Mobility and Security

Luca Cardelli

Microsoft Research

Marktoberdorf
"Foundations of Secure Computation"
July 28 - August 7, 1999

OUTLINE

- Wide Area Computation
- The Ambient Calculus (with Andrew Gordon)
- Equivalences (with Andrew Gordon)
- Types (with Giorgio Ghelli and Andrew Gordon)
- Logics (with Andrew Gordon)

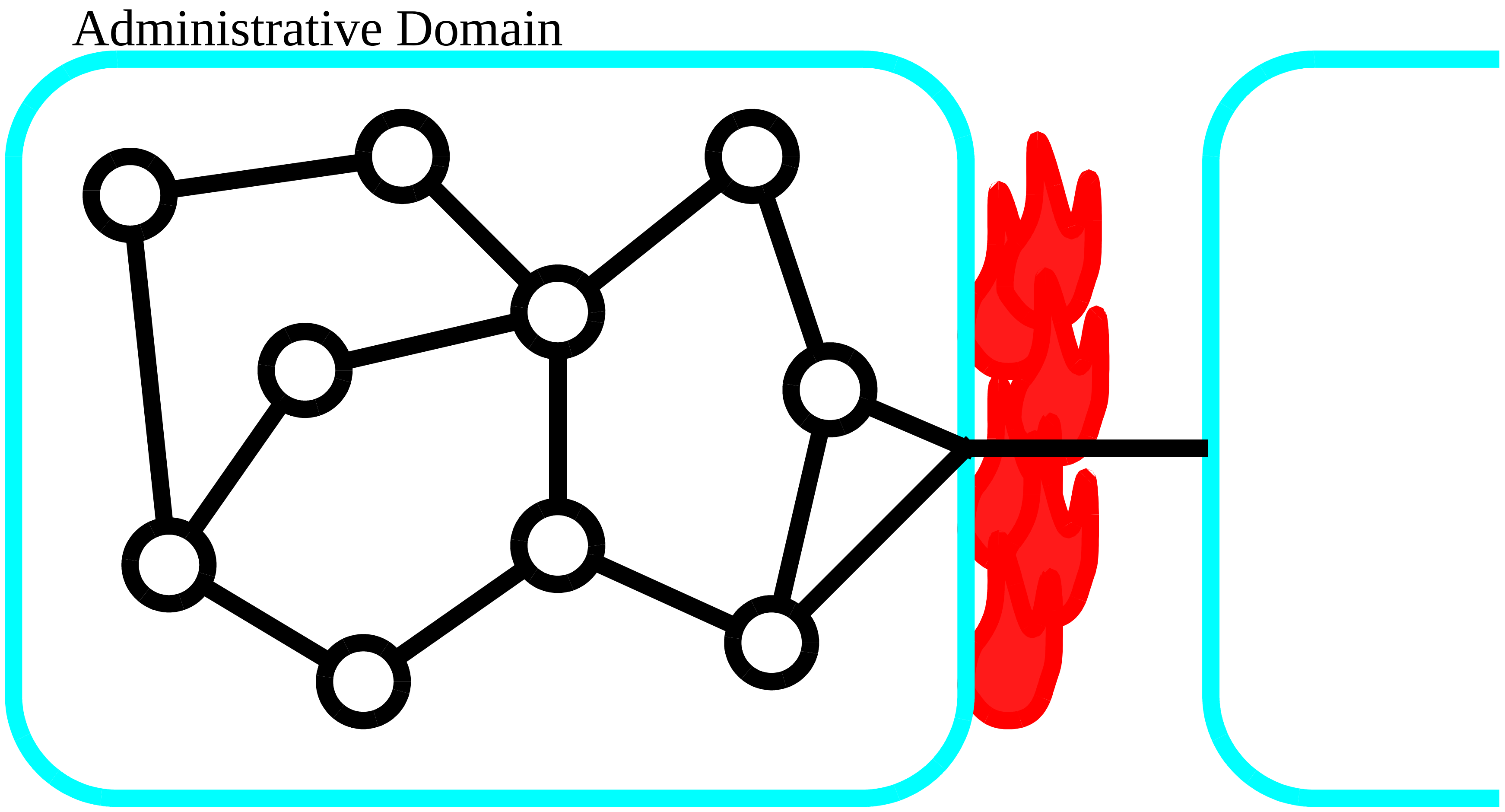
WIDE AREA COMPUTATION

Three Mental Pictures

Three views of computation:

- Local area networks
- Wide area networks
- Mobile networks

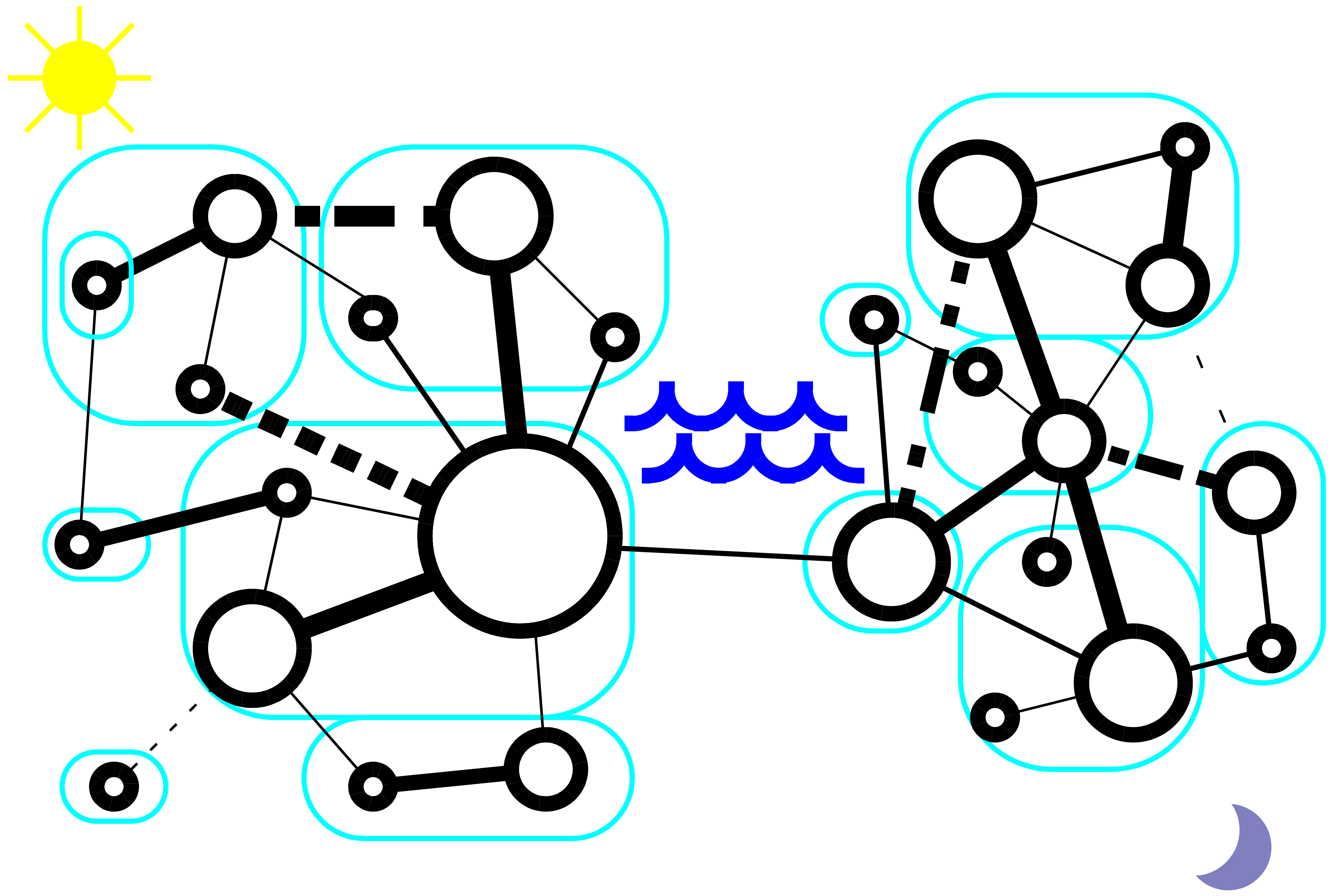
1: LANs and (Traditional) Distributed Computing



LAN Characteristics

- Static and often trivial topology (everything 1 logical step apart).
- Immobility (initially) of software, computers, and links.
- Software evolution: control mobility, data mobility, link mobility.
- Hardware evolution: laptops, wireless LANs.
- Traditional Distributed Object Systems: RPC/RMI.
  (CORBA, OLE, Modula-3 Network Objects, Java RMI.)
- Traditionally, no code mobility, no thread/process mobility.
- More recently, pre-Web: code mobility (Tcl), agent mobility (Tele-script), closure mobility (Obliq).

2: The Web



WAN Characteristics

- Internet/Web: a federated WAN infrastructure that spans the planet. We would like to program it.
- Unfortunately, federated WANs violate many familiar assumptions about the behavior of distributed systems.
- Three phenomena that remain largely hidden in LANs become readily observable:
  - ***Virtual locations***.
  - ***Physical locations***.
  - ***Bandwidth fluctuations***.
- Another phenomenon becomes unobservable:
  - ***Failures***.

A WAN is not a big LAN

- To emulate a LAN on top of a WAN we would have to:
  - ***(A) Hide virtual locations***. By semi-transparent security. But is it possible to guarantee the integrity of mobile code?
  - ***(B) Hide physical locations.*** Cannot "hide" the speed of light, other than by slowing down the whole network.
  - ***(C) Hide bandwidth fluctuations***. Service guarantees eliminate bandwidth fluctuations, but introduce access failures.
  - ***(D) Reveal failures.*** Impossible in principle, since the Web is an asynchronous network.
- Hard problems: (A) may be unsolvable for mobile code; (B) is only solvable (in full) by introducing unacceptable delays; (C) can be solved in a way that reduces it to (D); (D) is unsolvable in principle, while probabilistic solutions run into (B).

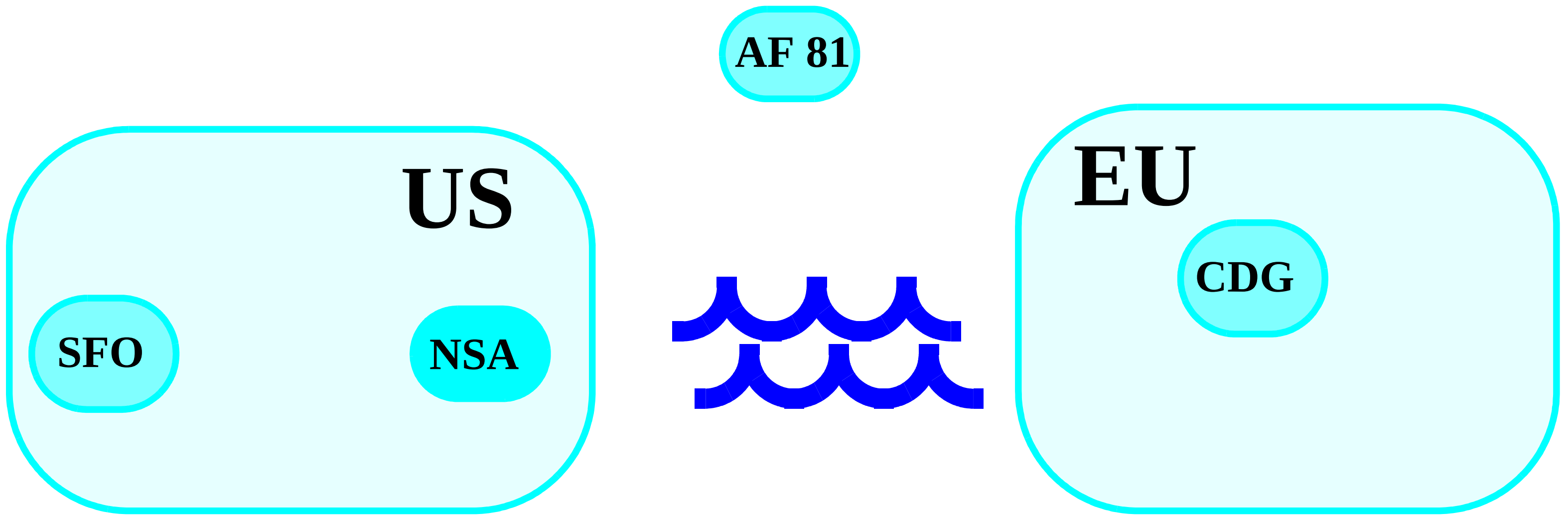
Observables

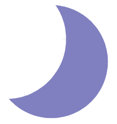
- WAN *observables* are different (and not reducible to) LAN observables.
- Observables determine programming constructs, and therefore influence programs and programming languages.
- We need a complete set of programming constructs that can detect and react to the available observables and, of course, we do not want programming constructs that attempt to detect or react to non-observables.
- Something has already emerged to cope with these observables: Mobile Computation.

Mobile Computation

- Mobile computation can cope with the observables characteristic of a wide-area network such as the Web.
  - ***Virtual locations***. Trust mechanisms to cross virtual barriers.
  - ***Physical locations***. Mobility to optimize placement.
  - ***Bandwidth fluctuations***. Mobility to split applications and establish optimized communication protocols.
  - ***Failures***. Running around or away from failures.
- Code mobility, post-Web.
  - Basic Java Applets.
  - Countless Tcl-based and Java-based ongoing projects.
  - Still no (native) thread mobility.
    (Many attempts; see Agent Mobility meetings)

3: Mobile Computing





- Mobile devices also move computations. In this sense, we cannot avoid the issues raised by mobile computation.

Hardware Mobility

- Lots of gadgets

  Smart cards (wired).
  Active badges, pagers (wireless).
  Cellphones, GPS receivers (wireless).
  Palm/Laptops (wired, wireless).

- Nested gadgets and networks, some mobile:

  Personal gadgets
  Personal Area Networks
  Car, train, ship, airplane local networks

Two Overlapping Views of Mobility

- Mobile Computing.
  - I.e. mobile hardware, physical mobility.
- Mobile Computation.
  - I.e. mobile software, virtual mobility.
- But the borders are fuzzy:
  - Agents may move by traversing a network (virtually), or by being carried on a laptop (physically).
  - Computers may move by lugging them around (physically), or by telecontrol software (virtually).
  - Boundaries may be physical (buildings) or virtual (firewalls).

A Unifying Difficulty: Security Barriers

- A (nasty) fundamental change in the way we compute.
  - Bye bye, flat IP addressing, transparent routing.
  - Bye bye, single universal address space.
  - Bye bye, transparent distributed object systems.
  - Bye bye, roaming agents.
  - Bye bye, *action-at-a-distance computing*.
- Big firewalls (for intranets), small firewalls (for applets). Becoming pervasive: 1 PC Firewall = $99.95.
- Firewall are designed impede access. But we need to make rightful access simple.

Mobility Postulates

- Separate locations exist. They may be difficult to reach.
- Since different locations have different properties, both people and programs (and sublocations!) will want to move between them.
- Barriers to mobility will be erected to preserve certain properties of certain locations.
- Some people and some programs will still need to cross those barriers.

This is the situation wide area computing has to cope with.

Related Work

- Broadly classifiable in two categories:
  - Agents (Actors, Process Calculi, Telescript, etc.)
  - Spaces (Linda, Distributed Lindas, JavaSpace, etc.)
- With our work on Ambients, we aim to unify and extend those basic concepts.

Modeling Mobility

Modeling Mobility

- It’s all about barriers:
  - Locality = **barrier topology**.
  - Process mobility = **barrier crossing**.
  - Security = **(In)ability to cross barriers**.
  - Interaction by **shared position within a barrier**, with no action at a distance.

Formalisms for Concurrency/Distribution

- CSP/CCS. (Static/immutable connectivity.)
- π-calculus. (Channel mobility.)
  N.B. "mobility" in this context is not process mobility.
- Process mobility is reduced to channel mobility.

- Ambient Calculus:
  Process mobility = Barrier crossing.

... in particular, π

- In the π-calculus (our starting point):
  - processes exist in a single **contiguous** location
  - interaction is by **shared names**, used as I/O channels
  - there is no direct account of access control

- In our ambient calculus:
  - processes exist in multiple **disjoint** locations
  - interaction is by **shared position**, with no action at a distance
  - **capabilities**, derived from ambient names, regulate access

Formalisms for Locality

- Join calculus. (Channel mobility and locality.)
- Various calculi with failure. (Locality = Partial Failure.)

- Ambient calculus:
  Locality = Barrier topology.

Formalisms for Security

- (BAN logic, etc.)
- Spi-calculus. (Channel mobility and cryptography)

- Ambient calculus:
  Security = (In)ability to cross barriers.

Ambient Calculus

Approach

- We want to capture in an abstract way, notions of locality, of mobility, and of ability to cross barriers.
- An *ambient* is a place, delimited by a boundary, where computation happens.
- Ambients have a *name*, a collection of local *processes*, and a collection of *subambients*.
- Ambients can move in an out of other ambients, subject to *capabilities* that are associated with ambient names.
- Ambient names are unforgeable (as in π and spi).

The Ambient Calculus

$P ::=$	$(\nu n)\ P$	new name *n* in a scope
	0	inactivity
	$P \mid P$	parallel
	$!P$	replication
	$M[P]$	ambient
	$M.P$	exercise a capability
	$(n).P$	input locally, bind to *n*
	$\langle M\rangle$	output locally (async)
$M ::=$	n	name
	in M	entry capability
	out M	exit capability
	open M	open capability
	ε	empty path
	$M.M'$	composite path

scoping

standard in process calculi

data structures

ambient-specific

actions

ambient I/O

basic capabilities

useful with I/O

Semantics

- Behavior
  - The semantics of the ambient calculus is given in non-deterministic “chemical style” (as in Berry&Boudol’s Chemical Abstract Machine, and in Milner’s π-calculus).
  - The semantics is factored into a reduction relation $P \longrightarrow P'$ describing the evolution of a process P into a process P', and a process equivalence indicated by $Q \equiv Q'$.
  - Here, $\longrightarrow$ is real computation, while $\equiv$ is “rearrangement”.
- Equivalence
  - On the basis of behavior, a substitutive *observational equivalence*, $P \approx Q$, is defined between processes.
  - Standard process calculi reasoning techniques (context lemmas, bisimulation, etc.) can be adapted.

Parallel

- Parallel execution is denoted by a binary operator:

$$P \mid Q$$

- It is commutative and associative:

$$\begin{aligned} P \mid Q &\equiv Q \mid P \\ (P \mid Q) \mid R &\equiv P \mid (Q \mid R) \end{aligned}$$

- It obeys the reduction rule:

$$P \longrightarrow Q \quad \Rightarrow \quad P \mid R \longrightarrow Q \mid R$$

Replication

- Replication is a technically convenient way of representing iteration and recursion.

$$!P$$

- It denotes the unbounded replication of a process P.

$$!P \quad \equiv \quad P \mid !P$$

- There are no reduction rules for $!P$; in particular, the process P under ! cannot begin to reduce until it is expanded out as $P|!P$.

Restriction

- The restriction operator creates a new (forever unique) ambient name n within a scope P.

$$(\nu n)P$$

- As in the π-calculus, the (νn) binder can float as necessary to extend or restrict the scope of a name. E.g.:

$$(\nu n)(P \mid Q) \quad \equiv \quad P \mid (\nu n)Q \qquad \text{if } n \notin fn(P)$$

- Reduction rule:

$$P \longrightarrow Q \quad \Rightarrow \quad (\nu n)P \longrightarrow (\nu n)Q$$

Inaction

- The process that does nothing:

$$\mathbf{0}$$

- Some garbage-collection equivalences:

$$\begin{aligned} P \mid \mathbf{0} &\equiv P \\ !\mathbf{0} &\equiv \mathbf{0} \\ (\nu n)\mathbf{0} &\equiv \mathbf{0} \end{aligned}$$

- This process does not reduce.

Ambients

- An ambient is written as follows, where n is the name of the ambient, and P is the process running inside of it.

$$n[P]$$

- In $n[P]$, it is understood that P is actively running:

$$P \longrightarrow Q \quad \Rightarrow \quad n[P] \longrightarrow n[Q]$$

- Multiple ambients may have the same name, (e.g., replicated servers).

Actions and Capabilities

- Operations that change the hierarchical structure of ambients are sensitive. They can be interpreted as the crossing of firewalls or the decoding of ciphertexts.
- Hence these operations are restricted by *capabilities*.

$$M.\ P$$

  This executes an action regulated by the capability M, and then continues as the process P.

- The reduction rules for $M.\ P$ depend on M.

