Journal of Universal Computer Science, vol. 10, no. 9 (2004), 1156-1198
submitted: 16/9/03, accepted: 2/4/04, appeared: 28/9/04 © J.UCS

Symbolic Approach to the Analysis of Security Protocols

Stéphane Lafrance
(École Polytechnique de Montréal, Canada
stephane.lafrance@polymtl.ca)

Abstract: The specification and validation of security protocols often requires viewing function calls – like encryption/decryption and the generation of fake messages – explicitly as actions within the process semantics. Following this approach, this paper introduces a symbolic framework based on value-passing processes able to handle symbolic values like fresh nonces, fresh keys, fake addresses and fake messages. The main idea in our approach is to assign to each value-passing process a formula describing the symbolic values conveyed by its semantics. In such symbolic processes, called *constrained processes*, the formulas are drawn from a logic based on a message algebra equipped with encryption, signature and hashing primitives. The symbolic operational semantics of a constrained process is then established through semantic rules updating formulas by adding restrictions over the symbolic values, as required for the process to evolve. We then prove that the logic required from the semantic rules is decidable. We also define a bisimulation equivalence between constrained processes; this amounts to a generalisation of the standard bisimulation equivalence between (non-symbolic) value-passing processes. Finally, we provide a complete symbolic bisimulation method for constructing the bisimulation between constrained processes.
Key Words: symbolic, bisimulation, protocols, non-interference, process algebra, equivalence-checking, formal methods.
Category: C.2.2, C.2.4

1 Introduction

The sudden expansion of electronic commerce has introduced an urgent need to establish strong security policies for the design of security protocols. The formal validation of security protocols has since become one of the primary tasks in computer science. In recent years, equivalence-checking has proved to be useful for the verification of security protocols [1, 4, 6, 19]. The main idea behind this approach of formal verification is to verify a security property by testing whether a process (specifying a protocol) is bisimilar to its intended behaviour. The success of these methods relies on two facts: 1) process algebras are suitable for the specification of such protocols, including cryptographic protocols; 2) bisimulation offers an expressive semantics to process calculi. Many other methods inspired by a wide range of approaches have been proposed in the literature to analyse security protocols, but very few offer the possibility to explicitly analyse function calls used, for example, in encrypting, decrypting, signing and hashing. In cryptographic based process calculi like Abadi & Gordon's *spi-calculus* [2] and Focardi & Martinelli's *CryptoSPA* [11], encryption and decryption manipulations are done in a parallel inference system, and therefore they are not directly observable from the process semantics. For instance, a principal sending a message m encrypted with a key k is modeled as an output action "$\overline{c}(\{m\}_k)$" (where $\{m\}_k$ stands for the message m encrypted by k) whenever $\{m\}_k$ can be inferred from the principal's current knowledge.

However, information flow properties (e.g. non-interference [10] and admissible interference [18]) usually require such manipulations to be observable. For that purpose, we work within the framework of an extension of value-passing CCS [16], called *Security Protocols Process Algebra* (SPPA) [14], in which function calls made by principals are explicitly modeled as actions. For instance, a principal sending a message m encrypted with a key k is modeled as the action "enc_{id}" (where id is an identifier for the principal encrypting the message) followed by the output action "$\overline{c}(\{m\}_k)$". Moreover, the specification of intruders in SPPA allows us to analyse the effects on the information flow of a protocol of an intruder generating fake messages and fake addresses. In addition, compared with a process calculus using an inference system for encryption manipulations, SPPA is more suited for analysing restricted attacks based on the repetition of the same attempt. For instance, distributed denial of service attacks have been specified in SPPA [14]. In order to deal with the notion of fake message, around which most attacks are built, we need to extend SPPA in order to specify functions generating random values. But the introduction of such generating function calls as actions requires interpreting their output as symbolic values. Thus, we need to consider symbolic value-passing processes along with a symbolic operational semantics able to handle symbolic variables without a specific value but satisfying certain constraints.

This paper introduces a symbolic framework for the specification of security protocols which is based on the novel concept of *constrained process*. A constrained process is a pair composed of a value-passing process (SPPA process with, possibly, free variables standing for symbolic values) and a formula expressing a statement about symbolic values. The formula pertains to a message logic whose terms are taken from a message algebra relying on atomic sets of numbers and identifiers (addresses), and cryptographic primitives (encryption, signing and hashing operators). Therefore, the purpose of a formula within a constrained process is to bind the free variables occurring in the course of process execution. For instance, to a process generating a fresh key for a protocol run and allocating this key to some free variable x, we assign the formula which states that x stands for a key. The operational semantics of constrained processes is thus achieved from the process behaviour, subject to the restrictions imposed by its formula. Hence, a process whose definition requires the execution of an action and evolution into another process will only occur if the whole transition satisfies the formula enforced at this point. Roughly speaking, the formula within a constrained process stands for the set of messages that can be assigned to its free variables; this set of possible values evolves, along with the process, by either adding new free variables or restricting (or binding) the ones already present. In one of the main results of this paper, we prove the decidability of every formula derivable from the process algebra's operational semantics. Moreover, we feel that our symbolic framework can be applied to any other process algebra, from value-passing CCS to more expressive process algebras like Milner's π-*calculus* [17] and Abadi & Gordon's *spi-calculus* [2].

The use of value-passing processes over infinite messages-domain leads to non finite-branching transition graphs on which trace equivalence and bisimulation equivalence fail to be decidable. An attractive solution to this challenge was proposed by Hennessy-Lin [12] who defined a notion of symbolic bisimulation. It is primary based on a symbolic semantics which may express value-passing CCS processes in terms of finite symbolic transition graphs instead of possibly infinite ones. The main idea behind Hennessy-Lin's approach is to assign to every action (transition) a formula describing the symbolic values (free variables) used in the action. Within this framework, they introduce two generalisations of Milner's strong bisimulation equivalence for value-passing processes called *early* and *late* bisimulation. Although our paper aims at a similar goal, we introduce a symbolic semantics in which the description of symbolic values is done within the processes (states) instead of within the transitions. In fact, our symbolic transition graphs could be directly obtained from their symbolic transition graphs (by considering every path). Moreover, our approach, compared to Hennessy-Lin's, takes advantage of an expressive message logic capable of stating cryptographic relations. For instance, we can bind free variables x_1, x_2, x_3 through the formula $(x_1 == \{x_2\}_{x_3}) \wedge \mathcal{K}(x_3)$ which states that x_1 stands for x_2 encrypted with the key x_3. In addition, we feel that the concept of constrained process is more suited for security protocol analysis than Hennessy-Lin's symbolic transition graph: a constrained process allows us to get a quick view of the symbolic values at a given state of the protocol, rather than retrieving successively every path leading to this state.

This paper is organised as follows. In section 2, we introduce a logic for cryptographic messages. In section 3, we present the SPPA process algebra and we describe a symbolic semantics for constrained processes. In section 4, we introduce a bisimulation equivalence relation for constrained processes, for which we give, in section 5, a sound and complete proof method called symbolic bisimulation. In section 6, we offer a brief overview on the application of our symbolic framework to security protocols analysis. We conclude this paper with a short talk on related work and on our future work.

2 Message Specification

2.1 Message Algebra

We consider the following message algebra which relies on disjoint syntactic categories of numbers, principal identifiers and variables respectively ranging over sets $\mathcal{N}$, $\mathcal{I}$ and $\mathcal{V}$. The set $\mathcal{T}$ of *terms* is constructed as follows:

$$\begin{aligned} t ::= \;& n \;\; (number) \;\; \mid id \;\; (identifier) \mid \;\; x \;\; (variable) \mid (t, t) \; (pair) \\ & \mid \{t\}_t \; (encryption) \mid [t]_t \; (signature) \mid h(t) \; (hashing). \end{aligned}$$

It is important to note that we only consider finite terms. For any term t, we denote $\mathrm{fv}(t)$ the set of variables occurring in t and we say that t is a *message* whenever it contains no variable. The set of all messages is denoted by $\mathcal{M}$. Furthermore, given a valuation $\varrho : \mathcal{V} \to \mathcal{M}$ and a term t such that $\mathrm{fv}(t) = \{x_1, \ldots, x_n\}$, $\varrho(t)$ stands for the message $t[\varrho(x_1)/x_1] \ldots [\varrho(x_n)/x_n]$ i.e., the message obtained from t by substituting each variable x_i with its valuation $\varrho(x_i)$ $(i = 1, \ldots, n)$. Note that if a variable is substituted more than once in the expression $t[\varrho(x_1/x_1] \ldots [\varrho(x_n)/x_n]$, then the lefter-most substitution always prevails.

For the sake of clarity, we will discriminate a subset $\mathcal{K} \subseteq \mathcal{M}$ of messages that may be used as encryption keys. Note that the definition of the set $\mathcal{K}$ usually depends on the cryptosystem used by the protocol. For instance, in the case of a symmetric block-cypher algorithm, we have $\mathcal{K} = \{v \in \mathcal{N} \mid \text{length of } v = N\}$ for some $N \in \mathbb{N}$; or, more generally, we may have $\mathcal{K} = \mathcal{N} \cup \bigcup_{m \geq 1} \{h^m(n) \mid n \in \mathcal{N}\}$ where we write $h^m(n)$ instead of $h(\ldots h(n) \ldots)$ (m times). However, for simplicity purposes, this paper simply uses the set $\mathcal{K} = \mathcal{N}$. Moreover, in order to deal with public-key encryption, we use an idempotent operator $[-]^{-1} : \mathcal{K} \to \mathcal{K}$ such that a^{-1} denotes the private decryption key corresponding to the public encryption key a, or vice versa. For symmetric encryption, let $a^{-1} = a$. Moreover, one assumes perfect encryption and hashing.

2.2 A Logic for Messages

In the following, we consider the logic based on the terms of our message algebra and the following *predicates*:

$$\begin{aligned} \mathcal{P} ::= \;& t == t \; (term\ equation) \quad \mid \mathcal{M}(t) \; (message\ predicate) \\ & \mid \; \mathcal{N}(t) \;\; (number\ predicate) \mid \; \mathcal{I}(t) \;\; (identifier\ predicate) \\ & \mid \; \mathcal{K}(t) \;\; (key\ predicate). \end{aligned}$$

The *formulas* of our logic are then obtained as follows:

$$\phi ::= \mathbf{0} \;\mid\; \mathbf{1} \;\mid\; \mathcal{P} \;\mid\; \phi \wedge \phi \;\mid\; \exists_x \, \phi \, .$$

The set of ϕ's free variables is denoted by $\mathrm{fv}(\phi)$ and ϕ is said to be closed whenever $\mathrm{fv}(\phi) = \emptyset$. The satisfaction of a closed formula ϕ, denoted by $\models \phi$, is defined recursively as follows:

- $\models \mathbf{1}$ and $\not\models \mathbf{0}$
- $\models\ a == b$ iff messages a and b are syntactically identical i.e.,
  - $\models\ n == n$ for every $n \in \mathcal{N}$,
  - $\models\ id == id$ for every $id \in \mathcal{I}$,
  - $\models\ (a_1, a_2) == (b_1, b_2)$ iff $\models\ a_1 == b_1 \ \wedge\ a_2 == b_2$,
  - $\models\ \{a_2\}_{a_1} == \{b_2\}_{b_1}$ iff $\models\ a_1 == b_1 \ \wedge\ a_2 == b_2$,
  - $\models\ [a_2]_{a_1} == [b_2]_{b_1}$ iff $\models\ a_1 == b_1 \ \wedge\ a_2 == b_2$, and
  - $\models\ h(a) == h(b)$ iff $\models\ a == b$;
- $\models\ \mathcal{M}(a)$ for every message $a \in \mathcal{M}$;
- $\models\ \mathcal{N}(a)$ iff $a \in \mathcal{N}$;
- $\models\ \mathcal{I}(a)$ iff $a \in \mathcal{I}$;
- $\models\ \mathcal{K}(a)$ iff $a \in \mathcal{K}$;
- $\models\ \phi \wedge \phi'$ iff $\models\ \phi$ and $\models\ \phi'$;
- $\models\ \exists_x\, \phi$ iff $\models\ \phi[a/x]$ for some $a \in \mathcal{M}$.

(Notation $\phi[a/x]$ stands for the substitution of every free occurrence of variable x in ϕ, by message a.) We assume that each predicate is decidable i.e., the satisfiability problems $\models \mathcal{N}(a)$, $\models \mathcal{I}(a)$ and $\models \mathcal{K}(a)$ are decidable for any $a \in \mathcal{M}$, and they are never satisfied whenever a is a non-atomic message (recall that we assumed earlier that $\mathcal{K} = \mathcal{N}$). For instance, $\not\models \mathcal{I}(h(a))$ and $\not\models \mathcal{N}(\{a\}_b)$ for any $a, b \in \mathcal{M}$. Moreover, we recall that $\mathcal{I}$ and $\mathcal{N}$ are disjoint sets.

Given a valuation $\varrho : \mathcal{V} \to \mathcal{M}$, the satisfaction of a formula ϕ by ϱ, denoted by $\varrho \models \phi$, is defined as follows:

$$\varrho \models \phi \quad \text{iff} \quad \models \varrho(\phi).$$

Example 1. Formula

$$\phi ::= \ \exists_{x_1}\exists_{x_2}\ (x == (x_1, x_2)\ \wedge\ \mathcal{K}(x_1)\ \wedge\ \mathcal{M}(x_2))$$

with $\mathrm{fv}(\phi) = \{x\}$ states that variable x must be a couple composed of a key and a message. Hence, if $\varrho_1(x) = (k, a)$, for some $k \in \mathcal{K}$ and $a \in \mathcal{M}$, then we see that $\varrho_1 \models \phi$. However, if $\varrho_2(x) = (a, a)$, then $\varrho_2 \not\models \phi$ unless $a \in \mathcal{K}$.

Two formulas ϕ and ϕ' are said to be *equivalent* – which is denoted by $\phi \Leftrightarrow \phi'$ – whenever

$$\varrho \models \phi \quad \text{iff} \quad \varrho \models \phi'$$

for every valuation ϱ. In particular, a formula ϕ is equivalent to $\mathbf{0}$ – which is denoted by $\phi \Leftrightarrow \mathbf{0}$ – whenever $\varrho \not\models \phi$ for every valuation ϱ. We also consider the equivalence relation $\phi \overset{\mathcal{V}}{\Leftrightarrow} \phi'$ defined as follows:

$$\phi \overset{\mathcal{V}}{\Leftrightarrow} \phi' \quad \text{iff} \quad (\phi \Leftrightarrow \phi' \text{ and } \mathrm{fv}(\phi) = \mathrm{fv}(\phi')).$$

The equivalence class of some formula ϕ under $\overset{\mathcal{V}}{\Leftrightarrow}$ is defined by $[\![\phi]\!] = \{\phi' \mid \phi \overset{\mathcal{V}}{\Leftrightarrow} \phi'\}$.

2.3 Decidability

We can prove that every (closed) formula from our logic is decidable. This result follows from the fact that our logic for messages is restricted to conjunction and existential operators. Hence, the decidability of a formula boils down to the satisfaction of a number of predicates and equations, which we assume to be decidable.

Theorem 1. *Every formula is decidable.*

Proof of Theorem 1 is given in Appendix A.

2.4 Functions

We consider a finite set $\mathcal{F}$ of *functions* mapping messages to new messages constructed from the grammar rules above. Each function $f(x_1, \ldots, x_n)$ has a *characterisation formula* from our logic, denoted by $\phi_{f(x_1,\ldots,x_n)}$ or simply ϕ_f, which is satisfied only by messages within its domain, and such that $\mathrm{fv}(\phi_f) = \{x_1, \ldots, x_n\}$. Therefore, $\models \phi_f[a_1/x_1]\ldots[a_n/x_n]$ if and only if $f(a_1, \ldots, a_n)$ is defined. We often write $\phi_f(a_1, \ldots, a_n)$ instead of $\phi_f[a_1/x_1]\ldots[a_n/x_n]$, or simply $\phi_f(a)$ where $a = (a_1, \ldots, a_n)$. In addition, we consider the notion of *generating functions*, extremely useful for the specification of security protocols requiring fresh nonces, fresh keys and random numbers, and for the specification of intruders generating fake addresses and fake messages. Generating functions are functions which may generate symbolic values without any input. Each generating function $new \in \mathcal{F}$ (often denoted by $new(-)$) is assigned to a formula ϕ_{new}, also called *characterisation formula*, which is satisfied only by messages within its range. We usually consider the following functions:

- $\mathrm{pair}(x_1, x_2) = (x_1, x_2)$ with $\phi_{\mathrm{pair}(x_1,x_2)} ::= \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)$;
- $\mathrm{enc}(x_1, x_2) = \{x_2\}_{x_1}$ with $\phi_{\mathrm{enc}(x_1,x_2)} ::= \mathcal{K}(x_1) \wedge \mathcal{M}(x_2)$;
- $\mathrm{hash}(x) = h(x)$ with $\phi_{\mathrm{hash}(x)} ::= \mathcal{M}(x)$;
- $\mathrm{sign}(x_1, x_2) = [x_2]_{x_1}$ with $\phi_{\mathrm{sign}(x_1,x_2)} ::= \mathcal{K}(x_1) \wedge \mathcal{M}(x_2)$;
- $\mathrm{newMessage}(-)$ with $\phi_{\mathrm{newMessage}} ::= \mathcal{M}(x)$;
- $\mathrm{newNumber}(-)$ with $\phi_{\mathrm{newNumber}} ::= \mathcal{N}(x)$;

- newId(−) with $\phi_{\text{newId}} ::= \mathcal{I}(x)$;
- newKey(−) with $\phi_{\text{newKey}} ::= \mathcal{K}(x)$.

3 Security Protocol Process Algebra

Our first step toward validation of security protocols is to find a language which may express both the protocols and the security policies we want to enforce. Process algebra has been used for some years to specify protocols as a cluster of concurrent processes, representing principals participating in the protocol, which are able to communicate in order to exchange data. Process algebras CSP [13] and CCS [16] have been extensively used with this objective [15, 20]. In this section, we introduce a generic symbolic framework which we feel could be applied to numerous process algebras. Given a process algebra, we proceed by extending its syntax in order to view generating function calls "let $x = new(-)$ in…" as prefixes. Messages generated in this way are explicitly typed with the characterisation formula ϕ_{new}. For simplicity, our symbolic framework follows the *Security Protocols Process Algebra* (SPPA) [14], an extension of value-passing CCS in which local function calls are viewed as visible actions. Up to these extensions tailored just to fit to the ideas presented here, SPPA is very similar to SPA presented by Focardi & Gorrieri [7]. SPPA's syntax follows Abadi & Gordon's Spi-Calculus [2], but without scope extrusion and replication, and the input and output prefixes do not carry channels. Also, the purpose here is not to introduce a new process algebra but just to define a generic process algebraic symbolic framework as well-suited as possible to analyse cryptographic protocols.

3.1 Syntax of SPPA

First, we consider a finite set C of *public channels*. Public channels are used to specify message exchanges between principals (commonly, there is one channel for every step of a protocol run). We assume that public channels have no specific domains: any message can be sent or received over them.

The *agents* of SPPA are constructed from the following grammar:

$$
\begin{array}{rcll|cl}
S ::= & & \mathbf{0} & (\textit{nil}) & \texttt{let } x = f(t) \texttt{ in } S & (\textit{function call}) \\
 & | & \overline{c}(t).S & (\textit{output}) & \texttt{let } (x,y) = t \texttt{ in } S & (\textit{pair splitting}) \\
 & | & c(x).S & (\textit{input}) & \texttt{case } t \texttt{ of } \{x\}_{t'} \texttt{ in } S & (\textit{decryption}) \\
 & | & [t = t']\ S & (\textit{match}) & \texttt{case } t \texttt{ of } [t'']_{t'} \texttt{ in } S & (\textit{signature verification}) \\
 & | & S + S & (\textit{sum}) & S|S & (\textit{parallel composition}) \\
 & | & S \backslash L & (\textit{restriction}) & S/\mathcal{O} & (\textit{observation})
\end{array}
$$

where L is a set and $\mathcal{O}$ is a partial mapping (both to be clarified in Section 3.2). Whenever f is a generating function, we usually write let $x = f(-)$ in S.

In order to prevent name clashes for variables (e.g. to prevent the sum or parallel composition of agents having free variables in common), we assume that a variable is never used twice to define agents (renaming variables when necessary). Given an agent S, we define its set of *free variables*, denoted by $\mathrm{fv}(S)$, as the set of variables x appearing in S which are not in the scope of an input prefix $c(x)$, a pair splitting $\texttt{let}\ (x, y) = t\ \texttt{in}$, a function call $\texttt{let}\ x = f(t)\ \texttt{in}$, or a decryption $\texttt{case}\ \{t'\}_t\ \texttt{of}\ \{x\}_t\ \texttt{in}$; otherwise the variable x is said to be *bound*. Given a free variable $x \in \mathrm{fv}(S)$ and a term t, we consider the substitution operator $S[t/x]$ where every free occurrence of x in S is set to t. A *closed agent* is an agent S such that $\mathrm{fv}(S) = \emptyset$.

