Proof Techniques for Cryptographic Processes*

Michele Boreale Rocco De Nicola Rosario Pugliese
Dipartimento di Sistemi e Informatica, Università di Firenze
e-mail: {boreale,denicola,pugliese}@dsi.unifi.it

Abstract

Contextual equivalences for cryptographic process calculi, like the spi-calculus, can be used to reason about correctness of protocols, but their definition suffers from quantification over all possible contexts. Here, we focus on two such equivalences, namely may-testing and barbed equivalence, and investigate tractable proof methods for them. To this aim, we design an enriched labelled transition system, where transitions are constrained by the knowledge the environment has of names and keys. The new transition system is then used to define a trace equivalence and a weak bisimulation equivalence, that avoid quantification over contexts. Our main results are soundness and completeness of trace and weak bisimulation equivalence with respect to may-testing and barbed equivalence, respectively. They lead to more direct proof methods for equivalence checking. The use of these methods is illustrated with a few examples, concerning implementation of secure channels and verification of protocol correctness.

Keywords: Process calculi, Reasoning about security, Semantics, Formal methods.

*Appeared in *SIAM Journal on Computing*, vol. 31(3), pp. 947-986, 2002. Extended and revised version of [9].

Contents

1 Introduction **3**

2 The Language **6**
2.1 Syntax 7
2.2 Operational semantics 9
2.3 May-testing and barbed equivalence 11

3 Trace and Bisimulation Semantics **11**
3.1 An environment-sensitive lts 12
3.2 Trace and bisimulation semantics 13

4 Soundness and Completeness **16**
4.1 May-testing and trace semantics 17
4.2 Barbed equivalence and bisimilarity 20

5 Applications **25**
5.1 Some useful laws 25
5.2 Secure channels implementation 27
5.3 Verification of a small protocol 30

6 A Calculus with Pairs **32**

7 Final Remarks and Related Work **34**

A Results on Environment Equivalence **37**

B Results on Trace and Bisimulation Semantics **41**
B.1 Trace semantics 42
B.2 Bisimilarity 43

C Characteristic Formula: Calculus with Pairing **45**

References **47**

1 Introduction

Recently, there has been much interest toward using formal methods for analysing cryptographic protocols. Here, we focus on a specific approach, which aims at modelling protocols as concurrent processes, described as terms of a process calculus (e.g. the spi-calculus [7, 5], a cryptographic version of the π-calculus [17]). As an example, consider the very simple protocol where two principals A and B share a private key k, and A wants to send B a datum d encrypted under k, through a public channel c:

$$\text{Message 1} \qquad A \to B\colon \{d\}_k \text{ on } c.$$

This informal notation can be expressed in the spi-calculus as follows:

$$\begin{array}{rcl} A(d) & \stackrel{\text{def}}{=} & \overline{c}\{d\}_k.\mathbf{0} \\ B & \stackrel{\text{def}}{=} & c(x).F(x) \\ P(d) & \stackrel{\text{def}}{=} & (\nu\, k)(A(d) \mid B). \end{array}$$

Here, $\overline{c}\{d\}_k.$ means *output* of message $\{d\}_k$ at channel c and $\mathbf{0}$ stands for *termination*. The prefix $c(x).$ indicates the intention to *input* a message at channel c and to bind it to x, and $F(x)$ is some expression describing the behaviour of B after the reception of x. The whole protocol $P(d)$ is the *parallel composition* $A(d) \mid B$, with the *restriction* $(\nu\, k)$ indicating that the key k is only known to $A(d)$ and B.

The main advantage of this kind of description is that process calculi have a formal yet simple semantics that permits rendering rigorously such notions as 'attacker' and 'secrecy'. Continuing with the example above, a way of asserting that $P(d)$ keeps d secret is requiring that $P(d)$ be *equivalent* to $P(d')$, for every other d'. An appropriate notion of equivalence is *may-testing* [11, 8, 7]; its intuition is precisely that no external context (which in the present setting can be read as 'attacker') may notice any difference when running in parallel with $P(d')$ or $P(d)$. A similar intuition is supported by other contextual equivalences, like *barbed equivalence* [18]. While rigorous and intuitive, the definitions of these equivalences suffer from universal quantification over contexts (attackers), that makes equivalence checking very hard. It is then important to devise proof techniques that avoid such quantification. Results in this direction have been obtained for traditional process calculi (for example, in CCS [11, 15] and in π-calculus [17], may-testing is easily proven to coincide with trace equivalence), but little has been done for cryptographic calculi.

In this paper, we consider may-testing and barbed equivalence for a variant of the spi-calculus with shared-key encryption primitives [7]. We develop an 'environment-sensitive' labelled transition system (lts), whose transitions are constrained by the knowledge the environment has of names and keys. A trace-based equivalence and a purely co-inductive notion of weak bisimulation, that avoid quantification over contexts, are defined on the new lts and it is shown that they are in agreement with may-testing and barbed equivalence, respectively. A more detailed account of our work follows.

The handling of names is a crucial aspect in the semantics of process calculi. Let us first consider the nature of the transitions in the non-cryptographic π-calculus. There are three kinds of basic moves which correspond to: output of a message, input of a message

and internal computation. An output transition like $P \xrightarrow{(\nu\, b)\overline{a}\langle b\rangle} P'$ says that process P passes a *new* (or *fresh*, $(\nu\,\cdot)$ means 'new') name b to the environment along channel a, and becomes P' in doing so. The environment can use at will names he has got to know. For example, the two-steps sequence $P \xrightarrow{(\nu\, b)\overline{a}\langle b\rangle} P' \xrightarrow{b\, c} P''$ (where $b\, c$ means 'input c along b') is possible. In general, if a process is ready to perform some action, the environment will always be able to react. The reason is that, in the π-calculus, environment and process share at each stage the *same* knowledge of names. Thus, to determine whether two processes are, say, may-testing equivalent it is sufficient to establish that they can perform the *same* sequences of (input/output) actions.

The correspondence between environment and process knowledge is lost when moving to the spi-calculus, i.e. when adding encryption and decryption primitives to the π-calculus. Indeed, two new facts must be taken into account.

(a) When the environment receives a new name encrypted under a fresh key, it does not acquire the knowledge of that name immediately. For instance, after P outputs a new name b encrypted under a fresh key k (written $P \xrightarrow{(\nu\, b,k)\overline{a}\langle\{b\}_k\rangle} P'$), name b is part of the knowledge of P', but not part of the knowledge of the environment. Thus, if P' is willing to input something at b (say $P' = b(c).P''$), the environment cannot satisfy P''s expectations: a sequence like $P \xrightarrow{(\nu\, b,k)\overline{a}\langle\{b\}_k\rangle} P' \xrightarrow{b\, c} P''$ (that is possible in the traditional-style transition system) cannot be considered as meaningful in the spi-calculus. For similar reasons, a sequence like $P \xrightarrow{(\nu\, b,k)\overline{a}\langle\{b\}_k\rangle} P' \xrightarrow{a'\, b} P''$, where the environment is expected to send back the cleartext b, cannot be considered as meaningful.

(b) Equivalent processes need not exhibit the same sequences of transitions. The process that performs the single output $(\nu\, k)\overline{a}\langle\{b\}_k\rangle$ and terminates, and the one that performs $(\nu\, k)\overline{a}\langle\{c\}_k\rangle$ and terminates, are equivalent, because messages $\{b\}_k$ and $\{c\}_k$ cannot be distinguished by the environment (as it cannot open something encrypted with k). However, the two messages could be distinguished if the environment got the key k. Thus, the two processes $(\nu\, k)\overline{a}\{b\}_k.\,\overline{a}k.\,\mathbf{0}$ and $(\nu\, k)\overline{a}\{c\}_k.\,\overline{a}k.\,\mathbf{0}$ are not equivalent.

To cope with these issues and recover the correspondence between environment and process actions, we introduce an enriched labelled transition system that explicitly keeps track of the environment's knowledge. The states of the new lts are *configurations* $\sigma \rhd P$, where P is a process and σ is the current environment's knowledge, modelled as a mapping from a set of variables to a set of messages. Informally, σ plays the role of a database storing the messages received by the environment; each entry of the database is referenced by a distinct variable. Transitions represent interactions between the environment and the process and take the form

$$\sigma \rhd P \overset{\mu}{\underset{\delta}{\longmapsto}} \sigma' \rhd P'$$

where μ is the action of process P and δ is the 'complementary' *environment action*.

We have three different kinds of situations.

1. The process performs an output and the environment an input. As a consequence, the environment's knowledge gets updated. For instance:

$$\sigma \rhd P \overset{(\nu\, \widetilde{b})\overline{a}\langle M\rangle}{\underset{z(x)}{\longmapsto}} \sigma[^{M}\!/\!_{x}] \rhd P'$$

   here $\sigma[^{M}\!/\!_{x}]$ is the update of σ with the new entry $[^{M}\!/\!_{x}]$, for a fresh variable x. Moreover, $\widetilde{b}$ is the set of new names the process has just created. For the transition to take place, channel a must belong to the knowledge of σ, which in this case amounts to saying that $\sigma(z) = a$.

2. The process performs an input and the environment an output. As discussed previously, messages from the environment cannot be arbitrary, they must be built, via encryption and decryption, using only the knowledge stored in σ. Thus, a transition might be:

$$\sigma \rhd P \overset{a\, M}{\underset{(\nu\, \widetilde{b})\overline{z}\langle\zeta\rangle}{\longmapsto}} \sigma[^{\widetilde{b}}\!/\!_{\widetilde{b}}] \rhd P'.$$

   Informally, $\widetilde{b}$ is the set of new names the environment has just created and added to its knowledge, while ζ is an expression describing how M has been built out of σ and $\widetilde{b}$. For example, if $\sigma(x_1) = \{c\}_k$ and $\sigma(x_2) = k$ and $M = c$ then ζ might be $\mathtt{dec}_{x_2}(x_1)$, indicating that message c results from decrypting the x_1-entry using the x_2-entry as a key. Again, a must belong to the knowledge of σ, thus $\sigma(z) = a$.

3. The process performs an internal move and the environment does nothing:

$$\sigma \rhd P \overset{\tau}{\underset{-}{\longmapsto}} \sigma \rhd P'.$$

When defining trace and bisimulation semantics (Section 3) on the top of the new lts, the point of view is taken that *equivalent configurations should exhibit the same environment actions*. As an example, take σ with entries $\sigma(x) = a$, $\sigma(y) = b$ and $\sigma(z) = c$ and consider the configurations $C_1 \overset{\text{def}}{=} \sigma \rhd (\nu\, k)\overline{a}\{b\}_k.\, \mathbf{0}$ and $C_2 \overset{\text{def}}{=} \sigma \rhd (\nu\, k)\overline{a}\{c\}_k.\, \mathbf{0}$. These configurations are both trace and bisimulation equivalent, because the only transitions they have are:

$$C_1 \overset{(\nu\, k)\overline{a}\langle\{b\}_k\rangle}{\underset{x(w)}{\longmapsto}} \sigma[^{\{b\}_k}\!/\!_{w}] \rhd \mathbf{0}$$

and

$$C_2 \overset{(\nu\, k)\overline{a}\langle\{c\}_k\rangle}{\underset{x(w)}{\longmapsto}} \sigma[^{\{c\}_k}\!/\!_{w}] \rhd \mathbf{0}$$

which exhibit the same environment action, $x(w)$. On the other hand, as discussed above, $C_3 \overset{\text{def}}{=} \sigma \rhd (\nu\, k)\overline{a}\{b\}_k.\, \overline{a}k.\, \mathbf{0}$ and $C_4 \overset{\text{def}}{=} \sigma \rhd (\nu\, k)\overline{a}\{c\}_k.\, \overline{a}k.\, \mathbf{0}$ should not be regarded as equivalent. Indeed, after two steps, C_3 reaches a state where the environment is $\sigma[^{\{b\}_k}\!/\!_{w}][^{k}\!/\!_{v}]$, which cannot be considered 'equivalent' to the environment reachable from C_4, i.e. $\sigma[^{\{c\}_k}\!/\!_{w}][^{k}\!/\!_{v}]$: The decryption of the entry w, that is now possible because k is known, yields distinct names,

b and c, in the two cases. Equivalence on environments is a notion crucial to our approach, and will be formalized in terms of logical equivalence. For both trace and bisimulation equivalence, we shall insist that *matching transitions should take equivalent environments to equivalent environments*. We shall show that these equivalences imply their contextual counterparts (soundness), hence the formers can be used as proof techniques for the latters. The converse implication (completeness) is also proven for trace equivalence. As to bisimulation, we establish completeness relatively to a broad class of processes (which includes all the *image-finite* ones [15]).

Trace and bisimulation equivalences avoid quantification over contexts and only require considering transitions of the enriched lts. As such, they make reasoning on processes much easier than the contextual definitions. While trace semantics is sufficient for expressing many security properties (especially secrecy and autenthicity ones, [7]), bisimulation is sometimes preferrable because it embodies a notion of fairness and is supported by a nice, purely coinductive proof technique. The latter can be enhanced by tailoring, as we do, some 'up-to' techniques [21, 10] to the cryptographic setting. Another advantage of our semantics are the congruence rules that make compositional proofs possible. The use of trace and bisimulation semantics as proof techniques is illustrated with a few examples; some of them concern the problem of implementing secure channels using encrypted public channels (like in [4]). Some of the equalities we establish are hard and lengthy to prove if relying on the original, contextual definitions (see, e.g., the secure channel implementations in Section 5).

The rest of the paper is organized as follows. The language is presented in Section 2; there, we also introduce the contextual semantics, may-testing and barbed equivalences. Section 3 introduces the new lts for the spi-calculus and, based on that, trace and bisimulation semantics. In Section 4, we establish soundness and completeness of trace semantics with respect to may-testing, and of bisimulation with respect to barbed equivalence. Section 5 presents a number of properties of trace and bisimulation semantics and their applications. In Section 6, we present the extension of the theory to a richer calculus, that permits handling pairs of messages. Comparisons to related works and a few concluding remarks are reported in Section 7. The most technical proofs and definitions are relegated to appendices A, B and C.

2 The Language

We first present syntax and (conventional) operational semantics of the language, which is a variant of the spi-calculus. We then define the contextual semantics, may-testing and barbed equivalence. In the definition of the language, there are a few implicit assumptions on the underlying system of shared-key cryptography. We make them explicit below:

1. A message M encrypted under a key k, written $\{M\}_k$, can only be decrypted using k. The only way to produce the ciphertext $\{M\}_k$ is to encrypt M under k. If k is secret, the attacker cannot guess or forge k (*perfect encryption*).

2. There is enough redundancy in the structure of messages to tell whether decryption of a message with a given key has actually succeeded or not.

3. There is enough redundancy in the structure of messages to tell their role (key or compound ciphertext).

4. The only way to form a new key is to get a fresh name from a primitive set of *names*.

These assumptions are quite common in the literature. In particular, the first two are also found in the original spi-calculus [7]. The third and fourth assumptions might be regarded as a limitation over the practice of crypto-protocols, where new keys are sometimes formed by assembling pieces of old messages together (especially if random bits are considered as an expensive resource). However, many interesting protocols do fulfill these assumptions.

2.1 Syntax

The syntax of the calculus is summarized in Table 1. A countable set $\mathcal{N}$ of *names* $a, b \ldots, h, k, \ldots, x, y, z \ldots$ is assumed. Names can be used as communication channels, primitive data or encryption keys: we do not distinguish between these three kinds of objects (notationally, we prefer letters $h, k, \ldots$, when we want to stress the use of a name as a key). In the standard π-calculus, names are the only transmissible objects. In the spi-calculus the possibility has been added to communicate *messages* obtained via shared-key encryption: message $\{M\}_k$ represents the ciphertext obtained by encrypting message M under key k, using a shared-key encryption system. Encryptions can be arbitrarily nested. *Expressions* are obtained applying encryption and decryption operators to names and ciphertexts. For example, the result of evaluating $\mathtt{dec}_\eta(\zeta)$ is the text obtained by decrypting the ciphertext ζ using the value of η as a key. Expressions are also used to represent dummy terms that can be generated at run but do not represent proper messages (such as $\{a\}_{\{b\}_k}$, where a compound term $\{b\}_k$ is used as a key instead of an atomic name). Logical *formulae* generalize the usual equality operator of the π-calculus with a predicate $name(\cdot)$, which tests for the format of the argument (plain name or a compound ciphertext), and with a `let` construct that binds the value of some expression ζ to a name z. *Processes* are built using a set of operators which include those from the standard π-calculus, plus two new operators: boolean guard and a `let` construct. An informal explanation of the operators might be the following:

- $\mathbf{0}$ is the process that does nothing;
- $\eta(x).P$ represents input of a generic message x along η: the only useful case is when η is a name, otherwise the whole process is stuck;
- $\overline{\eta}\zeta.P$ represents output of ζ along η: the only useful case is when η is a name and ζ is a message, otherwise the whole process is stuck;
- $P + Q$ can behave either as P or Q: the choice might either be triggered by the environment, or by internal computations of P or Q;
- $P \mid Q$ is the parallel execution of P and Q;
- $(\nu\, a)P$ creates a new name a which is only known to P;
- $!\,P$ behaves like unboudedly many copies of P running in parallel, i.e. $P \mid P \mid P \mid \cdots$;

$a, b \ldots, h, k, \ldots, x, y, z \ldots$			*names* $\mathcal{N}$
M, N	::=	$a \mid \{M\}_k$	*messages* $\mathcal{M}$
η, ζ	::=	$a \mid \{\eta\}_\zeta \mid \mathtt{dec}_\eta(\zeta)$	*expressions* $\mathcal{Z}$
ϕ, ψ	::=	$tt \mid name(\zeta) \mid [\zeta = \eta]$ $\mid \mathtt{let}\ z = \zeta\ \mathtt{in}\ \phi \mid \phi \wedge \psi \mid \neg\phi$	*formulae* Φ
P, Q	::=		*processes* $\mathcal{P}$
		$\mathbf{0}$	(*null*)
	|	$\eta(x).\, P$	(*input prefix*)
	|	$\overline{\eta}\zeta.\, P$	(*output prefix*)
	|	$P + Q$	(*non − deterministic choice*)
	|	$P \mid Q$	(*parallel composition*)
	|	$(\nu\, a)P$	(*restriction*)
	|	$!\, P$	(*replication*)
	|	ϕP	(*boolean guard*)
	|	$\mathtt{let}\ z = \zeta\ \mathtt{in}\ P$	(*encryption/decryption*)

It is assumed that $\mathtt{dec}.(\cdot)$ does not occur in $name(\zeta)$, $[\zeta = \eta]$, $\eta(x).$ and $\overline{\eta}\zeta.$.
Operators $a(x).\cdot$, $(\nu\, a)\cdot$ and $\mathtt{let}\ z = \zeta\ \mathtt{in}\ \cdot$ are *binders*, with the obvious scope, for names x, a and z, respectively. In $\mathtt{let}\ z = \zeta\ \mathtt{in}\ \cdot$, it is assumed that z does not appear in ζ.

Table 1: Syntax of the calculus

- ϕP behaves like P if the formula ϕ is logically true, otherwise it is stuck;
- $\mathtt{let}\ z = \zeta\ \mathtt{in}\ P$ attempts evaluation of ζ: if the evaluation succeeds the result is bound to z within P, otherwise the whole process is stuck.

There are a few differences from Abadi and Gordon's spi-calculus [7]. In particular:

- Our decryption keys cannot be compound messages, as already noted.
- For decryption, we use a 'let' construct instead of the 'case' construct of [7]. This enables us to write process expressions in a more compact form: for instance, the spi-calculus process $\mathtt{case}\ M\ \mathtt{of}\ \{k\}_h\ \mathtt{in}\ (\mathtt{case}\ N\ \mathtt{of}\ \{z\}_k\ \mathtt{in}\ P)$ can be written as $\mathtt{let}\ z = \mathtt{dec}_{\mathtt{dec}_h(M)}(N)\ \mathtt{in}\ P$ in our syntax.
- We have included a non-deterministic choice $+$, which is sometimes useful for specification purposes. Another – technical – reason for including $+$ in the language is that this operator appears to be necessary when proving coincidence of barbed equivalence and bisimilarity.
- We do not consider public key and hash functions, which are present in the original spi-calculus.

A minor difference is that the syntax above does not mention *tuples*, which we have preferred to treat separately (in Section 6). This permits a cleaner presentation of the overall approach.

We shall often abbreviate $\alpha.\mathbf{0}$ as α, where α is an input or output prefix, and $(\nu\, a)(\nu\, b)P$ as $(\nu\, a, b)P$. We shall use the tilde $\tilde{\cdot}$ to denote tuples of objects (e.g. $\tilde{x}$ is a generic tuple of names); this will sometimes be written as $\tilde{x}_{i\in I}$, for an appropriate index-set I. If $\tilde{x} = (x_1, \ldots, x_n)$ and $\tilde{y} = (y_1, \ldots, y_m)$ then $\tilde{x}\tilde{y}$ will denote the tuple $(x_1, \ldots, x_n, y_1, \ldots, y_m)$. When convenient, we shall regard a tuple simply as a set (writing e.g. $\tilde{x} \subseteq S$ to mean that all components of $\tilde{x}$ are in S). All our notations are extended to tuples component-wise. In particular, if $\tilde{k} = (k_1, \ldots, k_n)$, then $\{M\}_{\tilde{k}}$ means $\{\cdots\{M\}_{k_1}\cdots\}_{k_n}$ and $\mathtt{dec}_{\tilde{k}}(M)$ means $\mathtt{dec}_{k_n}(\cdots\mathtt{dec}_{k_1}(M)\cdots)$.

Notions of *free names* of a process P, $fn(P)$, of *bound names* of P, $bn(P)$, and of alpha-equivalence arise as expected; $n(P)$ is $fn(P) \cup bn(P)$. Often, we shall write $fn(P, Q)$ in place of $fn(P) \cup fn(Q)$ (similarly for $bn(\cdot)$ and $n(\cdot)$). Similar notations are used for formulae, expressions and messages.

A *substitution* σ is a finite partial map from $\mathcal{N}$ to the set of messages $\mathcal{M}$. The domain and proper co-domain of σ are written $dom(\sigma)$ and $range(\sigma)$, respectively. We let $n(\sigma) = dom(\sigma) \cup (\cup_{M\in range(\sigma)} n(M))$. Given a tuple of distinct names $\tilde{x} = (x_1, \ldots, x_n)$ and a tuple of messages $\widetilde{M} = (M_1, \ldots, M_n)$, the substitution mapping each x_i to M_i will be sometimes written as $[\widetilde{M}/\tilde{x}]$ or $[M_i/x_i]_{i\in 1..n}$. When $\tilde{x} \cap dom(\sigma) = \emptyset$, we write $\sigma[\widetilde{M}/\tilde{x}]$ for the substitution σ' which is the union of σ and $[\widetilde{M}/\tilde{x}]$ (in this case we say that σ' *extends* σ). For a given $V \subseteq_{\text{fin}} \mathcal{N}$, we write ϵ_V for the substitution with $dom(\epsilon_V) = V$ that acts as the identity on V. For any term (name/expression/formula/process) t, $t\sigma$ denotes the term obtained by simultaneously replacing each $x \in fn(t) \cap dom(\sigma)$ with $\sigma(x)$, with renaming of bound names of t possibly involved to avoid captures.

2.2 Operational semantics

The (conventional) operational semantics defined below only accounts for process intentions. In fact, the subsequent definitions of contextual equivalences only use the part of this semantics that describes 'internal computation' (denoted by $\xrightarrow{\tau}$) of processes.

