Deciding equivalence-based properties using constraint solving[1]

Vincent Cheval[1], Véronique Cortier[2], and Stéphanie Delaune[1]

[1] LSV, ENS Cachan & CNRS & INRIA Saclay Île-de-France, France
[2] LORIA, CNRS, 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 a notion of *behavioural equivalence*. Typical examples are anonymity, privacy related properties or statements closer to security properties used in cryptography.

In this paper, we consider three notions of equivalence defined in the applied pi calculus: *observational equivalence*, *may-testing equivalence*, and *trace equivalence*. First, we study the relationship between these three notions. 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. While trace equivalence and may-testing equivalence seem very similar, we show that may-testing equivalence is actually strictly stronger than trace equivalence. We prove that the two notions coincide for *image-finite* processes, such as processes without replication.

Second, we reduce the decidability of trace equivalence (for finite processes) to deciding symbolic equivalence between sets of constraint systems. For simple processes without replication and with trivial else branches, it turns out that it is actually sufficient to decide symbolic equivalence between pairs of positive constraint systems. Thanks to this reduction and relying on a result first proved by M. Baudet, this yields the first decidability result of observational equivalence for a general class of equational theories (for processes without else branch nor replication). Moreover, based on another decidability result for deciding equivalence between sets of constraint systems, we get decidability of trace equivalence for processes with else branch for standard primitives.

Keywords: security protocols, formal verification, applied-pi calculus, behavioural equivalences, constraint systems

1. Introduction

Security protocols aim at securing communications over insecure networks such as the Internet, where dishonest users may listen to communications and interfere with them. A *secure communication* has a different meaning depending on the underlying application. It ranges from the confidentiality of a data (medical files, secret keys, *etc.*) to e.g. verifiability in electronic voting systems. Another example of a security notion is privacy. As soon as personal data are manipulated, security mechanisms should enforce that entities access to some information only when they are entitled to. For example, passports are no more pure paper documents. Instead, they

[1]The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement n° 258865, project ProSecure, and the ANR project JCJC VIP n° 11 JS02 006 01.

contain a chip that stores additional information such as pictures and fingerprints of its holder. In order to ensure privacy, these chips include a mechanism that do not let the passport disclose private information to external users. However, it has been shown that it is nonetheless possible to recognize a previously observed passport, potentially tracing passport holders [8]. This is just a single example but of course privacy appears in many other contexts such as RDFIDs technologies or electronic voting. Protection for privacy typically relies on protocols (and/or access control) making use of cryptography. 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. They rely on symbolic models for protocols, where messages are represented by terms, an algebraic structure that abstracts away the underlying cryptography. Even if a basic property such as confidentiality is undecidable in general, many techniques and tools have been proposed. For example, secrecy preservation is co-NP-complete [6, 43, 48] for the case of a bounded number of sessions (*i.e.* assuming the protocol is executed a finite number of times). For an unbounded number of sessions, several decidable classes have been identified, e.g. when protocols have a simple structure [41] or when cyphertexts contain a label [15, 47]. Several tools have also been developed to verify cryptographic protocols. They can automatically find attacks or prove security. Popular tools are e.g. ProVerif [12], Avispa [9], or Scyther [30].

However most of these 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: a data remains confidential if, for any execution, the attacker is not able to *produce* the data. But privacy properties cannot be defined (or cannot be naturally defined) as trace properties. Indeed, consider for example the simple case of anonymity: a user Alice does not want her identity to be revealed. A first idea would be to request the confidentiality of the data "Alice". This is however unrealistic: identities are usually known to the public. And actually, the confidential information is not the data "Alice" but the fact that Alice did take part in a particular communication. So anonymity is usually defined as the fact that an observer cannot distinguish the case where A is talking from the case where B is talking (see [3]). This notion of indistinguishability requires a notion of *behavioural equivalence*. Roughly, two processes P and Q are equivalent ($P \approx Q$) if no process O can observe any difference between the processes P and Q. The notion of behavioural equivalence is crucial when specifying properties like anonymity and more generally, privacy. Privacy related properties involved in electronic voting protocols (*e.g.* [32]) also use equivalence as a key notion and cannot be expressed in linear temporal logic. For example, ballot secrecy is typically defined by the fact that an observer should not observe when two voters swap their votes: $A(yes) \mid B(no) \approx A(no) \mid B(yes)$. Other examples of privacy properties are untraceability in the context of RFIDs [17] or confidentiality for cloud computing conference management systems [7]. All these properties are again modeled using behavioural equivalences. The use of behavioural equivalences is not limited to privacy properties. Equivalence is also used for defining a stronger notion of secrecy, called "strong secrecy" [13] or even for defining authentication [4]. More generally, it is a notion that allows one to express flexible notions of security by requiring equivalence between a protocol and an idealized version of it, that magically realizes the desired properties.

Behavioural equivalences. Several definitions of equivalence have been proposed. In this paper, we focus on three main definitions: observational equivalence, may-testing equivalence, and trace equivalence, in the context of applied pi-calculus [2], which is well-suited for the analysis of security protocols. Roughly, two processes P and Q are may-testing equivalent, denoted by $P \approx_m Q$, if for any process O the processes $P \mid O$ and $Q \mid O$ are equally able to emit on a given channel. If the processes $P \mid O$ and $Q \mid O$ are in addition weakly bisimilar, we say that they are in observational equivalence ($P \approx Q$) . For trace equivalence ($P \approx_t Q$), P and Q must have the same sets of observable traces and have sent statically equivalent (*i.e.* indistinguishable for the attacker) sequences of messages.

Any of these three notions can be used to define privacy-like properties. While the exact choice of behavioral equivalence does not seem crucial to distinguish between secure and insecure

protocols in real examples, these three notions are not identical. For example, requiring two processes to be weakly bisimilar appears often to be too strong with respect to what an attacker can really observe. Instead, trace equivalence seems better appropriate to capture the ability of attackers. For example, the proof of [26] indicates that standard cryptographic definitions for security, based on indistinguishable games, are soundly abstracted by trace equivalence, with actually no need for observational equivalence. Conversely, two processes may not be weakly bisimilar and yet be indistinguishable from an attacker that is seen as an external program. This can be illustrated by the following example: $\mathsf{out}(a).(\mathsf{out}(b) + \mathsf{out}(c))$ and $\mathsf{out}(a).\mathsf{out}(b) + \mathsf{out}(a).\mathsf{out}(c)$.

Our contributions - in brief. In this paper, we provide two main contributions. First, we formally compare observational equivalence, may-testing equivalence, and trace equivalence. We show that observational equivalence and may-testing equivalence are both strictly stronger than trace equivalence and we provide sufficient conditions for these notions to coincide. Then, focusing on trace equivalence, we devise a decision procedure for equivalence properties. For protocols without replication, we show that deciding trace equivalence can be reduced to deciding equivalence of sets of set constraints, a structure that abstracts the protocol behavior and is easier to reason with. As a application, we can use two theorems [10, 11, 20] of the literature to deduce two decidability results for trace equivalence of protocols.

- Trace equivalence is decidable for a bounded number of sessions and for standard cryptographic primitives.
- Trace equivalence is decidable for a bounded number of sessions and for a larger class of cryptographic primitives (defined by subterm convergent equational theories) provided that protocols do not use else branches.

We further describe our contributions in the remaining of this section.

Comparing behavioural equivalences. Our first main contribution is to compare trace equivalence with the two other notions. Our results are summarized in Figure 1. As expected, observational equivalence is strictly stronger than trace equivalence. The converse implication was studied by J. Engelfriet. He has shown that observational equivalence and trace equivalence actually coincide in a general model of parallel computation with atomic actions (*e.g.* CCS [44], CSP [37]), when processes are *determinate* [36]. 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 $P' \approx P''$. We generalize this result to a framework well adapted to security protocols, namely the applied pi calculus, which consists in the pi calculus algebra enriched with terms and equational theories on terms (input and output actions are not necessarily atomic and may involve complex terms). We also 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 arbitrary equational theory. Consequently, this class captures most existing security protocols and cryptographic primitives.

May-testing equivalence and trace equivalence are two very close notions. Intuitively, both definitions require that equivalent processes have the same sets of traces. We show that while trace equivalence does imply may-testing equivalence, the converse does not hold. The difference between the two notions is subtle. We exhibit a counter-example that relies on the fact that may-testing equivalence requires the attacker to commit in advance on part of its behavior, yielding a slightly weaker attacker. We further show that may-testing equivalence does imply trace equivalence in case the processes have finitely many successors (*e.g* for processes without replication): trace equivalence and may-testing equivalence coincide for image-finite processes.

Towards decidability. Our second main contribution is to propose a proof techniques for deciding equivalence. Since replication very quickly yields to undecidability even in the simpler case

of trace properties [35, 5] such as secrecy, we focus here on processes without replication for which we have just seen that trace equivalence and may-testing equivalence coincide. For processes that are determinate, we have furthermore established that observational equivalence coincides with trace equivalence. We therefore concentrate our efforts on trace equivalence, which is also more amendable to automation.

The applied pi calculus is elegant and convenient for expressing security protocols. However, its syntax and semantics (in particular name restriction and parallel composition) do not ease the verification task. In contrast, constraint systems are much simpler and capture exactly the core of protocol executions. They have shown their usefulness for analysing security protocols, at least for secrecy and authentication properties (*e.g.* [43, 28, 27, 21]). We show a reduction result for general processes without replication and for arbitrary equational theories. We reduce the decidability of trace equivalence (for finite processes) to deciding symbolic equivalence between sets of constraint systems. For simple processes without replication and with trivial else branches, it turns out that it is actually sufficient to decide symbolic equivalence between pairs of positive constraint systems. To transfer executions from the applied pi calculus to constraint systems, we introduce an intermediate calculus.

Equivalence of constraint systems is easier to analyse and has already been studied:

- Equivalence of sets of constraint systems is decidable for the fixed theory corresponding to standard cryptographic primitives (symmetric and asymmetric encryption, signatures, hashes) [20].
- Equivalence of two positive constraint systems is also decidable for any subterm convergent theory [10, 11] (a large family of equational theories [1]).

Applying our approach, the first result immediately implies that trace equivalence is decidable for standard primitives (with some technical work to make the two frameworks actually coincide). The second result does not allow to compare arbitrary sets of constraint systems but only pairs of constraint systems. This result is however sufficient to deduce decidability of trace equivalence for simple processes without replication and with trivial else branches, for any subterm convergent theory.

For the sake of clarity, we provide an overview of all our results in Figure 1.

Related work. In contrast to the case of trace properties (e.g. secrecy, authentication), there are few results on automating the analysis of equivalence-based properties. The first decidability result was provided by [38], for a fragment of the spi-calculus (with no replication nor else branches), and a fixed set of primitives. This approach cannot be implemented due to its complexity. Both Boreale *et. al.* [42] and Durante *et. al.* [34] compare trace equivalence and may-testing equivalence for processes with no replication nor else branches. They also provide proof techniques for trace equivalence by developing some proofs on examples.

As already mentioned, Baudet [10, 11] proves decidability of equivalence between positive constraint systems for subterm convergent theories. In [23], a shorter proof of the result by Baudet is given. The result is based on an extension of the small attack property. It is shown that if two processes are not equivalent, then there must exist a small witness of non-equivalence. A decision of equivalence can be derived by checking every possible small witness. As in Baudet's approach, the main issue is the practicality. The number of small witnesses is very large as all terms of size smaller than a given bound have to be considered. Consequently, this method has not been implemented. [20] shows decidability of trace equivalence for standard primitives and processes with no replication (but possibly else branches). An early version of the decision procedure described in [20] has been implemented in the ADECS tool. Up to our knowledge, there are no other decidability results for trace nor observational equivalences, at least for process algebra dedicated to security protocols.

[18] has proposed a procedure based on Horn clauses for the class of optimally reducing theories, which encompasses subterm convergent theories. The procedure is sound and complete but its termination is not guaranteed. It applies to determinate processes without replication

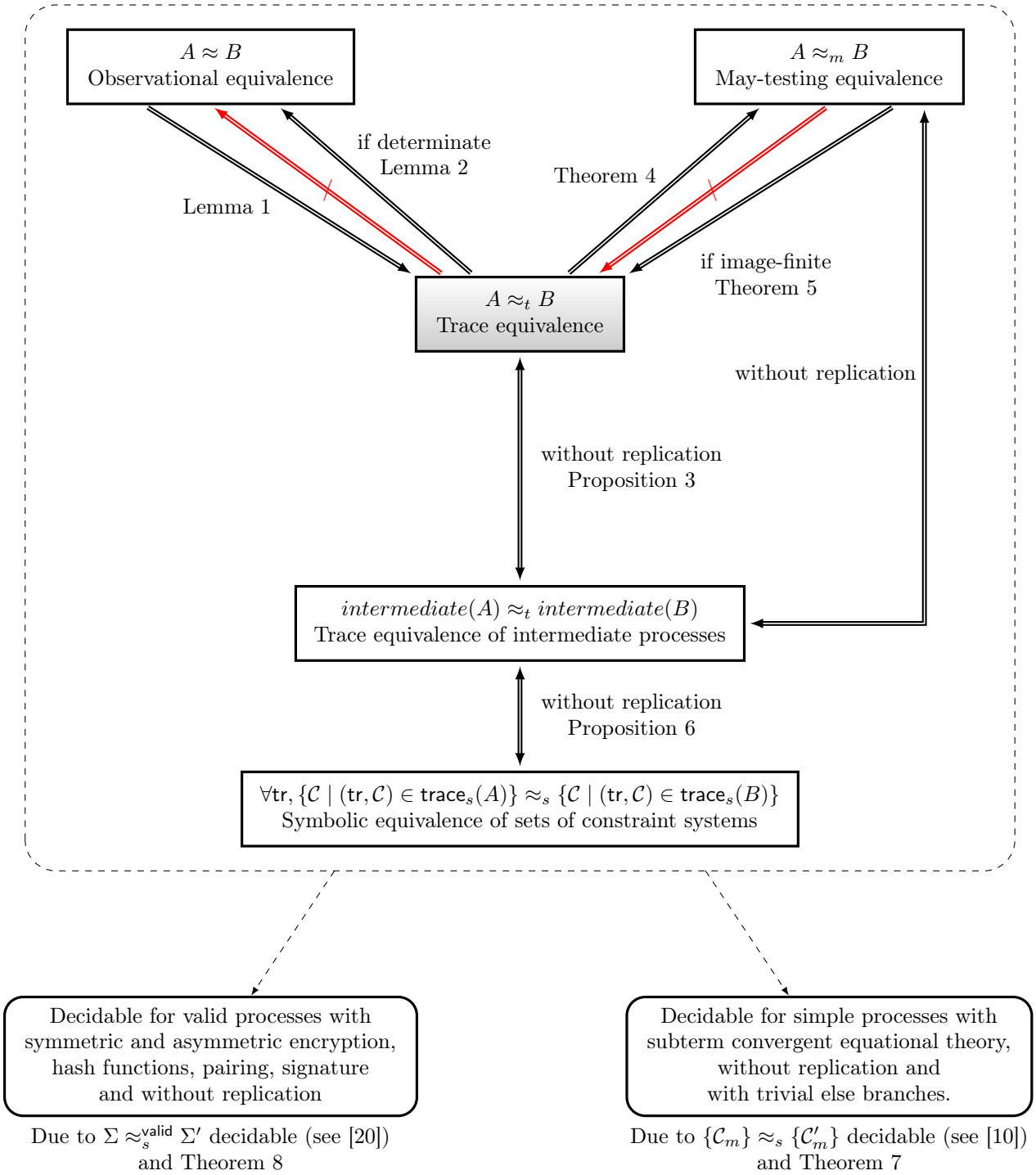

Figure 1: Overview of the results.

nor else branches. Moreover, when processes are not determinate, the procedure can be used for both under- and over-approximations of trace equivalence. A slightly different approach consists in designing and deciding a stronger notion. In particular, B. Blanchet *et. al.* [14] checks for diff-equivalence. The procedure has been implemented in ProVerif and works well in practice for properties such as strong properties. It is however too strong in general. For the particular case of ballot secrecy for electronic voting protocols, B. Smyth *et. al.* [33] have proposed a transformation (implemented in the ProSwapper tool), that helps ProVerif to prove equivalences. [49] proposes a notion of strong bisimulation that strictly implies trace equivalence. They provide a decision procedure implemented in a tool named SPEC. [31] similarly proposes a notion of symbolic bisimulation, stronger than trace equivalence. A proof system, sound but incomplete has been proposed for observational equivalence [39, 40]. Except for [20], none of these results allow one to deal with processes with non-trivial else branches.

This paper can be seen as an extended and enriched version of [29] and a part of [20]. In [29], it was shown that observational and trace equivalence coincide for determinate processes, identifying the class of simple processes, which are always determinate. We retrieve decidability of trace equivalence for processes without replication and with trivial else branches, for any subterm convergent theory using [10]. The reduction of checking trace equivalence to checking equivalence of sets of constraint systems has been presented, for a more limited framework, in [20]. However, the core of [20] is the decidability proof for equivalence of sets of constraint systems, which is not studied here. Our result on comparing may-testing equivalence and trace equivalence is completely new.

Structure of the paper. We have split the paper in two main parts. Part I defines our three notions of equivalence (Section 2) and compares trace equivalence with observational equivalence (Section 3) and may-testing equivalence (Section 4). Part II develops techniques for proving trace equivalence. Reducing trace equivalence to equivalence between sets of constraints require to introduce an intermediate calculus (Section 5) and a symbolic calculus (Section 6). We then show in Section 7 how to relate trace equivalence to symbolic trace equivalence (which reduces to checking equivalence between sets of constraint systems), first for general processes (without replication) and then for the particular case of simple processes without else branches. We show how our framework can be applied to [20] and [10] to deduce two decidability results for trace equivalence of processes in the applied pi calculus.

Contents

1 Introduction **1**

I Trace, observational, and may-testing equivalences **9**

2 The applied pi calculus **9**
2.1 Syntax . . . 9
2.2 Semantics . . . 10
2.3 Behavioural equivalences . . . 11
2.3.1 Trace equivalence . . . 12
2.3.2 May-testing equivalence . . . 12
2.3.3 Observational equivalence . . . 13

3 Relating observational and trace equivalences **13**
3.1 Determinacy . . . 14
3.2 An expressive class of determinate processes . . . 15

4 Relating may-testing and trace equivalences **18**
4.1 Trace equivalence implies may-testing equivalence . . . 19
4.2 May-testing equivalence does not imply trace equivalence . . . 19
4.3 May-testing equivalence implies trace equivalence for image-finite processes . . . 20

II Towards deciding trace equivalence **22**

5 Intermediate calculus **22**
5.1 Syntax . . . 22
5.2 Semantics . . . 23
5.3 Equivalence . . . 24

6 Symbolic calculus **25**
6.1 Constraint system . . . 25
6.2 Syntax and semantics . . . 26
6.3 Soundness and completeness . . . 27
6.4 Symbolic equivalence of constraint systems . . . 28

7 Relating trace equivalence and symbolic trace equivalence **28**
7.1 General processes . . . 29
7.2 Simple processes with trivial else branches . . . 30
7.3 Decidability results . . . 32
7.3.1 Decidability of trace equivalence for standard primitives . . . 32
7.3.2 Decidability of trace equivalence for simple processes . . . 33

8 Conclusion **34**

Appendix **39**

A Testing equivalence vs trace equivalence **39**

B Intermediate calculus **42**

C Symbolic calculus **44**

D Decidability of trace equivalence for standard primitives **49**

Part I
Trace, observational, and may-testing equivalences

The goal of this first part is to define and compare three notions of equivalence: trace equivalence, observational equivalence, and may-testing equivalence.

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*, denoted by $\mathcal{N} = \{a, b, \ldots, sk, k, n, \ldots\}$, which is split into the set $\mathcal{N}_b$ of names of *base types* and the set $\mathcal{C}h$ of names of *channel types* (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 $fv(T) = \emptyset$.

Example 1. *Consider the following signature*

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

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*.*

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

where M and N are terms, n is a name, x is a variable and u is a term of channel type, *i.e.* a name or a variable.

Figure 2: Syntax of processes

The applied pi calculus defines *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 contain terms rather than just names. Extended processes add *active substitutions* and restriction on variables (see Figure 2).

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

"let $x = M$ in P".

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). The domain $\mathrm{dom}(A)$ of an extended process A is the domain of $\phi(A)$.

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

$$\nu s.\nu sk.\nu x_1.(\mathit{out}(c_1, x_1) \mid \mathit{in}(c_1, y).\mathit{out}(c_2, \mathsf{adec}(y, sk)) \mid \{^{\mathsf{aenc}(s,\mathsf{pk}(sk))}/_{x_1}\}).$$

Its first component publishes the message $\mathsf{aenc}(s, \mathsf{pk}(sk))$ 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 $\phi(A) = \nu s, sk, x_1.\{^{\mathsf{aenc}(s,\mathsf{pk}(sk))}/_{x_1}\}$ 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 $\mathcal{F}$, which can *e.g.* represent the behaviour of the cryptographic primitives. An equational theory is a relation on terms that is closed under substitutions of terms for variables. We further require the equational theory to be closed under one-to-one renaming, but not necessarily closed under substitutions of arbitrary terms for names. Usually, an equational theory is generated from a finite set of equations $M = N$ with $M, N \in \mathcal{T}(\emptyset, \mathcal{X})$. In this case, we have that E is closed by substitutions of terms for names.

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

$$\mathsf{adec}(\mathsf{aenc}(x, \mathsf{pk}(y)), y) \;=\; x \qquad \pi_i(\langle x_1, x_2\rangle) \;=\; x_i \quad \textit{for } i \in \{1, 2\}.$$

The first equation represents asymmetric decryption while the second ones represent the first and second projections of the pair.
For example, we have that $\pi_1(\mathsf{adec}(\mathsf{aenc}(\langle n_1, n_2\rangle, \mathsf{pk}(sk)), sk)) =_{\mathsf{E}_{\mathsf{aenc}}} n_1$.

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 $|$, binding-operator-like behavior of ν, and when $M =_{\mathsf{E}} N$ the equivalences:

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

Example 4. *Let P be the following process:*

$$\nu s.\nu sk.\ \big(\mathit{out}(c_1, \mathsf{aenc}(s, \mathsf{pk}(sk))) \mid \mathit{in}(c_1, y).\mathit{out}(c_2, \mathsf{adec}(y, sk))\big).$$

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:

$$\begin{array}{rcll} \mathtt{out}(a,M).P \mid \mathtt{in}(a,x).Q & \xrightarrow{\tau} & P \mid Q\{^M/_x\} & \\ \mathtt{if}\ M=N\ \mathtt{then}\ P\ \mathtt{else}\ Q & \xrightarrow{\tau} & P & \text{if } M =_{\mathsf{E}} N \\ \mathtt{if}\ M=N\ \mathtt{then}\ P\ \mathtt{else}\ Q & \xrightarrow{\tau} & Q & \text{if } M,N \text{ ground terms such that } M \neq_{\mathsf{E}} N \end{array}$$

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.

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

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

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

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

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

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

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

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 [31] that the two underlying notions of observational equivalence coincide.

Example 5. *Let P be the process defined in Example 4. We have that:*

$$\begin{array}{rcl} P & \xrightarrow{\boldsymbol{out}(c_1,x_1)} & \nu s.\nu sk.(\{^{\mathsf{aenc}(s,\mathsf{pk}(sk))}/_{x_1}\} \mid \boldsymbol{in}(c_1,y).\boldsymbol{out}(c_2,\mathsf{adec}(y,sk))) \\ & \xrightarrow{\boldsymbol{in}(c_1,x_1)} & \nu s.\nu sk.(\{^{\mathsf{aenc}(s,\mathsf{pk}(sk))}/_{x_1}\} \mid \boldsymbol{out}(c_2,s)) \end{array}$$

2.3. Behavioural equivalences

Behavioural equivalences intuitively define the fact that no observer can see the difference between two processes. They can be used to formalize many interesting security properties, in particular privacy related properties, such as those studied in [3, 32, 8, 17]. Slightly more specifically, two processes can be said equivalent if an observer, whatever how he behaves, observes the same (or equivalent) outputs from the two processes. This can be formally defined as *trace equivalence* as introduced in [42]. An alternative definition is *testing equivalence* as defined for example by M. Abadi and A. Gordon [4]. In the context of the pi calculus, may-testing and trace equivalences are known to be difficult to prove. Therefore, a stronger notion has been proposed: *observational equivalence*, which requires in addition the two processes to be (weakly) bisimilar.

This section is devoted to the definition of these three notions. Sections 3 and 4 then study their relation in the context of the applied pi calculus.

Notations:. 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 $s \in (\mathcal{A} \setminus \{\tau\})^*$, the relation $\overset{s}{\Rightarrow}$ on extended processes is defined by: $A \overset{s}{\Rightarrow} B$ 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, $A \overset{s}{\Rightarrow} B$ means that A transforms into B by experiment s. We also consider the relation $A \overset{w}{\mapsto} B$ and $A \overset{s}{\Mapsto} B$ that are the restriction of the relations $\xrightarrow{w}$ and $\overset{s}{\Rightarrow}$ on closed extended processes.

2.3.1. Trace equivalence

Before defining trace equivalence, we introduce the notion of *static equivalence* that compares sequences of messages, a notion of intruder's knowledge that has been extensively studied (*e.g.* [1]).

Definition 1 (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: $(M =_{\mathsf{E}} N)\phi_1$ if and only if $(M =_{\mathsf{E}} N)\phi_2$.*

Example 6. *Consider the theory $\mathsf{E}_{\mathsf{aenc}}$ (see Example 3), $\varphi_a = \{^{\mathsf{aenc}(a,\mathsf{pk}(sk))}/_{x_1}\}$, and $\varphi_b = \{^{\mathsf{aenc}(b,\mathsf{pk}(sk))}/_{x_1}\}$. We have that $(\mathsf{adec}(x_1, sk) =_{\mathsf{E}_{\mathsf{aenc}}} a)\varphi_a$ whereas $(\mathsf{adec}(x_1, sk) \neq_{\mathsf{E}_{\mathsf{aenc}}} a)\varphi_b$, 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.

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 of base type are exclusively stored in the frame and not in the sequence s (the outputs are made by "reference").

Two processes are trace equivalent if, whatever the messages they received (built upon previously sent messages), the resulting frames are in static equivalence.

Definition 2 (trace equivalence $\approx_t$). *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 $\varphi \sim \varphi'$.*

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

2.3.2. May-testing equivalence

We write $A \Downarrow c$ when A can send a message on c, that is, when $A \overset{\epsilon}{\Rightarrow} C[\mathtt{out}(c, M).P]$ for some evaluation context C that does not bind c. A *test* consists of any evaluation context C and any channel name c. A closed extended process A *passes* the test if and only if $C[A]\Downarrow c$. The notion of testing gives rise to a *may-testing preorder* $\sqsubseteq_m$ and to a *may-testing equivalence* $\approx_m$ on the set of closed extended processes.

Definition 3 (may-testing equivalence). *Let A and B be two closed extended processes such that* $\mathrm{dom}(A) = \mathrm{dom}(B)$, *$A \sqsubseteq_m B$ if for any test (C, c) such that C is a closing evaluation context for A (and B) we have that $C[A]\Downarrow c$ implies that $C[B]\Downarrow c$. Two closed extended processes A and B are in* may-testing equivalence, *denoted by $A \approx_m B$, if $A \sqsubseteq_m B$ and $B \sqsubseteq_m A$.*

The idea of may-testing equivalence comes from the work of R. De Nicola and M. Hennessy [46]. Our definition is similar to their notion of may-testing equivalence. Since then, this notion of may-testing equivalence has been used in several cryptographic calculi, *e.g.* spi-calculus [4]. This notion is usually shown to be equivalent to a notion of trace equivalence (as the one given in Definition 2) that is easier to manipulate since the universal quantification over all contexts has been removed (see [34] for a proof of this result for the spi-calculus). We study the relation between these two relations in Section 4.

2.3.3. Observational equivalence

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

Definition 4 (observational equivalence). Observational equivalence *is the largest symmetric relation $\mathcal{R}$ between closed extended processes with the same domain such that $A \mathcal{R} B$ implies:*

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

However, proofs of observational equivalences are difficult because of the universal quantification over all contexts. Therefore, an alternative definition has been proposed, considering labeled transitions for the processes.

Definition 5 (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 $B \overset{\epsilon}{\Rightarrow} B'$ and $A' \mathcal{R} B'$ for some B',*
3. *if $A \overset{\ell}{\mapsto} A'$ and $bn(\ell) \cap fn(B) = \emptyset$ then $B \overset{\ell}{\Rightarrow} B'$ and $A' \mathcal{R} B'$ for some B'.*

Example 7. *Consider the theory $\mathsf{E}_{\mathsf{aenc}}$ and the two processes $P_a = \mathtt{out}(c, \mathsf{aenc}(a, \mathsf{pk}(sk)))$ and $P_b = \mathtt{out}(c, \mathsf{aenc}(b, \mathsf{pk}(sk)))$. 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 6.*

It has been shown that observational equivalence coincides with labeled bisimilarity [2].

Theorem 1 ([2]). *Two processes A and B are observationally equivalent if and only if they are labeled bisimilar ($A \approx B$).*

We therefore adopt the same notation ($\approx$) for both relations.

3. Relating observational and trace equivalences

Observational equivalence has been initially introduced as a mean for proving trace equivalence. In this section, we discuss the relation between these two notions.

It is easy to see that observational equivalence (or labeled bisimilarity) implies trace equivalence while the converse is false in general (see Example 8).

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. $\square$

Example 8. *Consider the two following processes:*

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

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.

J. Engelfriet has shown that observational and trace equivalence coincide for a process algebra with atomic actions (processes are only allowed to input and output names) when processes are *determinate* [36]. In the remaining of the section, we show a similar result (Section 3.1) in the context of the applied-pi calculus (where actions are no longer atomic) and we identify a large family of processes, called *simple processes*, that enjoy the determinacy property (Section 3.2).

3.1. Determinacy

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

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 5). 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. □

Example 8 shows that trace and observational equivalences do not always coincide. But it is important to notice that such an example makes use of a non determinate process.

Example 9. *Consider the closed extended process A given in Example 8. We have that $A \overset{\tau}{\mapsto} A_1$ and $A \overset{\tau}{\mapsto} A_2$ for A_1 and A_2 given below:*

$$\begin{aligned} A_1 &= \nu c'.(\mathtt{out}(c,a).\mathtt{out}(c,b_1) \mid \mathtt{in}(c',x).\mathtt{out}(c,a).\mathtt{out}(c,b_2)) \\ A_2 &= \nu c'.(\mathtt{in}(c',x).\mathtt{out}(c,a).\mathtt{out}(c,b_1) \mid \mathtt{out}(c,a).\mathtt{out}(c,b_2)). \end{aligned}$$

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.

We extend the result of J. Engelfriet [36] to processes of the applied pi calculus, showing that observational equivalence and trace equivalence coincide when processes are determinate.

Theorem 2. *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 5). 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. □

3.2. An expressive class of determinate processes

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. Our class of simple processes is close to the fragment of processes considered in [26] for proving cryptographic indistinguishability using observational equivalence. However, the fragment of [26] does not enjoy the determinacy property (since it was not designed for it).

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 J. Clark and J. Jacob library [24] 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 $B_1, B_2 \in \mathcal{B}(c, \mathcal{V})$, $M, N \in \mathcal{T}(\mathcal{N}_b, \mathcal{V})$, then*
$$\texttt{if } M = N \texttt{ then } B_1 \texttt{ else } B_2 \in \mathcal{B}(c, \mathcal{V}).$$
- *if $B \in \mathcal{B}(c, \mathcal{V})$, $u \in \mathcal{T}(\mathcal{N}_b, \mathcal{V})$, then $\texttt{out}(c, u).B \in \mathcal{B}(c, \mathcal{V})$.*
- *if $B \in \mathcal{B}(c, \mathcal{V} \uplus \{x\})$, x of base type ($x \notin \mathcal{V}$), then $\texttt{in}(c, x).B \in \mathcal{B}(c, \mathcal{V})$.*

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 10. *We consider 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:*

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

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: $\mathsf{aenc}(N_b, \mathsf{pub}(B))$. 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 $M = \mathsf{adec}(x, ska)$. 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.

$$\begin{array}{rl} A(a, b) \stackrel{\mathit{def}}{=} & \texttt{out}(c_A, \mathsf{aenc}(\langle n_a, \mathsf{pk}(ska)\rangle, \mathsf{pk}(skb))). \\ & \texttt{in}(c_A, x). \\ & \texttt{if } \langle \pi_1(M), \pi_2(\pi_2(M))\rangle = \langle n_a, \mathsf{pk}(skb)\rangle \texttt{ then } 0 \texttt{ else } 0. \end{array}$$

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

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

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:*

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

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 $P_1 = \mathtt{out}(c,a).\mathtt{out}(c,a).0$ while two roles emitting each the constant a would be modeled by $P_2 = \mathtt{out}(c,a).0 \mid \mathtt{out}(c,a).0$. Then P_1 and P_2 are observationally equivalent while the two protocols could be distinguished in practice (since they would use different IP adresses for example), which is reflected in our modeling in simple processes.

