

# Checking Security Policies through an Enhanced Control Flow Analysis <sup>\*</sup>

Chiara Bodei<sup>1</sup> Pierpaolo Degano<sup>1</sup> Corrado Priami<sup>2</sup>

<sup>1</sup> Dipartimento di Informatica, Università di Pisa  
Via F. Buonarroti, 2, I-56127 Pisa, Italy – {chiara,degano}@di.unipi.it

<sup>2</sup> Dipartimento di Informatica e Telecomunicazioni, Università di Trento  
Via Sommarive, 1438050 Povo (TN), Italy – priami@science.unitn.it

**Abstract.** We introduce a Control Flow Analysis that statically approximates the dynamic behaviour of mobile processes, expressed in (a variant of) the  $\pi$ -calculus. Our analysis of a system is able to describe the essential behaviour of each sub-system, tracking where and between whom communications may occur. Therefore, we can safely approximate the behaviour of a system plugged in a larger and mainly unknown context, without explicitly analysing it. Quite a lot of possible properties fan out, among which some related to confidentiality and with security policies.

## 1 Introduction

Systems of mobile processes are open systems: they are considered as part of larger and mainly unknown contexts. Their analysis is particularly challenging, because it should describe and predict the behaviour of systems and of their sub-systems, regardless of the possible hostile environment in which they run. Protecting data and resources is crucial in any information management system for secrecy, for integrity and for availability. Enforcing protection requires to state which information flows are allowed and which are not, i.e. to enforce an access control policy w.r.t. principals and resources.

To abstractly represent and study systems, we use (a variant of) the  $\pi$ -calculus [21], which is a model of concurrent communicating processes based on name passing. Names may represent both data and channels that processes exchange. In our framework resources are names (channels and data), while principals are sub-processes. We further assume that read and write are the only modes to use and access resources. We can then exploit the precision of the technique briefly outlined below to statically predict when processes respect a few security policies.

---

<sup>\*</sup> Work partially supported by EU-project DEGAS (IST-2001-32072) and the Progetto MIUR Metodi Formali per la Sicurezza e il Tempo (MEFISTO).

The analysis we present here approximates the behaviour of a system  $P$  and of its sub-systems, plugged in any environment  $E$ , without explicitly analysing  $E$ . In particular, we approximate the possible interactions of  $P$  with  $E$ , that may occur on the channels they share. More precisely,  $E$  can listen on these channels and can therefore acquire new names. Moreover  $E$  can send to  $P$  all the names it has acquired and exploit them as channels. This is reminiscent of the Dolev and Yao model [15]. A richer structure on data, e.g. that of the spi-calculus [1] does not affect deeply our approach and these extensions can be dealt with as in [7].

We use here a specific static technique, Control Flow Analysis (CFA), based on Flow Logic [22] and propose an analysis for the  $\pi$ -calculus that refines the one in [8]. The main idea is to exploit in its definition the notion of “logical” *addresses* of sub-processes, in the style of the enhanced operational semantics [13, 9]. Through them, our CFA can safely approximate the behaviour of each sub-system, tracking where and between which communications may occur. More in detail, the result of our analysis, or *estimate*, is a triple  $(\rho, \eta, \Phi)$ . Actually, things are a bit more complex; we shall come on this issue later on. The first component,  $\rho$ , gives information about the set of channels to which names can be bound at run-time; the second component,  $\eta$ , gives information about the set of channels that can be communicated on given channels and about the *addresses* of participants in the communication. Additionally, an estimate establishes a super-set  $\Phi$  of the knowledge of the environment. This last component is therefore used to implicitly approximate the behaviour of  $E$ , by collecting all the names the environment initially knows and those it can acquire from  $P$ , as described above.

The analysis is carried out by taking into account the abstract syntax tree  $T$  of a process, built by considering parallel composition  $|$  and non-deterministic choice  $+$  as main operators. The path from the root of  $T$  – the whole process – to a node corresponding to a sub-process  $P$  we call *address* of  $P$ . Addresses *implicitly* annotate sub-processes. So, they also offer a way of determining the principals involved in a system. Consider the following three processes, where just for explanation, we make the annotation explicit:

$$P_1 = (\underbrace{a(y)}_{\parallel_0} | \underbrace{\bar{a}b}_{\parallel_1}) \quad P_2 = (\underbrace{a(y)}_{\pm_0} + \underbrace{\bar{a}b}_{\pm_1}) \quad P_3 = (a(y).\bar{a}b)$$

In our CFA, addresses are then used to selectively collect the information of interest, e.g. the values a variable  $y$  may assume at run-time. Clearly,

these values may vary, depending on the different sub-processes where  $y$  occur. So, using addresses we obtain better estimates, *without explicitly* resorting to contexts.

In our example, the operational semantics of the  $\pi$ -calculus tells us that the variable  $y$  in  $P_1$  can be bound to  $b$ , while in  $P_2$  and in  $P_3$  it cannot, unless  $E$  knows  $b$  and sends it along  $a$ . Our present analysis reflects this situation, by statically considering possible only synchronizations between sub-processes in parallel. A communication is predicted only if the receiving sub-process is in parallel with the sending sub-process; technically, if the two co-actions lie on different sides of the same  $|$ . Instead, the analysis proposed in [8] (safely, but imprecisely) predicts that variable  $y$  may be bound to  $b$  in all cases above, even if receiving and sending are mutually exclusive as in  $P_2$ , or reading must occur before sending  $b$ , as in  $P_3$ .

The addresses of the sub-processes within a replicated process require a special handling. Indeed, at run time replication alters significantly the structure of syntax trees, yet in a predictable way. Consider for example the process  $!P$ , where  $P = (a(y).Q + \bar{a}b.R)$ ; a communication between two copies of  $P$  is possible, resulting in the following transition:  $!P \xrightarrow{\tau} (Q | R) !P$ . Our CFA must then take care of a possibly infinite set of addresses. Roughly, in the example above, before the transition, the address of  $a(y).Q$  is  $\pm_0$  (prefixed by a special tag giving information on the fact that it is inside a  $!$ ), while after the transition there is a copy of  $a(y).Q$  at address  $\|_1\pm_0$  (prefixed by the additional special tag); further communications will increase the number of tags  $\|_1$ . We shall tackle this problem and maintain our estimates finite in the following way. First, a *pre-estimate* is computed, considering that  $!P$  originates two copies only in parallel—which agrees with the operational semantics—, but with addresses  $\!_0$  and  $\!_1$  that are the special tags mentioned above. This is the triple  $(\rho, \eta, \Phi)$  or pre-estimate we referred to above. In this way, our analysis uses a finite number of addresses, only. The actual estimate, possibly involving an unbound number of addresses, is then obtained by a suitable closure operation on a pre-estimate, actually only of its  $\eta$  component. The major point here is that we show that *no information* is lost if one considers pre-estimates in place of estimates, as long as one is interested in predicting the flow of information between sub-process, including an attacker. So, we can safely use pre-estimates when checking security properties and leave estimates play a technical role, only.

We shall establish the *semantic correctness* of our analysis in the form of a *reduction theorem* and we shall show that *least pre-estimates always*

*exist*. There is also a *constructive procedure* for obtaining the least solution (pre-estimates, actually), whose complexity we argue is polynomial.

We validate our proposal on a variety of security properties taken from the literature, in particular those based on access policies, thus showing it quite flexible and expressive. The general scheme consists in selecting a specific dynamic security property and defining a static check on estimates that implies the dynamic property, i.e. if a system passes a static security check, then all its computations do enjoy the dynamic property. The major point is that an estimate is generated *once and for all*, while the properties are checked by performing different static tests on the *single* computed estimate. Since our analysis approximates the behaviour of each component of the system under consideration, we can statically predict:

- which kind of actions (read or write) are possible at certain addresses, including those arising from replicated processes,
- which principals can perform them,
- on which channels these actions may take place,
- which objects flow on which channels and
- with which partner communications take place.

The paper is organized as follows. The next section briefly surveys our version of the  $\pi$ -calculus. Section 3 introduces addresses of sub-processes, based on the structure of abstract syntax trees. We develop in Section 4 a static analysis for tracking communications in our calculus: we show its semantic correctness and demonstrate that pre-estimates always exist. We also sketch how to obtain them. Finally, in Section 5 we apply our CFA to establish properties related to secrecy. We conclude with Section 7. Appendix contains the proofs of our main results.

## 2 The Process Calculus

In our approach, channels are interfaces and processes are interested in filtering received values, e.g. in accepting a valid password or a PIN and refusing invalid ones. For this reason, we slightly change the standard  $\pi$ -calculus [21] obtaining an equally expressive variant of it.

The main difference with the standard calculus is the absence of an explicit matching construct, replaced by a selection on the values received in communications. Values are accepted only if they are included in the selection set. In other words, inputs only succeed on values matching one of the names in the set. The remaining part of the calculus is fairly standard.

*Syntax* For the following treatment, it is convenient to partition the set of names in *values* and *variables*. Intuitively, the first set,  $Val$ , ranged over by  $a, b, c, d \dots$ , contains all those names that may occur free or restricted in processes. When a value  $a$  is used as a channel on which other values flow, we sometimes shall call  $a$  a channel. The variables in  $Var$ , ranged over by  $x, y, w, z \dots$ , are those names occurring within input prefixes.

Furthermore, to simplify the definition of our control flow analysis in Section 4, we discipline the  $\alpha$ -renaming of bound values and variables. To do it in a simple and “implicit” way, we assume that values and variables are “stable”, i.e. that for each value  $a \in Val$  there is a canonical representative  $[a]$  for the set  $\{a, a_0, a_1, \dots\}$  and similarly, for each variable  $x \in Var$  there is a canonical representative  $[x]$  for the set  $\{x, x_0, x_1, \dots\}$ . Then, we discipline  $\alpha$ -conversion as follows: two values (resp. variables) are  $\alpha$ -convertible only when they have the same canonical value (resp. variable). In this way, we statically maintain the identity of values and variables that may be lost by freely applying  $\alpha$ -conversions. Hereafter, we shall simply write  $a$  (resp.  $x$ ) for  $[a]$  (resp.  $[x]$ ). We also assume that all the bound values and variables are kept distinct and that the bound values never clash with the free ones.

**Definition 1.** *Processes are defined according to the following syntax, where  $Y \subseteq (Val \cup Var)$  is a finite set.*

$\mathcal{P} \ni P, Q, R ::=$		<b>processes</b>	
$\mathbf{0}$	<i>inaction</i>	$\pi ::=$	<b>prefixes</b>
$\pi.P$	<i>prefix</i>	$\tau$	<i>silent prefix</i>
$(\nu a)P$	<i>restriction</i>	$x(y \in Y)$	<i>selective input</i>
$P + P$	<i>nondeterministic choice</i>	$\bar{x}y$	<i>output</i>
$P P$	<i>parallel composition</i>		
$!P$	<i>replication</i>		

The prefix  $\pi$  is the first atomic action that the process  $\pi.P$  can perform. The silent prefix  $\tau$  denotes an action which is invisible to an external observer of the system. The output prefix does not bind the name  $a$  which is sent along  $x$ .

The input prefix  $x(y \in Y)$  binds the name  $y$  in the prefixed process  $P$ . Intuitively, some name  $b$  will be received along the link named  $x$  activating the process  $P$  (where  $b$  substitutes  $y$ ) only if  $b \in Y$ . Variables can occur in  $Y$ , some of which can be instantiated to names by firing the input prefixes that bind them; we assume that  $y \notin Y$ .

Summation denotes nondeterministic choice. The operator  $|$  describes parallel composition of processes. The operator  $(\nu a)$  acts as a static binder

$Tau : \tau.P \xrightarrow{\tau} P$	$Out : \bar{a}b.P \xrightarrow{\bar{a}b} P$	$Sel\_Ein : a(y \in Y).P \xrightarrow{ab} P\{b/y\}, b \in Y$
$Sum_0 : \frac{P_0 \xrightarrow{\mu} Q_0}{P_0 + P_1 \xrightarrow{\mu} Q_0 + \mathbf{0}}$	$Par_0 : \frac{P_0 \xrightarrow{\mu} Q_0}{P_0 P_1 \xrightarrow{\mu} Q_0 P_1}, bn(\mu) \cap fn(P_1) = \emptyset$	
$Open : \frac{P \xrightarrow{\bar{a}b} Q}{(\nu b)P \xrightarrow{\bar{a}(b)} Q}, b \neq a$	$Res : \frac{P \xrightarrow{\mu} Q}{(\nu a)P \xrightarrow{\mu} (\nu a)Q}, a \notin vals(\mu)$	
$Com_0 : \frac{P_0 \xrightarrow{\bar{a}b} Q_0, P_1 \xrightarrow{ab} Q_1}{P_0 P_1 \xrightarrow{\tau} Q_0 Q_1}$	$Close_0 : \frac{P_0 \xrightarrow{\bar{a}(b)} Q_0, P_1 \xrightarrow{ab} Q_1}{P_0 P_1 \xrightarrow{\tau} (\nu b)(Q_0 Q_1)}$	
$Bang : \frac{P P \xrightarrow{\mu} Q}{!P \xrightarrow{\mu} Q !P}$	$Var : \frac{P' \equiv P \xrightarrow{\mu} Q \equiv Q'}{P' \xrightarrow{\mu} Q'}$	

**Table 1.** Early transition system for the  $\pi$ -calculus (symmetric rules are omitted).

for the name  $a$  in the process  $P$  that it prefixes. In other words,  $a$  is a unique name in  $P$  which is different from all the external names. Finally,  $!P$  behaves as many copies of  $P$  as needed, put in parallel.

*Semantics* Our operational semantics for the  $\pi$ -calculus is an *early* one and is defined in SOS style; the labels for transitions are  $\tau$  for silent action,  $ab$  for input,  $\bar{a}b$  for free output and  $\bar{a}(b)$  for bound output. We use  $\mu$  as metavariable for the labels of transitions, distinct from the metavariable  $\pi$  of prefixes. The sets of free values  $fn(\cdot)$ , of bound values  $bn(\cdot)$  (and of values  $vals(\cdot) = fn(\cdot) \cup bn(\cdot)$ ), as well as those of free variables  $fv(\cdot)$ , of bound variables  $bv(\cdot)$  (and of variables  $vars(\cdot) = fv(\cdot) \cup bv(\cdot)$ ) are defined much in the standard way. E.g.  $fn(a(y \in Y).P) = \{a\} \cup vals(Y) \cup fn(P)$ , and  $fv(a(y \in Y).P) = (vars(Y) \cup fv(P)) \setminus \{y\}$ .

Our static approximations exploit the structure of processes, so we need to preserve it as much as possible under this dynamic evolution. This is the main reason why our semantics slightly differs from the standard one.

In particular, our congruence has *no* rule for making the parallel and nondeterministic operators associative and commutative or for considering inaction as a neutral element, i.e. monoidal laws are not valid here. In this way, the abstract syntax tree of a process, or more precisely its structure, cannot be altered by the application of a congruence rule. The

*structural congruence*  $\equiv$  on processes is then the least congruence satisfying:

- $P \equiv Q$  if  $P$  and  $Q$  are  $\alpha$ -equivalent in the disciplined way discussed above
- $(\nu a)(\nu b)P \equiv (\nu b)(\nu a)P$ ;
- $(\nu a)P \equiv P$  and  $(\nu a)(P | Q) \equiv (\nu a)P | Q$ , if  $a \notin fn(Q)$

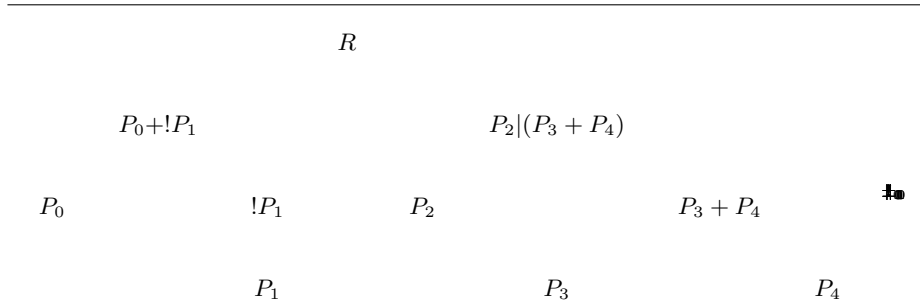
Tab. 1 displays the rules of the operational semantics of our calculus. The symmetric rules for  $Sum_0$ ,  $Par_0$ ,  $Close_0$  and  $Com_0$  are omitted. As written above, our rules slightly differ from the standard *early* ones, in particular input and summation are treated differently. So we discuss these cases in some detail below. Also, we use the variant of the standard rules for replication of [27], in which two copies in parallel  $P|P$  of  $!P$  make a move: either a single copy does or the two communicate each other.

