An overview of Boxed Ambients
(Abstract)

Giuseppe Castagna [1]

Département d'Informatique, École Normale Supérieure, Paris, France

Lecture on joint work with

Michele Bugliesi and Silvia Crafa [2]

Dipartimento d'Informatica, Università Ca' Foscari, Venezia, Italy

In this lecture we present some work we published in [2,3] and hint at some new current lines of research on information flow and security.

More precisely, we describe the calculus of *Boxed Ambients* a variant of Cardelli and Gordon's *Mobile Ambients*[4] a calculus of mobile and dynamically reconfigurable agents. Boxed Ambients inherit from Mobile Ambients (part of) the mobility primitives but rely on a completely different model of communication. The new communication primitives fit nicely the design principles of Mobile Ambients, and complement the existing constructs for ambient mobility with finer-grained, and more effective, mechanisms for ambient interaction. As a result Boxed Ambients retain the expressive power and the computational flavor of Ambient Calculus, as well as the elegance of its formal presentation. In addition, they enhance the flexibility of typed communications over Mobile Ambients, and provide new insight into the relationship between synchronous and asynchronous input-output.

1. Mobile Ambients

Ambients are named process of the form $a[\,P\,]$ where a is a name and P a process. Processes can be composed in parallel, as in $P \mid Q$, exercise a capability, as in $M.P$, declare local names as in $(\boldsymbol{\nu} x)P$, or simply do nothing as in $\mathbf{0}$. Ambients may be nested to form a tree structure that can be dynamically reconfigured by exercising the capabilities $\mathsf{in}, \mathsf{out}$ and open. In addition, ambients and processes may communicate. Communication is anonymous, and happens inside ambients. The configuration $(\boldsymbol{x})P \mid \langle M \rangle$ represents the parallel composition of two processes, the output process $\langle M \rangle$ "dropping" the message M, and the input process $(\boldsymbol{x})P$ reading the message M and continuing as $P\{\boldsymbol{x} := M\}$. The open capability has a

[1] Email: Giuseppe.Castagna@ens.fr
[2] Email: {michele,silvia}@dsi.unive.it

fundamental interplay with communication: in fact, communication results from a combination of mobility and opening control. To exemplify, consider two ambients running in parallel as in the following configuration $a[\![\,(\boldsymbol{x})P \mid Q\,]\!] \mid b[\![\,\langle M\rangle \mid R\,]\!]$. The exchange of the value M from b to the process P enclosed in a happens as a result of, say, first b moving inside a, and then a opening b. Thus, if Q is the process $\mathsf{open}\ b$, and R is $\mathsf{in}\ a$, communication is the result of the following sequence of reductions:

$$
\begin{array}{ll}
a[\![\,(\boldsymbol{x})P \mid \mathsf{open}\ b\,]\!] \mid b[\![\,\langle M\rangle \mid \mathsf{in}\ a\,]\!] & \\
\quad \rightarrow a[\![\,(\boldsymbol{x})P \mid \mathsf{open}\ b \mid b[\![\,\langle M\rangle\,]\!]\,]\!] & \text{by exercising } \mathsf{in}\ a \\
\quad \rightarrow a[\![\,(\boldsymbol{x})P \mid \langle M\rangle\,]\!] & \text{by } \mathsf{open}\ b\text{, unleashing } M \\
\quad \rightarrow a[\![\,P\{\boldsymbol{x} := M\}\,]\!] & \text{by local communication}
\end{array}
$$

2. A case against ambient opening

While fundamental to the Ambient Calculus for the reasons just illustrated, an unrestricted use of the open capability appears to bring about serious security concerns in wide-area distributed applications.

Consider a scenario where a client agent c wants to access some information M in a resource r on a remote host h. This situation can be modeled by the configuration $c[\![\,\mathsf{in}\ h.P\,]\!] \mid h[\![\,r[\![\,\langle M\rangle \mid Q\,]\!]\,]\!]$. As a result of c exercising the capability $\mathsf{in}\ h$, the system evolves into the new configuration $h[\![\,a[\![\,P\,]\!] \mid r[\![\,\langle M\rangle \mid Q\,]\!]\,]\!]$. Now how can the client access the resource? A first solution is that the r enters c and once there it is opened; but it seems strange that in order to access a resource an agent must dissolve it. A second solution is that the client enters the resource and then it is opened; but it is even stranger that in order to access a resource an agent has in some sense to commit a suicide. The only reasonable solution is then that the resource and the client use some protocol relying on exchanges of some temporary agents; but in that case it becomes difficult to determine whether it is c that accesses r or vice-versa.

