PERGAMON

Computer Languages, Systems & Structures 28 (2002) 101–127

COMPUTER LANGUAGES, SYSTEMS & STRUCTURES

www.elsevier.com/locate/cl

Security boundaries in mobile ambients ☆

Chiara Braghin *, Agostino Cortesi, Riccardo Focardi

Dipartimento di Informatica, Università Ca' Foscari di Venezia, Via Torino 155, 30173 Venezia-Mestre, Italy

Received 11 April 2002

Abstract

A new notion of security boundary is introduced to model multilevel security policies in the scenario of mobile systems, within Cardelli and Gordon's "pure" mobile ambients calculus. Information leakage may be expressed in terms of the possibility for a hostile ambient to access confidential data that are not protected inside a security boundary. A control flow analysis is defined, as a refinement of the Hansen–Jensen–Nielsons's CFA, that allows to properly capture boundary crossings. In this way, direct information leakage may be statically detected. © 2002 Elsevier Science Ltd. All rights reserved.

Keywords: Mobile ambients; Security; Static analysis

1. Introduction

1.1. The problem

In distributed systems, resources and data are shared among users with different locations, yielding to a number of security issues that are object of an increasing number of research projects. Mobile ambients calculus has been introduced by Cardelli and Gordon in [3] to model both mobile computation and mobile systems. A challenging problem, within this setting, is to properly tackle security issues.

We focus on *multilevel security*, a particular *mandatory access control* security policy: every entity is bound to a security level (for simplicity, we consider only two levels: high and low), and information may just flow from the low-level to the high one. Typically, two access rules are

☆ Partially supported by MURST Projects "Interpretazione Astratta, Type Systems e Analisi Control-Flow", and "Modelli formali per la sicurezza" and the EU Contract IST-2001-32617 "Models and Types for Security in Mobile Distributed Systems". The paper is an extended and revised version of [1,2].

* Corresponding author.

E-mail addresses: braghin@dsi.unive.it (C. Braghin), cortesi@dsi.unive.it (A. Cortesi), focardi@dsi.unive.it (R. Focardi).

PII: S0096-0551(02)00009-7

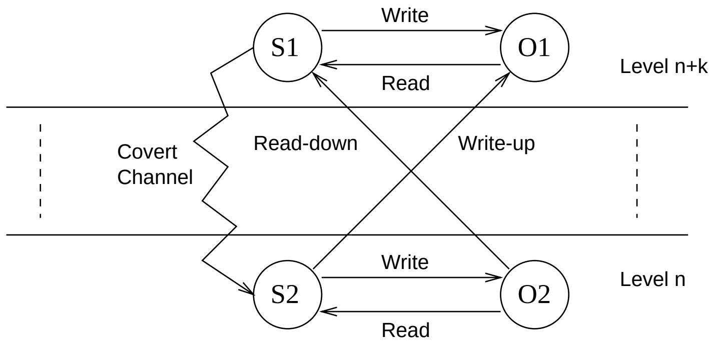


Fig. 1. Multilevel security policy.

imposed: (i) *No Read Up*, a low-level entity cannot access information of a high-level entity; (ii) *No Write Down*, a high-level entity cannot leak information to a low-level entity. Sometimes, these two access controls are not enough as information may be indirectly leaked, through, e.g., some system side effect: a typical example is represented by a resource shared among the security levels which may be alternatively overloaded by some high-level Trojan horse program (causing, e.g., longer response time at all security levels) in order to transmit information to a malicious low-level entity. These indirect ways of transmitting information are called *covert channels*. Fig. 1 summarizes this policy.

In order to detect information leakages, a typical approach (see, e.g., [4–9]) consists in directly defining what is an information flow from one level to another one. Then, it is sufficient to verify that, in any system execution, no information flow is possible from level high to level low. This is the approach we follow in this paper.

1.2. The scenario

We will consider information flow security in the scenario of mobile systems. This particular setting, where code may migrate from one security level to another one, complicates even further the problem of capturing all the possible information leakages. As an example, confidential data may be read by an authorized agent which, moving around, could expose them to unexpected attacks. Moreover, the code itself could be confidential, and so not allowed to be read/executed by lower levels.

In order to study this problem as much abstractly as possible, we consider the "pure" mobile ambients calculus [3], in which no communication channels are present and the only possible actions are represented by the moves performed by mobile processes. This allows to study a very general notion of information flow which should be applicable also to more "concrete" versions of the calculus (e.g., [10,11]).

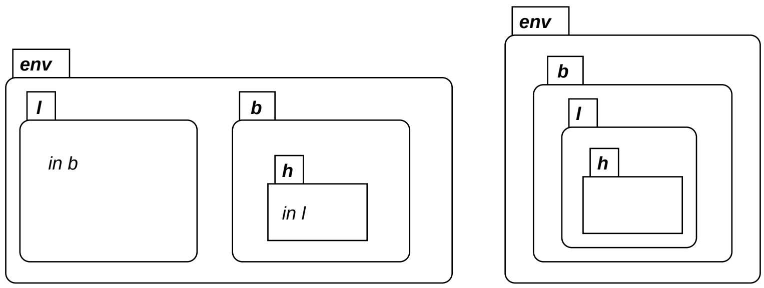
1.3. Verification

We introduce the new notion of "security boundary", that allows to identify ambients that may guarantee to properly protect their content. The intuition is the following: to guarantee absence of information leakage, every high-level data or process should be encapsulated into a boundary ambient, and a boundary ambient can be opened only when it is nested into another boundary ambient. Then, we may even allow that low-level information/processes interfere with high-level data inside a boundary ambient, but we want to be sure that once this happens, the low information/process is trapped: it can neither carry high-level information out of the boundary ambient nor move out of it.

Instead of proposing a new variant of ambient calculus to apply and develop this ideas, we adopt a static analysis approach: given a process P, and a partition of its ambient and capability labels into "high", "low", and "boundary" sets, we aim at providing methods and tools to *verify* that in every execution of process P no direct information leakage occurs.

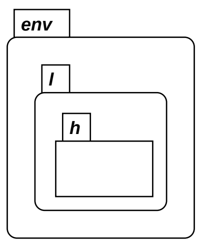
A first approach to this issue may consist in introducing syntactic restrictions on process P, as initially proposed in [2]. A less restrictive and more accurate solution can be obtained by refining the control flow analysis by Hansen et al. [12].

The aim of the analysis proposed in [12] is to calculate an over approximation of ambient nestings, i.e., the analysis returns a set which contains all the ambient nestings that are possible at run-time. However, the fact that all nestings are collected in a unique set, may introduce some "fake" nestings in the result. As an example, consider a low-level ambient ℓ that moves inside a boundary b and, after that, is entered by a high-level ambient h. The initial and the final situation may be depicted as



Initial State Final State

If we write down all the nestings in the form (x, y) meaning that "x may contain y", we obtain $(env, \ell), (env, b), (b, h)$ from the initial state, plus (ℓ, h) from the final one. From this "flat" representation, we obtain (env, ℓ) and (ℓ, h), i.e., a fake nesting showing that h is unprotected (because outside of b), thus possibly exposed to environment attacks.



Here, we propose a more accurate abstract domain that separately considers nestings inside and outside boundaries, yielding to a much more sophisticated control flow analysis. The example above will be analyzed through two separate sets of nestings, $(env, \ell), (env, b)$ and $(b, h), (b, \ell)(\ell, h)$, representing the possible nestings outside and inside boundaries, respectively. Note that now we do not have a nesting path leading from *env* to h. The fake nesting above is not derived by our analysis as it keeps track that h is inside ℓ only when ℓ is inside some boundaries.

The rest of the paper is organized as follows. In Section 2 we introduce the basic terminology on ambient calculus and we briefly report the control flow analysis of [12]. In Section 3, we present the model of multilevel security for mobile agents and we show how to guarantee absence of unwanted information flows through simple syntactic restrictions. In Section 4, we introduce the refined control flow analysis, which is proved to be correct in Section 5. Section 6 concludes the paper with final remarks and comparisons with related works.

2. Background

In this section we introduce the basic terminology of ambient calculus and we briefly report the control flow analysis of [12].

2.1. Mobile ambients

The mobile ambients calculus has been introduced in [3] with the main purpose of explicitly modeling mobility. Indeed, ambients are arbitrarily nested boundaries which can move around through suitable capabilities. The syntax of processes is given as follows, where $n \in \mathbf{Amb}$ denotes an ambient name.

$$
\begin{array}{llll}
P, Q & ::= & (\nu n)P & \text{restriction} \\
 & | & \mathbf{0} & \text{inactivity} \\
 & | & P \mid Q & \text{composition} \\
 & | & !P & \text{replication} \\
 & | & n^{\ell^a}[P] & \text{ambient} \\
 & | & \mathbf{in}^{\ell^t}\, n.P & \text{capability to enter } n \\
 & | & \mathbf{out}^{\ell^t}\, n.P & \text{capability to exit } n \\
 & | & \mathbf{Open}^{\ell^t}\, n.P & \text{capability to open } n
\end{array}
$$

Labels $\ell^a \in \mathsf{Lab}^a$ on ambients and labels $\ell^t \in \mathsf{Lab}^t$ on transitions are introduced as it is customary in static analysis to indicate "program points". They will be useful in the next section when developing the analysis.

Intuitively, the restriction $(\nu n)P$ introduces the new name n and limits its scope to P; process $\mathbf{0}$ does nothing; $P \mid Q$ is P and Q running in parallel; replication provides recursion and iteration as $!P$ represents any number of copies of P in parallel. By $n^{\ell^a}[P]$ we denote the ambient named n with the process P running inside it. The capabilities $\mathbf{in}^{\ell^t} n$ and $\mathbf{out}^{\ell^t} n$ move their enclosing ambients in and out ambient n, respectively; the capability $\mathbf{open}^{\ell^t} n$ is used to dissolve the boundary of a sibling ambient n. The operational semantics of a process P is given through a suitable reduction relation $\rightarrow$ and a structural congruence $\equiv$ between processes (see Appendix A for more details). Intuitively, $P \rightarrow Q$ represents the possibility for P of reducing to Q through some computation.

$$
\begin{array}{lll}
(res) & \beta^{\mathrm{CF}}_{\ell}((\nu n)P) & = \beta^{\mathrm{CF}}_{\ell}(P) \\
(zero) & \beta^{\mathrm{CF}}_{\ell}(\mathbf{0}) & = (\emptyset, \emptyset) \\
(par) & \beta^{\mathrm{CF}}_{\ell}(P \mid Q) & = \beta^{\mathrm{CF}}_{\ell}(P) \sqcup \beta^{\mathrm{CF}}_{\ell}(Q) \\
(repl) & \beta^{\mathrm{CF}}_{\ell}(!P) & = \beta^{\mathrm{CF}}_{\ell}(P) \\
(amb) & \beta^{\mathrm{CF}}_{\ell}(n^{\ell^a}[\,P\,]) & = \beta^{\mathrm{CF}}_{\ell^a}(P) \sqcup (\{(\ell, \ell^a)\}, \{(\ell^a, n)\}) \\
(in) & \beta^{\mathrm{CF}}_{\ell}(\mathbf{in}^{\ell^t}\, n.P) & = \beta^{\mathrm{CF}}_{\ell}(P) \sqcup (\{(\ell, \ell^t)\}, \emptyset) \\
(out) & \beta^{\mathrm{CF}}_{\ell}(\mathbf{out}^{\ell^t}\, n.P) & = \beta^{\mathrm{CF}}_{\ell}(P) \sqcup (\{(\ell, \ell^t)\}, \emptyset) \\
(open) & \beta^{\mathrm{CF}}_{\ell}(\mathbf{open}^{\ell^t}\, n.P) & = \beta^{\mathrm{CF}}_{\ell}(P) \sqcup (\{(\ell, \ell^t)\}, \emptyset)
\end{array}
$$

Fig. 2. Representation function for the control flow analysis.

2.2. Control flow analysis

The control flow analysis described in [12] aims at modeling which processes can be inside what other processes. It works on pairs $(\hat{I}, \hat{H})$, where:

- The first component $\hat{I}$ is an element of $\wp(\mathsf{Lab}^a \times (\mathsf{Lab}^a \cup \mathsf{Lab}^t))$. If a process contains an ambient labeled ℓ^a having inside either a capability or an ambient labeled ℓ, then (ℓ^a, ℓ) is expected to belong to $\hat{I}$.
- The second component $\hat{H} \in \wp(\mathsf{Lab}^a \times \mathbf{Amb})$ keeps track of the correspondence between names and labels. If a process contains an ambient labeled ℓ^a with name n, then (ℓ^a, n) is expected to belong to $\hat{H}$.
- The pairs are component-wise partially ordered.

The analysis is defined by a representation function and a specification. [1] They are recalled, respectively, in Figs. 2 and 3.

The representation function aims at mapping concrete values to their best abstract representation. It is given in terms of a function $\beta^{\mathrm{CF}}_{\ell}(P)$ which basically builds sets $\hat{I}$ and $\hat{H}$ corresponding to process P, with respect to an enclosing ambient labeled ℓ. The representation of a process P is defined as $\beta^{\mathrm{CF}}_{env}(P)$, where *env* is a special label corresponding to the environment.