The placeholder of the input is instantiated in the axiom for selective input (*Sel\_Ein*):  $a(y \in Y).P \xrightarrow{ab} P\{b/y\}$ , only if  $b \in Y$ . For any  $b$  and  $y$ ,  $P\{b/y\}$  denotes the operation of substituting  $b$  for the free occurrences of  $y$  in the process  $P$ . In particular, if  $P$  is on the form  $c(z \in \{y\} \cup Y).P'$ , then  $P\{b/y\} = c(z \in \{b\} \cup Y).P'\{b/y\}$ . Here we are somehow more demanding than the standard semantics that allows everything to be read from outside, with a so-called free input. Intuitively, our selective input implements a filter on the messages that a process is willing to accept. Note that our selective input suffices to encode the standard matching construct  $[x = y]P$  of the  $\pi$ -calculus as  $(\nu a)(\bar{a}x \mid a(y \in \{x\}).P)$ . Additional work is needed to encode the selective input in the  $\pi$ -calculus; roughly one has an input followed by a sequence of matchings covering  $Y$ , taking care that the input can anyway occur even if a matching fails.

As for the *Sum* rules, the “structure” with respect to  $+$  of the process is preserved leaving a  $\mathbf{0}$  in the place of the process not chosen, i.e. if  $P_0 \xrightarrow{\mu} Q_0$  then  $P_0 + P_1 \xrightarrow{\mu} Q_0 + \mathbf{0}$  (and, symmetrically, if  $P_1 \xrightarrow{\mu} Q_1$  then  $P_0 + P_1 \xrightarrow{\mu} \mathbf{0} + Q_1$ ). Note that this change does not affect at all the standard meaning of the nondeterministic operator.

Our peculiar rules for summation and the absence of monoidal laws for summation and parallel composition preserve the structure of a process, in a way made precise in the next section. Instead, as we have seen in the Introduction, replication alters this structure, yet in a predictable manner<sup>3</sup>. Process structure is particularly useful for our analysis and we will see below how we are able to exploit it and to handle its changes.

<sup>3</sup> Of course the structure of the target of the transition  $\bar{a}b.(P_0|P_1) \xrightarrow{\bar{a}b} (P_0|P_1)$  is somehow different from that of the source.



**Fig. 1.** The tree of (sequential) processes of  $R = (P_0+!P_1)|(P_2|(P_3+P_4))$ .

---

### 3 On the Structure of Processes

To approximate the behaviour of sub-processes in a process, we need a way to single them out. As already said, we do it by exploiting the abstract syntax trees of processes, where arcs are ordered, so we use no explicit centralized naming service. These trees are built using the binary parallel composition  $|$ , the nondeterministic choice  $+$  and replication  $!$  as main operators. Given a process  $R$ , the nodes of its tree (see e.g. Fig. 1) correspond to the occurrences of  $|$ ,  $+$  and  $!$  in  $R$ , and its leaves are the sequential components of  $R$  (roughly, those processes whose top-level operator is a prefix).

We sketch now how to use these abstract syntax trees. We consider first the operators  $|$  and  $+$ , and we recall that they are neither commutative nor associative nor they have a neutral element; the operator  $!$  will be discussed later on. Since we cannot swap the arguments of  $|$  and  $+$ , it comes natural to assume that the left (resp. right) branches of an abstract syntax tree denote the left (resp. right) component of a  $|$  or of a  $+$  operator, and so we label their arcs with tags  $\|_0$  and  $\pm_0$  (resp.  $\|_1$  and  $\pm_1$ ). Now, consider the path from the root of the abstract syntax tree to one of its leaves. For example, in Fig. 1, consider the path labelled  $\|_1\|_1\pm_0$  that leads from the root, i.e. the whole process  $R$  to the sub-process  $P_3$ . This label we call *address* of  $P_3$  *within*  $R = (P_0+!P_1)|(P_2|(P_3+P_4))$ . As a matter of fact, each address *uniquely identifies* a sub-process within a given process.

We consider now the  $!$  operator. In our example we only have  $!P_1$  and we do not make further explicit the internal structure of the body  $P_1$  itself. In order to deduce a transition for  $!P_1$ , one has to “create” two copies of  $P_1$  (see rule *Bang* in Tab. 1) and possibly unfold these copies inductively. The format of the rule *Bang* thus suggests us to signal that

the body of  $!P_1$  is reachable in two ways: one leading to the left copy of  $P_1|P_1$ , the other to the right one. In analogy with  $|$  and  $+$ , we therefore use two tags  $!_0, !_1$  to label the *same* arc outgoing from the node  $!$ . In Fig. 1, the path  $||_0\pm_1!_0$  leads to  $P_1$ , just as the equivalent path  $||_0\pm_1!_1$ .

Recapitulating, given a process  $R$ , any one of its sub-processes  $P$  is identified by a string  $\vartheta \in \mathcal{A} = \{||_i, \pm_i, !_i \mid i = 0, 1\}^*$ . The string  $\vartheta$  is called *address* of  $P$  and corresponds to the path from the root, i.e. the whole process  $R$ , to  $P$ . By the sake of explanation, we give the addresses of all the other sub-processes of the process leaves of the abstract syntax tree in Fig. 1. The address of  $P_0$  is  $||_0\pm_0$ , the address of  $P_2$  is  $||_1||_0$  and the address of  $P_4$  is  $||_1||_1\pm_1$ .

Conversely, given any process  $R$ , it is possible to localize its sub-process  $P$ , if any, whose address is  $\vartheta$ . This is done via the operator  $@\vartheta$ , defined below. In Fig. 1, e.g.  $R@||_1||_1\pm_0 = P_3$  and  $R@||_1||_1 = P_3 + P_4$ , but  $R@||_1||_1||_1$  is undefined. In fact,  $@$  is not a total operator, and, given a process  $P$ , it will be convenient to collect in a set the addresses that lead to sub-processes of  $P$ .

Since the two operators  $|$  and  $+$  often play a similar role from now onwards, we shall use the following

**Notation.** For  $i = 0, 1$ , let  $\diamond_i \in \{||_i, \pm_i\}$  and let  $op_\diamond$  be  $|$  whenever  $\diamond_i = ||_i$  and  $+$  whenever  $\diamond_i = \pm_i$ .

**Definition 2.** The localization operator  $@\vartheta$  is defined on processes by induction ( $\epsilon$  is the empty address):

1.  $P@\epsilon = P$ ;
2.  $((\nu a)P)@\vartheta = P@\vartheta$ ;
3.  $(P_0 op_\diamond P_1)@\diamond_i \vartheta = P_i@\vartheta$ ;
4.  $!P@!_i \vartheta = P@\vartheta$ , for  $i \in \{0, 1\}$ .

The set  $Addr(P) = \{\vartheta \mid \exists Q. P@\vartheta = Q\}$  collects all the addresses of the sub-processes of  $P$ .

From a semantic point of view, a communication can take place only if the two complementary actions are *compatible*, i.e. if they lay on the *same* side of a  $+$ , if any, and on *different* sides of the same parallel composition operator  $|$ . Furthermore, we have that all the actions inside the replicated part of a process are considered compatible with the ones in another copy of the same process and compatible with those in the rest of the process. Therefore, two addresses  $\vartheta$  and  $\vartheta'$  are compatible if they share the same prefix, followed by any two addresses that however begin

with different tags, recording the presence of a parallel or a replication operator. Formally, we define the following symmetric relation.

**Definition 3.** *Given two addresses  $\vartheta, \vartheta' \in \mathcal{A}$ ,  $\vartheta$  and  $\vartheta'$  are compatible, written as  $\text{comp}(\vartheta, \vartheta')$ , if and only if:*

$$\vartheta = \vartheta_0 \parallel_i \vartheta_1 \quad \text{and} \quad \vartheta' = \vartheta_0 \parallel_{1-i} \vartheta'_1 \quad \text{for } i \in \{0, 1\} \quad \text{or}$$

$$\vartheta = \vartheta_0 !!_i \vartheta_1 \quad \text{and} \quad \vartheta' = \vartheta_0 !!_{1-i} \vartheta'_1 \quad \text{for } i \in \{0, 1\}$$

Clearly, for all  $\vartheta$ ,  $\text{comp}(\vartheta \vartheta_0, \vartheta \vartheta_1)$  if  $\text{comp}(\vartheta_0, \vartheta_1)$ .

Back to our example in Fig. 1,  $P_0$  and  $P_3$ , i.e.  $R@ \parallel_0 \pm_0$  and  $R@ \parallel_1 \parallel_1 \pm_0$  (here  $\vartheta_0$  is  $\epsilon$ ) may communicate and so do two parallel sub-components of  $P_0$ , if any (for which  $\vartheta_0$  is  $\parallel_0 \pm_0$ ). On the contrary,  $P_0$  can communicate with no copy of  $!P_1$ : their addresses are  $\parallel_0 \pm_0$  and  $\parallel_0 \pm_1$ , showing that only one out of  $P_0$  and (a copy of)  $!P_1$  can be active. Instead, any two copies of  $!P_1$  can communicate, provided that  $P_1$  may fire two complementary prefixes; indeed, the address  $\parallel_0 \pm_1 !!_1$  is compatible with  $\parallel_0 \pm_1 !!_0$ .

The notion of address compatibility is vindicated by the Property 1, that needs an auxiliary definition first. Its purpose is to single out in a process, performing a transition  $t$ , the sub-process (or the sub-processes) that acts in  $t$ . This is done exploiting the localization operator and visiting backwards the proof of  $t$  until an axiom is reached.

**Definition 4.** *A transition  $P \xrightarrow{\mu} Q$  involves, i.e. has been deduced with, the axiom  $P@ \vartheta \xrightarrow{\mu'} R$  iff*

- $\vartheta = \epsilon$ ,  $\mu = \mu'$ , and  $P \equiv \mu.R$ ;
- $P \equiv (\nu a)P'$  then for suitable  $Q'$  the transition  $P' \xrightarrow{\mu} Q'$  involves  $P@ \vartheta \xrightarrow{\mu'} R$ ;
- $\vartheta = \pm_i \vartheta_i$ ,  $P \equiv P_0 + P_1$ , and for suitable  $Q_i$  the transition  $P_i \xrightarrow{\mu} Q_i$  involves  $P_i@ \vartheta_i \xrightarrow{\mu'} R$ ;
- $\vartheta = \parallel_i \vartheta_i$ ,  $P \equiv P_0 | P_1$ , and for suitable  $Q_i$  the transition  $P_i \xrightarrow{\mu''} Q_i$  involves  $P_i@ \vartheta_i \xrightarrow{\mu} R$ , with  $\mu'' \in \{\mu, \mu'\}$ ;  $R'; \xrightarrow{\mu} R$ .
- $\vartheta = !!_i \vartheta$ ,  $P \equiv !S$  and for suitable  $Q'$  the transition  $S|S \xrightarrow{\mu} Q'$  involves  $S@ \vartheta \xrightarrow{\mu'} R$ .

Note that a single transition may involve one or two axioms. In the second case, the transition is on the form  $P \xrightarrow{\tau} Q$  and the involved axioms are  $P@ \vartheta_0 \xrightarrow{\mu_0} R_0$  and  $P@ \vartheta_1 \xrightarrow{\mu_1} R_1$ . In particular, in the fourth case, if  $\vartheta_0 = \parallel_i \vartheta'_0$  and  $\vartheta_1 = \parallel_i \vartheta'_1$  the label is then  $\mu'' = \mu = \tau$ , while

if  $\vartheta_0 = \parallel_i \vartheta'_0$  and  $\vartheta_1 = \parallel_{1-i} \vartheta'_1$  then the label is  $\mu'' = \mu_i$ , for  $i \in \{0, 1\}$ . Actually, if the transition is due to the application of a *Close*, we have that the axiom involved is on the form  $\mu_i = \bar{a}b$  and  $\mu'' = \bar{a}(b)$ .

*Example 1.* Consider the following process

$$\begin{aligned} \text{Simp\_Sys} &= P \mid Q \\ P &= \bar{a}c.b(z \in \{a, c\}) + \bar{a}e \\ Q &= a(x \in \{c\}).Q' \end{aligned}$$

We have that  $\text{Addr}(\text{Simp\_Sys}) = \{\parallel_0 \pm_0\} \cup \{\parallel_0 \pm_1\} \cup \{\parallel_1.\text{Addr}(Q')\}$ . The following transition is possible,

$$\text{Simp\_Sys} \xrightarrow{\tau} (b(z \in \{a, c\}) + \mathbf{0}) \mid Q'\{c/x\},$$

in which  $Q$  receives on channel  $a$  the message  $c$  sent by  $P$  and  $c$  replaces  $x$  in  $Q'$ . The transition involves  $\text{Simp\_Sys}@_{\parallel_0 \pm_0} = \bar{a}c.b(z \in \{a, c\}) \xrightarrow{\bar{a}c} b(z \in \{a, c\})$  because  $(\bar{a}c.b(z \in \{a, c\}) + \bar{a}e)@_{\pm_0} \xrightarrow{\bar{a}c} b(z \in \{a, c\})$  involves the axiom  $(\bar{a}c.b(z \in \{a, c\}) + \bar{a}e)@_{\pm_0} \xrightarrow{\bar{a}c} b(z \in \{a, c\})$ . Furthermore, it involves  $\text{Simp\_Sys}@_{\parallel_1} = a(x \in \{c\}).Q' \xrightarrow{ac} Q'\{c/x\}$ . The communication above involves two axioms, and indeed its label  $\mu = \tau$  is different from the ones of the two axioms ( $\bar{a}c$  and  $ac$ ). Instead, if a single axiom is involved, then  $\mu = \mu'$ . E.g., the transition  $\text{Simp\_Sys} \xrightarrow{\bar{a}c} (b(z \in \{a, c\}) + \mathbf{0}) \mid Q$  involves  $\bar{a}c.b(z \in \{a, c\}) \xrightarrow{\bar{a}c} b(z \in \{a, c\})$ .

The following property links compatibility with the transitions that originate a communication; its proof is by straightforward induction.

*Property 1.* If  $P \xrightarrow{\tau} P'$  involves both  $P@_{\vartheta_0} \xrightarrow{\mu_0} R_0$  and  $P@_{\vartheta_1} \xrightarrow{\mu_1} R_1$ , then  $\text{comp}(\vartheta_0, \vartheta_1)$ .

Just as the definition of the axiom(s) involved in a transition  $t$  talks about the source of  $t$ , the following definitions describe how  $t$  transforms its source into its target. In absence of replication, the effects of a transition involving only the axiom  $P@_{\vartheta} \xrightarrow{\mu} R$  consists in substituting  $R$  for the sub-process  $P@_{\vartheta}$  within  $P$ ; additionally, whenever  $\vartheta$  can be split in  $\vartheta_0 \pm_i \vartheta_1$ , i.e. whenever a choice  $P@_{\vartheta_0} = P@_{\vartheta_0 \pm_0} + P@_{\vartheta_0 \pm_1}$  has been resolved in favour of  $P@_{\vartheta_0 \pm_i}$ , the discarded process  $P@_{\vartheta_0 \pm_{1-i}}$  has to be replaced by  $\mathbf{0}$ , due to our peculiar rules *Sum*. Firing a replicated process  $!P$  affects considerably the target: a way of seeing it is to expand the source  $!P$  into  $(P \mid P) \mid !P$  and proceed as for the bang-free case.

We will formally describe the above transformations in two steps:

- a function  $h$  first transforms the structure of the source in the structure of the target, (a) by substituting  $\mathbf{0}$  for the discarded processes and (b) by expanding the relevant replicated sub-processes  $!S$  in  $(S|S)!S$ .
- a second function, called *localized substitution*, substitutes  $R_0$  for  $P@v_0$  at the right addresses (and  $R_1$  for  $P@v_1$ ) within the process, if the transition involves  $P@v_0 \xrightarrow{\mu_0} R_0$  ( $P@v_1 \xrightarrow{\mu_1} R_1$ ) and leaves the rest as it is.