A SPPA *principal* is a couple (S, id) where S is an agent and $id \in \mathcal{I}$. The purpose of this notation is to relate an SPPA agent S and its sub-agents, to their unique owner (principal) via its identifier id. When no confusion is possible, we often use A as a reference to the principal (S_A, id_A) where S_A is the *initial agent* of A i.e., the agent specifying the entire behaviour of the principal A within the protocol. Moreover, we commonly make use of the identifier id_A as a message containing its address, while we simply use A to refer to the principal's entity (i.e. the party involved with the protocol). For simplicity, given $A_1 ::= (S_1, id)$ and $A_2 ::= (S_2, id)$ (they must have the same identifier) we often write $[t = t']A_1$ instead of $([t = t']S_1, id)$, $A_1|A_2$ instead of $(S_1|S_2, id)$, $A_1 + A_2$ instead of $(S_1 + S_2, id)$, and so on.

In order to specify a security protocol in SPPA, we use the classic approach [9, 20] of specifying the principals as concurrent agents. Given a principal A, SPPA *processes* are constructed as follows:

$$\begin{aligned} P ::= \quad & A \quad (\textit{principal}) \quad \mid A \parallel P \quad (\textit{protocol}) \\ & \mid P\backslash L \quad (\textit{restriction}) \mid \ P/\mathcal{O} \quad (\textit{observation}). \end{aligned}$$

where $\parallel$ is an associative and commutative operator forcing communication over public channels.

A *constrained process* is an expression of the form $\langle P,\ \phi\rangle$ where P is a process and ϕ is a formula designed to constrain the free variables occurring in P. Commonly, notation $\langle P,\ \phi\rangle$ stands for the pair $(P,\ [\![\phi]\!])$, where $[\![\phi]\!]$ is the equivalence class of ϕ under the relation $\overset{\mathcal{V}}{\Leftrightarrow}$ (see Section 2.2). Thus, if $\phi \overset{\mathcal{V}}{\Leftrightarrow} \phi'$ (i.e. formulas ϕ and ϕ' are equivalent and have the same free variables), then the constrained processes $\langle P,\ \phi\rangle$ and $\langle P,\ \phi'\rangle$ are considered to be the same.

Example 2. Consider the following one-step protocol

$$\text{Message 1:} \quad A \overset{\{n_A\}_{k_B}}{\longrightarrow} B$$

in which principal A generates a fresh nonce n_A and sends to the principal B this nonce encrypted with B's public key k_B. Principals A and B are specified, respectively, as the SPPA principals $A ::= (S_A, id_A)$ and $B ::= (S_B, id_B)$, where id_A is A's identifier, id_B is B's identifier, and the initial agents S_A and S_B are defined as follows:

$$S_A ::= \texttt{let}\ x_1 = \text{newNumber}(-)\ \texttt{in}\ \texttt{let}\ x_2 = \text{enc}(k_B, x_1)\ \texttt{in}\ \overline{c}(x_2).\mathbf{0}$$
$$S_B ::= c(y_1).\ \texttt{case}\ y_1\ \texttt{of}\ \{y_2\}_{k_B}\ \texttt{in}\ \mathbf{0}\ .$$

The protocol is then specified as the SPPA process $P ::= A \parallel B$, in which principals A and B can communicate over the public channel c. Since the process P has no free variable, the protocol is then specified as the constrained process $\langle P,\ \mathbf{1}\rangle$.

3.2 Symbolic Semantics

The value-passing operational semantics of an SPPA process is defined in Appendix B. Note that this value-passing semantics is only defined for closed processes. Also note that, because of the value-passing semantics, the obtained transition graph could be infinite. In this section, we establish a symbolic operational semantics for constrained processes, which correspond to finite labeled transition graphs.

Given a term t, the *actions* of SPPA are defined as follows:

$$\begin{array}{llll} \alpha ::= \overline{c_{id}}(t) & (\mathit{output}) & \mid c_{id}(x) & (\mathit{input}) \\ \quad\mid f_{id} & (\mathit{function\ call}) & \mid \text{split}_{id} & (\mathit{splitting}) \\ \quad\mid \text{dec}_{id} & (\mathit{decryption}) & \mid \text{signv}_{id} & (\mathit{signature\ verification}) \\ \quad\mid \delta(t) & (\mathit{marker\ action}) & \mid \tau & (\mathit{silent\ action}) \end{array}$$

For instance, function call action enc_{id_A} stands for principal A encrypting some message a with some key k; output action $\overline{c_{id_A}}(\{a\}_k)$ stands for principal A sending message $\{a\}_k$ over the public channel c; and decryption action dec_{id_A} stands for principal A successfully decrypting some message $\{a\}_k$. The silent action τ is used to express non-observable behaviours. We often use C to denote both the set of public channels and the set of output and input actions.

In value-passing process algebra, communication is commonly expressed by replacing the matching output action and input action by the silent action τ. However, this interpretation of communication causes a drastic loss of information on the content of the exchanged values and the parties involved. Using a marker action $\delta(t)$ instead of τ in those situations helps to parry this problem. Marker actions are therefore introduced in an attempt to establish an annotation to the semantics of an SPPA process; they do not occur in the syntax of processes and their specific semantics restricts their occurrence in order to tag communications between principals. A marker action has three parameters: a principal identifier, a channel and a term (message). Roughly speaking, the occurrence of an *output marker* $\overline{\delta^c_{id_A}}(a)$ stands for "*the principal A has sent message a over* *the channel c*", and the occurrence of an *input marker* $\delta^c_{id_A}(a)$ stands for "*the principal A has received message a over the channel c*".

We write Act to denote the set of all actions and we consider the set Act_A of actions that may be launched by the principal A, defined by:

$$\begin{aligned} Act_A = \{&\overline{c_{id_A}}(t),\ c_{id_A}(t),\ \overline{\delta^c_{id_A}}(t),\ \delta^c_{id_A}(t) \in Act \mid c \in C \text{ and } t \in \mathcal{T}\} \\ &\cup \{f_{id_A} \mid f \in \mathcal{F}\} \ \cup \ \{\mathrm{split}_{id_A},\ \mathrm{dec}_{id_A},\ \mathrm{signv}_{id_A}\} \end{aligned}$$

An *observation criterion* is a partial mapping $\mathcal{O} : Act^* \mapsto Act$ which intends to express equivalence between process behaviours. Two sequences of actions γ_1 and γ_2 are said to carry out the same observation α whenever $\gamma_1, \gamma_2 \in \mathcal{O}^{-1}(\alpha)$. Given a subset $L \subseteq Act \setminus \{\tau\}$, we consider the observation criterion $\mathcal{O}_L$ defined as follows:

$$\mathcal{O}_L^{-1}(\alpha) = \begin{cases} (Act \setminus L)^* \ \alpha \ (Act \setminus L)^* & \text{if } \alpha \in L \\ (Act \setminus L)^* & \text{if } \alpha = \tau. \end{cases}$$

Only behaviours from the set L are observable through this observation criterion. In particular, we have a natural observation criterion $\mathcal{O}_{Act_A \cup C}$, often denoted by $\mathcal{O}_A$, describing the actions observable by a principal A.

The symbolic operational semantics for constrained processes is given in Fig. 1 and Fig. 2. It is inspired by Hennessy-Lin's symbolic operational semantics [12] where boolean values guarding actions are replaced by formulas ϕ restricting free variables within the processes. Note that any transition $\langle P,\ \phi\rangle \xrightarrow{\alpha} \langle P',\ \phi'\rangle$ is dismissed whenever either ϕ or ϕ' is equivalent to $\mathbf{0}$.

Rules `Output` and `Input` allow principals to, respectively, send and receive messages over public channels. Rules `Function` and `Generator` allow the execution of local function calls made by principals. Rule `Split` allows to extract pairs. Rules `Decryption` and `Signature-Verif` allow to, respectively, recover encrypted messages and verify signed messages. Rule `Match` allows the verification of equality between two messages. Rules `Sum` and `Parallel` allow the specification of non-deterministic sum and parallel product of agents (with matching identifier). Rules `Protocol` and `Synchronisation` allow the specification of protocols, where the operator $\|$ is similar to a parallel product in which communication between principals is achieved (and forced) through public channels. Rules `Sum`, `Parallel`, `Protocol` and `Synchronisation` are assumed to be both associative and commutative (i.e. $P + (Q + R)$ behaves as $(P + Q) + R$, $Q + P$ behaves as $P + Q$, and so on). Moreover, recall that constrained processes $\langle P,\ \phi\rangle$ and $\langle Q,\ \psi\rangle$ must be defined with different variables, thus $\mathrm{fv}(\phi) \cap \mathrm{fv}(\psi) = \emptyset$. Rule `Restriction` interprets $P \setminus L$ as process P with the actions in L forbidden. In the `Restriction` rule, we assume that formula ϕ^L_α (which forbids instantiations of α to be in L) is such that $\mathrm{fv}(\phi^L_\alpha) = \mathrm{fv}(\alpha)$. Moreover, we need to restrict rule `Restriction` to the sets L such that formula ϕ^L_α is definable within our logic.

`Output`	$\dfrac{-}{\langle \overline{c}(t).A,\ \phi\rangle \xrightarrow{\overline{c_{id_A}}(t)} \langle A,\ \phi\rangle}$
`Input`	$\dfrac{-}{\langle c(x).A,\ \phi\rangle \xrightarrow{c_{id_A}(x)} \langle A,\ (\exists_x \phi)\ \wedge\ \mathcal{M}(x)\rangle}$
`Function`	$\dfrac{f\ \in\ \mathcal{F}}{\langle \texttt{let}\ x{=}f(t)\ \texttt{in}\ A,\ \phi\rangle \xrightarrow{f_{id_A}} \langle A,\ (\exists_x \phi)\ \wedge\ \phi_f(t)\ \wedge\ x{==}f(t)\rangle}$
`Generator`	$\dfrac{new\ \in\ \mathcal{F}}{\langle \texttt{let}\ x{=}f(-)\ \texttt{in}\ A,\ \phi\rangle \xrightarrow{new_{id_A}} \langle A,\ (\exists_x \phi)\ \wedge\ \phi_{new}(x)\rangle}$
`Split`	$\dfrac{-}{\langle \texttt{let}\ (x,y){=}(t,t')\ \texttt{in}\ A,\ \phi\rangle \xrightarrow{\mathrm{split}_{id_A}} \langle A,\ (\exists_x \exists_y \phi)\ \wedge\ x{==}t\ \wedge\ y{==}t'\rangle}$
`Decryption`	$\dfrac{-}{\langle \texttt{case}\ t\ \texttt{of}\ \{x\}_{t'}\ \texttt{in}\ A,\ \phi\rangle \xrightarrow{\mathrm{dec}_{id_A}} \langle A,\ (\exists_x \phi)\ \wedge\ \mathcal{K}(t')\ \wedge\ t{==}\{x\}_{t'}\rangle}$
`Signature-Verif`	$\dfrac{-}{\langle \texttt{case}\ t\ \texttt{of}\ [t'']_{t'}\ \texttt{in}\ A,\ \phi\rangle \xrightarrow{\mathrm{signv}_{id_A}} \langle A,\ \phi\ \wedge\ \mathcal{K}(t')\ \wedge\ t{==}[t'']_{t'}\rangle}$

Figure 1: *Semantics of constrained processes.*

Finally, rule `Observation` interprets the observation of a process through an observation criterion $\mathcal{O}$, where the *computation* $\langle P,\ \phi\rangle \xrightarrow{\gamma} \langle P',\ \phi'\rangle$, for a sequence of actions $\gamma = \alpha_0\alpha_1 \dots \alpha_n \in Act^*$, stands for the finite string of transitions satisfying

$$\langle P,\ \phi\rangle \xrightarrow{\alpha_0} \langle P_1,\ \phi_1\rangle \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} \langle P',\ \phi'\rangle.$$

Thus, $P/\mathcal{O}_L$ (where L is a set of actions) means P with the actions outside L ignored (set to τ).

A constrained process $\langle P',\ \phi'\rangle$ is a *derivative* of $\langle P,\ \phi\rangle$ if there is a computation $\langle P,\ \phi\rangle \xrightarrow{\gamma} \langle P',\ \phi'\rangle$ for some $\gamma \in Act^*$. Hence, the set of $\langle P,\ \phi\rangle$'s derivatives is defined by

$$\mathcal{D}(\langle P,\ \phi\rangle) = \{\langle P',\ \phi'\rangle \mid \exists_{\gamma \in Act^*}\ \langle P,\ \phi\rangle \xrightarrow{\gamma} \langle P',\ \phi'\rangle\}.$$

The following theorem states that the transition graph associated to any constrained process is always finite.

Theorem 2. *For every constrained process $\langle P,\ \phi\rangle$, the set $\mathcal{D}(\langle P,\ \phi\rangle)$ is finite.*

Proof of Theorem 2 is given in Appendix C.

$$\texttt{Match} \qquad \frac{\langle A,\ \phi\rangle \xrightarrow{\alpha} \langle A',\ \phi'\rangle}{\langle [t{=}t']A,\ \phi\rangle \xrightarrow{\alpha} \langle A',\ \psi\rangle}$$

$$\text{with } \psi ::= \begin{cases} (\exists_x(\phi\ \wedge\ t{=}{=}t'))\ \wedge\ \psi' \text{ if } \alpha{=}\ c_{id}(x),\ f_{id},\ \mathrm{split}_{id} \text{ or } \mathrm{dec}_{id} \\ \qquad\qquad \text{and } \phi'{::=}(\exists_x\phi)\ \wedge\ \psi' \\ \phi'\ \wedge\ t{=}{=}t' \qquad \text{otherwise.} \end{cases}$$

$$\texttt{Sum} \qquad \frac{\langle P,\ \phi\rangle \xrightarrow{\alpha} \langle P',\ \phi'\rangle \quad \text{and} \quad \mathrm{fv}(\phi){\cap}\mathrm{fv}(\psi){=}\emptyset}{\langle P{+}Q,\ \phi\ \wedge\ \psi\rangle \xrightarrow{\alpha} \langle P',\ \phi'\ \wedge\ \psi\rangle}$$

$$\texttt{Parallel} \qquad \frac{\langle P,\ \phi\rangle \xrightarrow{\alpha} \langle P',\ \phi'\rangle \quad \text{and} \quad \mathrm{fv}(\phi){\cap}\mathrm{fv}(\psi){=}\emptyset}{\langle P|Q,\ \phi\ \wedge\ \psi\rangle \xrightarrow{\alpha} \langle P'|Q,\ \phi'\ \wedge\ \psi\rangle}$$

$$\texttt{Protocol} \qquad \frac{\langle P,\ \phi\rangle \xrightarrow{\alpha} \langle P',\ \phi'\rangle,\quad \alpha{\notin}C \quad \text{and} \quad \mathrm{fv}(\phi){\cap}\mathrm{fv}(\psi){=}\emptyset}{\langle P\|Q,\ \phi\ \wedge\ \psi\rangle \xrightarrow{\alpha} \langle P'\|Q,\ \phi'\ \wedge\ \psi\rangle}$$

$$\texttt{Synchronisation} \qquad \frac{\langle P,\ \phi\rangle \xrightarrow{\overline{c_{id}}(t)} \langle P',\ \phi'\rangle,\ \langle Q,\ \psi\rangle \xrightarrow{c_{id'}(x)} \langle Q',\ \psi'\rangle \text{ and } \mathrm{fv}(\phi){\cap}\mathrm{fv}(\psi){=}\emptyset}{\langle P\|Q,\ \phi\ \wedge\ \psi\rangle \xrightarrow{\overline{\delta^c_{id}}(t)} \langle P'\|Q,\ \varphi_1\rangle \xrightarrow{\delta^c_{id'}(t)} \langle P'\|Q',\ \varphi_2\rangle}$$

with $\varphi_1 ::= \phi' \wedge \psi$ and $\varphi_2 ::= \phi' \wedge \psi' \wedge x == t$.

$$\texttt{Restriction} \qquad \frac{\langle P,\ \phi\rangle \xrightarrow{\alpha} \langle P',\ \phi'\rangle}{\langle P\backslash L,\ \phi\rangle \xrightarrow{\alpha} \langle P'\backslash L,\ \phi'\ \wedge\ \phi^L_\alpha\rangle}$$

where ϕ^L_α is such that $\forall_\varrho\ (\varrho \models \phi^L_\alpha \quad \text{iff} \quad \varrho(\alpha) \in Act \setminus L)$.

$$\texttt{Observation} \qquad \frac{\langle P,\ \phi\rangle \xrightarrow{\gamma} \langle P',\ \phi'\rangle \quad \text{and} \quad \gamma\ \in\ \mathcal{O}^{-1}(\alpha)}{\langle P/\mathcal{O},\ \phi\rangle \xrightarrow{\alpha} \langle P'/\mathcal{O},\ \phi'\rangle}$$

Figure 2: *Semantics of constrained processes.*

Example 3. Consider the following SPPA processes:

$$A ::= \ c(x_1).A_1, \quad A_1 ::= \ c(x_2).A_2 \quad \text{and} \quad A_2 ::= \ [x_1 = x_2]\ \overline{c}(x_1).A_1\ .$$

The semantics of the constrained process $\langle A,\ \mathbf{1}\rangle$ is illustrated in Fig. 3. Notice that rule `Match` yields the transition

$$\langle A_2,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle \xrightarrow{\overline{c}_{id_A}(x_1)} \langle A_1,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2) \wedge x_1 == x_2\rangle,$$

but we write $\langle A_2,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle \stackrel{\overline{c}_{id_A}(x_1)}{\longrightarrow} \langle A_1,\ x_1 == x_2\rangle$ since

$$\mathcal{M}(x_1) \wedge \mathcal{M}(x_2) \wedge x_1 == x_2 \quad \stackrel{\mathcal{V}}{\Leftrightarrow} \quad x_1 == x_2.$$

Thus $\langle A_1,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2) \wedge x_1 == x_2\rangle$ and $\langle A_1,\ x_1 == x_2\rangle$ correspond to the same constrained process. Similarly, the transition

$$\langle A_1,\ x_1 == x_2\rangle \stackrel{\overline{c}_{id_A}(x_1)}{\longrightarrow} \langle A_2,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle$$

follows from the `Input` rule and the fact that

$$(\exists_{x_2}\ x_1 == x_2) \wedge \mathcal{M}(x_2) \quad \stackrel{\mathcal{V}}{\Leftrightarrow} \quad \mathcal{M}(x_1) \wedge \mathcal{M}(x_2).$$

$$\begin{tikzcd}
\langle A,\ \mathbf{1}\rangle \arrow[r, "c_{id_A}(x_1)"] & \langle A_1,\ \mathcal{M}(x_1)\rangle \arrow[r, "c_{id_A}(x_2)"] & \langle A_2,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle \arrow[d, shift right, "\overline{c}_{id_A}(x_1)"'] \\
 & & \langle A_1,\ x_1 == x_2\rangle \arrow[u, shift right, "c_{id_A}(x_2)"']
\end{tikzcd}$$

Figure 3: *Symbolic Semantics of* $\langle A,\ \mathbf{1}\rangle$.

Example 4. Consider the following processes:

$$B ::= \ c(y_2).B_1, \quad B_1 ::= \ \texttt{let}\ y_3 = \mathrm{enc}(y_1, y_2)\ \texttt{in}\ B_2 \quad \text{and} \quad B_2 ::= \overline{c}(y_3).\mathbf{0}$$

where $\mathrm{fv}(B) = \{y_1\}$. The symbolic semantics of the constrained process $\langle B,\ \mathcal{M}(y_1)\rangle$ is given in Fig. 4, where $\phi ::= \ \mathcal{K}(y_1) \ \wedge \ \mathcal{M}(y_2) \ \wedge \ y_3 == \{y_2\}_{y_1}$.

$$\langle B,\ \mathcal{M}(y_1)\rangle \xrightarrow{c_{id_B}(y_2)} \langle B_1,\ \mathcal{M}(y_1) \wedge \mathcal{M}(y_2)\rangle \xrightarrow{\mathrm{enc}_{id_B}} \langle B_2,\ \phi\rangle \xrightarrow{\overline{c_{id_B}}(y_3)} \langle \mathbf{0},\ \phi\rangle$$

Figure 4: *Symbolic Semantics of* $\langle B,\ \mathcal{M}(y_1)\rangle$.

3.3 Symbolic Semantics vs Value-Passing Semantics

The relationship between the symbolic operational semantics of constrained processes and the value-passing operational semantics of processes (see Appendix B) is detailed in the following lemmas. Every sequence of transitions between SPPA processes can be unwound to a sequence of transitions between constrained processes. Conversely, every transition between constrained processes can be interpreted as a set of transitions between processes.