Example 2.1. Let P be a process of the form: $P = n^{\ell^a_1}[m^{\ell^a_2}[\mathbf{out}^{\ell^t}\, n]\,]$, thus the representation function of P is the following: $\beta^{\mathrm{CF}}_{env}(P) = (\{(env, \ell^a_1), (\ell^a_1, \ell^a_2), (\ell^a_2, \ell^t)\}, \{(\ell^a_1, n), (\ell^a_2, m)\})$.

The specification depicts how the process transforms one abstract representation to another one, and it mostly amounts on recursive check of subprocesses except for the three capabilities *open*, *in*, and *out*. The rule for *open*-capability says that if some ambient labeled ℓ^a has an *open*-capability ℓ^t on an ambient n, that may apply due to the presence of a sibling ambient labeled $\ell^{a'}$ whose

[1] In ambient calculus bound names may be α-converted. For the sake of simplicity, here we are assuming that ambient names are *stable*, i.e., n is indeed a representative for a class of α-convertible names. See [12] for more details on how this can be handled.

$$
\begin{array}{lll}
(res) & (\hat{I},\hat{H}) \models^{\mathrm{CF}} (\nu n)P & \text{iff } (\hat{I},\hat{H}) \models^{\mathrm{CF}} P \\
(zero) & (\hat{I},\hat{H}) \models^{\mathrm{CF}} \mathbf{0} & \text{always} \\
(par) & (\hat{I},\hat{H}) \models^{\mathrm{CF}} P \mid Q & \text{iff } (\hat{I},\hat{H}) \models^{\mathrm{CF}} P \wedge (\hat{I},\hat{H}) \models^{\mathrm{CF}} Q \\
(repl) & (\hat{I},\hat{H}) \models^{\mathrm{CF}} !P & \text{iff } (\hat{I},\hat{H}) \models^{\mathrm{CF}} P \\
(amb) & (\hat{I},\hat{H}) \models^{\mathrm{CF}} n^{\ell^a}[\,P\,] & \text{iff } (\hat{I},\hat{H}) \models^{\mathrm{CF}} P \\
(in) & (\hat{I},\hat{H}) \models^{\mathrm{CF}} \mathbf{in}^{\ell^t}\, n.P & \text{iff } (\hat{I},\hat{H}) \models^{\mathrm{CF}} P \wedge \\
& \multicolumn{2}{l}{\quad \forall \ell^a, \ell^{a'}, \ell^{a''} \in \mathbf{Lab}^a : ((\ell^a,\ell^t) \in \hat{I} \wedge (\ell^{a''},\ell^a) \in \hat{I} \wedge (\ell^{a''},\ell^{a'}) \in \hat{I}} \\
& \multicolumn{2}{l}{\qquad \wedge (\ell^{a'}, n) \in \hat{H}) \implies (\ell^{a'},\ell^a) \in \hat{I}} \\
(out) & (\hat{I},\hat{H}) \models^{\mathrm{CF}} \mathbf{out}^{\ell^t}\, n.P & \text{iff } (\hat{I},\hat{H}) \models^{\mathrm{CF}} P \wedge \\
& \multicolumn{2}{l}{\quad \forall \ell^a, \ell^{a'}, \ell^{a''} \in \mathbf{Lab}^a : ((\ell^a,\ell^t) \in \hat{I} \wedge (\ell^{a'},\ell^a) \in \hat{I} \wedge (\ell^{a''},\ell^{a'}) \in \hat{I}} \\
& \multicolumn{2}{l}{\qquad \wedge (\ell^{a'}, n) \in \hat{H}) \implies (\ell^{a''},\ell^a) \in \hat{I}} \\
(open) & (\hat{I},\hat{H}) \models^{\mathrm{CF}} \mathbf{open}^{\ell^t}\, n.P & \text{iff } (\hat{I},\hat{H}) \models^{\mathrm{CF}} P \wedge \\
& \multicolumn{2}{l}{\quad \forall \ell^a, \ell^{a'} \in \mathbf{Lab}^a : ((\ell^a,\ell^t) \in \hat{I} \wedge (\ell^a,\ell^{a'}) \in \hat{I} \wedge (\ell^{a'}, n) \in \hat{H})} \\
& \multicolumn{2}{l}{\qquad \implies \left\{(\ell^a,\ell') \mid (\ell^{a'},\ell') \in \hat{I}\right\} \subseteq \hat{I}}
\end{array}
$$

Fig. 3. Specification of the control flow analysis.

name is n, then the result of performing that capability should also be recorded in $\hat{I}$, i.e. all the ambients/capabilities nested in $\ell^{a'}$ have to be nested also in ℓ^a. The *in* and *out* capabilities behave similarly.

Example 2.2. Let P be the process of Example 2.1, thus the least solution of P is the pair $(\hat{I},\hat{H})$ where $\hat{I} = \{(env,\ell_1^a),(env,\ell_2^a),(\ell_1^a,\ell_2^a),(\ell_2^a,\ell^t)\}$ and $\hat{H} = \{(\ell_1^a,n),(\ell_2^a,m)\}$.

The correctness of the analysis is proven by showing that every reduction of the semantics is properly mimicked in the analysis.

Theorem 2.3. *Let P and Q be two processes such that $\beta^{\mathrm{CF}}_{env}(P) \sqsubseteq (\hat{I},\hat{H}) \wedge (\hat{I},\hat{H}) \models^{\mathrm{CF}} P \wedge P \to Q$. Then, $\beta^{\mathrm{CF}}_{env}(Q) \sqsubseteq (\hat{I},\hat{H}) \wedge (\hat{I},\hat{H}) \models^{\mathrm{CF}} Q$.*

Intuitively, whenever $(\hat{I},\hat{H}) \models^{\mathrm{CF}} P$ and the representation of P is contained in $(\hat{I},\hat{H})$, we are assured that every nesting of ambients and capabilities in every possible derivative of P is also captured in $(\hat{I},\hat{H})$.

It is important to recall that the resulting control flow analysis applies to any process, and that every process enjoys a *least* analysis.

3. Information flow

In this section, we present a formalization of multilevel security in the setting of mobile ambients (Section 3.1). We apply the control flow of Section 2.2 to the verification of multilevel security and we show that in some cases it is too approximate (Section 3.2). Then, a simple syntactic property is given which allows to very efficiently verify the absence of unwanted information flows and which also solves some of the approximation problems of the control flow (Section 3.3).

3.1. Modeling multilevel security

In order to define multilevel security in mobile ambients we first need to classify information into different levels of confidentiality. We do this by exploiting the labeling of ambients. In particular, we partition the set of ambient labels Lab^a into three disjoint sets $\mathsf{Lab}^a_H, \mathsf{Lab}^a_L$ and Lab^a_B, which stand for *high*, *low* and *boundary* labels.

Given a process, the multilevel security policy may be established by deciding which ambients are the ones responsible for confining confidential information. These are all labeled with boundary labels from set Lab^a_B and we will refer to them as *boundary ambients*. Thus, all the *high-level ambients* must be contained in a boundary ambient and labeled with labels from set Lab^a_H. Finally, all the external ambients are considered *low-level* ones and they are consequently labeled with labels from set Lab^a_L. This is how we will always label processes, and it corresponds to defining the security policy: what is secret, what is not, what is a container of possible secrets. In all the examples, we will use the following notation for labels: $b \in \mathsf{Lab}^a_B$, $h \in \mathsf{Lab}^a_H$, $m, m' \in \mathsf{Lab}^a_L$ and $c, ch, cl, cm, cm' \in \mathsf{Lab}^t$.

As an example consider the following process:

$$P_1 = container^b[hdata^h[\mathbf{out}^c\ container\ \mathbf{0}]\ ]\,|\,Q,$$

where Q contains some low-level ambients. Ambient *container* is a boundary for the high-level data *hdata* (note that data are abstractly represented as ambients). This process is an example of a direct information flow. Indeed, P may evolve to $container^b[\ ]\,|\,hdata^h[\ ]\,|\,Q$, where the high-level *hdata* is out of any boundary ambient, thus vulnerable and potentially accessible by any ambient or process in Q.[2] This flow of high-level data/ambients outside the security boundaries is exactly what we intend to control and avoid. It may be formalized as follows.

Definition 3.1 (*Unprotected*). Given a process P and a labeling Lab^a, $\mathtt{Unprotected}\ (\ell, (\hat{I}, \hat{H})) =$ *true iff* $\exists \ell_1, \ldots, \ell_n \in \mathsf{Lab}^a_L$ *s.t* $(env, \ell_1), (\ell_1, \ell_2), \ldots, (\ell_{n-1}, \ell_n), (\ell_n, \ell) \sqsubseteq \hat{I}$.

Definition 3.2 (*Protected*). Given a process P and a labeling Lab^a, $\mathtt{Protected}\ (\ell, (\hat{I}, \hat{H})) =$ *true iff* $\neg$ $\mathtt{Unprotected}\ (\ell, (\hat{I}, \hat{H}))$.

Definition 3.3 (*Information leakage*). Given a process P and a labeling Lab^a, P leaks secret $h \in \mathsf{Lab}^a_H$ iff $\exists Q$, $P \rightarrow^* Q$ such that $\mathtt{Unprotected}\ (h, \beta^{CF}(Q))$.

In distributed and mobile systems, it is unrealistic to consider a unique boundary, containing all the confidential information. As an example consider two different sites *venice* and *london*, each with some set of confidential information that need to be protected. This can be modeled by defining

[2] Note that the presence of an ambient may be tested by trying to open it or by entering and then exiting from it. A low-level ambient may thus test if *hdata* is present. This may be seen as reading high-level information.

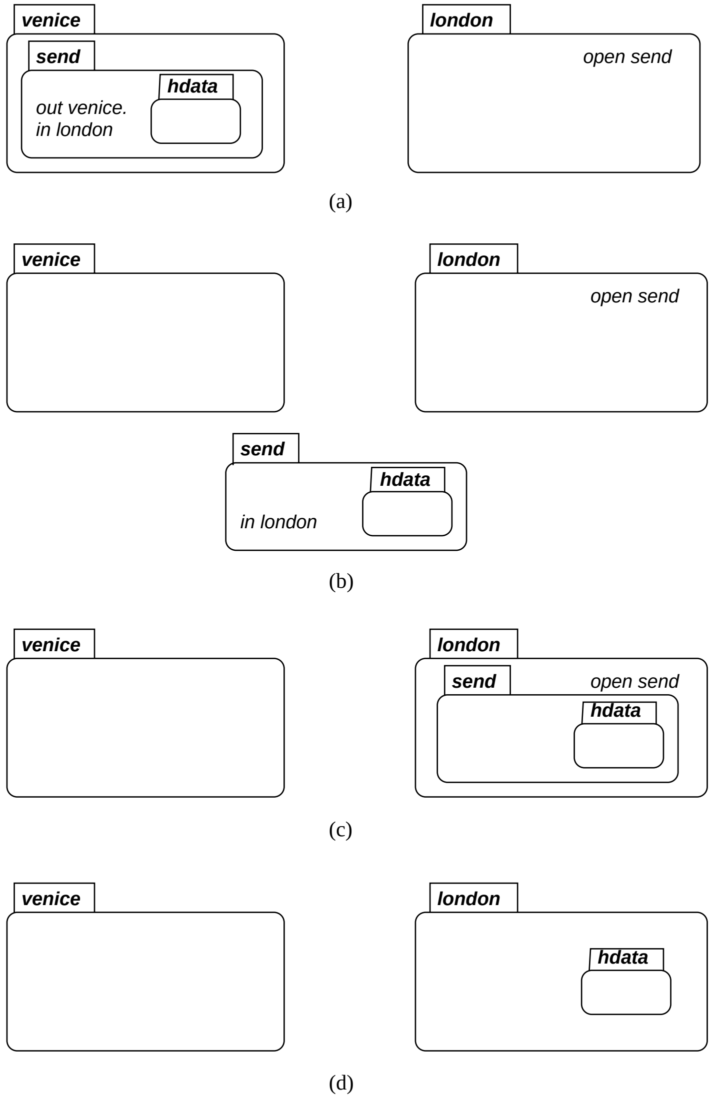


Fig. 4. Venice and london exchange confidential information. (a) *venice* needs to send confidential data *hdata* to london; (b) The confidential data is sent inside the secure "envelope" *send*; (c) The confidential data safely arrive in *london*; (d) The envelope is dissolved to allow confididential data to be accessed in *london*.

two boundary ambients, one for each site: $venice^b[Q_1] \mid london^b[Q_2] \mid Q$. In order to make the model applicable, it is certainly needed a mechanism for moving confidential data from one boundary to another one. This is achieved through another boundary ambient which moves out from the first protected area and into the second one. The following example, also depicted in Fig. 4, describes the exchange of confidential information between the two sites *venice* and *london*:

$$P_2 = venice^b[send^b[\mathbf{out}^c\, venice.\mathbf{in}^c\, london \mid hdata^h[\ ]]\ ] \mid london^b[\mathbf{open}^c\, send] \mid Q.$$

The process may evolve to the following one (see steps (b) and (c) of Fig. 4):

$$venice^b\mathbf{[\ ]} \mid london^b\mathbf{[open}^c send \mid send^b\mathbf{[}hdata^h\mathbf{[\ ]\ ]\ ]} \mid Q$$

and finally to (see step (d) of Fig. 4):

$$venice^b\mathbf{[\ ]} \mid london^b\mathbf{[}hdata^h\mathbf{[\ ]\ ]} \mid Q.$$

Note that *send* is labeled as a boundary ambient. Thus, high-level data *hdata* is always protected by boundary ambients, during the whole execution. Observe that in this example we use the same label for all boundary ambients. If we were interested to distinguish ambient nestings inside different boundaries, it would be enough to assign different boundary labels.

