigma, \mathcal{P})$.*

The input completeness condition essentially requires that each input (from the environment) that can be performed by a system Σ is satisfied by some content policy in $\mathcal{P}^{\mathrm{ch}}$. Note that this condition is met by all systems that can only communicate internally (with τ-actions), such as the Σ_{MD} of our motivating example. Hence continuing with Example 13, we can finally conclude $Sec^{\Delta}(\Sigma_{\mathrm{MD}}, \mathcal{P}_{\mathrm{MD}})$ for all schedulers Δ such that $Obl_{\mathcal{P}_{\mathrm{MD}}}(\Delta)$ holds.

8 Conclusion and Discussion

This paper studies information flow problems with the use of content-dependent confidentiality policies in a concurrent language. In our language, processes use local variables and communicate along channels with each other and the environment. A bisimulation-based noninterference condition is formulated to characterize security under the selection of different confidentiality policies according to the current memory and communication content. "Presence" and "content" are treated as separate aspects of communication, and a "rely-guarantee" pattern in picking the policies relevant to output and input leads to a compositionality result. A link is then established from this property to a more concisely formulated property concerning the use of a deterministic scheduler. The enforcement is achieved by a static type system that employs a Hoare logic component, which provides information on the possible memory content at different program points.

A major scenario related to our development is a concurrent system in which the destination of messages depends on their tagging. This is illustrated by our running example involving a communicating pair of multiplexer and demultiplexer, that separates confidential and public traffic between system partitions. This example is shown to be well-typed and secure under the control of schedulers whose behaviors are only influenced by their observation of the public memory.

The CA-bisimulation formulated in Section 6 does not universally quantify over states, but rather carries the states along with the executions. This has the flavor of the "flat bisimulation" considered in [9] that further goes back to [4]. "Flat bisimulations" do not give rise to a compositional notion of security when shared-memory is used, since memory modifications by other concurrent processes are not captured. However, the use of local variables with communication rectifies this issue and makes our CA-security compositional. Not surprisingly, this compositionality result allows us to focus on systems with single processes in our soundness proof.

As mentioned in the introduction, several developments [1, 5, 20] where information flow policies depend on certain conditions exist for *sequential languages*. On the other hand, type/-proof systems and noninterference properties have been studied extensively for concurrent systems (e.g., [4, 21, 9]), using information flow policies from *simple security lattices*.

For concurrent program security, the scheduling problem is an important area of research in its own (e.g., [29, 22, 26]). We have considered simple deterministic schedulers modeled neatly as Mealy automata. Understanding the security implications of *probabilistic schedulers* [26] with the use of disjunctive policies is an interesting line of future work. Another relevant topic of future work is dealing with *asynchronous communication*, which is important for widely distributed systems.

References

1 Torben Amtoft, Josiah Dodds, Zhi Zhang, Andrew W. Appel, Lennart Beringer, John Hatcliff, Xinming Ou, and Andrew Cousino. A certificate infrastructure for machine-checked proofs of conditional information flow. In *POST '12*.

2 Krzysztof R. Apt. Ten years of Hoare's logic: A survey - part 1. *ACM Trans. Program. Lang. Syst.*, 3(4):431–483, 1981.

3 Frédéric Besson, Nataliia Bielova, and Thomas Jensen. Hybrid information flow monitoring against web tracking. In *CSF '13*.

4 Gérard Boudol and Ilaria Castellani. Noninterference for concurrent programs and thread systems. *Theoretical Computer Science*, 281(1):109–130, 2002.

5 Niklas Broberg and David Sands. Paralocks: role-based information flow control and beyond. In *POPL '10*, pages 431–444.

6 Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. Civitas: Toward a secure voting system. In *S&P '08*, pages 354–368.

7 Ellis S. Cohen. Information transmission in computational systems. In *SOSP '77*.

8 The Coq Proof Assistant. Webpage: http://coq.inria.fr.

9 Mads Dam. Decidability and proof systems for language-based noninterference relations. POPL '06.

10 Dorothy E. Denning and Peter J. Denning. Certification of programs for secure information flow. *Commun. ACM*, 20(7):504–513, 1977.

11 Sebastian Eggert, Ron van der Meyden, Henning Schnoor, and Thomas Wilke. The complexity of intransitive noninterference. In *32nd IEEE Symposium on Security and Privacy, S&P 2011, 22-25 May 2011, Berkeley, California, USA*, pages 196–211, 2011.

12 Jeffrey S. Fenton. Memoryless subsystems. *The Computer Journal*, 17(2):143–147, 1974.

13 Riccardo Focardi and Sabina Rossi. Information flow security in dynamic contexts. *Journal of Computer Security*, 14(1):65–110, 2006.

14 Daniel Hedin, Arnar Birgisson, Luciano Bello, and Andrei Sabelfeld. Jsflow: tracking information flow in javascript and its apis. In *SAC '14*, pages 1663–1671.

15 Daniel Hedin and Andrei Sabelfeld. A perspective on information-flow control. In *Software Safety and Security - Tools for Analysis and Verification*, pages 319–347. 2012.

16 C. B. Jones. *Development Methods for Computer Programs including a Notion of Interference.* PhD thesis, Oxford University, June 1981.

17 Nenad Jovanovic, Christopher Kruegel, and Engin Kirda. Pixy: A static analysis tool for detecting web application vulnerabilities. pages 6–pp, 2006.

18 Naoki Kobayashi. Type-based information flow analysis for the pi-calculus. *Acta Inf.*, 42(4-5):291–347, 2005.

19 Ximeng Li. Proofs that are formalized in coq.
http://lbtweb.pbworks.com/w/file/fetch/97133580/dif_com_coq.zip.

20 Luísa Lourenço and Luís Caires. Dependent information flow types. POPL '15.

21 Heiko Mantel and Andrei Sabelfeld. A unifying approach to the security of distributed and multi-threaded programs. *Journal of Computer Security*, 11(4):615–676, 2003.

22 Heiko Mantel and Henning Sudbrock. Flexible scheduler-independent security. In *ESORICS '10*, pages 116–133.

23 Robin Milner. *Communication and concurrency*, volume 84. Prentice hall, 1989.

24 Benjamin C. Pierce. *Types and programming languages.* MIT Press, 2002.

25 John Rushby. Separation and integration in mils (the mils constitution). *Computer Science Laboratory SRI International, Technical Report SRI-CSL-08-XX*, 2008.

26 Andrei Sabelfeld. Confidentiality for multithreaded programs via bisimulation. In *PSI '03*.

27 Andrei Sabelfeld and Andrew C Myers. Language-based information-flow security. *Selected Areas in Communications, IEEE Journal on*, 21(1):5–19, 2003.

28 Geoffrey Smith. Improved typings for probabilistic noninterference in a multi-threaded language. *Journal of Computer Security*, 14(6):591–623, 2006.

29 Ron van der Meyden and Chenyi Zhang. Information flow in systems with schedulers, part I: definitions. *Theor. Comput. Sci.*, 467:68–88, 2013.

30 Dennis M. Volpano, Cynthia E. Irvine, and Geoffrey Smith. A sound type system for secure flow analysis. *Journal of Computer Security*, 4(2/3):167–188, 1996.

A Proof Sketch

Subject Reduction

The **subject reduction** result (Theorem 7) can be shown using the process-local result of Lemma 20, which can be proved by a straightforward induction on the derivation of the typing of S.