First, we need two evaluation functions: one for expressions, the other for formula. The *evaluation function for expressions*, $\widehat{\cdot} : \mathcal{Z} \to \mathcal{M} \cup \{\bot\}$ (where $\bot$ is a distinct symbol), is defined by induction on ζ as follows:

- $\widehat{a} = a$
- $\widehat{\{\zeta_1\}_{\zeta_2}} = \begin{cases} \{M\}_k & \text{if } \widehat{\zeta_1} = M \text{ and } \widehat{\zeta_2} = k \in \mathcal{N}, \text{ for some } M \text{ and } k \\ \bot & \text{otherwise} \end{cases}$
- $\widehat{\mathtt{dec}_{\zeta_2}(\zeta_1)} = \begin{cases} M & \text{if } \widehat{\zeta_1} = \{M\}_k \text{ and } \widehat{\zeta_2} = k \in \mathcal{N}, \text{ for some } M \text{ and } k \\ \bot & \text{otherwise} \end{cases}$

Note that the evaluation of an expression 'fails', i.e. returns the value $\bot$, whenever an encryption/decryption with something different from a name is attempted, or whenever a

(Inp) $a(x).\,P \xrightarrow{a\,M} P[M/x]$ (Out) $\overline{a}M.\,P \xrightarrow{\overline{a}\langle M\rangle} P$

(Sum) $\dfrac{P \xrightarrow{\mu} P'}{P+Q \xrightarrow{\mu} P'}$ (Rep) $\dfrac{P \mid !\,P \xrightarrow{\mu} P'}{!\,P \xrightarrow{\mu} P'}$

(Par) $\dfrac{P \xrightarrow{\mu} P'}{P \mid Q \xrightarrow{\mu} P' \mid Q}$ (Com) $\dfrac{P \xrightarrow{(\nu\,\widetilde{b})\overline{a}\langle M\rangle} P' \quad Q \xrightarrow{a\,M} Q'}{P \mid Q \xrightarrow{\tau} (\nu\,\widetilde{b})(P' \mid Q')}$

(Res) $\dfrac{P \xrightarrow{\mu} P'}{(\nu\,c)P \xrightarrow{\mu} (\nu\,c)P'}\ c \notin n(\mu)$ (Open) $\dfrac{P \xrightarrow{(\nu\,\widetilde{b})\overline{a}\langle M\rangle} P'}{(\nu\,c)P \xrightarrow{(\nu\,\widetilde{b}c)\overline{a}\langle M\rangle} P'}\ c \neq a, c \in n(M) - \widetilde{b}$

(Guard) $\dfrac{[\![\,\phi\,]\!] = tt \quad P \xrightarrow{\mu} P'}{\phi P \xrightarrow{\mu} P'}$ (Let) $\dfrac{\widehat{\zeta} \neq \bot \quad P[\widehat{\zeta}/z] \xrightarrow{\mu} P'}{\mathtt{let}\ z = \zeta\ \mathtt{in}\ P \xrightarrow{\mu} P'}$

Table 2: Operational semantics (symmetric versions of (Sum), (Par) and (Com) omitted)

decryption with something different from the encryption key is attempted. For instance, the evaluation of $\{a\}_{\{b\}_c}$ and $\mathtt{dec}_b(\{a\}_c)$ is $\bot$, while $\mathtt{dec}_c(\{a\}_c)$ evaluates to a.

The *evaluation function for formulae*, $[\![\,\cdot\,]\!] : \Phi \to \{tt, ff\}$, is defined by induction on ϕ. The only non-standard clauses are those for $name(\zeta)$ and for $\mathtt{let}\ z = \zeta\ \mathtt{in}\ \phi$:

- $[\![\,name(\zeta)\,]\!] = \begin{cases} tt & \text{if } \zeta \in \mathcal{N} \\ ff & \text{otherwise.} \end{cases}$
- $[\![\,\mathtt{let}\ z = \zeta\ \mathtt{in}\ \phi\,]\!] = \begin{cases} [\![\,\phi[\widehat{\zeta}/z]\,]\!] & \text{if } \widehat{\zeta} \neq \bot \\ ff & \text{otherwise.} \end{cases}$

For any substitution σ, $\sigma \models \phi$ means that $[\![\,\phi\,\sigma\,]\!] = tt$.

The operational semantics is defined by the early-style inference rules of Table 2. All rules but the last two are standard from π-calculus. Rule (Guard) says that process ϕP behaves like P provided that ϕ evaluates to true; otherwise, process ϕP is stuck. Rule (Let) attempts evaluation of expression ζ: if the evaluation succeeds, then process $\mathtt{let}\ z = \zeta\ \mathtt{in}\ P$ behaves like process $P[\widehat{\zeta}/z]$, otherwise $\mathtt{let}\ z = \zeta\ \mathtt{in}\ P$ is stuck.

Process *actions* (i.e. labels of the transition system), ranged over by $\mu, \lambda, \ldots$, can be of three forms: τ (internal action), $a\,M$ (input at a where message M is received) and $(\nu\,\widetilde{b})\overline{a}\langle M\rangle$ (output at a where message M containing the fresh, private names $\widetilde{b}$ is sent). We shall write $\overline{a}\langle M\rangle$ instead of $(\nu\,\widetilde{b})\overline{a}\langle M\rangle$ whenever $\widetilde{b} = \emptyset$. Input and output actions will be called *visible* actions. We use s to range over sequences of visible actions (traces), and write $\Longrightarrow$ or $\xRightarrow{\epsilon}$ to denote the reflexive and transitive closure of $\xrightarrow{\tau}$ and, inductively, $\xRightarrow{s}$ for $\Longrightarrow\xrightarrow{\mu}\xRightarrow{s'}$ when $s = \mu \cdot s'$. $P \xRightarrow{s}$ will stand for 'there is P' s.t. $P \xRightarrow{s} P'$ for some P''.
From now on, we shall adopt the following:

Convention 2.1 We identify alpha-equivalent processes and formulae. Moreover, both in actions and in sequences of actions, *we shall assume that bound names can be freely renamed*

with fresh names. In particular, we shall always assume that bound names are distinct from each other and from the free names, and not touched by substitutions.

2.3 May-testing and barbed equivalence

We instantiate the general framework of may-testing [11] to our calculus. *Observers*, ranged over by $O, O', \ldots$, are processes that can perform a distinct 'success' action ω. Informally, the latter is used to signal that the observed process has passed a test. For instance, the observer $(\nu\, b)\overline{a}b.\, b(x).\, [x = c]\omega$, when run in parallel with any process, tests for the ability of the process to receive a new name b on channel a and then to send name c along b. The may-testing preorder is defined in terms of the ability of processes to pass tests proposed by observers. Since we work in a non-deterministic setting, a process *may* or *may not* pass a specific test. If one interprets 'passing a test' as 'revealing a piece of information', then two processes that may pass the same tests may potentially reveal the same information to observers: as such, they should be considered as equivalent from a security point of view. We can formalize this concept solely in terms of sequences of internal computations ($\Longrightarrow$) and success action (ω).

Definition 2.2 (may-testing preorder)
$P \sqsubseteq_{\approx} Q$ if, for every observer O, $P \mid O \stackrel{\omega}{\Longrightarrow}$ implies $Q \mid O \stackrel{\omega}{\Longrightarrow}$. $\diamond$

The equivalence obtained as the kernel of the preorder $\sqsubseteq_{\approx}$ is denoted by $\simeq$ ($\simeq = \sqsubseteq_{\approx} \cap \sqsubseteq_{\approx}^{-1}$) and is called *testing equivalence*.

The intuition behind barbed equivalence [18] is somehow similar to that of testing, but is based on a notion of step-by-step simulation between two processes. In what follows, we say that a process P *commits to* a, and write $P \downarrow a$, if $P \xrightarrow{a\, M}$ or $P \xrightarrow{(\nu\, b)\overline{a}\langle M\rangle}$, for some M and $\widetilde{b}$. We also write $P \Downarrow a$ if $P \Longrightarrow P'$ and $P' \downarrow a$, for some P'.

Definition 2.3 (barbed equivalence) *A symmetric relation $\mathcal{S} \subseteq \mathcal{P} \times \mathcal{P}$ is a* barbed bisimulation *if whenever $P \mathcal{S} Q$ then:*

1. *for each P', if $P \xrightarrow{\tau} P'$ then there is Q' such that $Q \Longrightarrow Q'$ and $P' \mathcal{S} Q'$, and*
2. *for each a, if $P \downarrow a$ then $Q \Downarrow a$.*

Barbed bisimilarity*, written $\dot{\cong}$, is the largest barbed bisimulation relation. Two processes P and Q are* barbed equivalent*, written $P \cong Q$, if for all R we have that $P \mid R \dot{\cong} Q \mid R$.* $\diamond$

It is worthwhile to notice that neither $\sqsubseteq_{\approx}$ nor $\cong$ are (pre-)congruences, because they are not preserved by input prefix: this is standard in name-passing languages (see, e.g., [17, 10]).

3 Trace and Bisimulation Semantics

In this section we shall first introduce an 'environment-sensitive' lts for our calculus and then define trace and bisimulation equivalences over the new lts.

It is assumed that $n(\eta) \subseteq dom(\sigma)$ and that names in $\widetilde{b}$ are fresh for σ and P.

$$\text{(E-Out)}\ \frac{P \xrightarrow{(\nu\,\widetilde{b})\overline{a}\langle M\rangle} P' \qquad \widehat{\eta\sigma} = a}{\sigma \rhd P \overset{(\nu\,\widetilde{b})\overline{a}\langle M\rangle}{\underset{\eta(x)}{\longmapsto}} \sigma[M/x] \rhd P'} \qquad \text{(E-Tau)}\ \frac{P \xrightarrow{\tau} P'}{\sigma \rhd P \overset{\tau}{\underset{-}{\longmapsto}} \sigma \rhd P'}$$

$$\text{(E-Inp)}\ \frac{P \xrightarrow{a\,M} P' \qquad \widehat{\eta\sigma} = a \qquad M = \widehat{\zeta\sigma} \qquad \widetilde{b} \overset{\text{def}}{=} (n(\zeta) - dom(\sigma))}{\sigma \rhd P \overset{a\,M}{\underset{(\nu\,\widetilde{b})\overline{\eta}\langle\zeta\rangle}{\longmapsto}} \sigma[\widetilde{b}/\widetilde{b}] \rhd P'}$$

Table 3: Rules for the environment-sensitive lts

3.1 An environment-sensitive lts

States are *configurations* of the form $\sigma \rhd P$, where P is a process and substitution σ represents the environment (from now on, terms 'substitution' and 'environment' will be used interchangeably). Transitions take the form

$$\sigma \rhd P \overset{\mu}{\underset{\delta}{\longmapsto}} \sigma' \rhd P'$$

and represent atomic interactions between process P and environment σ, μ is the process action (i.e. input, output or τ) and δ is the complementary *environment action*. The latter can be of three forms, output, input and 'no action':

$$\delta ::= (\nu\,\widetilde{b})\overline{\eta}\langle\zeta\rangle \quad | \quad \eta(x) \quad | \quad - \,.$$

The upper transition labels are not strictly necessary for the development of our theory. However, they are useful because they show the process action that triggers the transition.

Free names and bound names of δ are defined as expected, in particular $bn(\eta(x)) = \{x\}$. The *visible* environment actions are input and output. We shall use u to range over sequences of visible environment actions. The inference rules for the transition relation $\overset{\mu}{\underset{\delta}{\longmapsto}}$ are displayed in Table 3. Note that judgements from the conventional transition system are used in the premises.

In rule (E-Out), the environment receives a message M and updates its knowledge accordingly. For the transition to take place, channel a must belong to the knowledge of the environment, thus η is some expression describing how a can be obtained out of σ (this is what $\widehat{\eta\sigma} = a$ means). In rule (E-Inp), the environment sends a message M to the process: expression ζ describes how message M is built out of σ and $\widetilde{b}$. The update $[\widetilde{b}/\widetilde{b}]$ records the creation of the new names $\widetilde{b}$[1]. Like in the previous rule, a must belong to the knowledge of σ.

In the following, we write $\Longmapsto$ to denote the reflexive and transitive closure of $\overset{\tau}{\underset{-}{\longmapsto}}$ and, inductively, write $\overset{s}{\underset{u}{\Longmapsto}}$ for $\Longmapsto\overset{\mu}{\underset{\delta}{\longmapsto}}\overset{s'}{\underset{u'}{\Longmapsto}}$ when $s = \mu \cdot s'$ and $u = \delta \cdot u'$.

[1]Actually, any update $[\widetilde{b}/\widetilde{x}]$ – with $\widetilde{x}$ a tuple of fresh names – could have been used instead.

3.2 Trace and bisimulation semantics

In order to define observational semantics based on $\overset{\mu}{\underset{\delta}{\longmapsto}}$, we have to precisely define when two environments, represented by substitutions σ and σ', can be considered as equivalent. Informally, two environments σ and σ' are equivalent whenever they are logically indistinguishable.

Definition 3.1 (equivalence on environments) *Two substitutions σ and σ' are* equivalent*, written $\sigma \sim \sigma'$, if $dom(\sigma) = dom(\sigma')$ and for each formula ϕ with $fn(\phi) \subseteq dom(\sigma)$, it holds that $\sigma \models \phi$ if and only if $\sigma' \models \phi$.* $\diamond$

This logical characterization is difficult to check, as it contains a quantification on all formulae. Below, we shall give an equivalent definition that is easy to check. To do this, we start by making precise the concept of knowledge of an environment σ, that is, all the information that can be deduced from σ.

Definition 3.2 (decryption closure and knowledge) *Let W be a set of messages. The* decryption closure *of W, written $dc(W)$, is the set of messages defined inductively as follows:*

(i) $W \subseteq dc(W)$, and

(ii) if $k \in dc(W)$ and $\{M\}_k \in dc(W)$ then $M \in dc(W)$.

The knowledge *of W, written $kn(W)$, is the set of names in $dc(W)$, i.e.: $kn(W) \stackrel{\text{def}}{=} dc(W) \cap \mathcal{N}$.*

Let σ be a substitution; we let $dc(\sigma) \stackrel{\text{def}}{=} dc(range(\sigma))$ and $kn(\sigma) \stackrel{\text{def}}{=} kn(range(\sigma))$. $\diamond$

Note that $kn(\sigma)$ can be computed in a finite number of steps. For instance, it is not difficult to see that $kn([\{b\}_l/w, \{a\}_k/x, \{k\}_h/y, h/z]) = \{a, h, k\}$. Next, a couple of notational shorthands.

- Given any substitution $\sigma = [M_i/x_i]_{i \in I}$ and $i \in I$, we denote by $core(\sigma, x_i)$ what is left of M_i after decrypting as much as possible using the keys in $kn(\sigma)$. Formally, we define $core(\sigma, x_i)$ as the message N such that: for some $\widetilde{k} \subseteq kn(\sigma)$, it holds $M_i = \{N\}_{\widetilde{k}}$ and either N is a name, or $N = \{N'\}_h$ for some N' and $h \notin kn(\sigma)$. For example, given $\sigma = [\{a\}_{hk}/x_1,\ k/x_2]$, then $core(\sigma, x_1) = \{a\}_h$ and $core(\sigma, x_2) = k$.

- Given a tuple $\widetilde{x} = x_{i \in I}$ and a tuple of indices $\widetilde{\jmath} = (j_1, \ldots, j_k) \subseteq I$, we let $\widetilde{x}[\widetilde{\jmath}]$ denote the tuple $(x_{j_1}, \ldots, x_{j_k})$. For instance, if $\widetilde{x} = (x_1, x_2, x_3)$ and $\widetilde{\jmath} = (3, 1, 1, 2)$ then $\widetilde{x}[\widetilde{\jmath}] = (x_3, x_1, x_1, x_2)$.

We are now set to give an alternative definition of equivalence on environments. The intuition behind the definition below is that, for any two equivalent environments, it should not be possible to tell apart two messages M_i and M'_i referenced by the same variable x_i, by (a) trying decryption with different keys, or by (b) format mismatch (name vs. compound ciphertext), or by (c) syntactic comparison.

Definition 3.3 (equivalence on environments: alternative definition) *Let* $\sigma = [M_i/x_i]_{i\in I}$ *and* $\sigma' = [M'_i/x_i]_{i\in I}$ *be two substitutions with the same domain. For each* $i \in I$*, let* $N_i = core(\sigma, x_i)$ *and* $N'_i = core(\sigma', x_i)$*, and let* $\widetilde{N} = N_{i\in I}$ *and* $\widetilde{N'} = N'_{i\in I}$*. We write* $\sigma \sim' \sigma'$*, if for each* $i \in I$ *the following three conditions hold:*

(a) for some tuple of indices $\widetilde{\jmath_i} \subseteq I$*, it holds that* $M_i = \{N_i\}_{\widetilde{N}[\widetilde{\jmath_i}]}$ *and* $M'_i = \{N'_i\}_{\widetilde{N'}[\widetilde{\jmath_i}]}$*;*

(b) $N_i \in \mathcal{N}$ *iff* $N'_i \in \mathcal{N}$*;*

(c) for each $j \in I$*, it holds that* $N_i = N_j$ *iff* $N'_i = N'_j$*.* ◇

As an example, $\sigma_1 = [b/x_1, c/x_2, \{b\}_k/x_3]$ and $\sigma_2 = [b/x_1, c/x_2, \{c\}_k/x_3]$ are equivalent. On the contrary, $\sigma_3 = \sigma_1[k/x_4]$ and $\sigma_4 = \sigma_3[k/x_4]$ are not equivalent, because $core(\sigma_3, x_3) = b = core(\sigma_3, x_1)$, while $core(\sigma_4, x_3) = c \neq core(\sigma_4, x_1)$, thus condition (c) is violated[2]. Also note that equivalent environments need not have the same $kn(\cdot)$: for instance, the environments $[\{b\}_{kh}/x_1, h/x_2]$ and $[\{b\}_{k'h'}/x_1, h'/x_2]$ are equivalent, though they have different knowledge. Environment pairs of this sort may arise when comparing two processes (this is due to the interplay between encryption and restriction – see Example 3.9).

The following theorem, whose proof can be found in Appendix A, allows us to freely interchange the use of $\sim$ and of $\sim'$ in the rest of the paper.

Theorem 3.4 (coincidence of $\sim$ and $\sim'$) *For any two substitutions* σ *and* σ'*, it holds that* $\sigma \sim \sigma'$ *if and only if* $\sigma \sim' \sigma'$*.*

We are now ready to define a trace-based preorder. Recall that bound names of s and u below are assumed to be fresh. A similar remark applies to μ and δ in Definition 3.8.

Definition 3.5 (trace preorder)
Let $\sigma_1 \sim \sigma_2$*. Given two processes* P *and* Q*, we write* $(\sigma_1, \sigma_2) \vdash P \ll Q$ *if whenever* $\sigma_1 \rhd P \stackrel{s}{\underset{u}{\Longrightarrow}} \sigma'_1 \rhd P'$ *then there are* s'*,* σ'_2 *and* Q' *such that* $\sigma_2 \rhd Q \stackrel{s'}{\underset{u}{\Longrightarrow}} \sigma'_2 \rhd Q'$ *and* $\sigma'_1 \sim \sigma'_2$*.* ◇

Note that, when comparing configurations, just the lower transition labels are considered, while the upper labels (in s and s') are ignored: we give them to help reading the definition. We revise below the example given in the Introduction.

Example 3.6 Define $\sigma = [a/x, b/y, c/z]$. Then $\sigma \rhd (\nu\, k)\overline{a}\{b\}_k$ and $\sigma \rhd (\nu\, k)\overline{a}\{c\}_k$ are $\ll$-equivalent. On the contrary, $\sigma \rhd (\nu\, k)\overline{a}\{b\}_k.\overline{a}k$ and $\sigma \rhd (\nu\, k)\overline{a}\{c\}_k.\overline{a}k$ are not related, because $\sigma[\{b\}_k/v, k/w] \not\sim \sigma[\{c\}_k/v, k/w]$, for any fresh v, w.

[2]In general, once $kn(\sigma_1)$ and $kn(\sigma_2)$ have been computed, $\sigma_1 \sim' \sigma_2$ can be very easily checked. In particular, the existential on (a) does not imply any search among the tuples $\widetilde{\jmath}$: given $i \in I$, we just choose any tuple $\widetilde{\jmath_i}$ s.t. $M_i = \{N_i\}_{\widetilde{N}[\widetilde{\jmath_i}]}$ and check whether $M'_i = \{N'_i\}_{\widetilde{N'}[\widetilde{\jmath_i}]}$. If this is the case then condition (a) is verified for i, otherwise we can immediately conclude that $\sigma \not\sim' \sigma'$: indeed, if condition (a) were validated by a different tuple $\widetilde{\jmath'}$, then we would get that $\widetilde{N}[\widetilde{\jmath_i}] = \widetilde{N}[\widetilde{\jmath'_i}]$, but $\widetilde{N'}[\widetilde{\jmath_i}] \neq \widetilde{N'}[\widetilde{\jmath'_i}]$, thus condition (c) would be violated.

Example 3.7 A subtler example. Consider

$$P \stackrel{\text{def}}{=} (\nu\, a, k)\overline{c}\{k\}_k.\overline{c}a \qquad \text{and} \qquad Q \stackrel{\text{def}}{=} (\nu\, a, k)\overline{c}\{k\}_{ak}.\overline{c}a.$$

It is easy to check that, for $\sigma \stackrel{\text{def}}{=} [c/\!\!/c]$, it holds both that $(\sigma, \sigma) \vdash P \ll Q$ and that $(\sigma, \sigma) \vdash Q \ll P$. The difference between P and Q is that Q's first message contains a private name (a) that is later disclosed to the environment: however the environment cannot detect this difference without the key k, which is never disclosed. Indeed, P and Q are may (and barbed) equivalent.

More examples will be given in Section 5. Let us switch to bisimulation. In what follows $\sigma \rhd P \stackrel{\widehat{\mu}}{\underset{\delta}{\Longrightarrow}} \sigma' \rhd P'$ stands for $\sigma \rhd P \stackrel{\mu}{\underset{\delta}{\Longrightarrow}} \sigma' \rhd P'$ if $\mu \neq \tau$, and for $\sigma \rhd P \Longrightarrow \sigma' \rhd P'$ if $\mu = \tau$ (the $\widehat{\cdot}$ defined here has of course nothing to do with the evaluation function on expressions defined in the previous section). We say that a pair of configurations $(\sigma_1 \rhd P, \sigma_2 \rhd Q)$ is *compatible* if σ_1 and σ_2 are equivalent. A relation $\mathcal{R}$ is *compatible* if it only contains compatible pairs of configurations. Given a binary relation $\mathcal{R}$, we write $(\sigma_1, \sigma_2) \vdash P\,\mathcal{R}\,Q$ if $(\sigma_1 \rhd P\,,\, \sigma_2 \rhd Q) \in \mathcal{R}$.

