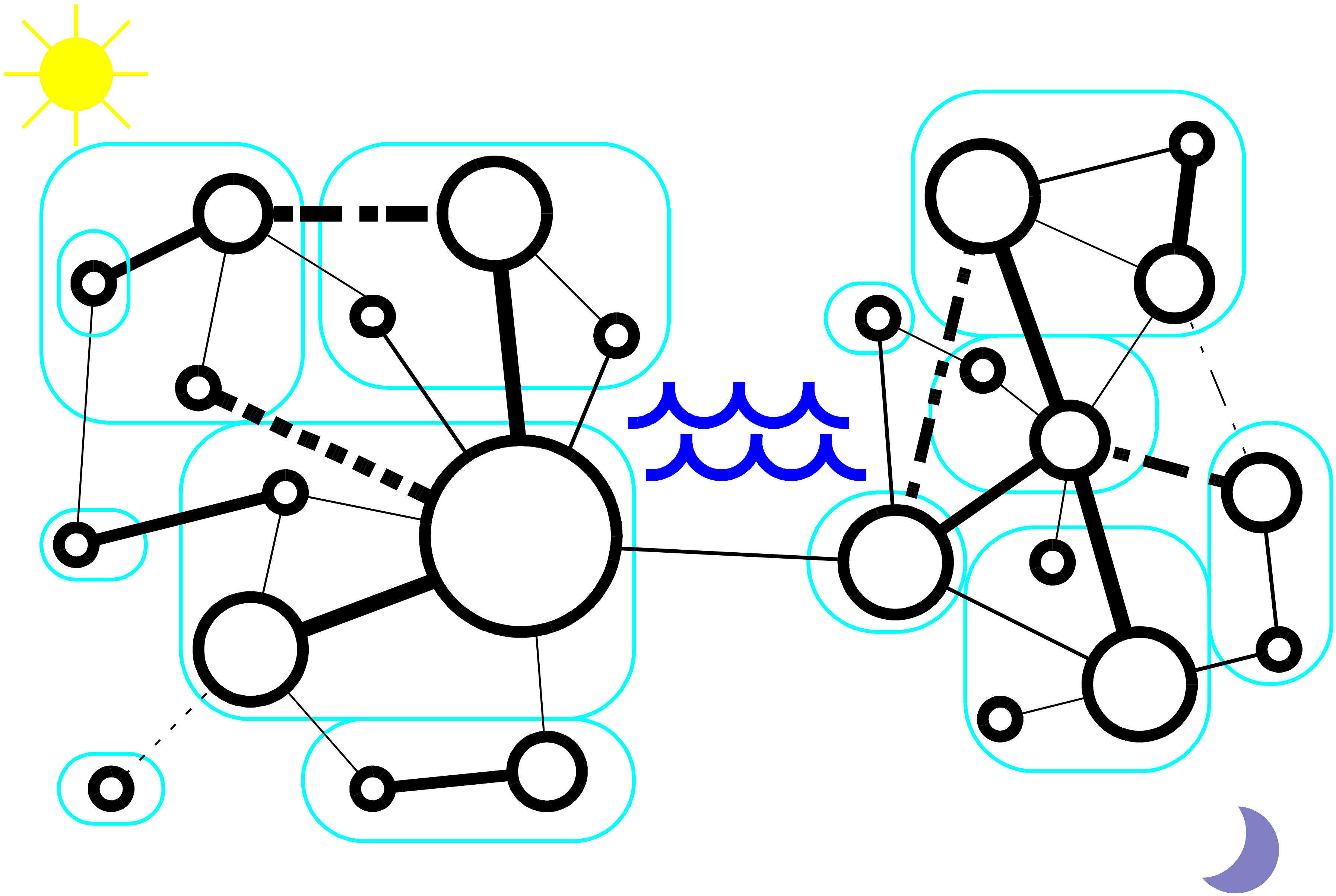
Mobile Ambients

Luca Cardelli
Andrew D. Gordon

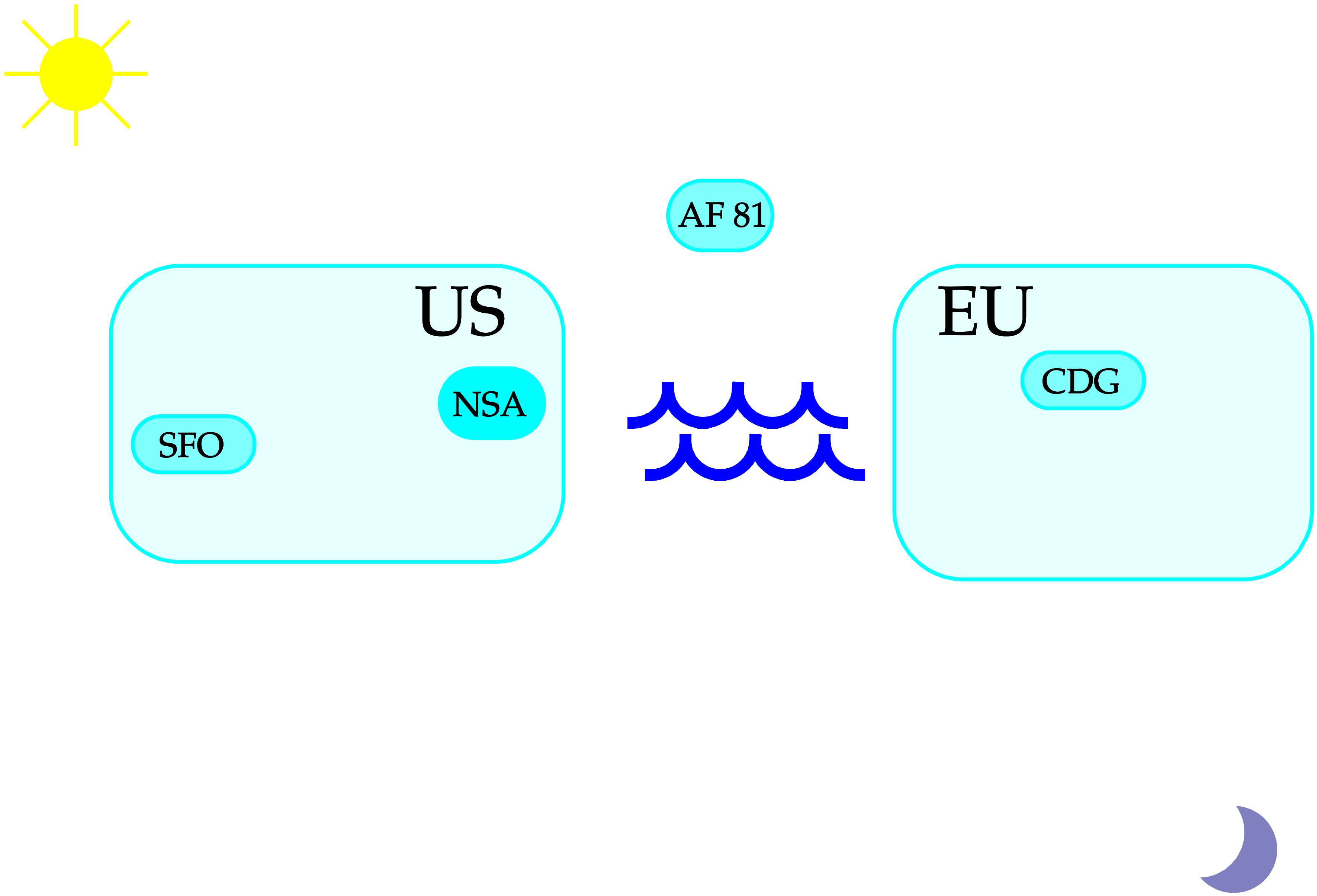
Microsoft Research

ETAPS'98

Virtuality



Reality



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).

Mobility Postulates

- Distinct locations exist.
- If different locations have different properties, then both people and programs 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.

Formalisms for Concurrency/Distribution

- In the π-like calculi (our starting point):
  - ~ processes exist in a single **contiguous** location; interaction **shared names**, used as I/O channels
  - ~ process mobility = channel passing
  - ~ locality (and location failures) are added
  - ~ 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
  - ~ process mobility = barrier crossing
  - ~ integrated locality = topology; failure = unreachability
  - ~ **capabilities**, derived from ambient names, regulate access

Ambients

- 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

$P ::=$	an activity
$(\nu n)\ P$	new name n in a scope
0	inactivity
$P \mid P$	parallel
$!P$	replication
$M[P]$	ambient ($M = n$ or x)
$M.\ P$	exercise a capability
$(x).\ P$	input locally, bind to x
$\langle M \rangle$	output locally (async)

scoping

standard in process calculi

data structures

ambient-specific

actions

ambient I/O

$M ::=$	a capability
n	name
$in\ a$	entry capability
$out\ a$	exit capability
$open\ a$	open capability
x	variable
$M.\ M'$	path

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, enabling reasoning.
  - ~ Standard process calculi proof techniques (context lemmas,

bisimulation, etc.) can be adapted.

Straight from the π-calculus

- Parallel execution, $P \mid Q$:

$$P \mid Q \equiv Q \mid P$$

$$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$$

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

- Replication, $!P$:

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

- Restriction, $(\nu n)P$:

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

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

- Inaction, 0:

$$P \mid 0 \equiv P$$

$$!0 \equiv 0$$

$$(\nu n)0 \equiv 0$$

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:

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

- 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.