Example 11. *Continuing Example 10, a simple process representing an unbounded number of sessions in which a plays A (with b) and b plays B with a is:*

$$\nu ska, skb. \quad (\; !(\nu n_a, c_A.\mathtt{out}(p_A, c_A).A(a,b)) \mid !(\nu n_b, c_B.\mathtt{out}(p_B, c_B).B(b,a)) \;)$$

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 and a sequence of actions tr, there is a unique process B (up to some internal reduction steps) such that $A \overset{\mathsf{tr}}{\Mapsto} B$.

Theorem 3. *Any simple process is determinate.*

Before proving Theorem 3, we introduce some definitions and notations. Each process of the form $!(\nu c, \tilde{n}.\mathtt{out}(p,c).B)$ 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 $A \overset{s(w)}{\Mapsto} B$ 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 basic processes such that either $B_i = B'_i$ or $B_i \overset{\epsilon}{\Mapsto} B'_i$ or $B'_i \overset{\epsilon}{\Mapsto} B_i$. It is easy to check that $\leftrightarrow$ is an equivalence relation on simple processes.

We are now ready to prove Theorem 3.

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

Assume that $P \overset{w}{\mapsto} P_1 \overset{\alpha}{\mapsto} P'_1$ and $P \overset{s(w.\alpha)}{\Mapsto} P'_2$ 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 basic processes such that either $B_i = B'_i$ or $B_i \overset{\epsilon}{\Mapsto} B'_i$ or $B'_i \overset{\epsilon}{\Mapsto} B_i$. 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 $P_2 \overset{\epsilon}{\Mapsto} P'_2$, thus $P_2 \leftrightarrow P'_2$. Since $\leftrightarrow$ is an equivalence relation, we deduce $P'_1 \leftrightarrow P'_2$.

Case $\alpha = \nu c.\mathtt{out}(p, c)$ *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 name for a fresh instance of the role p. Let $R_i \equiv !(\nu c, \tilde{m} \cdot \mathtt{out}(p, c).B)$ 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 $P''_2 \overset{\epsilon}{\Mapsto} P'_2$, we also have $P''_2 \leftrightarrow P'_2$ thus $P'_1 \leftrightarrow P'_2$.

Case $\alpha = \nu x.\mathtt{out}(c, x)$ *where* x *is a variable.* Let B_i the basic process such that $B_i = \mathtt{out}(c, s).B''_i$. 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 $\alpha = \mathtt{in}(c, M)$. Let B_i the basic process such that $B_i = \mathtt{in}(c, x).B''_i$. 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 $P'_1 \leftrightarrow P'_2$. □

Applying Theorems 2 and 3, we get that, on simple processes, observational equivalence and trace equivalence coincide. Note that these results hold for any signature and equational theory.

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

4. Relating may-testing and trace equivalences

May-testing and trace equivalences are usually considered as equivalent. Actually, this result has been proved in two variants of the spi-calculus [42, 34] and is in the spirit of the main theorem stated in [2]. We tried to adapt their proof, but it happens that the two notions do not coincide in the applied pi calculus setting. We indeed exhibit two processes that are may-testing equivalent but not trace equivalent (Section 4.2). However, these two notions are similar and try to capture the same concept. We show in Section 4.1 that trace equivalence implies may-testing equivalence (without any additional restriction) and the other implication holds as soon as we consider processes without replication (Section 4.3). More precisely, this implication holds for processes that are image-finite (up to static equivalence).

4.1. Trace equivalence implies may-testing equivalence

Trace equivalence implies may-testing equivalence in general.

Theorem 4. *Let A and B two closed extended processes. We have that:*

$$A \approx_t B \text{ implies that } A \approx_m B.$$

Proof (sketch). The proof of Theorem 4 is quite technical as it requires to characterize any evolution of the contextualized process A, that is, any evolution of $C[A]$.

Given a closed extended process A and an evaluation context $C[_]$, the proposition below allows us to map a derivation issued from $C[A]$ to a labeled trace issued from A. Note that we only consider evaluation contexts having a special shape (there is no restriction in front of the hole). This additional assumption is actually fulfilled when proving Theorem 4.

Proposition 1. *Let A and B be two closed extended process with* $\mathrm{dom}(A) = \mathrm{dom}(B)$*, and $C[_] = \nu\tilde{n}.(D \mid _)$ be an evaluation context closing for A. If $C[A] \mapsto^* A''$ for some process A'', then there exist a closed extended process A', an evaluation context $C' = \nu\tilde{n'}.(D' \mid _)$ closing for A', and a trace $\mathsf{tr} \in (\mathcal{A} \smallsetminus \{\tau\})^*$ such that $A'' \equiv C'[A']$, $A \overset{\mathsf{tr}}{\Rightarrow} A'$, and for all closed extended process B', we have that:*

$$B \overset{\mathsf{tr}}{\Rightarrow} B' \text{ and } \phi(B') \sim \phi(A') \text{ implies that } C[B] \mapsto^* C'[B'].$$

Proposition 1 is proved in Appendix A.

The proof of Theorem 4 then works intuitively as follows. Assume that A and B are trace equivalent. We have to show that they are may-testing equivalent. Assume $C[A] \mapsto^* A''$ for some process A'', then by Proposition 1, there exist a closed extended process A', an evaluation context C' closing for A', and a trace $\mathsf{tr} \in (\mathcal{A} \smallsetminus \{\tau\})^*$ such that $A'' \equiv C'[A']$, $A \overset{\mathsf{tr}}{\Rightarrow} A'$. Since A and B are trace equivalent, we know that $B \overset{\mathsf{tr}}{\Rightarrow} B'$ with $\phi(B') \sim \phi(A')$. Proposition 1 allows then to conclude that $C[B] \mapsto^* C'[B']$. This holds provided that C is of the form $\nu\tilde{n}.(D \mid _)$. How contexts not of this form are handled is described in Appendix A.

It then remains to show that $C'[A']$ and $C'[B']$ emit on the same channels. This is proved by distinguishing two cases. Either C' can emit on some channel c, in which case both $C'[A']$ and $C'[B']$ may emit on c. Or A' may emit on c, in which case B' may emit on c too, using trace equivalence. □

4.2. May-testing equivalence does not imply trace equivalence

While testing and trace equivalences seem to be two very similar notions, may-testing equivalence actually does not imply trace equivalence. Roughly, in may-testing equivalence, the attacker is allowed to perform a sequence of inputs/outputs followed by a finite number of tests, but he has to commit on these tests before knowing how the communication actions will be used. In trace equivalence, all the tests are considered through static equivalence. The attacker does not have to commit on the tests he wants to use in advance. So, the attacker is more adaptive in trace equivalence and this gives him more power.

We describe two processes A and B such that $A \not\approx_t B$ whereas $A \approx_m B$. Let A and B be the two following processes:

- $A = \nu b.\nu c_1.\big(\mathtt{out}(c_1, \mathit{token}) \mid \mathtt{in}(c_1, x).\mathtt{out}(c, b) \mid \mathtt{in}(c_1, x).B\big)$; and
- $B = \nu c_2.\big(\mathtt{out}(c_2, \mathsf{h}(a)) \mid \mathtt{in}(c_2, x).\mathtt{out}(c, x) \ \mid !\mathtt{in}(c_2, x).\mathtt{out}(c_2, \mathsf{h}(x))\big)$.

We consider the empty equational theory and a signature which only contains the symbol h of arity 1 (with no equation). The private channel c_1 in A is used to model the choice operator. The behavior of A consists of outputting a fresh name b or executing the process B. The processes B relies also on a private channel, namely c_2, whose purpose is to model the choice operator. An

execution of B will output a message of the form $\mathsf{h}(\mathsf{h}(\ldots\mathsf{h}(a)\ldots))$ on the public channel c, for an arbitrary number of h. In the remaining, we denote by $\mathsf{h}^n(a)$ the term obtained by applying n times h on top of a.

Regarding trace equivalence, we have that:

- $\mathsf{trace}(A) = \{(\epsilon, 0);\ (\mathsf{tr}, \nu b.\{b/_x\});\ (\mathsf{tr}, \{\mathsf{h}(a)/_x\});\ \ldots;\ (\mathsf{tr}, \{\mathsf{h}^n(a)/_x\});\ \ldots\}$
- $\mathsf{trace}(B) = \{(\epsilon, 0);\ (\mathsf{tr}, \{\mathsf{h}(a)/_x\});\ \ldots;\ (\mathsf{tr}, \{\mathsf{h}^n(a)/_x\});\ \ldots\}$

with $\mathsf{tr} = \nu x.\mathtt{out}(c, x)$. We can easily see that $A \not\approx_t B$: the frame $\nu b.\{b/_x\}$ is not statically equivalent to any other frame in $\mathsf{trace}(B)$.

On the other hand, we have that $A \approx_m B$. This is a non trivial equivalence that is not easy to prove. Below, we only give an intuition. First, it seems clear that $B \sqsubseteq_m A$ since A can easily mimic the process B by performing first an internal communication and giving the token to the last part of the process. Regarding the other inclusion, consider a test (C, c_0) such that C is a closing evaluation context for A. The most interesting case is when $C[A]$ outputs the fresh name b on the channel c and then performs some internal actions until it is ready to emit on c_0. In such a case, we have that:

$$C[A] \rightarrow \xrightarrow{\nu x.\mathtt{out}(c,x)} C[\nu b.\nu c_1.(\{b/_x\} \mid \mathtt{in}(c_1, x).B)] \rightarrow \ldots \rightarrow C_A[\mathtt{out}(c_0, M_A)]$$

for some term M_A and some evaluation context C_A that does not bind c_0. However, once the execution trace is fixed, we know the tests that have been perfomed along this derivation, and we can compute n_0 such that the same tests will be satisfied if the outputted message was $\mathsf{h}^{n_0}(a)$ instead of b (for instance, we can choose for n_0 the number of occurences of h in the tests plus one). Typically, those tests will allow one to check that the value stored in x (i.e. b here) is different from $\mathsf{h}^{n_1}(a)$, ..., $\mathsf{h}^{n_k}(a)$ for some integers $n_1 \leq, \ldots \leq n_k$. Regarding the process $C[B]$, we have to trigger sufficiently many internal communications so that the term outputted on the public channel c will be of the form $\mathsf{h}^{n_0}(a)$ with $n_0 \geq n_k$, and thus will allow the resulting process $C[\nu c_2.(\{\mathsf{h}^{n_0}(a)/_x\} \mid !\mathtt{in}(c_2, x).\mathtt{out}(c_2, \mathsf{h}(x)))]$ to pass all the tests and emit on c_0. We have that:

$$C[B] \rightarrow \ldots \rightarrow \xrightarrow{\nu x.\mathtt{out}(c,x)} C[\nu c_2.(\{\mathsf{h}^{n_0}(a)/_x\} \mid !\mathtt{in}(c_2, x).\mathtt{out}(c_2, \mathsf{h}(x)))] \rightarrow \ldots \rightarrow C_B[\mathtt{out}(c_0, M_B)]$$

for some term M_B and some evaluation context C_B that does not bind c_0.

4.3. May-testing equivalence implies trace equivalence for image-finite processes

As illustrated in Section 4.2, may-testing equivalence does not imply trace equivalence in general. However, the implication holds as soon as we consider processes without replication. More precisely, this implication holds for processes that are image-finite (up to static equivalence).

Definition 9 (image-finite). *An extended process A is* image-finite *if for each trace* tr*, the set of equivalence classes $\{\phi(A') \mid A \stackrel{\mathsf{tr}}{\Rightarrow} A'\}/\sim$ is finite.*

Theorem 5. *Let A and B two closed extended processes with* $\mathrm{dom}(A) = \mathrm{dom}(B)$ *and such that A and B are image-finite. We have that:*

$$A \approx_m B \text{ implies that } A \approx_t B$$

Proof. Assume that $A \not\approx_t B$. We assume w.l.o.g. that $A \not\sqsubseteq_t B$. In such a case, there exists a witness for the non equivalence. This means that there exists $(\mathsf{tr}, \phi) \in \mathsf{trace}(A)$ such that $bn(\mathsf{tr}) \cap fn(B) = \emptyset$, and

1. either there does not exist ϕ' such that $(\mathsf{tr}, \phi') \in \mathsf{trace}(B)$;
2. or for all $(\mathsf{tr}, \phi') \in \mathsf{trace}(B)$, we have that $\phi' \not\sim \phi$.

Moreover, we assume that no name in tr is bound twice (*i.e.* $\nu a.$ can not occur twice in tr) and bound names in tr are distinct from free names that occur in A, B, and tr.

We build an evaluation context C according to the trace tr and also the tests that witness the fact that static equivalence does not hold. Let $S_{\mathsf{tr}} = \{\phi' \mid (\mathsf{tr}, \phi') \in \mathsf{trace}(B)\}$. Since B is image-finite, we know that $S/\sim$ is finite. Let $\{\phi'_1, \ldots, \phi'_m\} = S/\sim$. If we are in the first case, *i.e.* $S = \emptyset$, we have that $m = 0$.

We know that $\{1, \ldots, m\} = T^+ \uplus T^-$ with:

- for each $i \in T^+$, there exist two terms M_i and N_i such that $fv(M_i) \cup fv(N_i) \subseteq \mathrm{dom}(\phi)$, $(M_i =_{\mathsf{E}} N_i)\phi$, and $(M_i \neq_{\mathsf{E}} N_i)\phi'_i$; and
- for each $i \in T^-$, there exist two terms M_i and N_i such that $fv(M_i) \cup fv(N_i) \subseteq \mathrm{dom}(\phi)$, $(M_i \neq_{\mathsf{E}} N_i)\phi$, and $(M_i =_{\mathsf{E}} N_i)\phi'_i$.

Let bad be a fresh channel name that does not occur in A and B. Let $P_1, \ldots, P_m, P_{m+1}$ be the plain processes defined as follows:

- $P_{m+1} \stackrel{\mathsf{def}}{=} \mathtt{out}(\mathsf{bad}, \mathsf{bad}).0$
- for $1 \leq i \leq m$, we define P_i as follows:

$$P_i \stackrel{\mathsf{def}}{=} \mathtt{if}\ M_i = N_i\ \mathtt{then}\ P_{i+1}\ \mathtt{else}\ 0 \qquad \text{when } i \in T^+$$
$$P_i \stackrel{\mathsf{def}}{=} \mathtt{if}\ M_i = N_i\ \mathtt{then}\ 0\ \mathtt{else}\ P_{i+1} \qquad \text{when } i \in T^-$$

Let $\{a_1, \ldots, a_k\}$ be channel names that occur free in A, B, and tr. Let $\mathcal{X}^0_{ch} = \{x_{a_1}, \ldots, x_{a_k}\}$ be a set of variables of channel type, and $\sigma = \{x_{a_1} \mapsto a_1, \ldots, x_{a_k} \mapsto a_k\}$. We define C such that $C = Q(\mathsf{tr}, \mathcal{X}^0_{ch}) \mid _$ where $Q(\mathsf{tr}, \mathcal{X}_{ch})$ is defined by recurrence on tr as follows:

- if $\mathsf{tr} = \epsilon$ then $Q(\mathsf{tr}, \mathcal{X}_{ch}) = P_1$;
- if $\mathsf{tr} = \mathtt{in}(a, M).\mathsf{tr}'$ then $Q(\mathsf{tr}, \mathcal{X}_{ch}) = \mathtt{out}(x_a\sigma, M).Q(\mathsf{tr}', \mathcal{X}_{ch})$;
- it $\mathsf{tr} = \mathtt{out}(a, c).\mathsf{tr}'$ then $Q(\mathsf{tr}, \mathcal{X}_{ch}) = \mathtt{in}(x_a\sigma, y).\ \mathtt{if}\ y = x_c\sigma\ \mathtt{then}\ Q(\mathsf{tr}', \mathcal{X}_{ch})\ \mathtt{else}\ 0$ where y is fresh variable of channel type; and
- if $\mathsf{tr} = \nu c.\mathtt{out}(a, c)$ then $Q(\mathsf{tr}, \mathcal{X}_{ch}) = \mathtt{in}(x_a\sigma, x_c).\ \mathtt{if}\ x_c \in \mathcal{X}_{ch}\sigma\ \mathtt{then}\ 0\ \mathtt{else}\ Q(\mathsf{tr}', \mathcal{X}'_{ch})$ where $\mathcal{X}'_{ch} = \mathcal{X}_{ch} \uplus \{x_c\}$.

We use the conditional $\mathtt{if}\ u \in \{u_1, \ldots, u_k\}\ \mathtt{then}\ 0\ \mathtt{else}\ P$ as a shortcut for

$$\mathtt{if}\ u = u_1\ \mathtt{then}\ 0\ \mathtt{else}\ (\mathtt{if}\ u = u_2\ \mathtt{then}\ 0\ \mathtt{else}\ (\ldots.\ (\mathtt{if}\ u = u_k\ \mathtt{then}\ 0\ \mathtt{else}\ P))).$$

We can see that $C[A]\Downarrow_{\mathsf{bad}}$ since $(\mathsf{tr}, \phi) \in \mathsf{trace}(A)$ and ϕ satisfies by definition all the tests that are tested in P_m. However, by construction of C, we have that $C[B]$ can not emit on bad. $\square$

Part II
Towards deciding trace equivalence

Several procedures for accessibility-based properties rely on solving constraint systems, as introduced by J. Millen & V. Shmatikov and subsequently H. Comon-Lundh & V. Shmatikov [43, 28].

In this part, we provide an alternative framework for reasoning about trace equivalence, based on constraint systems. Namely, we show that, for process without replication, checking for trace equivalence amounts into checking equivalence of sets of constraint systems. The advantage of reasoning directly with constraint systems is to get rid of the process algebra (with non determinism, name passing, restriction, etc.). To illustrate the advantage of our approach, we show how to apply two existing decidability results for constraint systems to obtain two decidability results for trace equivalence:

- Trace equivalence is decidable for determinate processes without replication and with trivial else branches, for any subterm convergent theory. This result relies the decision procedure developed by M. Baudet in [10] for deciding S-equivalence between two constraint systems.
- Trace equivalence is decidable for processes with non-trivial else branches and without replication, for the standard theory of symmetric and asymmetric encryption, signatures and hash. This result was first shown in [20], in a slightly different framework (not the applied pi calculus).

We first introduce an intermediate calculus (Section 5) and then our symbolic calculus based constraint systems (Section 6), for mapping step-by-step trace equivalence to equivalence between sets of constraint systems (Sections 7.1 and 7.2). We illustrate our approach by retrieving (and unifying) two decidability results (Section 7.3).

5. Intermediate calculus

Reasoning on processes of the applied pi calculus is quite involved. In particular, as mentioned in [31], the semantics of the applied pi calculus is not well-suited for defining a symbolic semantics. Thus we use the class of *intermediate processes*, defined in [31], that is easier to manipulate. These intermediate processes are a selected (but sufficient) subset of the original processes. One may think of them as being processes in some kind of normal form. In [31], it has been shown that observational equivalence of processes without replication is equivalent to observational equivalence of their corresponding intermediate processes. In a similar way, it is quite easy to establish this result for trace equivalence.

In this section, we review this class of intermediate processes by using a slightly different syntax. Moreover, we give a non-compositional semantics that is easier to manipulate than its compositional counterpart as defined in [31].

5.1. Syntax

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

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

where u is a term of channel type, M_1, M_2 are terms having the same type, x is a variable, N is a term, and $\mathcal{P}$ (resp.$\mathcal{Q}$) is a multiset of plain intermediate processes. Terms M_1, M_2 and N can also use variables. For sake of clarity, when the multiset $\mathcal{P}$ is a singleton, *i.e.* $\mathcal{P} = \{P\}$, we write P instead of $\{P\}$.

Definition 10 (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 *where null processes are removed and such that $fv(\mathcal{P}) = \emptyset$.*

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 and such that $\mathrm{dom}(A) = \{w_1, \ldots, w_n\}$ for some n, we can easily transform it into an intermediate process $\tilde{A} = (\mathcal{E}; \mathcal{P}; \Phi)$ such that "$A \approx \nu\mathcal{E}.(\mathcal{P} \mid \Phi)$". More formally, we can show the following lemma whose proof can be found in Appendix B.

Proposition 2. *Let A be a closed extended process. There exists a plain process P that does not contain name restriction, and a context evaluation C that only contains name restrictions, parallel compositions and active substitutions such that $P \approx C[A]$.*

The idea of the transformation is to rename names and variables to avoid clashes, to apply the active substitutions, to remove the restrictions on variables, 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 12. *Consider the extended process A described below (M is a term such that $n \notin fn(M)$) and* $\mathrm{dom}(A) = \emptyset$.

$$A = \nu sk.\nu x.(\mathtt{out}(c, \mathsf{aenc}(x, \mathsf{pk}(sk))).\nu n.\mathtt{out}(c, n) \mid \{^{M}/_{x}\}).$$

An intermediate process A' associated to A is:

$$A' \;=\; (\mathcal{E}; \mathcal{P}; \Phi) \;=\; (\{sk, n\}; \mathtt{out}(c, \mathsf{aenc}(M, \mathsf{pk}(sk))).\mathtt{out}(c, n); \emptyset).$$

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.

Consider the extended process $B = \nu s.\nu sk.(\mathtt{out}(c_2, s) \mid \{^{\mathsf{aenc}(s, \mathsf{pk}(sk))}/_{w_1}\})$. An intermediate process B' associated to B is:

$$B' \;=\; (\{s, sk\}; \mathtt{out}(c_2, s); \{w_1 \rhd \mathsf{aenc}(s, \mathsf{pk}(sk))\}).$$

5.2. Semantics

The semantics for intermediate processes is given in Figure 3. 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 $s \in (\mathcal{A}_i \setminus \{\tau\})^*$, the relation $\overset{s}{\Rightarrow}_i$ on intermediate processes is defined by: $A \overset{s}{\Rightarrow}_i B$ 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.

$$(\mathcal{E}; \{\texttt{if } u = v \texttt{ then } \mathcal{Q}_1 \texttt{ else } \mathcal{Q}_2\} \uplus \mathcal{P}; \Phi) \xrightarrow{\tau}_i (\mathcal{E}; \mathcal{Q}_1 \uplus \mathcal{P}; \Phi) \text{ if } u =_{\mathsf{E}} v \qquad (\textsc{Then}_i)$$

$$(\mathcal{E}; \{\texttt{if } u = v \texttt{ then } \mathcal{Q}_1 \texttt{ else } \mathcal{Q}_2\} \uplus \mathcal{P}; \Phi) \xrightarrow{\tau}_i (\mathcal{E}; \mathcal{Q}_2 \uplus \mathcal{P}; \Phi) \text{ if } u \neq_{\mathsf{E}} v \qquad (\textsc{Else}_i)$$

$$(\mathcal{E}; \{\texttt{out}(p, u).\mathcal{Q}_1; \texttt{in}(p, x).\mathcal{Q}_2\} \uplus \mathcal{P}; \Phi) \xrightarrow{\tau}_i (\mathcal{E}; \mathcal{Q}_1 \uplus \mathcal{Q}_2\{x \mapsto u\} \uplus \mathcal{P}; \Phi) \qquad (\textsc{Comm}_i)$$

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

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

$$(\mathcal{E}; \{\texttt{out}(p, c).\mathcal{Q}\} \uplus \mathcal{P}; \Phi) \xrightarrow{\texttt{out}(p,c)}_i (\mathcal{E}; \mathcal{Q} \uplus \mathcal{P}; \Phi) \qquad (\textsc{Out-Ch}_i)$$
$$\text{if } p, c \notin \mathcal{E}$$

$$(\mathcal{E}; \{\texttt{out}(p, c).\mathcal{Q}\} \uplus \mathcal{P}; \Phi) \xrightarrow{\nu ch_n.\texttt{out}(p,ch_n)}_i (\mathcal{E}; (\mathcal{Q} \uplus \mathcal{P})\{c \mapsto ch_n\}; \Phi) \qquad (\textsc{Open-Ch}_i)$$
$$\text{if } p \notin \mathcal{E},\ c \in \mathcal{E},\ ch_n \text{ is a fresh channel name}$$

where p, c are channel names, u, v are ground terms, and x is a variable.

Figure 3: Intermediate semantics

5.3. Equivalence

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

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

Two intermediate processes are in (intermediate trace) equivalence if for any trace, the two corresponding frames are statically equivalent.

Definition 11 ($\approx_t$ 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 (see Proposition 3) that the two notions of trace equivalence coincide (this is done in [31] for the more involved notion of observational equivalence).

Proposition 3. *Let A and B be two processes without replication. Consider the two associated intermediate processes: $(\mathcal{E}; \mathcal{P}_A; \Phi_A)$ and $(\mathcal{E}; \mathcal{P}_B; \Phi_B)$.*

*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, $(\mathcal{E}; \mathcal{P}_A; \Phi_A) \approx_t (\mathcal{E}; \mathcal{P}_B; \Phi_B)$.*

Example 13. *Continuing Example 10, we have that $(\mathcal{E}; \{B(b, a)\}; \Phi_0) \approx_t (\mathcal{E}; \{B(b, a')\}; \Phi_0)$ where:*

- *$\mathcal{E} = \{ska, ska', skb, nb\}$; and*
- *$\Phi_0 = \{w_1 \rhd \mathsf{pk}(ska), w_2 \rhd \mathsf{pk}(ska'), w_3 \rhd \mathsf{pk}(skb)\}$.*

This is a non-trivial equivalence that illustrates the anonymity property. Intuitively, the 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 $B(b, a')$). For modelling and verification purposes, we may want to disclose the public keys in order to make them available to the attacker. This is done by means of the frame Φ_0.

However, the "decoy" message plays an important role. Indeed, considering now the process $B^-(b, a)$, we have that: $(\mathcal{E}; \{B^-(b, a)\}; \Phi_0) \not\approx_t (\mathcal{E}; \{B^-(b, a')\}; \Phi_0)$. This can be easily shown

by considering the sequence of actions $s = \mathtt{in}(c, \mathsf{aenc}(\langle n_i, w_1\rangle, w_3)) \cdot \mathtt{out}(c, w_4)$. *We have that* $(s, \nu\mathcal{E}.\Phi_1) \in \mathsf{trace}_i((\mathcal{E}; \{B^-(b,a)\}; \Phi_0))$. *However, this sequence* s *does not exist for the process* $(\mathcal{E}; \{B^-(b,a')\}; \Phi_0)$.

6. Symbolic calculus

In this section, we propose a symbolic semantics for our calculus following *e.g.* [11]. By treating inputs symbolically, our symbolic semantics avoids potentially infinite branching of execution trees due to inputs from the environment. Correctness is maintained by associating with each process a set of constraints on terms.

6.1. Constraint system

Following the notations of [10], we consider a new set $\mathcal{X}^2$ of variables called *second order variables* $X, Y, \ldots$, each variable with an arity, denoted $ar(X)$.

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

Definition 12 (constraint system). *A* constraint system *is a triple* $(\mathcal{E}; \Phi; \mathcal{D})$*:*

- $\mathcal{E}$ *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.*
- $\mathcal{D}$ *is a set of constraints of the form* $X \rhd^? x$ *with* $ar(X) \leq n$*, or of the form* $s =^?_{\mathsf{E}} s'$ *or* $s \neq^?_{\mathsf{E}} s'$ *where* s, s' *are first-order terms of same type. 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. Given a set $\mathcal{D}$ of constraints, we denote by $\mathsf{var}^1(\mathcal{D})$ (resp. $\mathsf{var}^2(\mathcal{D})$) the first order (resp. second order) variables of $\mathcal{D}$, that is $\mathsf{var}^1(\mathcal{D}) = fv(\mathcal{D}) \cap \mathcal{X}$ (resp. $\mathsf{var}^2(\mathcal{D}) = fv(\mathcal{D}) \cap \mathcal{X}^2$).

We also assume the following conditions are satisfied on a constraint system:

1. for every $x \in \mathsf{var}^1(\mathcal{D})$, there exists a unique X such that $(X \rhd^? x) \in \mathcal{D}$, and each variable X occurs at most once in $\mathcal{D}$.
2. for every $1 \leq k \leq n$, for every $x \in \mathsf{var}^1(t_k)$, there exists $(X \rhd^? x) \in \mathcal{D}$ 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 a constraint system $\mathcal{C} = (\mathcal{E}; \Phi; \mathcal{D})$ is given by $\mathcal{E}$, $|\Phi|$ and $\mathsf{var}^2(\mathcal{D})$ with their arity. A *positive constraint system* is a constraint system that does not contain any constraint of the form $s \neq^?_{\mathsf{E}} s'$.

Example 14. *The triple* $\mathcal{C}_1 = (\mathcal{E}; \Phi_0 \cup \{w_4 \rhd t\}; \mathcal{D}_1)$ *where*

- $\mathcal{E} = \{ska, ska', skb, n_b\}$,
- $\Phi_0 = \{w_1 \rhd \mathsf{pk}(ska), w_2 \rhd \mathsf{pk}(ska'), w_3 \rhd \mathsf{pk}(skb)\}$,
- $t = \mathsf{aenc}(\langle \pi_1(\mathsf{adec}(y, skb)), \langle n_b, \mathsf{pk}(skb)\rangle\rangle, \mathsf{pk}(ska))$,
- $\mathcal{D}_1 = \{Z_1 \rhd^? z_1; Y \rhd^? y; z_1 =^?_{\mathsf{E}} c_B; \pi_2(\mathsf{adec}(y, skb)) =^?_{\mathsf{E}} \mathsf{pk}(ska); Z_2 \rhd^? z_2; z_2 =^?_{\mathsf{E}} c_B\}$ *with* $ar(Z_1) = ar(Z_2) = ar(Y) = 3$.

is a constraint system. We will see that it corresponds to an execution of the process $B(b,a)$ *(and* $B^-(b,a)$*) presented in Example 10.*

Definition 13 (solution). *A* solution *of a constraint system* $\mathcal{C} = (\mathcal{E}; \Phi; \mathcal{D})$ *is a substitution* θ *such that*

- $\mathrm{dom}(\theta) = \mathsf{var}^2(\mathcal{C})$, *and*
- $X\theta \in \mathcal{T}(\mathcal{N} \smallsetminus \{\mathcal{E}\}, \{w_1, \ldots, w_k\})$ *for any* $X \in \mathrm{dom}(\theta)$ *with* $ar(X) = k$.

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

1. *for every* $(X \rhd^? x) \in \mathcal{D}$, *we have that* $(X\theta)(\Phi\lambda) = x\lambda$;
2. *for every* $(s =^?_{\mathsf{E}} s') \in \mathcal{D}$, *we have that* $s\lambda =_{\mathsf{E}} s'\lambda$;
3. *for every* $(s \neq^?_{\mathsf{E}} s') \in \mathcal{D}$, *we have that* $s\lambda \neq_{\mathsf{E}} s'\lambda$.

The substitution λ *is called the* first-order solution *of* $\mathcal{C}$ *associated to* θ. *The set of solutions of a constraint system* $\mathcal{C}$ *is denoted* $\mathsf{Sol}(\mathcal{C})$. *A constraint system* $\mathcal{C}$ *is* satisfiable *if* $\mathsf{Sol}(\mathcal{C}) \neq \emptyset$.

Intuitively, in the preceding definition the substitution θ stores the computation done by the adversary in order to compute the messages he sends (stored in λ) during the execution.

Example 15. *Continuing Example 14, a solution to* $\mathcal{C}_1 = (\mathcal{E}_1; \Phi_1; \mathcal{D}_1)$ *is* θ *where* $\mathrm{dom}(\theta) = \{Z_1, Z_2, Y\}$, $\theta(Z_1) = \theta(Z_2) = c_B$, *and* $\theta(Y) = \mathsf{aenc}(\langle n_i, w_1\rangle, w_3)$ *with* n_i *a public name of base type (*i.e. $n_i \notin \mathcal{E}_1$*). The first-order solution* λ *of* $\mathcal{C}_1$ *associated to* θ *is a substitution whose domain is* $\{z_1, z_2, y\}$ *and such that* $\lambda(z_1) = \lambda(z_2) = c_B$, *and* $\lambda(y) = \mathsf{aenc}(\langle n_i, \mathsf{pk}(ska)\rangle, \mathsf{pk}(skb))$.

6.2. Syntax and semantics

From an intermediate process $(\mathcal{E}; \mathcal{P}; \Phi)$, we can compute the set of constraint systems capturing its possible executions, starting from the symbolic process $(\mathcal{E}; \mathcal{P}; \Phi; \emptyset)$ and applying the rules defined in Figure 4.