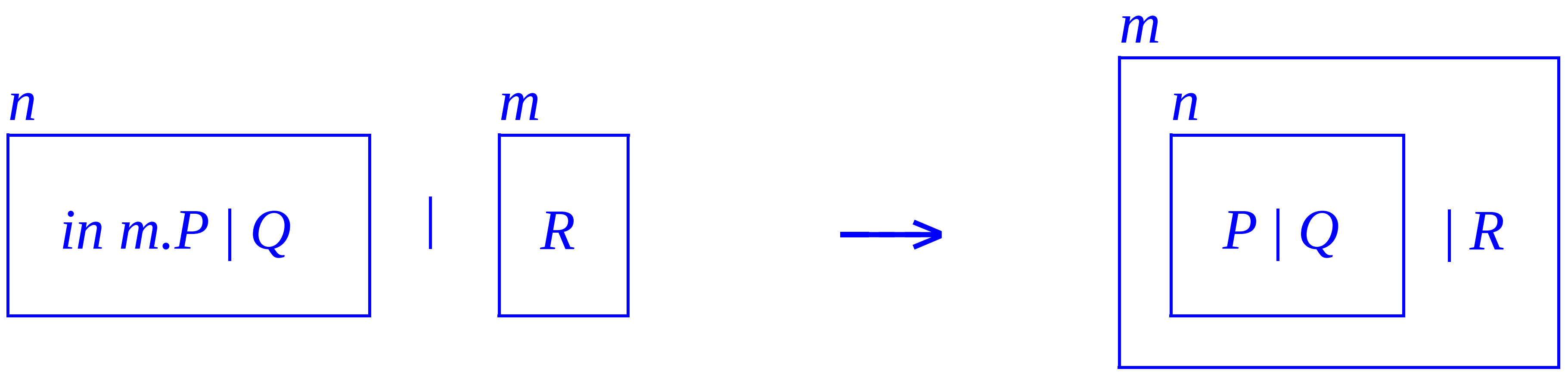
Entry Capability

- An entry capability, *in m*, can be used in the action:

$$in\ m.\ P$$

- The reduction rule (non-deterministic and blocking) is:

$$n[in\ m.\ P \mid Q] \mid m[R] \quad \longrightarrow \quad m[n[P \mid Q] \mid R]$$



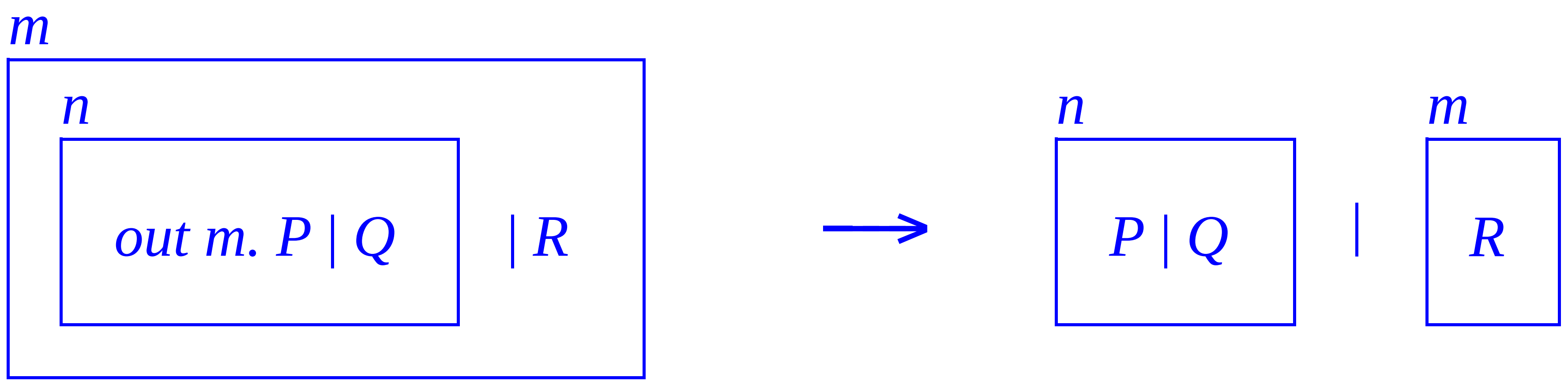
Exit Capability

- An exit capability, *out m*, can be used in the action:

$$out\ m.\ P$$

- The reduction rule (non-deterministic and blocking) is:

$$m[n[out\ m.\ P \mid Q] \mid R] \ \longrightarrow\ n[P \mid Q] \mid m[R]$$



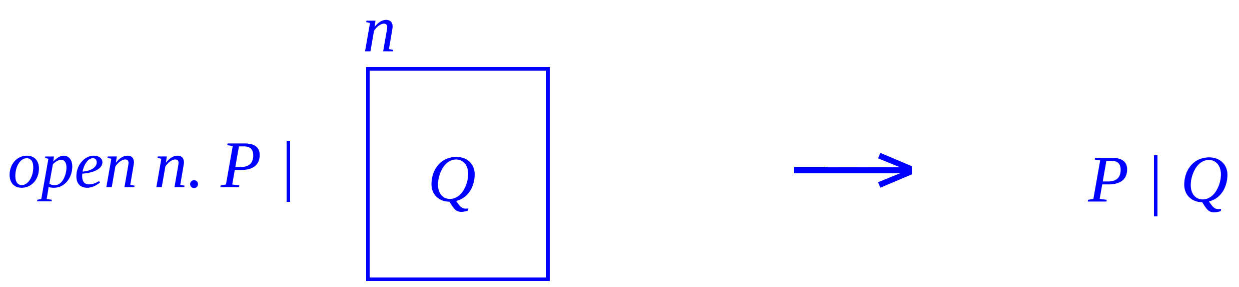
Open Capability

- An opening capability, *open m*, can be used in the action:

$$open\ n.\ P$$

- The reduction rule (non-deterministic and blocking) is:

$$open\ n.\ P \mid n[Q] \quad \longrightarrow \quad P \mid Q$$



- An *open* operation may be upsetting to both P and Q above.
  - From the point of view of P, there is no telling in general what Q might do when unleashed.
  - From the point of view of Q, its environment is being ripped open.
- Still, this operation is relatively well-behaved because:
  - The dissolution is initiated by the agent *open* $n.\ P$, so that the appearance of Q at the same level as P is not totally unexpected;
  - *open* n is a capability that is given out by n, so $n[Q]$ cannot be dissolved if it does not wish to be.

Design Principle

- An ambient should not get killed or trapped unless:
  - It talks too much. (By making its capabilities public.)
  - It poisons itself. (By opening an untrusted intruder.)
  - It steps into quicksand. (By entering an untrusted ambient.)
- Some natural primitives violate this principle. E.g.:

$$n[\underline{burst\ n.\ P} \mid Q] \quad \longrightarrow \quad P \mid Q$$

Then a mere *in* capability gives a kidnapping ability:

$$entrap(M) \quad \triangleq \quad (\nu\ k\ m)\ (m[M.\ burst\ m.\ in\ k] \mid k[])$$

$$\begin{aligned} entrap(in\ n) \mid n[P] \ &\longrightarrow^* \quad (\nu k)\ (n[in\ k \mid P] \mid k[]) \\ &\longrightarrow^* \quad (\nu k)\ k[n[P]] \end{aligned}$$

- Moral
  - One can imagine lots of different mobility primitives.
  - But one must think hard about the "security" implications of combinations of these primitives.

Ambient I/O

- Local anonymous communication within an ambient:

$(x).\ P$	input action
$\langle M \rangle$	async output action

- We have the reduction:

$$(x).\ P \mid \langle M \rangle \longrightarrow P\{x{\leftarrow}M\}$$

- This mechanism fits well with the ambient intuitions.
  - Long-range communication, like long-range movement, should not happen automatically because messages may have to cross firewalls and other obstacles. (C.f., Telescript.)
  - Still, this is sufficient to emulate communication over named channels, etc.

Reduction Summary

$n[in\ m.\ P \mid Q] \mid m[R] \longrightarrow m[n[P \mid Q] \mid R]$	(Red In)
$m[n[out\ m.\ P \mid Q] \mid R] \longrightarrow n[P \mid Q] \mid m[R]$	(Red Out)
$open\ n.\ P \mid n[Q] \longrightarrow P \mid Q$	(Red Open)
$(n).\ P \mid \langle M \rangle \longrightarrow P\{n{\leftarrow}M\}$	(Red Comm)
$P \longrightarrow Q \ \Rightarrow\ (\nu n)P \longrightarrow (\nu n)Q$	(Red Res)
$P \longrightarrow Q \ \Rightarrow\ n[P] \longrightarrow n[Q]$	(Red Amb)
$P \longrightarrow Q \ \Rightarrow\ P \mid R \longrightarrow Q \mid R$	(Red Par)
$P' \equiv P,\ P \longrightarrow Q,\ Q \equiv Q' \ \Rightarrow\ P' \longrightarrow Q'$	(Red $\equiv$)
$\longrightarrow^*$	reflexive and transitive closure of $\longrightarrow$

In addition, we identify terms up to renaming of bound names:

$$(\nu n)P \quad = \quad (\nu m)P\{n{\leftarrow}m\} \qquad \text{if } m \notin fn(P)$$

$$(n).P \quad = \quad (m).P\{n{\leftarrow}m\} \qquad \text{if } m \notin fn(P)$$

Structural Congruence Summary

$P \equiv P$	(Struct Refl)
$P \equiv Q \quad \Rightarrow \quad Q \equiv P$	(Struct Symm)
$P \equiv Q, Q \equiv R \quad \Rightarrow \quad P \equiv R$	(Struct Trans)
$P \equiv Q \quad \Rightarrow \quad (\nu n)P \equiv (\nu n)Q$	(Struct Res)
$P \equiv Q \quad \Rightarrow \quad P \mid R \equiv Q \mid R$	(Struct Par)
$P \equiv Q \quad \Rightarrow \quad !P \equiv !Q$	(Struct Repl)
$P \equiv Q \quad \Rightarrow \quad M[P] \equiv M[Q]$	(Struct Amb)
$P \equiv Q \quad \Rightarrow \quad M.P \equiv M.Q$	(Struct Action)
$P \equiv Q \quad \Rightarrow \quad (n).P \equiv (n).Q$	(Struct Input)

$P \mid Q \equiv Q \mid P$	(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)
$(\nu n)(P \mid Q) \equiv P \mid (\nu n)Q \quad \text{if } n \notin fn(P)$	(Struct Res Par)
$(\nu n)(m[P]) \equiv m[(\nu n)P] \quad \text{if } n \neq m$	(Struct Res Amb)
$P \mid \mathbf{0} \equiv P$	(Struct Zero Par)
$(\nu n)\mathbf{0} \equiv \mathbf{0}$	(Struct Zero Res)
$!\mathbf{0} \equiv \mathbf{0}$	(Struct Zero Repl)
$\varepsilon.P \equiv P$	(Struct ε)
$(M.M').P \equiv M.(M'.P)$	(Struct .)

Example

$$\overbrace{A[msg[\langle M\rangle | out\ A.\ in\ B]]}^{Principal\ A}_{\underbrace{\qquad\qquad}_{send\ M\ :\ A{=>}B}} \mid \overbrace{B[open\ msg.\ (x).\ P]}^{Principal\ B}_{\underbrace{\qquad\qquad}_{receive\ x;\ P}}$$

$$\begin{aligned}
& A[msg[\langle M\rangle | out\ A.\ in\ B]] \mid B[open\ msg.\ (x).\ P] \\
\longrightarrow\ & A[] \mid msg[\langle M\rangle | in\ B] \mid B[open\ msg.\ (x).\ P] \\
\longrightarrow\ & A[] \mid B[msg[\langle M\rangle] | open\ msg.\ (x).\ P] \\
\longrightarrow\ & A[] \mid B[\langle M\rangle | (x).\ P] \\
\longrightarrow\ & A[] \mid B[P\{x \leftarrow M\}]
\end{aligned}$$

Noticeable Inequivalences

- Replication creates new names:

$$!(\nu n)P \quad \not\equiv \quad (\nu n)!P$$

- Multiple *n* ambients have separate identity:

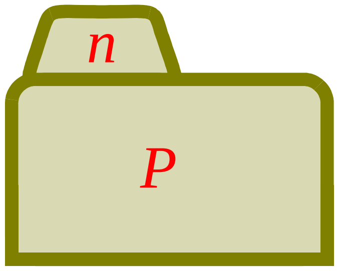
$$n[P] \mid n[Q] \quad \not\equiv \quad n[P \mid Q]$$

Folder Calculus

The Folder Calculus

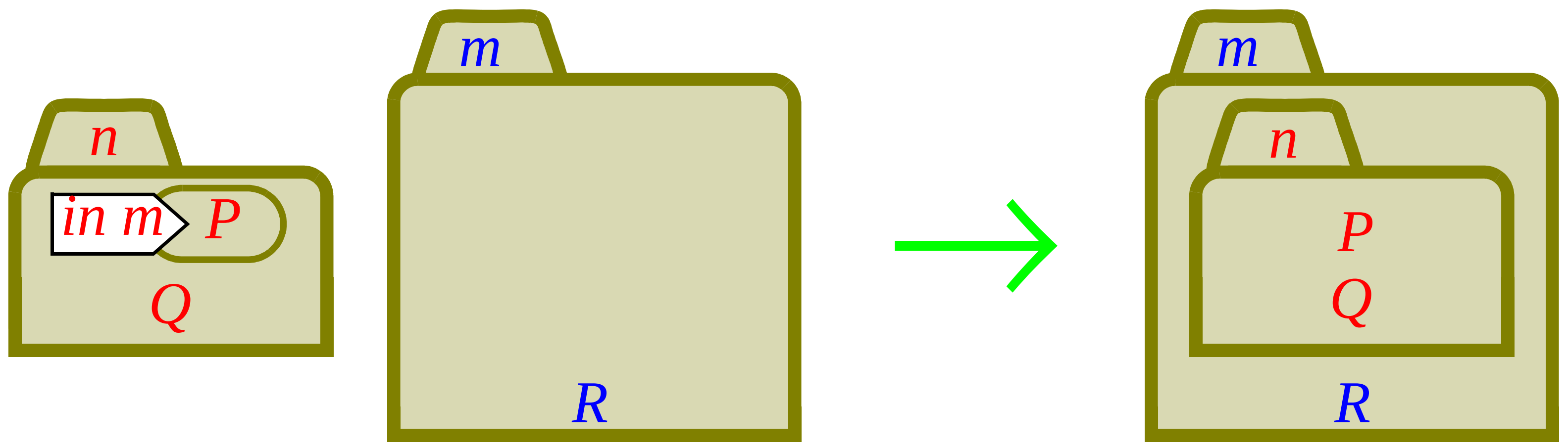
- A graphical office metaphor to explain the ambient calculus.
- A precise metaphor, isomorphic to the formal ambient calculus.
- Based on wide-area computation principles: locality, mobility, nested domains, asynchronous communication, authentication.

Folders (Nested Domains)

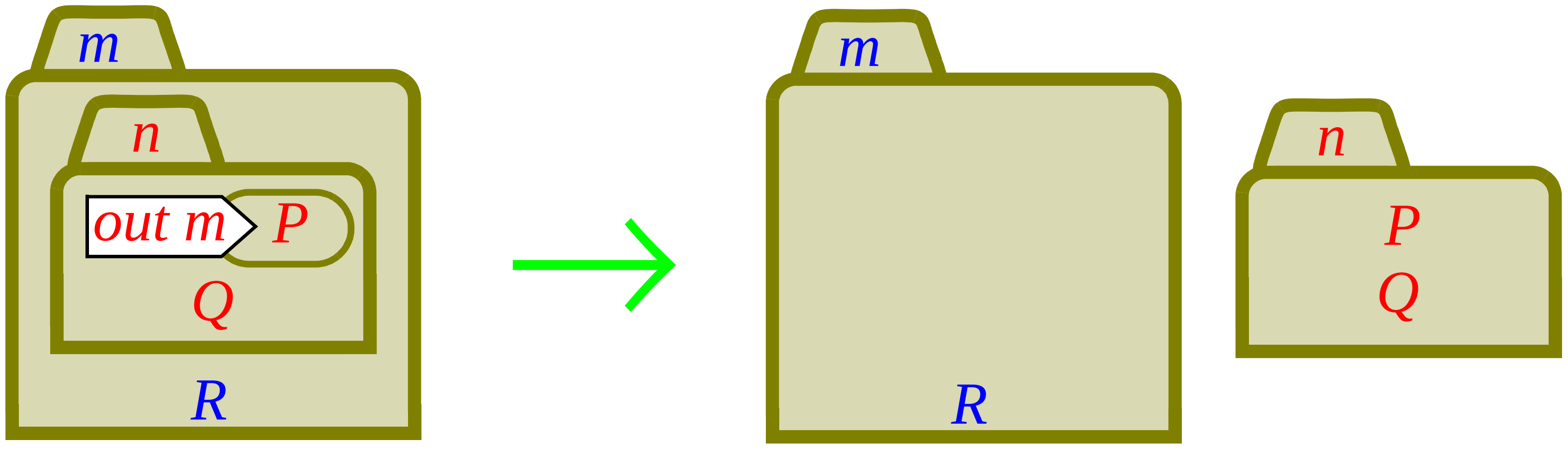


- A folder name *n*.
- Active contents *P*:
  - hierarchical data and “gremlins”.
  - computational primitives for mobility and communication.

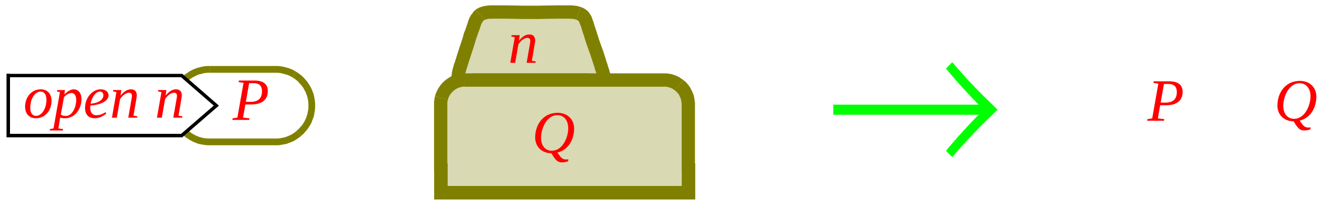
Enter Reduction (Mobility)



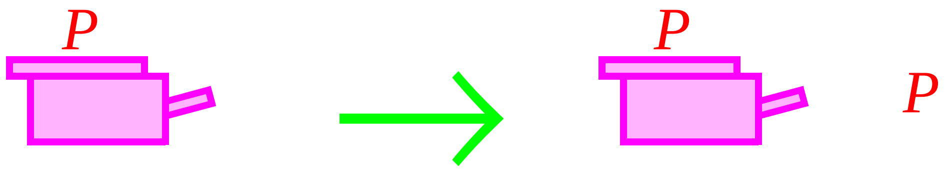
Exit Reduction (Mobility)



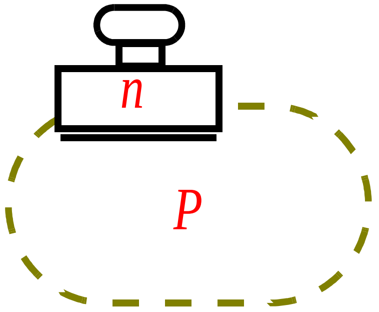
Open Reduction (Assimilation)



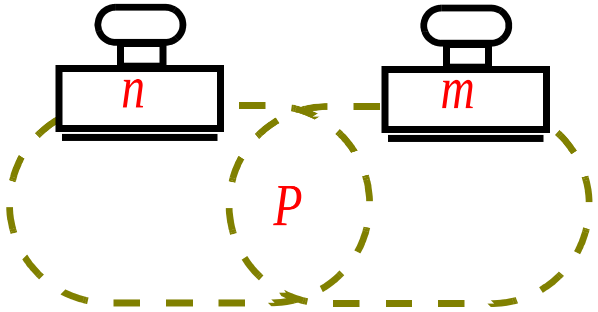
Copy Reduction (Iteration/Recursion)



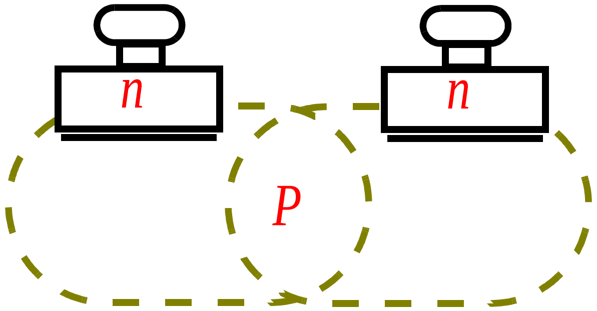
Rubber Stamps (Authentication)



- Give authenticity to folders.
- Copiers are unable to accurately duplicate rubber stamps.



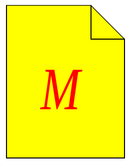
allowed



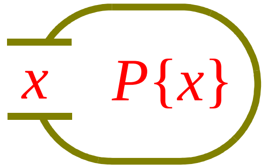
forbidden

Post-It Notes (Local Communication)

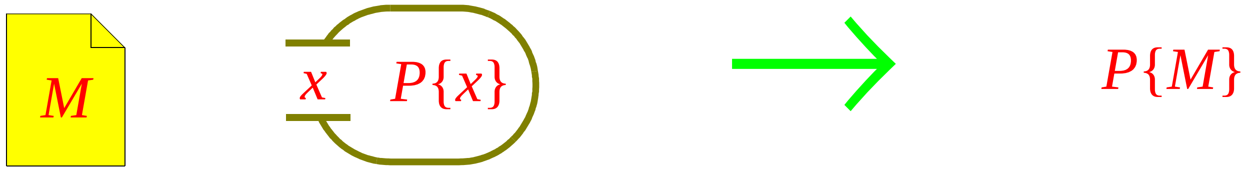
- A Post-It Note (Nameless file / Asynchronous message).



- A gremlin grabbing (reading and removing) a note.