Definition 3.8 (weak bisimulation) *Let $\mathcal{R}$ be a binary compatible relation of configurations. We say that $\mathcal{R}$ is a* weak bisimulation *if whenever $(\sigma_1, \sigma_2) \vdash P\,\mathcal{R}\,Q$ and $\sigma_1 \rhd P \stackrel{\mu}{\underset{\delta}{\longmapsto}} \sigma_1' \rhd P'$ then there are μ', σ_2' and Q' such that $\sigma_2 \rhd Q \stackrel{\widehat{\mu'}}{\underset{\delta}{\Longrightarrow}} \sigma_2' \rhd Q'$ and $(\sigma_1', \sigma_2') \vdash P'\,\mathcal{R}\,Q'$, and the converse on the transitions of Q and P.* Bisimilarity*, written $\approx$, is the largest weak bisimulation relation.* $\diamond$

Example 3.9 This example shows that, when establishing process equivalence, pairs of equivalent environments with different $kn(\cdot)$ may arise, even when starting from a pair of identical environments. Consider

$$P \stackrel{\text{def}}{=} (\nu\, a, k)\overline{c}\{k\}_k.\overline{c}a.\overline{a}c \qquad \text{and} \qquad Q \stackrel{\text{def}}{=} (\nu\, a, k)\overline{c}\{k\}_{ak}.\overline{c}a.\overline{a}c$$

and let $\sigma \stackrel{\text{def}}{=} [c/\!\!/c]$. It is easy to see that $(\sigma, \sigma) \vdash P \approx Q$. In particular, the move

$$\sigma \rhd P \stackrel{(\nu\, k)\overline{c}\langle\{k\}_k\rangle}{\underset{c(x)}{\longmapsto}} \sigma[\{k\}_k/\!\!/x] \rhd (\nu\, a)\overline{c}a.\overline{a}c \stackrel{\text{def}}{=} P'$$

is matched by

$$\sigma \rhd Q \stackrel{(\nu\, k,a)\overline{c}\langle\{k\}_{ak}\rangle}{\underset{c(x)}{\longmapsto}} \sigma[\{k\}_{ak}/\!\!/x] \rhd \overline{c}a.\overline{a}c \stackrel{\text{def}}{=} Q'$$

where $(\sigma[\{k\}_k/\!\!/x], \sigma[\{k\}_{ak}/\!\!/x]) \vdash P' \approx Q'$. Thus, the move

$$\sigma[\{k\}_k/\!\!/x] \rhd P' \stackrel{(\nu\, b)\overline{c}\langle b\rangle}{\underset{c(y)}{\longmapsto}} \sigma[\{k\}_k/\!\!/x,\, b/\!\!/y] \rhd \overline{b}c \stackrel{\text{def}}{=} P''$$

(where we have alpha-renamed a into a fresh b) must be matched by

$$\sigma[\{k\}_{ak}/\!\!/x] \rhd Q' \stackrel{\overline{c}\langle a\rangle}{\underset{c(y)}{\longmapsto}} \sigma[\{k\}_{ak}/\!\!/x,\, a/\!\!/y] \rhd \overline{a}c \stackrel{\text{def}}{=} Q''$$

where $(\sigma[\{k\}_k/\!\!/x,\, b/\!\!/y], \sigma[\{k\}_{ak}/\!\!/x,\, a/\!\!/y]) \vdash P'' \approx Q''$. In particular, the process action $\overline{b}\langle c\rangle$ originating from P'' is matched by the process action $\overline{a}\langle c\rangle$ originating from Q''.

To end the section, we note that bisimilarity $\approx$ is strictly included in the trace preorder $\ll$. This fact holds for labelled transition systems in general, it is not specific to cryptography. As an example, it is easily checked that, if $V = \{a, b, c\}$, then

$$(\epsilon_V, \epsilon_V) \vdash P \stackrel{\text{def}}{=} \overline{a}a.(\nu\, z)((\overline{z}z.\overline{b}b \mid \overline{z}z.\overline{c}c) \mid z(x)) \ll (\nu\, z)((\overline{z}z.\overline{a}a.\overline{b}b \mid \overline{z}z.\overline{a}a.\overline{c}c) \mid z(x)) \stackrel{\text{def}}{=} Q$$

but

$$(\epsilon_V, \epsilon_V) \vdash P \not\approx Q\,.$$

Indeed, $\epsilon_V \rhd P$ has a $\overline{a}a$-move that $\epsilon_V \rhd Q$ cannot match.

4 Soundness and Completeness

In this section we show the agreement between the contextual semantics of Section 2 and the semantics based on the environment-sensitive lts of Section 3. More precisely, we will prove that:

- the trace preorder ($\ll$) coincides with the may-testing preorder ($\sqsubseteq_{\approx}$), and that:
- bisimilarity ($\approx$) is included in barbed equivalence ($\cong$), while the opposite inclusions holds for the class of structurally image-finite processes (defined later in this section).

The inclusions $\ll \subseteq \sqsubseteq_{\approx}$ and $\approx \subseteq \cong$ will be referred to as *soundness*, while the opposite inclusions will be referred to as *completeness*. There are a few basic ingredients for the proofs of soundness and completeness, which we list below. First, it is technically convenient to introduce a notion of structural equivalence, $\equiv$, in the same vein of [16].

Definition 4.1 (structural equivalence) Structural equivalence *is the least equivalence relation $\equiv$ over processes that is preserved by parallel composition and restriction, and satisfies the structural laws of [16], i.e.*

- the monoid laws for parallel composition*: $P|\mathbf{0} \equiv P$, $P|Q \equiv Q|P$ and $P|(Q|R) \equiv (P|Q)|R$,*
- the laws for restriction*: $(\nu\, b)\mathbf{0} \equiv \mathbf{0}$, $(\nu\, a)(\nu\, b)P \equiv (\nu\, b)(\nu\, a)P$ and $(\nu\, a)(P|Q) \equiv P|(\nu\, a)Q$ if $a \notin fn(P)$,*
- the law for replication*: $!P \equiv P \mid !P$,*

plus the law:

$$(\texttt{let}\ z = \zeta\ \texttt{in}\ P) \equiv P[\widehat{\zeta}/z] \quad \text{if } \widehat{\zeta} \neq \bot\,. \qquad \diamond$$

A property of structural equivalence that we shall widely use in the sequel is that $\equiv$ commutes with $\stackrel{\mu}{\longrightarrow}$, i.e.: if $P \equiv Q$ and $P \stackrel{\mu}{\longrightarrow} P'$ then there exists Q' such that $Q \stackrel{\mu}{\longrightarrow} Q'$ and $P' \equiv Q'$ (the proof goes by inspection of the rules; see also [16]).

The key to soundness is the following proposition, that relates equivalence on environments ($\sim$) to the (conventional) operational semantics of Table 2 (its proof can be found in Appendix B).

Proposition 4.2 *Consider two equivalent substitutions σ_1 and σ_2. Let R be any process or observer such that $fn(R) \subseteq dom(\sigma_1)$.*

1. *Suppose that $R\sigma_1 \xrightarrow{(\nu\,\widetilde{b})\overline{a}\langle M\rangle} R_1$. Then: (i) there is η s.t. $n(\eta) \subseteq dom(\sigma_1)$ and $\widehat{\eta\sigma_1} = a$; (ii) there are ζ and R' s.t. $fn(\zeta, R') \subseteq dom(\sigma_1) \cup \widetilde{b}$ and $M = \widehat{\zeta\sigma_1}$ and $R_1 \equiv R'\sigma_1$; (iii) it holds that $R\sigma_2 \xrightarrow{(\nu\,\widetilde{b})\overline{a'}\langle M'\rangle} R_2$, where $a' = \widehat{\eta\sigma_2}$, $M' = \widehat{\zeta\sigma_2}$ and $R_2 \equiv R'\sigma_2$.*
2. *Suppose that $R\sigma_1 \xrightarrow{a\,M} R_1$. Then: (i) there is η s.t. $n(\eta) \subseteq dom(\sigma_1)$ and $\widehat{\eta\sigma_1} = a$; (ii) taken any fresh y and $\sigma_1' \stackrel{\text{def}}{=} \sigma_1[M/y]$, there is R' s.t. $fn(R') \subseteq dom(\sigma_1')$ and $R_1 \equiv R'\sigma_1'$; (iii) for any M', it holds that $R\sigma_2 \xrightarrow{a'\,M'} R_2$, where $a' = \widehat{\eta\sigma_2}$ and $R_2 \equiv R'\sigma_2'$ and $\sigma_2' \stackrel{\text{def}}{=} \sigma_2[M'/y]$.*
3. *Suppose that $R\sigma_1 \xrightarrow{\mu} R_1$ with $\mu = \tau$ or $\mu = \omega$. Then: (i) there is R' s.t. $fn(R') \subseteq dom(\sigma_1)$ and $R_1 \equiv R'\sigma_1$; (ii) we have $R\sigma_2 \xrightarrow{\mu} R_2$, where $R_2 \equiv R'\sigma_2$.*

The main ingredient for completeness is the notion of *characteristic formula* of an environment σ, written ϕ_σ. The exact definition of ϕ_σ can be found in Appendix A. Here, we only wish to remind the properties that will be used below: it holds that $n(\phi_\sigma) \subseteq dom(\sigma)$ and that $\sigma \models \phi_\sigma$; moreover, the following crucial theorem (whose proof can be found in Appendix A) says that ϕ_σ characterizes all and only those environments σ' equivalent to σ.

Theorem 4.3 *Let σ and σ' be two substitutions such that $dom(\sigma) = dom(\sigma')$. We have that $\sigma \sim \sigma'$ if and only if $\sigma' \models \phi_\sigma$.*

We will now proceed to prove soundness and completeness, first for may-testing and then for barbed equivalence.

4.1 May-testing and trace semantics

Soundness

It is convenient to prove soundness of $\ll$ with respect to a notion more general than $\sqsubseteq_\approx$, which is introduced below.

Definition 4.4 (generalized may-testing) *For equivalent σ_1 and σ_2, we write $(\sigma_1, \sigma_2) \vdash P \sqsubseteq_\approx Q$ if, for each observer O with $fn(O) \subseteq dom(\sigma_1)$, $P \,|\, O\sigma_1 \stackrel{\omega}{\Longrightarrow}$ implies $Q \,|\, O\sigma_2 \stackrel{\omega}{\Longrightarrow}$.* $\diamond$

The above definition subsumes that of may preorder, as $P \sqsubseteq_\approx Q$ holds if and only if $(\epsilon_V, \epsilon_V) \vdash P \sqsubseteq_\approx Q$ for some $V \supseteq fn(P, Q)$. The 'only if' part of this statement is trivial. To see that the 'if' part is true, use the fact that, for any R and O, $R|O \stackrel{\omega}{\Longrightarrow}$ iff $(\nu\,\widetilde{b})(R|O) \equiv R|(\nu\,\widetilde{b})O \stackrel{\omega}{\Longrightarrow}$, where $\widetilde{b} = fn(O) - fn(R)$.

We need some properties of sequences of transitions. To state these properties we introduce some additional notions.

Notation 4.5 *Given two process actions μ and λ, we write $\mu \underline{compl}\, \lambda$ when $\mu = a\,M$ and $\lambda = (\nu\,\widetilde{b})\overline{a}\langle M\rangle$, or vice-versa, for some a, $\widetilde{b}$ and M. The notation extends to sequences of visible actions of the same length ($s \underline{compl}\, r$) as expected.*

Note that whenever $P \mid O \stackrel{\omega}{\Longrightarrow}$, then we can find s and r (possibly empty) s.t. $P \stackrel{s}{\Longrightarrow}$, $O \stackrel{r\omega}{\Longrightarrow}$ and $s \underline{compl}\, r$. Conversely, $P \stackrel{s}{\Longrightarrow}$ and $O \stackrel{r\omega}{\Longrightarrow}$ with $s \underline{compl}\, r$ can be composed to yield $P \mid O \stackrel{\omega}{\Longrightarrow}$.

Definition 4.6 *Given an environment action δ and an environment σ, $\widehat{\delta\sigma}$ is the process action defined as:*

$$\widehat{\delta\sigma} = \begin{cases} (\nu\, \widetilde{b})\overline{a}\langle M\rangle & \text{if } \delta = (\nu\, \widetilde{b})\overline{\eta}\langle\zeta\rangle,\ \widehat{\eta\sigma} = a \text{ and } M = \widehat{\zeta\sigma} \\ a\, M & \text{if } \delta = \eta(x),\ \widehat{\eta\sigma} = a \text{ and } M = x\sigma \\ \textit{undefined} & \textit{otherwise.} \end{cases}$$

Given a trace u with $n(u) \subseteq dom(\sigma)$, $\widehat{u\sigma}$ is defined as expected. $\diamond$

For proving the soundness theorem we make use of the following three auxiliary lemmas whose proofs can be found in Appendix B. The first lemma is a generalization of Proposition 4.2 to sequences of transitions.

Lemma 4.7 *Suppose $\sigma_1 \sim \sigma_2$ and let O be any observer such that $fn(O) \subseteq dom(\sigma_1)$. Suppose that $O\sigma_1 \stackrel{r}{\Longrightarrow} O_1$, where, for some u and σ_1' extending σ_1, it is $r = \widehat{u\sigma_1'}$. Then, there is O' with $fn(O') \subseteq dom(\sigma_1')$ such that $O_1 \equiv O'\sigma_1'$. Furthermore, for any σ_2' extending σ_2 and such that $\sigma_2' \sim \sigma_1'$, it holds that $O\sigma_2 \stackrel{r'}{\Longrightarrow} \equiv O'\sigma_2'$, with $r' = \widehat{u\sigma_2'}$.*

The next lemma gives a sufficient condition to infer the existence of a sequence of transitions $\sigma \rhd P \stackrel{s}{\underset{u}{\Longmapsto}} \sigma' \rhd P'$.

Lemma 4.8 *Consider σ, P and any observer O with $fn(O) \subseteq dom(\sigma)$. Suppose that $P \stackrel{s}{\Longrightarrow} P'$ and that $O\sigma \stackrel{r}{\Longrightarrow}$ with $s \underline{compl}\, r$. Then there are u and σ' extending σ such that $r = \widehat{u\sigma'}$ and $\sigma \rhd P \stackrel{s}{\underset{u}{\Longmapsto}} \sigma' \rhd P'$.*

Finally, a simple result relates the form of s to that of u in a sequence of transitions $\stackrel{s}{\underset{u}{\Longmapsto}}$.

Lemma 4.9 *Suppose that $\sigma \rhd P \stackrel{s}{\underset{u}{\Longmapsto}} \sigma' \rhd P'$. Then it holds that $P \stackrel{s}{\Longrightarrow} P'$, that σ' extends σ and that $s \underline{compl}\, r$, where $r = \widehat{u\sigma'}$.*

We can now state and prove the soundness theorem.

Theorem 4.10 (soundness of trace semantics) *If $(\sigma_1, \sigma_2) \vdash P \ll Q$ then $(\sigma_1, \sigma_2) \vdash P \sqsubseteq\!\!\!\sim Q$.*

Proof: Suppose that $(\sigma_1, \sigma_2) \vdash P \ll Q$ and let O be any observer with $fn(O) \subseteq dom(\sigma_1)$. Suppose that $P \mid O\sigma_1 \stackrel{\omega}{\Longrightarrow}$; we have to show that $Q \mid O\sigma_2 \stackrel{\omega}{\Longrightarrow}$. Since $P \mid O\sigma_1 \stackrel{\omega}{\Longrightarrow}$ we can find P', s and r such that

$$P \stackrel{s}{\Longrightarrow} P' \text{ and } O\sigma_1 \stackrel{r\omega}{\Longrightarrow} \text{ with } s \underline{compl}\, r$$

(as usual, we suppose that $bn(s,r)$ are fresh). Due to Lemma 4.8, we can find u and σ_1' extending σ_1 such that

$$\sigma_1 \rhd P \stackrel{s}{\stackrel{\Longrightarrow}{u}} \sigma_1' \rhd P' \text{ and } r = \widehat{u\sigma_1'}\,.$$

Thus, by hypothesis, there are s', σ_2' extending σ_2 and Q' such that

$$\sigma_2 \rhd Q \stackrel{s'}{\stackrel{\Longrightarrow}{u}} \sigma_2' \rhd Q' \text{ with } \sigma_1' \sim \sigma_2'\,.$$

Moreover, due to Lemma 4.9, it holds that $Q \stackrel{s'}{\Longrightarrow} Q'$ and $s' \underline{compl}\, r'$ where $r' = \widehat{u\sigma_2'}$. From Lemma 4.7 applied to $O\sigma_1 \stackrel{r\omega}{\Longrightarrow}$ and to σ_2', we get that $O\sigma_2 \stackrel{r'\omega}{\Longrightarrow}$. From this fact, $Q \stackrel{s'}{\Longrightarrow} Q'$ and $s' \underline{compl}\, r'$, we get the wanted $Q \mid O\sigma_2 \stackrel{\omega}{\Longrightarrow}$. □

Completeness

We begin by introducing some notation. We write $\langle\overline{\eta}\zeta\rangle.P$ instead of $\mathtt{let}\ z_1 = \eta\ \mathtt{in}\ (\mathtt{let}\ z_2 = \zeta\ \mathtt{in}\ \overline{z_1}z_2.P)$, and $\langle\eta(x)\rangle.P$ instead of $\mathtt{let}\ z = \eta\ \mathtt{in}\ z(x).P$ (z, z_1, and z_2 fresh). The *output-bound names* of δ, $obn(\delta)$, are defined as follows: $obn((\nu\,\widetilde{b})\overline{\eta}\zeta) = \widetilde{b}$ and $obn(\delta) = \emptyset$ if δ is not an output action. This notations extend to traces ($obn(u)$) as expected. Given u and σ, we say that u *is consistent with* σ if: (i) $n(u) \subseteq dom(\sigma)$, and (ii) σ extends $[\widetilde{b}/\widetilde{b}]$ where $\widetilde{b} = obn(u)$, and (iii) $\widehat{u\sigma}$ is defined.
Based on ϕ_σ, we can define a class of canonical observers $o(u,\sigma)$, depending on specific u and σ.

Definition 4.11 (canonical observers) *Consider u consistent with σ. The observers $o(u,\sigma)$ are defined by induction on u as follows:*

$$\begin{array}{rcl} o(\epsilon,\sigma) & \stackrel{\text{def}}{=} & \phi_\sigma\omega \\ o((\nu\,\widetilde{b})\overline{\eta}\langle\zeta\rangle\cdot u,\sigma) & \stackrel{\text{def}}{=} & (\nu\,\widetilde{b})\,\langle\overline{\eta}\zeta\rangle.o(u,\sigma) \\ o(\eta(x)\cdot u,\sigma) & \stackrel{\text{def}}{=} & \langle\eta(x)\rangle.o(u,\sigma). \end{array}$$ ◇

Note that $fn(o(u,\sigma)) \subseteq dom(\sigma) - bn(u)$. We need now two technical lemmas.

Lemma 4.12 *Consider u consistent with σ. Let σ_0 be the restriction of σ to $dom(\sigma) - bn(u)$.*

1. *We have $o(u,\sigma)\sigma_0 \stackrel{r\omega}{\Longrightarrow}$, where $r = \widehat{u\sigma}$.*
2. *Consider any $\sigma_0' \sim \sigma_0$. If $o(u,\sigma)\sigma_0' \stackrel{r\omega}{\Longrightarrow}$ then (up to renaming of bound names of r) $r = \widehat{u\sigma'}$ for some σ' extending σ_0' and s.t. $\sigma \sim \sigma'$.*

Proof: Both parts are proven by straightforward induction on u and relying on Theorem 4.3 for the base case. □

Lemma 4.13 *Consider P, σ and σ' extending σ. Suppose that $P \stackrel{s}{\Longrightarrow} P'$, with s $\underline{compl}$ $\widehat{u\sigma'}$, for some u consistent with σ' and s.t. $bn(u) = dom(\sigma') - dom(\sigma)$. Then $\sigma \rhd P \stackrel{s}{\stackrel{\Longrightarrow}{u}} \sigma' \rhd P'$.*

PROOF: A straightforward induction based on u which relies on Proposition 4.2. □

We are now ready to prove completeness.

Theorem 4.14 (completeness for may-testing) *If* $(\sigma_1, \sigma_2) \vdash P \sqsubseteq\!\!\!\!\sim Q$ *then* $(\sigma_1, \sigma_2) \vdash P \ll Q$.

PROOF: Suppose that $(\sigma_1, \sigma_2) \vdash P \sqsubseteq\!\!\!\!\sim Q$ and that $\sigma_1 \rhd P \stackrel{s}{\underset{u}{\Longmapsto}} \sigma_1' \rhd P'$, where as usual $bn(s,u)$ are taken fresh. We have to show that for some s', σ_2' and Q', it holds that $\sigma_2 \rhd Q \stackrel{s'}{\underset{u}{\Longmapsto}} \sigma_2' \rhd Q'$ and $\sigma_1' \sim \sigma_2'$.

Due to Lemma 4.9, we know that $s \underline{compl}\, r \stackrel{\text{def}}{=} \widehat{u\sigma_1'}$. Furthermore, it is easy to show that u is consistent with σ_1'. Since σ_1 is the restriction of σ_1' to $dom(\sigma_1') - bn(u)$, by virtue of Lemma 4.12(1) we get that $o(u,\sigma_1')\sigma_1 \stackrel{r\omega}{\Longrightarrow}$. From this, $P \stackrel{s}{\Longrightarrow} P'$ and $s \underline{compl}\, r$ we get that $P \mid o(u,\sigma_1')\sigma_1 \stackrel{\omega}{\Longrightarrow}$. Thus, by hypothesis, we have also $Q \mid o(u,\sigma_1')\sigma_2 \stackrel{\omega}{\Longrightarrow}$. This implies that there are s', Q' and r' such that

$$Q \stackrel{s'}{\Longrightarrow} Q' \text{ and } o(u,\sigma_1')\sigma_2 \stackrel{r'\omega}{\Longrightarrow} \text{ with } s' \underline{compl}\, r' .$$

From this and Lemma 4.12(2), we obtain that $r' = \widehat{u\sigma_2'}$, for some σ_2' extending σ_2 and s.t. $\sigma_1' \sim \sigma_2'$. From this fact, $Q \stackrel{s'}{\Longrightarrow} Q'$ and Lemma 4.13 we get the wanted $\sigma_2 \rhd Q \stackrel{s'}{\underset{u}{\Longmapsto}} \sigma_2' \rhd Q'$. □

4.2 Barbed equivalence and bisimilarity

Soundness

It is convenient to generalize Definition 2.3 of barbed equivalence.

Definition 4.15 (generalized barbed equivalence)
Let $\sigma_1 = [M_i/x_i]_{i\in I}$ *and* $\sigma_2 = [M_i'/x_i]_{i\in I}$ *be equivalent substitutions. For each* $i \in I$, *let* $N_i = core(\sigma_1, x_i)$ *and* $N_i' = core(\sigma_2, x_i)$. *A binary relation* $\mathcal{S}$ *of processes is a* (σ_1,σ_2)-barbed bisimulation *if whenever* $P\mathcal{S}Q$ *then:*

1. *for each* P', *if* $P \stackrel{\tau}{\longrightarrow} P'$ *then there is* Q' *such that* $Q \Longrightarrow Q'$ *and* $P'\mathcal{S}Q'$, *and*
2. *for each* $i \in I$, *if* $P \downarrow N_i$ *then* $Q \Downarrow N_i'$

and the converse on the transitions and commitments of Q *and* P.
Two processes P *and* Q *are* (σ_1,σ_2)-barbed bisimilar, *written* $(\sigma_1,\sigma_2) \vdash P \dot{\cong} Q$, *if* (P,Q) *belongs to the largest* (σ_1,σ_2)*-barbed bisimulation.*
Two processes P *and* Q *are* (σ_1,σ_2)-barbed equivalent, *written* $(\sigma_1,\sigma_2) \vdash P \cong Q$, *if for all* R *with* $fn(R) \subseteq dom(\sigma_1)$ *we have that* $(\sigma_1,\sigma_2) \vdash P \mid R\sigma_1 \dot{\cong} Q \mid R\sigma_2$. ◇

Note the differences of this definition from Definition 2.3. First, the 'barbs' (the $\downarrow$ predicate) are only checked relatively to those names that are known to the environments, which are precisely those N_i's and those N_i''s that are in $\mathcal{N}$. Second, for each $i \in I$, names N_i

and N_i' are not required to be the same, but are just required to be the 'cores' of the same environment entry x_i. Most important, $\stackrel{\cdot}{\cong}$ is just closed under those contexts that can be obtained via instantiation with σ_1 and σ_2. Of course, $P \cong Q$ holds iff $(\epsilon_V, \epsilon_V) \vdash P \cong Q$, for some V containing $fn(P, Q)$.

