Mobile Ambients

Luca Cardelli

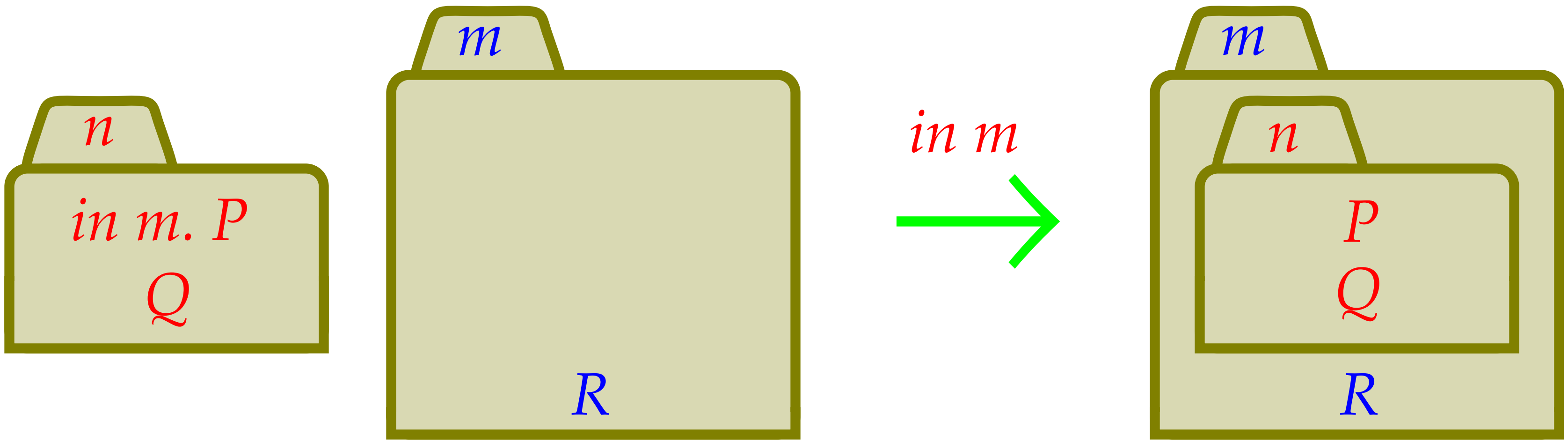
Andrew D. Gordon

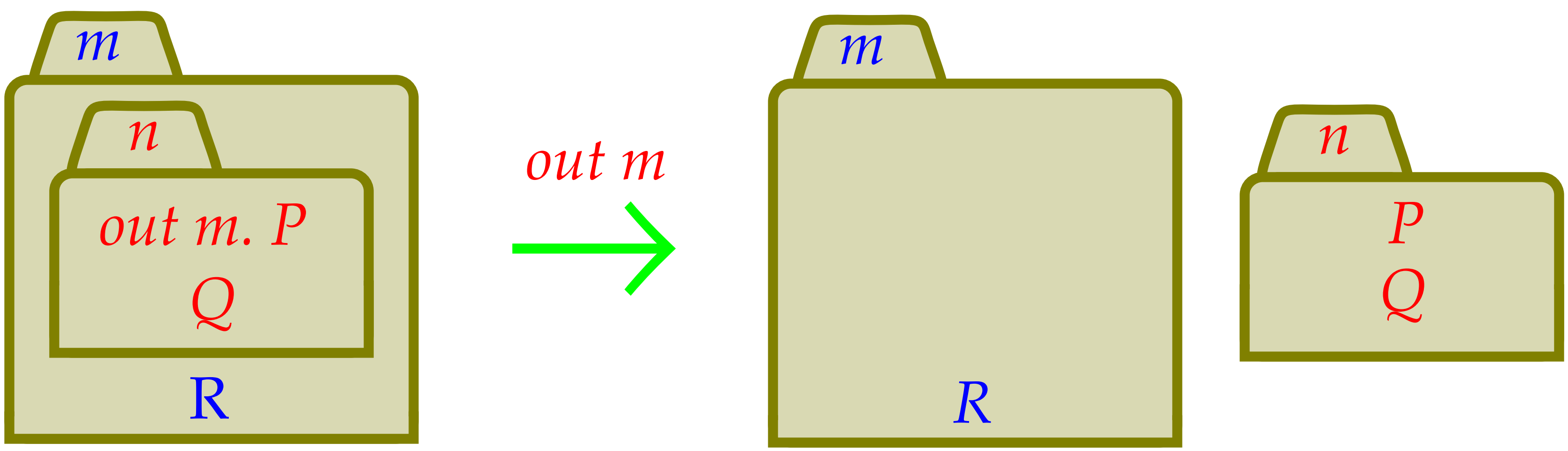
Status Report

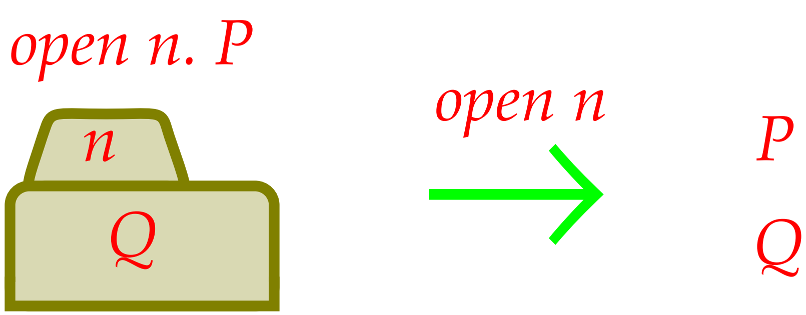
- Goal of the week:
  - ~ Settle on a set of ambient primitives.
  - ~ Study their practical and theoretical expressive power.
- Esthetics for the week:
  - ~ Small size. Theoretical power. Cute examples.
  - ~ "It's so advanced, it's simple".

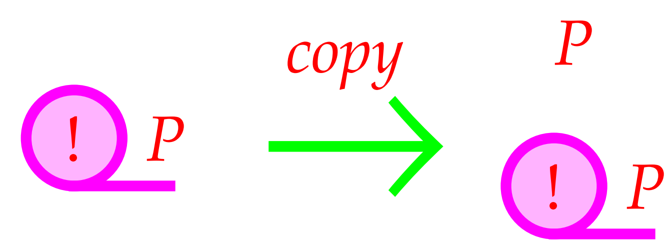
- Outcome:
  - A paper draft, a bunch of examples, a few (almost-) theorems.
  - Surprise: the primitives invented for mobility ended up being meaningful for cryptography. The combination of mobility and cryptography in the same formal framework seems novel and intriguing.
  - E.g.: we have a simple example of an agent authenticating itself with a firewall, obtaining a pass (securely), and then “physically” crossing the firewall.

Ambient Dynamics









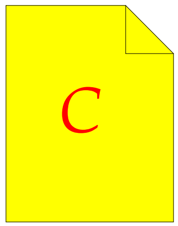
Comments

- We can look at ambients as **active folders**; each folder has a name on its tab, and can contain other folders. Each folder can also contain a whole bunch of concurrent **gremlins** that tell the folder what do and where to go. Each horizontal script line in a folder represent one (or more) gremlins.
- A folder with dynamic content can send out gremlins to find information, represented by other folders, and persuade those folders to follow the gremlins to their home folder.
- The *open* operation throws away a folder and spills its content into the current folder (where *open* $n.P$ lives). It requires a capability *open* n, that must have been given out by folder n.
- The ! operation is a copy machine: if P is a folder, $!P$ can make as many copies of P as desired.
- All transitions block when they cannot fire.
- The $!P$ transition never blocks: it is a very idealized copy machine that never breaks and never runs out of paper. However, copying takes computation, so we can imagine that the operation is blocked until a new copy of P has been produced.

The set of operations on this slide (including folder creation) is Turing-complete.