For the remainder of this paper, we will discriminate between two types of actions: actions $\overline{c_{id}}(t)$, $\delta^c_{id}(t)$, $\overline{\delta^c_{id}}(t)$, signv_{id} and τ (denoted by α), and actions $c_{id}(x)$, f_{id}, split_{id} and dec_{id} (denoted by β). From the symbolic operational semantics, we see that an action β introduces a new variable x (or two, x and y, in the case of action split_{id}), while an action α does not. In the following, we will assume that, given an action β, x is always the introduced variable. Moreover, for simplicity purposes, we will omit the case of action split_{id} which introduces two variables and treat it as any other action β. It is easy to see that this last assumption will not affect the results presented is this paper since complete proofs can be obtained by adding special cases for action split_{id}.

Lemma 3. *Let P, P' be SPPA processes, let $\alpha' = \overline{c_{id}}(t)$, $\delta^c_{id}(t)$, $\overline{\delta^c_{id}}(t)$, signv_{id} or τ, for some term t such that $\mathrm{fv}(t) \subseteq \{x_1, \ldots, x_n\}$, and let $\beta' = c_{id}(x)$, f_{id}, split_{id} or dec_{id}.*

- *If $P[a_1/x_1]\ldots[a_n/x_n] \xrightarrow{\alpha} P'[a_1/x_1]\ldots[a_n/x_n]$, for some $a_1, \ldots, a_n \in \mathcal{M}$ ($n \geq 0$) and $\alpha = \alpha'[a_1/x_1]\ldots[a_n/x_n]$, then, for any formula $\phi \not\Leftrightarrow \mathbf{0}$ such that $\mathrm{fv}(\phi) = \{x_1, \ldots, x_n\}$, we have*

$$\langle P,\ \phi\rangle \xrightarrow{\alpha'} \langle P',\ \phi'\rangle$$

where ϕ' is the formula given by the symbolic operational semantics (with $\phi' \not\Leftrightarrow \mathbf{0}$).

- *If $P[a_1/x_1]\ldots[a_n/x_n] \xrightarrow{\beta} P'[a/x][a_1/x_1]\ldots[a_n/x_n]$, for some $a_1, \ldots, a_n, a \in \mathcal{M}$ ($n \geq 0$) and $\beta = \beta'[a/x]$, then, for any formula $\phi \not\Leftrightarrow \mathbf{0}$ such that $\mathrm{fv}(\phi) = \{x_1, \ldots, x_n\}$, we have*

$$\langle P,\ \phi\rangle \xrightarrow{\beta'} \langle P',\ \phi'\rangle$$

where ϕ' is the formula given by the symbolic operational semantics (with $\phi' \not\Leftrightarrow \mathbf{0}$).

Lemma 4. *Let P, P' be SPPA processes, let $\alpha' = \overline{c_{id}}(t)$, $\delta^c_{id}(t)$, $\overline{\delta^c_{id}}(t)$, signv_{id} or τ, for some term t such that $\mathrm{fv}(t) \subseteq \{x_1, \ldots, x_n\}$, and let $\beta' = c_{id}(x)$, f_{id}, split_{id} or dec_{id}.*

- *If $\langle P,\ \phi\rangle \xrightarrow{\alpha'} \langle P',\ \phi'\rangle$, with $\mathrm{fv}(\phi) = \mathrm{fv}(\phi') = \{x_1, \ldots, x_n\}$, then, for every valuation ϱ such that $\varrho \models \phi'$,*

$$P[\varrho(x_1)/x_1]\ldots[\varrho(x_n)/x_n] \xrightarrow{\alpha} P'[\varrho(x_1)/x_1]\ldots[\varrho(x_1)/x_n]$$

where $\alpha = \varrho(\alpha')$.

– *If* $\langle P,\ \phi\rangle \xrightarrow{\beta'} \langle P',\ \phi'\rangle$*, with* $\mathrm{fv}(\phi) = \{x_1, \ldots, x_n\}$ *and* $\mathrm{fv}(\phi') = \mathrm{fv}(\phi) \cup \{x\}$*, then, for every valuation* ϱ *such that* $\varrho \models \phi'$*,*

$$P[\varrho(x_1)/x_1] \ldots [\varrho(x_n)/x_n] \xrightarrow{\beta} P'[\varrho(x)/x][\varrho(x_1)/x_1] \ldots [\varrho(x_n)/x_n]$$

where $\beta = \varrho(\beta')$*.*

Lemma 5. *Let* P, P' *be SPPA processes, let* $\alpha' = \overline{c_{id}}(t)$*,* $\delta^c_{id}(t)$*,* $\overline{\delta^c_{id}}(t)$*,* signv_{id} *or* τ*, for some term* t *such that* $\mathrm{fv}(t) \subseteq \{x_1, \ldots, x_n\}$*, and let* $\beta' = c_{id}(x)$*,* f_{id}*,* split_{id} *or* dec_{id}*. Consider* $a, a_1, \ldots, a_n \in \mathcal{M}$*, and let* $\alpha = \alpha'[a_1/x_1] \ldots [a_n/x_n]$ *and* $\beta = \beta'[a/x]$*.*

– *If* $P[a_1/x_1] \ldots [a_n/x_n] \xrightarrow{\alpha} P'[a_1/x_1] \ldots [a_n/x_n]$ *and* $\models \phi[a_1/x_1] \ldots [a_n/x_n]$*, then* $\models \phi'[a_1/x_1] \ldots [a_n/x_n]$ *whenever*

$$\langle P,\ \phi\rangle \xrightarrow{\alpha'} \langle P',\ \phi'\rangle.$$

– *If* $P[a_1/x_1] \ldots [a_n/x_n] \xrightarrow{\beta} P'[a/x][a_1/x_1] \ldots [a_n/x_n]$ *and* $\models \phi[a_1/x_1] \ldots [a_n/x_n]$*, then* $\models \phi'[a/x][a_1/x_1] \ldots [a_n/x_n]$ *whenever*

$$\langle P,\ \phi\rangle \xrightarrow{\beta'} \langle P',\ \phi'\rangle.$$

The proofs of Lemma 3, Lemma 4 and Lemma 5 are given in Appendix D.

4 Bisimulation Equivalence for Constrained Processes

In this section, we extend Milner's notion of strong bisimulation [16] to handle constrained processes. But first, in order to bind the variables of the compared constrained processes, we need to consider finite relations between their free variables. Recall that whenever we compare two processes P and Q, we always assume that no variable has been used in both definitions.

4.1 Relations Between Variables

Consider the family $\mathfrak{R}$ of all relation between finite subsets of variables, hence $\mathfrak{R} = \{R \mid R \subseteq \mathcal{V}_1 \times \mathcal{V}_2$ for some finite sets $\mathcal{V}_1, \mathcal{V}_2 \subseteq \mathcal{V}\}$. A relation $R \in \mathfrak{R}$ is said to be a *full relation between* $\mathcal{V}_1$ *and* $\mathcal{V}_2$ whenever $\forall_{x \in \mathcal{V}_1} \exists_{y \in \mathcal{V}_2}\ (x, y) \in R$ and $\forall_{y \in \mathcal{V}_2} \exists_{x \in \mathcal{V}_1}\ (x, y) \in R$ (or $R = \emptyset$ if either $\mathcal{V}_1 = \emptyset$ or $\mathcal{V}_2 = \emptyset$).

For any variable $x \in \mathcal{V}$ and any relation $R \in \mathfrak{R}$, we consider the relation $R[x] \in \mathfrak{R}$ defined by $R[x] = \{(x', y') \in R \mid x' \neq x$ and $y' \neq x\}$. Moreover, for any $x, y \in \mathcal{V}$, we consider the relation $R[\![(x, y)]\!] \in \mathfrak{R}$ defined by

$$R[\![(x, y)]\!] = R[x][y] \ \cup\ \{(x, y)\}.$$

The relation $R[\![(x,y)]\!]$ is therefore obtained from R by, first removing every occurrence of x and y, and then adding (x,y).

A valuation ϱ is said to be *consistent* with the relation $R \in \mathfrak{R}$ if $\varrho(x) = \varrho(y)$ whenever $(x,y) \in R$. Given $x, y \in \mathcal{V}$, we define $\varrho[x/y]$ as the valuation obtained from ϱ by setting $\varrho[x/y](y) = \varrho(x)$ (and $\varrho[x/y](z) = \varrho(z)$ otherwise). It is easy to see that the valuation $\varrho[x/y]$ is consistent with the relation $R[\![(x,y)]\!]$ whenever ϱ is consistent with $R[x]$.

4.2 Bisimulation

For the following definition, recall from Section 3.3 that, given an action $\beta = c_{id}(x)$, f_{id}, split_{id} or dec_{id}, we assume that x is always the variable introduced by β.

Definition 6. (Bisimulation) Let $\langle P,\ \phi\rangle$ and $\langle Q,\ \psi\rangle$ be constrained processes and let $R \in \mathfrak{R}$ be a full relation between $\text{fv}(\phi)$ and $\text{fv}(\psi)$. A *bisimulation* between $\langle P,\ \phi\rangle$ and $\langle Q,\ \psi\rangle$ with respect to R is a family of relations $\mathcal{R} = \{\mathcal{R}^\varrho\}_\varrho$, for every valuation ϱ, where each relation $\mathcal{R}^\varrho \subseteq \mathcal{D}(\langle P,\ \phi\rangle) \times \mathcal{D}(\langle Q,\ \psi\rangle) \times \mathfrak{R}$ satisfies the following conditions:

1. If $\varrho \models \phi$, $\varrho \models \psi$ and ϱ is consistent with R, then $(\langle P,\ \phi\rangle, \langle Q,\ \psi\rangle, R) \in \mathcal{R}^\varrho$;
2. Whenever $(\langle P_1,\ \phi_1\rangle, \langle Q_1,\ \psi_1\rangle, R_1) \in \mathcal{R}^\varrho$, for any action $\alpha = \overline{c_{id}}(t)$, $\delta^c_{id}(t)$, $\overline{\delta^c_{id}}(t)$, signv_{id} or τ, and any action $\beta = c_{id}(x)$, f_{id}, split_{id} or dec_{id}, we have
   - if $\langle P_1,\ \phi_1\rangle \xrightarrow{\alpha} \langle P_2,\ \phi_2\rangle$ and $\varrho \models \phi_2$, then $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1) \in \mathcal{R}^\varrho$, where Q_2 and ψ_2 are such that $\varrho \models \psi_2$ and $\langle Q_1,\ \psi_1\rangle \xrightarrow{\alpha'} \langle Q_2,\ \psi_2\rangle$, and α' is such that $\alpha' = \alpha[y_1/x_1]\ldots[y_n/x_n]$ for some $(x_i, y_i) \in R_1$;
   - if $\langle Q_1,\ \psi_1\rangle \xrightarrow{\alpha} \langle Q_2,\ \psi_2\rangle$ and $\varrho \models \psi_2$, then $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1) \in \mathcal{R}^\varrho$ where P_2 and ϕ_2 are such that $\varrho \models \phi_2$ and $\langle P_1,\ \phi_1\rangle \xrightarrow{\alpha'} \langle P_2,\ \phi_2\rangle$, and $\alpha' = \alpha[y_1/x_1]\ldots[y_n/x_n]$ for some $(y_i, x_i) \in R_1$;
   - if $\langle P_1,\ \phi_1\rangle \xrightarrow{\beta} \langle P_2,\ \phi_2\rangle$, then $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1[\![(x,y)]\!]) \in \mathcal{R}^{\varrho'[x/y]}$ for every valuation ϱ' consistent with $R_1[x]$ such that $\varrho' \models \phi_2$, where Q_2 and ψ_2 are such that $\varrho'[x/y] \models \psi_2$ and $\langle Q_1,\ \psi_1\rangle \xrightarrow{\beta'} \langle Q_2,\ \psi_2\rangle$, and $\beta' = \beta[y/x]$;
   - if $\langle Q_1,\ \psi_1\rangle \xrightarrow{\beta} \langle Q_2,\ \psi_2\rangle$, then $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1[\![(x,y)]\!]) \in \mathcal{R}^{\varrho'[x/y]}$ for every valuation ϱ' consistent with $R_1[x]$ such that $\varrho' \models \psi_2$, where P_2 and ϕ_2 are such that $\varrho'[x/y] \models \phi_2$ and $\langle P_1,\ \phi_1\rangle \xrightarrow{\beta'} \langle P_2,\ \phi_2\rangle$, and $\beta' = \beta[y/x]$.

Two constrained processes $\langle P,\ \phi\rangle$ and $\langle Q,\ \psi\rangle$ are *bisimilar* if there exists a bisimulation which relates $\langle P,\ \phi\rangle$ and $\langle Q,\ \psi\rangle$ with respect to some full relation R between $\mathrm{fv}(\phi)$ and $\mathrm{fv}(\psi)$. In that case, we write $\langle P,\ \phi\rangle \simeq \langle Q,\ \psi\rangle$.

Note that if $\mathcal{R} = \{\mathcal{R}^{\varrho}\}_{\varrho}$ is a bisimulation, $\varrho \models \phi$ and $\varrho \models \psi$ whenever $(\langle P,\ \phi\rangle, \langle Q,\ \psi\rangle, R) \in \mathcal{R}^{\varrho}$. Moreover, ϱ must be consistent with the relation R.

4.3 Equivalence of the Bisimulations

In the following theorem, we see that the bisimulation equivalence between SPPA constrained processes, as defined above, corresponds to the strong bisimulation equivalence between SPPA processes. Obviously, this result holds only when the SPPA processes under comparison are compatible i.e., for processes without free variables. (We say that an SPPA process P is *closed* whenever $\mathrm{fv}(P) = \emptyset$.) First, we define strong bisimulation between SPPA (value-passing) processes.

Definition 7. A bisimulation between closed processes P and Q is a relation $\mathcal{R} \subseteq \mathcal{D}(P) \times \mathcal{D}(Q)$ such that

– $(P, Q) \in \mathcal{R}$;

– If $(P_1, Q_1) \in \mathcal{R}$ and $P_1 \xrightarrow{\alpha} P_2$, then $(P_2, Q_2) \in \mathcal{R}$, where Q_2 is such that $Q_1 \xrightarrow{\alpha} Q_2$;

– If $(P_1, Q_1) \in R$ and $Q_1 \xrightarrow{\alpha} Q_2$, then $(P_2, Q_2) \in \mathcal{R}$, where P_2 is such that $P_1 \xrightarrow{\alpha} P_2$,

where $\alpha \in \mathit{Act}$ is any (variable-free) action. We write $P \simeq Q$ whenever P and Q are related by some bisimulation.

Theorem 8. *Let P and Q be closed processes. Then, $P \simeq Q$ if and only if $\langle P,\ \mathbf{1}\rangle \simeq \langle Q,\ \mathbf{1}\rangle$.*

Proof. First, assume that $P \simeq Q$ and let $\mathcal{R}$ be a bisimulation between P and Q. Consider the family of relations $\mathcal{R}' = \{\mathcal{R}'^{\varrho}\}_{\varrho}$, where, given a valuation ϱ, the relation $\mathcal{R}'^{\varrho}$ is defined as follows:

– for every $(P', Q') \in \mathcal{R}$, $(\langle P',\ \mathbf{1}\rangle, \langle Q',\ \mathbf{1}\rangle, \emptyset) \in \mathcal{R}'^{\varrho}$;

– for every $(P'[\varrho(x_1)/x_1] \ldots [\varrho(x_n)/x_n], Q'[\varrho(y_1)/y_1] \ldots [\varrho(y_m)/y_m]) \in \mathcal{R}$, for every formulas ϕ and ψ, with $\mathrm{fv}(\phi) = \{x_1, \ldots, x_n\}$ and $\mathrm{fv}(\psi) = \{y_1, \ldots, y_m\}$, and for every relation $R \subseteq \mathrm{fv}(\phi) \times fv(\psi)$, $(\langle P',\ \phi\rangle, \langle Q',\ \psi\rangle, R) \in \mathcal{R}'^{\varrho}$ whenever $\varrho \models \phi$, $\varrho \models \psi$ and ϱ is consistent with R.

In the following, we show that $\mathcal{R}' = \{\mathcal{R}'^{\varrho}\}_{\varrho}$ is a bisimulation between $\langle P,\ \mathbf{1}\rangle$ and $\langle Q,\ \mathbf{1}\rangle$ with respect to the empty relation $\emptyset$. To achieve this goal, we show that each relation $\mathcal{R}'^{\varrho}$ fits the conditions from Definition 6.

1. First, we see that $(\langle P,\ \mathbf{1}\rangle, \langle Q,\ \mathbf{1}\rangle, \emptyset) \in \mathcal{R}'^{\varrho}$ (with $\varrho \models \mathbf{1}$ and ϱ is consistent with the empty relation $\emptyset$).

2. Let $(\langle P_1,\ \phi_1\rangle, \langle Q_1,\ \psi_1\rangle, R) \in \mathcal{R}'^{\varrho}$. First, we see that $\varrho \models \phi_1$, $\varrho \models \psi_1$ and ϱ is consistent with R. Moreover, we may assume that $\mathrm{fv}(\phi_1) = \{x_1, \ldots, x_n\}$ and $\mathrm{fv}(\psi_1) = \{y_1, \ldots, y_m\}$, with $R \subseteq \mathrm{fv}(\phi_1) \times \mathrm{fv}(\psi_1)$. Also, we have $(P_1', Q_1') \in \mathcal{R}$, with $P_1' ::= P_1[\varrho(x_1)/x_1] \ldots [\varrho(x_n)/x_n]$ and $Q_1' ::= Q_1[\varrho(y_1)/y_1] \ldots [\varrho(y_m)/y_m]$.

   Assume that $\varrho \models \phi_2$ and $\langle P_1,\ \phi_1\rangle \xrightarrow{\alpha'} \langle P_2,\ \phi_2\rangle$ with $\alpha' = \overline{c_{id}}(t)$, $\delta^c_{id}(t)$, $\overline{\delta^c_{id}}(t)$, signv_{id} or τ. By Lemma 4,

   $$P_1' \xrightarrow{\alpha} P_2'$$

   where $P_2' ::= P_2[\varrho(x_1)/x_1] \ldots [\varrho(x_n)/x_n]$ and $\alpha = \varrho(\alpha')$. Since $(P_1', Q_1') \in \mathcal{R}$, there is a transition

   $$Q_1' \xrightarrow{\alpha} Q_2'$$

   such that $(P_2', Q_2') \in \mathcal{R}$, where $Q_2' ::= Q_2[\varrho(y_1)/y_1] \ldots [\varrho(y_m)/y_m]$. By Lemma 3, there is a transition

   $$\langle Q_1,\ \psi_1\rangle \xrightarrow{\alpha''} \langle Q_2,\ \psi_2\rangle$$

   with $\alpha = \alpha''[\varrho(y_1)/y_1] \ldots [\varrho(y_m)/y_m]$ and $\varrho \models \psi_2$ (by Lemma 5). Moreover, since ϱ is consistent with R, we may assume that $\alpha' = \alpha''[y_1'/x_1] \ldots [y_n'/x_n]$ with $\{y_1', \ldots, y_n'\} = \{y_1, \ldots, y_m\}$ and $(x_i, y_i') \in R$. Therefore, since $\varrho \models \phi_2$, $\varrho \models \psi_2$ and ϱ is consistent with R, we see that $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R) \in \mathcal{R}'^{\varrho}$. The case where $\langle Q_1,\ \psi_1\rangle \xrightarrow{\alpha'} \langle Q_2,\ \psi_2\rangle$ is similar.

   Now assume that $\langle P_1,\ \phi_1\rangle \xrightarrow{\beta'} \langle P_2,\ \phi_2\rangle$, with $\beta' = c_{id}(x)$, f_{id}, split_{id} or dec_{id}, and let ϱ' be a valuation consistent with $R[x]$ such that $\varrho' \models \phi_2$. By Lemma 4,

   $$P_1' \xrightarrow{\beta} P_2'$$

   where $\beta = \varrho'(\beta')$ and $P_2' ::= P_2[\varrho'(x)/x][\varrho'(x_1)/x_1] \ldots [\varrho'(x_n)/x_n]$. Since $(P_1', Q_1') \in \mathcal{R}$, there is a transition

   $$Q_1' \xrightarrow{\beta} Q_2'$$

   such that $(P_2', Q_2') \in \mathcal{R}$, where $Q_2' ::= Q_2[\varrho'(x)/y][\varrho'(y_1)/y_1] \ldots [\varrho'(y_m)/y_m]$ for some y. By Lemma 3, there is a transition

   $$\langle Q_1,\ \phi_1\rangle \xrightarrow{\beta''} \langle Q_2,\ \psi_2\rangle$$

   with $\beta = \beta''[\varrho'(x)/y] = \varrho'(\beta'')$ and $\varrho'[x/y] \models \psi_2$ (by Lemma 5). Moreover, since ϱ' is consistent with $R[x]$, we may assume that $\beta' = \beta''[y/x]$. Therefore, since $\varrho'[x/y] \models \phi_2$, $\varrho'[x/y] \models \psi_2$ and $\varrho'[x/y]$ is consistent with $R[\![(x,y)]\!]$, we see that $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R[\![(x,y)]\!]) \in \mathcal{R}'^{\varrho'[x/y]}$. The case where $\langle Q_1,\ \psi_1\rangle \xrightarrow{\beta'} \langle Q_2,\ \psi_2\rangle$ is similar.

