A Calculus of Mobile Ambients

Luca Cardelli

Digital Equipment Corporation

Systems Research Center

Andrew D. Gordon

University of Cambridge

Computer Laboratory

The Network is the Computer

The Internet: a single, flat, global address space?

No, since we want computing to be secure, ubiquitous and personal.

- Firewalls partition the net into regions.
- Mobile devices move from one net connection to another.
- Mobile agents move with data from one host to another.

A notion of **administrative domain** underlies these situations.

Our goal: a calculus to describe secure and mobile administrative domains, and to suggest flexible ways of programming them.

Design Principles

Some goals:

- a small, expressive calculus of primitive ambients
- an ambient has a boundary—no action at a distance
- an ambient may be nested inside another
- an ambient may move as a whole

Some non-goals:

- primitive global channels (but may be encoded)
- a design optimised for distribution or including node failure
- types

So what's the formalism?

Milner, Parrow and Walker's π-calculus is our starting point:

- **centralised**: contiguous group of communicating processes
- **nominal**: has names and name generation
- communication achieved via **shared names**

Instead, we want a nominal calculus of nested, distributed processes, with no action at a distance, and some account of access control.

A Calculus with Mobility Primitives

$P, Q, R ::=$	process
$\quad (\nu n)P$	restriction
$\quad \mathbf{0}$	inactivity
$\quad P \mid Q$	composition
$\quad !P$	replication
$\quad n[P]$	ambient
$\quad M.P$	action
$M ::=$	term
$\quad \mathit{in}\ n$	can enter n
$\quad \mathit{out}\ n$	can exit n
$\quad \mathit{open}\ n$	can open n

General Form of an Ambient

Nested ambients induce a tree-structure. Each node takes the form:

$$n[\, \underbrace{P_1 \mid \cdots \mid P_q}_{\text{local processes}} \mid \underbrace{m_1[\cdots] \mid \cdots \mid m_p[\cdots]}_{\text{child ambients}}] \qquad P_j \neq p_j[\cdots]$$

Local processes **govern** ambient behaviour.

Via suitable capabilities, a local process:

- may move the whole of ambient n into a sibling
- may move the whole of ambient n out of its parent ambient
- dissolve the boundary of one of the children of ambient n

Reduction I: Mobility

Entering an ambient:

$$n[\mathit{in}\ m.P \mid Q] \mid m[R] \longrightarrow m[n[P \mid Q] \mid R] \qquad \text{(Red In)}$$

Exiting an ambient:

$$m[n[\mathit{out}\ m.P \mid Q] \mid R] \longrightarrow n[P \mid Q] \mid m[R] \qquad \text{(Red Out)}$$

Possession of a capability *in* m or *out* m confers a right of passage across the boundary of ambient m.

Passage preserves the boundary of ambient n.

Reduction II: Open

Opening an ambient:

$$\textit{open}\ n.P \mid n[Q] \longrightarrow P \mid Q \quad \text{(Red Open)}$$

Capability *open* n confers the right to dissolve the boundary of ambient n.

Reduction III: CHAM-style Rules

The following rules complete the definition of $P \longrightarrow Q$:

$P \longrightarrow Q \Rightarrow P \mid R \longrightarrow Q \mid R$	(Red Par)
$P \longrightarrow Q \Rightarrow (\nu n)P \longrightarrow (\nu n)Q$	(Red Res)
$P \longrightarrow Q \Rightarrow n[P] \longrightarrow n[Q]$	(Red Amb)
$P' \equiv P, P \longrightarrow Q, Q \equiv Q' \Rightarrow P' \longrightarrow Q'$	(Red $\equiv$)

Structural Equivalence I

$P \equiv Q$ is an equivalence relation satisfying:

$P \equiv Q \Rightarrow (\nu n)P \equiv (\nu n)Q$	(Struct Res)
$P \equiv Q \Rightarrow P \mid R \equiv Q \mid R$	(Struct Par)
$P \mid Q \equiv Q \mid P$	(Struct Par Comm)
$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$	(Struct Par Assoc)
$!P \equiv P \mid !P$	(Struct Repl Par)
$(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$	(Struct Res Res)
$n \notin \mathit{fn}(P) \Rightarrow (\nu n)(P \mid Q) \equiv P \mid (\nu n)Q$	(Struct Res Par)