Static or dynamic analysis of incoming code are often advocated as solutions to the above problem but these solutions may not be always possible, or feasible, in practice. This is not meant to dismiss the role of static analysis. To the contrary, it should be taken as a motivation to seek new design principles enabling a more effective use of static analysis. One such principle for Mobile Ambients, which leaded to the definition of Boxed Ambients is that resource access should not rely on some kind of protocol but instead be already accounted for at the level of communication primitives.

3. Boxed Ambients: overview and main results

Boxed Ambients result from Cardelli and Gordon's Mobile Ambients essentially by dropping the open capability while retaining the in and out capabilities for mobility. Disallowing open represents a rather drastic departure from the Ambient Calculus, and requires new primitives for process communication.

As in the Ambient Calculus, processes in the new calculus communicate via anonymous channels, inside ambients. This is a formal parsimony that simplifies the definition of the new calculus while not involving any loss of expressive power: in fact, named communication channels can be coded in faithful ways using the existing primitives. In addition, to compensate for the absence of `open`, Boxed Ambients are equipped with primitives for communication across ambient boundaries, between parent and children. Syntactically, this is obtained by means of tags specifying the *location* where the communication has to take place: for instance, $(x)^n P$ indicates an input from child ambient n, while $\langle M \rangle^{\uparrow}$ is an output to the parent ambient.

The choice of these primitives, and the resulting model of communication is inspired to Castagna and Vitek's *Seal Calculus* [7], from which Boxed Ambients also inherit the two principles of *mediation* and *locality*. Mediation implies that remote communication, i.e., between sibling ambients, is not possible: it either requires mobility, or intervention by the ambients' parent. Locality means that communication resources are *local* to ambients, and message exchanges result from explicit read and write requests on those resources. To exemplify, consider the following nested configuration:

$$n[\,(x)^p P \mid p[\,\langle M \rangle \mid (x)Q \mid q[\,\langle N \rangle^{\uparrow}\,]\,]\,]$$

Ambient n makes a downward request to read p's local value M, while ambient q makes an upward write request to communicate its value N to its parent. The downward input request $(x)^p P$ may only synchronize with the output $\langle M \rangle$ local to p. Instead, $(x)Q$ may non-deterministically synchronize with either output: of course, type safety requires that M and N be of the same type. Interestingly, however, exchanges of different types may take place within the same ambient without type confusion:

$$n[\,(x)^p P \mid (x)^q Q \mid p[\,\langle M \rangle\,] \mid q[\,\langle N \rangle\,]\,]$$

The two values M and N are local to p and q, respectively, and may very well have different types: there is no risk of type confusion, as $(x)^p P$ requests a read from p, while $(x)^q Q$ requests a read from q.

The ressource access case of previous section can thus be handled by the definition in the host h of a monitor process R that manages the accesses to the ressource (for example $h[\,a[\,P\,] \mid !((x)^r\langle x \rangle^a) \mid r[\,Q\,]\,]$ where $!((x)^r\langle x \rangle^a)$ is the monitor process).

This flexibility of communication results from the combination of two design choices: directed input/output operations, and resource locality. In fact, these choices have other interesting consequences.

- They provide the calculus with fine-grained primitives for ambient interaction, and with clear notions of resource ownership and access request. Based on that, Boxed Ambients enable a rather direct formalization of classical security policies for resource protection and access control: this is not easy (if at all possible) with Mobile Ambients (see [3]).
- They ease the design of type systems providing precise accounts of ambient be-

havior. As we show in [2], a rather simple structure of types suffices for that purpose. Ambient and process types are defined simply as two-place constructors describing the types of exchanges that may take place locally, and with the enclosing context. Interestingly, this simple type structure is all that is needed to give a full account of ambient interaction. This is a consequence of (i) there being no way for ambients to communicate directly across more than one boundary, and (ii) communication being the only means for ambient to interact. Based on that, the typing of Boxed Ambients provides for more flexibility of communication and mobility than existing type systems for Mobile Ambients.

- Finally, resource locality and directed input/output provide new insight into the relation between the synchronous and asynchronous models of communication. Specifically, the classic π-calculus relations between synchronous and asynchronous output, as stated by Boudol in [1], no longer hold as a result of combining remote communications, resource locality and mobility. More precisely asynchronous output may *no longer* be viewed as a special case of synchronous output with null continuation, neither can it be encoded by structural equivalence, by stipulating that $\langle M\rangle P \equiv \langle M\rangle \mid P$. As we show in [2] these two solutions, which are equivalent in the π-calculus, have rather different consequences in Boxed Ambients.