The purely co-inductive formulation of $\approx$ given in Definition 3.8 gives us a powerful proof technique when proving equalities between two processes: it is sufficient to exhibit *any* bisimulation relation containing the given pair. This technique can be enhanced using the so called *up-to* techniques (similar to those in, e.g., [21, 10]), which often permit to reduce the size of the relation to exhibit. We introduce below some useful up-to techniques, which will be used in later proofs and examples. *Up to structural equivalence* allows one to freely identify structurally equivalent processes; *up to weakening* permits discarding environment entries, while *up to contraction* permits adding redundant (hence harmless) entries to the environments. Finally, *up to restriction* and *up to parallel composition* permit cutting away top-level restrictions and common parallel contexts, respectively, in process derivatives.

Definition 4.16 (up-to techniques) *Given a compatible relation $\mathcal{R}$, define $\mathcal{R}_t$, for $t \in \{s, w, c, r, p\}$, as the least binary relations over configurations that satisfy the following rules:*

- *Up to structural equivalence*

$$\frac{P' \equiv P,\ Q' \equiv Q \text{ and } (\sigma_1, \sigma_2) \vdash P' \,\mathcal{R}\, Q'}{(\sigma_1, \sigma_2) \vdash P \,\mathcal{R}_s\, Q}.$$

- *Up to weakening:*

$$\frac{(\sigma_1[\widetilde{M}/\widetilde{x}], \sigma_2[\widetilde{M'}/\widetilde{x}]) \vdash P \,\mathcal{R}\, Q}{(\sigma_1, \sigma_2) \vdash P \,\mathcal{R}_w\, Q}.$$

- *Up to contraction:*

$$\frac{(\sigma_1, \sigma_2) \vdash P \,\mathcal{R}\, Q,\ \bot \notin \widetilde{\sigma_1(\zeta)} \text{ and } n(\widetilde{\zeta}) - dom(\sigma_1) \text{ are fresh for } \sigma_1,\ \sigma_2,\ P \text{ and } Q}{(\sigma_1[\widetilde{\sigma_1(y)}/\widetilde{y}], \sigma_2[\widetilde{\sigma_2(y)}/\widetilde{y}]) \vdash P \,\mathcal{R}_c\, Q}.$$

- *Up to restriction:*

$$\frac{\widetilde{h} \cap n(\sigma_1) = \emptyset\ ,\ \widetilde{k} \cap n(\sigma_2) = \emptyset \text{ and } (\sigma_1, \sigma_2) \vdash P \,\mathcal{R}\, Q}{(\sigma_1, \sigma_2) \vdash (\nu\, \widetilde{h}) P \,\mathcal{R}_r\, (\nu\, \widetilde{k}) Q}.$$

- *Up to parallel composition:*

$$\frac{A \equiv P \mid R\sigma_1,\ B \equiv Q \mid R\sigma_2 \text{ and } (\sigma_1, \sigma_2) \vdash P \,\mathcal{R}\, Q \text{ and } fn(R) \subseteq dom(\sigma_1)}{(\sigma_1, \sigma_2) \vdash A \,\mathcal{R}_p\, B}.$$

A relation $\mathcal{R}$ is a weak bisimulation up to structural equivalence *if $\mathcal{R}$ satisfies the definition of weak bisimulation (Definition 3.8), but with the condition on the derivatives '$(\sigma_1', \sigma_2') \vdash P' \,\mathcal{R}\, Q'$' replaced by the weaker '$(\sigma_1', \sigma_2') \vdash P' \,\mathcal{R}_s\, Q'$'. Weak bisimulation* up to weakening, contraction, restriction, *and* parallel composition *are defined similarly.* $\diamond$

Thus an up-to technique t is essentially a functional $(\cdot)_t$ from compatible relations to compatible relations. We say that an up-to technique t is *sound* if, whenever $\mathcal{R}$ is a bisimulation up to t, then $\mathcal{R} \subseteq \approx$. Our first task is therefore to prove that the techniques we have defined above are sound. Next, it should be obvious that different up to techniques can be combined to get new techniques: as an example, weak bisimulation up parallel composition and contraction is defined by replacing '$(\sigma_1', \sigma_2') \vdash P' \mathcal{R} Q'$' with '$(\sigma_1', \sigma_2') \vdash P' (\mathcal{R}_p)_c Q'$' in Definition 3.8. Formally, a combination of techniques is a composition of the corresponding functionals (see [21]). The results of [21] ensure that, if the techniques we have introduced in Definition 4.16 are sound, then any combination of them is sound too[3]. The next proposition, whose proof can be found in Appendix B, states soundness of our up-to techniques.

Proposition 4.17 *Let $\mathcal{R}$ be a weak bisimulation up to structural equivalence (resp. weakening, contraction, restriction, parallel composition). Then $\mathcal{R} \subseteq \mathcal{R}_s \subseteq \approx$ (resp. $\mathcal{R} \subseteq \mathcal{R}_t \subseteq \approx$, for $t = w, c, r, p$).*

We are now ready to prove the soundness theorem.

Theorem 4.18 (soundness of weak bisimilarity) *Let P and Q be processes and σ_1 and σ_2 be equivalent substitutions. If $(\sigma_1, \sigma_2) \vdash P \approx Q$ then $(\sigma_1, \sigma_2) \vdash P \cong Q$.*

PROOF: Note that $\approx$ is trivially a weak bisimulation up to parallel composition, thus $(\approx)_p \subseteq \approx$ due to Proposition 4.17. Hence $(\sigma_1, \sigma_2) \vdash P \approx Q$ implies $(\sigma_1, \sigma_2) \vdash P \mid R\sigma_1 \approx Q \mid R\sigma_2$ for each R with $fn(R) \subseteq dom(\sigma_1)$. Since $\approx$ is finer than $\dot{\cong}$, this fact implies the wanted $(\sigma_1, \sigma_2) \vdash P \cong Q$. □

Completeness

We shall prove completeness of $\approx$ relatively to a class of processes that have an image-finiteness property (defined below). This property makes the proof relatively simple (and not far from, e.g., the proof for asynchronous bisimilarity given in [3]). On the other hand, the class of processes enjoying the property is broad enough to ensure that $\approx$ is a fairly general proof technique. At present, we do not know whether the proof can be extended to the full language.

Formally, a process P is *structurally image-finite* if for each visible trace s, the set of equivalence classes $\{P' : P \stackrel{s}{\Longrightarrow} P'\}/\equiv$ is finite. Note that this notion is slightly more general than the usual image-finiteness (as considered, e.g., in [3]): this is due to our use of $\equiv$ to quotient the set of s-derivatives. As an example, process $P \stackrel{\text{def}}{=} !\, a(x).\overline{b}x$ is structurally image-finite but not image-finite: indeed, for each s, P has infinitely many s-derivatives, which are however finite up to structural equivalence (use the law $!P \mid P \equiv !P$). On the contrary, process $Q \stackrel{\text{def}}{=} !\,(\tau.\overline{a} + \tau.\overline{b})$ is not structurally image-finite (hence not image-finite): for instance, for any $n, m \geq 0$, Q has an ϵ-derivative of the form $\overline{a} \mid \cdots \mid \overline{a} \mid \overline{b} \mid \cdots \mid \overline{b} \mid Q$, with n $\overline{a}$'s and m $\overline{b}$'s in parallel.

[3]Technically, our techniques enjoy the 'respectfulness' property of [21] w.r.t. the transition relation $\stackrel{\mu}{\underset{\delta}{\longmapsto}}$.

Next, it is convenient to introduce a chain of relations $\approx_i$, $i \geq 0$, which are used to approximate bisimilarity $\approx$ over configurations. In the sequel, $|E|$ is used to denote the syntactic size of some term E. Given any σ, we let $rk(\sigma) \stackrel{\text{def}}{=} |\phi_\sigma|$.

Definition 4.19 *The binary relations $\approx_i$ over configurations are defined by induction on i as follows:*

- $(\sigma_1, \sigma_2) \vdash P \approx_0 Q$ *if* $\sigma_1 \sim \sigma_2$;
- $(\sigma_1, \sigma_2) \vdash P \approx_i Q$, *for* $i > 0$ *and* $\sigma_1 \sim \sigma_2$, *if whenever* $\sigma_1 \rhd P \stackrel{\mu}{\underset{\delta}{\longmapsto}} \sigma_1' \rhd P'$ *with* $|\delta| \leq i$ *and* $rk(\sigma_1') \leq i$, *then there are* μ', σ_2' *and* Q' *s.t.* $\sigma_2 \rhd Q \stackrel{\hat{\mu}}{\underset{\delta}{\Longmapsto}} \sigma_2' \rhd Q'$ *and* $rk(\sigma_2') \leq i$ *and* $(\sigma_1', \sigma_2') \vdash P' \approx_{i-1} Q'$, *and the converse on the transitions of Q and P.*

We let $\approx_\omega \stackrel{\text{def}}{=} \cap_{i \geq 0} \approx_i$. $\diamond$

The following is a variation on a standard result for bisimulation (see e.g. [15, 20]).

Lemma 4.20 *Let P and Q be structurally image-finite processes. Then $(\sigma_1, \sigma_2) \vdash P \approx Q$ if and only if $(\sigma_1, \sigma_2) \vdash P \approx_\omega Q$.*

We show now that $\cong$ implies $\approx_\omega$, from which completeness of $\approx$ for structurally image-finite processes will follow. In order to do this, we exploit the formula ϕ_σ and define a class of canonical contexts $R_{i,\sigma}$, depending on some $i \geq 0$ and σ, which can be used to test whether two configurations are related by $\approx_i$. In what follows, we shall use some of the process notation introduced at the beginning of the subsection on completeness for may testing. Furthermore, we shall sometimes omit the object part of action prefixes, writing e.g. $\overline{c}$ instead of $\overline{c}x$, when x is not relevant. We will let $\tau.P$ denote the process $(\nu\, c)(c.P \mid \overline{c})$ $(c \notin fn(P))$ and, for any finite set of processes $\{P_1, \ldots, P_k\}$, we will let $\sum \{P_1, \ldots, P_k\}$ denote the process $P_1 + \cdots + P_k$ (the exact way the summands are arranged does not matter).

Definition 4.21 (canonical contexts) *Define the processes $R_{i,\sigma}$, for $i \geq 0$, by induction on i as follows. $R_{0,\sigma} \stackrel{\text{def}}{=} \mathbf{0}$ and, for $i > 0$:*

$$
\begin{aligned}
R_{n,\sigma} &\stackrel{\text{def}}{=} \textstyle\sum \{\, R_\eta^{inp} + R_\eta^{out} \;:\; n(\eta) \subseteq dom(\sigma),\, \widehat{\eta\sigma} \in \mathcal{N} \text{ and } |\eta| < i \,\} + R_\epsilon + \overline{e}_i, \quad \textit{where:} \\
R_\eta^{inp} &\stackrel{\text{def}}{=} \langle \eta(x) \rangle.\textstyle\sum \{\, \phi_{\sigma'}\big(\overline{f}_{\eta,\phi_{\sigma'},i} + \tau.R_{i-1,\sigma'}\big) \;:\; \text{there is } M \text{ s.t. } \sigma' = \sigma[{}^{M}\!/_{x}], \text{ and } rk(\sigma') \leq i \,\} \\
R_\eta^{out} &\stackrel{\text{def}}{=} \textstyle\sum \{\, (\nu\, \widetilde{b})\langle \overline{\eta}\zeta \rangle.\big(\overline{g}_{\eta,\zeta,i} + \tau.R_{i-1,\sigma'}\big) \;:\; fn((\nu\, \widetilde{b})\overline{\eta}\langle\zeta\rangle) \subseteq dom(\sigma), \sigma' = \sigma[{}^{\widetilde{b}}\!/_{\widetilde{b}}] \text{ and } |\zeta| < i \,\} \\
R_\epsilon &\stackrel{\text{def}}{=} \tau.\big(\overline{h}_i + \tau.R_{i-1,\sigma}\big)
\end{aligned}
$$

where the names e_j, $f_{\eta,\phi_{\sigma'},j}$, $g_{\eta,\zeta,j}$ and h_j $(0 \leq j \leq i)$ are all distinct and fresh. $\diamond$

Note that, in the above definition, the sum in R_η^{inp} is finite, because, for fixed $\widetilde{x}$ and i, there are finitely many ϕ_σ's and $R_{i,\sigma}$ s.t. $dom(\sigma) = \widetilde{x}$ and $rk(\sigma) \leq i$ (this is formally proven by induction on i). The sum in R_η^{out} is finite as well, because actions $(\nu\, \widetilde{b})\overline{\eta}\langle\zeta\rangle$ are considered up to alpha-equivalence. Note also that $fn(R_{i,\sigma}) \subseteq dom(\sigma) \cup \widetilde{l}$, where $\widetilde{l}$ is the set of all names e_j, $f_{\eta,\phi_{\sigma'},j}$, $g_{\eta,\zeta,j}$ and h_j $(0 \leq j \leq i)$ occurring in $R_{i,\sigma}$.

An easy lemma on characteristic formulae (the proof can be found in Appendix B).

Lemma 4.22 *If $\sigma \sim \sigma'$ then $rk(\sigma) = rk(\sigma')$.*

We are now ready to prove completeness.

Theorem 4.23 (completeness for barbed equivalence) *Let P and Q be structurally image-finite processes. If $(\sigma_1, \sigma_2) \vdash P \cong Q$ then $(\sigma_1, \sigma_2) \vdash P \approx Q$.*

PROOF: By virtue of Lemma 4.20, it is sufficient to prove that for each $i \geq 0$ it holds $(\sigma_1, \sigma_2) \vdash P \approx_i Q$. Consider R_{i,σ_1} and let $\rho_1 \stackrel{\text{def}}{=} \sigma_1[\widetilde{l}/\widetilde{l}]$ and $\rho_2 \stackrel{\text{def}}{=} \sigma_2[\widetilde{l}/\widetilde{l}]$, where $\widetilde{l}$ is the set of all names e_j, $f_{\eta,\phi_{\sigma'},j}$, $g_{\eta,\zeta,j}$ and h_j $(0 \leq j \leq i)$ occurring in R_{i,σ_1} (we suppose that $\widetilde{l}$ has been chosen fresh for σ_1, σ_2, P and Q). We prove that if $(\rho_1, \rho_2) \vdash (P \mid R_{i,\sigma_1}\sigma_1) \dot{\cong} (Q \mid R_{i,\sigma_1}\sigma_2)$, then $(\sigma_1, \sigma_2) \vdash P \approx_i Q$. From this fact and $(\sigma_1, \sigma_2) \vdash P \cong Q$ the thesis will follow (note that $(\sigma_1, \sigma_2) \vdash P \cong Q$ trivially implies $(\rho_1, \rho_2) \vdash P \cong Q$).

We proceed by induction on i. The case $i = 0$ is trivial, thus suppose $i > 0$. We only consider the case of a (C-INP) transition, that is:

$$\sigma_1 \rhd P \stackrel{(\nu\,\widetilde{b})\overline{a}\langle M\rangle}{\underset{\eta(x)}{\longmapsto}} \sigma_1' \rhd P' \tag{1}$$

with $|\eta(x)| \leq i$, $\sigma_1' = \sigma_1[M/x]$ and $rk(\sigma_1') \leq i$, as the other cases are similar or easier. We show the existence of a transition of $\sigma_2 \rhd Q$ that matches this one. From (1) above, we can infer that:

$$P \mid R_{i,\sigma_1}\sigma_1 \xrightarrow{\tau} (\nu\,\widetilde{b})\big(P' \mid (\phi_{\sigma_1'}(\overline{f}_{\eta,\phi_{\sigma_1'},i} + \tau.R_{i-1,\sigma_1'}))\sigma_1'\big) \stackrel{\text{def}}{=} A.$$

Since $(\rho_1, \rho_2) \vdash P|R_{i,\sigma_1}\sigma_1 \dot{\cong} Q|R_{i,\sigma_1}\sigma_2$ and $A \downarrow f_{\eta,\phi_{\sigma_1'},i}$ we deduce the existence of a transition

$$Q \mid R_{i,\sigma_1}\sigma_2 \Longrightarrow (\nu\,\widetilde{b}')\big(Q' \mid (\phi_{\sigma_1'}(\overline{f}_{\eta,\phi_{\sigma_1'},i} + \tau.R_{i-1,\sigma_1'}))\sigma_2'\big) \stackrel{\text{def}}{=} B$$

with $(\rho_1, \rho_2) \vdash A \dot{\cong} B$ and $B \downarrow f_{\eta,\phi_{\sigma_1'},i}$, where $\sigma_2' = \sigma_2[M'/x]$, for some M' s.t. $Q \xrightarrow{(\nu\,\widetilde{b}')\overline{a'}\langle M'\rangle} Q'$ $(a' = \widehat{\eta\sigma_2})$. Hence

$$\sigma_2 \rhd Q \stackrel{(\nu\,\widetilde{b}')\overline{a'}\langle M'\rangle}{\underset{\eta(x)}{\Longmapsto}} \sigma_2' \rhd Q'. \tag{2}$$

Moreover, since $B \downarrow f_{\eta,\phi_{\sigma_1'},i}$, it holds that $\sigma_2' \models \phi_{\sigma_1'}$.

Now, from $A \xrightarrow{\tau} \equiv (\nu\,\widetilde{b})\big((P' \mid R_{i-1,\sigma_1'})\sigma_1'\big) \stackrel{\text{def}}{=} A'$ we deduce that $B \Longrightarrow B'$ with $(\rho_1, \rho_2) \vdash A' \dot{\cong} B'$. Since $A' \downarrow e_{i-1}$, it must hold $B' \Downarrow e_{i-1}$, hence it must be $B' \equiv (\nu\,\widetilde{b}')\big((Q''|R_{i-1,\sigma_1'})\sigma_2'\big)$, with $Q' \Longrightarrow Q''$. We can strip the restrictions $(\nu\,\widetilde{b})$ and $(\nu\,\widetilde{b}')$ away from $(\rho_1, \rho_2) \vdash A' \dot{\cong} B'$ and deduce that

$$(\rho_1, \rho_2) \vdash (P' \mid R_{i-1,\sigma_1'}\sigma_1') \dot{\cong} (Q'' \mid R_{i-1,\sigma_1'}\sigma_2')$$

which, by induction, implies that $(\sigma_1', \sigma_2') \vdash P' \approx_{i-1} Q''$. Now, from $Q' \Longrightarrow Q''$ and transition (2), we deduce

$$\sigma_2 \rhd Q \stackrel{(\nu\,\widetilde{b}')\overline{a'}\langle M'\rangle}{\underset{\eta(x)}{\Longmapsto}} \sigma_2' \rhd Q''\,.$$

We show that this transition matches (1). Indeed, we have that: $\sigma_1' \sim \sigma_2'$ (by $\sigma_2' \models \phi_{\sigma_1'}$ and Theorem 4.3), that $rk(\sigma_2') = rk(\sigma_1') \leq i$ (Lemma 4.22) and that $(\sigma_1', \sigma_2') \vdash P' \approx_{i-1} Q''$. □

5 Applications

In this section we first give some properties which are useful when reasoning on cryptographic processes and then use them in a few examples.

5.1 Some useful laws

We start by stating some simple properties.

Proposition 5.1 *Let* $\mathbf{rel} \in \{\approx, \ll\}$.

- (Reflexivity) *For any* σ *and* P, $(\sigma, \sigma) \vdash P \,\mathbf{rel}\, P$.
- (Transitivity) *If* $(\sigma_1, \sigma_2) \vdash P \,\mathbf{rel}\, Q$ *and* $(\sigma_2, \sigma_3) \vdash Q \,\mathbf{rel}\, R$ *then* $(\sigma_1, \sigma_3) \vdash P \,\mathbf{rel}\, R$.
- (Weakening) *Suppose that* $(\sigma_1[M/x], \sigma_2[N/x]) \vdash P \,\mathbf{rel}\, Q$. *Then* $(\sigma_1, \sigma_2) \vdash P \,\mathbf{rel}\, Q$.
- (Contraction) *Suppose that* $(\sigma_1, \sigma_2) \vdash P \,\mathbf{rel}\, Q$ *and consider any* ζ *such that* $n(\zeta) \subseteq dom(\sigma_1)$ *and* $\widehat{\zeta\sigma_1} \neq \bot$. *Then* $(\sigma_1[\widehat{\zeta\sigma_1}/x], \sigma_2[\widehat{\zeta\sigma_2}/x]) \vdash P \,\mathbf{rel}\, Q$.
- (Structural equivalence) *Suppose that* $P \equiv Q$. *Then for any* σ $(\sigma, \sigma) \vdash P \,\mathbf{rel}\, Q$.

Proof: Reflexivity and Transitivity and Structural equivalence are trivial. The other cases are consequences of Proposition 4.17 for $\approx$. For $\ll$, the proof becomes trivial when one switches to the original definition of $\underset{\sim}{\sqsubset}$; for contraction, note that $O\sigma_i[\widehat{\zeta\sigma_i}/x] \equiv (\mathtt{let}\ x = \zeta\ \mathtt{in}\ O)\sigma_i$ for $i = 1, 2$ and $fn(O) \subseteq dom(\sigma_i)$. □

Some congruence laws are listed in Table 4. These laws are very useful (especially (C-Par) and (C-Res)) because they permit a kind of compositional reasoning, as we shall see in later examples in this section.

Proposition 5.2 *The laws listed in Table 4 are correct.*

Proof: The proof for (C-Inp) and (C-Out) is trivial. Laws (C-Par) and (C-Res) are a consequence of Proposition 4.17 in the case of $\approx$. In the case of $\ll$, the proof becomes trivial if one switches to the original definition $\underset{\sim}{\sqsubset}$. □

We shall also need a few rules to reason on environments. They are given in the following two lemmas (whose proofs can be found in Appendix A). The first lemma characterizes $kn(\sigma)$ in terms of the expressions that can be formed using the variables in $dom(\sigma)$.

Lemma 5.3 *Let* σ *be an environment. Then* $kn(\sigma) = \{\widehat{\zeta\sigma} \in \mathcal{N} : n(\zeta) \subseteq dom(\sigma)\}$.

The next lemma is about the effect of evaluating the same expression ζ under two equivalent environments, σ and σ'.

Lemma 5.4 *Let* $\sigma = [M_i/x_i]_{i\in I}$ *and* $\sigma' = [M'_i/x_i]_{i\in I}$ *be two substitutions such that* $\sigma \sim' \sigma'$. *Define* $\widetilde{N} = core(\sigma, x_i)_{i\in I}$ *and* $\widetilde{N}' = core(\sigma', x_i)_{i\in I}$. *For each* ζ *s.t.* $n(\zeta) \subseteq \widetilde{x}$, *either*

(a) $\widehat{\zeta\sigma} = \widehat{\zeta\sigma'} = \bot$, *or*

Let **rel** $\in \{\approx, \ll\}$.