3.2. Verifying absence of information leakage

In this section, we study how to verify that no leakage of secret data/ambients outside the boundary ambients is possible. A natural approach could be the direct application of the control flow of [12] reported in Section 2.2.

Corollary 3.4. *Let P be a process such that $\beta^{\mathrm{CF}}_{env}(P) \sqsubseteq (\hat{I},\hat{H}) \wedge (\hat{I},\hat{H}) \models^{\mathrm{CF}} P \wedge h \in \mathbf{Lab}^a_H$ s.t.* `Protected` $(h,(\hat{I},\hat{H})) \Rightarrow P$ *does not leak secret h.*

Proof. By contradiction, assume that P leaks secret h, thus there exists a process Q s.t. $P \to^* Q$ and `Unprotected` $(h,\beta^{\mathrm{CF}}(Q))$. By iterating Theorem 2.3, we obtain that $\beta^{\mathrm{CF}}(Q) \sqsubseteq (\hat{I},\hat{H})$. From `Unprotected` $(h,\beta^{\mathrm{CF}}(Q))$ and $\beta^{\mathrm{CF}}(Q) \sqsubseteq (\hat{I},\hat{H})$, it follows that `Unprotected` $(h,(\hat{I},\hat{H}))$, leading to a contradiction. □

As an example, consider again the process presented above:

$$P_2 = venice^b\mathbf{[}send^b\mathbf{[out}^c venice.\mathbf{in}^c london \mid hdata^h\mathbf{[\ ]\ ]\ ]} \mid london^b\mathbf{[open}^c send\mathbf{]}.$$

The least analysis for this process can be easily shown to be the following:

$$\hat{I}_2 = \{(env,b),(b,b),(b,h),(b,c)\},$$

$$\hat{H}_2 = \{(b,venice),(b,send),(b,london),(h,hdata)\}.$$

Notice that h is always contained inside b, i.e., it is protected by a boundary ambient. Formally, $\beta^{\mathrm{CF}}_{env}(P_2) \sqsubseteq (\hat{I}_2,\hat{H}_2)$, $(\hat{I}_2,\hat{H}_2) \models^{\mathrm{CF}} P_2$ and `Protected` $(h,(\hat{I}_2,\hat{H}_2))$, thus Corollary 3.4 guarantees that P_2 does not leak secret h.

However, the fact that the analysis simply collects all the potential nesting without considering the temporal ordering of the events, may sometimes lead to unnecessary approximations. As an example, consider again process P_2 above, and suppose that high-level data is willing to enter some filter process, which could possibly be low-level code:

$$P_3 = venice^b\mathbf{[}send^b\mathbf{[out}^c venice.\mathbf{in}^c london \mid hdata^h\mathbf{[in}^{ch} filter\mathbf{]\ ]\ ]} \mid$$
$$\mid london^b\mathbf{[open}^c send\mathbf{]} \mid filter^m\mathbf{[in}^c send\mathbf{]} \mid \mathbf{open}^{cl} filter.$$

Note that the filter behaves correctly with respect to multilevel security rules, i.e., it only enters boundaries. In particular, this means that it will never carry high-level data outside the security boundaries. However, if we perform the control flow analysis we obtain the following least solution:

$$\hat{I}_3 = \{(env,b),(env,h),(env,m),(env,cl),(env,c),(b,b),(b,h),(b,m),(b,c),$$
$$(h,ch),(m,h),(m,c)\},$$
$$\hat{H}_3 = \{(b,venice),(b,send),(b,london),(h,hdata),(m,filter)\}.$$

Note that in this solution h appears at the environment level, leading to `Unprotected` $(h,(\hat{I}_3,\hat{H}_3)))$, showing a potential attack. However, as observed before, there is no execution leading to such a situation. The reason why the analysis looses precision here, is due to the fact that h enters an m ambient which might be opened at the environment level, but the analysis does not capture the fact that h enters m only after it has crossed the boundary, and then it never gets out of it.

3.3. A syntactic approach

Syntactic conditions may be imposed on processes that are sufficient to prove the absence of leakage of secret data/ambients outside the boundary ambients. Moreover, such conditions properly deal with the situation discussed before. Let us briefly recall the main results in this direction, as first presented in [2].

The idea is to control the $\mathbf{out}^l n$ and $\mathbf{open}^l n$ capabilities executed on a boundary ambient n. In particular, we require that such capabilities may only be performed by boundary ambients.

First, we characterize a subset of capability labels, in order to mark *out* and *open* capabilities that refer to boundary ambients. Let $\mathsf{Lab}^t_O \subseteq \mathsf{Lab}^t$ be the subset of labels that refer to *out* and *open* capabilities, and let $\mathsf{Lab}^t_{BM} \subseteq \mathsf{Lab}^t_O$ be a subset of this set of *out* and *open* capability labels (*BM* stands for *boundary move* capabilities). Let also $\phi : \mathsf{Lab}^t \to \wp(\mathbf{Amb})$ be a function that maps a capability label ℓ^t to the set of ambient names on which all the capabilities labeled by ℓ^t operate.

Given a process P, the conditions that should be imposed on $(\hat{I}_0,\hat{H}_0) = \beta^{\mathrm{CF}}_{env}(P)$ to guarantee absence of information leakage are the following:

(i) $(\ell^a,n) \in \hat{H}_0, \ell^a \in \mathsf{Lab}^a_B, n \in \phi(\ell^t), \ell^t \in \mathsf{Lab}^t_O \Rightarrow \ell^t \in \mathsf{Lab}^t_{BM}$,
(ii) $(\ell,\ell') \in \hat{I}_0, \ell' \in \mathsf{Lab}^t_{BM} \Rightarrow \ell \in \mathsf{Lab}^a_B$.

Observe that condition (i) turns out to be a well-formedness condition over labeling. It requires that all the *out* and *open* capabilities that operate on boundary ambients are labeled as boundary moves (i.e., with labels in set Lab^t_{BM}). If this condition is initially satisfied by P (i.e., by $\beta^{\mathrm{CF}}_{env}(P)$), then it will hold also for every derivative of P, as the labeling cannot change during process execution.

Condition (ii) requires that every *out* and *open* boundary move is executed inside a boundary ambient. Note that, in general, this may be not preserved when P evolves. Indeed, the following theorem states that also condition (ii) is preserved, within every execution of P.

Theorem 3.5. *If the representation function* $\beta^{\mathrm{CF}}_{env}(P)$ *fulfills conditions* (i)–(ii), *then the least solution* $(\hat{I},\hat{H}) \models^{\mathrm{CF}} P$ *to the control flow analysis in* [12] *enjoys these conditions as well.*

Proof. Let $(\hat{I}_\beta, \hat{H}_\beta) = \beta^{\mathrm{CF}}_{env}(P)$, and assume $(\hat{I}_\beta, \hat{H}_\beta)$ fulfills conditions (i) and (ii). Now consider the least solution $(\hat{I}, \hat{H}) \models^{\mathrm{CF}} P$ (such that $\beta^{\mathrm{CF}}_{env}(P) \sqsubseteq (\hat{I}, \hat{H})$). To be a least solution, a pair $(\hat{I}, \hat{H})$ must satisfy $\hat{H} = \hat{H}_\beta$. In fact, notice that the analysis requires no changes to $\hat{H}$, thus the least $\hat{H}$ containing $\hat{H}_\beta$ is $\hat{H}_\beta$ itself. This proves that the least solution $(\hat{I}, \hat{H})$ satisfies (i).

Let us now turn to the first component of $(\hat{I}, \hat{H})$. A way for calculating the least solution is to start from $\beta^{\mathrm{CF}}_{env}(P)$, and then add all the pairs required by the analysis, until a fix point is reached. Notice that every nesting in $\hat{I} \setminus \hat{I}_\beta$ (i.e., all the nestings added by the analysis) should have been required by at least one of the capability rules. If this is not the case, we could remove this kind of nestings, obtaining a smaller analysis, leading to a contradiction.

Let $(\ell, \ell') \in \hat{I} \setminus \hat{I}_\beta$ and $\ell' \in \mathsf{Lab}^t_{BM}$. Notice that ℓ' is a transition label, thus the pair (ℓ, ℓ') may only have been added by the *open* rule (in fact, *in* and *out* rules add pairs $(\ell^{a'}, \ell^a)$ with $\ell^a \in \mathsf{Lab}^a$). By induction over the fix point algorithm steps, we may assume that before the application of the *open* rule the pair $(\hat{I}, \hat{H})$ satisfies both (i) and (ii) conditions. Now, we consider $\hat{I} \cup \{(\ell^a, \ell') \mid (\ell^{a'}, \ell') \in \hat{I}\}$ with $(\ell^a, \ell^t) \in \hat{I} \wedge (\ell^a, \ell^{a'}) \in \hat{I} \wedge (\ell^{a'}, n) \in \hat{H}$. We focus on the case $\ell' \in \mathsf{Lab}^t_{BM}$.

Notice that we have $open^{\ell^t} n$, thus $n \in \phi(\ell^t)$. As $\ell' \in \mathsf{Lab}^t_{BM}$ and $(\ell^{a'}, \ell') \in \hat{I}$, by induction on (ii), we have that $\ell^{a'} \in \mathsf{Lab}^a_B$. By induction on (i), we obtain that $\ell^t \in \mathsf{Lab}^t_{BM}$, too. By applying again induction on (ii), we also obtain that $\ell^a \in \mathsf{Lab}^a_B$. This means that the set $\{(\ell^a, \ell') \mid (\ell^{a'}, \ell') \in \hat{I}\}$, where $\ell' \in \mathsf{Lab}^t_{BM}$ adds to $\hat{I}$ only pairs with $\ell^a \in \mathsf{Lab}^a_B$, thus still satisfying (ii). $\square$

Condition (ii) basically states two important behavioral properties: every time a boundary ambient is opened, this is done inside another boundary ambient; the only ambients that may exit from boundary ambients are boundary ambients. By induction on reduction rules of mobile ambients it is easy to prove the following information flow theorem.

Theorem 3.6. *If $\beta^{\mathrm{CF}}_{env}(P)$ fulfills conditions* (i)–(ii), *then process P does not leak any secret $h \in \mathsf{Lab}^a_H$.*

Note that the two conditions above are definitely easy to check. Consider again the examples presented above. In particular,

$$P_1 = container^b[hdata^h[\mathbf{out}^c container.\mathbf{0}]\,]\mid Q$$

does not satisfy condition (ii). In fact, $\mathbf{out}^c container$ should be labeled as a boundary move by condition (i). However this makes a boundary move executable in a high-level ambient, invalidating condition (ii). On the other side, the second example

$$P_2 = venice^b[send^b[\mathbf{out}^c venice.\mathbf{in}^c london \mid hdata^h[\,]\,]\,]\mid london^b[\mathbf{open}^c send]\mid Q$$

fulfills both the conditions, with $c \in \mathsf{Lab}^t_{BM}$. This proves that *hdata*, in every execution, is always nested inside a boundary ambient, i.e., P_2 does not leak secret h.

The syntactic conditions successfully applies also to the extended example with *hdata* entering the filter:

$$P_3 = venice^b[send^b[\mathbf{out}^c venice.\mathbf{in}^c london \mid hdata^h[\mathbf{in}^{ch} filter]\,]\,]\mid$$
$$\mid london^b[\mathbf{open}^c send]\mid filter^m[\mathbf{in}^c send]\mid \mathbf{open}^{cl} filter.$$

Also in this case, we are able to prove that P_3 does not leak h. Note that this was not provable through the control flow analysis introduced in Section 2.2.

The approach above may also be adapted to the case in which the external environment (e.g., any malicious process put in parallel with the analyzed process P) does not fulfill the required conditions. This is indeed reasonable in a distributed open system. The idea is to suitably restrict the scope of boundary ambients and provide low-level ambients with some "taxi" processes that, once entered, bring the client inside restricted boundaries. Let $b_1,\dots,b_n$ represent all the boundary ambients of process P. Then consider process

$$(\nu b_1,\dots,b_n)(P \mid !t_1[\mathbf{in}^l b_1] \mid \dots \mid !t_n[\mathbf{in}^l b_n]) \mid Q.$$

As $b_1,\dots,b_n$ are restricted names, they may not appear in Q. As a consequence, if P fulfills the conditions (i)–(ii), this is sufficient to prove that the whole system (whatever Q is considered) satisfies such conditions, too. It is indeed simple to prove the following.