Definition 14 (symbolic process). *A* symbolic process *is a tuple* $(\mathcal{E}; \mathcal{P}; \Phi; \mathcal{D})$ *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* $fv(\mathcal{P}) \subseteq \{x \mid X \rhd^? x \in \mathcal{D}\}$;
- $(\mathcal{E}, \Phi, \mathcal{D})$ *is a constraint system.*

The rules of Figure 4 define the semantics of symbolic processes. This relation transforms a symbolic process into a symbolic process. 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 $\textsc{Then}_s$ (resp. $\textsc{Else}_s$) rule allows the process to pass a conditional. The corresponding constraint is added in the set of constraints $\mathcal{D}$. When a process is ready to input a term on a public channel p, a deduction constraint is added in the set $\mathcal{D}$ (rule $\textsc{In}_s$). When a process is ready to output a term u on a public channel p, the outputted term is added to the frame Φ (rule $\textsc{Out-T}_s$), which means that this term is made available to the attacker. Note that when this term is actually a channel name, say c, the situation is slightly different. We distinguish two cases depending on whether c is restricted or not. In particular, in case of an output of a bound channel name, the rule $\textsc{Open-Ch}_s$ requires renaming the channel name (as this is done in the rule $\textsc{Out-T}_s$). This is needed for our symbolic trace equivalence relation since we require both the left- and right-hand processes to use the same label without allowing α-conversion.

The relations $\xrightarrow{w}_s$ and $\overset{s}{\Rightarrow}_s$ are defined as for our intermediate semantics.

Example 16. *Continuing Example 10, the constraint system* $\mathcal{C}_1 = (\mathcal{E}; \Phi_0 \cup \{w_4 \rhd t\}; \mathcal{D}_1)$ *(see Example 14) is one of the constraint systems obtained by applying our symbolic rules on the process* $(\mathcal{E}; B(b, a); \Phi_0; \emptyset)$ *and considering the symbolic trace* $\mathsf{tr}_s = \mathtt{in}(Z_1, Y) \cdot \mathtt{out}(Z_2, w_4)$. *The other one (for the same sequence* tr_s*) is* $\mathcal{C}_2 = (\mathcal{E}; \Phi_2; \mathcal{D}_2)$ *where:*

$$(\mathcal{E}; \{\texttt{if } u = v \texttt{ then } \mathcal{Q}_1 \texttt{ else } \mathcal{Q}_2\} \uplus \mathcal{P}; \Phi; \mathcal{D}) \xrightarrow{\tau}_s (\mathcal{E}; \mathcal{Q}_1 \uplus \mathcal{P}; \Phi; \mathcal{D} \cup \{u =^?_{\mathsf{E}} v\}) \qquad (\textsc{Then}_s)$$

$$(\mathcal{E}; \{\texttt{if } u = v \texttt{ then } \mathcal{Q}_1 \texttt{ else } \mathcal{Q}_2\} \uplus \mathcal{P}; \Phi; \mathcal{D}) \xrightarrow{\tau}_s (\mathcal{E}; \mathcal{Q}_2 \uplus \mathcal{P}; \Phi; \mathcal{D} \cup \{u \neq^?_{\mathsf{E}} v\}) \qquad (\textsc{Else}_s)$$

$$(\mathcal{E}; \{\texttt{out}(p, u).\mathcal{Q}_1; \texttt{in}(q, x).\mathcal{Q}_2\} \uplus \mathcal{P}; \Phi; \mathcal{D}) \xrightarrow{\tau}_s (\mathcal{E}; \mathcal{Q}_1 \uplus \mathcal{Q}_2\{x \mapsto u\} \uplus \mathcal{P}; \Phi; \mathcal{D} \cup \{p =^?_{\mathsf{E}} q\}) \qquad (\textsc{Comm}_s)$$

$$(\mathcal{E}; \{\texttt{in}(p, x).\mathcal{Q}\} \uplus \mathcal{P}; \Phi; \mathcal{D}) \xrightarrow{\texttt{in}(Z,Y)}_s (\mathcal{E}; \mathcal{Q}\{x \mapsto y\} \uplus \mathcal{P}; \Phi; \mathcal{D} \cup \{Y \rhd^? y; Z \rhd^? z; z =^?_{\mathsf{E}} p\})$$
if Y, y, Z, z are fresh variables, $ar(Y) = ar(Z) = |\Phi|$ $(\textsc{In}_s)$

$$(\mathcal{E}; \{\texttt{out}(p, u).\mathcal{Q}\} \uplus \mathcal{P}; \Phi; \mathcal{D}) \xrightarrow{\nu w_n.\texttt{out}(Z,w_n)}_s (\mathcal{E}; \mathcal{Q} \uplus \mathcal{P}; \Phi \cup \{w_n \rhd u\}; \mathcal{D} \cup \{Z \rhd^? z\ ; z =^?_{\mathsf{E}} p\})$$
if w_n is a variable such that $n = |\Phi| + 1$, Z, z are fresh variables $ar(Z) = |\Phi|$ $(\textsc{Out-T}_s)$

$$(\mathcal{E}; \{\texttt{out}(p, c).\mathcal{Q}\} \uplus \mathcal{P}; \Phi; \mathcal{D}) \xrightarrow{\texttt{out}(Z,Y)}_s (\mathcal{E}; \mathcal{Q} \uplus \mathcal{P}; \Phi; \mathcal{D} \cup \{Z \rhd^? z\ ; Y \rhd^? y\ ; z =^?_{\mathsf{E}} p\ ; y =^?_{\mathsf{E}} c\})$$
if $c \notin \mathcal{E}$ and Y, y, Z, z are fresh variables, $ar(Y) = ar(Z) = |\Phi|$ $(\textsc{Out-Ch}_s)$

$$(\mathcal{E}; \{\texttt{out}(p, c).\mathcal{Q}\} \uplus \mathcal{P}; \Phi; \mathcal{D}) \xrightarrow{\nu ch_n.\texttt{out}(Z,ch_n)}_s (\mathcal{E}; (\mathcal{Q} \uplus \mathcal{P})\{c \mapsto ch_n\}; \Phi; \mathcal{D} \cup \{Z \rhd^? z; z =^?_{\mathsf{E}} p\})$$
if $c \in \mathcal{E}$, Z, z are fresh variables with $ar(Z) = |\Phi|$, ch_n a fresh channel name $(\textsc{Open-Ch}_s)$

u, v are terms having the same type, x is a variable of any type, and p, q, c are terms of channel type, *i.e.* names or variables.

Figure 4: Symbolic semantics

- $\Phi_2 = \Phi_0 \cup \{w_4 \rhd \mathsf{aenc}(n_b, \mathsf{pk}(skb))\}$*; and*
- $\mathcal{D}_2 = \{Z_1 \rhd^? z_1; Y \rhd^? y; z_1 =_{\mathsf{E}} c_B; \pi_2(\mathsf{adec}(y, skb)) \neq^?_{\mathsf{E}} \mathsf{pk}(ska); Z_2 \rhd^? z_2; z_2 =_{\mathsf{E}} c_B\}$.

For the same sequence tr_s*, similar constraint systems, denoted* $\mathcal{C}'_1$ *and* $\mathcal{C}'_2$ *can be derived for the process* $(\mathcal{E}; \{B(b, a')\}; \Phi_0)$*. The occurrences of ska will be replaced by ska′ (except the occurrence in* Φ_0*).*

6.3. Soundness and completeness

We show that the set of symbolic processes obtained from an intermediate process $(\mathcal{E}; \mathcal{P}; \Phi)$ exactly captures the set of execution traces of $(\mathcal{E}; \mathcal{P}; \Phi)$ through θ-concretization.

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

Proposition 4 (soundness). *Let* $(\mathcal{E}_1; \mathcal{P}_1; \Phi_1; \mathcal{D}_1)$*, and* $(\mathcal{E}_2; \mathcal{P}_2; \Phi_2; \mathcal{D}_2)$ *be two symbolic processes such that*

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

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

1. $\theta_1 \in \mathsf{Sol}((\mathcal{E}_1; \Phi_1; \mathcal{D}_1))$*, 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* $(\mathcal{E}_1; \mathcal{P}_1; \Phi_1; \mathcal{D}_1)$ *(resp.* $(\mathcal{E}_2; \mathcal{P}_2; \Phi_2; \mathcal{D}_2)$*).*

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 5 (completeness). *Let $(\mathcal{E}_1; \mathcal{P}_1; \Phi_1; \mathcal{D}_1)$ be a symbolic process, $(\mathcal{E}_1; \mathcal{P}'_1; \Phi'_1)$ its θ_1-concretization where $\theta_1 \in \mathsf{Sol}((\mathcal{E}_1; \Phi_1; \mathcal{D}_1))$. 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 $(\mathcal{E}_2; \mathcal{P}_2; \Phi_2; \mathcal{D}_2)$, a substitution θ_2, and a symbolic action α_s such that:*

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

6.4. Symbolic equivalence of constraint systems

As for processes in the applied-pi calculus, it is also possible to define equivalence of (sets of) constraint systems. Two sets of constraint systems Σ and Σ' are in equivalence if for any solution of a constraint system in Σ, there exists a constraint system in Σ' that has the same solution and such that the resulting frames are in static equivalence.

Definition 16 (symbolic equivalence). *Let Σ and Σ' be two sets of constraint systems that contain constraint systems having the same structure. We say that Σ and Σ' are in symbolic equivalence, denoted by $\Sigma \approx_s \Sigma'$, if for all $\mathcal{C} \in \Sigma$, for all $\theta \in \mathsf{Sol}(\mathcal{C})$, there exists $\mathcal{C}' \in \Sigma'$ such that $\theta \in \mathsf{Sol}(\mathcal{C}')$ and $\nu\mathcal{E}.\Phi\lambda_\theta \sim \nu\mathcal{E}.\Phi\lambda'_\theta$ (and conversely) where:*

- $\mathcal{C} = (\mathcal{E}; \Phi; D)$, *and* $\mathcal{C}' = (\mathcal{E}; \Phi'; D')$,
- λ_θ *(resp.* λ'_θ*) is the first-order substitution associated to* $\mathcal{C}$ *(resp.* $\mathcal{C}'$*) and* θ.

Example 17. *Consider the sets of constraint systems $\Sigma = \{\mathcal{C}_1, \mathcal{C}_2\}$ and $\Sigma' = \{\mathcal{C}'_1, \mathcal{C}'_2\}$ as defined in Example 16. These two sets are in symbolic equivalence whereas the sets $\{\mathcal{C}_1\}$ and $\{\mathcal{C}'_1\}$ are not in symbolic equivalence. Indeed, consider the substitution θ given in Example 14. We have that $\theta \in \mathsf{Sol}(\mathcal{C}_1)$ whereas $\theta \notin \mathsf{Sol}(\mathcal{C}'_1)$. Indeed, we have that $\theta = \{Z_1 \mapsto c_B, Z_2 \mapsto c_B, Y \mapsto \mathsf{aenc}(\langle n_i, w_1\rangle, w_3)\}$. Thus, in order to satisfy item 1 of Definition 13, the substitution λ' should be $\lambda' = \{z_1 \mapsto c_B, z_2 \mapsto c_B, y \mapsto \mathsf{aenc}(\langle n_i, \mathsf{pk}(ska)\rangle, \mathsf{pk}(skb))\}$. However, such a substitution λ' will not satisfy item 2 since the equality $\pi_2(\mathsf{adec}(\lambda'(y), skb)) =_{\mathsf{E}} \mathsf{pk}(ska')$ does not hold. This allows us to conclude that $\{\mathcal{C}_1\} \not\approx_s \{\mathcal{C}'_1\}$.*

The next section will be devoted to proving that grouping traces by constraint systems is sound w.r.t. trace equivalence.

7. Relating trace equivalence and symbolic trace equivalence

In this section, we reduce the decidability of trace equivalence (for processes without replication) to symbolic equivalence, *i.e.* deciding equivalence between sets of constraint systems. This is a first step towards deciding trace equivalence since symbolic equivalence is easier to manipulate than trace equivalence. As a first application, we combine our reduction result with the decidability result of [20] for equivalence between sets of constraint systems and retrieve that trace equivalence is decidable for processes without replication and with standard primitives (asymmetric and symmetric encryption, signatures and hashes). In the particular case of simple processes with trivial else branches, we further show that we can reduce the decidability of trace equivalence to deciding symbolic equivalence of sets of positive constraints systems that are reduced to singletons. This corresponds to the notion of equivalence introduced by M. Baudet in [10], namely S-equivalence, and for which several decision procedures already exist [10, 23]. We therefore retrieve as a second application that trace equivalence is decidable for simple processes with trivial else branches and for subterm-convergent equational theories.

7.1. General processes

Due to Proposition 3, it is sufficient to establish the link between symbolic equivalence and intermediate trace equivalence of intermediate processes.

Following the approach of [11], we compute from an intermediate process $A = (\mathcal{E}; \mathcal{P}; \Phi)$ the set of constraint systems capturing the possible executions of A, starting from its associated symbolic process $A_s \stackrel{\mathsf{def}}{=} (\mathcal{E}; \mathcal{P}; \Phi; \emptyset)$ and applying the rules defined in Figure 4. Formally, we define the set of traces of a symbolic process as follows:

$$\mathsf{trace}_s(A_s) = \{(\mathsf{tr}_s, (\mathcal{E}_2; \Phi_2; \mathcal{D}_2)) \mid A_s \stackrel{\mathsf{tr}_s}{\Rightarrow}_s (\mathcal{E}_2; \mathcal{P}_2; \Phi_2; \mathcal{D}_2) \text{ for some } (\mathcal{E}_2; \mathcal{P}_2; \Phi_2; \mathcal{D}_2)\}$$

When tr_s is fixed, we also write $(\mathsf{tr}_s, \Sigma) \in \mathsf{trace}_s(A_s)$ to define the set Σ as the set of constraint systems $\{\mathcal{C} \mid (\mathsf{tr}_s, \mathcal{C}) \in \mathsf{trace}_s(A_s)\}$.

Two intermediate processes are in intermediate trace equivalence if and only if, for any symbolic trace, their corresponding sets of constraints are in symbolic equivalence, which we call symbolic trace equivalence.

Definition 17 (symbolic trace equivalence). *Let A and B be two intermediate processes. They are in* symbolic trace equivalence *if for every sequence* tr *of symbolic actions, we have that:*

$$\{\mathcal{C} \mid (\mathsf{tr}, \mathcal{C}) \in \mathit{trace}_s(A_s)\} \approx_s \{\mathcal{C} \mid (\mathsf{tr}, \mathcal{C}) \in \mathit{trace}_s(B_s)\}$$

where A_s and B_s are the symbolic processes associated to A and B.

Proposition 6. *Let A and B be two intermediate processes: $A \approx_t B$ if, and only if A and B are in symbolic trace equivalence.*

Proof. We show the two directions separately. Let $A = (\mathcal{E}; \mathcal{P}_A; \Phi_A)$ and $B = (\mathcal{E}; \mathcal{P}_B; \Phi_B)$.

($\Leftarrow$) 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 $(\mathcal{E}; \mathcal{P}_A; \Phi_A) \stackrel{w}{\Rightarrow}_i (\mathcal{E}; \mathcal{P}'_A; \Phi'_A)$, 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 $\theta \in \mathsf{Sol}((\mathcal{E}; \Phi_A; \emptyset))$. Thanks to Proposition 5, we have that there exist a symbolic process $A'_s = (\mathcal{E}; \mathcal{P}'^s_A; \Phi'^s_A; \mathcal{D}'_A)$, a substitution θ', and a sequence w_s of symbolic actions such that:

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

Let $\mathcal{C}'_A = (\mathcal{E}; \Phi'^s_A; \mathcal{D}'_A)$. By definition of $\mathsf{trace}_s(A_s)$, we have that $(w_s, \mathcal{C}'_A) \in \mathsf{trace}_s(A_s)$. Since A and B are in symbolic trace equivalence, we deduce that there exists $\mathcal{C}'_B = (\mathcal{E}; \Phi'^s_B; \mathcal{D}'_B)$ such that $(w_s, \mathcal{C}'_B) \in \mathsf{trace}_s(B_s)$ with $\theta' \in \mathsf{Sol}(\mathcal{C}'_B)$ and $\nu\mathcal{E}.\Phi'^s_A\lambda^A_{\theta'} \sim \nu\mathcal{E}.\Phi'^s_B\lambda^B_{\theta'}$ where $\lambda^A_{\theta'}$ (resp. $\lambda^B_{\theta'}$) is the first-order solution of $\mathcal{C}'_A$ (resp. $\mathcal{C}'_B$) associated to θ'. By definition of $\mathsf{trace}_s(B_s)$, we have that $(\mathcal{E}; \mathcal{P}_B; \Phi_B; \emptyset) \stackrel{w_s}{\Rightarrow}_s (\mathcal{E}; \mathcal{P}'^s_B; \Phi'^s_B; \mathcal{D}'_B)$ for some $\mathcal{P}'^s_B$. Now, we apply Proposition 4, we have that $(\mathcal{E}; \mathcal{P}_B; \Phi_B) \stackrel{w_s\theta'}{\Rightarrow}_i (\mathcal{E}; \mathcal{P}'^s_B\lambda^B_{\theta'}; \Phi'^s_B\lambda^B_{\theta'})$. Let $\varphi_B = \nu\mathcal{E}.\Phi'^s_B\lambda^B_{\theta'}$. Clearly, we have that $(w_s\theta', \varphi_B) \in \mathsf{trace}_i(B)$. From the fact that $\nu\mathcal{E}.\Phi'^s_A\lambda^A_{\theta'} \sim \nu\mathcal{E}.\Phi'^s_B\lambda^B_{\theta'}$, we deduce that $\varphi_A \sim \varphi_B$. The other inclusion can be shown in a similar way.

($\Rightarrow$) We have to show that A and B are in symbolic trace equivalence, *i.e.* for every sequence w_s of symbolic actions, we have to show that, for any symbolic trace tr:

$$\{\mathcal{C} \mid (\mathsf{tr}, \mathcal{C}) \in \mathsf{trace}_s(A_s)\} \approx_s \{\mathcal{C} \mid (\mathsf{tr}, \mathcal{C}) \in \mathsf{trace}_s(B_s)\}$$

Let $(w_s, \mathcal{C}'_A) \in \mathsf{trace}_s(A_s)$ and $\theta \in \mathsf{Sol}(\mathcal{C}'_A)$. By definition of $\mathsf{trace}_s(A_s)$, we know that there exists $(\mathcal{E}; \mathcal{P}'^s_A; \Phi'^s_A; \mathcal{D}'_A)$ such that $A_s \stackrel{w_s}{\Rightarrow}_s (\mathcal{E}; \mathcal{P}'^s_A; \Phi'^s_A; \mathcal{D}'_A)$ and $\mathcal{C}'_A = (\mathcal{E}; \Phi'^s_A; \mathcal{D}'_A)$. Thanks to

Proposition 4, we have that $A \overset{w_s\theta}{\Rightarrow}_i (\mathcal{E}; \mathcal{P}'^s_A \lambda^A_\theta; \Phi'^s_A \lambda^A_\theta)$ where λ^A_θ is the first-order solution of $\mathcal{C}'_A$ associated to θ. Since A and B are in trace equivalence, we deduce that there exists $(\mathcal{E}; \mathcal{P}'_B; \Phi'_B)$ such that $B \overset{w_s\theta}{\Rightarrow}_i (\mathcal{E}; \mathcal{P}'_B; \Phi'_B)$ and $\nu\mathcal{E}.\Phi'^s_A \lambda^A_\theta \sim \nu\mathcal{E}.\Phi'_B$. Thanks to Proposition 5, we deduce that there exists $(\mathcal{E}; \mathcal{P}'^s_B; \Phi'^s_B; \mathcal{D}'_B)$, a substitution θ', and a sequence w'_s of symbolic actions such that:

1. $B_s \overset{w'_s}{\Rightarrow}_s (\mathcal{E}; \mathcal{P}'^s_B; \Phi'^s_B; \mathcal{D}'_B)$;
2. $\theta' \in \mathsf{Sol}((\mathcal{E}; \Phi'^s_B; \mathcal{D}'_B))$;
3. the process $(\mathcal{E}; \mathcal{P}'_B; \Phi'_B)$ is the θ'-concretization of $(\mathcal{E}; \mathcal{P}'^s_B; \Phi'^s_B; \mathcal{D}'_B)$; and
4. $w'_s\theta' = w_s\theta$.

Let $\mathcal{C}'_B = (\mathcal{E}; \Phi'^s_B; \mathcal{D}'_B)$. Actually, we can assume that $w'_s = w_s$ (by renaming the second order variables that occur in w'_s). Moreover, since $w_s\theta' = w_s\theta$, we have that $\theta' = \theta$. Lastly, since $\nu\mathcal{E}.\Phi'^s_A \lambda^A_\theta \sim \nu\mathcal{E}.\Phi'_B$, we easily deduce that $\nu\mathcal{E}.\Phi'^s_A \lambda^A_{\theta'} \sim \nu\mathcal{E}.\Phi'^s_B \lambda^B_{\theta'}$ where $\lambda^A_{\theta'}$ (resp. $\lambda^B_{\theta'}$) is the first-order solution of $\mathcal{C}'_A$ (resp. $\mathcal{C}'_B$) associated to $\theta = \theta'$. □

7.2. Simple processes with trivial else branches

For simple processes with trivial else branches, there is no need to consider sets of constraint systems, a single constraint system suffices. Intuitively, this is due to the fact that, given a symbolic action in which the variables of channel type have been instantiated, only one process can move using an instance of this symbolic action. Roughly, this allows us to ensure that once the symbolic trace and the communication channels have been fixed, there is only one symbolic execution that corresponds to this symbolic trace. Actually, because of silent actions, we can not ensure that the set of contraint systems (in the definition of symbolic trace equivalence) is a singleton but we will see that the set of constraint systems always contains a most general one (allowing us to forget the others). Moreover, we can show that such a constraint system is necessarily a positive one.

Definition 18 (constraint system with fixed channels). *Let $\mathcal{C} = (\mathcal{E}; \Phi; \mathcal{D})$ be a constraint system. We say that $\mathcal{C}$ is a* constraint system with fixed channels *if for all $(X \rhd^? x) \in \mathcal{D}$ with x a variable of channel type, there exists a unique equation $(x =^?_{\mathsf{E}} c) \in \mathcal{D}$, where c is a name of channel type. We denote by $\theta_{\mathcal{C}h(\mathcal{C})}$ the substitution whose domain is $\{X \mid (X \rhd^? x) \in \mathcal{D}$ and x is a variable of channel type$\}$ and such that $X\theta_{\mathcal{C}h(\mathcal{C})} = c$ when $(x =^?_{\mathsf{E}} c) \in \mathcal{D}$.*

Note that given a constraint system $\mathcal{C}$ with fixed channels, for any $\theta \in \mathsf{Sol}(\mathcal{C})$, we have that $\theta|_{\mathrm{dom}(\theta_{\mathcal{C}h(\mathcal{C})})} = \theta_{\mathcal{C}h(\mathcal{C})}$

Actually, it is quite easy to see that constraint systems generated during the symbolic execution of a symbolic process derived from a simple process satisfy this requirement.

Lemma 3. *Let A be a simple process without replication and A_s its associated symbolic process. Let tr_s be a sequence of symbolic actions. Let $\Sigma = \{\mathcal{C} \mid (\mathsf{tr}_s, \mathcal{C}) \in \mathsf{trace}_s(A_s)\}$. Each constraint system in Σ has fixed channels.*

Proof. Note that for symbolic processes that are obtained from simple processes without replication, we only need to consider the rules $\textsc{Then}_s$, $\textsc{Else}_s$, $\textsc{In}_s$, and $\textsc{Out-T}_s$. The other rules will never be applied in a symbolic derivation. Relying on Definition 7 of basic process, it is then easy to see that constraint of the form $X \rhd^? x$ can only be introduced by the rules $\textsc{Out-T}_s$ or $\textsc{In}_s$. In both case, an equality $x =^?_{\mathsf{E}} c$ with $c \in \mathcal{C}h$ is added in $\mathcal{D}$ (where x is a fresh variable). □

The execution of a simple process may yield to multiple constraint systems. But they are actually very similar and one of them is actually more general in the sense that it contains less constraints. Formally, we introduce the notion of most general constraint system.

Definition 19 (most general constraint system). *Let $\mathcal{C} = (\mathcal{E}; \Phi; \mathcal{D})$ and $\mathcal{C}' = (\mathcal{E}'; \Phi'; \mathcal{D}')$ be two constraint systems. The constraint system $\mathcal{C}$ is* more general than *$\mathcal{C}'$, denoted by $\mathcal{C}' \preceq \mathcal{C}$, if $\mathcal{E}' = \mathcal{E}$, $\Phi' = \Phi$, and $\mathcal{D}' = \mathcal{D} \uplus \mathcal{D}_0$ where $\mathcal{D}_0$ only contains equality and disequality constraints.*

Given a set of constraint systems Σ, we say that $\mathcal{C} \in \Sigma$ is the most general constraint system of Σ if $\mathcal{C}' \preceq \mathcal{C}$ for every $\mathcal{C}' \in \Sigma$.

Note that such a most general constraint system does not necessarily exist, but when it exists, it is unique.

Example 18. *Consider the simple process* $(\mathcal{E}; B(b,a); \Phi_0)$ *(see Example 13) and the constraint systems generated through the symbolic trace* $\mathsf{tr}_s = \mathit{in}(Z_1, Y)$*. We obtain the following constraint systems:*

- $\mathcal{C}_0 = (\mathcal{E}; Z_1 \rhd^? z_1; Y \rhd^? y; z_1 =^?_{\mathsf{E}} c_B; \Phi_0)$*;*
- $\mathcal{C}_{\mathsf{eq}} = (\mathcal{E}; Z_1 \rhd^? z_1; Y \rhd^? y; z_1 =^?_{\mathsf{E}} c_B; \pi_2(\mathsf{adec}(y, skb)) =^?_{\mathsf{E}} \mathsf{pk}(ska); \Phi_0)$*;*
- $\mathcal{C}_{\mathsf{deq}} = (\mathcal{E}; Z_1 \rhd^? z_1; Y \rhd^? y; z_1 =^?_{\mathsf{E}} c_B; \pi_2(\mathsf{adec}(y, skb)) \neq^?_{\mathsf{E}} \mathsf{pk}(ska); \Phi_0)$*.*

These constraint systems have fixed channels. Moreover, we have that $\mathcal{C}_0$ *is the most general constraint system of* $\{\mathcal{C}_0, \mathcal{C}_{\mathsf{eq}}, \mathcal{C}_{\mathsf{deq}}\}$*.*

We can show that an execution of a simple process admits a most general constraint system.

Proposition 7. *Let* A *be a simple process with trivial else branches and without replication, and let* A_s *be its associated symbolic process. Let* tr_s *be a sequence of symbolic actions.*

Let $\Sigma = \{\mathcal{C} \mid (\mathsf{tr}_s, \mathcal{C}) \in \mathit{trace}_s(A_s)\}$ *and* $\mathcal{C}_0 \in \Sigma$*. We have that* $\Sigma_0 = \{\mathcal{C} \mid \mathcal{C} \in \Sigma$ *and* $\theta_{\mathcal{C}h(\mathcal{C})} = \theta_{\mathcal{C}h(\mathcal{C}_0)}\}$ *admits a most general constraint system and this constraint system is a positive one.*

Proof. Note that for symbolic processes that are obtained from simple processes with trivial else branches, we only need to consider the rules $\textsc{Then}_s$, $\textsc{Else}_s$, $\textsc{In}_s$, and $\textsc{Out-T}_s$. The other rules will never be applied in a symbolic derivation.

Let tr_s be a sequence of symbolic actions, and let $\Sigma = \{\mathcal{C} \mid (\mathsf{tr}_s, \mathcal{C}) \in \mathsf{trace}_s(A_s)\}$ and $\mathcal{C}_0 \in \Sigma$. Let $\Sigma_0 = \{\mathcal{C} \mid \mathcal{C} \in \Sigma$ and $\theta_{\mathcal{C}h(\mathcal{C})} = \theta_{\mathcal{C}h(\mathcal{C}_0)}\}$. Since the outputs/inputs have been performed in the same order in each derivation (those have to be done as specified by the symbolic trace) and since we specified the channels used in the symbolic actions, we have that $\mathcal{E}_1 = \mathcal{E}_2$, $\Phi_1 = \Phi_2$, and the two sets $\mathcal{D}_1$ and $\mathcal{D}_2$ contains the same deduction constraints. Then, we consider a symbolic process such that $A_s \overset{\mathsf{tr}_s}{\Rightarrow} (\mathcal{E}; \mathcal{P}; \Phi; \mathcal{D})$, with $(\mathcal{E}; \Phi; \mathcal{D}) \in \Sigma_0$, using a derivation of minimal length (including the τ action). In such a derivation, rule $\textsc{Else}_s$ has never been applied, and thus $\mathcal{C}$ is a positive constraint system. Moreover, the only symbolic action $\textsc{Then}_s$ that have been performed are those that are mandatory. Hence, we have that $\mathcal{C}' \preceq \mathcal{C}$ for any $\mathcal{C}' \in \Sigma_0$. □

Then, it remains to ensure that it is sufficient to consider symbolic equivalence between these pairs of constraint systems when we want to decide symbolic equivalence between sets of constraint systems. This is the purpose of the following proposition.

Proposition 8. *Let* Σ *and* Σ' *be two sets of constraint systems having the same structure and fixed channels. We have that* $\Sigma \approx_s \Sigma'$ *if, and only if,* $\{\mathcal{C}_m\} \approx_s \{\mathcal{C}'_m\}$ *for every* $\mathcal{C}_0 \in \Sigma \cup \Sigma'$ *where:*

- $\mathcal{C}_m$ *is the most general constraint system of* $\Sigma_0 = \{\mathcal{C} \mid \mathcal{C} \in \Sigma$ *and* $\theta_{\mathcal{C}h(\mathcal{C})} = \theta_{\mathcal{C}h(\mathcal{C}_0)}\}$*, and*
- $\mathcal{C}'_m$ *is the most general constraint system of* $\Sigma'_0 = \{\mathcal{C} \mid \mathcal{C} \in \Sigma'$ *and* $\theta_{\mathcal{C}h(\mathcal{C})} = \theta_{\mathcal{C}h(\mathcal{C}_0)}\}$*.*

Proof. We prove this result in two steps.

1. We show that $\Sigma \approx_s \Sigma'$ if, and only if $\Sigma_0 \approx_s \Sigma'_0$ for every $\mathcal{C}_0 \in \Sigma \cup \Sigma'$.

   The converse implication is easy to established. We only consider the direct implication. Let $\mathcal{C}_0 \in \Sigma \cup \Sigma'$, $\mathcal{C} \in \Sigma_0$ with $\mathcal{C} = (\mathcal{E}; \Phi; \mathcal{D})$, and $\theta \in \mathsf{Sol}(\mathcal{C})$. We have that $\mathcal{C} \in \Sigma$. Since $\Sigma \approx_s \Sigma'$, we know that there exists $\mathcal{C}' \in \Sigma'$ such that $\theta \in \mathsf{Sol}(\mathcal{C}')$ and $\nu\mathcal{E}.\Phi\lambda_\theta \approx \nu\mathcal{E}'.\Phi'\lambda'_\theta$ where:
   - $\mathcal{C}' = (\mathcal{E}'; \Phi'; \mathcal{D}')$, and
   - λ_θ (resp. λ'_θ) is the first-order solution associated to $\mathcal{C}$ (resp. $\mathcal{C}'$) and θ.

This implies that $\theta_{\mathcal{C}h(\mathcal{C})} = \theta_{\mathcal{C}h(\mathcal{C}')}$. Indeed, since $\mathcal{C}$ and $\mathcal{C}'$ have fixed channels and same structure, we have that $\mathrm{dom}(\theta_{\mathcal{C}h(\mathcal{C})}) = \mathrm{dom}(\theta_{\mathcal{C}h(\mathcal{C}')})$, $\theta|_{\mathrm{dom}(\theta_{\mathcal{C}h(\mathcal{C})})} = \theta_{\mathcal{C}h(\mathcal{C})}$, and $\theta|_{\mathrm{dom}(\theta_{\mathcal{C}h(\mathcal{C}')})} = \theta_{\mathcal{C}h(\mathcal{C}')}$. Hence, by definition of Σ'_0, we have that $\mathcal{C}' \in \Sigma'_0$.

2. We show that given two sets Σ and Σ' of constraint systems having a most general constraint system $\mathcal{C}_\mathsf{m}$ and $\mathcal{C}'_\mathsf{m}$, we have that $\Sigma \approx_s \Sigma'$ if, and only if, $\{\mathcal{C}_\mathsf{m}\} \approx_s \{\mathcal{C}'_\mathsf{m}\}$.

   Relying on Definition 19 (most general constraint system), it is quite easy to see that $\{\mathcal{C}_\mathsf{m}\} \approx_s \{\mathcal{C}'_\mathsf{m}\}$ implies $\Sigma \approx_s \Sigma'$. We now consider the other implication. Let $\mathcal{C}_\mathsf{m} = (\mathcal{E}_\mathsf{m}; \Phi_\mathsf{m}; \mathcal{D}_\mathsf{m})$ and $\mathcal{C}'_\mathsf{m} = (\mathcal{E}'_\mathsf{m}; \Phi'_\mathsf{m}; \mathcal{D}'_\mathsf{m})$. Let $\theta \in \mathsf{Sol}(\mathcal{C}_\mathsf{m})$. Since $\mathcal{C}_\mathsf{m} \in \Sigma$, by hypothesis, we know that there exists $\mathcal{C}' \in \Sigma'$ such that $\theta \in \mathsf{Sol}(\Sigma')$ and $\nu\mathcal{E}_\mathsf{m}.\Phi_\mathsf{m}\lambda_\theta \sim \nu\mathcal{E}'.\Phi'\lambda'_\theta$ where:

   - $\mathcal{C}' = (\mathcal{E}'; \Phi'; \mathcal{D}')$, and
   - λ_θ (resp. λ'_θ) is the first-order solution associated to $\mathcal{C}$ (resp. $\mathcal{C}'$) and θ.