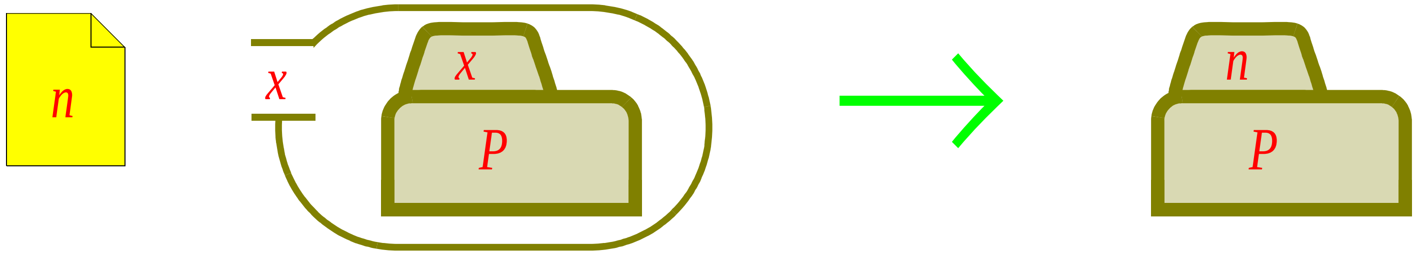
- Read reduction

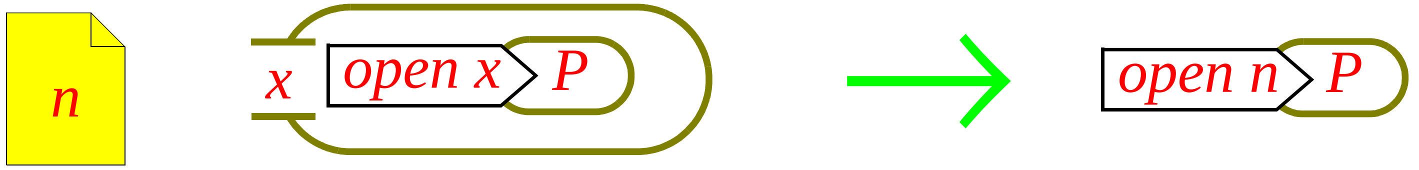


Messages (Names or Capabilities)

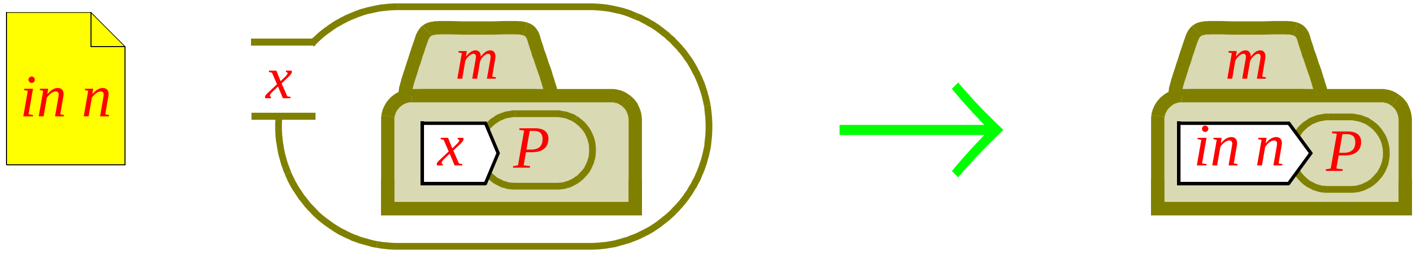
A message *M* can be either:

- The name of a folder (danger: spoofing, killing):

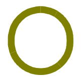




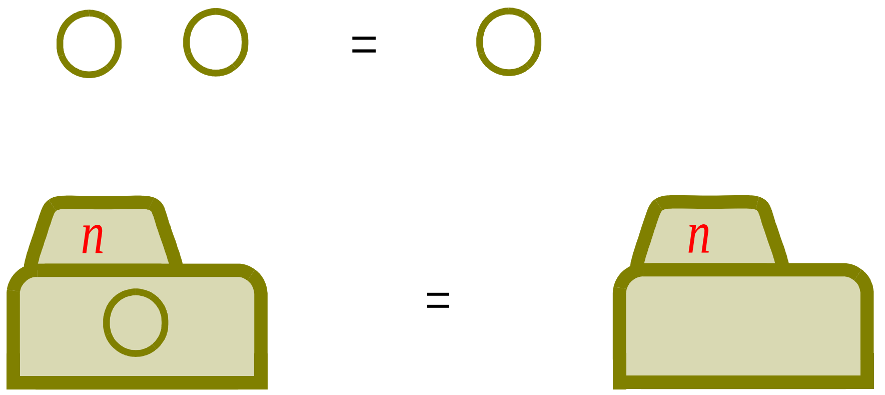
- A capability (no danger of recovering the name):