Proposition 3.7. *If $\beta^{\mathrm{CF}}_{env}(P)$ fulfills conditions* (i)–(ii), *then, for all Q (labeled in $\mathsf{Lab}^a_L \cup \mathsf{Lab}^t \setminus \mathsf{Lab}^t_{BM}$),*

$$\beta^{\mathrm{CF}}_{env}((\nu b_1,\dots,b_n)(P \mid !t_1[\mathbf{in}^l b_1] \mid \dots \mid !t_n[\mathbf{in}^l b_n]) \mid Q)$$

fulfills conditions (i)–(ii).

Note that processes $!t_i[\mathbf{in}^l b_i]$ allow any low-level ambient to enter boundary b_i. So, legitimate flows from low to high-level are possible even if boundaries are restricted. Note also that the condition on the labeling of Q simply means that Q only contains low-level ambients and its capabilities are not (incorrectly) labeled as boundary moves.

4. Refining the control flow analysis

In this section we introduce a refinement of the control flow analysis of [12] reported in Section 2.2, that incorporates into the analysis the ideas discussed in Section 3, thus yielding to a more accurate tool for detecting unwanted boundary crossings. The resulting analysis improves also with respect to the syntactic properties presented in Section 3.3.

The main idea is to split the $\hat{I}$ component of the abstract domain in two (not necessarily disjoint) sets, that maintain nesting information about ambients protected by boundaries, and about "unprotected" ones, respectively.

The refined control flow analysis works on triplets $(\hat{I}_B, \hat{I}_E, \hat{H})$, where:

- The first component $\hat{I}_B$ is an element of $\wp(\mathsf{Lab}^a \times (\mathsf{Lab}^a \cup \mathsf{Lab}^t))$. If a process contains either a capability or an ambient labeled ℓ inside an ambient labeled ℓ^a which is a boundary or an ambient nested inside a boundary (referred as *protected ambient*) then (ℓ^a, ℓ) is expected to belong to $\hat{I}_B$. As long as a high-level datum is contained inside a protected ambient there is no unwanted information flow.
- The second component $\hat{I}_E$ is still an element of $\wp(\mathsf{Lab}^a \times (\mathsf{Lab}^a \cup \mathsf{Lab}^t))$. If a process contains either a capability or an ambient labeled ℓ inside an ambient labeled ℓ^a which is not protected, then (ℓ^a, ℓ) is expected to belong to $\hat{I}_E$.

$$
\begin{array}{lll}
 & \beta^{\mathrm{B}}(P) & = \beta^{\mathrm{B}}_{env,False}(P) \\
(res) & \beta^{\mathrm{B}}_{\ell,Protected}((\nu n)P) & = \beta^{\mathrm{B}}_{\ell,Protected}(P) \\
(zero) & \beta^{\mathrm{B}}_{\ell,Protected}(\mathbf{0}) & = (\emptyset,\emptyset,\emptyset) \\
(par) & \beta^{\mathrm{B}}_{\ell,Protected}(P \mid Q) & = \beta^{\mathrm{B}}_{\ell,Protected}(P) \sqcup \beta^{\mathrm{B}}_{\ell,Protected}(Q) \\
(repl) & \beta^{\mathrm{B}}_{\ell,Protected}(!P) & = \beta^{\mathrm{B}}_{\ell,Protected}(P) \\
(amb) & \beta^{\mathrm{B}}_{\ell,Protected}(n^{\ell^a}[\,P\,]) & = \text{case } Protected \text{ of} \\
 & & \quad \text{True} : \beta^{\mathrm{B}}_{\ell^a,Protected}(P) \sqcup (\{(\ell,\ell^a)\},\emptyset,\{(\ell^a,n)\}) \\
 & & \quad \text{False: if } (\ell_a \in \mathbf{Lab}^a_B) \text{ then} \\
 & & \qquad \text{let Protected}' = \text{True else Protected}' = \text{False in} \\
 & & \qquad \beta^{\mathrm{B}}_{\ell^a,Protected'}(P) \sqcup (\emptyset,\{(\ell,\ell^a)\},\{(\ell^a,n)\}) \\
(in) & \beta^{\mathrm{B}}_{\ell,Protected}(\mathbf{in}^{\ell^t}\, n.P) & = \text{case } Protected \text{ of} \\
 & & \quad \text{True} : \beta^{\mathrm{B}}_{\ell,Protected}(P) \sqcup (\{(\ell,\ell^t)\},\emptyset,\emptyset) \\
 & & \quad \text{False: } \beta^{\mathrm{B}}_{\ell,Protected}(P) \sqcup (\emptyset,\{(\ell,\ell^t)\},\emptyset) \\
(out) & \beta^{\mathrm{B}}_{\ell,Protected}(\mathbf{out}^{\ell^t}\, n.P) & = \text{case } Protected \text{ of} \\
 & & \quad \text{True} : \beta^{\mathrm{B}}_{\ell,Protected}(P) \sqcup (\{(\ell,\ell^t)\},\emptyset,\emptyset) \\
 & & \quad \text{False: } \beta^{\mathrm{B}}_{\ell,Protected}(P) \sqcup (\emptyset,\{(\ell,\ell^t)\},\emptyset) \\
(open) & \beta^{\mathrm{B}}_{\ell,Protected}(\mathbf{open}^{\ell^t}\, n.P) & = \text{case } Protected \text{ of} \\
 & & \quad \text{True} : \beta^{\mathrm{B}}_{\ell,Protected}(P) \sqcup (\{(\ell,\ell^t)\},\emptyset,\emptyset) \\
 & & \quad \text{False: } \beta^{\mathrm{B}}_{\ell,Protected}(P) \sqcup (\emptyset,\{(\ell,\ell^t)\},\emptyset)
\end{array}
$$

Fig. 5. Representation function for the refined control flow analysis.

- The third component $\hat{H}$, as before, keeps track of the correspondence between names and labels. If a process contains an ambient labeled ℓ^a with name n, then (ℓ^a, n) is expected to belong to $\hat{H}$.

Also in this case, the analysis is defined by a representation function and a specification. They are depicted, respectively, in Figs. 5 and 6.

Example 4.1. Let P be the process of Example 2.1, i.e., of the form: $P = n^{\ell^a_1}[m^{\ell^a_2}[\mathbf{out}^{\ell^t} n]\,]$, with $\ell^a_1 \in \mathbf{Lab}^a_B$ and $\ell^a_2 \in \mathbf{Lab}^a_L$, thus the representation function of P is the following: $\beta^{\mathrm{B}}_{env}(P) = (\{(\ell^a_1, \ell^a_2), (\ell^a_2, \ell^t)\}, \{(env, \ell^a_1)\}, \{(\ell^a_1, n), (\ell^a_2, m)\})$.

Example 4.2. Let P be the process of Example 4.1. The least solution of P is the triplet $(\hat{I}_B, \hat{I}_E, \hat{H})$ where $\hat{I}_B = \{(\ell^a_1, \ell^a_2), (\ell^a_2, \ell^t)\}$, $\hat{I}_E = \{(env, \ell^a_1), (env, \ell^a_2), (\ell^a_2, \ell^t)\}$, and $\hat{H} = \{(\ell^a_1, n), (\ell^a_2, m)\}$. Observe that $(\hat{I}_B, \hat{I}_E, \hat{H})$ strictly contains $\beta^{\mathrm{B}}_{env}(P)$, as expected being $(\hat{I}_B, \hat{I}_E, \hat{H})$ a safe approximation.

$$
\begin{array}{lll}
(res) & (\hat{I_B}, \hat{I_E}, \hat{H}) \models^{\mathrm{B}} (\nu n)P & \text{iff } (\hat{I_B}, \hat{I_E}, \hat{H}) \models^{\mathrm{B}} P \\
(zero) & (\hat{I_B}, \hat{I_E}, \hat{H}) \models^{\mathrm{B}} \mathbf{0} & \text{always} \\
(par) & (\hat{I_B}, \hat{I_E}, \hat{H}) \models^{\mathrm{B}} P \mid Q & \text{iff } (\hat{I_B}, \hat{I_E}, \hat{H}) \models^{\mathrm{B}} P \ \wedge\ (\hat{I_B}, \hat{I_E}, \hat{H}) \models^{\mathrm{B}} Q \\
(repl) & (\hat{I_B}, \hat{I_E}, \hat{H}) \models^{\mathrm{B}} !P & \text{iff } (\hat{I_B}, \hat{I_E}, \hat{H}) \models^{\mathrm{B}} P \\
(amb) & (\hat{I_B}, \hat{I_E}, \hat{H}) \models^{\mathrm{B}} n^{\ell^a}[\,P\,] & \text{iff } (\hat{I_B}, \hat{I_E}, \hat{H}) \models^{\mathrm{B}} P
\end{array}
$$

$$
\begin{array}{ll}
(in) & (\hat{I_B}, \hat{I_E}, \hat{H}) \models^{\mathrm{B}} \mathbf{in}^{\ell^t}\, n.P \quad \text{iff } (\hat{I_B}, \hat{I_E}, \hat{H}) \models^{\mathrm{B}} P \wedge \\
& \forall \ell^a, \ell^{a'}, \ell^{a''} \in \mathbf{Lab}^a : \\
& \quad \text{case } ((\ell^a, \ell^t) \in \hat{I_B} \ \wedge\ (\ell^{a''}, \ell^a) \in \hat{I_B} \ \wedge\ (\ell^{a''}, \ell^{a'}) \in \hat{I_B} \ \wedge\ (\ell^{a'}, n) \in \hat{H}) \\
& \qquad \Longrightarrow\ (\ell^{a'}, \ell^a) \in \hat{I_B} \\
& \quad \text{case } ((\ell^a, \ell^t) \in \hat{I_B} \ \wedge\ (\ell^{a''}, \ell^a) \in \hat{I_E} \ \wedge\ (\ell^{a''}, \ell^{a'}) \in \hat{I_E} \ \wedge\ \ell^a \in \mathbf{Lab}^a_B \ \wedge\ (\ell^{a'}, n) \in \hat{H}) \ \Longrightarrow \\
& \qquad \text{if } (\ \ell^{a'} \in \mathbf{Lab}^a_B) \text{ then } (\ell^{a'}, \ell^a) \in \hat{I_B} \\
& \qquad \text{else } (\ell^{a'}, \ell^a) \in \hat{I_E} \\
& \quad \text{case } ((\ell^a, \ell^t) \in \hat{I_E} \ \wedge\ (\ell^{a''}, \ell^a) \in \hat{I_E} \ \wedge\ (\ell^{a''}, \ell^{a'}) \in \hat{I_E} \ \wedge\ (\ell^{a'}, n) \in \hat{H}) \ \Longrightarrow \\
& \qquad \text{if } (\ \ell^{a'} \in \mathbf{Lab}^a_B) \\
& \qquad\quad \text{then } (\ell^{a'}, \ell^a) \in \hat{I_B} \ \wedge\ \left\{(\ell, \ell') \in \hat{I_E} \mid path_E(\ell^a, \ell)\right\} \subseteq \hat{I_B} \\
& \qquad \text{else } (\ell^{a'}, \ell^a) \in \hat{I_E} \\
(out) & (\hat{I_B}, \hat{I_E}, \hat{H}) \models^{\mathrm{B}} \mathbf{out}^{\ell^t}\, n.P \quad \text{iff } (\hat{I_B}, \hat{I_E}, \hat{H}) \models^{\mathrm{B}} P \wedge \\
& \forall \ell^a, \ell^{a'}, \ell^{a''} \in \mathbf{Lab}^a : \\
& \quad \text{case } ((\ell^a, \ell^t) \in \hat{I_B} \ \wedge\ (\ell^{a'}, \ell^a) \in \hat{I_E} \cup \hat{I_B} \ \wedge\ (\ell^{a''}, \ell^{a'}) \in \hat{I_E} \ \wedge\ (\ell^{a'}, n) \in \hat{H}) \ \Longrightarrow \\
& \qquad \text{if } (\ \ell^a \in \mathbf{Lab}^a_B) \text{ then } (\ell^{a''}, \ell^a) \in \hat{I_E} \\
& \qquad \text{else } (\ell^{a''}, \ell^a) \in \hat{I_E} \ \wedge\ \left\{(\ell, \ell') \in \hat{I_B} \mid path_B(\ell^a, \ell)\right\} \subseteq \hat{I_E} \\
& \quad \text{case } ((\ell^a, \ell^t) \in \hat{I_B} \ \wedge\ (\ell^{a'}, \ell^a) \in \hat{I_B} \ \wedge\ (\ell^{a''}, \ell^{a'}) \in \hat{I_B} \ \wedge\ (\ell^{a'}, n) \in \hat{H}) \\
& \qquad \Longrightarrow\ (\ell^{a''}, \ell^a) \in \hat{I_B} \\
& \quad \text{case } ((\ell^a, \ell^t) \in \hat{I_E} \ \wedge\ (\ell^{a'}, \ell^a) \in \hat{I_E} \ \wedge\ (\ell^{a''}, \ell^{a'}) \in \hat{I_E} \ \wedge\ (\ell^{a'}, n) \in \hat{H}) \\
& \qquad \Longrightarrow\ (\ell^{a''}, \ell^a) \in \hat{I_E} \\
(open) & (\hat{I_B}, \hat{I_E}, \hat{H}) \models^{\mathrm{B}} \mathbf{open}^{\ell^t}\, n.P \text{ iff } (\hat{I_B}, \hat{I_E}, \hat{H}) \models^{\mathrm{B}} P \wedge \\
& \forall \ell^a, \ell^{a'} \in \mathbf{Lab}^a : \\
& \quad \text{case } ((\ell^a, \ell^t) \in \hat{I_E} \ \wedge\ (\ell^a, \ell^{a'}) \in \hat{I_E} \ \wedge\ (\ell^{a'}, n) \in \hat{H}) \ \Longrightarrow \\
& \qquad \text{if } (\ \ell^{a'} \in \mathbf{Lab}^a_B) \text{ then } \left\{(\ell^a, \ell^{a''}) \mid (\ell^{a'}, \ell^{a''}) \in \hat{I_B}\right\} \subseteq \hat{I_E} \wedge \\
& \qquad\quad \left\{(\ell, \ell') \mid (\ell, \ell') \in \hat{I_B} \ \wedge\ (\ell^{a'}, \ell'') \in \hat{I_B} \ \wedge\ path_B(\ell'', \ell)\right\} \subseteq \hat{I_E} \\
& \qquad \text{else } \left\{(\ell^a, \ell) \mid (\ell^{a'}, \ell) \in \hat{I_E}\right\} \subseteq \hat{I_E} \\
& \quad \text{case } ((\ell^a, \ell^t) \in \hat{I_B} \ \wedge\ (\ell^a, \ell^{a'}) \in \hat{I_B} \ \wedge\ (\ell^{a'}, n) \in \hat{H}) \\
& \qquad \Longrightarrow\ \left\{(\ell^a, \ell) \mid (\ell^{a'}, \ell) \in \hat{I_B}\right\} \subseteq \hat{I_B}
\end{array}
$$