   Since $\mathcal{C}'_\mathsf{m}$ is the most general constraint system of Σ', we have that $\mathcal{E}' = \mathcal{E}'_\mathsf{m}$, $\Phi' = \Phi'_\mathsf{m}$, and $\mathcal{D}' = \mathcal{D}'_m \cup \mathcal{D}_0$ where $\mathcal{D}_0$ contains only some equality and disequality constraints. Hence, we have that $\theta \in \mathsf{Sol}(\mathcal{C}'_\mathsf{m})$ and the first-order solution associated to $\mathcal{C}'_\mathsf{m}$ and θ is also λ'_θ. Hence, we have that $\nu\mathcal{E}'.\Phi'\lambda'_\theta = \nu\mathcal{E}'_\mathsf{m}.\Phi'_\mathsf{m}\lambda'_\theta$. This allows us to conclude. □

7.3. Decidability results

We have shown that deciding trace equivalence of processes in the applied pi calculus can be reduced to deciding symbolic equivalence of sets of constraint systems or even just pairs of constraint systems. We show in this section how this can be applied to retrieve decidability results of trace equivalence, reusing existing decision procedures.

Theorem 6. *Given an algorithm for deciding symbolic equivalence between sets of constraint systems, we can derive an algorithm for deciding trace equivalence between processes without replication.*

The algorithm for deciding trace equivalence follows from Proposition 6 since there are a finite number of symbolic traces and each symbolic trace leads to a finite number of constraint systems.

Example 19. *We have that:*

1. $(\mathcal{E}; \{B(b,a)\}; \Phi_0) \approx_t (\mathcal{E}; \{B(b,a')\}; \Phi_0)$. *Relying on Proposition 6, this non-trivial equivalence can be established by checking symbolic equivalence of several pairs of sets of constraint systems. This equivalence requires in particular to check that $\Sigma \approx_s \Sigma'$ (see Example 17). These two sets Σ and Σ' are those generated by considering the symbolic trace $\mathsf{tr}_s = \mathit{in}(Z_1, Y) \cdot \mathit{out}(Z_2, w_4)$.*
2. $(\mathcal{E}; \{B^-(b,a)\}; \Phi_0) \not\approx_t (\mathcal{E}; \{B^-(b,a')\}; \Phi_0)$. *Regarding this equivalence, we have seen that $\{\mathcal{C}_1\} \not\approx_s \{\mathcal{C}'_1\}$. Since these two sets correspond to those generated by considering the symbolic trace $\mathsf{tr}_s = \mathit{in}(Z_1, Y) \cdot \mathit{out}(Z_2, w_4)$, we conclude that symbolic trace equivalence does not hold between the two associated symbolic processes. Thus, relying again on Proposition 6, we can establish that $(\mathcal{E}; \{B^-(b,a)\}; \Phi_0) \not\approx_t (\mathcal{E}; \{B^-(b,a')\}; \Phi_0)$.*

We are now ready to establish two particular decidability results.

7.3.1. Decidability of trace equivalence for standard primitives

The only result that considers symbolic equivalence between sets of constraint systems is [20]. However, their decision procedure works in a slightly different setting than the one presented in this paper. In fact, they consider a fixed set of cryptographic primitives, namely signature, hash function, pairing, symmetric and asymmetric encryptions. Moreover, they split the function symbols into a set $\mathcal{F}_c$ of *constructors symbols* and a set $\mathcal{F}_d$ of *destructors symbols*. Destructors are used to model the fact that some operations fail.

$$\mathcal{F}_c = \{\mathsf{senc}/2,\ \mathsf{aenc}/2,\ \mathsf{pub}/1,\ \mathsf{sign}/2,\ \mathsf{vk}/1,\ \langle\,\rangle/2,\ \mathsf{h}/1\}$$
$$\mathcal{F}_d = \{\mathsf{sdec}/2,\ \mathsf{adec}/2,\ \mathsf{check}/2,\ \mathsf{proj}_1/1,\ \mathsf{proj}_2/1\}.$$

The behavior of the cryptographic primitives is described by the means of a convergent rewriting system where $t{\downarrow}$ denotes the normal form of a term t.

$$\begin{array}{rcl} \mathsf{sdec}(\mathsf{senc}(x,y),y) & \to & x \\ \mathsf{adec}(\mathsf{aenc}(x,\mathsf{pub}(y)),y) & \to & x \end{array} \qquad \begin{array}{rcl} \mathsf{proj}_1(\langle x,y\rangle) & \to & x \\ \mathsf{proj}_2(\langle x,y\rangle) & \to & y \end{array} \qquad \begin{array}{rcl} \mathsf{check}(\mathsf{sign}(x,y),\mathsf{vk}(y)) & \to & x \end{array}$$

Moreover, to represent messages, they only consider *valid terms*. A term t is valid, denoted $\mathsf{valid}(t)$, if for any syntactic subterm u of t, we have that $u{\downarrow}$ is a constructor term, *i.e.* u does not contain any destructor symbol. For instance, the terms $\mathsf{sdec}(a,b)$, $\mathsf{proj}_1(\langle a,\mathsf{sdec}(a,b)\rangle)$, and $\mathsf{proj}_1(a)$ are not valid. In order to use their algorithm in the setting presented in this paper, we first have to fix the equational theory that will allow one to model their rewrite system and the validity checks. To do this, we consider the signature $\mathcal{F}_0 = \mathcal{F}_c \uplus \mathcal{F}_d \uplus \{\mathsf{fail}/0\}$, and the smallest equivalence relation $=_{\mathsf{E}_0}$ that is closed by application of function symbols, and that contains:

- $u =_{\mathsf{E}_0} v$, for any ground terms u, v with $\mathsf{valid}(u)$, $\mathsf{valid}(v)$, $u{\downarrow} = v{\downarrow}$; and
- $u =_{\mathsf{E}_0} \mathsf{fail}$, for any ground term u and $\neg\mathsf{valid}(u)$.

Intuitively, in [20], the messages that are exchanged during an execution have to be valid. In order to apply the result of [20], we need to restrict the class of processes in order to only consider processes that exchange *valid* messages. Formally, we consider processes that are *valid*, *i.e.* those that are generated using the following grammar (which is a special case of the grammar of plain processes that is given in Section 2).

$$\begin{array}{rcl} P,Q,R & := & 0 \\ & | & P \mid Q \\ & | & !P \\ & | & \nu n.P \\ & | & \mathtt{if}\ M_1 = M_2\ \mathtt{and}\ \mathsf{fail} \neq M_1\ \mathtt{and}\ \mathsf{fail} \neq M_2\ \mathtt{then}\ P\ \mathtt{else}\ Q \\ & | & \mathtt{in}(u,x).\mathtt{if}\ x = \mathsf{fail}\ \mathtt{then}\ 0\ \mathtt{else}\ P \\ & | & \mathtt{if}\ N = \mathsf{fail}\ \mathtt{then}\ 0\ \mathtt{else}\ \mathtt{out}(u,N).P \end{array}$$

where n is a name, u is a metavariable of channel type, M_1, M_2 are terms having the same type, x is a variable of any type, N is a term of any type, and $\mathcal{P}$ (resp. $\mathcal{Q}$) is a multiset of plain intermediate processes that are valid. Moreover, we assume that the constant fail does not occur in M_1, M_2 and N. The logical connector $\mathtt{and}$ and the disequations $\neq$ are syntactic sugar that can be easily encoded using nested conditionals.

Relying on the algorithm that has been proposed in [20] for deciding a notion of symbolic equivalence between pair of sets of contraint systems, we get that observational equivalence is decidable for valid processes under the equational theory E_0.

Corollary 2. *Let E_0 be the equational theory defined above. Let A and B be two valid processes without replication. The problem whether A and B are observationally (or trace) equivalent is decidable.*

The proof follows from [20] and Theorem 6. The main technicality consists in proving that valid processes under E_0 coincide with the framework developed in [20]. This is detailed in Appendix D.

7.3.2. Decidability of trace equivalence for simple processes

Several procedures have been proposed to decide symbolic equivalence between pair of positive constraint systems [10, 49, 23]. Those procedures can be used for deciding trace equivalence between simple processes with trivial else branches.

Theorem 7. *Given an algorithm for deciding symbolic equivalence between two positive constraint systems, we can derive an algorithm for deciding trace equivalence (and observational equivalence) between simple processes without replication and with trivial else branches.*

Our algorithm consists of generating all the symbolic traces and their associated constraint systems. The difficult point consists in showing that we can indeed restrict us to consider symbolic equivalence between two positive constraint systems. This is the purpose of Proposition 7 and Proposition 8 stated and proved in Section 7.2.

Our decision procedure for trace equivalence works as follows: We consider each symbolic trace tr in turn (note that there are only finitely many symbolic traces):

- Compute (in polynomial time) the sets of constraint systems Σ and Σ' such that $(\mathsf{tr}, \Sigma) \in \mathsf{trace}_s(A)$ and $(\mathsf{tr}, \Sigma') \in \mathsf{trace}_s(B)$;
- For each $\mathcal{C}_0 \in \Sigma \cup \Sigma'$, compute $\mathcal{C}_\mathsf{m}$ (resp. $\mathcal{C}'_\mathsf{m}$) the most general constraint system of $\{\mathcal{C} \mid \mathcal{C} \in \Sigma \text{ and } \theta_{\mathcal{C}h(\mathcal{C})} = \theta_{\mathcal{C}h(\mathcal{C}_0)}\}$ (resp. $\{\mathcal{C} \mid \mathcal{C} \in \Sigma' \text{ and } \theta_{\mathcal{C}h(\mathcal{C})} = \theta_{\mathcal{C}h(\mathcal{C}_0)}\}$);
- Check whether the two (positive) constraint systems $\{\mathcal{C}_\mathsf{m}\}$ and $\{\mathcal{C}'_\mathsf{m}\}$ are in symbolic equivalence using the oracle.

If our algorithm finds a symbolic trace tr and a constraint system $\mathcal{C}_0$ for which the two resulting positive constraint systems $\mathcal{C}$ and $\mathcal{C}'$ are not in symbolic equivalence then it return *no* (not equivalent), otherwise it returns *yes* (equivalent). To show the correctness of our algorithm, we have to show that $\{\mathcal{C} \mid \mathcal{C} \in \Sigma \text{ and } \theta_{\mathcal{C}h(\mathcal{C})} = \theta_{\mathcal{C}h(\mathcal{C}_0)}\}$ (resp. $\{\mathcal{C} \mid \mathcal{C} \in \Sigma' \text{ and } \theta_{\mathcal{C}h(\mathcal{C})} = \theta_{\mathcal{C}h(\mathcal{C}_0)}\}$) has a most general constraint system which is positive (see Proposition 7)

As already mentioned in Section 6.4, a result by M. Baudet [10] shows that checking whether two (positive) constraints systems are in symbolic equivalence 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. $\mathcal{R}$. For example, the equational theory presented in Example 3 is a convergent subterm theory. Many other examples can be found *e.g.* in [1]. Since this result has been proved in the same setting as the one of this paper, we can derive the following result.

Corollary 3. *Let E be a subterm convergent equational theory. Let A and B be two simple processes with trivial else branches and without replication. The problem whether A and B are observationally (or trace) equivalent is co-NP-complete.*

8. Conclusion

Trace equivalence is a central notion for expressing privacy-like properties. We have shown that it coincides with may-testing equivalence for image-finite processes, thus in particular for processes without replication. Observational equivalence is a stronger notion (and often too strong to express privacy). It nevertheless coincides with trace equivalence for determinate processes, thus in particular for the class of simple processes. We have focused on three behavioral equivalences, namely trace, may-testing, and observational equivalence. This choice comes from the fact that they are the three notions that are used in the context of security protocols (up to our knowledge). However, several other behavioral equivalences have been proposed such as must-testing equivalence [16, 46] or failure-equivalence [36]. It would be interesting to study their relationships and to understand which ones are the most appropriate in the context of security protocols.

We have proposed a proof technique for trace equivalence, showing that it can be reduced to checking equivalence between sets of constraint systems. Our reduction result is very general and holds for arbitrary processes without replication and for arbitrary equational theories. It does not provide immediately decidability results (trace equivalence being undecidable in general) but constraint systems are much simpler and more amendable to automation. As an illustration, we

have reused two existing decidability results on equivalence of constraint systems, getting two decidability results for trace equivalence in the applied pi calculus.

- Trace equivalence is decidable for general processes without replication and for standard primitives (symmetric and asymmetric encryption, signatures, hashes).
- Trace equivalence is decidable for simple processes without replication nor else branches and for subterm-convergent theories.

However, in order to get an efficient procedure for trace equivalence, it is necessary to come with some optimisations to reduce the number of interleavings that have to be considered, and so the number of equivalence between sets of constraint systems that have to be checked. This problem has already been studied in the context of trace properties [25, 45] but seems to be more challenging for trace equivalence.

We would also like to develop further decision procedures for equivalence between constraint systems. In particular, less standard primitives such as blind signatures or trapdoor commitment functions are crucial in the context of e-voting but do not fall in any existing decidability results. So, we aim at deciding equivalence between sets of constraint systems for more general primitives. This could be done in a modular way. For this, we have to provide an algorithm to combine the decision procedures that we will obtain for various cryptographic primitives as it was done for instance in [22] in the context of accessibility-based properties.

Lastly, it would be nice to provide some composition results in the context of privacy-type security properties. It is well-known that composition works when the protocols do not share secrets. However, there is no result allowing us to derive some interesting results when the processes rely on some shared secrets such as long term keys, in the context of equivalence-based properties. This kind of composition results will be very useful to analyse a whole system in a modular way.

[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 and W. Charatonik. On name generation and set-based analysis in the Dolev-Yao model. In *Proc. of the 13th International Conference on Concurrency Theory (CONCUR'02)*, LNCS, pages 499–514, Brno, Czech Republic, 2002. Springer Verlag.

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

[7] M. Arapinis, S. Bursuc, and M. Ryan. Privacy supporting cloud computing: Confichair, a case study. In *Proc. 1st Conference on Principles of Security and Trust (POST'12)*, 2012. To appear.

[8] M. Arapinis, T. Chothia, E. Ritter, and M. Ryan. Analysing unlinkability and anonymity using the applied pi calculus. In *Proc. 23rd IEEE Computer Security Foundations Symposium (CSF'10)*, pages 107–121. IEEE Computer Society Press, 2010.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

[14] B. Blanchet, M. Abadi, and C. Fournet. Automated Verification of Selected Equivalences for Security Protocols. In *20th Symposium on Logic in Computer Science*, pages 331–340, 2005.

[15] B. Blanchet and A. Podelski. Verification of cryptographic protocols: Tagging enforces termination. In *Foundations of Software Science and Computation Structures (FoSSaCS'03)*, volume 2620 of *LNCS*, Warsaw, Poland, April 2003. Springer Verlag.

[16] M. Boreale and R. D. Nicola. Testing equivalence for mobile processes. In W. Cleaveland, editor, *CONCUR '92*, volume 630 of *Lecture Notes in Computer Science*, pages 2–16. Springer Berlin Heidelberg, 1992.

[17] M. Bruso, K. Chatzikokolakis, and J. den Hartog. Formal verification of privacy for RFID systems. In *Proc. 23rd IEEE Computer Security Foundations Symposium (CSF'10)*. IEEE Computer Society Press, 2010.

[18] R. Chadha, Ş. Ciobâcă, and S. Kremer. Automated verification of equivalence properties of cryptographic protocols. In H. Seidl, editor, *Programming Languages and Systems — Proceedings of the 21th European Symposium on Programming (ESOP'12)*, Lecture Notes in Computer Science, Tallinn, Estonia, Mar. 2012. Springer. To appear.

[19] V. Cheval and B. Blanchet. Proving more observational equivalences with ProVerif. Research Report LSV-12-19, Laboratoire Spécification et Vérification, ENS Cachan, France, Oct. 2012. 34 pages.

[20] V. Cheval, H. Comon-Lundh, and S. Delaune. Trace equivalence decision: Negative tests and non-determinism. In *Proc. 18th ACM Conference on Computer and Communications Security (CCS'11)*, pages 321–330, Chicago, Illinois, USA, Oct. 2011. ACM Press.

[21] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP decision procedure for protocol insecurity with XOR. In *Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science (LICS'03)*, pages 261–270, Ottawa (Canada), 2003. IEEE Computer Society Press.

[22] Y. Chevalier and M. Rusinowitch. Hierarchical combination of intruder theories. *Information and Computation*, 206(2-4):352–377, 2008.

[23] Y. Chevalier and M. Rusinowitch. Decidability of symbolic equivalence of derivations. *Journal of Automated Reasoning*, 2011. To appear.

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

[25] E. M. Clarke, S. Jha, and W. R. Marrero. Efficient verification of security protocols using partial-order reductions. *International Journal on Software Tools for Technology Transfer*, 4(2):173–188, 2003.

[26] 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.

[27] H. Comon-Lundh, S. Delaune, and J. Millen. Constraint solving techniques and enriching the model with equational theories. In V. Cortier and S. Kremer, editors, *Formal Models and Techniques for Analyzing Security Protocols*, volume 5 of *Cryptology and Information Security Series*, pages 35–61. IOS Press, 2011.

[28] H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of Exclusive Or. In *Proc. 18th Annual IEEE Symposium on Logic in Computer Science (LICS '03)*, pages 271–280, Los Alamitos, CA, 2003. IEEE Computer Society.

[29] V. Cortier and S. Delaune. A method for proving observational equivalence. In *Proceedings of the 22nd IEEE Computer Security Foundations Symposium (CSF'09)*, pages 266–276, Port Jefferson, NY, USA, July 2009. IEEE Computer Society Press.

[30] C. Cremers. The Scyther Tool: Verification, falsification, and analysis of security protocols. In *Computer Aided Verification, 20th International Conference, CAV 2008, Princeton, USA, Proc.*, volume 5123/2008 of *Lecture Notes in Computer Science*, pages 414–418. Springer, 2008.

[31] 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.

[32] S. Delaune, S. Kremer, and M. D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, (4):435–487, July 2008.

[33] S. Delaune, M. D. Ryan, and B. Smyth. Automatic verification of privacy properties in the applied pi-calculus. In Y. Karabulut, J. Mitchell, P. Herrmann, and C. D. Jensen, editors, *Proceedings of the 2nd Joint iTrust and PST Conferences on Privacy, Trust Management and Security (IFIPTM'08)*, volume 263 of *IFIP Conference Proceedings*, pages 263–278, Trondheim, Norway, June 2008. Springer.

[34] 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.

[35] N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *Workshop on Formal Methods and Security Protocols*, 1999.

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

[37] C. A. R. Hoare. Communicating sequential processes. *Commun. ACM*, 21(8):666–677, 1978.

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

[39] J. Liu and H. Lin. A complete symbolic bisimulation for full applied pi calculus. In *Proceedings of the 36th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM '10)*, LNCS, pages 552–563. Springer-Verlag, 2010.

[40] J. Liu and H. Lin. Proof system for applied pi calculus. In *IFIP TCS 2010*, pages 229–243, 2010.

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

[42] M.Boreale, R. D. Nicola, and R. Pugliese. Proof techniques for cryptographic processes. *SIAM Journal on Computing*, 31(3):947–986, 2002.

[43] 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.

[44] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980.

[45] S. Mödersheim, L. Viganò, and D. A. Basin. Constraint differentiation: Search-space reduction for the constraint-based analysis of security protocols. *Journal of Computer Security*, 18(4):575–618, 2010.

[46] R. D. Nicola and M. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984.

[47] R. Ramanujam and S. Suresh. Tagging makes secrecy decidable for unbounded nonces as well. In *Proc. 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'03)*, 2003.

[48] 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.

[49] A. Tiu and J. E. Dawson. Automating open bisimulation checking for the spi calculus. In *Proc. 23rd IEEE Computer Security Foundations Symposium (CSF'10)*, pages 307–321. IEEE Computer Society Press, 2010.

A. Testing equivalence vs trace equivalence

Proposition 1. *Let A and B be two closed extended process with* $\mathrm{dom}(A) = \mathrm{dom}(B)$*, and* $C[_] = \nu\tilde{n}.(D \mid _)$ *be an evaluation context closing for A. If $C[A] \mapsto^* A''$ for some process A'', then there exist a closed extended process A', an evaluation context $C' = \nu\tilde{n'}.(D' \mid _)$ closing for A', and a trace* $\mathsf{tr} \in (\mathcal{A} \smallsetminus \{\tau\})^*$ *such that $A'' \equiv C'[A']$, $A \overset{\mathsf{tr}}{\Rightarrow} A'$, and for all closed extended process B', we have that:*

$$B \overset{\mathsf{tr}}{\Rightarrow} B' \text{ and } \phi(B') \sim \phi(A') \text{ implies that } C[B] \mapsto^* C'[B'].$$

Proof. Let A and B be two extended processes with $\mathrm{dom}(A) = \mathrm{dom}(B)$ and C be an evaluation context closing for A. Let A'' be such that $C[A] \mapsto^* A''$. We prove the result by induction on the length ℓ of the derivation.

Base case $\ell = 0$: In such a case, we have that $A'' \equiv C[A]$. Let $A' = A$, $C' = C$ and $\mathsf{tr} = \epsilon$, we have that $A'' \equiv C'[A']$, and $A \overset{\epsilon}{\Rightarrow} A'$. Let B' be a closed extended process such that $B \overset{\epsilon}{\Rightarrow} B'$ and $\Phi(B') \sim \Phi(A')$ for some B'. Clearly, we have that $C[B] \mapsto^* C'[B']$ since $C' = C$ and $B \to^* B'$.

Inductive case $\ell > 0$: In such a case, we have that there exists a closed extended process A_1 such that $C[A] \mapsto^* A_1$ with a derivation whose length is smaller than ℓ, and $A_1 \overset{\tau}{\mapsto} A''$. Thus, we can apply our induction hypothesis allowing us to conclude that there exist an extended process A'_1, an evaluation context $C'_1[_] = \nu\tilde{n'_1}.(D'_1 \mid _)$ closing for A'_1, and a trace $\mathsf{tr}_1 \in (\mathcal{A} \smallsetminus \{\tau\})^*$ such that $A_1 \equiv C'_1[A'_1]$, $A \overset{\mathsf{tr}_1}{\Rightarrow} A'_1$, and for all closed extended processes B'_1, we have that:

$$B \overset{\mathsf{tr}_1}{\Rightarrow} B'_1 \text{ and } \Phi(B'_1) \sim \Phi(A'_1) \text{ implies that } C[B] \mapsto^* C'_1[B'_1].$$

Since $A_1 \equiv C'_1[A'_1]$ and $A_1 \overset{\tau}{\mapsto} A''$, we have that $C'_1[A'_1] \overset{\tau}{\mapsto} A''$ (internal reduction is closed under structural equivalence). We do a case analysis on the rule involved in this reduction.

Case of an internal reduction in A'_1, i.e. there exists A' such that $A'_1 \overset{\tau}{\to} A'$ and $A'' \equiv C'_1[A']$. In such a case, we have that $C'_1[A'_1] \overset{\tau}{\to} C'_1[A']$. Let $C' = C'_1$ and $\mathsf{tr} = \mathsf{tr}_1$. We have that $A'' \equiv C'_1[A'] = C'[A']$ and $A \overset{\mathsf{tr}_1}{\Rightarrow} A'_1 \overset{\tau}{\to} A'$, *i.e.* $A \overset{\mathsf{tr}}{\Rightarrow} A'$. Lastly, let B' be a closed extended process such that $B \overset{\mathsf{tr}}{\Rightarrow} B'$ and $\Phi(B') \sim \Phi(A')$. We have that $B \overset{\mathsf{tr}_1}{\Rightarrow} B'$ and $\Phi(B') \sim \Phi(A'_1) \equiv \Phi(A')$, and thus relying on our induction hypothesis, we conclude that $C[B] \mapsto^* C'_1[B'] = C'[B']$. This allows us to conclude.

Case of an internal reduction in C'_1, i.e. there exists D'_2 such that $\nu\tilde{n}'_1(D'_1 \mid B) \overset{\tau}{\to} \nu\tilde{n}'_1.(D'_2 \mid B)$ for any process B such that $\phi(B) \sim \phi(A'_1)$, and $A'' \equiv \nu\tilde{n}'_1.(D'_2 \mid A'_1)$. In such a case, we have that $C'_1[A'_1] \overset{\tau}{\to} \nu\tilde{n}'_1.(D'_2 \mid A'_1)$. Let $A' = A'_1$, $C'[_] = \nu\tilde{n}'_1.(D'_2 \mid _)$ and $\mathsf{tr} = \mathsf{tr}_1$. We have that $A'' \equiv \nu\tilde{n}'_1.(D'_2 \mid A'_1) = C'[A']$ and $A \overset{\mathsf{tr}_1}{\Rightarrow} A'_1 = A'$, *i.e.* $A \overset{\mathsf{tr}}{\Rightarrow} A'$. Lastly, let B' be a closed extended process such that $B \overset{\mathsf{tr}}{\Rightarrow} B'$ and $\Phi(B') \sim \Phi(A')$. We have that $B \overset{\mathsf{tr}_1}{\Rightarrow} B'$ and $\Phi(B') \sim \Phi(A'_1) \equiv \Phi(A')$, and thus relying on our induction hypothesis, we have that $C[B] \mapsto^* C'_1[B']$. But by our hypothesis on the internal reduction, $\Phi(B') \sim \Phi(A'_1)$ implies that $C'_1[B'] \overset{\tau}{\mapsto} \nu\tilde{n}'_1.(D'_2 \mid B') = C'[B']$ and so $C[B] \mapsto^* C'[B']$. This allows us to conclude.

*Case of a rule (*Comm*) between C'_1 (output) and A'_1 (input), i.e. $D'_1 \equiv \nu\tilde{n}.(\mathtt{out}(c, M).P \mid D)$, $A'_1 \equiv A'_2 = \nu\tilde{r}.(\mathtt{in}(c, z).Q \mid A_2)$ for some c, M, P, D, A'_2, $\tilde{r}$, z, Q, and A_2 such that z is a fresh variable, $fv(M) \subseteq \mathrm{dom}(A'_2)$, $\tilde{r} \cap (fn(M) \cup fv(M)) = \emptyset$, and $\tilde{n} \cap (fn(A'_2) \cup fv(A'_2)) = \emptyset$. We assume in addition that names and variables in $\tilde{n}$ do not occur in $fn(B)$, $fv(B)$, and* tr. In such a case, we have that

$$\begin{array}{rcl} C'_1[A'_1] & \equiv & \nu\tilde{n}'_1.[\nu\tilde{n}.(\mathtt{out}(c, M).P \mid D) \mid A'_2) \\ & \equiv & \nu\tilde{n}'_1.\nu\tilde{n}.[\mathtt{out}(c, M).P \mid D \mid \nu\tilde{r}.(\mathtt{in}(c, z).Q \mid A_2)] \\ & \to & \nu\tilde{n}'_1.\nu\tilde{n}.[P \mid D \mid \nu\tilde{r}.(Q\{^M/_z\} \mid A_2)] \end{array}$$

and $A'' \equiv \nu\tilde{n}'_1, \tilde{n}.[P \mid D \mid \nu\tilde{r}.(Q\{^M/_z\} \mid A_2)]$.

Let $A' = \nu\tilde{r}.(Q\{^M/_z\} \mid A_2)$, $C'[_] = \nu\tilde{n}'_1.\nu\tilde{n}.(P \mid D \mid _)$, and $\mathsf{tr} = \mathsf{tr}_1 \cdot \mathtt{in}(c, M)$. We have that $A'' \equiv C'[A']$. By induction hypothesis, we have that $A \overset{\mathsf{tr}_1}{\Rightarrow} A'_1$ and $A'_1 \overset{\mathtt{in}(c,M)}{\mapsto} A'$. This allows us to conclude that $A \overset{\mathsf{tr}}{\Rightarrow} A'$. Note that A' is a closed extended process ($fv(M) \subseteq \mathrm{dom}(A'_2) = \mathrm{dom}(A'_1)$).

Lastly, let B' be a closed extended processes such that $B \overset{\mathsf{tr}}{\Rightarrow} B'$ and $\Phi(B') \sim \Phi(A')$. We have that there exists B'_1 such that $B \overset{\mathsf{tr}_1}{\Rightarrow} B'_1 \xrightarrow{\mathtt{in}(c,M)} B'$. Moreover, we can assume w.l.o.g. that $\tilde{n}$ do not occur in $fn(B'_1)$ and $fv(B'_1)$ since $\tilde{n}$ do not occur in $fn(B)$, $fv(B)$ and tr_1. Since $\Phi(B') \sim \Phi(A')$, we have also that $\Phi(B'_1) \sim \Phi(A'_1)$. Thus, we can apply our induction hypothesis on B'_1. This allows us to deduce that $C[B] \rightarrow^* C'_1[B'_1]$. In order to conclude, it remains to show that $C'_1[B'_1] \rightarrow C'[B']$.

We have seen that $B'_1 \xrightarrow{\mathtt{in}(c,M)} B'$. Hence, we know that there exists $\tilde{m}, P_2, B_2$ such that $B'_1 \equiv \nu\tilde{m}.(\mathtt{in}(c, z).P_2 \mid B_2)$, $B' \equiv \nu\tilde{m}.(P_2\{M/_z\} \mid B_2)$, and $\tilde{m} \cap (fv(M) \cup fn(M)) = \emptyset$. Moreover, we have already seen that $\tilde{n} \cap (fn(B'_1) \cup fv(B'_1)) = \emptyset$. Hence, we have that:

$$\begin{array}{rcl}
C'_1[B'_1] & \equiv & \nu\tilde{n}'_1.[\nu\tilde{n}.(\mathtt{out}(c, M).P \mid D) \mid B'_1] \\
& \equiv & \nu\tilde{n}'_1.\nu\tilde{n}.[\mathtt{out}(c, M).P \mid D \mid B'_1] \\
& \equiv & \nu\tilde{n}'_1.\nu\tilde{n}.[\mathtt{out}(c, M).P \mid D \mid \nu\tilde{m}.(\mathtt{in}(c, z).P_2 \mid B_2)] \\
& \rightarrow & \nu\tilde{n}'_1.\nu\tilde{n}.(P \mid D \mid \nu\tilde{m}.(P_2\{^M/_z\} \mid B_2)) \\
& \equiv & C'[B']
\end{array}$$

*Case of a rule (*Comm*) between C'_1 (input) and A'_1 (output), i.e. $D'_1 \equiv \nu\tilde{n}.(\mathtt{in}(c, z).P \mid D)$, $A'_1 \equiv A'_2 = \nu\tilde{r}.(\mathtt{out}(c, M).Q \mid A_2)$ for some c, M, P, Q, D, A'_2, $\tilde{r}$, z, and A_2 such that z is a fresh variable, $fv(M) = \emptyset$, $\tilde{n} \cap (fn(A'_2) \cup fv(A'_2)) = \emptyset$, $\tilde{r} \cap (fn(\mathtt{in}(c, z).P) \cup fv(\mathtt{in}(c, z).P)) = \emptyset$. We assume in addition that names and variables in $\tilde{n}$ do not occur in $fn(B)$, $fv(B)$, and tr.* In such a case, we have that :

$$\begin{array}{rcl}
C'_1[A'_1] & \equiv & \nu\tilde{n}'_1.[\nu\tilde{n}.(\mathtt{in}(c, z).P \mid D) \mid A'_2] \\
& \equiv & \nu\tilde{n}'_1.\nu\tilde{n}.(\mathtt{in}(c, z).P \mid D \mid A'_2) \\
& \equiv & \nu\tilde{n}'_1.\nu\tilde{n}.[\mathtt{in}(c, z).P \mid D \mid \nu\tilde{r}.(\mathtt{out}(c, M).Q \mid A_2)] \\
& \equiv & \nu\tilde{n}'_1.\nu\tilde{n}.[\nu\tilde{r}.(P\{^M/_z\} \mid Q \mid A_2) \mid D]
\end{array}$$

and $A'' \equiv \nu\tilde{n}'_1.\nu\tilde{n}.[\nu\tilde{r}.(P\{^M/_z\} \mid Q \mid A_2) \mid D]$.

To determine C', A' and tr, we distinguish several cases depending on the term M:

- *M is a name of channel type and $M \notin \tilde{r}$.* In such a case, we have that $A'' \equiv \nu\tilde{n}'_1.\nu\tilde{n}.[P\{^M/_z\} \mid D \mid \nu\tilde{r}.(Q \mid A_2)]$. Thus, let $C'[_] = \nu\tilde{n}'_1.\nu\tilde{n}.[P\{^M/_z\} \mid D \mid _]$, $A' = \nu\tilde{r}.(Q \mid A_2)$ and $\mathsf{tr} = \mathsf{tr}_1 \cdot \mathtt{out}(c, M)$. Clearly, we have that $A'' \equiv C'[A']$ and $A'_1 \xrightarrow{\mathtt{out}(c,M)} A'$.