Conversely, assume that $\langle P,\ \mathbf{1}\rangle \simeq \langle Q,\ \mathbf{1}\rangle$ and let $\mathcal{R} = \{\mathcal{R}^\varrho\}$ be a bisimulation between $\langle P,\ \mathbf{1}\rangle$ and $\langle Q,\ \mathbf{1}\rangle$ with respect to $\emptyset$. Consider the relation $\mathcal{R}' \subseteq \mathcal{D}(P) \times \mathcal{D}(Q)$ defined as follows:

– If $(\langle P',\ \mathbf{1}\rangle,\ \langle Q',\ \mathbf{1}\rangle,\ \emptyset) \in \mathcal{R}^\varrho$ for some ϱ, then $(P', Q') \in \mathcal{R}'$;

– If $(\langle P',\ \phi\rangle,\ \langle Q',\ \psi\rangle,\ R) \in \mathcal{R}^\varrho$ for some ϱ, then

$$(P'[\varrho(x_1)/x_1]\ldots[\varrho(x_1)/x_n],\ Q'[\varrho(y_1)/y_1]\ldots[\varrho(y_m)/y_m]) \in \mathcal{R}'$$

where $\mathrm{fv}(\phi) = \{x_1, \ldots, x_n\}$ and $\mathrm{fv}(\psi) = \{y_1, \ldots, y_m\}$.

In the following, we show that $\mathcal{R}'$ is a bisimulation between P and Q.

1. First, we see that $(P, Q) \in \mathcal{R}'$ since $(\langle P,\ \mathbf{1}\rangle,\ \langle Q,\ \mathbf{1}\rangle, \emptyset) \in \mathcal{R}^\varrho$. for any valuation ϱ.

2. Let $P_1' ::= P_1[a_1/x_1]\ldots[a_n/x_n]$ and $Q_1' ::= Q_1[b_1/y_1]\ldots[b_m/y_m]$, for some $a_i, b_j \in \mathcal{M}$, be such that $(P_1', Q_1') \in \mathcal{R}'$. By the definition of $\mathcal{R}'$, there is a valuation ϱ, with $\varrho(x_i) = a_i$ and $\varrho(y_j) = b_j$, such that $(\langle P_1,\ \phi_1\rangle,\ \langle Q_1,\ \psi_1\rangle,\ R) \in \mathcal{R}^\varrho$ with $\mathrm{fv}(\phi_1) = \{x_1, \ldots, x_n\}$ and $\mathrm{fv}(\psi_1) = \{y_1, \ldots, y_m\}$. Moreover, we have $\varrho \models \phi_1$, $\varrho \models \psi_1$ and ϱ is consistent with R.

   Let $\alpha' = \overline{c}(t)$, $\delta^c_{id}(t)$, $\overline{\delta^c_{id}}(t)$, signv_{id} or τ, and assume that $P_1' \xrightarrow{\alpha} P_2'$ where $\alpha = \alpha'[\varrho(x_1)/x_1]\ldots[\varrho(x_n)/x_n] = \varrho(\alpha)$ and $P_2' ::= P_2[\varrho(x_1)/x_1]\ldots[\varrho(x_n)/x_n]$. By Lemma 3 and since $\mathrm{fv}(\phi_1) = \{x_1, \ldots, x_n\}$, we have

   $$\langle P_1,\ \phi_1\rangle \xrightarrow{\alpha'} \langle P_2,\ \phi_2\rangle$$

   with $\varrho \models \phi_2$ (by Lemma 5). But since $(\langle P_1,\ \phi_1\rangle,\ \langle Q_1,\ \psi_1\rangle,\ R) \in \mathcal{R}^\varrho$, there is a transition

   $$\langle Q_1,\ \psi_1\rangle \xrightarrow{\alpha''} \langle Q_2,\ \psi_2\rangle$$

   for $\alpha' = \alpha[y_1'/x_1]\ldots[y_n'/x_n]$ with $\{y_1', \ldots, y_n'\} = \{y_1, \ldots, y_m\}$ and $(x_i, y_i') \in R$, such that $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R) \in \mathcal{R}^\varrho$ and $\varrho \models \psi_2$. Also, since ϱ is consistent with R, we have $\varrho(\alpha'') = \varrho(\alpha') = \alpha$. Therefore, by Lemma 4 and since $\varrho \models \psi_2$, we have

   $$Q_1' \xrightarrow{\alpha} Q_2'$$

   where $Q_2' ::= Q_2[\varrho(y_1)/y_1]\ldots[\varrho(y_m)/y_m]$. Moreover, since $\mathrm{fv}(\phi_2) = \mathrm{fv}(\phi_1) = \{x_1, \ldots, x_n\}$ and $\mathrm{fv}(\psi_2) = \mathrm{fv}(\psi_1) = \{y_1, \ldots, y_m\}$, and since $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R) \in \mathcal{R}^\varrho$, we see from the definition of $\mathcal{R}'$ that $(P_2', Q_2') \in \mathcal{R}'$. The case where $Q_1' \xrightarrow{\alpha} Q_2'$ is similar.

Now assume that $P'_1 \xrightarrow{\beta} P'_2$ where $\beta = \beta'[a/x]$ for some $\beta' = c_{id}(x)$, f_{id}, split_{id} or dec_{id}, and $P'_2 ::= P_2[a/x][\varrho(x_1)/x_1]\dots[\varrho(x_n)/x_n]$. Consider the valuation ϱ' defined as follows:

$$\varrho'(x) = a \quad \text{and} \quad \varrho'(z) = \varrho(z) \text{ otherwise.}$$

Since ϱ is consistent with R, then ϱ' is consistent with $R[x]$ and $\beta = \varrho'(\beta')$. Furthermore, we see that $P'_2 = P_2[\varrho'(x)/x][\varrho'(x_1)/x_1]\dots[\varrho'(x_n)/x_n]$. By Lemma 3, and since $\mathrm{fv}(\phi_1) = \{x_1, \dots, x_n\}$, we have

$$\langle P_1,\ \phi_1\rangle \xrightarrow{\beta'} \langle P_2,\ \phi_2\rangle$$

with $\varrho' \models \phi_2$ (by Lemma 5). But since $(\langle P_1,\ \phi_1\rangle,\ \langle Q_1,\ \psi_1\rangle,\ R) \in \mathcal{R}^{\varrho}$, there is a transition

$$\langle Q_1,\ \psi_1\rangle \xrightarrow{\beta''} \langle Q_2,\ \psi_2\rangle$$

for $\beta' = \beta[y/x]$, such that $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R[\![(x, y)]\!]) \in \mathcal{R}^{\varrho'[x/y]}$ and $\varrho'[x/y] \models \psi_2$. Also, since ϱ' is consistent with $R[x]$, therefore ϱ' is consistent with $R[\![(x, y)]\!]$, we have $\varrho'[x/y](\beta'') = \varrho'[x/y](\beta') = \beta$. Put $\varrho'' = \varrho'[x/y]$. By Lemma 4 and since $\varrho'' \models \psi_2$, we have

$$Q'_1 \xrightarrow{\beta} Q'_2$$

where $Q'_2 ::= Q_2[\varrho''(y)/y][\varrho''(y_1)/y_1]\dots[\varrho''(y_m)/y_m]$ with $\varrho''(y) = a$, $\varrho''(y_1) = \varrho(y_1), \dots, \varrho''(y_m) = \varrho(y_m)$. Moreover, since $\mathrm{fv}(\phi_2) = \mathrm{fv}(\phi_1) \cup \{x\} = \{x_1, \dots, x_n, x\}$ and $\mathrm{fv}(\psi_2) = \mathrm{fv}(\psi_1) \cup \{y\} = \{y_1, \dots, y_m, y\}$, and since $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R[\![(x, y)]\!]) \in \mathcal{R}^{\varrho''}$, we see from the definition of $\mathcal{R}'$ that $(P'_2, Q'_2) \in \mathcal{R}'$. The case where $Q'_1 \xrightarrow{\beta} Q'_2$ is similar. $\square$

5 Symbolic Bisimulation: A Proof Method for Bisimulation

In this section, we introduce a symbolic bisimulation relation for constrained processes which can be constructed within a finite number of steps. We also show that this symbolic bisimulation relation is equivalent to the bisimulation relation introduced in Definition 6. Our symbolic bisimulation may therefore serve as a sound and complete finite proof method for the bisimulation of constrained processes.

5.1 Equivalence Relation over Valuations

For the following, we consider two constrained processes $\langle P,\ \phi\rangle$ and $\langle Q,\ \psi\rangle$. We also consider the following sets:

$$\{\phi' \mid \langle P',\ \phi'\rangle \in \mathcal{D}(\langle P,\ \phi\rangle) \text{ for some } P'\} = \{\phi_1, \dots, \phi_m\},$$

$$\{\psi' \mid \langle Q',\ \psi'\rangle \in \mathcal{D}(\langle Q,\ \psi\rangle) \text{ for some } Q'\} = \{\psi_1, \ldots, \psi_{m'}\},$$
$$\bigcup_{1\leq j\leq m} \mathrm{fv}(\phi_j) \ = \ \{x_1, \ldots, x_n\} \quad \text{and} \quad \bigcup_{1\leq j\leq m'} \mathrm{fv}(\psi_j) \ = \ \{y_1, \ldots, y_{n'}\}.$$

Such sets of formulas and variables are finite (up to formula equivalence) since the sets $\mathcal{D}(\langle P,\ \phi\rangle)$ and $\mathcal{D}(\langle Q,\ \psi\rangle)$ are finite by Theorem 2. Now consider the equivalence relation over valuations defined as follows: $\varrho \equiv \varrho'$ if

- for every $1 \leq i \leq m$, $\quad \varrho \models \phi_i \quad$ iff $\quad \varrho' \models \phi_i$;
- for every $1 \leq i \leq m'$, $\quad \varrho \models \psi_i \quad$ iff $\quad \varrho' \models \psi_i$;
- for every $z, z' \in \{x_1, \ldots, x_n\} \cup \{y_1, \ldots, y_{n'}\}$,
$$\varrho(z) = \varrho(z') \quad \text{iff} \quad \varrho'(z) = \varrho'(z').$$

Also consider the equivalence class (with respect to $\langle P,\ \phi\rangle$ and $\langle Q,\ \psi\rangle$) of a valuation ϱ defined as follows:

$$[\![\varrho]\!] \ = \ \{\varrho' \mid \varrho' \equiv \varrho\}.$$

Lemma 9. *Let ϱ and ϱ' be valuations such that $\varrho \equiv \varrho'$.*

1. *For any relation $R \subseteq \{x_1, \ldots, x_n\} \times \{y_1, \ldots, y_{n'}\}$,*
$$\varrho \textit{ is consistent with } R \textit{ if and only if } \varrho' \textit{ is consistent with } R.$$
2. *For any $x, y \in \{x_1, \ldots, x_n\} \cup \{y_1, \ldots, y_{n'}\}$,*
$$\varrho[x/y] \equiv \varrho'[x/y] \quad \textit{and} \quad \varrho[y/x] \equiv \varrho'[y/x].$$

The proof of Lemma 9 is straightforward from the definition of the equivalence relation $\equiv$.

Lemma 10. *There are finitely many equivalence classes $[\![\varrho]\!]$ i.e., the set $\{[\![\varrho]\!] \mid \varrho$ is a valuation$\}$ is finite.*

Proof. Follows from the fact that the sets $\{\phi' \mid \langle P',\ \phi'\rangle \in \mathcal{D}(\langle P,\ \phi\rangle)$ for some $P'\}$ and $\{\psi' \mid \langle Q',\ \psi'\rangle \in \mathcal{D}(\langle Q,\ \psi\rangle)$ for some $Q'\}$ are finite (up to formula equivalence). Therefore, using the notation established above, we see that

- there are at most 2^m equivalence classes for the relation:
$$\forall_{1\leq i\leq m} \quad \varrho \models \phi_i \quad \text{iff} \quad \varrho' \models \phi_i;$$
- there are at most $2^{m'}$ equivalence classes for the relation:
$$\forall_{1\leq i\leq m'} \quad \varrho \models \psi_i \quad \text{iff} \quad \varrho' \models \psi_i;$$
- there are at most $2^{(n+n')^2}$ equivalence classes for the relation:
$$\forall_{z,z'\in\{x_1,\ldots,x_n\}\cup\{y_1,\ldots,y_{n'}\}} \quad \varrho(z) = \varrho(z') \quad \text{iff} \quad \varrho'(z) = \varrho'(z').$$

Hence, there are at most $2^{m+m'+(n+n')^2}$ equivalence classes $[\![\varrho]\!]$. $\square$

5.2 Symbolic Bisimulation

Definition 11. (Symbolic Bisimulation) Let $\langle P,\ \phi\rangle$ and $\langle Q,\ \psi\rangle$ be constrained processes and let $R \in \mathfrak{R}$ be a full relation between $\mathrm{fv}(\phi)$ and $\mathrm{fv}(\psi)$. A *symbolic bisimulation* between $\langle P,\ \phi\rangle$ and $\langle Q,\ \psi\rangle$ with respect to R is a finite family $\mathcal{R} = \{\mathcal{R}^{[\![\varrho]\!]}\}_\varrho$, for every valuation ϱ, where each relation $\mathcal{R}^{[\![\varrho]\!]}$ satisfies the following conditions:

1. If $\varrho \models \phi$, $\varrho \models \psi$ and ϱ is consistent with R, then $(\langle P,\ \phi\rangle,\ \langle Q,\ \psi\rangle,\ R) \in \mathcal{R}^{[\![\varrho]\!]}$;
2. Whenever $(\langle P_1,\ \phi_1\rangle,\ \langle Q_1,\ \psi_1\rangle,\ R_1) \in \mathcal{R}^{[\![\varrho]\!]}$, for any action $\alpha = \overline{c_{id}}(t)$, $\delta^c_{id}(t)$, $\overline{\delta^c_{id}}(t)$, signv_{id} or τ, and any action $\beta = c_{id}(x)$, f_{id}, split_{id} or dec_{id}, we have
   – if $\langle P_1,\ \phi_1\rangle \xrightarrow{\alpha} \langle P_2,\ \phi_2\rangle$ and $\varrho \models \phi_2$, then $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1) \in \mathcal{R}^{[\![\varrho]\!]}$, where Q_2 and ψ_2 are such that $\varrho \models \psi_2$ and $\langle Q_1,\ \psi_1\rangle \xrightarrow{\alpha'} \langle Q_2,\ \psi_2\rangle$, and $\alpha' = \alpha[y_1/x_1]\ldots[y_n/x_n]$ for some $(x_i, y_i) \in R_1$;

   – if $\langle Q_1,\ \psi_1\rangle \xrightarrow{\alpha} \langle Q_2,\ \psi_2\rangle$ and $\varrho \models \psi_2$, then $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1) \in \mathcal{R}^{[\![\varrho]\!]}$, where P_2 and ϕ_2 are such that $\varrho \models \phi_2$ and $\langle P_1,\ \phi_1\rangle \xrightarrow{\alpha'} \langle P_2,\ \phi_2\rangle$, and $\alpha' = \alpha[y_1/x_1]\ldots[y_n/x_n]$ for some $(y_i, x_i) \in R_1$;

   – if $\langle P_1,\ \phi_1\rangle \xrightarrow{\beta} \langle P_2,\ \phi_2\rangle$, then $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1[\![(x,y)]\!]) \in \mathcal{R}^{[\![\varrho'[x/y]]\!]}$ for every valuation ϱ' consistent with $R_1[x]$ such that $\varrho' \models \phi_2$, where Q_2 and ψ_2 are such that $\varrho'[x/y] \models \psi_2$ and $\langle Q_1,\ \psi_1\rangle \xrightarrow{\beta'} \langle Q_2,\ \psi_2\rangle$, and $\beta' = \beta[y/x]$;

   – if $\langle Q_1,\ \psi_1\rangle \xrightarrow{\beta} \langle Q_2,\ \psi_2\rangle$, then $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1[\![(x,y)]\!]) \in \mathcal{R}^{[\![\varrho'[x/y]]\!]}$ for every valuation ϱ' consistent with $R_1[x]$ such that $\varrho' \models \psi_2$, where P_2 and ϕ_2 are such that $\varrho'[x/y] \models \phi_2$ and $\langle P_1,\ \phi_1\rangle \xrightarrow{\beta'} \langle P_2,\ \phi_2\rangle$, and $\beta' = \beta[y/x]$.

We write $\langle P,\ \phi\rangle \simeq_{\mathrm{s}} \langle Q,\ \psi\rangle$ whenever there exists a symbolic bisimulation which relates $\langle P,\ \phi\rangle$ and $\langle Q,\ \psi\rangle$ with respect to some full relation R between $\mathrm{fv}(\phi)$ and $\mathrm{fv}(\psi)$.

The following theorem states that symbolic bisimulation is a sound and complete proof method for verifying bisimilarity between constrained processes.

Theorem 12. $\langle P,\ \phi\rangle \simeq \langle Q,\ \psi\rangle$ *if and only if* $\langle P,\ \phi\rangle \simeq_{\mathrm{s}} \langle Q,\ \psi\rangle$.

Proof. First, assume that $\langle P,\ \phi\rangle \simeq \langle Q,\ \psi\rangle$, and let $\mathcal{R} = \{\mathcal{R}^\varrho\}_\varrho$ be a bisimulation with respect to some full relation R between $\mathrm{fv}(\phi)$ and $\mathrm{fv}(\psi)$. For every equivalence class $[\![\varrho]\!]$, consider the relation

$$\mathcal{R}'^{[\![\varrho]\!]} = \bigcup_{\varrho' \in [\![\varrho]\!]} \mathcal{R}^{\varrho'}.$$

Hence $\mathcal{R}^{\varrho} \subseteq \mathcal{R}'^{[\![\varrho]\!]}$ for every ϱ. Then it is enough to show that the (finite) family $\mathcal{R}' = \{\mathcal{R}'^{[\![\varrho]\!]}\}_{\varrho}$ is a symbolic bisimulation with respect to R, therefore $\langle P,\ \phi\rangle \simeq_{\mathrm{s}} \langle Q,\ \psi\rangle$. Indeed, given an equivalence class $[\![\varrho]\!]$, we see that the $\mathcal{R}'^{[\![\varrho]\!]}$ satisfies every conditions from Definition 11.

1. If $\varrho \models \phi$, $\varrho \models \psi$, and ϱ is consistent with R, then $(\langle P,\ \phi\rangle, \langle Q,\ \psi\rangle, R) \in \mathcal{R}^{\varrho}$, thus $(\langle P,\ \phi\rangle, \langle Q,\ \psi\rangle, R) \in \mathcal{R}^{[\![\varrho]\!]}$.

2. Assume $(\langle P_1,\ \phi_1\rangle, \langle Q_1,\ \psi_1\rangle, R_1) \in \mathcal{R}^{[\![\varrho]\!]}$, with $\varrho \models \phi_1$ and $\varrho \models \psi_1$. Then we have $(\langle P_1,\ \phi_1\rangle, \langle Q_1,\ \psi_1\rangle, R_1) \in \mathcal{R}^{\varrho'}$, for some $\varrho' \equiv \varrho$, with $\varrho' \models \phi_1$ and $\varrho' \models \psi_1$. Therefore, if $\langle P_1,\ \phi_1\rangle \xrightarrow{\alpha} \langle P_2,\ \phi_2\rangle$ for $\alpha =\ \overline{c_{id}}(t)$, $\delta^c_{id}(t)$, $\overline{\delta^c_{id}}(t)$, signv$_{id}$ or τ, and $\varrho \models \phi_2$, thus $\varrho' \models \phi_2$, then there is a transition $\langle Q_1,\ \psi_1\rangle \xrightarrow{\alpha'} \langle Q_2,\ \psi_2\rangle$, for some $\alpha' = \alpha[y_1/x_1]\ldots[y_n/x_n]$ with $(x_i, y_i) \in R_1$, such that $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1) \in \mathcal{R}^{\varrho'}$ and $\varrho' \models \psi_2$. Therefore $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1) \in \mathcal{R}^{[\![\varrho']\!]} = \mathcal{R}^{[\![\varrho]\!]}$ and $\varrho \models \psi_2$. The case where $\langle Q_1,\ \psi_1\rangle \xrightarrow{\alpha} \langle Q_2,\ \psi_2\rangle$ is similar.