▶ **Lemma 20.** *If* $X, l_1 \vdash_{\mathcal{K}} \{\phi\}\, S\, \{\psi\} : Y, l_2$ *holds,* $\vdash_i \langle S; \sigma\rangle \xrightarrow{\alpha} \langle S'; \sigma'\rangle$, $\sigma \models \phi$, $P \in \mathcal{K}$ *and* $\sigma \models P$*, then*
1. $\exists P^\bullet : \alpha \models_! P^\bullet$*, and*
2. if $\exists P^\bullet$ *s.t.* $\alpha \models_? P^\bullet$ *then* $\exists P' \in \mathcal{K}, \phi' : X, l_1 \vdash_{\mathcal{K}} \{\phi'\}\, S'\, \{\psi\} : Y, l_2$ *and* $\sigma' \models \phi'$ *and* $\sigma' \models P'$.

Compositionality

▶ **Lemma 21.** *For all* Σ_1, Σ_2, σ_1, σ_2, i*, and* j*, such that* $i \in \mathit{Pid}(\Sigma_1)$ *and* $j \in \mathit{Pid}(\Sigma_2)$*, if there do not exist* c, ρ *and* $\vec{v}$ *such that* $\langle \Sigma_1; \sigma_1\rangle \xrightarrow{c\rho\vec{v}}_i$ *and* $\langle \Sigma_2; \sigma_2\rangle \xrightarrow{\widetilde{c\rho\vec{v}}}_j$*, then we have* $\langle \Sigma_1 || \Sigma_2; \sigma_1 \uplus \sigma_2\rangle \overset{\square}{\twoheadrightarrow}_{i,j} \langle \Sigma_1 || \Sigma_2; \sigma_1 \uplus \sigma_2\rangle$.

▶ **Lemma 22.** *For all* Σ, σ, Σ'_1, σ'_1, Σ'_2, σ'_2, c_1, ρ_1, $\vec{v}_1$, c_2, ρ_2, $\vec{v}_2$*, and* i*, if* $\langle \Sigma; \sigma\rangle \xrightarrow{c_1\rho_1\vec{v}_1}_i \langle \Sigma'_1; \sigma'_1\rangle$*, then*
1. $\langle \Sigma; \sigma\rangle \xrightarrow{c_2\rho_2\vec{v}_2}_i \langle \Sigma'_2; \sigma'_2\rangle \;\Rightarrow\; c_1 = c_2 \wedge \rho_1 = \rho_2$*, and*
2. $\langle \Sigma; \sigma\rangle \not\xrightarrow{\tau}_i$.

▶ **Lemma 23.** *The following statements hold.*

1. If $c\rho'\vec{v} \underset{\rho}{\overset{P^\bullet}{\sim}} \epsilon$ *then* $\widetilde{c\rho'}\vec{v} \underset{\widetilde{\rho}}{\overset{P^\bullet}{\sim}} \epsilon$*, and*
2. if $c\rho'\vec{v}_1 \underset{\rho}{\overset{P^\bullet}{\sim}} c\rho'\vec{v}_2$ *then* $\widetilde{c\rho'}\vec{v}_1 \underset{\widetilde{\rho}}{\overset{P^\bullet}{\sim}} \widetilde{c\rho'}\vec{v}_2$.

The proof of the **compositionality** result (Theorem 11) is sketched below.

Proof of Theorem 11.

Security of $\Sigma_1 \setminus \Omega$

We construct the following relation $R_\setminus$.

$$R_\setminus = \{((\langle \Sigma_{\mathrm{a}} \setminus \Omega;\, \sigma_{\mathrm{a}}\rangle, P_{\mathrm{a}}), (\langle \Sigma_{\mathrm{b}} \setminus \Omega;\, \sigma_{\mathrm{b}}\rangle, P_{\mathrm{b}})) \mid (\langle \Sigma_{\mathrm{a}}; \sigma_{\mathrm{a}}\rangle, P_{\mathrm{a}}) \underset{\mathcal{P}_1}{\overset{\mathrm{com}}{\sim}} (\langle \Sigma_{\mathrm{b}}; \sigma_{\mathrm{b}}\rangle, P_{\mathrm{b}})\}$$

By $\mathit{Sec}_{\mathrm{com}}(\Sigma_1, \mathcal{P}_1)$, we have

$$\forall \sigma_1, \sigma_2, P \text{ s.t. } \langle \Sigma_1; \sigma_1\rangle \underset{\mathcal{P}_1}{\overset{P}{=}} \langle \Sigma_1; \sigma_2\rangle : (\langle \Sigma_1; \sigma_1\rangle, P) \underset{\mathcal{P}_1}{\overset{\mathrm{com}}{\sim}} (\langle \Sigma_1; \sigma_2\rangle, P).$$

Take arbitrary σ'_1, σ'_2 and P' such that $\langle \Sigma_1 \setminus \Omega;\, \sigma'_1\rangle \underset{\mathcal{P}_1}{\overset{P'}{=}} \langle \Sigma_1 \setminus \Omega;\, \sigma'_2\rangle$. We also have $\langle \Sigma_1;\, \sigma'_1\rangle \underset{\mathcal{P}_1}{\overset{P'}{=}} \langle \Sigma_1;\, \sigma'_2\rangle$. Hence $(\langle \Sigma_1; \sigma'_1\rangle, P') \underset{\mathcal{P}_1}{\overset{\mathrm{com}}{\sim}} (\langle \Sigma_1; \sigma'_2\rangle, P')$ holds, and $(\langle \Sigma_1 \setminus \Omega; \sigma'_1\rangle, P')\; R_\setminus\; (\langle \Sigma_1 \setminus \Omega; \sigma'_2\rangle, P')$ holds.

It remains to be shown that $R_\setminus$ is a CA-bisimulation. The symmetry of $R_\setminus$ is obvious by its construction.

Take arbitrary pair $((\langle \Sigma_\mathrm{a} \setminus \Omega;\, \sigma_\mathrm{a}\rangle, P_\mathrm{a}), (\langle \Sigma_\mathrm{b} \setminus \Omega;\, \sigma_\mathrm{b}\rangle, P_\mathrm{b}))$ from $R_\setminus$, and arbitrary α, η, and C_1' such that $\langle \Sigma_\mathrm{a} \setminus \Omega; \sigma_\mathrm{a}\rangle \xrightarrow{\alpha}_\eta C_1'$. There must exist some C_1'' such that $\langle \Sigma_\mathrm{a}; \sigma_\mathrm{a}\rangle \xrightarrow{\alpha}_\eta C_1''$ and the channel used by α, if any, is not in C. The rest of the reasoning goes by a case split on α, on which we do not elaborate any further.

Security of $\Sigma_1||\Sigma_2$

We construct the following relation $R_{||}$.

$$\begin{aligned} R_{||} = \{\; & ((\langle \Sigma_{11}||\Sigma_{12}; \sigma_{11} \uplus \sigma_{12}\rangle, P_1 \uplus P_2), (\langle \Sigma_{21}||\Sigma_{22}; \sigma_{21} \uplus \sigma_{22}\rangle, P_1 \uplus P_2)) \mid \\ & ((\langle \Sigma_{11}; \sigma_{11}\rangle, P_1) \overset{\mathrm{com}}{\underset{\mathcal{P}_1}{\sim}} (\langle \Sigma_{21}; \sigma_{21}\rangle, P_1) \;\wedge\; (\langle \Sigma_{12}; \sigma_{12}\rangle, P_2) \overset{\mathrm{com}}{\underset{\mathcal{P}_2}{\sim}} (\langle \Sigma_{22}; \sigma_{22}\rangle, P_2) \;\} \end{aligned}$$

By $\mathit{Sec}_\mathrm{com}(\Sigma_1, \mathcal{P}_1)$ and $\mathit{Sec}_\mathrm{com}(\Sigma_2, \mathcal{P}_2)$, for all P, σ_a and σ_b such that $\langle \Sigma_1||\Sigma_2; \sigma_\mathrm{a}\rangle \overset{P}{\underset{\mathcal{P}}{=}} \langle \Sigma_1||\Sigma_2; \sigma_\mathrm{b}\rangle$, $(\langle \Sigma_1||\Sigma_2; \sigma_\mathrm{a}\rangle, P)\; R_{||}\; (\langle \Sigma_1||\Sigma_2; \sigma_\mathrm{b}\rangle, P)$ holds.

