Security for Mobility

Hanne Riis Nielson Flemming Nielson Mikael Buchholtz

Informatics and Mathematical Modelling, Technical University of Denmark,
Richard Petersens Plads, Building 321, DK-2800 Lyngby, Denmark.
E-mail: {riis,nielson,mib}@imm.dtu.dk

Abstract. We show how to use static analysis to provide information about security issues related to mobility. First the syntax and semantics of Mobile Ambients is reviewed and we show how to obtain a so-called 0CFA analysis that can be implemented in polynomial time. Next we consider discretionary access control where we devise Discretionary Ambients, based on Safe Ambients, and we adapt the semantics and 0CFA analysis; to strengthen the analysis we incorporate context-sensitivity to obtain a 1CFA analysis. This paves the way for dealing with mandatory access control where we express both a Bell-LaPadula model for confidentiality as well as a Biba model for integrity. Finally, we use Boxed Ambients as a means for expressing cryptographic key exchange protocols and we adapt the operational semantics and the 0CFA analysis.

1 Introduction

Mobile Ambients (see Section 2) were introduced by Cardelli and Gordon [13, 16] as a formalism for reasoning about mobility. Ambients present a high-level view of mobile computation and give rise to a high-level treatment of the related security issues.

An ambient is a named bounded place and the boundary determines what is outside and what is inside. Ambients can be nested inside each other and thereby form a tree structure. Mobility is then represented as navigation inside this hierarchy. Each ambient contains a number of multi-threaded running processes; the top-level processes of each ambient have direct control over it and can instruct it to move and thereby change its future behaviour. The ambient names are unforgeable and are essential for controlling access to the ambients. As in [12] we shall impose a simple type structure by assigning groups to ambients.

The basic calculus has three so-called subjective mobility capabilities: an enclosing ambient can be instructed to move into a sibling ambient, it can be instructed to move out of its enclosing ambient, and a sibling ambient can be dissolved. The literature contains a number of extensions to the basic calculus: so-called objective moves, various forms of communication and primitives for access control etc; we shall begin by considering the basic calculus in Section 2, then add access control features in Sections 3 and 4, and finally revert to the basic calculus in order to add communication primitives in Section 5.

The operational semantics is a standard reduction semantics with a structural congruence relation. The static analysis is modelled on the simple 0CFA analysis originally developed for functional programs. In the case of Mobile Ambients the control structure is expressed by the hierarchical structure of ambients (with separate components taking care of the communication, if present). Hence we aim at modelling the *father-son* relationship between the nodes of the tree structure [31, 30].

The precision of the 0CFA analysis is roughly comparable to that of early type systems for Mobile Ambients [11, 14] and may be used for validating security properties related to *crossing control* and *opening control* [12]. In the spirit of type systems the main semantic result showing the correctness of the 0CFA analysis is a subject-reduction result expressing that the analysis information remains valid during execution.

The efficiency of the analysis is good and the worst-case complexity is cubic [36]. In practical terms we find it convenient to translate the analysis into a fragment of first order logic known as Alternation-free Least Fixed Point Logic (ALFP) and implemented by our Succinct Solver [35].

Discretionary access control (see Section 3) imposes conditions on when an ambient can perform a given mobility primitive on another ambient. As an example, an ambient (the *subject*) may move into another ambient (the *object*) by executing a suitable capability (an *access operation*). In the *Safe Ambients* of Levi and Sangiorgi [24] there is a simple notion of access control; here the object must agree to being entered and this is expressed by requiring the object to execute the corresponding co-capability (an *access right*).

This rudimentary kind of access control does not fully model the usual notion of access control where an *access control matrix* lists the set of capabilities that each subject may perform on each object. (In the classical setting [22], the subjects correspond to programs, the objects correspond to files, and the access operations could be the read, write, and execute permissions of UNIX.) We overcome this shortcoming by designing the *Discretionary Ambients* where co-capabilities not only indicate the access rights but also the subject that is allowed to perform it.

We then adapt the semantics to incorporate the necessary checks and hence to block execution whenever an inadmissible access operation is performed. Similarly we adapt the analysis and later strengthen it using context-sensitivity; this is a standard technique from data flow and control flow analysis that can be used to improve the precision of a simple 0CFA analysis in order to obtain a so-called 1CFA analysis [29]. As mentioned above the 0CFA analysis approximates the hierarchical structure of the ambients by a binary *father-son* relationship. Context-sensitivity then is based on the observation that more precise results are likely to be obtained using a ternary *grandfather-father-son* relationship between ambients. This 1CFA analysis still has reasonable complexity and we report on practical experiments confirming that the use of ternary relations strikes a useful balance between precision and efficiency. (A considerably more precise and costly analysis is presented in [37, 38].)

Mandatory access control (see Section 4) imposes confidentiality and/or integrity by combining the access control matrices with additional information [22]. The Bell-LaPadula model assigns security levels to objects and subjects and imposes *confidentiality* by preventing information from flowing downwards from a high security level to a low security level. The Biba model assigns integrity levels to objects and subjects and then imposes *integrity* by preventing trusted high-level entities from being corrupted by dubious low-level entities — thus information is prevented from flowing upwards.

These security models were originally developed in a setting much more static than the one of Discretionary Ambients. For comparison, an ambient may be viewed as a system with a distributed access control matrix that dynamically evolves and that is concerned with multiplicities whereas classically the access control matrix is partly centralised and static. In this paper we show how the security policies may be re-formulated in the dynamic and mobile setting of the Discretionary Ambients.

The formal development amounts to adapting the semantics so as to incorporate reference monitors that block execution whenever an inadmissible access operation is performed (according to the mandatory access control policy considered). The analysis is extended to perform tests comparable to those of the reference monitors, and as an extension of the subject reduction theorem we show that if all static tests are satisfied then the reference monitor can be dispensed with.

Cryptographic protocols (see Section 5) are most naturally expressed using communication. The full calculus of Mobile Ambients includes a notion of local communication where there is a communication box inside each ambient; this naturally leads to dealing with asynchronous communication. For some purposes it is more convenient to allow communication between adjacent layers of ambients and this motivated the design of Boxed Ambients [9, 8]. Here an ambient can directly access not only its local communication box but also the communication box of its parent (but not grandparents) as well as its children (but not grandchildren). We show that perfect symmetric cryptography as well as a number of cryptographic key exchange protocols (Wide Mouthed Frog, Yahalom and the Needham-Schroeder symmetric key protocol) can be expressed in a rather natural manner in Boxed Ambients. We adapt the semantics and the 0CFA analysis to this setting and prove the usual results; thanks to a small simplification also the implementation is relatively straightforward. This analysis may additionally be used for validating security properties related to *exchange analysis* as presented in [12].

In the *Conclusion* (see Section 6) we summarise the development performed and briefly discuss extensions of the work as well as directions for future research: the

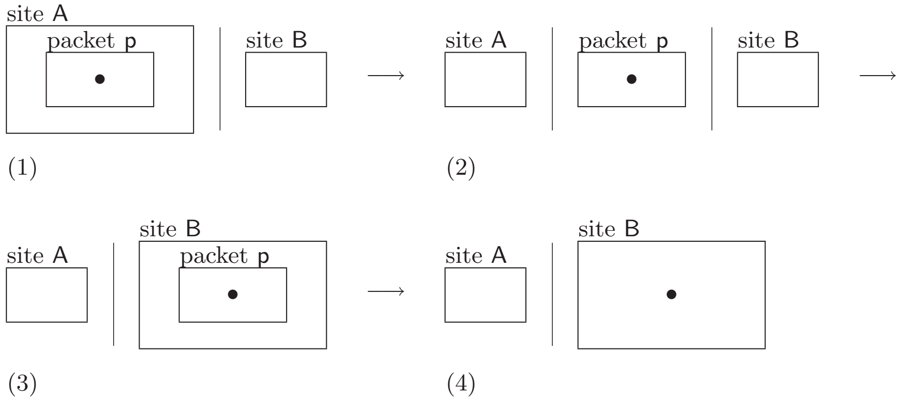


Fig. 1. A packet p moves from site A to site B and finally gets dissolved.

notion of hardest attackers as a means for characterising all Dolev-Yao attackers to a firewall [31, 30], and the possibility of extending this to capture all Dolev-Yao attackers to the protocols considered [5].

2 Mobile Ambients

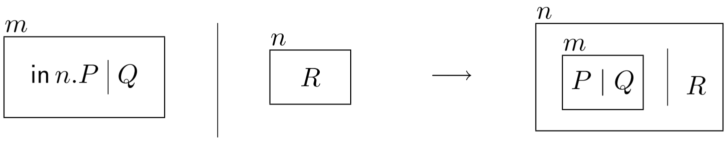
In the ambient view of the world each ambient is a bounded place where computations take place. The boundary determines what is inside and what is outside and as such it represents is a high-level security abstraction. Additionally it provides a powerful abstraction of mobility where ambients move as a whole. This view is sufficiently flexible to apply to a variety of scenarios: applets, agents, laptops, etc.

Ambients can be nested inside other ambients forming a tree structure. Mobility is then represented as navigation within this hierarchy of ambients. As an example, consider Figure 1 where a packet p moves from one site A into another site B. First we move the packet out the enclosing ambient (2) and then into the new enclosing ambient (3). Finally in (4), the payload of the packet is opened inside site B and the packet p is, thereby, dissolved.

Each ambient contains a number of multi-threaded running processes. The top-level processes of an ambient have direct control over it and can instruct it to move and thereby change the future behaviour of its processes and sub-ambients; consequently, the processes of sub-ambients only control the sub-ambient in which they are placed. Processes continue to run while being moved.

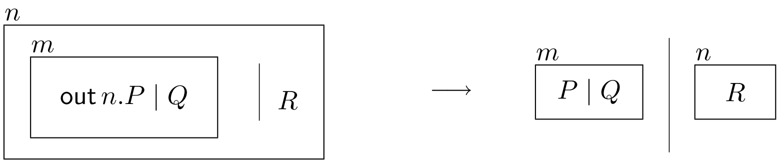
Each ambient has a name. Only the name can be used to control the access to the ambient (entry, exit, etc.) and the ambient names are unforgeable.

The mobility primitives of ambients are based on the notion of subjective moves. Here the movements of an ambient are caused by the threads running at top-level inside it. The in-capability directs an ambient to move into a sibling (i.e. another ambient running in parallel). This can be depicted as:

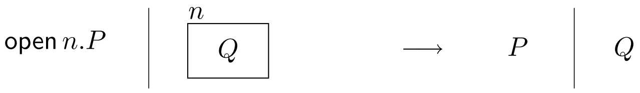


The left hand side shows two sibling ambients named m and n; the ambient m has a thread instructing it to move into the ambient n. The right hand side of the figure then shows the result of this action.

The out-capability directs an ambient to move out of its father (i.e. the enclosing ambient). In the figure below the ambient m contains a thread instructing it to move out of its father named n:



The open-capability allows a process to dissolve the boundary around a sibling ambient (named n below):



The ambient view of systems directly focuses on the ability to express a number of high-level security issues related to mobility; as for example, ensuring that packets with sensitive information can only pass through classified sites, or that packets with sensitive information may pass through unclassified sites but can only be opened at classified sites.

2.1 Syntax and Semantics of Mobile Ambients

To make this precise we formally define the syntax of processes, P, and capabilities, M, by the following grammar:

Processes based on the π-calculus:

$P ::= (\nu\, n\colon \mu)\, P$	introduces a process with private name n in group μ
$\mid\ (\nu\, \mu)\, P$	introduces a new group named μ with its scope
$\mid\ \mathbf{0}$	the inactive process
$\mid\ P_1 \mid P_2$	two concurrent processes
$\mid\ !P$	replication: any number of occurrences of P
$\mid\ n\,[P]$	an ambient named n containing P (drawn as $\overset{n}{\boxed{P}}$)
$\mid\ M.P$	a capability M followed by P

$P \equiv P$	$!P \equiv P \mid !P$	
$P \equiv Q \wedge Q \equiv R \Rightarrow P \equiv R$	$!\mathbf{0} \equiv \mathbf{0}$	
$P \equiv Q \Rightarrow Q \equiv P$	$(\nu\, n\colon \mu)\, \mathbf{0} \equiv \mathbf{0}$	
	$(\nu\, \mu)\, \mathbf{0} \equiv \mathbf{0}$	
$P \equiv Q \Rightarrow (\nu\, n\colon \mu)\, P \equiv (\nu\, n\colon \mu)\, Q$		
$P \equiv Q \Rightarrow (\nu\, \mu)\, P \equiv (\nu\, \mu)\, Q$	$(\nu\, n\colon \mu)\, (\nu\, n'\colon \mu')\, P \equiv$	
$P \equiv Q \Rightarrow P \mid R \equiv Q \mid R$	$(\nu\, n'\colon \mu')\, (\nu\, n\colon \mu)\, P$	if $n \neq n'$
$P \equiv Q \Rightarrow !P \equiv !Q$	$(\nu\, \mu)\, (\nu\, \mu')\, P \equiv (\nu\, \mu')\, (\nu\, \mu)\, P$	
$P \equiv Q \Rightarrow n\, [P] \equiv n\, [Q]$	$(\nu\, n\colon \mu)\, (\nu\, \mu')\, P \equiv (\nu\, \mu')\, (\nu\, n\colon \mu)\, P$	if $\mu \neq \mu'$
$P \equiv Q \Rightarrow M.\, P \equiv M.\, Q$		
	$(\nu\, n\colon \mu)\, (P \mid Q) \equiv P \mid (\nu\, n\colon \mu)\, Q$	if $n \notin \mathrm{fn}(P)$
$P \mid Q \equiv Q \mid P$	$(\nu\, \mu)\, (P \mid Q) \equiv P \mid (\nu\, \mu)\, Q$	if $\mu \notin \mathrm{fg}(P)$
$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$		
$P \mid \mathbf{0} \equiv P$	$(\nu\, n'\colon \mu)\, (n\, [P]) \equiv n\, [(\nu\, n'\colon \mu)\, P]$	if $n \neq n'$
	$(\nu\, \mu)\, (n\, [P]) \equiv n\, [(\nu\, \mu)\, P]$	
	$(\nu\, n\colon \mu)\, P \equiv (\nu\, n'\colon \mu)\, (P\{n \leftarrow n'\})$	if $n' \notin \mathrm{fn}(P)$

Table 1. The structural congruence relation.

Capabilities of the core calculus:

$M ::=$	$\mathsf{in}\, n$	move the enclosing ambient into a sibling named n
$\mid$	$\mathsf{out}\, n$	move the enclosing ambient out of a parent named n
$\mid$	$\mathsf{open}\, n$	dissolve a sibling ambient named n

In the graphical representation the inactive process is usually not written explicitly. Our syntax of ambients follows [12] and extends the basic calculus of [13, 16] in not having an operation $(\nu\, n)P$ for introducing a new private name n for use in P but instead two operations: $(\nu\, \mu)\, P$ for introducing a new group name μ for use in P and then $(\nu\, n\colon \mu)\, P$ for introducing a new private name n belonging to the group μ. A group can be viewed as the "type" of a name and has no semantic consequence.

The importance of groups becomes clear in the 0CFA analysis where we will need that the group name is stable under α-renaming of names. We achieve this by means of a careful definition of the structural congruence (see Table 1 and the explanation below). For simplicity there will be no α-renaming of group names; for this to work we make the simplifying assumption that all $(\nu\, \mu)$ must be distinct and must not occur inside some replication operator (!).

The semantics of mobile ambients consists of a structural congruence relation, written $P \equiv Q$, and a transition relation, written $P \to Q$. The structural congruence relation allows to rearrange the syntactic appearance of processes as the pictorial representation suggests (e.g. $P \mid Q \equiv Q \mid P$), it deals with the semantics of replication (e.g. $!P \equiv !P \mid P$) and allows to perform α-renaming; the details are fairly standard and may be found in Table 1. We write $\mathrm{fn}(P)$

$$\frac{P \to Q}{(\nu\, n \colon \mu)\, P \to (\nu\, n \colon \mu)\, Q} \qquad \frac{P \to Q}{(\nu\, \mu)\, P \to (\nu\, \mu)\, Q}$$

$$\frac{P \to Q}{n\,[P] \to n\,[Q]} \qquad \frac{P \to Q}{P \mid R \to Q \mid R} \qquad \frac{P \equiv P' \wedge P' \to Q' \wedge Q' \equiv Q}{P \to Q}$$

$$m\,[\mathsf{in}\, n.\, P \mid Q] \mid n\,[R] \;\to\; n\,[m\,[P \mid Q] \mid R]$$

$$n\,[m\,[\mathsf{out}\, n.\, P \mid Q] \mid R] \;\to\; m\,[P \mid Q] \mid n\,[R]$$

$$\mathsf{open}\, n.\, P \mid n\,[Q] \;\to\; P \mid Q$$

Table 2. The transition relation for Mobile Ambients.

and fg(P) for the free names and the free groups of P, respectively. The transition relation formalises the three subjective moves and clarifies that capabilities deeply nested inside ambients may execute provide they are not prefixed with other capabilities; the details are fairly standard and may be found in Table 2.

Example 1. Let us consider a packet p moving from a site A to another site B as in Figure 1. The first configuration (1) in Figure 1 could be the process:

$$\mathsf{A}\,[\mathsf{p}\,[\mathsf{out}\,\mathsf{A}.\ \mathsf{in}\ \mathsf{B}]] \mid \mathsf{B}\,[\mathsf{open}\ \mathsf{p}]$$

Using the transition relation of Table 2 we can get an execution sequence corresponding to that of Figure 1:

$$\begin{array}{lll}
(1) & \mathsf{A}\,[\mathsf{p}\,[\mathsf{out}\,\mathsf{A}.\ \mathsf{in}\ \mathsf{B}]] \mid \mathsf{B}\,[\mathsf{open}\ \mathsf{p}] & \to \\
(2) & \mathsf{A}\,[\;] \mid \mathsf{p}\,[\mathsf{in}\ \mathsf{B}] \mid \mathsf{B}\,[\mathsf{open}\ \mathsf{p}] & \to \\
(3) & \mathsf{A}\,[\;] \mid \mathsf{B}\,[\mathsf{open}\ \mathsf{p} \mid \mathsf{p}\,[\;]] & \to \\
(4) & \mathsf{A}\,[\;] \mid \mathsf{B}\,[\;] &
\end{array}$$

□

2.2 A 0CFA Analysis for Mobile Ambients

The aim of the static analysis is to determine which ambients and capabilities may turn up inside given ambients. Fixing our attention on a given ambient process P our aim is to find an estimate $\mathcal{I}$ of this information that describes all configurations reachable from P. In the Flow Logic approach to static analysis [28, 39] we proceed in the following stages:

Specification: First we define what it means for the estimate $\mathcal{I}$ to be an acceptable description of the process P.

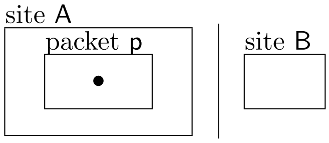
Correctness: Then we prove that all acceptable estimates will remain acceptable during execution.

Implementation: Finally, we show that a best acceptable estimate can be calculated in polynomial time.

This approach should be rather natural to readers familiar with type systems: first one formulates the type system (thereby making precise the notion of *type checking*), then one shows semantic soundness of the type system (usually in the form of a *subject-reduction* result), and finally one shows how to obtain principal types for processes (thereby making precise the notion of *type inference*).

Example 2. Consider the Mobile Ambient process of Example 1

$$\mathsf{A}\,[\mathsf{p}\,[\mathsf{out}\,\mathsf{A}.\,\mathsf{in}\,\mathsf{B}]]\mid \mathsf{B}\,[\mathsf{open}\,\mathsf{p}]$$



where ambients A and B belong to the group $\mathbb{S}$ of sites, and the ambient p belongs to the group $\mathbb{P}$ of packets. The analysis will provide a safe approximation of which ambients may turn up inside other ambients.

The exact answer is that p may turn up inside A, p may turn up inside B, but that A and B never turn up inside p nor inside each other. In terms of groups we shall simply say that $\mathbb{P}$ may turn up inside $\mathbb{S}$ but that $\mathbb{S}$ never turns up inside $\mathbb{P}$ nor inside $\mathbb{S}$. We shall represent this using a mapping

$$\mathcal{I} : \mathbf{Group} \to \mathcal{P}(\mathbf{Group} \cup \mathbf{Cap})$$

that for each ambient group $\mu \in \mathbf{Group}$ tells us not only which ambient groups may be inside an ambient in group μ but also which group capabilities may be possessed by an ambient in group μ; here a group capability $m \in \mathbf{Cap}$ is given by:

$$m ::= \mathsf{in}\,\mu \mid \mathsf{out}\,\mu \mid \mathsf{open}\,\mu$$

The optimum value of $\mathcal{I}$ for the example discussed above is given by

$$\begin{array}{rcl} \mathcal{I}(\star) & = & \{\mathbb{S},\, \mathbb{P}\} \\ \mathcal{I}(\mathbb{S}) & = & \{\mathbb{P},\, \mathsf{open}\,\mathbb{P}\} \\ \mathcal{I}(\mathbb{P}) & = & \{\mathsf{in}\,\mathbb{S},\, \mathsf{out}\,\mathbb{S}\} \end{array}$$

where $\star$ denotes the group of the overall system (i.e. the top level). A somewhat less precise *over-approximation* of $\mathcal{I}$, where extra elements are included, is

$$\begin{array}{rcl} \mathcal{I}(\star) & = & \{\mathbb{S},\, \mathbb{P}\} \\ \mathcal{I}(\mathbb{S}) & = & \{\mathbb{P},\, \mathbb{S},\, \mathsf{in}\,\mathbb{S},\, \mathsf{out}\,\mathbb{S},\, \mathsf{open}\,\mathbb{P}\} \\ \mathcal{I}(\mathbb{P}) & = & \{\mathsf{in}\,\mathbb{S},\, \mathsf{out}\,\mathbb{S}\} \end{array}$$

and it will turn out that this is the one that will be obtained using the simplest of our analyses (the 0CFA analysis developed in this subsection). □

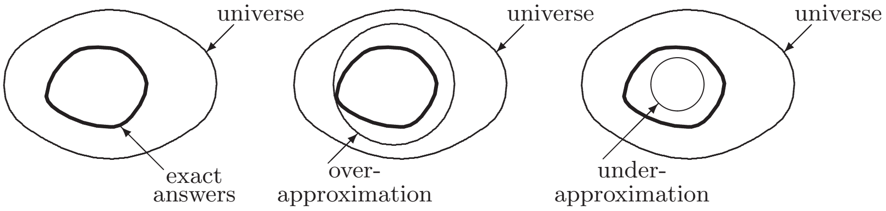


Fig. 2. The nature of approximation.

Remark 3. The choice of the domain of $\mathcal{I}$ determines which kind of information the analysis can provide about a process and, consequently, what it *cannot* record.

For example, with the choice of $\mathcal{I}$ as in Example 2 we can record whether an ambient or a capability is present inside some other ambient, but not *the number* of ambients and capabilities that are present. To get such information we will have to change the domain of the analysis estimate as done in e.g. [37, 23, 38].

Furthermore, we can only records the presence of capabilities but not the order in which they appear. Thus, we do not capture the order in which capabilities are executed and cannot determine whether one capability is executed before another; in other words the analysis is not flow-sensitive. Adding sequences of capabilities have been studied in [23].

In the remainder of this paper we *do not* consider neither multiplicities nor flow-sensitivity. As we will see, even these “simple” analyses are able to give analysis estimates that are sufficiently precise to determine interesting security properties. □

Specification of the 0CFA analysis. In the above example we displayed the best value of $\mathcal{I}$ that one can hope for. In general this is not always possible since the problem of finding the best value of $\mathcal{I}$ is really undecidable due to the Turing completeness of Mobile Ambients. Hence we will have to settle for more approximate estimates saying that $\mathbb{S}$ may turn up inside $\mathbb{P}$, whereas we shall never accept an estimate saying that $\mathbb{P}$ never turns up inside $\mathbb{S}$. In terms of approximation this means that we opt for an over-approximation of the set of containments; this is illustrated in Figure 2.

To make precise when an estimate $\mathcal{I} : \mathbf{Group} \to \mathcal{P}(\mathbf{Group} \cup \mathbf{Cap})$ describes an acceptable over-approximation of the behaviour of a process P under consideration we shall axiomatise a judgement

$$\mathcal{I} \models^{\mu}_{\Gamma} P$$

meaning that $\mathcal{I}$ is an acceptable analysis estimate for the process P when it occurs inside an ambient from the group μ and whenever the ambients are in groups as specified by the type environment Γ (e.g. $\Gamma(\mathsf{p}) = \mathbb{P}$ and $\Gamma(\mathsf{A}) = \Gamma(\mathsf{B}) = \mathbb{S}$). The judgement is defined by structural induction on the syntax of the process P (as shown below).

Analysis of composite processes. Each acceptable analysis estimate for a composite process must also be an acceptable analysis estimate for its sub-processes; perhaps more imprecise than need be. This is captured by the following clauses where Γ is the current type environment and $\star$ is the ambience i.e. the group associated with the enclosing ambient.

$\mathcal{I} \models^{\star}_{\Gamma} (\nu\, n\colon \mu)\, P$	iff $\mathcal{I} \models^{\star}_{\Gamma[n \mapsto \mu]} P$	update type env.; check process
$\mathcal{I} \models^{\star}_{\Gamma} (\nu\, \mu)\, P$	iff $\mathcal{I} \models^{\star}_{\Gamma[\mu \mapsto \diamond]} P$	update type env.; check process
$\mathcal{I} \models^{\star}_{\Gamma} \mathbf{0}$	iff true	nothing to check
$\mathcal{I} \models^{\star}_{\Gamma} P_1 \mid P_2$	iff $\mathcal{I} \models^{\star}_{\Gamma} P_1 \;\wedge\; \mathcal{I} \models^{\star}_{\Gamma} P_2$	check both branches
$\mathcal{I} \models^{\star}_{\Gamma} !P$	iff $\mathcal{I} \models^{\star}_{\Gamma} P$	check process; ignore multiplicity
$\mathcal{I} \models^{\star}_{\Gamma} n\,[P]$	iff $\mu \in \mathcal{I}(\star) \;\wedge\; \mathcal{I} \models^{\mu}_{\Gamma} P$ where $\mu = \Gamma(n)$	μ is inside $\star$; check process

In the first clause we update the type environment with the type of the newly introduced name; in the second clause we update the type environment with a special placeholder $\diamond$ indicating a group name; in the last clause we ensure that the analysis estimate $\mathcal{I}$ records that the group of n occurs inside the ambience $\star$ and we analyse the internals of n in an appropriately updated ambience.

Remark 4. Elaborating on the analogy to type systems explained above, one could coin the slogan:

Flow Logic is the approach to static analysis that presents Data and Control Flow Analysis as a Type System.

