Véronique Cortier and Stéphanie Delaune

A method for proving observational equivalence

Research Report LSV-09-04

February 2009

Laboratoire
Spécification
et
Vérification

CENTRE NATIONAL
DE LA RECHERCHE
SCIENTIFIQUE

Ecole Normale Supérieure de Cachan
61, avenue du Président Wilson
94235 Cachan Cedex France

A method for proving observational equivalence *

Véronique Cortier
LORIA, CNRS & INRIA Nancy Grand Est, France

Stéphanie Delaune
LSV, ENS Cachan & CNRS & INRIA Saclay, France

Abstract

Formal methods have proved their usefulness for analyzing the security of protocols. Most existing results focus on trace properties like secrecy or authentication. There are however several security properties, which cannot be defined (or cannot be naturally defined) as trace properties and require the notion of observational equivalence. *Typical examples are anonymity, privacy related properties or statements closer to security properties used in cryptography.*

In this paper, we consider the applied pi calculus and we show that for determinate *processes, observational equivalence actually coincides with trace equivalence, a notion simpler to reason with. We exhibit a large class of determinate processes, called* simple processes, *that capture most existing protocols and cryptographic primitives. Then, for simple processes without replication, we reduce the decidability of trace equivalence to deciding an equivalence relation introduced by M. Baudet. Altogether, this yields the first decidability result of observational equivalence for a general class of equational theories.*

1 Introduction

Security protocols are paramount in today's secure transactions through public channels. It is therefore essential to obtain as much confidence as possible in their correctness. Formal methods have proved their usefulness for precisely analyzing the security of protocols. In the case of a bounded number of sessions, secrecy preservation is co-NP-complete [5, 21, 22], and for an unbounded number of sessions, several decidable classes have been identified (e.g. [20]). Many tools have also been developed to automatically verify cryptographic protocols (e.g. [9, 6]).

Most existing results focus on trace properties, that is, statements that something bad never occurs on any execution trace of a protocol. Secrecy and authentication are typical examples of trace properties. There are however several security properties, which cannot be defined (or cannot be naturally defined) as trace properties and require the notion of *observational equivalence*. We focus here on the definition proposed in the context of applied pi-calculus [2], which is well-suited for the analysis of security protocols. Two processes P and Q are observationally equivalent, denoted by $P \approx Q$, if for any process O the processes $P \mid O$ and $Q \mid O$ are equally able to emit on a given channel and are (weakly) bisimilar. This means that the process O cannot observe any difference between the processes P and Q.

Observational equivalence is crucial when specifying properties like anonymity that states that an observer cannot distinguish the case where A is talking from the case where B is talking (see [3]). Privacy related properties involved in electronic voting protocols (e.g. [15]) also use equivalence as a key notion and cannot be expressed in linear temporal logic. Observational equivalence is also used for defining a stronger notion of secrecy, called "strong secrecy" [10] or even for defining authentication [4]. More generally, it is a notion that allows to express flexible notions of security by requiring observational equivalence between a protocol and an idealized version of it, that magically realizes the desired properties.

Related work. In contrast to the case of trace properties, there are very few results on automating the analysis of observational equivalence. Decidability results are limited to fixed cryptographic primitives in spi-calculus (e.g. [18, 16]). In applied-pi calculus, an alternative approach has been considered [14, 7, 11] for

*This work has been partially supported by the ANR project SeSur AVOTÉ.

arbitrary cryptographic primitives. The approach consists in designing stronger notions of equivalences that imply observational equivalence. One of these techniques has been implemented in ProVerif [11]. None of these are however complete, that is, there exist observationally equivalent processes that do not satisfy these stronger notions of equivalences.

Our contributions. One of the difficulties in proving observational equivalence is the bisimulation property. Although bisimulation-based equivalences may be simpler to check than trace equivalences [19], in the context of cryptographic protocols, it seems easier to simply check *trace equivalence*, that is, equality of the set of execution traces (modulo some equivalence relation between traces). In particular, most decision techniques have been developed for trace properties only. However, it is well-known that this is not sufficient to ensure observational equivalence. J. Engelfriet has shown that observational equivalence and trace equivalence actually coincide in a general model of parallel computation with atomic actions, when processes are *determinate* [17]. Intuitively, a process P is determinate if after the same experiment s, the resulting processes are equivalent, that is, if $P \stackrel{s}{\Rightarrow} P'$ and $P \stackrel{s}{\Rightarrow} P''$ then $P' \approx P''$. Our first contribution is to generalize this result to the applied pi-calculus, which consists in the pi-calculus algebra enriched with terms and equational theories on terms.

Then we show that a large class of processes enjoys the determinacy property. More precisely, we design the class of *simple processes* and show that simple processes are determinate. Simple processes allow replication, `else` branches and arbitrary term algebra modulo an equational theory. Consequently, this class captures most existing security protocols and cryptographic primitives. In addition, our simple processes are close to the fragment considered in [13] for which cryptographic guarantees can be deduced from observational equivalence. The class of processes defined in [13] is however not determinate but we believe that their result could be easily extended to our class of simple processes, yielding to a decision technique for proving indistinguishability in cryptographic models.

Our third contribution is a decidability result for simple processes without replication nor `else` branch and for *convergent subterm theories*. Convergent subterm theories capture a wide array of functions, e.g. pairing, projections, various flavors of encryption and decryption, digital signatures, one-way hash functions, *etc.* We show that trace equivalence of simple processes without replication can be reduced to deciding an equivalence relation introduced by M. Baudet and which has been shown decidable for convergent subterm theories in [7].

Putting our three contributions together, we obtain decidability of observational equivalence for a large and interesting class of processes of the applied pi-calculus. This is the first decidability result for a general class of equational theories. Some of the proofs are omitted but can be found in appendix.

2 The applied pi calculus

The *applied pi calculus* [2] is a derivative of the pi calculus that is specialized for modeling cryptographic protocols. Participants in a protocol are modeled as processes, and the communication between them is modeled by means of message passing.

2.1 Syntax

To describe processes in the applied-pi calculus, one starts with a set of *names* $\mathcal{N} = \{a, b, \ldots, sk, k, n, \ldots\}$, which is split into the set $\mathcal{N}_b$ of names of *basic types* and the set $\mathcal{C}h$ of names of *channel type* (which are used to name communication channels). We also consider a set of *variables* $\mathcal{X} = \{x, y, \ldots\}$, and a *signature* $\mathcal{F}$ consisting of a finite set of *function symbols*. We rely on a sort system for terms. The details of the sort system are unimportant, as long as *base types* differ from *channel types*. We suppose that function symbols only operate on and return terms of base type.

Terms are defined as names, variables, and function symbols applied to other terms. Let $\mathsf{N} \subseteq \mathcal{N}$ and $\mathsf{X} \subseteq \mathcal{X}$, the set of terms built from N and X by applying function symbols in $\mathcal{F}$ is denoted by $\mathcal{T}(\mathsf{N}, \mathsf{X})$. Of course function symbol application must respect sorts and arities. We write $fv(T)$ for the set of variables occurring in T. The term T is said to be a *ground* term if $fv(T) = \emptyset$. We shall use $u, v, \ldots$ to denote *metavariables* that range over both names and variables.

Example 1 *Consider the following signature*

$$\mathcal{F} = \{\mathsf{enc}/2,\ \mathsf{dec}/2,\ \mathsf{pk}/1,\ \langle\ \rangle/2,\ \pi_1/1,\ \pi_2/1\}$$

that contains function symbols for asymmetric encryption, decryption and pairing, each of arity 2*, as well as projection symbols and the function symbol* pk*, each of arity* 1*. The ground term* $\mathsf{pk}(sk)$ *represents the public counterpart of the private key* sk*.*

In the applied pi calculus, one has *plain processes*, denoted P, Q, R and *extended processes*, denoted by A, B, C. Plain processes are built up in a similar way to processes in pi calculus except that messages can

$$
\begin{array}{ll}
P, Q, R := 0 & \text{plain processes} \\
\quad P \mid Q & \\
\quad !P & \\
\quad \nu n.P & \\
\quad \texttt{if } M = N \texttt{ then } P \texttt{ else } Q & \\
\quad \mathrm{in}(u, x).P & \\
\quad \mathrm{out}(u, N).P &
\end{array}
\qquad
\begin{array}{ll}
A, B, C := & \text{extended processes} \\
\quad P & \\
\quad A \mid B & \\
\quad \nu n.A & \\
\quad \nu x.A & \\
\quad \{^M/_x\} &
\end{array}
$$

where M and N are terms, n is a name, x a variable and u is a metavariable.

Figure 1. Syntax of processes

contain terms rather than just names. Extended processes add *active substitutions* and restriction on variables (see Figure 1).

The substitution $\{^M/_x\}$ is an active substitution that replaces the variable x with the term M. Active substitutions generalize the "let" construct: $\nu x.(\{^M/_x\} \mid P)$ corresponds exactly to

$$\text{"let } x = M \text{ in } P\text{"}.$$

As usual, names and variables have scopes, which are delimited by restrictions and by inputs. We write $fv(A)$, $bv(A)$, $fn(A)$ and $bn(A)$ for the sets of *free* and *bound variables* and *free* and *bound names* of A, respectively. We say that an extended process is *closed* if all its variables are either bound or defined by an active substitution. An *evaluation context* $C[\_]$ is an extended process with a hole instead of an extended process.

Active substitutions are useful because they allow us to map an extended process A to its *frame*, denoted $\phi(A)$, by replacing every plain process in A with 0. Hence, a frame is an extended process built up from 0 and active substitutions by parallel composition and restriction. The frame $\phi(A)$ accounts for the set of terms statically possessed by the intruder (but does not take into account for A's dynamic behavior). The domain of a frame φ, denoted by $\mathrm{dom}(\varphi)$, is the set of variables for which φ defines a substitution (those variables x for which φ contains a substitution $\{^M/_x\}$ not under a restriction on x).

Example 2 *Consider the following process A made up of three components in parallel:*

$$
\begin{array}{ll}
\nu s, sk, x_1. & (out(c_1, x_1) \\
& \mid in(c_1, y).out(c_2, \mathsf{dec}(y, sk)) \\
& \mid \{^{\mathsf{enc}(s,\mathsf{pk}(sk))}/_{x_1}\}).
\end{array}
$$

Its first component publishes the message $\mathsf{enc}(s, \mathsf{pk}(sk))$ stored in x_1 by sending it on c_1. The second receives a message on c_1, uses the secret key sk to decrypt it, and forwards the result on c_2. We have $\phi(A) = \nu s, sk, x_1.\{^{\mathsf{enc}(s,\mathsf{pk}(sk))}/_{x_1}\}$ and $\mathrm{dom}(\phi(A)) = \emptyset$ (since x_1 is under a restriction).

2.2 Semantics

We briefly recall the operational semantics of the applied pi calculus (see [2] for details). First, we associate an *equational theory* E to the signature $\mathcal{F}$. The equational theory is defined by a set of equations $M = N$ with $M, N \in \mathcal{T}(\emptyset, \mathsf{X})$ and induces an equivalence relation over terms: $=_{\mathsf{E}}$ is the smallest equivalence relation on terms, which contains all equations $M = N$ in E and that is closed under application of contexts and substitution of terms for variables. Since the equations in E do not contain any names, we have that E is also closed by substitutions of terms for names.

Example 3 *Consider the signature $\mathcal{F}$ of Example 1. We define the equational theory $\mathsf{E}_{\mathsf{enc}}$ by the following equations:*

$$
\begin{array}{rcl}
\mathsf{dec}(\mathsf{enc}(x, \mathsf{pk}(y)), y) & = & x \\
\pi_i(\langle x_1, x_2 \rangle) & = & x_i \quad \text{for } i \in \{1, 2\}.
\end{array}
$$

We have that $\pi_1(\mathsf{dec}(\mathsf{enc}(\langle n_1, n_2\rangle, \mathsf{pk}(sk)), sk)) =_{\mathsf{E}_{\mathsf{enc}}} n_1$.

Structural equivalence, noted $\equiv$, is the smallest equivalence relation on extended processes that is closed under α-conversion of names and variables, by application of evaluation contexts, and satisfying some further basic structural rules such as $A \mid 0 \equiv A$, associativity and commutativity of $\mid$, binding-operator-like behavior of ν, and when $M =_{\mathsf{E}} N$ the equivalences:

$$
\nu x.\{^M/_x\} \equiv 0 \qquad \{^M/_x\} \equiv \{^N/_x\}
$$
$$
\{^M/_x\} \mid A \equiv \{^M/_x\} \mid A\{^M/_x\}.
$$

Example 4 *Let P be the following process:*

$$
\begin{array}{ll}
\nu s, sk. & (out(c_1, \mathsf{enc}(s, \mathsf{pk}(sk))) \\
& \mid in(c_1, y).out(c_2, \mathsf{dec}(y, sk))).
\end{array}
$$

The process P is structurally equivalent to the process A given in Example 2. We have that $\phi(P) = 0 \equiv \phi(A)$.

The operational semantics of processes in the applied pi calculus is defined by structural rules defining two relations: *structural equivalence* (described above) and *internal reduction*, noted $\xrightarrow{\tau}$. Internal reduction is the smallest relation on extended processes closed under structural equivalence and application of evaluation contexts such that:

$$\begin{array}{rcl} \mathrm{out}(a,x).P \mid \mathrm{in}(a,x).Q & \xrightarrow{\tau} & P \mid Q \\ \texttt{if } M = M \texttt{ then } P \texttt{ else } Q & \xrightarrow{\tau} & P \\ \texttt{if } M = N \texttt{ then } P \texttt{ else } Q & \xrightarrow{\tau} & Q \end{array}$$

where M, N are ground terms such that $M \neq_{\mathsf{E}} N$

The operational semantics is extended by a *labeled* operational semantics enabling us to reason about processes that interact with their environment. Labeled operational semantics defines the relation $\xrightarrow{\ell}$ where ℓ is either an input or an output. We adopt the following rules in addition to the internal reduction rules. Below, the names a and c are channel names whereas x is a variable of base type and y is a variable of any type.

$$\textsc{In} \qquad \mathrm{in}(a,y).P \xrightarrow{in(a,M)} P\{^M/_y\}$$

$$\textsc{Out-Ch} \qquad \mathrm{out}(a,c).P \xrightarrow{out(a,c)} P$$

$$\textsc{Open-Ch} \qquad \frac{A \xrightarrow{out(a,c)} A' \qquad c \neq a}{\nu c.A \xrightarrow{\nu c.out(a,c)} A'}$$

$$\textsc{Out-T} \qquad \mathrm{out}(a,M).P \xrightarrow{\nu x.out(a,x)} P \mid \{^M/_x\} \qquad x \notin fv(P) \cup fv(M)$$

$$\textsc{Scope} \qquad \frac{A \xrightarrow{\ell} A' \quad u \text{ does not occur in } \ell}{\nu u.A \xrightarrow{\ell} \nu u.A'}$$

$$\textsc{Par} \qquad \frac{A \xrightarrow{\ell} A' \quad \begin{array}{c} bn(\ell) \cap fn(B) = \emptyset \\ bv(\ell) \cap fv(B) = \emptyset \end{array}}{A \mid B \xrightarrow{\ell} A' \mid B}$$

$$\textsc{Struct} \qquad \frac{A \equiv B \quad B \xrightarrow{\ell} B' \quad B' \equiv A'}{A \xrightarrow{\ell} A'}$$

Note that the labeled transition is not closed under application of evaluation contexts. Moreover the output of a term M needs to be made "by reference" using a restricted variable and an active substitution. The rules differ slightly from those described in [2] but it has been shown in [14] that the two underlying notions of observational equivalence coincide.

3 Trace and observational equivalences

Let $\mathcal{A}$ be the alphabet of actions (in our case this alphabet is infinite) where the special symbol $\tau \in \mathcal{A}$ represents an unobservable action. For every $\alpha \in \mathcal{A}$ the relation $\xrightarrow{\alpha}$ has been defined in Section 2. For every $w \in \mathcal{A}^*$ the relation $\xrightarrow{w}$ on extended processes is defined in the usual way. By convention $A \xrightarrow{\epsilon} A$ where ϵ denotes the empty word.

For every $s \in (\mathcal{A} \smallsetminus \{\tau\})^*$, the relation $\stackrel{s}{\Rightarrow}$ on extended processes is defined by: $A \stackrel{s}{\Rightarrow} B$ if, and only if, there exists $w \in \mathcal{A}^*$ such that $A \xrightarrow{w} B$ and s is obtained from w by erasing all occurrences of τ. Intuitively, $A \stackrel{s}{\Rightarrow} B$ means that A transforms into B by experiment s. We also consider the relation $A \stackrel{w}{\mapsto} B$ and $A \stackrel{s}{\Mapsto} B$ that are the restriction of the relations $\xrightarrow{w}$ and $\stackrel{s}{\Rightarrow}$ on closed extended processes.