Both functions depend on the address(es)  $v$  (and on  $v'$ ) of the sub-process(es) of  $P$  involved by the axiom(s), whose tags drive the transformations. When a single axiom  $P@v \xrightarrow{\mu} R$  is involved, function  $h$  is on the form  $h(1, P, v, v')$ , where the fourth argument is dummy (and put equal to the third). If two axioms  $P@v_0 \xrightarrow{\mu_0} R_0$  and  $P@v_1 \xrightarrow{\mu_1} R_1$  are involved, then we have  $h(2, P, v, v')$ . Furthermore the localized substitution is applied twice.

**Definition 5.** Let  $P$  a process,  $v, v' \in \text{Addr}(P)$ ,  $i \in \{0, 1\}$  and  $k \in \{1, 2\}$ . We define the function  $h : \{0, 1\} \times \mathcal{P} \times \text{Addr} \times \text{Addr} \rightarrow \mathcal{P}$  as

- $h(1, P, \epsilon, \epsilon) = P$ ;
- $h(k, (\nu a)P, v, v') = (\nu a)h(k, P, v, v')$ ;
- $h(k, P_0 \text{ op}_\diamond P_1, \diamond_i v, \diamond_i v') = P'_0 \text{ op}_\diamond P'_1$ , where  $P'_i = h(k, P_i, v, v')$  and  $P'_{1-i} = \begin{cases} P_{1-i} & \text{if } \diamond_i = \parallel_i \\ \mathbf{0} & \text{if } \diamond_i = \pm_i \end{cases}$ ;
- $h(2, P_0 \mid P_1, \parallel_i v, \parallel_{1-i} v') = P'_0 \text{ op}_\diamond P'_1$ , where  $P'_i = h(k, P_i, v, v')$  and  $P'_{1-i} = h(k, P_{1-i}, v', v')$ ;
- $h(k, !P, !_i v, !_j v') = h(k, (P|P), \parallel_i v, \parallel_j v') \mid !P$ .

To determine the place where a particular substitution takes place, we exploit addresses, as shown in the next definition.

**Definition 6.** Let  $P$  and  $R$  be two processes and  $v \in \{\parallel_i, \pm_i \mid i = 0, 1\}^*$  be a bang-free address of  $P$ . Then, the localized substitution of  $R$  at  $v$  within  $P$ , written as  $P[v \mapsto R]$  is defined as:

- $P[\epsilon \mapsto R] = R$ ;
- $((\nu a)P)[v \mapsto R] = (\nu a)(P[v \mapsto R])$ ;
- $(P_0 \text{ op}_\diamond P_1)[\diamond_i v \mapsto R] = P'_0 \text{ op}_\diamond P'_1$ , where  $\begin{cases} P'_i = P_i[v \mapsto R] \\ P'_{1-i} = P_{1-i} \end{cases}$

Note that  $P[v'v'' \mapsto R] = P@v'[v'' \mapsto R]$ . Moreover, note that, due to the absence of occurrences of  $!_i$  in  $v$ ,  $P' = P[v \mapsto R]$  can alternatively be defined as the process such that  $P'@v' = \begin{cases} R & v' = v; \\ P@v' & \text{otherwise.} \end{cases}$

There is a further auxiliary definition we need to apply the localized substitution in order to obtain the target of a transition. Again, the problem consists in taking into account any possible expansion of processes replicated, in order to localize correctly the images of all sub-processes in the source, in the changed structure of the target. The auxiliary function  $w(\cdot)$  transforms each occurrence of  $\mathbb{!}_i$  in the corresponding sequence  $\|_0\|_i$ , thus obtaining a bang-free address.

**Definition 7.** *Given an address  $\vartheta$ , we define the function  $w : \text{Addr} \rightarrow \text{Addr}$  such that*

$$-w(\epsilon) = \epsilon; \quad -w(\diamond_i \vartheta) = \diamond_i w(\vartheta); \quad -w(\mathbb{!}_i \vartheta) = \|_0\|_i w(\vartheta).$$

Now, we have all the ingredients to determine the form of the target of a transition, in terms of the axioms involved, as follows.

**Lemma 1.** *If the transition  $P \xrightarrow{\alpha} Q$  involves*

- (a) *only the axiom  $P@i \xrightarrow{\alpha} R_i$  then  $Q = h(1, P, \vartheta_i, \vartheta_i)[w(\vartheta_i) \mapsto R_i]$ , with  $i \in \{0, 1\}$ ;*
- (b) *both axioms  $P@i_0 \xrightarrow{\mu_0} R_0$  and  $P@i_1 \xrightarrow{\mu_1} R_1$  then  $Q = h(2, P, \vartheta_0, \vartheta_1)[w(\vartheta_0) \mapsto R_0, w(\vartheta_1) \mapsto R_1]$ .*

Note that the two addresses  $\vartheta_0$  and  $\vartheta_1$  above are compatible and so are  $w(\vartheta_0)$  and  $w(\vartheta_1)$ ; thus the two localized substitutions do not interfere. Also, the structure of  $Q$  and the one of  $H = h(k, P, \vartheta_0, \vartheta_1)$  are the same: the main difference consists in the sub-processes at address  $w(\vartheta_i)$ , that in  $H$  coincide with  $P@i$  and in  $Q$  with  $R_i$ .

*Example 2.* Consider the following system  $Sys$ .

$$\begin{aligned} Sys &= !P \mid Q \\ P &= \bar{a}c.b(z \in \{a, c\}) + \bar{a}e \\ Q &= a(x \in \{c\}).Q' \\ Q' &= \bar{b}\langle x \rangle.\mathbf{0} \mid R \end{aligned}$$

$!P$  represents a source of infinitely many outputs on  $a$  of the message  $c$ . In the following transition,  $Q$  receives the message  $c$  sent on channel  $a$  by a copy of  $!P$ .

$$Sys \xrightarrow{\tau} Sys' = ((b(z \in \{a, c\}) + \mathbf{0}) \mid \bar{a}c.b(z \in \{a, c\}) + \bar{a}e) \mid !P \mid (\bar{b}\langle c \rangle.\mathbf{0} \mid R)$$

The axioms involved are:

$$Sys@_0\|_0\pm_0 \xrightarrow{\bar{a}c} b(z \in \{a, c\}) \quad \text{and} \quad Sys@_1\|_1 \xrightarrow{ac} Q'\{c/x\}.$$

Therefore the target process  $Sys'$  is  $Sys''[w(\|_0!_0\pm_0) \mapsto P', w(\|_1) \mapsto Q'\{c/x\}]$ , with  $Sys'' = h(2, Sys, \|_0!_0\pm_0, \|_1)$ . By definition of  $h$  and of  $w$ , we have that

- $w(\|_0!_0\pm_0) = \|_0\|_0\|_0\pm_0$  and  $w(\|_1) = \|_1$ ;
- $Sys'' = h(1, !P, \|_0\pm_0, \|_0\pm_0) \mid h(1, Q, \epsilon, \epsilon)$ ;
- $h(1, Q, \epsilon, \epsilon) = \bar{b}\langle x \rangle.\mathbf{0} \mid R$ ;
- $h(1, !P, \|_0\pm_0, \|_0\pm_0) = h(1, (P\mid P), \|_0\pm_0, \|_0\pm_0) \mid !P =$   
 $h(1, P, \pm_0, \pm_0) \mid P \mid !P = ((\bar{a}c.b(z \in \{a, c\}) + \mathbf{0}) \mid P) \mid !P$ .

By summarizing,  $Sys'' = ((\bar{a}c.b(z \in \{a, c\}) + \mathbf{0}) \mid P) \mid !P \mid (\bar{b}\langle x \rangle.\mathbf{0} \mid R)$  and  $Sys' = Sys''[\|_0\|_0\|_0\pm_0 \mapsto P', \|_1 \mapsto Q'\{c/x\}]$

It is easy to prove the following corollary, ensuring that a process  $P$  can make a transition leading in  $Q$  if and only if  $H$  can.

**Corollary 1.** *Given a process  $P$ , then  $P \xrightarrow{\mu} Q$  involves  $P@v_0 \xrightarrow{\mu_0} R_0$  (and  $P@v_1 \xrightarrow{\mu_1} R_1$ ) iff  $H = h(k, P, v_0, v_1) \xrightarrow{\mu} Q$  involves  $H@w(v_0) \xrightarrow{\mu_0} R_0$  (and  $H@w(v_1) \xrightarrow{\mu_1} R_1$ ).*

Back to our example, the transition  $Sys = !(\bar{a}c.b(z \in \{a, c\}) + \bar{a}e) \mid Q \xrightarrow{\tau} Sys' = ((b(z \in \{a, c\}) + \mathbf{0}) \mid \bar{a}c.b(z \in \{a, c\}) + \bar{a}e) \mid !P \mid (\bar{b}\langle c \rangle.\mathbf{0} \mid R)$ , that involves the axioms (i)  $Sys@_i \xrightarrow{\bar{a}c} b(z \in \{a, c\})$  and (ii)  $Sys@_1 \xrightarrow{ac} Q'\{c/x\}$  is such that that  $Sys'@w(\|_0!_1) = Sys'@_i$  and  $Sys'@w(\|_1) = Sys'@_1 = Sys@_i$ .

## 4 Control Flow Analysis

We are interested in analyzing the behaviour of a process  $P$  plugged in any environment  $E$ , without explicitly analysing  $E$ . More precisely, our implicit analysis of the environment amounts to considering the most powerful context sharing public names with  $P$ . (Technically, in process algebras like  $\pi$ -calculus, a process  $Q$  cannot know the restricted names of another process  $P$ , unless  $P$  did not send them to  $Q$ .)

### 4.1 Validation

The aim of the analysis is to provide a safe approximation to the dynamic behaviour of processes in terms of which messages are exchanged and between which. More precisely, the result of analysing a process  $P$  is a triple  $(\rho, \widehat{\eta}, \Phi)$ , called *estimate* or *solution*, that, roughly, establishes a super-set of the set of (abstract) values to which the program objects

$(\rho, \eta, \Phi) \models^\vartheta \mathbf{0}$	iff <i>true</i>
$(\rho, \eta, \Phi) \models^\vartheta \tau.P$	iff $(\rho, \eta, \Phi) \models^\vartheta P$
$(\rho, \eta, \Phi) \models^\vartheta \bar{x}y.P$	iff $(\rho, \eta, \Phi) \models^\vartheta P \wedge$ $\forall a \in \rho(x) : \rho(y) \subseteq \eta_1(\vartheta)(a) = J \wedge$ $(a \in \Phi \wedge J \neq \emptyset) \Rightarrow \begin{cases} \eta_1(\vartheta)(a) \subseteq \Phi \\ \eta_2(\vartheta)(a) \ni \vartheta_E \end{cases}$
$(\rho, \eta, \Phi) \models^\vartheta x(y \in Y).P$	iff $(\rho, \eta, \Phi) \models^\vartheta P \wedge$ $\forall a \in \rho(x), \forall \vartheta' : \text{comp}(\vartheta, \vartheta') :$ $J = (\eta_1(\vartheta')(a) \cap \rho(Y)), J' = (\Phi \cap \rho(Y))$ $(J \neq \emptyset) \Rightarrow \begin{cases} \eta_1(\vartheta')(a) \cap \rho(Y) \subseteq \rho(y) \\ \eta_2(\vartheta')(a) \ni \vartheta \end{cases}$ $(a \in \Phi \wedge J' \neq \emptyset) \Rightarrow \begin{cases} (\Phi \cap \rho(Y)) \subseteq \rho(y) \\ \eta_2(\vartheta_E)(a) \ni \vartheta \end{cases}$
$(\rho, \eta, \Phi) \models^\vartheta P_0 + P_1$	iff $(\rho, \eta, \Phi) \models^{\vartheta \pm 0} P_0 \wedge (\rho, \eta, \Phi) \models^{\vartheta \pm 1} P_1$
$(\rho, \eta, \Phi) \models^\vartheta P_0   P_1$	iff $(\rho, \eta, \Phi) \models^{\vartheta \parallel 0} P_0 \wedge (\rho, \eta, \Phi) \models^{\vartheta \parallel 1} P_1$
$(\rho, \eta, \Phi) \models^\vartheta (\nu a)P$	iff $(\rho, \eta, \Phi) \models^\vartheta P$
$(\rho, \eta, \Phi) \models^\vartheta !P$	iff $(\rho, \eta, \Phi) \models^{\vartheta \sharp 0} P \wedge (\rho, \eta, \Phi) \models^{\vartheta \sharp 1} P$

**Table 2.** Control Flow Analysis for the  $\pi$ -calculus.

can be bound to and, in our framework, giving information on where and between which the communications take place.

From a static point of view, this is not trivial a task. The analysis has a particular process  $P$  as input and should predict something on all the possible continuations of  $P$ , i.e. all the possible processes  $Q$  to which a sequence of computations can lead. As we have shown in the previous sections, the structure of a process – in terms of its addresses – may change each time a transition is fired. Mainly, the structure changes due to transitions of replicated sub-processes. Indeed, if the process under analysis includes some replications, then its behaviour is potentially infinite. We would like to have instead a finite analysis, able to safely approximate this infinite behaviour, without paying so high a price. For this reason, we first define a *finite* approximation, that we will call *pre-estimate* and on that basis, we will define the real estimate or *solution*, as a result of our analysis.

More precisely, pre-estimates are computed considering that each replication  $!P$  originates exactly two copies of  $P$  in parallel. This means that our analysis uses a finite number of addresses, only. The actual estimate, possibly involving an unbound number of addresses, is then obtained by a suitable closure operation on a pre-estimate. Luckily enough, for our aim, generating and investigating pre-estimates will be sufficient. So our proposal is feasible (and efficient).

We then begin with the description of pre-estimates and then we proceed with solutions. In details, the pre-estimate components are:

- $\rho : Var \rightarrow \wp(Val)$  is the *abstract environment* that associates a set of values with values; more precisely,  $\rho(x)$  must include the set of values that  $x$  could assume at run-time. We shall allow to regard the abstract environment as a function such that  $\forall a \in Val : \rho(a) = \{a\}$ . Also, we will use  $\rho(Y)$  as a shorthand for  $\bigcup\{\rho(y_i) \mid y_i \in Y\}$ .
- $\eta = \langle \eta_1, \eta_2 \rangle$  is the *abstract communication structure*:
  - $\eta_1 : \mathcal{A} \rightarrow (Val \rightarrow \wp(Val))$  associates values, actually channels, with the values that can be sent over them at certain positions; more precisely,  $\eta_1(\vartheta)(a)$  must include the set of values that can be sent over the channel  $a$  by the sub-process  $P@{\vartheta}$ .
  - $\eta_2 : \mathcal{A} \rightarrow (Val \rightarrow \wp(\mathcal{A}))$  associates channels with the addresses of sub-processes that can receive and send values on them; more precisely,  $\eta_2(\vartheta)(a)$  must include the set of addresses  $\vartheta'$  such that  $P@{\vartheta'}$  can receive values on  $a$ , sent by  $P@{\vartheta}$ . For instance, if  $\vartheta \in \eta_2(\vartheta')(a)$ , then an internal action on channel  $a$  having  $P@{\vartheta'}$  as a sender and  $P@{\vartheta}$  as a receiver is possible. (Note that  $\vartheta$  and  $\vartheta'$  are compatible.) We also predict communications with the attacker and designate  $\vartheta_E$  to be the generic address of an unknown process hosted in  $E$ , by definition compatible with all the addresses in  $P$ .
- $\Phi \subseteq Val$  represents the *external environment* (see also [5]). This component establishes a super-set of the values the environment can send and receive on the channels it knows. According to the Dolev and Yao model [15, 2], (i)  $E$  initially knows a subset of the values of  $P$ ; (ii) afterwards,  $E$  may increase its knowledge by communicating with  $P$  and (iii)  $E$  can use all the known values to communicate with  $P$ , possibly affecting its behaviour.

Once defined the form of analysis pre-estimates, a Flow Logic for when pre-estimates are acceptable consists in defining a number of clauses. These clauses operate upon judgments of the form:  $(\rho, \eta, \Phi) \models^{\vartheta} P$ . A pre-estimate for a process  $P$  is then a triple  $(\rho, \eta, \Phi)$  s.t.  $(\rho, \eta, \Phi) \models^{\epsilon} P$ ,

where  $fn(P) \subseteq \Phi$  and  $fv(P) = \emptyset$ . Our Control Flow Analysis is defined by the Flow Logic clauses in Tab. 2.