   Now assume that $\langle P_1,\ \phi_1\rangle \xrightarrow{\beta} \langle P_2,\ \phi_2\rangle$, for $\beta =\ c_{id}(x)$, f_{id}, split$_{id}$ or dec$_{id}$, and let ϱ_1 be a valuation consistent with $R_1[x]$ and such that $\varrho_1 \models \phi_2$. There is a transition $\langle Q_1,\ \psi_1\rangle \xrightarrow{\beta'} \langle Q_2,\ \psi_2\rangle$, for some $\beta' = \beta[y/x]$, such that $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1[\![(x,y)]\!]) \in \mathcal{R}^{\varrho_1[x/y]}$ and $\varrho_1[x/y] \models \psi_2$. Moreover, since $\mathcal{R}^{\varrho_1[x/y]} \subseteq \mathcal{R}^{[\![\varrho_1[x/y]]\!]}$, we see that $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1[\![(x,y)]\!]) \in \mathcal{R}^{[\![\varrho_1[x/y]]\!]}$. The case where $\langle Q_1,\ \psi_1\rangle \xrightarrow{\beta} \langle Q_2,\ \psi_2\rangle$ is similar.

Conversely, assume that $\langle P,\ \phi\rangle \simeq_{\mathrm{s}} \langle Q,\ \psi\rangle$, and let $\mathcal{R} = \{\mathcal{R}^{[\![\varrho]\!]}\}_{\varrho}$ be a symbolic bisimulation with respect to some full relation R between fv(ϕ) and fv(ψ). Consider the family $\mathcal{R}' = \{\mathcal{R}^{\varrho}\}_{\varrho}$ where $\mathcal{R}^{\varrho} = \mathcal{R}^{[\![\varrho]\!]}$. We see that $\mathcal{R}'$ is a bisimulation with respect to R, thus $\langle P,\ \phi\rangle \simeq \langle Q,\ \psi\rangle$. Indeed, given a valuation ϱ, we show that the relation $\mathcal{R}^{\varrho} = \mathcal{R}^{[\![\varrho]\!]}$ satisfies the conditions from Definition 6.

1. If $\varrho \models \phi$, $\varrho \models \psi$, and ϱ is consistent with R, then $(\langle P,\ \phi\rangle, \langle Q,\ \psi\rangle, R) \in \mathcal{R}^{[\![\varrho]\!]}$, thus $(\langle P,\ \phi\rangle, \langle Q,\ \psi\rangle, R) \in \mathcal{R}^{\varrho}$.

2. Assume $(\langle P_1,\ \phi_1\rangle, \langle Q_1,\ \psi_1\rangle, R_1) \in \mathcal{R}^{\varrho}$ with $\varrho \models \phi_1$ and $\varrho \models \psi_1$. Then, we have $(\langle P_1,\ \phi_1\rangle, \langle Q_1,\ \psi_1\rangle, R_1) \in \mathcal{R}^{[\![\varrho]\!]}$. Therefore, if $\langle P_1,\ \phi_1\rangle \xrightarrow{\alpha} \langle P_2,\ \phi_2\rangle$ for $\alpha =\ \overline{c_{id}}(t)$, $\delta^c_{id}(t)$, $\overline{\delta^c_{id}}(t)$, signv$_{id}$ or τ, and $\varrho \models \phi_2$, then there is a transition $\langle Q_1,\ \psi_1\rangle \xrightarrow{\alpha'} \langle Q_2,\ \psi_2\rangle$, for some $\alpha' = \alpha[y_1/x_1]\ldots[y_n/x_n]$ with $(x_i, y_i) \in R_1$, such that $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1) \in \mathcal{R}^{[\![\varrho]\!]}$ and $\varrho \models \psi_2$. Therefore $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1) \in \mathcal{R}^{\varrho}$. The case where $\langle Q_1,\ \psi_1\rangle \xrightarrow{\alpha} \langle Q_2,\ \psi_2\rangle$ is similar.

   Now assume that $\langle P_1,\ \phi_1\rangle \xrightarrow{\beta} \langle P_2,\ \phi_2\rangle$, for $\beta =\ c_{id}(x)$, f_{id}, split$_{id}$ or dec$_{id}$, and let ϱ_1 be a valuation consistent with $R[x]$ and such that $\varrho_1 \models \phi_2$. Therefore, there is a transition $\langle Q_1,\ \psi_1\rangle \xrightarrow{\beta'} \langle Q_2,\ \psi_2\rangle$, for some $\beta' = \beta[y/x]$, such that $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1[\![(x,y)]\!]) \in \mathcal{R}^{[\![\varrho_1[x/y]]\!]}$ and $\varrho_1[x/y] \models \psi_2$. Moreover, since $\mathcal{R}^{\varrho_1[x/y]} = \mathcal{R}^{[\![\varrho_1[x/y]]\!]}$, we have $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1[\![(x,y)]\!]) \in \mathcal{R}^{\varrho_1[x/y]}$. The case where $\langle Q_1,\ \psi_1\rangle \xrightarrow{\beta} \langle Q_2,\ \psi_2\rangle$ is similar. $\square$

Theorem 12 allows us to construct bisimulation between constrained processes using only finitely many valuations (one from each equivalence class $[\![\varrho]\!]$). Therefore, it gives us a finite proof method for verifying bisimilarity between any two constrained processes.

5.3 An Example

Example 5. Consider processes A and B defined as follows:

$$A ::=\ c(x_1).A', \quad A' ::= c(x_2).A \quad \text{and} \quad B ::=\ c(y).B$$

with $id_A = id_B = id$. The symbolic semantics of $\langle A,\ \mathbf{1}\rangle$ and $\langle B,\ \mathbf{1}\rangle$ are given in Fig. 5. We show that the constrained processes $\langle A,\ \mathbf{1}\rangle$ and $\langle B,\ \mathbf{1}\rangle$ are bisimilar. From Theorem 12, it is enough to construct a symbolic bisimulation between $\langle A,\ \mathbf{1}\rangle$ and $\langle B,\ \mathbf{1}\rangle$ with respect to the empty relation $R = \emptyset$.

$$\begin{tikzcd}
\langle A,\ \mathbf{1}\rangle \arrow[r, "c_{id}(x_1)"] & \langle A',\ \mathcal{M}(x_1)\rangle \arrow[r, "c_{id}(x_2)"] & \langle A,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle \arrow[d, shift right, "c_{id}(x_1)"'] \\
& & \langle A',\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle \arrow[u, shift right, "c_{id}(x_2)"']
\end{tikzcd}$$

$$\begin{tikzcd}
\langle B,\ \mathbf{1}\rangle \arrow[r, "c_{id}(y)"] & \langle B,\ \mathcal{M}(y)\rangle \arrow[loop right, "c_{id}(y)"]
\end{tikzcd}$$

Figure 5: *Symbolic Semantics of* $\langle A,\ \mathbf{1}\rangle$ *and* $\langle B,\ \mathbf{1}\rangle$*.*

First we consider the sets of formulas occurring in $\mathcal{D}(\langle A,\ \phi\rangle)$ and $\mathcal{D}(\langle B,\ \psi\rangle)$:

$$\{\mathbf{1},\ \mathcal{M}(x_1),\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\} \quad \text{and} \quad \{\mathbf{1},\ \mathcal{M}(y)\}.$$

Thus, we only need to consider sets of variables $\{x_1, x_2\}$ and $\{y\}$. Since these formulas are satisfied by every valuation, there are five equivalence classes with respect to $\equiv$, namely those that cover all the possibilities of the variables having identical values (resp. different values):

$$\begin{aligned}
\llbracket\varrho_1\rrbracket &= \{\varrho' \mid \varrho'(x_1) \neq \varrho'(x_2),\ \varrho'(x_1) \neq \varrho'(y) \text{ and } \varrho'(x_2) \neq \varrho'(y)\},\\
\llbracket\varrho_2\rrbracket &= \{\varrho' \mid \varrho'(x_1) = \varrho'(x_2) \text{ and } \varrho'(x_1) \neq \varrho'(y)\},\\
\llbracket\varrho_3\rrbracket &= \{\varrho' \mid \varrho'(x_1) = \varrho'(y) \text{ and } \varrho'(x_1) \neq \varrho'(x_2)\},\\
\llbracket\varrho_4\rrbracket &= \{\varrho' \mid \varrho'(x_1) \neq \varrho'(x_2) \text{ and } \varrho'(x_2) = \varrho'(y)\}, \text{ and}\\
\llbracket\varrho_5\rrbracket &= \{\varrho' \mid \varrho'(x_1) = \varrho'(x_2) = \varrho'(y)\}.
\end{aligned}$$

Note that we have $\llbracket\varrho_i[x_1/y]\rrbracket = \llbracket\varrho_i[y/x_1]\rrbracket = \llbracket\varrho_3\rrbracket$ for $i = 1, 3$, $\llbracket\varrho_i[x_1/y]\rrbracket = \llbracket\varrho_i[y/x_1]\rrbracket = \llbracket\varrho_5\rrbracket$ for $i = 2, 4, 5$, $\llbracket\varrho_i[x_2/y]\rrbracket = \llbracket\varrho_i[y/x_2]\rrbracket = \llbracket\varrho_4\rrbracket$ for $i = 1, 4$, and $\llbracket\varrho_i[x_2/y]\rrbracket = \llbracket\varrho_i[y/x_2]\rrbracket = \llbracket\varrho_5\rrbracket$ for $i = 2, 3, 5$.

For each equivalence class $\llbracket\varrho_i\rrbracket$, the steps for constructing the relation $\mathcal{R}^{\llbracket\varrho_i\rrbracket}$ are illustrated in Fig. 6. Thus, for each class $\llbracket\varrho_i\rrbracket$, we need go to through the

$\mathcal{R}^{\llbracket\varrho_1\rrbracket}$

$\mathcal{D}(\langle A,\ \mathbf{1}\rangle)$	$\mathcal{D}(\langle B,\ \mathbf{1}\rangle)$	R
$\langle A,\ \mathbf{1}\rangle$	$\langle B,\ \mathbf{1}\rangle)$	$\emptyset$

`go to relations` $\mathcal{R}^{\llbracket\varrho_3\rrbracket}$ and $\mathcal{R}^{\llbracket\varrho_5\rrbracket}$

$\mathcal{R}^{\llbracket\varrho_2\rrbracket}$

$\mathcal{D}(\langle A,\ \mathbf{1}\rangle)$	$\mathcal{D}(\langle B,\ \mathbf{1}\rangle)$	R
$\langle A,\ \mathbf{1}\rangle$	$\langle B,\ \mathbf{1}\rangle)$	$\emptyset$

`go to relations` $\mathcal{R}^{\llbracket\varrho_3\rrbracket}$ and $\mathcal{R}^{\llbracket\varrho_5\rrbracket}$

$\mathcal{R}^{\llbracket\varrho_3\rrbracket}$

$\mathcal{D}(\langle A,\ \mathbf{1}\rangle)$	$\mathcal{D}(\langle B,\ \mathbf{1}\rangle)$	R
$\langle A,\ \mathbf{1}\rangle$	$\langle B,\ \mathbf{1}\rangle)$	$\emptyset$
$\langle A',\ \mathcal{M}(x_1)\rangle$	$\langle B,\ \mathcal{M}(y))\rangle$	$\{(x_1, y)\}$
$\langle A',\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle$	$\langle B,\ \mathcal{M}(y))\rangle$	$\{(x_1, y)\}$

`go to relation` $\mathcal{R}^{\llbracket\varrho_4\rrbracket}$ and $\mathcal{R}^{\llbracket\varrho_5\rrbracket}$

$\mathcal{R}^{\llbracket\varrho_4\rrbracket}$

$\mathcal{D}(\langle A,\ \mathbf{1}\rangle)$	$\mathcal{D}(\langle B,\ \mathbf{1}\rangle)$	R
$\langle A,\ \mathbf{1}\rangle$	$\langle B,\ \mathbf{1}\rangle)$	$\emptyset$
$\langle A,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle$	$\langle B,\ \mathcal{M}(y)\rangle$	$\{(x_2, y)\}$

`go to relations` $\mathcal{R}^{\llbracket\varrho_3\rrbracket}$ and $\mathcal{R}^{\llbracket\varrho_5\rrbracket}$

$\mathcal{R}^{\llbracket\varrho_5\rrbracket}$

$\mathcal{D}(\langle A,\ \mathbf{1}\rangle)$	$\mathcal{D}(\langle B,\ \mathbf{1}\rangle)$	R
$\langle A,\ \mathbf{1}\rangle$	$\langle B,\ \mathbf{1}\rangle)$	$\emptyset$
$\langle A',\ \mathcal{M}(x_1)\rangle$	$\langle B,\ \mathcal{M}(y)\rangle$	$\{(x_1, y)\}$
$\langle A,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle$	$\langle B,\ \mathcal{M}(y)\rangle$	$\{(x_2, y)\}$
$\langle A',\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle$	$\langle B,\ \mathcal{M}(y)\rangle$	$\{(x_1, y)\}$
$\langle A,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle$	$\langle B,\ \mathcal{M}(y)\rangle$	$\{(x_2, y)\}$

`halt: already processed (success)`

Figure 6: *Symbolic Bisimulation of* $\langle A,\ \mathbf{1}\rangle$ *and* $\langle B,\ \mathbf{1}\rangle$.

conditions imposed by Definition 11 and determine which pairs of derivatives from the set $\mathcal{D}(\langle A,\ \mathbf{1}\rangle) \times \mathcal{D}(\langle B,\ \mathbf{1}\rangle)$ belong to $\mathcal{R}^{[\![\varrho_i]\!]}$, along with their finite relation R.

- For the class $[\![\varrho_1]\!]$, first we add $(\langle A,\ \mathbf{1}\rangle, \langle B,\ \mathbf{1}\rangle, \emptyset)$ to $\mathcal{R}^{[\![\varrho_1]\!]}$. Then, since the constrained process $\langle A,\ \mathbf{1}\rangle$ can execute the input action $c_{id}(x_1)$, we need to verify whether $\langle B,\ \mathbf{1}\rangle$ can simulate this input action with its own $c_{id}(y)$, where x_1 is mapped to y. But since both ϱ_3 and ϱ_5 are consistent with the relation $\{(x_1, y)\}$, we need to add $(\langle A',\ \mathcal{M}(x_1)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_1, y)\})$ to both $\mathcal{R}^{[\![\varrho_3]\!]}$ and $\mathcal{R}^{[\![\varrho_5]\!]}$. Similarly, simulating the input action $c_{id}(y)$ from $\langle B,\ \mathbf{1}\rangle$ by the input action $c_{id}(x_1)$ from $\langle A,\ \mathbf{1}\rangle$ requires the same additions on $\mathcal{R}^{[\![\varrho_3]\!]}$ and $\mathcal{R}^{[\![\varrho_5]\!]}$.
- For the class $[\![\varrho_2]\!]$, first we add $(\langle A,\ \mathbf{1}\rangle, \langle B,\ \mathbf{1}\rangle, \emptyset)$ to $\mathcal{R}^{[\![\varrho_2]\!]}$. As for the class $[\![\varrho_1]\!]$, we then need to add $(\langle A',\ \mathcal{M}(x_1)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_1, y)\})$ to both $\mathcal{R}^{[\![\varrho_3]\!]}$ (already added) and $\mathcal{R}^{[\![\varrho_5]\!]}$ (already added).
- For the class $[\![\varrho_3]\!]$, first we add $(\langle A,\ \mathbf{1}\rangle, \langle B,\ \mathbf{1}\rangle, \emptyset)$ to $\mathcal{R}^{[\![\varrho_3]\!]}$. As above, we also need to add $(\langle A',\ \mathcal{M}(x_1)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_1, y)\})$ to both $\mathcal{R}^{[\![\varrho_3]\!]}$ (already added) and $\mathcal{R}^{[\![\varrho_5]\!]}$ (already added). The addition of the tuplet $(\langle A',\ \mathcal{M}(x_1)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_1, y)\})$ to $\mathcal{R}^{[\![\varrho_3]\!]}$ (imposed by the other relations) requires the bisimulation of the input action $c_{id}(x_2)$ form $\langle A',\ \mathcal{M}(x_1)\rangle$ by the input action $c_{id}(y)$ $\langle B,\ \mathcal{M}(y)\rangle$, through the relation $\{(x_2, y)\}$. Hence, since the valuations ϱ_4 and ϱ_5 are consistent with the relation $\{(x_2, y)\}$, we need to add $(\langle A,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_2, y)\})$ to both $\mathcal{R}^{[\![\varrho_4]\!]}$ and $\mathcal{R}^{[\![\varrho_5]\!]}$. Moreover, the addition of $(\langle A',\ \mathcal{M}(x_1)\wedge\mathcal{M}(x_2)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_1, y)\})$ to $\mathcal{R}^{[\![\varrho_3]\!]}$ (imposed by the relation $\mathcal{R}^{[\![\varrho_4]\!]}$) requires the bisimulation of the input action $c_{id}(x_2)$ from $\langle A',\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle$ by the input action $c_{id}(y)$ from $\langle B,\ \mathcal{M}(y)\rangle$, through the relation $\{(x_2, y)\}$. Hence, we need to add $(\langle A,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_2, y)\})$ to both $\mathcal{R}^{[\![\varrho_4]\!]}$ (already added) and $\mathcal{R}^{[\![\varrho_5]\!]}$ (already added).
- For the class $[\![\varrho_4]\!]$, first we add $(\langle A,\ \mathbf{1}\rangle, \langle B,\ \mathbf{1}\rangle, \emptyset)$ to $\mathcal{R}^{[\![\varrho_4]\!]}$. As above, we then need to add $(\langle A',\ \mathcal{M}(x_1)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_1, y)\})$ to both $\mathcal{R}^{[\![\varrho_3]\!]}$ (already added) and $\mathcal{R}^{[\![\varrho_5]\!]}$ (already added). The addition of the tuplet $(\langle A,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_2, y)\})$ to $\mathcal{R}^{[\![\varrho_4]\!]}$ (imposed by the relation $\mathcal{R}^{[\![\varrho_3]\!]}$) requires the bisimulation of the input action $c_{id}(x_1)$ from $\langle A,\ \mathcal{M}(x_1)\wedge\mathcal{M}(x_2)\rangle$ by the input action $c_{id}(y)$ from $\langle B,\ \mathcal{M}(y)\rangle$, through the relation $\{(x_1, y)\}$. Hence, since ϱ_3 and ϱ_5 are consistent with the relation $\{(x_1, y)\}$, we need to add $(\langle A',\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_1, y)\})$ to both $\mathcal{R}^{[\![\varrho_3]\!]}$ and $\mathcal{R}^{[\![\varrho_5]\!]}$.
- For the class $[\![\varrho_5]\!]$, first we add $(\langle A,\ \mathbf{1}\rangle, \langle B,\ \mathbf{1}\rangle, \emptyset)$ to $\mathcal{R}^{[\![\varrho_5]\!]}$. As above, we then need to add $(\langle A',\ \mathcal{M}(x_1)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_1, y)\})$ to both $\mathcal{R}^{[\![\varrho_3]\!]}$ (already added) and $\mathcal{R}^{[\![\varrho_5]\!]}$ (already added). This last additon to $\mathcal{R}^{[\![\varrho_5]\!]}$ requires to show the bisimulation of the input action $c_{id}(x_2)$ from $\langle A',\ \mathcal{M}(x_1)\rangle$ by the

input action $c_{id}(y)$ from $\langle B,\ \mathcal{M}(y)\rangle$, through the relation $\{(x_2, y)\}$. Since the valuations ϱ_4 and ϱ_5 are consistent with the relation $\{(x_2, y)\}$, we need to add $(\langle A,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_2, y)\})$ to both $\mathcal{R}^{[\![\varrho_4]\!]}$ (already added) and $\mathcal{R}^{[\![\varrho_5]\!]}$ (already added). This last addition to $\mathcal{R}^{[\![\varrho_5]\!]}$ then leads us to look at constrained processes $\langle A,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle$ and $\langle B,\ \mathcal{M}(y)\rangle$, hence their respective input actions $c_{id}(x_1)$ and $c_{id}(y)$ through the relation $\{(x_1, y)\}$. But since ϱ_3 and ϱ_5 are consistent with the relation $\{(x_1, y)\}$, we need to add the tuplet $(\langle A',\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_1, y)\}$ to both $\mathcal{R}^{[\![\varrho_3]\!]}$ (already added) and $\mathcal{R}^{[\![\varrho_5]\!]}$ (already added).