3.1 Observational equivalence

Intuitively, two processes are *observationally equivalent* if they cannot be distinguished by any active attacker represented by any context.

We write $A \Downarrow c$ when A can send a message on c, that is, when $A \rightarrow^* C[\mathrm{out}(c, M).P]$ for some evaluation context C that does not bind c.

Definition 1 Observational equivalence *is the largest symmetric relation $\mathcal{R}$ between closed extended processes with the same domain such that $A \mathrel{\mathcal{R}} B$ implies:*

1. *if $A \Downarrow c$, then $B \Downarrow c$;*
2. *if $A \rightarrow^* A'$, then $B \rightarrow^* B'$ and $A' \mathrel{\mathcal{R}} B'$ for some B';*
3. *$C[A] \mathrel{\mathcal{R}} C[B]$ for all closing evaluation contexts C.*

Observational equivalence can be used to formalize many interesting security properties, in particular privacy related properties, such as those studied in [3, 15]. However, proofs of observational equivalences are difficult because of the universal quantification over all contexts. It has been shown that observational equivalence coincides with labeled bisimilarity [2]. Before defining this notion, we introduce a notion of intruder's knowledge that has been extensively studied (e.g. [1]).

Definition 2 (static equivalence $\sim$) *Two terms M and N are* equal in the frame ϕ, *written $(M =_{\mathsf{E}} N)\phi$, if there exists $\tilde{n}$ and a substitution σ such that $\phi \equiv \nu\tilde{n}.\sigma$, $\tilde{n} \cap (fn(M) \cup fn(N)) = \emptyset$, and $M\sigma =_{\mathsf{E}} N\sigma$.*

Two closed frames ϕ_1 and ϕ_2 are statically equivalent, *written $\phi_1 \sim \phi_2$, when:*

- $\mathrm{dom}(\phi_1) = \mathrm{dom}(\phi_2)$, *and*

- *for all terms M, N we have that*

$$(M =_{\mathsf{E}} N)\phi_1 \text{ if and only if } (M =_{\mathsf{E}} N)\phi_2.$$

Example 5 *Consider the theory $\mathsf{E}_{\mathsf{enc}}$ described in Example 3, and the two frames*

- $\varphi_a = \{^{\mathsf{enc}(a,\mathsf{pk}(sk))}/_{x_1}\}$*, and*
- $\varphi_b = \{^{\mathsf{enc}(b,\mathsf{pk}(sk))}/_{x_1}\}$.

We have that $(\mathsf{dec}(x_1, sk) =_{\mathsf{E}_{\mathsf{enc}}} a)\varphi_a$ *whereas* $(\mathsf{dec}(x_1, sk) \neq_{\mathsf{E}_{\mathsf{enc}}} a)\varphi_b$*, thus we have that* $\varphi_a \not\sim \varphi_b$.

However, we have that $\nu sk.\varphi \sim \nu sk.\varphi'$*. This is a non trivial equivalence. Intuitively, there is no test that allows one to distinguish the two frames since the decryption key and the encryption key are not available.*

Definition 3 (labeled bisimilarity $\approx$) Labeled bisimilarity *is the largest symmetric relation $\mathcal{R}$ on closed extended processes such that $A \;\mathcal{R}\; B$ implies*

1. $\phi(A) \sim \phi(B)$,
2. *if* $A \overset{\tau}{\mapsto} A'$*, then* $B \overset{\epsilon}{\Mapsto} B'$ *and* $A' \;\mathcal{R}\; B'$ *for some* B',
3. *if* $A \overset{\ell}{\mapsto} A'$ *and* $bn(\ell) \cap fn(B) = \emptyset$ *then* $B \overset{\ell}{\Mapsto} B'$ *and* $A' \;\mathcal{R}\; B'$ *for some* B'.

Example 6 *Consider the theory $\mathsf{E}_{\mathsf{enc}}$ and the two processes* $P_a = out(c, \mathsf{enc}(a, \mathsf{pk}(sk)))$ *and* $P_b = out(c, \mathsf{enc}(b, \mathsf{pk}(sk)))$*. We have that* $\nu sk.P_a \approx \nu sk.P_b$ *whereas* $P_a \not\approx P_b$*. These results are direct consequences of the static (in)equivalence relations stated and discussed in Example 5.*

3.2 Trace equivalence

For every closed extended process A we define its set of traces, each trace consisting in a sequence of actions together with the sequence of sent messages:

$$\mathsf{trace}(A) = \{(s, \phi(B)) \mid A \overset{s}{\Mapsto} B \text{ for some } B\}.$$

Note that, in the applied pi calculus, the sent messages are exclusively stored in the frame and not in the sequence s (the outputs are made by "reference").

Definition 4 (trace inclusion $\sqsubseteq_t$) *Let A and B be two closed extended processes, $A \sqsubseteq_t B$ if for every $(s, \varphi) \in \mathsf{trace}(A)$ such that $bn(s) \cap fn(B) = \emptyset$, there exists $(s', \varphi') \in \mathsf{trace}(B)$ such that $s = s'$ and $\varphi \sim \varphi'$.*

Definition 5 (trace equivalence $\approx_t$) *Two closed extended processes A and B are* trace equivalent*, denoted by $A \approx_t B$, if $A \sqsubseteq_t B$ and $B \sqsubseteq_t A$.*

It is easy to see that observational equivalence (or labeled bisimilarity) implies trace equivalence while the converse is false in general (see Example 7).

Lemma 1 *Let A and B be two closed extended processes: $A \approx B$ implies $A \approx_t B$.*

Example 7 *Consider the two following processes:*

$$\begin{aligned} A = \nu c'.(out(c', ok) \;&\mid in(c', x).out(c, a).out(c, b_1) \\ &\mid in(c', x).out(c, a).out(c, b_2)) \end{aligned}$$

$$\begin{aligned} B = out(c, a).\nu c'.(out(c', ok) \;&\mid in(c', x).out(c, b_1) \\ &\mid in(c', x).out(c, b_2)). \end{aligned}$$

We have that $A \approx_t B$ whereas $A \not\approx B$. Intuitively, after B's first move, B still has the choice of emitting b_1 or b_2, while A, trying to follow B's first move, is forced to choose between two states from which she can only emit one of the two.

3.3 Determinacy

J. Engelfriet has shown that observational and trace equivalence coincide for a process algebra with atomic actions, when processes are *determinate* [17].

Definition 6 (determinacy) *Let $\cong$ be an equivalence relation on closed extended processes. A closed extended process A is* $\cong$-determinate *if whenever $A \overset{s}{\Mapsto} B$, $A \overset{s}{\Mapsto} B'$ and $\phi(B) \sim \phi(B')$ then $B \cong B'$.*

Fixing the equivalence relation yields to potentially different notions of determinacy. We define two of them: *observation determinacy* (for $\cong := \approx$) and *trace determinacy* (for $\cong := \approx_t$). By using the techniques of J. Engelfriet, we can show that these two notions of determinacy actually coincide. So we say that an extended process is *determinate* if it satisfies any of these two notions.

Lemma 2 *Let A be a closed extended process. The process A is observation determinate if, and only if, it is trace determinate.*

Example 8 *Consider for instance the closed extended process A given in Example 7. We have that $A \overset{\tau}{\mapsto} A_1$ and $A \overset{\tau}{\mapsto} A_2$ for A_1 and A_2 given below:*

$$\begin{aligned} A_1 = \nu c'.\quad &(out(c, a).out(c, b_1) \\ &\mid in(c', x).out(c, a).out(c, b_2)) \end{aligned}$$

$$\begin{aligned} A_2 = \nu c'.\quad &(in(c', x).out(c, a).out(c, b_1) \\ &\mid out(c, a).out(c, b_2)). \end{aligned}$$

The process A_1 can output the messages a and then b_1 whereas the process A_2 can output a and then b_2. Thus, the process A is neither observation determinate, nor trace determinate.

Our first main contribution is to extend the result of J. Engelfriet [17] to processes of the applied-pi calculus, showing that observational equivalence and trace equivalence coincide when processes are determinate.

Theorem 1 *Let A and B be two closed extended processes that are determinate.*

$$A \approx_t B \text{ implies } A \approx B.$$

Proof (sketch). Let A and B be two closed extended processes that are determinate, and assume that $A \approx_t B$. We consider the relation $\mathcal{R}$ defined as follows:

$$A' \mathrel{\mathcal{R}} B' \text{ iff there exists } s \text{ such that } A \overset{s}{\Rightarrow} A',\ B \overset{s}{\Rightarrow} B', \text{ and } \phi(A') \sim \phi(B').$$

We have that $A \mathrel{\mathcal{R}} B$. It remains to check that $\mathcal{R}$ satisfies the three points of Definition 3. $\square$

4 An expressive class of determinate processes

In what follows, we consider any signature and equational theory. We do not need the full applied pi-calculus to represent security protocols. For example, it is generally assumed that all communications are controlled by the attacker thus private channels between processes are not accurate (they should rather be implemented using cryptography). In addition, the attacker schedules the communications between processes thus he knows exactly to whom he is sending messages and from whom he is listening. Thus we assume that each process communicates on a personal channel.

Formally, we consider the fragment of *simple processes* built on *basic processes*. A basic process represents a session of a protocol role where a party waits for a message of a certain form or checks some equalities and outputs a message accordingly. Then the party waits for another message or checks for other equalities and so on.

Intuitively, any protocol whose roles have a deterministic behavior can be modeled as a simple process. Most of the roles are indeed deterministic since an agent should usually exactly know what to do once he has received a message. In particular, all protocols of the Clark and Jacob library [12] can be modeled as simple processes.

Definition 7 (basic process) *The set $\mathcal{B}(c, \mathcal{V})$ of* basic processes *built from $c \in \mathcal{C}h$ and $\mathcal{V} \subseteq \mathcal{X}$ (variables of base type) is the least set of processes that contains* 0 *and such that*

- *if $B_1, B_2 \in \mathcal{B}(c, \mathcal{V})$, $M, N, s_1, s_2 \in \mathcal{T}(\mathcal{N}_b, \mathcal{V})$, then*
$$\mathsf{if}\ M = N\ \mathsf{then}\ out(c, s_1).B_1\ \mathsf{else}\ out(c, s_2).B_2 \in \mathcal{B}(c, \mathcal{V}).$$
- *if $B \in \mathcal{B}(c, \mathcal{V} \uplus \{x\})$, x of base type ($x \notin \mathcal{V}$), then*
$$in(c, x) \cdot B \in \mathcal{B}(c, \mathcal{V}).$$

Intuitively, in a basic process, depending on the outcome of the test, the process sends on its channel c a message depending on its inputs. A basic process may also input messages on its channel c.

Example 9 *We consider a slightly simplified version of a protocol given in [3] designed for transmitting a secret without revealing its identity to other participants. In this protocol, A is willing to engage in communication with B and wants to reveal its identity to B. However, A does not want to compromise its privacy by revealing its identity or the identity of B more broadly. The participants A and B proceed as follows:*

$$\begin{array}{lcl} A \to B & : & \mathsf{enc}(\langle N_a, \mathsf{pub}(A)\rangle, \mathsf{pub}(B)) \\ B \to A & : & \mathsf{enc}(\langle N_a, \langle N_b, \mathsf{pub}(B)\rangle\rangle, \mathsf{pub}(A)) \end{array}$$

First A sends to B a nonce N_a and its public key encrypted with the public key of B. If the message is of the expected form then B sends to A the nonce N_a, a freshly generated nonce N_b and its public key, all of this being encrypted with the public key of A. Otherwise, B sends out a "decoy" message: $\mathsf{enc}(N_b, \mathsf{pub}(B))$. This message should basically look like B's other message from the point of view of an outsider. This is important since the protocol is supposed to protect the identity of the participants.

A session of role A played by agent a with b can be modeled by the following basic process where true *denotes a test that is always satisfied and $M = \mathsf{dec}(x, ska)$. Note that A is not given the value skb but is directly given the value $\mathsf{pk}(skb)$, that is the public key corresponding to B's private key.*

$$\begin{array}{l} A(a,b) \overset{def}{=} \\ \quad \mathtt{if}\ \mathsf{true}\ \mathtt{then} \\ \qquad out(c_A, \mathsf{enc}(\langle n_a, \mathsf{pk}(ska)\rangle, \mathsf{pk}(skb))). \\ \qquad in(c_A, x). \\ \qquad \mathtt{if}\ \langle \pi_1(M), \pi_2(\pi_2(M))\rangle = \langle n_a, \mathsf{pk}(skb)\rangle\ \mathtt{then}\ 0 \\ \qquad\qquad \mathtt{else}\ 0 \\ \qquad\quad \mathtt{else}\ 0. \end{array}$$

Similarly, a session of role B played by agent b with a can be modeled by the basic process $B(b,a)$ where $N = \mathsf{dec}(y, skb)$.

$$\begin{array}{rl} B(b,a) \stackrel{def}{=} & in(c_B, y). \\ & \quad \texttt{if } \pi_2(N) = \mathsf{pk}(ska) \texttt{ then} \\ & out(c_B, \mathsf{enc}(\langle \pi_1(N), \langle n_b, \mathsf{pk}(skb)\rangle\rangle, \mathsf{pk}(ska))).0 \\ & \texttt{else } out(c_B, \mathsf{enc}(nb, \mathsf{pk}(skb))).0. \end{array}$$

Intuitively, this protocol preserves anonymity if an attacker cannot distinguish whether b is willing to talk to a (represented by the process $B(b,a)$) or willing to talk to a' (represented by the process $B(b,a')$), provided a, a' and b are honest participants. For illustration purposes, we also consider the process $B'(b,a)$ obtained from $B(b,a)$ by replacing the `else` *branch by* `else` *0. We will see that the "decoy" message plays a crucial role to ensure privacy.*

Definition 8 (simple process) *A* simple process *is obtained by composing and replicating basic processes and frames, hiding some names:*

$$\begin{array}{lllll} \nu\tilde{n}. (& \nu\tilde{n}_1.(B_1 \mid \sigma_1) & \mid & !(\nu c'_1, \tilde{m}_1.out(p_1, c'_1).B'_1) & \\ & \vdots & & \vdots & \\ & \nu\tilde{n}_k.(B_k \mid \sigma_k) & \mid & !(\nu c'_n, \tilde{m}_n.out(p_n, c'_n).B'_n) &) \end{array}$$

where $B_j \in \mathcal{B}(c_j, \emptyset)$, $B'_j \in \mathcal{B}(c'_j, \emptyset)$ and c_j are channel names that are pairwise distinct. The names $p_1, \ldots, p_n$ are distinct channel names that do not appear elsewhere and $\sigma_1, \ldots, \sigma_k$ are frames without restricted names (i.e. substitutions).

Each basic process B'_j first publishes its channel name c'_j on the public channel p_j so that an attacker can communicate with it. Intuitively the public channels $p_1, \ldots, p_n$ indicate from which role the channel name c'_i is emitted. Names of base types may be shared between processes, this is the purpose of $\tilde{n}$.

It is interesting to notice that protocols with deterministic behavior are usually not modeled within our fragment (see e.g. [2]) since a single channel is used for all communications. We think however that using a single channel does not provide enough information to the attacker since he is not able to schedule exactly the messages to the processes and he does not know from which process a message comes from while this information is usually available (via e.g. IP adresses). For example, a role emitting the constant a twice would be modeled by $P_1 = \mathrm{out}(c,a).\mathrm{out}(c,a).0$ while two roles emitting each the constant a would be modeled by $P_2 = \mathrm{out}(c,a).0 \mid \mathrm{out}(c,a).0$. Then P_1 and P_2 are observationally equivalent while the two protocols could be distinguished in practice, which is reflected in our modeling in simple processes.

Example 10 *Continuing Example 9, a simple process representing unbounded number of sessions in which a plays A (with b) and b plays B with a is:*

$$\begin{array}{ll} \nu ska, skb. & (\, !(\nu n_a, c_A.out(p_A, c_A).A(a,b)) \\ & \mid \, !(\nu n_b, c_B.out(p_B, c_B).B(b,a)) \,) \end{array}$$

For modelling and verification purposes, we may want to disclose the public keys in order to make them available to the attacker. This can be done by means of an additional basic process

$$K(a,b) = out(c_K, \mathsf{pk}(ska)) \cdot out(c_K, \mathsf{pk}(skb)).0.$$

Simple processes is a large class of processes that are determinate. Indeed, since each basic process has its own channel to send and receive messages, all the communications are visible to the attacker. Moreover, the attacker knows exactly who is sending a message or from whom he is receiving a message. Actually, given a simple process A a sequence of actions tr, there is a unique process B (up to some internal reduction steps) such that $A \stackrel{\mathsf{tr}}{\Rightarrow} B$.

Theorem 2 *Any simple process is determinate.*

Applying Theorems 1 and 2, we get that, on simple processes, it is sufficient to check trace equivalence to prove observational equivalence.