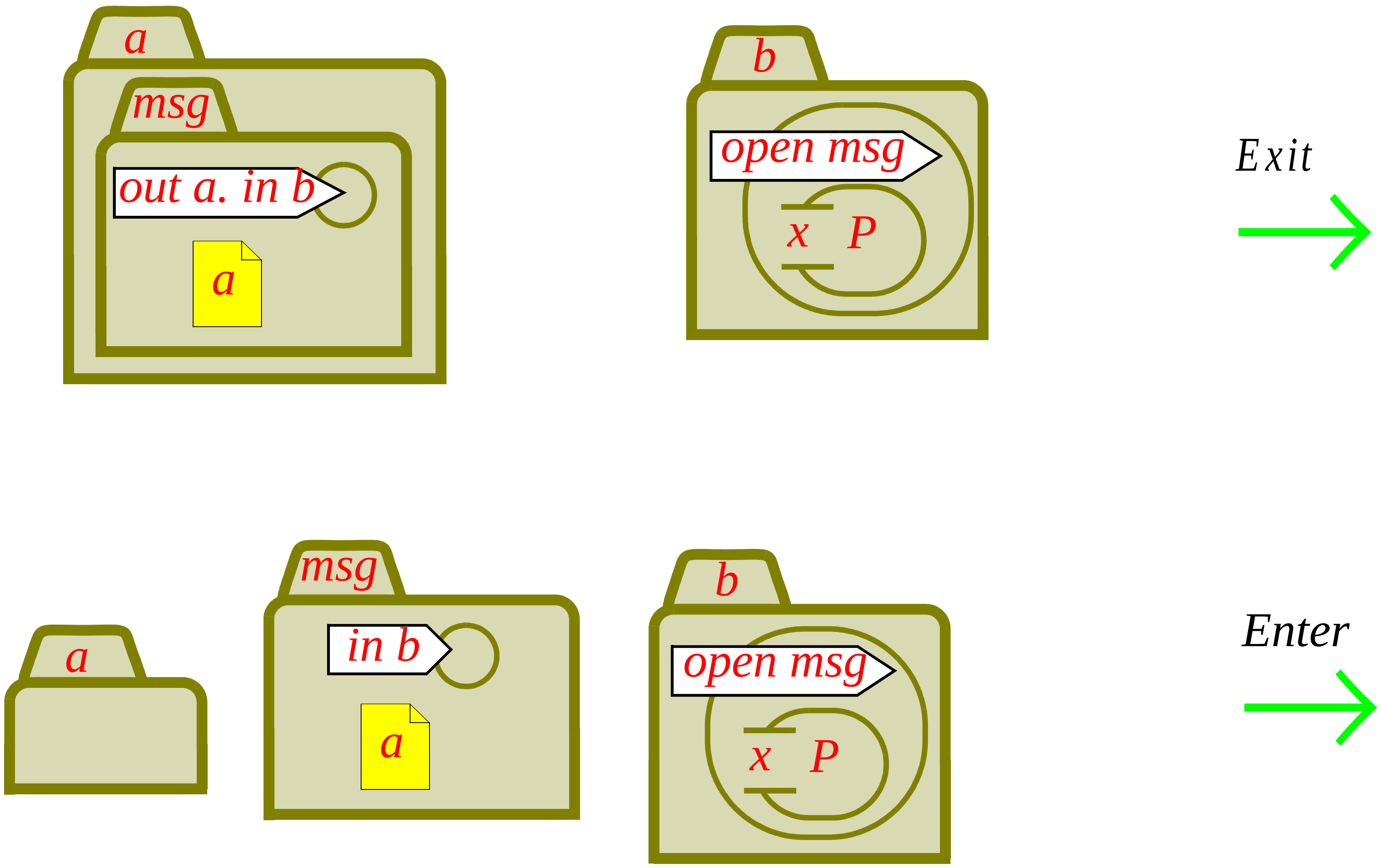
Leaves of the Syntax

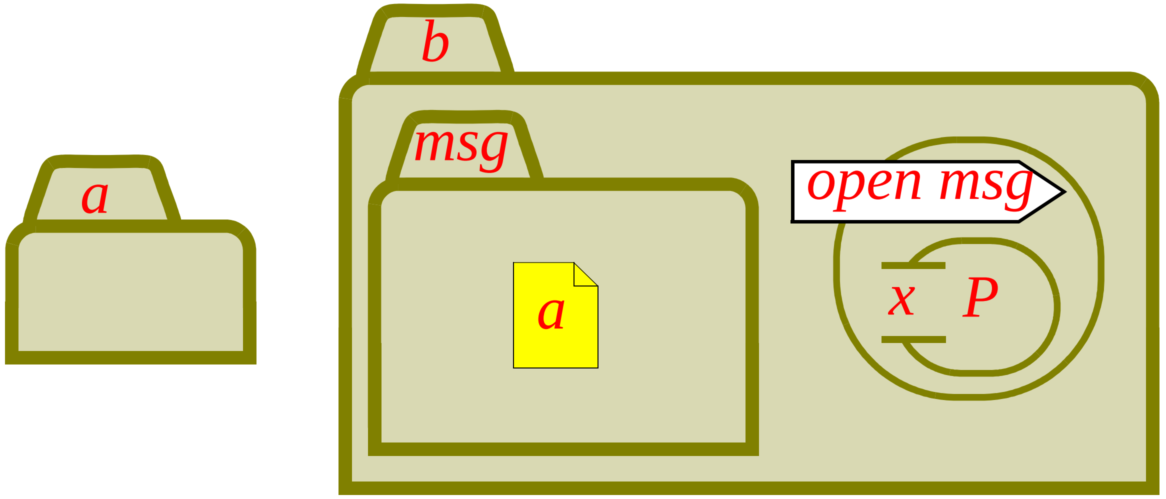


Inactive gremlin



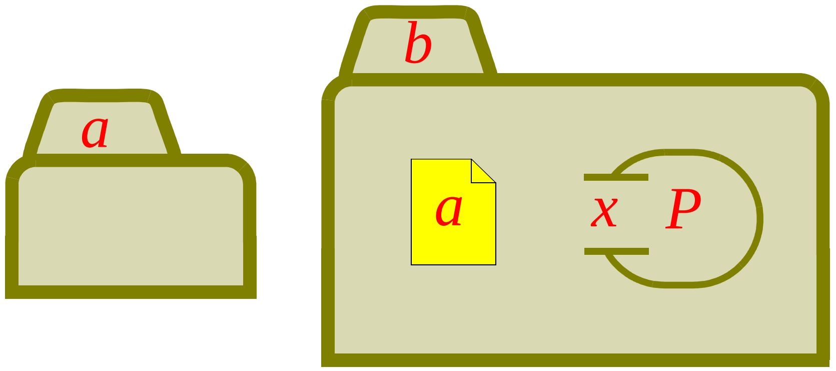
Example: Message from A to B





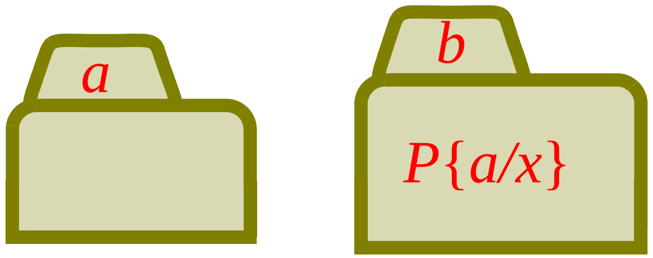
Open

$\rightarrow$

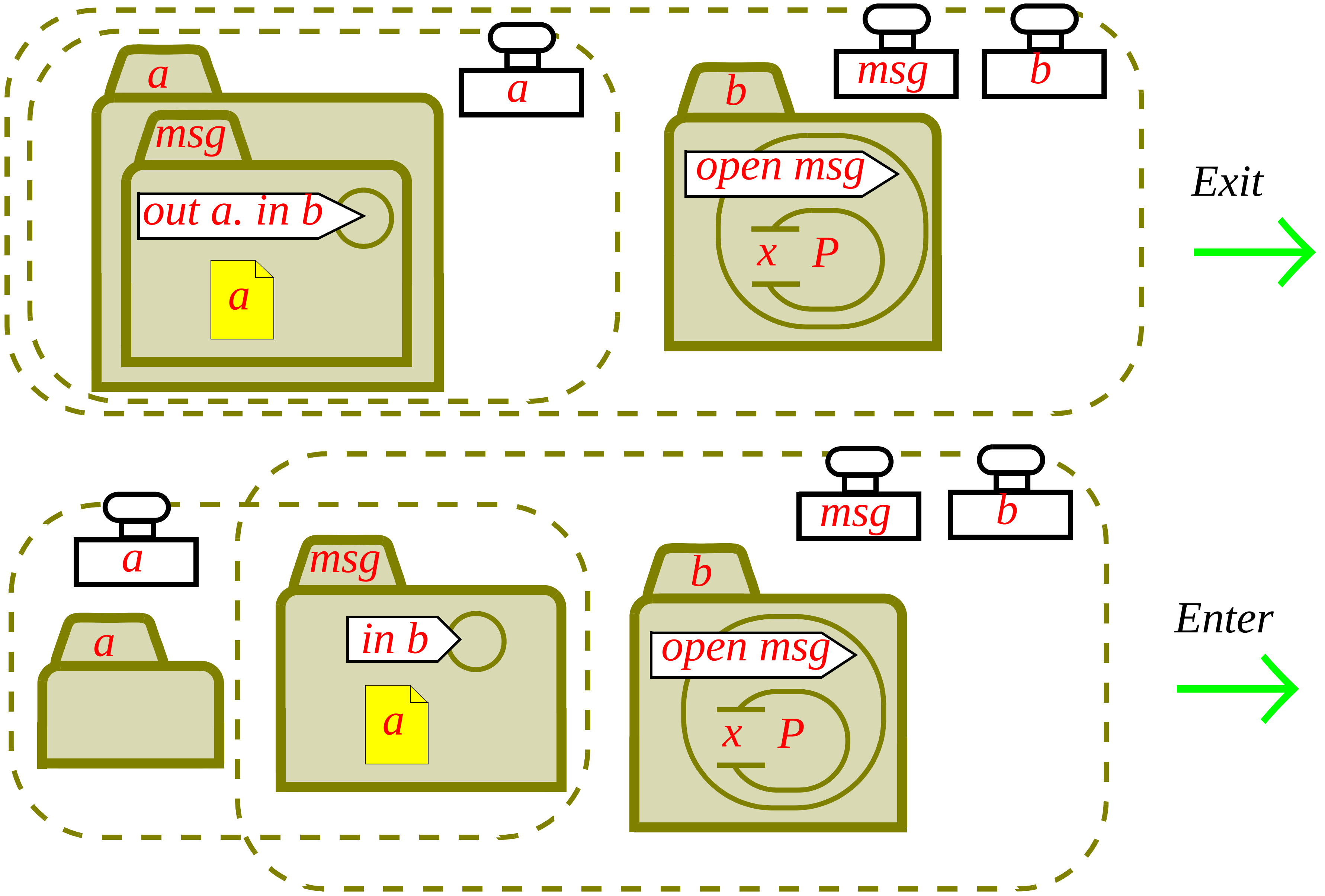


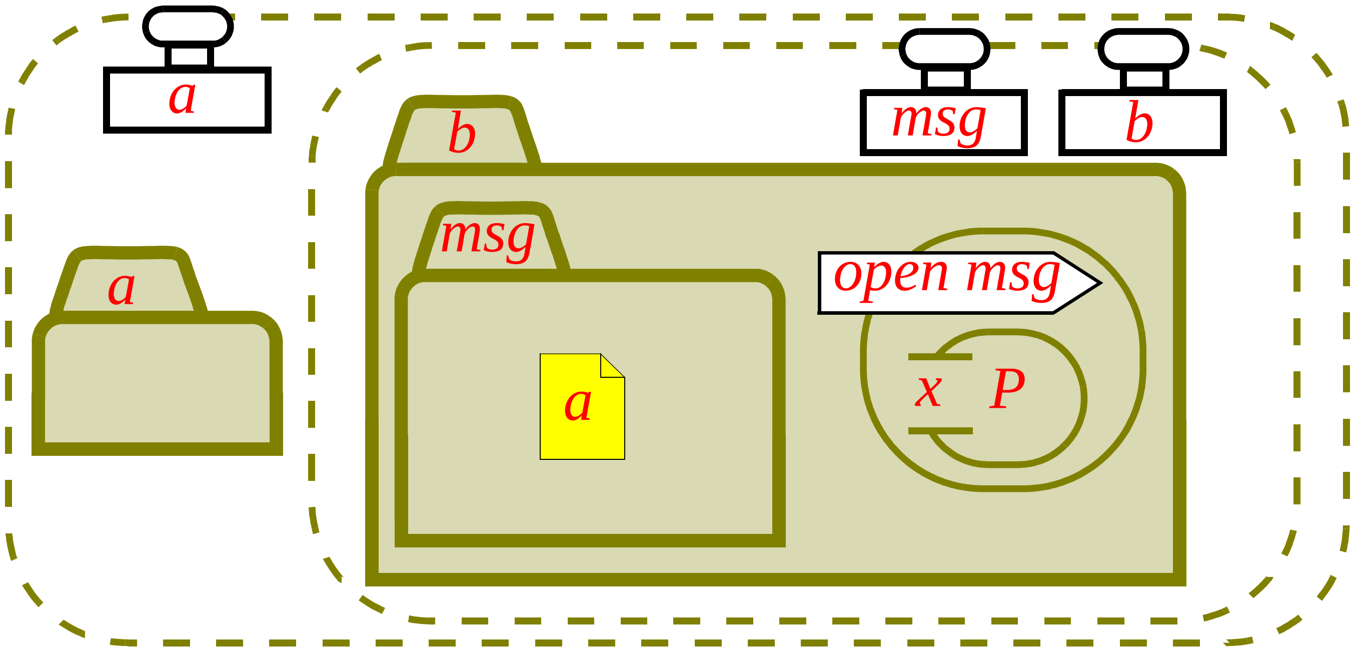
Read

$\rightarrow$


a
b
$P\{a/x\}$

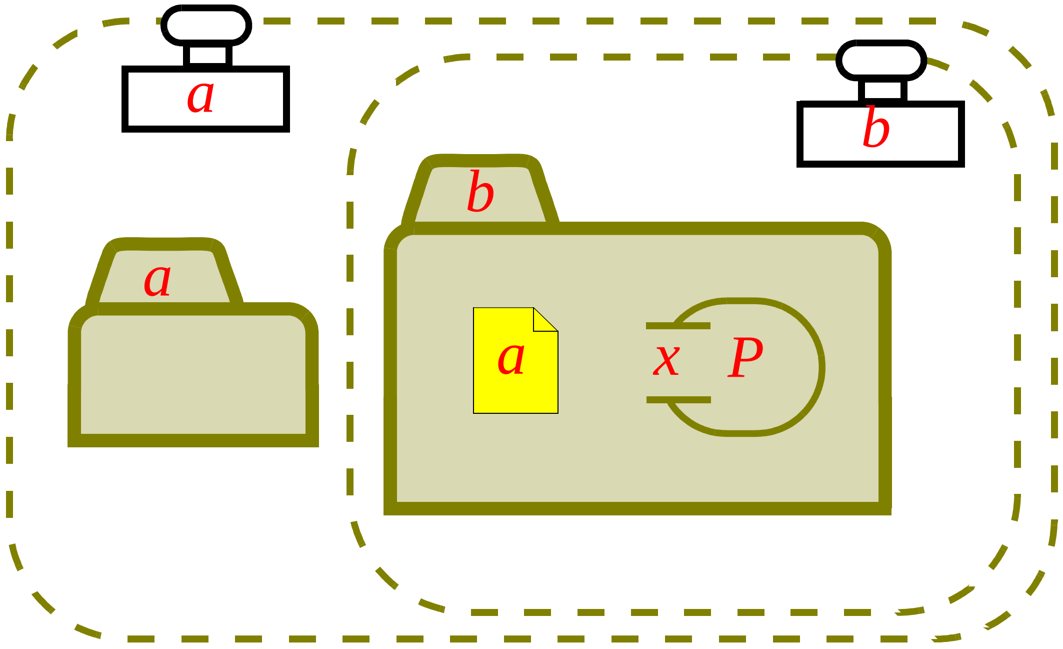
Same Example, with scoping





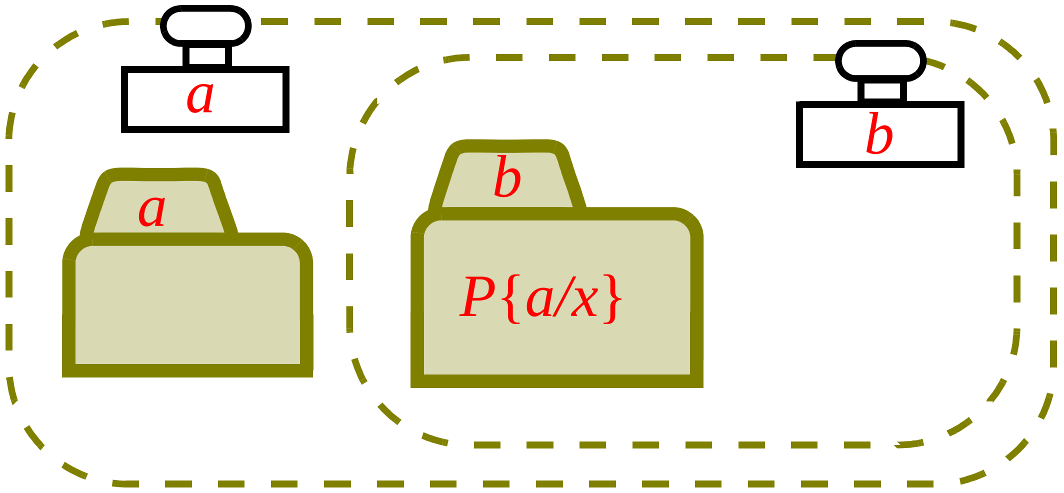
Open

$\longrightarrow$

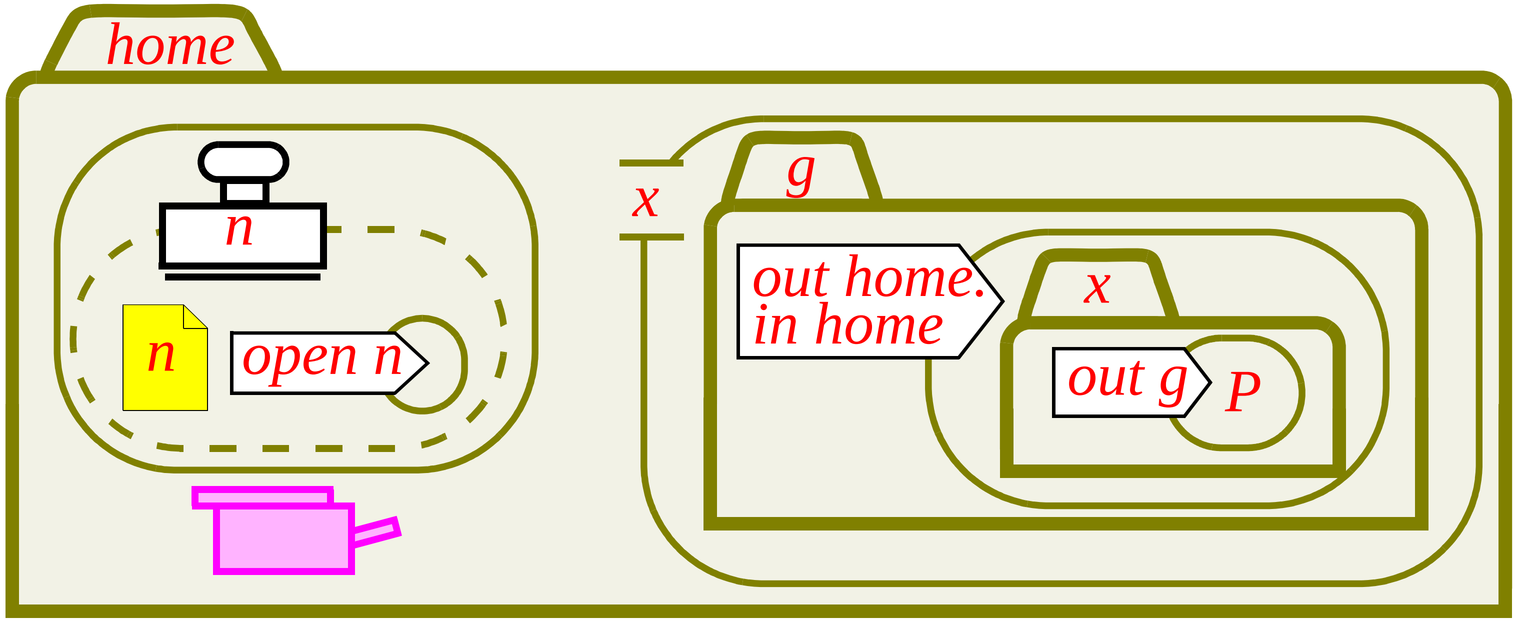


Read

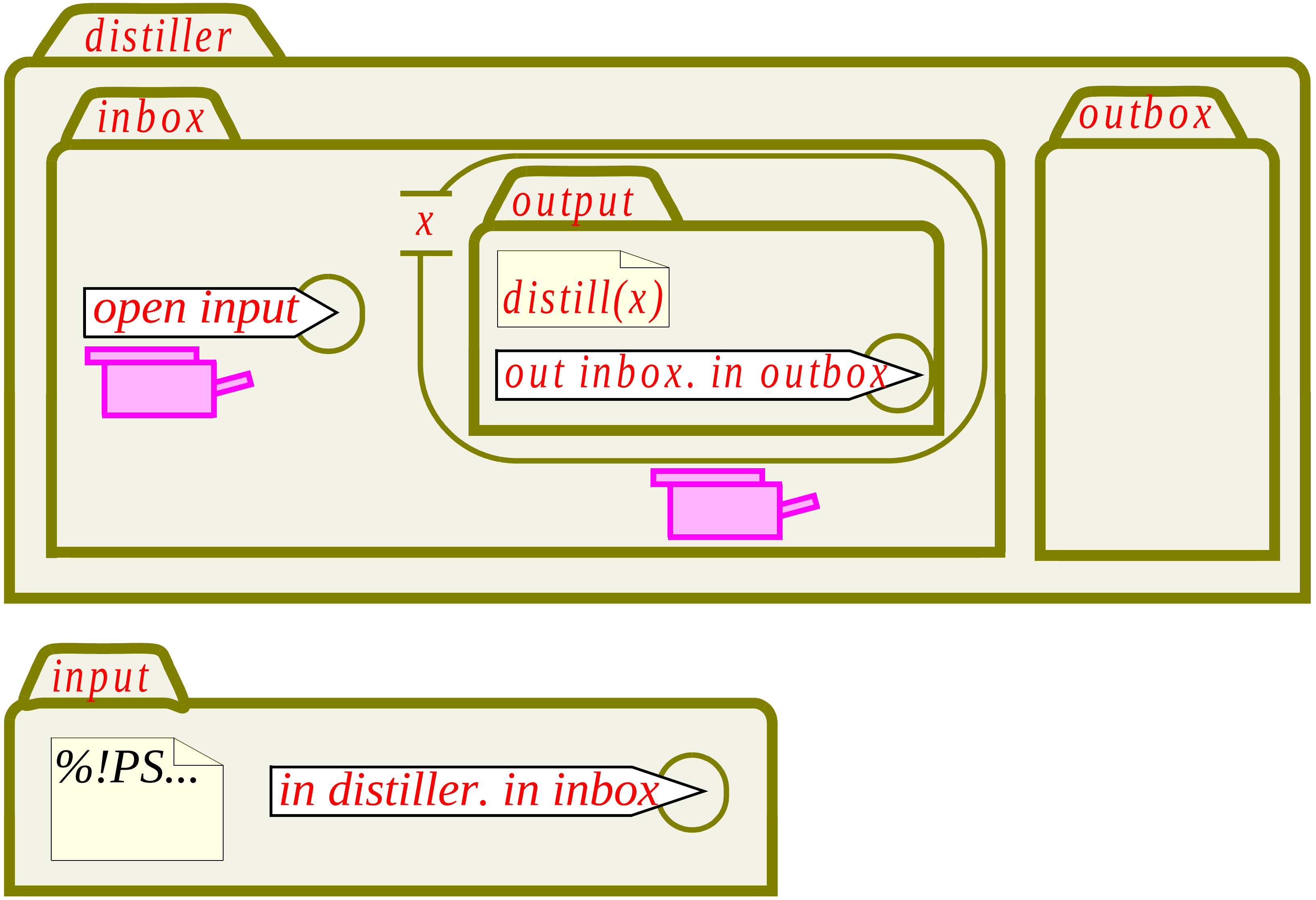
$\longrightarrow$


a
a
b
b
P{a/x}

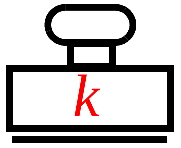
Example: Agent Authentication



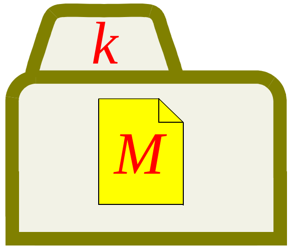
Example: A Distiller Server



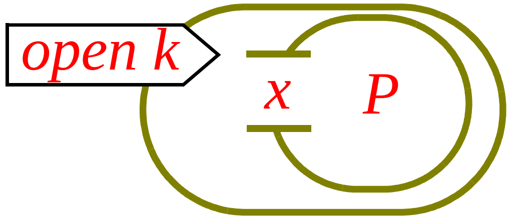
Example: Keys



generation of a fresh key k



encryption:
plaintext M inside a k-envelope



decryption:
opening a k-envelope and reading the contents

AMBIENT CALCULUS EXPRESSIVENESS

Expressiveness

- Some new features
  - The primitives invented for process mobility end up being meaningful for security.
  - The combination of mobility and security in the same formal framework seems novel and intriguing.
  - E.g., we can represent both mobility and (some) security aspects of “crossing a firewall”.

Old Concepts

- Old concepts that can be represented:
  - Synchronization and communication mechanisms.
  - Turing machines. (Natural encoding, no I/O required.)
  - Arithmetic. (Tricky, no I/O required.)
  - Data structures.
  - π-calculus. (Easy, channels are ambients.)
  - λ-calculus. (Hard, different than encoding λ in π.)
  - Spi-calculus concepts. (Being debated.)

New Concepts

- Net-centric concepts that can be represented:
  - Named machines and services on complex networks.
  - Agents, applets, RPC.
  - Encrypted data and firewalls.
  - Data packets, routing, active networks.
  - Dynamically linked libraries, plug-ins.
  - Mobile devices.
  - Public transportation.

Ambients as Locks

- We can use *open* to encode locks:

$$\mathit{release}\ n.\ P \triangleq n[\,] \mid P$$

$$\mathit{acquire}\ n.\ P \triangleq \mathit{open}\ n.\ P$$

- This way, two processes can "shake hands" before proceeding with their execution:

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

Turing Machines

```
end[extendLft | S_0 |
        square[S_1 |
                square[S_2 |
                        ...
                                square[S_i | head |
                                        ...
                                                square[S_{n-1} |
                                                        square[S_n | extendRht]] .. ] .. ]]]
```

Ambients as Mobile Processes

$tourist \quad \triangleq \quad (x).\ joe[x.\ enjoy]$

$ticket\text{-}desk \quad \triangleq \quad !\ \langle in\ AF81SFO.\ out\ AF81CDG \rangle$

$SFO[ticket\text{-}desk \mid tourist \mid AF81SFO[route]]$

$\longrightarrow^* \qquad SFO[ticket\text{-}desk \mid$
$\qquad\qquad joe[in\ AF81SFO.\ out\ AF81CDG.\ enjoy] \mid$
$\qquad\qquad AF81SFO[route]]$

$\longrightarrow^* \qquad SFO[ticket\text{-}desk \mid$
$\qquad\qquad AF81SFO[route \mid joe[out\ AF81CDG.\ enjoy]]]$

Ambients as Firewalls (buggy)

- Assume that the shared key k is already known to the firewall and the client.

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

$Cleo \mid Wally$

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

Ambients as Firewalls

- Assume that the shared key k is already known to the firewall and the client.

$$\begin{aligned} \mathit{Wally} &\triangleq (\nu w)\,(k[\mathit{in}\ k.\ \mathit{in}\ w] \mid w[\mathit{open}\ k.\ P]) \\ \mathit{Cleo} &\triangleq k[\mathit{open}\ k.\ C] \end{aligned}$$

$$\begin{aligned} &\mathit{Cleo} \mid \mathit{Wally} \\ &\quad \longrightarrow^* (\nu w)\,(\ k[\mathit{open}\ k.\ C] \mid k[\mathit{in}\ k.\ \mathit{in}\ w] \mid w[\mathit{open}\ k.\ P]\ ) \\ &\quad \longrightarrow^* (\nu w)\,(\ k[k[\mathit{in}\ w] \mid \mathit{open}\ k.\ C] \mid w[\mathit{open}\ k.\ P]\ ) \\ &\quad \longrightarrow^* (\nu w)\,(\ k[\mathit{in}\ w \mid C] \mid w[\mathit{open}\ k.\ P]\ ) \\ &\quad \longrightarrow^* (\nu w)\quad w[k[C] \mid \mathit{open}\ k.\ P] \\ &\quad \longrightarrow^* (\nu w)\quad w[C \mid P] \end{aligned}$$

Comments

- One secret names is introduced: w is the secret name of the firewall.
- We want to verify that Cleo knows the key k: this is done by $in\ k$. After that, Cleo gives control to $in\ w$ to enter the firewall.

The Asynchronous π-calculus

- A named channel is represented by an ambient.
  - The name of the channel is the name of the ambient.
  - Communication on a channel is becomes local I/O inside a channel-ambient.
  - A conventional name, *io*, is used to transport I/O requests into the channel.

$$\begin{array}{lcl}
(ch\ n)P & \triangleq & (\nu n)\ (n[!open\ io]\ |\ P) \\
n(x).P & \triangleq & (\nu p)\ (io[in\ n.\ (x).\ p[out\ n.\ P]]\ |\ open\ p) \\
n\langle M\rangle & \triangleq & io[in\ n.\ \langle M\rangle]
\end{array}$$

- These definitions satisfy the expected reduction in presence of a channel for *n:*

$$n(x).P\ |\ n\langle M\rangle \quad \longrightarrow^* \quad P\{x\leftarrow M\}$$

- Therefore:

$$
\begin{array}{lcl}
\langle\!\langle (\nu n)P \rangle\!\rangle & \triangleq & (\nu n)\ (n[!open\ io] | \langle\!\langle P \rangle\!\rangle) \\
\langle\!\langle n(x).P \rangle\!\rangle & \triangleq & (\nu p)\ (io[in\ n.\ (x).\ p[out\ n.\ \langle\!\langle P \rangle\!\rangle]] | open\ p) \\
\langle\!\langle n\langle m \rangle \rangle\!\rangle & \triangleq & io[in\ n.\ \langle m \rangle] \\
\langle\!\langle P|Q \rangle\!\rangle & \triangleq & \langle\!\langle P \rangle\!\rangle | \langle\!\langle Q \rangle\!\rangle \\
\langle\!\langle !\mathrm{P} \rangle\!\rangle & \triangleq & !\langle\!\langle \mathrm{P} \rangle\!\rangle
\end{array}
$$

  – The choice-free synchronous π-calculus, can be encoded within the asynchronous π-calculus.

  – The λ-calculus can be encoded within the asynchronous π-calculus.

WAN Languages

From Calculi to Languages

- The ambient/folder calculus is a minimal formalism designed for theoretical study. As such, it is not a “programming language”.
- Still, the ambient calculus is designed to match fundamental WAN characteristics.
- By building languages on top of a well-understood WAN semantics, we can be confident that languages will embody the intended semantics.
- We now discuss how ambient characteristics might look like when extrapolated to programming languages.

WAN Phenomena

- Physical Locations
  - Observable because of the speed of light limit
  - Preclude instantaneous actions
  - Require mobile code
- Virtual Locations
  - Observable because of administrative domains
  - Preclude unfettered actions
  - Require security model and disconnected operation

- Variable Connectivity
  - Observable because of free-will actions, physical mobility
  - Precludes purely static networks
  - Requires bandwidth adaptability
- Failures
  - Unobservable because of asynchrony, domain walls
  - Preclude reliance on other parts of the system
  - Require blocking behavior, transaction model

Mobility and Barriers

- Mobility is all about barriers:
  - Locality = **barrier topology**.
  - Process mobility = **barrier crossing**.
  - Security = **(in)ability to cross barriers**.
  - Communication = **interaction within a barrier.**
  - No immediate *action at a distance* (= **across barriers**).
- Ambients embed this barrier-based view of mobility, which is grounded on WAN observables.
- A "wide-area language" is one that does not contain features violating this view of computation.

Wide Area Languages

- WAN-sound
  - No action-at-a-distance assumption
  - No continued connectivity assumption
  - No security bypasses
- WAN-complete
  - Able to emulate and support surfer/roamer/road-warrior behavior
- Some steps towards Wide Area Languages:
  - Agent languages (although many are effectively LAN-bound)
  - Ambit (with Mads Torgesen)
  - Service Combinators (with Rowan Davies)
    WebL (Hannes Marais)

Ambients as a Programming Abstraction

- Our basic abstraction is that of mobile computational ambients.
- The ambient calculus brings this abstraction to an extreme, by representing *everything* in terms of ambients at a very fine grain.
- In practice, ambients would have to be medium or large-grained entities. Ambient contents should include standard programming subsystems such as modules, classes, objects, and threads.
- But: the ability to smoothly move a collection of running threads is almost unheard of in current software infrastructures. Ambients would be a novel and non-trivial addition to our collection of programming abstractions.

Names vs. Pointers

- The only way to denote an ambient is by its name.
  - One may possess a name without having immediate access to any ambient of that name (unlike pointers).
  - Name references are never “broken” but may be “blocked” until a suitable ambient becomes available.
- Uniformly replace pointers (to data structures etc.) by names.
  - At least across ambient boundaries.
  - This is necessary to allow ambients to move around freely without being restrained by immobile ties.

Locations

- Ambients can be used to model both physical and virtual locations.
  - Some physical locations are mobile (such as airplanes) while others are immobile (such as buildings).
  - Similarly, some virtual locations are mobile (such as agents) while others are immobile (such as mainframe computers).
- Mobility distinctions are not part of the basic semantics of ambients.
  - Can be added as a refinement of the basic model, or
  - Can be embedded in type systems that restrict the mobility of certain ambients.

Migration and Transportation

- Ambients offer a good paradigm for application migration.
  - If an ambient encloses a whole application, then the whole running application can be moved without need to restart it or reinitialize it.
  - In practice, an application will have ties to the local window system, the local file system, etc. These ties, however, should only be via ambient names.
  - After movement the application can smoothly move and reconnect its bindings to the new local environment. (Some care will still be needed to restart in a good state).

Communication

- The communication primitives of the ambient calculus (local to an ambient) do not support global consensus or failure detection.
- These properties should be preserved by any higher-level communication primitives that may be added to the basic model, so that the intended semantics of communication over a wide-area networks is preserved.
  - RPC, interpreted as mobile packets that transport and deposit messages to remote locations.
  - Parent-child communication
  - Communication between siblings.

Synchronization

- The ambient calculus is highly concurrent.
  - It has high-level synchronization primitives that are natural and effective (as shown in the examples).
  - It is easy to represent basic synchronization constructs, such as mutexes:

$release\ n; P$	$\triangleq$	$n[\,] \mid P$	release a mutex called n, and do P
$acquire\ n; P$	$\triangleq$	$open\ n.\ P$	acquire a mutex called n, then do P

- Still, additional synchronization primitives are desirable.
  - A useful technique is to synchronize on the change of name of an ambient:

$$n[be\ m.P \mid Q] \quad \rightarrow \quad m[P \mid Q]$$

  - (See also the Seal calculus by Castagna and Vitek.)

Static and Dynamic Binding

- The names of the ambient calculus represent an unusual combination of static and dynamic binding.
  - The names obey the classical rules of static scoping, including consistent renaming, capture-avoidance, and block nesting.
  - The navigation primitives behave by dynamically binding/linking a name to any ambient that has the right name.
- Definitional facilities can similarly be derived in static or dynamic binding style. E.g.:
  - Statically bound function definitions.
  - Dynamically bound resource definitions.

Modules

- An ambient containing *definitions* is similar to a module/class.
  - Remote invocation is like qualified module access.
  - *open* is like inheritance.
  - *copy* is like object generation from a prototype.
- Unusual “module” features:
  - Ambients are *first class modules*: one can choose at run time which particular instance of a module to use.
  - Ambients support *dynamic linking*: missing subsystems can be added to a running system by placing them in the right spot.
  - Ambients support *dynamic reconfiguration*. Module identity is maintained at run time. The blocking semantics allows smooth suspension and reactivation. The dynamic hierarchical structure allows replacement of subsystems.

Security

- Ambient security is based on boundaries and capabilities, as opposed to a cryptography, or access-control.
- These three models are all interdefinable. In our case:
  - Access control is obtained by using ambients to implement RPC-like invocations that have to cross boundaries and authenticate every time.
  - Cryptography is obtained by interpreting ambient names (by assumption unforgeable) as encryption keys.
- The ambient security model is high level.
  - It maps naturally to administrative domains and sandboxes.
  - It allows the direct discussion of virus, trojan horses, infection of mobile agents, firewall crossing, etc.

Summary of WAL Features

- No “hard” pointers.

  Remote references are URLs, symbolic links, or such.

- Migration/Transportation

  Thread migration.

  Data migration.

  Whole-application migration.

- Dynamic linking.

  A missing library or plug-in may suddenly show up.

- Patient communication.

  Blocking/exactly-once semantics.

- Built-in security primitives.

EQUIVALENCE

Contextual Equivalence

- A standard semantic equivalence: $P \simeq Q$ iff the observable behavior of a system with component P is unaffected by replacing P with Q.
- In the ambient calculus:
  - $P{\downarrow}n$ means that process P exhibits n (observable by *open* n).

$$P{\downarrow}n \quad \Leftrightarrow \quad P \equiv (\nu n_1 \ldots n_p)(n[Q] \mid R) \ \wedge \ n \notin \{n_1 \ldots n_p\}$$

  - $P{\Downarrow}n$ means that process P converges n, i.e., it evolves into a process that exhibits n.

$$P{\Downarrow}n \quad \Leftrightarrow \quad P \longrightarrow^* Q \ \wedge \ Q{\downarrow}n$$

  - A *context* $\boldsymbol{C(\bullet)}$ is a process with a hole. Ex.: $\bullet$, $n[\bullet]$, $in\ n.\bullet$, $\boldsymbol{!}\bullet$.
  - Contextual Equivalence:

$$P \simeq Q \quad \Leftrightarrow \quad \forall n.\ \forall \boldsymbol{C(\bullet)}.\ \boldsymbol{C(}P\boldsymbol{)}{\Downarrow}n \Leftrightarrow \boldsymbol{C(}Q\boldsymbol{)}{\Downarrow}n$$

Example: Some Basic Equations

- Some easy inequivalences:
  - $p[]$ and $q[]$ distinguished by $\bullet$.
  - $open\ p.\mathbf{0}$ and $open\ q.\mathbf{0}$ distinguished by $p[n[]] \mid \bullet$.
  - $in\ p.\mathbf{0}$ and $out\ p.\mathbf{0}$ distinguished by $m[n[\bullet \mid out\ p.\ out\ m] \mid p[]]$.
- As in most process calculi, reduction does not imply equivalence:
  - $n[] \mid open\ n.\mathbf{0} \longrightarrow \mathbf{0}$ but $n[] \mid open\ n.\mathbf{0} \not\simeq \mathbf{0}$.
- A general law:

$$(\nu n)(n[] \mid open\ n.P) \simeq (\nu n)P$$

Example: Perfect Firewalls

- An ambient $n[P]$ can abstractly model an internet firewall named n enclosing a group of machines P (or a sandbox named n enclosing a group of applets P).

  The *Perfect Firewall Equation*:

  > If nobody inside or outside an ambient knows its name, then it forms a perfect boundary between its inside and outside:
  >
  > $$n \notin fn(P) \quad \Rightarrow \quad (\nu n)n[P] \simeq \mathbf{0}$$
  >
  > (Alternatively: $(\nu n)n[(\nu n)P] \simeq \mathbf{0}$.)

Example: Perfect Encryption

- An ambient $k[\langle M\rangle]$ abstractly models a ciphertext $\{M\}_k$ obtained by encrypting a plaintext M with a key k.

  The *Perfect Encryption Equation*:

  $$(\nu k)k[\langle M\rangle] \simeq (\nu k)k[\langle M'\rangle]$$

  (Follows from the Perfect Firewall Equation if $k \notin fn(M)$.)

  Compare with $(\nu k)c\langle\{M\}_k\rangle \simeq (\nu k)c\langle\{M'\}_k\rangle$ from spi.

Example: Piloting an Agent Across a Firewall

- Pre-arranged passwords k, k', k'' allow an agent to cross a firewall:

  $$Firewall \triangleq (\nu w)w[k[out\ w.\ in\ k'.\ in\ w] \mid open\ k'.\ open\ k''.P]$$
  $$Agent \triangleq k'[open\ k.k''[C]]$$

  Assuming k, k', k'' do not occur in C or P and w does not occur in C, we get the safety property:

  $$(\nu k\ k'\ k'')\ (Agent \mid Firewall) \simeq (\nu w)w[C \mid P]$$

- Contextual equivalence here means that:
  - The agent may successfully cross the firewall. In addition:
  - The agent will successfully cross the firewall under any attack (since "in every context" includes both "in parallel with any attacking process" and "inside any attacking environment").
  - Caveat: only ambient-level attacks. E.g. applet-to-applet.