We then show that $R_{||}$ is a CA-bisimulation. By construction $R_{||}$ is symmetric.

Take a pair $((\langle \Sigma_{11}||\Sigma_{12}; \sigma_{11} \uplus \sigma_{12}\rangle, P_1 \uplus P_2), (\langle \Sigma_{21}||\Sigma_{22}; \sigma_{21} \uplus \sigma_{22}\rangle, P_1 \uplus P_2))$ from $R_{||}$ (where we have directly exposed the structures of the systems, states, and policies). We have

$$(\langle \Sigma_{11}; \sigma_{11}\rangle, P_1) \overset{\mathrm{com}}{\underset{\mathcal{P}_1}{\sim}} (\langle \Sigma_{21}; \sigma_{21}\rangle, P_1) \tag{5}$$

$$(\langle \Sigma_{12}; \sigma_{12}\rangle, P_2) \overset{\mathrm{com}}{\underset{\mathcal{P}_2}{\sim}} (\langle \Sigma_{22}; \sigma_{22}\rangle, P_2) \tag{6}$$

Suppose $\langle \Sigma_{11}||\Sigma_{12}; \sigma_{11} \uplus \sigma_{12}\rangle \xrightarrow{\alpha}_\eta \langle \Sigma_{11}'||\Sigma_{12}'; \sigma_{11}' \uplus \sigma_{12}'\rangle$ for some α, Σ_{11}', Σ_{12}', σ_{11}' and σ_{12}'. A case split on this transition is to be made, where the most interesting case is the communication between Σ_{11} and Σ_{12}. We give detailed arguments for this case.

Suppose without loss of generality that $\langle \Sigma_{11}; \sigma_{11}\rangle \xrightarrow{c!\vec{v}}_i \langle \Sigma_{11}'; \sigma_{11}'\rangle$, $\langle \Sigma_{12}; \sigma_{12}\rangle \xrightarrow{c?\vec{v}}_j \langle \Sigma_{12}'; \sigma_{12}'\rangle$, and $\eta = i, j$. We then discuss whether $P^\circ(c)$ is H or L.

1. $P^\circ(c) = H$. We make a further case split on whether Σ_{21} can communicate with Σ_{22}.
   a. Σ_{21} cannot communicate with Σ_{22}. Using Lemma 21, we have

$$\langle \Sigma_{21}||\Sigma_{22}; \sigma_{21} \uplus \sigma_{22}\rangle \overset{\square}{\twoheadrightarrow}_{i,j} \langle \Sigma_{21}||\Sigma_{22}; \sigma_{21} \uplus \sigma_{22}\rangle \tag{7}$$

Using (5), there exists some content policy $P_0^\bullet$ and β such that $c!\vec{v} \overset{P_0^\bullet}{\underset{!}{\sim}} \beta$. By $P^\circ(c) = H$, $\beta = \epsilon$. For all content policy $P_?^\bullet$ and $\vec{v}$ such that $c!\vec{v} \overset{P_?^\bullet}{\underset{?}{\sim}} \epsilon(\vec{v})$ (which boilds down to $\forall P_?^\bullet \in \mathcal{P}_\bullet^\mathrm{ch}$), there exists some variable policy P_1' such that the transition from $\langle \Sigma_{11}; \sigma_{11}\rangle$ is simulated by an ϵ-transition from $\langle \Sigma_{21}; \sigma_{21}\rangle$, with

$$(\langle \Sigma_{11}'; \sigma_{11}'\rangle, P_1') \overset{\mathrm{com}}{\underset{\mathcal{P}_1}{\sim}} (\langle \Sigma_{21}; \sigma_{21}\rangle, P_1'). \tag{8}$$

Using (6), and the fact that $c?\vec{v} \overset{P_0^\bullet}{\underset{?}{\sim}} \epsilon$ holds with the $P_0^\bullet$ guaranteed by the output $c!\vec{v}$ (using Lemma 23), there exists some variable policy P_2' such that

$$(\langle \Sigma_{12}'; \sigma_{12}'\rangle, P_2') \overset{\mathrm{com}}{\underset{\mathcal{P}_2}{\sim}} (\langle \Sigma_{22}; \sigma_{22}\rangle, P_2'). \tag{9}$$

By the construction of $R_{||}$, we thus have

$$(\langle \Sigma'_{11} || \Sigma'_{12}; \sigma'_{11} \uplus \sigma'_{12} \rangle, P'_1 \uplus P'_2) \; R_{||} \; (\langle \Sigma_{21} || \Sigma_{22}; \sigma_{21} \uplus \sigma_{22} \rangle, P'_1 \uplus P'_2).$$

Since $\tau \overset{P_0^\bullet}{\underset{!}{\sim}} \square$ holds, the internal communication τ between Σ_{11} and Σ_{12} is thus simulated by transition (7).

b. Suppose we have c', $\vec{v}'$, Σ'_{21}, Σ'_{22}, σ'_{21} and σ'_{22} such that

$$\langle \Sigma_{21}; \sigma_{21} \rangle \xrightarrow{c'!\vec{v}'}_i \langle \Sigma'_{21}; \sigma'_{21} \rangle \tag{10}$$

$$\langle \Sigma_{22}; \sigma_{22} \rangle \xrightarrow{c'?\vec{v}'}_j \langle \Sigma'_{22}; \sigma'_{22} \rangle \tag{11}$$

We show that $P^\circ(c') = H$ must be the case. Assume per absurdum that $P^\circ(c') = L$. By (5), the symmetry of $\overset{\text{com}}{\underset{\mathcal{P}_1}{\sim}}$, and transition (10), there exists $P_1^\bullet$ and β such that $c'!\vec{v}' \overset{P_1^\bullet}{\underset{!}{\sim}} \beta$, and for all $P_?^\bullet$ such that $c'!\vec{v}' \overset{P_?^\bullet}{\underset{?}{\sim}} \beta$ (which boils down to $\forall P_?^\bullet \in \mathcal{P}_\bullet^{\text{ch}}$), we have $\langle \Sigma_{11}; \sigma_{11} \rangle \overset{\beta}{\twoheadrightarrow}_i \langle \Sigma'_{11}; \sigma'_{11} \rangle$. By $P^\circ(c') = L$, $\beta = c'!\vec{v}''$ for some $\vec{v}''$. Hence we have $\langle \Sigma_{11}; \sigma_{11} \rangle \xrightarrow{c'!\vec{v}''}_i \langle \Sigma'_{11}; \sigma'_{11} \rangle$ by the non-emptiness of $\mathcal{P}_\bullet^{\text{ch}}$. By Lemma 22 we have $c' = c$ and a contradiction.