Corollary 1 *Let A and B be two simple processes: $A \approx_t B$ if, and only if, $A \approx B$.*

5 Intermediate calculus

Reasoning on processes of the applied-pi calculus is quite involved since it requires one to consider all the rules defining the labeled transition relation $\xrightarrow{\alpha}$. Thus we use a simplified fragment of the class of *intermediate processes*, defined in [14], that are easier to manipulate and such that trace equivalence of simple processes without replication nor else branch is equivalent to trace equivalence of their corresponding intermediate processes.

5.1 Syntax

The grammar of the *plain intermediate processes* is as follows:

$$\begin{array}{l} P, Q, R := 0 \\ \quad \texttt{if } M_1 = M_2 \texttt{ then } P \texttt{ else } Q \\ \quad \mathrm{in}(c,x).P \\ \quad \mathrm{out}(c,N).P \end{array}$$

where $c \in \mathcal{C}h$ is channel name, M_1, M_2 are terms of base type, x is a variable of base type, and N is a message of base type. Terms M_1, M_2 and N can also use variables.

Definition 9 (intermediate process) *An* intermediate process *is a triple* $(\mathcal{E}; \mathcal{P}; \Phi)$ *where:*

- $\mathcal{E}$ *is a set of names that represents the names restricted in* $\mathcal{P}$*;*
- $\Phi = \{w_1 \rhd t_1, \ldots, w_n \rhd t_n\}$ *where* $t_1, \ldots, t_n$ *are ground terms, and* $w_1, \ldots, w_n$ *are variables;*
- $\mathcal{P}$ *is a multiset of* plain intermediate processes *(defined below) where null processes are removed and such that* $fv(\mathcal{P}) \subseteq \{w_1, \ldots, w_m\}$.

Additionally, we require intermediate processes to be variable distinct*, i.e. any variable is at most bound once.*

Given a sequence $\Phi = \{w_1 \rhd t_1, \ldots, w_n \rhd t_n\}$ where $t_1, \ldots, t_n$ are terms, we also denote by Φ its associated frame, i.e. $\{^{t_1}/_{w_1}\} \mid \ldots \mid \{^{t_n}/_{w_n}\}$.

Given a closed extended process A of the original applied pi without replication, we can easily transform it into an intermediate process $\tilde{A} = (\mathcal{E}; \mathcal{P}; \Phi)$ such that $A \approx \nu\mathcal{E}.(\mathcal{P} \mid \Phi)$. The idea is to rename names and variables to avoid clashes, to apply the active substitutions (Subst), to remove the restrictions on variables (Alias), and finally to push the restrictions on names in front of the process. We can also add some restricted names not appearing in the process in front of it. This will be useful to obtain two intermediate processes with the same set of restricted names.

Example 11 *Consider the extended process* A *described below (*M *is a term such that* $n \notin fn(M)$*):*

$$\nu sk.\nu x.(out(c, \mathsf{enc}(x, \mathsf{pk}(sk))).\nu n.out(c, n) \mid \{^M/_x\}).$$

An intermediate process A' *associated to* A *is:*

$$\begin{aligned} A' &= (\mathcal{E}; \mathcal{P}; \Phi) \\ &= (\{sk, n\}; out(c, \mathsf{enc}(M, \mathsf{pk}(sk))).out(c, n); \emptyset). \end{aligned}$$

We have that $A \approx \nu\mathcal{E}.(\mathcal{P} \mid \Phi)$*. However, note that* A *and* $\nu\mathcal{E}.(\mathcal{P} \mid \Phi)$ *are not in structural equivalence. Indeed, structural equivalence does not allow one to push all the restrictions in front of a process.*

5.2 Semantics

From now on, we consider intermediate processes without else branch, that is we assume that any subprocess of the form `if` $M = N$ `then` P `else` Q is such that $Q = 0$. The semantics for intermediate processes (without else branch) is given in Figure 2. Let $\mathcal{A}_i$ be the alphabet of actions for the intermediate semantics. For every $w \in \mathcal{A}_i^*$ the relation $\xrightarrow{w}_i$ on intermediate processes is defined in the usual way. For $s \in (\mathcal{A}_i \smallsetminus \{\tau\})^*$, the relation $\stackrel{s}{\Rightarrow}_i$ on intermediate processes is defined by: $A \stackrel{s}{\Rightarrow}_i B$ if, and only if there exists $w \in \mathcal{A}_i^*$ such that $A \xrightarrow{w}_i B$ and s is obtained by erasing all occurrences of τ. Note that by definition, intermediate processes are closed.

5.3 Equivalence

Let $A = (\mathcal{E}_1; \mathcal{P}_1; \Phi_1)$ be an intermediate process. We define the following set:

$$\mathsf{trace}_i(A) = \{(s, \nu\mathcal{E}_2.\Phi_2) \mid (\mathcal{E}_1; \mathcal{P}_1; \Phi_1) \stackrel{s}{\Rightarrow}_i (\mathcal{E}_2; \mathcal{P}_2; \Phi_2) \text{ for some } (\mathcal{E}_2; \mathcal{P}_2; \Phi_2)\}$$

Definition 10 ($\approx_t$ for intermediate processes) *Let* A *and* B *be two intermediate processes having the same set of restricted names, i.e.* $A = (\mathcal{E}; \mathcal{P}_1; \Phi_1)$ *and* $B = (\mathcal{E}; \mathcal{P}_2; \Phi_2)$.

The processes A *and* B *are* intermediate trace equivalent*, denoted by* $A \approx_t B$*, if for every* $(s, \varphi) \in \mathsf{trace}_i(A)$ *there exists* $(s', \varphi') \in \mathsf{trace}_i(B)$ *such that* $s = s'$ *and* $\varphi \sim \varphi'$ *(and conversely).*

Despite the differences between the two semantics, it can be shown that the two notions of trace equivalence coincide [14]. For intermediate processes derived from simple processes, we wish to obtain a similar result for a more detailed notion of trace, called *annotated trace.*

Annotated traces are obtained by replacing the label τ of the rule Then$_i$ in Figure 2 with test_p where p is the identity of the process, i.e. the name of its channel. If $A_i \xrightarrow{a_1}_i \ldots \xrightarrow{a_n}_i A'_i$, we denote by $\overline{a_1 \cdot \ldots \cdot a_n}$ the trace obtained from $a_1 \cdot \ldots \cdot a_n$ by replacing any test_p by τ, recovering a trace for the previous definition of trace. We can easily adapt the definition of trace and trace equivalence, yielding to annotated trace and annotated trace equivalence.

We show that on simple processes without else branch nor replication, trace equivalence coincides with annotated trace equivalence.

Proposition 1 *Let* A *and* B *be two simple processes without else branch nor replication. Let* $\tilde{A} = (\mathcal{E}; \mathcal{P}_A; \Phi_A)$ *and* $\tilde{B} = (\mathcal{E}; \mathcal{P}_B; \Phi_B)$ *be the two associated intermediate processes.*

The processes A *and* B *are trace equivalent (i.e.* $A \approx_t B$ *in the original applied pi calculus semantics) if, and only if,* $\tilde{A}$ *and* $\tilde{B}$ *are annotated trace equivalent.*

$$(\mathcal{E}; \{\texttt{if } u = v \texttt{ then } P \texttt{ else } Q\} \uplus \mathcal{P}; \Phi) \xrightarrow{\tau}_i (\mathcal{E}; \{P\} \uplus \mathcal{P}; \Phi) \text{ if } u =_{\mathsf{E}} v \qquad (\text{Then}_i)$$

$$(\mathcal{E}; \{\text{in}(p, x).P\} \uplus \mathcal{P}; \Phi) \xrightarrow{\text{in}(p,M)}_i (\mathcal{E}; \{P\{x \mapsto u\}\} \uplus \mathcal{P}; \Phi) \qquad (\text{In}_i)$$
$$M\Phi = u,\ fv(M) \subseteq \text{dom}(\Phi) \text{ and } fn(M) \cap \mathcal{E} = \emptyset$$

$$(\mathcal{E}; \{\text{out}(p, u).P\} \uplus \mathcal{P}; \Phi) \xrightarrow{\nu w_n.\text{out}(p,w_n)}_i (\mathcal{E}; \{P\} \uplus \mathcal{P}; \Phi \cup \{w_n \rhd u\}) \qquad (\text{Out-T}_i)$$
$$w_n \text{ variable such that } n = |\Phi| + 1$$

u, v and x are terms of base type whereas p is a channel name.

Figure 2. Intermediate semantics of simple processes

The proof relies on the result of [14] that states that two processes are trace equivalent if and only if the corresponding intermediate processes are intermediate trace equivalent. We then need to show that traces can be grouped following the annotation, which is due to the determinism of simple processes.

6 A decision procedure for observational equivalence

The aim of the section is to provide a decision procedure for trace equivalence and for a large class of processes (namely the class of simple processes), for the class of convergent subterm equational theories. Starting from intermediate processes that are obtained from simple processes without else branch nor replication, we reduce trace equivalence to equivalence of constraint systems. We can then conclude by using the decision procedure proposed in [7, 8] for constraint systems for the class of convergent subterm equational theories.

6.1 Constraint system

Following the notations of [7], we consider a new set $\mathcal{X}^2$ of variables called *second order variables* $X, Y, \ldots$, each variable with an arity, denoted $ar(X)$. We denote by $\mathsf{var}^1(\mathcal{C})$ (resp. $\mathsf{var}^2(\mathcal{C})$) the first order (resp. second order) variables of $\mathcal{C}$, that is $\mathsf{var}^1(\mathcal{C}) = fv(\mathcal{C}) \cap \mathcal{X}$ (resp. $\mathsf{var}^2(\mathcal{C}) = fv(\mathcal{C}) \cap \mathcal{X}^2$).

A constraint system represents the possible executions of a protocol once an interleaving has been fixed.

Definition 11 (constraint system [7]) *A* constraint system *is a triple* $(\mathcal{E}; \Phi; \mathcal{C})$*:*

- $\mathcal{E}$ *is a set of names (names that are initially unknown to the attacker);*
- Φ *is a sequence of the form* $\{w_1 \rhd t_1, \ldots, w_n \rhd t_n\}$ *where* t_i *are terms and* w_i *are variables. The* t_i *represent the terms sent on the network, their variables represent messages sent by the attacker.*
- $\mathcal{C}$ *is a set of constraints of the form* $X \rhd^? x$ *with* $ar(X) \leq n$*, or of the form* $s =^?_{\mathsf{E}} s'$ *where* s, s' *are first-order terms. Intuitively, the constraint* $X \rhd^? x$ *is meant to ensure that* x *will be replaced by a deducible term.*

The *size* of Φ, denoted $|\Phi|$ is its length n. We also assume the following conditions:

1. for every $x \in \mathsf{var}^1(\mathcal{C})$, there exists a unique X such that $(X \rhd^? x) \in \mathcal{C}$, and each variable X occurs at most once in $\mathcal{C}$.
2. for every $1 \leq k \leq n$, for every $x \in \mathsf{var}^1(t_k)$, there exists $(X \rhd^? x) \in \mathcal{C}$ such that $ar(X) < k$.

Given a term T with variables $w_1, \ldots, w_k$ and $\Phi = \{w_1 \rhd t_1, \ldots, w_n \rhd t_n\}$, $n \geq k$, $T\Phi$ denotes the term T where each w_i has been replaced by t_i. The *structure* of $(\mathcal{E}; \Phi; \mathcal{C})$ is given by $\mathcal{E}$, $|\Phi|$ and $\mathsf{var}^2(\mathcal{C})$ with their arity.

Example 12 *The triple* $\Sigma_s = (\mathcal{E}_s; \Phi^0_s \cup \{w_4 \rhd t\}; \mathcal{C}_s)$ *where*

$\mathcal{E}_s = \{ska, ska', skb, n_a, n_b\}$,
$\Phi^0_s = \{w_1 \rhd \mathsf{pk}(ska), w_2 \rhd \mathsf{pk}(ska'), w_3 \rhd \mathsf{pk}(skb)\}$,
$t = \mathsf{enc}(\langle \pi_1(\mathsf{dec}(y, skb)), \langle n_b, \mathsf{pk}(skb)\rangle\rangle, \mathsf{pk}(ska))$,
$\mathcal{C}_s = \{Y \rhd^? y,\ \pi_2(\mathsf{dec}(y, skb)) =^?_{\mathsf{E}} \mathsf{pk}(ska)\}, ar(Y) = 3$

is a constraint system. We will see that it corresponds to the execution of the process $B'(b, a)$ *presented in Example 9. We consider three agents (*a*,* a' *and* b*) so that the attacker can try to learn whether* b *is willing to talk to* a *or to* a'*. Their public keys are made available to the attacker.*

Definition 12 (solution) *A* solution *of a constraint system* $\Sigma = (\mathcal{E}; \Phi; \mathcal{C})$ *is a substitution* θ *such that*

- $\text{dom}(\theta) = \mathsf{var}^2(\mathcal{C})$*, and*
- $X\theta \in \mathcal{T}(\mathcal{N}_b \smallsetminus \{\mathcal{E}\}, \text{dom}(\Phi))$ *for any* $X \in \text{dom}(\theta)$*.*

Moreover, we require that there exists a closed substitution λ *with* $\text{dom}(\lambda) = \mathsf{var}^1(\mathcal{C})$ *such that:*

1. *for every* $(X \rhd^? x) \in \mathcal{C}$, $(X\theta)(\Phi\lambda) = x\lambda$;

2. *for every* $(s =^?_{\mathsf{E}} s') \in \mathcal{C}$, $s\lambda =_{\mathsf{E}} s'\lambda$;

The substitution λ *is called* first order solution *of* Σ *associated to* θ. *The set of solutions of a constraint system* Σ *is denoted* $Sol(\Sigma)$.

Example 13 *Continuing Example 12, a solution to* $\Sigma_s = (\mathcal{E}_s; \Phi_s; \mathcal{C}_s)$ *is* θ *where* $\mathrm{dom}(\theta) = \{Y\}$ *and* $\theta(Y) = \mathsf{enc}(\langle n_i, w_1\rangle, w_3)$ *with* n_i *a public name (i.e.* $n_i \notin \mathcal{E}_s$*). The first order-solution* λ *of* Σ_s *associated to* θ *is a substitution whose domain is* $\{y\}$ *and such that* $\lambda(y) = \mathsf{enc}(\langle n_i, \mathsf{pk}(ska)\rangle, \mathsf{pk}(skb))$.

A constraint system Σ is *satisfiable* if $Sol(\Sigma) \neq \emptyset$. Two constraint systems Σ_1 and Σ_2 with the same structures are *equivalent* if and only if $Sol(\Sigma_1) = Sol(\Sigma_2)$. We further define S-equivalence [7] that will be useful to capture static equivalence.

Definition 13 (S-equivalence) *Let* $\Sigma_1 = (\mathcal{E}; \Phi_1; \mathcal{C}_1)$ *and* $\Sigma_2 = (\mathcal{E}; \Phi_2; \mathcal{C}_2)$ *be two constraint systems with the same structure and consider* $x, y \notin \mathsf{var}^1(\mathcal{C}_i)$ *and* $X, Y \notin \mathsf{var}^2(\mathcal{C}_i)$ *for* $i = 1, 2$. *The two systems* Σ_1 *and* Σ_2 *are* S-equivalent *if the constraint systems:*

- $(\mathcal{E}; \Phi_1; \mathcal{C}_1 \cup \{X \rhd^? x,\, Y \rhd^? y,\, x =^?_{\mathsf{E}} y\})$, *and*
- $(\mathcal{E}; \Phi_2; \mathcal{C}_2 \cup \{X \rhd^? x,\, Y \rhd^? y,\, x =^?_{\mathsf{E}} y\})$

are equivalent.

Example 14 *Let* Σ'_s *be the constraint system below:*

$$(\mathcal{E}_s; \Phi^0_s \cup \{w_4 \rhd t'\}; Y \rhd^? y, \pi_2(\mathsf{dec}(y, skb)) =^?_{\mathsf{E}} \mathsf{pk}(ska'))$$

where $t' = \mathsf{enc}(\langle \pi_1(\mathsf{dec}(y, skb)), \langle n_b, \mathsf{pk}(skb)\rangle\rangle, \mathsf{pk}(ska'))$, *and* $\mathcal{E}_s$, Φ^0_s *are defined as in Example 12. We will see that this system corresponds to the system obtained after a symbolic execution of the process* $B'(b, a')$ *presented in Example 9.*

The system Σ_s *(given in Example 12) is not equivalent to* Σ'_s. *Indeed, the substitution* θ *given in Example 13 is such that* $\theta \in Sol(\Sigma_s)$ *whereas* $\theta \notin Sol(\Sigma'_s)$. *We conclude that the constraint systems* Σ_s *and* Σ'_s *are not equivalent, and thus not in* S*-equivalence. Actually, this corresponds to the fact that an attacker can distinguish between* $B'(b, a)$ *and* $B'(b, a')$ *by sending a message* $\mathsf{enc}(\langle n, \mathsf{pk}(ska)\rangle, \mathsf{pk}(skb))$ *and see whether* b *answers or not.*

6.2 Symbolic calculus

Following the approach of [8], we compute from an intermediate process $P = (\mathcal{E}; \mathcal{P}; \Phi)$ the set of constraints systems capturing the possible executions of P, starting from $P_s \stackrel{\text{def}}{=} (\mathcal{E}; \mathcal{P}; \Phi; \emptyset)$ and applying the rules defined in Figure 3.