To make this more apparent we could formulate the first clause above as an inference rule

$$\frac{\mathcal{I} \models^{\star}_{\Gamma[n \mapsto \mu]} P}{\mathcal{I} \models^{\star}_{\Gamma} (\nu\, n\colon \mu)\, P}$$

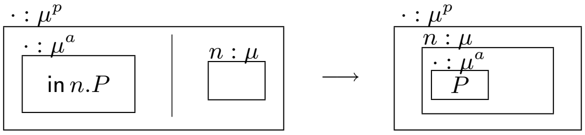
and similarly for the other clauses presented here. These formulations are equivalent[1] whenever the judgement is defined by structural induction on processes. The formulation chosen here is perhaps more in the spirit of the equational approach of Data Flow Analysis. □

[1] For some applications of Flow Logic to programming languages with higher-order features this need not be the case and then the inference system presentation amounts to an inductive definition rather than the desired co-inductive definition [29, 39]; however, this subtle but important point will not be an issue in the present paper where the inductive definition turns out to coincide with the co-inductive definition.

In-*capability.* Each acceptable analysis estimate must mimic the semantics: if the semantics allows one configuration to evolve into another then it must be reflected in the analysis estimate. For the in-capability this is achieved by the following clause:

$$
\begin{array}{lll}
\mathcal{I} \models^{\star}_{\Gamma} \mathsf{in}\, n.\ P \text{ iff} & \mathsf{in}\ \mu \in \mathcal{I}(\star)\ \wedge\ \mathcal{I} \models^{\star}_{\Gamma} P\ \wedge & \\
& \forall\ \mu^a, \mu^p : \mathsf{in}\ \mu \in \mathcal{I}(\mu^a)\ \wedge & \mu^a \text{ has the capability } \mathsf{in}\, \mu \\
& \qquad\qquad \mu^a \in \mathcal{I}(\mu^p)\ \wedge & \mu^p \text{ is the parent of } \mu^a \\
& \qquad\qquad \mu \in \mathcal{I}(\mu^p) & \mu^a \text{ has a sibling } \mu \\
& \qquad\quad \Rightarrow \mu^a \in \mathcal{I}(\mu) & \mu^a \text{ may move into } \mu \\
& \text{where } \mu = \Gamma(n) &
\end{array}
$$

Here the first line records the presence of the actual capability and also analyses the continuation – this is in line with what happened for ambients above. The remaining lines model the semantics. To understand the formulation it may be helpful to recall the semantics of the in-capability as follows (writing $n : \mu$ to indicate that $\Gamma(n) = \mu$ and writing $\cdot : \mu$ when the ambient name is of no importance for the analysis):



The precondition of the universally quantified implication above recognises the structure depicted to the left of the arrow by querying if the relevant entries already are in $\mathcal{I}$; the conclusion then records the only new structural ingredient depicted to the right of the arrow.

Example 5. Let Γ be given by $\Gamma(\mathsf{p}) = \mathbb{P}$ and $\Gamma(\mathsf{A}) = \Gamma(\mathsf{B}) = \mathbb{S}$ and let $\mathcal{I}$ be given by the second estimate in Example 2:

$$
\begin{array}{lcl}
\mathcal{I}(\star) & = & \{\mathbb{S},\, \mathbb{P}\} \\
\mathcal{I}(\mathbb{S}) & = & \{\mathbb{P},\, \mathbb{S},\, \mathsf{in}\, \mathbb{S},\, \mathsf{out}\, \mathbb{S},\, \mathsf{open}\, \mathbb{P}\} \\
\mathcal{I}(\mathbb{P}) & = & \{\mathsf{in}\, \mathbb{S},\, \mathsf{out}\, \mathbb{S}\ \}
\end{array}
$$

Checking that

$$\mathcal{I} \models^{\star}_{\Gamma} \mathsf{A}\, [\mathsf{p}\, [\mathsf{out}\, \mathsf{A}.\ \mathsf{in}\, \mathsf{B}\ ]\ ]\ \ |\ \ \mathsf{B}\, [\mathsf{open}\ \mathsf{p}]$$

involves checking

$$\mathcal{I} \models^{\mathbb{P}}_{\Gamma} \mathsf{in}\, \mathsf{B}$$

which holds because $\mathsf{in}\, \mathbb{S} \in \mathcal{I}(\mathbb{P})$ and

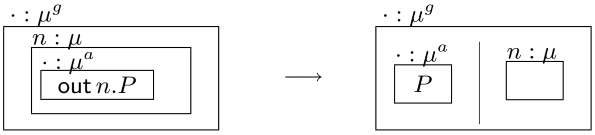
$$\mathsf{in}\, \mathbb{S} \in \mathcal{I}(\mu^a) \wedge \mu^a \in \mathcal{I}(\mu^p) \wedge \mathbb{S} \in \mathcal{I}(\mu^p) \Rightarrow \mu^a \in \mathcal{I}(\mathbb{S})$$

holds for all non-trivial $(\mu^a, \mu^p) \in \{(\mathbb{S}, \star), (\mathbb{S}, \mathbb{S}), (\mathbb{P}, \star), (\mathbb{P}, \mathbb{S})\}$. □

Out-*capability.* For the out-capability the clause is

$$\begin{array}{lll}
\mathcal{I} \models^{\star}_{\Gamma} \mathsf{out}\, n.\ P \text{ iff } & \mathsf{out}\ \mu \in \mathcal{I}(\star)\ \wedge\ \mathcal{I} \models^{\star}_{\Gamma} P\ \wedge & \\
& \forall\ \mu^a, \mu^g : \mathsf{out}\ \mu \in \mathcal{I}(\mu^a) \wedge & \mu^a \text{ has the capability } \mathsf{out}\,\mu \\
& \qquad\qquad \mu^a \in \mathcal{I}(\mu)\ \wedge & \mu \text{ is the parent of } \mu^a \\
& \qquad\qquad \mu \in \mathcal{I}(\mu^g) & \mu^g \text{ is the grandparent of } \mu^a \\
& \qquad\quad \Rightarrow \mu^a \in \mathcal{I}(\mu^g) & \mu^a \text{ may move out of } \mu \\
& \text{where } \mu = \Gamma(n) &
\end{array}$$

corresponding to the operational semantics:



Example 6. Continuing Example 5, checking

$$\mathcal{I} \models^{\star}_{\Gamma} \mathsf{A}\,[\mathsf{p}\,[\mathsf{out}\,\mathsf{A}.\ \mathsf{in}\,\mathsf{B}]\ ]\ \mid\ \mathsf{B}\,[\mathsf{open}\ \mathsf{p}]$$

involves checking

$$\mathcal{I} \models^{\mathbb{P}}_{\Gamma} \mathsf{out}\,\mathsf{A}.\ \mathsf{in}\,\mathsf{B}$$

which holds because $\mathcal{I} \models^{\mathbb{P}}_{\Gamma} \mathsf{in}\,\mathsf{B}$ (see Example 5) and $\mathsf{out}\,\mathbb{S} \in \mathcal{I}(\mathbb{P})$ and

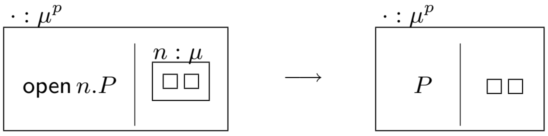
$$\mathsf{out}\,\mathbb{S} \in \mathcal{I}(\mu^a) \wedge \mu^a \in \mathcal{I}(\mathbb{S}) \wedge \mathbb{S} \in \mathcal{I}(\mu^g) \Rightarrow \mu^a \in \mathcal{I}(\mu^g)$$

holds for all $(\mu^a, \mu^g) \in \{(\mathbb{S}, \star), (\mathbb{S}, \mathbb{S}), (\mathbb{P}, \star), (\mathbb{P}, \mathbb{S})\}$. □

Open-*capability.* For the open-capability the clause is

$$\begin{array}{lll}
\mathcal{I} \models^{\star}_{\Gamma} \mathsf{open}\, n.\ P \text{ iff } & \mathsf{open}\ \mu \in \mathcal{I}(\star)\ \wedge\ \mathcal{I} \models^{\star}_{\Gamma} P\ \wedge & \\
& \forall\ \mu^p : \mathsf{open}\ \mu \in \mathcal{I}(\mu^p)\ \wedge & \mu^p \text{ has the capability } \mathsf{open}\,\mu \\
& \qquad\quad \mu \in \mathcal{I}(\mu^p) & \mu \text{ is a sibling of } \mathsf{open}\,\mu \\
& \quad \Rightarrow \mathcal{I}(\mu) \subseteq \mathcal{I}(\mu^p) & \text{everything in } \mu \text{ may be in } \mu^p \\
& \text{where } \mu = \Gamma(n) &
\end{array}$$

corresponding to the operational semantics:



Example 7. Continuing Example 5 and Example 6, checking that

$$\mathcal{I} \models^{\star}_{\Gamma} \mathsf{A}\,[\mathsf{p}\,[\mathsf{out}\,\mathsf{A}.\ \mathsf{in}\,\mathsf{B}\ ]]\ \mid\ \mathsf{B}\,[\mathsf{open}\ \mathsf{p}]$$

involves checking

$$\mathcal{I} \models^{\mathbb{S}}_{\Gamma} \mathsf{open}\ \mathsf{p}$$

$$\begin{tikzcd} P_1 \arrow[r] \arrow[d, Leftrightarrow, "\models^{\star}_{\Gamma}"] & P_2 \arrow[r] \arrow[d, Leftrightarrow, "\models^{\star}_{\Gamma}"] & \cdots \arrow[r] & P_i \arrow[r] \arrow[d, Leftrightarrow, "\models^{\star}_{\Gamma}"] & \cdots \\ \mathcal{I} & \mathcal{I} & \cdots & \mathcal{I} & \cdots \end{tikzcd}$$

Fig. 3. Subject reduction: the analysis estimate remains acceptable under execution.

which holds because $\mathsf{open}\,\mathbb{P} \in \mathcal{I}(\mathbb{S})$ and

$$\mathsf{open}\,\mathbb{P} \in \mathcal{I}(\mu^p) \land \mathbb{P} \in \mathcal{I}(\mathbb{S}) \Rightarrow \mathcal{I}(\mathbb{P}) \subseteq \mathcal{I}(\mu^p)$$

holds for $\mu^p = \mathbb{S}$. □

This concludes the Flow Logic definition of the judgement $\mathcal{I} \models^{\star}_{\Gamma} P$ in a style close to that of a type system.

Example 8. Ensuring that the analysis estimate $\mathcal{I}$ of Example 5 is an acceptable analysis estimate for *the entire process* $\mathsf{A}\,[\mathsf{p}\,[\mathsf{out}\,\mathsf{A}.\ \mathsf{in}\,\mathsf{B}]\ ]\ \mid\ \mathsf{B}\,[\mathsf{open}\,\mathsf{p}]$ amounts to checking that

$$\mathcal{I} \models^{\star}_{\Gamma} \mathsf{A}\,[\mathsf{p}\,[\mathsf{out}\,\mathsf{A}.\ \mathsf{in}\,\mathsf{B}]\ ]\ \mid\ \mathsf{B}\,[\mathsf{open}\,\mathsf{p}]$$

This involves checking the clauses for composite processes. The top-level parallel composition gives rise to the two checks that

$$\mathcal{I} \models^{\star}_{\Gamma} \mathsf{A}\,[\mathsf{p}\,[\mathsf{out}\,\mathsf{A}.\ \mathsf{in}\,\mathsf{B}]] \quad \text{and} \quad \mathcal{I} \models^{\star}_{\Gamma} \mathsf{B}\,[\mathsf{open}\,\mathsf{p}]$$

which, for the first part, leads to the checks that

$$\mathbb{S} \in \mathcal{I}(\star) \quad \text{and} \quad \mathcal{I} \models^{\mathbb{S}}_{\Gamma} \mathsf{p}\,[\mathsf{out}\,\mathsf{A}.\ \mathsf{in}\,\mathsf{B}]$$

In turn, checking the clauses for composite processes leads to the checks of capabilities performed in Example 5, 6, and 7. Performing all the required checks we find that the analysis estimate $\mathcal{I}$ of Example 5 is indeed an acceptable analysis estimate for the 0CFA analysis. □

Correctness of the 0CFA analysis. Although the specification of the judgement in the previous subsection was motivated by the transition relation it was not formally linked to it. To do so we take the rather natural approach, familiar from type systems, that the analysis estimate should not only describe the initial configuration in an acceptable way but it must remain acceptable under execution; then we know that all reachable configurations will be described by the analysis estimate.

This is the "subject reduction" approach to correctness; it is illustrated in Figure 3 and formalised by the following theorem:

Theorem 9. *If* $\quad \mathcal{I} \models^\star_\Gamma P \quad$ *and* $\quad P \rightarrow^* Q \quad$ *then* $\quad \mathcal{I} \models^\star_\Gamma Q$.

For the proof we first show that the analysis estimate is invariant under the structural congruence:

$$\text{If} \quad P \equiv Q \quad \text{then} \quad \mathcal{I} \models^\star_\Gamma P \quad \text{if and only if} \quad \mathcal{I} \models^\star_\Gamma Q.$$

This amounts to a straightforward induction on the inference of $P \equiv Q$.

Next we prove that the analysis estimate is preserved under the transition relation:

$$\text{If} \quad P \rightarrow Q \quad \text{and} \quad \mathcal{I} \models^\star_\Gamma P \quad \text{then} \quad \mathcal{I} \models^\star_\Gamma Q.$$

This amounts to a straightforward induction on the inference of $P \rightarrow Q$.

Finally we prove the theorem by a simple numerical induction on the number of steps k in $P \rightarrow^k Q$.

Implementation of the 0CFA analysis. An abstract argument for why there always is a best acceptable analysis estimate borrows from abstract interpretation [19, 20, 29]. The notion of "best estimate" means "least estimate" since we decided to opt for over-approximation and hence we are looking for a value of $\mathcal{I}$ that contains as few elements as possible. The abstract argument then amounts to the Moore Family result (or model intersection property in model-theoretic terminology):

Theorem 10. *For each P, the set $\{\mathcal{I} \mid \mathcal{I} \models^\star_\Gamma P\}$ is a Moore family; in other words: for each P, if $\mathcal{Y} \subseteq \{\mathcal{I} \mid \mathcal{I} \models^\star_\Gamma P\}$ then $\sqcap\mathcal{Y} \models^\star_\Gamma P$ where $(\sqcap\mathcal{Y})(\mu) = \bigcap\{\mathcal{I}(\mu) \mid \mathcal{I} \in \mathcal{Y}\}$.*

The proof is by structural induction on P. We are interested in the Moore family property because a Moore family *always* contains a unique least element. Thus it follows that the least analysis estimate can be expressed as $\sqcap\{\mathcal{I} \mid \mathcal{I} \models^\star_\Gamma P\}$ and in the world of type systems this corresponds to each process admitting a single *principal type*.

The ALFP logic. To actually compute the intended solution in polynomial time we shall follow a rather general and elegant method where the specification is translated into an extension of Horn clauses known as Alternation-free Least Fixed Point Logic (ALFP). This is a first-order logic where the set of formulae (or clauses), clause, and the set of preconditions, pre, are given by the following grammar (subject to a notion of stratification limiting the use of negation):

$$\begin{array}{lll}
\mathsf{pre} & ::= & R\,(x_1,\ldots,x_k) \quad | \quad \neg R\,(x_1,\ldots,x_k) \quad | \quad x = y \quad | \quad x \neq y \\
 & & \mathsf{pre}_1 \wedge \mathsf{pre}_2 \quad | \quad \mathsf{pre}_1 \vee \mathsf{pre}_2 \quad | \quad \forall x : \mathsf{pre} \quad | \quad \exists x : \mathsf{pre} \\
\mathsf{clause} & ::= & R\,(x_1,\ldots,x_k) \quad | \quad \mathbf{1} \quad | \quad \mathsf{clause}_1 \wedge \mathsf{clause}_2 \quad | \\
 & & \mathsf{pre} \implies \mathsf{clause} \quad | \quad \forall x : \mathsf{clause}
\end{array}$$

$(\rho,\sigma) \models R\,(x_1,\ldots,x_k)$	$\Leftrightarrow$	$(\sigma\,x_1,\ldots,\sigma\,x_k) \in \rho\,R$
$(\rho,\sigma) \models \neg R\,(x_1,\ldots,x_k)$	$\Leftrightarrow$	$(\sigma\,x_1,\ldots,\sigma\,x_k) \notin \rho\,R$
$(\rho,\sigma) \models x = y$	$\Leftrightarrow$	$\sigma\,x = \sigma\,y$
$(\rho,\sigma) \models x \neq y$	$\Leftrightarrow$	$\sigma\,x \neq \sigma\,y$
$(\rho,\sigma) \models \mathsf{pre}_1 \wedge \mathsf{pre}_2$	$\Leftrightarrow$	$(\rho,\sigma) \models \mathsf{pre}_1$ and $(\rho,\sigma) \models \mathsf{pre}_2$
$(\rho,\sigma) \models \mathsf{pre}_1 \vee \mathsf{pre}_2$	$\Leftrightarrow$	$(\rho,\sigma) \models \mathsf{pre}_1$ or $(\rho,\sigma) \models \mathsf{pre}_2$
$(\rho,\sigma) \models \forall x : \mathsf{pre}$	$\Leftrightarrow$	$(\rho,\sigma[x \mapsto a]) \models \mathsf{pre}$ for all $a \in \mathcal{U}$
$(\rho,\sigma) \models \exists x : \mathsf{pre}$	$\Leftrightarrow$	$(\rho,\sigma[x \mapsto a]) \models \mathsf{pre}$ for some $a \in \mathcal{U}$
$(\rho,\sigma) \models R\,(x_1,\ldots,x_k)$	$\Leftrightarrow$	$(\sigma\,x_1,\ldots,\sigma\,x_k) \in \rho\,R$
$(\rho,\sigma) \models \mathbf{1}$	$\Leftrightarrow$	true
$(\rho,\sigma) \models \mathsf{clause}_1 \wedge \mathsf{clause}_2$	$\Leftrightarrow$	$(\rho,\sigma) \models \mathsf{clause}_1$ and $(\rho,\sigma) \models \mathsf{clause}_2$
$(\rho,\sigma) \models \mathsf{pre} \implies \mathsf{clause}$	$\Leftrightarrow$	$(\rho,\sigma) \models \mathsf{clause}$ whenever $(\rho,\sigma) \models \mathsf{pre}$
$(\rho,\sigma) \models \forall x : \mathsf{clause}$	$\Leftrightarrow$	$(\rho,\sigma[x \mapsto a]) \models \mathsf{clause}$ for all $a \in \mathcal{U}$

Table 3. Semantics of ALFP.

Here R is a k-ary predicate symbol for $k \geq 0$, and $y, x, x_1, \ldots$ denote arbitrary variables, while $\mathbf{1}$ is the always true clause. (Since we shall not use negation in this paper we dispense with explaining the notion of stratification.)

Given a universe finite $\mathcal{U}$ of atomic values and interpretations ρ and σ for predicate symbols and free variables, respectively, the satisfaction relations $(\rho,\sigma) \models \mathsf{pre}$ for pre-conditions and $(\rho,\sigma) \models \mathsf{clause}$ for clauses are defined in a straightforward manner as shown in Table 3. Note that the definitions for pre-conditions and clauses are similar for predicates $R\,(x_1,\ldots,x_k))$, conjunction $(\wedge)$, and universal quantification $(\forall)$. We view the free variables occurring in a formula as constant symbols or atoms from the finite universe $\mathcal{U}$. Thus, given an interpretation σ of the constant symbols, in the clause clause, we call an interpretation ρ of the predicate symbols a *solution* provided $(\rho,\sigma) \models \mathsf{clause}$.

Implementation using ALFP. Calculating a least analysis estimate is done by finding the least solution to an ALFP clause that is logically equivalent to the specification of the analysis. The calculation of the least solution ρ is done *automatically* using our Succinct Solver [35]. Recall that being the *least* interpretation means that it contains as few elements as possible while still being acceptable.

In the specification of an analysis we are free to use any mathematical notation, while in the implementation we are limited by what we can express in ALFP. For simple powerset based analyses, such as the ones considered in this paper, the transformation from the specification to ALFP is relatively straightforward. For the analysis considered here the following transformations suffice:

- The mapping $\mathcal{I} : \mathbf{Group} \to \mathcal{P}(\mathbf{Group} \cup \mathbf{Cap})$ is encoded as a *binary predicate* of sort $\mathbf{Group} \times (\mathbf{Group} \cup \mathbf{Cap})$.

- Correspondingly, set membership such as $\mu' \in \mathcal{I}(\mu)$ is written as $\mathcal{I}(\mu, \mu')$.
- Subset relations such as $\mathcal{I}(\mu) \subseteq \mathcal{I}(\mu')$ are written by explicitly quantifying the elements in the first set: $\forall u : \mathcal{I}(\mu, u) \Rightarrow \mathcal{I}(\mu', u)$.
- All groups μ and group capabilities $\mathsf{in}\,\mu, \mathsf{out}\,\mu$, and $\mathsf{open}\,\mu$ are elements in the universe $\mathcal{U}$; for a given process this universe will be finite.

It is straightforward to establish a formal relationship between the specification and the implementation by giving a mapping (actually an isomorphism) between the two representations of $\mathcal{I}$ and showing that the specification and the implementation are logically equivalent under this mapping.

We apply the above encodings systematically to the specification of the analysis thus getting a new formulation from which we can generate ALFP clauses for any given process P. The analysis of composite processes remains unchanged except for the analysis of the ambient construct, which is changed into:

$$\begin{array}{l} \mathcal{I} \models^{\star}_{\Gamma} n\,[P] \ \text{ iff } \ \mathcal{I}(\star, \mu) \ \wedge \ \mathcal{I} \models^{\mu}_{\Gamma} P \\ \qquad\qquad\qquad\quad \text{where } \mu = \Gamma(n) \end{array}$$

That is, the set membership is now written $\mathcal{I}(\star, \mu)$. Similarly, the analysis of the capabilities are changed and in the case of $\mathsf{open}\,\mu$ (where subset is used) the clause becomes:

$$\begin{array}{ll} \mathcal{I} \models^{\star}_{\Gamma} \mathsf{open}\,n.\ P \text{ iff} & \mathcal{I}(\star, \mathsf{open}\ \mu) \wedge \mathcal{I} \models^{\star}_{\Gamma} P \ \ \wedge \\ & \forall\ \mu^p : \mathcal{I}(\mu^p, \mathsf{open}\ \mu) \wedge \\ & \qquad\quad \mathcal{I}(\mu^p, \mu) \\ & \qquad\quad \Rightarrow \forall u : \mathcal{I}(\mu, u) \Rightarrow \mathcal{I}(\mu^p, u) \\ & \text{where } \mu = \Gamma(n) \end{array}$$

The rules for in- and out-capabilities are obtained in an analogous way.

Example 11. Finding the least analysis estimate for the 0CFA analysis of the process

$$\mathsf{A}\,[\mathsf{p}\,[\mathsf{out}\,\mathsf{A}.\,\mathsf{in}\,\mathsf{B}]]\ |\ \mathsf{B}\,[\mathsf{open}\,\mathsf{p}]$$

with Γ given by $\Gamma(\mathsf{p}) = \mathbb{P}$ and $\Gamma(\mathsf{A}) = \Gamma(\mathsf{B}) = \mathbb{S}$ now amounts to finding the solution to the ALFP clause:

$$\begin{array}{ll} \mathcal{I}(\star, \mathbb{S}) \wedge & \text{Ambient } \mathsf{A} \\ \mathcal{I}(\mathbb{S}, \mathbb{P}) \wedge & \text{Ambient } \mathsf{p} \\ \mathcal{I}(\mathbb{P}, \mathsf{out}\,\mathbb{S}) \wedge & \text{Capability } \mathsf{out}\,\mathsf{A} \\ (\forall \mu^a, \mu^g : \mathcal{I}(\mu^a, \mathsf{out}\,\mathbb{S}) \wedge \mathcal{I}(\mathbb{S}, \mu^a) \wedge \mathcal{I}(\mu^g, \mathbb{S}) & \\ \qquad\quad \Rightarrow \mathcal{I}(\mu^g, \mu^a)) \wedge & \\ \mathcal{I}(\mathbb{P}, \mathsf{in}\,\mathbb{S}) \wedge & \text{Capability } \mathsf{in}\,\mathsf{B} \\ (\forall \mu^a, \mu^p : \mathcal{I}(\mu^a, \mathsf{in}\,\mathbb{S}) \wedge \mathcal{I}(\mu^p, \mu^a) \wedge \mathcal{I}(\mu^p, \mathbb{S}) & \\ \qquad\quad \Rightarrow \mathcal{I}(\mathbb{S}, \mu^a)) \wedge & \\ \mathcal{I}(\star, \mathbb{S}) \wedge & \text{Ambient } \mathsf{B} \\ \mathcal{I}(\mathbb{S}, \mathsf{open}\,\mathbb{P}) \wedge & \text{Capability } \mathsf{open}\,\mathsf{p} \\ (\forall \mu^p : \mathcal{I}(\mu^p, \mathsf{open}\,\mathbb{P}) \wedge \mathcal{I}(\mu^p, \mathbb{P}) & \\ \qquad \Rightarrow \forall u : \mathcal{I}(\mathbb{P}, u) \Rightarrow \mathcal{I}(\mu^p, u)) & \end{array}$$

The resulting least solution $\mathcal{I}$ which satisfies the clause is:

$$
\begin{array}{ll}
\mathcal{I} : (\star, \mathbb{S}), (\mathbb{S}, \mathbb{P}), & \text{from ambients} \\
\quad (\mathbb{P}, \mathsf{out}\,\mathbb{S}), (\star, \mathbb{P}), & \text{from } \mathsf{out} \\
\quad (\mathbb{P}, \mathsf{in}\,\mathbb{S}), (\mathbb{S}, \mathbb{S}), & \text{from } \mathsf{in} \\
\quad (\mathbb{S}, \mathsf{open}\,\mathbb{P}), (\mathbb{S}, \mathsf{out}\,\mathbb{S}), (\mathbb{S}, \mathsf{in}\,\mathbb{S}) & \text{from } \mathsf{open}
\end{array}
$$

This corresponds to the solution displayed in Example 5. □

The solution of an ALFP clause can be found in polynomial time in the size of the universe i.e. in the number of groups and capabilities. This complexity is due to a generalisation [35] of a meta-complexity result for Horn Clauses by McAllester [25], which states that:

– The time needed to *compute* a solution is asymptotically the same as the time needed to *check* the validity of an estimate.
– The degree of the complexity polynomial is dominated by one plus the nesting depth of quantifiers occurring in the clause.

Consequently, the complexity bound can sometimes be improved by reformulating the clause to reduce the amount of quantifier nesting. Rather than improving formulae using a general transformation scheme, like the use of *tiling* to reduce quantifier nesting [36], or *automatically estimating* the run-time [32], we take the more pragmatic, and more precise, approach of estimating the run-time empirically [7], and use this as a basis for transforming the clause so as to improve its running time [7].

A result of such an experiment is shown in Figure 4. Here the analysis has been run on a number of processes with the same overall structure where a packet is routed through a grid of $m \times m$ sites. The actual time that the Succinct Solver spends on computing a solution is plotted against the size of the process.

The plot is shown using logarithmic scales on both axes so that a power function shows up as a straight line. A crude least-squares estimate of the degree of the complexity polynomial is displayed in the legend of the plot and we see that the solving times are *linear* in the size of the process being analysed. This is typical for most processes, though the analysis in some cases runs in time that is quadratic in the size of the process.