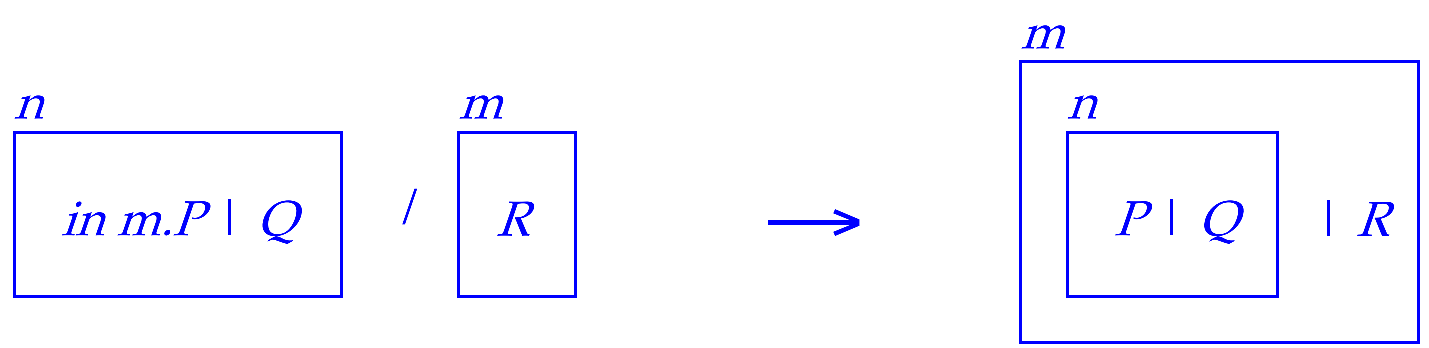
Entry Capability

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

$$in\ m.\ P$$

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

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



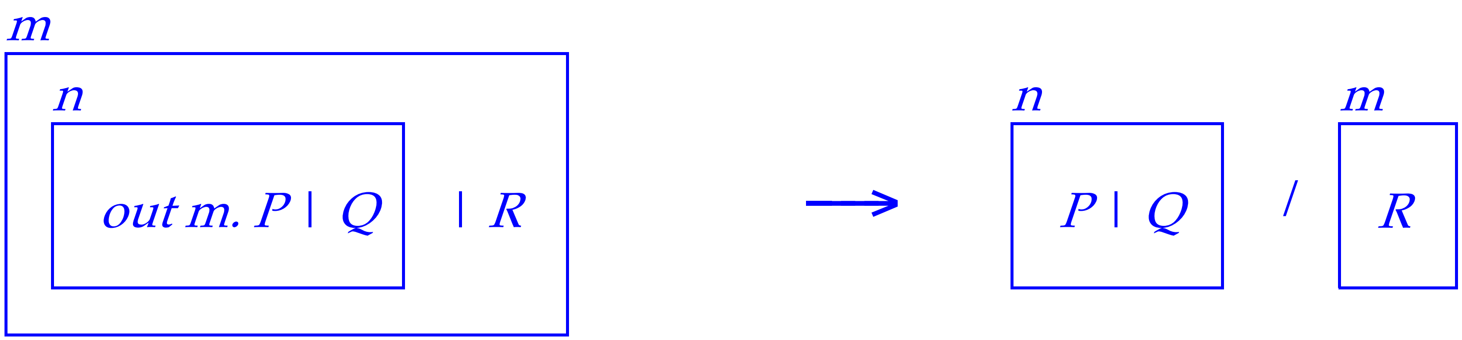
Exit Capability

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

$$out\ m.\ P$$

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

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



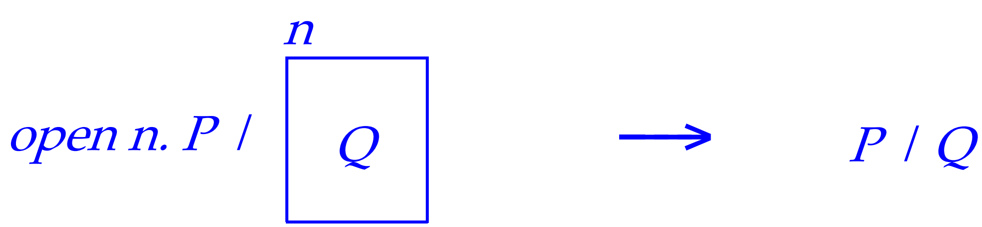
Open Capability

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

$$open\ n.\ P$$

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

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



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.