Definition 14 (symbolic process) *A* symbolic process *is a tuple* $(\mathcal{E}; \mathcal{P}; \Phi; \mathcal{C})$ *where:*

- $\mathcal{E}$ *is a set of names;*
- $\mathcal{P}$ *is a multiset of plain intermediate processes where null processes are removed and such that* $fv(\mathcal{P}) \subseteq \{x \mid X \rhd^? x \in \mathcal{C}\}$;
- $(\mathcal{E}, \Phi, \mathcal{C})$ *is a constraint system.*

The rules of Figure 3 define the semantics of symbolic processes. The aim of this symbolic semantics is to avoid the infinite branching due to the inputs of the environment. This is achieved by keeping variables rather than the input terms. The constraint system gives a finite representation of the value that these variables are allowed to take.

The $\text{T}\textsc{hen}_s$ (resp. IN_s) rule allows the process to pass a test (resp. an input). The corresponding constraint is added in the set of constraints $\mathcal{C}$. When a process is ready to output a term on a public channel p, the outputted term is added to the frame Φ, which means that this term is made available to the attacker.

Example 15 *We consider one session of the protocol presented in Example 9, in which* b *plays the role* B' *(with* a*) and* a *plays the role* A *with* b*. We consider the following process* $K(a, a', b)$ *that models keys disclosure, i.e.*

$$out(c_K, \mathsf{pk}(ska)).out(c_K, \mathsf{pk}(ska')).out(c_K, \mathsf{pk}(skb)).$$

Let $\mathcal{E}$ *be the set of names* $\{ska, ska', skb, n_a, n_b\}$, *and* P^s_{ex} *the following symbolic process:*

$$P^s_{\mathsf{ex}} = (\mathcal{E}; \{A(a, b),\, B'(b, a),\, K(a, a', b)\}; \emptyset; \emptyset).$$

We have that $P^s_{\mathsf{ex}} \stackrel{\mathsf{tr}}{\Rightarrow}_s (\mathcal{E}_s; \mathcal{P}_s; \Phi_s; \mathcal{C}_s)$ *where*

- $\mathsf{tr} = \nu w_1.out(c_K, w_1) \cdot \nu w_2.out(c_K, w_2) \cdot \nu w_3.out(c_K, w_3) \cdot in(c_B, y) \cdot \nu w_4.out(c_B, w_4)$,
- $\mathcal{P}_s = \{A(a, b)\}$, *and*
- $(\mathcal{E}_s; \Phi_s; \mathcal{C}_s)$ *is the constraint system* Σ_s *defined in Example 12.*

We show that the set of symbolic processes obtained from an intermediate process $(\mathcal{E}; \mathcal{P}; \Phi)$ without else branch exactly captures the set of execution traces of $(\mathcal{E}; \mathcal{P}; \Phi)$ though θ-concretization.

$$\begin{array}{lll}
\text{THEN}_s & (\mathcal{E};\{\texttt{if}\ u=v\ \texttt{then}\ P\ \texttt{else}\ 0\}\uplus\mathcal{P};\Phi;\mathcal{C}) \xrightarrow{\tau}_s (\mathcal{E};\{P\}\uplus\mathcal{P};\Phi;\mathcal{C}\cup\{u =_{\mathsf{E}}^{?} v\}) \\
\text{IN}_s & (\mathcal{E};\{\text{in}(p,x).P\}\uplus\mathcal{P};\Phi;\mathcal{C}) \xrightarrow{\text{in}(p,Y)}_s (\mathcal{E};\{P\{x\mapsto y\}\}\uplus\mathcal{P};\Phi;\mathcal{C}\cup\{Y \rhd^? y\}) \\
& \qquad \text{where } Y, y \text{ are fresh variables, } ar(Y) = |\Phi| \\
\text{OUT-T}_s & (\mathcal{E};\{\text{out}(p,u).P\}\uplus\mathcal{P};\Phi;\mathcal{C}) \xrightarrow{\nu w_n.\text{out}(p,w_n)}_s (\mathcal{E};\{P\}\uplus\mathcal{P};\Phi\cup\{w_n \rhd u\};\mathcal{C}) \\
& \qquad \text{where } w_n \text{ is a variable such that } n = |\Phi|+1
\end{array}$$

u, v, and x are terms of base type whereas p is a channel name.

Figure 3. Symbolic execution of simple processes

Definition 15 (θ-concretization) *Consider the symbolic process $(\mathcal{E}_1;\mathcal{P}_1;\Phi_1;\mathcal{C}_1)$ and let θ be a substitution in $Sol((\mathcal{E}_1;\Phi_1;\mathcal{C}_1))$. The intermediate process $(\mathcal{E}_1;\mathcal{P}_1\lambda_\theta;\Phi_1\lambda_\theta)$ is the* θ-concretization *of $(\mathcal{E}_1;\mathcal{P}_1;\Phi_1;\mathcal{C}_1)$ where λ_θ is the first order solution of $(\mathcal{E}_1;\Phi_1;\mathcal{C}_1)$ associated to θ.*

We now show soundness of $\xrightarrow{\alpha_s}_s$ w.r.t. $\xrightarrow{\alpha}_i$: whenever this relation holds between two symbolic processes, the relation in the intermediate semantics holds for each θ-concretization. Actually, we need such a result for the more detailed notion of annotated traces (see page 7): the label τ of the rules THEN_s and THEN_i is replaced by test_p where p is the identity of the process, i.e. the name of its channel.

Proposition 2 (soundness) *Let $(\mathcal{E}_1;\mathcal{P}_1;\Phi_1;\mathcal{C}_1)$, $(\mathcal{E}_2;\mathcal{P}_2;\Phi_2;\mathcal{C}_2)$ be two symbolic processes such that*

- *$(\mathcal{E}_1;\mathcal{P}_1;\Phi_1;\mathcal{C}_1) \xrightarrow{\alpha_s}_s (\mathcal{E}_2;\mathcal{P}_2;\Phi_2;\mathcal{C}_2)$, and*
- *$\theta_2 \in Sol((\mathcal{E}_2;\Phi_2;\mathcal{C}_2))$.*

Let $\theta_1 = \theta_2|_{\mathsf{var}^2(\mathcal{C}_1)}$. We have that:

1. *$\theta_1 \in Sol((\mathcal{E}_1;\Phi_1;\mathcal{C}_1))$, and*
2. *$(\mathcal{E}_1;\mathcal{P}'_1;\Phi'_1) \xrightarrow{\alpha_s\theta_2}_i (\mathcal{E}_2;\mathcal{P}'_2;\Phi'_2)$ where $(\mathcal{E}_1;\mathcal{P}'_1;\Phi'_1)$ (resp. $(\mathcal{E}_2;\mathcal{P}'_2;\Phi'_2)$) is the θ_1-concretization (resp. θ_2) of $(\mathcal{E}_1;\mathcal{P}_1;\Phi_1;\mathcal{C}_1)$ (resp. $(\mathcal{E}_2;\mathcal{P}_2;\Phi_2;\mathcal{C}_2)$).*

We also show completeness of the symbolic semantics w.r.t. the intermediate one: each time a θ-concretization of a symbolic process reduces to another intermediate process, the symbolic process also reduces to a corresponding symbolic process.

Proposition 3 (completeness) *Let $(\mathcal{E}_1;\mathcal{P}_1;\Phi_1;\mathcal{C}_1)$ be a symbolic process, $(\mathcal{E}_1;\mathcal{P}'_1;\Phi'_1)$ its θ_1-concretization where $\theta_1 \in Sol((\mathcal{E}_1;\Phi_1;\mathcal{C}_1))$. Let $(\mathcal{E};\mathcal{P};\Phi)$ be an intermediate process such that $(\mathcal{E}_1;\mathcal{P}'_1;\Phi'_1) \xrightarrow{\alpha}_i (\mathcal{E};\mathcal{P};\Phi)$. There exist a symbolic process $(\mathcal{E}_2;\mathcal{P}_2;\Phi_2;\mathcal{C}_2)$ and θ_2 such that:*

1. *$(\mathcal{E}_1;\mathcal{P}_1;\Phi_1;\mathcal{C}_1) \xrightarrow{\alpha_s}_s (\mathcal{E}_2;\mathcal{P}_2;\Phi_2;\mathcal{C}_2)$;*
2. *$\theta_2 \in Sol((\mathcal{E}_2;\Phi_2;\mathcal{C}_2))$;*
3. *the process $(\mathcal{E};\mathcal{P};\Phi)$ is the θ_2-concretization of $(\mathcal{E}_2;\mathcal{P}_2;\Phi_2;\mathcal{C}_2)$; and*
4. *$\alpha_s\theta_2 = \alpha$.*

6.3 Symbolic equivalence

Definition 16 (symbolic trace equivalence) *Let A be a simple process without else branch nor replication. We define the set of its symbolic traces as follows:*

$$\mathit{trace}_s(A) = \{(\mathsf{tr},\Sigma) \mid A_s \stackrel{\mathsf{tr}}{\Rightarrow}_s (\mathcal{E}';\mathcal{P}';\Phi';\mathcal{C}') \text{ and } \Sigma = (\mathcal{E}';\Phi';\mathcal{C}') \text{ satisfiable.}\}$$

Let A and B be two simple processes. They are in symbolic trace equivalence *if for every $(\mathsf{tr},\Sigma) \in \mathit{trace}_s(A)$ there exists $(\mathsf{tr}',\Sigma') \in \mathit{trace}_s(B)$ such that $\mathsf{tr} = \mathsf{tr}'$ and Σ, Σ' are S-equivalent (and conversely).*

We show that symbolic trace equivalence exactly captures trace equivalence.

Proposition 4 *Let $A = (\mathcal{E};\mathcal{P}_A;\Phi_A)$ and $B = (\mathcal{E};\mathcal{P}_B;\Phi_B)$ be two intermediate processes derived from simple processes without else branch nor replication. We have that A and B are in annotated trace equivalence if, and only if, they are in annotated symbolic trace equivalence.*

The proof relies on the fact that, when $A \approx_t B$, execution traces can be grouped in the same way for A and B, forming symbolic traces with S-equivalent constraint systems.

The following proposition is an immediate consequence of Proposition 1 and Proposition 4.

Proposition 5 *Let A and B be two simple processes without else branch nor replication: $A \approx_t B$ if, and only if A and B are in annotated symbolic trace equivalence.*

Example 16 *Relying on our technique, we can now prove that the two following processes P_{ex} and P'_{ex} are not in observational equivalence:*

- $P_{\mathsf{ex}} = \nu\tilde{n}.[\,A(a,b) \mid B'(b,a) \mid K(a,a',b)\,]$, *and*
- $P'_{\mathsf{ex}} = \nu\tilde{n}.[\,A(a',b) \mid B'(b,a') \mid K(a,a',b)\,]$.

Continuing Example 15, we have that $(\mathsf{tr}, \Sigma_s) \in \mathsf{trace}_s(P^s_{\mathsf{ex}})$ and Σ_s satisfiable (see Example 13). The only constraint system reachable from

$$P'^s_{\mathsf{ex}} = (\mathcal{E}; \{A(a',b),\, B'(b,a'),\, K(a,a',b)\}; \emptyset; \emptyset)$$

by the sequence tr is Σ'_s as defined in Example 14. We have seen that Σ'_s is not in S-equivalence with Σ_s. This allows us to conclude that the simple processes P_{ex} and P'_{ex} are not in symbolic trace equivalence, and thanks to Proposition 5, Theorem 1 and Theorem 2, we conclude that $P_{\mathsf{ex}} \not\approx P'_{\mathsf{ex}}$.

Similarly, our techniques allow to prove that two processes are in observational equivalence. An example is provided in Appendix E.

6.4 Decidability result

It remains to show how to decide symbolic trace equivalence. We mainly rely on the result of [7] that ensures that checking whether two constraints systems are S-equivalent is NP-complete, for the class of convergent subterm theories.

An equational theory E is a *convergent subterm theory* if it is generated by a convergent rewriting system $\mathcal{R}$ such that any rule $l \to r \in \mathcal{R}$ satisfies that either r is a strict subterm of l or r is a closed term in normal form w.r.t. $\mathcal{R}$. The equational theory presented in Example 3 is a convergent subterm theory. Many other examples can be found e.g. in [1].

Now, we are able to state our main result.

Theorem 3 *Let E be a subterm convergent equational theory. Let A and B be two simple processes without else branch nor replication. The problem whether A and B are observationally equivalent is co-NP-complete.*

The decidability of observational equivalence follows from Proposition 5 since there are a finite number of symbolic traces and non S-equivalence of constraint systems is decidable [7]. Actually, since we consider annotated trace, we have that for any simple process P and any annotated trace tr, there is at most one Σ such that $(\mathsf{tr}, \Sigma) \in \mathsf{trace}_s(P)$. We show that two simple processes A and B without else branch nor replication are in trace equivalence if, and only if, for any annotated trace $(\mathsf{tr}, \Sigma) \in \mathsf{trace}_s(A)$, there exists a (unique) annotated trace $(\mathsf{tr}, \Sigma') \in \mathsf{trace}_s(B)$ such that Σ and Σ' are S-equivalent. We show this result in two steps: we go from applied pi to the intermediate calculus (see Proposition 1) and then we go from intermediate calculus to our symbolic calculus (see Proposition 4). Then the NP-TIME decision procedure for non observational equivalence works as follows:

- Guess a symbolic (annotated) trace tr;
- Compute (in polynomial time) Σ and Σ' such that $(\mathsf{tr}, \Sigma) \in \mathsf{trace}_s(A)$ and $(\mathsf{tr}, \Sigma') \in \mathsf{trace}_s(B)$;
- check whether Σ and Σ' are not S-equivalent.

Due to [7], we know that the last step can be done in NP-TIME for convergent subterm theories thus we deduce that the overall procedure is NP-TIME. NP-hardness is obtained using the usual encoding [22].

7 Conclusion

In this paper, we consider the class of *determinate* processes and we show that observational equivalence actually coincides with trace equivalence, a notion simpler to reason with. We exhibit a large class of processes that are determinate and we show how to reduce the decidability of trace equivalence to deciding an equivalence relation introduced by M. Baudet. Altogether, this yields the first decidability result of observational equivalence for a general class of processes.

As future work, it would be interesting to extend this class of processes in different ways. For example, we would like to extend our decision result to else branches. This would require adding disequality tests in set of constraints and adapt the procedure of [7] accordingly. Moreover, some protocols such as e-voting protocols are divided in several phases. It does not seem difficult to add a "phase" operator to the applied pi-calculus and obtain a corresponding decision result for observational equivalence. It would be also interesting to consider larger classes of equational theories such as those considered for e-voting protocols [15].

Our class of simple processes is close to the fragment of processes considered in [13] for proving cryptographic indistinguishability using observational equivalence. However, the fragment of [13] does not enjoy the determinacy property (since it was not designed for it). We plan to extend their result to our class of simple processes, yielding to a decision technique for proving indistinguishability in cryptographic models.

References

[1] M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. *Theoretical Computer Science*, 387(1-2):2–32, 2006.

[2] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. 28th Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115. ACM Press, 2001.

[3] M. Abadi and C. Fournet. Private authentication. *Theoretical Computer Science*, 322(3):427–476, 2004.

[4] M. Abadi and A. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Proc. 4th Conference on Computer and Communications Security (CCS'97)*, pages 36–47. ACM Press, 1997.

[5] R. Amadio, D. Lugiez, and V. Vanackère. On the symbolic reduction of processes with cryptographic functions. *Theoretical Computer Science*, 290:695–740, 2002.

[6] A. Armando et al. The AVISPA Tool for the automated validation of internet security protocols and applications. In *Proc. 17th Int. Conference on Computer Aided Verification (CAV'05)*, volume 3576 of *LNCS*, pages 281–285. Springer, 2005.

[7] M. Baudet. Deciding security of protocols against off-line guessing attacks. In *Proc. 12th Conference on Computer and Communications Security (CCS'05)*, pages 16–25. ACM Press, 2005.

[8] M. Baudet. *Sécurité des protocoles cryptographiques : aspects logiques et calculatoires*. Phd thesis, École Normale Supérieure de Cachan, France, 2007.

[9] B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *Proc. 14th Computer Security Foundations Workshop (CSFW'01)*, pages 82–96. IEEE Comp. Soc. Press, 2001.

[10] B. Blanchet. Automatic proof of strong secrecy for security protocols. In *Proc. Symposium on Security and Privacy*, pages 86–100. IEEE Comp. Soc. Press, 2004.