Remark 12. We already said that for ALFP the time needed to *compute* the least solution is asymptotically the same as the time needed to *check* the acceptability of an estimate. Elaborating on the analogy to type systems explained above, one could then coin the slogan:

> ALFP-based Flow Logic studies a class of simple Type Systems where *type checking* and *type inference* have the same asymptotic complexity.

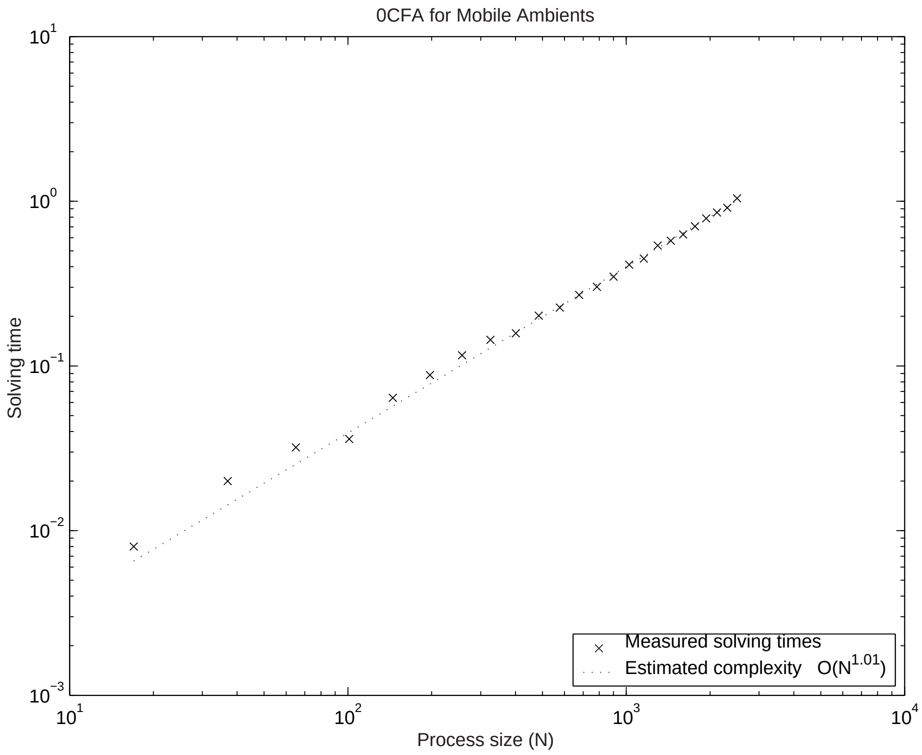


Fig. 4. Estimating the complexity empirically.

In the absence of principal types this is quite different from type systems based on subtyping where *type checking* usually takes polynomial time but where *type inference* often would seem to require non-deterministic polynomial time (in practise exponential time) due to the need to search for the right types. □

2.3 Crossing Control and Opening Control

The analysis not only approximates the hierarchical structure of the ambients but also the access operations that an ambient may possess. This facilitates validating the following security properties [12]:

- *Crossing control:* may an ambient m cross the boundary of another ambient n either by entering it (using in-capabilities) or by exiting it (using out-capabilities)?
- *Opening control:* may an ambient n be dissolved by another ambient m (using open-capabilities)?

In each case we proceed as follows:

- First, we describe the desired property *dynamically*, i.e. we express it using the concepts and notation of the reduction semantics.
- Second, we describe the property *statically*, i.e. we re-express it using the concepts and notation of the static analysis.
- Third, we show the semantic correctness of these formulations: that the static formulation of the property implies the dynamic formulation.
- Finally, we argue that the test can be performed by means of the techniques used for implementing the analysis, which in our case means that the static properties can be determined in polynomial time.

Crossing control. The dynamic notion amounts to saying that an ambient n can cross the ambient n' during the execution of a process P whenever in some reachable configuration n executes the $\mathsf{in}\, n'$ or the $\mathsf{out}\, n'$ capability. This can be reformulated in terms of groups:

Definition 13 (Dynamic notion of crossing). *Ambients of group μ* **can cross** *ambients of group μ' during the execution of P* whenever

1. $P \rightarrow^* Q$,
2. *some ambient n in Q contains an executable capability* $\mathsf{in}\, n'$ *or an executable capability* $\mathsf{out}\, n'$*, and*
3. *n is of group μ and n' is of group μ'.*

We could choose to define the dynamic notion both with and without groups but we shall focus on the former since it more directly relates to the static notion studied below.

The static notion is expressed in terms of the 0CFA analysis. It amounts to saying that ambients of group μ may cross ambients of group μ' during the execution of P whenever the precondition in the clause for in-capabilities is satisfied or the precondition in the clause for out-capabilities is satisfied. To express this in a succinct manner we decide to introduce an"observation predicate", named $\mathcal{D}$ for dynamics, to keep track of the capabilities recorded by $\mathcal{I}$ that may actually execute according to the analysis. We let $\mathcal{D}$ be a mapping $\mathcal{D} : \mathbf{Group} \rightarrow \mathcal{P}(\mathbf{Cap})$ and modify the clauses for in and out to read:

$$\begin{array}{rl} (\mathcal{I},\mathcal{D}) \models^\star_\Gamma \mathsf{in}\, n.\, P \text{ iff} & \mathsf{in}\ \mu \in \mathcal{I}(\star) \ \wedge \ (\mathcal{I},\mathcal{D}) \models^\star_\Gamma P \ \wedge \\ & \forall\ \mu^a, \mu^p : \mathsf{in}\ \mu \in \mathcal{I}(\mu^a) \ \wedge \\ & \qquad\qquad \mu^a \in \mathcal{I}(\mu^p) \ \wedge \\ & \qquad\qquad \mu \in \mathcal{I}(\mu^p) \\ & \qquad\quad \Rightarrow \mu^a \in \mathcal{I}(\mu) \ \wedge \ \mathsf{in}\ \mu \in \mathcal{D}(\mu^a) \\ & \text{where } \mu = \Gamma(n) \end{array}$$

$$\begin{array}{rl} (\mathcal{I},\mathcal{D}) \models^{\star}_{\Gamma} \mathsf{out}\, n.\ P \text{ iff } & \mathsf{out}\ \mu \in \mathcal{I}(\star)\ \wedge\ (\mathcal{I},\mathcal{D}) \models^{\star}_{\Gamma} P\ \wedge \\ & \forall\ \mu^a, \mu^g : \mathsf{out}\ \mu \in \mathcal{I}(\mu^a) \wedge \\ & \qquad\qquad \mu^a \in \mathcal{I}(\mu)\ \wedge \\ & \qquad\qquad \mu \in \mathcal{I}(\mu^g) \\ & \qquad \Rightarrow \mu^a \in \mathcal{I}(\mu^g)\ \wedge\ \mathsf{out}\ \mu \in \mathcal{D}(\mu^a) \\ & \text{where } \mu = \Gamma(n) \end{array}$$

Using the information in the "observation predicate" $\mathcal{D}$ the static notion of what it means for an ambient to cross the boundary of another ambient can be defined as follows:

Definition 14 (Static notion of crossing). *Ambients of group μ* **possibly may cross** *ambients of group μ' during the execution of P* whenever

$$\mathsf{in}\,\mu' \in \mathcal{D}(\mu) \quad \vee \quad \mathsf{out}\,\mu' \in \mathcal{D}(\mu)$$

for the least $\mathcal{I}$ and $\mathcal{D}$ such that $(\mathcal{I},\mathcal{D}) \models^{\star}_{\Gamma} P$.

This static condition is checkable in polynomial time.

Example 15. With respect to Γ and $\mathcal{I}$ as displayed in Example 5 the least estimate for the modified analysis will produce a relation $\mathcal{D}$ that contains exactly the same capabilities as recorded in $\mathcal{I}$; i.e. $\forall \mu : \mathcal{D}(\mu) = \mathcal{I}(\mu) \cap \mathbf{Cap}$. Hence the analysis can be used to validate (where "will never" is the negation of "possibly may"):

- Ambients of group $\mathbb{P}$ possibly may cross ambients in group $\mathbb{S}$; because $\mathsf{in}\,\mathbb{S} \in \mathcal{D}(\mathbb{P})\ \vee\ \mathsf{out}\,\mathbb{S} \in \mathcal{D}(\mathbb{P})$.
- Ambients in group $\mathbb{S}$ will never cross ambients in group $\mathbb{P}$; because $\mathsf{in}\,\mathbb{P} \notin \mathcal{D}(\mathbb{S}) \wedge\ \mathsf{out}\,\mathbb{P} \notin \mathcal{D}(\mathbb{S})$.

It is interesting to observe that a more precise analysis is needed to validate that ambients of group $\mathbb{S}$ will never cross ambients in group $\mathbb{S}$ since we do *not* have that $\mathsf{in}\,\mathbb{S} \notin \mathcal{D}(\mathbb{S}) \wedge \mathsf{out}\,\mathbb{S} \notin \mathcal{D}(\mathbb{S})$. And indeed, we also have $\mathbb{S} \in \mathcal{I}(\mathbb{S})$ indicating that as far as the analysis can see, some ambient of group $\mathbb{S}$ may turn up inside some ambient of group $\mathbb{S}$. □

The correctness of the static test with respect to the dynamic semantics is formally expressed as follows:

Theorem 16 (Crossing Control).

1. If *ambients of group μ* **can cross** *ambients of group μ' during the execution of P* then *ambients of group μ* **possibly may cross** *ambients of group μ' during the execution of P.*

2. If *ambients of group μ* **will never cross** *ambients of group μ' during the execution of P* then *ambients of group μ* **cannot cross** *ambients of group μ' during the execution of P.*

The proposition is a corollary of the subject reduction result (Theorem 9); also note that the second statement is the contrapositive version of the first statement (and hence that they are logically equivalent).

Opening control. The dynamic notion amounts to saying that an ambient n can open the ambient n' during the execution of P whenever some n executes the $\mathsf{open}\, n'$ capability in some reachable configuration. Again we define the notion in terms of groups.

Definition 17 (Dynamic notion of opening). *Ambients of group μ* **can open** *ambients of group μ' during the execution of P* whenever

1. *$P \rightarrow^* Q$,*
2. *some ambient n in Q contains an executable capability $\mathsf{open}\, n'$, and*
3. *n is of group μ and n' is of group μ'.*

The static notion is once again expressed in terms of the 0CFA analysis. It amounts to saying that ambients of group μ may open ambients in group μ' whenever the precondition in the clause for open-capabilities is satisfied. As before we use a modified clause for extracting the "executable" capabilities in $\mathcal{D}$:

$$\begin{array}{rl}(\mathcal{I},\mathcal{D}) \models^{\star}_{\Gamma} \mathsf{open}\, n.\ P \text{ iff} & \mathsf{open}\ \mu \in \mathcal{I}(\star)\ \wedge\ (\mathcal{I},\mathcal{D}) \models^{\star}_{\Gamma} P\ \wedge \\ & \forall\ \mu^p : \mathsf{open}\ \mu \in \mathcal{I}(\mu^p)\ \wedge \\ & \qquad\quad \mu \in \mathcal{I}(\mu^p) \\ & \quad \Rightarrow\ \mathcal{I}(\mu) \subseteq \mathcal{I}(\mu^p) \wedge \mathsf{open}\, \mu \in \mathcal{D}(\mu^p) \\ & \text{where } \mu = \Gamma(n)\end{array}$$

and the static property can be defined accordingly:

Definition 18 (Static notion of opening). *Ambients of group μ* **possibly may open** *ambients of group μ' during the execution of P* whenever

$$\mathsf{open}\, \mu' \in \mathcal{D}(\mu)$$

for the least $\mathcal{I}$ and $\mathcal{D}$ such that $(\mathcal{I},\mathcal{D}) \models^{\star}_{\Gamma} P$.

As before the condition is checkable in polynomial time.

Example 19. With respect to Γ, $\mathcal{I}$ and $\mathcal{D}$ as given in Examples 5 and 15, respectively, the analysis can be used to validate that ambients of group $\mathbb{S}$ possibly may open ambients in group $\mathbb{P}$, because $\mathsf{open}\, \mathbb{P} \in \mathcal{D}(\mathbb{S})$, and that ambients in group $\mathbb{P}$ will never open any ambients, because $\forall \mu : \mathsf{open}\, \mu \notin \mathcal{D}(\mathbb{P})$. □

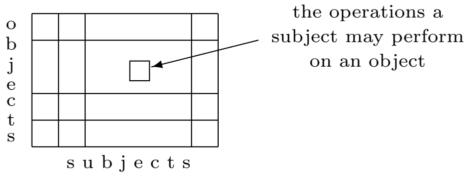
Theorem 20 (Opening Control).

1. If *ambients of group* μ **can open** *ambients in group* μ' *during the execution of* P then *ambients of group* μ **possibly may open** *ambients in group* μ' *during the execution of* P.
2. If *ambients of group* μ **will never open** *ambients in group* μ' *during the execution of* P then *ambients of group* μ **cannot open** *ambients in group* μ' *during the execution of* P.

As before this is a corollary of the subject reduction result (Theorem 9).

3 Discretionary Access Control

The notion of discretionary access control originates from operating systems where it is used to define a reference monitor for governing the access operations (typically read, write and execute) that active subjects (typically programs or users) can perform on passive objects (typically files or external devices). The reference monitor is then implemented as part of the operating system. Although traditionally implemented as access control lists, often based on grouping users into three layers (the user, a group of users, all users), conceptually the specification of access control takes the form of a matrix [22]:



When adapting discretionary access control to Mobile Ambients we should rethink the concepts of subject, object and access operation. It seems very natural to let the access operations be the basic capabilities (in, out and open) of Mobile Ambients since the notions of read, write and execute are indeed the basic operations of a traditional operating system. Then subjects and objects will both be ambients; the subject will be the ambient containing the capability and the object the other ambient involved (typically the one being moved into or being moved out of).

Safe Ambients [24] extend Mobile Ambients to deal with discretionary access control. Since it is not in the distributed nature of Mobile Ambients to have a single global access control matrix it is implemented as access rights, or co-capabilities, placed inside the objects. Syntactically this leads to modifying the syntax of Mobile Ambients as follows:

$$\begin{array}{rll} P ::= & \cdots \text{ as before } \cdots & \\ M ::= & \mathsf{in}\ n \mid \mathsf{out}\ n \mid \mathsf{open}\ n & \text{capabilities} \approx \text{access operations} \\ \mid & \overline{\mathsf{in}}\ n \mid \overline{\mathsf{out}}\ n \mid \overline{\mathsf{open}}\ n & \text{co-capabilities} \approx \text{access rights} \end{array}$$

A transition only takes place if a capability of the subject is matched by a corresponding co-capability in the object:

- If m wants to move into n then n should be willing to let ambients enter; i.e. n must have the co-capability $\overline{\mathsf{in}}\ n$:

$$\overset{m}{\boxed{\mathsf{in}\, n.P \mid Q}} \;\Big|\; \overset{n}{\boxed{\overline{\mathsf{in}}\ n.R \mid S}} \quad\longrightarrow\quad \overset{n}{\boxed{\overset{m}{\boxed{P \mid Q}} \;\Big|\; R \mid S}}$$

- If m wants to move out of n then n should be willing to let ambients leave; i.e. n must have the co-capability $\overline{\mathsf{out}}\ n$:

$$\overset{n}{\boxed{\overset{m}{\boxed{\mathsf{out}\, n.P \mid Q}} \;\Big|\; \overline{\mathsf{out}}\ n.R \mid S}} \quad\longrightarrow\quad \overset{m}{\boxed{P \mid Q}} \;\Big|\; \overset{n}{\boxed{R \mid S}}$$

- If m wants to dissolve n then n should be willing to be dissolved; i.e. n must have the co-capability $\overline{\mathsf{open}}\ n$:

$$\mathsf{open}\, n.P \;\Big|\; \overset{n}{\boxed{\overline{\mathsf{open}}\ n.Q \mid R}} \quad\longrightarrow\quad P \;\mid\; Q \;\mid\; R$$

This amounts to integrating the reference monitor into the semantics i.e. the transition relation.

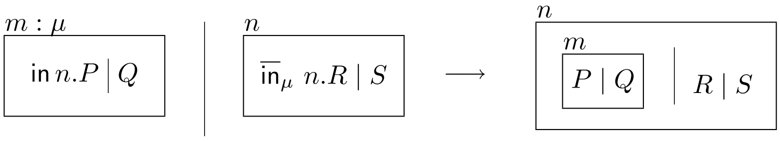
3.1 Syntax and Semantics of Discretionary Ambients

Discretionary Ambients goes one step further in giving an account of discretionary access control that is as refined as illustrated by the access control matrix above. We do so by augmenting co-capabilities with a subscript indicating the group of ambients permitted to perform the corresponding capability:

$$\begin{aligned} P &::= (\nu\mu)P \;\mid\; (\nu n : \mu)P \;\mid\; \mathbf{0} \;\mid\; P_1 \mid P_2 \;\mid !P \;\mid\; n[P] \;\mid\; M.\,P \\ M &::= \mathsf{in}\ n \;\mid\; \mathsf{out}\ n \;\mid\; \mathsf{open}\ n \;\mid\; \overline{\mathsf{in}}_\mu n \;\mid\; \overline{\mathsf{out}}_\mu n \;\mid\; \overline{\mathsf{open}}_\mu n \end{aligned}$$

Hence the basic transitions need to be determined relative to a type environment Γ mapping ambient names to groups; below we write $n : \mu$ to indicate that $\Gamma(n) = \mu$.

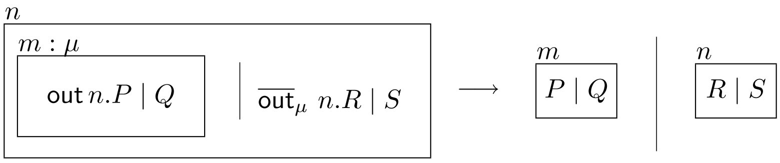
– For the in-capability we have



$m : \mu$
$\mathsf{in}\, n.P \mid Q$
n
$\overline{\mathsf{in}}_\mu\ n.R \mid S$
$\longrightarrow$
n
m
$P \mid Q$
$R \mid S$

so n is willing to let ambients of group μ enter.

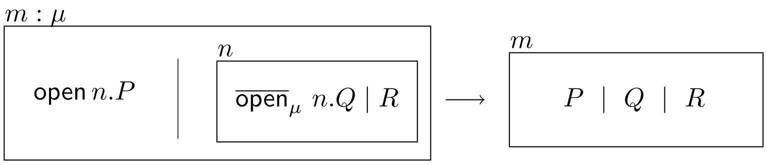
– For the out-capability we have



n
$m : \mu$
$\mathsf{out}\, n.P \mid Q$
$\overline{\mathsf{out}}_\mu\ n.R \mid S$
$\longrightarrow$
m
$P \mid Q$
n
$R \mid S$

so n is willing to let ambients of group μ leave.

– For the open-capability we have



so n is willing to be dissolved within ambients of group μ.

The semantics of Discretionary Ambients is a straightforward extension of the semantics of Mobile Ambients. It consists of the structural congruence relation, $P \equiv Q$, defined in Table 1 and the transition relation, $\Gamma \vdash P \to Q$, defined in Table 4. Here Γ is a type environment mapping names to groups and groups to the special token $\diamond$.

Example 21. We may express the process of Figure 1 in Discretionary Ambients as follows:

$$\mathsf{A}[\mathsf{p}[\mathsf{out\ A.in\ B}.\overline{\mathsf{open}}_{\mathbb{S}}\,\mathsf{p}] \mid \overline{\mathsf{out}}_{\mathbb{P}}\,\mathsf{A}] \mid \mathsf{B}[\overline{\mathsf{in}}_{\mathbb{P}}\,\mathsf{B.open\ p}]$$

where we assume that $\Gamma(\mathsf{A}) = \Gamma(\mathsf{B}) = \mathbb{S}$ and $\Gamma(\mathsf{p}) = \mathbb{P}$ i.e. that the sites A and B are in the group $\mathbb{S}$ and the packet p is in the group $\mathbb{P}$. The packet may move out of A since it has the capability $\mathsf{out\ A}$ and furthermore A grants it the right to do so because it has the co-capability $\overline{\mathsf{out}}_{\mathbb{P}}\,\mathsf{A}$ (and exploiting that p is in the group $\mathbb{P}$). So in the first step the system may evolve into:

$$\mathsf{A}[\,] \mid \mathsf{p}[\mathsf{in\ B}.\overline{\mathsf{open}}_{\mathbb{S}}\,\mathsf{p}] \mid \mathsf{B}[\overline{\mathsf{in}}_{\mathbb{P}}\,\mathsf{B.open\ p}]$$

Now p has the capability to enter B and B has the co-capability to let it do so; the system becomes:

$$\mathsf{A}[\,] \mid \mathsf{B}[\mathsf{p}[\overline{\mathsf{open}}_{\mathbb{S}}\,\mathsf{p}] \mid \mathsf{open\ p}]$$

$$\frac{\Gamma[\mu \mapsto \diamond] \vdash P \to Q}{\Gamma \vdash (\nu\mu)P \to (\nu\mu)Q} \qquad \frac{\Gamma[n \mapsto \mu] \vdash P \to Q}{\Gamma \vdash (\nu n : \mu)P \to (\nu n : \mu)Q} \quad \text{if } \mu \in \mathrm{dom}(\Gamma)$$

$$\frac{\Gamma \vdash P \to Q}{\Gamma \vdash n[P] \to n[Q]} \qquad \begin{array}{l} \Gamma \vdash m[\mathsf{in}\ n.\, P \mid Q] \mid n[\overline{\mathsf{in}}_{\mu} n.\, R \mid S] \\ \quad \to\ n[m[P \mid Q] \mid R \mid S] \qquad \text{if } \Gamma(m) = \mu \end{array}$$

$$\frac{\Gamma \vdash P \to Q}{\Gamma \vdash P \mid R \to Q \mid R} \qquad \begin{array}{l} \Gamma \vdash n[m[\mathsf{out}\ n.\, P \mid Q] \mid \overline{\mathsf{out}}_{\mu} n.\, R \mid S] \\ \quad \to\ m[P \mid Q] \mid n[R \mid S] \qquad \text{if } \Gamma(m) = \mu \end{array}$$

$$\frac{P \equiv P' \quad \Gamma \vdash P' \to Q' \quad Q' \equiv Q}{\Gamma \vdash P \to Q} \qquad \begin{array}{l} \Gamma \vdash m[\mathsf{open}\ n.\, P \mid n[\overline{\mathsf{open}}_{\mu} n.\, Q \mid R]] \\ \quad \to\ m[P \mid Q \mid R] \qquad \text{if } \Gamma(m) = \mu \end{array}$$

Table 4. Transition relation for Discretionary Ambients.

In the final step B has the capability to open p and p grants it the right to do so since B is in the group $\mathbb{S}$ and we then obtain:

$$\mathsf{A}[\,] \mid \mathsf{B}[\,]$$

as the final configuration. □

Remark 22. The classical Mobile Ambients do not support access control: an object has no way of restricting which operations the subjects may perform on it. Safe Ambients [24] allows to model a very rudimentary form of access control as the objects may use the co-capabilities $\overline{\mathsf{in}}\, n$, $\overline{\mathsf{out}}\, n$ and $\overline{\mathsf{open}}\, n$ to control which operations they engage in. However, the possession of one of these co-capabilities gives access to *any* subject with a corresponding capability; there is no way of allowing some subjects but not others to perform the operation. Discretionary Ambients models the more general form of access control corresponding to the classical developments because the co-capabilities put restrictions on the subjects allowed to perform the operations.

The difference can be illustrated for the running example where the access control matrices could be viewed as being:

Safe Ambients		subject		
		A	B	p
object	A	$\overline{\mathsf{out}}$ A	$\overline{\mathsf{out}}$ A	$\overline{\mathsf{out}}$ A
	B	$\overline{\mathsf{in}}$ B	$\overline{\mathsf{in}}$ B	$\overline{\mathsf{in}}$ B
	p	$\overline{\mathsf{open}}$ p	$\overline{\mathsf{open}}$ p	$\overline{\mathsf{open}}$ p

Discretionary Ambients		subject		
		A : $\mathbb{S}$	B : $\mathbb{S}$	p : $\mathbb{P}$
object	A	–	–	$\overline{\mathsf{out}}_{\mathbb{P}}$ A
	B	–	–	$\overline{\mathsf{in}}_{\mathbb{P}}$ B
	p	$\overline{\mathsf{open}}_{\mathbb{S}}$ p	$\overline{\mathsf{open}}_{\mathbb{S}}$ p	–

Note that for Safe Ambients columns must necessarily be equal.

Compared to the classical setting the access control matrices in Discretionary Ambients are much more dynamic structures that may evolve as the process executes. This is due to the fact that co-capabilities vanish once they have been used. If we want to model the classical setting more faithfully we should therefore always use co-capabilities that are individual threads and prefixed with the replication operator: e.g. $!\,\overline{\mathsf{in}}_{\mathbb{P}}\ \mathsf{B}$ will continue to grant subjects of group $\mathbb{P}$ permission to enter the ambient B as many times as needed. □

3.2 Adapting the 0CFA Analysis to Discretionary Ambients

To adapt the 0CFA analysis to deal with Discretionary (or Safe) Ambients we must modify the functionality of $\mathcal{I}$ to record

- as before: which ambient groups may be inside an ambient in group μ,
- as before: which access operations (capabilities) may be possessed by an ambient in group μ (as subject), and
- additionally: which access rights (co-capabilities) may be possessed by an ambient in group μ (as object).

Hence we shall use

$$\mathcal{I} : \mathbf{Group} \rightarrow \mathcal{P}(\mathbf{Group} \cup \mathbf{Cap} \cup \overline{\mathbf{Cap}})$$

where capabilities and co-capabilities are given by:

$$\begin{array}{lll} \text{capabilities} & m \in \mathbf{Cap} & m ::= \mathsf{in}\ \mu \ \mid\ \mathsf{out}\ \mu \ \mid\ \mathsf{open}\ \mu \\ \text{co-capabilities} & \overline{m} \in \overline{\mathbf{Cap}} & \overline{m} ::= \overline{\mathsf{in}}_{\mu'}\ \mu \ \mid\ \overline{\mathsf{out}}_{\mu'}\ \mu \ \mid\ \overline{\mathsf{open}}_{\mu'}\ \mu \end{array}$$

We shall find it useful also to incorporate the observations $\mathcal{D}$ as discussed in Subsection 2.3 and thus have:

$$\mathcal{D} : \mathbf{Group} \rightarrow \mathcal{P}(\mathbf{Cap} \cup \overline{\mathbf{Cap}})$$

Specification of the adapted 0CFA analysis. It is straightforward to adapt the specification of the 0CFA analysis from Section 2.2 to deal with Discretionary Ambients. In particular no changes are needed for the analysis of the composite processes.