Problem 1: Chemical Soups

- Structural congruence [Berry & Boudol, Milner] allows for:
  - rearrangement: $P \mid (\nu n)Q \equiv (\nu n)(P \mid Q)$ if $n \notin fn(P)$, ...
  - garbage collection: $P \mid \mathbf{0} \equiv \mathbf{0}$, ...
  - replication: $\mathbf{!}P \equiv P \mid \mathbf{!}P$, $\mathbf{!0} \equiv \mathbf{0}$, ...
  - reduction factored by congruence, based on the rule:

$$P \equiv P' \;\wedge\; P' \longrightarrow Q' \;\wedge\; Q' \equiv Q \;\;\Rightarrow\;\; P \longrightarrow Q$$

- This is great for many purposes!
  - Calculating reductions, validating type systems, obeying page limits...
- But it’s dreadful for others.
  - Try proving that if $\mathbf{C(0)}{\Downarrow}n$ then $\mathbf{C(}P\mathbf{)}{\Downarrow}n$.

Solution: Hardenings and Labelled Transitions

- A *concretion* is a phrase $(\nu\overline{p})\langle P'\rangle P''$ [Milner].
- New idea: a *hardening* $P > (\nu\overline{p})\langle P'\rangle P''$ identifies a top-level process P' of P, the residue P'', and the bindings $\overline{p}$ they share.
- For example:

$$P = (\nu p)(\nu q)(n[p[]] \mid q[])$$

has two hardenings:

$$P > (\nu)\ \langle(\nu p)n[p[]]\rangle\ (\nu q)(\mathbf{0} \mid q[])$$
$$P > (\nu q)\ \langle q[]\rangle\ (\nu p)(n[p[]] \mid \mathbf{0})$$

Definition of Hardening

Hardening $P > (\nu\overline{p})\langle P'\rangle P''$

$$\frac{M \in \{in\ n,\ out\ n,\ open\ n\}}{M.P > (\nu)\langle M.P\rangle \mathbf{0}} \qquad \frac{}{n[P] > (\nu)\langle n[P]\rangle \mathbf{0}}$$

$$\frac{P > (\nu\overline{p})\langle P'\rangle P'' \quad \{\overline{p}\} \cap fn(Q) = \emptyset}{P \mid Q > (\nu\overline{p})\langle P'\rangle (P'' \mid Q)} \qquad \frac{P > (\nu\overline{p})\langle P'\rangle P''}{(\nu n)P > (\nu n)(\nu\overline{p})\langle P'\rangle P''}$$

$$\frac{Q > (\nu\overline{q})\langle Q'\rangle Q'' \quad \{\overline{q}\} \cap fn(P) = \emptyset}{P \mid Q > (\nu\overline{q})\langle Q'\rangle (P \mid Q'')} \qquad \frac{P > (\nu\overline{p})\langle P'\rangle P''}{\mathbf{!}P > (\nu\overline{p})\langle P'\rangle (P'' \mid \mathbf{!}P)}$$

Definition of Labelled Transitions

- First, a *labelled transition* $P \xrightarrow{M} P'$ for $M \in \{in\ n,\ out\ n,\ open\ n\}$, means that P has effect M on parent of sibling n, and evolves to P'.

M-transitions, where $M \in \{in\ n,\ out\ n,\ open\ n\}$:

$$\frac{P > (\nu \overline{p})\langle M.P'\rangle P'' \qquad \{\overline{p}\} \cap fn(M) = \varnothing}{P \xrightarrow{M} (\nu \overline{p})(P' \mid P'')}$$

Ex: $$\frac{out\ a.in\ b.\langle M\rangle > (\nu)\langle out\ a.in\ b.\langle M\rangle\rangle \mathbf{0}}{out\ a.in\ b.\langle M\rangle \xrightarrow{out\ a} in\ b.\langle M\rangle \mid \mathbf{0}}$$

- Second, a labelled transition $P \xrightarrow{\tau} P'$ means that P internally evolves in one step to P'.:

τ-transitions:

$$\frac{P > (\nu\overline{p})\langle n[Q]\rangle P' \quad Q > (\nu\overline{q})\langle m[R]\rangle Q' \quad R \xrightarrow{out\ n} R' \quad n \notin \{\overline{q}\}}{P \xrightarrow{\tau} (\nu\overline{p})((\nu\overline{q})(m[R'] \mid n[Q']) \mid P')}$$

Ex:

$$\frac{\begin{array}{c} a[msg[out\ a.in\ b.\langle M\rangle]] > (\nu)\langle a[msg[out\ a.in\ b.\langle M\rangle]]\rangle\mathbf{0} \\ msg[out\ a.in\ b.\langle M\rangle]] > (\nu)\langle msg[out\ a.in\ b.\langle M\rangle]\rangle\mathbf{0} \\ out\ a.in\ b.\langle M\rangle \xrightarrow{out\ a} in\ b.\langle M\rangle \mid \mathbf{0} \end{array}}{a[msg[out\ a.in\ b.\langle M\rangle]] \xrightarrow{\tau} msg[in\ b.\langle M\rangle \mid \mathbf{0}] \mid a[\mathbf{0}] \mid \mathbf{0}}$$

Theorems about Hardening and Labelled Transitions

- If $P > (\nu\overline{p})\langle P'\rangle P''$ then $P \equiv (\nu\overline{p})(P' \mid P'')$.
- If $P \equiv Q$ and $Q > (\nu\overline{r})\langle Q'\rangle Q''$ then there are P' and P'' with $P > (\nu\overline{r})\langle P'\rangle P''$, $P' \equiv Q'$, and $P'' \equiv Q''$.
- $P \longrightarrow Q$ if and only if $P \xrightarrow{\tau} \equiv Q$.
- $P{\Downarrow}n$ if and only if there are Q, $\overline{q}$, Q', Q'' such that $P \xrightarrow{\tau}{}^{*} Q$, $Q > (\nu\overline{q})\langle n[Q']\rangle Q''$, and $n \notin \{\overline{q}\}$.

Hence, we solve the problem of proving $\boldsymbol{C(0)}{\Downarrow}n$ implies $\boldsymbol{C(P)}{\Downarrow}n$.

Problem 2: The Trouble with Contexts

- General contexts $\boldsymbol{C(\bullet)}$ possess neither the desirable properties:
  - they have a unique hole, preserved by reduction.
  - they are identified up to alpha-conversion.
- Usual solution: identify a limited set of contexts satisfying those properties, and:
  - a *Context Lemma*: $P \simeq Q$ if and only if for all limited context H and names n, that $H\{P\}\Downarrow n \Leftrightarrow H\{Q\}\Downarrow n$.
- In CCS and the π-calculus, the limited contexts are simply parallel observers; $H ::= \bullet \mid R$. [De Nicola and Hennessy]

Parallel Observers in the Ambient Calculus

- Let $P = out\ p.\mathbf{0}$ and $Q = \mathbf{0}$.
  - We have $P \mid R \Downarrow n \Leftrightarrow Q \mid R \Downarrow n$ for all n and R,
  - while if $\boldsymbol{C}(\bullet) = p[m[\bullet]]$ we have $\boldsymbol{C}(P){\Downarrow}m$ but not $\boldsymbol{C}(Q){\Downarrow}m$.
- Therefore, the set of parallel observers does not satisfy a context lemma in the ambient calculus.

Solution: Harnesses

- Our solution is to augment parallel observers with ambients:

Harnesses:

$H ::=$	harnesses
$\bullet$	unique hole
$(\nu n)P$	restriction
$P \mid H$	left composition
$H \mid Q$	right composition
$n[H]$	ambient

- We identify harnesses up to alpha-conversion: if $H = (\nu n)\bullet$, its instantiation $H\{n[]\}$ is $(\nu n')n[]$.
- By a careful analysis, we have proved that harnesses satisfy the desired context lemma.

Problem 3: Analyzing $H\{P\} \longrightarrow Q$

- In lots of proofs that appeal to our context lemma, we have the problem of analyzing reductions like $H\{P\} \longrightarrow Q$.
- Intuitively, either P or H evolves on its won, or they interact. The best formalization we have found of this is:
  - An *Activity Lemma*: $H\{P\} \longrightarrow R$ if and only if:

(Act Proc)	$P \longrightarrow P'$ with $R \equiv H\{P'\}$, or
(Act Har)	$H \longrightarrow H'$ with $R \equiv H'\{P\}$, or
(Act Inter)	$H \circ R \rightsquigarrow P$

(large case analysis of all possible labeled-transition interactions of a process with a harness)

- Although the formulation of the activity lemma is very complex (due to numerous ways in which ambients may interact), the activity lemma can be used comfortably in specific proofs.

Summary: Difficulty of Deriving Ambient Equations

- Problem 1: Reduction defined using structural congruence.
  - Solution: use $P > (\nu\overline{p})\langle P'\rangle P''$ and $P \xrightarrow{\tau} P'$ instead.

    Thm 1: $P \longrightarrow Q \iff P \xrightarrow{\tau} \equiv Q$
- Problem 2: Equivalence defined using arbitrary contexts.
  - Solution: use a context lemma for limited contexts.

    Thm 2: $P \simeq Q \iff \forall H,n.\ H\{P\}{\Downarrow}n \Leftrightarrow H\{Q\}{\Downarrow}n$
- Problem 3: Analyzing the reductions of a process in harness.
  - Solution: use an activity lemma.

    Thm 3: $H\{P\} \longrightarrow R \iff (P \longrightarrow P' \wedge R \equiv H\{P'\}) \vee$
    $(H \longrightarrow H' \wedge R \equiv H'\{P\}) \vee (H \circ R \rightsquigarrow P)$
- Examples: Perfect firewall and cipher equations; firewall crossing.

Assessment

- We wanted to prove equations asserting simple security properties of ambients, and we succeeded. But:
- Reasoning about higher-order hierarchical processes is difficult (others have found similar difficulties). Is there an alternative?
- Defining reduction from structural congruence is a mixed blessing. Laws $P \mid \mathbf{0} \equiv \mathbf{0}$ and $!\mathbf{0} \equiv \mathbf{0}$, are not so innocent!
- Defining labeled transitions from hardening works well.
- Activity lemma is useful but suspicious. Is there a less monolithic definition?

Types for Mobile Ambients

Need for Distinctions

The syntax does not distinguish between names and capabilities, therefore it permits strange terms like:

$in\ n[P]$	(stuck)
$n.P$	(stuck)

This cannot be avoided by a more precise syntax, because such terms may be generated by interactions:

$$\langle in\ n\rangle \mid (m).m[P] \longrightarrow in\ n[P]$$

$$\langle n\rangle \mid (m).m.P \longrightarrow n.P$$

$$(m).\ (m.P \mid m[Q])$$ (tests whether m is a name or a capability!)

We have two sorts of things (ambient names and capabilities) that we want to use consistently. A type system should do the job.
Desired property: a well-typed program does not produce insane terms like $in\ n[P]$ and $n.P$.

Exchange Types

$W ::=$	message types
$\quad Amb[T]$	ambient name allowing T exchange
$\quad Cap[T]$	capability unleashing T exchange
$T ::=$	exchange types
$\quad Shh$	no exchange
$\quad W_1 \times \ldots \times W_k$	tuple exchange (**1** is the null product)

- A quiet ambient: $Amb \triangleq Amb[Shh]$
- A harmless capability: $Cap \triangleq Cap[Shh]$
- A synchronization ambient: $Amb[\mathbf{1}]$
- Ambient containing harmless capabilities: $Amb[Cap[Shh]]$
- Ex.: A capability that may unleash the exchange of names for quiet ambients: $Cap[Amb[Shh]]$

Polyadic Ambient Calculus

$P,Q ::=$

$(\nu n{:}W)P$

$\mathbf{0}$

$P|Q$

$!P$

$M[P]$

$M.P$

$(n_1{:}W_1, \ldots, n_k{:}W_k).P$

$\langle M_1, \ldots, M_k \rangle$

$M,N ::=$

n

$in\ M$

$out\ M$

$open\ M$

ε

$M.N$

Reduction

$$n[\mathit{in}\ m.\ P \mid Q] \mid m[R] \longrightarrow m[n[P \mid Q] \mid R]$$
$$m[n[\mathit{out}\ m.\ P \mid Q] \mid R] \longrightarrow n[P \mid Q] \mid m[R]$$
$$\mathit{open}\ n.\ P \mid n[Q] \longrightarrow P \mid Q$$
$$(n_1{:}W_1, ..., n_k{:}W_k).P \mid \langle M_1, ..., M_k\rangle \longrightarrow P\{n_1{\leftarrow}M_1, ..., n_k{\leftarrow}M_k\}$$

$$P \longrightarrow Q \quad \Rightarrow \quad (\nu n{:}W)P \longrightarrow (\nu n{:}W)Q$$
$$P \longrightarrow Q \quad \Rightarrow \quad n[P] \longrightarrow n[Q]$$
$$P \longrightarrow Q \quad \Rightarrow \quad P \mid R \longrightarrow Q \mid R$$
$$P' \equiv P, P \longrightarrow Q, Q \equiv Q' \;\Rightarrow\; P' \longrightarrow Q'$$

Intuitions: Typing of Processes

- If M is a T, then $\langle M\rangle$ is a process that exchanges T.

  $$M : W \Rightarrow \langle M\rangle : W$$

- If P is a process that may exchange T, then $(n{:}T).P$ is too.

  $$P : W \Rightarrow (n{:}W).P : W$$

- If P and Q are processes that may exchange T, then $P|Q$ is too. (Similarly for $!P$.)

  $$P : T, Q : T \Rightarrow P|Q : T$$

- Both **0** and $n[P]$ exchange nothing at the current level, so they can have any exchange type, and can be added in parallel freely.

- Therefore, W-inputs and W-outputs are tracked so that they match correctly when placed in parallel.

Intuitions: Typing of Open

- We have to worry about *open*, which might open-up a *T*-ambient and unleash *T*-exchanges inside an *S*-ambient.
- We decorate each ambient name with the *T* that can be exchanged in ambients of that name. Different ambients may permit internal exchanges of different types.

$$n : Amb[T],\ P : T \Rightarrow n[P] \textit{ is legal and } n[P] : S$$

- If *n* permits *T*-exchanges, then *open n* may unleash *T*-exchanges in the current location.

$$n : Amb[T] \Rightarrow open\ n : Cap[T]$$

- Any process that uses a *Cap*[*T*] had better be a process that already exchanges *T*, because new *T*-exchanges may be unleashed.

$$M : Cap[T],\ P : T \Rightarrow M.P : T$$

Judgments

$E \vdash \diamond$	good environment
$E \vdash M : W$	good expression of type W
$E \vdash P : T$	good process that exchanges T

Rules

$$\frac{}{\emptyset \vdash \Diamond} \qquad \frac{E \vdash \Diamond \quad n \notin dom(E)}{E, n{:}W \vdash \Diamond} \qquad \frac{E', n{:}W, E'' \vdash \Diamond}{E', n{:}W, E'' \vdash n : W}$$

$$\frac{E \vdash \Diamond}{E \vdash \varepsilon : Cap[T]} \qquad \frac{E \vdash M : Cap[T] \quad E \vdash M' : Cap[T]}{E \vdash M.M' : Cap[T]}$$

$$\frac{E \vdash M : Amb[S]}{E \vdash in\ M : Cap[T]} \qquad \frac{E \vdash M : Amb[S]}{E \vdash out\ M : Cap[T]} \qquad \frac{E \vdash M : Amb[T]}{E \vdash open\ M : Cap[T]}$$

$$\frac{E \vdash M : Cap[T] \quad E \vdash P : T}{E \vdash M.P : T} \qquad \frac{E \vdash M : Amb[T] \quad E \vdash P : T}{E \vdash M[P] : S}$$

$$\frac{E, n_1{:}W_1, \ldots, n_k{:}W_k \vdash P : W_1 \times \ldots \times W_k}{E \vdash (n_1{:}W_1, \ldots, n_k{:}W_k).P : W_1 \times \ldots \times W_k} \qquad \frac{E \vdash M_1{:}W_1 \quad \ldots \quad E \vdash M_k{:}W_k}{E \vdash \langle M_1, \ldots, M_k \rangle : W_1 \times \ldots \times W_k}$$

$$\frac{E, n{:}Amb[T] \vdash P : S}{E \vdash (\nu n{:}Amb[T])P : S} \qquad \frac{E \vdash \diamond}{E \vdash \mathbf{0} : T}$$

$$\frac{E \vdash P : T \qquad E \vdash Q : T}{E \vdash P|Q : T} \qquad \frac{E \vdash P : T}{E \vdash\, !P : T}$$

- Ex.: A process that outputs names of quiet ambients:

$$E \vdash\, !(\nu n{:}Amb)\langle n \rangle : Amb$$

- Ex.: A capability that may unleash *S*-exchanges:

$$n{:}Amb[T], m{:}Amb[S] \vdash in\ n.\ open\ m : Cap[S]$$

Prop (Subject Reduction)

If $E \vdash P : U$ and $P \longrightarrow Q$ then $E \vdash Q : U$.

Typed Polyadic Asynchronous π-calculus

$\langle\!\langle Ch[T_1, ..., T_k] \rangle\!\rangle \quad \triangleq$

$\quad Amb[\langle\!\langle T_1 \rangle\!\rangle \times \langle\!\langle T_1 \rangle\!\rangle \times ... \times \langle\!\langle T_k \rangle\!\rangle \times \langle\!\langle T_k \rangle\!\rangle]$

$\langle\!\langle (\nu^\pi n{:}Ch[T])P \rangle\!\rangle \quad \triangleq$

$\quad (\nu n{:}\langle\!\langle Ch[T] \rangle\!\rangle)\ (\nu n^p{:}\langle\!\langle Ch[T] \rangle\!\rangle)\ n[!open\ n^p] \mid \langle\!\langle P \rangle\!\rangle$

$\langle\!\langle n(x_1{:}T_1, ..., x_k{:}T_k).P \rangle\!\rangle \quad \triangleq$

$\quad (\nu p{:}Amb)\ (open\ p \mid$

$\quad\quad n^p[in\ n.\ (x_1,x^p{}_1{:}\langle\!\langle T_1 \rangle\!\rangle, ..., x_k,x^p{}_k{:}\langle\!\langle T_k \rangle\!\rangle).\ p[out\ n.\ \langle\!\langle P \rangle\!\rangle]])$

$\langle\!\langle n\langle n_1, ..., n_k\rangle \rangle\!\rangle \quad \triangleq$

$\quad n^p[in\ n.\ \langle n_1, n^p{}_1, ..., n_k, n^p{}_k\rangle]$

$\langle\!\langle P|Q \rangle\!\rangle \quad \triangleq \quad \langle\!\langle P \rangle\!\rangle | \langle\!\langle Q \rangle\!\rangle$

$\langle\!\langle !P \rangle\!\rangle \quad \triangleq \quad !\langle\!\langle P \rangle\!\rangle$

Derived Rules

$$\langle\!\langle E \vdash P \rangle\!\rangle \quad \triangleq \quad \langle\!\langle E \rangle\!\rangle \vdash \langle\!\langle P \rangle\!\rangle : Shh$$

$$\langle\!\langle \emptyset, n_1{:}W_1, \ldots, n_k{:}W_k \rangle\!\rangle \quad \triangleq$$
$$\emptyset, n_1{:}\langle\!\langle W_1 \rangle\!\rangle, n_1{}^p{:}\langle\!\langle W_1 \rangle\!\rangle, \ldots, n_k{:}\langle\!\langle W_k \rangle\!\rangle, n_k{}^p{:}\langle\!\langle W_k \rangle\!\rangle$$

$$\langle\!\langle E, n{:}Ch[W_1, \ldots, W_k] \vdash P \rangle\!\rangle \quad \Rightarrow \quad \langle\!\langle E \vdash (\nu^\pi n{:}Ch[W_1, \ldots, W_k])P \rangle\!\rangle$$

$$\langle\!\langle E \vdash n : Ch[W_1, \ldots, W_k] \rangle\!\rangle, \langle\!\langle E \vdash n_1 : W_1 \rangle\!\rangle, \ldots, \langle\!\langle E \vdash n_k : W_k \rangle\!\rangle$$
$$\Rightarrow \quad \langle\!\langle E \vdash n\langle n_1, \ldots, n_k \rangle \rangle\!\rangle$$

$$\langle\!\langle E \vdash n : Ch[W_1, \ldots, W_k] \rangle\!\rangle, \langle\!\langle E, n_1{:}W_1, \ldots, n_k{:}W_k \vdash P \rangle\!\rangle$$
$$\Rightarrow \quad \langle\!\langle E \vdash n(n_1{:}W_1, \ldots, n_k{:}W_k).P \rangle\!\rangle$$

$$\langle\!\langle E \vdash P \rangle\!\rangle, \langle\!\langle E \vdash Q \rangle\!\rangle \quad \Rightarrow \quad \langle\!\langle E \vdash P \mid Q \rangle\!\rangle$$

$$\langle\!\langle E \vdash P \rangle\!\rangle \quad \Rightarrow \quad \langle\!\langle E \vdash !P \rangle\!\rangle$$

Typed Call-by-Value λ-calculus

$$[\![A{\rightarrow}B]\!] \;\triangleq\; Ch[[\![A]\!], Ch[[\![B]\!]]]$$

$$[\![E \vdash b{:}T]\!] \;\triangleq\; [\![E]\!] \vdash (\nu^{\pi} k{:}Ch[[\![T]\!]])\, [\![b]\!]_k : Shh$$

$$[\![x_T]\!]_k \;\triangleq\; k\langle x\rangle$$

$$[\![\lambda x{:}A.b_{A\rightarrow B}]\!]_k \;\triangleq$$
$$\qquad (\nu^{\pi} n{:}[\![A{\rightarrow}B]\!])\, (k\langle n\rangle \mid !n(x{:}[\![A]\!], k'{:}Ch[[\![B]\!]]).\, [\![b_B]\!]_{k'})$$

$$[\![b_{A\rightarrow B}(a_A)]\!]_k \;\triangleq$$
$$\qquad (\nu^{\pi} k'{:}Ch[[\![A{\rightarrow}B]\!]], k''{:}Ch[[\![A]\!]])$$
$$\qquad\qquad ([\![b]\!]_{k'} \mid k'(x{:}[\![A{\rightarrow}B]\!]).\, ([\![a]\!]_{k''} \mid k''(y{:}[\![A]\!]).\, x\langle y, k\rangle))$$

$$[\![x{:}T \vdash x{:}T]\!] \;=\; [\![x{:}T]\!] \vdash (\nu^{\pi} k{:}Ch[[\![T]\!]])\, k\langle x\rangle : Shh$$
$$=\; x{:}[\![T]\!], x^p{:}[\![T]\!] \vdash (\nu k{:}Amb[[\![T]\!]\times[\![T]\!]])\, k[!open\ k^p] \mid k^p[in\ k.\langle x, x^p\rangle] : Shh$$

Records

A record r containing cells c_i has the general structure $r[\ \dots \mid c_i^{buf}[\langle M_i\rangle \mid\ !open\ c_i^{ip}] \mid \dots\ ]$, where r is the cell container, c_i^{buf} are the value containers for each cell, c_i^{ip} are input packets for reading and writing cell contents, and M_i are the cell contents.