[11] B. Blanchet, M. Abadi, and C. Fournet. Automated verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 75(1):3–51, 2008.

[12] J. Clark and J. Jacob. A survey of authentication protocol literature. `http://www.cs.york.ac.uk/~jac/papers/drareviewps.ps`, 1997.

[13] H. Comon-Lundh and V. Cortier. Computational soundness of observational equivalence. In *Proc. 15th Conference on Computer and Communications Security (CCS'08)*, pages 109–118. ACM Press, 2008.

[14] S. Delaune, S. Kremer, and M. D. Ryan. Symbolic bisimulation for the applied pi-calculus. In *Proc. 27th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'07)*, pages 133–145, 2007.

[15] S. Delaune, S. Kremer, and M. D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 2008. To appear.

[16] L. Durante, R. Sisto, and A. Valenzano. Automatic testing equivalence verification of spi calculus specifications. *ACM Transactions on Software Engineering and Methodology*, 12(2):222–284, 2003.

[17] J. Engelfriet. Determinacy implies (observation equivalence = trace equivalence). *Theoretical Computer Science*, 36:21–25, 1985.

[18] H. Hüttel. Deciding framed bisimulation. In *Proc. 4th Int. Workshop on Verification of Infinite State Systems (INFINITY'02)*, pages 1–20, 2002.

[19] P. C. Kanellakis and S. A. Smolka. Ccs expressions, finite state processes, and three problems of equivalence. In ACM, editor, *Proceedings of the second annual ACM symposium on Principles of distributed computing*, pages 228 – 240, 1983.

[20] G. Lowe. Towards a completeness result for model checking of security protocols. In *Proc. 11th Computer Security Foundations Workshop (CSFW'98)*, 1998.

[21] J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. 8th ACM Conference on Computer and Communications Security (CCS'01)*. ACM Press, 2001.

[22] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *Proc. 14th Computer Security Foundations Workshop (CSFW'01)*, pages 174–190. IEEE Comp. Soc. Press, 2001.

A Proofs of Section 3

Trace equivalence implies observational equivalence.

Lemma 1 *Let A and B be two closed extended processes: $A \approx B$ implies $A \approx_t B$.*

Proof. Let $(s, \varphi) \in \mathsf{trace}(A)$ with $bn(s) \cap fn(B) = \emptyset$. By definition of $\mathsf{trace}(A)$ we have that there exists A' such that $A \overset{s}{\Rightarrow} A'$, and $\varphi = \phi(A')$. By relying on the fact that $A \approx B$, we can show by induction on the length of the derivation $A \overset{s}{\Rightarrow} A'$ that there exists B' such that $B \overset{s}{\Rightarrow} B'$ and $A' \approx B'$. Thus, we have that $(s, \phi(B')) \in \mathsf{trace}(B)$ and since $A' \approx B'$ we have that $\phi(A') \sim \phi(B')$. This allows us to conclude that $A \sqsubseteq_t B$. The other direction can be proved in a similar way. □

Observation and trace determinacy coincide.

Lemma 2 *Let A be a closed extended process. The process A is observation determinate if, and only if, it is trace determinate.*

Proof. We prove the two directions separately.
($\Rightarrow$) Let A be a closed extended process that is observation determinate. Let B and B' be two closed extended processes and s be a sequence of actions such that $A \overset{s}{\Rightarrow} B$, $A \overset{s}{\Rightarrow} B'$ and $\phi(B) \sim \phi(B')$. In order to show that A is trace determinate, we have to show that $B \approx_t B'$. Actually, since A is observation determinate, we have that $B \approx B'$ and thanks to Lemma 1, we easily conclude.

($\Leftarrow$) Let A be a closed extended process that is trace determinate. Let B and B' be two closed extended processes and s_1 be a sequence of actions such that $A \overset{s_1}{\Rightarrow} B$, $A \overset{s_1}{\Rightarrow} B'$ and $\phi(B) \sim \phi(B')$. By hypothesis we have that $B \approx_t B'$. In order to show that A is observation determinate, we have to show that $B \approx B'$. To prove this, first we define a relation $\mathcal{R}$ (which depends on B, B' and A) on closed extended processes and then we will show that $\mathcal{R}$ is a labeled bisimulation witnessing $B \approx B'$.

(i) Definition of $\mathcal{R}$. $\tilde{B} \mathcal{R} \tilde{B}'$ if, and only if, there exists s_2 such that $B \overset{s_2}{\Rightarrow} \tilde{B}$, $B' \overset{s_2}{\Rightarrow} \tilde{B}'$, and $\phi(\tilde{B}) \sim \phi(\tilde{B}')$.

(ii) $\mathcal{R}$ is a bisimulation relation witnessing $B \approx B'$. First note that $B \mathcal{R} B'$. Now, we have to show that $\mathcal{R}$ satisfies the three points of the definition of labeled bisimulation (Definition 3). Let $\tilde{B}$ and $\tilde{B}'$ be two closed extended processes such that $\tilde{B} \mathcal{R} \tilde{B}'$. Note that, by definition of $\mathcal{R}$, we have that there exists s_2 such that $B \overset{s_2}{\Rightarrow} \tilde{B}$, $B' \overset{s_2}{\Rightarrow} \tilde{B}'$, and $\phi(\tilde{B}) \sim \phi(\tilde{B}')$.

1. $\phi(\tilde{B}) \sim \phi(\tilde{B}')$. This is an easy consequence of the definition of $\mathcal{R}$.
2. If $\tilde{B} \overset{\tau}{\mapsto} \tilde{A}$ then there exists an extended process $\tilde{A}'$ such that $\tilde{B}' \Rightarrow \tilde{A}'$ and $\tilde{A} \mathcal{R} \tilde{A}'$.

   Let $\tilde{A}' = \tilde{B}'$. We have that $\tilde{B}' \Rightarrow \tilde{A}'$ and $\tilde{A} \mathcal{R} \tilde{A}'$ since $B \overset{s_2}{\Rightarrow} \tilde{B} \overset{\tau}{\mapsto} \tilde{A}$, $B' \overset{s_2}{\Rightarrow} \tilde{B}' = \tilde{A}'$, and $\phi(\tilde{A}) \sim \phi(\tilde{A}')$. This last point is due to the following relations:

   $$\phi(\tilde{A}) = \phi(\tilde{B}) \sim \phi(\tilde{B}') = \phi(\tilde{A}').$$

3. If $\tilde{B} \overset{\ell}{\mapsto} \tilde{A}$ with $bn(\ell) \cap fn(\tilde{B}') = \emptyset$ then $\tilde{B}' \overset{\ell}{\Rightarrow} \tilde{A}'$ and $\tilde{A} \mathcal{R} \tilde{A}'$ for some extended process $\tilde{A}'$.

   Since A is trace determinate, we have that $\tilde{B} \approx_t \tilde{B}'$. We have that $(\ell, \phi(\tilde{A})) \in \mathsf{trace}(\tilde{B})$. Since, $\tilde{B} \sqsubseteq_t \tilde{B}'$ and $bn(\ell) \cap fn(\tilde{B}') = \emptyset$, we deduce that there exists $\tilde{A}'$ such that $\tilde{B}' \overset{\ell}{\Rightarrow} \tilde{A}'$ and $\phi(\tilde{A}) \sim \phi(\tilde{A}')$. We deduce that $\tilde{A} \mathcal{R} \tilde{A}'$ (by the sequence $s_2 \cdot \ell$). This allows us to conclude. □

Theorem 1 *Let A and B be two closed extended processes that are determinate.*

$$A \approx_t B \text{ implies } A \approx B.$$

Proof. Let A and B be two closed extended processes that are determinate, and assume that $A \approx_t B$. We have to show that $A \approx B$. Define $\mathcal{R}$ as in the proof of Lemma 2:

$$A' \mathcal{R} B' \text{ iff there exists } s \text{ such that } A \overset{s}{\Rightarrow} A',\ B \overset{s}{\Rightarrow} B', \text{ and } \phi(A') \sim \phi(B').$$

First note that $A \mathcal{R} B$. We show that $\mathcal{R}$ satisfies the three points of the definition of labeled bisimulation (Definition 3). Let A' and B' be two extended processes such that $A' \mathcal{R} B'$.

1. $\phi(A') \sim \phi(B')$. This is an easy consequence of the definition of $\mathcal{R}$.
2. If $A' \overset{\tau}{\mapsto} A''$ then there exists an extended process B'' such that $B' \Rightarrow B''$ and $A'' \mathcal{R} B''$.

   Let $B'' = B'$. We have that $B' \Rightarrow B''$ and $A'' \mathcal{R} B''$ since $A \overset{s}{\Rightarrow} A''$, $B \overset{s}{\Rightarrow} B''$, and $\phi(A'') \sim \phi(B'')$. Indeed, we have that

   $$\phi(A') = \phi(A'') \sim \phi(B'') = \phi(B').$$

3. If $A' \overset{\ell}{\mapsto} A''$ with $bn(\ell) \cap fn(B') = \emptyset$ then $B' \overset{\ell}{\Rightarrow} B''$ and $A'' \mathcal{R} B''$ for some extended process B''.

   We have that $A \overset{s}{\Rightarrow} A' \overset{\ell}{\mapsto} A''$. Since $A \approx_t B$, we easily deduce that there exist two closed extended processes D' and D'' such that $B \overset{s}{\Rightarrow} D' \overset{\ell}{\Rightarrow} D''$ such that $\phi(A'') \sim \phi(D'')$. Now, since B is determinate and $B \overset{s}{\Rightarrow} B'$ and $B \overset{s}{\Rightarrow} D'$, we have that $B' \approx_t D'$. Hence there exists B'' such that $B' \overset{\ell}{\Rightarrow} B''$ and $\phi(D'') \sim \phi(B'')$. We have that $A \overset{s\cdot\ell}{\Rightarrow} A''$, $B \overset{s\cdot\ell}{\Rightarrow} B''$ and $\phi(A'') \sim \phi(B'')$, i.e. $A'' \mathcal{R} B''$. This allows us to conclude. □

B Proof of Section 4

Before to prove Theorem 2, we introduce some definitions and notations. Let B be a basic process such that $B \in \mathcal{B}(c, \mathcal{V})$. If $B \overset{\tau}{\mapsto} B'$ then it is easy to see that B' is of the form $B' \equiv \text{out}(c, s).B''$ for some term s and some $B'' \in \mathcal{B}(c, \mathcal{V})$. We say that B' is a *derived basic process*.

A simple process built from basic processes that are possibly derived is a *derived simple process*. Each process of the form $!(\nu c, \tilde{n}.\text{out}(p, c).B)$ with $B \in \mathcal{B}(c, \emptyset)$ is called a *replicated process for the role p*. For every $w \in \mathcal{A}^*$, we denote by $s(w)$ the trace obtained from w by erasing all occurrences of τ. By definition, we have that $A \overset{s(w)}{\Rightarrow} B$ when $A \overset{w}{\mapsto} B$.

We say that two processes P_1 and P_2 are *in relation*, denoted by $P_1 \leftrightarrow P_2$, if

- $P_1 \equiv \nu\tilde{n}.(B_1 \mid \cdots \mid B_k \mid R_1 \mid \cdots R_l \mid \sigma)$,
- $P_2 \equiv \nu\tilde{n}.(B'_1 \mid \cdots \mid B'_k \mid R_1 \mid \cdots R_l \mid \sigma)$,

where σ is a sequence of active substitutions, the R_i are replicated processes and the B_i, B'_i are (possibly derived) basic processes such that either $B_i = B'_i$ or $B_i \overset{\tau}{\mapsto} B'_i$ or $B'_i \overset{\tau}{\mapsto} B_i$. It is easy to check that $\leftrightarrow$ is an equivalence relation on derived simple processes.

Theorem 2 *Any simple process is determinate.*

Proof. Let P be (possibly derived) simple process. We show by induction on the length of the trace w that whenever $P \overset{w}{\mapsto} P_1$ and $P \overset{s(w)}{\Rightarrow} P_2$ then $P_1 \leftrightarrow P_2$. This implies $P_1 \approx P_2$ thus this would prove that P is determinate.

Assume that $P \overset{w}{\mapsto} P_1 \overset{\alpha}{\mapsto} P'_1$ and $P \overset{s(w.\alpha)}{\Rightarrow} P'_2$ where $\alpha \in \mathcal{A}$. We have $P \overset{w'}{\mapsto} P_2 \overset{\alpha}{\mapsto} P''_2 \overset{w''}{\mapsto} P'_2$ for some w and w' such that $s(w) = s(w')$ and $w'' \in \{\tau\}^*$ or possibly $P'_2 = P_2$ in case $\alpha = \tau$. By induction hypothesis, we have $P_1 \leftrightarrow P_2$ thus we have that

- $P_1 \equiv \nu\tilde{n}.(B_1 \mid \cdots \mid B_k \mid R_1 \mid \cdots R_l \mid \sigma)$,
- $P_2 \equiv \nu\tilde{n}.(B'_1 \mid \cdots \mid B'_k \mid R_1 \mid \cdots R_l \mid \sigma)$,

where the R_i are replicated processes and the B_i, B'_i are (possibly derived) basic processes such that either $B_i = B'_i$ or $B_i \overset{\tau}{\mapsto} B'_i$ or $B'_i \overset{\tau}{\mapsto} B_i$. Let us show that $P'_1 \leftrightarrow P'_2$ by case analysis on α.

Case $\alpha = \tau$. Then $P_1 \overset{\tau}{\mapsto} P'_1$ thus $P_1 \leftrightarrow P'_1$. We have also that $P_2 \overset{w}{\mapsto} P'_2$ for $w \in \{\tau\}^*$, thus $P_2 \leftrightarrow P'_2$. Since $\leftrightarrow$ is an equivalence relation, we deduce $P'_1 \leftrightarrow P'_2$.

Case $\alpha = \nu c.\text{out}(p, c)$ *where c is a name channel.* It must be the case that one process R_i is a replicated process for the role p and has sent a new channel names for a fresh instance of the role p. Let $R_i \equiv !(\nu c, \tilde{m} \cdot \text{out}(p, c).B)$ with $B \in \mathcal{B}(c, \emptyset)$. We must have $P'_1 \equiv \nu\tilde{n}, \tilde{m}.(B_1 \mid \cdots \mid B_k \mid B \mid R_1 \mid \cdots R_l \mid \sigma)$. Similarly, since $P_2 \overset{\alpha}{\rightarrow} P''_2$, we must have $P''_2 \equiv \nu\tilde{n}, \tilde{m}.(B'_1 \mid \cdots \mid B'_k \mid B \mid R_1 \mid \cdots R_l \mid \sigma)$. Thus $P'_1 \leftrightarrow P''_2$. Since $P''_2 \rightarrow^* P'_2$, we also have $P''_2 \leftrightarrow P'_2$ thus $P'_1 \leftrightarrow P'_2$.

Case $\alpha = \nu x.\text{out}(c, x)$ *where x is a variable.* Let B_i the (derived) basic process such that $B_i = \text{out}(c, s).B''_i$. Such a B_i is unique and we must have $B'_i = B_i$. Indeed the other cases, i.e. $B'_i \overset{\tau}{\mapsto} B_i$ and $B_i \overset{\tau}{\mapsto} B'_i$ are impossible since B_i and B'_i are ready to emit. We deduce that $P'_1 \equiv \nu\tilde{n}.(B_1 \mid \cdots \mid B''_i \mid \cdots \mid B_k \mid R_1 \mid \cdots R_l \mid \sigma \mid \{^s/_x\})$ and $P''_2 \equiv \nu\tilde{n}.(B'_1 \mid \cdots \mid B''_i \mid \cdots \mid B'_k \mid B \mid R_1 \mid \cdots R_l \mid \sigma \mid \{^s/_x\})$. We deduce similarly that $P'_1 \leftrightarrow P'_2$.

Case $\alpha = \text{in}(c, M)$. Let B_i the basic process such that $B_i = \text{in}(c, x).B''_i$. Such a B_i is unique and we must have $B'_i = B_i$. Since B'_i and B_i get the same input, they both evolved into $B''_i\{^M/_x\}$ thus we have that $P'_1 \leftrightarrow P'_2$. □

C Proofs of Section 5

Proposition 1 is a consequence of Propositions 6 and 7 stated and proved below.

Despite the difference between the two semantics (the original one and the intermediate one), it can be shown that the two notions of trace equivalence coincide. The following proposition is a direct consequence of results established in [14].

Proposition 6 *Let A and B be two simple processes s (of the original applied pi) without replication nor else branch. Let $\tilde{A} = (\mathcal{E}_1; \mathcal{P}_1; \Phi_1)$ (resp. $\tilde{B} = (\mathcal{E}_2; \mathcal{P}_2; \Phi_2)$) be an intermediate process associated to A (resp. B). Moreover we assume that $\mathcal{E}_1 = \mathcal{E}_2$. We have that $A \approx_t B$ (in the original applied pi calculus semantics) if, and only if, the two processes $\tilde{A}$ and $\tilde{B}$ are intermediate trace equivalent.*

Actually, we want to obtain such a result but for the more detailed notion of annotated trace. For this, it remains to prove the following proposition.