By reasoning similar to that of case (1a), it can be shown that $c!\vec{v}$ and $c?\vec{v}$ performed from $\langle \Sigma_{11}; \sigma_{11} \rangle$ and $\langle \Sigma_{12}; \sigma_{12} \rangle$ can both be simulated by ϵ, resulting again in (8) and (9). By the symmetry of $\overset{\text{com}}{\underset{\mathcal{P}_1}{\sim}}$ and $\overset{\text{com}}{\underset{\mathcal{P}_2}{\sim}}$, and $P^\circ(c') = H$, the actions $c'!\vec{v}'$ and $c'?\vec{v}'$ can be simulated by ϵ-transitions from $\langle \Sigma'_{11}; \sigma'_{11} \rangle$ and $\langle \Sigma'_{12}; \sigma'_{12} \rangle$, resulting in

$$(\langle \Sigma'_{11}; \sigma'_{11} \rangle, P''_1) \overset{\text{com}}{\underset{\mathcal{P}_1}{\sim}} (\langle \Sigma'_{21}; \sigma'_{21} \rangle, P''_1) \tag{12}$$

$$(\langle \Sigma'_{12}; \sigma'_{12} \rangle, P''_2) \overset{\text{com}}{\underset{\mathcal{P}_2}{\sim}} (\langle \Sigma'_{22}; \sigma'_{22} \rangle, P''_2) \tag{13}$$

for some P''_1 and P''_2. By the construction of $R_{||}$, we have

$$(\langle \Sigma'_{11} || \Sigma'_{12}; \sigma'_{11} \uplus \sigma'_{12} \rangle, P''_1 \uplus P''_2) \; R_{||} \; (\langle \Sigma'_{21} || \Sigma'_{22}; \sigma'_{21} \uplus \sigma'_{22} \rangle, P''_1 \uplus P''_2).$$

There must exist $P_!^\bullet$ and τ such that $\tau \overset{P_!^\bullet}{\underset{!}{\sim}} \tau$ (any $P^\bullet$ in $\mathcal{P}_\bullet^{\text{ch}}$ will do), and for all $P_?^\bullet$ and $\vec{v}$ such that $\tau \overset{P_?^\bullet}{\underset{?}{\sim}} \tau(\vec{v})$, there exists the transition $\langle \Sigma_{21} || \Sigma_{22}; \sigma_{21} \uplus \sigma_{22} \rangle \xrightarrow{\tau}_{i,j} \langle \Sigma'_{21} || \Sigma'_{22}; \sigma'_{21} \uplus \sigma'_{22} \rangle$ simulating the internal communication between Σ_{11} and Σ_{21}.

The case where $\langle \Sigma_{21}; \sigma_{21} \rangle$ performs an input $c'?\vec{v}'$ and $\langle \Sigma_{22}; \sigma_{22} \rangle$ performs the corresponding output $c'!\vec{v}'$ is analogous.

2. $P^\circ(c) = L$. In this case the output and input by Σ_{11} and Σ_{12}, respectively, can be simulated by an output and an input over the same channel, from Σ_{21} and Σ_{22}. The communication between Σ_{11} and Σ_{12} can be simulated by a communication between Σ_{21} and Σ_{22} over the same channel c. The statement 2 of Lemma 23 will be useful.

The other cases are less involved. ◀

Soundness

The proof of the **soundness** result (Theorem 12) is sketched below. The establishment of the compositionality result enables us to focus on systems of single processes.

Table 4 The low equivalence relation

$$(\mathrm{LO}_{\mathrm{EQ}})\quad \frac{\exists\phi : lo^{\mathcal{K}}_{\phi}(\langle S;\sigma_1\rangle, P) \wedge lo^{\mathcal{K}}_{\phi}(\langle S;\sigma_2\rangle, P) \quad \sigma_1 \overset{\{P\}}{=} \sigma_2 \quad nip(\mathcal{K})}{(\langle i : S;\, \sigma_1\rangle, P) \overset{[i\mapsto\mathcal{K}]}{\simeq} (\langle i : S;\, \sigma_2\rangle, P)}$$

$$(\mathrm{HI}_{\mathrm{EQ}})\quad \frac{hi^{\mathcal{K}}_{\phi_1}(\langle S_1;\sigma_1\rangle, P) \wedge hi^{\mathcal{K}}_{\phi_2}(\langle S_2;\sigma_2\rangle, P) \quad \sigma_1 \overset{\{P\}}{=} \sigma_2 \quad nip(\mathcal{K})}{(\langle i : S_1;\, \sigma_1\rangle, P) \overset{[i\mapsto\mathcal{K}]}{\simeq} (\langle i : S_2;\, \sigma_2\rangle, P)}$$

▶ **Definition 24.** A process S with set $\mathcal{K}$ of variable policies is *high at state* σ *and with policy* P*, under pre-condition* ϕ, written $hi^{\mathcal{K}}_{\phi}(\langle S;\sigma\rangle, P)$, if and only if

$$\begin{array}{l}\sigma \models P \;\wedge\; \sigma \models \phi \;\wedge \\ \exists X, l, X', l', \psi : \sigma \models \phi \;\wedge\; X, l \vdash_{\mathcal{K}} \{\phi\}\, S\, \{\psi\} : X', l' \;\wedge\; (l = H \vee \exists x \in X : P_{\mathsf{S}}(x) = H)\end{array}$$

▶ **Definition 25.** A process S with set $\mathcal{K}$ of variable policies is *low at state* σ *and with policy* P*, under pre-condition* ϕ, written $lo^{\mathcal{K}}_{\phi}(\langle S;\sigma\rangle, P)$, if and only if

$$\begin{array}{l}\sigma \models P \;\wedge\; \sigma \models \phi \;\wedge \\ (\exists X, l, X', l', \psi : X, l \vdash_{\mathcal{K}} \{\phi\}\, S\, \{\psi\} : X', l') \;\wedge \\ (\forall X, l, X', l', \psi : X, l \vdash_{\mathcal{K}} \{\phi\}\, S\, \{\psi\} : X', l' \Rightarrow (l = L \wedge \forall x \in X : P_{\mathsf{S}}(x) = L))\end{array}$$

We then define a low-equivalence relation $\overset{[i\mapsto\mathcal{K}]}{\simeq}$ concerned with systems of the form $i : S$ in Table 4.

Several lemmas are then established to build up to the following results:

- $\overset{[i\mapsto\mathcal{K}]}{\simeq}$ qualifies as a CA-bisimulation, and
- if $[i \mapsto \mathcal{K}] \vdash \{[i \mapsto \mathbf{tt}]\}\, i : S\, \{[i \mapsto \mathbf{tt}]\}$ holds, and $\langle i : S;\sigma_1\rangle \underset{[i\mapsto\mathcal{K}]}{\overset{P}{=}} \langle i : S;\sigma_2\rangle$, then $(\langle i : S;\, \sigma_1\rangle, P) \overset{[i\mapsto\mathcal{K}]}{\simeq} (\langle i : S;\, \sigma_2\rangle, P)$ holds.

In other words, if $[i \mapsto \mathcal{K}] \vdash \{[i \mapsto \mathbf{tt}]\}\, i : S\, \{[i \mapsto \mathbf{tt}]\}$ can be established, then $Sec_{\mathrm{com}}(i : S, [i \mapsto \mathcal{K}])$. Suppose a sysmtem $\Sigma_\star$ has the forms $\Sigma_1||\Sigma_2$, or $\Sigma\backslash\Omega$, and the policy environment $\mathcal{P}_\star$. By the typing rule for systems, and the compositionality theorem (Theorem 11), we can then easily show that $\mathcal{P}_\star \vdash \{\mathbf{T}^{\Sigma_\star}\}\, \Sigma_\star\, \{\mathbf{T}^{\Sigma_\star}\}$ implies $Sec_{\mathrm{com}}(\Sigma_\star, \mathcal{P}_\star)$.