To understand how processes are validated, consider the most crucial rules, those for parallel composition, for output, for input and for replication. All the rules for a compound process require that the components are validated, and so we feel free to omit these conjuncts below. Note that in the clauses, addresses are only used and updated when the top-level operator is a  $|$ , a  $+$ , or a  $!$ , i.e. in the parallel composition, in the summation or replication rules.

Consider the parallel composition rule as an example. A pre-estimate is valid for the sub-process  $(P_0|P_1)$  with address  $\vartheta$  with respect to the initial process  $P$  under analysis, if it is valid for each  $P_i$  ( $i = 0, 1$ ) with address  $\vartheta||_i$ . Indeed, if  $(P_0|P_1) = P@ \vartheta$  then, by definition,  $P_i = (P_0|P_1)@ \vartheta||_i$ . Furthermore, note that  $(\rho, \eta, \Phi) \models^\vartheta \pi.(P_0|P_1)$  implies that  $(\rho, \eta, \Phi) \models^\vartheta (P_0|P_1)$  and therefore  $(\rho, \eta, \Phi) \models^{\vartheta||_0} P_0$  and  $(\rho, \eta, \Phi) \models^{\vartheta||_1} P_1$ . The same considerations are valid for  $P_0 + P_1$ .

Consider now the output case. To validate the process  $\bar{x}y.P$  the rule requires that

$\dots \forall a \in \rho(x) :$  for each value  $a$  that can be bound to  $x$   
 $\rho(y) \subseteq \eta_1(\vartheta)(a)$  the set of values  $\eta_1(\vartheta)(a)$  that can be sent at  $\vartheta$  along each  $a$ ,  
 $= J$  must include the values to which  $y$  can evaluate, i.e.  $\rho(y)$ .  
 $\wedge (a \in \Phi$  if  $a$  belongs to the environment knowledge and  
 $\wedge J \neq \emptyset)$  there is a possible flow of values on  $a$ , i.e.  $\eta_1(\vartheta)(a) = J \neq \emptyset$ , then  
 $\Rightarrow (\eta_1(\vartheta)(a) \subseteq \Phi$   $E$  knows the values that flow on  $a$ , i.e. they are in  $\Phi$ , and therefore,  
 $\wedge \eta_2(\vartheta)(a) \ni \vartheta_E)$  a communication is possible between the sub-process at  $\vartheta$  and  $E$

The more demanding rule is the rule for input. For validating the process  $x(y \in Y).P$ , we require that

$\dots \forall a \in \rho(x),$  for each value  $a$  that can be bound to  $x$  and  
 $\forall \vartheta' : comp(\vartheta, \vartheta') :$  for each compatible address  $\vartheta'$ ,  
 $J = (\eta_1(\vartheta')(a) \cap \rho(Y))$   
 $J' = (\Phi \cap \rho(Y))$  some of the values that can be sent on  $a$   
 $(J \neq \emptyset) \Rightarrow$   
 $\left\{ \begin{array}{l} \eta_1(\vartheta')(a) \cap \rho(Y) \subseteq \rho(y) \\ \eta_2(\vartheta')(a) \ni \vartheta \end{array} \right.$  at  $\vartheta'$ , i.e.  $\eta_1(\vartheta')(a)$ , are included in the set of values,  
to which  $y \in Y$  can evaluate, provided that they  
belong also to  $\rho(Y)$ , i.e.  $J \neq \emptyset$ . In this case,  
a communication is possible between  
the sub-processes at  $\vartheta'$  and at  $\vartheta$

$(a \in \Phi \wedge J' \neq \emptyset) \Rightarrow$  Also, if  $a$  belongs to the environment knowledge, some of  
the values (in  $\Phi$ ) that can be sent on  $a$  by  $E$ , are  
included in the set of values to which  $y \in Y$   
can evaluate, provided that they belong also to  $\rho(Y)$ ,  
i.e.  $J' \neq \emptyset$ , and a communication is possible  
between the sub-process at  $\vartheta'$  and the one in  $E$

$\left\{ \begin{array}{l} (\Phi \cap \rho(Y)) \subseteq \rho(y) \\ \eta_2(\vartheta_E)(a) \ni \vartheta \end{array} \right.$

Note that whenever we test for emptiness, we address the reachability issue; indeed the condition  $\eta_1(\vartheta)(a) = J = \emptyset$  in the rule for output amounts to saying that an output action in that position, on channel  $a$ , is not reachable (e.g. the output action  $\bar{a}x$  in the process  $(\nu b)b(x \in \{c\}).\bar{a}x$ ). Similarly for input conditions: if  $J = (\eta_1(\vartheta')(b) \cap \rho(Y)) = \emptyset$  and  $J' = \Phi \cap \rho(Y) = \emptyset$  then the input action on  $b$  is not reachable (e.g. the action  $b(x \in \{c\})$  in the example above).

Finally, the rule for replication asks for the validation of the replicated process at the addresses  $\vartheta!_0$  and  $\vartheta!_1$ . The form of this rule corresponds to the idea that the infinite behaviour of  $!P$  can be approximated by the finite one of a pair of its copies  $P$ , i.e.  $!P@!_0$  and  $!P@!_1$ . They can:

- communicate each other (the two addresses are compatible);
- separately communicate to the external environment, with processes in parallel with  $!P$ ; and
- make an internal communication, as any other process.

*Example 3.* To give the flavour of the analysis, we use our running example.

$$Sys = !P \mid Q = \underbrace{!(\bar{a}c.b(z \in \{a, c\}))}_{\|_0!_i \pm 0} + \underbrace{\bar{a}e}_{\|_0!_i \pm 1} \mid \underbrace{a(x \in \{c\})}_{\|_1} . \underbrace{(\bar{b}(x).\mathbf{0})}_{\|_1\|_0} \mid \underbrace{R}_{\|_1\|_1}$$

The analysis should provide us with the following information:

- which are the values that can be bound to each variable. In terms of the CFA, we need to know  $\rho(z)$  and  $\rho(x)$ ;
- the set of values that can be sent over the channels  $a$  and  $b$ , i.e.  $\eta_1(\vartheta)(a)$  and  $\eta_1(\vartheta)(b)$  for  $\vartheta \in \{\|_0!_i \pm 0, \|_0!_i \pm 1, \|_1, \|_1\|_0\}$ ;
- the set of addresses of sub-processes that can send and receive values on the channels, i.e.  $\eta_2(\vartheta)(a)$  and  $\eta_2(\vartheta)(b)$  for  $\vartheta$  ranging over  $\{\|_0!_i \pm 0, \|_0!_i \pm 1, \|_1, \|_1\|_0\}$ ;
- finally, the contribute  $\Phi$  of the environment. Initially, the environment knows all the free values  $\{a, b, c, e\}$  of the process  $Sys$ .

To establish  $(\rho, \eta, \Phi) \models^\epsilon Sys$  it is necessary to establish  $(\rho, \eta, \Phi) \models^{\|_0} !P = Sys@!_0$  and  $(\rho, \eta, \Phi) \models^{\|_1} Q = Sys@!_1$ ; to establish  $(\rho, \eta, \Phi) \models^{\|_0!_i} P$  for each  $i \in \{0, 1\}$ , it is necessary to establish  $(\rho, \eta, \Phi) \models^{\|_0!_i \pm 0} \bar{a}c.b(z \in \{a, c\})$  and  $(\rho, \eta, \Phi) \models^{\|_0!_i \pm 1} \bar{a}e$ . We can compute a pre-estimate giving the following results:

- $\rho(z) \supseteq \{c, \mathbf{a}\}$  and  $\rho(x) \supseteq \{c\}$
- $\eta_1(\|_0!_i \pm 0)(a) = \{c\}$ ,  $\eta_1(\|_0!_i \pm 1)(a) = \{e\}$ ,  $\eta_1(\|_1)(a) = \emptyset$ ,  $\eta_1(\|_0!_i \pm 0)(b) = \emptyset$  and  $\eta_1(\|_1\|_0)(b) \supseteq \{c\}$

- $\eta_2(\|_0!_i\pm_0)(a) \supseteq \{\vartheta_E, \|_1\}$ ,  $\eta_2(\|_0!_i\pm_1)(a) \not\supseteq \{\|_1\}$  and  $\eta_2(\|_1\|_0)(b) \supseteq \{\vartheta_E, \|_0!_i\pm_0\}$
- $\Phi \supseteq \{a, b, c, e\}$

We write in boldface the value  $\mathbf{a}$  in  $\rho(z)$  to stress the contribute of the environment. In fact  $\eta_1(\|_1\|_0)(b) \cap \{c, a\} \supseteq \{c\}$  and  $\Phi \cap \{c, a\} = \{c, a\}$ . As a matter of fact, if the process were in isolation (i.e. initially with no free names, e.g. if it were on the form  $(\nu a)(\nu b)Sys$ ), the only value that could be bound to  $z$  would have been  $c$ , i.e. the value passed from the only compatible action  $\bar{b}x$ . Furthermore,  $\rho(x)$  includes  $\{c\}$ , that coincides with  $(\eta_1(\|_0!_i\pm_0)(a) \cap \{c\})$ . Instead,  $\rho(x)$  does not include  $\{e\}$  because  $\eta_1(\|_0!_i\pm_1)(a) = \{e\} \cap \{c\} = \emptyset$ .

It is easy to show that validating a process  $P$ , corresponds to validating all its sub-processes  $P@v'$ . There is a subtlety here. Actually, validating a process  $P$  on the form  $\pi.P'$  amounts to validating all the sub-processes of  $P'$  as well.

**Lemma 2.** *For each process  $P$ ,  $(\rho, \eta, \Phi) \models^v P$  iff  $\forall v' \in Addr(P) : (\rho, \eta, \Phi) \models^{v'} P@v'$ . Also,  $(\rho, \eta, \Phi) \models^{v'} P'@v'$  if  $P = \pi.P'$ .*

## 4.2 Validation of Solutions

To take into account all the possible expansion of processes due to the application of rule *Bang*, we introduce estimates on the form  $(\rho, \widehat{\eta}, \Phi)$ , built from  $(\rho, \eta, \Phi)$ . The only difference between them is that the component  $\widehat{\eta}$  handles an unbound number of addresses. Instead, the other components  $\rho$  and  $\Phi$  do not change at all with respect to the pre-estimate<sup>4</sup>.

Estimates only play a technical role. Indeed, for our purposes, given a process  $P$  and its sub-processes, we need to know where and between which sub-processes communications may occur. In particular, we are interested in the inputs/outputs possible at a given address  $v$ , such that  $P@v$  is on the form  $!S$ . As a matter of fact, all the processes arising from replicating  $S$  exhibit the same static behaviour of  $P@v$ , and therefore pre-estimates are enough. To consider all the possible unfoldings due to replications (and therefore the unbound number of addresses of the expanded process) we use a *closure* function that returns the set of addresses generated by expanding all the occurrence of  $!_i$  along the path  $v$ . Its definition is reminiscent of that of  $w(\cdot)$ .

<sup>4</sup> Actually,  $\rho$  as well could be bound to addresses and made more precise. Nevertheless, this would add irrelevant information with respect to the aspects of process behaviour that we are interested in. Additionally, such an extension will require a much more complex treatment (see e.g. [16]).

**Definition 8.** Given a process  $P$ , let  $\widehat{w}(\cdot)$  the function closure  $\widehat{w}: \mathcal{A} \rightarrow \wp(\mathcal{A})$  such that  $\widehat{w}(\vartheta) \ni \vartheta$  and:

- $\widehat{w}(\epsilon) = \epsilon$ ;
- $\widehat{w}(\diamond_i \vartheta) = \diamond_i \widehat{w}(\vartheta)$ ;
- $\widehat{w}(!_0 \vartheta) = \widehat{w}(!_1 \vartheta) = \{\|_0\|_0, \|_0\|_1\} \widehat{w}(\vartheta) \cup \|_1 \widehat{w}(!_0 \vartheta) \cup \|_1 \widehat{w}(!_1 \vartheta)$ .

Note that  $w(\vartheta) \in \widehat{w}(\vartheta)$  and that, in particular, for  $i \in \{0, 1\}$ ,  $\widehat{w}(!_i) \supseteq \{\|_1^q \|_0 \|_j \mid j \in \{0, 1\} \wedge n \in \mathbb{N}\}$ .

We define estimates as follows; note that now the functions in the second component may have an infinite domain.

**Definition 9.** Given a process  $P$  and a pre-estimate  $(\rho, \eta, \Phi)$  for it, the corresponding solution is  $(\rho, \widehat{\eta}, \Phi)$ , where  $\widehat{\eta}$  is such that

- $\forall a, b, \vartheta : b \in \eta_1(\vartheta)(a)$  iff  $\forall \vartheta' \in \widehat{w}(\vartheta) : b \in \widehat{\eta}_1(\vartheta')(a)$ ;
- $\forall \vartheta, \vartheta', a : \vartheta' \in \eta_2(\vartheta)(a)$  iff  $\forall \vartheta_0 \in \widehat{w}(\vartheta'), \forall \vartheta_1 \in \widehat{w}(\vartheta) : \vartheta_0 \in \widehat{\eta}_2(\vartheta_1)(a)$ .

Consequently, each time a pre-estimate validates the process  $P$  at address  $\vartheta$ , then the corresponding estimate validates the same process at each address  $\vartheta' \in \widehat{w}(\vartheta)$ . In particular, from  $\vartheta_0 !_0 \vartheta \in \eta_2(\vartheta_0 !_1 \vartheta')$ , we have that  $\forall l \neq m : \vartheta_0 \|_1^l \|_0 \|_0 \vartheta \in \eta_2(\vartheta_0 \|_1^m \|_0 \|_1 \vartheta')$  and  $comp(\vartheta_0 \|_1^l \|_0 \|_0 \vartheta, \vartheta_0 \|_1^m \|_0 \|_1 \vartheta')$ .

As mentioned above, pre-estimates are enough for approximating the usage of a specific channel for input or output by a given sub-process. In particular, if a certain communication is not predicted by a pre-estimate, then also the estimate does not predict it. Indeed, the following follows by construction.

**Proposition 1.** Given a process  $P$ , let  $(\rho, \eta, \Phi)$  is a pre-estimate of  $P$  and  $(\rho, \widehat{\eta}, \Phi)$  is the corresponding estimate. Then, for all  $a, b$  and for all  $\vartheta, \vartheta'$ , we have:

- if  $b \notin \eta_1(\vartheta)(a)$  then  $\forall \vartheta' \in \widehat{w}(\vartheta) : b \notin \widehat{\eta}_1(\vartheta')(a)$
- if  $\vartheta' \notin \eta_2(\vartheta)(a)$  then  $\forall \vartheta_0 \in \widehat{w}(\vartheta'), \forall \vartheta_1 \in \widehat{w}(\vartheta) : \vartheta_0 \notin \widehat{\eta}_2(\vartheta_1)$ .

The following lemma shows that our solution is able to validate any possible process resulting from one or more expansions, due to replications; it will be help in proving the subject reduction theorem.

**Lemma 3.** If  $(\rho, \eta, \Phi) \models^\vartheta P$  then  $\forall \vartheta_0, \vartheta_1$  such that  $\exists H = h(k, P, \vartheta_0, \vartheta_1)$ ,

- (a)  $(\rho, \eta, \Phi) \models^\vartheta P$  implies  $(\rho, \widehat{\eta}, \Phi) \models^\vartheta P$ ;
- (b)  $(\rho, \widehat{\eta}, \Phi) \models^\vartheta P$  iff  $(\rho, \eta, \Phi) \models^\vartheta H$ .

The above lemma and Corollary 1 furnish the basis to prove the Subject Reduction Theorem 1.

### 4.3 Correctness

To establish the semantic correctness of our analysis we rely on the definition of the early semantics in Tab. 1 as well as on that of estimate, based on the analysis in Tab. 2.