Proposition 7 *Let A and B be two simple processes without else branch nor replication. Let $\tilde{A} = (\mathcal{E}; \mathcal{P}_A; \Phi_A)$ and $\tilde{B} = (\mathcal{E}; \mathcal{P}_B; \Phi_B)$ be the two associated intermediate processes. The processes $\tilde{A}$ and $\tilde{B}$ are trace equivalent if, and only if, $\tilde{A}$ and $\tilde{B}$ are annotated trace equivalent.*

Proof. We show the two directions separately.

($\Leftarrow$) Let $\tilde{A}$ and $\tilde{B}$ be two intermediate processes that are annotated trace equivalent. Then they are of course trace equivalent.

($\Rightarrow$) Assume that $\tilde{A}$ and $\tilde{B}$ are in trace equivalence. Now, it remains to show that they are in trace equivalence w.r.t. annotated traces that seems at first sight a stronger notion of trace equivalence.

Let w be an annotated trace such that $\tilde{A} \overset{w}{\Rightarrow}_i \tilde{A}'$. We show that there exists $\tilde{B}'$ such that $\tilde{B} \overset{w}{\Rightarrow}_i \tilde{B}'$ with the same annotated trace w and $\phi(\tilde{B}) \sim \phi(\tilde{B}')$. First, we force $\tilde{A}'$ to emit as much as possible, i.e. $\tilde{A}' \overset{w'}{\Rightarrow}_i \tilde{A}''$ such that w' only consists in actions of the form $\nu w.out(c, w)$ and $\tilde{A}''$ cannot perform any out action of a message of base type. By trace equivalence, we have that there exists $\tilde{B}''$ such that $\tilde{B} \overset{s(\overline{w \cdot w'})}{\Rightarrow}_i \tilde{B}''$ and $\phi(\tilde{A}'') \sim \phi(\tilde{B}'')$ (Definition of $s(w)$ is given page 14 and Definition of $\overline{w}$ can be found on page 7).

Let $\mathcal{V}_B$ be the set of annotated traces such that $\tilde{B} \overset{v_B}{\Rightarrow}_i \tilde{B}''$. Note that $\mathcal{V}_B \neq \emptyset$ and every annotated trace v_B in $\mathcal{V}_B$ is such that $v_B = w_B \cdot w'_B$, $s(\overline{w_B}) = s(\overline{w})$ for some w_B and w'_B. Let v_B be an annotated trace in $\mathcal{V}_B$ of maximal common prefix with w. We show that w is a prefix of v_B.

Indeed, if not, we consider the first time where the two annotated traces w and v_B differ. We have that $w = u{\cdot}a{\cdot}u'$ and $v_B = u{\cdot}b{\cdot}v'$ with $a \neq b$. Clearly a and b can not be both observable actions. One of them (at least) is an annotation. Assume w.l.o.g. that $a = \mathsf{test}_p$ for some p. We consider the first occurrence (after the action a) of the process identified by p in the trace $w{\cdot}w'$. Note that such an occurrence necessarily exists and is of the form $\nu w.out(p, w)$ since each test is followed by an emission and we have force $\tilde{A}$ to emit. Thus, v_B must contain an action of the form $\nu w.out(p, w)$, which enforces that test_p also occur in v_B. Thus, we can remove this action test_p to insert it just before the occurrence of b. We obtain another trace that is also in $\mathcal{V}_B$ and which contradicts the maximality of v_B.

Hence, we have that w is a prefix of v_B. Let $\tilde{B}'$ be the process obtained after the execution of w in the derivation $\tilde{B} \overset{v_B}{\Rightarrow}_i \tilde{B}''$. We have that $\tilde{B} \overset{w}{\Rightarrow}_i \tilde{B}'$ by the annotated trace w and $\phi(\tilde{A}') \sim \phi(\tilde{B}')$ since $\phi(\tilde{A}'') \sim \phi(\tilde{B}'')$. □

D Proofs of Section 6

Proposition 2 (soundness) *Let $(\mathcal{E}_1; \mathcal{P}_1; \Phi_1; \mathcal{C}_1)$, $(\mathcal{E}_2; \mathcal{P}_2; \Phi_2; \mathcal{C}_2)$ be two symbolic processes such that*

- $(\mathcal{E}_1; \mathcal{P}_1; \Phi_1; \mathcal{C}_1) \xrightarrow{\alpha_s}_s (\mathcal{E}_2; \mathcal{P}_2; \Phi_2; \mathcal{C}_2)$, *and*
- $\theta_2 \in Sol((\mathcal{E}_2; \Phi_2; \mathcal{C}_2))$.

Let $\theta_1 = \theta_2|_{\mathsf{var}^2(\mathcal{C}_1)}$. We have that:

1. $\theta_1 \in Sol((\mathcal{E}_1; \Phi_1; \mathcal{C}_1))$, *and*
2. $(\mathcal{E}_1; \mathcal{P}'_1; \Phi'_1) \xrightarrow{\alpha_s\theta_2}_i (\mathcal{E}_2; \mathcal{P}'_2; \Phi'_2)$ *where $(\mathcal{E}_1; \mathcal{P}'_1; \Phi'_1)$ (resp. $(\mathcal{E}_2; \mathcal{P}'_2; \Phi'_2)$ is the θ_1-concretization (resp. θ_2) of $(\mathcal{E}_1; \mathcal{P}_1; \Phi_1; \mathcal{C}_1)$ (resp. $(\mathcal{E}_2; \mathcal{P}_2; \Phi_2; \mathcal{C}_2)$).*

Proof. We prove this result by case analysis on the rule involved in the reduction step:

$$(\mathcal{E}_1; \mathcal{P}_1; \Phi_1; \mathcal{C}_1) \xrightarrow{\alpha_s}_s (\mathcal{E}_2; \mathcal{P}_2; \Phi_2; \mathcal{C}_2).$$

Case THEN$_s$**.** In such a case there exist u, v, P, $\mathcal{P}$ and c such that $\mathcal{P}_1 = \{\mathtt{if}\ u = v\ \mathtt{then}\ P\ \mathtt{else}\ 0\} \uplus \mathcal{P}$, $\mathcal{C}_2 = \mathcal{C}_1 \cup \{u =^?_{\mathsf{E}} v\}$, $\mathcal{P}_2 = \{P\} \uplus \mathcal{P}$, $\mathcal{E}_1 = \mathcal{E}_2$, $\Phi_1 = \Phi_2$, and $\alpha_s = \mathsf{test}_c$ where c is the channel name used in the process P.

1. Since $\mathsf{var}^2(\mathcal{C}_1) = \mathsf{var}^2(\mathcal{C}_2)$, we have $\theta_1 = \theta_2$, and it is now easy to see that $\theta_1 \in Sol((\mathcal{E}_1; \Phi_1; \mathcal{C}_1))$.
2. Since $\theta_2 \in Sol((\mathcal{E}_2; \Phi_2; \mathcal{C}_1 \cup \{u =^?_{\mathsf{E}} v\}))$, $\theta_2 = \theta_1$, $\Phi_2 = \Phi_1$, we have $\lambda_{\theta_1} = \lambda_{\theta_2}$ and thus $u\lambda_{\theta_1} =_{\mathsf{E}} v\lambda_{\theta_1}$. Hence we have that

$$(\mathcal{E}_1; \{\mathtt{if}\ u\lambda_{\theta_1} = v\lambda_{\theta_1}\ \mathtt{then}\ P\lambda_{\theta_1}\ \mathtt{else}\ 0\} \uplus \mathcal{P}\lambda_{\theta_1}; \Phi_1\lambda_{\theta_1}) \xrightarrow{\mathsf{test}_c}_i (\mathcal{E}_1; \{P\lambda_{\theta_2}\} \uplus \mathcal{P}\lambda_{\theta_2}; \Phi_2\lambda_{\theta_2}),$$

   i.e. $(\mathcal{E}_1; \mathcal{P}'_1; \Phi'_1) \xrightarrow{\alpha_s\theta_2}_i (\mathcal{E}_2; \mathcal{P}'_2; \Phi'_2)$ where $(\mathcal{E}_1; \mathcal{P}'_1; \Phi'_1)$ (resp. $(\mathcal{E}_2; \mathcal{P}'_2; \Phi'_2)$ is the θ_1-concretization (resp. θ_2-concretization) of $(\mathcal{E}_1; \mathcal{P}_1; \Phi_1; \mathcal{C}_1)$ (resp. $(\mathcal{E}_2; \mathcal{P}_2; \Phi_2; \mathcal{C}_2)$).

Case IN$_s$**.** In such a case, there exist p, x, P, $\mathcal{P}$ and fresh variables y and Y with $ar(Y) = |\Phi_1|$ and such that $\alpha_s = \mathrm{in}(p, Y)$, $\mathcal{P}_1 = \{\mathrm{in}(p, x).P\} \uplus \mathcal{P}$, $\mathcal{P}_2 = \{P\{x \mapsto y\}\} \uplus \mathcal{P}$, $\mathcal{C}_2 = \mathcal{C}_1 \cup \{Y \rhd^? y\}$, $\mathcal{E}_1 = \mathcal{E}_2$ and $\Phi_1 = \Phi_2$.

1. We have that $\mathsf{var}^2(\mathcal{C}_1) = \mathsf{var}^2(\mathcal{C}_2) \smallsetminus \{Y\}$, $\mathsf{var}^1(\mathcal{C}_1) = \mathsf{var}^1(\mathcal{C}_2) \smallsetminus \{y\}$ and $\Phi_1 = \Phi_2$. Thus, we have that $\lambda_{\theta_1} = \lambda_{\theta_2}|_{\mathsf{var}^1(\mathcal{C}_1)}$ and we easily deduce that $\theta_1 \in Sol((\mathcal{E}_1; \Phi_1; \mathcal{C}_1))$.

2. Let $M = Y\theta_2$ and $u = y\lambda_{\theta_2}$. We have that $fn(M) \cap \mathcal{E}_1 = \emptyset$ since $\theta_2 \in Sol((\mathcal{E}_2;\Phi_2;\mathcal{C}_2))$ and $\mathcal{E}_2 = \mathcal{E}_1$. We have also that $u = y\lambda_{\theta_2} = M(\Phi_1\lambda_{\theta_1}) = M\Phi'_1$. Lastly, by definition of a solution, we have that $fv(M) \subseteq \mathrm{dom}(\Phi'_1)$. Hence, we have that

$$(\mathcal{E}_1;\{\mathrm{in}(p,x).P\lambda_{\theta_1}\} \uplus \mathcal{P}\lambda_{\theta_1};\Phi_1\lambda_{\theta_1}) \xrightarrow{in(p,M)}_i (\mathcal{E}_1;\{P\lambda_{\theta_1}\{x \mapsto u\}\} \uplus \mathcal{P}\lambda_{\theta_1};\Phi_2\lambda_{\theta_1}),$$

i.e. $(\mathcal{E}_1;\mathcal{P}_1\lambda_{\theta_1};\Phi_1\lambda_{\theta_1}) \xrightarrow{\alpha_s\theta_2} (\mathcal{E}_2;(P\{x \mapsto y\})\lambda_{\theta_2} \uplus \mathcal{P}\lambda_{\theta_2};\Phi_2\lambda_{\theta_2})$ since $\mathcal{E}_2 = \mathcal{E}_1$, $\Phi_2 = \Phi_1$, $\mathsf{var}^1(\Phi_1) = \mathsf{var}^1(\Phi_2)$ and $\lambda_{\theta_2} = \lambda_{\theta_1} \cup \{y \mapsto u\}$. Hence, we have that $(\mathcal{E}_1;\mathcal{P}'_1;\Phi'_1) \xrightarrow{\alpha_s\theta_2} (\mathcal{E}_2;\mathcal{P}'_2;\Phi'_2)$ where $(\mathcal{E}_1;\mathcal{P}'_1;\Phi'_1)$ (resp. $(\mathcal{E}_2;\mathcal{P}'_2;\Phi'_2)$ is the θ_1-concretization (resp. θ_2) of $(\mathcal{E}_1;\mathcal{P}_1;\Phi_1;\mathcal{C}_1)$ (resp. $(\mathcal{E}_2;\mathcal{P}_2;\Phi_2;\mathcal{C}_2)$).

Case OUT-T$_s$. In such a case, there exist p, u, P, $\mathcal{P}$ and w_l such that $l = |\Phi_1| + 1$, $\alpha_s = \nu w_l.out(p,w_l)$, $\mathcal{P}_1 = \{\mathrm{out}(p,u).P\} \uplus \mathcal{P}$, $\mathcal{P}_2 = \{P\} \uplus \mathcal{P}$, $\mathcal{C}_2 = \mathcal{C}_1$, $\mathcal{E}_2 = \mathcal{E}_1$, and $\Phi_2 = \Phi_1 \cup \{w_l \rhd u\}$.

1. We have that $\theta_1 = \theta_2$ and w_l does not occur in θ_1. Hence we have that $\lambda_{\theta_1} = \lambda_{\theta_2}$. This allows us to deduce that $\theta_1 \in Sol((\mathcal{E}_1;\Phi_1;\mathcal{C}_1))$.

2. Let $v = u\lambda_{\theta_1}$. We have that:

$$(\mathcal{E}_1;\{\mathrm{out}(p,u\lambda_{\theta_1}).P\lambda_{\theta_1}\} \uplus \mathcal{P}\lambda_{\theta_1};\Phi_1\lambda_{\theta_1}) \xrightarrow{\nu w_l.out(p,w_l)}_i (\mathcal{E}_1;\{P\lambda_{\theta_1}\} \uplus \mathcal{P}\lambda_{\theta_1};\Phi_1\lambda_{\theta_1} \cup \{w_l \rhd u\lambda_{\theta_1}\}),$$

i.e. $(\mathcal{E}_1;\mathcal{P}'_1;\Phi'_1) \xrightarrow{\nu w_l.out(p,w_l)}_i (\mathcal{E}_2;\mathcal{P}_2\lambda_{\theta_1};\Phi_2\lambda_{\theta_1})$, and thus $(\mathcal{E}_1;\mathcal{P}'_1;\Phi'_1) \xrightarrow{\nu w_l.out(p,w_l)}_i (\mathcal{E}_2;\mathcal{P}'_2;\Phi'_2)$ since $\lambda_{\theta_1} = \lambda_{\theta_2}$. ☐

Proposition 3 (completeness) *Let $(\mathcal{E}_1;\mathcal{P}_1;\Phi_1;\mathcal{C}_1)$ be a symbolic process, $(\mathcal{E}_1;\mathcal{P}'_1;\Phi'_1)$ its θ_1-concretization where $\theta_1 \in Sol((\mathcal{E}_1;\Phi_1;\mathcal{C}_1))$. Let $(\mathcal{E};\mathcal{P};\Phi)$ be an intermediate process such that $(\mathcal{E}_1;\mathcal{P}'_1;\Phi'_1) \xrightarrow{\alpha}_i (\mathcal{E};\mathcal{P};\Phi)$. There exist a symbolic process $(\mathcal{E}_2;\mathcal{P}_2;\Phi_2;\mathcal{C}_2)$ and θ_2 such that:*

1. $(\mathcal{E}_1;\mathcal{P}_1;\Phi_1;\mathcal{C}_1) \xrightarrow{\alpha_s}_s (\mathcal{E}_2;\mathcal{P}_2;\Phi_2;\mathcal{C}_2)$;

2. $\theta_2 \in Sol((\mathcal{E}_2;\Phi_2;\mathcal{C}_2))$;

3. *the process $(\mathcal{E};\mathcal{P};\Phi)$ is the θ_2-concretization of $(\mathcal{E}_2;\mathcal{P}_2;\Phi_2;\mathcal{C}_2)$; and*

4. $\alpha_s\theta_2 = \alpha$.