Structural Equivalence II

Moreover, $P \equiv Q$ satisfies:

$$P \equiv Q \Rightarrow n[P] \equiv n[Q] \qquad \text{(Struct Amb)}$$

$$n \neq m \Rightarrow (\nu n)m[P] \equiv m[(\nu n)P] \qquad \text{(Struct Res Amb)}$$

But note that in general:

$$n[P] \mid n[Q] \;\not\equiv\; n[P \mid Q]$$

$$!n[P] \;\not\equiv\; n[!P]$$

Variation: Subjective versus Objective Moves

Subjective: "I move. I become a child ambient."

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

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

Objective: "I make you move. You become a local process."

$$\mathit{mv\ in}\ n.P \mid n[Q] \;\longrightarrow\; n[P \mid Q]$$

$$n[\mathit{mv\ out}\ n.P \mid Q] \;\longrightarrow\; P \mid n[Q]$$

Variation: Open versus Acid

Ambient acid: "I dissolve my own boundary."

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

Objective moves derivable:

$$\mathit{mv}\ \mathit{in}\ n.P \;\triangleq\; (\nu q)q[\mathit{in}\ n.\mathit{acid}.P]$$

$$\mathit{mv}\ \mathit{out}\ n.P \;\triangleq\; (\nu q)q[\mathit{out}\ n.\mathit{acid}.P]$$

But the risk is that acid allows ambient kidnap:

$$\mathit{entrap}\ m \;\triangleq\; (\nu k\ q)k[] \mid q[\mathit{in}\ m.\mathit{acid}.\mathit{in}\ k.\mathbf{0}]$$

$$\mathit{entrap}\ m \mid m[P] \;\rightarrow^*\; (\nu k)k[m[P]]$$

Ambients as Administrative Domains

An ambient is like a republic with border guards:

- Local processes govern movement of the ambient.
- Disposition of capabilities governs movement across the border.
- Capabilities *in* $\mathfrak{n}$ and *out* $\mathfrak{n}$ are like visas.

  (But acid or objective moves turn them into charters to govern.)
- Capability *open* $\mathfrak{n}$ lets the local processes of an ambient decide who may join them.

Encoding Locks

We can use *open* to encode locks:

$$\textit{acquire}\; n.P \triangleq \textit{open}\; n.P$$

$$\textit{release}\; n.P \triangleq n[] \mid P$$

Two processes may synchronise as follows:

$$\textit{acquire}\; n.\textit{release}\; m.P \mid \textit{release}\; n.\textit{acquire}\; m.Q$$

Encoding Objective Moves I

$$
\begin{aligned}
\textit{allow}\ n &\triangleq\ !\textit{open}\ n \\
\textit{mv in}\ n.P &\triangleq (\nu k)k[\textit{in}\ n.\mathrm{in}[\textit{out}\ k.P]] \\
\textit{mv out}\ n.P &\triangleq (\nu k)k[\textit{out}\ n.\mathrm{out}[\textit{out}\ k.P]] \\
n^{\downarrow}[P] &\triangleq n[P \mid \textit{allow}\ \mathrm{in}] \\
n^{\uparrow}[P] &\triangleq n[P] \mid \textit{allow}\ \mathrm{out} \\
n^{\updownarrow}[P] &\triangleq n[P \mid \textit{allow}\ \mathrm{in}] \mid \textit{allow}\ \mathrm{out}
\end{aligned}
$$

Encoding Objective Moves II

We get:

$$\begin{array}{rcl} \textit{mv in}\, n.P \mid n^{\updownarrow}[Q] & \rightarrow & n^{\updownarrow}[(\nu k)k[in[\textit{out}\; k.P]] \mid Q] \\ & \rightarrow & n^{\updownarrow}[(\nu k)k[] \mid in[P] \mid Q] \\ & \rightarrow & n^{\updownarrow}[(\nu k)k[] \mid P \mid Q] \\ & \approx & n^{\updownarrow}[P \mid Q] \end{array}$$

Similarly:

$$n^{\updownarrow}[\textit{mv out}\; n.P \mid Q] \rightarrow^{*}\approx P \mid n^{\updownarrow}[Q]$$