For co-capabilities we simply record the presence of the co-capability much as was the case for ambients:

$$\begin{array}{lll} (\mathcal{I},\mathcal{D}) \models^{\star}_{\Gamma} \overline{\mathsf{in}}_{\mu^a}\ n.\ P & \text{iff} & \overline{\mathsf{in}}_{\mu^a}\mu \in \mathcal{I}(\star) \wedge (\mathcal{I},\mathcal{D}) \models^{\star}_{\Gamma} P \\ & & \text{where } \mu = \Gamma(n) \\[2ex] (\mathcal{I},\mathcal{D}) \models^{\star}_{\Gamma} \overline{\mathsf{out}}_{\mu^a}\ n.\ P & \text{iff} & \overline{\mathsf{out}}_{\mu^a}\mu \in \mathcal{I}(\star) \wedge (\mathcal{I},\mathcal{D}) \models^{\star}_{\Gamma} P \\ & & \text{where } \mu = \Gamma(n) \\[2ex] (\mathcal{I},\mathcal{D}) \models^{\star}_{\Gamma} \overline{\mathsf{open}}_{\mu^p}\ n.\ P & \text{iff} & \overline{\mathsf{open}}_{\mu^p}\mu \in \mathcal{I}(\star) \wedge (\mathcal{I},\mathcal{D}) \models^{\star}_{\Gamma} P \\ & & \text{where } \mu = \Gamma(n) \end{array}$$

For capabilities we need to add one conjunct in each precondition that ensures that the capability is matched by a corresponding co-capability:

$$
\begin{array}{l}
(\mathcal{I},\mathcal{D}) \models^{\star}_{\Gamma} \mathsf{in}\, n.\ P \text{ iff } \mathsf{in}\ \mu \in \mathcal{I}(\star) \ \wedge\ (\mathcal{I},\mathcal{D}) \models^{\star}_{\Gamma} P \ \wedge \\
\qquad\qquad\qquad \forall\ \mu^a, \mu^p :\ \mathsf{in}\ \mu \in \mathcal{I}(\mu^a) \ \wedge\ \mu^a \in \mathcal{I}(\mu^p) \ \wedge\ \mu \in \mathcal{I}(\mu^p) \wedge \\
\qquad\qquad\qquad\qquad\quad \overline{\mathsf{in}}_{\mu^a}\mu \in \mathcal{I}(\mu) \qquad \mu \text{ provides the access right to } \mu^a \\
\qquad\qquad\qquad\quad \Rightarrow\ \mu^a \in \mathcal{I}(\mu) \ \wedge\ \mathsf{in}\ \mu \in \mathcal{D}(\mu^a) \wedge \overline{\mathsf{in}}_{\mu^a}\mu \in \mathcal{D}(\mu) \\
\qquad\qquad\qquad \text{where } \mu = \Gamma(n)
\end{array}
$$

$$
\begin{array}{l}
(\mathcal{I},\mathcal{D}) \models^{\star}_{\Gamma} \mathsf{out}\, n.\ P \text{ iff } \mathsf{out}\ \mu \in \mathcal{I}(\star) \ \wedge\ (\mathcal{I},\mathcal{D}) \models^{\star}_{\Gamma} P \ \wedge \\
\qquad\qquad\qquad \forall\ \mu^a, \mu^g : \mathsf{out}\ \mu \in \mathcal{I}(\mu^a) \wedge \mu^a \in \mathcal{I}(\mu) \wedge \mu \in \mathcal{I}(\mu^g) \wedge \\
\qquad\qquad\qquad\qquad\quad \overline{\mathsf{out}}_{\mu^a}\mu \in \mathcal{I}(\mu) \quad \mu \text{ provides the access right to } \mu^a \\
\qquad\qquad\qquad\quad \Rightarrow\ \mu^a \in \mathcal{I}(\mu^g) \wedge \\
\qquad\qquad\qquad\qquad \mathsf{out}\ \mu \in \mathcal{D}(\mu^a) \wedge \overline{\mathsf{out}}_{\mu^a}\mu \in \mathcal{D}(\mu) \\
\qquad\qquad\qquad \text{where } \mu = \Gamma(n)
\end{array}
$$

$$
\begin{array}{l}
(\mathcal{I},\mathcal{D}) \models^{\star}_{\Gamma} \mathsf{open}\, n.\ P \text{ iff } \mathsf{open}\ \mu \in \mathcal{I}(\star) \ \wedge\ (\mathcal{I},\mathcal{D}) \models^{\star}_{\Gamma} P \ \wedge \\
\qquad\qquad\qquad \forall\ \mu^p :\ \mathsf{open}\ \mu \in \mathcal{I}(\mu^p) \ \wedge\ \mu \in \mathcal{I}(\mu^p) \ \wedge \\
\qquad\qquad\qquad\qquad\quad \overline{\mathsf{open}}_{\mu^p}\mu \in \mathcal{I}(\mu) \quad \mu \text{ provides the access right to } \mu^p \\
\qquad\qquad\qquad\quad \Rightarrow\ \mathcal{I}(\mu) \subseteq \mathcal{I}(\mu^p) \wedge \\
\qquad\qquad\qquad\qquad \mathsf{open}\ \mu \in \mathcal{D}(\mu^p) \wedge \overline{\mathsf{open}}_{\mu^p}\mu \in \mathcal{D}(\mu) \\
\qquad\qquad\qquad \text{where } \mu = \Gamma(n)
\end{array}
$$

Here $\mathcal{D}$ records capabilities and co-capabilities whenever they may be executed.

Example 23. Continuing Example 21 we take Γ to be as before (i.e. $\Gamma(\mathsf{p}) = \mathbb{P}$ and $\Gamma(\mathsf{A}) = \Gamma(\mathsf{B}) = \mathbb{S}$) and obtain the following best estimates of $\mathcal{I}$ and $\mathcal{D}$ of the 0CFA given above:

$$
\begin{array}{rcl}
\mathcal{I}(\star) & = & \{\mathbb{S},\ \mathbb{P}\} \\
\mathcal{I}(\mathbb{S}) & = & \{\mathbb{P},\ \mathsf{in}\,\mathbb{S},\ \mathsf{out}\,\mathbb{S},\ \mathsf{open}\,\mathbb{P},\ \overline{\mathsf{in}}_{\mathbb{P}}\ \mathbb{S},\ \overline{\mathsf{out}}_{\mathbb{P}}\ \mathbb{S},\ \overline{\mathsf{open}}_{\mathbb{S}}\ \mathbb{P}\} \\
\mathcal{I}(\mathbb{P}) & = & \{\mathsf{in}\,\mathbb{S},\ \mathsf{out}\,\mathbb{S},\ \overline{\mathsf{open}}_{\mathbb{S}}\ \mathbb{P}\} \\
\\
\mathcal{D}(\star) & = & \emptyset \\
\mathcal{D}(\mathbb{S}) & = & \{\mathsf{open}\,\mathbb{P},\ \overline{\mathsf{in}}_{\mathbb{P}}\ \mathbb{S},\ \overline{\mathsf{out}}_{\mathbb{P}}\ \mathbb{S}\} \\
\mathcal{D}(\mathbb{P}) & = & \{\mathsf{in}\,\mathbb{S},\ \mathsf{out}\,\mathbb{S},\ \overline{\mathsf{open}}_{\mathbb{S}}\ \mathbb{P}\}
\end{array}
$$

The notion of crossing control from Section 2.3 can easily be adapted to the setting of Discretionary Ambients. Now the analysis finds that ambients of group $\mathbb{S}$ will never cross ambients of group $\mathbb{S}$ since $\mathsf{in}\,\mathbb{S} \notin \mathcal{D}(\mathbb{S})$ and $\mathsf{out}\,\mathbb{S} \notin \mathcal{D}(\mathbb{S})$. This is unlike what was the case for the analysis of Mobile Ambients in Example 15. □

Remark 24. Clearly the analysis can be simplified to deal with Safe Ambients, rather than Discretionary Ambients, although with a loss in precision. This is done by "ignoring" the groups in the co-capabilities in the specification of the analysis above. Continuing Example 23 we can express the system in Safe Ambients by omitting the subscripts to the co-capabilities:

$$\mathsf{A}\,[\ \mathsf{p}\,[\ \mathsf{out}\,\mathsf{A}.\ \mathsf{in}\,\mathsf{B}.\ \overline{\mathsf{open}}\ \mathsf{p}\ ]\ \mid\ \overline{\mathsf{out}}\ \mathsf{A}\ ]\ \mid\ \mathsf{B}\,[\ \overline{\mathsf{in}}\ \mathsf{B}.\ \mathsf{open}\,\mathsf{p}\ ]$$

When we apply the (modified) analysis from above to this process, however, the best analysis estimate will resemble the one found for Mobile Ambients in Example 15 and is no longer able to ensure that ambients of group $\mathbb{S}$ cannot cross ambients of group $\mathbb{S}$. □

Correctness of the adapted 0CFA analysis. The correctness of the analysis for Discretionary Ambients is still a "subject reduction" result saying that the validity of an analysis estimate is preserved during execution:

Theorem 25. *If* $(\mathcal{I}, \mathcal{D}) \models^{\star}_{\Gamma} P$ *and* $\Gamma \vdash P \rightarrow^{*} Q$ *then* $(\mathcal{I}, \mathcal{D}) \models^{\star}_{\Gamma} Q$*.*

Implementation of the adapted 0CFA analysis. The Moore Family property still ensures that all processes admit a least analysis estimate:

Theorem 26. *The set* $\{(\mathcal{I}, \mathcal{D}) \mid (\mathcal{I}, \mathcal{D}) \models^{\star}_{\Gamma} P\}$ *is a Moore family.*

Also the implementation in ALFP proceeds exactly as in Section 2.2.

Precision of the adapted 0CFA analysis. The 0CFA analysis of Discretionary Ambients seems to be more precise than the corresponding 0CFA analysis of Mobile Ambients. An occurrence of this phenomenon is Example 23 where the analysis of Discretionary Ambients reveals that ambients of group $\mathbb{S}$ will never cross ambients of $\mathbb{S}$; on the other hand the analysis of the Mobile Ambients version of the same process in Example 15 is not able to give the same precise result.

The better precision can be ascribed to the extra information that co-capabilities give about the behaviour of the process. This extra information allows for additional constraints on the analysis result, which in turn makes the analysis more precise. For example, accumulated errors where one "incorrect element" in the solution gives rise to several more incorrect elements are less likely to occur because it is improbable for an incorrect element to fulfil the extra constraints inferred by the co-capabilities.

Example 27. The analysis gains precision by the way access control is added to the processes. Consider for example the process:

$$\mathsf{a}[\,] \mid \mathsf{b}[\,] \mid \mathsf{c}[\mathsf{b}[\mathsf{in}\ \mathsf{a}]]$$

which is analysed in a type environment where $\Gamma(\mathsf{a}) = \mathbb{A}$, $\Gamma(\mathsf{b}) = \mathbb{B}$, and $\Gamma(\mathsf{c}) = \mathbb{C}$. When analysed with the 0CFA analysis for Mobile Ambients we get the correct but imprecise result $\mathcal{I}_1$ indicating that that b may turn up inside a.

$$
\begin{array}{lll}
\mathcal{I}_1(\star) \ = \{\mathbb{A}, \mathbb{B}, \mathbb{C}\} & \mathcal{I}_2(\star) \ = \{\mathbb{A}, \mathbb{B}, \mathbb{C}\} & \mathcal{I}_3(\star) \ = \{\mathbb{A}, \mathbb{B}, \mathbb{C}\} \\
\mathcal{I}_1(\mathbb{A}) = \{\mathbb{B}\} & \mathcal{I}_2(\mathbb{A}) = \emptyset & \mathcal{I}_3(\mathbb{A}) = \{\overline{\mathsf{in}}_{\mathbb{B}}\mathbb{A}, \mathbb{B}\} \\
\mathcal{I}_1(\mathbb{B}) = \{\mathsf{in}\ \mathbb{A}\} & \mathcal{I}_2(\mathbb{B}) = \{\mathsf{in}\ \mathbb{A}\} & \mathcal{I}_3(\mathbb{B}) = \{\mathsf{in}\ \mathbb{A}\} \\
\mathcal{I}_1(\mathbb{C}) = \{\mathbb{B}\} & \mathcal{I}_2(\mathbb{C}) = \{\mathbb{B}\} & \mathcal{I}_3(\mathbb{C}) = \{\mathbb{B}\}
\end{array}
$$

Suppose that we add access rights to the above process in order *not* to allow b to enter a. In that case, we do not add any co-capabilities and the process above is just viewed as a Discretionary Ambient process. The analysis result $\mathcal{I}_2$ found using the 0CFA analysis of Discretionary Ambients, however, now correctly shows that b cannot show up inside a.

Suppose on the other hand that we add access rights in order to *allow* b to enter a. Then we add the co-capability $\overline{\mathsf{in}}_{\mathbb{B}}\mathsf{a}$ and get the process:

$$\mathsf{a}[\overline{\mathsf{in}}_{\mathbb{B}}\mathsf{a}] \mid \mathsf{b}[\,] \mid \mathsf{c}[\mathsf{b}[\mathsf{in}\ \mathsf{a}]]$$

Now, we get the analysis result $\mathcal{I}_3$ that imprecisely shows that b can turn up in a. We conclude that the additional precision strongly depends on how access rights are added. □

3.3 A Context-Sensitive 1CFA Analysis

In preparation for the study of mandatory access control in the next section we shall now develop a more precise analysis of Discretionary Ambients. Instead of merely recording the *father-son* relationship it takes the grandfather into account and directly records the *grandfather-father-son* relationship by means of a ternary relation.

As before the analysis approximates the behaviour of a process by a *single* abstract configuration that describes all the possible derivatives that the process may have. It distinguishes between the various groups of ambients but not between the individual ambients. Unlike before the analysis is context-sensitive in keeping track of the grandfather relevant for the father-son relationship. Hence the analysis represents the tree structure of the processes by a ternary relation

$$\mathcal{I} : \mathbf{Group} \times \mathbf{Group} \to \mathcal{P}(\mathbf{Group} \cup \mathbf{Cap} \cup \overline{\mathbf{Cap}})$$

so that $\mu_s \in \mathcal{I}\langle\mu_g, \mu_f\rangle$ means that μ_s is a son of μ_f while at the same time μ_f is a son of μ_g. In a similar way the "observation predicate" becomes a ternary relation

$$\mathcal{D} : \mathbf{Group} \times \mathbf{Group} \to \mathcal{P}(\mathbf{Cap} \cup \overline{\mathbf{Cap}})$$

Example 28. For the running example of Example 21 we may use the following definition of $\mathcal{I}$. Here $\top$ is the father of $\star$, i.e. $\top$ is the grandfather of the top-level ambients in the process being analysed. The entries specify the set of sons with a given combination of grandfather and father:

			grandfather		
	$\mathcal{I}$	$\top$	$\star$	$\mathbb{S}$	$\mathbb{P}$
father	$\star$	$\{\mathbb{P},\mathbb{S}\}$			
	$\mathbb{S}$		$\{\mathbb{P}, \mathsf{in}\ \mathbb{S}, \mathsf{out}\ \mathbb{S}, \mathsf{open}\ \mathbb{P},$ $\overline{\mathsf{in}}_{\mathbb{P}}\mathbb{S}, \overline{\mathsf{out}}_{\mathbb{P}}\mathbb{S}, \overline{\mathsf{open}}_{\mathbb{S}}\mathbb{P}\}$		
	$\mathbb{P}$		$\{\mathsf{in}\ \mathbb{S}, \mathsf{out}\ \mathbb{S}, \overline{\mathsf{open}}_{\mathbb{S}}\mathbb{P}\}$	$\{\mathsf{in}\ \mathbb{S}, \mathsf{out}\ \mathbb{S}, \overline{\mathsf{open}}_{\mathbb{S}}\mathbb{P}\}$	

To be more specific, the fragment $\mathsf{p}[\mathsf{out}\ \mathsf{A}.\mathsf{in}\ \mathsf{B}.\overline{\mathsf{open}}_{\mathbb{S}}\,\mathsf{p}]$ will give rise to $\mathsf{in}\ \mathbb{S} \in \mathcal{I}(\mathbb{S},\mathbb{P})$, $\mathsf{out}\ \mathbb{S} \in \mathcal{I}(\mathbb{S},\mathbb{P})$, $\overline{\mathsf{open}}_{\mathbb{S}}\mathbb{P} \in \mathcal{I}(\mathbb{S},\mathbb{P})$ as shown above. We shall come back to $\mathcal{D}$ in Example 29. □

Specification of the 1CFA analysis. The judgement of the analysis takes the form

$$(\mathcal{I},\mathcal{D}) \models_{\Gamma}^{\langle\top,\star\rangle} P$$

and expresses that $\mathcal{I}$ and $\mathcal{D}$ are safe approximations of the configurations that P may evolve into when ambient names are mapped to groups as specified by Γ and when $\top$ and $\star$ are the ambient groups of the grandfather and father, respectively.

Analysis of composite processes. It is rather straightforward to adapt the clauses for analysing composite processes:

$$\begin{array}{lll}
(\mathcal{I},\mathcal{D}) \models_{\Gamma}^{\langle\top,\star\rangle} (\nu n:\mu)P & \text{iff} & (\mathcal{I},\mathcal{D}) \models_{\Gamma[n\mapsto\mu]}^{\langle\top,\star\rangle} P \\
(\mathcal{I},\mathcal{D}) \models_{\Gamma}^{\langle\top,\star\rangle} (\nu\mu)P & \text{iff} & (\mathcal{I},\mathcal{D}) \models_{\Gamma[\mu\mapsto\diamond]}^{\langle\top,\star\rangle} P \\
(\mathcal{I},\mathcal{D}) \models_{\Gamma}^{\langle\top,\star\rangle} \mathbf{0} & \text{iff} & \text{true} \\
(\mathcal{I},\mathcal{D}) \models_{\Gamma}^{\langle\top,\star\rangle} P_1 \mid P_2 & \text{iff} & (\mathcal{I},\mathcal{D}) \models_{\Gamma}^{\langle\top,\star\rangle} P_1 \ \wedge \ (\mathcal{I},\mathcal{D}) \models_{\Gamma}^{\langle\top,\star\rangle} P_2 \\
(\mathcal{I},\mathcal{D}) \models_{\Gamma}^{\langle\top,\star\rangle} !P & \text{iff} & (\mathcal{I},\mathcal{D}) \models_{\Gamma}^{\langle\top,\star\rangle} P \\
(\mathcal{I},\mathcal{D}) \models_{\Gamma}^{\langle\top,\star\rangle} n[P] & \text{iff} & \mu \in \mathcal{I}\langle\top,\star\rangle \ \wedge \ (\mathcal{I},\mathcal{D}) \models_{\Gamma}^{\langle\star,\mu\rangle} P \\
 & & \quad \text{where } \mu = \Gamma(n)
\end{array}$$

The only modification worth observing is the change of ambience in the final rule: the father $\star$ now becomes the grandfather and the ambient μ now becomes the father when analysing the process P.

Analysis of co-capabilities hardly requires any changes:

$$\begin{array}{c}
(\mathcal{I},\mathcal{D}) \models_{\Gamma}^{\langle\top,\star\rangle} \overline{\mathsf{in}}_{\mu} n.\,P \ \text{ iff } \ \overline{\mathsf{in}}_{\mu}\mu' \in \mathcal{I}\langle\top,\star\rangle \wedge (\mathcal{I},\mathcal{D}) \models_{\Gamma}^{\langle\top,\star\rangle} P \\
\text{where } \mu' = \Gamma(n)
\end{array}$$

$$(\mathcal{I},\mathcal{D}) \models_{\Gamma}^{\langle\top,\star\rangle} \overline{\mathsf{out}}_{\mu} n.\,P \;\text{ iff }\; \overline{\mathsf{out}}_{\mu}\mu' \in \mathcal{I}\langle\top,\star\rangle \wedge (\mathcal{I},\mathcal{D}) \models_{\Gamma}^{\langle\top,\star\rangle} P \quad \text{where } \mu' = \Gamma(n)$$

$$(\mathcal{I},\mathcal{D}) \models_{\Gamma}^{\langle\top,\star\rangle} \overline{\mathsf{open}}_{\mu} n.\,P \;\text{ iff }\; \overline{\mathsf{open}}_{\mu}\mu' \in \mathcal{I}\langle\top,\star\rangle \wedge (\mathcal{I},\mathcal{D}) \models_{\Gamma}^{\langle\top,\star\rangle} P \quad \text{where } \mu' = \Gamma(n)$$

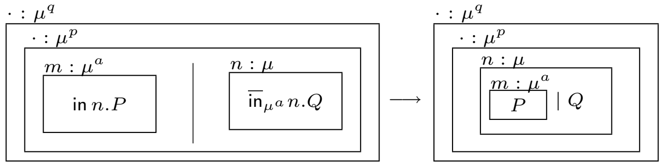
Analysis of capabilities require a number of changes; we begin with the in-capability (explained below):

$$(\mathcal{I},\mathcal{D}) \models_{\Gamma}^{\langle\top,\star\rangle} \mathsf{in}\ n.\,P \text{ iff}$$

$$\left[\begin{array}{l} \mathsf{in}\ \mu \in \mathcal{I}\langle\top,\star\rangle \wedge (\mathcal{I},\mathcal{D}) \models_{\Gamma}^{\langle\top,\star\rangle} P \wedge \\ \forall\mu^a,\mu^p,\mu^q : \left[\begin{array}{l} \mathsf{in}\ \mu \in \mathcal{I}\langle\mu^p,\mu^a\rangle \wedge \\ \mu^a \in \mathcal{I}\langle\mu^q,\mu^p\rangle \wedge \\ \mu \in \mathcal{I}\langle\mu^q,\mu^p\rangle \wedge \\ \overline{\mathsf{in}}_{\mu^a}\mu \in \mathcal{I}\langle\mu^p,\mu\rangle \end{array}\right] \Rightarrow \left[\begin{array}{l} \mu^a \in \mathcal{I}\langle\mu^p,\mu\rangle \wedge \\ \mathcal{I}\langle\mu^p,\mu^a\rangle \subseteq \mathcal{I}\langle\mu,\mu^a\rangle \wedge \\ \mathsf{in}\ \mu \in \mathcal{D}\langle\mu^p,\mu^a\rangle \wedge \\ \overline{\mathsf{in}}_{\mu^a}\mu \in \mathcal{D}\langle\mu^p,\mu\rangle \end{array}\right] \end{array}\right]$$

$$\text{where } \mu = \Gamma(n)$$

Recall that the semantic rule is:



As before the first step is to identify a potential redex:

- $\mathsf{in}\ \mu \in \mathcal{I}\langle\mu^p,\mu^a\rangle$ ensures that the $\mathsf{in}\ n$ capability is present inside some ambient m; here n has group μ and m has group μ^a whereas μ^p is the group of m's father.
- $\mu^a \in \mathcal{I}\langle\mu^q,\mu^p\rangle$ establishes more of m's (i.e. μ^a's) context: its father is μ^p and its grandfather is μ^q.
- $\mu \in \mathcal{I}\langle\mu^q,\mu^p\rangle$ will now ensure that n (i.e. μ) is a sibling of m: it has the same father μ^p and grandfather μ^q.
- $\overline{\mathsf{in}}_{\mu^a}\mu \in \mathcal{I}\langle\mu^p,\mu\rangle$ finally ensures that n (i.e. μ) grants the access right to m (i.e. μ^a) in the context established by the father μ^p of μ.

In addition to identifying possible n's and m's of the semantic rule these steps also identify the context of the redex and thereby rule out some of the confusion that is inevitable when the ambient names are replaced by groups.

Having identified a potential redex in this way the next step is to record in $\mathcal{I}$ the effect of reducing the redex. This is expressed by:

- $\mu^a \in \mathcal{I}\langle\mu^p,\mu\rangle$ records that m (i.e. μ^a) is moved into n (i.e. μ) and this happens only when μ^p is the father.

- $\mathcal{I}\langle\mu^p,\mu^a\rangle \subseteq \mathcal{I}\langle\mu,\mu^a\rangle$ records that everything inside μ^a with grandfather μ^p (the processes P and Q in the semantic rule) as a result of the reduction also may have grandfather μ.

Note that the latter step is considerably more involved than in the 0CFA analysis due to the need to update the context of all entities moved. Finally we need to update the "observation predicate" $\mathcal{D}$:

- $\mathsf{in}\ \mu \in \mathcal{D}\langle\mu^p,\mu^a\rangle$ records that the in-capability was executed.
- $\overline{\mathsf{in}}_{\mu^a}\mu \in \mathcal{D}\langle\mu^p,\mu\rangle$ records that the $\overline{\mathsf{in}}$-co-capability was executed.