Proof. We prove this result by case analysis on the rule involved in the reduction step:

$$(\mathcal{E}_1;\mathcal{P}'_1;\Phi'_1) \xrightarrow{\alpha}_i (\mathcal{E};\mathcal{P};\Phi).$$

Case THEN$_i$. In such a case we have that $\mathcal{E} = \mathcal{E}_1$ and there exist u', v', P', $\mathcal{P}'$, and c such that $u' =_{\mathsf{E}} v'$, $\mathcal{P}'_1 = \{\mathtt{if}\ u' = v'\ \mathtt{then}\ P'\ \mathtt{else}\ 0\} \uplus \mathcal{P}'$, $\mathcal{P} = \{P'\} \uplus \mathcal{P}'$, $\Phi = \Phi'_1$, and $\alpha = \mathsf{test}_c$ where c is the channel name used in P'. Since $(\mathcal{E}_1;\mathcal{P}'_1;\Phi'_1)$ is the θ_1-concretization of $(\mathcal{E}_1;\mathcal{P}_1;\Phi_1;\mathcal{C}_1)$, we have that $\mathcal{P}'_1 = \mathcal{P}_1\lambda_{\theta_1}$ and $\Phi'_1 = \Phi_1\lambda_{\theta_1}$. Hence we deduce that that there exist u, v, P and $\mathcal{P}_0$ such that $\mathcal{P}_1 = \{\mathtt{if}\ u = v\ \mathtt{then}\ P\ \mathtt{else}\ 0\} \uplus \mathcal{P}_0$, and thus we have that $u\lambda_{\theta_1} = u'$, $v\lambda_{\theta_1} = v'$, $P\lambda_{\theta_1} = P'$ and $\mathcal{P}_0\lambda_{\theta_1} = \mathcal{P}'$. Let $\mathcal{E}_2 = \mathcal{E}_1$, $\mathcal{P}_2 = \{P\} \uplus \mathcal{P}_0$, $\Phi_2 = \Phi_1$, $\mathcal{C}_2 = \mathcal{C}_1 \cup \{u = v\}$, $\alpha_s = \mathsf{test}_c$ and $\theta_2 = \theta_1$. We have that:

1. $(\mathcal{E}_1;\mathcal{P}_1;\Phi_1;\mathcal{C}_1) \xrightarrow{\alpha_s}_s (\mathcal{E}_2;\mathcal{P}_2;\Phi_2;\mathcal{C}_2)$. Indeed, we have that

$$(\mathcal{E}_1;\{\mathtt{if}\ u = v\ \mathtt{then}\ P\ \mathtt{else}\ 0\} \uplus \mathcal{P}_0;\Phi_1;\mathcal{C}_1) \xrightarrow{\alpha_s}_s (\mathcal{E}_1;\{P\} \uplus \mathcal{P}_0;\Phi_1;\mathcal{C}_1 \cup \{u = v\}).$$

2. We have that $\mathsf{var}^2(\mathcal{C}_2) = \mathsf{var}^2(\mathcal{C}_1)$, $\mathcal{E}_2 = \mathcal{E}_1$ and $\Phi_2 = \Phi_1$. To check that θ_2 is a solution, it remains to show that λ_{θ_2} satisfies the constraints in $\mathcal{C}_2 = \mathcal{C}_1 \cup \{u = v\}$. Actually we have that $\lambda_{\theta_2} = \lambda_{\theta_1}$. Moreover, we have that $u\lambda_{\theta_1} =_{\mathsf{E}} v\lambda_{\theta_1}$, thus we deduce that $\theta_2 \in Sol_{\mathsf{E}}((\mathcal{E}_2;\Phi_2;\mathcal{C}_2))$.

3. We have that

$$\begin{aligned}(\mathcal{E}_2;\mathcal{P}_2\lambda_{\theta_2};\Phi_2\lambda_{\theta_2}) &= (\mathcal{E}_1;(\{P\} \uplus \mathcal{P}_0)\lambda_{\theta_1};\Phi_1\lambda_{\theta_1})\\ &= (\mathcal{E};\{P'\} \uplus \mathcal{P}';\Phi'_1)\\ &= (\mathcal{E};\mathcal{P};\Phi).\end{aligned}$$

4. We have that $\alpha_s\theta_2 = \alpha_s = \alpha = \mathsf{test}_c$.

Case IN$_i$. In such a case we have that $\mathcal{E} = \mathcal{E}_1$, $\Phi = \Phi'_1$ and there exist p, x, P', $\mathcal{P}'$, M and u such that $\mathcal{P}'_1 = \{\mathrm{in}(p,x).P'\} \uplus \mathcal{P}'$, $\mathcal{P} = \{P'\{x \mapsto u\}\} \uplus \mathcal{P}'$, $M\Phi'_1 = u$, $fv(M) \subseteq \mathrm{dom}(\Phi'_1)$, $fn(M) \cap \mathcal{E}_1 = \emptyset$, and $\alpha = in(p,M)$. Since $(\mathcal{E}_1;\mathcal{P}'_1;\Phi'_1)$ is the θ_1-concretization of $(\mathcal{E}_1;\mathcal{P}_1;\Phi_1;\mathcal{C}_1)$, we have that $\mathcal{P}'_1 = \mathcal{P}_1\lambda_{\theta_1}$, and $\Phi'_1 = \Phi\lambda_{\theta_1}$. Hence, we deduce that there exist P, $\mathcal{P}_0$ such that $\mathcal{P}_1 = \{\mathrm{in}(p,x).P\} \uplus \mathcal{P}_0$ with $P\lambda_{\theta_1} = P'$ and $\mathcal{P}_0\lambda_{\theta_1} = \mathcal{P}'$. (Note that λ_{θ_1} can only instantiate variables of base type, thus the channel name is necessarily p.)

Let Y be a second order variable of arity $|\Phi_1|$ and y be a fresh first order variable. Let $\mathcal{E}_2 = \mathcal{E}_1$, $\mathcal{P}_2 = \{P\{x \mapsto y\}\} \uplus \mathcal{P}_0$, $\Phi_2 = \Phi_1$, $\mathcal{C}_2 = \mathcal{C}_1 \cup \{Y \rhd^? y\}$, and $\alpha_s = in(p,Y)$. Let θ_2 be the substitution such that $\mathrm{dom}(\theta_2) = \mathrm{dom}(\theta_1) \cup \{Y\}$, $\theta_2 = \theta_1|_{\mathsf{var}^2(\mathcal{C}_1)}$, and $Y\theta_2 = M$. We have that:

1. $(\mathcal{E}_1; \mathcal{P}_1; \Phi_1; \mathcal{C}_1) \xrightarrow{\alpha_s}_s (\mathcal{E}_2; \mathcal{P}_2; \Phi_2; \mathcal{C}_2)$. Indeed, we have that

$$(\mathcal{E}_1; \{\text{in}(p,x).P\} \uplus \mathcal{P}_0; \Phi_1; \mathcal{C}_1) \xrightarrow{in(p,Y)}_s (\mathcal{E}_1; \{P\{x \mapsto y\}\} \uplus \mathcal{P}_0; \Phi_1; \mathcal{C}_1 \cup \{Y \rhd^? y\}).$$

2. We have $\text{dom}(\theta_2) = \mathsf{var}^2(\mathcal{C}_2)$ and $fn(\theta_2) \cap \mathcal{E}_2 = \emptyset$ since $fn(\theta_1) \cap \mathcal{E}_1 = \emptyset$ and $fn(M) \cap \mathcal{E}_1 = \emptyset$. Now, it remains to show that λ_{θ_2} satisfies the constraints in $\mathcal{C}_2$. Actually, we have that $\lambda_{\theta_2} = \lambda_{\theta_1}|_{\mathsf{var}^1(\mathcal{C}_1)}$ and $y\lambda_{\theta_2} = u$. This allows us to conclude since $(Y\theta_2)\Phi_1 = M\Phi_1 = u = y\lambda_{\theta_2}$.

3. We have that

$$\begin{array}{rl} & (\mathcal{E}_2; \mathcal{P}_2\lambda_{\theta_2}; \Phi_2\lambda_{\theta_2}) \\ = & (\mathcal{E}_1; \{P\{x \mapsto y\}\lambda_{\theta_2}\} \uplus \mathcal{P}_0\lambda_{\theta_2}; \Phi_1\lambda_{\theta_1}) \\ = & (\mathcal{E}_1; P\lambda_{\theta_1}\{x \mapsto u\}\} \uplus \mathcal{P}'; \Phi'_1) \\ = & (\mathcal{E}; \{P'\{c \mapsto u\}\} \uplus \mathcal{P}'; \Phi) \\ = & (\mathcal{E}; \mathcal{P}; \Phi). \end{array}$$

4. We have that $\alpha_s\theta_2 = \text{in}(p,Y)\theta_2 = \text{in}(p,M) = \alpha$.

Case OUT-T$_i$**.** In such a case we have that $\mathcal{E} = \mathcal{E}_1$ and there exist p, u', P', and $\mathcal{P}'$ such that $\mathcal{P}'_1 = \{\text{out}(p,u').P'\} \uplus \mathcal{P}'$, $\Phi = \Phi'_1 \cup \{w_l \rhd u'\}$ where $l = |\Phi'_1| + 1$, $\mathcal{P} = \{P'\} \uplus \mathcal{P}'$, and $\alpha = \nu w_l.\text{out}(p, w_l)$. Since $(\mathcal{E}_1; \mathcal{P}'_1; \Phi'_1)$ is the θ_1-concretization of $(\mathcal{E}_1; \mathcal{P}_1; \Phi_1; \mathcal{C}_1)$, we have that $\mathcal{P}'_1 = \mathcal{P}_1\lambda_{\theta_1}$ and $\Phi'_1 = \Phi_1\lambda_{\theta_1}$. Hence, we deduce that there exist u, P and $\mathcal{P}_0$ such that $\mathcal{P}_1 = \{\text{out}(p,u).P\} \uplus \mathcal{P}_0$, with $u\lambda_{\theta_1} = u'$, $P\lambda_{\theta_1} = P'$, and $\mathcal{P}_0\lambda_{\theta_1} = \mathcal{P}'$.

Let $\mathcal{E}_2 = \mathcal{E}_1$, $\mathcal{P}_2 = \{P\} \uplus \mathcal{P}_0$, $\Phi_2 = \Phi_1 \cup \{w_l \rhd u\}$, $\mathcal{C}_2 = \mathcal{C}_1$, $\alpha_s = \nu w_l.\text{out}(p, w_l)$ and $\theta_2 = \theta_1$. We have that:

1. $(\mathcal{E}_1; \mathcal{P}_1; \Phi_1; \mathcal{C}_1) \xrightarrow{\alpha_s}_s (\mathcal{E}_2; \mathcal{P}_2; \Phi_2; \mathcal{C}_2)$. Indeed, we have that

$$(\mathcal{E}_1; \{\text{out}(p,u).P\} \uplus \mathcal{P}_0; \Phi_1; \mathcal{C}_1) \xrightarrow{\nu w_l.out(p,w_l)}_s (\mathcal{E}_1; \{P\} \uplus \mathcal{P}_0; \Phi_1 \cup \{w_l \rhd u\}; \mathcal{C}_1).$$

2. We have that $\theta_2 = \theta_1$, $\mathcal{C}_2 = \mathcal{C}_1$ and $\Phi_2 = \Phi_1 \cup \{w_l \rhd u\}$. Since for all $X \in \mathsf{var}^2(\mathcal{C}_2)$, we have that $ar(X) \leq |\Phi_1|$, we deduce that $\lambda_{\theta_1} = \lambda_{\theta_2}$ and $\theta_2 \in Sol((\mathcal{E}_2; \Phi_2; \mathcal{C}_2))$.

3. We have that

$$\begin{array}{rl} & (\mathcal{E}_2; \mathcal{P}_2\lambda_{\theta_2}; \Phi_2\lambda_{\theta_2}) \\ = & (\mathcal{E}_1; \{P\lambda_{\theta_1}\} \uplus \mathcal{P}_0\lambda_{\theta_1}; \Phi_1\lambda_{\theta_1} \cup \{w_l \rhd u\lambda_{\theta_1}\}) \\ = & (\mathcal{E}; \{P'\} \uplus \mathcal{P}'; \Phi'_1 \cup \{w_l \rhd u'\}) \\ = & (\mathcal{E}; \mathcal{P}; \Phi) \end{array}$$

4. We have that $\alpha_s\theta_2 = \nu w_l.\text{out}(p, w_l) = \alpha$. □

Before to prove that for the class of simple processes without else branch nor replication, trace equivalence and symbolic trace equivalence coincide, we need to relate S-equivalence to static equivalence. As shown in [7], S-equivalence captures static equivalence.

Proposition 8 *Let $\Sigma_1 = (\mathcal{E}; \Phi_1; \mathcal{C}_1)$ and $\Sigma_2 = (\mathcal{E}; \Phi_2; \mathcal{C}_2)$ be two constraint systems with the same structure. Σ_1 and Σ_2 are S-equivalent iff*

1. Σ_1 and Σ_2 are equivalent; and

2. for every $\theta \in Sol(\Sigma_1)$, $\nu\mathcal{E}.\Phi_1\lambda_\theta \sim \nu\mathcal{E}.\Phi_2\lambda_\theta$.

The following proposition can be easily proved by relying on Propositions 2 and 3. We also use Proposition 8 to deal with static equivalence.

Proposition 4 *Let $A = (\mathcal{E}; \mathcal{P}_A; \Phi_A)$ and $B = (\mathcal{E}; \mathcal{P}_B; \Phi_B)$ be two intermediate processes derived from simple processes without else branch nor replication. We have that A and B are in annotated trace equivalence if, and only if, they are in annotated symbolic trace equivalence.*

Proof. We show the two directions separately.

($\Leftarrow$) We consider annotated traces. We have to show that for every $(w, \varphi_A) \in \mathsf{trace}_i(A)$ there exists $(w, \varphi_B) \in \mathsf{trace}_i(B)$ such that $\varphi_A \sim \varphi_B$ (and reciprocally). Let $(w, \varphi_A) \in \mathsf{trace}_i(A)$. By definition of $\mathsf{trace}_i(A)$, this means that there exists $(\mathcal{E}'; \mathcal{P}'_A; \Phi'_A)$ such that $(\mathcal{E}; \mathcal{P}_A; \Phi_A) \overset{w}{\Rightarrow}_i (\mathcal{E}'; \mathcal{P}'_A; \Phi'_A)$, and $\varphi_A = \nu\mathcal{E}'.\Phi'_A$. Let $A_s = (\mathcal{E}; \mathcal{P}_A; \Phi_A; \emptyset)$ and θ be the identity. We have that A is the θ-concretization of A_s and $\theta \in Sol((\mathcal{E}; \Phi_A; \emptyset))$. Thanks to Proposition 3, we have that there exist a symbolic process $A'_s = (\mathcal{E}'; \mathcal{P}'^s_A; \Phi'^s_A; \mathcal{C}'_A)$ and a substitution θ' such that

1. $(\mathcal{E}; \mathcal{P}_A; \Phi_A; \emptyset) \overset{w_s}{\Rightarrow}_s (\mathcal{E}'; \mathcal{P}'^s_A; \Phi'^s_A; \mathcal{C}'_A)$;
2. $\theta' \in Sol((\mathcal{E}'; \Phi'^s_A; \mathcal{C}'_A))$;
3. $(\mathcal{E}'; \mathcal{P}'_A; \Phi'_A)$ is the θ'-concretization of $(\mathcal{E}'; \mathcal{P}'^s_A; \Phi'^s_A; \mathcal{C}'_A)$; and
4. $w_s\theta' = w$.

Let $\Sigma'_A = (\mathcal{E}'; \Phi'^s_A; \mathcal{C}'_A)$. By definition of $\mathsf{trace}_s(A_s)$, we have that $(w_s, \Sigma'_A) \in \mathsf{trace}_s(A_s)$. Since A_s and B_s are in symbolic trace equivalence, we deduce that there exists $\Sigma'_B = (\mathcal{E}'; \Phi'^s_B; \mathcal{C}'_B)$ such that $(w_s, \Sigma'_B) \in \mathsf{trace}_s(B_s)$ and Σ'_A, Σ'_B are S-equivalent. By definition of $\mathsf{trace}_s(B_s)$, we have that

$$(\mathcal{E}; \mathcal{Q}_B; \Phi_B; \emptyset) \overset{w_s}{\Rightarrow}_s (\mathcal{E}'; \mathcal{Q}'^s_B; \Phi'^s_B; \mathcal{C}'_B)$$

for some $\mathcal{Q}'^s_B$, Φ'^s_B and $\mathcal{C}'_B$.

Since Σ'_A and Σ'_B are S-equivalent, they are also equivalent (Proposition 8) and thus $\theta' \in Sol(\Sigma'_B)$. Moreover, we have that $\nu\mathcal{E}'.\Phi'^s_A\lambda_{\theta'} \sim \nu\mathcal{E}'.\Phi'^s_B\lambda_{\theta'}$ (again thanks to Proposition 8). Now, we apply Proposition 2, we have that $(\mathcal{E}; \mathcal{P}_B; \Phi_B) \overset{w_s\theta'}{\Rightarrow}_i (\mathcal{E}'; \mathcal{P}'^s_B\lambda_{\theta'}; \Phi'^s_B\lambda_{\theta'})$. Let $\varphi_B = \nu\mathcal{E}'.\Phi'^s_B\lambda_{\theta'}$. Clearly, we have that $(w_s\theta', \varphi_B) \in \mathsf{trace}_i(B)$. From the fact that $\nu\mathcal{E}'.\Phi'^s_A\lambda_{\theta'} \sim \nu\mathcal{E}'.\Phi'^s_B\lambda_{\theta'}$, we deduce that $\varphi_A \sim \varphi_B$. The other inclusion can be shown in a similar way.