▶ **Lemma 26.** *For all* S*,* σ_1*,* σ_2*,* i*,* $\mathcal{K}$ *and* $P \in \mathcal{K}$*, if* $[i \mapsto \mathcal{K}] \vdash \{[i \mapsto \phi]\}\, [i \mapsto S]\, \{\Psi\}$ *for some* ϕ *and* Ψ*,* $\sigma_1 \models P$*,* $\sigma_2 \models P$*,* $\sigma_1 \models \phi$*, and* $\sigma_2 \models \phi$*, then either of the following holds.*

- $lo^{\mathcal{K}}_{\phi}(\langle S;\sigma_1\rangle, P) \wedge lo^{\mathcal{K}}_{\phi}(\langle S;\sigma_2\rangle, P)$
- $hi^{\mathcal{K}}_{\phi}(\langle S;\sigma_1\rangle, P) \wedge hi^{\mathcal{K}}_{\phi}(\langle S;\sigma_2\rangle, P)$

▶ **Definition 27.** $UpdNext(S)$ represents the set of variables that the process S attempts to update immediately. Formally, $UpdNext(S) = \emptyset$ except for the following cases.

$$\begin{array}{l}UpdNext(x := a) = \{x\} \\ UpdNext(c?\vec{x}) = \{x_1, ..., x_k\} \text{ where } \vec{x} = (x_1, ..., x_k) \\ UpdNext(S_1; S_2) = UpdNext(S_1)\end{array}$$

▶ **Lemma 28.** *If* $x \notin UpdNext(S)$*, and* $\vdash_i \langle S;\sigma\rangle \xrightarrow{\alpha} \langle S';\sigma'\rangle$*, then* $\sigma(x) = \sigma'(x)$*.*

▶ **Lemma 29.** *If* $hi^{\mathcal{K}}_{\phi}(\langle S;\sigma\rangle, P)$*, and* $\vdash_i \langle S;\sigma\rangle \xrightarrow{\alpha} \langle S';\sigma'\rangle$*, then*

- $\forall c, \rho, \vec{v} : \alpha = c\rho\vec{v} \Rightarrow P^{\circ}(c) = H$*,*
- $\exists\alpha \models_{!} P^{\bullet}$*, and*
- *if* $\exists\alpha \models_{?} P^{\bullet}$*, then*
  $\exists P' : (\forall x \in \mathbf{Var}_{\Sigma,i} : (x \in \mathit{UpdNext}(S) \Rightarrow P'_{\mathsf{S}}(x) = H) \ \land (x \notin \mathit{UpdNext}(S) \Rightarrow P_{\mathsf{S}}(x) \sqsubseteq P'_{\mathsf{S}}(x))) \land$
  $\exists\psi : hi^{\mathcal{K}}_{\psi}(\langle S';\sigma'\rangle, P')$*.*

▶ **Lemma 30.** *Given* $S = S_1; S_2$*,* σ*,* P*, and* ϕ *such that* S *is typable with precondition* ϕ*,* $\sigma \models \phi$*, and* $\sigma \models P_{\mathsf{V}}$*, if one of the following holds*

- $S_1 = c!\vec{a}$*, where* $P^{\circ}(c) = H$*,*
- $S_1 = c?\vec{x}$*, where* $P^{\circ}(c) = H$*,*
- $S_1 = \textsf{if } b \textsf{ then } S' \textsf{ else } S''$*, where* $\exists x \in \mathit{fv}(b) : P_{\mathsf{S}}(x) = H$*, or*
- $S_1 = \textsf{while } b \textsf{ do } S'$*, where* $\exists x \in \mathit{fv}(b) : P_{\mathsf{S}}(x) = H$*,*

then $hi^{\mathcal{K}}_{\phi}(\langle S;\sigma\rangle, P)$ *holds.*

▶ **Lemma 31.** *For all* S*, if* S *contains a communication binder over channel* c*, then for all* $\mathcal{K}$*,* X*,* l*,* X'*,* l'*,* ϕ*, and* ψ*, such that* $X, l \vdash_{\mathcal{K}} \{\phi\}\, S\, \{\psi\} : X', l'$*, it holds that* $P^{\circ}(c) \sqsubseteq l'$*.*

Scheduler-Specific Security

We turn to the proof that CA-security implies security under deterministic scheduling.

▶ **Lemma 32.** *For all* Σ_1*,* σ_1*,* Σ'_1*,* σ'_1*,* Σ_2*,* σ_2*,* α'*,* Σ'_2*,* σ'_2*,* η *and* P*, if we have* $(\langle\Sigma_1;\sigma_1\rangle, P) \overset{\text{com}}{\underset{\mathcal{P}}{\sim}} (\langle\Sigma_2;\sigma_2\rangle, P)$*,* $\langle\Sigma_1;\sigma_1\rangle \overset{\square}{\twoheadrightarrow}_{\eta} \langle\Sigma'_1;\sigma'_1\rangle$*, and* $\langle\Sigma_2;\sigma_2\rangle \xrightarrow{\alpha'}_{\eta} \langle\Sigma'_2;\sigma'_2\rangle$*, then* $\alpha' = \tau$ *or* $P^{\circ}(ch(\alpha')) = H$*.*

Proof. This lemma can be shown using Lemma 22 and the symmetry of $- \overset{\text{com}}{\underset{\mathcal{P}}{\sim}} -$. Note that all possible cases for α' are $\alpha' = \tau$, $P^{\circ}(ch(\alpha')) = H$ and $P^{\circ}(ch(\alpha')) = L$. Hence we just need to show that $P^{\circ}(ch(\alpha')) = L$ is impossible. Assume per absurdum that $P^{\circ}(ch(\alpha')) = L$. By symmetry of $- \overset{\text{com}}{\underset{\mathcal{P}}{\sim}} -$ we have

$$(\langle\Sigma_2;\sigma_2\rangle, P) \overset{\text{com}}{\underset{\mathcal{P}}{\sim}} (\langle\Sigma_1;\sigma_1\rangle, P).$$

By $\langle\Sigma_2;\sigma_2\rangle \xrightarrow{\alpha'}_{\eta} \langle\Sigma'_2;\sigma'_2\rangle$, there exist $P^{\bullet}_{!}$, β' such that $\alpha' \overset{P^{\bullet}_{!}}{\underset{!}{\sim}} \beta'$, and for all $P^{\bullet}_{?}$, $\vec{v}'$ such that $\alpha' \overset{P^{\bullet}_{?}}{\underset{?}{\sim}} \beta'(\vec{v}')$, there exist Σ'_1, σ'_1, and P' such that

$$\langle\Sigma_1;\sigma_1\rangle \overset{\beta'(\vec{v}')}{\twoheadrightarrow}_{\eta} \langle\Sigma'_1;\sigma'_1\rangle \tag{14}$$

$$(\langle\Sigma'_2;\sigma'_2\rangle, P') \overset{\text{com}}{\underset{\mathcal{P}}{\sim}} (\langle\Sigma'_1;\sigma'_1\rangle, P') \tag{15}$$

Since $P^{\circ}(ch(\alpha')) = L$, by $\alpha' \overset{P^{\bullet}_{!}}{\underset{!}{\sim}} \beta'$ we have the following cases:

1. $\alpha' = c!\vec{v}$ and $\beta' = c!\vec{v}''$ for some c, $\vec{v}$ and $\vec{v}''$ such that $v_j = v''_j$ whenever $P^{\bullet}(c.j) = L$,
2. $\alpha' = c?\vec{v}$ and $\beta' = c?[\,]$ for some c and $\vec{v}$.