Ambient I/O

Input: $(x).\ P\{x\}$

Output: 

$\xrightarrow{\textit{read}}$ $P\{C\}$

A Post-it can hold a *capability*:

n	a name
$in\ n$	an action capability
$out\ n$	an action capability
$open\ n$	an action capability
$C.\ C'$	a path (e.g.: $out\ n.\ in\ m$)

Ambient Expressions

$P ::=$	an activity	
$(\nu n)\ P$	new name in a scope	standard in process calculi
0	inactivity	
$P \mid P$	parallel	
$!P$	replication	
$n[P]$	ambient	ambient-specific
$C.\ P$	exercise of a capability	
$(n).\ P$	input from ether, bind to n	ambient I/O
$\langle C \rangle$	output to ether (async)	
$C ::=$	a capability	
$in\ n$	entry capability	basic capabilities
$out\ n$	exit capability	
$open\ n$	open capability	
n	name or input variable	useful with I/O
$C.\ C'$	path	

Ambients as Mobile Processes

- $tourist \triangleq (x).\ joe[x.\ Enjoy]$
- $ticket\text{-}desk \triangleq \langle in\ AF81atSFO.\ out\ AF81atCDG \rangle$

Ambients as Locks

- $\textit{release n and do Q} \;\triangleq\; n[\;] \mid Q$
- $\textit{acquire n then do P} \;\triangleq\; \textit{open } n.\, P$

Ambients as Firewalls

- $n[P]$ is a firewall called n protecting P.
- $in\ n$ is the capability needed to enter the firewall.
- $out\ n$ is the capability needed to exit the firewall.
- The *context* is the Internet.

- The <u>Perfect-Firewall Equation</u>:

$$(\nu n)\, n[P] \;\approx\; 0 \qquad \text{(if } n \text{ not in } P\text{)}$$

Ambients as Ciphertext

- $k[\langle txt \rangle]$ is the plaintext txt encrypted with key k.
- $open\ k$ is the capability needed to open a k-envelope, i.e. to decrypt for k (without knowing k).
- $in\ k$ is the capability needed to put stuff in a k-envelope, i.e. to encrypt for k (without knowing k).
- The *context* is the attacker.

$P \approx Q$ == no attacker can tell P from Q

- The <u>Perfect-Cipher Equation</u>:

$$(\nu k_1)\, k_1[\langle txt_1\rangle] \;\approx\; (\nu k_2)\, k_2[\langle txt_2\rangle]$$

~ because $(\nu k_1)\, k_1[\langle txt_1\rangle] \approx 0 \approx (\nu k_2)\, k_2[\langle txt_2\rangle]$.

Firewall Access

- (Very simplified.) Assume that the shared key k is already known to the firewall and the client.

$$Wally \triangleq (\nu\, w\, r)\, (\langle in\ r\rangle \mid r[open\ k.\ in\ w] \mid w[open\ r.\ P])$$

$$Cleo \triangleq (x).\ k[x.\ C]$$

$$\begin{aligned}
&Cleo \mid Wally \\
&\quad = (\nu\, w\, r)\, (\ (x).\ k[x.\ C] \mid \langle in\ r\rangle \mid r[open\ k.\ in\ w] \mid w[open\ r.\ P]\ ) \\
&\quad = (\nu\, w\, r)\, (\ k[in\ r.\ C] \mid r[open\ k.\ in\ w] \mid w[open\ r.\ P]\ ) \\
&\quad = (\nu\, w\, r)\, (\ r[k[C] \mid open\ k.\ in\ w] \mid w[open\ r.\ P]\ ) \\
&\quad = (\nu\, w\, r)\, (\ r[C \mid in\ w] \mid w[open\ r.\ P]\ ) \\
&\quad = (\nu\, w\, r)\, (\ w[r[C] \mid open\ r.\ P]\ ) \\
&\quad = (\nu\, w)\ \ (\ w[C \mid P]\ )
\end{aligned}$$

Comments

- Two secret names are introduced: w is the name of the firewall, and r is the name of a private room used as a customs checkpoint.
- We want to verify that Cleo knows the key k: this is done by *open* k. After that, we want to give Cleo a capability *in* w to enter the firewall. The communication of this capability must happen in a private place: we don't want some other process to snatch *in* w in transit. The private room r is used for this purpose.
- The room r has a secret name, and a single capability *in* r is made available for entering the room. Therefore we are sure that only one process enters r (we assume that Cleo is honest).

Turing Machine

$end[extendLft \mid S_0 \mid$

$square[S_1 \mid$

$square[S_2 \mid$

$\ldots$

$square[S_i \mid head \mid$

$\ldots$

$square[S_n \mid extendRht] \,..\,] \,..\,]]]$