$\langle\!\langle record\ r\rangle\!\rangle \ \triangleq\ r[]$

$\langle\!\langle add\ r\ c\ M\rangle\!\rangle \ \triangleq\ c^{buf}[!open\ c^{ip} \mid in\ r.\ \langle M\rangle]$

$\langle\!\langle get\ r\ c\ (x{:}W).\ P_S\rangle\!\rangle \ \triangleq$

$\quad (\nu op{:}Amb[S])\ (open\ op.\ op[] \mid$

$\quad\quad c^{ip}[in\ r.\ in\ c^{buf}.\ (x{:}W).$

$\quad\quad\quad (\langle x\rangle \mid op[out\ c^{buf}.\ out\ r.\ open\ op.\ \langle\!\langle P\rangle\!\rangle])])$

$\langle\!\langle set\ r\ c\ \langle M_W\rangle.\ P_S\rangle\!\rangle \ \triangleq$

$\quad (\nu op{:}Amb[S])\ (open\ op.\ op[] \mid$

$\quad\quad c^{ip}[in\ r.\ in\ c^{buf}.\ (x{:}W).$

$\quad\quad\quad (\langle M\rangle \mid op[out\ c^{buf}.\ out\ r.\ open\ op.\ \langle\!\langle P\rangle\!\rangle])])$

Field Types

The type of names of a record field holding W may be denoted by:

$$\mathit{Field}[W]$$

This notation can be translated to the ambient calculus by mapping each environment name $n : \mathit{Field}[W]$ to two environment names n^{buf}, $n^{ip} : \mathit{Amb}[W]$, and by mapping each restriction $(\nu n{:}\mathit{Field}[W])\ P$ to the restrictions $(\nu n^{buf}{:}\mathit{Amb}[W])\ (\nu n^{ip}{:}\mathit{Amb}[W])\ P$.

Telestrip'd

$[\![\mathit{Agent}[W_1, \ldots, W_k]]\!] \quad \triangleq \quad \mathit{Amb}[[\![W_1]\!] \times \ldots \times [\![W_k]\!]]$

Typing: each agent can *welcome* only agents talking about a single type, but can *meet* different agents that talk about different types.

$[\![\mathrm{Net}]\!] ::= \qquad\qquad : \mathit{Shh}$

$[\![\mathit{noplace}]\!] \quad \triangleq \quad \mathbf{0}$

$[\![\mathit{place}\ p[\mathit{Arena}]]\!] \quad \triangleq \quad p[[\![\mathit{Arena}]\!]_p] \quad (\text{if } p{:}\mathit{Amb}[\mathit{Shh}])$

$[\![\mathit{Net} \mid \mathit{Net}]\!] \quad \triangleq \quad [\![\mathit{Net}]\!] \mid [\![\mathit{Net}]\!]$

$[\![\mathrm{Arena}]\!]_p ::= \qquad : \mathit{Shh} \quad (\text{if } p{:}\mathit{Amb}[\mathit{Shh}])$

$[\![\mathit{empty}]\!]_p \triangleq \mathbf{0}$

$[\![\mathit{agent}\ (a{:}\mathit{Agent}[W_1, ..., W_k])[\mathit{Code}]]\!]_p \triangleq$

$\quad (\nu a{:}[\![\mathit{Agent}[W_1, ..., W_k]]\!])$

$\qquad a[\mathit{record\ sut} \mid \mathit{add\ sut\ at\ p} \mid [\![\mathit{Code}]\!]_n]$

$[\![\mathit{Arena} \mid \mathit{Arena}]\!]_p \triangleq [\![\mathit{Arena}]\!]_p \mid [\![\mathit{Arena}]\!]_p$

(Built-in "resources" of places can be modeled as static agents.)

$[\![Code]\!]_a ::= \quad : [\![W_1]\!]\times\ldots\times[\![W_k]\!] \quad (\text{if } a : [\![Agent[W_1, \ldots, W_k]]\!])$

$[\![stop]\!]_a \triangleq \mathbf{0}$

$[\![go\ p.\ Code]\!]_a \triangleq$

$\quad get\ sut\ at(q{:}Amb[Shh]).\ set\ \ sut\ at\langle p\rangle.\ out\ q.\ in\ p.\ [\![Code]\!]_a$

$[\![spawn\ (a'{:}Agent[W_1, \ldots, W_k])[Code'].\ Code]\!]_a \triangleq \quad (\text{for } a' \neq a)$

$\quad get\ sut\ at(p{:}Amb[Shh]).\ (\nu a', u{:}\langle\!\langle Agent[W_1, \ldots, W_k]\rangle\!\rangle)$

$\quad\quad (a'[record\ sut \mid add\ sut\ at\ p \mid out\ a.\ open\ u.\ \langle\!\langle Code'\rangle\!\rangle_{a'}]$

$\quad\quad \mid open\ u.\ \langle\!\langle Code\rangle\!\rangle_a$

$\quad\quad \mid (\nu t{:}Amb[Shh])\ t[out\ a.\ in\ a'.\ out\ a'.$

$\quad\quad\quad\quad (u[out\ t.\ in\ a'] \mid u[out\ t.\ in\ a])])$

$$[\![\mathit{meet}\ n\langle n_1, \ldots, n_k\rangle.\ \mathit{Code}]\!]_a \triangleq$$
$$(\nu z{:}\langle\!\langle \mathit{Agent}[W_1, \ldots, W_k]\rangle\!\rangle)$$
$$z[\mathit{out}\ a.\ \mathit{in}\ n.\ n[\mathit{out}\ z.\ \mathit{open}\ z.\ \langle n_1, \ldots, n_k\rangle]] \mid \langle\!\langle \mathit{Code}\rangle\!\rangle_a$$

$$[\![\mathit{welcome}\ (x_1, \ldots, x_m).\ \mathit{Code}]\!]_a \triangleq$$
$$\mathit{open}\ a \mid (n_1{:}\langle\!\langle W_1\rangle\!\rangle, \ldots, n_k{:}\langle\!\langle W_k\rangle\!\rangle).\ \langle\!\langle \mathit{Code}\rangle\!\rangle_a$$

$$[\![\mathit{folder}\ n\ m.\ \mathit{Code}]\!]_a \triangleq (\nu n{:}\mathit{Field}[\langle\!\langle W\rangle\!\rangle])\ (\mathit{add\ sut}\ n\ m \mid \langle\!\langle \mathit{Code}\rangle\!\rangle_a)$$

$$[\![\mathit{get}\ n(x{:}W).\ \mathit{Code}]\!]_a \triangleq \mathit{get\ sut}\ n(x{:}\langle\!\langle W\rangle\!\rangle).\ \langle\!\langle \mathit{Code}\rangle\!\rangle_a$$

$$[\![\mathit{set}\ n\langle m\rangle.\ \mathit{Code}]\!]_a \triangleq \mathit{set\ sut}\ n\langle m\rangle.\ \langle\!\langle \mathit{Code}\rangle\!\rangle_a$$

...

Static Constraints on Mobility

Introduction

- The Ambient Calculus is a process calculus based on:
  - Local communication.
  - Mobility regulated by capabilities.
- The calculus can be decorated with static information to restrict either aspect (or both):
  - *Exchange control systems* to restrict communication.
  - *Access control systems* to restrict mobility.

- In [Cardelli-Gordon POPL'99] we have investigated *exchange types* (which subsume type systems for processes and functions).
- In [Cardelli-Ghelli-Gordon ICALP'99] we investigate *immobility* and *locking* annotations, which are simple predicates about mobility.
- Here we consider further static predicates about mobility.

Name Groups (Sorts)

- We would like to say, within the type system, something like:

  The ambient named *n* can enter the ambient named *m*.

  But this would bring us straight into dependent types, since names are value-level entities. This is no fun.

- Instead, we introduce type-level name groups *G*,*H*, and we say:

  Ambients of group *G* can enter ambients of group *H*.

- Groups can be seen as related to π-calculus sorting mechanisms. We add group *creation*.

- Group creation is surprisingly interesting. For example, it has the effect of statically blocking certain communications, and can therefore prevent the “accidental” escape of capabilities that is a major concern in practical systems.

Grand Plan

- Investigate the notion of name groups and group creation. (This can be applied directly to the π-calculus as well.)
- Study the judgment "*process P may cross ambients of group G*". (This reduces to immobility annotations when a process can cross no groups.)
- Study the predicate "*process P may open ambients of group G*". (This reduces to locking annotations, when a group can be opened by no processes.)
- Recursive types. (To capture mutually-recursive sorting, π-calculus style.)
- Mix and stir well.

Typed Ambient Calculus with Groups

- Just one new process construct:

  $(\nu G)P$

  to create a new group G with scope P.

- Just one new type construct:

  $G[T]$

  as the type of names of group G that name ambients that contain T exchanges.

  The construct $G[T]$ replaces $Amb[T]$, where Amb can now be seen as the group of all names.

- So we can now write, e.g.: $(\nu G)\ (\nu n{:}G[Int])\ n[\langle 3\rangle \mid (x{:}Int).\ P]$

Types

$W ::=$	message types
$\quad G[T]$	ambient name in group G with T exchanges
$\quad Cap[T]$	capability unleashing T exchanges
$T ::=$	exchange types
$\quad Shh$	no exchange
$\quad W_1 \times \ldots \times W_k$	tuple exchange ($\mathbf{1}$ is the null product)

- A quiet ambient: $G[Shh]$
- A harmless capability: $Cap[Shh]$
- A synchronization ambient: $G[\mathbf{1}]$
- Ambient containing harmless capabilities: $G[Cap[Shh]]$
- A capability that may unleash the exchange of names for quiet ambients: $Cap[G[Shh]]$

Processes and Messages

$P,Q ::=$

- $(\nu G)P$
- $(\nu n{:}W)P$
- $\mathbf{0}$
- $P|Q$
- $!P$
- $M[P]$
- $M.P$
- $(n_1{:}W_1, \ldots, n_k{:}W_k).P$
- $\langle M_1, \ldots, M_k \rangle$

$M,N ::=$

- n
- $in\ M$
- $out\ M$
- $open\ M$
- ε
- $M.N$

Reduction

$$n[in\ m.\ P \mid Q] \mid m[R] \longrightarrow m[n[P \mid Q] \mid R]$$
$$m[n[out\ m.\ P \mid Q] \mid R] \longrightarrow n[P \mid Q] \mid m[R]$$
$$open\ n.\ P \mid n[Q] \longrightarrow P \mid Q$$
$$(n_1{:}W_1, ..., n_k{:}W_k).P \mid \langle M_1, ..., M_k\rangle \longrightarrow P\{n_1{\leftarrow}M_1, ..., n_k{\leftarrow}M_k\}$$

$$P \longrightarrow Q \quad \Rightarrow \quad (\nu G)P \longrightarrow (\nu G)Q$$
$$P \longrightarrow Q \quad \Rightarrow \quad (\nu n{:}W)P \longrightarrow (\nu n{:}W)Q$$
$$P \longrightarrow Q \quad \Rightarrow \quad n[P] \longrightarrow n[Q]$$
$$P \longrightarrow Q \quad \Rightarrow \quad P \mid R \longrightarrow Q \mid R$$
$$P' \equiv P,\ P \longrightarrow Q,\ Q \equiv Q' \ \Rightarrow \quad P' \longrightarrow Q'$$

Structural Congruence

$P \equiv P$	(Struct Refl)
$P \equiv Q \quad \Rightarrow \quad Q \equiv P$	(Struct Symm)
$P \equiv Q,\ Q \equiv R \quad \Rightarrow \quad P \equiv R$	(Struct Trans)
$P \equiv Q \quad \Rightarrow \quad (\nu G)P \equiv (\nu G)Q$	(Struct GRes)
$P \equiv Q \quad \Rightarrow \quad (\nu n{:}W)P \equiv (\nu n{:}W)Q$	(Struct Res)
$P \equiv Q \quad \Rightarrow \quad P \mid R \equiv Q \mid R$	(Struct Par)
$P \equiv Q \quad \Rightarrow \quad !P \equiv !Q$	(Struct Repl)
$P \equiv Q \quad \Rightarrow \quad M[P] \equiv M[Q]$	(Struct Amb)
$P \equiv Q \quad \Rightarrow \quad M.P \equiv M.Q$	(Struct Action)
$P \equiv Q \quad \Rightarrow$	(Struct Input)
$\quad (n_1{:}W_1, \ldots, n_k{:}W_k).P \equiv (n_1{:}W_1, \ldots, n_k{:}W_k).Q$	

$P \mid Q \equiv Q \mid P$	(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)
$(\nu n{:}W)(\nu m{:}W')P \equiv (\nu m{:}W')(\nu n{:}W)P \quad \text{if } n \neq m$	(Struct Res Res)
$(\nu n{:}W)(P \mid Q) \equiv P \mid (\nu n{:}W)Q \quad \text{if } n \notin fn(P)$	(Struct Res Par)
$(\nu n{:}W)(m[P]) \equiv m[(\nu n{:}W)P] \quad \text{if } n \neq m$	(Struct Res Amb)
$(\nu G)(\nu G')P \equiv (\nu G')(\nu G)P$	(Struct GRes GRes)
$(\nu G)(\nu n{:}W)P \equiv (\nu n{:}W)(\nu G)P \quad \text{if } G \notin fn(W)$	(Struct GRes Res)
$(\nu G)(P \mid Q) \equiv P \mid (\nu G)Q \quad \text{if } G \notin fn(P)$	(Struct GRes Par)
$(\nu G)(m[P]) \equiv m[(\nu G)P]$	(Struct GRes Amb)
$(\nu G)\mathbf{0} \equiv \mathbf{0}$	(Struct Zero GRes)
$(\nu n{:}W)\mathbf{0} \equiv \mathbf{0}$	(Struct Zero Res)
$P \mid \mathbf{0} \equiv P$	(Struct Zero Par)
$!\mathbf{0} \equiv \mathbf{0}$	(Struct Zero Repl)

$\varepsilon.P \equiv P$ (Struct ε)

$(M.M').P \equiv M.(M'.P)$ (Struct .)

Judgments

$E \vdash \diamond$	good environment
$E \vdash T$	good type
$E \vdash M : W$	good message of type W
$E \vdash P : T$	good process that exchanges T

Rules

Good Environments

$$\frac{}{\varnothing \vdash \Diamond} \qquad \frac{E \vdash W \quad n \notin dom(E)}{E, n{:}W \vdash \Diamond} \qquad \frac{E \vdash \Diamond \quad G \notin dom(E)}{E', G \vdash \Diamond}$$

Good Types

$$\frac{G \in dom(E) \quad E \vdash T}{E \vdash G[T]} \qquad \frac{E \vdash T}{E \vdash Cap[T]}$$

$$\frac{E \vdash \Diamond}{E \vdash Shh} \qquad \frac{E \vdash W_1 \quad \ldots \quad E \vdash W_k}{E \vdash W_1 \times \ldots \times W_m}$$

Good Messages

$$\frac{E', n{:}W, E'' \vdash \diamond}{E', n{:}W, E'' \vdash n : W}$$

$$\frac{E \vdash Cap[T]}{E \vdash \varepsilon : Cap[T]} \qquad \frac{E \vdash M : Cap[T] \quad E \vdash M' : Cap[T]}{E \vdash M.M' : Cap[T]}$$

$$\frac{E \vdash M : G[S] \quad E \vdash Cap[T]}{E \vdash in\ M : Cap[T]} \qquad \frac{E \vdash M : G[S] \quad E \vdash Cap[T]}{E \vdash out\ M : Cap[T]}$$

$$\frac{E \vdash M : G[T]}{E \vdash open\ M : Cap[T]}$$

Good Processes

$$\frac{E, G \vdash P : S \qquad G \notin fn(S)}{E \vdash (\nu G)P : S} \qquad \frac{E, n{:}G[T] \vdash P : S}{E \vdash (\nu n{:}G[T])P : S}$$

$$\frac{E \vdash M : Cap[T] \qquad E \vdash P : T}{E \vdash M.P : T} \qquad \frac{E \vdash M : G[T] \qquad E \vdash P : T \qquad E \vdash S}{E \vdash M[P] : S}$$

$$\frac{E \vdash T}{E \vdash \mathbf{0} : T} \qquad \frac{E \vdash P : T \qquad E \vdash Q : T}{E \vdash P \mid Q : T} \qquad \frac{E \vdash P : T}{E \vdash\, !P : T}$$

$$\frac{E, n_1{:}W_1, \ldots, n_k{:}W_k \vdash P : W_1 \times \ldots \times W_k}{E \vdash (n_1{:}W_1, \ldots, n_k{:}W_k).P : W_1 \times \ldots \times W_k} \qquad \frac{E \vdash M_1{:}W_1 \quad \ldots \quad E \vdash M_k{:}W_k}{E \vdash \langle M_1, \ldots, M_k \rangle : W_1 \times \ldots \times W_k}$$

Prop (Subject Reduction)

If $E \vdash P : T$ and $P \longrightarrow Q$ then there exists Gs such that $Gs, E \vdash Q : T$.

$\square$

Scope Extrusion

Consider:

$$(\nu G)\ (\nu n{:}G[Shh])\ \langle n\rangle$$

It cannot be typed, because G escapes along with the messages n:

$$G, n{:}G[Shh] \vdash \langle n\rangle : G[Shh]$$
$$\Rightarrow \quad G \vdash (\nu n{:}G[Shh])\ \langle n\rangle : G[Shh]$$
$$\not\Rightarrow \quad \vdash (\nu G)\ (\nu n{:}G[Shh])\ \langle n\rangle : G[Shh] \qquad (G \in fn(G[Shh]))$$

Similarly,

$$(\nu m{:}W)\ m[(\nu G)\ (\nu n{:}G[Shh])\ \langle n\rangle]$$

cannot be typed, because it contains an untypable term. But can we type this one, which is equivalent by extrusion:

$$(\nu m{:}W)\ (\nu G)\ m[(\nu n{:}G[Shh])\ \langle n\rangle]$$

Again:

$$(\nu m{:}W)\ (\nu G)\ m[(\nu n{:}G[Shh])\ \langle n\rangle]$$

This looks like it may be typable (breaking subject reduction) because the message $\langle n\rangle : G[Shh]$ is confined to the ambient m, and $m[...]$ can have an arbitrary type, e.g. Shh, that does not contain G. Therefore (νG) would not "see" any message escaping.

However, consider the type of m: it must have the form $H[T]$, where H is some group, and T is the type of messages exchanged inside m. Hmmm..., but that's $G[Shh]$. So we would have:

$$(\nu m{:}H[G[Shh]])\ (\nu G)\ m[(\nu n{:}G[Shh])\ \langle n\rangle]$$

which is not typable. Good.

Finally, this variation (not equivalent to the previous one) is typable:

$$(\nu G)\ (\nu m{:}H[G[Shh]])\ m[(\nu n{:}G[Shh])\ \langle n\rangle]$$

Therefore, we can create a new group G, and then ambients in which names of that group are exchanged.

We are guaranteed that names of group G can never be communicated to processes outside of the initial scope of G, simply because those processes cannot name G to receive the messages.

Secrecy

Therefore:

Names of group G are *secret* (forever) within the scope of νG.

This is in contrast with the situation with ordinary name restriction, where a name that is initially held secret (e.g. a key) may accidentally be given away and misused (e.g. to decrypt previous messages).

- Scoping of names can be extruded too far, inadvertently.
- Scoping of groups offers better protection. Still, it too can be extruded arbitrarily, so it does not impede the mobility of processes that carry secrets. It just prevents those processes from giving away the secrets.

Speculation: groups might help in establishing "forward secrecy" properties.

Crossing Control

$W ::=$	message types
$G[\frown Hs,T]$	ambient name in group G, containing processes that may cross Hs and exchange T
$Cap[\frown Hs,T]$	capability unleashing Hs crossings and T exchanges

$E \vdash P : \frown Hs,T$	process that exchanges T and crosses Hs
$\nu n{:}G[\frown\{\},T]$	a name for immobile ambients

Opening Control

$W ::=$	message types
$G[°Hs,T]$	ambient name in group G, containing processes that may open Hs and exchange T
$Cap[°Hs,T]$	capability unleashing Hs openings and T exchanges

$E \vdash P : °Hs,T$	process that exchanges T and opens Hs
$\nu n{:}G[°Hs,T]$	a name for locked ambients, where $G \notin Hs$

(We require $G \in Hs$ for *open n* to be typeable, because the opening of G may unleash further openings of Hs. With this rule the transitive closure of possible openings must be present already in the types.)

Modal Logics

Introduction

- We have been looking for ways to express properties of mobile computations, E.g.:
  - "Here today, gone tomorrow."
  - "Eventually the agent crosses the firewall."
  - "Every agent carries a suitcase."
  - "Somewhere there is a virus."
  - "There is always at most one ambient called *n* here."

- Options include equational reasoning, reasoning on traces, or...

Spatial Logic

- Devise a process logic that can talk about *space* as well as time.
- The ambient calculus has a spatial structure given by the nesting of ambients: we want a logic that can talk about that structure:

Process		*Formula*	
$\mathbf{0}$	(void)	$\mathbf{0}$	(there is nothing here)
$n[P]$	(location)	$n[\mathcal{A}]$	(there is one thing here)
$P \mid Q$	(composition)	$\mathcal{A} \mid \mathcal{B}$	(there are two things here)

- Could not find much of close relevance in the literature, except for Mads Dam's thesis and Urquhart's semantics, but we quickly diverge from both.