Fig. 6. Specification of the refined control flow analysis.

A pictorial representation of the most interesting application of the *in*-rule (i.e., the last one: a boundary crossing) is provided by Fig. 7: the sets $\hat{I}_E$ and $\hat{I}_B$ before and after the move of ambient k into ambient n are represented by graphs (a) and (b), respectively. In particular, ambient k labeled

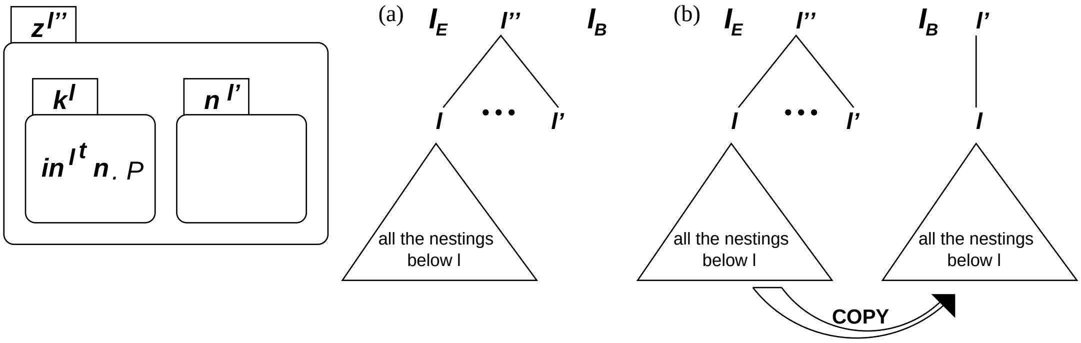


Fig. 7. The *in*-rule: boundary crossing.

as $\ell \notin \mathsf{Lab}^a_B$ is willing to enter ambient n labeled as $\ell' \in \mathsf{Lab}^a_B$. In this case, all the nestings below ℓ have to be copied in $\hat{I}_B$ since after the *in* move they become protected.

Observe that within the specification of the analysis (depicted in Fig. 6), some predicates are introduced that simplify the notation, namely

- $path_B(\ell^a,\ell) = \begin{cases} True & \text{if } \ell^a = \ell \vee \exists \ell_1,\ell_2,\ldots,\ell_n \notin \mathsf{Lab}^a_B\colon n \geqslant 0 \\ & (\ell^a,\ell_1),(\ell_1,\ell_2),\ldots,(\ell_n,\ell) \in \hat{I}_B \wedge \ell^a,\ell \notin \mathsf{Lab}^a_B, \\ False & \text{otherwise.} \end{cases}$
- $path_E(\ell^a,\ell) = \begin{cases} True & \text{if } \ell^a = \ell \vee \exists \ell_1,\ell_2,\ldots,\ell_n \notin \mathsf{Lab}^a_B\colon n \geqslant 0 \\ & (\ell^a,\ell_1),(\ell_1,\ell_2),\ldots,(\ell_n,\ell) \in \hat{I}_E \wedge \ell^a,\ell \notin \mathsf{Lab}^a_B, \\ False & \text{otherwise.} \end{cases}$

The following holds.

Theorem 4.3. *Let P and Q be two processes such that $\beta^{\mathrm{B}}_{env}(P) \sqsubseteq (\hat{I}_B,\hat{I}_E,\hat{H}) \wedge (\hat{I}_B,\hat{I}_E,\hat{H}) \models^{\mathrm{B}} P \wedge P \to Q$ then $\beta^{\mathrm{B}}_{env}(Q) \sqsubseteq (\hat{I}_B,\hat{I}_E,\hat{H}) \wedge (\hat{I}_B,\hat{I}_E,\hat{H}) \models^{\mathrm{B}} Q$.*

The proof of the Theorem can be obtained by showing that the new control flow analysis is induced from the analysis with occurrence counting of [13], and it can be found in Section 5.

The result of the analysis should be read, as expected, in terms of information flows. No leakage of secret data/ambients outside the boundary ambients is possible if in the analysis h (high-level datum) does not appear in any of the pairs belonging to $\hat{I}_E$.

Corollary 4.4. *Let P be a process and $h \in \mathsf{Lab}^a_H$ a high-level label. Let $\beta^{\mathrm{B}}(P) \sqsubseteq (\hat{I}_B,\hat{I}_E,\hat{H})$ and $(\hat{I}_B,\hat{I}_E,\hat{H}) \models^{\mathrm{B}} P$ and $\forall(\ell',\ell'') \in \hat{I}_E$, $\ell'' \neq h$. Then, P does not leak secret h.*

What about accuracy? The analysis just introduced is a refinement of the CFA in [12] and it properly deals with boundary nestings, in the spirit of Section 3. Referring again to example process

P_3, we may observe that our refined analysis properly captures the fact that the “filter” is entered by confidential data only once it inside of boundary “london”.

In addition, it definitely improves in accuracy with respect to the mentioned syntactic property introduced in [2]. Consider, for instance, the following example, where the process discussed in the previous sections is extended by allowing an *application* (say an applet) to be downloaded from the *web* within *london*; then, the application may open the ambient *send* and disappear.

$$P_4 = venice^{b_1}[send^{b_3}[\mathbf{out}^c venice.\mathbf{in}^c london \mid hdata^h[\mathbf{in}^{ch} filter]\ ]|$$

$$|download^{m'}[\mathbf{out}^{cm'} venice.\mathbf{in}^{cm'} web.\mathbf{in}^{cm'} london]\ ]|$$

$$|london^{b_2}[\mathbf{open}^c web.\mathbf{open}^c application]|$$

$$web^m[application^m[\mathbf{open}^{cm} send.filter^m[\ ]\ ] \mid \mathbf{open}^{cm} download].$$

Observe that in this case there is no information flow, as the application is not exporting any data out of the london boundary. In this case, our refined CFA yields to positive information (see the least solution reported below), whereas the syntactic property cannot be successfully applied. In fact, the (untrusted) application downloaded from the net is not a boundary, its *open* capability is labeled *BM* by the first rule, and thus the second rule cannot be satisfied since ambient *application* is not labeled as a boundary (i.e., $m \in \mathbf{Lab}^a_L$). The least solution of the analysis $S_4 = (\hat{I}_B, \hat{I}_E, Hc)$ is

$$\hat{I}_B = \{(b_1,b_3),(b_1,m'),(b_3,h),(b_3,c),(h,ch),(m',cm'),(b_2,b_3),(b_2,h),(b_2,m'),(b_2,b_2),$$
$$(b_2,m),(b_2,c),(b_2,cm'),(b_2,cm),(m,h),(m,m'),(m,b_2),$$
$$(m,m),(m,cm'),(m,cm)\},$$

$$\hat{I}_E = \{(env,b_1),(env,b_3),(env,m'),(env,b_2),(env,m),(m',cm'),(m,m'),(m,b_2),$$
$$(m,m),(m,cm'),(m,cm)\}$$

$$\hat{H} = \{(b_1,venice),(b_3,send),(b_2,london),(h,hdata),(m',download),$$
$$(m,web),(m,application),(m,filter)\}.$$

Observe that the result is also more accurate than the Hansen–Jensen–Nielsons’s CFA [12] which computes the following least solution:

$$\hat{I}_4 = \{(env,b_1),(env,b_3),(env,m'),(env,b_2),(env,m),(b_1,b_3),(b_1,m'),(b_3,h),(b_3,c),(h,ch),$$
$$(m',cm'),(b_2,b_3),(b_2,h),(b_2,m'),(b_2,b_2),(b_2,m),(b_2,c),(b_2,cm'),(b_2,cm),(m,h),$$
$$(m,m'),(m,b_2),(m,m),(m,cm'),(m,cm)\},$$

$$\hat{H}_4 = \{(b_1,venice),(b_3,send),(b_2,london),(h,hdata),(m',download),$$
$$(m,web),(m,application),(m,filter)\}.$$

Note that h appears inside an m ambient that at the beginning of the process is at the environmental level, but the analysis does not capture the fact that h enters m only after it has crossed the boundary and can never return back.

5. Correctness of the refined control flow analysis

In this section we prove the correctness of our refined CFA, as stated by Theorem 4.3. To do this, we strictly follow the approach of [13], i.e., we prove that the new control flow analysis is *induced* from the analysis with occurrence counting. As a consequence, the analysis is semantically correct: each reduction of the semantics is adequately mimicked in the analysis.

First, let us recall some auxiliary definitions in [13]:

- *Induced satisfaction relation.*

Let (α,γ) be a Galois connection of $\mathscr{A}$ in $\mathscr{A}'$, and let $\models: \mathscr{A} \times \textbf{Proc} \rightarrow \{\texttt{tt},\texttt{ff}\}$ and $\models'$: $\mathscr{A}' \times \textbf{Proc} \rightarrow \{\texttt{tt},\texttt{ff}\}$ be satisfaction relations.

The relation $\models$ is said to be `induced` from the relation $\models'$ when

$$\forall A \in \mathscr{A}, P \in \textbf{Proc}\colon \gamma(A) \models' P \quad\Leftrightarrow\quad A \models P.$$

- *Approximate satisfaction relation.*

Let (α,γ) be a Galois connection of $\mathscr{A}$ in $\mathscr{A}'$, and let $\models \ : \mathscr{A} \times \textbf{Proc} \rightarrow \{\texttt{tt},\texttt{ff}\}$ and $\models'$ $: \mathscr{A}' \times \textbf{Proc} \rightarrow \{\texttt{tt},\texttt{ff}\}$ be satisfaction relations.

The relation $\models$ is said to be `approximate` from the relation $\models'$ when

$$\forall A \in \mathscr{A}, P \in \textbf{Proc} : \gamma(A) \models' P \Longleftarrow A \models P.$$

- *Induced representation function.*

Let $\mathscr{A}'\mathscr{A}$ be a Galois connection, then a representation function $\beta : \textbf{Proc} \rightarrow \mathscr{A}$ is said to be `induced` from a representation function $\beta' : \textbf{Proc} \rightarrow \mathscr{A}'$ whenever:

$$\alpha \circ \beta' = \beta.$$

Proposition 5.1 (Preservation of "global" correctness Nielson et al. [13]). *Assume* $\models$, β, $\models'$ *and* β' *are given such that* $\models$ *approximates* $\models'$ *and* β *is induced from* β', *via the Galois connection* $\mathscr{A}'\mathscr{A}$. *Then*

$$\beta'(P) \sqsubseteq A' \wedge A' \models' P \wedge P \rightarrow^* Q \Rightarrow \beta'(Q) \sqsubseteq A'$$

implies

$$\beta'(P) \sqsubseteq A' \wedge A' \models' P \wedge P \rightarrow^* Q \Rightarrow \beta'(Q) \sqsubseteq A'$$

for all P, Q, A *and* A'.