4. Boxed Ambients and Information Flow

In [3] we describe in detail how it is possible to modify the Boxed Ambient type system in order to use for enforcing classic mandatory access control (MAC) security policies. However it is well know that such policies are not enough to ensure the absence of insecure information flows. They can ensure that a subject will never be allowed to access (either directly or indirectly) the resources it has not the right to access, but they cannot ensure that the subject will not get hold of the information stored in these resources. A classical example is

$$h[\,\ldots \mid (x{:}\mathsf{bool})(\mathsf{if}\ x\ \mathsf{then}\ t[\,]\ \mathsf{else}\ f[\,])\mid c[\ldots]\,]$$

Even though c does not perform an access to the local channel of h it can determine its value by testing (e.g., by exercising an in capability) the presence of the t or f ambient: the value of x flowed from h to c.

In order to determine the presence of such flows one relies on some notion of process equivalence to state a *non-interference* property. The rough idea is to partition agents (and resources) into top-secret and low-secret classes, and define as secure every system in which the behavior of top-secret processes does not "interfere" with the behavior of low secret ones, in the sense that the behavior of the latter does not depend on that of the former.

This needs the definition of some behavioral equivalence on processes. While the presence of the full-fledged communication primitives of [2] is interesting in dealing with MAC policies (as it naturally renders the complete palette of access modes), the definition of behavioral equivalences is greatly simplified by a reduced number of possible interaction. Moreover a tighter control on parent-child com-

munication offers better guarantees of the absence of implicit information flows. Therefore we modify the semantics of Boxed Ambients on the lines of the *shared channel* semantics of [5] and allow only two (instead of four) non-local basic interactions, described by the following reduction rules:

$$(\textit{input } n) \qquad (x)^n P \mid n[\![\langle M \rangle^{\uparrow} Q \mid R]\!] \rightarrow P\{x := M\} \mid n[\![Q \mid R]\!]$$
$$(\textit{output } n) \qquad \langle M \rangle^n P \mid n[\![(x)^{\uparrow} Q \mid R]\!] \rightarrow P \mid n[\![Q\{x := M\} \mid R]\!]$$

We then establish the convention to consider the former as a read access to n and the latter as a write access to n. This corresponds to the the view of an ambient as having two channels: a private channel which is only available for local exchanges, and an "upward channel" which the ambient offers to its enclosing context for read and write access.

Based on this intuition, one then may try to adapt to our framework the definition of *non-interference* first introduced in Goguen and Meseguer's seminal paper [6]. In that paper non-interference was defined as the property that the output of low-secret subjects did not depend on the input of top-secret subjects. In our framework subjects are ambients and their output is easily identifiable with upward communications (being the calculus synchronous the sole observation of the output actions may not suffice). So to study non-interference it is reasonable to use as barbs the actions of upward communication, to define equivalence in terms of weak barbed congruence (weak, as it does not matter "how long" to produce the ouput takes) and define as "interference free" any process whose behavior is insensitive to operations on top-secret names.

References

[1] G. Boudol. Asynchrony and the π-calculus. Research Report 1702, INRIA, http://www-sop.inria.fr/mimosa/personnel/Gerard.Boudol.html, 1992.

[2] M. Bugliesi, G. Castagna, and S. Crafa. Boxed ambients. In *TACS 2001*, number 2215 in LNCS, pages 38–63, Sendai, Japan, 2001. Springer.

[3] M. Bugliesi, G. Castagna, and S. Crafa. Reasoning about security in mobile ambients. In *CONCUR 2001*, number 2154 in LNCS, pages 102–120, 2001. Springer.

[4] L. Cardelli and A. Gordon. Mobile ambients. In *Proceedings of POPL '98*. ACM Press, 1998.

[5] G. Castagna, G. Ghelli, and F. Zappa. Typing mobility in the Seal Calculus. In *CONCUR 2001*, number 2154 in LNCS, pages 82–101, 2001. Springer.

[6] J.A. Goguen and J. Meseguer. Security policy and security models. In *Proceedings of Symposium on Secrecy and Privacy*, pages 11–20. IEEE Computer Society, 1982.

[7] J. Vitek and G. Castagna. Seal: A framework for secure mobile computations. In *Internet Programming Languages*, number 1686 in LNCS, pages 47–77. Springer, 1999.