Synchronisation on Named Channels

We represent a named CCS channel by the ambient,

$$\mathrm{n}^{\updownarrow}[]$$

and synchronisation on complementary ports $\mathrm{n}?$ and $\mathrm{n}!$ by:

$$\begin{aligned} \mathrm{n}?.\mathrm{P} &\triangleq \textit{mv in}\ \mathrm{n}.\textit{acquire}\ \mathrm{rd}.\textit{release}\ \mathrm{wr}.\textit{mv out}\ \mathrm{n}.\mathrm{P} \\ \mathrm{n}!.\mathrm{P} &\triangleq \textit{mv in}\ \mathrm{n}.\textit{release}\ \mathrm{rd}.\textit{acquire}\ \mathrm{wr}.\textit{mv out}\ \mathrm{n}.\mathrm{P} \end{aligned}$$

Encoding Boolean Flags

$$\mathit{flag}\ n \triangleq n[\,]$$

$$\begin{aligned}
&\mathit{if}\ tt\ P, \mathit{if}\ ff\ Q \triangleq \\
&\quad (\nu k)(k[\,] \mid \\
&\qquad \mathit{open}\ tt.\mathit{open}\ k.(\nu t)(ff[t[\,]] \mid \mathit{open}\ t.P) \mid \\
&\qquad \mathit{open}\ ff.\mathit{open}\ k.(\nu f)(tt[f[\,]] \mid \mathit{open}\ f.Q))
\end{aligned}$$

We have:

$$\mathit{flag}\ tt \mid \mathit{if}\ tt\ P, \mathit{if}\ ff\ Q \ \rightarrow^* \approx \ P$$

$$\mathit{flag}\ ff \mid \mathit{if}\ tt\ P, \mathit{if}\ ff\ Q \ \rightarrow^* \approx \ Q$$

Turing Machines

Idea: tape looks like $\mathrm{end}^{\updownarrow}[\mathrm{ff}[] \mid \mathrm{sq}^{\updownarrow}[\mathrm{ff}[] \mid \mathrm{sq}^{\updownarrow}[\mathrm{ff}[] \mid \mathrm{sq}^{\updownarrow}[\cdots]]]]$.

$$
\begin{aligned}
\textit{head} \quad &\triangleq \quad \mathrm{head}^{\updownarrow}[!\textit{open}\, S_1.\textit{mv out}\ \mathrm{head}.\\
&\qquad\qquad \textit{if}\ \mathrm{tt}\ (\mathrm{ff}[] \mid \textit{mv in}\ \mathrm{head}.\textit{in}\ \mathrm{sq}.S_2[]),\\
&\qquad\qquad \textit{if}\ \mathrm{ff}\ (\mathrm{tt}[] \mid \textit{mv in}\ \mathrm{head}.\textit{out}\ \mathrm{sq}.S_3[]) \mid\\
&\qquad\quad \ldots \mid\\
&\qquad\quad S_1[]]\\
\textit{stretchRht} \quad &\longrightarrow^* \quad \mathrm{sq}^{\updownarrow}[\mathrm{ff}[] \mid \textit{stretchRht}]\\
\textit{machine} \quad &\triangleq \quad \mathrm{end}^{\updownarrow}[\mathrm{ff}[] \mid \textit{head} \mid \textit{stretchRht}]
\end{aligned}
$$

Extensions for I/O and Paths

We include **variables**, x, and **names**, n, **null**, ϵ, and **paths**, $M.M'$, as terms.

We include **inputs**, $(x).P$, and **outputs**, $\langle M \rangle$, as processes.

$$
\begin{array}{rcll}
\epsilon.P & \equiv & P & \text{(Struct } \epsilon\text{)} \\
(M.M').P & \equiv & M.M'.P & \text{(Struct .)} \\
\langle M \rangle \mid (x).P & \rightarrow & P\{x \leftarrow M\} & \text{(Red I/O)}
\end{array}
$$

Routable Packets

$$\textit{packet}\ \mathrm{pkt} \triangleq \mathrm{pkt}[!(x).x \mid !\textit{open}\ \mathrm{route}]$$

$$\textit{route}\ \mathrm{pkt}\ \textit{with}\ P\ \textit{to}\ C \triangleq \mathrm{route}[\textit{in}\ \mathrm{pkt}.\langle C\rangle \mid P]$$

$$\textit{forward}\ \mathrm{pkt}\ \textit{to}\ C \triangleq \textit{route}\ \mathrm{pkt}\ \textit{with}\ \mathbf{0}\ \textit{to}\ C$$