The abstraction and concretization functions $\alpha^{\mathrm{B}}, \gamma^{\mathrm{B}}$ can be defined in terms of the occurrence counting domain, where a third component $\hat{A}$ maintains multiplicity information, as follows:

Let $\eta^B : \hat{I} \mapsto \hat{I}_B$ and $\eta^E : \hat{I} \mapsto \hat{I}_E$ be functions splitting $\hat{I}$ according to the boundary nestings, in [3]

$$\alpha^{\mathrm{B}}(\mathscr{C}) = \bigsqcup \{(\eta^B(\hat{I}), \eta^E(\hat{I}), \hat{H}) \mid (\hat{I}, \hat{H}, \hat{A}) \in \mathscr{C}\},$$

$$\gamma^{\mathrm{B}}(\hat{I}_B, \hat{I}_E, \hat{H}) = \{(\hat{I}', \hat{H}', \hat{A}') \mid (\eta^B(\hat{I}'), \eta^E(\hat{I}'), \hat{H}') \sqsubseteq (\hat{I}_B, \hat{I}_E, \hat{H}) \wedge (\hat{I}', \hat{H}', \hat{A}') \text{ is compatible}\}.$$

The two functions η^B and η^E are defined in terms of protected and unprotected path predicates as follows:

- $$p\_path(\ell^a) = \begin{cases} True & \text{iff } \exists \ell_0 \in \mathbf{Lab}_B^a \wedge \exists \ell_1, \ell_2, \ldots, \ell_n\colon n \geqslant 0 \\ & (\ell_0, \ell_1), (\ell_1, \ell_2), \ldots, (\ell_n, \ell^a) \in \hat{I}, \\ False & \text{otherwise.} \end{cases}$$
- $$u\_path(\ell^a) = \begin{cases} True & \text{iff } \exists \ell_1, \ell_2, \ldots, \ell_n\colon n \geqslant 0 \\ & (env, \ell_1), (\ell_1, \ell_2), \ldots, (\ell_n, \ell^a) \text{ s.t. } \forall k\ \ell_k \notin \mathbf{Lab}_B^a, \\ False & \text{otherwise.} \end{cases}$$

η^B: $\wp(\mathbf{Lab}^a \times (\mathbf{Lab}^a \cup \mathbf{Lab}^t)) \to \wp(\mathbf{Lab}^a \times (\mathbf{Lab}^a \cup \mathbf{Lab}^t))$
$\eta^B(\hat{I}) = \{(\ell^a, \ell) \mid p\_path(\ell^a) \vee \neg u\_path(\ell^a)\}$,
η^E: $\wp(\mathbf{Lab}^a \times (\mathbf{Lab}^a \cup \mathbf{Lab}^t)) \to \wp(\mathbf{Lab}^a \times (\mathbf{Lab}^a \cup \mathbf{Lab}^t))$
$\eta^E(\hat{I}) = \{(\ell^a, \ell) \mid u\_path(\ell^a)\}$.

The abstraction and concretization functions $\gamma^{\mathrm{B}}, \alpha^{\mathrm{B}}$ form a Galois connection of our abstract domain of triplets into **CountSet**; i.e.,

- both functions are monotone;
- $\mathscr{C} \subseteq \gamma^{\mathrm{B}}(\alpha^{\mathrm{B}}(\mathscr{C}))$ for any $\mathscr{C} \in$ **CountSet**;
- $\alpha^{\mathrm{B}}(\gamma^{\mathrm{B}}(\hat{I}_B, \hat{I}_E, \hat{H})) \sqsubseteq (\hat{I}_B, \hat{I}_E, \hat{H})$ for any triplet $(\hat{I}_B, \hat{I}_E, \hat{H})$.

The following proposition states that the refined control flow analysis is induced from the occurrence counting analysis.

Proposition 5.2. *If $P \in \mathbf{Proc}$ and $(\hat{I}_B, \hat{I}_E, \hat{H})$ is compatible then*

$$\gamma^{\mathrm{B}}(\hat{I}_B, \hat{I}_E, \hat{H}) \models^{\mathrm{OC}} P \quad \Leftrightarrow \quad (\hat{I}_B, \hat{I}_E, \hat{H}) \models^{\mathrm{B}} P$$

and

$$\alpha^{\mathrm{B}}(\beta^{\mathrm{OC}}(P)) = \beta^{\mathrm{B}}(P).$$

[3] According to [12], a triplet $(\hat{I}_B, \hat{I}_E, \hat{H})$ is *compatible* whenever the labels in $\hat{I}_E \cup \hat{I}_B$ are consistent with the mapping $\hat{H}$. More formally, if the following condition is satisfied: if $(\ell^a, \ell) \in \hat{I}_E \cup \hat{I}_B$ or $(\ell, \ell^a) \in \hat{I}_E \cup \hat{I}_B$ then there exists n such that $(\ell^a, n) \in (\hat{I}, \hat{H})$.

Proof. The proof follows the lines of the corresponding one in [13]. Namely, it suffices to prove

(i) $\gamma^{\mathrm{B}}(\hat{I}_B,\hat{I}_E,\hat{H}) \models^{\mathrm{OC}} P \Leftrightarrow (\hat{I}_B,\hat{I}_E,\hat{H}) \models^{\mathrm{B}} P$,
(ii) $\forall \ell \in \mathbf{Lab}^a$: $\alpha^{\mathrm{B}}(\{\eta_\ell^{\mathrm{OC}}(P)\}) = \beta_\ell^{\mathrm{B}}(P)$.

We just focus on proving point (i) by structural induction on process P (the proof of point (ii) is trivially obtained by rephrasing the corresponding proof in [13]). The base of the induction is case $P=0$, which is always verified. The interesting cases arise when capabilities *in*, *out* and *open* apply.

Case '*in*' (*Figs*. 8 *and* 9): Proof of "$\Longleftarrow$". Let us prove that

$$(\hat{I}_B,\hat{I}_E,\hat{H}) \models^{\mathrm{B}} in^{\ell^t} n.P \Rightarrow \gamma^{\mathrm{B}}(\hat{I}_B,\hat{I}_E,\hat{H}) \models^{\mathrm{OC}} in^{\ell^t} n.P.$$

From the specification of the analysis depicted in Fig. 9, it follows that $(\hat{I}_B,\hat{I}_E,\hat{H}) \models^{\mathrm{B}} P$. So, by induction hypothesis, we have $\gamma^{\mathrm{B}}(\hat{I}_B,\hat{I}_E,\hat{H}) \models^{\mathrm{OC}} P$.

Then, consider $(\hat{I}',\hat{H}',\hat{A}') \in \gamma^{\mathrm{B}}(\hat{I}_B,\hat{I}_E,\hat{H})$ and $\ell^a,\ell^{a'},\ell^{a''}$ such that

$$(\ell^a,\ell^t) \in \hat{I}' \wedge (\ell^{a''},\ell^a) \in \hat{I}' \wedge (\ell^{a''},\ell^{a'}) \in \hat{I}' \wedge (\ell^{a'},n) \in \hat{H}'.$$

By definition of γ^{B}, if $(\hat{I}',\hat{H}',\hat{A}') \in \gamma^{\mathrm{B}}(\hat{I}_B,\hat{I}_E,\hat{H})$, then $(\hat{I}',\hat{H}',\hat{A}')$ is compatible and $(\eta^B(\hat{I}'),\eta^E(\hat{I}'),\hat{H}') \sqsubseteq (\hat{I}_B,\hat{I}_E,\hat{H})$. It follows that:

$$\begin{aligned}&(\ell^a,\ell^t) \in \eta^B(\hat{I}') \cup \eta^E(\hat{I}') \wedge (\ell^{a''},\ell^a) \in \eta^B(\hat{I}') \cup \eta^E(\hat{I}') \wedge\\ &\quad (\ell^{a''},\ell^{a'}) \in \eta^B(\hat{I}') \cup \eta^E(\hat{I}') \wedge (\ell^{a'},n) \in \hat{H}.\end{aligned}$$

We have to consider the three cases in Fig. 9, the most informative one is case 3, when a boundary crossing occurs. In this case it also holds that either $(\ell^{a'},\ell^a) \in \eta^E(\hat{I}')$ or $(\ell^{a'},\ell^a) \in \eta^B(\hat{I}')$ and $\{(\ell,\ell') \in \eta^E(\hat{I}') \mid path_E(\ell^a,\ell)\} \subseteq \eta^B(\hat{I}')$.

Then, we may look at the occurrence counting analysis. Observe that all the four triplets of Fig. 8

$$\begin{aligned}&(\hat{I} \cup \{(\ell^{a'},\ell^a)\},\hat{H},\hat{A}),\\ &(\hat{I} \setminus \{(\ell^{a''},\ell^a)\} \cup \{(\ell^{a'},\ell^a)\},\hat{H},\hat{A}),\\ &(\hat{I} \setminus \{(\ell^a,\ell^t)\} \cup \{(\ell^{a'},\ell^a)\},\hat{H},\hat{A}),\\ &(\hat{I} \setminus \{(\ell^{a''},\ell^a),(\ell^a,\ell^t)\} \cup \{(\ell^{a'},\ell^a)\},\hat{H},\hat{A})\end{aligned}$$

are compatible, and only the pair $(\ell^{a'},\ell^a)$ is added to $\hat{I}$. This pair is already known to be either in $\hat{I}_E$ or in $\hat{I}_B$. Case 1 and case 2 of Fig. 9 are analogous.

Thus, it follows that all the triplets of Fig. 8 are elements of $\gamma^{\mathrm{B}}(\hat{I}_B,\hat{I}_E,\hat{H})$. This concludes the proof that $\gamma^{\mathrm{B}}(\hat{I}_B,\hat{I}_E,\hat{H}) \models^{\mathrm{OC}} in^{\ell^t} n.P$.

Case '*in*' (*Figs*. 8 *and* 9): Proof of "$\Rightarrow$". Let us now show that

$$\gamma^{\mathrm{B}}(\hat{I}_B,\hat{I}_E,\hat{H}) \models^{\mathrm{OC}} in^{\ell^t} n.P \Rightarrow (\hat{I}_B,\hat{I}_E,\hat{H}) \models^{\mathrm{B}} in^{\ell^t} n.P.$$

$$(in)\quad \mathcal{C} \models^{\mathrm{OC}} \mathbf{in}^{\ell^t}\, n.P \text{ iff } \mathcal{C} \models^{\mathrm{OC}} P \ \wedge\ \forall(\hat{I},\hat{H},\hat{A}) \in \mathcal{C}\ : \forall \ell^a, \ell^{a'}, \ell^{a''} :$$

$$\Big((\ell^a,\ell^t) \in \hat{I} \ \wedge\ (\ell^{a''},\ell^a) \in \hat{I} \ \wedge\ (\ell^{a''},\ell^{a'}) \in \hat{I} \ \wedge\ (\ell^{a'},n) \in \hat{H}\Big) \Rightarrow$$

$$\left(\begin{array}{l}
\text{case} \hat{A}(\ell^a) \text{of} \\[1em]
1 : (\hat{I} \setminus \{(\ell^{a''},\ell^a)\} \cup \{(\ell^{a'},\ell^a)\}, \hat{H}, \hat{A}) \in \mathcal{C} \\[1em]
\quad (\hat{I} \setminus \{(\ell^{a''},\ell^a),(\ell^a,\ell^t)\} \cup \{(\ell^{a'},\ell^a)\}, \hat{H}, \hat{A}) \in \mathcal{C} \\[1em]
\omega : (\hat{I} \cup \{(\ell^{a'},\ell^a)\}, \hat{H}, \hat{A}) \in \mathcal{C} \\[1em]
\quad (\hat{I} \setminus \{(\ell^{a''},\ell^a)\} \cup \{(\ell^{a'},\ell^a)\}, \hat{H}, \hat{A}) \in \mathcal{C} \\[1em]
\quad (\hat{I} \setminus \{(\ell^a,\ell^t)\} \cup \{(\ell^{a'},\ell^a)\}, \hat{H}, \hat{A}) \in \mathcal{C} \\[1em]
\quad (\hat{I} \setminus \{(\ell^{a''},\ell^a),(\ell^a,\ell^t)\} \cup \{(\ell^{a'},\ell^a)\}, \hat{H}, \hat{A}) \in \mathcal{C}
\end{array}\right)$$

Fig. 8. Specification of the analysis with occurrence counting (*in*-capabilities).

From the specification of the occurrence counting analysis (see Fig. 8), it follows that $\gamma^{\mathrm{B}}(\hat{I}_B,\hat{I}_E,\hat{H}) \models^{\mathrm{OC}} P$. By induction hypothesis, we have $(\hat{I}_B,\hat{I}_E,\hat{H}) \models^{\mathrm{B}} P$.