In both cases, the universally quantified $P_?^\bullet$ and $\vec{v}'$ can be instantiated with $P_!^\bullet$ and $\vec{v}$, for $\alpha' \overset{P_?^\bullet}{\underset{?}{\sim}} \beta'(\vec{v}')$ to be satisfied. We thus obtain (14) un-guarded. In more detail, we have

$$\langle \Sigma_1; \sigma_1 \rangle \xrightarrow{\beta'(\vec{v})}_\eta \langle \Sigma_1'; \sigma_1' \rangle \tag{16}$$

However, we also have $\langle \Sigma_1; \sigma_1 \rangle \overset{\square}{\twoheadrightarrow}_\eta \langle \Sigma_1'; \sigma_1' \rangle$ from the pre-conditions. This is impossible due to Lemma 22, and this contradiction completes the proof. ◀

▶ **Lemma 33.** *If* $IC(\mathcal{P}^{\text{ch}}, \Sigma)$, $\exists \sigma : \langle \Sigma; \sigma \rangle \longrightarrow^* \langle \Sigma_1; \sigma_1 \rangle \xrightarrow{\alpha}$, $P^\circ(ch(\alpha)) = H$, *and* $\alpha \overset{P_!^\bullet}{\underset{!}{\sim}} \beta$, *then* $\exists P_?^\bullet : \forall \vec{v} : \alpha \overset{P_?^\bullet}{\underset{?}{\sim}} \beta(\vec{v})$.

Proof. The proof is trivial by making a case split on the polarity of α. ◀

Finally, the proof of Theorem 19 is given below.

Proof. We unfold $Sec_{\text{com}}(\Sigma, \mathcal{P})$ into

$$\forall \sigma_1, \sigma_2, P \ s.t. \ \langle \Sigma; \sigma_1 \rangle \overset{P}{\underset{\mathcal{P}}{=}} \langle \Sigma; \sigma_2 \rangle : (\langle \Sigma; \sigma_1 \rangle, P) \overset{\text{com}}{\underset{\mathcal{P}}{\sim}} (\langle \Sigma; \sigma_2 \rangle, P) \tag{17}$$

Given Δ such that $Obl_{\mathcal{P}}(\Delta)$ holds, we construct the following relation R_Δ,

$$\begin{aligned} R_\Delta = \quad \{ ((\langle \Sigma_1; \sigma_1; q_1 \rangle, P_1), (\langle \Sigma_2; \sigma_2; q_2 \rangle, P_2)) \mid & \\ & q_1 \overset{\Delta}{\underset{\mathcal{P}}{\approx}} q_2 \ \wedge \\ & (\langle \Sigma_1; \sigma_1 \rangle, P_1) \overset{\text{com}}{\underset{\mathcal{P}}{\sim}} (\langle \Sigma_2; \sigma_2 \rangle, P_2) \ \wedge \\ & \exists \sigma : \langle \Sigma; \sigma \rangle \longrightarrow^* \langle \Sigma_1; \sigma_1 \rangle \wedge \exists \sigma : \langle \Sigma; \sigma \rangle \longrightarrow^* \langle \Sigma_2; \sigma_2 \rangle \}. \end{aligned}$$

Using (17), it is not difficult to show that for all σ_1, σ_2, and P such that $\langle \Sigma; \sigma_1; q_0 \rangle \overset{P}{\underset{\mathcal{P}}{=}} \langle \Sigma; \sigma_2; q_0 \rangle$, the pair $((\langle \Sigma; \sigma_1; q_0 \rangle, P), (\langle \Sigma; \sigma_2; q_0 \rangle, P))$ is in R_Δ.

We next show that R_Δ is a bisimulation for scheduled systems. The symmetry of R_Δ is obvious by construction.

Take pair $((\langle \Sigma_1; \sigma_1; q_1 \rangle, P), (\langle \Sigma_2; \sigma_2; q_2 \rangle, P))$ from R_Δ. We have

$$q_1 \overset{\Delta}{\underset{\mathcal{P}}{\approx}} q_2 \tag{18}$$

$$(\langle \Sigma_1; \sigma_1 \rangle, P) \overset{\text{com}}{\underset{\mathcal{P}}{\sim}} (\langle \Sigma_2; \sigma_2 \rangle, P) \tag{19}$$

$$\exists \sigma : \langle \Sigma; \sigma \rangle \longrightarrow^* \langle \Sigma_1; \sigma_1 \rangle \tag{20}$$

$$\exists \sigma : \langle \Sigma; \sigma \rangle \longrightarrow^* \langle \Sigma_2; \sigma_2 \rangle \tag{21}$$

We thus have $\langle \Sigma_1; \sigma_1 \rangle \overset{P}{\underset{\mathcal{P}}{=}} \langle \Sigma_2; \sigma_2 \rangle$. Hence $\langle \Sigma_1; \sigma_1; q_1 \rangle \overset{P,\Delta}{\underset{\mathcal{P}}{=}} \langle \Sigma_2; \sigma_2; q_2 \rangle$ holds.

Take arbitrary Σ_1', σ_1' and q_1' such that

$$\langle \Sigma_1; \sigma_1; q_1 \rangle \longrightarrow \langle \Sigma_1'; \sigma_1'; q_1' \rangle. \tag{22}$$

By (19), we have $\sigma_1 \overset{\{P\}}{=} \sigma_2$, and $P \mathrel{\mathsf{E}} \mathcal{P}$, thus $\sigma_1 \overset{\{P' \mid P' \mathrel{\mathsf{E}} \mathcal{P}\}}{=} \sigma_2$. By $Obl_{\mathcal{P}}(\Delta)$, we have

$$o(q_2, \sigma_2) = o(q_1, \sigma_1), \tag{23}$$

and

$$q_1' \overset{\Delta}{\underset{\mathcal{P}}{\approx}} q_2', \tag{24}$$

where $q_2' = \delta(q_2, \sigma_2)$ (and we know that $q_1' = \delta(q_1, \sigma_1)$).

With a case split on whether transition (22) is a $\xrightarrow{\square}$, we show that there is always a one-step simulation.

1. Transition (22) is $\langle \Sigma_1; \sigma_1; q_1 \rangle \xrightarrow{\alpha}_\eta \langle \Sigma_1'; \sigma_1'; q_1' \rangle$ where $\alpha \neq \square$. We have

   $$\langle \Sigma_1; \sigma_1 \rangle \xrightarrow{\alpha}_\eta \langle \Sigma_1'; \sigma_1' \rangle \tag{25}$$

   By (19), there exist some $P_!^\bullet$ and β such that $\alpha \overset{P_!^\bullet}{\underset{!}{\sim}} \beta$, and for all $P_?^\bullet$, $\vec{v}$ such that $\alpha \overset{P_?^\bullet}{\underset{?}{\sim}} \beta(\vec{v})$, there exist Σ_2', σ_2', and P' such that

   $$\langle \Sigma_2; \sigma_2 \rangle \overset{\beta(\vec{v})}{\twoheadrightarrow}_\eta \langle \Sigma_2'; \sigma_2' \rangle \tag{26}$$
   $$(\langle \Sigma_1'; \sigma_1' \rangle, P') \overset{\text{com}}{\underset{\mathcal{P}}{\sim}} (\langle \Sigma_2'; \sigma_2' \rangle, P') \tag{27}$$

   a. Suppose α is not an input. Then all $P_?^\bullet$ in $\mathcal{P}_\bullet^{\text{ch}}$, and all $\vec{v}$ will satisfy $\alpha \overset{P_?^\bullet}{\underset{?}{\sim}} \beta(\vec{v})$.

   b. Suppose $\alpha = c?\vec{v}_1$ and $P^\circ(c) = H$. By $\alpha \overset{P_!^\bullet}{\underset{!}{\sim}} \beta$, β must be ϵ. By $IC(\mathcal{P}^{\text{ch}}, \Sigma)$ and (20), there exists some $P_2^\bullet$ in $\mathcal{P}^{\text{ch}}$ such that for all $\vec{v}_2$, $\alpha \overset{P_2^\bullet}{\underset{?}{\sim}} \epsilon(\vec{v}_2)$.

   c. Suppose $\alpha = c?\vec{v}_1$ and $P^\circ(c) = L$. By $\alpha \overset{P_!^\bullet}{\underset{!}{\sim}} \beta$, $\beta = c?[\,]$. By $IC(\mathcal{P}^{\text{ch}}, \Sigma)$ and (20), there exists some $P_2^\bullet$, and $\vec{v}_2 = \vec{v}_1$, such that $\alpha \overset{P_2^\bullet}{\underset{?}{\sim}} c?[\,](\vec{v}_2)$.