  Lastly, let B' be a closed process such that $B \overset{\mathsf{tr}}{\Rightarrow} B'$ and $\Phi(B') \sim \Phi(A')$. We have that there exists B'_1 such that $B \overset{\mathsf{tr}_1}{\Rightarrow} B'_1 \xrightarrow{\mathtt{out}(c,M)} B'$. Moreover, we can assume w.l.o.g. that $\tilde{n}$ do not occur in $fn(B'_1)$, $fv(B'_1)$, and tr_1. We have also that $\Phi(A'_1) \sim \Phi(B'_1)$. Thus, we can apply our inductive hypothesis on B'_1 which means that $C[B] \rightarrow^* C'_1[B'_1]$. In order to conclude, it remains to show that $C'_1[B'_1] \rightarrow C'[B']$.

  We have seen that $B'_1 \xrightarrow{\mathtt{out}(c,M)} B'$. Hence, we know that there exists $\tilde{m}, Q_2, B_2$ such that $B'_1 \equiv \nu\tilde{m}.(\mathtt{out}(c, M).Q_2 \mid B_2)$, $B' \equiv \nu\tilde{m}.(Q_2 \mid B_2)$, $M \notin \tilde{m}$ and $\tilde{m} \cap (fv(D'_1) \cup fn(D'_1) = \emptyset$. Therefore, we have that:

$$\begin{array}{rcl}
C'_1[B'_1] & \equiv & \nu\tilde{n}'_1.[\nu\tilde{n}.(\mathtt{in}(c, z).P \mid D) \mid B'_1) \\
& \equiv & \nu\tilde{n}'_1.\nu\tilde{n}.[\mathtt{in}(c, z).P \mid D \mid B'_1] \\
& \equiv & \nu\tilde{n}'_1.\nu\tilde{n}.[\mathtt{in}(c, z).P \mid D \mid \nu\tilde{m}.(\mathtt{out}(c, M).Q_2 \mid B_2)] \\
& \rightarrow & \nu\tilde{n}'_1.\nu\tilde{n}.[P\{^M/_z\} \mid D \mid \nu\tilde{m}.(Q_2 \mid B_2)] \\
& \equiv & C'[B']
\end{array}$$

- *M is a name of channel type that occurs in $\tilde{r}$.* Let $\tilde{r}'$ be a sequence such that $\tilde{r} = \tilde{r'} \uplus M$. In such a case, we have that $A'' \equiv \nu\tilde{n}_1'.\nu\tilde{n}.\nu M'.[P\{^{M'}/_z\} \mid D \mid \nu\tilde{r}'.(Q\{^{M'}/_M\} \mid A_2\{^{M'}/_M\})]$ where $\tilde{M'}$ is a fresh name of channel type. Let $C'[_] = \nu\tilde{n}_1'.\nu\tilde{n}.\nu M'.[P\{^{M'}/_z\} \mid D \mid _]$, $A' = \nu\tilde{r}'.(Q\{^{M'}/_M\} \mid A_2\{^{M'}/_M\})$ and $\mathsf{tr} = \mathsf{tr}_1 \cdot \nu M'.\mathtt{out}(c, M')$. Clearly, we have that $A_1' \equiv \nu\tilde{r}'.\nu M.(\mathtt{out}(c, M).Q \mid A_2) \xrightarrow{\nu M'.\mathtt{out}(c,M')} \nu\tilde{r}'.(Q\{^{M'}/_M\} \mid A_2\{^{M'}/_M\}) \equiv A'$ and $A'' \equiv C'[A']$.

  Lastly, let B' be a closed process such that $B \overset{\mathsf{tr}}{\Rightarrow} B'$ and $\Phi(B') \sim \Phi(A')$. We have that there exists B_1' such that $B \overset{\mathsf{tr}_1}{\Rightarrow} B_1' \xrightarrow{\nu M'.\mathtt{out}(c,M')} B'$. Moreover, we can assume w.l.o.g. that $\tilde{n}$ do not occur in $fn(B_1')$, $fv(B_1')$, and tr_1. We also have that $\Phi(A_1') \sim \Phi(B_1')$. Thus, we can apply our induction hypothesis on B_1' which means that $C[B] \rightarrow^* C_1'[B_1']$. In order to conclude, it remains to show that $C_1'[B_1'] \rightarrow C'[B']$.

  We have seen that $B_1' \xrightarrow{\nu M'.\mathtt{out}(c,M')} B'$. Hence, we know that there exists $\tilde{m}, Q_2, B_2$ such that $B_1' \equiv \nu M'.\nu\tilde{m}.(\mathtt{out}(c, M').Q_2 \mid B_2)$, $B' \equiv \nu\tilde{m}.(Q_2 \mid B_2)$, $M' \notin \tilde{m}$ and $\tilde{m} \cap (fv(D_1') \cup fn(D_1') = \emptyset$. Therefore, we have that (where M'' is a fresh channel name):

$$\begin{array}{rcl}
C_1'[B_1'] & \equiv & \nu\tilde{n}_1'.[\nu\tilde{n}.(\mathtt{in}(c, z).P \mid D) \mid B_1') \\
& \equiv & \nu\tilde{n}_1'.\nu\tilde{n}.[\mathtt{in}(c, z).P \mid D \mid B_1'] \\
& \equiv & \nu\tilde{n}_1'.\nu\tilde{n}.[\mathtt{in}(c, z).P \mid D \mid \nu M'.\nu\tilde{m}.(\mathtt{out}(c, M').Q_2 \mid B_2)] \\
& \rightarrow & \nu\tilde{n}_1'.\nu\tilde{n}.\nu M''.[P\{^{M''}/_z\} \mid D \mid \nu\tilde{m}.(Q_2\{^{M''}/_{M'}\} \mid B_2\{^{M''}/_{M'}\})] \\
& \equiv & C'[B']
\end{array}$$

- *M is a term of base type.* In such a case, we have that $A'' \equiv \nu\tilde{n}_1'.\nu\tilde{n}.\nu z.[P \mid D \mid \nu\tilde{r}.(Q \mid A_2 \mid \{^M/_z\})]$. Let $C'[_] = \nu\tilde{n}_1'.\nu\tilde{n}.\nu z.[P \mid D \mid _]$, $A' = \nu\tilde{r}.(Q \mid A_2 \mid \{^M/_z\})$ and $\mathsf{tr} = \mathsf{tr}_1 \cdot \nu z.\mathtt{out}(c, z)$. Clearly, we have that $A_1' \equiv \nu\tilde{r}.(\mathtt{out}(c, M).Q \mid A_2) \xrightarrow{\nu z.\mathtt{out}(c,z)} \nu\tilde{r}.(Q \mid A_2 \mid \{^M/_z\}) \equiv A'$ and $A'' \equiv C'[A']$.

  Lastly, let B' be a closed process such that $B \overset{\mathsf{tr}}{\Rightarrow} B'$ and $\Phi(B') \sim \Phi(A')$. We have that there exists B_1' such that $B \overset{\mathsf{tr}_1}{\Rightarrow} B_1' \xrightarrow{\nu z.\mathtt{out}(c,z)} B'$. Moreover, we can assume w.l.o.g. that $\tilde{n}$ do not occur in $fn(B_1')$, $fv(B_1')$, and tr_1. We also have that $\Phi(A_1') \sim \Phi(B_1')$. Thus, we can apply our induction hypothesis on B_1' which means that $C[B] \rightarrow^* C_1'[B_1']$. In order to conclude, it remains to show that $C_1'[B_1'] \rightarrow C'[B']$.

  We have seen that $B_1' \xrightarrow{\nu z.\mathtt{out}(c,z)} B'$. Hence, we know that there exists $\tilde{m}, Q_2, B_2$ such that $B_1' \equiv \nu\tilde{m}.(\mathtt{out}(c, M).Q_2 \mid B_2)$, $B' \equiv \nu\tilde{m}.(Q_2 \mid B_2 \mid \{^M/_z\})$ and $\tilde{m} \cap (fv(D_1') \cup fn(D_1') = \emptyset$. Therefore, we have that:

$$\begin{array}{rcl}
C_1'[B_1'] & \equiv & \nu\tilde{n}_1'.[\nu\tilde{n}.(\mathtt{in}(c, z).P \mid D) \mid B_1') \\
& \equiv & \nu\tilde{n}_1'.\nu\tilde{n}.[\mathtt{in}(c, z).P \mid D \mid B_1'] \\
& \equiv & \nu\tilde{n}_1'.\nu\tilde{n}.[\mathtt{in}(c, z).P \mid D \mid \nu\tilde{m}.(\mathtt{out}(c, M).Q_2 \mid B_2)] \\
& \rightarrow & \nu\tilde{n}_1'.\nu\tilde{n}.\nu z.[P \mid D \mid \nu\tilde{m}.(Q_2 \mid B_2 \mid \{^M/_z\})] \\
& \equiv & C'[B']
\end{array}$$

□

Trace equivalence is closed by one-to-one renamings of free names. This is formally stated in the lemma below:

Lemma 4. *Let A and B be two closed extended processes such that $A \approx_t B$ and u be a name (resp. variable) that occurs in $fn(A) \cup fv(A) \cup fn(B) \cup fv(B)$ and not in $bn(A) \cup bv(A) \cup bn(B) \cup bv(B)$, and u' be a fresh name (resp. variable). We have that $A\{^{u'}/_u\} \approx_t B\{^{u'}/_u\}$.*

We are now able to show the following theorem.

Theorem 4. *Let A and B two closed extended processes. We have that:*

$$A \approx_t B \text{ implies that } A \approx_m B.$$

Proof. Let A, B be two closed extended processes such that $A \approx_t B$. Let $C[_]$ be an evaluation context closing for A (and B), and c be a channel name. We assume w.l.o.g. that $C[_] = \nu\tilde{n}.(D \mid \nu\tilde{m}._)$ for some extended process D and for some sequences of names and variables $\tilde{n}$, and $\tilde{m}$. We assume w.l.o.g. that $\tilde{m} \cap (bn(A) \cup bv(A)) = \emptyset$ and $\tilde{m} \cap (bn(B) \cup bv(B)) = \emptyset$. Let $A_2 = A\{^{\tilde{m}'}/_{\tilde{m}}\}$ and $B_2 = B\{^{\tilde{m}'}/_{\tilde{m}}\}$ where $\tilde{m}'$ is a sequence of fresh names and variables. Thanks to Lemma 4, we have that $A_2 \approx_t B_2$. Let $C_2[_] = \nu\tilde{n}.\nu\tilde{m}'.(D \mid _)$. We have that $C[A] \equiv C_2[A_2]$ and $C[B] \equiv C_2[B_2]$.

Assume now that $C[A]\Downarrow_c$. This means that there exist a closing evaluation context C_1 that does not bind c, a term M, and a plain process P such that $C[A] \equiv C_2[A_2] \rightarrow^* C_1[\mathtt{out}(c, M).P]$. Applying Proposition 1 on A_2, B_2 and $C_2[_]$, we know that there exist a closed extended process A_2', a closing evaluation context $C_2'[_] = \nu\tilde{r}.(E \mid _)$ for A_2' and $\mathsf{tr} \in (\mathcal{A} \setminus \{\tau\})^*$ such that $C_1[\mathtt{out}(c, M).P] \equiv C_2[A_2']$, and $A_2 \overset{\mathsf{tr}}{\Rightarrow} A_2'$, and for all closed extended process B_2' such that $B_2 \overset{\mathsf{tr}}{\Rightarrow} B_2'$ and $\Phi(B_2') \sim \Phi(A_2')$, we have that $C_2[B_2] \mapsto^* C_2'[B_2']$. Moreover, we assume w.l.o.g. that $bn(\mathsf{tr}) \cap fn(B_2) = \emptyset$.

Since $C_2' = \nu\tilde{r}.(E|_)$, we can deduce from $C_1[\mathtt{out}(c, M).P] \equiv C_2'[A_2']$ that the output $\mathtt{out}(c, M)$ comes either from the process E or from A_2'. We distinguish these two cases:

- *The output $\mathtt{out}(c, M)$ comes from E.* Since, we have that $A_2 \approx_t B_2$, we know that that there exists B_2' such that $B_2 \overset{\mathsf{tr}}{\Rightarrow} B_2'$ and $\Phi(A_2') \sim \Phi(B_2')$. Therefore, we have that $C_2[B_2] \mapsto^* C_2'[B_2'] \equiv \nu\tilde{r}.(E \mid B_2')$. But by hypothesis, we know that the output $\mathtt{out}(c, M)$ comes from E and $c \notin \tilde{r}$. Hence we have that $C_2[B_2]\Downarrow_c$, and since $C[B] \equiv C_2[B_2]$, we conclude that $C[B]\Downarrow_c$.

- *The output $\mathtt{out}(c, M)$ comes from A_2'.* Thus, we have that $A_2' \equiv \nu\tilde{v}.(\mathtt{out}(c, M).P \mid A_3)$ with $c \notin \tilde{v}, \tilde{r}$. Thus, we have that $A_2' \xrightarrow{\nu z.\mathtt{out}(c,z)} \nu\tilde{v}.(P \mid A_3 \mid \{^M/_z\})$ (if M is a term of channel type, the transition is different but the proof can be done in a similar way.) Let $A'' = \nu\tilde{v}.(P \mid A_3 \mid \{^M/_z\})$ and $\mathsf{tr}' = \mathsf{tr} \cdot \nu z.\mathtt{out}(c, z)$, we have that $A_2 \overset{\mathsf{tr}'}{\Rightarrow} A''$. Since we have that $A_2 \approx_t B_2$, we have that there exists B_2' such that $B_2 \overset{\mathsf{tr}'}{\Rightarrow} B_2'$ and $\Phi(A'') \sim \Phi(B_2')$. Thus, we can deduce that there exists B' such that $B_2 \overset{\mathsf{tr}}{\Rightarrow} B' \xrightarrow{\nu z.\mathtt{out}(c,z)} B_2'$. Therefore, we have that there exists a term N, an evaluation context C_3 and a process Q such that $B' \equiv C_3[\mathtt{out}(c, N).Q]$ and c is not bind by C_3. Furthermore, we have that $\Phi(A_2') \sim \Phi(B')$ which means that $C_2[B_2] \rightarrow^* C_2'[B']$, and thus $C_2[B_2] \rightarrow^* C_2'[C_3[\mathtt{out}(c, N).Q]]$. Hence, we have that $C_2[B_2]\Downarrow_c$, and since $C[B] \equiv C_2[B_2]$, we conclude that $C[B]\Downarrow_c$. □

B. Intermediate calculus

As usual, a *context* is an expression (a plain process or an extended process) with a hole. A context C is closing for P when $C[P]$ is closed. A substitution σ is closing for P when $P\sigma$ is closed. First, we have the following result that is fairly standard in the process calculi. It has been proved in [19] and also in [4, Appendix C.3] for the spi calculus.

Lemma 5. *Let P and Q be two plain processes (not necessarily closed) such that $P\sigma \approx Q\sigma$ for any substitution σ closing for P and Q. We have that $C[P] \approx C[Q]$ for any context C closing for P and Q.*

Proposition 2. *Let A be a closed extended process. There exists a plain process P that does not contain name restriction, and a context evaluation C that only contains name restrictions, parallel compositions and active substitutions such that $P \approx C[A]$.*

Proof. We first define a relation $\mathcal{R}$ between closed extended processes and we show that $A \mathcal{R} B$ implies that $A \approx B$. We then use this relation to conclude the proof.

Definition of the relation $\mathcal{R}$. We define $\mathcal{R}$ as the smallest equivalence relation on closed extended process such that for any variable x, any terms M, N, any channel name c, any plain processes P, Q, we have that:

1. $C[\nu a.\mathtt{in}(c,x).P] \mathcal{R} C[\mathtt{in}(c,x).\nu a.P]$ when $a \neq c$ and C is a closing context for $\mathtt{in}(c,x).P$;
2. $C[\nu a.\mathtt{out}(c,M).P] \mathcal{R} C[\mathtt{out}(c,M).\nu a.P]$ when $a \notin fn(M) \cup \{c\}$ and C is a closing context for $\mathtt{out}(c,M).P$;
3. $C[\mathtt{if}\ M = N\ \mathtt{then}\ \nu a.P\ \mathtt{else}\ Q] \mathcal{R} C[\nu a.\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q]$ when $a \notin fn(Q) \cup fn(M,N)$ and C is a closing context for $\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q$;
4. $C[\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ \nu a.Q] \mathcal{R} C[\nu a.\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q]$ when $a \notin fn(P) \cup fn(M,N)$ and C is a closing context for $\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q$.
5. $A \mathcal{R} B$ when A, B are closed extended processes such that $A \equiv B$;

Claim: $A \mathcal{R} B$ implies $A \approx B$. Actually, thanks to Lemma 5, it is sufficient to prove that the following relations hold (assuming that the processes under study are closed):

1. $\nu a.\mathtt{in}(c,x).P \approx \mathtt{in}(c,x).\nu a.P$ when $a \neq c$ and
2. $\nu a.\mathtt{out}(c,M).P \approx \mathtt{out}(c,M).\nu a.P$ when $a \notin fn(M) \cup \{c\}$;
3. $\mathtt{if}\ M = N\ \mathtt{then}\ \nu a.P\mathtt{else}\ Q \approx \nu a.\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q$ when $a \notin fn(Q) \cup fn(M,N)$;
4. $\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ \nu a.Q \approx \nu a.\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q$ when $a \notin fn(P) \cup fn(M,N)$;

We show each item separately. More precisely, in each case, we define an equivalence relation $\mathcal{R}_1$ (resp. $\mathcal{R}_2$, $\mathcal{R}_3$, and $\mathcal{R}_4$) on closed extended processes that satisfies the three conditions of Definition 5.

1. Consider the relation $\mathcal{R}_1$ such that $\nu a.\mathtt{in}(c,x).P \mathcal{R}_1 \mathtt{in}(c,x).\nu a.P$ for any plain process P with $fv(P) \subseteq \{x\}$, any channel name c, and any name $a \neq c$;
2. Consider the relation $\mathcal{R}_2$ such that $\nu a.\mathtt{out}(c,M).P \mathcal{R}_2 \mathtt{out}(c,M).\nu a.P$ for any closed plain process P, any channel name c, any term M, and any name $a \notin fn(M) \cup \{c\}$;
3. Consider the relation $\mathcal{R}_3$ such that
$$\mathtt{if}\ M = N\ \mathtt{then}\ \nu a.P\ \mathtt{else}\ Q \mathcal{R}_1 \nu a.\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q$$
for any closed plain processes P, Q, any ground terms M, N, any name $a \notin fn(Q) \cup fn(M,N)$;
4. Consider the relation $\mathcal{R}_4$ such that
$$\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ \nu a.Q \mathcal{R}_1 \nu a.\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q$$
for any closed plain processes P, Q, any ground terms M, N, any name $a \notin fn(P) \cup fn(M,N)$.

We can check that $\mathcal{R}_1$ (resp. $\mathcal{R}_2$, $\mathcal{R}_3$, and $\mathcal{R}_4$) satisfies the three conditions of Definition 5.

Conclusion: Given an extended process A, by definition of this relation $\mathcal{R}$, it is easy to find a plain process P that does not contain name restriction, and a context evaluation C, that contains only name restrictions, parallel compositions and active substitutions, such that $A \mathcal{R} C[P]$, and thus $A \approx C[P]$. □

C. Symbolic calculus

Proposition 4 (soundness). *Let* $(\mathcal{E}_1;\mathcal{P}_1;\Phi_1;\mathcal{D}_1)$*, and* $(\mathcal{E}_2;\mathcal{P}_2;\Phi_2;\mathcal{D}_2)$ *be two symbolic processes such that*

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

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

1. $\theta_1 \in \mathsf{Sol}((\mathcal{E}_1;\Phi_1;\mathcal{D}_1))$*, 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* $(\mathcal{E}_1;\mathcal{P}_1;\Phi_1;\mathcal{D}_1)$ *(resp.* $(\mathcal{E}_2;\mathcal{P}_2;\Phi_2;\mathcal{D}_2)$*).*

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

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

In the following, we will denote by λ_{θ_1} (resp. λ_{θ_2}) the first-order solution associated to θ_1 (resp. θ_2) and $(\mathcal{E}_1;\Phi_1;\mathcal{D}_1)$ (resp. $(\mathcal{E}_2;\Phi_2;D_2)$).

Case $\textsc{Then}_s$**.** In such a case there exist u, v, $\mathcal{Q}_1$, and $\mathcal{Q}_2$ such that $\mathcal{P}_1 = \{\texttt{if } u = v \texttt{ then } \mathcal{Q}_1 \texttt{ else } \mathcal{Q}_2\} \uplus \mathcal{P}$, $\mathcal{D}_2 = \mathcal{D}_1 \cup \{u =^?_{\mathsf{E}} v\}$, $\mathcal{P}_2 = \mathcal{Q}_1 \uplus \mathcal{P}$, $\mathcal{E}_1 = \mathcal{E}_2$, $\Phi_1 = \Phi_2$, and $\alpha_s = \tau$

1. Since $\mathsf{var}^2(\mathcal{D}_1) = \mathsf{var}^2(\mathcal{D}_2)$, we have $\theta_1 = \theta_2$. Furthermore, $\theta_2 \in \mathsf{Sol}((\mathcal{E}_2;\Phi_2;\mathcal{D}_2))$ implies that θ_2 satisfies the constraints of $\mathcal{D}_2 = \mathcal{D}_1 \cup \{u =^?_{\mathsf{E}} v\}$, and so satisfies $\mathcal{D}_1$. At last, since $\Phi_1 = \Phi_2$ and $\mathcal{E}_1 = \mathcal{E}_2$, we have that $\theta_1 \in \mathsf{Sol}((\mathcal{E}_1;\Phi_1;\mathcal{D}_1))$.
2. Since $\theta_2 \in \mathsf{Sol}((\mathcal{E}_2;\Phi_2;\mathcal{D}_1 \cup \{u =^?_{\mathsf{E}} v\}))$, $\theta_2 = \theta_1$, $\Phi_2 = \Phi_1$, we have $\lambda_{\theta_1} = \lambda_{\theta_2}$ and thus $u\lambda_{\theta_1} =_{\mathsf{E}} v\lambda_{\theta_1}$. Hence we have that:

   $$(\mathcal{E}_1;\{\texttt{if } u\lambda_{\theta_1} = v\lambda_{\theta_1} \texttt{ then } \mathcal{Q}_1\lambda_{\theta_1} \texttt{ else } \mathcal{Q}_2\lambda_{\theta_1}\} \uplus \mathcal{P}\lambda_{\theta_1};\Phi_1\lambda_{\theta_1}) \xrightarrow{\tau}_i (\mathcal{E}_1;\mathcal{Q}_1\lambda_{\theta_2} \uplus \mathcal{P}\lambda_{\theta_2};\Phi_1\lambda_{\theta_2}),$$

   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 $(\mathcal{E}_1;\mathcal{P}_1;\Phi_1;\mathcal{D}_1)$ (resp. $(\mathcal{E}_2;\mathcal{P}_2;\Phi_2;\mathcal{D}_2)$).

The case of the rule $\textsc{Else}_s$ can be done in a similar way.

Case $\textsc{Comm}_s$**.** In such a case, there exist p, q, u, x, $\mathcal{Q}_1$, $\mathcal{Q}_2$, and $\mathcal{P}$ such that $\mathcal{D}_2 = \mathcal{D}_1 \cup \{p =^?_{\mathsf{E}} q\}$, $\Phi_2 = \Phi_1$, $\mathcal{E}_2 = \mathcal{E}_1$, $\mathcal{P}_1 = \{\texttt{out}(p,u).\mathcal{Q}_1;\texttt{in}(q.x).\mathcal{Q}_2\} \uplus \mathcal{P}$, $\mathcal{P}_2 = \mathcal{Q}_1 \uplus \mathcal{Q}_2\{x \mapsto u\} \uplus \mathcal{P}$, and $\alpha_s = \tau$.

1. Since $\mathsf{var}^2(\mathcal{D}_1) = \mathsf{var}^2(\mathcal{D}_2)$, we have $\theta_1 = \theta_2$. Furthermore, $\theta_2 \in \mathsf{Sol}((\mathcal{E}_2;\Phi_2;\mathcal{D}_2))$ implies that θ_2 satisfies the constraints of $\mathcal{D}_2 = \mathcal{D}_1 \cup \{p =^?_{\mathsf{E}} q\}$, and so satisfies $\mathcal{D}_1$. At last, since $\Phi_1 = \Phi_2$ and $\mathcal{E}_1 = \mathcal{E}_2$, we have that $\theta_1 \in \mathsf{Sol}((\mathcal{E}_1;\Phi_1;\mathcal{D}_1))$.
2. Since $\theta_2 \in \mathsf{Sol}((\mathcal{E}_2;\Phi_2;\mathcal{D}_1 \cup \{p =^?_{\mathsf{E}} q\}))$, $\Phi_1 = \Phi_2$ and $\theta_1 = \theta_2$, we have that $\lambda_{\theta_1} = \lambda_{\theta_2}$ thus $p\lambda_{\theta_1} = q\lambda_{\theta_1}$. Hence, we have that:

   $$\begin{aligned}(\mathcal{E}_1;\{\texttt{out}(p\lambda_{\theta_1}, u\lambda_{\theta_1}).\mathcal{Q}_1\lambda_{\theta_1};\texttt{in}(p\lambda_{\theta_1}.x).\mathcal{Q}_2\lambda_{\theta_1}\} \uplus \mathcal{P}\lambda_{\theta_1};\Phi_1\lambda_{\theta_1})\\ \xrightarrow{\tau}_i (\mathcal{E}_1;\mathcal{Q}_1\lambda_{\theta_2} \uplus \mathcal{Q}_2\lambda_{\theta_2}\{x \mapsto u\lambda_{\theta_2}\} \uplus \mathcal{P}\lambda_{\theta_2};\Phi_1\lambda_{\theta_2}).\end{aligned}$$

   Since $\mathcal{Q}_2\lambda_{\theta_2}\{x \mapsto u\lambda_{\theta_2}\} = (\mathcal{Q}_2\{x \mapsto u\})\lambda_{\theta_2}$, we have that $(\mathcal{E}_1;\mathcal{P}_1';\Phi_1') \xrightarrow{\alpha_s\theta_2}_i (\mathcal{E}_2;\mathcal{P}_2';\Phi_2')$ 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 $(\mathcal{E}_1;\mathcal{P}_1;\Phi_1;\mathcal{D}_1)$ (resp. $(\mathcal{E}_2;\mathcal{P}_2;\Phi_2;\mathcal{D}_2)$).

Case $\textsc{In}_s$**.** In such a case, there exist p, x, $\mathcal{Q}$, $\mathcal{P}$ and fresh variables y, z and Y, Z with $ar(Y) = ar(Z) = |\Phi_1|$ and such that $\alpha_s = \texttt{in}(Z,Y)$, $\mathcal{P}_1 = \{\texttt{in}(p,x).\mathcal{Q}\} \uplus \mathcal{P}$, $\mathcal{P}_2 = \mathcal{Q}\{x \mapsto y\} \uplus \mathcal{P}$, $\mathcal{D}_2 = \mathcal{D}_1 \cup \{Y \rhd^? y\,; Z \rhd^? z\,; z =^?_{\mathsf{E}} p\}$, $\mathcal{E}_1 = \mathcal{E}_2$ and $\Phi_1 = \Phi_2$.

1. We have that $\mathsf{var}^2(\mathcal{D}_1) = \mathsf{var}^2(\mathcal{D}_2) \smallsetminus \{Y\ ; Z\}$, $\mathsf{var}^1(\mathcal{D}_1) = \mathsf{var}^1(\mathcal{D}_2) \smallsetminus \{y\ ; z\}$ and $\Phi_1 = \Phi_2$. Since $\theta_2 \in \mathsf{Sol}((\mathcal{E}_2; \Phi_2; \mathcal{D}_2))$, we have that θ_2 satisfies the constraints of $\mathcal{D}_2$ and so satisfies the constraints of $\mathcal{D}_1$. With $\Phi_1 = \Phi_2$ and $\mathcal{E}_1 = \mathcal{E}_2$, it implies that $\theta_1 \in \mathsf{Sol}((\mathcal{E}_1; \Phi_1; \mathcal{D}_1))$ with $\lambda_{\theta_1} = \lambda_{\theta_2}|_{\mathsf{var}^1(\mathcal{D}_1)}$.
2. Let $M = Y\theta_2$ and $u = y\lambda_{\theta_2}$. We have that $fn(M) \cap \mathcal{E}_1 = \emptyset$ since $\theta_2 \in \mathsf{Sol}((\mathcal{E}_2; \Phi_2; \mathcal{D}_2))$ and $\mathcal{E}_2 = \mathcal{E}_1$. We have also that $u = y\lambda_{\theta_2} = M(\Phi_2\lambda_{\theta_2})$. Since y and z are fresh variables, this means that $\mathsf{var}^1(\Phi_2) \subseteq \mathsf{var}^1(\mathcal{D}_1)$. Thus, we have that $u = M(\Phi_2\lambda_{\theta_2}) = M(\Phi_2\lambda_{\theta_1}) = M(\Phi_1\lambda_{\theta_1})$. Futhermore, by definition of a solution, we have that $fv(M) \subseteq \mathrm{dom}(\Phi'_1)$. Lastly, since $\theta_2 \in \mathsf{Sol}((\mathcal{E}_2; \Phi_2; \mathcal{D}_2))$, we have that $(Z\theta_2)(\Phi\lambda_{\theta_2}) = z\lambda_{\theta_2}$, $z\lambda_{\theta_2} =_{\mathsf{E}} p\lambda_{\theta_2}$ and $Z\theta_2 \in \mathcal{T}(\mathcal{N} \smallsetminus \{\mathcal{E}_2\}, \mathrm{dom}(\Phi_2))$. But p is a term of type channel, thus so does $p\lambda_{\theta_2}$. Since all the function symbol operate on and return term of base type and since all terms in Φ_2 are base type, we can deduce that $(Z\theta_2) \in \mathcal{N} \smallsetminus \{\mathcal{E}_2\}$ and so $Z\theta_2 = p\lambda_{\theta_2}$ with $p\lambda_{\theta_2} \notin \mathcal{E}_1$ ($\mathcal{E}_1 = \mathcal{E}_2$). Furthermore, since $\lambda_{\theta_1} = \lambda_{\theta_2}|_{\mathsf{var}^1(\mathcal{D}_1)}$ and $\mathsf{var}^1(\mathcal{D}_1) = \mathsf{var}^1(\mathcal{D}_2) \smallsetminus \{y\ ; z\}$ where y, z are fresh variables, then p is either a name or a variable in $\mathsf{var}^1(\mathcal{D}_1)$ and we have that $p\lambda_{\theta_2} = p\lambda_{\theta_1}$. Hence, we have that

$$(\mathcal{E}_1; \{\mathtt{in}(p\lambda_{\theta_1}, x).\mathcal{Q}\lambda_{\theta_1}\} \uplus \mathcal{P}\lambda_{\theta_1}; \Phi_1\lambda_{\theta_1}) \xrightarrow{in(Z\theta_2, M)}_i (\mathcal{E}_1; \mathcal{Q}\lambda_{\theta_1}\{x \mapsto u\} \uplus \mathcal{P}\lambda_{\theta_1}; \Phi_1\lambda_{\theta_1}),$$

i.e. $(\mathcal{E}_1; \mathcal{P}_1\lambda_{\theta_1}; \Phi_1\lambda_{\theta_1}) \xrightarrow{\alpha_s\theta_2}_i (\mathcal{E}_2; (\mathcal{Q}\{x \mapsto y\})\lambda_{\theta_2} \uplus \mathcal{P}\lambda_{\theta_2}; \Phi_2\lambda_{\theta_2})$ since $\mathcal{E}_2 = \mathcal{E}_1$, $\Phi_2 = \Phi_1$, $\mathsf{var}^1(\Phi_1) = \mathsf{var}^1(\Phi_2)$ and $\lambda_{\theta_2} = \lambda_{\theta_1} \cup \{y \mapsto u\ ; z \mapsto p\lambda_{\theta_1}\}$. Hence, we have that $(\mathcal{E}_1; \mathcal{P}'_1; \Phi'_1) \xrightarrow{\alpha_s\theta_2}_i (\mathcal{E}_2; \mathcal{P}'_2; \Phi'_2)$ 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 $(\mathcal{E}_1; \mathcal{P}_1; \Phi_1; \mathcal{D}_1)$ (resp. $(\mathcal{E}_2; \mathcal{P}_2; \Phi_2; \mathcal{D}_2)$).

Case Out-T$_s$**.** In such a case, there exist p, u, $\mathcal{Q}$, $\mathcal{P}$, fresh variables Z, z with $ar(Z) = |\Phi_1|$ and w_l such that $l = |\Phi_1| + 1$, $\alpha_s = \nu w_l.out(Z, w_l)$, $\mathcal{P}_1 = \{\mathtt{out}(p, u).\mathcal{Q}\} \uplus \mathcal{P}$, $\mathcal{P}_2 = \mathcal{Q} \uplus \mathcal{P}$, $\mathcal{D}_2 = \mathcal{D}_1 \cup \{Z \rhd^? z\ ; z =^?_{\mathsf{E}} p\}$, $\mathcal{E}_2 = \mathcal{E}_1$, and $\Phi_2 = \Phi_1 \cup \{w_l \rhd u\}$.