(C-Inp) Suppose that for all ζ such that $\widetilde{y} \stackrel{\text{def}}{=} (n(\zeta) - dom(\sigma_1))$ are fresh and $\widehat{\zeta\sigma_1} \neq \bot$ it holds: $(\sigma_1[\widetilde{y}/\widetilde{y}], \sigma_2[\widetilde{y}/\widetilde{y}]) \vdash P[\widehat{\zeta\sigma_1}/x]$ **rel** $Q[\widehat{\zeta\sigma_2}/x]$.
Suppose $a_i = \widehat{\eta\sigma_i}$ $(i = 1, 2)$ with $n(\eta) \subseteq dom(\sigma_1)$.
Then $(\sigma_1, \sigma_2) \vdash a_1(x).P$ **rel** $a_2(x).Q$.

(C-Out) Suppose that $(\sigma_1[M_1/x], \sigma_2[M_2/x]) \vdash P$ **rel** Q and that $a_i = \widehat{\eta\sigma_i}$ $(i = 1, 2)$ with $n(\eta) \subseteq dom(\sigma_1)$.
Then $(\sigma_1[M_1/x], \sigma_2[M_2/x]) \vdash \overline{a}_1 M_1.P$ **rel** $\overline{a}_2 M_2.Q$.

(C-Par) Suppose that $fn(R) \subseteq dom(\sigma_1)$ and $(\sigma_1, \sigma_2) \vdash P$ **rel** Q .
Then $(\sigma_1, \sigma_2) \vdash P \mid R\sigma_1$ **rel** $Q \mid R\sigma_2$.

(C-Res) Suppose that $(\sigma_1, \sigma_2) \vdash P$ **rel** Q, that $\widetilde{k} \cap n(\sigma_1) = \emptyset$ and that $\widetilde{h} \cap n(\sigma_2) = \emptyset$.
Then $(\sigma_1, \sigma_2) \vdash (\nu\, \widetilde{k})P$ **rel** $(\nu\, \widetilde{h})Q$.

Table 4: Some congruence rules for $\ll$ and $\approx$.

(b) there are $i \in I$ and a tuple $\widetilde{\jmath} \subseteq I$ such that: $\widehat{\zeta\sigma} = \{N_i\}_{\widetilde{N}[\widetilde{\jmath}]}$ and $\widehat{\zeta\sigma'} = \{N'_i\}_{\widetilde{N'}[\widetilde{\jmath}]}$.

We end this subsection with a small example (borrowed from [6]) that shows the use of our congruence laws.

Example 5.5 Let us consider the processes $P \stackrel{\text{def}}{=} (\nu\, k)\overline{c}\{d\}_k.\, c(x).\, [x = k]\overline{c}\{d\}_k$ and $Q \stackrel{\text{def}}{=} (\nu\, k)\overline{c}\{d\}_k.\, c(x)$. Process P creates a private key k, sends d encrypted under k, listens for an input, and if it receives k then resends $\{d\}_k$. Process Q behaves like P but, after the reception of one message, it becomes stuck. Since k is a private key that is never disclosed to the environment, P will never receive k back at c, as a consequence the matching $[x = k]$ will never become true: therefore P and Q should be considered as equivalent. Let $V = fn(P, Q) = \{c, d\}$; we want to show that $(\epsilon_V, \epsilon_V) \vdash P \simeq Q$.

We can prove that $(\epsilon_V, \epsilon_V) \vdash Q \ll P$ by simply noting that traces of Q are also traces of P. To prove that $(\epsilon_V, \epsilon_V) \vdash P \ll Q$, let z be any fresh name and let $\sigma \stackrel{\text{def}}{=} \epsilon_V[\{d\}_k/z]$. The crucial step is showing that

$$(\sigma, \sigma) \vdash c(x).\, [x = k]\overline{c}\{d\}_k \ll c(x).$$

Indeed, for any ζ with $\widetilde{y} \stackrel{\text{def}}{=} (n(\zeta) - dom(\sigma))$ fresh, it holds that $\widehat{\zeta\sigma[\widetilde{y}/\widetilde{y}]} = \widehat{\zeta\sigma} \neq k$ (because of $k \notin kn(\sigma)$ and of Lemma 5.3), hence we have: $(\sigma[\widetilde{y}/\widetilde{y}], \sigma[\widetilde{y}/\widetilde{y}]) \vdash [\widehat{\zeta\sigma} = k]\overline{c}\{d\}_k \ll \mathbf{0}$. The thesis for this step follows by using (C-Inp). Now, using (C-Out), we have that

$$(\sigma, \sigma) \vdash \overline{c}\{d\}_k.\, c(x).\, [x = k]\overline{c}\{d\}_k \ll \overline{c}\{d\}_k.\, c(x).$$

The thesis follows from this fact, using first weakening and then (C-Res).

5.2 Secure channels implementation

In the following examples, we show the use of our framework for proving security properties of communication protocols. In the same vein of [1, 4], the idea is that of implementing communication on secure (private) channels by means of encrypted communication on public channels. Let us consider the π-calculus process:

$$P \stackrel{\text{def}}{=} (\nu\, c)(\overline{c}d \mid c(z).\, R)$$

where c does not occur in R. Process P creates a private channel c which is used to transmit name d. Communication on c is secure because no execution context knows c. Since P consists of two concurrent subprocesses, the actual implementation could allocate them onto two different computers, whose interconnections are not guaranteed to be secure. Communication on c has to be implemented in terms of lower-level, encrypted communication on some public channel, say p. Thus, process P might be implemented as

$$I_P \stackrel{\text{def}}{=} (\nu\, k_c)(\overline{p}\{d\}_{k_c} \mid p(x).\mathtt{let}\ z = \mathtt{dec}_{k_c}(x)\ \mathtt{in}\ R) \qquad (k_c,\, x \notin fn(R))$$

In I_P, name k_c is a private encryption key that corresponds to channel c. The behaviour of the process is as follows: the process $\overline{p}\{d\}_{k_c}$ sends d encrypted under k_c, while $p(x).\mathtt{let}\ z = \mathtt{dec}_{k_c}(x)\ \mathtt{in}\ R$ tries to decrypt a ciphertext x received at p: if the decryption succeeds, a cleartext is obtained and bound to z and the process behaves like R, otherwise the process is stuck. Note that this implementation does not guarantee that d will eventually be passed to R: message $\{d\}_{k_c}$ could be captured by some context (attacker) listening at p. The last example of this section shall present an implementation that solves this problem.

A secrecy property Assume that R keeps z secret under any context, i.e. for every d and d', and $\sigma_1 \sim \sigma_2$, it holds $(\sigma_1, \sigma_2) \vdash R[d/z] \simeq R[d'/z]$. Under this hypothesis, we want to prove that the implementation scheme for P preserves secrecy. To see this, we consider a generic d', let $Q \stackrel{\text{def}}{=} (\nu\, c)(\overline{c}d' \mid c(z).\, R)$ and show that:

$$(\epsilon_V, \epsilon_V) \vdash I_P \simeq I_Q$$

where I_Q is the obvious implementation of Q and $V = fn(I_P, I_Q)$. In order to prove this, let y be any fresh name and define $\sigma_1 \stackrel{\text{def}}{=} \epsilon_V[\{d\}_{k_c}/y]$ and $\sigma_2 \stackrel{\text{def}}{=} \epsilon_V[\{d'\}_{k_c}/y]$. First, rule (C-Inp) allows one to prove that

$$(\sigma_1, \sigma_2) \vdash (p(x).\mathtt{let}\ z = \mathtt{dec}_{k_c}(x)\ \mathtt{in}\ R) \simeq (p(x).\mathtt{let}\ z = \mathtt{dec}_{k_c}(x)\ \mathtt{in}\ R)$$

To prove this, one exploits two facts: (1) for any ζ s.t. $n(\zeta) - dom(\sigma_1)$ are fresh, if $\widehat{\zeta\sigma_1} = \{M\}_{k_c}$ then $M = d$ and $\widehat{\zeta\sigma_2} = \{d'\}_{k_c}$ (by Lemma 5.4(b) and $k_c \notin kn(\sigma_1)$); (2) the hypothesis that $R[d/z]$ is may-equivalent to $R[d'/z]$ under σ_1 and σ_2. This fact and (C-Par) can be used to infer that:

$$(\sigma_1, \sigma_2) \vdash (\overline{p}\{d\}_{k_c} \mid p(x).\mathtt{let}\ z = \mathtt{dec}_{k_c}(x)\ \mathtt{in}\ R) \simeq (\overline{p}\{d'\}_{k_c} \mid p(x).\mathtt{let}\ z = \mathtt{dec}_{k_c}(x)\ \mathtt{in}\ R).$$

Finally, the wanted claim follows by applying weakening and then (C-Res) (with $(\nu\, k_c)$) to the equality above.

Preservation of may semantics Here we show that the previous implementation scheme also preserves may semantics. We relax the hypothesis that R keeps name z secret, and, for the sake of simplicity, assume that $R \stackrel{\text{def}}{=} \overline{b}z$. In π-calculus, process P is may-equivalent to process $\overline{b}d$. We want to show that the implementations of P and of process $\overline{b}d$ are still may-equivalent, under the assumption that the communication channel p is both *asynchronous* and *noisy*. Thus, the actual implementation also includes a buffer $B \stackrel{\text{def}}{=} !p(x).\overline{p}x$ and a noise generator $N \stackrel{\text{def}}{=} !(\nu\, k)\overline{p}\{k\}_k$ for p. Both noise and asynchrony are necessary to prevent the execution context from detecting traffic on the public channel p. Let $V = fn(I_P, \overline{b}d, N, B)$. To sum up, we want to show that

$$(\epsilon_V, \epsilon_V) \vdash (I_P \mid N \mid B) \simeq (\overline{b}d \mid N \mid B). \tag{3}$$

(Note that this equation is *not* valid for $\approx$). We do this in two steps.

- First, we prove that $(\epsilon_V, \epsilon_V) \vdash \overline{b}d \mid N \mid B \ll I_P \mid N \mid B$. The only possible trace for the configuration $\epsilon_V \rhd \overline{b}d$ is

  $$\epsilon_V \rhd \overline{b}d \stackrel{\overline{b}d}{\longmapsto}_{b(x)} \epsilon_V[d/x] \rhd \mathbf{0}.$$

  Configuration $\epsilon_V \rhd I_P$ can simulate the action above by first communicating $\{d\}_{k_c}$ on p, and then decrypting $\{d\}_{k_c}$:

  $$\epsilon_V \rhd I_P \stackrel{\overline{b}d}{\Longmapsto}_{b(x)} \epsilon_V[d/x] \rhd (\nu\, k_c)\mathbf{0}.$$

  Hence, $(\epsilon_V, \epsilon_V) \vdash \overline{b}d \ll I_P$ and the thesis follows by applying law (C-Par) in Table 4.

- Let us now prove that $(\epsilon_V, \epsilon_V) \vdash I_P \mid N \mid B \ll \overline{b}d \mid N \mid B$. Let y be any fresh name and let $\sigma \stackrel{\text{def}}{=} \epsilon_V[\{d\}_{k_c}/y]$.

  - The crucial step is showing that

    $$(\sigma, \sigma) \vdash p(x).\mathtt{let}\ z = \mathtt{dec}_{k_c}(x)\ \mathtt{in}\ \overline{b}z \ll p(x).\overline{b}d. \tag{4}$$

    Indeed, taking any ζ such that $\widetilde{w} \stackrel{\text{def}}{=} (n(\zeta) - dom(\sigma))$ are fresh names and such that $\widehat{\zeta\sigma} \neq \bot$, we have: $(\sigma[\widetilde{w}/\widetilde{w}], \sigma[\widetilde{w}/\widetilde{w}]) \vdash \mathtt{let}\ z = \mathtt{dec}_{k_c}(\widehat{\zeta\sigma})\ \mathtt{in}\ \overline{b}z \ll \overline{b}d$. In fact, the only case in which $\mathtt{dec}_{k_c}(\widehat{\zeta\sigma})$ does not evaluate to $\bot$ is when $\widehat{\zeta\sigma} = \{d\}_{k_c}$ (Lemma 5.4(b) and $k_c \notin kn(\sigma)$), which implies $\mathtt{let}\ z = \mathtt{dec}_{k_c}(\widehat{\zeta\sigma})\ \mathtt{in}\ \overline{b}z \equiv \overline{b}d$. Then (4) above follows using (C-Inp).
  - Now, using (C-Par) and (4) above, we have that

    $$(\sigma, \sigma) \vdash \overline{p}\{d\}_{k_c} \mid (p(x).\mathtt{let}\ z = \mathtt{dec}_{k_c}(x)\ \mathtt{in}\ \overline{b}z) \ll \overline{p}\{d\}_{k_c} \mid p(x).\overline{b}d$$

    hence, by weakening and (C-Res):

    $$\begin{aligned}(\epsilon_V, \epsilon_V) \vdash\ & (\nu\, k_c)(\overline{p}\{d\}_{k_c} \mid p(x).\mathtt{let}\ z = \mathtt{dec}_{k_c}(x)\ \mathtt{in}\ \overline{b}z) \ll \\ & (\nu\, k_c)(\overline{p}\{d\}_{k_c} \mid p(x).\overline{b}d) \equiv (\nu\, k_c)(\overline{p}\{d\}_{k_c}) \mid p(x).\overline{b}d\,.\end{aligned}$$

In the last step we have used a structural law for restriction ($(\nu\, a)(A_1 \mid A_2) \equiv ((\nu\, a)A_1) \mid A_2$ if $a \notin fn(A_2)$). Using (C-Par) again, we can put the context $N \mid B$ in parallel with the two processes:

$$(\epsilon_V, \epsilon_V) \vdash (\nu\, k_c)(\overline{p}\{d\}_{k_c} \mid p(x).\mathtt{let}\ z = \mathtt{dec}_{k_c}(x)\ \mathtt{in}\ \overline{b}z) \mid N \mid B \ll (\nu\, k_c)(\overline{p}\{d\}_{k_c}) \mid p(x).\overline{b}d \mid N \mid B.$$

Now, $(\nu\, k_c)(\overline{p}\{d\}_{k_c})$ in the right-hand side above can be turned into a particle of noise, because $(\epsilon_V, \epsilon_V) \vdash (\nu\, k_c)(\overline{p}\{d\}_{k_c}) \simeq (\nu\, k)(\overline{p}\{k\}_k)$. Using the structural law for replication ($!A \equiv A \mid !A$), this particle of noise can be absorbed by N, hence:

$$(\epsilon_V, \epsilon_V) \vdash (\nu\, k_c)(\overline{p}\{d\}_{k_c}) \mid p(x).\overline{b}d \mid N \mid B \ \simeq\ p(x).\overline{b}d \mid N \mid B.$$

Moreover, as an instance of a general law for asynchronous channels, we have that

$$(\epsilon_V, \epsilon_V) \vdash p(x).\overline{b}d \mid B \ll \overline{b}d \mid B$$

and the thesis easily follows by (C-Par).

Ensuring message delivery We consider a more sophisticated implementation scheme for process P, and prove that (under a fairness assumption) this scheme guarantees that a message sent on channel c is eventually delivered. Again, we implement c with an asynchronous and noisy public channel p. This time, however, we need a more complex source of noise: $N \stackrel{\text{def}}{=} !(\nu\, k)!\overline{p}\{k\}_k$. Note the difference from the previous example: N can now spawn at any time a process $(\nu\, k)!\overline{p}\{k\}_k$ which emits a constant noise $\{k\}_k$ at p. The buffer B for p is still $B \stackrel{\text{def}}{=} !p(x).\overline{p}x$.

In this example, we shall use recursive definitions of agent constants, of the kind $A \Longleftarrow S$ where A is an agent constant that may appear in the process expression S (these can be taken as primitive — the theory extends smoothly — or can be coded up using replication like in [15]). We also use the shorthand '$\mathtt{let}\ z = \zeta\ \mathtt{in}\ A\ \mathtt{else}\ B$' for '$(\mathtt{let}\ z = \zeta\ \mathtt{in}\ A) + \neg(\mathtt{let}\ z = \zeta\ \mathtt{in}\ \mathit{tt})B$'.

The implementation of P is the process

$$I_P \stackrel{\text{def}}{=} (\nu\, k_c)\big(!\overline{p}\{d\}_{k_c} \mid R\big) \qquad \text{where} \qquad R \Longleftarrow p(x).\mathtt{let}\ z = \mathtt{dec}_{k_c}(x)\ \mathtt{in}\ \overline{b}z\ \mathtt{else}\ (\overline{p}x \mid R).$$

Component $!\overline{p}\{d\}_{k_c}$ constantly emits d encrypted under key k_c on p, while R repeatedly tries to decrypt a ciphertext x received on p using k_c: when the decryption succeeds, the cleartext is sent on b.

Let $V = fn(I_P, \overline{b}d, B, N)$; we want to prove that:

$$(\epsilon_V, \epsilon_V) \vdash (I_P \mid B \mid N) \approx (\overline{b}d \mid B \mid N).$$

In order to see this, define $\sigma_1 \stackrel{\text{def}}{=} \epsilon_V[\{d\}_{k_c}/y]$ and $\sigma_2 \stackrel{\text{def}}{=} \epsilon_V[\{k\}_k/y]$ (y fresh). We first show that

$$(\sigma_1, \sigma_2) \vdash T \ \stackrel{\text{def}}{=}\ (!\overline{p}\{d\}_{k_c} \mid R \mid B \mid N) \ \approx\ (\overline{b}d \mid !\overline{p}\{k\}_k \mid B \mid N) \ \stackrel{\text{def}}{=}\ U \tag{5}$$

from which the thesis will follow by first applying weakening to discard the y-entry, then (C-Res) (with $(\nu\, k_c)$ on the left-hand side and $(\nu\, k)$ on the right-hand side) and then the

structural laws for restriction and the structural law for replication $(N \mid (\nu\, k)!\overline{p}\{k\}_k \equiv N)$ on the right-hand side. To prove (5), we consider a relation $\mathcal{R}$ consisting of *three* pairs (w is fresh):

$$\mathcal{R} = \left\{ \left(\sigma_1 \rhd T\,,\, \sigma_2 \rhd U\right)\ ,\ \left(\sigma_1 \rhd \overline{b}d\,,\, \sigma_2 \rhd \overline{b}d\right)\ ,\ \left(\sigma_1[^d\!/\!_w] \rhd \mathbf{0}\,,\, \sigma_2[^d\!/\!_w] \rhd \mathbf{0}\right) \right\}$$

and show that $\mathcal{R}$ is a weak bisimulation up to parallel composition and contraction. The proof consists in analyzing every transition of $\sigma_1 \rhd T$ and $\sigma_2 \rhd U$ and in showing that a matching transition exists in the other configuration in each case. In particular, note that:

- transitions of $\sigma_1 \rhd T$ originating from $!\overline{p}\{d\}_{k_c}$ are matched up to contraction via transitions from $!\overline{p}\{k\}_k$ in $\sigma_2 \rhd U$, and vice-versa;
- the transition of $\sigma_1 \rhd T$ originating from $R \xrightarrow{p\,\{d\}_{k_c}} \equiv \overline{b}d$ is matched up to parallel composition via $B \xrightarrow{p\,\{k\}_k} \overline{p}\{k\}_k \mid B$ in $\sigma_2 \rhd U$. To see this, first note that $\overline{p}\{k\}_k \mid !\overline{p}\{k\}_k \equiv !\overline{p}\{k\}_k$, and then cut away the parallel contexts $!\overline{p}\{d\}_{k_c} \mid B \mid N$ (from the LHS) and $!\overline{p}\{k\}_k \mid B \mid N$ (from the RHS);
- a transition of $\sigma_1 \rhd T$ originating from $R \xrightarrow{p\,M} \equiv \overline{p}M \mid R$, with $M = \widehat{\zeta\sigma_1} \neq \{d\}_{k_c}$, is matched up to parallel composition and contraction via a transition $B \xrightarrow{p\,M'} \equiv \overline{p}M' \mid B$ in $\sigma_2 \rhd U$, where $M' = \widehat{\zeta\sigma_2}$ (contraction may be used to discard any new name introduced by ζ);
- the transition of $\sigma_2 \rhd U$ originating from $\overline{b}d \xrightarrow{\overline{b}\langle d\rangle} \mathbf{0}$ is matched up to parallel composition via a communication between R and $!\overline{p}\{d\}_{k_c}$ followed by a $\overline{b}\langle d\rangle$-transition in $\sigma_1 \rhd T$.

Communications between R and $!\overline{p}\{d\}_{k_c}$ or R and N are treated like in the second and in the third item above, respectively. It should be now obvious how the other transitions match with each other.

5.3 Verification of a small protocol

Consider a system where two agents A and B share two secret keys, k_{AS} and k_{BS} respectively, with a server S. The purpose of the protocol is to establish a new secret key k between A and B, which A may use to pass some confidential information d to B. This is achieved with a version of the Wide Mouthed Frog Protocol (see, e.g., [7]). For the sake of simplicity, we suppose that the protocol is always started by A and that all communications occur on a public channel, say p. Informally, the protocol can be described as follows:

$$\begin{array}{lll} \text{Message 1} & A \longrightarrow S: & \{k\}_{k_{AS}} \\ \text{Message 2} & S \longrightarrow B: & \{k\}_{k_{BS}} \\ \text{Message 3} & A \longrightarrow B: & \{d\}_k. \end{array}$$

Our intent here is to verify one run of the protocol protocol (that is, we do not consider the case of multiple agents simultaneously executing the protocol). In our language, the above notation translates to a process $P(d)$ defined as follows (we use the notation $R(w)$ to stress that name w may occur free in R; for any M, $R(M)$ abbreviates $R[^M\!/\!_w]$. Bound names are all distinct.):

$$
\begin{array}{rcl}
A(d) & \stackrel{\text{def}}{=} & \overline{p}\{k\}_{k_{AS}}.\overline{p}\{d\}_k.\mathbf{0} \\
S & \stackrel{\text{def}}{=} & p(x).\mathtt{let}\ x' = \mathtt{dec}_{k_{AS}}(x)\ \mathtt{in}\ \overline{p}\{x'\}_{k_{BS}}.\mathbf{0} \\
B & \stackrel{\text{def}}{=} & p(y).\mathtt{let}\ y' = \mathtt{dec}_{k_{BS}}(y)\ \mathtt{in}\ p(z).\mathtt{let}\ z' = \mathtt{dec}_{y'}(z)\ \mathtt{in}\ \mathbf{0}\,. \\
P(d) & \stackrel{\text{def}}{=} & (\nu\, k_{AS},\, k_{BS})\big(((\nu\, k)A(d)) \mid S \mid B\big)\,.
\end{array}
$$

Here we assume that A, S and B terminate after the exchange of message 3: this assumption simplifies the reasoning below, and seems sensible, because the correctness of the protocol should be assessed independently of the subsequent behaviour of the participants. In the sequel, following Abadi and Gordon [7], we use the contextual equivalence $\cong$ to express properties of secrecy and integrity of $P(d)$ (below, we suppose by alpha-equivalence that names k_{AS}, k_{BS}, k, x, x', y, y', z, z' do not occur in messages M and M'):

Secrecy (*"$P(d)$ does not leak d"*)
For any M and M', it holds that $P(M) \cong P(M')$.