   Hence in all cases we can instantiate the universally quantified $P_?^\bullet$ and $\vec{v}$, and obtain the transition (26) such that (27) holds. We next make a case split on whether $\beta = \epsilon$.

   a. $\beta \neq \epsilon$.

      i. $\beta \neq \square$. We have $\langle \Sigma_2; \sigma_2; q_2 \rangle \longrightarrow_\eta \langle \Sigma_2'; \sigma_2'; q_2' \rangle$. Combining $q_1' \overset{\Delta}{\underset{\mathcal{P}}{\approx}} q_2'$ and (27), we have

      $$(\langle \Sigma_1'; \sigma_1'; q_1' \rangle, P') \; R_\Delta \; (\langle \Sigma_2'; \sigma_2'; q_2' \rangle, P').$$

      ii. $\beta = \square$. We have $\langle \Sigma_2; \sigma_2 \rangle \overset{\square}{\twoheadrightarrow}_\eta \langle \Sigma_2'; \sigma_2' \rangle$. We know that $(\forall \alpha' : \langle \Sigma_2; \sigma_2 \rangle \xrightarrow{\alpha'}_\eta \Rightarrow P^\circ(ch(\alpha')) = H) \wedge \Sigma_2' = \Sigma_2 \wedge \sigma_2' = \sigma_2$. There are two possibilities concerning the transition that can take place from $\langle \Sigma_2; \sigma_2 \rangle$.

      Suppose no transition can be performed from $\langle \Sigma_2; \sigma_2 \rangle$. In this case we have $\langle \Sigma_2; \sigma_2; q_2 \rangle \xrightarrow{\square}_\eta \langle \Sigma_2; \sigma_2'; q_2' \rangle$. This transition qualifies as simulation of the transition $\langle \Sigma_1; \sigma_1; q_1 \rangle \xrightarrow{\alpha}_\eta \langle \Sigma_1'; \sigma_1'; q_1' \rangle$ in R_Δ, by reasoning similar to the case 1(a)i.

      Suppose a transition $\langle \Sigma_2; \sigma_2 \rangle \xrightarrow{\alpha'}_\eta \langle \Sigma_2''; \sigma_2'' \rangle$ exists for some Σ_2'', σ_2'' and α', where α' is a communication such that $P^\circ(ch(\alpha')) = H$. By the symmetry of $- \overset{\text{com}}{\underset{\mathcal{P}}{\sim}} -$, (27), $\Sigma_2' = \Sigma_2$ and $\sigma_2' = \sigma_2$, we have

      $$(\langle \Sigma_2; \sigma_2 \rangle, P') \overset{\text{com}}{\underset{\mathcal{P}}{\sim}} (\langle \Sigma_1'; \sigma_1' \rangle, P'). \tag{28}$$

      Hence there exist β', $P_{!2}^\bullet$ such that $\alpha' \overset{P_{!2}^\bullet}{\underset{!}{\sim}} \beta'$, and for all $\vec{v}'$ and $P_{?2}^\bullet$ such that

$\alpha' \underset{?}{\overset{P^{\bullet}_{?2}}{\sim}} \beta'(\vec{v}')$, there exist Σ''_1, σ''_1, and P'', such that

$$\langle \Sigma'_1; \sigma'_1 \rangle \overset{\beta'(\vec{v}')}{\twoheadrightarrow}_{\eta} \langle \Sigma''_1; \sigma''_1 \rangle \tag{29}$$

$$(\langle \Sigma''_2; \sigma''_2 \rangle, P'') \underset{\mathcal{P}}{\overset{\text{com}}{\sim}} (\langle \Sigma''_1; \sigma''_1 \rangle, P'') \tag{30}$$

By Lemma 33, $IC(\mathcal{P}^{\text{ch}}, \Sigma)$, and (21), there exists a content policy instantiating $P^{\bullet}_{?2}$ such that for any value vector of the appropriate length instantiating $\vec{v}'$, $\alpha' \underset{?}{\overset{P^{\bullet}_{?2}}{\sim}} \beta'(\vec{v}')$ holds.

We thus obtain (29) and (30) no matter which is the case for α'. By (29) and $\beta' = \epsilon$, we have $\Sigma''_1 = \Sigma'_1$ and $\sigma''_1 = \sigma'_1$. By the symmetry of $- \underset{\mathcal{P}}{\overset{\text{com}}{\sim}} -$, we have

$$(\langle \Sigma'_1; \sigma'_1 \rangle, P'') \underset{\mathcal{P}}{\overset{\text{com}}{\sim}} (\langle \Sigma''_2; \sigma''_2 \rangle, P''). \tag{31}$$

Building on the transition $\langle \Sigma_2; \sigma_2 \rangle \xrightarrow{\alpha'}_{\eta} \langle \Sigma''_2; \sigma''_2 \rangle$ and using (23), we have

$$\langle \Sigma_2; \sigma_2; q_2 \rangle \xrightarrow{\alpha'}_{\eta} \langle \Sigma''_2; \sigma''_2; q'_2 \rangle.$$

It is not difficult to see that the transition above qualifies as a simulation of transition (22) in R_{Δ}, since the pair $((\langle \Sigma'_1; \sigma'_1; q'_1 \rangle, P''), (\langle \Sigma''_2; \sigma''_2; q'_2 \rangle, P''))$ is in R_{Δ} (which can be deduced using (24) and (31)).

b. $\beta = \epsilon$.

i. Suppose $\neg(\exists \alpha' : \langle \Sigma_2; \sigma_2 \rangle \xrightarrow{\alpha'}_{\eta})$. Then it is not difficult to see that $\langle \Sigma_2; \sigma_2; q_2 \rangle \xrightarrow{\square}_{\eta} \langle \Sigma_2; \sigma_2; q'_2 \rangle$ qualifies as a simulation of transition (22).

ii. Suppose there exists the transition

$$\langle \Sigma_2; \sigma_2 \rangle \xrightarrow{\alpha'}_{\eta} \langle \Sigma''_2; \sigma''_2 \rangle, \tag{32}$$

for some α', Σ''_2 and σ''_2. We know from $\alpha \underset{!}{\overset{P^{\bullet}_{!}}{\sim}} \epsilon$ that α is a communication such that $P^{\circ}(ch(\alpha)) = H$. By Lemma 22, we have $\forall \alpha'' : \langle \Sigma_1; \sigma_1 \rangle \xrightarrow{\alpha''}_{\eta} \Rightarrow P^{\circ}(ch(\alpha'')) = H$. Therefore we have $\langle \Sigma_1; \sigma_1 \rangle \overset{\square}{\twoheadrightarrow}_{\eta} \langle \Sigma_1; \sigma_1 \rangle$. By Lemma 32, $\alpha' = \tau$ or $P^{\circ}(ch(\alpha')) = H$. Suppose $P^{\circ}(ch(\alpha')) = H$. The proof is similar to the case 1(a)ii, where there is a high communication from $\langle \Sigma_2; \sigma_2 \rangle$.