1. We have that $\mathsf{var}^2(\mathcal{D}_1) = \mathsf{var}^2(\mathcal{D}_2) \smallsetminus \{Z\}$ thus $\theta_1 = \theta_2|_{\mathsf{var}^2(\mathcal{D}_1)}$ and w_l does not occur in θ_1 since $ar(X) \leq |\Phi_1|$, for any $X \in \mathrm{dom}(\theta_1)$. Furthermore, we have that $\mathsf{var}^1(\mathcal{D}_1) = \mathsf{var}^1(\mathcal{D}_2) \smallsetminus \{z\}$. Since $\theta_2 \in \mathsf{Sol}((\mathcal{E}_2; \Phi_2; \mathcal{D}_2))$ we have that θ_2 satisfies the constraints of $\mathcal{D}_2$ and so satisfies the constraints of $\mathcal{D}_1$. In addition, since w_l does not occur in θ_1, $\Phi_2 = \Phi_1 \cup \{w_l \rhd u\}$ and $\mathcal{E}_1 = \mathcal{E}_2$, we can conclude that $\theta_1 \in \mathsf{Sol}((\mathcal{E}_1; \Phi_1; \mathcal{D}_1))$ with $\lambda_{\theta_1} = \lambda_{\theta_2}|_{\mathsf{var}^1(\mathcal{D}_1)}$.
2. Since $\theta_2 \in \mathsf{Sol}((\mathcal{E}_2; \Phi_2; \mathcal{D}_2))$, we have that $(Z\theta_2)(\Phi\lambda_{\theta_2}) = z\lambda_{\theta_2}$, $z\lambda_{\theta_2} =_{\mathsf{E}} p\lambda_{\theta_2}$ and $(Z\theta_2) \in \mathcal{T}(\mathcal{N} \smallsetminus \{\mathcal{E}_2\}, \mathrm{dom}(\Phi_2))$. Since all the function symbol operate on and return term of base type and since all terms of Φ_2 are base type, we can deduce that $(Z\theta_2) \in \mathcal{N} \smallsetminus \{\mathcal{E}_2\}$ and so $Z\theta_2 = p\lambda_{\theta_2}$ with $p\lambda_{\theta_2} \notin \mathcal{E}_1$ ($\mathcal{E}_1 = \mathcal{E}_2$). Furthermore, since $\lambda_{\theta_1} = \lambda_{\theta_2}|_{\mathsf{var}^1(\mathcal{D}_1)}$ and $\mathsf{var}^1(\mathcal{D}_1) = \mathsf{var}^1(\mathcal{D}_2) \smallsetminus \{z\}$ where z is a fresh variable, we have that p is either a name or a variable in $\mathsf{var}^1(\mathcal{D}_1)$ and we have that $p\lambda_{\theta_2} = p\lambda_{\theta_1}$.
   Let $v = u\lambda_{\theta_1}$. We have that:

$$(\mathcal{E}_1; \{\mathtt{out}(p\lambda_{\theta_1}, u\lambda_{\theta_1}).\mathcal{Q}\lambda_{\theta_1}\} \uplus \mathcal{P}\lambda_{\theta_1}; \Phi_1\lambda_{\theta_1}) \xrightarrow{\nu w_l.out(Z\theta_2, w_l)}_i (\mathcal{E}_1; \mathcal{Q}\lambda_{\theta_1} \uplus \mathcal{P}\lambda_{\theta_1}; \Phi_1\lambda_{\theta_1} \cup \{w_l \rhd u\lambda_{\theta_1}\}),$$

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

Case Out-Ch$_s$**.** In such a case, there exist p, c, $\mathcal{Q}$, fresh variables Z, Y, z, y with $ar(Y) = ar(Z) = |\Phi_1|$, and $\mathcal{P}_1 = \{\mathtt{out}(p, c).\mathcal{Q}\} \uplus \mathcal{P}$, $\mathcal{P}_2 = \mathcal{Q} \uplus \mathcal{P}$, $\mathcal{E}_2 = \mathcal{E}_1$, $\Phi_2 = \Phi_1$, $\mathcal{D}_2 = \mathcal{D}_1 \cup \{Z \rhd^? z\ ; Y \rhd^? y\ ; z =^?_{\mathsf{E}} p\ ; y =^?_{\mathsf{E}} c\}$, $\alpha_s = \mathtt{out}(Z, Y)$ and $c \notin \mathcal{E}_1$.

1. We have that $\mathsf{var}^2(\mathcal{D}_1) = \mathsf{var}^2(\mathcal{D}_2) \smallsetminus \{Z\ ; Y\}$, $\mathsf{var}^1(\mathcal{D}_1) = \mathsf{var}^1(\mathcal{D}_2) \smallsetminus \{z\ ; y\}$. Since $\theta_2 \in \mathsf{Sol}((\mathcal{E}_2; \Phi_2; \mathcal{D}_2))$, we have that θ_2 satisfies the constraints of $\mathcal{D}_2$ and so satisfies the constraints of $\mathcal{D}_1$. Since $\Phi_1 = \Phi_2$ and $\mathcal{E}_1 = \mathcal{E}_2$, it implies that $\theta_1 \in \mathsf{Sol}((\mathcal{E}_1; \Phi_1; \mathcal{D}_1))$ with $\lambda_{\theta_1} = \lambda_{\theta_2}|_{\mathsf{var}^1(\mathcal{D}_1)}$.

2. Since $\theta_2 \in \mathsf{Sol}((\mathcal{E}_2;\Phi_2;\mathcal{D}_2))$, we have that $(Y\theta_2)(\Phi_2\lambda_{\theta_2}) = y\lambda_{\theta_2}$, $y\lambda_{\theta_2} = c\lambda_{\theta_2}$ and $(Y\theta_2) \in \mathcal{T}(\mathcal{N} \smallsetminus \{\mathcal{E}_2\}, \mathrm{dom}(\Phi_2))$. Since all the function symbol operate on and return term of base type and since all terms of Φ_2 are base type, we can deduce that $(Y\theta_2) \in \mathcal{N} \smallsetminus \{\mathcal{E}_2\}$ and so $Y\theta_2 = c\lambda_{\theta_2}$ with $c\lambda_{\theta_2} \notin \mathcal{E}_1$ ($\mathcal{E}_1 = \mathcal{E}_2$). Furthermore, since $\lambda_{\theta_1} = \lambda_{\theta_2}|_{\mathsf{var}^1(\mathcal{D}_1)}$ and $\mathsf{var}^1(\mathcal{D}_1) = \mathsf{var}^1(\mathcal{D}_2) \smallsetminus \{z\,;y\}$ where z, y are fresh variables, we have that c is either a name or a variable in $\mathsf{var}^1(\mathcal{D}_1)$ and we have that $c\lambda_{\theta_2} = c\lambda_{\theta_1}$.
Lastly, $\theta_2 \in \mathsf{Sol}((\mathcal{E}_2;\Phi_2;\mathcal{D}_2))$ also implies that $(Z\theta_2)(\Phi_2\lambda_{\theta_2}) = z\lambda_{\theta_2}$ and $z\lambda_{\theta_2} =_{\mathsf{E}} p\lambda_{\theta_2}$. We apply the same reasoning we used with $Y\theta_2$ to prove that $Z\theta_2 = p\lambda_{\theta_1}$ with $p\lambda_{\theta_1} \notin \mathcal{E}_1$.
Hence, we have that:

$$(\mathcal{E}_1;\{\mathtt{out}(p\lambda_{\theta_1}, c).\mathcal{Q}\lambda_{\theta_1}\} \uplus \mathcal{P}\lambda_{\theta_1};\Phi_1\lambda_{\theta_1}) \xrightarrow{\mathtt{out}(Z\theta_2, Y\theta_2)}_i (\mathcal{E}_1;\mathcal{Q}\lambda_{\theta_1} \uplus \mathcal{P}\lambda_{\theta_1};\Phi_1\lambda_{\theta_1}).$$

Hence, we have that $(\mathcal{E}_1;\mathcal{P}_1';\Phi_1') \xrightarrow{\alpha_s\theta_2}_i (\mathcal{E}_2;\mathcal{P}_2';\Phi_2')$ 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 $(\mathcal{E}_1;\mathcal{P}_1;\Phi_1;\mathcal{D}_1)$ (resp. $(\mathcal{E}_2;\mathcal{P}_2;\Phi_2;\mathcal{D}_2)$).

Case Open-Ch$_s$**.** In such a case, there exist p, c, $\mathcal{Q}$, $\mathcal{P}$ and fresh variables Z, z with $ar(Z) = |\Phi_1|$ such that $p \notin \mathcal{E}_1$, $c \in \mathcal{E}_1$, $\mathcal{P}_1 = \{\mathtt{out}(p, c).\mathcal{Q}\} \uplus \mathcal{P}$, $\mathcal{P}_2 = \mathcal{Q} \uplus \mathcal{P}$, $\mathcal{E}_2 = \mathcal{E}_1$, $\Phi_2 = \Phi_1$, $\mathcal{D}_2 = \mathcal{D}_1 \cup \{Z \rhd^? z\,; z =^?_{\mathsf{E}} p\}$, and $\alpha_s = \mathtt{out}(Z, c)$.

1. We have that $\mathsf{var}^2(\mathcal{D}_1) = \mathsf{var}^2(\mathcal{D}_2) \smallsetminus \{Z\}$, $\mathsf{var}^1(\mathcal{D}_1) = \mathsf{var}^1(\mathcal{D}_2) \smallsetminus \{z\}$. Since $\theta_2 \in \mathsf{Sol}((\mathcal{E}_2;\Phi_2;\mathcal{D}_2))$, we have that θ_2 satisfies the constraints of $\mathcal{D}_2$ and so satisfies the constraints of $\mathcal{D}_1$. Since $\Phi_1 = \Phi_2$ and $\mathcal{E}_1 = \mathcal{E}_2$, it implies that $\theta_1 \in \mathsf{Sol}((\mathcal{E}_1;\Phi_1;\mathcal{D}_1))$ with $\lambda_{\theta_1} = \lambda_{\theta_2}|_{\mathsf{var}^1(\mathcal{D}_1)}$.
2. Since $\theta_2 \in \mathsf{Sol}((\mathcal{E}_2;\Phi_2;\mathcal{D}_2))$, we have that $(Z\theta_2)(\Phi\lambda_{\theta_2}) = z\lambda_{\theta_2}$, $z\lambda_{\theta_2} = p\lambda_{\theta_2}$ and $(Z\theta_2) \in \mathcal{T}(\mathcal{N} \smallsetminus \{\mathcal{E}_2\}, \mathrm{dom}(\Phi_2))$. Since all the function symbol operate on and return term of base type and since all terms of Φ_2 are base type, we can deduce that $(Z\theta_2) \in \mathcal{N} \smallsetminus \{\mathcal{E}_2\}$ and so $Z\theta_2 = p\lambda_{\theta_2}$ with $p\lambda_{\theta_2} \notin \mathcal{E}_1$ ($\mathcal{E}_1 = \mathcal{E}_2$). Furthermore, since $\lambda_{\theta_1} = \lambda_{\theta_2}|_{\mathsf{var}^1(\mathcal{D}_1)}$ and $\mathsf{var}^1(\mathcal{D}_1) = \mathsf{var}^1(\mathcal{D}_2) \smallsetminus \{z\}$ where z is a fresh variable, then p is either a name or a variable and we have that $p\lambda_{\theta_2} = p\lambda_{\theta_1}$. Hence, we have that:

$$(\mathcal{E}_1;\{\mathtt{out}(p\lambda_{\theta_1}, c).\mathcal{Q}\lambda_{\theta_1}\} \uplus \mathcal{P}\lambda_{\theta_1};\Phi_1\lambda_{\theta_1}) \xrightarrow{\nu ch_n.\mathtt{out}(Z\theta_2, ch_n)}_i (\mathcal{E}_1;(\mathcal{Q}\lambda_{\theta_1})\{c \mapsto ch_n\} \uplus \mathcal{P}\lambda_{\theta_1};\Phi_1\lambda_{\theta_1}).$$

Since ch_n is fresh name, we have that $(\mathcal{Q}\lambda_{\theta_1})\{c \mapsto ch_n\} = (\mathcal{Q}\{c \mapsto ch_n\})\lambda_{\theta_1}$. Hence, we have that $(\mathcal{E}_1;\mathcal{P}_1';\Phi_1') \xrightarrow{\alpha_s\theta_2}_i (\mathcal{E}_2;\mathcal{P}_2';\Phi_2')$ 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 $(\mathcal{E}_1;\mathcal{P}_1;\Phi_1;\mathcal{D}_1)$ (resp. $(\mathcal{E}_2;\mathcal{P}_2;\Phi_2;\mathcal{D}_2)$). □

Proposition 5 (completeness). *Let $(\mathcal{E}_1;\mathcal{P}_1;\Phi_1;\mathcal{D}_1)$ be a symbolic process, $(\mathcal{E}_1;\mathcal{P}_1';\Phi_1')$ its θ_1-concretization where $\theta_1 \in \mathsf{Sol}((\mathcal{E}_1;\Phi_1;\mathcal{D}_1))$. 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 $(\mathcal{E}_2;\mathcal{P}_2;\Phi_2;\mathcal{D}_2)$, a substitution θ_2, and a symbolic action α_s such that:*

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

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

$$(\mathcal{E}_1;\mathcal{P}_1';\Phi_1') \xrightarrow{\alpha}_i (\mathcal{E};\mathcal{P};\Phi).$$

In the following, we will denote by λ_{θ_1} (resp. λ_{θ_2}) the first-order solution associated to θ_1 (resp. θ_2) and $(\mathcal{E}_1;\Phi_1;\mathcal{D}_1)$ (resp. $(\mathcal{E}_2;\Phi_2;D_2)$).
Case Then$_i$**.** In such a case we have that $\mathcal{E} = \mathcal{E}_1$ and there exist u', v', $\mathcal{Q}_1'$, $\mathcal{Q}_2'$, and $\mathcal{P}'$ such that $u' =_{\mathsf{E}} v'$, $\mathcal{P}_1' = \{\mathtt{if}\ u' = v'\ \mathtt{then}\ \mathcal{Q}_1'\ \mathtt{else}\ \mathcal{Q}_2'\} \uplus \mathcal{P}'$, $\mathcal{P} = \mathcal{Q}_1' \uplus \mathcal{P}'$, $\Phi = \Phi_1'$, and

$\alpha = \tau$. Since $(\mathcal{E}_1; \mathcal{P}'_1; \Phi'_1)$ is the θ_1-concretization of $(\mathcal{E}_1; \mathcal{P}_1; \Phi_1; \mathcal{D}_1)$, 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, $\mathcal{Q}_1$, $\mathcal{Q}_2$, and $\mathcal{P}_0$ such that $\mathcal{P}_1 = \{\texttt{if } u = v \texttt{ then } \mathcal{Q}_1 \texttt{ else } \mathcal{Q}_2\} \uplus \mathcal{P}_0$, and thus we have that $u\lambda_{\theta_1} = u'$, $v\lambda_{\theta_1} = v'$, $\mathcal{Q}_1\lambda_{\theta_1} = \mathcal{Q}'_1$, $\mathcal{Q}_2\lambda_{\theta_1} = \mathcal{Q}'_2$, and $\mathcal{P}_0\lambda_{\theta_1} = \mathcal{P}'$. Let $\mathcal{E}_2 = \mathcal{E}_1$, $\mathcal{P}_2 = \mathcal{Q}_1 \uplus \mathcal{P}_0$, $\Phi_2 = \Phi_1$, $\mathcal{D}_2 = \mathcal{D}_1 \cup \{u = v\}$, $\alpha_s = \tau$ and $\theta_2 = \theta_1$. We have that:

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

$$(\mathcal{E}_1; \{\texttt{if } u = v \texttt{ then } \mathcal{Q}_1 \texttt{ else } \mathcal{Q}_2\} \uplus \mathcal{P}_0; \Phi_1; \mathcal{D}_1) \xrightarrow{\alpha_s}_s (\mathcal{E}_1; \mathcal{Q}_1 \uplus \mathcal{P}_0; \Phi_1; \mathcal{D}_1 \cup \{u = v\}).$$

2. We have that $\mathsf{var}^2(\mathcal{D}_2) = \mathsf{var}^2(\mathcal{D}_1)$, $\mathcal{E}_2 = \mathcal{E}_1$ 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 $\mathcal{D}_2 = \mathcal{D}_1 \cup \{u = v\}$. 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 $\theta_2 \in \mathsf{Sol}((\mathcal{E}_2; \Phi_2; \mathcal{D}_2))$.
3. We have that

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

4. We have that $\alpha_s\theta_2 = \alpha_s = \alpha$.

The case of the rule $\textsc{Else}_i$ can be done in a similar way.

Case $\textsc{Comm}_i$**.** In such a case we have that $\mathcal{E} = \mathcal{E}_1$ and there exist p', u', x, $\mathcal{Q}'_1$, $\mathcal{Q}'_2$, and $\mathcal{P}'$ such that $\mathcal{P}'_1 = \{\texttt{out}(p', u').\mathcal{Q}'_1; \texttt{in}(p', x).\mathcal{Q}'_2\} \uplus \mathcal{P}'$, $\mathcal{P} = \mathcal{Q}'_1 \uplus \mathcal{Q}'_2\{x \mapsto u'\} \uplus \mathcal{P}'$, $\Phi = \Phi'_1$, and $\alpha = \tau$. Since $(\mathcal{E}_1; \mathcal{P}'_1; \Phi'_1)$ is the θ_1-concretization of $(\mathcal{E}_1; \mathcal{P}_1; \Phi_1; \mathcal{D}_1)$, 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, p, q, $\mathcal{Q}_1$, $\mathcal{Q}_2$, and $\mathcal{P}_0$ such that $\mathcal{P}_1 = \{\texttt{out}(p, u).\mathcal{Q}_1; \texttt{in}(q, x).\mathcal{Q}_2\} \uplus \mathcal{P}_0$, and thus we have that $u\lambda_{\theta_1} = u'$, $q\lambda_{\theta_1} = p\lambda_{\theta_1} = p'$, $\mathcal{Q}_1\lambda_{\theta_1} = \mathcal{Q}'_1$, $\mathcal{Q}_2\lambda_{\theta_1} = \mathcal{Q}'_2$, and $\mathcal{P}_0\lambda_{\theta_1} = \mathcal{P}'$. Let $\mathcal{E}_2 = \mathcal{E}_1$, $\mathcal{P}_2 = \mathcal{Q}_1 \uplus \mathcal{Q}_2\{x \mapsto u\} \uplus \mathcal{P}_0$, $\Phi_2 = \Phi_1$, $\mathcal{D}_2 = \mathcal{D}_1 \cup \{p =^?_{\mathsf{E}} q\}$, $\alpha_s = \tau$, $\theta_2 = \theta_1$ and $\lambda_{\theta_2} = \lambda_{\theta_1}$. We have that:

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

$$(\mathcal{E}_1; \{\texttt{out}(p, u).\mathcal{Q}_1; \texttt{in}(q, x).\mathcal{Q}_2\} \uplus \mathcal{P}_0; \Phi_1; \mathcal{D}_1) \xrightarrow{\alpha_s}_s (\mathcal{E}_1; \mathcal{Q}_1 \uplus \mathcal{Q}_2\{x \mapsto u\} \uplus \mathcal{P}_0; \Phi_1; \mathcal{D}_1 \cup \{p =^?_{\mathsf{E}} q\}).$$

2. Since $\mathcal{E}_2 = \mathcal{E}_1$, $\mathcal{D}_2 = \mathcal{D}_1 \cup \{p =^?_{\mathsf{E}} q\}$, $p\lambda_{\theta_2} = q\lambda_{\theta_2}$ and $\Phi_2 = \Phi_1$, we deduce that $\theta_2 \in \mathsf{Sol}((\mathcal{E}_2; \Phi_2; \mathcal{D}_2))$.
3. We have that

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

4. We have that $\alpha_s\theta_2 = \alpha_s = \alpha$.

Case $\textsc{In}_i$**.** In such a case we have that $\mathcal{E} = \mathcal{E}_1$, $\Phi = \Phi'_1$ and there exist p', x, $\mathcal{Q}'$, $\mathcal{P}'$, M and u such that $p' \notin \mathcal{E}_1$, $\mathcal{P}'_1 = \{\texttt{in}(p', x).\mathcal{Q}'\} \uplus \mathcal{P}'$, $\mathcal{P} = \mathcal{Q}'\{x \mapsto u\} \uplus \mathcal{P}'$, $M\Phi'_1 = u$, $fv(M) \subseteq \mathrm{dom}(\Phi'_1)$, $fn(M) \cap \mathcal{E}_1 = \emptyset$, and $\alpha = \texttt{in}(p', M)$. Since $(\mathcal{E}_1; \mathcal{P}'_1; \Phi'_1)$ is the θ_1-concretization of $(\mathcal{E}_1; \mathcal{P}_1; \Phi_1; \mathcal{D}_1)$, 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 p, $\mathcal{Q}$, $\mathcal{P}_0$ such that $\mathcal{P}_1 = \{\texttt{in}(p, x).\mathcal{Q}\} \uplus \mathcal{P}_0$ with $p\lambda_{\theta_1} = p'$, $\mathcal{Q}\lambda_{\theta_1} = \mathcal{Q}'$ and $\mathcal{P}_0\lambda_{\theta_1} = \mathcal{P}'$.

Let Y and Z be two second order variables with $ar(Y) = ar(Z) = |\Phi_1|$ and y, z be two fresh first-order variables. Let $\mathcal{E}_2 = \mathcal{E}_1$, $\mathcal{P}_2 = \mathcal{Q}\{x \mapsto y\} \uplus \mathcal{P}_0$, $\Phi_2 = \Phi_1$, $\mathcal{D}_2 = \mathcal{D}_1 \cup \{Y \rhd^? y\,; Z \rhd^? z\,; z =^?_{\mathsf{E}} p\}$ and $\alpha_s = \texttt{in}(Z, Y)$. Let θ_2 be the substitution such that $\theta_2 = \theta_1 \cup \{Y \mapsto M\,; Z \mapsto p'\}$. We have that:

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

$$\begin{aligned}&(\mathcal{E}_1; \{\texttt{in}(p, x).\mathcal{Q}\} \uplus \mathcal{P}_0; \Phi_1; \mathcal{D}_1) \xrightarrow{\texttt{in}(Z,Y)}_s \\ &\qquad (\mathcal{E}_1; \mathcal{Q}\{x \mapsto y\} \uplus \mathcal{P}_0; \Phi_1; \mathcal{D}_1 \cup \{Y \rhd^? y\,; Z \rhd^? z\,; z =^?_{\mathsf{E}} p\}).\end{aligned}$$

2. We have $\mathrm{dom}(\theta_2) = \mathsf{var}^2(\mathcal{D}_2)$ and $fn(\theta_2) \cap \mathcal{E}_2 = \emptyset$ since $fn(\theta_1) \cap \mathcal{E}_1 = \emptyset$, $fn(M) \cap \mathcal{E}_1 = \emptyset$ and $p' \notin \mathcal{E}_1$. Furthermore, since $fv(M) \subset \mathrm{dom}(\Phi_1)$, p' is a channel name and $ar(Z) = ar(Y) = |\Phi_1|$, we have that $Z\theta_2, Y\theta_2 \in \mathcal{T}(\mathcal{N} \smallsetminus \mathcal{E}_2, \mathrm{dom}(\Phi_1))$.
   Now, it remains to show that λ_{θ_2} satisfies the constraints in $\mathcal{D}_2$. Actually, we have that $\lambda_{\theta_2} = \lambda_{\theta_1} \cup \{y \mapsto u\ ; z \mapsto p'\}$ where y, z were fresh variable. Thus it implies that $\Phi_2\lambda_{\theta_2} = \Phi_2\lambda_{\theta_1}$, $p\lambda_{\theta_2} = p\lambda_{\theta_1}$ which allows us to conclude since $(Y\theta_2)(\Phi_2\lambda_{\theta_2}) = M(\Phi_2\lambda_{\theta_1}) = u = y\lambda_{\theta_2}$, $(Z\theta_2)(\Phi_1\lambda_{\theta_2}) = Z\theta_2 = p' = z\lambda_{\theta_2} = p\lambda_{\theta_2}$.
3. We have that

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

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

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

Let Z be a second order variable with $ar(Z) = |\Phi_1|$ and z be a fresh first order variable. Let $\mathcal{E}_2 = \mathcal{E}_1$, $\mathcal{P}_2 = \mathcal{Q} \uplus \mathcal{P}_0$, $\Phi_2 = \Phi_1 \cup \{w_l \rhd u\}$, $\mathcal{D}_2 = \mathcal{D}_1 \cup \{Z \rhd^? z\ ; z =^?_{\mathsf{E}} p\}$, $\alpha_s = \nu w_l.\mathtt{out}(Z, w_l)$ and $\theta_2 = \theta_1 \cup \{Z \mapsto p'\}$. We have that:

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

$$(\mathcal{E}_1; \{\mathtt{out}(p, u).\mathcal{Q}\} \uplus \mathcal{P}_0; \Phi_1; \mathcal{D}_1) \xrightarrow{\nu w_l.out(Z, w_l)}_s (\mathcal{E}_1; \mathcal{Q} \uplus \mathcal{P}_0; \Phi_1 \cup \{w_l \rhd u\}; \mathcal{D}_1 \cup \{Z \rhd^? z\ ; z =^?_{\mathsf{E}} p\}).$$

2. We have $\mathrm{dom}(\theta_2) = \mathsf{var}^2(\mathcal{D}_2)$ and $fn(\theta_2) \cap \mathcal{E}_2 = \emptyset$ since $fn(\theta_1) \cap \mathcal{E}_1 = \emptyset$ and $p' \notin \mathcal{E}_1$. Furthermore, for all $X \in \mathsf{var}^2(\mathcal{D}_1)$, we have that $ar(X) \leq |\Phi_1|$. At last, since $ar(Z) = |\Phi_1|$ and $Z\theta_2 = p'$, we deduce that $\lambda_{\theta_2} = \lambda_{\theta_1} \cup \{z \mapsto p'\}$, $z\lambda_{\theta_2} = p' = p\lambda_{\theta_1} = p\lambda_{\theta_2}$ and so $\theta_2 \in \mathsf{Sol}((\mathcal{E}_2; \Phi_2; \mathcal{D}_2))$.
3. We have that

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

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

Case Out-Ch$_i$**.** In such a case we have that $\mathcal{E} = \mathcal{E}_1$, $\Phi = \Phi'_1$ and there exist p', c', $\mathcal{Q}'$, and $\mathcal{P}'$ such that $\mathcal{P}'_1 = \{\mathtt{out}(p', c').\mathcal{Q}'\} \uplus \mathcal{P}'$, $\mathcal{P} = \mathcal{Q}' \uplus \mathcal{P}'$, $\Phi = \Phi'_1$, $p', c' \notin \mathcal{E}_1$, and $\alpha = \mathtt{out}(p', c')$. Since $(\mathcal{E}_1; \mathcal{P}'_1; \Phi'_1)$ is the θ_1-concretization of $(\mathcal{E}_1; \mathcal{P}_1; \Phi_1; \mathcal{D}_1)$, 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 p, c, $\mathcal{Q}$ and $\mathcal{P}_0$ such that $\mathcal{P}_1 = \{\mathtt{out}(p, c).\mathcal{Q}\} \uplus \mathcal{P}_0$ with $c\lambda_{\theta_1} = c'$, $p\lambda_{\theta_1} = p'$, $\mathcal{P}_0\lambda_{\theta_1} = \mathcal{P}'$ and $\mathcal{Q}\lambda_{\theta_1} = \mathcal{Q}'$.

Let Z and Y be second order variables with $ar(Y) = ar(Z) = |\Phi_1|$ and z, y be fresh first order variables. Let $\mathcal{E}_2 = \mathcal{E}_1$, $\mathcal{P}_2 = \mathcal{Q} \uplus \mathcal{P}_0$, $\Phi_2 = \Phi_1$, $\mathcal{D}_2 = \mathcal{D}_1 \cup \{Z \rhd^? z\ ; Y \rhd^? y\ ; z =^?_{\mathsf{E}} p\ ; y =^?_{\mathsf{E}} c\}$, $\alpha_s = \mathtt{out}(Z, Y)$, and $\theta_2 = \theta_1 \cup \{Z \mapsto p'\ ; Y \mapsto c'\}$. We have that:

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

$$(\mathcal{E}_1; \{\mathtt{out}(p, c).\mathcal{Q}\} \uplus \mathcal{P}_0; \Phi_1; \mathcal{D}_1) \xrightarrow{\mathtt{out}(Z, Y)}_s (\mathcal{E}_2; \mathcal{Q} \uplus \mathcal{P}_0; \Phi_1; \mathcal{D}_1 \cup \{Z \rhd^? z\ ; Y \rhd^? y\ ; z =^?_{\mathsf{E}} p\ ; y =^?_{\mathsf{E}} c\}).$$

2. First of all, we have $\mathrm{dom}(\theta_2) = \mathsf{var}^2(\mathcal{D}_2)$. Second, we know that p' and c' are both channel names such that $p', c' \notin \mathcal{E}_2$ ($\mathcal{E}_1 = \mathcal{E}_2$). Thus, it implies that $Z\theta_2, Y\theta_2 \in \mathcal{T}(\mathcal{N} \smallsetminus \mathcal{E}_2, \mathrm{dom}(\Phi_2))$. Now, it remains to show that λ_{θ_2} satisfies the constraints in $\mathcal{D}_2$. Actually, we have that $\lambda_{\theta_2} = \lambda_{\theta_1} \cup \{y \mapsto c' \,; z \mapsto p'\}$ where y,z were fresh variables. Thus it implies that $\Phi_2\lambda_{\theta_2} = \Phi_2\lambda_{\theta_1}$, $p\lambda_{\theta_2} = p\lambda_{\theta_1}$ and $c\lambda_{\theta_2} = c\lambda_{\theta_1}$ which allows us to conclude since
   - $(Y\theta_2)(\Phi_2\lambda_{\theta_2}) = c'(\Phi_2\lambda_{\theta_1}) = c' = y\lambda_{\theta_2}$ and so $y\lambda_{\theta_2} = c' = c\lambda_{\theta_1} = c\lambda_{\theta_2}$;
   - $(Z\theta_2)(\Phi_2\lambda_{\theta_2}) = p'(\Phi_2\lambda_{\theta_1}) = p' = z\lambda_{\theta_2}$ and so $z\lambda_{\theta_2} = p' = p\lambda_{\theta_1} = p\lambda_{\theta_2}$.
3. We have that
$$\begin{array}{rcl}(\mathcal{E}_2; \mathcal{P}_2\lambda_{\theta_2}; \Phi_2\lambda_{\theta_2}) & = & (\mathcal{E}_1; \mathcal{Q}\lambda_{\theta_1} \uplus \mathcal{P}_0\lambda_{\theta_1}; \Phi_1\lambda_{\theta_1}) \\ & = & (\mathcal{E}; \mathcal{Q}' \uplus \mathcal{P}'; \Phi_1') \\ & = & (\mathcal{E}; \mathcal{P}; \Phi)\end{array}$$
4. We have that $\alpha_s\theta_2 = \mathtt{out}(Z, Y)\theta_2 = \mathtt{out}(p', c') = \alpha$.

Case Open-Ch$_i$**.** In such a case we have that $\mathcal{E} = \mathcal{E}_1$, $\Phi = \Phi_1'$ and there exist p', c, $\mathcal{Q}'$, $\mathcal{P}'$ and a fresh channel name ch_n such that $\mathcal{P}_1' = \{\mathtt{out}(p', c).\mathcal{Q}'\} \uplus \mathcal{P}'$, $\mathcal{P} = (\mathcal{Q}' \uplus \mathcal{P}')\{c \mapsto ch_n\}$, $\Phi = \Phi_1'$, $p' \notin \mathcal{E}_1$, $c \in \mathcal{E}_1$, and $\alpha = \nu ch_n.\mathtt{out}(p, ch_n)$. Since $(\mathcal{E}_1; \mathcal{P}_1'; \Phi_1')$ is the θ_1-concretization of $(\mathcal{E}_1; \mathcal{P}_1; \Phi_1; \mathcal{D}_1)$, 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 p, $\mathcal{Q}$ and $\mathcal{P}_0$ such that $\mathcal{P}_1 = \{\mathtt{out}(p, c).\mathcal{Q}\} \uplus \mathcal{P}_0$ with $p\lambda_{\theta_1} = p'$, $\mathcal{P}_0\lambda_{\theta_1} = \mathcal{P}'$ and $\mathcal{Q}\lambda_{\theta_1} = \mathcal{Q}'$.