Restriction-free Ambient Calculus

$P,Q : \Pi ::=$	$M ::=$
$\mathbf{0}$	n
$P \mid Q$	$in\ M$
$!P$	$out\ M$
$M[P]$	$open\ M$
$M.P$	ε
$(n).P$	$M.M'$
$\langle M \rangle$	

$n[in\ m.\ P \mid Q] \mid m[R]$	$\longrightarrow$	$m[n[P \mid Q] \mid R]$	(enter reduction)
$m[n[out\ m.\ P \mid Q] \mid R]$	$\longrightarrow$	$n[P \mid Q] \mid m[R]$	(exit reduction)
$open\ m.\ P \mid m[Q]$	$\longrightarrow$	$P \mid Q$	(open reduction)
$(m).P \mid \langle M \rangle$	$\longrightarrow$	$P\{m \leftarrow M\}$	(read reduction)

Structural Congruence

$P \equiv P$	(Struct Refl)
$P \equiv Q \quad \Rightarrow \quad Q \equiv P$	(Struct Symm)
$P \equiv Q, Q \equiv R \quad \Rightarrow \quad P \equiv R$	(Struct Trans)
$P \equiv Q \quad \Rightarrow \quad P \mid R \equiv Q \mid R$	(Struct Par)
$P \equiv Q \quad \Rightarrow \quad !P \equiv !Q$	(Struct Repl)
$P \equiv Q \quad \Rightarrow \quad M[P] \equiv M[Q]$	(Struct Amb)
$P \equiv Q \quad \Rightarrow \quad M.P \equiv M.Q$	(Struct Action)
$P \equiv Q \quad \Rightarrow \quad (x).P \equiv (x).Q$	(Struct Input)
$\varepsilon.P \equiv P$	(Struct ε)
$(M.M').P \equiv M.M'.P$	(Struct .)
$P \mid Q \equiv Q \mid P$	(Struct Par Comm)
$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$	(Struct Par Assoc)
$P \mid \mathbf{0} \equiv P$	(Struct Par Zero)

$\mathbf{!}(P \mid Q) \equiv \mathbf{!}P \mid \mathbf{!}Q$ (Struct Repl Par)

$\mathbf{!0} \equiv \mathbf{0}$ (Struct Repl Zero)

$\mathbf{!}P \equiv P \mid \mathbf{!}P$ (Struct Repl Copy)

$\mathbf{!}P \equiv \mathbf{!!}P$ (Struct Repl Repl)

These axioms (particularly the ones for **!**) are sound and complete with respect to equality of *spatial trees*: edge-labeled finite-depth unordered trees, with infinite-branching but finitely many distinct labels under each node.

Reduction

$n[in\ m.\ P \mid Q] \mid m[R] \longrightarrow m[n[P \mid Q] \mid R]$	(Red In)
$m[n[out\ m.\ P \mid Q] \mid R] \longrightarrow n[P \mid Q] \mid m[R]$	(Red Out)
$open\ n.\ P \mid n[Q] \longrightarrow P \mid Q$	(Red Open)
$(n).P \mid \langle M \rangle \longrightarrow P\{n \leftarrow M\}$	(Red Comm)
$P \longrightarrow Q \quad \Rightarrow \quad n[P] \longrightarrow n[Q]$	(Red Amb)
$P \longrightarrow Q \quad \Rightarrow \quad P \mid R \longrightarrow Q \mid R$	(Red Par)
$P' \equiv P, P \longrightarrow Q, Q \equiv Q' \ \Rightarrow \ P' \longrightarrow Q'$	(Red $\equiv$)
$\longrightarrow^*$	refl-tran closure of $\longrightarrow$

Why a Logic?

A recurring issue for us was how to state behavioral properties of ambients. E.g., protocol specifications.

We have formal tools for establishing equational properties. But many properties cannot easily be formulated as equations.

For example, type systems for ambients guarantee certain properties, such as that some ambients are immobile, some are persistent. It’s hard to write down equations for immobility and persistence!

Our solution: use a (modal) logic tailored for ambients.

Modal Formulas

In a modal logic, the truth of a formula is relative to a state (world).

In our case, the truth of a *space-time* modal formula is relative to the *here and now* of a process. Each formula talks about the current time (before further evolution of the process) and the current place (the top-level of the process).

Therefore, the formula $n[\mathbf{0}]$ is read:

> there is *here and now* an empty location called n

The operator $n[\mathcal{A}]$ is a *single step in space* (akin to the temporal *next*), which allows us talk about that place one step down into n.

Other modal operators can be used to talk about undetermined times (in the future) and undetermined places (in the location tree).

Logical Formulas

$\mathcal{A}, \mathcal{B} : \mathbf{\Phi} ::=$

$\mathbf{T}$	true
$\neg\mathcal{A}$	negation (also $\mathcal{A}^{\neg}$)
$\mathcal{A} \vee \mathcal{B}$	disjunction
$\mathbf{0}$	void
$\eta[\mathcal{A}]$	location
$\mathcal{A} \mid \mathcal{B}$	composition
$\forall x.\mathcal{A}$	universal quantification over names
$\Diamond\mathcal{A}$	sometime modality (temporal)
✧$\mathcal{A}$	somewhere modality (spatial)
$\mathcal{A}@\eta$	location adjunct
$\mathcal{A} \triangleright \mathcal{B}$	composition adjunct

where η is a name n or a (quantifiable) variable x.

Satisfaction Relation

$$
\begin{array}{lcl}
P \vDash \mathbf{T} & & \\
P \vDash \neg\mathcal{A} & \triangleq & \neg\, P \vDash \mathcal{A} \\
P \vDash \mathcal{A}\vee\mathcal{B} & \triangleq & P \vDash \mathcal{A} \vee P \vDash \mathcal{B} \\
P \vDash \mathbf{0} & \triangleq & P \equiv \mathbf{0} \\
P \vDash n[\mathcal{A}] & \triangleq & \exists P' {:} \Pi.\ P \equiv n[P'] \wedge P' \vDash \mathcal{A} \\
P \vDash \mathcal{A} \mid \mathcal{B} & \triangleq & \exists P', P'' {:} \Pi.\ P \equiv P' | P'' \wedge P' \vDash \mathcal{A} \wedge P'' \vDash \mathcal{B} \\
P \vDash \forall x.\mathcal{A} & \triangleq & \forall m {:} \Lambda.\ P \vDash \mathcal{A}\{x \leftarrow m\} \\
P \vDash \Diamond\mathcal{A} & \triangleq & \exists P' {:} \Pi.\ P \rightarrow^{*} P' \wedge P' \vDash \mathcal{A} \\
P \vDash \diamondsuit\mathcal{A} & \triangleq & \exists P' {:} \Pi.\ P \downarrow^{*} P' \wedge P' \vDash \mathcal{A} \\
P \vDash \mathcal{A}@n & \triangleq & n[P] \vDash \mathcal{A} \\
P \vDash \mathcal{A} \triangleright \mathcal{B} & \triangleq & \forall P' {:} \Pi.\ P' \vDash \mathcal{A} \Rightarrow P|P' \vDash \mathcal{B}
\end{array}
$$

$P \downarrow P'$ iff $\exists n, P''.\ P \equiv n[P'] \mid P''$

$\downarrow^{*}$ is the reflexive and transitive closure of $\downarrow$

Claims

- The satisfaction relation is "utterly natural" (to us):
  - The definitions of $\mathbf{0}$, $\mathcal{A}|\mathcal{B}$, and $n[\mathcal{A}]$ seem inevitable, once we accept that formulas should be able to talk about the tree structure of locations, and that they should not distinguish processes that are surely indistinguishable (up to $\equiv$).
  - The connectives $\mathcal{A}@n$ and $\mathcal{A}\triangleright\mathcal{B}$ have security motivations.
  - The modalities $\Diamond\mathcal{A}$ and $✧\mathcal{A}$ talk about process evolution and structure in an undetermined way (good for specs).
  - The fragment $\mathbf{T}$, $\neg\mathcal{A}$, $\mathcal{A}\vee\mathcal{B}$, $\forall x.\mathcal{A}$, is classical: why not?
- The logic is induced by the satisfaction relation.
  - We did not have any preconceptions about what kind of logic this ought to be. We didn’t invent this logic, we discovered it!

Some Derived Formulas

$\mathbf{F}$	$\triangleq \neg\mathbf{T}$	
$\mathcal{A} \Rightarrow \mathcal{B}$	$\triangleq \neg\mathcal{A} \vee \mathcal{B}$	$P \vDash$ - iff $P \vDash \mathcal{A} \Rightarrow P \vDash \mathcal{B}$
$\mathcal{A} \wedge \mathcal{B}$	$\triangleq \neg(\neg\mathcal{A} \vee \neg\mathcal{B})$	$P \vDash$ - iff $P \vDash \mathcal{A} \wedge P \vDash \mathcal{B}$
$\exists x.\mathcal{A}$	$\triangleq \neg\forall x.\neg\mathcal{A}$	$P \vDash$ - iff $\exists m{:}\Lambda.\ P \vDash \mathcal{A}\{x\leftarrow m\}$
$\text{¤}\mathcal{A}$	$\triangleq \neg\text{✧}\neg\mathcal{A}$	$P \vDash$ - iff $\forall P'{:}\Pi.\ P\downarrow^{*}P' \Rightarrow P' \vDash \mathcal{A}$
$\Box\mathcal{A}$	$\triangleq \neg\Diamond\neg\mathcal{A}$	$P \vDash$ - iff $\forall P'{:}\Pi.\ P\rightarrow^{*}P' \Rightarrow P' \vDash \mathcal{A}$
$\mathcal{A}^{\mathbf{F}}$	$\triangleq \mathcal{A}\triangleright\mathbf{F}$	$P \vDash$ - iff $\forall P'{:}\Pi.\ P' \vDash \mathcal{A} \Rightarrow P \mid P' \vDash \mathbf{F}$
		iff $\forall P'{:}\Pi.\ \neg\ P' \vDash \mathcal{A}$
$\mathcal{A}^{\neg\mathbf{F}}$	$\mathcal{A}$ *valid*	$P \vDash$ - iff $\forall P'{:}\Pi.\ P' \vDash \mathcal{A}$
$\mathcal{A}^{\mathbf{F}\neg}$	$\mathcal{A}$ *satisfiable*	$P \vDash$ - iff $\exists P'{:}\Pi.\ P' \vDash \mathcal{A}$

Basic Fact

Satisfaction is invariant under structural congruence:

$$(P \vDash \mathcal{A} \wedge P \equiv P') \Rightarrow P' \vDash \mathcal{A}$$

I.e.: $\{P{:}\Pi \mid P \vDash \mathcal{A}\}$ is closed under $\equiv$.

Hence, formulas describe only congruence-invariant properties.

Simple Examples

(1) $p[\mathbf{T}] \mid \mathbf{T}$

there is a *p* here (and possibly something else)

(2) $\diamond(1)$

somewhere there is a *p*

(3) $(2) \Rightarrow \Box(2)$

if there is a *p* somewhere, then forever there is a *p* somewhere

(4) $p[q[\mathbf{T}] \mid \mathbf{T}] \mid \mathbf{T}$

there is a *p* with a child *q* here

(5) $\diamond(4)$

somewhere there is a *p* with a child *q*

From Satisfaction to Logic

Propositional validity

$\textbf{\textit{vld}}\ \mathcal{A} \quad \triangleq \quad \forall P{:}\Pi.\ P \vDash \mathcal{A} \qquad\qquad \mathcal{A}$ (closed) is valid

Sequents

$\mathcal{A} \vdash \mathcal{B} \quad \triangleq \quad \textbf{\textit{vld}}\ (\mathcal{A} \Rightarrow \mathcal{B})$

Rules

$\mathcal{A}_1 \vdash \mathcal{B}_1; \ldots; \mathcal{A}_n \vdash \mathcal{B}_n \;\}\; \mathcal{A} \vdash \mathcal{B} \quad \triangleq \qquad\qquad (n \geq 0)$

$\mathcal{A}_1 \vdash \mathcal{B}_1 \wedge \ldots \wedge \mathcal{A}_n \vdash \mathcal{B}_n \Rightarrow \mathcal{A} \vdash \mathcal{B}$

N.B.: All the rules shown later are validated accordingly.

Conventions:

$\dashv\vdash$ means $\vdash$ in both directions

$\{\}$ means $\}$ in both directions

"Neutral" Sequents

- The logic is formulated as a sequent calculus with single-premise, single-conclusion sequents. We don't pre-judge ",".
  - By taking $\wedge$ on the left and $\vee$ on the right of $\vdash$ as structural operators, all the standard rules of sequent and natural deduction systems with multiple premises/conclusions can be derived.
  - By taking $|$ on the left of $\vdash$ as a structural operator, all the rules of intuitionistic linear logic can be derived (by appropriate mappings of the ILL connectives).
  - By taking nestings of $\wedge$ and $|$ on the left of $\vdash$ as structural "bunches", we obtain a bunched logic, with its two associated implications, $\Rightarrow$ and $\triangleright$.
- This is convenient. We do not know much, however, about the metatheory of this presentation style.

Step 1: Propositional Rules

(A-L)	$\mathcal{A}\wedge(\mathcal{C}\wedge\mathcal{D}) \vdash \mathcal{B} \;\}\}\; (\mathcal{A}\wedge\mathcal{C})\wedge\mathcal{D} \vdash \mathcal{B}$
(A-R)	$\mathcal{A} \vdash (\mathcal{C}\vee\mathcal{D})\vee\mathcal{B} \;\}\}\; \mathcal{A} \vdash \mathcal{C}\vee(\mathcal{D}\vee\mathcal{B})$
(X-L)	$\mathcal{A}\wedge\mathcal{C} \vdash \mathcal{B} \;\}\; \mathcal{C}\wedge\mathcal{A} \vdash \mathcal{B}$
(X-R)	$\mathcal{A} \vdash \mathcal{C}\vee\mathcal{B} \;\}\; \mathcal{A} \vdash \mathcal{B}\vee\mathcal{C}$
(C-L)	$\mathcal{A}\wedge\mathcal{A} \vdash \mathcal{B} \;\}\; \mathcal{A} \vdash \mathcal{B}$
(C-R)	$\mathcal{A} \vdash \mathcal{B}\vee\mathcal{B} \;\}\; \mathcal{A} \vdash \mathcal{B}$
(W-L)	$\mathcal{A} \vdash \mathcal{B} \;\}\; \mathcal{A}\wedge\mathcal{C} \vdash \mathcal{B}$
(W-R)	$\mathcal{A} \vdash \mathcal{B} \;\}\; \mathcal{A} \vdash \mathcal{C}\vee\mathcal{B}$
(Id)	$\}\; \mathcal{A} \vdash \mathcal{A}$
(Cut)	$\mathcal{A} \vdash \mathcal{C}\vee\mathcal{B};\; \mathcal{A}'\wedge\mathcal{C} \vdash \mathcal{B}' \;\}\; \mathcal{A}\wedge\mathcal{A}' \vdash \mathcal{B}\vee\mathcal{B}'$
(T)	$\mathcal{A}\wedge\mathbf{T} \vdash \mathcal{B} \;\}\; \mathcal{A} \vdash \mathcal{B}$
(F)	$\mathcal{A} \vdash \mathbf{F}\vee\mathcal{B} \;\}\; \mathcal{A} \vdash \mathcal{B}$
($\neg$-L)	$\mathcal{A} \vdash \mathcal{C}\vee\mathcal{B} \;\}\; \mathcal{A}\wedge\neg\mathcal{C} \vdash \mathcal{B}$
($\neg$-R)	$\mathcal{A}\wedge\mathcal{C} \vdash \mathcal{B} \;\}\; \mathcal{A} \vdash \neg\mathcal{C}\vee\mathcal{B}$

Step 2: Concurrency Rules

- Apart from our interest in mobility and nested locations, a fragment of our logic makes sense just for ordinary concurrency (i.e., for a CCS-like process calculus with $\mathbf{0}$ and $|$). We examine this fragment first.
- (Small caveat. To get things off the ground, one needs some process that is definitely $\neg\mathbf{0}$. In our full logic, locations have this property, otherwise something must be introduced for this purpose.)

Concurrency Rules

$(\mid \mathbf{0})$	$\} \ \mathcal{A} \mid \mathbf{0} \dashv\vdash \mathcal{A}$	**0** is nothing
$(\mid \neg\mathbf{0})$	$\} \ \mathcal{A} \mid \neg\mathbf{0} \vdash \neg\mathbf{0}$	if a part is non-**0**, so is the whole
$(\mathrm{A} \mid)$	$\} \ \mathcal{A} \mid (\mathcal{B} \mid \mathcal{C}) \dashv\vdash (\mathcal{A} \mid \mathcal{B}) \mid \mathcal{C}$	| associativity
$(\mathrm{X} \mid)$	$\} \ \mathcal{A} \mid \mathcal{B} \vdash \mathcal{B} \mid \mathcal{A}$	| commutativity
$(\mid \vdash)$	$\mathcal{A}' \vdash \mathcal{B}';\ \mathcal{A}'' \vdash \mathcal{B}'' \ \} \ \mathcal{A}' \mid \mathcal{A}'' \vdash \mathcal{B}' \mid \mathcal{B}''$	| congruence
$(\mid \vee)$	$\} \ (\mathcal{A} \vee \mathcal{B}) \mid \mathcal{C} \vdash \mathcal{A} \mid \mathcal{C} \vee \mathcal{B} \mid \mathcal{C}$	|-∨ distribution
$(\mid \Vert)$	$\} \ \mathcal{A}' \mid \mathcal{A}'' \vdash \mathcal{A}' \mid \mathcal{B}'' \vee \mathcal{B}' \mid \mathcal{A}'' \vee \neg\mathcal{B}' \mid \neg\mathcal{B}''$	decomposition
$(\mid \triangleright)$	$\mathcal{A} \mid \mathcal{C} \vdash \mathcal{B} \ \{\} \ \mathcal{A} \vdash \mathcal{C} \triangleright \mathcal{B}$	|-▷ adjunction

N.B., neutral sequents make the rule $(\mid \vdash)$ (and others) particularly simple, even though $\mid$ does not distribute with $\wedge$ in the "useful" direction.

The Decomposition Operator

Consider the De Morgan dual of $|$:

$\mathcal{A} \,\|\, \mathcal{B}$	$\triangleq \neg(\neg\mathcal{A} \mid \neg\mathcal{B})$	$P \vDash$ - iff $\forall P', P'':\Pi.\ P \equiv P'\|P'' \Rightarrow P' \vDash \mathcal{A} \vee P'' \vDash \mathcal{B}$
$\mathcal{A}^{\forall}$	$\triangleq \mathcal{A} \,\|\, \mathbf{F}$	$P \vDash$ - iff $\forall P', P'':\Pi.\ P \equiv P'\|P'' \Rightarrow P' \vDash \mathcal{A}$
$\mathcal{A}^{\exists}$	$\triangleq \mathcal{A} \mid \mathbf{T}$	$P \vDash$ - iff $\exists P', P'':\Pi.\ P \equiv P'\|P'' \wedge P' \vDash \mathcal{A}$

$\mathcal{A} \,\|\, \mathcal{B}$		for every partition, one piece satisfies $\mathcal{A}$ or the other piece satisfies $\mathcal{B}$
$\mathcal{A}^{\forall}$	$\Leftrightarrow \neg((\neg\mathcal{A})^{\exists})$	every component satisfies $\mathcal{A}$
$\mathcal{A}^{\exists}$	$\Leftrightarrow \neg((\neg\mathcal{A})^{\forall})$	some component satisfies $\mathcal{A}$

Examples:

$(p[\mathbf{T}] \Rightarrow p[q[\mathbf{T}]^{\exists}])^{\forall}$	every p has a q child
$(p[\mathbf{T}] \Rightarrow p[q[\mathbf{T}] \mid (\neg q[\mathbf{T}])^{\forall}])^{\forall}$	every p has a unique q child

The Decomposition Axiom

$(\mid \|)$ — $(\mathcal{A}' \mid \mathcal{A}'') \vdash (\mathcal{A}' \mid \mathcal{B}'') \vee (\mathcal{B}' \mid \mathcal{A}'') \vee (\neg\mathcal{B}' \mid \neg\mathcal{B}'')$

Alternative formulations and special cases:

- $(\mathcal{A}' \mid \mathcal{A}'') \wedge (\mathcal{B}' \| \mathcal{B}'') \vdash (\mathcal{A}' \mid \mathcal{B}'') \vee (\mathcal{B}' \mid \mathcal{A}'')$

  "If P has a partition into pieces that satisfy $\mathcal{A}'$ and $\mathcal{A}''$, and every partition has one piece that satisfies $\mathcal{B}'$ or the other that satisfies $\mathcal{B}''$, then either P has a partition into pieces that satisfy $\mathcal{A}'$ and $\mathcal{B}''$, or it has a partition into pieces that satisfy $\mathcal{B}'$ and $\mathcal{A}''$."

- $\neg(\mathcal{A} \mid \mathcal{B}) \vdash (\mathcal{A} \mid \mathbf{T}) \Rightarrow (\mathbf{T} \mid \neg\mathcal{B})$

  "If P has no partition into pieces that satisfy $\mathcal{A}$ and $\mathcal{B}$, but P has a piece that satisfies $\mathcal{A}$, then P has a piece that does not satisfy $\mathcal{B}$."

- $\neg(\mathbf{T} \mid \mathcal{B}) \vdash \mathbf{T} \mid \neg\mathcal{B}$

The Composition Adjunct

$(\mid\triangleright)$ $\quad \mathcal{A} \mid \mathcal{C} \vdash \mathcal{B} \;\Updownarrow\; \mathcal{A} \vdash \mathcal{C} \triangleright \mathcal{B}$

"Assume that every process that has a partition into pieces that satisfy $\mathcal{A}$ and $\mathcal{C}$, also satisfies $\mathcal{B}$. Then, every process that satisfies $\mathcal{A}$, together with any process that satisfies $\mathcal{C}$, satisfies $\mathcal{B}$. (And vice versa.)" (*c.f.* ($\multimap$ R))

Interpretations of $\mathcal{A} \triangleright \mathcal{B}$:

- P provides $\mathcal{B}$ in any context that provides $\mathcal{A}$
- P ensures $\mathcal{B}$ under any attack that ensures $\mathcal{A}$

That is, $P \vDash \mathcal{A} \triangleright \mathcal{B}$ is a context-system spec (a concurrent version of a pre-post spec).

Moreover $\mathcal{A} \triangleright \mathcal{B}$ is, in a precise sense, linear implication: the context that satisfies $\mathcal{A}$ is used exactly once in the system that satisfies $\mathcal{B}$.

Some Derived Rules

$\} \; (\mathcal{A} \triangleright \mathcal{B}) \mid \mathcal{A} \vdash \mathcal{B}$

"If P provides $\mathcal{B}$ in any context that provides $\mathcal{A}$, and Q provides $\mathcal{A}$, then P and Q together provide $\mathcal{B}$."

