Mobile Ambients

(Extended Abstract)*

Luca Cardelli
Digital Equipment Corporation
Systems Research Center

Andrew D. Gordon
University of Cambridge
Computer Laboratory

There are two distinct areas of work in mobility: mobile computing, concerning computation that is carried out in mobile devices (laptops, personal digital assistants, etc.), and mobile computation, concerning mobile code that moves between devices (applets, agents, etc.). We aim to describe all these aspects of mobility within a single framework that encompasses mobile agents, the ambients where agents interact and the mobility of the ambients themselves.

This extended abstract of a longer paper [2] presents a minimal calculus of ambients that includes only mobility primitives. Section 1 gives its syntax and introduces its semantics informally. Section 2 gives a formal semantics. Section 3 concludes.

1 Syntax and Informal Semantics

We begin by defining the syntax of the calculus in the following table. The main syntactic categories are processes (including ambients and agents that execute actions) and capabilities.

Mobility and Communication Primitives

n	names
$M ::=$	capability
$\mathit{in}\ n$	can enter into n
$\mathit{out}\ n$	can exit out of n
$\mathit{open}\ n$	can open n

*This extended abstract is to appear in the proceedings of the workshop on *Higher Order Operational Techniques in Semantics*, held at Stanford University, December 8–11, 1997. The authors' current affiliation is Microsoft Research.

$P, Q, R ::=$	process
$(\nu n)P$	restriction
$\mathbf{0}$	inactivity
$P \mid Q$	composition
$!P$	replication
$n[P]$	ambient
$M.P$	action

The first four process primitives (restriction, inactivity, composition and replication) have the same meaning as in the π-calculus [4] namely: restriction is used to introduce new names and limit their scope; $\mathbf{0}$ has no behavior; $P \mid Q$ is the parallel composition of P and Q; and $!P$ is an unbounded number of parallel replicas of P. The main difference with respect to the π-calculus is that names are used to name ambients instead of channels. To these standard primitives we add ambients, $n[P]$, and the exercise of capabilities, $M.P$.

We now introduce the semantics of ambients informally. A reduction relation $P \rightarrow Q$ describes the evolution of a process P into a new process Q.

Ambients

An ambient is written $n[P]$, where n is the name of the ambient, and P is the process running inside the ambient. In $n[P]$, it is understood that P is actively running, and that P can be the parallel composition of several processes. We emphasize that P is running even when the surrounding ambient is moving. Running while moving may or may not be realistic, depending on the nature of the ambient and of the communication medium through which the ambient moves, but it is consistent to think in those terms.

In general, an ambient exhibits a tree structure induced by the nesting of ambient brackets. Each node of this tree structure may contain a collection of (non-ambient) processes running in parallel, in addition to subambients. We say that these processes are running in the ambient, in contrast to the ones running in subambients.

Nothing prevents the existence of two or more ambients with the same name, either nested or at the same level. Once a name is created, it can be used to name multiple ambients. Moreover, $!n[P]$ generates multiple ambients with the same name. This way, for example, one can easily model the replication of services.

Operations that change the hierarchical structure of ambients are sensitive. In particular such operations can be interpreted as the crossing of firewalls or the decoding of ciphertexts. Hence these operations are restricted by capabilities. Thanks to capabilities, an ambient can allow other ambients to perform certain operations without having to reveal its true name.

Actions and Capabilities

The process $M.P$ executes an action regulated by the capability M, and then continues as the process P. The process P does not start running until the action is executed. The reduction rules for $M.P$ depend on the capability M, and are described below case by case.

We consider three kinds of capabilities: one for entering an ambient, one for exiting an ambient and one for opening up an ambient. Capabilities are obtained from names; given a name n, the capability $\mathit{in}\, n$ allows entry into n, the capability $\mathit{out}\, n$ allows exit out of n and the capability $\mathit{open}\, n$ allows the opening of n. Implicitly, the possession of one or all of these capabilities for n is insufficient to reconstruct the original name n.

An entry capability, $\mathit{in}\ m$, can be used in the action $\mathit{in}\ m.P$, which instructs the ambient surrounding $\mathit{in}\, m.P$ to enter a sibling ambient named m. If no sibling m can be found, the operation blocks until a time when such a sibling exists. If more than one m sibling exists, any one of them can be chosen. The reduction rule is:

$$n[\mathit{in}\ m.P \mid Q] \mid m[R] \rightarrow m[n[P \mid Q] \mid R]$$

If successful, this reduction transforms a sibling n of an ambient m into a child of m. After the execution, the process $\mathit{in}\, m.P$ continues with P, and both P and Q find themselves at a lower level in the tree of ambients.