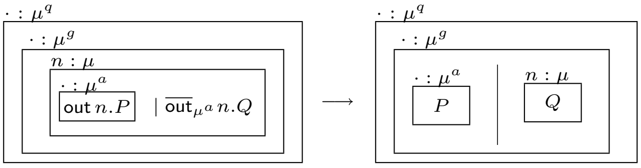
The out-capability follows much the same pattern

$$(\mathcal{I},\mathcal{D}) \models_{\Gamma}^{\langle\top,\star\rangle} \mathsf{out}\ n.\,P \text{ iff}$$

$$\left[\begin{array}{l} \mathsf{out}(\mu) \in \mathcal{I}\langle\top,\star\rangle \wedge (\mathcal{I},\mathcal{D}) \models_{\Gamma}^{\langle\top,\star\rangle} P \wedge \\ \forall\, \mu^a,\mu^g,\mu^q : \left[\begin{array}{l} \mathsf{out}(\mu) \in \mathcal{I}\langle\mu,\mu^a\rangle \wedge \\ \mu^a \in \mathcal{I}\langle\mu^g,\mu\rangle \wedge \\ \mu \in \mathcal{I}\langle\mu^q,\mu^g\rangle \wedge \\ \overline{\mathsf{out}}_{\mu^a}\mu \in \mathcal{I}\langle\mu^g,\mu\rangle \end{array}\right] \Rightarrow \left[\begin{array}{l} \mu^a \in \mathcal{I}\langle\mu^q,\mu^g\rangle \wedge \\ \mathcal{I}\langle\mu,\mu^a\rangle \subseteq \mathcal{I}\langle\mu^g,\mu^a\rangle \wedge \\ \mathsf{out}(\mu) \in \mathcal{D}\langle\mu,\mu^a\rangle \wedge \\ \overline{\mathsf{out}}_{\mu^a}\mu \in \mathcal{D}\langle\mu^g,\mu\rangle \end{array}\right] \end{array}\right]$$

where $\mu = \Gamma(n)$

corresponding to the semantics:



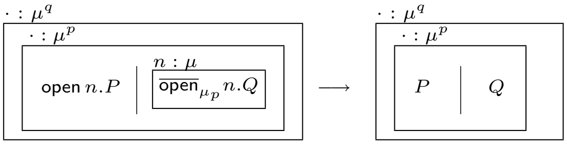
Also the open-capability follows much the same pattern

$$(\mathcal{I},\mathcal{D}) \models_{\Gamma}^{\langle\top,\star\rangle} \mathsf{open}\ n.\,P \text{ iff}$$

$$\left[\begin{array}{l} \mathsf{open}\ \mu \in \mathcal{I}\langle\top,\star\rangle \wedge (\mathcal{I},\mathcal{D}) \models_{\Gamma}^{\langle\top,\star\rangle} P \wedge \\ \forall\, \mu^p,\mu^q : \left[\begin{array}{l} \mathsf{open}\ \mu \in \mathcal{I}\langle\mu^q,\mu^p\rangle \wedge \\ \mu \in \mathcal{I}\langle\mu^q,\mu^p\rangle \wedge \\ \overline{\mathsf{open}}_{\mu^p}\mu \in \mathcal{I}\langle\mu^p,\mu\rangle \end{array}\right] \Rightarrow \left[\begin{array}{l} \mathcal{I}\langle\mu^p,\mu\rangle \subseteq \mathcal{I}\langle\mu^q,\mu^p\rangle \wedge \\ \mathsf{open}\ \mu \in \mathcal{D}\langle\mu^q,\mu^p\rangle \wedge \\ \overline{\mathsf{open}}_{\mu^p}\mu \in \mathcal{D}\langle\mu^p,\mu\rangle \end{array}\right] \end{array}\right]$$

where $\mu = \Gamma(n)$

corresponding to the semantics:



Example 29. Consider the analysis of our running example of Example 21 (again taking $\Gamma(\mathsf{A}) = \Gamma(\mathsf{B}) = \mathbb{S}$ and $\Gamma(\mathsf{p}) = \mathbb{P}$):

$$(\mathcal{I},\mathcal{D}) \models_{\Gamma}^{\langle\top,\star\rangle} \mathsf{A}\,[\ \mathsf{p}\,[\ \mathsf{out\,A.\ in\,B.}\ \overline{\mathsf{open}}_{\mathbb{S}}\ \mathsf{p}\ ]\ \mid\ \overline{\mathsf{out}}_{\mathbb{P}}\ \mathsf{A}\ ]\ \mid\ \mathsf{B}\,[\ \overline{\mathsf{in}}_{\mathbb{P}}\ \mathsf{B.\ open\,p}\ ]$$

This formula will be satisfied when $\mathcal{I}$ is (as in Example 28)

			grandfather		
	$\mathcal{I}$	$\top$	$\star$	$\mathbb{S}$	$\mathbb{P}$
father	$\star$	$\{\mathbb{P},\mathbb{S}\}$			
	$\mathbb{S}$		$\{\mathbb{P}, \mathsf{in}\ \mathbb{S}, \mathsf{out}\ \mathbb{S}, \mathsf{open}\ \mathbb{P}, \overline{\mathsf{in}}_{\mathbb{P}}\mathbb{S}, \overline{\mathsf{out}}_{\mathbb{P}}\mathbb{S}, \overline{\mathsf{open}}_{\mathbb{S}}\mathbb{P}\}$		
	$\mathbb{P}$		$\{\mathsf{in}\ \mathbb{S}, \mathsf{out}\ \mathbb{S}, \overline{\mathsf{open}}_{\mathbb{S}}\mathbb{P}\}$	$\{\mathsf{in}\ \mathbb{S}, \mathsf{out}\ \mathbb{S}, \overline{\mathsf{open}}_{\mathbb{S}}\mathbb{P}\}$	

and $\mathcal{D}$ is given by:

			grandfather		
	$\mathcal{D}$	$\top$	$\star$	$\mathbb{S}$	$\mathbb{P}$
father	$\star$				
	$\mathbb{S}$		$\{\mathsf{in}\ \mathbb{S}, \mathsf{open}\ \mathbb{P}, \overline{\mathsf{in}}_{\mathbb{P}}\mathbb{S}, \overline{\mathsf{out}}_{\mathbb{P}}\mathbb{S}\}$		
	$\mathbb{P}$		$\{\mathsf{in}\ \mathbb{S}\}$	$\{\mathsf{out}\ \mathbb{S}, \overline{\mathsf{open}}_{\mathbb{S}}\mathbb{P}\}$	

One may observe that $\mathcal{D}(\cdots)$ is often a *strict* subset of $\mathcal{I}(\cdots) \cap (\mathbf{Cap} \cup \overline{\mathbf{Cap}})$. This shows that a number of capabilities will never occur in a setting where they are allowed to execute. In particular, even though $\mathcal{I}(\star, \mathbb{P})$ contains $\mathsf{out}\ \mathbb{S}$ as well as $\overline{\mathsf{open}}_{\mathbb{S}}\mathbb{P}$, they are absent in $\mathcal{D}(\star, \mathbb{P})$ and hence cannot execute. □

Correctness of the 1CFA analysis. The semantic correctness of the analysis is expressed by the following subject reduction result:

Theorem 30. *If* $(\mathcal{I}, \mathcal{D}) \models_{\Gamma}^{\langle\top,\star\rangle} P$ *and* $\Gamma \vdash P \to^{*} Q$ *then* $(\mathcal{I}, \mathcal{D}) \models_{\Gamma}^{\langle\top,\star\rangle} Q$.

As before the proof is by induction in the length of the derivation sequence; each step is by induction on the inference in the semantics and uses that structurally congruent processes admit the same analysis results.

Implementation of the 1CFA analysis. The Moore family property ensures that all processes can be analysed and admit a least analysis estimate:

Theorem 31. *The set* $\{(\mathcal{I}, \mathcal{D}) \mid (\mathcal{I}, \mathcal{D}) \models_{\Gamma}^{\langle\top,\star\rangle} P\}$ *is a Moore family.*

Though the analysis is more complex that the 0CFA analysis it is still expressible in ALFP using the encodings described in Section 2.2. Hence, the implementation is again done using the Succinct Solver. However, the time complexity of the calculation of the analysis result for the 1CFA analysis is higher than that of the 0CFA analysis although still within polynomial time. We report in Table 5 on some practical experiments where the time spent for computing the analysis result is expressed in terms of the size N of the process for four scalable test

	A	B	C	D
0CFA	$O(N^{1.03})$	$O(N^{1.03})$	$O(N^{2.32})$	$O(N^{1.22})$
1CFA	$O(N^{1.98})$	$O(N^{2.00})$	$O(N^{3.34})$	$O(N^{2.01})$
Size of **Group**	$O(N^1)$	$O(N^1)$	$O(N^{1/2})$	$O(N^{2/3})$

Table 5. Running times for 0CFA versus 1CFA on four scalable test processes.

processes. The test processes describe a packet being routed though a network of sites with different network topology.

One reason for the higher complexity of the 1CFA is that the size of the analysis estimate for the 1CFA potentially is larger than size of the analysis estimate for the 0CFA by a factor corresponding to the number of groups in **Group** (i.e. the potential number of elements in $\mathbf{Group} \times \mathbf{Group} \to \mathcal{P}(\mathbf{Group} \cup \mathbf{Cap} \cup \overline{\mathbf{Cap}})$ versus $\mathbf{Group} \to \mathcal{P}(\mathbf{Group} \cup \mathbf{Cap} \cup \overline{\mathbf{Cap}})$). The results for the first two test processes, A and B behave exactly as expected. This is also largely the case for the last test process, D, taking into account that the number of groups here is not linear in the size of the process. The odd-one-out is the result for the test process C; here we conjecture that the 1CFA analysis is more costly than expected because the lower number of groups means that many ambients get mixed up, i.e. that many more contexts have occurrences of the same group. Thus the precision of the 1CFA is outweighed by the imprecision of the group information.

Precision of the 1CFA analysis. The 1CFA analysis is more precise than the 0CFA analysis in that it records more of the *context* in which capabilities can be used.

Example 32. Recall the process of Example 27

$$\mathsf{a}[\overline{\mathsf{in}}_{\mathbb{B}}\mathsf{a}] \mid \mathsf{b}[\,] \mid \mathsf{c}[\mathsf{b}[\mathsf{in}\ \mathsf{a}]]$$

where the 0CFA of Discretionary Ambients imprecisely finds that b may enter a (as shown by $\mathcal{I}_3$ of Example 27).

The 1CFA analysis gives rise the least estimate $\mathcal{I}$:

$\mathcal{I}$	$\top$	$\star$	$\mathbb{A}$	$\mathbb{B}$	$\mathbb{C}$
$\star$	$\{\mathbb{A}, \mathbb{B}, \mathbb{C}\}$				
$\mathbb{A}$		$\{\overline{\mathsf{in}}_{\mathbb{B}}\mathbb{A}\}$			
$\mathbb{B}$					$\{\mathsf{in}\ \mathbb{A}\}$
$\mathbb{C}$		$\{\mathbb{B}\}$			

It shows that the 1CFA analysis is able to record that the capability in a is not inside b in a context where b is actually a sibling to a. Thus, the analysis result show that b will not show up in a. □

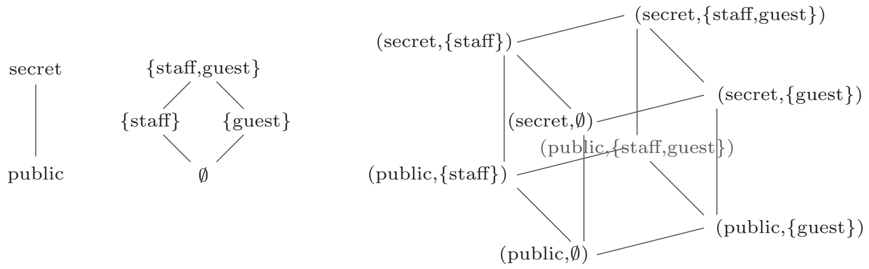


Fig. 5. Example security lattices for confidentiality.

4 Mandatory Access Control

The aim of this section is to show how the Bell-LaPadula [3, 22] and Biba [4, 22] models can be reformulated for Discretionary Ambients and thereby to construct the appropriate *reference monitor semantics*. The first design decision is to assign security levels/integrity levels to the groups rather than the ambients; an ambient then inherits the level of its group. We shall therefore extend the syntax of group introduction to have the form $(\nu\mu^{\ell})P$ meaning that μ has the level ℓ. It is now straightforward to extend the semantics to map groups to security/integrity levels; the key rule is:

$$\frac{\Gamma[\mu \mapsto \ell] \vdash P \to Q}{\Gamma \vdash (\nu\mu^{\ell})P \to (\nu\mu^{\ell})Q}$$

The security/integrity level information is then used in formalising reference monitors in the spirit of the Bell-LaPadula and Biba models; this is covered in the following subsections and takes the form of defining augmented semantics with judgements of the forms $\Gamma \vdash P \twoheadrightarrow Q$. We shall allow to write $\twoheadrightarrow_{\mathsf{BLP}}$ and $\twoheadrightarrow_{\mathsf{Biba}}$ to differentiate between the two choices.

4.1 Confidentiality: the Bell-LaPadula Model

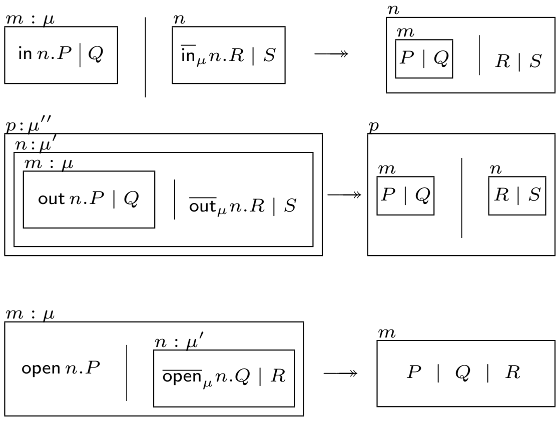
Dynamic formulation of the Bell-LaPadula model. The Bell-LaPadula security model [3, 22] is expressed using an access control matrix and an assignment of security levels to objects and subjects; the security levels are arranged in a lattice $(L, \leq)$ where $\ell_1 \leq \ell_2$ means that ℓ_1 has a lower security level than ℓ_2. The overall aim is then to enforce confidentiality by *preventing information from flowing downwards* from a high security level to a low security level.

To exemplify our interpretation of the Bell-LaPadula model for ambients we use a simple lattice $(L, \leq)$ with $L = \{\mathsf{public}, \mathsf{secret}\}$ and $\mathsf{public} \leq \mathsf{secret}$ as is one of the possibilities illustrated in Figure 5. Conceptually we regard a *secret ambient* as a protective boundary from which no information is allowed escape outwards to a *public ambience*. Thus, "anything" is allowed to happen inside or outside the boundary but restrictions are imposed on which ambients can leave it. This can be effectuated by making a number of restrictions on when operations (i.e. in, out, and open) on ambients are allowed. Informally, we state these restrictions as follows:

- any ambient can *enter* any other ambient;
- an ambient can only *leave* a secret ambient in a secret ambience; and
- a secret ambient can only *be dissolved* in a secret ambience.

The first item reflects that since nothing moves outwards when an in-capability is executed then confidentiality cannot be effected. This is in contrast to the situation for an out-capability where we must prevent movement from a secrets ambient out to a public ambience. Correspondingly, the third condition expresses that secret information inside a secret ambient is not allowed to flow into a public ambience when the secret ambient is opened.

These conditions can be formalised as side conditions to the semantic rules as shown below. The side conditions are modifications of the previous rules in that they incorporate the dynamic checks to be performed by the reference monitor. As an example, the rule for out now contains information about the encapsulating ambient in order to formalise the second condition. The formulation below is specialised to the security lattice $\{\mathsf{public}, \mathsf{secret}\}$ whereas the formulation in Table 6 is sufficiently general to deal with an arbitrary security lattice.



$m : \mu$
$\mathsf{in}\, n.P \mid Q$
n
$\overline{\mathsf{in}}_\mu n.R \mid S$
n
m
$P \mid Q$
$R \mid S$
$p : \mu''$
$n : \mu'$
$m : \mu$
$\mathsf{out}\, n.P \mid Q$
$\overline{\mathsf{out}}_\mu n.R \mid S$
p
m
$P \mid Q$
n
$R \mid S$
$m : \mu$
$\mathsf{open}\, n.P$
$n : \mu'$
$\overline{\mathsf{open}}_\mu n.Q \mid R$
m
$P \mid Q \mid R$

any ambient can enter any other ambient (hence, there is no side condition).

a secret ambient can leave an ambient in a secret ambience:
$\Gamma(\mu) = \mathsf{secret} \Rightarrow$
$\Gamma(\mu'') = \mathsf{secret}$

a secret ambient can be dissolved in a secret ambience:
$\Gamma(\mu') = \mathsf{secret} \Rightarrow$
$\Gamma(\mu) = \mathsf{secret}$

Note that the clause for the out-capability compares the security levels of p and m — and that p and m have a grandfather-son relationship; this is the key reason for why the 1CFA analysis is going to produce better results than the 0CFA analysis.

Example 33. Returning to the running example (expressed in Discretionary Ambients) of Example 21 we first assume that the sites are secret, that the packet is

$$\frac{\Gamma[\mu \mapsto \ell] \vdash P \twoheadrightarrow Q}{\Gamma \vdash (\nu\mu^{\ell})P \twoheadrightarrow (\nu\mu^{\ell})Q} \qquad \frac{\Gamma[n \mapsto \mu] \vdash P \twoheadrightarrow Q}{\Gamma \vdash (\nu n : \mu)P \twoheadrightarrow (\nu n : \mu)Q} \ \text{if } \mu \in \mathrm{dom}(\Gamma)$$

$$\frac{\Gamma \vdash P \to Q}{\Gamma \vdash n[P] \to n[Q]} \qquad \frac{\Gamma \vdash P \to Q}{\Gamma \vdash P \mid R \to Q \mid R} \qquad \frac{P \equiv P' \quad \Gamma \vdash P' \to Q' \quad Q' \equiv Q}{\Gamma \vdash P \to Q}$$

$$\Gamma \vdash m[\mathsf{in}\ n.\, P \mid Q] \mid n[\overline{\mathsf{in}}_{\mu} n.\, R \mid S] \twoheadrightarrow n[m[P \mid Q] \mid R \mid S] \quad \text{if } \Gamma(m) = \mu$$

$$\Gamma \vdash p[n[m[\mathsf{out}\ n.\, P \mid Q] \mid \overline{\mathsf{out}}_{\mu} n.\, R \mid S]] \twoheadrightarrow p[m[P \mid Q] \mid n[R \mid S]] \quad \text{if } \Gamma(m) = \mu \wedge \Gamma(\mu) \leq \Gamma(\Gamma(p))$$

$$\Gamma \vdash m[\mathsf{open}\ n.\, P \mid n[\overline{\mathsf{open}}_{\mu} n.\, Q \mid R]] \twoheadrightarrow m[P \mid Q \mid R] \quad \text{if } \Gamma(m) = \mu \wedge \Gamma(\Gamma(n)) \leq \Gamma(\mu)$$

Table 6. Reference monitor semantics for Bell-LaPadula.

public, and that the overall system is in a public ambience. The packet can move out of the site A because it is public and thus it does not impose the additional conditions on the ambience. The packet can always move into the site B and since it is public it can be opened inside B. So the execution explained in Example 21 is accepted by the reference monitor. This means that the transitions outlined in Example 21 hold for → as well as ↠.

Alternatively, let us assume that the sites are public but that the packet is secret. Now the packet is not allowed to leave A unless the overall system is in a secret ambience. The packet can always move into B but then it cannot be opened because it is secret and the ambience provided by B is indeed public. Thus in this case the reference monitor will "kick in" and prevent the execution from happening. This means that the transitions outlined in Example 21 hold for → but not for ↠. □

Static formulation of the Bell-LaPadula model. Having obtained an approximation to the behaviour of the processes the next step is to formulate the Bell-LaPadula conditions as checks on the analysis results — the idea being that if the analysis result passes these checks then the reference monitor will not intervene in the execution of the process.

The analysis result $(\mathcal{I}, \mathcal{D})$ satisfies the Bell-LaPadula security conditions with respect to the assignment Γ of security levels to groups, written $\mathsf{BLP}_{\Gamma}(\mathcal{I}, \mathcal{D})$, if the following conditions are fulfilled:

$$\forall \mu^a, \mu^g : \left[\exists \mu : \overline{\mathsf{out}}_{\mu^a} \mu \in \mathcal{D}\langle \mu^g, \mu \rangle\right] \Rightarrow \Gamma(\mu^a) \leq \Gamma(\mu^g)$$

$$\forall \mu^p, \mu : \left[\overline{\mathsf{open}}_{\mu^p} \mu \in \mathcal{D}\langle \mu^p, \mu \rangle\right] \Rightarrow \Gamma(\mu) \leq \Gamma(\mu^p)$$

The precondition of the first formula identifies a potential out-redex and the conclusion then requires that the security level of the subject (μ^a) is less than that of the ambience (μ^g) — exactly as required by the reference monitor. The second formula expresses the corresponding condition for the open-redex. In both cases we make good use of the "observation predicate" $\mathcal{D}$ in order to avoid copying large parts of the clauses in the 1CFA analysis.

Example 34. Corresponding to Example 33 let us assume that $\Gamma(\mathbb{S}) = \mathsf{secret}$ and $\Gamma(\mathbb{P}) = \Gamma(\star) = \Gamma(\top) = \mathsf{public}$. We shall check the Bell-LaPadula conditions imposed on the analysis result presented in Example 29. The precondition of the out-capability is only satisfied for $\mu = \mathbb{S}$, $\mu^a = \mathbb{P}$, and $\mu^g = \star$ and the check $\Gamma(\mu^a) \leq \Gamma(\mu^g)$ amounts to $\mathsf{public} \leq \mathsf{public}$, which clearly holds. The precondition for the open-capability is only satisfied for $\mu = \mathbb{P}$ and $\mu^p = \mathbb{S}$ and the check $\Gamma(\mu) \leq \Gamma(\mu^p)$ amounts to $\mathsf{public} \leq \mathsf{secret}$, which also holds. Consequently the analysis result ensures that the reference monitor will not "kick in" and, therefore, its tests can be dispensed with — as was also observed in Example 33.

Alternatively, we may assume that $\Gamma(\mathbb{P}) = \mathsf{secret}$ and $\Gamma(\mathbb{S}) = \Gamma(\star) = \Gamma(\top) = \mathsf{public}$. For the out-capability the check $\Gamma(\mu^a) \leq \Gamma(\mu^g)$ amounts to $\mathsf{secret} \leq \mathsf{public}$ and since this does not hold we cannot guarantee that the reference monitor will not "kick in" — as we reasoned in Example 33. □

The correctness of the static test can be expressed as follows; the result says that we can dispense with the reference monitor if the static checks are fulfilled:

Theorem 35. *Suppose* $\mathsf{BLP}_\Gamma(\mathcal{I}, \mathcal{D})$ *holds for some analysis estimate that satisfies* $(\mathcal{I}, \mathcal{D}) \models_\Gamma^{\langle\top,\star\rangle} P$*; then any execution* $\Gamma \vdash P \rightarrow^* Q$ *can be mimicked as an execution* $\Gamma \vdash P \twoheadrightarrow^*_{\mathsf{BLP}} Q$.

The proof is by induction in the length of the derivation sequence using the subject reduction theorem; each step is by induction on the inference in the semantics; we omit the details.

Efficient implementation of the 1CFA analysis is is as before. It is straightforward to translate $\mathsf{BLP}_\Gamma(\mathcal{I}, \mathcal{D})$ into ALFP and the test can be performed in low polynomial time.

4.2 Integrity: the Biba Model

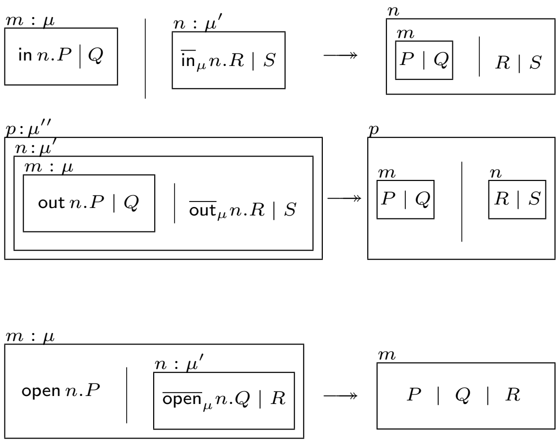
Dynamic formulation of the Biba model. The Biba model for integrity [4, 22] combines the access control matrix with an assignment of integrity levels to subjects and objects; the integrity levels are arranged in a lattice and the overall aim is to *prevent the corruption of high-level entities by low-level entities*:

As a simple example we use the lattice $\{\mathsf{dubious}, \mathsf{trusted}\}$ with $\mathsf{dubious} \leq \mathsf{trusted}$. Again we view ambients as protective boundaries but now we want to prevent *dubious ambients* from moving into *trusted ambients*. We can state this as the following requirements to operations on ambients:

- only trusted ambients can *enter* trusted ambients;
- only trusted ambients can *leave* in a trusted ambience;
- inside a trusted ambient it is only possible to *dissolve* trusted ambients that only contain trusted sub-ambients.

The first item reflects that a trusted ambient will be corrupted if it is entered by a dubious sibling. The second item reflects that a trusted ambient will also be corrupted if it is "entered" by a dubious grandchild. The third item again protects a trusted ambient against corruption by dubious grandchildren but this time as a result of a child being opened. Furthermore, we disallow dubious ambients to be opened in a trusted ambience, since this could unleash "dubious capabilities".

This is formalised by the following extension of the semantics. As before the formulation below is adapted to the security lattice $\{\mathsf{dubious}, \mathsf{trusted}\}$ whereas the formulation in Table 7 is sufficiently general to deal with an arbitrary security lattice. In the clause for open we write $\mathrm{top}(Q \mid R)$ for the ambients occurring at the *top-level* of the process $Q \mid R$; we demand that all of these ambients must have an integrity level that is at least as high as that of the encapsulating ambient.



$m : \mu$
$\mathsf{in}\, n.P \mid Q$
$n : \mu'$
$\overline{\mathsf{in}}_\mu n.R \mid S$
n
m
$P \mid Q$
$R \mid S$
$p : \mu''$
$n : \mu'$
$m : \mu$
$\mathsf{out}\, n.P \mid Q$
$\overline{\mathsf{out}}_\mu n.R \mid S$
p
m
$P \mid Q$
n
$R \mid S$
$m : \mu$
$\mathsf{open}\, n.P$
$n : \mu'$
$\overline{\mathsf{open}}_\mu n.Q \mid R$
m
$P \mid Q \mid R$

only trusted ambients can enter a trusted ambient:
$\Gamma(\mu') = \mathsf{trusted} \;\Rightarrow\; \Gamma(\mu) = \mathsf{trusted}$

a trusted ambience can only be entered by a trusted ambient:
$\Gamma(\mu'') = \mathsf{trusted} \;\Rightarrow\; \Gamma(\mu) = \mathsf{trusted}$

only trusted ambients with trusted sub-ambients can be dissolved inside a trusted ambient:
$\Gamma(\mu) = \mathsf{trusted} \;\Rightarrow$
$\Gamma(\mu') = \mathsf{trusted} \;\wedge$
$\forall p[\cdot] \in \mathrm{top}(Q \mid R) :$
$\Gamma(\Gamma(p)) = \mathsf{trusted}$

Note that also here the ambients of interest have a grandfather-son relationship; once again this motivates our study of the 1CFA analysis.

Example 36. Returning to Example 21 we now assume that sites are dubious, that the packet is trusted, and that the overall system is trusted. The packet can

$$\frac{\mathit{\Gamma}[\mu \mapsto \ell] \vdash P \twoheadrightarrow Q}{\mathit{\Gamma} \vdash (\nu\mu^{\ell})P \twoheadrightarrow (\nu\mu^{\ell})Q} \qquad \frac{\mathit{\Gamma}[n \mapsto \mu] \vdash P \twoheadrightarrow Q}{\mathit{\Gamma} \vdash (\nu n : \mu)P \twoheadrightarrow (\nu n : \mu)Q} \quad \text{if } \mu \in \mathrm{dom}(\mathit{\Gamma})$$

$$\frac{\mathit{\Gamma} \vdash P \to Q}{\mathit{\Gamma} \vdash n[P] \to n[Q]} \qquad \frac{\mathit{\Gamma} \vdash P \to Q}{\mathit{\Gamma} \vdash P \mid R \to Q \mid R} \qquad \frac{P \equiv P' \quad \mathit{\Gamma} \vdash P' \to Q' \quad Q' \equiv Q}{\mathit{\Gamma} \vdash P \to Q}$$

$$\mathit{\Gamma} \vdash m[\mathsf{in}\ n.\,P \mid Q] \mid n[\overline{\mathsf{in}}_{\mu} n.\,R \mid S] \twoheadrightarrow n[m[P \mid Q] \mid R \mid S] \quad \begin{array}{l} \text{if } \mathit{\Gamma}(m) = \mu\ \wedge \\ \mathit{\Gamma}(\mathit{\Gamma}(n)) \leq \mathit{\Gamma}(\mu) \end{array}$$

$$\mathit{\Gamma} \vdash p[n[m[\mathsf{out}\ n.\,P \mid Q] \mid \overline{\mathsf{out}}_{\mu} n.\,R \mid S]] \twoheadrightarrow p[m[P \mid Q] \mid n[R \mid S]] \quad \begin{array}{l} \text{if } \mathit{\Gamma}(m) = \mu\ \wedge \\ \mathit{\Gamma}(\mathit{\Gamma}(p)) \leq \mathit{\Gamma}(\mu) \end{array}$$

$$\mathit{\Gamma} \vdash m[\mathsf{open}\ n.\,P \mid n[\overline{\mathsf{open}}_{\mu} n.\,Q \mid R]] \twoheadrightarrow m[P \mid Q \mid R] \quad \begin{array}{l} \text{if } \mathit{\Gamma}(m) = \mu\ \wedge \\ \mu \leq \mathit{\Gamma}(n)\ \wedge \\ \forall p[\cdot] \in \mathrm{top}(Q \mid R) : \\ \quad \mathit{\Gamma}(\mu) \leq \mathit{\Gamma}(\mathit{\Gamma}(p)) \end{array}$$

Table 7. Reference monitor semantics for Biba.

move out of A because the overall system is trusted and it can now move into B because the sites are dubious. Since the site is dubious there is no problem opening the packet although it is trusted. So the execution of Example 21 will be accepted by the reference monitor. This means that the transitions outlined in Example 21 hold for → as well as ↠.