Proof: $\mathcal{A} \triangleright \mathcal{B} \vdash \mathcal{A} \triangleright \mathcal{B} \; \} \; (\mathcal{A} \triangleright \mathcal{B}) \mid \mathcal{A} \vdash \mathcal{B}$ by (Id), (| $\triangleright$)

$\mathcal{D} \vdash \mathcal{A}; \; \mathcal{B} \vdash \mathcal{C} \; \} \; \mathcal{D} \mid (\mathcal{A} \triangleright \mathcal{B}) \vdash \mathcal{C}$ (*c.f.* ($\multimap$ L))

"If anything that satisfies $\mathcal{D}$ satisfies $\mathcal{A}$, and anything that satisfies $\mathcal{B}$ satisfies $\mathcal{C}$, then: anything that has a partition into a piece satisfying $\mathcal{D}$ (and hence $\mathcal{A}$), and another piece satisfying $\mathcal{B}$ in a context that satisfies $\mathcal{A}$, it satisfies ($\mathcal{B}$ and hence) $\mathcal{C}$."

Proof:

$\mathcal{D} \vdash \mathcal{A}; \; \mathcal{A} \triangleright \mathcal{B} \vdash \mathcal{A} \triangleright \mathcal{B} \; \} \; \mathcal{D} \mid \mathcal{A} \triangleright \mathcal{B} \vdash \mathcal{A} \mid \mathcal{A} \triangleright \mathcal{B}$ assumption, (Id), (| $\vdash$)

$\mathcal{A} \mid \mathcal{A} \triangleright \mathcal{B} \vdash \mathcal{B}$ above

$\mathcal{B} \vdash \mathcal{C}$ assumption

More Derived Rules

$\} \; \mathcal{A} \vdash \mathbf{T} \mid \mathcal{A}$	you can always add more pieces (if they are **0**)
$\} \; \mathbf{F} \mid \mathcal{A} \vdash \mathbf{F}$	if a piece is absurd, so is the whole
$\} \; \mathbf{0} \vdash \neg(\neg\mathbf{0} \mid \neg\mathbf{0})$	**0** is single-threaded
$\} \; \mathcal{A} \mid \mathcal{B} \wedge \mathbf{0} \vdash \mathcal{A}$	you can split **0** (but you get **0**). Proof uses (| ||)
$\mathcal{A}' \vdash \mathcal{A};\; \mathcal{B} \vdash \mathcal{B}' \;\; \} \; \mathcal{A} \triangleright \mathcal{B} \vdash \mathcal{A}' \triangleright \mathcal{B}'$	$\triangleright$ is contravariant on the left
$\} \; \mathcal{A} \triangleright \mathcal{B} \mid \mathcal{B} \triangleright \mathcal{C} \vdash \mathcal{A} \triangleright \mathcal{C}$	$\triangleright$ is transitive
$\} \; (\mathcal{A} \mid \mathcal{B}) \triangleright \mathcal{C} \dashv\vdash \mathcal{A} \triangleright (\mathcal{B} \triangleright \mathcal{C})$	$\triangleright$ curry/uncurry
$\} \; \mathcal{A} \triangleright (\mathcal{B} \triangleright \mathcal{C}) \vdash \mathcal{B} \triangleright (\mathcal{A} \triangleright \mathcal{C})$	contexts commute
$\} \; \mathbf{T} \dashv\vdash \mathbf{T} \triangleright \mathbf{T}$	truth can withstand any attack
$\} \; \mathbf{T} \vdash \mathbf{F} \triangleright \mathcal{A}$	anything goes if you can find an absurd partner
$\} \; \mathbf{T} \triangleright \mathcal{A} \vdash \mathcal{A}$	if $\mathcal{A}$ resists any attack, then it holds

Step 3: Location Rules

$(n[]\ \neg\mathbf{0})$	$\} \ n[\mathcal{A}] \vdash \neg\mathbf{0}$	locations exist
$(n[]\ \neg \mid)$	$\} \ n[\mathcal{A}] \vdash \neg(\neg\mathbf{0} \mid \neg\mathbf{0})$	are not decomposable
$(n[]\ \vdash)$	$\mathcal{A} \vdash \mathcal{B} \ \{\} \ n[\mathcal{A}] \vdash n[\mathcal{B}]$	$n[]$ congruence
$(n[]\ \wedge)$	$\} \ n[\mathcal{A}] \wedge n[\mathcal{C}] \vdash n[\mathcal{A} \wedge \mathcal{C}]$	$n[]$-$\wedge$ distribution
$(n[]\ \vee)$	$\} \ n[\mathcal{C} \vee \mathcal{B}] \vdash n[\mathcal{C}] \vee n[\mathcal{B}]$	$n[]$-$\vee$ distribution
$(n[]\ \mathbf{F})$	$\} \ n[\mathbf{F}] \vdash \mathbf{F}$	can’t hold absurdity
$(n[]\ @)$	$n[\mathcal{A}] \vdash \mathcal{B} \ \{\} \ \mathcal{A} \vdash \mathcal{B}@n$	$n[]$-@ adjunction

Some Derived Rules

Consequences:

$\mathcal{A} \vdash \mathcal{B} \;\; \} \;\; \mathcal{A}@n \vdash \mathcal{B}@n$ @ congruence

$\} \;\; n[\mathcal{A}@n] \vdash \mathcal{A}$

$\} \;\; \mathcal{A} \dashv\vdash n[\mathcal{A}]@n$

$\} \;\; n[\neg\mathcal{A}] \vdash \neg n[\mathcal{A}]$

$\} \;\; \neg n[\mathcal{A}] \dashv\vdash \neg n[\mathbf{T}] \vee n[\neg\mathcal{A}]$

Examples

$\mathit{an}\ n \triangleq n[\mathbf{T}] \mid \mathbf{T}$	there is now an *n* here
$\mathit{no}\ n \triangleq \neg \mathit{an}\ n$	there is now no *n* here
$\mathit{one}\ n \triangleq n[\mathbf{T}] \mid \mathit{no}\ n$	there is now exactly one *n* here
$\mathcal{A}^{\forall} \triangleq \neg(\neg\mathcal{A} \mid \mathbf{T})$	everybody here satisfies $\mathcal{A}$
$(n[\mathbf{T}] \Rightarrow n[\mathcal{A}])^{\forall}$	every *n* here satisfies $\mathcal{A}$
$\Box((n[\mathbf{T}] \Rightarrow n[\mathcal{A}])^{\forall})$	every *n* everywhere satisfies $\mathcal{A}$

Step 4: Time and Space Modalities

(◇) $\rbrace$ $\Diamond\mathcal{A} \dashv\vdash \neg\Box\neg\mathcal{A}$

(□ K) $\rbrace$ $\Box(\mathcal{A} \Rightarrow \mathcal{B}) \vdash \Box\mathcal{A} \Rightarrow \Box\mathcal{B}$

(□ T) $\rbrace$ $\Box\mathcal{A} \vdash \mathcal{A}$

(□ 4) $\rbrace$ $\Box\mathcal{A} \vdash \Box\Box\mathcal{A}$

(□ **T**) $\rbrace$ $\mathbf{T} \vdash \Box\mathbf{T}$

(□ ⊢) $\mathcal{A} \vdash \mathcal{B}$ $\rbrace$ $\Box\mathcal{A} \vdash \Box\mathcal{B}$

(✧) $\rbrace$ ✧$\mathcal{A} \dashv\vdash \neg$ ¤ $\neg\mathcal{A}$

(¤ K) $\rbrace$ ¤$(\mathcal{A} \Rightarrow \mathcal{B}) \vdash$ ¤$\mathcal{A} \Rightarrow$ ¤$\mathcal{B}$

(¤ T) $\rbrace$ ¤$\mathcal{A} \vdash \mathcal{A}$

(¤ 4) $\rbrace$ ¤$\mathcal{A} \vdash$ ¤ ¤$\mathcal{A}$

(¤ **T**) $\rbrace$ $\mathbf{T} \vdash$ ¤$\mathbf{T}$

(¤ ⊢) $\mathcal{A} \vdash \mathcal{B}$ $\rbrace$ ¤$\mathcal{A} \vdash$ ¤$\mathcal{B}$

S4, but not S5: $\neg$ ***vld*** $\Diamond\mathcal{A} \vdash \Box\Diamond\mathcal{A}$ $\neg$ ***vld*** ✧$\mathcal{A} \vdash$ ¤ ✧$\mathcal{A}$

Additional Modality Rules

($\lozenge$ n[])	} $n[\lozenge\mathcal{A}] \vdash \lozenge n[\mathcal{A}]$
($\lozenge$ |)	} $(\lozenge\mathcal{A}) \mid (\lozenge\mathcal{B}) \vdash \lozenge(\mathcal{A} \mid \mathcal{B})$
(✧ n[])	} $n[$✧$\mathcal{A}] \vdash$ ✧$\mathcal{A}$
(✧ |)	} (✧$\mathcal{A}$) $\mid \mathcal{B} \vdash$ ✧$(\mathcal{A} \mid \mathbf{T})$
(✧$\lozenge$)	} ✧$\lozenge\mathcal{A} \vdash \lozenge$✧$\mathcal{A}$
	if somewhere sometime $\mathcal{A}$, then sometime somewhere $\mathcal{A}$

Step 5: Validity and Satisfiability

$P \vDash \mathcal{A}^{\mathbf{F}}$ iff $\forall P':\Pi.\ P' \vDash \mathcal{A} \Rightarrow P \mid P' \vDash \mathbf{F}$

iff $\forall P':\Pi.\ \neg P' \vDash \mathcal{A}$ iff $\mathcal{A}$ is unsatisfiable

$(\triangleright\mathbf{F}\ \neg)$ $\quad \vdash \mathcal{A}^{\mathbf{F}} \vdash \mathcal{A}^{\neg}$ $\quad$ if $\mathcal{A}$ is unsatisfiable then $\mathcal{A}$ is false

$(\neg\ \triangleright\mathbf{F})$ $\quad \vdash \mathcal{A}^{\mathbf{F}\neg} \vdash \mathcal{A}^{\mathbf{FF}}$ $\quad$ if $\mathcal{A}$ is satisfiable then $\mathcal{A}^{\mathbf{F}}$ is unsatisfiable

We can reflect validity and satisfiability within the logic:

$Vld\,\mathcal{A} \triangleq \mathcal{A}^{\neg\mathbf{F}}$ $\qquad P \vDash Vld\,\mathcal{A}$ iff $\forall P':\Pi.\ P' \vDash \mathcal{A}$

$Sat\,\mathcal{A} \triangleq \mathcal{A}^{\mathbf{F}\neg}$ $\qquad P \vDash Sat\,\mathcal{A}$ iff $\exists P':\Pi.\ P' \vDash \mathcal{A}$

Then, as derived rules we have that *Vld*,*Sat* are S5 modalities.
(That is, S4 plus: $\vdash Sat\,\mathcal{A} \vdash Vld\ Sat\,\mathcal{A}$.)

Reflecting Name Equality

Name equality can be defined within the logic:

$$\eta = \mu \quad \triangleq \quad \eta[\mathbf{T}]@\mu$$

Since (for any substitution applied to η,μ):

$P \vDash \eta[\mathbf{T}]@\mu$
iff $\mu[P] \vDash \eta[\mathbf{T}]$
iff $\eta = \mu \wedge P \vDash \mathbf{T}$
iff $\eta = \mu$

Example: "Any two ambients here have different names":

$$\forall x.\forall y.\ x[\mathbf{T}] \mid y[\mathbf{T}] \mid \mathbf{T} \Rightarrow \neg\ x{=}y$$

Thief!

A *shopper* is likely to pull out a wallet. A *thief* is likely to grab it.

$$\begin{aligned} \mathit{Shopper} \triangleq\ & \\ & \mathrm{Person}[\mathrm{Wallet}[£] \mid \mathbf{T}] \wedge \\ & \Diamond(\mathrm{Person}[\mathit{NoWallet}] \mid \mathrm{Wallet}[£]) \end{aligned}$$

$$\mathit{NoWallet} \triangleq \neg(\mathrm{Wallet}[£] \mid \mathbf{T})$$

$$\mathit{Thief} \triangleq \mathrm{Wallet}[£] \rhd \Diamond\mathit{NoWallet}$$

By simple logical deductions involving the laws of $\rhd$ and $\Diamond$:

$$\begin{aligned} \mathit{Shopper} \mid \mathit{Thief} \Rightarrow\ & \\ & (\mathrm{Person}[\mathrm{Wallet}[£] \mid \mathbf{T}] \mid \mathit{Thief}) \wedge \\ & \Diamond(\mathrm{Person}[\mathit{NoWallet}] \mid \mathit{NoWallet}) \end{aligned}$$

Syntactic Connections with Linear Logic

- Intuitionistic linear logic (ILL) can be embedded in our logic:

$\mathbf{1}_{\mathrm{ILL}}$	$\triangleq$	$\mathbf{0}$	$\mathcal{A} \oplus \mathcal{B}$	$\triangleq$	$\mathcal{A} \vee \mathcal{B}$
$\bot_{\mathrm{ILL}}$	$\triangleq$	$\mathbf{F}$	$\mathcal{A} \,\&\, \mathcal{B}$	$\triangleq$	$\mathcal{A} \wedge \mathcal{B}$
$\top_{\mathrm{ILL}}$	$\triangleq$	$\mathbf{T}$	$\mathcal{A} \otimes \mathcal{B}$	$\triangleq$	$\mathcal{A} \mid \mathcal{B}$
$\mathbf{0}_{\mathrm{ILL}}$	$\triangleq$	$\mathbf{F}$	$\mathcal{A} \multimap \mathcal{B}$	$\triangleq$	$\mathcal{A} \triangleright \mathcal{B}$
			$\mathbf{!}\mathcal{A}$	$\triangleq$	$\mathbf{0} \wedge (\mathbf{0} \Rightarrow \mathcal{A})^{\neg\mathbf{F}}$

- The rules of ILL can be logically derived from these definitions. (E.g.: the proof of $\mathbf{!}\mathcal{A} \vdash \mathbf{!}\mathcal{A} \otimes \mathbf{!}\mathcal{A}$ uses the decomposition axiom.)
- So, $\mathcal{A}_1, \ldots, \mathcal{A}_n \vdash_{\mathrm{ILL}} \mathcal{B}$ implies $\mathcal{A}_1 \mid \ldots \mid \mathcal{A}_n \vdash \mathcal{B}$.
- N.B. weakening/contraction are not valid (because $P \mid P \neq P$).
- But the additives, $\wedge$ and $\vee$, distribute (a derived rule).

Semantic Connections with Linear Logic

- A (commutative) quantale $\mathcal{Q}$ is a structure $<S : \text{Set}, \leq : S^2 \rightarrow \text{Bool}, \otimes : S^2 \rightarrow S, 1 : S, \bigvee : \mathcal{P}(S) \rightarrow S>$ such that:

  $\leq, \bigvee$: a complete join semilattice

  $\otimes, 1$: a commutative monoid

  $p \otimes \bigvee Q \;=\; \bigvee \{p \otimes q \mid q \in Q\}$

- They are complete models of Intuitionistic Linear Logic (ILL):

$$
\begin{array}{lcllcl}
[\![\mathcal{A} \oplus \mathcal{B}]\!] & \triangleq & \bigvee\{[\![\mathcal{A}]\!], [\![\mathcal{B}]\!]\} & [\![\mathbf{1}_{\text{ILL}}]\!] & \triangleq & 1 \\
[\![\mathcal{A} \,\&\, \mathcal{B}]\!] & \triangleq & \bigvee\{C \mid C \leq [\![\mathcal{A}]\!] \wedge C \leq [\![\mathcal{B}]\!]\} & [\![\bot_{\text{ILL}}]\!] & \triangleq & \text{any element of } S \\
[\![\mathcal{A} \otimes \mathcal{B}]\!] & \triangleq & [\![\mathcal{A}]\!] \otimes [\![\mathcal{B}]\!] & [\![\top_{\text{ILL}}]\!] & \triangleq & \bigvee S \\
[\![\mathcal{A} \multimap \mathcal{B}]\!] & \triangleq & \bigvee\{C \mid C \otimes [\![\mathcal{A}]\!] \leq [\![\mathcal{B}]\!]\} & [\![\mathbf{0}_{\text{ILL}}]\!] & \triangleq & \bigvee \emptyset
\end{array}
$$

$[\![!\mathcal{A}]\!] \;\triangleq\; \upsilon X.\, [\![\mathbf{1} \,\&\, \mathcal{A} \,\&\, X \otimes X]\!]$ where $\upsilon X.\, A\{X\} \triangleq \bigvee\{C \mid C \leq A\{C\}\}$

$$\mathbf{vld}_{\text{ILL}}(\mathcal{A}_1, \ldots, \mathcal{A}_n \vdash_{\text{ILL}} \mathcal{B})_{\mathcal{Q}} \;\triangleq\; [\![\mathcal{A}_1]\!]_{\mathcal{Q}} \otimes_{\mathcal{Q}} \ldots \otimes_{\mathcal{Q}} [\![\mathcal{A}_n]\!]_{\mathcal{Q}} \leq_{\mathcal{Q}} [\![\mathcal{B}]\!]_{\mathcal{Q}}$$

The Process Quantale

- The sets of processes closed under $\equiv$ and ordered by inclusion form a quantale (let $A^{\equiv} \triangleq \{P \,\|\, P \equiv Q \wedge Q \in A\}$):

$$\mathbf{\Phi} \triangleq <\mathbf{\Phi}, \subseteq, \otimes, \mathbf{1}, \bigcup> \qquad \text{where, for } A,B \subseteq \Pi\text{:}$$

$$\Phi \ \triangleq\ \{A^{\equiv} \,\|\, A \subseteq \Pi\}$$

$$1_{\Phi} \ \triangleq\ \{\mathbf{0}\}^{\equiv}, \qquad A \otimes_{\Phi} B \ \triangleq\ \{P \mid Q \,\|\, P \in A \wedge Q \in B\}^{\equiv}$$

- Our syntactic definitions of ILL operators match their quantale interpretation. (E.g.: $[\![\mathcal{A} \otimes \mathcal{B}]\!]_{\Phi} = [\![\mathcal{A}]\!]_{\Phi} \otimes_{\Phi} [\![\mathcal{B}]\!]_{\Phi}$, $[\![!\mathcal{A}]\!]_{\Phi} = !_{\Phi}[\![\mathcal{A}]\!]_{\Phi}$.)

- Interpretation of formulas:

$$[\![\mathcal{A}]\!] \ \triangleq\ \{P{:}\Pi \,\|\, P \vDash \mathcal{A}\} \qquad \text{where } [\![\mathcal{A}]\!] = [\![\mathcal{A}]\!]^{\equiv}$$

- Our validity matches ILL validity for ILL sequents:

$$\mathbf{vld}_{\mathrm{ILL}}(\mathcal{A}_1, \ldots, \mathcal{A}_n \vdash_{\mathrm{ILL}} \mathcal{B})_{\Phi} \iff \mathbf{vld}(\mathcal{A}_1 \mid \ldots \mid \mathcal{A}_n \vdash \mathcal{B})$$

Applications

- Model Checking
  - We have an algorithm for deciding the $\vDash$ relation for **!**-free processes and $\triangleright$-free formulas.
- Expressing Locking
  - If $E, n{:}Amb^{\bullet}[S] \vdash P : T$ (a typing judgment asserting that no ambient called n can ever be opened in P), then:

    $$P \vDash \Box(\Diamond an\ n \Rightarrow \Box\Diamond an\ n)$$
- Expressing Immobility
  - If $E, p{:}Amb^{\bullet}[S], q{:}Amb^{\bullet}[\underline{\curlyvee}S'] \vdash P : T$ (a typing judgment asserting that no ambient called q can ever move within P), then:

    $$P \vDash \Box(\Diamond(p\ parents\ q) \Rightarrow \Box\Diamond(p\ parents\ q))$$
    where $p\ parents\ q \ \triangleq\ p[q[\mathbf{T}] \mid \mathbf{T}] \mid \mathbf{T}$

Future Directions: Fixpoints

- Abadi, Lamport, and Plotkin and have described *reactive* specifications such that:

$$\mathcal{A} \rightarrow \mathcal{B} \mid \mathcal{B} \rightarrow \mathcal{A} \;\Rightarrow\; \mathcal{A} \wedge \mathcal{B}$$

  Define: $\mathcal{Y} \rightarrow \mathcal{Z} \;\triangleq\; \mu\mathcal{X}.\,(\mathcal{X} \triangleright \mathcal{Y}) \triangleright \mathcal{Z}$. Then:

$$\begin{aligned}
\mathcal{A} \rightarrow \mathcal{B} \;&=\; ((\mathcal{A} \rightarrow \mathcal{B}) \triangleright \mathcal{A}) \triangleright \mathcal{B} \;\Rightarrow\; (\mathcal{B} \rightarrow \mathcal{A}) \triangleright \mathcal{B} \\
\mathcal{B} \rightarrow \mathcal{A} \;&=\; ((\mathcal{B} \rightarrow \mathcal{A}) \triangleright \mathcal{B}) \triangleright \mathcal{A} \;\Rightarrow\; (\mathcal{A} \rightarrow \mathcal{B}) \triangleright \mathcal{A}
\end{aligned}$$

$$\begin{aligned}
\mathcal{A} \rightarrow \mathcal{B} \mid \mathcal{B} \rightarrow \mathcal{A} \;&\Rightarrow\; (\mathcal{B} \rightarrow \mathcal{A}) \triangleright \mathcal{B} \mid \mathcal{B} \rightarrow \mathcal{A} \;\Rightarrow\; \mathcal{B} \\
\mathcal{A} \rightarrow \mathcal{B} \mid \mathcal{B} \rightarrow \mathcal{A} \;&\Rightarrow\; \mathcal{A} \rightarrow \mathcal{B} \mid (\mathcal{A} \rightarrow \mathcal{B}) \triangleright \mathcal{A} \;\Rightarrow\; \mathcal{A}
\end{aligned}$$

- Modalities and their variations can be defined from fixpoints. Moreover, we can express new useful predicates:

$$\begin{aligned}
\cancel{n} \;&\triangleq\; \neg ✧(n[\mathbf{T}] \mid \mathbf{T}) \\
\mathit{unique}\ n \;&\triangleq\; \mu\mathcal{X}.\,\cancel{n} \mid (n[\cancel{n}] \vee \exists y{\neq}n.\; y[\mathcal{X}])
\end{aligned}$$

Conclusions

- The novel aspects of our logic lie in its treatment of *space* (spatial structures) and of the evolution of space over time (mobility).
- The logic has a strong intensional flavor, reflecting the fact that space has intensional properties. The logic has a linear flavor in the sense that space cannot be instantly created or deleted.
- These principles can be applied to any process calculus that embodies a distinction between topological and dynamic operators.
- The logic is based on strong computational intuitions. However, from a purely logical point of view, it seems to have unusual properties (perhaps accidental to our presentation style).