An exit capability, $\mathit{out}\ m$, can be used in the action $\mathit{out}\ m.P$, which instructs the ambient surrounding $\mathit{out}\, m.P$ to exit its parent ambient named m. If the parent is not named m, the operation blocks until a time when such a parent exists. The reduction rule is:

$$m[n[\mathit{out}\ m.P \mid Q] \mid R] \rightarrow n[P \mid Q] \mid m[R]$$

If successful, this reduction transforms a child n of an ambient m into a sibling of m. After the execution, the process $\mathit{in}\, m.P$ continues with P, and both P and Q find themselves at a higher level in the tree of ambients.

An opening capability, $\mathit{open}\, n$, can be used in the action $\mathit{open}\, n.P$. This action provides a way of dissolving the boundary of an ambient named n located at the same level as $\mathit{open}\ n.P$, according to the rule:

$$\mathit{open}\ n.P \mid n[Q] \rightarrow P \mid Q$$

If no ambient n can be found, the operation blocks until a time when such an ambient exists. If more than one ambient n exists, any one of them can be chosen.

2 Formal Semantics

We now give an operational semantics of our calculus, based on a structural congruence between processes, $\equiv$, and a reduction relation $\rightarrow$. This is a semantics in the style of Milner's reaction relation [3] for the π-calculus, which was itself inspired by the Chemical Abstract Machine of Berry and Boudol [1].

We let *structural congruence*, $\equiv$, be the least relation on processes that satisfies the following equations and rules:

Structural Congruence

$P \equiv P$	$P \mid Q \equiv Q \mid P$
$Q \equiv P \Rightarrow P \equiv Q$	$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$
$P \equiv Q, Q \equiv R \Rightarrow P \equiv R$	$!P \equiv P \mid !P$
	$(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$
$P \equiv Q \Rightarrow (\nu n)P \equiv (\nu n)Q$	$(\nu n)(P \mid Q) \equiv P \mid (\nu n)Q$ if $n \notin fn(P)$
$P \equiv Q \Rightarrow P \mid R \equiv Q \mid R$	$(\nu n)(m[P]) \equiv m[(\nu n)P]$ if $n \neq m$
$P \equiv Q \Rightarrow !P \equiv !Q$	$P \mid \mathbf{0} \equiv P$
$P \equiv Q \Rightarrow n[P] \equiv n[Q]$	$(\nu n)\mathbf{0} \equiv \mathbf{0}$
$P \equiv Q \Rightarrow M.P \equiv M.Q$	$!\mathbf{0} \equiv \mathbf{0}$

We let the *reduction relation*, $\rightarrow$, be the least relation on processes to satisfy the following rules:

Reduction

$n[\mathit{in}\ m.P \mid Q] \mid m[R] \rightarrow m[n[P \mid Q] \mid R]$	$P \rightarrow Q \Rightarrow P \mid R \rightarrow Q \mid R$
$m[n[\mathit{out}\ m.P \mid Q] \mid R] \rightarrow n[P \mid Q] \mid m[R]$	$P \rightarrow Q \Rightarrow (\nu n)P \rightarrow (\nu n)Q$
$\mathit{open}\ n.P \mid n[Q] \rightarrow P \mid Q$	$P \rightarrow Q \Rightarrow n[P] \rightarrow n[Q]$
$P' \equiv P, P \rightarrow Q, Q \equiv Q' \Rightarrow P' \rightarrow Q'$	

3 Summary

We presented the syntax and semantics of a minimal calculus of ambients. In the full paper [2] we provide more motivation and intuition, investigate a calculus with additional primitives for I/O, survey related work and include many examples. We discuss how the notion of a mobile ambient captures the structure of complex networks and the behavior of mobile computation. The full calculus is no more complex than common process calculi, but supports reasoning about mobility and, at least to some degree, security.

References

[1] G. Berry and G. Boudol. The chemical abstract machine. *Theoretical Computer Science*, 96(1):217–248, April 1992.

[2] A. D. Gordon and L. Cardelli. Mobile ambients. In *Foundations of System Specification and Computation Structures*, Lecture Notes in Computer Science. Springer-Verlag, 1998. To appear.

[3] R. Milner. Functions as processes. *Mathematical Structures in Computer Science*, 2:119–141, 1992.

[4] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, parts I and II. *Information and Computation*, 100:1–40 and 41–77, 1992.