Suppose $\alpha' = \tau$. By symmetry of $- \underset{\mathcal{P}}{\overset{\text{com}}{\sim}} -$, and (19), we have

$$(\langle \Sigma_2; \sigma_2 \rangle, P) \underset{\mathcal{P}}{\overset{\text{com}}{\sim}} (\langle \Sigma_1; \sigma_1 \rangle, P).$$

By (32), there exist $P^{\bullet}_{!2}$ and β' such that $\tau \underset{!}{\overset{P^{\bullet}_{!2}}{\sim}} \beta'$, and for all $P^{\bullet}_{?2}$ and $\vec{v}'$ such that $\tau \underset{?}{\overset{P^{\bullet}_{?2}}{\sim}} \beta'(\vec{v}')$, there exist Σ''_1, σ''_1 and P', such that

$$\langle \Sigma_1; \sigma_1 \rangle \overset{\beta'(\vec{v}')}{\twoheadrightarrow}_{\eta} \langle \Sigma''_1; \sigma''_1 \rangle \tag{33}$$

$$(\langle \Sigma''_2; \sigma''_2 \rangle, P'') \underset{\mathcal{P}}{\overset{\text{com}}{\sim}} (\langle \Sigma''_1; \sigma''_1 \rangle, P'') \tag{34}$$

By $\tau \underset{!}{\overset{P^{\bullet}_{!2}}{\sim}} \beta'$, we have $\beta' = \tau$ or $\beta' = \square$. The universally quantified $P^{\bullet}_{?2}$ can be instantiated with any policy in the non-empty $\mathcal{P}^{\text{ch}}$, and $\vec{v}'$ with any value vector

of the appropriate length, for $\tau \overset{P^\bullet_{?2}}{\underset{?}{\sim}} \beta'(\vec{v}')$ to be satisfied. We thus obtain (33) and (34). By $P^\circ(\mathit{ch}(\alpha)) = H$, Lemma 22, and (33), β' cannot be τ. Hence $\beta' = \square$, and $\Sigma_1'' = \Sigma_1$ and $\sigma_1'' = \sigma_1$.
By symmetry of $- \overset{\text{com}}{\underset{\mathcal{P}}{\sim}} -$, and (34), we have

$$(\langle \Sigma_1; \sigma_1 \rangle, P'') \overset{\text{com}}{\underset{\mathcal{P}}{\sim}} (\langle \Sigma_2''; \sigma_2'' \rangle, P'').$$

By transition (25), there exist $P^\bullet_{!3}$ and β'' such that $\alpha \overset{P^\bullet_{!3}}{\underset{!}{\sim}} \beta''$, and for all $P^\bullet_{?3}$ and $\vec{v}''$ satisfying $\alpha \overset{P^\bullet_{?3}}{\underset{?}{\sim}} \beta''(\vec{v}'')$, there exist Σ_2''', σ_2''' and P''' such that

$$\langle \Sigma_2''; \sigma_2'' \rangle \overset{\beta''(\vec{v}'')}{\twoheadrightarrow}_\eta \langle \Sigma_2'''; \sigma_2''' \rangle \tag{35}$$

$$(\langle \Sigma_1'; \sigma_1' \rangle, P''') \overset{\text{com}}{\underset{\mathcal{P}}{\sim}} (\langle \Sigma_2'''; \sigma_2''' \rangle, P''') \tag{36}$$

By $P^\circ(\mathit{ch}(\alpha)) = H$, (20), and Lemma 33, there exists a content policy instantiating $P^\bullet_{?3}$, such that for all value vectors of the appropriate length for $\vec{v}''$, $\alpha \overset{P^\bullet_{?3}}{\underset{?}{\sim}} \beta''(\vec{v}'')$ holds. We thus obtain (35) and (36). We also know $\beta'' = \epsilon$; hence $\Sigma_2''' = \Sigma_2''$ and $\sigma_2''' = \sigma_2''$. Hence transition (22) can be simulated by $\langle \Sigma_2; \sigma_2; q_2 \rangle \xrightarrow{\tau}_\eta \langle \Sigma_2''; \sigma_2''; q_2' \rangle$ in R_Δ, resulting in $((\langle \Sigma_1'; \sigma_1'; q_1' \rangle, P'''), (\langle \Sigma_2''; \sigma_2''; q_2' \rangle, P''')) \in R_\Delta$.

2. Transition (22) is $\langle \Sigma_1; \sigma_1; q_1 \rangle \xrightarrow{\square}_\eta \langle \Sigma_1'; \sigma_1'; q_1' \rangle$. We have $\Sigma_1' = \Sigma_1$, $\sigma_1' = \sigma_1$, and

$$\neg(\exists \alpha' : \langle \Sigma_1; \sigma_1 \rangle \xrightarrow{\alpha'})_\eta. \tag{37}$$

Compared with case 1, we argue more briefly, since no new insight is needed. A case split is made on the transition that can happen from $\langle \Sigma_2; \sigma_2 \rangle$, over the processes in η. By (37) we have $\langle \Sigma_1; \sigma_1 \rangle \overset{\square}{\twoheadrightarrow}_\eta \langle \Sigma_1'; \sigma_1' \rangle$. By Lemma 32, the only possible cases are the following.

a. There is transition $\langle \Sigma_2; \sigma_2 \rangle \xrightarrow{\alpha'}_\eta \langle \Sigma_2'; \sigma_2' \rangle$ for some Σ_2', σ_2' and α' such that $P^\circ(\mathit{ch}(\alpha')) = H$. It can be deduced that $(\langle \Sigma_1; \sigma_1 \rangle, P') \overset{\text{com}}{\underset{\mathcal{P}}{\sim}} (\langle \Sigma_2'; \sigma_2' \rangle, P')$ for some P'. Hence the transition $\langle \Sigma_1; \sigma_1; q_1 \rangle \xrightarrow{\square}_\eta \langle \Sigma_1'; \sigma_1'; q_1' \rangle$ can be simulated by $\langle \Sigma_2; \sigma_2; q_2 \rangle \xrightarrow{\alpha'}_\eta \langle \Sigma_2'; \sigma_2'; q_2' \rangle$ in R_Δ.

b. There is transition $\langle \Sigma_2; \sigma_2 \rangle \xrightarrow{\tau}_\eta \langle \Sigma_2'; \sigma_2' \rangle$ for some Σ_2' and σ_2'. We can still deduce $(\langle \Sigma_1; \sigma_1 \rangle, P') \overset{\text{com}}{\underset{\mathcal{P}}{\sim}} (\langle \Sigma_2'; \sigma_2' \rangle, P')$ for some P'. The simulation in R_Δ is by the transition $\langle \Sigma_2; \sigma_2; q_2 \rangle \xrightarrow{\alpha'}_\eta \langle \Sigma_2'; \sigma_2'; q_2' \rangle$.

c. $\neg(\exists \alpha' : \langle \Sigma_2; \sigma_2 \rangle \xrightarrow{\alpha'})$. The transition $\langle \Sigma_1; \sigma_1; q_1 \rangle \xrightarrow{\square}_\eta \langle \Sigma_1'; \sigma_1'; q_1' \rangle$ (note that $\Sigma_1' = \Sigma_1$ and $\sigma_1' = \sigma_1$) can be simulated by $\langle \Sigma_2; \sigma_2; q_2 \rangle \xrightarrow{\square}_\eta \langle \Sigma_2; \sigma_2; q_2' \rangle$, resulting in $((\langle \Sigma_1; \sigma_1; q_1' \rangle, P), (\langle \Sigma_2; \sigma_2; q_2' \rangle, P)) \in R_\Delta$.

The cases above are exhaustive and the proof is complete. ◀