Alternatively, assume that the sites are trusted but that the packet as well as the overall system are dubious. Then the reference monitor will prevent the packet from entering the site B as it will corrupt its integrity. This means that the transitions outlined in Example 21 hold for → but not for ↠. □

Static formulation of the Biba model. The analysis result $(\mathcal{I}, \mathcal{D})$ satisfies the Biba integrity condition with respect to the assignment $\mathit{\Gamma}$ of integrity levels to groups, written $\mathsf{Biba}_{\mathit{\Gamma}}(\mathcal{I}, \mathcal{D})$, if the following conditions are fulfilled:

$$\forall \mu, \mu^a :\ \left[\exists \mu^p : \overline{\mathsf{in}}_{\mu^a} \mu \in \mathcal{D}\langle \mu^p, \mu \rangle\right] \Rightarrow \mathit{\Gamma}(\mu) \leq \mathit{\Gamma}(\mu^a)$$

$$\forall \mu^a, \mu^g :\! \left[\exists \mu : \overline{\mathsf{out}}_{\mu^a} \mu \in \mathcal{D}\langle \mu^g, \mu \rangle\right] \Rightarrow \mathit{\Gamma}(\mu^g) \leq \mathit{\Gamma}(\mu^a)$$

$$\begin{array}{ll} \forall \mu^p, \mu : & [\overline{\mathsf{open}}_{\mu^p} \mu \in \mathcal{D}\langle \mu^p, \mu \rangle \Rightarrow \\ & \qquad [\mathit{\Gamma}(\mu^p) \leq \mathit{\Gamma}(\mu) \wedge \forall \mu^c : \mu^c \in \mathcal{I}(\mu^p, \mu) \Rightarrow \mathit{\Gamma}(\mu^p) \leq \mathit{\Gamma}(\mu^c)]] \end{array}$$

Again the preconditions express the presence of a potential redex and the conclusion then imposes the relevant integrity constraint of the reference monitor. Note that the top-level ambients (μ^c) occurring inside the subject (μ) easily can be accessed using the relation $\mathcal{I}$.

Example 37. Corresponding to Example 36 let us assume that $\Gamma(\mathbb{S}) = \mathsf{dubious}$ and $\Gamma(\mathbb{P}) = \Gamma(\star) = \Gamma(\top) = \mathsf{trusted}$ and let us check the Biba conditions on the analysis result of Example 29. The precondition for the in-capability is only satisfied for $\mu = \mathbb{S}$, $\mu^a = \mathbb{P}$, and $\mu^p = \star$ and the check $\Gamma(\mu) \leq \Gamma(\mu^a)$ amounts to $\mathsf{dubious} \leq \mathsf{trusted}$, which clearly holds. The precondition for the out-capability is only satisfied for $\mu^a = \mathbb{P}$, $\mu = \mathbb{S}$, and $\mu^g = \star$ and the check $\Gamma(\mu^g) \leq \Gamma(\mu^a)$ amounts to $\mathsf{trusted} \leq \mathsf{trusted}$, which also holds. The precondition for the open-capability only holds for $\mu = \mathbb{P}$ and $\mu^p = \mathbb{S}$. This leads to the two requirements that $\mathsf{dubious} \leq \mathsf{trusted}$ and that the universally quantified implication must hold for these values of μ and μ^p. The latter trivially holds since there exists no μ^c to fulfil the precondition reflecting that $\mathbb{P}$ never contains any sub-ambients. Consequently the analysis result ensures that the reference monitor will never "kick in" as we already observed in Example 36.

Alternatively we may assume that $\Gamma(\mathbb{S}) = \mathsf{trusted}$ but $\Gamma(\mathbb{P}) = \Gamma(\star) = \Gamma(\top) = \mathsf{dubious}$. For the in-capability the check $\Gamma(\mu) \leq \Gamma(\mu^a)$ amounts to $\mathsf{trusted} \leq \mathsf{dubious}$ and since this does not hold we cannot guarantee that the reference monitor will not "kick in" — as we already observed in Example 36. □

The correctness of the static test can be expressed as follows; the result says that we can dispense with the reference monitor if the static checks are fulfilled:

Theorem 38. *Suppose* $\mathsf{Biba}_\Gamma(\mathcal{I}, \mathcal{D})$ *holds for some analysis estimate that satisfies* $(\mathcal{I}, \mathcal{D}) \models_\Gamma^{\langle\top,\star\rangle} P$*; then any execution* $\Gamma \vdash P \rightarrow^* Q$ *can be mimicked as an execution* $\Gamma \vdash P \twoheadrightarrow^*_{\mathsf{Biba}} Q$*.*

The proof is by induction on the length of the derivation sequence using the subject reduction theorem; each step is by induction on the inference in the semantics; we omit the details.

Efficient implementation is as before: translating $\mathsf{Biba}_\Gamma(\mathcal{I}, \mathcal{D})$ into ALFP, the test can be performed in low polynomial time.

Remark 39. The static tests for Bell-LaPadula and Biba could also be phrased using the father-son relations found by the 0CFA. Since the tests are of a grandfather-child nature the 0CFA is, however, likely to be too imprecise. Another approach to improving the simple father-son analysis is considered by Braghin, Cortesi, and Focardi in [6] (for the classical Mobile Ambients with labels). Their idea is to extend the analysis with a third component holding the *security level* of the grandfather. While their analysis will in general be more precise than using our 0CFA it is coarser than the one developed here using the 1CFA because the security information of a grandfather may identify a rather large superset of the set of grandfathers possible.

Related work of studying mandatory access control within ambient calculi has also been done by Bugliesi, Castanga, and Crafa [9]. They interpret access control in an ambient setting as access to *communication* between ambients rather than *mobility* as we do. This leads to the definition of a new calculus called Boxed Ambients (see Section 5), which extends the communication primitives of the original Mobile Ambient calculus. Their main result is a type system which checks that communication does not violate a given access policy. The type system primarily builds on the *exchange types* of [14] (see Remark 50) and, as such, is quite far from our analysis which tracks mobility.

More recently, the same authors have studied *information flow* in a variant of Boxed Ambients [21]. They partition information (i.e. names and capabilities) into high and low security levels and define a type system which impose *access control* on low level processes. Next, they define processes to be contextually equivalent whenever they exhibit a certain kind of communication out of *low level* ambients. Finally, they show that a well-typed low level process is equivalent to itself composed with any well-typed high level process. Thus, a low level process cannot observe the difference between running on its own or running together with a well-typed high level process. Thereby they ensure that a low level process cannot *deduce* anything about well-typed high level processes. As the authors themselves point out, the requirement for high level processes to be well-typed is rather strict since it is not ensured that every process can be typed. Therefore, the high-level processes must be known so the method only applies to closed systems. □

5 Cryptographic Protocols

Mobile Ambients as originally introduced in [13] also admit communication primitives. However, rather than following full-fledged channel-based communication as in the π-calculus the designers opted for a more limited form of communication where each ambient has a mailbox that allows both ambient names as well as capabilities to be communicated. In this way all communication is local between the top-level processes of an ambient and one achieves long distance communication by a combination of local communication and movement within the ambient hierarchy. Also they opted for asynchronous rather than synchronous communication. In the case of monadic input and output the asynchronous[2] communication primitives are:

$\langle M \rangle$	outputs the message M asynchronously to the local mailbox;
$(x).P$	inputs a message from the mailbox, binds it to x and continues as P.

Boxed Ambients [9, 8] takes the view that ambients should not only be allowed to communicate locally but also with their children (but not grandchildren) and parents (but not grandparents). In the monadic calculus the new communication primitives are

[2] To obtain synchronous communication we should write $\langle M \rangle.P$ and modify the semantics.

- $\langle M\rangle^{\uparrow}$ outputs a message to the mailbox of the parent;
- $\langle M\rangle^{\circ}$ outputs a message to the local mailbox;
- $\langle M\rangle^{n}$ outputs a message to the mailbox of a child named n;
- $(x)^{\uparrow}.P$ inputs a message from the mailbox of the parent;
- $(x)^{\circ}.P$ inputs a message from the local mailbox;
- $(x)^{n}.P$ inputs a message from the mailbox of a child named n.

Example 40. Boxed Ambients are well suited for expressing *perfect symmetric* cryptography although there is no explicit cryptographic primitives. We shall code symmetric keys as names and introduce them using restriction; the perfect nature of the cryptography is then due to the semantics of restriction, $(\nu n : \mu)P$, that ensures that the name n introduced is distinct from all other names whether already introduced or yet to be introduced. In this model even a brute force attack cannot succeed.

A plain-text message msg encrypted under a key K is then coded as

$$\mathsf{K}[\langle \mathsf{msg}\rangle^{\circ}]$$

whereas decrypting a ciphertext cph under the key K is coded as:

$$\mathsf{cph} \mid (\mathsf{x})^{\mathsf{K}}.\cdots \mathsf{x} \cdots$$

Here the decryption only succeeds if indeed cph contains a top-level ambient of the form $\mathsf{K}[\langle \mathsf{msg}\rangle^{\circ}]$. If the encrypted message needs to survive for later decryption it can be "protected" from destruction by placing a replication operator (!) in front of it. □

5.1 Syntax and Semantics of Boxed Ambients

For definition of the syntax of Boxed Ambients we revert to Mobile Ambients as explained in Section 2 and add polyadic communication.

Syntax. As in other presentations of ambients we make a distinction between names (introduced by restrictions) and variables (introduced by input); in our view, this distinction adds clarity both to the semantics and to the analysis. We shall therefore find it helpful to introduce a new syntactic category N of namings that can be both variables and names and to use namings where names where used before. Furthermore we introduce an auxiliary syntactic category η for the communication direction. The syntax then reads:

$$\begin{aligned} P &::= (\nu\mu)P \ \mid \ (\nu n : \mu)P \ \mid \ \mathbf{0} \ \mid \ P_1 \mid P_2 \ \mid !P \ \mid \ N[P] \ \mid \ M.\,P \ \mid \\ &\quad\ \langle M_1, \cdots, M_k\rangle^{\eta} \mid (x_1, \cdots, x_k)^{\eta}.\,P \\ M &::= \mathsf{in}\ N \ \mid \ \mathsf{out}\ N \ \mid \ N \\ N &::= n \ \mid \ x \\ \eta &::= N \mid \uparrow \mid \ \circ \end{aligned}$$

We follow the designers of Boxed Ambients in *not* including the open-capability. For simplicity of presentation we have not allowed the formation of composite capabilities, i.e., we disallow the nil capability ϵ and the concatenation $M_1.M_2$ as would be needed for communicating complete routes along which movement could take place; most of the development would work for these extensions as well but the actual implementation would be more complex.

Semantics. The semantic changes are rather minor with respect to the specification of Tables 1 and 2. For the structural congruence we simply need to add the rule:

$$P \equiv Q \;\Rightarrow\; (x_1, \cdots, x_k)^\eta.\, P \equiv (x_1, \cdots, x_k)^\eta.\, Q$$

For the transition relation of Table 2 we add a number of rules for communication. The rules are depicted in their monadic version below, i.e. where only one message is communicated at a time. The polyadic version of the rules are summarised in Table 8 where $P\{x_1 \leftarrow M_1\} \cdots \{x_k \leftarrow M_k\}$ denotes P with M_i substituted for x_i with the usual α-renaming in order to avoid capturing free names in the M_i. We shall only apply the transition relation to closed processes, i.e. processes without any free variables, and hence M_i contains no variables. Consequently we shall dispense with α-renaming of variables (since this simplifies the specification of the 0CFA analysis).

First we have local communication, which takes place between any two sibling processes

$$\langle M\rangle^\circ \mid (x)^\circ.\, P \quad \longrightarrow \quad P\{x \leftarrow M\}$$

and binds the value of the message M to the variable x in the receiving process P. Next we add the following rules for output to a child, either by explicitly naming the child (and using the child's mailbox for the exchange of the message)

$$\langle M\rangle^n \quad \Bigg| \quad \overset{n}{\boxed{\;(x)^\circ.\, P \mid Q\;}} \qquad \longrightarrow \qquad \overset{n}{\boxed{P\{x \leftarrow M\} \mid Q}}$$

or anonymously (using the enclosing ambients mailbox for the exchange of the message)

$$\langle M\rangle^\circ \quad \Bigg| \quad \overset{n}{\boxed{\;(x)^\uparrow.\, P \mid Q\;}} \qquad \longrightarrow \qquad \overset{n}{\boxed{P\{x \leftarrow M\} \mid Q}}$$

Finally, we add the following rules for output to a parent

$$\begin{array}{lcl}
\langle M_1, \cdots, M_k\rangle^\circ \mid (x_1, \cdots, x_k)^\circ.\, P & \rightarrow & P\{x_1 \leftarrow M_1\}\cdots\{x_k \leftarrow M_k\} \\
\langle M_1, \cdots, M_k\rangle^n \mid n\,[(x_1, \cdots, x_k)^\circ.\, P \mid Q\,] & \rightarrow & n\,[P\{x_1 \leftarrow M_1\}\cdots\{x_k \leftarrow M_k\} \mid Q\,] \\
\langle M_1, \cdots, M_k\rangle^\circ \mid n\,[(x_1, \cdots, x_k)^\uparrow.\, P \mid Q\,] & \rightarrow & n\,[P\{x_1 \leftarrow M_1\}\cdots\{x_k \leftarrow M_k\} \mid Q\,] \\
(x_1, \cdots, x_k)^\circ.\, P \mid n\,[\langle M_1, \cdots, M_k\rangle^\uparrow \mid R\,] & \rightarrow & P\{x_1 \leftarrow M_1\}\cdots\{x_k \leftarrow M_k\} \mid n\,[R] \\
(x_1, \cdots, x_k)^n.\, P \mid n\,[\langle M_1, \cdots, M_k\rangle^\circ \mid R\,] & \rightarrow & P\{x_1 \leftarrow M_1\}\cdots\{x_k \leftarrow M_k\} \mid n\,[R]
\end{array}$$

Table 8. Transition relation for Boxed Ambients. Additions to Table 2.

$$(x)^\circ.\, P \;\Big|\; \overset{n}{\boxed{\langle M\rangle^\uparrow \mid R}} \quad\longrightarrow\quad P\{x \leftarrow M\} \;\Big|\; \overset{n}{\boxed{R}}$$

and

$$(x)^n.\, P \;\Big|\; \overset{n}{\boxed{\langle M\rangle^\circ \mid R}} \quad\longrightarrow\quad P\{x \leftarrow M\} \;\Big|\; \overset{n}{\boxed{R}}$$

In particular, we *do not* have a rule for output to grandchildren such as:

$$\langle M_1, \cdots, M_k\rangle^n \mid n\,[m\,[(x_1, \cdots, x_k)^\uparrow.\, P \mid Q\,] \mid R\,] \not\rightarrow \cdots$$

Instead communication between grandparent and its grandchildren will have to be forwarded e.g. as done by m below (for monadic output of a message M):

$$\langle M\rangle^m \mid m[(x)^\circ.\, \langle x\rangle^\circ \mid n[(x)^\uparrow. \cdots x \cdots]]$$

Example 41. Continuing Example 40 we have that

$$\mathsf{K}[\langle\mathsf{msg}\rangle^\circ] \mid (\mathsf{x})^{\mathsf{K}}. \cdots \mathsf{x} \cdots \quad\rightarrow\quad \mathsf{K}[\quad] \mid \cdots \mathsf{msg} \cdots$$

showing that after the decryption an empty message, $\mathsf{K}[\;]$, is left behind. □

Example 42. Boxed Ambients allows to code a package moving around on a network where it communicates the name of a new ambient to be created at the destination:

$$\begin{array}{l}
\mathsf{A}\,[\;\mathsf{p}\,[\;\mathsf{out\,A}.\;\mathsf{in\,B}.\;\langle\mathsf{C}\rangle^\uparrow\;]]\;\mid\;\mathsf{B}\,[\;(\mathsf{x})^\circ.\mathsf{x}\,[\;]] \\
\rightarrow \mathsf{A}\,[\;\;]\;\mid \mathsf{p}\,[\;\mathsf{in\,B}.\;\langle\mathsf{C}\rangle^\uparrow\;]\;\mid\;\mathsf{B}\,[\;(\mathsf{x})^\circ.\mathsf{x}\,[\;]] \\
\rightarrow \mathsf{A}\,[\;\;]\;\mid\;\mathsf{B}\,[\;\mathsf{p}\,[\;\langle\mathsf{C}\rangle^\uparrow\;]\;\mid\;(\mathsf{x})^\circ.\mathsf{x}\,[\;]] \\
\rightarrow \mathsf{A}\,[\;\;]\;\mid\;\mathsf{B}\,[\;\mathsf{p}\,[\;\;]\;\mid\;\mathsf{C}\,[\;\;]\,]
\end{array}$$

This example illustrates the usefulness of being able to communicate between adjacent layers and why this reduces (if not obviates) the need for the open-capability. □

5.2 Cryptographic Protocols in Boxed Ambients

Boxed Ambients seems rather well suited for expressing a number of cryptographic protocols. In this subsection we consider a number of protocols that involve a server S and agents (or principals) A and B.

Agents present themselves with their name and frequently there is the need to find the corresponding key. In traditional programming languages one might have an array that is indexed with the name and that produces the key, i.e. the key to be used by the server for encryption or decryption of messages from the agent. In Boxed Ambients it is natural to represent the "array" as a process of the form

$$\mathsf{KeyTable} \quad = \quad \mathsf{n}_1[!\langle \mathsf{K}_1\rangle^\circ] \quad | \quad \cdots \quad | \quad \mathsf{n}_\mathsf{m}[!\langle \mathsf{K}_\mathsf{m}\rangle^\circ]$$

corresponding to the name of the principal n_i being mapped to the key K_i. We use replication to ensure that the "array" can be queried any number of times. Hence, whenever a process performs the action

$$\mathsf{KeyTable} \quad | \quad (\mathsf{y}_\mathsf{K})^{\mathsf{x}_\mathsf{n}}. \cdots \mathsf{y}_\mathsf{K} \cdots$$

it will obtain the key y_K corresponding to the agent x_n.

Occasionally there is a need to test that two random numbers are equal before proceeding. In traditional programming languages one would test the equality of n and m using a conditional. In Mobile Ambients the traditional coding trick is to create an ambient, $\mathsf{n}[\quad]$ and then let an open-capability, $\mathsf{open\,m}$, guard the then-branch (ignoring the else-branch). In Boxed Ambients we do not have the open-capability and hence will use communication: we create an ambient, $\mathsf{n}[\langle\rangle^\circ]$, that performs a local nullary output and then let an input, $()^\mathsf{m}.\cdots$, guard the then-branch. In other words

$$\mathsf{n}[\langle\rangle^\circ] \quad | \quad ()^\mathsf{m}.\,P$$

will block the execution of P unless n equals m.

Whenever a principal sends a message to another principal it would be natural to encode the message in an anonymous packet (e.g p). Often the message consists of some public names together with a message encrypted by some key (e.g. K) and consisting of some secret names. One could then send the pair $(\mathsf{public}, \mathsf{secret})$ from A to B by means of the packet:

$$\mathsf{p}[\mathsf{out\,A}.\mathsf{in\,B}.(\langle\mathsf{public}\rangle^\circ \mid \mathsf{K}[\mathsf{out\,p}.\langle\mathsf{secret}\rangle^\circ])]$$

To avoid an overly heavy coding we shall generally prefer to dispense with the anonymous packet and instead reuse the cryptographic key. We therefore send $(\mathsf{public}, \mathsf{secret})$ from A to B by means of:

$$\mathsf{K}[\mathsf{out\,A}.\mathsf{in\,B}.(\langle\mathsf{public}\rangle^\uparrow \mid \langle\mathsf{secret}\rangle^\circ)]$$

Once the capabilities $\mathsf{out\,A}$ and $\mathsf{in\,B}$ have been executed the enclosing ambient will have access to the public parts of K without knowing the key, e.g.

$$\mathsf{K}[\langle\mathsf{public}\rangle^\uparrow \mid \langle\mathsf{secret}\rangle^\circ] \mid (\mathsf{x})^\circ.\,\mathsf{x} \longrightarrow \mathsf{K}[\langle\mathsf{secret}\rangle^\circ] \mid \mathsf{public}$$

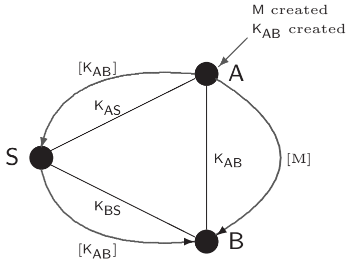


Fig. 6. Wide Mouthed Frog protocol.

In order to get hold of the secret parts of the message the enclosing ambient needs knowledge of the key K:

$$\mathsf{K}[\langle\mathsf{public}\rangle^{\uparrow} \mid \langle\mathsf{secret}\rangle^{\circ}] \mid (\mathsf{x})^{\mathsf{K}}.\,\mathsf{x} \longrightarrow \mathsf{K}[\langle\mathsf{public}\rangle^{\uparrow}] \mid \mathsf{secret}$$

Wide Mouthed Frog. We consider here the Wide Mouthed Frog protocol originally described in [10] in the simplified version of [1] where agents A and B are given together with a trusted server S. Also secret master keys are in place between the server and the agents: $\mathsf{K_{AS}}$ is known only to A and S, and $\mathsf{K_{BS}}$ is known only to B and S. Hence the $\mathsf{KeyTable}$ to be used in the server is

$$\mathsf{KeyTable} \quad = \quad \mathsf{A}[!\langle\mathsf{K_{AS}}\rangle^{\circ}] \quad \mid \quad \mathsf{B}[!\langle\mathsf{K_{BS}}\rangle^{\circ}]$$

The purpose of the protocol is first to exchange a secret session key $\mathsf{K_{AB}}$ for use between A and B, and then to communicate a secret message M using the session key. The protocol depicted in Figure 6 begins with A creating the session key $\mathsf{K_{AB}}$ and the message M. Next A forwards the key to S, encoded with $\mathsf{K_{AS}}$, thereby asking S to forward it to B. The key is forwarded to B, encoded with $\mathsf{K_{BS}}$. At this point B is ready to receive the message M communicated by A, this time encrypted with the secret session key $\mathsf{K_{AB}}$. The classical way of presenting protocols (see e.g. [18]) is to write a narration where the messages of the protocol are listed in order. For each message the principals involved in the message exchange are given along with the content of the message. For the Wide Mouthed Frog protocol this looks as follows (where we write $\mathsf{K}[\,\mathsf{M}\,]$ for the message M encrypted under the key K):

$$\begin{array}{lll} 1. & \mathsf{A} \rightarrow \mathsf{S}: & \mathsf{A}, \mathsf{K_{AS}}[\,\mathsf{B}, \mathsf{K_{AB}}\,] \\ 2. & \mathsf{S} \rightarrow \mathsf{B}: & \mathsf{K_{BS}}[\,\mathsf{A}, \mathsf{K_{AB}}\,] \\ 3. & \mathsf{A} \rightarrow \mathsf{B}: & \mathsf{K_{AB}}[\,\mathsf{M}\,] \end{array}$$

In more detail the steps are as follows (writing Alice for A, Bob for B and Server for S as is customary when discussing protocols):

1. Alice generates a new random session key ($\mathsf{K_{AB}}$)

$$(\nu\, \mathsf{K_{AB}} : \mathbb{K}_{\mathsf{AB}})$$

and then sends her name (A), Bobs name (B) and the session key ($\mathsf{K_{AB}}$) to the Server (S) encrypted by her master key $\mathsf{K_{AS}}$:

$$\mathsf{K_{AS}}[\mathsf{out\,A.\ in\,S.}\ (\langle \mathsf{A}\rangle^{\uparrow} \mid \langle \mathsf{B}, \mathsf{K_{AB}}\rangle^{\circ})\ ]$$

As discussed above, Alice's name (A) can be received by any enclosing ambient once $\mathsf{out\,A.\,in\,S}$ has been executed whereas Bob's name and the session key (B, $\mathsf{K_{AB}}$) can only be received by those ambients holding the master key $\mathsf{K_{AS}}$ under which the message is encrypted.

2. The Server first receives Alice's name ($\mathsf{y_A} = \mathsf{A}$) and obtains her master key ($\mathsf{y_{K_{AS}}} = \mathsf{K_{AS}}$) from the $\mathsf{KeyTable}$ and uses it to decrypt the remaining components ($\mathsf{y_B} = \mathsf{B}$ and $\mathsf{y_{K_{AB}}} = \mathsf{K_{AB}}$) of the message:

$$\mathsf{KeyTable} \mid (\mathsf{y_A})^{\circ}.\ (\mathsf{y_{K_{AS}}})^{\mathsf{y_A}}.\ (\mathsf{y_B}, \mathsf{y_{K_{AB}}})^{\mathsf{y_{K_{AS}}}}.$$

   The Server obtains Bob's master key ($\mathsf{y_{K_{BS}}} = \mathsf{K_{BS}}$) from $\mathsf{KeyTable}$ and uses it to encrypt Alice's name ($\mathsf{y_A} = \mathsf{A}$) and the random key ($\mathsf{y_{K_{AB}}} = \mathsf{K_{AB}}$) and sends it to Bob ($\mathsf{y_B} = \mathsf{B}$):

$$(\mathsf{y_{K_{BS}}})^{\mathsf{y_B}}.\ \mathsf{y_{K_{BS}}}[\mathsf{out\,S.\ in\,y_B.}\ \langle \mathsf{y_A}, \mathsf{y_{K_{AB}}}\rangle^{\circ}]$$

   Bob decrypts the message using his master key $\mathsf{K_{BS}}$ and obtains Alice's name ($\mathsf{z_A} = \mathsf{A}$) and the session key ($\mathsf{z_{K_{AB}}} = \mathsf{K_{AB}}$):

$$(\mathsf{z_A}, \mathsf{z_{K_{AB}}})^{\mathsf{K_{BS}}}.$$

3. Alice creates her message (M) and sends it encrypted with the session key $\mathsf{K_{AB}}$ to Bob

$$(\nu\, \mathsf{M} : \mathbb{M})\ \ \mathsf{K_{AB}}[\mathsf{out\,A.\ in\,B.}\ \langle \mathsf{M}\rangle^{\circ}]$$

   which Bob receives and decrypts using the session key ($\mathsf{z_{K_{AB}}} = \mathsf{K_{AB}}$):

$$(\mathsf{x})^{\mathsf{z_{K_{AB}}}}.\ \cdots \mathsf{x} \cdots$$

The overall protocol can be written as follows:

$$\begin{array}{l}
\mathsf{A}\ [(\nu\, \mathsf{K_{AB}} : \mathbb{K}_{\mathsf{AB}})\ \mathsf{K_{AS}}[\mathsf{out\,A.\ in\,S.}\ (\langle \mathsf{A}\rangle^{\uparrow} \mid \langle \mathsf{B}, \mathsf{K_{AB}}\rangle^{\circ})\ ]\ \ \mid \\
\qquad\qquad\qquad (\nu\, \mathsf{M} : \mathbb{M})\ \mathsf{K_{AB}}[\mathsf{out\,A.\ in\,B.}\ \langle \mathsf{M}\rangle^{\circ}]\ ] \\
\mid \\
\mathsf{S}\ [\,\mathsf{KeyTable} \mid \\
\quad (\mathsf{y_A})^{\circ}.\ (\mathsf{y_{K_{AS}}})^{\mathsf{y_A}}.\ (\mathsf{y_B}, \mathsf{y_{K_{AB}}})^{\mathsf{y_{K_{AS}}}}. \\
\qquad\qquad (\mathsf{y_{K_{BS}}})^{\mathsf{y_B}}.\ \mathsf{y_{K_{BS}}}[\mathsf{out\,S.\ in\,y_B.}\ \langle \mathsf{y_A}, \mathsf{y_{K_{AB}}}\rangle^{\circ}\ ]\ ] \\
\mid \\
\mathsf{B}\ [(\mathsf{z_A}, \mathsf{z_{K_{AB}}})^{\mathsf{K_{BS}}}.\ (\mathsf{x})^{\mathsf{z_{K_{AB}}}}.\ \cdots \mathsf{x} \cdots\ ]
\end{array}$$

We refer to the literature for the security properties of the Wide Mouthed Frog protocol. Actually, our encoding is a bit "more secure" than the original protocol. As an example, in step (1) we ensure that no one can listen to neither Alice's name nor Bob's name and the session key until after the package has been delivered. Rather than attempting to weaken the encoding to open up for more attacks we consider this a benefit of performing the encoding in Boxed Ambients.