Then, consider $\ell^a, \ell^{a'}, \ell^{a''}$ such that

$$(\ell^a,\ell^t) \in \eta^B(\hat{I}') \cup \eta^E(\hat{I}') \wedge (\ell^{a''},\ell^a) \in \eta^B(\hat{I}') \cup \eta^E(\hat{I}') \wedge$$
$$(\ell^{a''},\ell^{a'}) \in \eta^B(\hat{I}') \cup \eta^E(\hat{I}') \wedge (\ell^{a'},n) \in \hat{H}.$$

We know from [13] (Section 4.2) that

$$(\hat{I},\hat{H},[\{\ell^a \mid (\ell^a n) \in \hat{H}\} \mapsto \omega]) \in \gamma^{\mathrm{B}}(\hat{I}_B,\hat{I}_E,\hat{H})$$

therefore

$$(\hat{I} \cup \{(\ell^{a'},\ell^a)\},\hat{H},[\{\ell^a \mid (\ell^a n) \in \hat{H}\} \mapsto \omega]) \in \gamma^{\mathrm{B}}(\hat{I}_B,\hat{I}_E,\hat{H}).$$

Given the definition of γ^{B} it follows that $(\ell^{a'},\ell^a) \in \hat{I}_E \cup \hat{I}_B$, and this concludes the proof that $(\hat{I}_B,\hat{I}_E,\hat{H}) \models^{\mathrm{B}} in^{\ell^t} n.P$.

The proof for *out*-capabilities is analogous. Then, to conclude the whole proof we have to consider the *open* capability. In this case, we need to consider the multiplicity of the ambients dealt with. In particular,

Case '*open*' (*Figs.* 10 *and* 11): Proof of "$\Longleftarrow$". Let us prove that

$$(\hat{I}_B,\hat{I}_E,\hat{H}) \models^{\mathrm{B}} open^{\ell^t} n.P \ \Rightarrow \gamma^{\mathrm{B}}(\hat{I}_B,\hat{I}_E,\hat{H}) \models^{\mathrm{OC}} open^{\ell^t} n.P.$$

(in) $(\hat{I_B}, \hat{I_E}, \hat{H}) \models^{\mathrm{B}} \mathbf{in}^{\ell^t}\, n.P$ iff $(\hat{I_B}, \hat{I_E}, \hat{H}) \models^{\mathrm{B}} P \wedge$

$\forall \ell^a, \ell^{a'}, \ell^{a''} \in \mathsf{Lab}^a$:

case $((\ell^a, \ell^t) \in \hat{I_B} \wedge (\ell^{a''}, \ell^a) \in \hat{I_B} \wedge (\ell^{a''}, \ell^{a'}) \in \hat{I_B} \wedge (\ell^{a'}, n) \in \hat{H})$

$\Longrightarrow (\ell^{a'}, \ell^a) \in \hat{I_B}$

case $((\ell^a, \ell^t) \in \hat{I_B} \wedge (\ell^{a''}, \ell^a) \in \hat{I_E} \wedge (\ell^{a''}, \ell^{a'}) \in \hat{I_E} \wedge \ell^a \in \mathsf{Lab}^a_B \wedge (\ell^{a'}, n) \in \hat{H}) \Longrightarrow$

if ($\ell^{a'} \in \mathsf{Lab}^a_B$) then $(\ell^{a'}, \ell^a) \in \hat{I_B}$

else $(\ell^{a'}, \ell^a) \in \hat{I_E}$

case $((\ell^a, \ell^t) \in \hat{I_E} \wedge (\ell^{a''}, \ell^a) \in \hat{I_E} \wedge (\ell^{a''}, \ell^{a'}) \in \hat{I_E} \wedge (\ell^{a'}, n) \in \hat{H}) \Longrightarrow$

if ($\ell^{a'} \in \mathsf{Lab}^a_B$)

then $(\ell^{a'}, \ell^a) \in \hat{I_B} \wedge \left\{(\ell, \ell') \in \hat{I_E} \mid path_E(\ell^a, \ell)\right\} \subseteq \hat{I_B}$

else $(\ell^{a'}, \ell^a) \in \hat{I_E}$

Fig. 9. Specification of the refined control flow analysis (*in*-capabilities).

From the specification of the analysis depicted in Fig. 11, it follows that $(\hat{I}_B, \hat{I}_E, \hat{H}) \models^{\mathrm{B}} P$. So, by induction hypothesis we have $\gamma^{\mathrm{B}}(\hat{I}_B, \hat{I}_E, \hat{H}) \models^{\mathrm{OC}} P$.

Then, consider $(\hat{I}', \hat{H}', \hat{A}') \in \gamma^{\mathrm{B}}(\hat{I}_B, \hat{I}_E, \hat{H})$ and $\ell^a, \ell^{a'}$ such that

$$(\ell^a, \ell^t) \in \hat{I}' \wedge (\ell^a, \ell^{a'}) \in \hat{I}' \wedge (\ell^{a'}, n) \in \hat{H}'.$$

By definition of γ^{B}, if $(\hat{I}', \hat{H}', \hat{A}') \in \gamma^{\mathrm{B}}(\hat{I}_B, \hat{I}_E, \hat{H})$, then $(\hat{I}', \hat{H}', \hat{A}')$ is compatible and $(\eta^B(\hat{I}'), \eta^E(\hat{I}'), \hat{H}') \sqsubseteq (\hat{I}_B, \hat{I}_E, \hat{H})$. It follows that:

$$(\ell^a, \ell^t) \in \eta^B(\hat{I}') \cup \eta^E(\hat{I}') \wedge (\ell^a, \ell^{a'}) \in \eta^B(\hat{I}') \cup \eta^E(\hat{I}') \wedge (\ell^{a'}, n) \in \hat{H}.$$

We have to consider the two cases depicted in Fig. 11. The most interesting one is case 1, when a boundary is dissolved. In this case, it holds that either $\{(\ell^a, \ell^{a''}) \mid (\ell^{a'}, \ell^{a''}) \in \hat{I}_B\} \subseteq \eta^E(\hat{I}')$ and $\{(\ell, \ell') \mid (\ell, \ell') \in \hat{I}_B \wedge (\ell^{a'}, \ell'') \in \hat{I}_B \wedge path_B(\ell'', \ell)\} \subseteq \eta^E(\hat{I}')$ or $\{(\ell^a, \ell) \mid (\ell^{a'}, \ell) \in \hat{I}_E\} \subseteq \eta^E(\hat{I}')$.

By looking at the occurrence counting analysis, we observe that all the triplets of Fig. 10 are compatible and at most add the pairs

$$\{(\ell^a, \ell) \mid (\ell^{a'}, \ell) \in Z\} \subseteq \{(\ell^a, \ell) \mid (\ell^{a'}, \ell) \in \hat{I}'\} \subseteq \{(\ell^a, \ell) \mid (\ell^{a'}, \ell) \in \hat{I}_E \cup \hat{I}_B\}.$$

which are already known to be either in $\hat{I}_E$ or in $\hat{I}_B$. Case 2 of Fig. 11 is analogous.

$$(open)\quad \mathcal{C} \models^{\rm OC} \mathbf{open}^{\ell^t}\, n.P \text{ iff } \mathcal{C} \models^{\rm OC} P \;\wedge\; \forall(\hat{I},\hat{H},\hat{A}) \in \mathcal{C} \;: \forall \ell^a, \ell^{a'} :$$

$$\Big((\ell^a,\ell^t) \in \hat{I} \;\wedge\; (\ell^a,\ell^{a'}) \in \hat{I} \;\wedge\; (\ell^{a'},n) \in \hat{H}\Big) \Rightarrow$$

$$\left(\begin{array}{l}
\text{case}\hat{A}(\ell^{a'})\text{of} \\
1 : \forall X \subseteq \{(\ell^a,\ell^t)\} : \\
\quad (\hat{I} \setminus (\{(\ell^{a'},\ell) \in \hat{I} |\; \ell \in \mathbf{Lab}\} \cup \{(\ell^a,\ell^{a'})\} \cup X) \cup \{(\ell^a,\ell) |\; (\ell^{a'},\ell) \in \hat{I}\}, \\
\quad \hat{H} \setminus \{(\ell^{a'},m) |\; m \in \mathbf{SNam}\}, \hat{A} \setminus \{\ell^{a'}\}) \in C \\
\omega : \forall Z \subseteq \{(\ell^{a'},\ell) \in \hat{I} |\; \ell \in \mathbf{Lab}\} : \forall X \subseteq \{(\ell^a,\ell^t)\} : \forall \mathsf{c} \in \{1,\omega\} : \\
\quad \forall Y \subseteq \text{s.t. } Y \supseteq Z \cap \{(\ell^{a'},\ell) \in \hat{I} |\; \hat{A}(\ell) = 1\} : \\
\quad \text{case } \mathsf{c} \text{ of} \\
\quad \omega : \forall V \subseteq \{(\ell^{a'},\lfloor n \rfloor)\} \text{ s.t. } \exists m \in \mathbf{SNam} : (\ell^{a'},m) \in \hat{H} \setminus V : \\
\quad \forall W \subseteq \{(\ell^a,\ell^{a'})\} \text{ s.t. } \exists \ell : (\ell,\ell^{a'}) \in ((\hat{I} \setminus (Y \cup X)) \cup \{(\ell^a,\ell) |\; (\ell^{a'},\ell) \in Z\}) \setminus W : \\
\quad (((\hat{I} \setminus (Y \cup X)) \cup \{(\ell^a,\ell) |\; (\ell^{a'},\ell) \in Z\}) \setminus W, \hat{H} \setminus V, \hat{A}[\ell^{a'} \mapsto \mathsf{c}]) \in C \\
\quad 1 : \forall V \subseteq \{(\ell^{a'},m) |\; (\ell^{a'},m) \in \hat{H}\} \\
\quad \forall W \subseteq \{(\ell,\ell^{a'}) |\; (\ell,\ell^{a'}) \in ((\hat{I} \setminus (Y \cup X)) \cup \{(\ell^a,\ell) |\; (\ell^{a'},\ell) \in Z\})\} : \\
\quad (((\hat{I} \setminus (Y \cup X)) \cup \{(\ell^a,\ell) |\; (\ell^{a'},\ell) \in Z\}) \setminus W, \hat{H} \setminus V, \hat{A}[\ell^{a'} \mapsto \mathsf{c}]) \in C
\end{array}\right)$$

Fig. 10. Specification of the analysis with occurrence counting (*open*-capabilities).

Thus, it follows that all the triplets above are elements of $\gamma^{\rm B}(\hat{I}_B,\hat{I}_E,\hat{H})$. This concludes the proof that $\gamma^{\rm B}(\hat{I}_B,\hat{I}_E,\hat{H}) \models^{\rm OC} open^{\ell^t} n.P$.

Case '*open*' (Figs. 10 and 11): Proof of "⇒". Finally, let us show now that

$$\gamma^{\rm B}(\hat{I}_B,\hat{I}_E,\hat{H}) \models^{\rm OC} open^{\ell^t} n.P \;\Rightarrow (\hat{I}_B,\hat{I}_E,\hat{H}) \models^{\rm B} open^{\ell^t} n.P.$$

From the specification of the occurrence counting analysis (Fig. 10), we get that $\gamma^{\rm B}(\hat{I}_B,\hat{I}_E,\hat{H}) \models^{\rm OC} P$. By induction hypothesis, $(\hat{I}_B,\hat{I}_E,\hat{H}) \models^{\rm B} P$.

Then, consider $(\hat{I}',\hat{H}',\hat{A}') \in \gamma^{\rm B}(\hat{I}_B,\hat{I}_E,\hat{H})$ and $\ell^a, \ell^{a'}$ such that

$$(\ell^a,\ell^t) \in \hat{I} \wedge (\ell^a,\ell^{a'}) \in \hat{I} \wedge (\ell^{a'},n) \in \hat{H}.$$

$(open)$ $(\hat{I_B}, \hat{I_E}, \hat{H}) \models^{\mathrm{B}} \mathbf{open}^{\ell^t}\, n.P$ iff $(\hat{I_B}, \hat{I_E}, \hat{H}) \models^{\mathrm{B}} P \wedge$

$\forall \ell^a, \ell^{a'} \in \mathsf{Lab}^a$:

case 1: $((\ell^a, \ell^t) \in \hat{I_E} \ \wedge \ (\ell^a, \ell^{a'}) \in \hat{I_E} \ \wedge \ (\ell^{a'}, n) \in \hat{H}) \implies$

if ($\ell^{a'} \in \mathsf{Lab}^a_B$) then $\left\{(\ell^a, \ell^{a''}) \mid (\ell^{a'}, \ell^{a''}) \in \hat{I_B}\right\} \subseteq \hat{I_E} \wedge$

$\left\{(\ell, \ell') \mid (\ell, \ell') \in \hat{I_B} \ \wedge \ (\ell^{a'}, \ell'') \in \hat{I_B} \ \wedge \ path_B(\ell'', \ell)\right\} \subseteq \hat{I_E}$

else $\left\{(\ell^a, \ell) \mid (\ell^{a'}, \ell) \in \hat{I_E}\right\} \subseteq \hat{I_E}$

case 2 $((\ell^a, \ell^t) \in \hat{I_B} \ \wedge \ (\ell^a, \ell^{a'}) \in \hat{I_B} \ \wedge \ (\ell^{a'}, n) \in \hat{H})$

$\implies \left\{(\ell^a, \ell) \mid (\ell^{a'}, \ell) \in \hat{I_B}\right\} \subseteq \hat{I_B}$

Fig. 11. Specification of the refined control flow analysis (*open*-capabilities).