Integrity (*"if B accepts a message $\{N\}_k$ then $N = d$"*)
For any M, it holds that $P(M) \cong P^M_{spec}$, where

$$
\begin{array}{rcl}
P^M_{spec} & \stackrel{\text{def}}{=} & (\nu\, k_{AS},\, k_{BS})\big(((\nu\, k)A(d)) \mid S \mid B^M_{spec}\big) \\
B^M_{spec} & \stackrel{\text{def}}{=} & p(y).\mathtt{let}\ y' = \mathtt{dec}_{k_{BS}}(y)\ \mathtt{in}\ p(z).\mathtt{let}\ z' = \mathtt{dec}_{y'}(z)\ \mathtt{in}\ [z' \neq M]\,\overline{p}\,\mathtt{err}.\mathbf{0}\,.
\end{array}
$$

Output of name $\mathtt{err}$ on p is used to signal a violation of integrity, i.e. that some message $\{N\}_k$, with $N \neq M$ has been accepted: the fact that $P(M)$ and P^M_{spec} are equivalent means that action $\overline{p}\langle\mathtt{err}\rangle$, hence the violation, never occur (in this point our formalization differs a little from Abadi and Gordon's). Fix M and M' and let $V \stackrel{\text{def}}{=} fn(P(M), P(M'))$. By virtue of our soundness results, for secrecy, it will be sufficient to show that

$$(\epsilon_V, \epsilon_V) \vdash P(M) \approx P(M') \tag{6}$$

while, for integrity, it will be sufficient to establish that

$$(\epsilon_{V\cup\{\mathtt{err}\}}, \epsilon_{V\cup\{\mathtt{err}\}}) \vdash P(M) \approx P_{spec}(M). \tag{7}$$

We will prove the above equalities by reasoning compositionally on processes. To do this, we first show a compositionality result for $\approx$. Let us say that a process R is *σ-safe* if for each s, whenever $\sigma \rhd R \stackrel{s}{\underset{u}{\Longmapsto}} \sigma' \rhd R' \stackrel{(\nu\,\widetilde{b})\overline{a}\langle M\rangle}{\underset{\eta(x)}{\Longmapsto}}$, then $M \in dc(\sigma')$ (hence $\widetilde{b} = \emptyset$). Intuitively, R is σ-safe if R cannot increase the knowledge of σ. The following proposition strengthens the congruence rule for parallel composition (C-Par), under the assumption that the involved processes are safe for the appropriate environments (the proof is in Appendix B).

Proposition 5.6 *Suppose that $(\sigma_1, \sigma_2) \vdash Q_1 \approx Q_2$ and that $(\sigma_1, \sigma_2) \vdash R_1 \approx R_2$. Suppose that, for $i = 1, 2$, Q_i and R_i are σ_i-safe. Then $(\sigma_1, \sigma_2) \vdash Q_1 \mid R_1 \approx Q_2 \mid R_2$.*

Let us examine secrecy first. Define

$$\sigma \stackrel{\text{def}}{=} \epsilon_V[\{k\}_{k_{AS}}/x_1,\, \{k\}_{k_{BS}}/x_2,\, \{M\}_k/x_3] \qquad \text{and} \qquad \sigma' \stackrel{\text{def}}{=} \epsilon_V[\{k\}_{k_{AS}}/x_1,\, \{k\}_{k_{BS}}/x_2,\, \{M'\}_k/x_3]\,.$$

Clearly, $\sigma \sim \sigma'$. As a first step, check that:

$$\begin{array}{lrcl} (\sigma,\sigma') \vdash & A(M) & \approx & A(M') \\ (\sigma,\sigma') \vdash & S & \approx & S \\ (\sigma,\sigma') \vdash & B & \approx & B. \end{array}$$

The first equality follows from (C-Out), while the second and the third follow from (C-Inp) and Lemma 5.4. For instance, to establish the second equality, Lemma 5.4 is first used to check that whenever names $n(\zeta) - dom(\sigma)$ are fresh and $\widehat{\zeta\sigma_i} = \{N\}_{k_{BS}}$ then $N = k$; then (C-Inp) is applied. Next, it is easy to see that $A(M)$ is σ-safe and that $A(M')$ is σ'-safe, while S and B are both σ- and σ'-safe (again, this requires the use of Lemma 5.4). Applying Proposition 5.6, we can infer that:

$$(\sigma,\sigma') \vdash A(M) \mid S \mid B \approx A(M') \mid S \mid B.$$

Next, apply weakening (so as to discard entries x_1, x_2 and x_3) and then (C-Res) with $(\nu\, k)$ and a structural law $((\nu\, a)(Q \mid R) \equiv ((\nu\, a)Q) \mid R$ if $a \notin fn(R))$ and deduce that

$$(\epsilon_V,\epsilon_V) \vdash ((\nu\, k)A(M)) \mid S \mid B \approx ((\nu\, k)A(M')) \mid S \mid B.$$

Finally apply (C-Res) with $(\nu\, k_{AS},\, k_{BS})$ to get the wanted (6).

As to integrity, let $\sigma \stackrel{\text{def}}{=} \epsilon_{V\cup\{\mathtt{err}\}}[\{k\}_{k_{AS}}/x_1,\, \{k\}_{k_{BS}}/x_2,\, \{M\}_k/x_3]$. First, the crucial point is that $(\sigma,\sigma) \vdash B \approx B^M_{spec}$ (use twice Lemma 5.4 and (C-Inp)). Next, note that B, B^M_{spec}, A and S are all σ-safe (use Lemma 5.4 again). Thus we can compose these processes and proceed like in the case of secrecy to obtain (7).

6 A Calculus with Pairs

Our most relevant omission from the calculus of Abadi and Gordon has been *pairing*, that is, the possibility of transmitting pairs of messages of the form $\langle M_1, M_2\rangle$. It is however easy to extend our theory to a calculus with pairs: the necessary modifications are reported below.

The syntax of messages and of expressions is extended by introducing appropriate constructors and selectors for pairs:

$$\begin{array}{rcl} M & ::= \ldots & \mid \quad \langle M_1, M_2\rangle \\ \zeta & ::= \ldots & \mid \quad \pi_1(\zeta) \;\mid\; \pi_2(\zeta) \;\mid\; \langle\zeta_1,\zeta_2\rangle\,. \end{array}$$

The evaluation functions for expressions and formulae are extended accordingly (for example, $\widehat{\pi_1(\{a\}_k)} = \bot$). The definition of $dc(\cdot)$ is extended with the clause: if $\langle M_1, M_2\rangle \in dc(S)$ then $M_1, M_2 \in dc(S)$. The definition of *core* needs to be revised: informally, a message M can now have several cores, which are found at different positions inside M. If we code a position inside M as a string $p \in \{l, r\}^*$ (that is, a path through the nested pairs of M), then the core of M at position p w.r.t. σ, written $M[\sigma, p]$, can be formally defined as follows by induction on M:

- $a[\sigma, p] = \begin{cases} a & \text{if } p = \epsilon \\ \bot & \text{otherwise} \end{cases}$

- $\{M\}_k[\sigma,p] = \begin{cases} \{M\}_k & \text{if } k \notin kn(\sigma) \text{ and } p = \epsilon \\ M[\sigma,p] & \text{if } k \in kn(\sigma) \\ \bot & \text{otherwise} \end{cases}$

- $\langle M_1, M_2\rangle[\sigma,p] = \begin{cases} M_1[\sigma,p'] & \text{if } p = lp' \\ M_2[\sigma,p'] & \text{if } p = rp' \\ \bot & \text{otherwise.} \end{cases}$

As an example, consider message $M = \langle\langle\{b\}_{hk}, \{c\}_k\rangle, k\rangle$ and substitution $\sigma = [^{M}\!/_{x}, ^{\{h\}_d}\!/_{y}]$. We have $dc(\sigma) = \{M, M', k, \{h\}_d, \{b\}_{hk}, \{b\}_h, \{c\}_k, c\}$, where $M' = \langle\{b\}_{hk}, \{c\}_k\rangle$, and $kn(\sigma) = \{k, c\}$. Moreover, we have that $M[\sigma, r] = k$, $M[\sigma, ll] = \{b\}_h$, $M[\sigma, lr] = c$ and $M[\sigma, p] = \bot$ for $p \notin \{r, ll, lr\}$. Note also that the same would hold for every substitution σ' such that $kn(\sigma) = kn(\sigma')$.

Every core in $\sigma = [^{M_i}\!/_{x_i}]_{i\in I}$ is now determined by an index pair (index i, position p) which we write as ip. The following notation is useful: given a function f defined over $I \times \{l, r\}^*$ and a tuple of index pairs $\tilde{\jmath} = (i_1p_1, \ldots, i_kp_k)$, we let $f[\tilde{\jmath}]$ denote the tuple $(f_{i_1p_1}, \ldots, f_{i_kp_k})$ (we write f_{xy} instead of $f(x, y)$ for function application). We can now give the new definition of $\sim$.

Definition 6.1 (equivalence on environments: pairing) *Consider two substitutions $\sigma_1 = [^{M_i}\!/_{x_i}]_{i\in I}$ and $\sigma_2 = [^{M'_i}\!/_{x_i}]_{i\in I}$. Let $\widetilde{N}, \widetilde{N}' : I \times \{l, r\}^* \to \mathcal{M} \cup \{\bot\}$ be the functions defined as $N_{ip} \stackrel{\text{def}}{=} M_i[\sigma_1, p]$ and $N'_{ip} \stackrel{\text{def}}{=} M'_i[\sigma_2, p]$ for each index pair ip. We write $\sigma_1 \sim' \sigma_2$ iff for each $i \in I$ the following three conditions hold:*

(a) $(\sigma_1, \sigma_2) \vdash M_i \sim M'_i$;

(b) for each p, $N_{ip} \in \mathcal{N}$ iff $N'_{ip} \in \mathcal{N}$;

(c) for each p, $j \in I$ and q, $N_{ip} = N_{jq}$ iff $N'_{ip} = N'_{jq}$

where the predicate $(\sigma_1, \sigma_2) \vdash M \sim M'$ (a recursive version of condition (a) in the old definition) is defined by induction on M as follows:

> *$(\sigma_1, \sigma_2) \vdash M \sim M'$ iff there is a tuple $\tilde{\jmath}$ of index pairs such that $M = \{M_0\}_{\widetilde{N}[\tilde{\jmath}]}$ and $M' = \{M'_0\}_{\widetilde{N'}[\tilde{\jmath}]}$ for some M_0, M'_0 such that either (i) $M_0 = N_{ip}$ and $M'_0 = N'_{ip}$ for some i and p, or (ii) $M_0 = \langle M_1, M_2\rangle$, $M'_0 = \langle M'_1, M'_2\rangle$ and $(\sigma_1, \sigma_2) \vdash M_j \sim M'_j$ for $j = 1, 2$.* ◇

Note that the new definition of $\sim'$ is still effective, because for each $i \in I$ there are finitely many p's s.t. $N_{ip} \neq \bot$ (not more than $|M_i|$). With the new definitions, the results we obtained in the previous sections carry over smoothly, modulo a few notational changes. For instance, the crucial Lemma 5.4 enjoys now the more compact formulation:

> Suppose that $\sigma_1 \sim \sigma_2$ and that $n(\zeta) \subseteq dom(\sigma_1)$. Then either $\widehat{\zeta\sigma_1} = \widehat{\zeta\sigma_2} = \bot$ or $(\sigma_1, \sigma_2) \vdash \widehat{\zeta\sigma_1} \sim \widehat{\zeta\sigma_2}$.

that takes advantage of the predicate $(\sigma_1, \sigma_2) \vdash M \sim M'$.

The changes in the other statements and proofs are obvious and omitted, with the exception of the construction of ϕ_σ, which is given in Appendix C.

Example 6.2 The following example is used by Abadi and Gordon to discuss the incompleteness of their proof technique for cryptographic protocols, *framed bisimulation* [6]. Processes P and Q defined below are not equated by framed bisimilarity, but Abadi and Gordon conjecture that they are barbed congruent (hence barbed- and testing-equivalent). Here, we indeed prove that P and Q are barbed equivalent: this fact confirms that framed bisimulation is not complete w.r.t. barbed equivalence.

Fix some name 0 and define (we write $\{A, B\}_c$ instead of $\{\langle A, B\rangle\}_c$):

$$\begin{array}{rcl} P & \stackrel{\text{def}}{=} & (\nu\, k, k_{01})\ \overline{c}\{k_{01}\}_k.\, c(x).P' \\ Q & \stackrel{\text{def}}{=} & (\nu\, k, k_0, k_1)\ \overline{c}\{k_0, k_1\}_k.\, c(x).Q' \end{array}$$

where

$$\begin{array}{rcl} P' & \stackrel{\text{def}}{=} & [x = 0]\overline{c}k_{01}\ \mid\ [x \neq 0]\overline{c}k_{01} \\ Q' & \stackrel{\text{def}}{=} & [x = 0]\overline{c}k_0\ \mid\ [x \neq 0]\overline{c}k_1\,. \end{array}$$

for fresh and distinct k, k_0, k_1 and k_{01}. The difference between P and Q is that P discloses a single secret k_{01}, whereas Q may disclose either secret k_0 or secret k_1, but not both. The environment cannot detect this difference, because key k, under which the first message is encrypted, is never disclosed. To prove this, take $V = fn(P, Q)$ and let $\sigma_1 \stackrel{\text{def}}{=} \epsilon_V[\{k_{01}\}_k/y]$ and $\sigma_2 \stackrel{\text{def}}{=} \epsilon_V[\{k_0, k_1\}_k/y]$. Clearly $\sigma_1 \sim \sigma_2$ and, furthermore, for each ζ s.t. $\widetilde{w} \stackrel{\text{def}}{=} n(\zeta) - dom(\sigma_1)$ are fresh names and $\widehat{\zeta\sigma_1} \neq \bot$, we have that:

$$(\sigma_1[\widetilde{w}/\widetilde{w}], \sigma_2[\widetilde{w}/\widetilde{w}]) \vdash P'[\widehat{\zeta\sigma_1}/x]\ \approx\ Q'[\widehat{\zeta\sigma_2}/x]$$

In fact, it holds both that $\sigma_1[\widetilde{w}/\widetilde{w}][k_{01}/z] \sim \sigma_2[\widetilde{w}/\widetilde{w}][k_0/z]$ and that $\sigma_1[\widetilde{w}/\widetilde{w}][k_{01}/z] \sim \sigma_2[\widetilde{w}/\widetilde{w}][k_1/z]$, for any fresh z. Therefore, be $\widehat{\zeta\sigma_1}$ equal to 0 or not, we can infer the equivalence above. $(\epsilon_V, \epsilon_V) \vdash P \approx Q$ then follows by applying to the equivalence above first law (C-Inp), then law (C-Out), then weakening to discard the y-entry of the two environments, and finally law (C-Res) with $(\nu\, k, k_{01})$ on the left-hand side and $(\nu\, k, k_0, k_1)$ on the right-hand side. Thus we can conclude that $P \cong Q$.

7 Final Remarks and Related Work

We have studied contextual equivalences and relative proof techniques for a variant of the spi-calculus, an extension of the π-calculus proposed by Abadi and Gordon [7]. We have considered a few examples of verification, concerning secure channels implementation and protocol security, which demonstrate how these techniques can be used in practice.

In this paper, we have applied our techniques to small examples, as the emphasis was more on theory. However, we believe that our methodology can be used to reason on more complex systems. In this respect, a major advantage of our approach is the possibility of compositional reasoning offered by a set of congruence laws. A further step in this direction would be the design of a sound and complete proof system for the considered equivalences.

Another direction for future research is the study of algorithms for mechanical equivalence checking (especially in the case of bisimilarity). We are also considering extensions of the theory to public keys and digital signatures. A subtle point that remains to be solved is in connection with the so-called "known plaintext attack": for instance, knowing a, b and a public key k^+, an attacker could distinguish by comparison $\{b\}_{k^+}$ from $\{a\}_{k^+}$, even without knowing the private key k^- that 'opens' these messages. As a consequence, and in contrast with the shared key case, two environments like $\sigma = [a/x, \{a\}_{k^+}/y]$ and $\sigma' = [a/x, \{b\}_{k^+}/y]$ cannot be regarded as equivalent.

The relevance of may-testing to the analysis of security properties has been first pointed out by Abadi and Gordon in [7]. May-testing was originally introduced for CCS in [11], and subsequently studied for the π-calculus in [8].

Two papers closely related to our work are [10] and [6]. In [10], Boreale and Sangiorgi introduce a lts for a typed version of π-calculus, where the environment's input/output capabilities on names are explicitly described and updated. Here, we use a similar approach to model the environment's knowledge about names and keys.

Abadi and Gordon present in [6] *framed bisimulation*, a proof technique to analyze cryptographic protocols. In framed bisimulation, when comparing two processes P and Q, a *frame-theory pair* (fr, th) is used to represent the knowledge of P's and Q's environments. A judgement $(fr, th) \vdash M \leftrightarrow N$ is also introduced to express that the effect of message M on P's environment is the same as the effect of message N on Q's environment. The judgement is used to check indistinguishability of messages M and N that are exchanged by P and its environment and by Q and its environment. In our case, the indistinguishability of M and N is guaranteed by requiring matching transitions to exhibit the same environment action and to take equivalent environments into equivalent environments. This results in a relevant difference between the work in [6] and ours when considering output transitions. In our case, given an output transition, it is sufficient to check (like in standard bisimilarity) whether the other configuration can perform a matching output transition. Output transitions are, at least for finite–control processes, finitely many. In the case of [6], one must also build a new frame-theory pair that relates N to M and consistently extends the old one: this might be not completely trivial [12]. Moreover, in [6] there seems to be very few tools for compositional reasoning (congruence laws) and no obvious way of tailoring the 'up to' techniques to their setting. Finally, as shown in Example 6.2, framed bisimulation is not complete for barbed equivalence.

The process algebraic approach to cryptographic protocols has also been followed by Roscoe [19], Lowe [14] and Schneider [22], that consider model-checking of security protocols in a CSP-based framework. This approach requires explicitly designing a specific (powerful enough) attacker and carrying out the analysis with it. Of course, there is always a certain amount of arbitrarity in determining the attacker; any modification of the attacker would require a new analysis. In our paper, like in [7], a more radical approach is taken: the attacker may be be *any* process that can be defined in spi-calculus.

In [2], Abadi presents an approach to secrecy that combines the spi-calculus and the use of type systems: the idea is that a process $P(d)$ that type-checks guarantees secrecy of d (in a sense made precise via testing equivalence).

All the approaches mentioned so far, including ours, work under a perfect encryption hypothesis: this prevents the attacker from, for example, randomly guessing some bits of a secret key, or performing statistical analysis of messages. A first step towards relaxing this hypothesis has been made in [13], where probabilistic versions of the spi-calculus and of testing equivalence are introduced. Further research is required for a fuller understanding of these notions and for devising techniques to reason over them.

Acknowledgements We would like to thank the anonymous LICS referees for helpful comments. Discussions with Martin Abadi, Cedric Fournet and Andrew Gordon have helped us to improve the paper.

A Results on Environment Equivalence

In this appendix we keep the definitions of $\sim$ (Definition 3.1) and of $\sim'$ (Definition 3.3) separate and introduce the notion of characteristic formula, ϕ_σ. The main steps taken here are:

1. ϕ_σ characterizes all and only those environments $\sim'$-equivalent to σ (Theorem A.11);
2. using 1., one proves that $\sim$ and $\sim'$ coincide (Theorem 3.4/Theorem A.13);
3. hence, ϕ_σ characterizes all and only those environments $\sim$-equivalent to σ (Theorem 4.3/Theorem A.14).

Along the way, a few properties of $\sim$ are established which are of independent interest and are useful for the examples of Section 5 (Lemma 5.3 and Lemma 5.4).

First we show that evaluation $\widehat{\cdot}$ commutes with substitution.

Lemma A.1 *Let ζ and η be two expressions and σ a substitution. If $x \notin dom(\sigma)$, $n(\zeta) \subseteq dom(\sigma) \cup \{x\}$ and $n(\eta) \subseteq dom(\sigma)$ then $\widehat{\zeta\sigma[\widehat{\eta\sigma}/x]} = \widehat{(\zeta[\eta/x])\sigma}$.*

PROOF: Straightforward induction on ζ. □

The following lemma establishes a first relationship between ζ-expressions and the decryption closure $dc(\cdot)$, and between the knowledge $kn(\cdot)$ and the set of *core*'s of σ.

Lemma A.2 *Let $\sigma = [M_i/x_i]_{i\in I}$.*

1. *If $M \in dc(\sigma)$ then there is ζ s.t. $n(\zeta) \subseteq dom(\sigma)$ and $\widehat{\zeta\sigma} = M$.*
2. *If $a \in kn(\sigma)$ then $a = core(\sigma, x_i)$, for some $i \in I$.*

PROOF: Part 1 is easily proven by induction on the definition of $dc(\cdot)$. Part 2 follows from the following statement, easily proven by induction on the definition of $dc(\cdot)$: If $M \in dc(\sigma)$ then there are $i \in I$ and $\widetilde{k}$ s.t. $M = \{core(\sigma, x_i)\}_{\widetilde{k}}$. □

We proceed by showing that the knowledge $kn(\sigma)$ of σ is precisely the set of those names that can be obtained by arbitrary combinations of encryption and decryption operations (represented by ζ-expressions), starting from messages stored in σ.

Lemma A.3 (Lemma 5.3) *Let σ be an substitution. Then $kn(\sigma) = \{\widehat{\zeta\sigma} \in \mathcal{N} : n(\zeta) \subseteq dom(\sigma)\}$.*

PROOF: That $kn(\sigma) \subseteq \{\widehat{\zeta\sigma} \in \mathcal{N} : n(\zeta) \subseteq dom(\sigma)\}$ is a consequence of Lemma A.2(1). We now prove the opposite inclusion. Consider the set $S \stackrel{\text{def}}{=} \{\widehat{\zeta\sigma} \neq \bot : n(\zeta) \subseteq dom(\sigma)\}$ and the set $T \stackrel{\text{def}}{=} \{\{M\}_{\widetilde{k}} : M \in dc(\sigma) \text{ and } \widetilde{k} \subseteq kn(\sigma)\}$. We prove that $S \subseteq T$, from which the thesis follows, because $S \cap \mathcal{N} = \{\widehat{\zeta\sigma} \in \mathcal{N} \mid n(\zeta) \subseteq dom(\sigma)\}$ and $T \cap \mathcal{N} = kn(\sigma)$. The proof proceeds by induction on the structure of ζ. We explicitly show only the case $\zeta = \mathtt{dec}_{\zeta_1}(\zeta_2)$, as the other cases are similar or easier. Let $\widehat{\zeta\sigma} = M$. Since $\widehat{\zeta\sigma} \neq \bot$, there is a name k such

that $\widehat{\zeta_2\sigma} = \{M\}_k$ and $\widehat{\zeta_1\sigma} = k$. By induction, we can assume that $\{M\}_k \in T$, which, by definition of T, implies that there are $M' \in dc(\sigma)$ and $\widetilde{h} \subseteq kn(\sigma)$ such that $\{M\}_k = \{M'\}_{\widetilde{h}}$. Hence $M = \{M'\}_{\widetilde{h'}}$, for $\widetilde{h'}$ such that $\widetilde{h} = \widetilde{h'}k$, and the thesis follows. □

The next lemma is about the effect of applying two $\sim'$-equivalent substitutions σ and σ' onto the same expression ζ.

Lemma A.4 (Lemma 5.4) *Let $\sigma = [M_i/x_i]_{i\in I}$ and $\sigma' = [M'_i/x_i]_{i\in I}$ be two substitutions such that $\sigma \sim' \sigma'$. Define $\widetilde{N} = core(\sigma, x_i)_{i\in I}$ and $\widetilde{N'} = core(\sigma', x_i)_{i\in I}$. For each ζ s.t. $n(\zeta) \subseteq \widetilde{x}$, either*

(a) $\widehat{\zeta\sigma} = \widehat{\zeta\sigma'} = \bot$, or

(b) there are $i \in I$ and a tuple $\widetilde{\jmath} \subseteq I$ such that: $\widehat{\zeta\sigma} = \{N_i\}_{\widetilde{N}[\widetilde{\jmath}]}$ and $\widehat{\zeta\sigma'} = \{N'_i\}_{\widetilde{N'}[\widetilde{\jmath}]}$.