Let Z be a second order variable with $ar(Z) = |\Phi_1|$ and z be a fresh first order variable. Let $\mathcal{E}_2 = \mathcal{E}_1$, $\mathcal{P}_2 = (\mathcal{Q} \uplus \mathcal{P}_0)\{c \mapsto ch_n\}$, $\Phi_2 = \Phi_1$, $\mathcal{D}_2 = \mathcal{D}_1 \cup \{Z \rhd^? z \,; z =^?_{\mathsf{E}} p\}$, $\alpha_s = \nu ch_n.\mathtt{out}(Z, ch_n)$, and $\theta_2 = \theta_1 \cup \{Z \mapsto p'\}$. We have that:

1. $(\mathcal{E}_1; \mathcal{P}_1; \Phi_1; \mathcal{D}_1) \xrightarrow{\alpha_s}_s (\mathcal{E}_2; \mathcal{P}_2; \Phi_2; \mathcal{D}_2)$. Indeed, we have that
$$\begin{array}{l}(\mathcal{E}_1; \{\mathtt{out}(p, c).\mathcal{Q}\} \uplus \mathcal{P}_0; \Phi_1; \mathcal{D}_1) \xrightarrow{\nu ch_n.\mathtt{out}(Z, ch_n)}_s \\ \qquad\qquad (\mathcal{E}_2; (\mathcal{Q} \uplus \mathcal{P}_0)\{c \mapsto ch_n)\}; \Phi_1; \mathcal{D}_1 \cup \{Z \rhd^? z \,; z =^?_{\mathsf{E}} p\}).\end{array}$$
2. First of all, we have $\mathrm{dom}(\theta_2) = \mathsf{var}^2(\mathcal{D}_2)$. Second, we know that p' is a channel name such that $p' \notin \mathcal{E}_2$ ($\mathcal{E}_1 = \mathcal{E}_2$). Thus, it implies that $Z\theta_2 \in \mathcal{T}(\mathcal{N} \smallsetminus \mathcal{E}_2, \mathrm{dom}(\Phi_2))$. Now, it remains to show that λ_{θ_2} satisfies the constraints in $\mathcal{D}_2$. Actually, we have that $\lambda_{\theta_2} = \lambda_{\theta_1} \cup \{z \mapsto p'\}$ where z is a fresh variables. Thus it implies that $\Phi_1\lambda_{\theta_2} = \Phi_1\lambda_{\theta_1}$ and $p\lambda_{\theta_2} = p\lambda_{\theta_1}$ which allows us to conclude since $(Z\theta_2)(\Phi_1\lambda_{\theta_2}) = p'(\Phi_1\lambda_{\theta_1}) = p' = z\lambda_{\theta_2}$ and so $z\lambda_{\theta_2} = p' = p\lambda_{\theta_1} = p\lambda_{\theta_2}$.
3. We have that
$$\begin{array}{rcl}(\mathcal{E}_2; \mathcal{P}_2\lambda_{\theta_2}; \Phi_2\lambda_{\theta_2}) & = & (\mathcal{E}_1; (\mathcal{Q}\lambda_{\theta_1} \uplus \mathcal{P}_0\lambda_{\theta_1})\{c \mapsto ch_n\}; \Phi_1\lambda_{\theta_1}) \\ & = & (\mathcal{E}; (\mathcal{Q}' \uplus \mathcal{P}')\{c \mapsto ch_n\}; \Phi_1') \\ & = & (\mathcal{E}; \mathcal{P}; \Phi)\end{array}$$
4. We have that $\alpha_s\theta_2 = \nu ch_n.\mathtt{out}(Z, ch_n)\theta_2 = \nu ch_n.\mathtt{out}(p', ch_n) = \alpha$. □

D. Decidability of trace equivalence for standard primitives

We denote by $st(t)$ the subterms of t. The equationnal theory E_0 defined in Section 7.3 satisfies the following properties:

Proposition 9. *Let u and v be two ground terms, we have that:*

1. $\mathsf{fail} \in st(u)$ *implies* $u =_{\mathsf{E}_0} \mathsf{fail}$
2. $\mathsf{fail} \notin st(u)$ *and* $\mathsf{valid}(u)$*, if and only if,* $u \neq_{\mathsf{E}_0} \mathsf{fail}$
3. *if* $\mathsf{valid}(u)$,$\mathsf{valid}(v)$ *and* $\mathsf{fail} \notin st(u, v)$ *then* $u{\downarrow} = v{\downarrow}$*, if and only if,* $u =_{\mathsf{E}_0} v$.

Intuitively, in [20], the messages that are exchanged during an execution have to be valid. In order to fall into the class of constraint systems they consider, we have to restrict the class of processes we consider. From now on, we only consider plain intermediate processes that are *valid*, *i.e.* those that are generated using the following grammar (instead of using the grammar of plain intermediate processes that is given in Section 5.1).

$$\begin{array}{rcl} P, Q, R & := & 0 \\ & | & \texttt{if } M_1 = M_2 \texttt{ and } \mathsf{fail} \neq M_1 \texttt{ and } \mathsf{fail} \neq M_2 \texttt{ then } \mathcal{P} \texttt{ else } \mathcal{Q} \\ & | & \texttt{in}(u, x).\texttt{if } x = \mathsf{fail} \texttt{ then } 0 \texttt{ else } \mathcal{P} \\ & | & \texttt{if } N = \mathsf{fail} \texttt{ then } 0 \texttt{ else out}(u, N).\mathcal{P} \end{array}$$

where u is a metavariable of channel type, M_1, M_2 are terms having the same type, x is a variable of any type, N is a term of any type, and $\mathcal{P}$ (resp.$\mathcal{Q}$) is a multiset of plain intermediate processes that are valid. Moreover, we assume that the constant fail does not occur in M_1, M_2 and N. Then, a *valid intermediate processes* is defined in a similar way as an intermediate processes but considering plain intermediate processes that are valid. The logical connector `and` and the disequations $\neq$ are syntactic sugar that can be easily encoded using nested conditionals. However, when such an instruction is transformed into constraint systems, we obtain four constraint systems instead of the more usual two (one when the test fails and one when it succeeds). Thus, for sake of clarity in the proof, we will assume that $\texttt{if } u = v \texttt{ and } u \neq \mathsf{fail} \texttt{ and } v \neq \mathsf{fail}$ adds the formula $(u =^?_{\mathsf{E}_0} v \wedge u \neq^?_{\mathsf{E}_0} \mathsf{fail} \wedge v \neq^?_{\mathsf{E}_0} \mathsf{fail})$ in the constraint system for the rule $\textsc{Then}_s$ and $(u \neq^?_{\mathsf{E}_0} v \vee u =^?_{\mathsf{E}_0} \mathsf{fail} \vee v =^?_{\mathsf{E}_0} \mathsf{fail})$ for the rule $\textsc{Else}_s$.

We can now state some properties that are satisfied by any constraint system issued from a valid intermediate process.

Proposition 10. *Let A be a valid intermediate process. Let tr be a trace and let Σ be the set of constraint systems such that $(\mathsf{tr}, \Sigma) \in \mathsf{trace}_s(A_s)$ where A_s is the symbolic process associated to A. For all $\mathcal{C} = (\mathcal{E}, \Phi, \mathcal{D}) \in \Sigma$, we have that:*

- *for every $(X \rhd^? x) \in \mathcal{D}$, either $(x \neq^?_{\mathsf{E}_0} \mathsf{fail}) \in \mathcal{D}$ or x does not appear anywhere else in $\mathcal{C}$ unless possibly in an equation $(x =^?_{\mathsf{E}_0} \mathsf{fail}) \in \mathcal{D}$;*
- *for every $(w_i \rhd u) \in \Phi$, we have $(u \neq^?_{\mathsf{E}_0} \mathsf{fail}) \in \mathcal{D}$ and $\mathsf{fail} \notin st(u)$;*
- *the element in $\mathcal{D}$ are either deducibility constraints, or are of the form $(x =^?_{\mathsf{E}_0} \mathsf{fail})$, $(u \neq^?_{\mathsf{E}_0} \mathsf{fail})$, $(u =^?_{\mathsf{E}_0} v \wedge u \neq^?_{\mathsf{E}_0} \mathsf{fail} \wedge v \neq^?_{\mathsf{E}_0} \mathsf{fail})$ or $(u \neq^?_{\mathsf{E}_0} v \vee u =^?_{\mathsf{E}_0} \mathsf{fail} \vee v =^?_{\mathsf{E}_0} \mathsf{fail})$, for some variable x and some terms u,v that do not contain fail;*

Since the algorithm in [20] does not consider constants, the idea is to represent the constant fail as a public nonce. For this purpose, we introduce a special nonce n_{fail} that will be used for defining our transformation $\mathbf{Tr}(\cdot)$ that will allows us to transform two sets of constraint systems Σ and Σ' issued from valid intermediate processes into two sets $\mathbf{Tr}(\Sigma)$ and $\mathbf{Tr}(\Sigma')$ of constraint systems such that:

$$\Sigma \approx_s \Sigma' \text{ in } \mathsf{E}_0 \quad \Leftrightarrow \quad \mathbf{Tr}(\Sigma) \approx_s \mathbf{Tr}(\Sigma') \text{ using the algorithm in [20]}$$

Let $\mathcal{C} = (\mathcal{E}; \Phi; \mathcal{D})$ be a constraint system issued from a valid intermediate process. The transformation $\mathbf{Tr}(\mathcal{C})$ consists of removing or replacing (dis)equations of the form $u =^?_{\mathsf{E}_0} \mathsf{fail}$ (resp. $u \neq^?_{\mathsf{E}_0} \mathsf{fail}$). More precisely,

- we remove from $\mathcal{C}$ each equation (resp. disequation) of the form $u =^?_{\mathsf{E}_0} \mathsf{fail}$ (resp. $u \neq^?_{\mathsf{E}_0} \mathsf{fail}$) that corresponds to the additional check performed before an output;
- we remove from $\mathcal{C}$ the two disequations of the form $u \neq^?_{\mathsf{E}_0} \mathsf{fail}$ that correspond to the additional checks performed to pass a positive test;

- we remove from $\mathcal{C}$ the two equations of the form $u =^?_{\mathsf{E}_0} \mathsf{fail}$ that correspond to the checks performed to pass a negative test (more formally the equations are replaced by "false");
- we remove from $\mathcal{C}$ each equation of the form $x =^?_{\mathsf{E}_0} \mathsf{fail}$ that corresponds to the additional check performed after an input.
- we replace each disequation of the form $x \neq^?_{\mathsf{E}_0} \mathsf{fail}$ that corresponds to the additional check performed after an input with $x \neq^? n_{\mathsf{fail}}$.

We will denote $\mathbf{Tr}(\Sigma)$ the sets of constraint systems obtained by applying the transformation on each constraint system in Σ.

Therefore, the decision procedure for deciding trace equivalence on valid processes works as follows: We consider each symbolic trace tr_s

- Compute (in polynomial time) the sets of constraint systems Σ and Σ' such that $(\mathsf{tr}_s, \Sigma) \in \mathsf{trace}_s(A_s)$ and $(\mathsf{tr}_s, \Sigma') \in \mathsf{trace}_s(B_s)$;
- Check whether $\mathbf{Tr}(\Sigma)$ and $\mathbf{Tr}(\Sigma')$ are in symbolic equivalence using the algorithm in [20].

If there is a symbolic trace for which the two resulting sets of constraint systems are not in symbolic equivalence then return *no*, otherwise return *yes*.

We have already shown in Section 7.1 that two intermediate processes A and B are in trace equivalence if, and only if A and B are in symbolic trace equivalence (see Proposition 6). It remains to show that the algorithm proposed in [20] allows one to decide symbolic equivalence between our sets of constraint systems.

Actually, the algorithm proposed in [20] allows one to decide a notion of symbolic equivalence that is defined in the same way than $\approx_s$ replacing $\mathsf{Sol}(\mathcal{C})$ with $\mathsf{Sol}^{\mathsf{valid}}(\mathcal{C})$ and $\sim$ with $\sim^{\mathsf{valid}}$. We denote $\approx_s^{\mathsf{valid}}$ this notion of symbolic equivalence (namely *valid symbolic equivalence*) and the definitions of $\mathsf{Sol}^{\mathsf{valid}}(\mathcal{C})$ and $\sim^{\mathsf{valid}}$ are given below:

Definition 20. *A* valid solution *of a constraint system $\mathcal{C} = (\mathcal{E}; \Phi; \mathcal{D})$ is a substitution θ such that:*

- $\mathrm{dom}(\theta) = \mathsf{var}^2(\mathcal{C})$, *and*
- $X\theta \in \mathcal{T}(\mathcal{N}^+ \smallsetminus \{\mathcal{E}\}, \{w_1, \ldots, w_k\})$ *using function symbols in $\mathcal{F}_c \uplus \mathcal{F}_d$ for any $X \in \mathrm{dom}(\theta)$ with $ar(X) = k$.*

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

1. *for every $(X \rhd^? x) \in \mathcal{D}$, $(X\theta)(\Phi\lambda) = x\lambda$ and $\mathsf{valid}(x\lambda)$;*
2. *for every $(w_i \rhd u) \in \Phi$, $\mathsf{valid}(u\lambda)$;*
3. *for every $(s =^? s') \in \mathcal{D}$, $s\lambda\!\downarrow = s'\lambda\!\downarrow$, $\mathsf{valid}(s\lambda)$ and $\mathsf{valid}(s'\lambda)$;*
4. *for every $(s \neq^? s') \in \mathcal{D}$, $s\lambda\!\downarrow \neq s'\lambda\!\downarrow$, or $\neg\mathsf{valid}(s\lambda)$, or $\neg\mathsf{valid}(s'\lambda)$.*

where $\mathcal{N}^+ = \mathcal{N} \uplus \{n_{\mathsf{fail}}\}$. The set of valid solutions of a constraint system $\mathcal{C}$ is denoted $\mathsf{Sol}^{\mathsf{valid}}(\mathcal{C})$.

Definition 21. *A* term M is valid in a frame ϕ, *written* $(\mathsf{valid}(M))\phi$, *if there exists $\tilde{n}$ and a substitution σ such that $\phi \equiv \nu\tilde{n}.\sigma$, $\tilde{n} \cap fn(M) = \emptyset$, and $\mathsf{valid}(M\sigma)$.*

Two terms M and N are equal in a frame ϕ, *written $(M =_\downarrow N)\phi$, 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 $M\sigma\!\downarrow = N\sigma\!\downarrow$.*

Two closed frames ϕ_1 and ϕ_2 are in valid static equivalence, *denoted $\phi_1 \sim^{\mathsf{valid}} \phi_2$, when:*

- $\mathrm{dom}(\phi_1) = \mathrm{dom}(\phi_2)$, *and*
- *for all term M, we have that $\mathsf{valid}(M)\phi_1$, if and only if, $\mathsf{valid}(M)\phi_2$, and*
- *for all term M, N such that $\mathsf{valid}(M)\phi_1$, $\mathsf{valid}(N)\phi_1$, we have that $(M =_\downarrow N)\phi_1$, if and only if, $(M =_\downarrow N)\phi_2$.*

We first establish a lemma that allows us to show that static equivalence coincides with valid static equivalence when the frames do not involved the special constant fail and the special name n_{fail}.

Lemma 6. *Let ϕ_1 and ϕ_2 be two closed frames in which the constant* fail *and the name n_{fail} do not occur. We have that:*

$$\phi_1 \sim \phi_2, \text{ if and only if, } \phi_1 \sim^{\mathsf{valid}} \phi_2$$

Proof. We prove the two implications separately.

First implication: $\phi_1 \sim \phi_2$ implies $\phi_1 \sim^{\mathsf{valid}} \phi_2$. Since $\phi_1 \sim \phi_2$, we have that $\mathrm{dom}(\phi_1) = \mathrm{dom}(\phi_2)$.

Let M be a term such that $\mathsf{valid}(M)\phi_1$. We assume w.l.o.g. that $n_{\mathsf{fail}} \notin fn(M)$. Indeed, since n_{fail} does not occur in ϕ_1 and ϕ_2, we have that $\mathsf{valid}(M)\phi_1 \Leftrightarrow \mathsf{valid}(M')\phi_1$ and $\mathsf{valid}(M)\phi_2 \Leftrightarrow \mathsf{valid}(M')\phi_2$ where $M' = M\{^a/_{n_{\mathsf{fail}}}\}$ for some fresh name a. So, we can assume w.l.o.g. that $n_{\mathsf{fail}} \notin fn(M)$. By definition of $\mathsf{valid}(M)\phi_1$, we have that there exists $\tilde{n}_1$ and a substitution σ_1 such that $\phi_1 \equiv \nu\tilde{n}_1.\sigma_1$, $\tilde{n}_1 \cap fn(M) = \emptyset$ and $\mathsf{valid}(M\sigma_1)$. Moreover, we can assume that the constant fail and the name n_{fail} do not occur in $\nu\tilde{n}_1.\sigma_1$. Hence, we have that $n_{\mathsf{fail}} \notin fn(M\sigma_1)$. Furthermore, $\mathsf{valid}(M\sigma_1)$ implies that $M\sigma_1$ is a ground term. Thus, by Proposition 9 (item 2), we have that $\neg(M =_{\mathsf{E}_0} \mathsf{fail})\phi_1$.

By hypothesis, we can deduce that $\neg(M =_{\mathsf{E}_0} \mathsf{fail})\phi_2$ and so for any $\tilde{n}_2$ and substitution σ_2 such that $\phi_2 \equiv \nu\tilde{n}_2.\sigma_2$, and $\tilde{n}_2 \cap fn(M) = \emptyset$, we have that $\mathsf{valid}(M\sigma_2)$ and so $\mathsf{valid}(M)\phi_2$.

Let M and N two terms such that $\mathsf{valid}(M)\phi_1$ and $\mathsf{valid}(N)\phi_1$. Assume that $(M =_{\downarrow} N)\phi_1$. Once again, we assume w.l.o.g. that $n_{\mathsf{fail}} \notin fn(M, N)$. In such a case, there exists $\tilde{n}_1$ and a substitution σ_1 such that $\phi_1 \equiv \nu\tilde{n}_1.\sigma_1$, $\tilde{n}_1 \cap (fn(M) \cup fn(N)) = \emptyset$ and $M\sigma_1{\downarrow} = N\sigma_1{\downarrow}$. Hence, we have that $\mathsf{valid}(M\sigma_1)$, $\mathsf{valid}(N\sigma_1)$, and $M\sigma_1{\downarrow} = N\sigma_1{\downarrow}$ which means that $(M =_{\mathsf{E}_0} N)\phi_1$. Thus our hypothesis implies that $(M =_{\mathsf{E}_0} N)\phi_2$ and so there exists $\tilde{n}_2$, a substitution σ_2 such that $\phi_2 \equiv \nu\tilde{n}_2.\sigma_2$, $\tilde{n}_2 \cap (fn(M) \cup fn(N)) = \emptyset$ and $M\sigma_2 =_{\mathsf{E}_0} N\sigma_2$. Moreover, we can assume w.l.o.g. that fail does not occur in σ_2. We have that $M\sigma_2$ and $N\sigma_2$ are ground terms and fail does not occur in these terms. But by our hypothesis, we know that $\mathsf{valid}(M)\Phi_1$ and $\mathsf{valid}(N)\Phi_1$. We already proved that it implies $\mathsf{valid}(M)\Phi_2$ and $\mathsf{valid}(N)\Phi_2$, thus we have $\mathsf{valid}(M\sigma_2)$ and $\mathsf{valid}(N\sigma_2)$. Thanks to Proposition 9 (item 3), we deduce that: $M\sigma_2 =_{\mathsf{E}_0} N\sigma_2$ implies that $M\sigma_2{\downarrow} = N\sigma_2{\downarrow}$. We can conclude that $(M =_{\downarrow} N)\phi_2$.

Second implication: $\phi_1 \sim^{\mathsf{valid}} \phi_2$ implies $\phi_1 \sim \phi_2$. Let M, N two terms such that $(M =_{\mathsf{E}_0} N)\phi_1$. By definition, there exists $\tilde{n}$ and a substitution σ_1 such that $\phi_1 \equiv \nu\tilde{n}_1.\sigma_1$, $\tilde{n}_1 \cap (fn(M) \cup fn(N)) = \emptyset$ and $M\sigma_1 =_{\mathsf{E}_0} N\sigma_1$. We can assume w.l.o.g. that the constant fail and the name n_{fail} do not occur in $\nu\tilde{n}_1.\sigma_1$. We prove that $(M =_{\mathsf{E}_0} N)\phi_1$ implies that $(M =_{\mathsf{E}_0} N)\phi_2$ by induction on the proof tree witnessing the equality $M\sigma_1 =_{\mathsf{E}_0} N\sigma_1$.

- *Case $M\sigma_1$ and $N\sigma_1$ are ground term such that* $\mathsf{valid}(M\sigma_1)$, $\mathsf{valid}(N\sigma_1)$, *and $M\sigma_1{\downarrow} = N\sigma_1{\downarrow}$.* Thus we have that $\mathsf{valid}(M)\phi_1$, $\mathsf{valid}(M)\phi_1$ and $(M =_{\downarrow} N)\phi_1$. Thanks to our hypothesis, we can deduce that $\mathsf{valid}(M)\phi_2$, $\mathsf{valid}(M)\phi_2$ and $(M =_{\downarrow} N)\phi_2$, so there exists $\tilde{n}_2$ and a substitution σ_2 such that $\phi_2 \equiv \nu\tilde{n}_2.\sigma_2$, $\tilde{n}_2 \cap (fn(M) \cup fn(N) \cup \{n_{\mathsf{fail}}\}) = \emptyset$, $M\sigma_2{\downarrow} = N\sigma_2{\downarrow}$, $\mathsf{valid}(M\sigma_2)$ and $\mathsf{valid}(N\sigma_2)$. Therefore, by definition of $=_{\mathsf{E}_0}$, we have that $M\sigma_2 =_{\mathsf{E}_0} N\sigma_2$, and so $(M =_{\mathsf{E}_0} N)\phi_2$.

- *Case $M\sigma_1$ is a ground term with* $\neg\mathsf{valid}(M\sigma_1)$ *and* $N = \mathsf{fail}$. If fail occurs in $M\sigma_1$, then since fail does not occur in σ_1, we have that fail occur in M. Thus, for any $\tilde{n}_2$ and substitution σ_2 such that $\phi_2 \equiv \nu\tilde{n}_2.\sigma_2$ and $\tilde{n}_2 \cap fn(M) = \emptyset$, we have that fail occurs in $M\sigma_2$ and so $(M =_{\mathsf{E}_0} \mathsf{fail})\phi_2$, by Proposition 9 (item 1). Otherwise $\neg\mathsf{valid}(M\sigma_1)$ and so $\neg\mathsf{valid}(M)\phi_1$. Thus by our hypothesis, we deduce that $\neg\mathsf{valid}(M)\phi_2$ and so for any $\tilde{n}_2$ and substitution σ_2 such that $\tilde{n}_2 \cap fn(M) = \emptyset$, we have that $\neg\mathsf{valid}(M\sigma_2)$ and $M\sigma_2$ is a ground term, which means that $M\sigma_2 =_{\mathsf{E}_0} \mathsf{fail}$, and so $(M =_{\mathsf{E}_0} \mathsf{fail})\phi_2$.

The inductive cases (transitivity, closure by application of a context) are straightforward. □

Now, it remains to establish the link between the two definitions of solution of a constraint system. Proposition 10 gives us several properties that are satisfied by the constraint systems we are considering. Actually, we can consider constraint systems that satisfy stronger properties. Consider, for example, the process $P = \mathtt{in}(u, x).\mathtt{if}\ x = \mathsf{fail}\ \mathtt{then}\ 0\ \mathtt{else}\ \mathcal{P}$, for the symbolic labeled trace $\mathtt{in}(X, Y)$, our algorithm produces (at least) the following constraint systems:

- $\mathcal{C}_1$ with $\mathcal{D}_1 = \{X \rhd^? u\}$;
- $\mathcal{C}_2$ with $\mathcal{D}_2 = \{X \rhd^? u;\ x =^?_{\mathsf{E}_0} \mathsf{fail}\}$; and
- $\mathcal{C}_3$ with $\mathcal{D}_3 = \{X \rhd^? u;\ x \neq^?_{\mathsf{E}_0} \mathsf{fail}\}$.

The only difference between $\mathcal{D}_2$ and $\mathcal{D}_1$ is the addition of $x =^?_{\mathsf{E}_0} \mathsf{fail}$, thus we have that $\mathsf{Sol}(\mathcal{C}_2) \subseteq \mathsf{Sol}(\mathcal{C}_1)$. Furthermore, by definition of the transformation, we have that $\mathbf{Tr}(\mathcal{C}_1) = \mathbf{Tr}(\mathcal{C}_2)$. For these reasons, we will not consider constraint system like $\mathcal{D}_2$. This is formally stated below:

Proposition 11. *Let A be a valid intermediate process, tr be a trace and Σ be the set of constraint systems such that $(\mathsf{tr}, \Sigma) \in \mathit{trace}_s(A_s)$ where A_s is the symbolic process associated to A. For all $\mathcal{C} \in \Sigma$, there exists a constraint system $\mathcal{C}' = (\mathcal{E}'; \Phi'; \mathcal{D}')$ such that $\mathbf{Tr}(\mathcal{C}) = \mathbf{Tr}(\mathcal{C}')$ and $\mathsf{Sol}(\mathcal{C}) \subseteq \mathsf{Sol}(\mathcal{C}')$ that satisfies the following properties:*

- *for every $(X \rhd^? x) \in \mathcal{D}'$, either $(x \neq^?_{\mathsf{E}_0} \mathsf{fail}) \in \mathcal{D}'$ or x does not appear anywhere else in $\mathcal{C}'$;*
- *for every $(w_i \rhd u) \in \Phi'$, we have $(u \neq^?_{\mathsf{E}_0} \mathsf{fail}) \in \mathcal{D}$ and $\mathsf{fail} \notin st(u)$;*
- *the element in $\mathcal{D}'$ are either deducibility constraints, or are of the form $(u \neq^?_{\mathsf{E}_0} \mathsf{fail})$, $(u =^?_{\mathsf{E}_0} v \wedge u \neq^?_{\mathsf{E}_0} \mathsf{fail} \wedge v \neq^?_{\mathsf{E}_0} \mathsf{fail})$ or $(u \neq^?_{\mathsf{E}_0} v \vee u =^?_{\mathsf{E}_0} \mathsf{fail} \vee v =^?_{\mathsf{E}_0} \mathsf{fail})$, for some variable x and some terms u,v that do not contain fail;*
- *for all $(u \neq^?_{\mathsf{E}_0} \mathsf{fail}) \in \mathcal{D}'$, either u is a variable and there exists $(X \rhd^? u) \in \mathcal{D}'$, or there exist $(w_i \rhd u) \in \Phi'$.*

A constraint system that satisfies the four properties stated above is a valid constraint system.

With this notion of valid constraint system, we can now state and prove the two following lemmas that allow us to establish the link between the two notions of solutions.

Lemma 7. *Let $\mathcal{C} = (\mathcal{E}; \Phi; \mathcal{D})$ be a valid constraint system (satisfies all the properties given in Proposition 11). Let $\theta \in \mathsf{Sol}(\mathcal{C})$, λ be the first-order solution associated to θ and $\mathcal{C}$, and θ' be the substitution defined as follows:*

- $\mathrm{dom}(\theta) = \mathrm{dom}(\theta')$, *and*
- $X\theta' = n_{\mathsf{fail}}$, *for every X such that $(X\theta)(\Phi\lambda) =_{\mathsf{E}_0} \mathsf{fail}$, and*
- $X\theta' = X\theta$ *otherwise.*

We have that $\theta' \in \mathsf{Sol}^{\mathsf{valid}}(\mathbf{Tr}(\mathcal{C}))$ and if λ' is the first order solution associated to $\mathbf{Tr}(\mathcal{C})$ and θ', then $x\lambda' = n_{\mathsf{fail}}$ for every $x \in vars(\mathcal{C})$ such that $x\lambda =_{\mathsf{E}_0} \mathsf{fail}$, and $x\lambda' = x\lambda$ otherwise. Moreover, we have that $\Phi\lambda = \Phi\lambda'$.

Proof. We have that for all $X \in \mathrm{dom}(\theta)$ with $ar(X) = k$, $X\theta \in \mathcal{T}(\mathcal{F} \cup \mathcal{N} \smallsetminus \{\mathcal{E}\}, \{w_1, \ldots, w_k\})$. We first show that $X\theta'$ is built using function symbols in $\mathcal{F}_c \cup \mathcal{F}_d$, names in $(\mathcal{N} \cup \{n_{\mathsf{fail}}\}) \smallsetminus \{\mathcal{E}\}$, and variables in $\{w_1, \ldots, w_k\}$. Assume that the constant fail occurs in $X\theta$. In such a case, by Proposition 9 (item 1), we have that $(X\theta)(\Phi\lambda) =_{\mathsf{E}_0} \mathsf{fail}$ which means that $X\theta' = n_{\mathsf{fail}}$. Thus, if $(X\theta)(\Phi\lambda) \neq_{\mathsf{E}_0} \mathsf{fail}$, we have that fail does not occur in $X\theta$ which allows us to conclude.

Let λ' the substitution such that $x\lambda' = x\lambda$ if $x\lambda \neq_{\mathsf{E}_0} \mathsf{fail}$, and $x\lambda' = n_{\mathsf{fail}}$ otherwise. By Proposition 11, we know that for all $(X \rhd^? x) \in \mathcal{D}$, either x does not occur in Φ, or there exists

$(x \neq^?_{\mathsf{E}_0} \mathsf{fail}) \in \mathcal{D}$, and in such a case, we have that $x\lambda \neq_{\mathsf{E}_0} \mathsf{fail}$. Thus, for all $(X \rhd^? x) \in \mathcal{D}$, $x\lambda =_{\mathsf{E}_0} \mathsf{fail}$ implies that $x \notin vars(\Phi)$. Thus, for all $x \notin vars(\Phi)$, we have that $x\lambda = x\lambda'$ and so we deduce that $\Phi\lambda = \Phi\lambda'$. From this, we can deduce that for all X such that $(X\theta)(\Phi\lambda) \neq_{\mathsf{E}_0} \mathsf{fail}$, we have $(X\theta')(\Phi\lambda') = (X\theta)(\Phi\lambda) = x\lambda = x\lambda'$. At last, $x\lambda \neq_{\mathsf{E}_0} \mathsf{fail}$ implies that $\mathsf{valid}(x\lambda)$ and so $\mathsf{valid}(x\lambda')$.

By definition of θ', we have that $X\theta'(\Phi\lambda') = n_{\mathsf{fail}} = x\lambda'$ when $x\lambda =_{\mathsf{E}_0} \mathsf{fail}$ and we have that $\mathsf{valid}(x\lambda')$, for all $(X \rhd^? x) \in \mathcal{D}$.

Let $(w_i \rhd u) \in \Phi$. By Proposition 11, there exists $(u \neq^?_{\mathsf{E}_0} \mathsf{fail}) \in \mathcal{D}$. Since $\theta \in \mathsf{Sol}(\mathcal{C})$, we deduce that $\mathsf{valid}(u\lambda)$. For each $x \in vars(u)$, we have that $x \neq_{\mathsf{E}_0} \mathsf{fail}$ in $\mathcal{D}$, and so we have that $x\lambda' = x\lambda$. This allows us to conclude that $\mathsf{valid}(u\lambda')$.

Let $(u =^? v) \in \mathbf{Tr}(\mathcal{C})$. By definition of $\mathbf{Tr}(\mathcal{C})$, either $v = n_{\mathsf{fail}}$ and u is a variable, or else there exists $(u =^?_{\mathsf{E}_0} v \wedge u \neq^?_{\mathsf{E}_0} \mathsf{fail} \wedge v \neq^?_{\mathsf{E}_0} \mathsf{fail}) \in \mathcal{D}$. If $v = n_{\mathsf{fail}}$ and u is a variable, then there exists $(X \rhd^? u) \in \mathcal{D}$ with $X\theta' = n_{\mathsf{fail}}$ and $u\lambda' = n_{\mathsf{fail}}$ (indeed, in such a situation, we have that $u =^?_{\mathsf{E}_0} \mathsf{fail}$ in $\mathcal{D}$, and so $(X\theta)(\Phi\lambda) = u\lambda = \mathsf{fail}$). Hence, we have that $u\lambda' = n_{\mathsf{fail}}$ and $\mathsf{valid}(u\lambda')$. Assume now that there exists $(u =^?_{\mathsf{E}_0} v \wedge u \neq^?_{\mathsf{E}_0} \mathsf{fail} \wedge v \neq^?_{\mathsf{E}_0} \mathsf{fail}) \in \mathcal{D}$. Since $\theta \in \mathsf{Sol}(\mathcal{C})$, we deduce that $u\lambda =_{\mathsf{E}_0} v\lambda$, $u\lambda \neq_{\mathsf{E}_0} \mathsf{fail}$ and $v\lambda \neq_{\mathsf{E}_0} \mathsf{fail}$. By Proposition 9 (item 2), we $u\lambda \neq_{\mathsf{E}_0} \mathsf{fail}$ and $v\lambda \neq_{\mathsf{E}_0} \mathsf{fail}$ imply that $\mathsf{valid}(u\lambda)$ and $\mathsf{valid}(v\lambda)$. Thus, by Proposition 9 (item3) and since $u\lambda =_{\mathsf{E}_0} v\lambda$, we deduce that $u\lambda'{\downarrow} = v\lambda'{\downarrow}$.