Example:

$$a[\textit{packet}\ \mathrm{pkt} \mid \textit{route}\ \mathrm{pkt}\ \textit{with}\ \mathrm{data}[\langle \mathrm{msg}\rangle]\ \textit{to}\ (\textit{out}\ a.\textit{in}\ b.\textit{in}\ c)]] \mid$$

$$b[c[\textit{forward}\ \mathrm{pkt}\ \textit{to}\ (\textit{out}\ c.\textit{in}\ d)] \mid$$

$$d[\cdots]]$$

Gaining Entry Behind a Firewall

There is a previously arranged password, k.

The password is verified in the single-entry room r.

Access behind the firewall is given in the room w.

$$
\begin{array}{rcl}
\textit{Firewall} & \triangleq & (\nu w\, r)(\langle r \rangle \mid r[\textit{open}\, k.\textit{in}\, w] \mid w[\textit{open}\, r.P]) \\
\textit{Agent} & \triangleq & (x).k[x.Q] \\
\textit{Agent} \mid \textit{Firewall} & \longrightarrow^{*} & (\nu w)w[Q \mid P]
\end{array}
$$

Any agent gaining entry behind the firewall either knows k,

or is helped by someone who does.

Communication on Named Channels

Encoding the choice-free asynchronous π-calculus:

$$
\begin{aligned}
(\textit{new}\; n)P &\triangleq (\nu n)(n^{\updownarrow}[] \mid P) \\
\overline{n}\langle M\rangle &\triangleq \textit{mv in}\; n.\langle M\rangle \\
n(x).P &\triangleq \textit{mv in}\; n.(x).\textit{mv out}\; n.P
\end{aligned}
$$

Hence (and also directly) we can encode the λ-calculus.

Encoding the Spi Calculus

Abadi and Gordon's spi calculus is a value-passing π-calculus.

Names play the role of encryption keys; restriction models secrecy.

A value, V, either an S-expression or a ciphertext, may be coded as an ambient $\overline{V}$:

$$\begin{aligned}
\overline{\#n} &\triangleq \langle n \rangle \\
\overline{(V_1, V_2)} &\triangleq fst^{\downarrow}[\overline{V_1}] \mid snd^{\downarrow}[\overline{V_2}] \\
\overline{\{V\}_K} &\triangleq K[\overline{V}]
\end{aligned}$$

Encoding destructors (projection, decryption) is easy.

Basic Theory of Ambients

Reduction characterised by an LTS: $P \rightarrow Q$ iff $P \xrightarrow{\tau} \equiv Q$.

Morris-style **contextual equivalence**:

$$
\begin{aligned}
P \downarrow n &\triangleq \exists P_1, P_2, \vec{m}(P \equiv (\nu \vec{m})(P_1 \mid n[P_2]) \,\&\, n \notin \{\vec{m}\}) \\
P \Downarrow &\triangleq \exists Q, n(P \rightarrow^* Q \,\&\, Q \downarrow n) \\
P \approx Q &\triangleq \forall \text{contexts } \mathcal{C}(\mathcal{C}[P]\Downarrow \Leftrightarrow \mathcal{C}[Q]\Downarrow)
\end{aligned}
$$

The **Perfect Firewall Equation**: $(\nu n)n[P] \approx \mathbf{0}$, if $n \notin \mathit{fn}(P)$.

Some Related Formalisms

- The π-calculus (Milner, Parrow, Walker)
- Localities and failures (Amadio, Prasad)
- The distributed join calculus (Fournet, Gonthier, et al)
- Locality based Linda (De Nicola, Ferrari, Pugliese)
- λdist (Sekiguchi, Yonezawa)
- Distributed π-calculus (Sewell)
- The spi calculus (Abadi, Gordon)

Summary

Ambients are hierarchical, mobile, administrative domains, with no action at a distance.

Ambients incorporate both **classical formalisms** (TMs, π, λ) and **net-centric concepts** (distributed devices, packets, mobile code, encryption).

Theory based on both CHAM and LTS paradigms.

An implementation is in progress (Cardelli).