The algorithm halts (with success) when every triplet $(\langle P,\ \phi\rangle, \langle Q,\ \psi\rangle, R)$ has been processed, without any contradiction, for every relation $\mathcal{R}^{[\![\varrho]\!]}$ such that ϱ is consistent with R. In our example, the construction of the bisimulation halts since every triplet has been added whenever is was required. Therefore, we obtain the symbolic bisimulation $\mathcal{R} = \bigcup_{i=1}^{5} \{\mathcal{R}^{[\![\varrho_i]\!]}\}$ where

$$
\begin{aligned}
\mathcal{R}^{[\![\varrho_1]\!]} &= \{\ (\langle A,\ \mathbf{1}\rangle, \langle B,\ \mathbf{1}\rangle, \emptyset)\ \} \\
\mathcal{R}^{[\![\varrho_2]\!]} &= \{\ (\langle A,\ \mathbf{1}\rangle, \langle B,\ \mathbf{1}\rangle, \emptyset)\ \} \\
\mathcal{R}^{[\![\varrho_3]\!]} &= \{\ (\langle A,\ \mathbf{1}\rangle, \langle B,\ \mathbf{1}\rangle, \emptyset),\ (\langle A',\ \mathcal{M}(x_1)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_1, y)\}), \\
&\qquad (\langle A',\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_1, y)\})\ \} \\
\mathcal{R}^{[\![\varrho_4]\!]} &= \{\ (\langle A,\ \mathbf{1}\rangle, \langle B,\ \mathbf{1}\rangle, \emptyset),\ (\langle A,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_2, y)\})\ \} \\
\mathcal{R}^{[\![\varrho_5]\!]} &= \{\ (\langle A,\ \mathbf{1}\rangle, \langle B,\ \mathbf{1}\rangle, \emptyset),\ (\langle A',\ \mathcal{M}(x_1)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_1, y)\}), \\
&\qquad (\langle A,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_2, y)\}), \\
&\qquad (\langle A',\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_1, y)\})\ \}.
\end{aligned}
$$

6 Security Protocols Analysis

In this section, we give a quick overview on how to use the symbolic framework introduced in this paper to analyse security protocols. First, we show how to specify a protocol using the concept of constrained process, and, secondly, we show how to specify security properties using equivalence-checking methods.

6.1 Protocol Specification

Process algebra SPPA along with the notion of constrained process offer a useful framework for the specification of security protocols, including cryptographic protocols. Starting from a protocol P written in a notation *à la Alice and Bob*, the main idea behind our specification approach is to specify each principal involved in P as disjoint constrained processes. For instance, a principal A is specified as the constrained process $\langle A,\ \phi_A\rangle$, where $A ::= (S_A, id_A)$, S_A is the initial SPPA agent of A, and ϕ_A is a formula characterising the principal's initial knowledge (specified as free variables within S_A). Note that a principal's initial knowledge (e.g. its private keys and the keys of other principals) are commonly implicitly specified within the initial agent S_A as specific messages $m \in \mathcal{M}$ or keys $k \in \mathcal{K}$. Thus, an initial agent S_A is generally closed (i.e. $\mathrm{fv}(S_A) = \emptyset$) and, in that case, we have $\phi_A ::= \mathbf{1}$. However, our symbolic framework also allows for the specification of these initial knowledge as symbolic values (i.e. free variables). For instance, if $\mathrm{fv}(S_A) = \{x\}$ and x stands for A's private key in S_A, then we put $\phi_A ::= \mathcal{K}(x)$.

Given specifications of the protocol's principals, let say $\langle A,\ \phi_A\rangle$, $\langle B,\ \phi_B\rangle$ and $\langle S,\ \phi_S\rangle$, then the whole protocol is specified as the constrained process $\langle P,\ \phi_P\rangle$ with $P ::= A \parallel B \parallel S$ and $\phi_P ::= \phi_A \wedge \phi_B \wedge \phi_S$.

The intruders, namely the principals attacking the security protocols, are specified similarly as the other principals. Hence, an intruder is specified as a constrained process $\langle E,\ \phi_E\rangle$, commonly called *enemy process*, where $E ::= (S_E, id_E)$, S_E is the initial SPPA agent of E (i.e. the SPPA agent specifying the intruder's attack) and ϕ_E is a formula characterising the intruder's initial knowledge (as above). From this notation, the protocol P being attacked by the enemy process E is then specified as the constrained process $\langle P_E,\ \phi_{P_E}\rangle$ with $P_E ::= P \parallel E$ and $\phi_{P_E} ::= \phi_P \wedge \phi_E$.

6.2 Equivalence-Checking

We achieve security protocols analysis through a verification method called *equivalence-checking*. The main idea is to verify whether the protocol always acts correctly within an hostile environment. Roughly speaking, given a protocol P, we need to verify if the protocol being attacked, specified as the constrained process $\langle P_E,\ \phi_{P_E}\rangle$, is equivalent to the protocol not being attacked, specified as the constrained process $\langle P,\ \phi_P\rangle$. The equivalence relation used to compared the two constrained processes is a relation based on bisimulation (Definition 6), called $\mathcal{O}$-*bisimulation*.

The concept of $\mathcal{O}$-bisimulation [19], called $\mathcal{O}$-congruence by Boudol [5], captures the notion of behavioural indistinguishability through an observation criterion $\mathcal{O}$. Given an observation criterion $\mathcal{O}$, we say that the constrained process $\langle P,\ \phi\rangle$ is $\mathcal{O}$-*bisimilar* to the constrained process $\langle Q,\ \psi\rangle$ whenever $\langle P/\mathcal{O},\ \phi\rangle \simeq \langle Q/\mathcal{O},\ \psi\rangle$. In that case, we write $P \simeq_{\mathcal{O}} Q$.

From the concept of $\mathcal{O}$-bisimulation, security properties are captured through different interpretations of an information flow property called *bisimulation-based non-deterministic admissible interference* (BNAI) [19]. In the following, we offer a quick overview of previously defined security property based on BNAI (see respective reference for further details).

Confidentiality [19]. Protocol $\langle P,\ \phi_P\rangle$ *preserves the confidentiality* if, for every enemy process E,

$$\forall_{\langle Q,\ \psi\rangle \in \mathcal{D}(\langle P_E,\ \phi_{P_E}\rangle)} \quad \langle Q \setminus \Gamma,\ \psi\rangle \ \simeq_{\mathcal{O}_E}\ \langle Q \setminus (\Gamma\ \cup\ Act_{\mathrm{secret}}),\ \psi\rangle$$

where $\mathcal{O}_E = \mathcal{O}_{Act_E}$ (see Section 3.2 for notation), Act_{secret} is the set of actions containing a secret message, and Γ is a set of downgrading actions containing every encrypting action, hashing action and signing action (hence the actions causing admissible declassification of information). This confidentiality property requires that no intruder can discriminate, in an inadmissible way, the protocol's behaviour and the behaviour of the protocol exchanging no confidential information.

Authenticity [19]. Protocol $\langle P,\ \phi_P\rangle$ *preserves the authenticity* if, for every enemy process E,

$$\forall_{\langle Q,\ \psi\rangle \in \mathcal{D}(\langle P_E,\ \phi_{P_E}\rangle)} \quad \langle Q \setminus \Gamma,\ \psi\rangle \ \simeq_{\mathcal{O}_{\mathrm{auth}}}\ \langle Q \setminus Act_E,\ \psi\rangle$$

where $\mathcal{O}_{\mathrm{auth}} = \mathcal{O}_{Act_{\mathrm{auth}}}$, the set $Act_{\mathrm{auth}} \subseteq Act$ contains actions describing critical states of a process (i.e. the actions that should not occur when the protocol is being attacked), and $\Gamma \subseteq Act_E$ is a set of admissible attacks containing intruder's actions corresponding to harmless interference (e.g. intruder receiving an invitation for a protocol run or initiating an honest protocol run). This authenticity property requires that no intruder can interfere in an inadmissible way with the protocol.

Denial of Service [14]. Protocol P is *robust against denial of service* if, for every enemy process E,

$$\forall_{\langle Q,\ \psi\rangle \in \mathcal{D}(\langle P_E,\ \phi_{P_E}\rangle)} \quad \langle Q \setminus \Gamma,\ \psi\rangle \ \simeq_{\mathcal{O}_{\mathrm{costly}}}\ \langle Q \setminus Act_E,\ \psi\rangle$$

where $\mathcal{O}_{\mathrm{costly}} = \mathcal{O}_{Act_{\mathrm{costly}}}$, Act_{costly} is the set of costly actions (i.e. actions requiring large amounts of resources and which could lead to resource exhaustion for some principal), and $\Gamma \subseteq Act_E$ is a set of admissible attacks (defined similarly as above). This denial of service property requires no causal dependency between enemy behaviours and costly actions (hence, potentially exhausting actions) of other principals.

7 Future Work and Related Work

This paper presents a symbolic framework for the analysis of security protocols. It is based on a message algebra that handles cryptographic primitives and a logic over this message algebra. The notion of constrained processes is then introduced as a value-passing process paired with a formula. Processes are defined through SPPA, a process algebra which allows for the specification of local function calls as visible actions. SPPA also gives, through marker actions, a clearer view of communication between principals. Generating functions for random numbers, fresh nonces and fresh keys, are introduced into SPPA's syntax in order to specify intruders generating fake addresses and fake messages. From SPPA symbolic semantics for constrained processes, we then establish a bisimulation equivalence. Apart from introducing a new symbolic approach, the major results of this paper are the decidability of every formula in our logic (Theorem 1), the finiteness of the symbolic operational semantics of any constrained process (Theorem 2), and the fact that the bisimulation equivalence between constrained processes corresponds to Milner's strong bisimulation between value-passing processes (Theorem 8). Another main result of this paper is a sound and complete proof method, called symbolic bisimulation, to check bisimilarity between constrained processes.

The main difference between our approach and Hennessy-Lin's [12] is the symbolic transition graph: in our symbolic transition graph (symbolic semantics), we assign to each state (process) a formula giving a precise description of the free variables involved in the process; Hennessy-Lin's symbolic framework requires considering the formula built from some path leading to a given state (process). Our symbolic framework was developed with security analysis in mind – it is then essential to have an accurate description of the symbolic values at a given state in order to properly analyse a security protocol in a computer system. Indeed, security protocol analysis often requires checking the effect of random values (e.g. nonces, fresh keys or fake messages) on certain principals of the protocol. In this context, our notion of constrained process allows us to explicitly view which such random value could lead, at a certain point of the protocol, to either a confidentiality leak or a masquerade (authentication attack). For instance, in denial of service analysis, we commonly need to verify whether a fake message sent by an intruder can cause the execution of a function requiring a large amount of resources (e.g. decryption or signature verification). In that case, one strategy based on constrained processes would be to verify, for every process following such costly action, the restriction imposed by the formula to the variable representing the fake message: if every fake message satisfies the formula, then we should conclude that the protocol can not detect fake protocol runs. If only few fake messages satisfy the formula, then we should conclude that the protocol is safe since most fake protocol runs initiated by an intruder will have been detected previously. A similar method, based on SPPA, for detecting denial of service vulnerabilities was introduced in a previous paper [14].

Other significant symbolic methods applied to security protocols were proposed by Boreale [3] and Fiore & Abadi [8]. Starting from a process algebra similar to spi-calculus, Boreale introduces a symbolic operational semantics based on unification. Boreale then gives a method carrying out trace analysis directly on the symbolic model. Also starting from a process algebra similar to spi-calculus, Fiore & Abadi propose a decision procedure for knowledge checking and a symbolic procedure for knowledge analysis. In future work, we plan to establish more complete relationships between these methods and ours. This task would require introducing constrained processes containing π-calculus and spi-calculus processes, and establishing a symbolic semantics for such constrained processes.

Acknowledgment:

Many thanks to prof. J. Mullins who helped me to write this paper, and to S. Hamadou for verifying its cryptographic soundness.

References

1. M. Abadi and A. D. Gordon. A bisimulation method for cryptographic protocols. *Nordic Journal of Computing*, 5(4):267–303, 1998.
2. M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1–70, January 1999.
3. M. Boreale. Symbolic trace analysis of cryptographic protocols. In *Proceedings of ICALP'01*, 2001.
4. M. Boreale, R. De Nicola, and R. Pugliese. Proof techniques for cryptographic processes. In *Logic in Computer Science*, pages 157–166, 1999.
5. G. Boudol. Notes on algebraic calculi of processes. In *Logic and Models of Concurrent Systems*, NATO ASI Series F-13, pages 261–303. Springer, 1985.
6. V. Cortier. Observational equivalence and trace equivalence in an extension of spi-calculus. application to cryptographic protocols analysis. Technical Report LSV-02-3, Lab. Specification and Verification, ENS de Cachan, Cachan, France, March 2002.
7. A. Durante, R. Focardi, and R. Gorrieri. CVS: A compiler for the analysis of cryptographic protocols. In *Proceedings of 12th IEEE Computer Security Foundations Workshop*, pages 203–212. IEEE Computer Society, June 1999.
8. M. Fiore and M. Abadi. Computing symbolic models for verifying cryptographic protocols. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop*, pages pp. 160–173, 2001.
9. R. Focardi, A. Ghelli, and R. Gorrieri. Using non interference for the analysis of security protocols. In H. Orman and C. Meadows, editors, *Proceedings of the DIMACS Workshop on Design and Formal Verification of Security Protocols*, Rutgers University, September 1997.
10. R. Focardi and R. Gorrieri. A classification of security properties for process algebras. *Journal of Computer Security*, 3(1):5–33, 1994/1995.
11. R. Focardi and F. Martinelli. A uniform approach for the definition of security properties. In *Proceedings of World Congress on Formal Methods (FM'99)*, volume 1708 of *LNCS*, pages 794–813. Springer, 1999.
12. M. Hennessy and H. Lin. Symbolic bisimulations. *Theoretical Computer Science*, 138:353–389, 1995.
13. C.A.R. Hoare. *Communicating sequential processes*. Prentice-Hall, 1985.
14. S. Lafrance and J. Mullins. An information flow method to detect denial of service vulnerabilities. In V. Dvorak, M. Sveda, C. Rattray, and J. W. Rozenblit, editors, *Formal Specifications of Computer-Based Systems*, volume 9, pages 1258–1260. Journal of Universal Computer Science, November 2003.

15. G. Lowe. Breaking and fixing the needham-schroeder public-key protocol using FDR. In *Proceedings of TACAS'96*, volume 1055 of *LNCS*, pages 147–166. Springer-Verlag, 1996.
16. R. Milner. *Communication and concurrency.* Prentice-Hall, 1989.
17. R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, parts I and II. *Information and Computation*, 100(1):1–77, September 1992.
18. J. Mullins. Nondeterministic admissible interference. *Journal of Universal Computer Science*, 6(11):1054–1070, 2000.
19. J. Mullins and S Lafrance. Bisimulation-based non-deterministic admissible interference with applications to the analysis of cryptographic protocols. *International Journal in Information and Software Technology*, pages 1–25, 2002.
20. S. Schneider. Security properties and CSP. In *IEEE Symposium on Security and Privacy*, pages 174–187, 1996.

A Decidability of Formulas

A formula ϕ is decidable whenever there is a finite algorithm allowing to verify, for every valuation ϱ, whether $\varrho \models \phi$. But since $\varrho \models \phi$ is equivalent to $\models \varrho(\phi)$, and $\varrho(\phi)$ is a closed formula, we see that, in order to prove the decidability of every formula from our logic, it is enough to show that every closed formula ϕ is decidable i.e., to give an algorithm deciding whether $\models \phi$. The first step toward proving this result consists in showing that every closed formula is equivalent to a quantifier-free (closed) formula.

Given a formula $\phi ::= (\exists_x\ \phi_1) \wedge \phi_2$ and a variable y that does not occur in ϕ_2, it is easily seen that ϕ is equivalent to the formula $\exists_y\ (\phi_1[y/x] \wedge \phi_2)$. Hence, we may assume that a closed formula ϕ is always given in its normal form i.e.,

$$\phi ::= \ \exists_{x_1} \ldots \exists_{x_n}\ (\phi_1 \wedge \ldots \wedge \phi_m)$$

where the ϕ_i are either predicates ($\phi_i = \mathcal{K}(t),\ \mathcal{I}(t),\ \mathcal{N}(t)$ or $\mathcal{M}(t)$) or equations ($\phi_i = (t == t')$) with $\mathrm{fv}(t), \mathrm{fv}(t') \subseteq \{x_1, \ldots, x_n\}$. (We assume that formulas $\mathbf{1}$ and $\mathbf{0}$ coincide with their normal form.) Moreover, from the definition of $\models (a == b)$ given in Section 2.2, we see that every equation $t == t'$ is equivalent to a finite conjunction of irreducible equations $x == t''$. Thus, we may assume that $\phi ::= \ \exists_{x_1} \ldots \exists_{x_n}\ (\phi_1 \wedge \ldots \wedge \phi_m)$ where the sub-formulas ϕ_i are either predicates or equations $x == t$ (with $x \in \{x_1, \ldots, x_n\}$ and $\mathrm{fv}(t) \subseteq \{x_1, \ldots, x_n\}$). For the following, we consider the family of closed formulas:

$$\begin{aligned}\mathcal{F} \ = \{ &\exists_{x_1} \ldots \exists_{x_n}\ (\phi_1 \wedge \ldots \wedge \phi_m) \mid \phi_i = \mathcal{K}(t) \text{ or } \phi_i = \mathcal{I}(t) \text{ or } \phi_i = \mathcal{N}(t) \\ &\text{ or } \phi_i = \mathcal{M}(t) \text{ or } \phi_i = (x == t),\ \text{for some } n, m \in \mathbb{N}, \\ &\text{for some } x \in \{x_1, \ldots, x_n\} \ \text{ and for some } t \text{ such that } \mathrm{fv}(t) \subseteq \{x_1, \ldots, x_n\}\} \\ &\cup \{\mathbf{0}, \mathbf{1}\}.\end{aligned}$$

Therefore, given a closed formula ϕ, we may always assume that $\phi \in \mathcal{F}$; otherwise, an equivalent formula $\phi' \in \mathcal{F}$ can be easily constructed from ϕ following the steps above.

Given a formula $\phi \in \mathcal{F}$, we can construct an equivalent quantifier-free formula as follows. We proceed by induction on the number of existential quantifiers in ϕ. The case where ϕ has no quantifiers is trivial.

Let $n \geq 0$ and assume that every formula in $\mathcal{F}$ with at most n existential quantifiers is equivalent to some quantifier-free formula in $\mathcal{F}$. Now consider some formula $\phi \in \mathcal{F}$, where

$$\phi \ ::= \ \exists_x \exists_{x_1} \ldots \exists_{x_n} \ (\phi_1 \wedge \phi_2 \wedge \ldots \wedge \phi_m).$$

First assume that one of the $\phi_i = (x == t)$ (let say $i = 1$). If $t = x$ (i.e. $\phi_1 = (x == x)$), then we may drop ϕ_1 and assume that $\phi \ ::= \ \exists_x \exists_{x_1} \ldots \exists_{x_n} \ (\phi_2 \wedge \ldots \wedge \phi_m)$. Otherwise, if x occurs in t, then ϕ is equivalent to $\mathbf{0}$, hence $\not\models \phi$, since we do not allow infinite messages. If x does not occur in t, then we see that ϕ is equivalent to the formula

$$\exists_{x_1} \ldots \exists_{x_n} \ (\phi_2[t/x] \wedge \ldots \wedge \phi_m[t/x])$$

which has one less quantifier than ϕ. Moreover, every sub-formula $\phi_i = (t == t')$ can be replaced by an equivalent conjunction of equations $(x'_1 == t_1) \wedge \ldots \wedge (x'_k == t_k)$ with $x'_j \in \{x_1, \ldots, x_n\}$. Since the obtained formula belongs to $\mathcal{F}$, the proof is resolved using the induction hypothesis: there is a quantifier-free formula $\phi' \in \mathcal{F}$ equivalent to the formula above, and therefore equivalent to ϕ. Furthermore, we see that every equation can be withdrawn from ϕ by repeating the steps presented above for each variable x'_j.

Now assume that none of the ϕ_i is an equation. Moreover, assume that x occurs only in $\phi_1, \ldots, \phi_k$ (for $k \leq m$). Since formulas $\exists_x \mathcal{I}(t)$, $\exists_x \mathcal{N}(t)$ and $\exists_x \mathcal{K}(t)$ can only be true whenever $t = x$, none of the other quantified variables $x_1, \ldots, x_m$ occurs in the predicates $\phi_1, \ldots, \phi_k$ (otherwise $\phi \Leftrightarrow \mathbf{0}$). The formula ϕ is therefore equivalent to

$$\exists_x (\phi_1 \wedge \ldots \wedge \phi_k) \ \wedge \ \exists_{x_1} \ldots \exists_{x_n} \ (\phi_{k+1} \wedge \ldots \wedge \phi_m)$$

where $\phi_i \in \{\mathcal{K}(x), \mathcal{I}(x), \mathcal{N}(x)\}$ (for $1 \leq i \leq k$). Hence, it is enough to find a quantifier-free formula ψ equivalent to $\exists_x (\phi_1 \wedge \ldots \wedge \phi_k)$; we take $\psi ::= \mathbf{1}$ whenever

- $\phi_i = \mathcal{I}(x)$, for every $i = 1, \ldots, k$; or
- $\phi_i = \mathcal{N}(x)$ or $\phi_i = \mathcal{K}(x)$, for every $i = 1, \ldots, k$.