Proof: The proof proceeds by induction on ζ. We explicitly consider only the case $\zeta = \mathtt{dec}_{\zeta_2}(\zeta_1)$; the other cases are similar or easier. If $\widehat{\zeta_1\sigma} = \bot$ or $\widehat{\zeta_2\sigma} = \bot$, then by induction hypothesis it easily follows that $\widehat{\zeta\sigma} = \widehat{\zeta\sigma'} = \bot$. Otherwise, by induction hypothesis we have that for some tuples $\widetilde{\jmath}, \widetilde{\ell} \subseteq I$ and indices $i, j \in I$:

$$\begin{array}{rclcrcl} \widehat{\zeta_1\sigma} & = & \{N_i\}_{\widetilde{N}[\widetilde{\jmath}]} & \qquad & \widehat{\zeta_1\sigma'} & = & \{N'_i\}_{\widetilde{N'}[\widetilde{\jmath}]} \\ \widehat{\zeta_2\sigma} & = & \{N_j\}_{\widetilde{N}[\widetilde{\ell}]} & & \widehat{\zeta_2\sigma'} & = & \{N'_j\}_{\widetilde{N'}[\widetilde{\ell}]} \end{array}$$

There are two cases:

- $\widetilde{\ell} \neq \emptyset$ or $N_j \notin \mathcal{N}$. Then by definition $\widehat{\zeta\sigma} = \widehat{\zeta\sigma'} = \bot$ (note that $N_j \notin \mathcal{N}$ implies $N'_j \notin \mathcal{N}$, because $\sigma \sim' \sigma'$).
- $\widetilde{\ell} = \emptyset$ and $N_j = a \in \mathcal{N}$. Hence $N'_j = a' \in \mathcal{N}$, because $\sigma \sim' \sigma'$. Now, if the last component of $\widetilde{\jmath}$ is some j' s.t. $N_{j'} = a$, say $\widetilde{\jmath} = (\widetilde{\jmath}', j')$, then it is also $N'_{j'} = a'$, because $\sigma \sim' \sigma'$: thus $\widehat{\zeta\sigma} = \{N_i\}_{\widetilde{N}[\widetilde{\jmath}']}$ and $\widehat{\zeta\sigma'} = \{N'_i\}_{\widetilde{N'}[\widetilde{\jmath}']}$, which is the wanted claim for this case. Otherwise, $\widehat{\zeta\sigma} = \widehat{\zeta\sigma'} = \bot$. □

The intuition underlying the following lemma is that $\sim'$-equivalence is preserved when uniformly adding entries to two equivalent environments.

Lemma A.5 *If $\sigma_1 \sim' \sigma_2$ then $\sigma_1[\widehat{\zeta\sigma_1}/y] \sim' \sigma_2[\widehat{\zeta\sigma_2}/y]$, provided that $\widehat{\zeta\sigma_1} \neq \bot$ and that $n(\zeta) \subseteq dom(\sigma_1)$.*

Proof: Apply Lemma 5.4 to σ_1, σ_2 and ζ: the thesis then follows by definition of $\sim'$. □

We can now prove that of $\sim'$ implies $\sim$.

Lemma A.6 *Let σ and σ' be two substitutions. If $\sigma \sim' \sigma'$ then $\sigma \sim \sigma'$.*

Proof: Suppose that $\sigma \sim' \sigma'$. We must show that for each ϕ with $fn(\phi) \subseteq dom(\sigma)$ it holds that $\sigma \models \phi$ iff $\sigma' \models \phi$. The proof proceeds by induction on ϕ. Here, we explicitly consider only the case where ϕ is $\mathtt{let}\ z = \zeta\ \mathtt{in}\ \phi'$, as the other cases are similar or easier. By definition, $\sigma \models \mathtt{let}\ z = \zeta\ \mathtt{in}\ \phi'$ means that $\widehat{\zeta\sigma} \neq \bot$ and that $\sigma[\widehat{\zeta\sigma}/z] \models \phi'$. Now, we have

$$
\begin{array}{ll}
 & \texttt{let}_{j\in I}\ z_j = \zeta_j\ \texttt{in}\ \Big[\\
(*) & \bigwedge_{i,j\in I}[\texttt{dec}_{z_j}(z_i) = \bot] \wedge \\
(a) & \bigwedge_{i\in I}[x_i = \{z_i\}_{\widetilde{z}[\mathfrak{j}_i]}] \wedge \\
(b) & \bigwedge_{N_i\in\mathcal{N}} name(z_i) \ \wedge\ \bigwedge_{N_i\notin\mathcal{N}} \neg name(z_i) \wedge \\
(c) & \bigwedge_{i,j\in I \text{ s.t. } N_i=N_j}[z_i = z_j] \ \wedge\ \bigwedge_{i,j\in I \text{ s.t. } N_i\neq N_j}[z_i \neq z_j] \\
 & \Big]
\end{array}
$$

Table 5: The formula ϕ_σ

that $\widehat{\zeta\sigma'} \neq \bot$ (Lemma A.4) and $\sigma[\widehat{\zeta\sigma}/z] \sim \sigma'[\widehat{\zeta\sigma'}/z]$ (Lemma A.5). By induction, $\sigma'[\widehat{\zeta\sigma'}/z] \models \phi'$ and we can conclude that $\sigma' \models \texttt{let}\ z = \zeta\ \texttt{in}\ \phi'$. □

We proceed now to showing the converse implication, that $\sim$ implies $\sim'$. Two crucial ingredients for the proof will be the notion of *characteristic formula* of an environment σ, ϕ_σ, and Theorem A.11, that will give a logical characterization of $\sim'$. We need some notational shorthands.

Notation A.7

- *'*$\texttt{let}_{j\in 1..k}\ z_j = \zeta_j\ \texttt{in}\ \phi$*' stands for '*$\texttt{let}\ z_1 = \zeta_1\ \texttt{in}\ (\cdots(\texttt{let}\ z_k = \zeta_k\ \texttt{in}\ \phi)\cdots)$*'.*
- *'*$\zeta \neq \bot$*' stands for '*$\texttt{let}\ z = \zeta\ \texttt{in}\ tt$*', for any* z*, and '*$\zeta = \bot$*' stands for '*$\neg(\zeta \neq \bot)$*'.*
- *'*$w \in \{M_1, \ldots, M_m\}$*' stands for '*$\bigvee_{j=1}^{m}[w = M_j]$*'.*

Definition A.8 (characteristic formula) *Let* $\sigma = [M_i/x_i]_{i\in I}$ *be a substitution. For each* $i \in I$*, let* $N_i = core(\sigma, x_i)$ *and let* ζ_i *be the least*[4] *expression such that* $n(\zeta_i) \subseteq \widetilde{x}$ *and* $\widehat{\zeta_i\sigma} = N_i$*. Let* $\widetilde{N} = N_{i\in I}$ *and* $\widetilde{\zeta} = \zeta_{i\in I}$*. Finally, for each* $i \in I$*, let* $\widetilde{\mathfrak{j}_i} \subseteq I$ *be a tuple such that* $M_i = \{N_i\}_{\widetilde{N}[\widetilde{\mathfrak{j}_i}]}$*. The formula* ϕ_σ *is then defined as in Table 5.* ◇

About Definition A.8 we have the following:

Remark A.9 Note that the expressions $\zeta_{i\in I}$ mentioned in the definition above do exist by virtue of Lemma A.2(1). Furthermore, we are allowed to assume that $M_i = \{N_i\}_{\widetilde{N}[\widetilde{\mathfrak{j}_i}]}$, for some tuple $\widetilde{\mathfrak{j}_i}$, because, by virtue of Lemma A.2(2), $kn(\sigma) \subseteq \widetilde{N}$. Finally, note that $fn(\phi_\sigma) \subseteq dom(\sigma)$ and that $\sigma \models \phi_\sigma$.

A lemma that gives conditions under which a tuple of messages $\widetilde{Z}$ can be identified as the tuple of 'cores' of a given σ.

[4] w.r.t. some fixed total ordering of expressions.

Lemma A.10 *Consider* $\sigma = [M_i/x_i]_{i\in I}$ *and a tuple* $\widetilde{Z} = Z_{i\in I}$ *s.t.* $\widetilde{Z} \subseteq dc(\sigma)$. *Suppose that the following two conditions hold:*

(a) for each $i, j \in I$, $\widehat{\mathtt{dec}_{Z_j}(Z_i)} = \bot$;

(b) for each $i \in I$, *there is a tuple* $\widetilde{\jmath} \subseteq I$ *s.t.* $M_i = \{Z_i\}_{\widetilde{Z}[\widetilde{\jmath}]}$.

Then $\widetilde{Z} = core(\sigma, x_i)_{i\in I}$.

PROOF: The proof consists of two steps.

- We first show that $kn(\sigma) \subseteq \widetilde{Z}$. To this end, consider the set of messages $T \stackrel{\text{def}}{=} \{N : \{N\}_{\widetilde{k}} = M_i \text{ for some } i \in I \text{ and } \widetilde{k} \subseteq \widetilde{Z}\}$. First, one proves by induction on the definition of $dc(\cdot)$ that $dc(\sigma) \subseteq T$. Then, it is easy to see that $T \cap \mathcal{N} \subseteq \widetilde{Z}$: indeed if $\{a\}_{\widetilde{k}} = M_i$ for some $\widetilde{k} \subseteq \widetilde{Z}$, then it also holds that $\{a\}_{\widetilde{k}} = \{Z_i\}_{\widetilde{Z}[\widetilde{\jmath}]}$ (condition (b)), which, by virtue of condition (a) implies $a = Z_i$. Therefore we can conclude $\widetilde{Z} \supseteq kn(\sigma)$.
- Let $\widetilde{N} = core(\sigma, x_i)_{i\in I}$ and take any $i \in I$: we show that $Z_i = N_i$. By definition of *core* and by condition (b), we have that $M_i = \{N_i\}_{\widetilde{h}} = \{Z_i\}_{\widetilde{Z}[\widetilde{\jmath}]}$, for some tuple $\widetilde{\jmath} \subseteq I$ and $\widetilde{h} \subseteq kn(\sigma)$. There are two cases:
  - $Z_i = \{N_i\}_{\widetilde{k}}$ with $(\widetilde{k}, \widetilde{Z}[\widetilde{\jmath}]) = \widetilde{h}$. Since $\widetilde{k} \subseteq kn(\sigma)$ (as $\widetilde{h} \subseteq kn(\sigma)$), due to $kn(\sigma) \subseteq \widetilde{Z}$ and condition (a), we deduce that $\widetilde{k} = \emptyset$, hence $Z_i = N_i$.
  - $N_i = \{Z_i\}_{\widetilde{k}}$ with $(\widetilde{k}, \widetilde{h}) = \widetilde{Z}[\widetilde{\jmath}]$. Again, we have that $\widetilde{k} \subseteq kn(\sigma)$ (as $\widetilde{k} \subseteq \widetilde{Z} \cap \mathcal{N}$ and $\widetilde{Z} \subseteq dc(\sigma)$), hence, by definition of *core*, we get $\widetilde{k} = \emptyset$, hence $Z_i = N_i$. □

The crucial result on $\sim'$ and characteristic formulae.

Theorem A.11 *Let* σ *and* σ' *be two substitutions such that* $dom(\sigma) = dom(\sigma')$. *We have that* $\sigma \sim' \sigma'$ *if and only if* $\sigma' \models \phi_\sigma$.

PROOF: The 'only if' part is a consequence of Lemma A.6 (as $\sigma \models \phi_\sigma$). Let us see the 'if' part. Suppose that $\sigma = [M_i/x_i]_{i\in I}$, and take any $\sigma' = [M'_i/x_i]_{i\in I}$ s.t. $\sigma' \models \phi_\sigma$: we show that $\sigma' \sim \sigma$. With the notation of Table 5, let $Z_i = \widehat{\zeta_i \sigma'}$ and $\widetilde{Z} = Z_{i\in I}$. From $\sigma' \models \phi_\sigma$, we deduce that $\bot \notin \widetilde{Z}$ and that $\sigma'[\widetilde{Z}/\widetilde{z}] \models (*) \wedge (a) \wedge (b) \wedge (c)$. Now, we have that:

- $\sigma'[\widetilde{Z}/\widetilde{z}] \models (*)$ means that for each $i, j \in I$, $\widehat{\mathtt{dec}_{Z_j}(Z_i)} = \bot$.
- $\sigma'[\widetilde{Z}/\widetilde{z}] \models (a)$ means that for each $i \in I$ there is a tuple $\widetilde{\jmath} \subseteq I$ s.t. $M'_i = \{Z_i\}_{\widetilde{Z}[\widetilde{\jmath}]}$.

The above two facts and Lemma A.10 imply that $\widetilde{Z} = core(\sigma', x_i)_{i\in I}$. Thus $\sigma'[\widetilde{Z}/\widetilde{z}] \models (a) \wedge (b) \wedge (c)$ precisely says that $\sigma \sim \sigma'$. □

We can prove that $\sim$ implies $\sim'$.

Lemma A.12 *Let* σ *and* σ' *be substitutions. If* $\sigma \sim \sigma'$ *then* $\sigma \sim' \sigma'$.

PROOF: By definition $fn(\phi_\sigma) \subseteq dom(\sigma)$ and $\sigma \models \phi_\sigma$. Hence, by hypothesis, $\sigma' \models \phi_\sigma$ and the thesis immediately follows from Theorem A.11. □

The coincidence of $\sim$ and $\sim'$ is now an immediate consequence of Lemma A.6 and of Lemma A.12.

Theorem A.13 (Theorem 3.4) *Let σ and σ' be substitutions. Then $\sigma \sim \sigma'$ if and only if $\sigma \sim' \sigma'$.*

The following important property of characteristic formulae is an immediate consequence of Theorem A.11 and of Theorem A.13.

Theorem A.14 (Theorem 4.3) *Let σ and σ' be substitutions such that $dom(\sigma) = dom(\sigma')$. We have that $\sigma \sim \sigma'$ if and only if $\sigma' \models \phi_\sigma$.*

We end the section with two technical lemmas. The first lemma is useful for manipulating environments: it says that equivalence is preserved when uniformly removing entries from two equivalent environments. Its proof is an easy consequence of the definition of $\sim$.

Lemma A.15 *If $\sigma_1[M/y] \sim \sigma_2[M'/y]$ then $\sigma_1 \sim \sigma_2$.*

The second lemma is about the size of characteristic formulae (recall that $rk(\sigma) = |\phi_\sigma|$).

Lemma A.16 (Lemma 4.22) *If $\sigma \sim \sigma'$ then $rk(\sigma) = rk(\sigma')$.*

PROOF: Referring to the notation of Definition A.8, note that the size of ϕ_σ and $\phi_{\sigma'}$ may only differ due a different choice of some ζ_i, $i \in I$. But Lemma A.4(1) (which ensures that the same ζ_i's can be used in both ϕ_σ and $\phi_{\sigma'}$) and the requirement of minimality imply that the ζ_i's in ϕ_σ and those in $\phi_{\sigma'}$ are actually the same. □

B Results on Trace and Bisimulation Semantics

We begin the section with a crucial result on operational semantics.

Proposition B.1 (Proposition 4.2) *Consider two equivalent substitutions σ_1 and σ_2. Let R be any process or observer s.t. $fn(R) \subseteq dom(\sigma_1)$.*

1. *Suppose that $R\sigma_1 \xrightarrow{(\nu \widetilde{b})\overline{a}\langle M\rangle} R_1$. Then: (i) there is η with $n(\eta) \subseteq dom(\sigma_1)$ s.t. $\widehat{\eta\sigma_1} = a$; (ii) there are ζ and R' with $fn(\zeta, R') \subseteq dom(\sigma_1) \cup \widetilde{b}$ s.t. $M = \widehat{\zeta\sigma_1}$ and $R_1 \equiv R'\sigma_1$; (iii) it holds that $R\sigma_2 \xrightarrow{(\nu \widetilde{b})\overline{a'}\langle M'\rangle} R_2$, where $a' = \widehat{\eta\sigma_2}$, $M' = \widehat{\zeta\sigma_2}$ and $R_2 \equiv R'\sigma_2$.*
2. *Suppose that $R\sigma_1 \xrightarrow{a\,M} R_1$. Then: (i) there is η with $n(\eta) \subseteq dom(\sigma_1)$ s.t. $\widehat{\eta\sigma_1} = a$; (ii) taken any fresh y and $\sigma_1' \stackrel{\text{def}}{=} \sigma_1[M/y]$, there is R' with $fn(R') \subseteq dom(\sigma_1')$ s.t. $R_1 \equiv R'\sigma_1'$; (iii) for any M', it holds that $R\sigma_2 \xrightarrow{a'\,M'} R_2$, where $a' = \widehat{\eta\sigma_2}$ and $R_2 \equiv R'\sigma_2'$ with $\sigma_2' \stackrel{\text{def}}{=} \sigma_2[M'/y]$.*

3. Suppose that $R\sigma_1 \xrightarrow{\mu} R_1$ *with* $\mu = \tau$ *or* $\mu = \omega$. *Then: (i) there is* R' *with* $fn(R') \subseteq dom(\sigma_1)$ *such that* $R_1 \equiv R'\sigma_1$; *(ii) we have* $R\sigma_2 \xrightarrow{\mu} R_2$, *where* $R_2 \equiv R'\sigma_2$.

PROOF: An induction on the derivation of $R\sigma \xrightarrow{\mu} R_1$. Here, we explicitly consider only the case where the last rule applied is (LET), which is the most delicate.

Suppose that R is $\texttt{let}\ z = \xi\ \texttt{in}\ P$ and that $R \xrightarrow{\mu} R_1$ is inferred via (LET) from the premise $P\rho_1 \xrightarrow{\mu} R_1$, where $\rho_1 \stackrel{\text{def}}{=} \sigma_1[\widehat{\xi\sigma_1}/z]$ ($\widehat{\xi\sigma_1} \neq \perp$). Depending on the form of μ we have three cases, here we only consider the case when μ is an output, say $\mu = (\nu\,\widetilde{b})\overline{a}\langle M\rangle$. Since $\widehat{\xi\sigma_2} \neq \perp$ (Lemma A.4) we can define $\rho_2 \stackrel{\text{def}}{=} \sigma_2[\widehat{\xi\sigma_2}/z]$. Since $\rho_1 \sim \rho_2$ (Lemma A.5), by induction hypothesis applied to the premise $P\rho_1 \xrightarrow{\mu} R_1$ we have that:

(j) there is η' with $dom(\eta') \subseteq dom(\rho_1)$ and $\widehat{\eta'\rho_1} = a$

(jj) there are ζ' and R'' with $fn(\zeta', R'') \subseteq dom(\rho_1) \cup \widetilde{b}$ such that $M = \widehat{\zeta'\rho_1}$ and $R_1 \equiv R''\rho_1$

(jjj) $P\rho_2 \xrightarrow{(\nu\,\widetilde{b})\overline{a'}\langle M'\rangle} R_2$ where $a' = \widehat{\eta'\rho_2}$, $M' = \widehat{\zeta'\rho_2}$ and $R_2 \equiv R''\rho_2$.

Define $\eta \stackrel{\text{def}}{=} \eta'[\xi/z]$, $\zeta \stackrel{\text{def}}{=} \zeta'[\xi/z]$ and $R' \stackrel{\text{def}}{=} \texttt{let}\ z = \xi\ \texttt{in}\ R''$. Then we have that $\widehat{\eta\sigma_1} = a$, $\widehat{\zeta\sigma_1} = M$ (Lemma A.1) and that $R'\sigma_1 \equiv R_1$: this proves parts (i) and (ii) of the claim. Similarly, it holds that $\widehat{\eta\sigma_2} = a'$ and $\widehat{\zeta\sigma_2} = M'$ and that $R'\sigma_2 \equiv R_2$. Part (iii) follows from these equalities and applying (LET) to transition $P\rho_2 \xrightarrow{(\nu\,\widetilde{b})\overline{a'}\langle M'\rangle} R_2$. □

B.1 Trace semantics

Lemma B.2 (Lemma 4.7) *Suppose* $\sigma_1 \sim \sigma_2$ *and let* O *be any observer such that* $fn(O) \subseteq dom(\sigma_1)$. *Suppose that* $O\sigma_1 \stackrel{r}{\Longrightarrow} O_1$, *where, for some* u *and* σ_1' *extending* σ_1, *it is* $r = \widehat{u\sigma_1'}$. *Then, there is* O' *with* $fn(O') \subseteq dom(\sigma_1')$ *such that* $O_1 \equiv O'\sigma_1'$. *Furthermore, for any* σ_2' *extending* σ_2 *and such that* $\sigma_2' \sim \sigma_1'$, *it holds that* $O\sigma_2 \stackrel{r'}{\Longrightarrow} \equiv O'\sigma_2'$, *with* $r' = \widehat{u\sigma_2'}$.

PROOF: The proof consists in iterating the statement of Proposition 4.2. Formally one proceeds by induction on trace u. We only examine the case when $u = u' \cdot \eta(x)$. For any substitution σ, let us write σ^{-x} for the substitution that behaves like σ but is undefined on x. Then we can write $r = \widehat{u'\sigma_1'^{-x}} \cdot \widehat{\eta(x)\sigma_1'} \stackrel{\text{def}}{=} s \cdot aM$ for $\widehat{\eta\sigma_1^{-x}} = a$ and $M = x\sigma_1'$, and similarly $r' = \widehat{u'\sigma_2'^{-x}} \cdot \widehat{\eta(x)\sigma_2'} \stackrel{\text{def}}{=} s' \cdot a'M'$ for $\widehat{\eta\sigma_2^{-x}} = a'$ and $M' = x\sigma_2'$ (recall that, by our convention on bound names, name x does not occur in u'). Applying induction and Proposition 4.2(2) and (3), the sequence of transitions $O\sigma_1 \stackrel{r}{\Longrightarrow} O_1$ can be decomposed as follows, for suitable O'' with $fn(O'') \subseteq dom(\sigma_1'^{-x})$, and O''' and O' with $fn(O''', O') \subseteq dom(\sigma_1')$:

$$O\sigma_1 \stackrel{s}{\Longrightarrow} \equiv O''\sigma_1'^{-x} \xrightarrow{aM} \equiv O'''\sigma_1' \Longrightarrow \equiv O'\sigma_1' \equiv O_1.$$

Similarly

$$O\sigma_2 \stackrel{s'}{\Longrightarrow} \equiv O''\sigma_2'^{-x} \xrightarrow{a'M'} \equiv O'''\sigma_2' \Longrightarrow \equiv O'\sigma_2'$$

is inferred using again induction and Proposition 4.2(2) and (3) (note that $\sigma_1'^{-x} \sim \sigma_2'^{-x}$ by Lemma A.15). This implies $O\sigma_2 \stackrel{r'}{\Longrightarrow} \equiv O'\sigma_2'$ that concludes the proof for this case. □

Lemma B.3 (Lemma 4.8) *Consider σ, P and any observer O with $fn(O) \subseteq dom(\sigma)$. Suppose that $P \stackrel{s}{\Longrightarrow} P'$ and that $O\sigma \stackrel{r}{\Longrightarrow}$ with $s \underline{compl}\, r$. Then there are u and σ' extending σ such that $r = \widehat{u\sigma'}$ and $\sigma \rhd P \stackrel{s}{\stackrel{\Longrightarrow}{u}} \sigma' \rhd P'$.*

PROOF: The proof consists in iterating the following two statements (the first one is an easy consequence of Proposition 4.2):

1. Suppose that $P \stackrel{\mu}{\longrightarrow} P'$ $(\mu \neq \tau)$. Suppose that, for some observer O with $fn(O) \subseteq dom(\sigma)$, it holds that $O\sigma \stackrel{\lambda}{\longrightarrow} O_1$, with $\mu \underline{compl}\, \lambda$. Then there are σ' extending σ, δ and O' with $fn(O') \subseteq dom(\sigma')$ such that: $\sigma \rhd P \stackrel{\mu}{\stackrel{\longmapsto}{\delta}} \sigma' \rhd P'$ with $\lambda = \widehat{\delta\sigma'}$ and $O_1 \equiv O'\sigma'$.