**Theorem 1 (Subject reduction).** *Given a process  $P$ , if  $(\rho, \widehat{\eta}, \Phi) \models^\vartheta P$ , then*

- (a) *if  $P \equiv Q$ , then  $(\rho, \widehat{\eta}, \Phi) \models^\vartheta P$  iff  $(\rho, \widehat{\eta}, \Phi) \models^\vartheta Q$ ;*
- (b) *if  $P \xrightarrow{\mu} Q$  then we have:*
  - (1) *if  $\mu = \tau$  then  $(\rho, \widehat{\eta}, \Phi) \models^\vartheta Q$ ;*
  - (2) *if  $\mu = \bar{a}b$  or  $\mu = \bar{a}(b)$ , then  $(\rho, \widehat{\eta}, \Phi) \models^\vartheta Q$  and the transition involves  $P @ \vartheta_0 \xrightarrow{\mu} R_0$ , then  $b \in \widehat{\eta}_1(\vartheta\vartheta_0)(a)$ ; additionally,  $b \in \Phi \wedge \vartheta_E \in \widehat{\eta}_2(\vartheta\vartheta_0)(a)$ , provided that  $a \in \Phi$ ;*
  - (3) *if  $\mu = ab$ , the transition involves  $P @ \vartheta_1 = a(y \in Y).P_1 \xrightarrow{\mu} R_1$  and the condition  $(*)$  holds, then  $b \in \rho(Y)$  and  $(\rho, \widehat{\eta}, \Phi) \models^\vartheta Q$ , where  $(*)$  is  $(\exists \vartheta_0 : b \in \widehat{\eta}_1(\vartheta\vartheta_0)(a) \text{ and } \vartheta\vartheta_1 \in \widehat{\eta}_2(\vartheta\vartheta_0)(a))$  or (if  $a \in \Phi$  then  $b \in \Phi$  and  $\vartheta\vartheta_1 \in \widehat{\eta}_2(\vartheta_E)(a)$ ).*

### 4.4 Existence

So far we gave a procedure for validating whether or not a proposed pre-estimate  $(\rho, \eta, \Phi)$  is in fact acceptable. Remarkably, there always exists a least choice of  $(\rho, \eta, \Phi)$  that is acceptable in the manner of Tab. 2.

It is quite standard to partially order the set of proposed estimates. Recall that a *Moore family*  $\mathcal{I}$  is a set that contains  $\sqcap \mathcal{J}$  for all  $\mathcal{J} \subseteq \mathcal{I}$  (where  $\sqcap$  is the greatest lower bound operator), defined pointwise. One important property of a Moore family is that it always contains a least element.

The following theorem then guarantees that there is always a least pre-estimate to the specification in Tab. 2. Its proof is by induction and it is quite similar to the corresponding one in [8], so we omit it. The subject reduction result above applies to *all* the pre-estimates of the analysis, and hence in particular to the least.

**Theorem 2.**  $\forall P : \{(\rho, \eta, \Phi) \mid (\rho, \eta, \Phi) \models^\epsilon P\}$  *is a Moore family.*

## 4.5 Construction

There is also a constructive procedure for obtaining the least pre-solution. Actually, this construction is very similar to the one in [8], so we dispense here with the details. We argue that the complexity of obtaining a pre-estimate can be kept low-polynomial, following the technique presented in [25]. The main steps of our procedure are the following.

First, we attach to each prefix of the given process the address at which it occurs. E.g., consider the process of our running example:

$$Sys = ! \underbrace{(\bar{a}c.b(z \in \{a, c\}))}_{\|_0\|_i \pm_0} + \underbrace{(\bar{a}e)}_{\|_0\|_i \pm_1} \mid \underbrace{a(x \in \{c\})}_{\|_1} . \underbrace{(\bar{b}\langle x \rangle.\mathbf{0})}_{\|_1\|_0} \mid \underbrace{R}_{\|_1\|_1}$$

We pass from  $Sys @ \|_0\|_i \pm_1$  to  $[\|_0\|_0 \pm_0] \bar{a}c. [\|_0\|_0 \pm_0] b(z \in \{a, c\})$ . This initialization step is linear with the number of prefixes of the process, that usually are much more than the number of operators  $\mid$  and  $+$ . So, we assume that there are  $O(N)$  addresses, where  $N$  depends linearly on the symbols in  $P$ .

A second pre-processing step is determining the compatibility between the addresses of the process at hand. Here, one has to slightly extend the notion of address to take care also of the following case  $\mu.(P \mid Q) \mid R$ , the addresses of which include, besides  $\|_0$  and  $\|_1$  also  $\|_0\|_0$  and  $\|_0\|_1$ . More precisely, the addresses of  $\mu.P$  include also those of  $P$ . Note that the addresses can be collected while performing the first step above: so one may naively fill in a compatibility table in  $O(N^3)$  time and making checking compatibility cheap.

The actual generation of the pre-estimate is based on the observation that validating a solution  $(\rho, \eta, \Phi) \models^{\vartheta} P$  amounts to checking a number of individual constraints. Much in the style of [8] we then define a function  $\mathcal{G}_C[[T]]$ , where  $T$  is obtained from  $P$  in the first step above. This function explicitly extracts the set of constraints to be checked by inducing on the syntax of  $T$  itself. The algorithm in [8] works in  $O(N^5)$  time; a further improvement [25] reduced the exponent to 3. The basic difference of our construction with the algorithm of [8] concerns the treatment of inputs and outputs, as these have now attached addresses, and the constraint generator uses them according to the analysis in Tab. 2. There, the messages that flow on a given channel  $a$  were collected all together, regardless of the position where  $a$  occurs. Our present analysis is more detailed, and thus our construction generates more constraints. However, the overall complexity is still polynomial, (actually, our tuples  $(\rho, \eta, \Phi)$  contain  $O(N^2)$  items,  $O(N)$  for each element of  $P$  times  $O(N)$  for each

address, while the solutions of [8] have only  $O(N)$  items, so the exponent of the polynomial is increased by 3).

## 5 CFA & Security Properties

Our analysis statically approximates the behaviour of a process  $P$  under consideration, in particular it tracks where and between which communications may occur. This is essentially done through its component  $\eta_2$ . Actually, exploiting the soundness of our analysis, we can prove, among others, the following three basic facts. Suitable combinations of them enable us to define and check many security properties. In particular, those centered on access control, which prescribe which processes can communicate each other and in which direction.

- **[no output for  $P@v'$  on  $a$ ]** The process  $P@v'$  cannot use the channel  $a$  to send data to the environment, when  $v_E \notin \eta_2(v')(a)$ .
- **[no input for  $P@v$  on  $a$ ]** The process  $P@v$  cannot use the channel  $a$  to receive data from the environment, when  $v \notin \eta_2(v_E)(a)$ .
- **[no communication between  $P@v$  and  $P@v'$  on  $a$ ]** The process  $P@v'$  cannot communicate with  $P@v$  on channel  $a$ , when  $v \notin \eta_2(v')(a)$ , i.e. no exchange of information occurs on  $a$ , from site  $v'$  to site  $v$  within the system.

Formally, we have the following.

**Theorem 3.** *Let  $P$  be a process with pre-estimate  $(\rho, \eta, \Phi)$  and let  $a$  be a channel.*

1. **[no output for  $P@v'$  on  $a$ ]** *If  $a \in \text{fn}(P)$  and  $v_E \notin \eta_2(v')(a)$ , then whenever  $P \longrightarrow^* P' \xrightarrow{\mu} P''$ , with  $\mu \in \{\bar{a}b, \bar{a}(b)\}$ , and the last transition involves  $P'@v_0$ , then  $v' \neq v_0$ .*
2. **[no input for  $P@v$  on  $a$ ]** *If  $a \in \text{fn}(P)$  and  $v \notin \eta_2(v_E)(a)$ , then whenever  $P \longrightarrow^* P' \xrightarrow{ab} P''$ , and the last transition involves  $P'@v_1$  then  $v \neq v_1$ .*
3. **[no communication between  $P@v$  and  $P@v'$  on  $a$ ]** *If  $v \notin \eta_2(v')(a)$ , (with  $v, v' \neq v_E$ ), then, whenever  $P \longrightarrow^* P' \xrightarrow{\tau} P''$  and the last transition involves  $P'@v_0 = \bar{a}b.P_0$  and  $P'@v_1 = a(y \in Y).P_1$ , then  $v' \neq v_0$  or  $v \neq v_1$ .*

The check of absence of a specific output can be made more precise, by exploiting the sub-component  $\eta_1$  of a pre-estimate that permits to establish suitable tests for when a sub-process  $P@v'$  cannot send a selected

value  $b$  to the environment. If  $b$  is considered a sensible data, its *secrecy* with respect to a channel and a position is indeed guaranteed. Analogously, one can check that a inputted value  $b$  comes only from inside the system, i.e. that it is never received from the attacker. The following corollary immediately follows from Theorem 3.

**Corollary 2.** *Let  $P$  be a process with pre-estimate  $(\rho, \eta, \Phi)$  and let  $a$  be a channel.*

1. [ $P@v'$  **cannot output**  $b$ ] *If  $b \notin \eta_1(v')(a)$ , and whenever  $P \longrightarrow^* P' \xrightarrow{\mu} P''$ , with  $\mu \in \{\bar{a}b, \bar{a}(b)\}$ , and the last transition involves  $P'@v_0$  then  $v' \neq v_0$ .*
2. [ $P@v$  **cannot input**  $b$  on  $y$ ] *If  $b \notin \rho(y)$  with  $y \in bv(P@v)$  and whenever  $P \longrightarrow^* P' \xrightarrow{ab} P''$  and the last transition involves  $P'@v_0$  then  $v \neq v_0$ .*

## 5.1 Secrecy

We can exploit the above corollary to statically check secrecy. First we give the dynamic notion of secrecy: a process preserves the secrecy of a value  $b$ , when it never sends it on a free channel. Then, no attacker harboured in the environment can read  $b$ . We start with defining the dynamic notion. We then state a theorem showing that the component  $\Phi$  of a pre-estimate suffices for ensuring statically the dynamic property.

**Definition 10.** *Let  $P$  be a process and let  $a$  be a channel.  $P$  preserves the secrecy of the value  $b$  (i.e. does not disclose  $b$  to the external environment), if whenever  $P \longrightarrow^* P' \xrightarrow{\mu} P''$ , then  $\mu \notin \{\bar{a}b, \bar{a}(b)\}$ .*

**Theorem 4.** *Let  $P$  be a process with pre-estimate  $(\rho, \eta, \Phi)$ . If  $b \notin \Phi$ , then  $P$  preserves the secrecy of the value  $b$ .*

We now show that this property is preserved by any context satisfying a mild constraint: the context contains only values known to everyone, including the attacker. Actually this is not a limitation. Indeed, if a secret has to be sent on a public channel, it has to be protected otherwise, typically by encrypting it and making the resulting message public again. We have no encryption here: we refer the reader to [6] for a CFA taking care of that.

**Theorem 5.** *Let  $P$  be a process with pre-estimate  $(\rho, \eta, \Phi)$  and  $\mathcal{C}[-]$  be a one hole context with  $\text{vals}(\mathcal{C}[-]) \subseteq \Phi$ . If  $P$  preserves the secrecy of the value  $b$  then  $\mathcal{C}[P]$  still preserves the secrecy of the value  $b$ .*

## 5.2 Security policies

A *security policy* is used to state which information flows are to be allowed and which are to be prevented. Therefore, it establishes which privileges are accorded to which principals, in terms of which actions they can or cannot perform. This section discusses how policies can be characterized abstractly, and how the approach described so far helps in detecting whether a process violates or not a given policy. The check of absence of communications between a pair of sub-processes provides us with the basis for statically verifying whether a process follows a specific security policy. Indeed, our analysis approximates the usage of channels, essentially through its components  $\eta_1$  and  $\eta_2$ , which track where the read/write operations may occur.

In what follows, we shall consider some different dynamic security properties, showing that they all are captured by simple checks on pre-estimates. So, it suffices computing a pre-estimate  $\mathcal{R}$  for a process  $P$  *once and for all* and then check the properties of interest on  $\mathcal{R}$ . Our presentation below use the following paradigm, whose steps consist of

1. a(n abstract) policy  $\Sigma$ , as a set of capabilities that specify which are the authorized accesses to data;
2. the notion of when a process dynamically respects  $\Sigma$ ;
3. a static test on an estimate  $\mathcal{R}$  of  $P$  for when  $\mathcal{R}$  agrees with  $\Sigma$ ;
4. a theorem stating that if  $\mathcal{R}$  agrees with  $\Sigma$ , then  $P$  respects  $\Sigma$ ;
5. one or more instantiations of  $\Sigma$  mirroring some security properties discussed in the literature.

**Definition 11.** *Given a process  $P$ , two compatible addresses  $\vartheta, \vartheta'$ , and a channel  $a$ , a localized policy is a relation  $\Sigma$  such that  $(\vartheta, \vartheta') \in \Sigma(a)$ , whenever  $\text{comp}(\vartheta, \vartheta')$*

- $P@ \vartheta'$  has the capability to send on the channel  $a$  to  $P@ \vartheta$ ; and
- $P@ \vartheta$  has the capability to receive on the channel  $a$  from  $P@ \vartheta'$ .

For the sake of simplicity, hereafter, we consider pairs of addresses  $(\vartheta, \vartheta')$  with  $\vartheta, \vartheta' \neq \vartheta_E$ . By extension, we can also consider the capability to write (read, resp.) on the channel  $a$  having the environment as a receiver (sender, resp.).

Our next step makes it precise when a process respects the given policy.

**Definition 12.** *A given process  $P$  respects the security policy  $\Sigma$ , if  $\forall a: (\vartheta, \vartheta') \notin \Sigma(a)$  implies that, whenever  $P \longrightarrow^* P' \xrightarrow{\tau} P''$  and the last*

transition involves  $P'@v_0 = \bar{a}b.\tilde{P}'$  and  $P'@v_1 = a(y \in Y).\tilde{P}'$ , then  $v' \neq v_0$  or  $v \neq v_1$ .

The following definition specifies our static test.

**Definition 13.** A pre-estimate  $(\rho, \eta, \Phi)$  of a process  $P$  agrees with  $\Sigma$ , if  $(v, v') \notin \Sigma(a)$  implies  $v \notin \eta_2(v')(a)$ .

A pre-estimate that agrees with a policy correctly predicts the usage of channels. The result follows immediately, from Theorem 3.

**Corollary 3.** A process  $P$  with pre-estimate  $(\rho, \eta, \Phi)$  respects  $\Sigma$ , if  $(\rho, \eta, \Phi)$  agrees with  $\Sigma$ .

We now instantiate the abstract policy  $\Sigma$  to some policies, based on security levels. This requires to define a hierarchy of levels for processes. In the following, we assume to have a set of security levels  $SL$  (ranged over by  $l$ ) made into a lattice  $\langle SL, \leq \rangle$  by the partial order relation  $\leq$ . Then, we assign a security level to addresses via the function  $F$ , in such a way that  $F(v) = l$  means that the process  $P@v$  has clearance level  $l$ .

*No Read-Up/No Write-Down* As a first example of these policies, consider the following variant of the *no read-up/no write-down* property by Bell and LaPadula [3] (see also [17]). Processes receive clearance levels of security and a process with a high level cannot write any value to a process at low level, while the converse is allowed; symmetrically a process at low level cannot read data from one at a high level. In other words, the dynamic property requires that there is no communication between the low-level process  $P@v$  and the high-level process  $P@v'$ . More precisely, it requires that  $P@v'$  does not perform an output, i.e. a write down to  $P@v$  (see the simple example below). The corresponding static test is straightforward:  $v \notin \eta_2(v')$ .

**Definition 14.** A localized policy  $\Sigma$  is a no read-up/no write-down policy, if  $\forall (v, v') \in \Sigma(a)$  iff  $F(v) \geq F(v')$ .

**Corollary 4.** Given  $F$ , a process  $P$  is no read-up/no write-down, if agrees with a no read-up/no write-down policy  $\Sigma$ .

*No Read-Down/No Write-Up* Consider now the dual policy for integrity, based on the Biba model [4]. It requires that a sub-process  $P@v$  with a low clearance level cannot write any value to a sub-process  $P@v'$  at a higher level, while the converse is allowed. Again the point is that there should not be any communication between  $P@v$  and  $P@v'$  in the wrong sense. The static test also turns out to be easy:  $v' \notin \eta_2(v)$ .

**Definition 15.** A localized policy  $\Sigma$  is a no read-down/no write-up policy, if  $\forall(\vartheta, \vartheta') \in \Sigma(a)$  iff  $F(\vartheta) \leq F(\vartheta')$ .

**Corollary 5.** Given  $F$ , a process  $P$  is no read-down/no write-up, if agrees with the no read-down/no write-up policy  $\Sigma$ .

*Polarity* Some simpler policies focus on polarity (see also [26, 19]). These policies prescribe for each channel if a particular process can use it either for input or only for output or in both ways. Checking that a process obeys a given polarity policy is again straightforward, e.g.  $P@a$  uses  $a$  only for output, if for all compatible  $\vartheta'$ ,  $(\vartheta, \vartheta') \notin \Sigma(a)$ . This property is guaranteed by the following simple check:  $\vartheta \notin \eta_1(\vartheta')(a)$ .