Let $(u \neq^? v) \in \mathbf{Tr}(\mathcal{D})$. By definition of $\mathbf{Tr}(\mathcal{C})$, either $v = n_{\mathsf{fail}}$ and u is a variable, or else there exist $(u \neq^?_{\mathsf{E}_0} v \vee u =^?_{\mathsf{E}_0} \mathsf{fail} \vee v =^?_{\mathsf{E}_0} \mathsf{fail}) \in \mathcal{D}$. If $v = n_{\mathsf{fail}}$ and u is a variable, then there exists $(X \rhd^? u) \in \mathcal{D}$ with $X\theta' = X\theta$ and $u\lambda' = u\lambda$. Since $n_{\mathsf{fail}} \notin st(u\lambda)$, we can deduce that $u\lambda'{\downarrow} \neq n_{\mathsf{fail}}$. Assume now that there exists $(u \neq^?_{\mathsf{E}_0} v \vee u =^?_{\mathsf{E}_0} \mathsf{fail} \vee v =^?_{\mathsf{E}_0} \mathsf{fail}) \in \mathcal{D}$. In such a case, $\theta \in \mathsf{Sol}(\mathcal{C})$ implies that $u\lambda \neq_{\mathsf{E}_0} v\lambda$, or $u\lambda =_{\mathsf{E}_0} \mathsf{fail}$ or $v\lambda =_{\mathsf{E}_0} \mathsf{fail}$. Once again, by Proposition 9 (item 2) and 9 (item 3), we have that $u\lambda'{\downarrow} \neq v\lambda'{\downarrow}$, or $\neg\mathsf{valid}(u\lambda')$ or $\neg\mathsf{valid}(u\lambda')$.
This conclude the proof of $\theta' \in \mathsf{Sol}^{\mathsf{valid}}(\mathbf{Tr}(\mathcal{C}))$ and $\Phi\lambda = \Phi\lambda'$. □

Lemma 8. *Let $\mathcal{C} = (\mathcal{E}, \Phi, \mathcal{D})$ be a valid constraint system (satisfies all the properties given in Proposition 11). Let $\theta \in \mathsf{Sol}^{\mathsf{valid}}(\mathbf{Tr}(\mathcal{C}))$ and let λ be the first order solution associated to θ and $\mathbf{Tr}(\mathcal{C})$. Let θ' be the substitution that is defined as follows (where a is a fresh name in $\mathcal{N}$):*

- $\mathrm{dom}(\theta) = \mathrm{dom}(\theta')$, *and*
- $X\theta' = \mathsf{fail}$, *for every X such that $(X\theta)(\Phi\lambda){\downarrow} = n_{\mathsf{fail}}$, and*
- $X\theta' = X\theta\{^a/_{n_{\mathsf{fail}}}\}$ *otherwise.*

We have that $\theta' \in \mathsf{Sol}(\mathcal{C})$ and if λ' is the first order solution associated to θ' and $\mathcal{C}$, we have that $x\lambda' = x\lambda\{^a/_{n_{\mathsf{fail}}}\}$ for every x such that $x\lambda{\downarrow} \neq n_{\mathsf{fail}}$, and $x\lambda' = \mathsf{fail}$ otherwise. Moreover, we have that $\Phi\lambda' = \Phi\lambda\{^a/_{n_{\mathsf{fail}}}\}$.

Proof. Since $\theta \in \mathsf{Sol}(\mathbf{Tr}(\mathcal{C}))$, we have that for all $X \in \mathrm{dom}(\theta)$ with $ar(X) = k$, we have $X\theta$ is built using function symbols in $\mathcal{F}_c \cup \mathcal{F}_d$, names in $(\mathcal{N} \cup \{n_{\mathsf{fail}}\}) \smallsetminus \{\mathcal{E}\}$, and variables in $\{w_1, \ldots, w_k\}$. By definition of θ', for all X, we have that each occurence of n_{fail} in $X\theta$ is either replace with the fresh name a, or else $X\theta' = \mathsf{fail}$. Hence, we have that $X\theta'$ is built using function symbols in $\mathcal{F}$, names in $\mathcal{N} \smallsetminus \{\mathcal{E}\}$, and variables in $\{w_1, \ldots, w_k\}$.

Let λ' be the substitution such that $x\lambda' = x\lambda\{^a/_{n_{\mathsf{fail}}}\}$ for every x such that $x\lambda{\downarrow} \neq n_{\mathsf{fail}}$, and $x\lambda' = \mathsf{fail}$ otherwise. We know that n_{fail} does not occur in $\mathcal{C}$. Furthermore, by Proposition 11 and the definition of $\mathbf{Tr}(\mathcal{C})$, we know that for all $(X \rhd^? x) \in \mathbf{Tr}(\mathcal{D})$, either x does not occur in Φ or there exists $(x \neq^?_{\mathsf{E}_0} n_{\mathsf{fail}}) \in \mathbf{Tr}(\mathcal{D})$. But $\theta \in \mathsf{Sol}(\mathcal{C})$ which means that for each x that occurs in Φ, we have that $x\lambda{\downarrow} \neq n_{\mathsf{fail}}$ and so $x\lambda' = x\lambda\{^a/_{n_{\mathsf{fail}}}\}$. Hence, we can conclude that $\Phi\lambda' = \Phi\lambda\{^a/_{n_{\mathsf{fail}}}\}$ and for all $(X \rhd^? x) \in \mathbf{Tr}(\mathcal{D})$, $(X\theta)(\Phi\lambda){\downarrow} \neq n_{\mathsf{fail}}$ implies that $(X\theta')(\Phi\lambda') = (X\theta\{^a/_{n_{\mathsf{fail}}}\})(\Phi\lambda\{^a/_{n_{\mathsf{fail}}}\}) = (X\theta)(\Phi\lambda)\{^a/_{n_{\mathsf{fail}}}\} = x\lambda\{^a/_{n_{\mathsf{fail}}}\} = x\lambda'$.

By definition of our transformation, we know that the element of $\mathcal{D}$ are either constraints, or are of the form $(u =^?_{\mathsf{E}_0} \mathsf{fail})$, $(u \neq^?_{\mathsf{E}_0} \mathsf{fail})$, $(u =^?_{\mathsf{E}_0} v \wedge u \neq^?_{\mathsf{E}_0} \mathsf{fail} \wedge v \neq^?_{\mathsf{E}_0} \mathsf{fail})$ or $(u \neq^?_{\mathsf{E}_0} v \vee u =^?_{\mathsf{E}_0} \mathsf{fail} \vee v =^?_{\mathsf{E}_0} \mathsf{fail})$, for some terms u,v that do not contain fail. We consider each case separately:

- $(u \neq^?_{\mathsf{E}_0} \mathsf{fail}) \in \mathcal{D}$. In such a case, we have that either u is a variable and there exists $(X \rhd^? u) \in \mathcal{D}$, or there exist $(w_i \rhd u) \in \Phi$, thanks to Proposition 11. In both cases, $\theta \in \mathsf{Sol}^{\mathsf{valid}}(\mathbf{Tr}(\mathcal{C}))$ implies $\mathsf{valid}(u\lambda)$. In the first case, the definition of $\mathbf{Tr}(\mathcal{C})$ tells us that there exists $(u \neq^? n_{\mathsf{fail}}) \in \mathbf{Tr}(\mathcal{D})$ which means that $u\lambda\!\downarrow \neq n_{\mathsf{fail}}$ and so $u\lambda' = u\lambda\{^a/_{n_{\mathsf{fail}}}\}$. In the case where $(w_i \rhd u) \in \mathcal{D}$, we have already proved that $u\lambda' = u\lambda\{^a/_{n_{\mathsf{fail}}}\}$. Thus, in both cases, we have that $u\lambda' = u\lambda\{^a/_{n_{\mathsf{fail}}}\}$. We know that $u\lambda \neq_{\mathsf{E}} \mathsf{fail}$, and thus $u\lambda' \neq_{\mathsf{E}} \mathsf{fail}$.

- $(u =^?_{\mathsf{E}_0} v \wedge u \neq^?_{\mathsf{E}_0} \mathsf{fail} \wedge v \neq^?_{\mathsf{E}_0} \mathsf{fail}) \in \mathcal{D}$. By definition of $\mathbf{Tr}(\mathcal{C})$, we know that there exists $(u =^? v) \in \mathbf{Tr}(\mathcal{D})$. Since $\theta \in \mathsf{Sol}^{\mathsf{valid}}(\mathbf{Tr}(\mathcal{C}))$, we can deduce that $u\lambda\!\downarrow = v\lambda\!\downarrow \wedge \mathsf{valid}(u\lambda) \wedge \mathsf{valid}(v\lambda)$. Hence, we deduce that $u\lambda\{^a/_{n_{\mathsf{fail}}}\}\!\downarrow = v\lambda\{^a/_{n_{\mathsf{fail}}}\}\!\downarrow$, $\mathsf{valid}(u\lambda\{^a/_{n_{\mathsf{fail}}}\})$, and $\mathsf{valid}(v\lambda\{^a/_{n_{\mathsf{fail}}}\})$. Hence, we have that $u\lambda'\!\downarrow = v\lambda'\!\downarrow \wedge \mathsf{valid}(u\lambda') \wedge \mathsf{valid}(v\lambda')$.

  By Proposition 11, we have that $\mathsf{fail} \notin st(u, v)$ and since for all $x \in st(u, v)$, we have $x\lambda' = x\lambda\{^a/_{n_{\mathsf{fail}}}\}$, we can deduce that $n_{\mathsf{fail}} \notin st(u\lambda', v\lambda')$. Furthermore, by Proposition 9 (item 3), we can deduce that $u\lambda' =_{\mathsf{E}} v\lambda'$. At last, by Proposition 9 (item 2), we deduce that $u\lambda' \neq_{\mathsf{E}} \mathsf{fail}$ and $v\lambda' \neq_{\mathsf{E}} \mathsf{fail}$.

- $(u \neq^?_{\mathsf{E}_0} v \vee u =^?_{\mathsf{E}_0} \mathsf{fail} \vee v =^?_{\mathsf{E}_0} \mathsf{fail}) \in \mathcal{D}$. The proof is similar to the previous case.

This conclude our proof that $\theta \in \mathsf{Sol}(\mathcal{C})$ and $\Phi\lambda' = \Phi\lambda\{^a/_{n_{\mathsf{fail}}}\}$. □

Theorem 8. *Let A and B two valid intermediate processes, and* tr *be a sequence of actions. Let Σ and Σ' be two sets of constraint systems such that $(\mathsf{tr}, \Sigma) \in \mathit{trace}_s(A_s)$ and $(\mathsf{tr}, \Sigma') \in \mathit{trace}_s(B_s)$ where A_s and B_s are the symbolic processes associated to A and B. We have that:*

$$\Sigma \approx_s \Sigma', \text{ if and only if, } \mathbf{Tr}(\Sigma) \approx_s^{\mathsf{valid}} \mathbf{Tr}(\Sigma')$$

Proof. We prove the two directions separately.

First implication: $\mathbf{Tr}(\Sigma) \approx_s^{\mathsf{valid}} \mathbf{Tr}(\Sigma')$ *implies that* $\Sigma \approx_s \Sigma'$. Let $\mathcal{C} = (\mathcal{E}; \Phi; \mathcal{D})$ be a constraint system such that $\mathcal{C} \in \Sigma$. Let $\theta \in \mathsf{Sol}(\mathcal{C})$. By Proposition 11, we can assume w.l.o.g. that $\mathcal{C}$ is a valid constraint system. Let λ be the first order solution associated to θ and $\mathcal{C}$. Let θ_{valid} be the substitution that is defined as follows:

- $\mathrm{dom}(\theta_{\mathsf{valid}}) = \mathrm{dom}(\theta)$, and
- $X\theta_{\mathsf{valid}} = n_{\mathsf{fail}}$, for every X such that $(X\theta)(\Phi\lambda) =_{\mathsf{E}_0} \mathsf{fail}$, and
- $X\theta_{\mathsf{valid}} = X\theta$ otherwise.

By Lemma 7, we know that $\theta_{\mathsf{valid}} \in \mathsf{Sol}^{\mathsf{valid}}(\mathbf{Tr}(\mathcal{C}))$ and if λ_{valid} is the first order solution associated to θ_{valid} and $\mathbf{Tr}(\mathcal{C})$, then we have $x\lambda_{\mathsf{valid}} = n_{\mathsf{fail}}$ for every $x \in \mathit{vars}(\mathcal{C})$ such that $x\lambda =_{\mathsf{E}_0} \mathsf{fail}$, and $x\lambda_{\mathsf{valid}} = x\lambda$ otherwise. Moreover, we have that $\Phi\lambda = \Phi\lambda_{\mathsf{valid}}$. Since by hypothesis we know that $\mathbf{Tr}(\Sigma) \approx_s^{\mathsf{valid}} \mathbf{Tr}(\Sigma')$, we deduce that there exists $\mathcal{C}' = (\mathcal{E}'; \Phi'; \mathcal{D}') \in \Sigma'$ such that $\theta_{\mathsf{valid}} \in \mathsf{Sol}_{\mathsf{valid}}(\mathbf{Tr}(\mathcal{C}'))$ and if $\lambda'_{\mathsf{valid}}$ is the first order solution associated to θ_{valid} and $\mathbf{Tr}(\mathcal{C}')$, then $\Phi\lambda_{\mathsf{valid}} \sim^{\mathsf{valid}} \Phi'\lambda'_{\mathsf{valid}}$. By Proposition 11, we can assume w.l.o.g. that $\mathcal{C}'$ is a valid constraint system By definition of θ_{valid}, we know that for all $(X \rhd^? x) \in \mathcal{D}'$, either $X\theta_{\mathsf{valid}} = n_{\mathsf{fail}}$ or n_{fail} does not occur in $X\theta_{\mathsf{valid}}$. If $X\theta_{\mathsf{valid}} = n_{\mathsf{fail}}$, it implies that there is no disequation $x \neq^?_{\mathsf{E}_0} n_{\mathsf{fail}}$ in $\mathcal{D}'$. Thus by definition of $\mathbf{Tr}(\mathcal{C}')$ and by Proposition 11, we deduce that if $X\theta_{\mathsf{valid}} = n_{\mathsf{fail}}$, then x only occurs once in $\mathcal{C}'$, which means that $x \notin st(\Phi')$. Furthermore, by Proposition 11 and definition of $\mathbf{Tr}(\mathcal{C}')$, we know that for all $(w_i \rhd u) \in \Phi'$, $n_{\mathsf{fail}} \notin st(u)$. Thus, we can deduce that $x\lambda'_{\mathsf{valid}}\{^a/_{n_{\mathsf{fail}}}\} = x\lambda'_{\mathsf{valid}}$, for every x such that $x\lambda'_{\mathsf{valid}} \neq n_{\mathsf{fail}}$; and $\Phi'\lambda'_{\mathsf{valid}}\{^a/_{n_{\mathsf{fail}}}\} = \Phi'\lambda'_{\mathsf{valid}}$. At last, it allows us to prove that $(X\theta_{\mathsf{valid}})(\Phi'\lambda'_{\mathsf{valid}})\!\downarrow = n_{\mathsf{fail}}$ implies that $X\theta_{\mathsf{valid}} = n_{\mathsf{fail}}$.

Thus, by Lemma 8, we deduce that $\theta' \in \mathsf{Sol}(\mathcal{C}')$ where θ' is defined such that:

- $\mathrm{dom}(\theta_{\mathsf{valid}}) = \mathrm{dom}(\theta')$
- $X\theta' = \mathsf{fail}$ for every X such that $X\theta_{\mathsf{valid}} = n_{\mathsf{fail}}$, and

- $X\theta' = X\theta_{\mathsf{valid}}\{^a/_{n_{\mathsf{fail}}}\} = X\theta_{\mathsf{valid}}$.

Furthermore, if λ' is the first order solution associated to θ' and $\mathcal{C}'$, we have that

$$x\lambda' = x\lambda'_{\mathsf{valid}}\{^a/_{n_{\mathsf{fail}}}\} = x\lambda'_{\mathsf{valid}}, \text{ for every } x\lambda'_{\mathsf{valid}} \neq n_{\mathsf{fail}}, \text{ and } x\lambda' = \mathsf{fail} \text{ otherwise.}$$

Moreover, we have $\Phi'\lambda' = \Phi'\lambda'_{\mathsf{valid}}\{^a/_{n_{\mathsf{fail}}}\} = \Phi'\lambda'_{\mathsf{valid}}$.

Thus $\Phi\lambda_{\mathsf{valid}} \sim^{\mathsf{valid}} \Phi'\lambda'_{\mathsf{valid}}$ implies that $\Phi\lambda \sim^{\mathsf{valid}} \Phi'\lambda'$. We know that $n_{\mathsf{fail}} \notin st(\Phi'\lambda') \cup st(\Phi\lambda)$. But by Proposition 11, we also know that for all $(X \rhd^? x) \in \mathcal{D}$ (resp $\mathcal{D}'$) such that $x \in st(\Phi)$ (resp. $st(\Phi')$), there exists $x \neq^?_{\mathsf{E}_0} \mathsf{fail} \in \mathcal{D}$ (resp. $\mathcal{D}'$). Since θ (resp θ') is solution of $\mathcal{C}$ (resp. $\mathcal{C}'$), we can deduce that $x\lambda \neq_{\mathsf{E}_0} \mathsf{fail}$ (resp. $x\lambda' \neq_{\mathsf{E}_0} \mathsf{fail}$). Thus, by Proposition 9 (item 2), we can deduce that $\mathsf{fail} \notin st(x\lambda)$ (resp. $st(x\lambda')$) and so $\mathsf{fail} \notin st(\Phi'\lambda') \cup st(\Phi\lambda)$. By applying Lemma 6, we deduce that $\Phi\lambda \sim \Phi'\lambda'$.

Let $(X \rhd^? x) \in \mathcal{D}'$. We know that either x only occur once in $\mathcal{C}'$ or there exists $(x \neq^?_{\mathsf{E}_0} \mathsf{fail}) \in \mathcal{D}'$. But when $X\theta' = \mathsf{fail}$ and $\theta' \in \mathsf{Sol}(\mathcal{C}')$ implies $x\lambda' =_{\mathsf{E}_0} \mathsf{fail}$ and so x only occurs once in $\mathcal{C}'$. Thus, in such a case, since there is no restriction on x other than the recipe $X\theta'$, we can replace $X\theta' = \mathsf{fail}$ by any other recipe and the replacement will still be a solution of $\mathcal{C}'$. Thus, we can replace fail by $X\theta$, which means that if we denote λ'' the substitution such that $x\lambda'' = x\lambda'$ for every x such that $x\lambda' \neq_{\mathsf{E}_0} \mathsf{fail}$, and $x\lambda'' = (X\theta)(\Phi'\lambda')$ otherwise, then we have that $\theta \in \mathsf{Sol}(\mathcal{C}')$ and λ'' is the first order solution associated to θ and $\mathcal{C}'$, and we have also that $\Phi'\lambda'' = \Phi'\lambda'$. We can now conclude that $\theta \in \mathsf{Sol}(\mathcal{C}')$ and $\Phi\lambda \sim \Phi'\lambda''$.

Second implication: $\Sigma \approx_s \Sigma'$ *implies* $\mathbf{Tr}(\Sigma) \approx_s^{\mathsf{valid}} \mathbf{Tr}(\Sigma')$. Let $\mathcal{C} = (\mathcal{E}; \Phi; \mathcal{D})$ be a constraint system in Σ and $\theta_{\mathsf{valid}} \in \mathsf{Sol}^{\mathsf{valid}}(\mathbf{Tr}(\mathcal{C}))$ and λ_{valid} be the first order solution associated to θ_{valid} and $\mathbf{Tr}(\mathcal{C})$. By Proposition 11, we can assume w.l.o.g. that $\mathcal{C}$ is a valid constraint system. Let a be a fresh name in $\mathcal{N}$ (*i.e.* a doesn't appears in Σ or Σ') and let θ be the substitution such that:

- $\mathrm{dom}(\theta) = \mathrm{dom}(\theta_{\mathsf{valid}})$, and
- $X\theta = \mathsf{fail}$, for every X such that $(X\theta_{\mathsf{valid}})(\Phi\lambda_{\mathsf{valid}})\downarrow = n_{\mathsf{fail}}$, and
- $X\theta = X\theta_{\mathsf{valid}}\{^a/_{n_{\mathsf{fail}}}\}$ otherwise.

By Lemma 8, we have that $\theta \in \mathsf{Sol}(\mathcal{C})$ and if λ is the first order solution associated to θ and $\mathcal{C}$, we have that $x\lambda = x\lambda_{\mathsf{valid}}\{^a/_{n_{\mathsf{fail}}}\}$ for every x such that $x\lambda_{\mathsf{valid}}\downarrow \neq n_{\mathsf{fail}}$, and $x\lambda = \mathsf{fail}$ otherwise. Moreover, we have that $\Phi\lambda = \Phi\lambda_{\mathsf{valid}}\{^a/_{n_{\mathsf{fail}}}\}$.

Since $\Sigma \approx_s \Sigma'$, we know that there exists $\mathcal{C}' = (\mathcal{E}'; \Phi'; \mathcal{D}') \in \Sigma'$ such that $\theta \in \mathsf{Sol}(\mathcal{C}')$ and if λ' is the first order solution associated to $\mathcal{C}'$ and θ, then $\Phi\lambda \sim \Phi'\lambda'$. By Proposition 11, we can assume w.l.o.g. that $\mathcal{C}'$ is a valid constraint system. But by Proposition 11, we know that $\mathsf{fail}, n_{\mathsf{fail}} \notin st(\Phi')$. Furthermore, for all $x \in st(\Phi')$, we also know that there exists $(x \neq^?_{\mathsf{E}_0} \mathsf{fail}) \in \mathcal{D}'$ and so $x\lambda' \neq_{\mathsf{E}_0} \mathsf{fail}$. Thus, by Proposition 9 (item 1), we have that $\mathsf{fail} \notin st(x\lambda')$. Thus, we can apply Lemma 6 to deduce that $\Phi\lambda \sim^{\mathsf{valid}} \Phi'\lambda'$. But a does not appear in Φ' (a was fresh) and a, n_{fail} are public names in $\Phi\lambda$ and $\Phi'\lambda'$. Since the static equivalence is closed under one-to-one renaming, we can deduce that $\Phi(\lambda\{^{n_{\mathsf{fail}}}/_a\}) \sim^{\mathsf{valid}} \Phi'(\lambda'\{^{n_{\mathsf{fail}}}/_a\})$ and so $\Phi\lambda_{\mathsf{valid}} \sim^{\mathsf{valid}} \Phi'(\lambda'\{^{n_{\mathsf{fail}}}/_a\})$.

Applying Lemma 7, we know that $\theta'_{\mathsf{valid}} \in \mathsf{Sol}^{\mathsf{valid}}(\mathbf{Tr}(\mathcal{C}'))$ where θ'_{valid} is defined as follows:

- $\mathrm{dom}(\theta'_{\mathsf{valid}}) = \mathrm{dom}(\theta)$;
- $X\theta'_{\mathsf{valid}} = n_{\mathsf{fail}}$, for every X such that $(X\theta)(\Phi'\lambda') =_{\mathsf{E}_0} \mathsf{fail}$, and
- $X\theta'_{\mathsf{valid}} = X\theta$ otherwise.

Furthermore, if $\lambda'_{\mathsf{valid}}$ is the first order solution associated to θ'_{valid} and $\mathbf{Tr}(\mathcal{C}')$, we have that $x\lambda'_{\mathsf{valid}} = n_{\mathsf{fail}}$ for every $x \in vars(\mathcal{C}')$ such that $x\lambda' =_{\mathsf{E}_0} \mathsf{fail}$, and $x\lambda'_{\mathsf{valid}} = x\lambda'$ otherwise. Moreover, we have that $\Phi'\lambda'_{\mathsf{valid}} = \Phi'\lambda'$.

Let $(X \rhd^? x) \in \mathcal{D}$ and $(X \rhd^? y) \in \mathcal{D}'$ such that $(X\theta)(\Phi'\lambda') =_{\mathsf{E}_0} \mathsf{fail}$. Since $\Phi\lambda \sim \Phi'\lambda'$, we have that $(X\theta)(\Phi\lambda) =_{\mathsf{E}_0} \mathsf{fail}$. By definition of θ, we know that either $X\theta = \mathsf{fail}$ or $X\theta = X\theta_{\mathsf{valid}}\{^a/_{n_{\mathsf{fail}}}\}$. We show that in fact the case $X\theta = X\theta_{\mathsf{valid}}\{^a/_{n_{\mathsf{fail}}}\}$ is not possible:

Assume that $X\theta = X\theta_{\mathsf{valid}}\{^a/_{n_{\mathsf{fail}}}\}$ and $X\theta \neq \mathsf{fail}$. Since $\theta_{\mathsf{valid}} \in \mathsf{Sol}^{\mathsf{valid}}(\mathbf{Tr}(\mathcal{C}))$, we can deduce that $\mathsf{valid}(x\lambda_{\mathsf{valid}})$. Moreover our rewrite rule doesn't contain any name, which means that we also have $\mathsf{valid}(x\lambda_{\mathsf{valid}}\{^a/_{n_{\mathsf{fail}}}\})$. Since we assumed that $X\theta \neq \mathsf{fail}$, we have that $x\lambda_{\mathsf{valid}}{\downarrow} \neq n_{\mathsf{fail}}$ by definition of θ and so $x\lambda = x\lambda_{\mathsf{valid}}\{^a/_{n_{\mathsf{fail}}}\}$. Thus, we have that $\mathsf{valid}(x\lambda)$. At last, since $\mathsf{fail} \notin st(X\theta)$ and $\mathsf{fail} \notin st(\Phi\lambda_{\mathsf{valid}}\{^a/_{n_{\mathsf{fail}}}\}) = st(\Phi\lambda)$, we can conclude, thanks to Proposition 9 (item 2), that $x\lambda \neq_{\mathsf{E}_0} \mathsf{fail}$. Since by hypothesis, we assumed that $x\lambda = (X\theta)(\Phi\lambda) =_{\mathsf{E}_0} \mathsf{fail}$, there is a contradiction.

We deduce that $(X\theta)(\Phi'\lambda') =_{\mathsf{E}_0} \mathsf{fail}$ implies that $X\theta = \mathsf{fail}$ and so $(X\theta_{\mathsf{valid}})(\Phi\lambda_{\mathsf{valid}}){\downarrow} = n_{\mathsf{fail}}$. By the construction of θ and θ'_{valid}, it implies that $X\theta'_{\mathsf{valid}} = X\theta_{\mathsf{valid}}\{^a/_{n_{\mathsf{fail}}}\}$, when $(X\theta_{\mathsf{valid}})(\Phi\lambda_{\mathsf{valid}}){\downarrow} \neq n_{\mathsf{fail}}$. On the other hand, we have that $X\theta'_{\mathsf{valid}} = n_{\mathsf{fail}}$ when $(X\theta_{\mathsf{valid}})(\Phi\lambda_{\mathsf{valid}}){\downarrow} = n_{\mathsf{fail}}$.

We want to show that $\theta_{\mathsf{valid}} \in \mathsf{Sol}_v(\mathbf{Tr}(\mathcal{C}'))$ and if $\lambda''_{\mathsf{valid}}$ is the first order solution associated to $\mathbf{Tr}(\mathcal{C}')$ and θ_{valid}, we have that $x\lambda''_{\mathsf{valid}} = x\lambda'_{\mathsf{valid}}\{^{n_{\mathsf{fail}}}/_a\}$, for every x such that $x\lambda_{\mathsf{valid}}{\downarrow} \neq n_{\mathsf{fail}}$.

Let $(X \rhd^? x) \in \mathbf{Tr}(\mathcal{C}')$ such that $(x \neq^? n_{\mathsf{fail}}) \in \mathbf{Tr}(\mathcal{C}')$. We showed earlier that either $X\theta'_{\mathsf{valid}} = X\theta_{\mathsf{valid}}\{^a/_{n_{\mathsf{fail}}}\}$ or $X\theta'_{\mathsf{valid}} = n_{\mathsf{fail}}$. Since θ'_{valid} is solution of $\mathbf{Tr}(\mathcal{C}')$, we can deduce that $X\theta'_{\mathsf{valid}} = X\theta_{\mathsf{valid}}\{^a/_{n_{\mathsf{fail}}}\}$.

But a does not occur in Σ', n_{fail} does not occur in Φ' and by Proposition 11, for all $x \in st(\Phi')$, $(x \neq^? n_{\mathsf{fail}}) \in \mathbf{Tr}(\mathcal{C}')$. Thus for all $(X \rhd^? x) \in \mathbf{Tr}(\mathcal{C}')$ such that $x \neq^? n_{\mathsf{fail}} \in \mathbf{Tr}(\mathcal{C}')$, we have that $x\lambda'_{\mathsf{valid}}\{^{n_{\mathsf{fail}}}/_a\} = (X\theta'_{\mathsf{valid}})(\Phi'\lambda'_{\mathsf{valid}})\{^{n_{\mathsf{fail}}}/_a\}$ and so $x\lambda'_{\mathsf{valid}}\{^{n_{\mathsf{fail}}}/_a\} = (X\theta_{\mathsf{valid}})(\Phi'\lambda'_{\mathsf{valid}}\{^{n_{\mathsf{fail}}}/_a\}) = (X\theta_{\mathsf{valid}})(\Phi'\lambda''_{\mathsf{valid}})$.

It remains to show that θ_{valid} and $\lambda''_{\mathsf{valid}}$ satisfies the disequations. We know that n_{fail} only occurs in the disequations of the form $x \neq^? n_{\mathsf{fail}}$. Since $\theta'_{\mathsf{valid}} \in \mathsf{Sol}(\mathbf{Tr}(\mathcal{C}'))$, we know that $x\lambda'_{\mathsf{valid}}{\downarrow} \neq n_{\mathsf{fail}}$ but it doesn't implies that $x\lambda'_{\mathsf{valid}}\{^{n_{\mathsf{fail}}}/_a\}{\downarrow} \neq n_{\mathsf{fail}}$. However, we have that $\Phi\lambda_{\mathsf{valid}} \sim^{\mathsf{valid}} \Phi'\lambda'\{^{n_{\mathsf{fail}}}/_a\}$ and so $\Phi\lambda_{\mathsf{valid}} \sim^{\mathsf{valid}} \Phi'\lambda''_{\mathsf{valid}}$. Thus, $(X\theta_{\mathsf{valid}})(\Phi\lambda_{\mathsf{valid}}){\downarrow} \neq n_{\mathsf{fail}}$ implies that $(X\theta_{\mathsf{valid}})(\Phi'\lambda''_{\mathsf{valid}}){\downarrow} \neq n_{\mathsf{fail}}$.

For any constraint $(X \rhd^? x) \in \mathbf{Tr}(\mathcal{C}')$ such that $(x \neq^? n_{\mathsf{fail}}) \notin \mathbf{Tr}(\mathcal{C}')$, thanks to Proposition 11, we know that x occurs once in $\mathbf{Tr}(\mathcal{C}')$ and so there is no constraint on x other than the recipe $X\theta_{\mathsf{valid}}$.

It conclude our proof that $\theta_{\mathsf{valid}} \in \mathsf{Sol}^{\mathsf{valid}}(\mathbf{Tr}(\mathcal{C}'))$ and $\Phi\lambda_{\mathsf{valid}} \sim^{\mathsf{valid}} \Phi'\lambda''_{\mathsf{valid}}$. □

Corollary 2. *Let E_0 be the equational theory defined above. Let A and B be two valid processes without replication. The problem whether A and B are observationally (or trace) equivalent is decidable.*

Proof. Let A and B be two valid processes without replication. They can easily be transformed into valid intermediate processes $\tilde{A}$ and $\tilde{B}$.

Thanks to Proposition 3, we have that $A \approx_t B$ if, and only if, $\tilde{A} \approx_t \tilde{B}$, and relying on Proposition 6, we have that $\tilde{A} \approx_t \tilde{B}$ if, and only if, $\tilde{A}_s$ and $\tilde{B}_s$ are in symbolic trace equivalence, *i.e.* if for every sequence tr of symbolic actions, we have that:

$$\{\mathcal{C} \mid (\mathsf{tr}, \mathcal{C}) \in \mathsf{trace}_s(\tilde{A}_s)\} \approx_s \{\mathcal{C} \mid (\mathsf{tr}, \mathcal{C}) \in \mathsf{trace}_s(\tilde{B}_s)\}.$$

We have show in Theorem 8 that deciding this symbolic equivalence amounts to decide whether

$$\mathbf{Tr}(\{\mathcal{C} \mid (\mathsf{tr}, \mathcal{C}) \in \mathsf{trace}_s(\tilde{A}_s)\}) \approx_s^{\mathsf{valid}} \mathbf{Tr}(\{\mathcal{C} \mid (\mathsf{tr}, \mathcal{C}) \in \mathsf{trace}_s(\tilde{B}_s)\})$$

and this can be done by using the algorithm provided in [20]. □