2. Suppose that $P \stackrel{\tau}{\longrightarrow} P'$. Then $\sigma \rhd P \stackrel{\tau}{\stackrel{\longmapsto}{-}} \sigma \rhd P'$. □

Lemma B.4 (Lemma 4.9) *Suppose that $\sigma \rhd P \stackrel{s}{\stackrel{\Longrightarrow}{u}} \sigma' \rhd P'$. Then it holds that $P \stackrel{s}{\Longrightarrow} P'$, that σ' extends σ and that $s \underline{compl}\, r$, where $r = \widehat{u\sigma'}$.*

PROOF: An easy induction on u. □

B.2 Bisimilarity

Proposition B.5 (Proposition 4.17) *Let $\mathcal{R}$ be a weak bisimulation up to structural equivalence (resp. weakening, contraction, restriction, parallel composition). Then $\mathcal{R} \subseteq \mathcal{R}_s \subseteq \approx$ (resp. $\mathcal{R} \subseteq \mathcal{R}_t \subseteq \approx$, for $t = w, c, r, p$).*

PROOF: It is obvious from the definition that $\mathcal{R} \subseteq \mathcal{R}_t$, for any t. Let us examine the other inclusion. For $t \in \{s, w, c\}$ the proof simply consists in showing that $\mathcal{R}_t$ is a weak bisimulation. This is straightforward by relying on the the fact that $(\mathcal{R}_t)_t = \mathcal{R}_t$; in the case $t = c$, we also use the fact that, when $\bot \notin \widetilde{\widehat{\sigma}}$ and $n(\widetilde{\zeta}) \subseteq dom(\sigma)$, it holds $\sigma[\widetilde{\widehat{\zeta}}/\widetilde{y}] \rhd P \stackrel{\mu}{\stackrel{\longmapsto}{\delta}} \sigma'[\widetilde{\widehat{\zeta}}/\widetilde{y}] \rhd P'$ if and only if $\sigma \rhd P \stackrel{\mu}{\stackrel{\longmapsto}{\delta'}} \sigma' \rhd P'$ where: if δ is an input then $\delta' = \delta[\widetilde{\zeta}/\widetilde{y}]$, if δ is an output then $\delta' = (\nu\, \widetilde{b})\delta[\widetilde{\zeta}/\widetilde{y}]$, with $\widetilde{b} = n(\zeta) - dom(\sigma)$, and $\delta' = \delta$ otherwise (the proof of this fact uses Lemma A.1). Next, we examine in more detail the two most interesting cases, up to restriction and parallel composition.

- *Up to restriction.* The proof is a variation on the proof for π-calculus found, e.g., in [21]. In order to cope with the fact that the output of a name which is bound by a restriction (like k in $(\nu\, k)\overline{a}k.P$) gives rise to infinitely many transitions, all of which differ by some injective renaming of this bound name, it is convenient to introduce bisimulation *up to injective renaming*.

  A substitution of names $\rho : \mathcal{N} \to \mathcal{N}$ is *injective on* $V \subseteq \mathcal{N}$ if for each $x, y \in V$, $x\rho = y\rho$ implies $x = y$; given any $\sigma = [M_i/x_i]_{i \in I}$ we denote by $\sigma \circ \rho$ the substitution $[M_i\rho/x_i]_{i \in I}$.

We adapt the definition in [21] to our setting and define up to injective renaming as the technique that corresponds to the functional $(\cdot)_{in}$ defined by the rule:

$$\frac{(\sigma_1,\sigma_2) \vdash P_1 \,\mathcal{R}\, P_2 \qquad \rho_i \text{ injective on } \cup_{i=1,2}\, n(\sigma_i) \cup fn(P_i)}{(\sigma_1\circ\rho_1, \sigma_2\circ\rho_2) \vdash P_1\rho_1 \,\mathcal{R}_{in}\, P_2\rho_2}\,.$$

It is straightforward to show that if $\mathcal{R}$ is a bisimulation up to injective renaming then $\mathcal{R}_{in} \subseteq \approx$ (this relies on the fact that the both the transition relation $\xrightarrow{\mu}$ and the relation $\sim$ on environments are preserved by injective renaming). Next, it is easy to show that if $\mathcal{R}$ is a bisimulation up to restriction, then $\mathcal{R}_r$ is a bisimulation up to injective renaming (the proof also uses the fact that $(\mathcal{R}_r)_r = \mathcal{R}_r$), thus we have that $\mathcal{R}_r \subseteq \approx$.

- *Up to parallel composition.* We will show that $\mathcal{R}_p$ is a weak bisimulation up to weakening, restriction and structural equivalence: since these techniques have already been proven sound, this implies $\mathcal{R}_p \subseteq \approx$. Suppose that $(\sigma_1,\sigma_2) \vdash A\,\mathcal{R}_p\,B$, with $A \equiv P \,|\, R\sigma_1$ and $B \equiv Q \,|\, R\sigma_2$ as given by the definition of $\mathcal{R}_p$. One analyzes the possible transitions from $\sigma_1 \rhd A$ and in each case finds a matching (weak) transition from $\sigma_2 \rhd B$.

  We examine only the most delicate case, which is when P and $R\sigma_1$ interact with each other: $\sigma_1 \rhd A \overset{\tau}{\longmapsto} \sigma_1 \rhd A'$, where $A \equiv P|R\sigma_1 \xrightarrow{\tau} (\nu\,\widetilde{h})(P'|R_1) \equiv A'$, $P \xrightarrow{(\nu\,\widetilde{h})\overline{a}\langle M\rangle} P'$ and $R\sigma_1 \xrightarrow{a\,M} R_1$. By virtue of Proposition 4.2(2)(i) applied to $R\sigma_1 \xrightarrow{a\,M} R_1$, there must be some η with $n(\eta) \subseteq dom(\sigma_1)$ s.t. $\widehat{\eta\sigma_1} = a$. Since $(\sigma_1,\sigma_2) \vdash P\,\mathcal{R}\,Q$ and $\mathcal{R}$ is a bisimulation up to parallel composition, it holds that

  $$Q \xRightarrow{(\nu\,\widetilde{k})\overline{a'}\langle M'\rangle} Q' \text{ with } a' = \widehat{\eta\sigma_2} \text{ and } (\sigma_1[M/x], \sigma_2[M'/x]) \vdash P'\,\mathcal{R}_p\,Q' \ (x \text{ fresh}).$$

  Moreover, by virtue of Proposition 4.2(2)(iii) applied to $R\sigma_1 \xrightarrow{a\,M} R_1$ and M', there is R' with $fn(R') \subseteq dom(\sigma_1) \cup \{x\}$ s.t. $R_1 \equiv R'\sigma_1[M/x]$ and $R\sigma_2 \xrightarrow{a'\,M'} \equiv R'\sigma_2[M'/x]$. Hence

  $$Q \,|\, R\sigma_2 \equiv B \Longrightarrow B' \equiv (\nu\,\widetilde{k})(Q' \,|\, R'\sigma_2[M'/x]).$$

  By definition, $(\sigma_1[M/x], \sigma_2[M'/x]) \vdash (P' \,|\, R'\sigma_1[M/x])\,\mathcal{R}_p\,(Q' \,|\, R'\sigma_2[M'/x])$ (here we have used the fact that $(\mathcal{R}_p)_p = \mathcal{R}_p$ for any $\mathcal{R}$), hence, by weakening (to discard the x-entries) and restriction, we get that

  $$(\sigma_1,\sigma_2) \vdash (\nu\,\widetilde{h})(P' \,|\, R'\sigma_1[M/x])\ ((\mathcal{R}_p)_w)_r\ (\nu\,\widetilde{k})(Q' \,|\, R'\sigma_2[M'/x]);$$

  finally, by structural equivalence, $(\sigma_1,\sigma_2) \vdash A'(((\mathcal{R}_p)_w)_r)_s B'$, which is the wanted claim for this case. □

The following Proposition strengthens the congruence rule for parallel composition (C-Par), under the assumption that the involved processes are safe for the appropriate environments. Recall that a process R is *σ-safe* if for each s, whenever $\sigma \rhd R \overset{s}{\underset{u}{\Longmapsto}} \sigma' \rhd R' \overset{(\nu\,\widetilde{b})\overline{a}\langle M\rangle}{\underset{\eta(x)}{\Longmapsto}}$, then $M \in dc(\sigma')$ (hence $\widetilde{b} = \emptyset$).

Proposition B.6 (Proposition 5.6) *Suppose that $(\sigma_1,\sigma_2) \vdash Q_1 \approx Q_2$ and that $(\sigma_1,\sigma_2) \vdash R_1 \approx R_2$. Suppose that, for $i = 1, 2$, Q_i and R_i are σ_i-safe. Then $(\sigma_1,\sigma_2) \vdash Q_1\,|\,R_1 \approx Q_2\,|\,R_2$.*

PROOF: Consider the relation $\mathcal{R}$ defined by:

$(\sigma_1, \sigma_2) \vdash (Q_1 \mid R_1)\ \mathcal{R}\ (Q_2 \mid R_2)$ if and only if:
(a) $(\sigma_1, \sigma_2) \vdash Q_1 \approx Q_2$ and $(\sigma_1, \sigma_2) \vdash R_1 \approx R_2$, and
(b) for $i = 1, 2$, Q_i and R_i are σ_i-safe.

One shows that $\mathcal{R}$ is a weak bisimulation. Suppose that $(\sigma_1, \sigma_2) \vdash Q_1 \,|\, R_1\, \mathcal{R}\, Q_2 \,|\, R_2$. We only examine the most delicate case, that is when Q_1 and R_1 interact with one another. Thus, suppose that

$$\sigma_1 \rhd Q_1 \mid R_1 \stackrel{\tau}{\longmapsto}_{-} \sigma_1 \rhd (\nu\, \widetilde{h})(Q_1' \mid R_1') \tag{8}$$

where $Q_1 \xrightarrow{(\nu\, \widetilde{h})\overline{a}\langle M\rangle} Q_1'$ and $R_1 \xrightarrow{a\, M} R_1'$ (the other cases follow by symmetry). Note that it must be $\widetilde{h} = \emptyset$, because $M \in dc(\sigma_1)$. Since $(\sigma_1, \sigma_2) \vdash Q_1 \approx Q_2$, we get that $Q_2 \xRightarrow{(\nu\, \widetilde{k})\overline{a'}\langle M'\rangle} Q_2'$, for some a', M' and Q_2', such that $(\sigma_1', \sigma_2') \vdash Q_1' \approx Q_2'$ (here $\sigma_1' \stackrel{\text{def}}{=} \sigma_1[M/x]$ and $\sigma_2' \stackrel{\text{def}}{=} \sigma_2[M'/x]$, for a fresh x). Also in this case, $\widetilde{k} = \emptyset$, because $M' \in dc(\sigma_2)$.

Now, since $M \in dc(\sigma_1)$ there is ζ s.t. $n(\zeta) \subseteq dom(\sigma_1)$ and $M = \widehat{\zeta\sigma_1}$ (Lemma A.2(1)). From Lemma A.4, it follows that $\widehat{\zeta\sigma_1} = \{N_i\}_{\widetilde{N}[\widetilde{\jmath}]}$ and that $\widehat{\zeta\sigma_2} = \{N_i'\}_{\widetilde{N'}[\widetilde{\jmath}]}$, for appropriate indices i, $\widetilde{\jmath}$ and cores $\widetilde{N}$ and $\widetilde{N'}$. Hence, from $\sigma_1' \sim \sigma_2'$, we get that $M' = \{N_i'\}_{\widetilde{N'}[\widetilde{\jmath}]}$, hence $M' = \widehat{\zeta\sigma_2}$. Similarly, one finds an η s.t. $n(\eta) \subseteq dom(\sigma_1)$ and $a = \widehat{\eta\sigma_1}$ and $a' = \widehat{\eta\sigma_2}$

Now, using the facts on M and M' established above, from $R_1 \xrightarrow{a\, M} R_1'$ we deduce $\sigma_1 \rhd R_1 \stackrel{a\, M}{\longmapsto}_{\overline{\eta}\zeta} \sigma_1 \rhd R_1'$; hence, from $(\sigma_1, \sigma_2) \vdash R_1 \approx R_2$, we get $\sigma_2 \rhd R_2 \stackrel{a'\, M'}{\Longmapsto}_{\overline{\eta}\zeta} \sigma_2 \rhd R_2'$, where $(\sigma_1, \sigma_2) \vdash R_1' \approx R_2'$. Combining $Q_2 \xRightarrow{\overline{a'}\langle M'\rangle} Q_2'$ and $R_2 \xRightarrow{a'\, M'} R_2'$, we get $Q_2 | R_2 \Longrightarrow Q_2' \mid R_2'$, hence

$$\sigma_2 \rhd Q_2 \mid R_2 \Longmapsto_{-} \sigma_2 \rhd Q_2' \mid R_2'\,.$$

To see that the above weak transition matches (8), let us check conditions (a) and (b) of the definition of $\mathcal{R}$. Indeed, $(\sigma_1, \sigma_2) \vdash Q_1' \approx Q_2'$ (by $(\sigma_1', \sigma_2') \vdash Q_1' \approx Q_2'$ and weakening) and $(\sigma_1, \sigma_2) \vdash R_1' \approx R_2'$ imply condition (a); moreover, for $i = 1, 2$, R_i' and Q_i' are σ_i-safe, because R_i and Q_i are, thus condition (b) is true. □

C Characteristic Formula: Calculus with Pairing

Definition C.1 *Let $\sigma = [M_i/x_i]_{i\in I}$ be a substitution. For each $i \in I$ and $p \in \{l, r\}^*$, let:*

- *N_{ip} be $M_i[\sigma, p]$;*
- *ζ_{ip} be the least expression s.t. $n(\zeta_{ip}) \subseteq \widetilde{x}$ and $\widehat{\zeta_{ip}\sigma} = N_{ip}$;*
- *z_{ip} be some fixed fresh name.*

Let $\widetilde{N}$ be the function that maps each ip to N_{ip}, $\widetilde{\zeta}$ be the function that maps each ip to ζ_{ip} and $\widetilde{z}$ be the function that maps each ip to z_{ip} ($\widetilde{z}$ will also denote the set of all z_{ip}'s). Let ρ_σ be the substitution that maps each z_{ip} to N_{ip}, if $N_{ip} \neq \bot$.

Finally, for any $i \in I$, let M_i^σ be a message s.t.: $n(M_i^\sigma) \subseteq \{z_{ip} : p \in \{l, r\}^*\}$ and $(M_i^\sigma)\rho_\sigma = M_i$. The formula ϕ_σ is then defined as in Table 6. ◇

$$
\begin{array}{ll}
 & \mathtt{let}_{N_{jq}\neq\bot}\ z_{jq} = \zeta_{jq}\ \mathtt{in}\ \Big[\\
(*) & \bigwedge_{N_{ip},N_{jq}\neq\bot} [\mathtt{dec}_{z_{jq}}(z_{ip}) = \bot]\ \wedge [\pi_1(z_{ip}) = \bot]\ \wedge\ [\pi_2(z_{ip}) = \bot] \wedge \\
(a) & \bigwedge_{i\in I} [x_i = M_i^\sigma] \wedge \\
(b) & \bigwedge_{N_{ip}\in\mathcal{N}} name(z_{ip})\ \wedge \bigwedge_{N_{ip}\neq\bot \text{ and } N_{ip}\notin\mathcal{N}} \neg name(z_{ip}) \wedge \\
(c) & \bigwedge_{N_{ip},N_{jq}\neq\bot \text{ and } N_{ip}=N_{jq}} [z_{ip} = z_{jq}]\ \wedge \bigwedge_{N_{ip},N_{jq}\neq\bot \text{ and } N_{ip}\neq N_{jq}} [z_{ip} \neq z_{jq}] \\
 & \Big]
\end{array}
$$

Table 6: The formula ϕ_σ (calculus with pairing)

The existence of expressions ζ_{ip}'s can be easily proven by relying on (the analogue of) Lemma A.2, while in the case of the M_i^σ's, it is sufficient to prove that for each $M \in dc(\sigma)$ there is M' with $n(M') \subseteq \widetilde{z}_{ip}$ s.t. $M'\rho_\sigma = M$ (induction on M). The interested reader can easily supply the details. Note that $fn(\phi_\sigma) \subseteq dom(\sigma)$ and that $\sigma \models \phi_\sigma$.

The proof of Theorem A.11 is easily extended via the following lemma (the analogue of Lemma A.10), where we refer to the notation introduced in Definition C.1:

Lemma C.2 *Consider $\sigma = [M_i/x_i]_{i\in I}$ and $\sigma' = [M'_i/x_i]_{i\in I}$, and let ρ' be a substitution s.t. $dom(\rho') = dom(\rho_\sigma)$ and $\widetilde{Z} \stackrel{\text{def}}{=} range(\rho') \subseteq dc(\sigma')$. Suppose that the following two conditions hold:*

(a) for each Z_1, $Z_2 \in \widetilde{Z}$, $\widehat{\mathtt{dec}_{Z_2}(Z_1)} = \bot$ and Z_1 is not a pair,

(b) for each $i \in I$, it holds that $M'_i = (M_i^\sigma)\rho'$.

Then for each i and p s.t. $M'_i[\sigma', p] \neq \bot$, it holds that $M'_i[\sigma', p] = \rho'(z_{ip})$. Furthermore, for each $i \in I$ it holds that $(\sigma, \sigma') \vdash M_i \sim M'_i$.

Proof: We just outline the proof. There are three steps:

- $kn(\sigma') \subseteq \widetilde{Z}$. To prove this, consider the set $T \stackrel{\text{def}}{=} \{M\rho' : n(M) \subseteq dom(\rho')\}$ and show that $dc(\sigma') \subseteq T$ (by induction on the definition of $dc(\cdot)$). The thesis then follows because $T \cap \mathcal{N} \subseteq \widetilde{Z}$.

- For any M and context $C[\cdot]$ (a message with a 'hole'), if $C[M] = M_i^\sigma$ and $(M\rho')[\sigma', p] \neq \bot$, then $(M\rho')[\sigma', p] = \rho'(z_{i(qp)})$, where q is the position of the hole $[\cdot]$ inside $C[\cdot]$ (this implies $M'_i[\sigma', p] = \rho'(z_{ip})$ when $C[\cdot]$ is the empty context $[\cdot]$). The proof of this fact proceeds by induction on M and relies, for the base case, on the fact that $kn(\sigma') \subseteq \widetilde{Z}$.

- For any M that is a sub-message of M_i^σ it holds that $(\sigma, \sigma') \vdash M\rho_\sigma \sim M\rho'$ (this implies $(\sigma, \sigma') \vdash M_i \sim M'_i$ when M is M_i^σ, by virtue of hypothesis (b)). The proof of this fact proceeds by induction on M, and relies on the fact that $M'_i[\sigma', p] = \rho'(z_{ip})$, which has been proven above. □

References

[1] M. Abadi. Protection in Programming-Language Translations. *ICALP'98*, *Proceedings* (K.G. Larsen, S. Skyum, G. Winskel, Eds.), *LNCS* 1443, pp.868-883, Springer-Verlag, 1998.

[2] M. Abadi. Secrecy by Typing in Security Protocols. *STACS'97*, *Proceedings*, *LNCS* 1281, pp.611-638, Springer-Verlag, 1997. Full version to appear in *Journal of ACM*.

[3] R. Amadio, I. Castellani, D. Sangiorgi. On Bisimulations for the Asynchronous π–calculus. *Theoretical Computer Science*, 195,Elsevier, 1998.

[4] M. Abadi, C. Fournet, G. Gonthier. Secure implementation of channel abstractions. In *Proc. of the 13th IEEE Symposium Logic In Computer Science (LICS'98)*, IEEE Computer Society Press, pp. 105-116, 1998.

[5] M. Abadi, A.D. Gordon. Reasoning about cryptographic protocols in the spi calculus. *CONCUR'97*, *Proceedings* (A. Mazurkiewicz, J. Winkowsky, Eds.), *LNCS* 1243, pp.59-73, Springer-Verlag, 1997.

[6] M. Abadi, A.D. Gordon. A Bisimulation Method for Cryptographic Protocols. *Nordic Journal of Computing*, 5(4):267-303, 1998.

[7] M. Abadi, A.D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1-70, Academic Press, 1999.

[8] M. Boreale, R. De Nicola. Testing equivalence for mobile processes. *Information and Computation*, 120:279-303, Academic Press, 1995.

[9] M. Boreale, R. De Nicola, R. Pugliese. Proof Techniques for Cryptographic Processes. In *Proc. of the 14th IEEE Symposium Logic In Computer Science (LICS'99)*, IEEE Computer Society Press, pp.157-166, 1999.

[10] M. Boreale, D. Sangiorgi. Bisimulation in Name-Passing Calculi without Matching. In *Proc. of the 13th IEEE Symposium Logic In Computer Science (LICS'98)*, IEEE Computer Society Press, pp. 165-175, 1998.

[11] R. De Nicola, M.C.B. Hennessy. Testing Equivalence for Processes. *Theoretical Computers Science*, 34:83-133, Elsevier, 1984.

[12] A.S. Elkjaer, M. Höhle, H. Hüttel, K.O. Nielsen. Towards Automatic Bisimilarity Checking in the Spi Calculus, *Proc. of DMTCS'99+CATS'99*, 1999.

[13] P.D. Lincoln, J.C. Mitchell, M. Mitchell, A. Scedrov. A Probabilistic Poly-time Framework for Protocol Analysis, *ACM Computer and Communication Security (CCS-5)*, pp. 112–121, 1998.

[14] G. Lowe. Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR. *TACAS'96*, *Proceedings* (T. Margaria, B. Steffen, Eds.), *LNCS* 1055, pp. 147-166, Springer-Verlag, 1996.

[15] R. Milner. *Communication and Concurrency*. Prentice Hall International, 1989.

[16] R. Milner. The Polyadic π-calculus: a Tutorial. In *Logic and Algebra of Specification* (F.L. Hamer, W. Brauer, H. Schwichtenberg, Eds.), Springer-Verlag, 1993.

[17] R. Milner, J. Parrow, D. Walker. A calculus of mobile processes, (Part I and II). *Information and Computation*, 100:1-77, Academic Press, 1992.

[18] R. Milner, D. Sangiorgi. Barbed Bisimulation. *ICALP'92*, *Proceedings* (W. Kuich, Ed.), *LNCS* 623, pp.685-695, Springer-Verlag, 1992.

[19] A.W. Roscoe. Modelling and verifying key-exchange using CSP and FDR. In *8th Computer Security Foundations Workshop*, IEEE Computer Society Press, 1995.

[20] M.C. Sanderson. Proof techniques for CCS. Internal Report CST-19-82, Department of Computer Science, University of Edinburgh, 1982.

[21] D. Sangiorgi. On the Bisimulation Proof Method. *Mathematical Structures in Computer Science*, 8:447-479, 1998.

[22] S. Schneider. Verifying Authentication Protocols in CSP. *IEEE Transactions on Software Engineering*, 24(8):743-758, IEEE Computer Society Press, 1998.