Yahalom. The Yahalom protocol is described in [10]; its classical narration is as follows:

$$\begin{array}{lll}
1. & \mathsf{A} \rightarrow \mathsf{B}: & \mathsf{A}, \mathsf{R_A} \\
2. & \mathsf{B} \rightarrow \mathsf{S}: & \mathsf{B}, \mathsf{K_{BS}}[\,\mathsf{A}, \mathsf{R_A}, \mathsf{R_B}\,] \\
3. & \mathsf{S} \rightarrow \mathsf{A}: & \mathsf{K_{AS}}[\,\mathsf{B}, \mathsf{K_{AB}}, \mathsf{R_A}, \mathsf{R_B}\,], \mathsf{K_{BS}}[\,\mathsf{A}, \mathsf{K_{AB}}\,] \\
4. & \mathsf{A} \rightarrow \mathsf{B}: & \mathsf{K_{BS}}[\,\mathsf{A}, \mathsf{K_{AB}}\,], \mathsf{K_{AB}}[\,\mathsf{R_B}\,] \\
5. & \mathsf{A} \rightarrow \mathsf{B}: & \mathsf{K_{AB}}[\,\mathsf{M}\,]
\end{array}$$

In Boxed Ambients it can be encoded as follows:

1. Alice sends her name and a random number to Bob:

   $$(\nu\, \mathsf{R_A} : \mathbb{R})\ \mathsf{p}[\mathsf{out\,A.\ in\,B.}\ \langle \mathsf{A}, \mathsf{R_A}\rangle^{\uparrow}]$$

   Here there is no encryption in the message and we have to revert to the use of an anonymous package.

2. Bob receives Alice's name and random number and generates his own random number; he then encrypts Alice's name, her random number and his own random number with his own master key and sends it to the Server together with his name:

   $$(\mathsf{y_A}, \mathsf{y_R})^{\circ}.\ \ (\nu\, \mathsf{R_B} : \mathbb{R})\ \ \mathsf{K_{BS}}[\mathsf{out\,B.\ in\,S.}\ (\langle \mathsf{B}\rangle^{\uparrow} \mid \langle \mathsf{y_A}, \mathsf{y_R}, \mathsf{R_B}\rangle^{\circ})]$$

3. The Server receives Bob's name and obtains his master key from $\mathsf{KeyTable}$; he then decrypts Alice's name and the two random numbers and obtains Alice's master key and creates a random session key. Then he constructs *two* messages. The first message is encrypted with Alice's master key and is sent to Alice; it contains Bob's name, the session key, Alice's random number and Bob's random number. The other message is sent to Alice too although intended for Bob (and hence may be instructed to go there) and is encrypted with Bob's master key and consists of Alice's name and the session key:

   $$\begin{array}{l}
   \mathsf{KeyTable} \mid \\
   (\mathsf{z_B})^{\circ}.\ (\mathsf{z_{K_{BS}}})^{\mathsf{z_B}}.\ (\mathsf{z_A}, \mathsf{z_R}, \mathsf{z'_R})^{\mathsf{z_{K_{BS}}}}.\ (\mathsf{z_{K_{AS}}})^{\mathsf{z_A}}.\ (\nu\, \mathsf{K} : \mathbb{K}) \\
   \qquad\qquad \mathsf{z_{K_{AS}}}[\mathsf{out\,S.\ in\,z_A}.\langle \mathsf{z_B}, \mathsf{K}, \mathsf{z_R}, \mathsf{z'_R}\rangle^{\circ}] \mid \\
   \qquad\qquad \mathsf{z_{K_{BS}}}[\mathsf{out\,S.\ in\,z_A}.\ (\mathsf{y_1}, \mathsf{y_2})^{\uparrow}.\ \mathsf{y_1}.\, \mathsf{y_2}.\ \langle \mathsf{z_A}, \mathsf{K}\rangle^{\circ}]
   \end{array}$$

4. Alice decrypts the message intended for her and checks that the random number is her own. Then she sends two messages to Bob: one being the message from the Server and the other being Bob's random number encrypted with the session key:

   $$\begin{array}{l}
   (\mathsf{x_B}, \mathsf{x_K}, \mathsf{x_R}, \mathsf{x'_R})^{\mathsf{K_{AS}}}.\ (\mathsf{x_R}[\langle\rangle^{\circ}] \mid ()^{\mathsf{R_A}}.\ (\ \langle \mathsf{out\,A,\ in\,B}\rangle^{\circ} \ \mid \\
   \qquad\qquad\qquad\qquad\qquad\qquad \mathsf{x_K}[\mathsf{out\,A.\ in\,B.}\ \langle \mathsf{x'_R}\rangle^{\circ}]))
   \end{array}$$

   Note that we test the equality of $\mathsf{x_R}$ and $\mathsf{R_A}$ as illustrated above; in a similar way we could have decided to test the equality of $\mathsf{x_B}$ and B (but protocol

narrations are often a bit unclear about how many tests actually have to be carried out).
Bob decrypts the first message with his master key thereby obtaining the session key and uses it to decrypt the other message and checks that it equals his random number:

$$(\mathsf{y_A}, \mathsf{y_K})^{\mathsf{K_{BS}}}.\ (\mathsf{y_R})^{\mathsf{y_K}}.\ (\mathsf{y_R}[\langle\rangle^{\circ}] \mid ()^{\mathsf{R_B}}.\cdots)$$

5. Alice creates her message and sends it encrypted with the session key to Bob

$$(\nu\, \mathsf{M} : \mathbb{M})\ \mathsf{x_K}[\mathsf{out\,A.\ in\,B.}\ \langle \mathsf{M}\rangle^{\circ}]$$

which Bob receives and decrypts using the session key:

$$(\mathsf{y_M})^{\mathsf{y_K}}.\cdots \mathsf{y_M}\cdots$$

The protocol may be summarised as follows:

$$\begin{array}{l}
\mathsf{A}\ [(\nu\, \mathsf{R_A} : \mathbb{R})\ \mathsf{p}[\mathsf{out\,A.\ in\,B.}\ \langle \mathsf{A}, \mathsf{R_A}\rangle^{\uparrow}]\ \mid \\
\qquad\qquad\qquad (\mathsf{x_B}, \mathsf{x_K}, \mathsf{x_R}, \mathsf{x'_R})^{\mathsf{K_{AS}}}.\ (\mathsf{x_R}[\langle\rangle^{\circ}] \mid \\
\qquad\qquad\qquad\qquad\qquad\qquad ()^{\mathsf{R_A}}.\ (\langle \mathsf{out\,A,\ in\,B}\rangle^{\circ}\ \mid \\
\qquad\qquad\qquad\qquad\qquad\qquad \mathsf{x_K}[\mathsf{out\,A.\,in\,B.}\ \langle \mathsf{x'_R}\rangle^{\circ}]\ \mid \\
\qquad\qquad\qquad\qquad\qquad\qquad (\nu\, \mathsf{M} : \mathbb{M})\ \mathsf{x_K}[\mathsf{out\,A.\ in\,B.}\ \langle \mathsf{M}\rangle^{\circ}])))] \\
\mid \\
\mathsf{S}\ [\,\mathsf{KeyTable} \mid \\
\quad (\mathsf{z_B})^{\circ}.\ (\mathsf{z_{K_{BS}}})^{\mathsf{z_B}}.\ (\mathsf{z_A}, \mathsf{z_R}, \mathsf{z'_R})^{\mathsf{z_{K_{BS}}}}.\ (\mathsf{z_{K_{AS}}})^{\mathsf{z_A}}.\ (\nu\, \mathsf{K} : \mathbb{K}) \\
\qquad\qquad\qquad\qquad\quad \mathsf{z_{K_{AS}}}[\mathsf{out\,S.\,in\,z_A}.\langle \mathsf{z_B}, \mathsf{K}, \mathsf{z_R}, \mathsf{z'_R}\rangle^{\circ}]\ \mid \\
\qquad\qquad\qquad\qquad\quad \mathsf{z_{K_{BS}}}[\mathsf{out\,S.\,in\,z_A}.\ (\mathsf{y_1}, \mathsf{y_2})^{\uparrow}.\ \mathsf{y_1}.\,\mathsf{y_2}.\ \langle \mathsf{z_A}, \mathsf{K}\rangle^{\circ}]\,] \\
\mid \\
\mathsf{B}\ [(\mathsf{y_A}, \mathsf{y_R})^{\circ}.\ \ (\nu\, \mathsf{R_B} : \mathbb{R})\ \ \mathsf{K_{BS}}[\mathsf{out\,B.\ in\,S.}\ (\langle \mathsf{B}\rangle^{\uparrow} \mid \langle \mathsf{y_A}, \mathsf{y_R}, \mathsf{R_B}\rangle^{\circ})]\ \mid \\
\qquad\qquad\qquad\qquad\qquad (\mathsf{y_A}, \mathsf{y_K})^{\mathsf{K_{BS}}}.\ (\mathsf{y_R})^{\mathsf{y_K}}.\ (\mathsf{y_R}[\langle\rangle^{\circ}] \mid ()^{\mathsf{R_B}}.\,(\mathsf{y_M})^{\mathsf{y_K}}.\cdots \mathsf{y_M}\cdots)]
\end{array}$$

We refer to the literature for the security properties of the Yahalom protocol.

Needham-Schroeder. The Needham-Schroeder symmetric key protocol is described in [27]; its classical protocol narration is as follows

$$\begin{array}{lll}
1. & \mathsf{A} \rightarrow \mathsf{S} : & \mathsf{A}, \mathsf{B}, \mathsf{R_A} \\
2. & \mathsf{S} \rightarrow \mathsf{A} : & \mathsf{K_{AS}}[\,\mathsf{R_A}, \mathsf{B}, \mathsf{K_{AB}}, \mathsf{K_{BS}}[\,\mathsf{A}, \mathsf{K_{AB}}\,]\,] \\
3. & \mathsf{A} \rightarrow \mathsf{B} : & \mathsf{K_{BS}}[\,\mathsf{A}, \mathsf{K_{AB}}\,] \\
4. & \mathsf{B} \rightarrow \mathsf{A} : & \mathsf{K_{AB}}[\,\mathsf{R_B}\,] \\
5. & \mathsf{A} \rightarrow \mathsf{B} : & \mathsf{K_{AB}}[\,\mathsf{R_B} - 1\,] \\
6. & \mathsf{A} \rightarrow \mathsf{B} : & \mathsf{K_{AB}}[\,\mathsf{M}\,]
\end{array}$$

In Boxed Ambients this can be encoded as follows:

1. Alice sends her name, Bob's name and a random number to the Server:

$$(\nu\, \mathsf{R_A} : \mathbb{R})\ \mathsf{p}[\mathsf{out\,A.\ in\,S.}\ \langle \mathsf{A}, \mathsf{B}, \mathsf{R_A}\rangle^{\uparrow}\,]$$

2. The Server receives the message, generates a random session key K, and sends a single message to Alice encrypted with her master key. It contains Alice's random number and the session key. It also contains a message intended for Bob (and hence may be instructed to go there) and encrypted with Bob's master key containing the session key and Alice's name. To make this work the message sent to Alice must act as a forwarder from the ambience of the message to the message intended for Bob (as discussed previously):

$$\begin{aligned}\mathsf{KeyTable}\ \mid\ & (\mathsf{y_A}, \mathsf{y_B}, \mathsf{y_R})^{\circ}.\, (\mathsf{y_{K_{AS}}})^{\mathsf{y_A}}.\, (\mathsf{y_{K_{BS}}})^{\mathsf{y_B}}.\, (\nu\, \mathsf{K} : \mathbb{K})\\ & \quad \mathsf{y_{K_{AS}}}[\mathsf{out\,S.\ in\,y_A}.\ (\langle \mathsf{y_R}, \mathsf{K}\rangle^{\circ}\ \mid\\ & \qquad\qquad (\mathsf{y_1}, \mathsf{y_2}, \mathsf{y_3})^{\circ}.\, \langle \mathsf{y_1}, \mathsf{y_2}, \mathsf{y_3}\rangle^{\circ}\ \mid\\ & \qquad\qquad \mathsf{y_{K_{BS}}}[(\mathsf{y_1}, \mathsf{y_2}, \mathsf{y_3})^{\uparrow}.\ \mathsf{y_1}.\, \mathsf{y_2}.\, \mathsf{y_3}.\ \langle \mathsf{y_A}, \mathsf{K}\rangle^{\circ}])\ ]\end{aligned}$$

3. Alice decrypts the message and checks that she got her random number back; then she sends the included message to Bob:

$$(\mathsf{x_R}, \mathsf{x_K})^{\mathsf{K_{AS}}}.\, (\mathsf{x_R}[\langle\rangle^{\circ}] \mid ()^{\mathsf{R_A}}.\, \langle \mathsf{out\,K_{AS}},\ \mathsf{out\,A},\ \mathsf{in\,B}\rangle^{\mathsf{K_{AS}}})$$

4. Bob decrypts the message using his master key, generates a random number, encrypts it with the session key and sends it to Alice:

$$(\mathsf{z_A}, \mathsf{z_K})^{\mathsf{K_{BS}}}.\ \ (\nu\, \mathsf{R_B} : \mathbb{R})\, \mathsf{z_K}[\mathsf{out\,B.\ in\,z_A}.\ \langle \mathsf{R_B}\rangle^{\circ}]$$

5. Alice decrypts the message using the session key, she modifies Bobs random number (by duplicating it rather than subtracting one), encrypts it with the session key and sends it back to Bob:

$$(\mathsf{x'_R})^{\mathsf{x_K}}.\ \ \mathsf{x_K}[\mathsf{out\,A.\ in\,B}.\ \langle \mathsf{x'_R}, \mathsf{x'_R}\rangle^{\circ}]$$

   Bob decrypts the message with the session key and verifies that it is obtained from his random number:

$$(\mathsf{z_R}, \mathsf{z_{R'}})^{\mathsf{x_K}}.\, (\mathsf{z_R}[\langle\rangle^{\circ}] \mid \mathsf{z_{R'}}[\langle\rangle^{\circ}] \mid\ ()^{\mathsf{R_B}}.\, ()^{\mathsf{R_B}}.\ \cdots)$$

6. Alice creates her message and sends it encrypted with the session key to Bob

$$(\nu\, \mathsf{M} : \mathbb{M})\ \ \mathsf{x_K}[\mathsf{out\,A.\ in\,B}.\ \langle \mathsf{M}\rangle^{\circ}]$$

   which Bob receives and decrypts using the session key:

$$(\mathsf{z_M})^{\mathsf{z_K}}.\cdots \mathsf{z_M} \cdots$$

The protocol may be summarised as follows:

$$
\begin{array}{l}
\mathsf{A}\ [(\nu\,\mathsf{R_A}:\mathbb{R})\ \mathsf{p}[\mathsf{out\,A}.\ \mathsf{in\,S}.\ \langle \mathsf{A},\mathsf{B},\mathsf{R_A}\rangle^{\uparrow}\,]\ \mid \\
\qquad\qquad\qquad (\mathsf{x_R},\mathsf{x_K})^{\mathsf{K_{AS}}}.\ (\mathsf{x_R}[\langle\rangle^{\circ}] \mid ()^{\mathsf{R_A}}.\ (\langle \mathsf{out\,K_{AS}},\ \mathsf{out\,A},\ \mathsf{in\,B}\rangle^{\mathsf{K_{AS}}}\ \mid \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (\mathsf{x'_R})^{\mathsf{x_K}}.\ \mathsf{x_K}[\mathsf{out\,A}.\ \mathsf{in\,B}.\ \langle \mathsf{x'_R},\mathsf{x'_R}\rangle^{\circ}] \mid \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (\nu\,\mathsf{M}:\mathbb{M})\ \mathsf{x_K}[\mathsf{out\,A}.\ \mathsf{in\,B}.\ \langle \mathsf{M}\rangle^{\circ}]\ ))\ ] \\
\mid \\
\mathsf{S}\ [\mathsf{KeyTable}\ \mid\ (\mathsf{y_A},\mathsf{y_B},\mathsf{y_R})^{\circ}.\ (\mathsf{y_{K_{AS}}})^{\mathsf{y_A}}.\ (\mathsf{y_{K_{BS}}})^{\mathsf{y_B}}.\ (\nu\,\mathsf{K}:\mathbb{K}) \\
\qquad\qquad\qquad\qquad\quad \mathsf{y_{K_{AS}}}[\mathsf{out\,S}.\ \mathsf{in\,y_A}.\ (\langle \mathsf{y_R},\mathsf{K}\rangle^{\circ}\ \mid \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad (\mathsf{y_1},\mathsf{y_2},\mathsf{y_3})^{\circ}.\ \langle \mathsf{y_1},\mathsf{y_2},\mathsf{y_3}\rangle^{\circ} \mid \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathsf{y_{K_{BS}}}[(\mathsf{y_1},\mathsf{y_2},\mathsf{y_3})^{\uparrow}.\ \mathsf{y_1}.\,\mathsf{y_2}.\,\mathsf{y_3}.\ \langle \mathsf{y_A},\mathsf{K}\rangle^{\circ}])\ ]\ ] \\
\mid \\
\mathsf{B}\ [(\mathsf{z_A},\mathsf{z_K})^{\mathsf{K_{BS}}}.\ \ (\nu\,\mathsf{R_B}:\mathbb{R})\ \mathsf{z_K}[\mathsf{out\,B}.\ \mathsf{in\,z_A}.\ \langle \mathsf{R_B}\rangle^{\circ}]\ \mid \\
\qquad\qquad\qquad\qquad\qquad\quad (\mathsf{z_R},\mathsf{z_{R'}})^{\mathsf{x_K}}.\ (\,\mathsf{z_R}[\langle\rangle^{\circ}] \mid \mathsf{z_{R'}}[\langle\rangle^{\circ}]\ \mid \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad ()^{\mathsf{R_B}}.\ ()^{\mathsf{R_B}}.\ (\mathsf{z_M})^{\mathsf{z_K}}.\cdots \mathsf{z_M}\cdots)\ ]
\end{array}
$$

We refer to the literature for the security properties of the Needham-Schroeder protocol.

5.3 Adapting the 0CFA Analysis to Deal with Communication

As far as the analysis is concerned we revert to the 0CFA based analysis presented in Section 2.2. As before we need to have a component

$$\mathcal{I} : \mathbf{Group} \to \mathcal{P}(\mathbf{Group} \cup \mathbf{Cap})$$

keeping track of whom is inside whom. Additionally we need a component keeping track of the values bound to variables

$$\mathcal{R} : \mathbf{Var} \to \mathcal{P}(\mathbf{Group} \cup \mathbf{Cap})$$

and a component keeping track of the contents of the mailboxes:

$$\mathcal{C} : \mathbf{Group} \to \mathcal{P}((\mathbf{Group} \cup \mathbf{Cap})^{*})$$

Judgements then take the form

$$(\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\mu}_{\Gamma} P$$

where there is only one superscript to $\models$ because we are reverting to the context-insensitive 0CFA based analysis.

Example 43. Suppose that an ambient A has gotten hold of a message encrypted under the key K and that the message instructs A where to move. Assuming that A knows the key K this may be illustrated by the process

$$\mathsf{A}\,[\mathsf{K}\,[\langle \mathsf{in\,S}\rangle^{\circ}] \mid (\mathsf{x})^{\mathsf{K}}.\,\mathsf{x}] \mid \mathsf{S}\,[\,]$$

Assuming that we analyse the process in an environment Γ with $\Gamma(\mathsf{A}) = \mathbb{A}$, $\Gamma(\mathsf{K}) = \mathbb{K}$, and $\Gamma(\mathsf{S}) = \mathbb{S}$ the analysis estimate

$$\begin{array}{ll}
\mathcal{I}(\star) = \{\mathbb{A}, \mathbb{S}\} & \mathcal{C}(\mathbb{K}) = \{\mathsf{in}\,\mathbb{S}\} \\
\mathcal{I}(\mathbb{A}) = \{\mathbb{K}, \mathsf{in}\,\mathbb{S}\} & \mathcal{C}(\star) = \mathcal{C}(\mathbb{A}) = \mathcal{C}(\mathbb{S}) = \emptyset \\
\mathcal{I}(\mathbb{S}) = \{\mathbb{A}\} & \\
\mathcal{I}(\mathbb{K}) = \emptyset & \mathcal{R}(\mathsf{x}) = \{\mathsf{in}\,\mathbb{S}\}
\end{array}$$

will show the possible behaviour of the process. This estimate is in fact the best estimate found by the analysis specified below. The ambient hierarchy is once again recorded in $\mathcal{I}$ and records that A may turn up inside S (i.e. $\{\mathbb{A}\} \subseteq \mathcal{I}(\mathbb{S})$). Inspection of $\mathcal{C}$ shows that communication may only take place within K where the capability $\mathsf{in}\,\mathsf{S}$ may be communicated. The variable environment $\mathcal{R}$ shows that the variable x may be bound to exactly this value.

Note that while capabilities are recorded directly in $\mathcal{I}$, communication primitives are recorded indirectly. Outputs are recorded by the effect they have on the mailboxes, i.e. on the content of $\mathcal{C}$, while inputs show up as the possible values in $\mathcal{R}$ of the variables that will be bound by the input. □

Specification of the communication analysis. As before each acceptable analysis estimate for a composite process must also be an acceptable analysis estimate for its sub-processes:

$$\begin{array}{lll}
(\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} (\nu\, n\colon \mu)\, P & \text{iff} & (\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma[n\mapsto\mu]} P \\
(\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} (\nu\, \mu)\, P & \text{iff} & (\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma[\mu\mapsto\diamond]} P \\
(\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} \mathbf{0} & \text{iff} & \text{true} \\
(\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} P_1 \mid P_2 & \text{iff} & (\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} P_1 \ \wedge\ (\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} P_2 \\
(\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} !P & \text{iff} & (\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} P \\
(\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} N\,[P] & \text{iff} & \forall \mu \in \mathcal{N}_{\Gamma,\mathcal{R}}(N) : \mu \in \mathcal{I}(\star) \ \wedge\ (\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\mu}_{\Gamma} P
\end{array}$$

One change from before is that the name of an ambient can also be given by a variable. We therefore use the auxiliary function $\mathcal{N}_{\Gamma,\mathcal{R}}$ to map namings to sets of groups:

$$\begin{aligned}
\mathcal{N}_{\Gamma,\mathcal{R}}(x) &= \mathcal{R}(x) \cap \mathbf{Group} \\
\mathcal{N}_{\Gamma,\mathcal{R}}(n) &= \{\mu\} \text{ where } \mu = \Gamma(n)
\end{aligned}$$

This function can be extended to obtain the function $\mathcal{M}_{\Gamma,\mathcal{R}}$ for mapping capabilities to sets of values (capabilities *or* names):

$$\begin{aligned}
\mathcal{M}_{\Gamma,\mathcal{R}}(\mathsf{in}\,N) &= \{\mathsf{in}\,\mu \mid \mu \in \mathcal{N}_{\Gamma,\mathcal{R}}(N)\} \\
\mathcal{M}_{\Gamma,\mathcal{R}}(\mathsf{out}\,N) &= \{\mathsf{out}\,\mu \mid \mu \in \mathcal{N}_{\Gamma,\mathcal{R}}(N)\} \\
\mathcal{M}_{\Gamma,\mathcal{R}}(x) &= \mathcal{R}(x) \\
\mathcal{M}_{\Gamma,\mathcal{R}}(n) &= \{\mu\} \text{ where } \mu = \Gamma(n)
\end{aligned}$$

In Boxed Ambients there is no open-capability so we only need to adapt the clauses for the in- and out-capabilities. Since we shall use the function $\mathcal{M}_{\Gamma,\mathcal{R}}$ we have an additional quantification over μ; for in-capabilities we have

$$\begin{aligned}(\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} \mathsf{in}\, N.\ P \text{ iff } & \mathcal{M}_{\Gamma,\mathcal{R}}(\mathsf{in}\, N) \subseteq \mathcal{I}(\star) \ \wedge\ (\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} P \ \wedge \\ & \forall\, \mathsf{in}\, \mu \in \mathcal{M}_{\Gamma,\mathcal{R}}(\mathsf{in}\, N) : \varphi_{\mathsf{in}}(\mu)\end{aligned}$$

where the "closure condition" φ_{in} is defined by

$$\begin{aligned}\varphi_{\mathsf{in}}(\mu) \text{ iff } \forall\, \mu^a, \mu^p : \ & \mathsf{in}\ \mu \in \mathcal{I}(\mu^a) \wedge \\ & \mu^a \in \mathcal{I}(\mu^p) \ \wedge \\ & \mu \in \mathcal{I}(\mu^p) \\ & \Rightarrow\ \mu^a \in \mathcal{I}(\mu)\end{aligned}$$

For out-capabilities we have

$$\begin{aligned}(\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} \mathsf{out}\, N.\ P \text{ iff } & \mathcal{M}_{\Gamma,\mathcal{R}}(\mathsf{out}\, N) \subseteq \mathcal{I}(\star) \ \wedge\ (\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} P \ \wedge \\ & \forall\, \mathsf{out}\, \mu \in \mathcal{M}_{\Gamma,\mathcal{R}}(\mathsf{out}\, N) : \varphi_{\mathsf{out}}(\mu)\end{aligned}$$

where the "closure condition" φ_{out} is defined by

$$\begin{aligned}\varphi_{\mathsf{out}}(\mu) \text{ iff } \forall\, \mu^a, \mu^g : \ & \mathsf{out}\ \mu \in \mathcal{I}(\mu^a) \wedge \\ & \mu^a \in \mathcal{I}(\mu) \ \wedge \\ & \mu \in \mathcal{I}(\mu^g) \\ & \Rightarrow\ \mu^a \in \mathcal{I}(\mu^g)\end{aligned}$$

Finally, for "mixed capabilities" we have:

$$\begin{aligned}(\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} N.\ P \text{ iff } & \mathcal{M}_{\Gamma,\mathcal{R}}(N) \cap \mathbf{Cap} \subseteq \mathcal{I}(\star) \ \wedge\ (\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} P \ \wedge \\ & \forall\, \mathsf{in}\, \mu \in \mathcal{M}_{\Gamma,\mathcal{R}}(N) : \varphi_{\mathsf{in}}(\mu) \ \wedge \\ & \forall\, \mathsf{out}\, \mu \in \mathcal{M}_{\Gamma,\mathcal{R}}(N) : \varphi_{\mathsf{out}}(\mu)\end{aligned}$$

Thus, if N is a variable containing a capability then the capability must be in $\mathcal{I}(\star)$ and the "closure conditions" ensures that this capability is analysed all ambients where it may end up.