First, we define the actual dynamic policy as an instance of that in Def. 11; then we state a corollary of Corollary 3.

**Definition 16.** A localized policy  $\Sigma$  for a process  $P$  is a polarity policy if, on each channel  $a$ , it assigns to  $P$  the following capabilities:

1. read-only if and only if:  $\forall \vartheta' \in \text{Addr}(P) : (\vartheta, \vartheta') \notin \Sigma(a)$  and  $(\vartheta, \vartheta_E) \notin \Sigma(a)$
2. write-only if and only if:  $\forall \vartheta' \in \text{Addr}(P) : (\vartheta, \vartheta') \notin \Sigma(a)$  and  $(\vartheta_E, \vartheta) \notin \Sigma(a)$
3. no-read&no-write if and only if:  $\forall \vartheta' \in \text{Addr}(P) : (\vartheta, \vartheta'), (\vartheta', \vartheta) \notin \Sigma(a)$  and  $(\vartheta, \vartheta_E) \notin \Sigma(a)$  and  $(\vartheta_E, \vartheta) \notin \Sigma(a)$ .
4. read&write if anyone of the above cases is verified.

Note that if a value is used neither in input, nor in output, then it is a passive object, i.e. it is not used as a channel.

Checking polarities is now straightforward: it suffices to control if an estimate agrees with the policy given.

**Corollary 6.** A process  $P$  with estimate  $(\rho, \eta, \Phi)$ , respects a polarity policy  $\Sigma$ , if  $(\rho, \eta, \Phi)$  agrees with  $\Sigma$ .

Finally, we observe that often principals can be collected in groups and policies generalized to groups. To this aim we do not need to change neither our analysis, nor our syntax: we just establish a class of groups as a partition on addresses. Then a policy is given by relating which groups can write/read to which. The properties discussed above scale up immediately also to groups.

## 6 An example

Assume that within an organization, *Alice* ( $A$ ), a top manager, creates a file *trend* containing an analysis of the organization trend, i.e. a collection of important and sensitive data. This data should not be disclosed to anybody besides *Alice*. Consider now *Bob* ( $B$ ), one of *Alice*'s subordinates, who wants to acquire some information about the file to exploit it within a competitor organization. *Bob* modifies a web consultant application generally used by *Alice*, to include one hidden operation: a write operation on the channel *hidden* (he uses it as a Trojan horse). Then he gives this application to *Alice*. The application receives through the channel *line* the information on the trend of the organization and if it is negative, i.e. if it is *bad*, then sends *Alice* some suggestion. Otherwise, it sends an acknowledgement of the communication, through channel *ack*. When *Alice* will execute the application, also the hidden write operation can be performed. As a result, *Bob* will acquire the desired information, through the channel *hidden* and he can use it to decide, e.g. to sell his actions, through the channel *sell*. Recall that in the  $\pi$ -calculus, the restriction operator acts as a static binder, i.e. the restricted value is private of the process prefixed by the restriction: for instance, the restricted value *good* is not known outside  $A$ , until it is explicitly sent, as in the example with the action  $\overline{line}\langle good \rangle$ . Below, we present a specification, in which  $A'$  and  $B'$  represent two suitable continuations of the processes  $A$  and  $B$ .

$$\begin{aligned}
 System &= (\nu line) \left( \underbrace{A}_{\|_0\|_0} \mid \underbrace{WebConsultant}_{\|_0\|_1} \right) \mid \underbrace{B}_{\|_1} \\
 A &= (\nu good)(\nu bad) \left( \overline{line}\langle good \rangle.ack(x_1 \in \{ok\}).A' \mid \right. \\
 &\quad \left. \overline{line}\langle bad \rangle.line(x_2 \in \{suggestion\}).A'' \right) \\
 WebConsultant &= (line(z_{trend} \in \{good\}).\overline{ack}\langle ok \rangle) \mid \\
 &\quad (line(y_{trend} \in \{bad\}).\overline{line}\langle suggestion \rangle.\overline{hidden}\langle SideEffect \rangle) \\
 B &= hidden(w \in \{SideEffect\}).\overline{sell}\langle actions \rangle.B'
 \end{aligned}$$

Assign now a high security level  $H$  to  $A$  ( $\|_0\|_0$ ) and to her sub-components ( $\|_0\|_0\|_i$ ), as well as to the *WebConsultant* application ( $\|_0\|_1$ ) and to its sub-components ( $\|_0\|_1\|_i$ ); assign instead low security level  $L$  to  $B$  ( $\|_1$ ). The following is the relevant part of an acceptable estimate.

- $\eta_1(\|_0\|_0\|_0)(line), \rho(z_{trend}) \ni good$  and  $\|_0\|_1\|_0 \in \eta_2(\|_0\|_0\|_0)(line)$ ;
- $\eta_1(\|_0\|_0\|_1)(line), \rho(y_{trend}) \ni bad$  and  $\|_0\|_1\|_1 \in \eta_2(\|_0\|_0\|_1)(line)$ ;
- $\eta_1(\|_0\|_1\|_0)(ack), \rho(x_1) \ni ok$  and  $\|_0\|_0\|_0 \in \eta_2(\|_0\|_1\|_0)(ack)$ ;
- $\eta_1(\|_0\|_1\|_1)(line), \rho(x_2) \ni suggestion$  and  $\|_0\|_0\|_1 \in \eta_2(\|_0\|_1\|_1)(line)$ ;

- $\eta_1(\|_0\|_1\|_1)(hidden), \rho(w) \ni SideEffect$  and  $\|_1 \in \eta_2(\|_0\|_1\|_1)(hidden)$ ;
- $\eta_1(\|_1)(sell) \ni actions$  and  $\vartheta_E \in \eta_2(\|_1)(sell)$ .

It is immediate to check that our system violates the *no read-up/no write-down* policy. In fact, the estimate is such that  $\|_1 \in \eta_2(\|_0\|_1\|_1)(hidden)$ , with  $\|_1 \in L$  and  $\|_0\|_1\|_1 \in H$ . This corresponds to the fact that the high-level sub-process  $System@_{\|_0\|_1\|_1}$  performs a *write down* to the low sub-process  $System@_{\|_1}$  (that, in turn, performs a *read up*), that corresponds to a undesired flow of information from high to low. The same system respects instead the dual property: the integrity of data of high-level processes is indeed preserved.

## 7 Concluding Remarks

We defined a Control Flow Analysis that over-approximates statically the values that can be exchanged along channels during the evolution of a system of mobile processes. We built on the proposal of [8] and exploit to a further extent the syntactic structure of systems. In this way, we traced which component of the system is which?, and so we could carefully approximate the “local” behaviour of each component in terms of the values it can send and receive on a particular channel. Technically, this is done by giving a static interpretation to the main operators of the calculus, which is closer to their dynamic interpretation. This results in a better estimate of the behaviour of the system under consideration, as well as of its components, in terms of the communications predicted. For instance, we discard here some of those considered possible in [8], which actually were not (false positives). Also, we improved the accuracy of the analysis of replicated processes, still maintaining finite our approximations. Much in the style of [5, 7, 6], our Control Flow Analysis analyses a process, plugged in any environment  $E$ , possibly hosting a Dolev-Yao attacker, without explicitly analysing  $E$ . The estimates are invariant under computation steps, and there always exists a least estimate for a given system; also we sketched a procedure for computing it in polynomial time.

We exploited the accuracy of our estimates in predicting communications to statically establish when a given system respects specific security properties. Those considered here mainly cover access control and confidentiality, but many other properties can easily be verified. An important point is that all these checks are made on the *same* estimate. It is then sufficient to compute an estimate *once and for all* and then inspect it. As a matter of fact, all the checks we proposed result from simple combinations of two basic controls: whether a value is sent or is received along

a channel by a specific component of the system. This gives evidence that our approach offers a uniform, flexible and expressive framework for analysing security properties.

The expressivity and generality sketched above make our approach complementary to the more classical Type Systems, where quite often specific type systems are defined *ad hoc* for a particular security property or policy. There is a great number of papers on the subject, among which the seminal ones [2, 17, 26]; below we only consider the most closely related to our present work. General type systems have been proposed to be instantiated for analysing more properties at the same time, as we do. A first example is [19], where the same type system is used to study the input/output behaviour of channels, the upper and lower bounds to the number of active channels, and also confluence. In [17], the authors introduce a security  $\pi$ -calculus, where processes are given security levels. The corresponding type system guarantees that processes of security level  $\sigma$  cannot access resources with a security level higher than  $\sigma$ . In [18], Hennesy and Riley presents a Type System for resource protection in the  $D\pi$ -calculus, a distributed variant of  $\pi$ -calculus: channels are associated with read and write capabilities and the static check controls that processes have the appropriate capabilities. The most related work to ours is [16], where the author present an abstract interpretation based analysis for the  $\pi$ -calculus, and uses it to check confidentiality issues in open systems. Feret's non-uniform analysis distinguishes several instances of the same replicated process, therefore gaining precision with respect to our work.

Also tightly related with our present is [23], because we followed the same approach based on Flow Logic. In this paper, the authors define a control flow analysis for Mobile Ambients, improving [24], and use it to establish properties of mandatory access control.

As a matter of fact, Mobile Ambients [11] and related calculi have been often used to study and establish different security properties. Since [12], several type systems have been proposed, among which the one of [10], used to check access policies.

We conclude this concise list of related work by citing two further papers on types and access control: the first [14] uses a variant of Linda with multiple tuple spaces, and the second [20] deals with typed applets.

**Acknowledgement.** We thank Flemming Nielson and Hanne Riis Nielson for their helpful comments and suggestions concerning this work.

## References

1. M. Abadi and A. D. Gordon. A calculus for cryptographic protocols - The Spi calculus. *Information and Computation*,148(1):1–70, January 1999.
2. M. Abadi, Security protocols and specifications, In *Proc. FoSSaCS'99*, LNCS 1578, pp. 1–13, Springer, Berlin, 1999.
3. D.E. Bell and L.J. LaPadula. Secure computer systems: Unified exposition and Multics interpretation. Technical Report ESD-TR-75-306, Mitre C., 1976.
4. K.J. Biba. Integrity Considerations for Secure Computer Systems. US Airforce Electronic Systems Division, 1977.
5. C. Bodei, P. Degano, R. Focardi, R. Gorrieri, F. Martinelli. Techniques for security checking: Non-Interference vs Control Flow Analysis *Proc. of the Final Workshop Tosca 2001*, ENTCS 62, 2001.
6. C. Bodei, M. Buchholtz, P. Degano, F. Nielson, and H. Riis Nielson. Automatic Validation of Protocol Narration. *Proc. of 16th IEEE Computer Security Foundations Workshop*. IEEE Computer Society, 2003.
7. C. Bodei, P. Degano, F. Nielson, and H. Riis Nielson. Flow Logic for Dolev-Yao Secrecy in Cryptographic Processes. *Future Generation Computer Systems* 18 (6): 747-756, 2002, Elsevier.
8. C. Bodei, P. Degano, F. Nielson, and H. Riis Nielson. Static analysis for the  $\pi$ -calculus with their application to security. *Information and Computation*168: 68-92, 2001.
9. C. Bodei, P. Degano, and C. Priami. “Names of the  $\pi$ -Calculus Agents Handled Locally”. *Theoretical Computer Science*, 253(2):155–184, 2001.
10. M. Bugliesi, G. Castagna and S. Crafa. *Reasoning about security in mobile ambients*. In Proc. of CONCUR'01, LNCS 2154, 2001.
11. L. Cardelli, A. D. Gordon. *Mobile Ambients*. In Proc. of FoSSaCS'98, LNCS 1378, 1998.
12. L. Cardelli, A. D. Gordon. *Types for Mobile Ambients*. In Proc. 26 POPL, ACM,1999.
13. P. Degano and C. Priami. “Enhanced Operational Semantics.”. *ACM Computing Surveys*, 33, 2 (June 2001), 135-176.
14. R. De Nicola, G. Ferrari and R. Pugliese. *Types as specifications of access policies*. In *Security Internet Programming: Security Issues for Mobile and Distributed Objects*, LNCS 1603, 1999.
15. D. Dolev and A.C. Yao, On the security of public key protocols, IEEE TIT, IT-29(12) (1983) 198–208.
16. J. Feret. Confidentiality Analysis of Mobile Systems. In *Proc. of SAS'00*, LNCS 1824, pp. 135–154. Springer, 2000.
17. M. Hennessy and J. Riely. Information Flow vs. Resource Access in the Asynchronous Pi-Calculus. In *Proc. of ICALP'00*, LNCS 1853, pp. 415–427, Springer, 2000.
18. M. Hennessy and J. Riely. *Resource access in systems of mobile agents*. *Information and Computation*, 173:82-120, 2002.
19. B. König. Analysing Input/Output-Capabilities of Mobile Processes with a Generic Type System. In *Proc. of ICALP'00*, LNCS 1853, pp. 403–414, Springer, 2000.
20. X.Leroy and F. Rouaix. *Security properties of typed applets*. In *Security Internet Programming: Security Issues for Mobile and Distributed Objects*, LNCS 1603, 1999.

21. R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes (I and II). *Information and Computation*, 100(1):1–77, 1992.
22. F. Nielson and H. Riis Nielson. Flow logics and operational semantics. *ENTCS*, 10, 1998.
23. F. Nielson, H. Riis Nielson, and M. Buchholtz. Security for mobility. IMM Technical Report, 2002.
24. F. Nielson, H. Riis Nielson, and R. R. Hansen. Validating firewalls using flow logics. *Theoretical Computer Science*, 283(2):381–418, 2002.
25. F. Nielson and H. Seidl, Control-Flow Analysis in Cubic Time, in: Proc. ESOP’01, Lecture Notes in Computer Science, Vol. 2028 (Springer, Berlin, 2001) 252–268.
26. B.C. Pierce, and D. Sangiorgi. Typing and Sub-typing for Mobile Processes. *MSCS* 6(5), pp.409-454,1996.
27. D. Sangiorgi and D. Walker. The Pi-Calculus - A Theory of Mobile Processes Cambridge University Press, 2001.

## A Proofs of Section 3

**Lemma 1** *If the transition  $P \xrightarrow{\alpha} Q$  involves*

- (a) *only the axiom  $P@v \xrightarrow{\alpha} R_i$  then  $Q = h(1, P, v_i, v_i)[w(v_i) \mapsto R_i]$ , with  $i \in \{0, 1\}$ ;*
- (b) *both axioms  $P@v_0 \xrightarrow{\mu_0} R_0$  and  $P@v_1 \xrightarrow{\mu_1} R_1$  then  $Q = h(2, P, v_0, v_1)[w(v_0) \mapsto R_0, w(v_1) \mapsto R_1]$ .*

*Proof.* The proof is by induction on the structure of the derivation, by cases on the last rule used, and by further sub-cases depending on whether a single axiom or two are involved.

**Tau, Out, Sel\_Ein.** In this case, the axiom is  $P@e \xrightarrow{\mu} R_i$ , by Def. 2,  $P@e = P$ , therefore the transition coincides with the axiom and  $Q = R_i = h(1, P, e, e)[w(e) \mapsto R_i]$ .

**Sum<sub>i</sub>.** The process  $P$  is on the form  $P_0 + P_1$  for suitable  $P_0, P_1$  and  $v = \pm_i v_i$ . The transition then involves the axiom  $P_i@v_i = (P_0 + P_1)@ \pm_i v_i \xrightarrow{\mu} R_i$  and its premise involves  $P_i \xrightarrow{\alpha} P'_i$ .

(a) If the transition involves a single axiom, then  $\mu = \alpha$  and by induction hypothesis, we have that  $P'_i = h(1, P_i, v_i, v_i)[w(v_i) \mapsto R_i]$ .

(b) If the transition involves two axioms, then  $\mu = \mu_i$  and let the other axiom involved be  $P_i@v'_i = (P_0 + P_1)@ \pm_i v'_i \xrightarrow{\mu'_i} R'_i$ . In this case, by induction hypothesis we have that  $P'_i = h(2, P_i, v_i, v'_i)[w(v_i) \mapsto R_i][w(v'_i) \mapsto R'_i]$ .