($\Rightarrow$) First, note that for the class of symbolic processes we consider, we have that $A_s \overset{w^a_s}{\Rightarrow} A'_s$ and $A_s \overset{w^a_s}{\Rightarrow} A''_s$ for some annotated trace w^a_s implies that $A'_s = A''_s$. We have to show that A_s and B_s are in symbolic trace equivalence. Let $(w_s, \Sigma'_A) \in \mathsf{trace}_s(A_s)$ and let w^a_s be the associated annotated trace (w^a_s is uniquely defined).

Let $\Sigma'_A = (\mathcal{E}'; \Phi'^s_A; \mathcal{C}'_A)$. Thanks to Proposition 2, we have that $(w^a_s\theta, \nu\mathcal{E}'.\Phi'^s_A\lambda_\theta)$ is an annotated trace of A for every $\theta \in Sol(\Sigma'_A)$. Since A and B are in annotated trace equivalence, we deduce that for every $\theta \in Sol(\Sigma'_A)$ there exists an annotated trace $(w^a_s\theta, \varphi^\theta_B)$ issued from B and such $\nu\mathcal{E}'.\Phi'^s_A\lambda_\theta \sim \varphi^\theta_B$. By completeness, we deduce that for each $\theta \in Sol(\Sigma'_A)$, there exists B'^θ_s whose associated constraint system is Σ'^θ_B and an annotated sequence w^θ_s such that $B_s \overset{w^\theta_s}{\Rightarrow}_s B'^\theta_s$ and $\theta \in Sol(\Sigma'^\theta_B)$. Actually, there exists a unique symbolic derivation that is possible.

Let $\Sigma'_B = (\mathcal{E}'; \Phi'^s_B; \mathcal{C}'_B)$ be the constraint system obtained after such a derivation. Moreover, we necessarily have that $w^\theta_s = w^a_s$. Hence we have that

$$\begin{array}{c}\{(w^a_s\theta, \nu\mathcal{E}'.\Phi'^s_A\lambda_\theta) \mid \theta \in Sol(\Sigma'_A)\}\\ \subseteq \\ \{(w^a_s\theta, \nu\mathcal{E}'.\Phi'^s_B\lambda_\theta) \mid \theta \in Sol(\Sigma'_B)\}.\end{array}$$

Actually, we have that

$$\begin{array}{c}\{(w^a_s\theta, \nu\mathcal{E}'.\Phi'^s_A\lambda_\theta) \mid \theta \in Sol(\Sigma'_A)\}\\ = \\ \{(w^a_s\theta, \nu\mathcal{E}'.\Phi'^s_B\lambda_\theta) \mid \theta \in Sol(\Sigma'_B)\}.\end{array}$$

Indeed, let $(w^a, \varphi'_B) \in \{(w^a_s\theta, \nu\mathcal{E}'.\Phi'^s_B\lambda_\theta) \mid \theta \in Sol(\Sigma'_B)\}$. Since A and B are in annotated trace equivalence, we know that there exists A' such that $A \overset{w^a}{\Rightarrow}_i A'$. Thanks to Proposition 3, we know that A_s can mimic this derivation and it is easy to see that the only possible trace is actually w^a_s, hence the result. $\square$

E Example

This subsection contains an example that illustrates our techniques and the different notions we used.

Example 17 *Consider the theory* $\mathsf{E}_{\mathsf{enc}}$ *described in Example 3, and the two frames:*

- $\varphi_0 = \nu sks.\{^{\mathsf{pk}(sks)}/_{x_1}, {}^{\mathsf{enc}(\langle n_i, \mathsf{v}_0\rangle, \mathsf{pk}(sks))}/_{x_2}\}$,
- $\varphi_1 = \nu sks.\{^{\mathsf{pk}(sks)}/_{x_1}, {}^{\mathsf{enc}(\langle n_i, \mathsf{v}_1\rangle, \mathsf{pk}(sks))}/_{x_2}\}$

We have that $(\mathsf{enc}(\langle n_i, \mathsf{v}_0\rangle, x_1) =_{\mathsf{E}_{\mathsf{enc}}} x_2)\varphi_0$ *whereas* $(\mathsf{enc}(\langle n_i, \mathsf{v}_0\rangle, x_1) \neq_{\mathsf{E}_{\mathsf{enc}}} x_2)\varphi_1$.

However, we have that $\nu n_i.\varphi_0 \sim \nu n_i.\varphi_1$. *This is a non trivial equivalence. Intuitively, there is no test that allows one to distinguish the two frames since neither the decryption key nor the nonce* n_i *are available.*

Example 18 *Consider the theory* $\mathsf{E}_{\mathsf{enc}}$ *and the following process*

$P(v) = \nu sks.out(c, \mathsf{pk}(sks)).out(c, \mathsf{enc}(\langle n_i, v\rangle, \mathsf{pk}(sks)))$.

We have that $\nu n_i.P(\mathsf{v}_0) \approx \nu n_i.P(\mathsf{v}_1)$ *whereas* $P(\mathsf{v}_0) \not\approx P(\mathsf{v}_1)$. *These results are direct consequences of the static (in)equivalence relations stated and discussed in Example 17.*

Example 19 *We consider a protocol that could be used for the voting phase of an e-voting protocol. In this protocol,* S *sends a token, i.e. a nonce* N_s*, to the voter* A*. Then* A *votes by sending the token together with his vote, the whole encrypted by the public key of* S*. We will see that the nonce* N_a *used in this answer is crucial to ensure privacy, i.e. the fact that* A *does not want to reveal his vote* v*. The participants* A *and* S *proceed as follows:*

$$\begin{array}{lcl} S \to A & : & \mathsf{enc}(\langle N_s, \mathsf{pub}(A)\rangle, \mathsf{pub}(A)) \\ A \to S & : & \mathsf{enc}(\langle N_s, \langle v, N_a\rangle\rangle, \mathsf{pub}(S)) \end{array}$$

Firstly S *sends to* A *a nonce* N_s *concatenated with the public key of* A *(in order to identify* A*), all of this being encrypted with the public key of* A*. If the message is of the expected form then* A *sends to* S *the nonce* N_s*, a freshly generated nonce* N_a *and his vote* v*, all of this being encrypted with the public key of the server* S*. From the point of view of an outsider, the messages should basically look similar for any voting option. This is important since the protocol is supposed to ensure privacy.*

A session of role A *played by agent* a *with* s *to cast the vote* v *can be modeled by the following basic process where* true *denotes a test that is always satisfied and* $M = \mathsf{dec}(x, ska)$*. Note that* A *is not given the value* sks *but is directly given the value* $\mathsf{pk}(sks)$*, that is the public key corresponding to* S*'s private key.*

$$\begin{array}{rl} A(a,b,v) \overset{def}{=} & in(c_A, x). \\ & \textit{if } \pi_2(M) = \mathsf{pk}(ska) \textit{ then} \\ & out(c_A, \mathsf{enc}(\langle \pi_1(M), \langle v, n_a\rangle\rangle, \mathsf{pk}(sks))).0 \\ & \textit{else } 0. \end{array}$$

where $M = \mathsf{dec}(x, sks)$.

Similarly, a session of role S *played by agent* s *with* a *can be modeled by the basic process* $S(s, a)$ *where* $N = \mathsf{dec}(y, sks)$.

$$
\begin{array}{rl}
S(s,a) \stackrel{def}{=} & \mathit{if}\ \mathsf{true}\ \mathit{then} \\
& \quad out(c_S, \mathsf{enc}(\langle n_s, \mathsf{pk}(ska)\rangle, \mathsf{pk}(ska))) \\
& \quad in(c_S, y). \\
& \quad \mathit{if}\ \pi_1(N) = n_s\, \mathit{then}\ out(c_S, \mathsf{ok}).0 \\
& \quad \mathit{else}\ 0 \\
& \mathit{else}\ 0.
\end{array}
$$

Intuitively, this protocol preserves privacy if an attacker cannot distinguish whether a *is casting* v_0 *or* v_1. *For illustration purposes, we also consider the process* $A'(a, s, v)$ *obtained from* $A(a, s, v)$ *by removing the nonce* n_a *in the outputted message. We will see that this nonce plays a crucial role to ensure privacy.*

Example 20 *Continuing Example 19, a simple process representing an unbounded number of sessions in which* a *plays* A *(with* s*) to cast vote* v *and* s *plays* S *with* a *is:*

$$\nu ska, sks.(\ !(\nu n_a.A(a, s, v))\ \mid\ (\nu n_s.S(s, a))\ )$$

This represents the case where A *may try to vote several times while the server* S *will provide only one token to* A. *For modeling and verification purposes, we wish to disclose the public keys in order to make them available to the attacker. This can be done by means of an additional basic process*

$$K(a, s) = out(c_K, \mathsf{pk}(ska)) \cdot out(c_K, \mathsf{pk}(sks)).0$$

Example 21 *The triple* $\Sigma_s = (\mathcal{E}_s; \Phi^0_s \cup \{w_3 \rhd t\}; \mathcal{C}_s)$ *where*

$$
\begin{array}{l}
\mathcal{E}_s = \{ska, sks, n_a, n_s\}, \\
\Phi^0_s = \{w_1 \rhd \mathsf{pk}(ska), w_2 \rhd \mathsf{pk}(sks)\}, \\
t = \mathsf{enc}(\langle \pi_1(\mathsf{dec}(x, ska)), \mathsf{v}_0\rangle, \mathsf{pk}(sks)), \\
\mathcal{C}_s = \{X \rhd^? x,\ \pi_2(\mathsf{dec}(x, ska)) =^?_{\mathsf{E}} \mathsf{pk}(ska)\}, ar(X) = 2
\end{array}
$$

is a constraint system. We will see that it corresponds to the execution of the process $A'(a, s, \mathsf{v}_0)$ *presented in Example 19. Note that the public keys of the participants are made available to the attacker since they appear in* Φ^0_s.

Example 22 *Continuing Example 21, a solution to* $\Sigma_s = (\mathcal{E}_s; \Phi_s; \mathcal{C}_s)$ *is* θ *where* $\mathrm{dom}(\theta) = \{X\}$ *and* $\theta(X) = \mathsf{enc}(\langle n_i, w_1\rangle, w_1)$ *with* n_i *a public name (i.e.* $n_i \notin \mathcal{E}_s$*). The first order-solution* λ *of* Σ_s *associated to* θ *is a substitution whose domain is* $\{x\}$ *and such that* $\lambda(x) = \mathsf{enc}(\langle n_i, \mathsf{pk}(ska)\rangle, \mathsf{pk}(ska))$.

Example 23 *Let* Σ'_s *be the following constraint system:*

$$\Sigma'_s = (\mathcal{E}_s; \Phi^0_s \cup \{w_3 \rhd t_1\}; \mathcal{C}_s)$$

where $t_1 = \mathsf{enc}(\langle \pi_1(\mathsf{dec}(x, ska)), \mathsf{v}_1\rangle, \mathsf{pk}(sks))\}$, *and* $\mathcal{E}_s$, $\mathcal{C}_s$ *and* Φ^0_s *are defined as in Example 21. This system corresponds to the system obtained after a symbolic execution of the process* $A'(a, s, \mathsf{v}_1)$ *presented in Example 19.*

The system Σ_s *(given in Example 21) is not* S*-equivalent to* Σ'_s. *Indeed, the two following constraint systems are not equivalent:*

- $(\mathcal{E}_s; \Phi_s; \mathcal{C}_s \cup \{X_1 \rhd^? x_1, X_2 \rhd^? x_2, x_1 =^?_{\mathsf{E}} x_2\})$, *and*
- $(\mathcal{E}_s; \Phi'_s; \mathcal{C}_s \cup \{X_1 \rhd^? x_1, X_2 \rhd^? x_2, x_1 =^?_{\mathsf{E}} x_2\})$.

Let θ *be the substitution such that* $\mathrm{dom}(\theta) = \{X, X_1, X_2\}$ *and* $\theta(X) = \mathsf{enc}(\langle n_i, w_1\rangle, w_1)$, $\theta(X_1) = \mathsf{enc}(\langle n_i, \mathsf{v}_0\rangle, w_2)$ *and* $\theta(X_2) = w_3$. *We have that* $\theta \in Sol(\Sigma_s)$ *whereas* $\theta \notin Sol(\Sigma'_s)$. *We conclude that the constraint systems* Σ_s *and* Σ'_s *are not in* S*-equivalence. Actually, this corresponds to the fact that an attacker can distinguish between* $A'(a, s, \mathsf{v}_0)$ *and* $A'(a, s, \mathsf{v}_1)$ *by sending the message* $\mathsf{enc}(\langle n_i, \mathsf{pk}(ska)\rangle, \mathsf{pk}(ska))$ *to* a *and see whether the answer of* a *is equal to* $\mathsf{enc}(\langle n_i, \mathsf{v}_0\rangle, \mathsf{pk}(sks))$ *or not.*

Example 24 *We consider one session of the protocol presented in Example 19, in which* a *plays the role* A' *(with* s*) to cast the vote* v_0 *and* s *plays the role* S *with* a. *We consider the process* $K(a, s)$ *defined in Example 20. Let* $\mathcal{E}$ *be the set of names* $\{ska, sks, n_a, n_s\}$, *and* P^s_{ex} *be the following symbolic process:*

$$P^s_{\mathsf{ex}} = (\mathcal{E}; \{A'(a, b, \mathsf{v}_0),\ S(s, a),\ K(a, s)\}; \emptyset; \emptyset).$$

We have that $P^s_{\mathsf{ex}} \stackrel{\mathsf{tr}}{\Rightarrow}_s (\mathcal{E}_s; \mathcal{P}_s; \Phi_s; \mathcal{C}_s)$ *where*

- $\mathsf{tr} = \nu w_1.out(c_K, w_1) \cdot \nu w_2.out(c_K, w_2) \cdot in(c_A, x) \cdot \nu w_3.out(c_A, w_3)$,
- $\mathcal{P}_s = \{S(s, a)\}$, *and*
- $(\mathcal{E}_s; \Phi_s; \mathcal{C}_s)$ *is the constraint system* Σ_s *defined in Example 21.*

Example 25 *Relying on our techniques, we can now prove that the two processes* P_{ex} *and* P'_{ex} *are not in observational equivalence, where*

- $P_{\mathsf{ex}} = \nu\tilde{n}.[\, A'(a, s, \mathsf{v}_0) \mid S(s, a) \mid K(a, s)\,]$, *and*
- $P'_{\mathsf{ex}} = \nu\tilde{n}.[\, A'(a, s, \mathsf{v}_1) \mid S(s, a) \mid K(a, s)\,]$.

Continuing Example 24, we have that $(\mathsf{tr}, \Sigma_s) \in \mathit{trace}_s(P^s_{\mathsf{ex}})$ *and* Σ_s *is satisfiable (see Example 22). The only constraint system reachable from*

$$P'^s_{\mathsf{ex}} = (\mathcal{E}; \{A'(a, s, \mathsf{v}_1),\, S(s, a),\, K(a, s)\}; \emptyset; \emptyset)$$

by the sequence tr *is* Σ'_s *as defined in Example 23. We have seen that* Σ'_s *is not in* S*-equivalence with* Σ_s*. This allows us to conclude that the simple processes* P_{ex} *and* P'_{ex} *are not in symbolic trace equivalence, and thanks to Proposition 5, Theorem 1 and Theorem 2, we conclude that* P_{ex} *and* P'_{ex} *are not in observational equivalence.*

Our techniques also allow one to show that the processes Q_{ex} *and* Q'_{ex} *are in observational equivalence, where*

- $Q_{\mathsf{ex}} = \nu\tilde{n}.[\, A(a, s, \mathsf{v}_0) \mid S(s, a) \mid K(a, s)\,]$*, and*
- $Q'_{\mathsf{ex}} = \nu\tilde{n}.[\, A(a, s, \mathsf{v}_1) \mid S(s, a) \mid K(a, s)\,]$.

Let

$$\begin{array}{rcl} Q^s_{\mathsf{ex}} & = & (\mathcal{E}; \{A(a, b, \mathsf{v}_0),\, S(s, a),\, K(a, s)\}; \emptyset; \emptyset) \\ Q'^s_{\mathsf{ex}} & = & (\mathcal{E}; \{A(a, b, \mathsf{v}_1),\, S(s, a),\, K(a, s)\}; \emptyset; \emptyset) \end{array}$$

Thanks to Proposition 5, Theorem 1 and Theorem 2, it is sufficient to check that, for any $(\mathsf{tr}, \Sigma_s) \in \mathit{trace}_s(Q^s_{\mathsf{ex}})$*, there exists* $(\mathsf{tr}, \Sigma'_s) \in \mathit{trace}_s(Q'^s_{\mathsf{ex}})$ *such that* Σ_s *and* Σ'_s *are in* S*-equivalence, and reciprocally.* S*-equivalence can be checked by hand in this example or using the decision procedure of [7]. This shows that the nonce* N_a *plays a crucial role to ensure privacy.*