The new clauses deal with polyadic input and output for each of the three directions. For local communication the clauses are

$$\begin{aligned}(\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} \langle M_1, \cdots, M_k\rangle^{\circ} \quad & \text{iff } \ \mathcal{M}_{\Gamma,\mathcal{R}}(M_1) \times \cdots \times \mathcal{M}_{\Gamma,\mathcal{R}}(M_k) \subseteq \mathcal{C}(\star) \\ (\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} (x_1, \cdots, x_k)^{\circ}.\, P \quad & \text{iff } \ \forall (v_1, \cdots, v_k) \in \mathcal{C}(\star) : \\ & \qquad v_1 \in \mathcal{R}(x_1) \wedge \cdots \wedge v_k \in \mathcal{R}(x_k) \ \wedge \\ & \quad (\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} P\end{aligned}$$

where output ensures that the values are "put" into the local mailbox $\mathcal{C}(\star)$ while input "copies" values from $\mathcal{C}(\star)$ into the variables. For communication with a child we add the clauses:

$$\begin{array}{lll}(\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} \langle M_1,\cdots,M_k\rangle^N & \text{iff} & \forall\mu \in \mathcal{N}_{\Gamma,\mathcal{R}}(N) : \mu \in \mathcal{I}(\star) \\ & & \Rightarrow\ \mathcal{M}_{\Gamma,\mathcal{R}}(M_1) \times \cdots \times \mathcal{M}_{\Gamma,\mathcal{R}}(M_k) \subseteq \mathcal{C}(\mu)\end{array}$$

$$\begin{array}{lll}(\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} (x_1,\cdots,x_k)^N.P & \text{iff} & \forall\mu \in \mathcal{N}_{\Gamma,\mathcal{R}}(N) : \mu \in \mathcal{I}(\star) \\ & & \Rightarrow\ \forall(v_1,\cdots,v_k) \in \mathcal{C}(\mu) : \\ & & \qquad v_1 \in \mathcal{R}(x_1) \wedge \cdots \wedge v_k \in \mathcal{R}(x_k) \wedge \\ & & \qquad (\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} P\end{array}$$

Here we obtain all the possible groups μ of the child N and check that the corresponding ambient indeed occurs in the ambience $\star$; for each successful group μ the communication is recorded by adapting the clause for local communication.

For communication with a parent we add the clauses:

$$\begin{array}{lll}(\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} \langle M_1,\cdots,M_k\rangle^{\uparrow} & \text{iff} & \forall\mu : \star \in \mathcal{I}(\mu) \\ & & \Rightarrow\ \mathcal{M}_{\Gamma,\mathcal{R}}(M_1) \times \cdots \times \mathcal{M}_{\Gamma,\mathcal{R}}(M_k) \subseteq \mathcal{C}(\mu)\end{array}$$

$$\begin{array}{lll}(\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} (x_1,\cdots,x_k)^{\uparrow}.P & \text{iff} & \forall\mu : \star \in \mathcal{I}(\mu) \\ & & \Rightarrow\ \forall(v_1,\cdots,v_k) \in \mathcal{C}(\mu) : \\ & & \qquad v_1 \in \mathcal{R}(x_1) \wedge \cdots \wedge v_k \in \mathcal{R}(x_k) \wedge \\ & & \qquad (\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} P\end{array}$$

One may note that the analysis is a bit imprecise with respect to the semantics because although $\langle M\rangle^n \mid n\,[m\,[(x)^{\uparrow}.\,P \mid Q\,] \mid R\,] \not\rightarrow \cdots$ the analysis will pretend that the communication succeeds. One should also point out that the analysis of the communication primitives would have been somewhat more complex if the designers of Boxed Ambients had decided to keep the open-primitive; the reason is that then the communication may take place in other ambiences than where first encountered in the analysis (see [30] for how to deal with this.)

Correctness of the communication analysis. Once more the correctness of the analysis amounts to a "subject reduction" result:

Theorem 44. *If* $(\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} P$ *and* $P \rightarrow^* Q$ *then* $(\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} Q$.

Implementation of the communication analysis. Once more the existence of a least analysis is a consequence of the Moore family result:

Theorem 45. *For each* P*, the set* $\{(\mathcal{I},\mathcal{C},\mathcal{R}) \mid (\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\star}_{\Gamma} P\}$ *is a Moore family.*

The implementation in ALFP proceeds much as before. One caveat is the use of the universal quantification over μ in the clause for ambients:

$$\forall\mu \in \mathcal{N}_{\Gamma,\mathcal{R}}(N) :\ (\mathcal{I},\mathcal{C},\mathcal{R}) \models^{\mu}_{\Gamma} P$$

Here we must make sure that the variable μ is not in the domain or range of Γ.

Another issue is that the communication component $\mathcal{C}$ uses *sequences* to represent messages in the mailboxes. Instead of encoding sequences in general into a relational form that can be described in ALFP we use the fact that the analysis (and the semantics) only compares messages of the same length. Therefore, we can split $\mathcal{C}$ into mappings $\mathcal{C}_k : \mathbf{Group} \to \mathcal{P}((\mathbf{Group} \cup \mathbf{Cap})^k)$ for each message arity k occurring in the process we analysis. This intuitively corresponds to having different mailboxes for messages of different length. We reformulate the analysis into an equivalent analysis where analysing a communication primitive of arity k only gives requirements to the content of the communication component $\mathcal{C}_k$. In turn, this allows us to encode each communication component $\mathcal{C}_k$ as a relation of *fixed* arity $k+1$.

We then obtain an implementation much as before. However, now it is no longer the case that there is a fixed upper bound on the nesting depth of quantifiers and therefore the worst-case complexity is exponential; in practice it will be polynomial if there is a small nesting depth of ambients in the original process (or at least when the nesting depth of variable-named ambients is less than linear in the size of the process).

If we were to admit composite capabilities (i.e. $M.M$) we would face the problem that the universe is no longer finite for a given process since the process may output a few simple capabilities and then repeatedly input two capabilities and then output their composition. A rather crude way of dealing with this (taken in [30]) is to abandon recording the causal structure of capabilities. The better way is to adapt the solving technology to deal with a possibly infinite universe. A possibly infinite subset of the universe is then described using a tree grammar (or tree automaton). We can express this using our Succinct Solver by manually translating the specification into one that constructs the tree grammar; we refer to [33] for an account of how to do so for the Spi-calculus [1]. Alternatively we may replace the Succinct Solver with a more appropriate solver based on set-constraints [2] or H3 [34].

Remark 46. Our restriction to finitary calculi is in line with some recent developments for Mobile Ambients. In [17] Charatonik, Gordon, and Talbot provide a type system, which can check whether a process has finite behaviour. It is then possible to model check such a process against a so-called ambient logic [15]. Teller, Zimmer, and Hirschkoff [40] models *a resource* as an ambient R with a fixed capacity given by the maximal number of other ambients allowed inside R at top level. They provide a type system for resource control that checks whether the resource capacities in a system may be exceeded at run-time. □

5.4 Protocol and Exchange Analysis

Protocol analysis. We now show a number of properties of the Wide Mouthed Frog protocol that can be obtained using the 0CFA analysis.

Example 47. We can analyse the Wide Mouthed Frog protocol using the 0CFA analysis and aim at "maximal precision" by keeping as many groups distinct as possible. This means that we analyse the protocol in a group environment where $\Gamma(\mathsf{A}) = \mathbb{A}$, $\Gamma(\mathsf{B}) = \mathbb{B}$, $\Gamma(\mathsf{S}) = \mathbb{S}$, $\Gamma(\mathsf{K_{AS}}) = \mathbb{K}_{\mathsf{AS}}$ and $\Gamma(\mathsf{K_{BS}}) = \mathbb{K}_{\mathsf{BS}}$. Recall that the protocol additionally specifies the groups of the session key and the message (i.e. $(\nu\,\mathsf{K_{AB}} : \mathbb{K}_{\mathsf{AB}})$ and $(\nu\,\mathsf{M} : \mathbb{M})$). Then the possible values of variables in the analysis component $\mathcal{R}$ are given by:

$\mathcal{R}$	x	$\mathsf{z_{K_{AB}}}$	$\mathsf{z_A}$	$\mathsf{y_{K_{BS}}}$	$\mathsf{y_{K_{AB}}}$	$\mathsf{y_B}$	$\mathsf{y_{K_{AS}}}$	$\mathsf{y_A}$
	$\{\mathbb{M}\}$	$\{\mathbb{K}_{\mathsf{AB}}\}$	$\{\mathbb{A}\}$	$\{\mathbb{K}_{\mathsf{BS}}\}$	$\{\mathbb{K}_{\mathsf{AB}}\}$	$\{\mathbb{B}\}$	$\{\mathbb{A}, \mathbb{K}_{\mathsf{AS}}\}$	$\{\mathbb{A}\}$

Note in particular that $\mathsf{z_{K_{AB}}}$ can only be bound to the session key created by Alice. Consequently Bob may receive the messages Alice sends encrypted under the session key as shown by the values of $\mathcal{R}(\mathsf{x})$. The reason that both $\mathbb{A}$ and $\mathbb{K}_{\mathsf{AS}}$ show up in $\mathcal{R}(\mathsf{y_{K_{AS}}})$ is that the analysis does not distinguish the ambient A that represents Alice and the ambient A in the $\mathsf{KeyTable}$. □

Example 48. In the specification of the Wide Mouthed Frog protocol the server only allows *known principals* to participate in the protocol, since keys shared between the server and a principal needs to appear in $\mathsf{KeyTable}$. Suppose that an unknown principal E (for Enemy) tries to participate in the protocol by carrying out the part of Alice. That is, suppose that the Enemy is as the process describing Alice with A replaced by E everywhere. This is the process below where $\mathsf{K_{ES}}, \mathsf{M}', \mathbb{K}_{\mathsf{EB}}$, and $\mathbb{M}'$ are arbitrarily chosen free names and free groups, respectively.

$$\begin{aligned}\mathsf{E}\ [(\nu\,\mathsf{K_{EB}} : \mathbb{K}_{\mathsf{EB}})\ \mathsf{K_{ES}}[\mathsf{out}\,\mathsf{E}.\ \mathsf{in}\,\mathsf{S}.\ (\langle \mathsf{E}\rangle^{\uparrow} \mid \langle \mathsf{B}, \mathsf{K_{EB}}\rangle^{\circ})\ ]\ \mid \\ (\nu\,\mathsf{M}' : \mathbb{M}')\ \mathsf{K_{EB}}[\mathsf{out}\,\mathsf{E}.\ \mathsf{in}\,\mathsf{B}.\ \langle \mathsf{M}'\rangle^{\circ}]\ ]\end{aligned}$$

Since the Server does not know the key $\mathsf{K_{ES}}$ the attempt to participate in the protocol will fail. This can in fact be guaranteed using the 0CFA analysis by analysing the process implementing the Wide Mouthed Frog protocol in parallel with the ambient E above. The interesting part of the analysis result is:

$\mathcal{R}$	x	$\mathsf{z_{K_{AB}}}$
	$\{\mathbb{M}\}$	$\{\mathbb{K}_{\mathsf{AB}}\}$

which guarantees that Bob *will never* receive the message M' in the variable x. Since $\mathsf{K_{ES}}$ and M' are arbitrarily chosen the result guaranties that *any* such attempt by the Enemy to conform with the protocol will indeed fail.

Suppose on the other hand that the Server does know the key $\mathsf{K_{ES}}$, i.e. that $\mathsf{KeyTable}$ is extended to:

$$\mathsf{KeyTable}' \quad = \quad \mathsf{A}\,[!\langle \mathsf{K_{AS}}\rangle^{\circ}] \mid \mathsf{B}\,[!\langle \mathsf{K_{BS}}\rangle^{\circ}] \mid \mathsf{E}\,[!\langle \mathsf{K_{ES}}\rangle^{\circ}]$$

Now the Server may successfully let E participate in the protocol as confirmed by the analysis result:

$\mathcal{R}$	x	$\mathsf{z_{K_{AB}}}$
	$\{\mathbb{M}, \mathbb{M}'\}$	$\{\mathbb{K}_{\mathsf{AB}}, \mathbb{K}_{\mathsf{EB}}\}$

This corresponds to the Enemy being a dishonest principal rather than an intruder. □

Example 49. Assume that an enemy E shares a key $\mathsf{K_{ES}}$ with the Server in the Wide Mouthed Frog protocol (i.e. that the Server uses $\mathsf{KeyTable}'$ of Example 48). The enemy may now attempt to "impersonate Alice" by sending an A in the unencrypted part of the first message. This can be encoded as the process

$$\begin{array}{l} \mathsf{E}\ [(\nu\, \mathsf{K_{EB}} : \mathbb{K}_{\mathsf{EB}})\ \mathsf{K_{ES}}[\mathsf{out\,E.\ in\,S.}\ (\langle \mathbf{A} \rangle^{\uparrow} \mid \langle \mathsf{B}, \mathsf{K_{EB}} \rangle^{\circ})\ ]\ \mid \\ \qquad\qquad\qquad (\nu\, \mathsf{M}' : \mathbb{M}')\ \mathsf{K_{EB}}[\mathsf{out\,E.\ in\,B.}\ \langle \mathsf{M}' \rangle^{\circ}]\ ] \end{array}$$

The goal of the Enemy is to "fool" Bob into believing that the key $\mathsf{K_{EB}}$ was a session key generated by Alice. However, the 0CFA analysis guarantees us that this never happens. By analysing the above ambient E in parallel with the Wide Mouth Frog protocol (using $\mathsf{KeyTable}'$ instead of $\mathsf{KeyTable}$) we get:

$\mathcal{R}$	x	$\mathsf{z_{K_{AB}}}$
	$\{\mathbb{M}\}$	$\{\mathbb{K}_{\mathsf{AB}}\}$

This in fact ensures that the key $\mathsf{K_{EB}}$ *will never* even reach Bob i.e. it will never be bound to the variable $\mathsf{z_{K_{AB}}}$. Thus, in particular, this attack cannot fool Bob into believing that the key came from Alice. □

Exchange analysis. Besides opening and crossing control also *exchange analysis* of ambients is considered in [12] (and links back to the type system of [14]). Exchange analysis deals with determining what the topic of conversation is for some ambient n: is there no communication at all, is there an exchange of ambient names only, or is there an exchange of capabilities only.

These questions can be answered using the analysis as follows. Suppose that $(\mathcal{I}, \mathcal{C}, \mathcal{R}) \models^{\star}_{\Gamma} P$ and let $\mu = \Gamma(n)$.

- No communication at all takes place in case $\mathcal{C}(\mu) = \emptyset$;.
- There is exchange of ambient names only, in case $\mathcal{C}(\mu) \subseteq \mathbf{Group}^*$;
- There is exchange of capabilities only, in case $\mathcal{C}(\mu) \subseteq \mathbf{Cap}^*$.

The correctness of these claims are immediate consequences of Theorem 44. We obtain the best answer by using the least analysis estimate as guaranteed by Theorem 45.

Remark 50. Numerous type systems for ambient calculi deals with communication and much of the work is based on the original type system by Cardelli and Gordon [14] including type systems for Boxed Ambients [9, 8, 21, 26]. Furthermore, some type systems incorporates information about mobility (e.g. [11, 12]), which is also the case for Merro and Sassone's recent type system [26] for Boxed Ambients.

The type system of [26] checks whether *exchange types* of send and receive are compatible both for local and non-local communication. If any pairs of communications within a process are incompatible the process does not type check. In comparison, our 0CFA analysis will analyse *any* process; as such, our 0CFA analysis is closer to soft typing. The information conveyed by the exchange types is, in essence, similar to the content of the $\mathcal{C}$ component of our 0CFA analysis as described in the section above. A notable difference is that [26] forbids communications of different arities within the same ambient. This restriction is present in many type systems for ambient calculi and probably goes back to the type system of [14].

Also the type system of [26] incorporates *mobility types* that for a given ambient n gives the set of other ambients in which n is allowed to occur; this is similar to the $\mathcal{I}$ component of our 0CFA analysis. All types are ordered by a subtyping relation that allows to say that some types are "better" than others and to define a best type. Though listed as work-in-progress, they do not provide a *type inference* algorithm so they cannot automatically calculate the mobility and communication behaviour of a given process. □

6 Conclusion

Mobile Ambients and their variants have established themselves as a useful class of process algebras in which to study mobility. Our first aim was to extend the calculus to express *discretionary access control* in a manner compatible with the classical studies of operating systems; we achieved this goal by developing the Discretionary Ambients (and thereby generalising the Safe Ambients). Our second aim was to extend the calculus to express *mandatory access control* for confidentiality as well as integrity; we achieved this goal by modifying the semantics to enforce the checks of the reference monitor. Our third aim was to show that cryptographic key exchange protocols could be coded rather naturally in Boxed Ambients where we make use of the more general communication primitives of Boxed Ambients over those of Mobile Ambients; as far as we are aware this is the first treatment of key exchange protocols in a calculus for mobility.

Throughout we have defined the semantics and developed 0CFA or 1CFA analyses for the calculi studied. They could be implemented in our Succinct Solver by re-expressing the specification in a fragment of Alternation-free Least Fixed Point Logic (ALFP). Except for Boxed Ambients we could guarantee a worst-case complexity being a polynomial of a low degree; for Boxed Ambients the degree of the polynomial is proportional to the nesting depth of ambients in the original process (and hence exponential to the size of the process in the worst case).

We believe that our Flow Logic approach to analysis gives us a number of conveniences. We share with type systems the convenience of separating specification from implementation thereby obtaining a conceptually cleaner formulation of the analysis that interacts well with semantic correctness and state-of-the-art techniques for efficient implementation. But unlike most type systems based on subtyping we achieve polynomial time complexity for most of the analyses of interest.

Perhaps more importantly the logical or constraint-based nature of our approach lead us to the formulation of "hardest attackers": a finite process characterising all possible malicious processes (somewhat in the manner of hard problems for a given complexity class). The key element in our success is that we limit our attention to the finitary world of the static analysis. In the original development [31, 30] we considered the firewall described in [13]. Here only agents knowing the required passwords are supposed to enter, and it is shown that all agents in a special form will in fact enter. However, it is at least as important to ensure that an attacker not knowing the required passwords cannot enter, since this presents a useful technique for screening a system against attackers. This is achieved using the "hardest attacker" [31, 30]. We conjecture that a similar development may be possible for the key exchange protocols considered in Subsection 5.2; a preliminary study for protocols expressed in the LySa-calculus is reported in [5].

Acknowledgements. This research was funded in part by the SecSaf project (number 9901590) funded by the Danish Natural Science Research Council and by the DEGAS project (number IST-2001-32072) funded by the European Union and this paper is also published as DEGAS report number WP6-IMM-I01-Int-001.

References

1. M. Abadi and A. D. Gordon. A calculus for cryptographic protocols – The Spi calculus. *Information and Computation*, 148(1):1–70, 1999.
2. A. Aiken. Introduction to set constraint-based program analysis. *Science of Computer Programming (SCP)*, 35(2):79–111, 1999.
3. D. Bell and L. LaPadula. Secure computer system: Unified exposition and Multics interpretation. Technical Report ESDTR-75-306, MTR-2547, MITRE Corporation, 1975.
4. K. J. Biba. Integrity consideration for secure computer systems. Technical Report ESDTR-76-372, MTR-3153, MITRE Corporation, 1977.
5. C. Bodei, M. Buchholtz, P. Degano, F. Nielson, and H. Riis Nielson. Polynomial-time validation of protocol narration. Manuscript, 2002.
6. C. Braghin, A. Cortesi, and R. Focardi. Control flow analysis for information flow security in Mobile Ambients. In *Proceedings of NordSec 2001.* Technical Report IMM-TR-2001-14. Technical University of Denmark, 2001.
7. M. Buchholtz, F. Nielson, and H. Riis Nielson. Experiments with Succinct Solvers. Technical Report IMM-TR-2002-4, Technical University of Denmark, 2002.
8. M. Bugliesi, G. Castagna, and S. Crafa. Boxed Ambients. In *Theoretical Aspects in Computer Science (TACS 2001)*, volume 2215 of *Lecture Notes in Computer Science*, pages 37–63. Springer, 2001.
9. M. Bugliesi, G. Castagna, and S. Crafa. Reasoning about security in Mobile Ambients. In *CONCUR 2001 – Concurrency Theory*, volume 2154 of *Lecture Notes in Computer Science*, pages 102–120. Springer, 2001.

10. M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, pages 18–36, 1990.
11. L. Cardelli, G. Ghelli, and A. D. Gordon. Mobility types for Mobile Ambients. In *Automata, Languages, and Programming. 26th International Colloquium, ICALP 1999*, volume 1644 of *Lecture Notes in Computer Science*, pages 230–239. Springer, 1999.
12. L. Cardelli, G. Ghelli, and A. D. Gordon. Ambient groups and mobility types. In *Theoretical Computer Science. International Conference IFIP TCS 2000*, volume 1872 of *Lecture Notes in Computer Science*, pages 333–347. Springer, 2000.
13. L. Cardelli and A. D. Gordon. Mobile Ambients. In *Foundations of Software Science and Computation Structures (FoSSaCS 1998)*, volume 1378 of *Lecture Notes in Computer Science*, pages 140–155. Springer, 1998.
14. L. Cardelli and A. D. Gordon. Types for Mobile Ambients. In *Proceedings of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 1999)*, pages 79–92. ACM Press, 1999.
15. L. Cardelli and A. D. Gordon. Anytime, anywhere: Modal logics for Mobile Ambients. In *Proceedings of the 27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2000)*, pages 365–377. ACM Press, 2000.
16. L. Cardelli and A. D. Gordon. Mobile Ambients. *Theoretical Computer Science*, 240(1):177–213, 2000.
17. W. Charatonik, A. D. Gordon, and J.-M. Talbot. Finite-control Mobile Ambients. In *Programming Languages and Systems. 11th European Symposium on Programming (ESOP 2001)*, volume 2305 of *Lecture Notes in Computer Science*. Springer, 2001.
18. J. Clark and J. Jacob. A survey of authentication protocol literature: Version 1.0. http://www-users.cs.york.ac.uk/∼jac/papers/drareviewps.ps, 1997.
19. P. Cousot and R. Cousot. Abstract Interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the 4th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 1977)*, pages 238–252. ACM Press, 1977.
20. P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *Proceedings of the 6th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 1979)*, pages 269–282. ACM Press, 1979.
21. S. Crafa, M. Bugliesi, and G. Castagna. Information flow security for Boxed Ambients. In *F-WAN: Foundations of Wide Area Network Computing*, volume 63 of *Electronic Notes in Theoretical Computer Science*, 2002.
22. D. Gollmann. *Computer Security*. Wiley, 1999.
23. F. Levi and S. Maffeis. An abstract interpretation framework for analysing Mobile Ambients. In *Static Analysis, 8th International Symposium, SAS 2001*, pages 395–411. Springer, 2001.
24. F. Levi and D. Sangiorgi. Controlling interference in ambients. In *Proceedings of the 27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2000)*, pages 352–364. ACM Press, 2000.
25. D. McAllester. On the complexity analysis of static analyses. In *Static Analysis, 6th International Symposium, SAS 1999*, volume 1694 of *Lecture Notes in Computer Science*, pages 312–329. Springer, 1999.
26. M. Merro and V. Sassone. Typing and subtyping mobility in Boxed Ambients. In *CONCUR 2002 – Concurrency Theory*, volume 2421 of *Lecture Notes in Computer Science*, pages 304–320. Springer, 2002.
27. R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

28. F. Nielson and H. Riis Nielson. Flow Logics and operational semantics. *Electronic Notes in Theoretical Computer Science*, 10, 1998.
29. F. Nielson, H. Riis Nielson, and C. Hankin. *Principles of Program Analysis*. Springer, 1999.
30. F. Nielson, H. Riis Nielson, and R. R. Hansen. Validating firewalls using flow logics. *Theoretical Computer Science*, 283(2):381–418, 2002.
31. F. Nielson, H. Riis Nielson, R. R. Hansen, and J. G. Jensen. Validating firewalls in Mobile Ambients. In *CONCUR 1999 – Concurrency Theory*, volume 1664 of *Lecture Notes in Computer Science*, pages 463–477. Springer, 1999.
32. F. Nielson, H. Riis Nielson, and H. Seidl. Automatic complexity analysis. In *Programming Languages and Systems. 11th European Symposium on Programming (ESOP 2002)*, number 2305 in Lecture Notes in Computer Science, pages 243–261. Springer, 2002.
33. F. Nielson, H. Riis Nielson, and H. Seidl. Cryptographic analysis in cubic time. *Electronic Notes in Theoretical Computer Science*, 62, 2002.
34. F. Nielson, H. Riis Nielson, and H. Seidl. Normalizable Horn clauses, strongly recognizable relations and Spi. In *Static Analysis, 9th International Symposium, SAS 2002*, volume 2477 of *Lecture Notes in Computer Science*, pages 20–35. Springer, 2002.
35. F. Nielson, H. Riis Nielson, and H. Seidl. Succinct Solvers. Manuscript, 2002.
36. F. Nielson and H. Seidl. Control-flow analysis in cubic time. In *Programming Languages and Systems. 10th European Symposium on Programming (ESOP 2001)*, volume 2028 of *Lecture Notes in Computer Science*, pages 252–268. Springer, 2001.
37. H. Riis Nielson and F. Nielson. Shape analysis for Mobile Ambients. In *Proceedings of the 27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2000)*, pages 142–154. ACM Press, 2000.
38. H. Riis Nielson and F. Nielson. Shape analysis for Mobile Ambients. *Nordic Journal of Computing*, 8:233–275, 2001.
39. H. Riis Nielson and F. Nielson. Flow Logic: a multi-paradigmatic approach to static analysis. In *The Essence of Computation: Complexity, Analysis, Transformation*, Lecture Notes in Computer Science. Springer, to appear.
40. D. T. Teller, P. Zimmer, and D. Hirschkoff. Using ambients to control resources. In *CONCUR 2002 – Concurrency Theory*, volume 2421 of *Lecture Notes in Computer Science*, pages 288–303. Springer, 2002.