In both cases, the conclusion of the rule leads  $P$  to a process  $Q =$

$Q_0 + Q_1$ , for suitable  $Q_i$  such that  $\begin{cases} Q_{1-i} = \mathbf{0} \\ Q_i = P'_i \end{cases}$  and therefore, by

Def. 5, in (a) we have that  $Q = h(1, P_i, \pm_i v_i, \pm_i v_i)[w(\pm_i v_i) \mapsto R_i]$ ,

while in (b)  $Q = h(2, P_i, \pm_i v_i, \pm_i v'_i)[w(\pm_i v_i) \mapsto R_i, w(\pm_i v'_i) \mapsto R'_i]$ .

**Par<sub>i</sub>.** The process  $P$  is on the form  $P_0|P_1$  for suitable  $P_0, P_1$  and  $\vartheta = \parallel_i \vartheta_i$ . The transition then involves the axiom  $(P_0 | P_1)@ \parallel_i \vartheta_i = P_i @ \vartheta_i \xrightarrow{\mu_0} R_0$  and its premise involves  $P_i \xrightarrow{\alpha} P'_i$ .

(a) If the transition involves a single axiom, then  $\mu = \alpha$  and, by induction hypothesis, we have that  $P'_i = h(1, P_i, \vartheta_i, \vartheta_i)[w(\vartheta_i) \mapsto R_i]$ .

(b) If the transition involves two axioms, then  $\mu = \mu_i$  and let the other axiom involved be  $(P_0 | P_1)@ \parallel_i \vartheta'_i = P_i @ \vartheta'_i \xrightarrow{\mu'_i} R_1$  involved and, then by induction hypothesis, we have that  $P'_i = h(2, P_i, \vartheta_i, \vartheta'_i)[w(\vartheta_i) \mapsto R_i, w(\vartheta'_i) \mapsto R'_i]$ .

In both cases, the conclusion of the rule leads  $P$  to a process  $Q = Q_0|Q_1$ , for suitable  $Q_i$  such that  $\begin{cases} Q_{1-i} = P_{1-i} \\ Q_i = P'_i \end{cases}$  and therefore, by Def. 5, in (a) we have that  $Q = h(1, P, \parallel_i \vartheta_i, \parallel_i \vartheta_i)[w(\parallel_i \vartheta_i) \mapsto R_i]$ , while in in (b)  $Q = h(2, P_i, \parallel_i \vartheta_i, \parallel_i \vartheta'_i)[w(\parallel_i \vartheta_i) \mapsto R_i, w(\parallel_i \vartheta'_i) \mapsto R'_i]$ .

**Com.** The process  $P$  is on the form  $P_0|P_1$ , for suitable  $P_0, P_1$ , and the transition involves both  $(P_0 | P_1)@ \parallel_i \vartheta_i = P_i @ \vartheta_i \xrightarrow{\mu_0} R_i$  and  $(P_0 | P_1)@ \parallel_{1-i} \vartheta_{1-i} = P_{1-i} @ \vartheta_{1-i} \xrightarrow{\mu_1} R_{1-i}$ . The premises of the transition are  $P_i \xrightarrow{\mu_0} P'_i$  and  $P_{1-i} \xrightarrow{\mu_1} P'_{1-i}$ , for suitable  $P'_i$  and  $P'_{1-i}$ , and involve  $P_i @ \vartheta_i \xrightarrow{\mu_0} R_i$  and  $P_{1-i} @ \vartheta_{1-i} \xrightarrow{\mu_1} R_{1-i}$ . The conclusion of the rule *Com* leads  $P$  to a process  $Q = Q_0|Q_1$ , for suitable  $Q_i$  s.t.  $\begin{cases} Q_{1-i} = P'_{1-i} \\ Q_i = P'_i \end{cases}$  and by induction hypothesis,  $P'_i = h(1, P, \vartheta_i, \vartheta_i)[w(\vartheta_i) \mapsto R_i]$  and  $P'_{1-i} = h(1, P, \vartheta_{1-i}, \vartheta_{1-i})[w(\vartheta_{1-i}) \mapsto R_{1-i}]$ . By Def. 5,  $Q = h(2, P, \parallel_i \vartheta_i, \parallel_{1-i} \vartheta_{1-i})[w(\parallel_i \vartheta_i) \mapsto R_i, w(\parallel_{1-i} \vartheta_{1-i}) \mapsto R_{1-i}]$ .

**Close.** Analogous to the previous case.

**Bang.** The process is on the form  $!S$  for suitable  $S$ , and  $\vartheta \#_i \vartheta_i$ . The transition involves the axiom  $P @ \#_i \vartheta_i \xrightarrow{\mu} R_i$ , i.e.  $S @ \vartheta_i \xrightarrow{\mu} R_i$  and so does its premise  $S|S \xrightarrow{\mu} (Q_0|Q_1)$  that involves the axiom  $(S|S)@ \parallel_i \vartheta_i \xrightarrow{\mu} R_i$ .

(a) If the transition involves a single axiom, then  $\mu = \alpha$ , and  $Q_i$ , by

inductive hypothesis, is such that  $\begin{cases} Q_{1-i} = S \\ Q_i = h(1, S, \vartheta_i, \vartheta_i)[w(\vartheta_i) \mapsto R_i] \end{cases}$

(b) If the transition involves two axioms, then  $\mu = \mu_i$  and let the other axiom involved be  $P @ \#_j \vartheta'_i = P_j @ \vartheta'_j \xrightarrow{\mu'_j} R'_j$ , with  $j \in \{0, 1\}$ . Two further cases arise.

(i) If  $j = i$  (the penultimate rule is a *Par*) then, by inductive hypothesis,

esis,  $\begin{cases} Q_{1-i} = S \\ Q_i = h(2, S, \vartheta_i, \vartheta'_j)[w(\vartheta_i) \mapsto R_i, w(\vartheta'_j) \mapsto R_j] \end{cases}$

- (ii) If  $j \neq i$  (the penultimate rule is a *Com*) then, by inductive hypothesis,  $\begin{cases} Q_{1-i} = h(1, S, \vartheta_j, \vartheta_j)[w(\vartheta'_j) \mapsto R_j] \\ Q_i = h(1, S, \vartheta_i, \vartheta_i)[w(\vartheta_i) \mapsto R_i] \end{cases}$
- In both cases, the conclusion of the rule leads  $P$  to a process  $Q = (Q_0|Q_1)!S$  and therefore, by Def. 5,  
 $Q = h(P, \mathbb{!}_i \vartheta_i, \mathbb{!}_j \vartheta'_j)[w(\mathbb{!}_i \vartheta_i) \mapsto R_i, w(\mathbb{!}_j \vartheta'_j) \mapsto R_j]$ .

## B Proofs of Section 4

**Lemma 2** For each process  $P$ ,  $(\rho, \eta, \Phi) \models^\vartheta P$  iff  $\forall \vartheta' \in \text{Addr}(P) : (\rho, \eta, \Phi) \models^{\vartheta \vartheta'} P @ \vartheta'$ . Also,  $(\rho, \eta, \Phi) \models^{\vartheta \vartheta'} P' @ \vartheta'$  if  $P = \pi.P'$ .

*Proof.* By induction on the form of  $P$ .

- if  $P \equiv \mathbf{0}$ : trivial;
- if  $P \equiv (\nu a)P'$  then  $(\rho, \eta, \Phi) \models^\vartheta (\nu a)P'$  iff  $(\rho, \eta, \Phi) \models^\vartheta P'$ ;
- if  $P \equiv (P_0 \text{op}_\diamond P_1)$  then, by definition,  $(\rho, \eta, \Phi) \models^\vartheta (P_0 \text{op}_\diamond P_1)$  if and only if  $(\rho, \eta, \Phi) \models^{\vartheta \diamond_0} P_0 \wedge (\rho, \eta, \Phi) \models^{\vartheta \diamond_1} P_1$  and  $\text{Addr}(P_0 \text{op}_\diamond P_1) = \diamond_0.\text{Addr}(P_0) \cup \diamond_1.\text{Addr}(P_1)$ ;
- If  $P @ \vartheta' \equiv !S$  then, by definition,  $(\rho, \eta, \Phi) \models^\vartheta !S$  iff  $(\rho, \eta, \Phi) \models^{\vartheta \mathbb{!}_0} S$  and  $(\rho, \eta, \Phi) \models^{\vartheta \mathbb{!}_1} S \text{SAddr}(!S) = \mathbb{!}_0.\text{Addr}(S) \cup \mathbb{!}_1.\text{Addr}(S)$ .

Furthermore, if  $P = \pi.P'$ , then by the rules for prefixes  $(\rho, \eta, \Phi) \models^\vartheta \pi.P'$  if and only if  $(\rho, \eta, \Phi) \models^\vartheta P'$ ;

**Lemma 3** If  $(\rho, \eta, \Phi) \models^\vartheta P$  then  $\forall \vartheta_0, \vartheta_1$  such that  $\exists H = h(k, P, \vartheta_0, \vartheta_1)$ ,

- (a)  $(\rho, \eta, \Phi) \models^\vartheta P$  implies  $(\rho, \widehat{\eta}, \Phi) \models^\vartheta P$ ;  
(b)  $(\rho, \widehat{\eta}, \Phi) \models^\vartheta P$  iff  $(\rho, \widehat{\eta}, \Phi) \models^\vartheta H$ .

*Proof.* Item (a) holds by construction of  $(\rho, \widehat{\eta}, \Phi)$ , since  $\vartheta \in \widehat{w}(\vartheta)$ .

We now prove (b), by inducing on the structure of  $P$  and by sub-cases on the form of  $\vartheta_0, \vartheta_1$ .

- If  $P = \mathbf{0}$ : trivial.
- If  $P = \mu.P$  then  $\vartheta_0 = \vartheta_1 = \epsilon$  and  $H = h(1, P, \epsilon, \epsilon) = P$ ;
- If  $P = (\nu a)P'$  then  $h(k, (\nu a)P', \vartheta_0, \vartheta_1) = (\nu a)h(k, P', \vartheta_0, \vartheta_1)$  and, by induction hypothesis on  $P'$ , the thesis holds.
- If  $P = P_0 \text{op}_\diamond P_1$ ,  $\vartheta_0 = \diamond_i \vartheta'_0$  and  $\vartheta_1 = \diamond_i \vartheta'_1$  then we have that  $(\rho, \widehat{\eta}, \Phi) \models^\vartheta P$  iff  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta \diamond_i} P_i \wedge (\rho, \widehat{\eta}, \Phi) \models^{\vartheta \diamond_{1-i}} P_{1-i}$ . Now,  $H = h(k, P_0 \text{op}_\diamond P_1, \diamond_i \vartheta'_0, \diamond_i \vartheta'_1) = P'_0 \text{op}_\diamond P'_1$ , where  $P'_i = h(k, P_i, \vartheta'_0, \vartheta'_1)$  and  $P'_{1-i} = \begin{cases} P_{1-i} & \text{if } \diamond_i = \parallel_i \\ \mathbf{0} & \text{if } \diamond_i = \pm_i \end{cases}$ . We have that  $(\rho, \widehat{\eta}, \Phi) \models^\vartheta H$  if and

- only if  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta \circ_i} P'_i$  and  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta \circ_{1-i}} P'_{1-i}$ . The first holds by induction hypothesis on  $P_i$  and the other holds either because  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta \circ_{1-i}} P_{1-i}$  or because  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta \circ_{1-i}} \mathbf{0}$ .
- If  $P = P_0 \mid P_1$ ,  $\vartheta_0 = \parallel_i \vartheta'_0$  and  $\vartheta_1 = \parallel_{1-i} \vartheta'_1$  then we have that  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta} P$  iff  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta \parallel_i} P_i \wedge (\rho, \widehat{\eta}, \Phi) \models^{\vartheta \parallel_{1-i}} P_{1-i}$ . Now,  $H = h(2, P_0 \mid P_1, \parallel_i \vartheta, \parallel_{1-i} \vartheta') = P'_0 \text{ op}_{\circ} P'_1$ , where  $P'_i = h(k, P_i, \vartheta, \vartheta')$  and  $P'_{1-i} = h(k, P_{1-i}, \vartheta', \vartheta')$ . Now,  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta} H$  holds because both  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta \parallel_i} P'_i$  and  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta \parallel_{1-i}} P'_{1-i}$  hold by induction hypothesis on  $P_i$  and  $P_{1-i}$ .
  - If  $P = !S$  and  $\vartheta_0 = \parallel_i \vartheta'_0$  and  $\vartheta_1 = \parallel_j \vartheta'_1$ , we prove the stronger result  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta} (S \mid S) \mid !S$ , because  $H = h(k, P, \parallel_i \vartheta'_0, \parallel_j \vartheta'_1)$  differs from  $(S \mid S) \mid !S$  only because it may have some inaction  $\mathbf{0}$  in place of other processes within a  $+$ -context. Now, we have to show that the following items hold: (a)  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta \parallel_0} S$ , (b)  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta \parallel_0 \parallel_1} S$  and (c)  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta \parallel_1} !S$ . By hypothesis,  $(\rho, \eta, \Phi) \models^{\vartheta} !S$  and therefore  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta \parallel_0} S$  and  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta \parallel_1} S$ . By definition of estimates, items (a), (b), (c) follow immediately, because  $\parallel_0 \parallel_0, \parallel_0 \parallel_1, \parallel_1 \parallel_0, \parallel_1 \parallel_1 \in \widehat{w}(\parallel_i)$ .

**Theorem 1 [Subject reduction]** *Given a process  $P$ , if  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta} P$ , then*

- (a) *if  $P \equiv Q$ , then  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta} P$  iff  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta} Q$ ;*
- (b) *if  $P \xrightarrow{\mu} Q$  then we have:*
  - (1) *if  $\mu = \tau$  then  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta} Q$ ;*
  - (2) *if  $\mu = \bar{a}b$  or  $\mu = \bar{a}(b)$ , then  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta} Q$  and the transition involves  $P @ \vartheta_0 \xrightarrow{\mu} R_0$ , then  $b \in \widehat{\eta}_1(\vartheta \vartheta_0)(a)$ ; additionally,  $b \in \Phi \wedge \vartheta_E \in \widehat{\eta}_2(\vartheta \vartheta_0)(a)$ , provided that  $a \in \Phi$ ;*
  - (3) *if  $\mu = ab$ , the transition involves  $P @ \vartheta_1 = a(y \in Y).P_1 \xrightarrow{\mu} R_1$  and the condition (\*) holds, then  $b \in \rho(Y)$  and  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta} Q$ , where (\*) is  $(\exists \vartheta_0 : b \in \widehat{\eta}_1(\vartheta \vartheta_0)(a) \text{ and } \vartheta \vartheta_1 \in \widehat{\eta}_2(\vartheta \vartheta_0)(a))$  or (if  $a \in \Phi$  then  $b \in \Phi$  and  $\vartheta \vartheta_1 \in \widehat{\eta}_2(\vartheta_E)(a)$ ).*

*Proof.* The proof for (a) is straightforward, while the proof for (b) is by induction on the construction of  $P \xrightarrow{\mu} Q$  and with sub-cases depending on whether case (1), (2), or (3) applies. Throughout assume that  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta} P$  and  $P \xrightarrow{\mu} Q$  has been deduced with axioms  $P @ \vartheta_0 \xrightarrow{\mu_0} R_0$  (and  $P @ \vartheta_1 \xrightarrow{\mu_1} R_1$ ). Let  $H = h(1, P, \vartheta_i, \vartheta_i)$  and  $Q = H[w(\vartheta_i) \mapsto R_i]$

if the transition has been deduced with only the axiom  $P@v_i \xrightarrow{\mu_i} R_i$  and let  $H = h(2, P, v_0, v_1)$  and  $Q = H[w(v_0) \mapsto R_0, w(v_1) \mapsto R_1]$  if the transition has been deduced with both axioms.