We know from [13] that

$$(\hat{I}, \hat{H}, [\{\ell^a \mid (\ell^a, n) \in \hat{H}\} \mapsto \omega]) \in \gamma^{\mathrm{B}}(\hat{I}_B, \hat{I}_E, \hat{H}).$$

For this choice of element in $\gamma^{\mathrm{B}}(\hat{I}_B, \hat{I}_E, \hat{H})$, in the specification of the analysis depicted in Fig. 10 we take

$$Z = \{(\ell^{a'} \in \hat{I}) \mid \ell \in \mathsf{Lab})\},$$

$$Y = \emptyset,$$

$$X = \emptyset,$$

$$c = \omega,$$

$$V = \emptyset,$$

$$W = \emptyset.$$

Then, it follows that:

$$(((\hat{I} \setminus (Y \cup X)) \cup \{(\ell^a, \ell) \mid (\ell^{a'}, \ell) \in Z\}) \setminus W, \hat{H} \setminus V, \hat{A}[\ell^{a'} \mapsto C]) \in \gamma^{\mathrm{B}}(\hat{I}_B, \hat{I}_E, \hat{H}).$$

Given the definition of γ^{B}, it follows that:

$$\{(\ell^a, \ell) \mid (\ell^{a'}, \ell) \in \hat{I}\} = \{(\ell^a, \ell) \mid (\ell^{a'}, \ell) \in Z\} \subseteq \hat{I}$$

and this concludes the proof that $(\hat{I}_B, \hat{I}_E, \hat{H}) \models^{\mathrm{B}} open^{\ell^t} n.P$. $\square$

Proposition 5.3 (Theorem 4.3). *Let P and Q be two processes such that* $\beta^{\mathrm{B}}_{env}(P) \sqsubseteq (\hat{I}_B, \hat{I}_E, \hat{H}) \wedge (\hat{I}_B, \hat{I}_E, \hat{H}) \models^{\mathrm{B}} (P) \wedge P \to Q$ *then* $\beta^{\mathrm{B}}_{env}(Q) \sqsubseteq (\hat{I}_B, \hat{I}_E, \hat{H}) \wedge (\hat{I}_B, \hat{I}_E, \hat{H}) \models^{\mathrm{B}} Q$.

Proof. It follows immediately from Propositions 5.1 and 5.2. □

6. Related works and conclusions

The main novelty of the approach presented in this paper is that we model multilevel information flow within a "pure" mobile ambient setting based on the seminal ideas presented in [12,14] on ambient nesting analysis. We do not consider either channels or recently proposed restrictions of mobile ambients designed for security issues. There have been hardly any research efforts in this general direction. In fact, the most interesting recent contributions in the area mainly focused on either extending the ambient calculus thus enhancing its expressive power, or on building suitable type systems to verify security properties.

Among the type systems approaches, it is worth to mention [15], where the authors introduce a new type system for tracking the behavior of mobile computations. They introduce the concept of name group, which represents a collection of ambient names. Using groups, the type system can impose to an ambient behavioral constraints on the set of ambients it may cross and the set of ambients it may open. It has the effect of statically preventing certain communications through a mandatory access control policy, and can block accidental or malicious leakage of secrets.

Dezani and Salvo [16] extend the work of Cardelli et al. just mentioned, with a type system that also expresses security levels associated with ambients and provide further control over ambient movement and opening.

Among the language extensions, valuable proposals are described in [17,10,18,19,11]. *Safe ambients* is a modification of mobile ambients, where a movement or an ambient dissolution can take place only when the affected ambient agrees, offering the corresponding coaction. *Boxed ambients* [10] is another variant of the ambient calculus with a completely different model of communication, which results from dropping the *open* capability. In their paper, Bugliesi et al. define also a type system that provides an effective mechanism for resource protection and access control.

Few efforts have been put on the analysis approach to security issues in mobile ambients. In [20], Degano et al. present a control flow analysis that mainly focuses on access control. The analysis relies on the use of coactions as a filter to control access to resources.

We are currently extending our analysis to deal with coactions in order to evaluate benefits and disadvantages of the different formalisms of "information leakage" and of the corresponding analyses, and we are currently investigating how to extend our analysis to communication channels as well. Finally, the way we propose to express a multilevel information flow policy through the notion of boundary ambient, in our opinion, may also lead to interesting applications to model cryptographic protocols.

Appendix A. Operational semantics of mobile ambients

For the sake of completeness, we report the operational semantics of mobile ambients as described in [3].

A.1. Structural congruence

$$
\begin{array}{ll}
P \equiv P, & P \mid Q \equiv Q \mid P, \\
P \equiv Q \Rightarrow Q \equiv P, & (P \mid Q) \mid R \equiv P \mid (Q \mid R), \\
P \equiv Q, Q \equiv R \Rightarrow P \equiv R, & !P \equiv P \mid !P, \\
P \equiv Q \Rightarrow (\nu n)P \equiv (\nu n)Q, & (\nu n)(\nu m)P \equiv (\nu m)(\nu n)P, \\
P \equiv Q \Rightarrow P \mid R \equiv Q \mid R, & (\nu n)P \mid Q \equiv P \mid (\nu n)Q \text{ if } n \notin fn(P), \\
P \equiv Q \Rightarrow !P \equiv !Q, & (\nu n)m^{\ell^a}[P] \equiv m^{\ell^a}[(\nu n)P] \text{ if } n \neq m, \\
P \equiv Q \Rightarrow n^{\ell}[P] \equiv n^{\ell}[Q], & P \mid 0 \equiv P, \\
P \equiv Q \Rightarrow M.P \equiv M.Q, & (\nu n)0 \equiv 0, \\
 & !0 \equiv 0.
\end{array}
$$

A.2. Reduction rules

$$
\begin{array}{l}
n^{\ell^a}[\mathbf{in}^c m.P \mid Q] \mid m^{\ell}[R] \to m^{\ell}[n^{\ell^a}[P \mid Q] \mid R], \\
m^{\ell^a}[n^{\ell}[\mathbf{out}^c m.P \mid Q] \mid R] \to n^{\ell}[P \mid Q] | m^{\ell^a}[R], \\
\mathbf{open}^c n.P \mid n^{\ell^a}[Q] \to P \mid Q, \\
P' \equiv P,\ P \to Q,\ Q \equiv Q' \Rightarrow P' \to Q', \\
P \to Q \Rightarrow (\nu n)P \to (\nu n)Q, \\
P \to Q \Rightarrow n^{\ell}[P] \to n^{\ell}[Q], \\
P \to Q \Rightarrow P \mid R \to Q \mid R.
\end{array}
$$

References

[1] Braghin C, Cortesi A, Focardi R. Control flow analysis of mobile ambients with security boundaries. In: Jacobs B, Rensink A, editors. Proceedings of Fifth IFIP International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS '02). Dordrecht: Kluwer Academic Publisher, 2002. p. 197–212.

[2] Cortesi A, Focardi R. Information flow security in mobile ambients. In: Proceedings of International Workshop on Concurrency and Coordination (ConCoord '01), Electronic Notes on Theoretical Computer Science, ENTCS, vol. 54. Amsterdam: Elsevier, July 6–8, 2001.

[3] Cardelli L, Gordon AD. Mobile ambients. In: Nivat M, editor. Proceedings of Foundations of Software Science and Computation Structures (FoSSaCS), Lecture Notes in Computer Science, vol. 1378, Berlin, Germany, Springer, March 1998, p. 140–55.

[4] Bodei C, Degano P, Nielson F, Nielson HR. Static analysis of processes for No read-up and No write-down. In: Proceedings of FoSSaCS '99, Lecture Notes in Computer Science, No. 1578. Berlin: Springer, 1999. p. 120–34.

[5] Focardi R, Gorrieri G, Martinelli F. Information flow analysis in a discrete-time process algebra. In: Proceedings of the 13th Computer Security Foundations Workshop (CSFW). Silver Spring, MD: IEEE Computer Society Press, 2000. p. 170–84.
[6] Focardi R, Gorrieri R. A classification of security properties for process algebras. Journal of Computer Security 1995;3(1):5–33.
[7] Focardi R, Gorrieri R. The compositional security checker: a tool for the verification of information flow security properties IEEE Transactions on Software Engineering 1997;23(9):550–71.
[8] Hennessy M, Riely J. Information flow vs. resource access in the asynchronous pi-calculus. In: Montanari U, Rolim J, Welzl E, editors. Proceedings of the International Colloquium on Automata Languages and Programming, Lecture Notes in Computer Science, vol. 1853. Berlin, Geneva: Springer, August 2000, p. 415–27.
[9] Smith G, Volpano D. Secure information flow in a multi-threaded imperative language. In: Conference Record of POPL 98: The 25TH ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Diego, California, New York, NY, 1998. p. 355–64.
[10] Bugliesi M, Castagna G, Crafa S. Boxed ambients. In: Proceedings of the Fourth International Conference on Theoretical Aspects of Computer Science (TACS '01), Lecture Notes in Computer Science, vol. 2215. Berlin: Springer, 2000. p. 38–63.
[11] Levi F, Sangiorgi D. Controlling interference in ambients. In: Proceedings of the 28th ACM Symposium on Principles of Programming Languages (POPL '01), 2000. p. 352–64.
[12] Hansen RR, Jensen JG, Nielson F, Nielson HR. Abstract interpretation of mobile ambients. In: Cortesi A, File' G, editors. Proceedings of Static Analysis Symposium (SAS '99). Lecture Notes in Computer Science, No. 1694. Berlin: Springer, 1999. p. 134–48.
[13] Nielson F, Hansen RR, Nielson HR. Abstract interpretation of mobile ambients. In: Cortesi A, File' G, editors. Science of computer programming (Special Issue on Static Analysis), to appear.
[14] Bodei C, Degano P, Nielson F, Nielson HR. Control flow analysis for the π-calculus. In: Proceedings of CONCUR '98, Lecture Notes in Computer Science, No. 1466. Berlin: Springer, 1998. p. 84–98.
[15] Cardelli L, Ghelli G, Gordon AD. Ambient groups and mobility types. In: van Leeuwen J, Watanabe O, Hagiya M, Mosses PD, Ito T, editors. Theoretical computer science: exploring new frontiers of theoretical informatics. Proceedings of the International IFIP Conference TCS 2000 (Sendai, Japan), Lecture Notes in Computer Science, vol. 1872. New York, Berlin: IFIP, Springer, August 2000. p. 333–47.
[16] Dezani-Ciancaglini M, Salvo I. Security types for mobile safe ambients. In: He J, Sato M, editors. Proceedings of Advances in Computing Science—ASIAN '00 (Sixth Asian Computing Science Conference, Penang, Malaysia), Lecture Notes in Computer Science, vol. 1961, Berlin, Springer, December 2000, p. 215–36.
[17] Degano P, Levi F, Bodei C. Safe ambients: control flow analysis and security. In: He J, Sato M, editors. Proceedings of Advances in Computing Science—ASIAN '00 (Sixth Asian Computing Science Conference, Penang, Malaysia), Lecture Notes in Computer Science, vol. 1961, Berlin, Springer, December 2000, p. 199–214.
[18] Bugliesi M, Castagna G. Secure safe ambients. In: Proceedings of 28th ACM Symposium on Principles of Programming Languages (POPL '01). New York: ACM Press, 2001. p. 222–35.
[19] Bugliesi M, Castagna G. Behavioural typing of safe ambients. Journal of Computer Security, this issue.
[20] Levi F, Bodei C. Security analysis for mobile ambients. In: Electronic Proceedings of IFIP WG 1.7 Workshop on Issues on the Theory of Security (WITS '00), Geneve, 2000.

Chiara Braghin received her Laurea degree in Computer Science from Ca' Foscari University of Venice, Italy, in 2001. She is currently a Ph.D. Student at the same university. Her research activities include security, process algebras and static analysis techniques.

Agostino Cortesi received the Laurea degree in Mathematics in 1986, and the Ph.D. in Applied Mathematics and Computer Science in 1992, both from the University of Padova, Italy. Then, he spent 1 year in the Computer Science Department of Brown University, USA, as post doc visiting scientist. In 1994, he joined the Computer Science Department of Ca' Foscari University, Italy. Since 1998, he has been Associate Professor, holding the chairs on computer programming, and on software engineering, and on program analysis and verification. His current research interests are related to programming languages and static analysis techniques, in particular to the study of domains in abstract interpretation.

Riccardo Focardi received the Laurea degree in Computer Science in 1993 and the Ph.D. degree in Computer Science in 1998, both from the University of Bologna, Italy. Since October 1996, he has been an assistant professor at the University of Venice, Italy. His research interests are related to security, in particular to the automatic inference and verification of systems and protocols security properties. He is also interested in the semantics of concurrent languages.