Structural Congruence Summary

$P \equiv P$	(Struct Refl)
$P \equiv Q \;\Rightarrow\; Q \equiv P$	(Struct Symm)
$P \equiv Q,\, Q \equiv R \;\Rightarrow\; P \equiv R$	(Struct Trans)
$P \equiv Q \;\Rightarrow\; (\nu n)P \equiv (\nu n)Q$	(Struct Res)
$P \equiv Q \;\Rightarrow\; P \mid R \equiv Q \mid R$	(Struct Par)
$P \equiv Q \;\Rightarrow\; !P \equiv !Q$	(Struct Repl)
$P \equiv Q \;\Rightarrow\; M[P] \equiv M[Q]$	(Struct Amb)
$P \equiv Q \;\Rightarrow\; M.P \equiv M.Q$	(Struct Action)
$P \mid Q \equiv Q \mid P$	(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)
$(\nu n)(P \mid Q) \equiv P \mid (\nu n)Q$ if $n \notin fn(P)$	(Struct Res Par)
$(\nu n)(m[P]) \equiv m[(\nu n)P]$ if $n \neq m$	(Struct Res Amb)
$P \mid 0 \equiv P$	(Struct Zero Par)
$(\nu n)0 \equiv 0$	(Struct Zero Res)
$!0 \equiv 0$	(Struct Zero Repl)
$P \equiv Q \;\Rightarrow\; (x).P \equiv (x).Q$	(Struct Input)
$\varepsilon.P \equiv P$	(Struct ε)
$(M.M').P \equiv M.M'.P$	(Struct .)

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

$$(\nu n)P = (\nu m)P\{n \leftarrow m\} \quad \text{if } m \notin fn(P)$$
$$(x).P = (y).P\{x \leftarrow y\} \quad \text{if } y \notin fv(P)$$

Noticeable Inequivalences

- Replication creates new names:

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

- Multiple n ambients have separate identity:

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

Expressiveness

- 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.)

- 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.

Locks

- We can use *open* to encode locks:

$$release\ n.\ P \triangleq n[\,]\ |\ P$$

$$acquire\ n.\ P \triangleq open\ n.\ P$$

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

$$acquire\ n.\ release\ m.\ P\ |\ release\ n.\ acquire\ m.\ Q$$

Turing Machines

$end[extendLft \mid S_0 \mid$

$square[S_1 \mid$

$square[S_2 \mid$

...

$square[S_i \mid head \mid$

...

$square[S_{n\text{-}1} \mid$

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

Mobile Agents

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

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

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

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

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

Firewalls

- Assume the keys k, k', k'' are shared.

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

$$Agent \triangleq k'[open\ k.\ k''[Q]]$$

$Agent \mid Firewall$

$\longrightarrow^* (\nu w)\ (\ k'[open\ k.\ k''[Q]] \mid k[in\ k'.\ in\ w] \mid w[open\ k'.\ open\ k''.$

$P]\,)$

$\rightarrow^* (\nu w)\,(\,k'[k[in\ w] \mid open\ k.\ k''[Q]] \mid w[open\ k'.\ open\ k''.\ P]\,)$

$\rightarrow^* (\nu w)\,(\,k'[in\ w \mid k''[Q]] \mid w[open\ k'.\ open\ k''.\ P]\,)$

$\rightarrow^* (\nu w)\quad w[k'[k''[Q]] \mid open\ k'.\ open\ k''.\ P]$

$\rightarrow^* (\nu w)\quad w[k''[Q] \mid open\ k''.\ P]$

$\rightarrow^* (\nu w)\quad w[Q \mid P]$

- Desired Property:

$$(\nu\ k\ k'\ k'')\,(Agent \mid Firewall) \;\simeq\; (\nu w)\ w[Q \mid P]$$

Contextual Equivalence

- Exhibition

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

- Convergence

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

- Contextual Equivalence

$$P \simeq Q \iff \forall C\{\!\{\bullet\}\!\}.\ \forall n.\ C\{\!\{P\}\!\}{\Downarrow}n \Leftrightarrow C\{\!\{Q\}\!\}{\Downarrow}n$$

- Ex.: the Perfect-Firewall Equation:

$$(\nu n)\ n[P] \simeq 0 \qquad \text{if } n \text{ not free in } P$$

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.

$$(ch\ n)P \triangleq (\nu n)\ (n[!open\ io] \mid P)$$
$$n(x).P \triangleq (\nu p)\ (io[in\ n.\ (x).\ p[out\ n.\ P]] \mid open\ p)$$
$$n\langle M\rangle \triangleq io[in\ n.\ \langle M\rangle]$$

- These definitions satisfy the expected reduction:

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

in presence of a channel for *n*.

Conclusions

- The notion of *named, active, hierarchical, mobile ambients* captures the structure of complex networks and of mobile computing/computation.
- The ambient calculus formalizes ambient notions simply and powerfully.
  - ~ It is no more complex than common process calculi.
  - ~ It supports reasoning about mobility and (hopefully) security.
- It provides a basis for envisioning new programming methodologies/libraries/languages for global computation.