- (1) We have to show that  $(\rho, \widehat{\eta}, \Phi) \models^\vartheta Q$ . In the case of **Tau** this is immediate; and clearly the axioms **Out** and **Sel\_Ein** do not apply. Due to (a) and the induction hypothesis (1), the property is preserved by the rules **Var** and **Res**; and clearly the rule **Open** does not apply. Now, we consider only one of the pairs of the remaining, symmetric rules, and we omit the analogous proofs for the others. In the case of **Sum<sub>0</sub>**, let  $P = P_0 + P_1$ , such that for suitable  $Q_0$ ,  $P_0 \xrightarrow{\tau} Q_0$  (therefore  $Q = Q_0 + \mathbf{0}$ ). Since  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta \pm 1} \mathbf{0}$  is always true and, by induction hypothesis,  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta \pm 0} Q_0$  we can establish  $(\rho, \widehat{\eta}, \Phi) \models^\vartheta Q_0 + \mathbf{0}$ . In the case of rule **Par<sub>0</sub>**, let  $P = P_0 | P_1$ , s.t. for suitable  $Q_0$ ,  $P_0 \xrightarrow{\mu} Q_0$  (therefore  $Q = Q_0 | P_1$ ). From  $(\rho, \widehat{\eta}, \Phi) \models^\vartheta P_0 | P_1$  we have that, for  $i = 0, 1$ ,  $(\rho, \widehat{\eta}, \Phi) \models^{\|i\vartheta} P_i$ . Then the induction hypothesis ensures that  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta \| 0} Q_0$  thereby establishing the desired  $(\rho, \widehat{\eta}, \Phi) \models^\vartheta Q_0 | P_1$ . In the case of rule **Bang**, Lemma 3 guarantees that  $(\rho, \widehat{\eta}, \Phi) \models^\vartheta P$  implies  $(\rho, \widehat{\eta}, \Phi) \models^\vartheta H = h(k, P, v_0, v_1)$ . Now the top-level operator of  $H$  is a “|”, and the induction hypothesis (1) suffices. In the case of **Close**, let  $P = P_0 | P_1$  and, for suitable  $Q_0$  and  $Q_1$ , let  $Q = Q_0 | Q_1$ . Suppose that the transition involves both  $P_0@v_0 \xrightarrow{\bar{a}b} R_0$  and  $P_1@v_1 \xrightarrow{ab} R_1$ , with  $P_0@v_0 = \bar{a}b.R_0$  and  $P_1@v_1 = a(y \in Y).R_1$ . The induction hypothesis (2) ensures that  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta \| 0} Q_0$  and that  $b \in \widehat{\eta}_1 (\vartheta \|_0 v_0)(a)$  and the induction hypothesis (3) ensures that  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta \| 1} Q_1$  and  $b \in \rho(Y)$ , if (\*) holds. Now,  $b \in \widehat{\eta}_1 (\vartheta \|_0 v_0)(a)$  as said above, by Def. 3,  $\text{comp}(\vartheta \|_0 v_0, \vartheta \|_1 v_1)$  thus  $\vartheta \|_1 v_1 \in \widehat{\eta}_2 (\vartheta \|_0 v_0)$ . From  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta \| 0} Q_0$  and  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta \| 1} Q_1$ , we obtain the desired  $(\rho, \widehat{\eta}, \Phi) \models^\vartheta (\nu b)(Q_0 | Q_1)$ . The case of rule **Com** is similar; note however that the first conjunct of condition (\*) may not hold and the output  $\bar{a}b$  can be read by the attacker as reported by  $\vartheta_E \in \widehat{\eta}_2 (\vartheta \|_0 v_0)(a)$ . Additionally, in this case, if  $b \in \Phi$ , the induction hypothesis (2) ensures that  $\vartheta_E \in \widehat{\eta}_2 (\vartheta \|_0 v_0)(a)$ , provided that  $a \in \Phi$ , in which case  $\vartheta \|_1 v_1 \in \widehat{\eta}_2 (\vartheta_E)(a)$ .

- (2) [Case  $\mu = \bar{a}b$ ] We have to show that  $(\rho, \widehat{\eta}, \Phi) \models^\vartheta Q \wedge b \in \widehat{\eta}_1 (\vartheta v_0)(a)$ . The axioms **Tau** and **Sel\_Ein** do not apply. Due to (a) and the in-

duction hypothesis (2) the property is preserved by the rules **Var** and **Res**. The rules **Open**, **Close** and **Com** do not apply. The only interesting case is that of **Out**. Let the transition involve  $P@v_0 = \bar{a}b.R_0 \xrightarrow{\bar{a}b} R_0$ , then, by Lemma 1,  $Q = H[w(v_0) \mapsto R_0]$ , with  $H = h(1, P, v_0, v_0)$  and, by Lemma 3,  $(\rho, \widehat{\eta}, \Phi) \models^v H$ . By definition of  $Q$ , we know that  $H@w(v_0) = P@v_0$  and  $Q@w(v_0) = R_0$ . In particular, from  $(\rho, \widehat{\eta}, \Phi) \models^v H$ , we can deduce that  $(\rho, \widehat{\eta}, \Phi) \models^{v w(v_0)} H@w(v_0) = \bar{a}b.R_0$ , and, by the clause for output in Tab. 2,  $(\rho, \widehat{\eta}, \Phi) \models^{v w(v_0)} R_0 = Q@w(v_0)$ . If  $v_0 = v'_0 v''_0$ , by Lemma 2,  $(\rho, \widehat{\eta}, \Phi) \models^{v w(v'_0 v''_0)} Q@w(v'_0 v''_0)$  if and only if  $(\rho, \widehat{\eta}, \Phi) \models^{v w(v'_0)} Q@w(v'_0)$ . For all other addresses  $v''$ , we have that  $Q@v'' = H@v''$ , and therefore  $(\rho, \widehat{\eta}, \Phi) \models^{v v''} Q@v''$  if and only if  $(\rho, \widehat{\eta}, \Phi) \models^{v v''} H@v''$ . Consequently, by Lemma 2,  $(\rho, \widehat{\eta}, \Phi) \models^v Q$ . The rules **Par**, **Sum** and **Bang** are as before.

[Case  $\mu = \bar{a}(b)$ ] We have to show that  $(\rho, \widehat{\eta}, \Phi) \models^v Q$  and  $b \in \widehat{\eta}_1 (v\vartheta_0)(a)$ . The axioms **Tau**, **Out** and **Sel.Ein** do not apply. Due to (a) and the induction hypothesis (2) the property is preserved by the rules **Var** and **Res**. The rules **Close** and **Com** do not apply. The only interesting case is that of **Open**. Let  $P@v_0 \equiv \bar{a}b.R_0$  and  $Q@v_0 = R_0$ . The induction hypothesis (2) and  $a \neq b$  ensure that  $(\rho, \widehat{\eta}, \Phi) \models^{v\vartheta_0} (v\bar{b})R_0$  and that  $b \in \widehat{\eta}_2 (v\vartheta_0)(a)$ . Thus  $(\rho, \widehat{\eta}, \Phi) \models^{v\vartheta_0} R_0$  and following the same argument of the case above on  $Q = h(1, P, v_0, v_1)[w(v_1) \mapsto R_0]$ , we establish the desired result  $(\rho, \widehat{\eta}, \Phi) \models^v Q$ . Additionally, if  $a \in \Phi$ , then  $b \in \Phi$  and  $v_E \in \widehat{\eta} (v\vartheta_0)(a)$ . The rules **Par**, **Sum** and **Bang** are as before.

- (3) Neither the axioms **Tau**, **Out**, nor the rules **Open**, **Close** and **Com** are applicable. Due to (a) and induction hypothesis (3) the property is preserved by the rules **Var** and **Res**. The only interesting case is that of **Sel.Ein**. Let the transition involve  $P@v_1 = a(y \in Y).P_1 \xrightarrow{ab} R_1 = P_1\{b/y\}$ . By induction hypothesis,  $(\exists v_0 : b \in \widehat{\eta}_1 (v\vartheta_0)(a)$  and  $v\vartheta_1 \in \widehat{\eta}_2 (v\vartheta_0)(a)$ ) or (if  $a \in \Phi$  then  $b \in \Phi$  and  $v\vartheta_1 \in \widehat{\eta}_2 (v_E)(a)$ ). We have to prove that  $b \in \rho(Y)$  and that  $(\rho, \widehat{\eta}, \Phi) \models^v Q$ .

By Lemma 1,  $Q = H[w(v_1) \mapsto R_1]$  and, by Lemma 3,  $(\rho, \widehat{\eta}, \Phi) \models^v H$ . By definition of  $Q$ , we know that  $H@w(v_1) = P@v_1$  and  $Q@w(v_1) = R_1$ . In particular, from  $(\rho, \widehat{\eta}, \Phi) \models^v H$ , we can deduce that

$(\rho, \widehat{\eta}, \Phi) \models^{\vartheta w(\vartheta_1)} H@w(\vartheta_1) = a(y \in Y).P_1$ , and, by the clause for input in Tab. 2,  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta w(\vartheta_0)} P_1$ . Since the transition is fired,  $b \in Y$ ,  $b \in \rho(b) \in \rho(Y)$ .

Also, by hypothesis, either  $b \in \widehat{\eta}_1(\vartheta\vartheta_0)(a)$  or  $b \in \Phi$ . Therefore,  $b \in \widehat{\eta}_1(\vartheta\vartheta_0)(a) \cap \rho(Y)$  or  $b \in \Phi \cap \rho(Y)$ , by the rule for input in Tab. 2 (lines 4 and 6) we also have that  $b \in \rho(y)$  and thus  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta\vartheta_1} P_1\{b/y\} = R_1 = Q@w(\vartheta_1)$ . Now, by Lemma 2, let  $\vartheta_1 = \vartheta'_1\vartheta''_1$ ,  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta w(\vartheta'_1\vartheta''_1)} Q@w(\vartheta'_1\vartheta''_1)$  iff  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta w(\vartheta'_1)} Q@w(\vartheta'_1)$ . For all other addresses  $\vartheta''$ ,  $Q@w(\vartheta''_1) = S@w(\vartheta''_1)$ , therefore:  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta w(\vartheta''_1)} Q@w(\vartheta''_1)$  iff  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta w(\vartheta''_1)} S@w(\vartheta''_1)$ . Consequently, by Lemma 2,  $(\rho, \widehat{\eta}, \Phi) \models^{\vartheta} Q$ . The rules **Par**, **Sum** and **Bang** are as before.

## C Proofs of Section 5

**Theorem 3** *Let  $P$  be a process with pre-estimate  $(\rho, \eta, \Phi)$  and let  $a$  be a channel.*

1. **[no output for  $P@w$  on  $a$ ]** *If  $a \in \text{fn}(P)$  and  $\vartheta_E \notin \eta_2(\vartheta')(a)$ , then whenever  $P \longrightarrow^* P' \xrightarrow{\mu} P''$ , with  $\mu \in \{\bar{a}b, \bar{a}(b)\}$ , and the last transition involves  $P'@w$ , then  $\vartheta' \neq \vartheta_0$ .*
2. **[no input for  $P@w$  on  $a$ ]** *If  $a \in \text{fn}(P)$  and  $\vartheta \notin \eta_2(\vartheta_E)(a)$ , then whenever  $P \longrightarrow^* P' \xrightarrow{ab} P''$ , and the last transition involves  $P'@w$  then  $\vartheta \neq \vartheta_1$ .*
3. **[no communication between  $P@w$  and  $P@w'$  on  $a$ ]** *If  $\vartheta \notin \eta_2(\vartheta')(a)$ , (with  $\text{comp}(\vartheta, \vartheta')$ ), and  $\vartheta, \vartheta' \neq \vartheta_E$ ), then, whenever  $P \longrightarrow^* P' \xrightarrow{\tau} P''$  and the last transition involves  $P'@w_0 = \bar{a}b.P_0$  and  $P'@w_1 = a(y \in Y).P_1$ , then  $\vartheta' \neq \vartheta_0$  or  $\vartheta \neq \vartheta_1$ .*

*Proof.* By Proposition 1, we know that  $\vartheta \in \eta_2(\vartheta')(a)$  implies that  $\forall \vartheta_0 \in \widehat{w}(\vartheta'), \forall \vartheta_1 \in \widehat{w}(\vartheta) : \vartheta_0 \notin \widehat{\eta}_2(\vartheta_1)$ . Moreover, by Theorem 1 and by Lemma 3, we know that a solution  $(\rho, \widehat{\eta}, \Phi)$  is valid for  $P$  and for its continuations and so in the pre-estimate. Thus, it is enough to prove our claim in the case  $P = P'$ .

1. Let  $P@w_0$  be  $\bar{a}b.P_0$  for some  $b$ . Assume, per absurdum, that  $\vartheta' = \vartheta_0$ . Since the transition involves  $P@w_0$ , i.e.  $P@w'$ , Lemma 2 and the hypothesis ensure that  $(\rho, \eta, \Phi) \models^\epsilon P$  implies  $(\rho, \eta, \Phi) \models^{\vartheta'} \bar{a}b.P_0$  and therefore  $(\rho, \eta, \Phi) \models^{\vartheta'} P_0$  and  $b \in \eta_1(\vartheta')(a)$  and since  $a \in \Phi$ , then  $\vartheta_E \in \eta_2(\vartheta')(a)$ . This is against the hypothesis.

2. Let  $P@v_1$  be  $a(y \in Y).P_1$ . Assume, per absurdum, that  $v = v_1$ . Since the transition involves  $P@v_1$ , i.e.  $P@v$ , Lemma 2 and the hypothesis ensure that  $(\rho, \eta, \Phi) \models^\epsilon P$  implies  $(\rho, \eta, \Phi) \models^v a(y \in Y).P_1$  and therefore  $(\rho, \eta, \Phi) \models^v P_1$  and since  $a \in \Phi$  and  $b \in \Phi \cap \rho(Y)$  (in fact,  $b \in \text{fn}(P)$  and  $b \in Y$ ), then  $b \in \rho(y)$  and  $v \in \eta_2(v_E)(a)$ . This is against the hypothesis.
3. Assume, per absurdum, that  $v' = v_0$  and  $v = v_1$ . Since the transition involves  $P@v_0$ , i.e.  $P@v'$ , Lemma 2 and the hypothesis ensure that  $(\rho, \eta, \Phi) \models^\epsilon P$  implies  $(\rho, \eta, \Phi) \models^{v'} \bar{a}b.P_0$  and therefore  $(\rho, \eta, \Phi) \models^{v'} P_0$  and, in particular,  $b \in \eta_1(v')(a)$ . Since the transition involves  $P@v_1$ , i.e.  $P@v$ , Lemma 2 and the hypothesis ensure that  $(\rho, \eta, \Phi) \models^\epsilon P$  implies  $(\rho, \eta, \Phi) \models^v a(y \in Y).P_1$ . Since  $v, v'$  are compatible and  $b \in \rho(Y) \cap \eta_1(v')(a)$  then  $b \in \rho(y)$  and  $v \in \eta_2(v')(a)$ . This is against the hypothesis. Note that  $b \in Y$  implies  $b \in \rho(Y)$ .

**Theorem 4** *Let  $P$  be a process with pre-estimate  $(\rho, \eta, \Phi)$ . If  $b \notin \Phi$ , then  $P$  preserves the secrecy of the value  $b$ .*

*Proof.* We exploit Theorem 1 and use  $P = P'$ . Suppose, per absurdum, that  $\exists a : \mu = \bar{a}b$  (the other case is analogous) and that  $P@v_0 = \bar{a}b.\tilde{P}'$ . Since the transition involves  $P@v_0$ , Lemma 2 and the hypothesis ensure that  $(\rho, \eta, \Phi) \models^\epsilon P$  implies  $(\rho, \eta, \Phi) \models^v \bar{a}b.\tilde{P}'$  and therefore  $(\rho, \eta, \Phi) \models^v \tilde{P}'$  and  $b \in \eta_1(v')(a)$ . Since  $a \in \text{fn}(P)$ , otherwise the transition would not have been possible, then  $a \in \Phi$  and therefore  $b \in \Phi$ . This is against the hypothesis.

**Theorem 6.** *5 Let  $P$  be a process with pre-estimate  $(\rho, \eta, \Phi)$  and  $\mathcal{C}[-]$  be a one hole context with  $\text{vals}(\mathcal{C}[-]) \subseteq \Phi$ . If  $P$  preserves the secrecy of the value  $b \notin \text{fn}(P)$  then  $\mathcal{C}[P]$  still preserves the secrecy of the value  $b$ .*

*Proof.* Suppose, per absurdum, that  $\mathcal{C}[P] \xrightarrow{\mu} \mathcal{C}[P']$ , with  $\mu = \bar{a}b$  (the case  $\mu = \bar{a}(b)$  is analogous). Since  $b \notin \Phi$  as  $P$  preserves the secrecy of  $b$ , then, by Theorem 4,  $b \notin \text{vals}(\mathcal{C}[-])$ . Therefore,  $\mathcal{C}[P] \xrightarrow{\bar{a}(b)} \mathcal{C}[P']$  only if  $P \xrightarrow{\mu} P'$ : contradiction.