Otherwise we take $\psi ::= \mathbf{0}$. Finally, it is straightforward to see that the resulting formula (either $\mathbf{0}$ or $\exists_{x_1} \ldots \exists_{x_n} \ (\phi_{k+1} \wedge \ldots \wedge \phi_m)$) still belongs to $\mathcal{F}$ and has at most n quantifiers. Thus, by the induction hypothesis, we can found an equivalent formula $\phi' \in \mathcal{F}$, which is also equivalent to ϕ.

For the next lemma, recall that every formula in $\mathcal{F}$ in closed, including the quantifier-free formulas.

Lemma 13. *Every quantifier-free formula in $\mathcal{F}$ is decidable.*

Proof. Let $\phi \in \mathcal{F}$ be any quantifier-free formula. If $\phi = \mathbf{1}$ or $\phi = \mathbf{0}$, then the statement is trivial. Now assume that $\phi = \phi_1 \wedge \ldots \wedge \phi_n$ with $n \geq 1$. Since ϕ is closed, every sub-formula ϕ_i is either a predicate $\mathcal{I}(a)$, $\mathcal{K}(a)$ or $\mathcal{N}(a)$ (every predicate $\mathcal{M}(a)$ can be replaced right away with $\mathbf{1}$), or an equation $a == a'$, for some messages $a, a' \in \mathcal{M}$. But, as we saw in Section 2.2, each predicate $\mathcal{I}(a)$, $\mathcal{K}(a)$ or $\mathcal{N}(a)$ is assumed to be decidable, and each equation $a == a'$ is also decidable by successive reductions. Hence, each sub-formulas ϕ_i may therefore be individually replaced by either $\mathbf{1}$ or $\mathbf{0}$. Any such conjunction of $\mathbf{1}$ and $\mathbf{0}$ is clearly decidable. $\square$

The proof of Theorem 1 follows from Lemma 13 and the fact that every closed formula is equivalent to a formula from $\mathcal{F}$.

B Operational Semantics of SPPA.

The operational semantics of SPPA is given in Fig. 7 and Fig. 8. It is a value-passing-based semantics defined only for closed processes i.e., processes P such that $\mathrm{fv}(P) = \emptyset$. Rules `Sum`, `Parallel`, `Protocol` and `Synchronisation` are assumed to be associative and commutative.

A process P' is a *derivative* of P if there is a computation $P \xrightarrow{\gamma} P'$ for some $\gamma \in Act^*$. We also consider the set of P's derivatives defined as follows:

$$\mathcal{D}(P) = \{P' \mid \exists_{\gamma \in Act^*}\ P \xrightarrow{\gamma} P'\}.$$

C Proof of the Finiteness of Symbolic Semantics

In this section, we show that, for any constrained process $\langle P,\ \phi\rangle$, the transition graph associated to $\langle P,\ \phi\rangle$ using SPPA's operational symbolic semantics is always finite. But in order to obtain this result, we first need to establish the following restriction on SPPA's syntax (often applied on other process algebras for similar purposes): we do not allow recursive definitions $P := P_1 \setminus L$, $P := P_1/\mathcal{O}$, $P := P_1|P_2$, or $P := P_1 \parallel P_2$ such that P occurs somewhere within either P_1's or P_2's definition. Hence, we assume that any recursive definition of some SPPA's agent or process P (i.e. where P is defined using a self reference P) never uses a restriction operator, nor an observation operator, nor a parallel composition operator, nor a protocol operator. Such recursive definitions often lead to "infinite" processes, i.e. SPPA processes with infinitely many derivatives. For instance, processes $P ::= (c(x).P) \setminus L$ and $P ::= P|P'$ are refrained, while processes $P :=$ `let` $x = f(t)$ `in` P and $P := c(x).P + P'$ are retained.

We say that the process P' is a *sub-process* of the constrained process $\langle P,\ \phi\rangle$ whenever there is some formula ϕ' such that $\langle P',\ \phi'\rangle \in \mathcal{D}(\langle P,\ \phi\rangle)$.

Output $$\frac{-}{\overline{c}(a).A \xrightarrow{\overline{c_{id_A}}(a)} A}$$

Input $$\frac{a\in\mathcal{M}}{c(x).A \xrightarrow{c_{id_A}(a)} A[a/x]}$$

Function $$\frac{f\in\mathcal{F},\ \models\phi_f(a) \text{ and } a'=f(a)}{\texttt{let } x=f(a) \texttt{ in } A \xrightarrow{f_{id_A}} A[a'/x]}$$

Generator $$\frac{new\in\mathcal{F} \text{ and } \models\phi_{new}(a)}{\texttt{let } x=new(-) \texttt{ in } A \xrightarrow{new_{id_A}} A[a/x]}$$

Split $$\frac{-}{\texttt{let } (x,y)=(a,a') \texttt{ in } A \xrightarrow{\text{split}_{id_A}} A[a/x][a'/y]}$$

Decryption $$\frac{-}{\texttt{case } \{a\}_k \texttt{ of } \{x\}_k \texttt{ in } A \xrightarrow{\text{dec}_{id_A}} A[a/x]}$$

Signature-Verif $$\frac{-}{\texttt{case } [a]_k \texttt{ of } [a]_k \texttt{ in } A \xrightarrow{\text{signv}_{id_A}} A}$$

Figure 7: *Semantics of SPPA processes.*

Lemma 14. *Every constrained process $\langle P,\ \phi\rangle$ has finitely many sub-processes i.e., the set $\{P' \mid \langle P',\ \phi'\rangle \in \mathcal{D}(\langle P,\ \phi\rangle)$ for some $\phi'\}$ is finite.*

Proof. The proof follows from the fact that SPPA's symbolic semantics rules (Fig. 1 and Fig. 2) never alter the initial definition of P (and its sub-processes), neither through substitution $P'[t/x]$ or variable renaming. Hence, any sub-process P' occurring in $\mathcal{D}(\langle P,\ \phi\rangle)$ must be syntactically identical to its initial definition within P. We may therefore conclude that the cardinality of the set $\{P' \mid \langle P',\ \phi'\rangle \in \mathcal{D}(\langle P,\ \phi\rangle)$ for some $\phi'\}$ is at most $N_u + 2N_b$, where N_u is the number of unary SPPA operators (output, input, function call, match, restriction, etc.) used in the syntactical definition of P, and N_b is the number of binary operators (sum, parallel composition, etc.) used in the syntactical definition of P. □

It follows from Lemma 14 that, for any constrained process $\langle P,\ \phi\rangle$, there are only finitely many variables occurring in P and its sub-processes. Moreover, these variables are exactly the ones used within P's syntactical definition. Let $\{x_1, \ldots, x_n\}$ be the finite set containing those variables. We may also conclude from observing the semantics rules that $\text{fv}(\phi') \subseteq \{x_1, \ldots, x_n\}$ for any formula $\phi' \in \Phi$, where $\Phi = \{\phi' \mid \langle P',\ \phi'\rangle \in \mathcal{D}(\langle P,\ \phi\rangle)$ for some $P'\}$. Thus, any variable

Match $\dfrac{A \xrightarrow{\alpha} A'}{[a{=}a]\ A \xrightarrow{\alpha} A'}$

Sum $\dfrac{P \xrightarrow{\alpha} P'}{P{+}Q \xrightarrow{\alpha} P'}$

Parallel $\dfrac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}$

Protocol $\dfrac{P \xrightarrow{\alpha} P' \text{ and } \alpha{\notin}C}{P\|Q \xrightarrow{\alpha} P'\|Q}$

Synchronisation $\dfrac{P \xrightarrow{\overline{c_{id}}(a)} P' \quad \text{and} \quad Q \xrightarrow{c_{id'}(a)} Q'}{P\|Q \xrightarrow{\overline{\delta^c_{id}}(a)} P'\|Q \xrightarrow{\delta^c_{id'}(a)} P'\|Q'}$

Restriction $\dfrac{P \xrightarrow{\alpha} P' \text{ and } \alpha{\notin}L}{P\backslash L \xrightarrow{\alpha} P'\backslash L}$

Observation $\dfrac{P \xrightarrow{\gamma} P' \text{ and } \gamma \in \mathcal{O}^{-1}(\alpha)}{P/\mathcal{O} \xrightarrow{\alpha} P'/\mathcal{O}}$

Figure 8: *Semantics of SPPA processes.*

occurring in some derivative $\langle P',\ \phi'\rangle \in \mathcal{D}(\langle P,\ \phi\rangle)$ (either in P' or in ϕ') must also occur somewhere in the initial definition of P (or its sub-processes).

A computation $\langle P,\ \phi\rangle \xrightarrow{\gamma} \langle P',\ \phi'\rangle$ is said to be *minimal* whenever no constrained process appears more than once during the computation, including constrained processes $\langle P,\ \phi\rangle$ and $\langle P',\ \phi'\rangle$. Hence, neither $\langle P,\ \phi\rangle$ nor $\langle P',\ \phi'\rangle$ may appear within the minimal computation, except at the extremities, although we allow them to be the same constrained process. Any computation $\langle P,\ \phi\rangle \xrightarrow{\gamma} \langle P',\ \phi'\rangle$ may clearly be reduced to a minimal sub-computation $\langle P,\ \phi\rangle \xrightarrow{\gamma'} \langle P',\ \phi'\rangle$ (where γ' is a sub-sequence of γ) by taking out any loop within the computation of γ.

Lemma 15. *For every constrained process $\langle P,\ \phi\rangle$, there are only finitely many minimal computations $\langle P,\ \phi\rangle \xrightarrow{\gamma} \langle P,\ \phi'\rangle$.*

Proof. First, we see that any constrained process $\langle P,\ \phi\rangle$ as finitely many transitions emanating from it i.e., the set

$$\{\alpha \in Act \mid \langle P,\ \phi\rangle \xrightarrow{\alpha} \langle Q,\ \psi\rangle \text{ for some } \langle Q,\ \psi\rangle\}$$

is finite. Indeed, we see from the semantics rules that the existence of a transition $\langle P,\ \phi\rangle \xrightarrow{\alpha} \langle Q,\ \psi\rangle$, along with the value of the action α, depends only on the process P and not on the formula ϕ (as long it is not equivalent to $\mathbf{0}$). Moreover, since $\langle P,\ \phi\rangle$ has only a finite number of sub-processes (by Lemma 14), the total number of actions α occurring within $\langle P,\ \phi\rangle$'s semantics must be finite.

We may therefore conclude that given any two constrained processes, there are finitely many minimal computations between them. Indeed, if N_p denotes the number of P's sub-processes and N_a denotes the number of actions occurring within $\langle P,\ \phi\rangle$'s semantics, then the number of minimal computations between any two constrained processes is at most $N_p! \cdot {N_p}^{N_a+1}$, where

- $N_p!$ is a bound on the number of possible sequences of sub-processes corresponding to some minimal computation between the two constrained processes (i.e. no sub-process may occur twice), and
- ${N_p}^{N_a+1}$ is a bound on the number of possible sequences of actions corresponding to some minimal computation between the two constrained processes.

In particular, there are only finitely many minimal computations between $\langle P,\ \phi\rangle$ and $\langle P,\ \phi'\rangle$. $\square$

Proof of Theorem 2. Let $\langle P,\ \phi\rangle$ be a constrained process and assume that it has infinitely many derivative i.e., $\mathcal{D}(\langle P,\ \phi\rangle)$ is infinite. In that case, and since the number of transitions emanating from some constrained process is always bounded, there is an infinite computation

$$\langle P,\ \phi\rangle \xrightarrow{\alpha} \langle P',\ \phi'\rangle \xrightarrow{\alpha'} \ldots \tag{1}$$

with pairwise different constrained processes (i.e. no constrained process occurs more than once during the computation). Let $\{x_1, \ldots, x_n\}$ be the set of variables occurring in the computation (1) (we saw above that this set must be finite). Since every formula ψ occurring in the computation (1) is such that $\mathrm{fv}(\psi) \subseteq \{x_1, \ldots, x_n\}$, we may assume that the infinite computation has a tail

$$\langle P_1,\ \phi_1\rangle \xrightarrow{\alpha_1} \langle P_2,\ \psi_2\rangle \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_{k-1}} \langle P_k,\ \psi_k\rangle \xrightarrow{\alpha_k} \ldots$$

such that $\mathrm{fv}(\phi_1) = \mathrm{fv}(\psi_k) = \{x_1, \ldots, x_n\}$, for $k \geq 2$. Moreover, since P has finitely many sub-processes (by Lemma 14), we may assume that process P_1 occurs infinitely often in this computation. Hence, we can write

$$\langle P_1,\ \phi_1\rangle \xrightarrow{\gamma_1} \langle P_1,\ \phi_2\rangle \xrightarrow{\gamma_2} \ldots \xrightarrow{\gamma_{k-1}} \langle P_1,\ \phi_k\rangle \xrightarrow{\gamma_k} \ldots \tag{2}$$

with $\gamma_k \in Act^*$ and $\mathrm{fv}(\phi_k) = \{x_1, \ldots, x_n\}$, for $k \geq 1$. We may also assume that each computation $\langle P_1,\ \phi_k\rangle \xrightarrow{\gamma_k} \langle P_1,\ \phi_{k+1}\rangle$ is minimal. Moreover, by Lemma 15, there are finitely many minimal computations between any $\langle P_1,\ \phi_k\rangle$ and $\langle P_1,\ \phi_{k'}\rangle$, thus there are finitely many different sequences of actions γ_k. Assume that these possible sequences of actions are $\gamma'_1, \gamma'_2, \ldots, \gamma'_m$, hence any γ_k from the computation (2) is such that $\gamma_k \in \{\gamma'_1, \gamma'_2, \ldots, \gamma'_m\}$. Furthermore, we can assume that each γ'_k occurs infinitely often within the computation (2). Otherwise, if one of the sequence of actions γ'_k occurs only a finite number of times within the infinite computation, then we consider the infinite computation obtained by cutting the computation (2) after the last occurrence of γ'_k; this computation contains no γ'_k.

From the proof of Lemma 15, we know that the fact that a constrained process $\langle P',\ \phi'\rangle$ may execute an action depends only on the definition of the process P' (as long ϕ' is not equivalent to $\mathbf{0}$). Hence, given any sequence of action γ'_k, every computation $\langle P_1,\ \phi_l\rangle \xrightarrow{\gamma'_k} \langle P_1,\ \phi_{l+1}\rangle$ transforms the formula ϕ_l to the formula ϕ_{l+1} following the exact same rule (for any l). Moreover, we see from the symbolic semantics rules that we must have

$$\phi_{l+1} ::= \ \exists_{x_1^{(k)},\ldots,x_{n_k}^{(k)}}(\phi_l \wedge \psi_k) \ \wedge \ \psi'_k$$

for some formulas ψ_k and ψ'_k (which do not depend on l), and where $\{x_1^{(k)}, \ldots, x_{n_k}^{(k)}\} \subseteq \{x_1, \ldots, x_n\}$. Also notice that the formula ϕ_{l+1} is equivalent to the formula

$$\exists_{y_1^{(k)},\ldots,y_{n_k}^{(k)}}(\phi_l[y_1^{(k)}/x_1^{(k)}]\ldots[y_{n_k}^{(k)}/x_{n_k}^{(k)}] \ \wedge \ \psi_k[y_1^{(k)}/x_1^{(k)}]\ldots[y_{n_k}^{(k)}/x_{n_k}^{(k)}] \ \wedge \ \psi'_k),$$

where the $y_i^{(k)}$ are new variables. For simplicity, we use the following notation:

- we write $\exists_{y_i^{(k)}}$ instead of $\exists_{y_1^{(k)},\ldots,y_{n_k}^{(k)}}$, and
- we write $\phi[y_i^{(k)}]$ instead of $\phi[y_1^{(k)}/x_1^{(k)}]\ldots[y_{n_k}^{(k)}/x_{n_k}^{(k)}]$.

Using this notation, we have

$$\phi_{l+1} \Leftrightarrow \exists_{y_i^{(k)}}(\phi_l[y_i^{(k)}] \ \wedge \ \psi_k[y_i^{(k)}] \ \wedge \ \psi'_k).$$

Now consider the family of formula mappings $\{\Gamma_k\}_{k=1}^m$ with

$$\Gamma_k : \ \phi \ \mapsto \ \exists_{y_i^{(k)}}(\phi[y_i^{(k)}] \ \wedge \ \psi_k[y_i^{(k)}] \ \wedge \ \psi'_k)$$

where the $y_i^{(k)}$ are always new variables i.e., any mapping Γ_k (for $1 \leq k \leq m$) never uses the same variables $y_1^{(k)}, \ldots, y_{n_k}^{(k)}$ twice. From our arguments above, we see that $\Gamma_k(\phi_l) \Leftrightarrow \phi_{l+1}$ for any $l \geq 1$ such that $\langle P_1,\ \phi_l\rangle \xrightarrow{\gamma'_k} \langle P_1,\ \phi_{l+1}\rangle$, and the sequence of formulas $\phi_1, \phi_2, \phi_3, \ldots$ from the computation (2) can therefore be written as

$$\phi_1,\ \Gamma_{k_1}(\phi_1),\ \Gamma_{k_2}(\Gamma_{k_1}(\phi_1)),\ \Gamma_{k_3}(\Gamma_{k_2}(\Gamma_{k_1}(\phi_1))), \ldots$$

with $\gamma_l = \gamma'_{k_l}$.

But each mapping Γ_k essentially does two things: on one hand Γ_k substitutes the variables $x_1^{(k)}, \ldots, x_{n_k}^{(k)}$ with some new variables $y_1^{(k)}, \ldots, y_{n_k}^{(k)}$, and, on the other hand Γ_k introduces (through conjunction) formulas ψ_k and ψ'_k. Since Γ_k always introduces new variables, we may assume that every formula ϕ_l from the computation (2) has the form $\exists_{y_1,\ldots,y_n}\psi$ where the $y_1, \ldots, y_n$ are variables $y_1^{(k)}, \ldots, y_{n_k}^{(k)}$ introduced during previous applications of the mappings Γ_k (for $1 \le k \le m$), and ψ is some quantifier-free formula. Moreover, the formula ψ has the form $\phi_1[y_i^{(k_1)}]\ldots[y_i^{(k_l)}] \;\wedge\; \psi'$, where the formula ψ' is a conjunction of formulas ψ_k and ψ'_k (for $1 \le k \le m$) on which substitutions $[y_i^{(k_1)}]\ldots[y_i^{(k_l)}]$ where applied. Although there will be infinitely many new variables $y_1^{(k)}, \ldots, y_{n_k}^{(k)}$ introduced during the infinite computation, we show that, at some point during the computation, the introduction of those new variables by some Γ_k, and their substitution with the variables $x_1^{(k)}, \ldots, x_{n_l}^{(k)}$, will create a formula ϕ_l equivalent to a previous formula $\phi_{l'}$.

First, we see that Γ_k's substitutions are applied at most once on the formulas within ψ'. Indeed, consecutive substitutions

$$\phi_1[y_1^{(k)}/x_1^{(k)}]\ldots[y_{n_k}^{(k)}/x_{n_k}^{(k)}][z_1^{(k)}/x_1^{(k)}]\ldots[z_{n_k}^{(k)}/x_{n_k}^{(k)}]$$

gives the formula $\phi_1[y_1^{(k)}/x_1^{(k)}]\ldots[y_{n_k}^{(k)}/x_{n_k}^{(k)}]$, hence $\phi_l[y_i^{(k)}]$. The same remark is true for any formula ψ_k and ψ'_k, for $1 \le k \le m$, and consequently for the formula ψ'. Since there are m mappings Γ_k, and therefore m sets of variables $\{x_1^{(k)}, \ldots, x_{n_k}^{(k)}\}$ to be substituted, there at most $m!$ non-equivalent composed substitutions of the form $[y_i^{(k_1)}]\ldots[y_i^{(k_l)}]$ that can be obtained by composing the Γ_k; two composed substitutions are equivalent whenever the order in which the sets of variables $\{x_1^{(k)}, \ldots, x_{n_k}^{(k)}\}$ are substituted is exactly the same (the names of the new variables $y_1^{(k)}, \ldots, y_{n_k}^{(k)}$ does not matter). Moreover, since each mapping Γ_k always introduces the same formulas ψ_k and ψ'_k, the number of possible formulas we can obtain from the ψ_k and ψ'_k, and any composed substitution over the sets of variables $\{x_1^{(k)}, \ldots, x_{n_k}^{(k)}\}$ is at most $2m(m!)$, thus finite. But since we assumed that every sequence of actions γ'_k occurs infinitely often in the computation (2), at some point, for any new variables $y_1^{(k)}, \ldots, y_{n_k}^{(k)}$ introduced by some mapping Γ_k (through a substitution of $x_1^{(k)}, \ldots, x_{n_k}^{(k)}$) there are previously introduced variables $z_1^{(k)}, \ldots, z_{n_k}^{(k)}$ which are present within the exact same equivalent composed substitutions applied to the exact same formulas ψ_l and ψ'_l. In that case, the new set of variables $y_1^{(k)}, \ldots, y_{n_k}^{(k)}$ can be replaced by the $z_1^{(k)}, \ldots, z_{n_k}^{(k)}$, and the obtained formula is equivalent to a previous formula ϕ_l. In fact, we can see that at some point in the infinite computation (2), more precisely when each γ'_k occurred at least $2m(m!)$ times, any new application of the mapping Γ_k gives a formula equivalent to a previous application of Γ_k. Thus, there are formulas

$\theta_1, \ldots, \theta_m$ such that $\phi_{l+1} \Leftrightarrow \theta_k$ whenever

$$\langle P_1,\ \phi_l \rangle \xrightarrow{\gamma'_k} \langle P_1,\ \phi_{l+1} \rangle$$

for every $l \geq K$ (for some $K \geq 1$ large enough). Hence, $\Gamma_k(\theta_{k'}) \Leftrightarrow \theta_k$. But this contradicts the fact that the formulas ϕ_l (from the computation (2)) are pairwise not equivalent, and therefore contradicts the existence of the infinite computation (2). Hence, the set $\mathcal{D}(\langle P,\ \phi \rangle)$ must be finite. □

Remark. The formulas θ_k from the previous proof are too large to be explicitly written down in this paper, but we can see that they have the following form

$$\theta_k ::= \exists_{y_1,\ldots,y_{n'}} (\ \phi_1[y_i^{(k_1)}] \ldots [y_i^{(k_m)}]\ \wedge\ \theta\ \wedge\ \psi_k[z_i^{(k)}]\ \wedge\ \psi'_k\ )$$

where $y_1, \ldots, y_{n'}$ are new variables introduced by the mappings Γ_k, with $\{k_1, \ldots, k_m\} = \{1, \ldots, m\}$ and $y_i^{(k_1)}, \ldots, y_i^{(k_m)}, z_i^{(k)} \in \{y_1, \ldots, y_{n'}\}$. The formula θ, present in every θ_k, is the formula obtained by considering the conjunction of every possible combination of composed substitutions $[y_i^{(k_1)}] \ldots [y_i^{(k_m)}]$ (for new variables $y_i^{(k)}$) applied on every formulas ψ_k and ψ'_k (for $k = 1, \ldots, m$). The formula θ may be a very large formula (although it is rather small in most practical cases) which turns out to be some sort of fixed point for every mapping Γ_k. Indeed, since θ contains every composed substitution applied on every formulas introduced by the mappings, then any new substitution introduced by some Γ_k will have no effect on θ. The family of formulas $\{\theta_k\}_{k=1}^m$ will therefore be such that $\Gamma_k(\theta_{k'}) \Leftrightarrow \theta_k$, for any k, k'.

D Proofs of Lemma 3, Lemma 4 and Lemma 5

Proof of Lemma 3. In order to shorten the proof, the two statements are proved simultaneously by induction on the structure of P. Depending on the statement to prove, we put either

$$Q' ::=\ P'[a_1/x_1] \ldots [a_n/x_n] \quad \text{or} \quad Q' ::=\ P'[a/x][a_1/x_1] \ldots [a_n/x_n].$$

The case where $P = \mathbf{0}$ is trivial. If $P = \overline{c}(t).P'$, or $P = c(x).P'$, or $P =$ `let` $x = f(t)$ `in` P', or $P =$ `let` $(x, y) = t$ `in` P', or $P =$ `case` t `of` $\{x\}_{t'}$ `in` P', or $P =$ `case` t `of` $[t'']_{t'}$ `in` P', then the conclusion follows from rules `Output`, `Input`, `Function`, `Generator`, `Split`, `Decryption` and `Signature-Verif`.

If $P = P_1 + P_2$, or $P = P_1 | P_2$, or $P = P_1 \parallel P_2$ (and α is not a marker action), then, from semantics rules `Sum`, `Parallel` and `Protocol`, we may assume that

$$P_1[a_1/x_1] \ldots [a_n/x_n] \xrightarrow{\alpha} P'_1[a_1/x_1] \ldots [a_n/x_n]$$

$$(\text{resp. } P_1[a_1/x_1] \ldots [a_n/x_n] \xrightarrow{\beta} P'_1[a/x][a_1/x_1] \ldots [a_n/x_n])$$

with either $P' ::= P'_1$, or $P' ::= P'_1|P_2$, or $P' ::= P'_1 \parallel P_2$. Thus, by the induction hypothesis,

$$\langle P_1,\ \phi\rangle \xrightarrow{\alpha'} \langle P'_1,\ \phi'\rangle \quad (\text{resp. } \langle P_1,\ \phi\rangle \xrightarrow{\beta'} \langle P'_1,\ \phi'\rangle).$$

Therefore, $\langle P,\ \phi\rangle \xrightarrow{\alpha'} \langle P',\ \phi'\rangle$ (resp. $\langle P,\ \phi\rangle \xrightarrow{\beta'} \langle P',\ \phi'\rangle$). If $P = P_1 \parallel P_2$ and α is a marker action, then we may assume that

$$P_1[a_1/x_1]\dots[a_n/x_n] \xrightarrow{\overline{c_{id_1}}(a)} P'_1[a_1/x_1]\dots[a_n/x_n]$$

and

$$P_2[a_1/x_1]\dots[a_n/x_n] \xrightarrow{c_{id_2}(a)} P'_2[a/x][a_1/x_1]\dots[a_n/x_n]$$

with $P' ::= P'_1 \parallel P_2$ or $P' ::= P'_1 \parallel P'_2$. By the induction hypothesis, we have $\langle P_1,\ \phi\rangle \xrightarrow{\overline{c_{id_1}}(t)} \langle P'_1,\ \phi'\rangle$ (with $a = t[a_1/x_1]\dots[a_n/x_n]$) and $\langle P_2,\ \phi\rangle \xrightarrow{c_{id_2}(x)} \langle P'_2,\ \phi'\rangle$. Hence $\langle P,\ \phi\rangle \xrightarrow{\alpha'} \langle P',\ \phi'\rangle$ by `Synchronisation`.

If $P = [t = t']P_1$ (with $t[a_1/x_1]\dots[a_n/x_n] = t'[a_1/x_1]\dots[a_n/x_n]$), then $P_1[a_1/x_1]\dots[a_n/x_n] \xrightarrow{\alpha} Q'$ (resp. $P_1[a_1/x_1]\dots[a_n/x_n] \xrightarrow{\beta} Q'$) and, by the induction hypothesis, we see that

$$\langle P_1,\ \phi\rangle \xrightarrow{\alpha'} \langle P',\ \phi'\rangle \quad (\text{resp. } \langle P_1,\ \phi\rangle \xrightarrow{\beta'} \langle P',\ \phi'\rangle).$$

Therefore, by semantics rule `Match`, $\langle P,\ \phi\rangle \xrightarrow{\alpha'} \langle P',\ \phi'\rangle$ (resp. $\langle P,\ \phi\rangle \xrightarrow{\beta'} \langle P',\ \phi'\rangle$) with $\phi' \not\Leftrightarrow \mathbf{0}$ since $\varrho \models (t == t')$ for any ϱ such that $\varrho(x_i) = a_i$.

If $P = P_1 \setminus L$ (with $\alpha, \beta \notin L$), then $P_1[a_1/x_1]\dots[a_n/x_n] \xrightarrow{\alpha} Q'$ (resp. $P_1[a_1/x_1]\dots[a_n/x_n] \xrightarrow{\beta} Q'$) and, by the induction hypothesis, we have $\langle P_1,\ \phi\rangle \xrightarrow{\alpha'} \langle P',\ \phi''\rangle$ (resp. $\langle P_1,\ \phi\rangle \xrightarrow{\beta'} \langle P',\ \phi''\rangle$) with $\phi'' \not\Leftrightarrow \mathbf{0}$. Therefore, $\langle P,\ \phi\rangle \xrightarrow{\alpha'} \langle P',\ \phi'\rangle$ (resp. $\langle P,\ \phi\rangle \xrightarrow{\beta'} \langle P',\ \phi'\rangle$) with $\phi' \not\Leftrightarrow \mathbf{0}$ since $\varrho \models \phi^L_{\alpha'}$ (resp. $\varrho \models \phi^L_{\beta'}$) for any ϱ such that $\varrho(x_i) = a_i$.

Finally, if $P = P_1/\mathcal{O}$, then there is a computation

$$P_1[a_1/x_1]\dots[a_n/x_n] \xrightarrow{\gamma} P'_1[a_1/x_1]\dots[a_n/x_n]$$

$$(\text{resp. } P_1[a_1/x_1]\dots[a_n/x_n] \xrightarrow{\gamma} P'_1[a/x][a_1/x_1]\dots[a_n/x_n])$$

such that $\gamma \in \mathcal{O}^{-1}(\alpha)$ (resp. $\gamma \in \mathcal{O}^{-1}(\beta)$) and with $Q' = P'_1[a_1/x_1]\dots[a_n/x_n]/\mathcal{O}$ (resp. $Q' = P'_1[a/x][a_1/x_1]\dots[a_n/x_n]/\mathcal{O}$). Thus, by the induction hypothesis, we have $\langle P_1,\ \phi\rangle \xrightarrow{\gamma'} \langle P'_1,\ \phi'\rangle$ where $\gamma = \gamma'[a_1/x_1]\dots[a_n/x_n]$ (resp. $\gamma = \gamma'[a/x][a_1/x_1]\dots[a_n/x_n]$). Hence, by semantics rule `Observation` and since $P = P_1/\mathcal{O}$ and $P' = P'_1/\mathcal{O}$, we see that $\langle P,\ \phi\rangle \xrightarrow{\alpha'} \langle P',\ \phi'\rangle$, (resp. $\langle P,\ \phi\rangle \xrightarrow{\beta'} \langle P',\ \phi'\rangle$) which concludes the proof. □

Proof of Lemma 4. For simplicity, the two statements are proved simultaneously by induction on the structure of P. The case where $P = \mathbf{0}$ is trivial.

Let ϱ be a valuation such that $\varrho \models \phi'$ and put $Q ::= P[\varrho(x_1)/x_1]\ldots[\varrho(x_n)/x_n]$ and, depending on the statement proved, put either

$$Q' ::= P'[\varrho(x_1)/x_1]\ldots[\varrho(x_n)/x_n]$$

or

$$Q' ::= P'[\varrho(x)/x][\varrho(x_1)/x_1]\ldots[\varrho(x_n)/x_n].$$

Also put $\alpha ::= \varrho(\alpha')$ and $\beta ::= \varrho(\beta')$.

If $P = \overline{c}(t).P'$, or $P = c(x).P'$, or $P = \texttt{let}\ x = f(t)\ \texttt{in}\ P'$, or $P = \texttt{let}\ (x, y) = t\ \texttt{in}\ P'$, or $P = \texttt{case}\ t\ \texttt{of}\ \{x\}_{t'}\ \texttt{in}\ P'$, or $P = \texttt{case}\ t\ \texttt{of}\ [t'']_{t'}\ \texttt{in}\ P'$, then we have either $Q = \overline{c}(\varrho(t)).Q'$, or $Q = c(x).Q'$, or $Q = \texttt{let}\ x = f(\varrho(t))\ \texttt{in}\ Q'$, or $Q = \texttt{let}\ (x, y) = \varrho(t)\ \texttt{in}\ Q'$, or $Q = \texttt{case}\ \{\varrho(t')\}_{\varrho(t)}\ \texttt{of}\ \{x\}_{\varrho(t)}\ \texttt{in}\ Q'$, or $Q = \texttt{case}\ [\varrho(t')]_{\varrho(t)}\ \texttt{of}\ [x]_{\varrho(t)}\ \texttt{in}\ Q'$. Thus $Q \xrightarrow{\alpha} Q'$ (resp. $Q \xrightarrow{\beta} Q'$).

If $P = P_1 + P_2$, or $P = P_1|P_2$, or $P = P_1 \parallel P_2$, then either $Q = Q_1 + Q_2$, or $Q = Q_1|Q_2$, or $Q = Q_1 \parallel Q_2$, where $Q_1 ::= P_1[\varrho(x_1)/x_1]\ldots[\varrho(x_n)/x_n]$ and $Q_2 ::= P_2[\varrho(x_1)/x_1]\ldots[\varrho(x_n)/x_n]$. Whenever α' is not a marker action, we may assume that $\langle P_1,\ \phi\rangle \xrightarrow{\alpha'} \langle P_1',\ \phi'\rangle$ (resp. $\langle P_1,\ \phi\rangle \xrightarrow{\beta'} \langle P_1',\ \phi'\rangle$) with either $P' ::= P_1'$, or $P' ::= P_1'|P_2$, or $P' ::= P_1' \parallel P_2$. Hence, by the induction hypothesis, we see that

$$Q_1 \xrightarrow{\alpha} P_1'[\varrho(x_1)/x_1]\ldots[\varrho(x_n)/x_n]$$

$$(\text{resp. } Q_1 \xrightarrow{\beta} P_1'[\varrho(x)/x][\varrho(x_1)/x_1]\ldots[\varrho(x_n)/x_n]),$$

thus, $Q \xrightarrow{\alpha} Q'$ (resp. $Q \xrightarrow{\beta} Q'$). Now assume that α' is a marker action, with $P = P_1 \parallel P_2$. Then we may assume that $\langle P_1,\ \psi_1\rangle \xrightarrow{\overline{c_{id_1}}(t)} \langle P_1',\ \psi_1'\rangle$ and $\langle P_2,\ \psi_2\rangle \xrightarrow{c_{id_2}(x)} \langle P_2',\ \psi_2'\rangle$ with $\phi = \psi_1 \wedge \psi_2$. By the induction hypothesis, we have

$$Q_1 \xrightarrow{\overline{c_{id_1}}(\varrho(t))} P_1'[\varrho(x_1)/x_1]\ldots[\varrho(x_n)/x_n]$$

and

$$Q_2 \xrightarrow{c_{id_2}(\varrho(t))} P_2'[a/x][\varrho(x_1)/x_1]\ldots[\varrho(x_n)/x_n].$$

Therefore, $P \xrightarrow{\alpha} P'$ by rule `Synchronisation`.

If $P = [t = t']P_1$, then we have $\langle P_1,\ \phi\rangle \xrightarrow{\alpha'} \langle P',\ \psi\rangle$ (resp. $\langle P_1,\ \phi\rangle \xrightarrow{\beta'} \langle P',\ \psi\rangle$) with $\varrho \models \psi$ and $\varrho \models (t == t')$ since $\varrho \models \phi'$. Therefore, by the induction hypothesis, we have $\langle Q_1,\ \phi\rangle \xrightarrow{\alpha'} \langle Q',\ \psi\rangle$ (resp. $\langle Q_1,\ \phi\rangle \xrightarrow{\beta'} \langle Q',\ \psi\rangle$) where $Q_1 ::= P_1[\varrho(x_1)/x_1]\ldots[\varrho(x_n)/x_n]$. Thus, by rule `Match`, $Q \xrightarrow{\alpha} Q'$ (resp.$Q \xrightarrow{\beta} Q'$) since $\varrho(t) = \varrho(t')$.

If $P = P_1 \setminus L$, then we have $\langle P_1,\ \phi\rangle \xrightarrow{\alpha'} \langle P_1',\ \psi\rangle$ (resp. $\langle P_1,\ \phi\rangle \xrightarrow{\beta'} \langle P_1',\ \psi\rangle$) with $P' = P_1' \setminus L$ and $\phi = \psi \wedge \phi^L_{\alpha'}$ (resp. $\phi = \psi \wedge \phi^L_{\beta'}$). Furthermore, we see that

$Q = Q_1 \setminus L$ where $Q_1 ::= P_1[\varrho(x_1)/x_1] \dots [\varrho(x_n)/x_n]$. Since $\varrho \models \psi$, we see, by the induction hypothesis, that

$$Q_1 \xrightarrow{\alpha} P_1'[\varrho(x_1)/x_1] \dots [\varrho(x_n)/x_n]$$

$$(\text{resp. } Q_1 \xrightarrow{\beta} P_1'[\varrho(x)/x][\varrho(x_1)/x_1] \dots [\varrho(x_n)/x_n]).$$

But $\varrho \models \phi'$, therefore $\varrho \models \phi^L_{\alpha'}$ (resp. $\varrho \models \phi^L_{\beta'}$), we have $\alpha, \beta \notin L$. Hence $Q \xrightarrow{\alpha} Q'$ (resp. $Q \xrightarrow{\beta} Q'$).

If $P = P_1/\mathcal{O}$, then there is a computation $\langle P_1,\ \phi\rangle \xrightarrow{\gamma'} \langle P_1',\ \phi'\rangle$ such that $\gamma' \in \mathcal{O}^{-1}(\alpha)$ (resp. $\gamma' \in \mathcal{O}^{-1}(\beta)$) and with $P' = P_1'/\mathcal{O}$. By the induction hypothesis, we see that $Q_1 \xrightarrow{\gamma} Q_1'$ where $Q_1' ::= P_1'[\varrho(x_1)/x_1] \dots [\varrho(x_n)/x_n]$ (resp. $Q_1' ::= P_1'[\varrho(x)/x][\varrho(x_1)/x_1] \dots [\varrho(x_n)/x_n]$) and $\gamma = \varrho(\gamma')$. Thus, $Q \xrightarrow{\alpha} Q'$ since $\gamma \in \mathcal{O}^{-1}(\alpha)$ (resp. $Q \xrightarrow{\beta} Q'$ since $\gamma \in \mathcal{O}^{-1}(\beta)$). $\square$

Proof of Lemma 5. For simplicity, the two statements are proved simultaneously by induction on the structure of P. If the transition $\langle P,\ \phi\rangle \xrightarrow{\alpha'} \langle P',\ \phi'\rangle$ (resp. $\langle P,\ \phi\rangle \xrightarrow{\beta'} \langle P',\ \phi'\rangle$) comes from either `Output`, `Input`, `Function`, `Generator`, `Split`, `Decryption`, `Signature-Verif`, then we directly see from the definition of ϕ' that $\models \phi'[a_1/x_1] \dots [a_n/x_n]$ (resp. $\models \phi'[a/x][a_1/x_1] \dots [a_n/x_n]$) whenever $\models \phi[a_1/x_1] \dots [a_n/x_n]$.

If $P = P_1 + P_2$, or $P = P_1|P_2$, or $P = P_1 \parallel P_2$ (and α is not a marker action), then we may assume that $\phi = \phi_1 \wedge \phi_2$ and $\phi' = \phi_1' \wedge \phi_2$, with $\langle P_1,\ \phi_1\rangle \xrightarrow{\alpha'} \langle P_1',\ \phi_1'\rangle$ (resp. $\langle P_1,\ \phi_1\rangle \xrightarrow{\beta'} \langle P_1',\ \phi_1'\rangle$). Therefore, since $\models \phi[a_1/x_1] \dots [a_n/x_n]$, we must have $\models \phi_1[a_1/x_1] \dots [a_n/x_n]$, therefore by the induction hypothesis we see that $\models \phi_1'[a_1/x_1] \dots [a_n/x_n]$ (resp. $\models \phi_1'[a/x][a_1/x_1] \dots [a_n/x_n]$). Hence, we have $\models \phi'[a/x][a_1/x_1] \dots [a_n/x_n]$ (resp. $\models \phi'[a/x][a_1/x_1] \dots [a_n/x_n]$) since $\models \phi_2[a_1/x_1] \dots [a_n/x_n]$. The case where α is a marker action is similar. Now, if $P = [t = t']P_1$, then we see, for any definition for ϕ', that the statement holds since $t[a_1/x_1] \dots [a_n/x_n] = t'[a_1/x_1] \dots [a_n/x_n]$.

If $P = P_1 \setminus L$, then $\phi' = \phi_1 \wedge \phi^L_{\alpha'}$ (resp. $\phi' = \phi_1 \wedge \phi^L_{\beta'}$). By the induction hypothesis, we see that $\models \phi_2[a_1/x_1] \dots [a_n/x_n]$ (resp. $\models \phi_1[a/x][a_1/x_1] \dots [a_n/x_n]$), and since $\alpha, \beta \notin L$, $\models \phi^L_{\alpha'}[a_1/x_1] \dots [a_n/x_n]$ (resp. $\models \phi^L_{\beta'}[a/x][a_1/x_1] \dots [a_n/x_n]$). Therefore, $\models \phi'[a_1/x_1] \dots [a_n/x_n]$ (resp. $\models \phi'[a/x][a_1/x_1] \dots [a_n/x_n]$). Finally, assume that $P = P_1/\mathcal{O}$. Then $\langle P_1,\ \phi\rangle \xrightarrow{\gamma'} \langle P_1',\ \phi'\rangle$, where $P' = P_1'$ and $\gamma' \in \mathcal{O}^{-1}(\alpha')$ (resp. $\gamma' \in \mathcal{O}^{-1}(\beta')$). Thus, by using the induction hypothesis for each action within the sequence γ', we see that $\models \phi'[a_1/x_1] \dots [a_n/x_n]$ (resp. $\models \phi'[a/x][a_1/x_1] \dots [a_n/x_n]$) whenever $\models \phi[a_1/x_1] \dots [a_n/x_n]$. $\square$