Located Process Calculi an introduction

- Introduction
- Dπ: anonymous autonomous agents
- Nomadic Pict: named autonomous agents
- Seal Calculus: multi-level dependent agents
- Ambients: multi-level autonomous agents

Review:

- Typing
- Behavioural theories

Wide Area Networks

cf Cardelli: Wide Area Computation

- Virtual locations: administrative domains, resource control
- Physical locations: remote versus local rcps
- Bandwidth fluctuations: efficiency of connections vary
- Failures: site failure is unpredictable

New programming and analysis tools are required

The Computational Model underlying Dπ

locations/sites: where computations occur. Locations have a flat structure but may be generated dynamically.
In Dπ all communication is local.

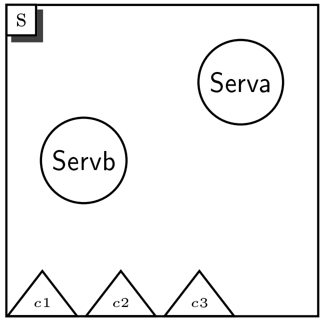
mobile agents: perform computations; move from site to site.
Described by augmented π-calculus terms.

resources: (communication channels) available at specific locations.
In Dπ names of resources only have local significance.

types:

guarantee controlled access to resources.
Agents should only use resources in accordance with capabilities/permissions they have acquired.

Dπ Processes



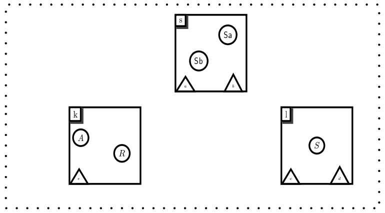
S - site name

$c1, c2, c3$ -resources available at S

Serva, Servb - anonymous threads/agents

Agents may move autonomously from site to site

Distributed Systems



A collection of independent distributed *sites* offering **services/resources** to migrating **agents**.

Resources: Modelled by picalculus channels

Agents: Modelled by (augmented) picalculus processes

Dπ– a π-calculus with explicit locations

A **System** is a collection of autonomous **threads/agents** each running at explicit locations

Syntax of Systems M-N:

- $l[\![P]\!]$ - the thread P running at site l
- $M \mid N$ - threads running in parallel
- $(\mathbf{new}\, e@l : \mathrm{T})\ M$ - sharing information
- 0 - the empty system

Example Systems:

$$l[\![P]\!] \mid (\mathbf{new}\, a@k)\ (l[\![Q]\!] \mid k[\![R]\!])$$

$$l[\![d?(x@z)\, T]\!] \mid k[\![(\mathbf{new}\, a)\ \ \mathbf{goto}\, l.\, d!\langle a@k\rangle]\!]$$

Threads/Agents in Dπ

Syntax of threads, $P - R$:

- $u?(X : \mathrm{T})\, Q$ - *local* input on channel u
- $u!\langle V\rangle\, Q$ - *local* output on channel u
- $\mathbf{goto}\, u.\, P$ - code movement to site u
- $\mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q$ - testing of names
- $(\mathbf{new}\, e : \mathrm{T})\ \ P$ - generation of new names channels or locations
- $P \mid Q$ - concurrent code
- $*P$ - iteration
- **stop** - finished

Values V are locations l, local channels c, or remote channels $c@k$.

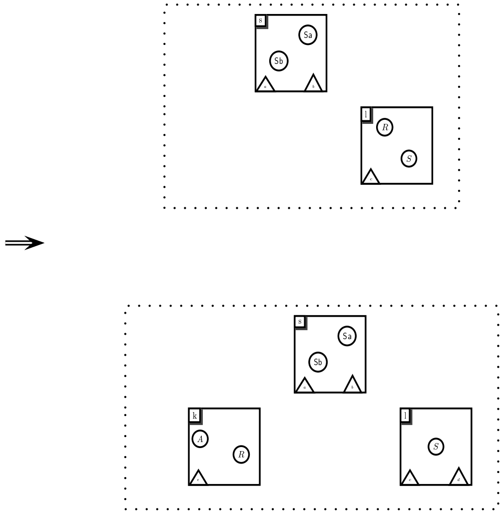
A cell in Dπ

g - inputs a location; the location **must** have a channel called ret on which to return value

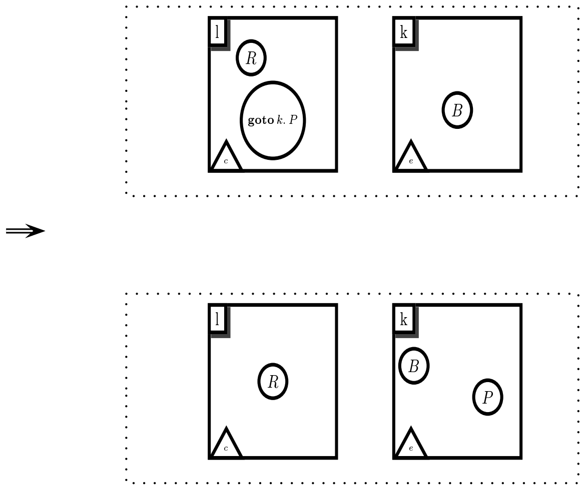
p - inputs a location and new value; location **must** have a channel called ack on which to send acknowledgement.

$$\begin{array}{rcl}
\mathbf{System} & \Leftarrow & l[\![\mathbf{Cell}(v)]\!] \mid h[\![\mathbf{User}]\!] \\
\mathbf{User} & \Leftarrow & \mathbf{goto}\, l.\, \mathrm{p}!\langle h, 0\rangle\ \mid\ \mathrm{ack}?()\ \mathbf{goto}\, l.\, \mathrm{g}!\langle h\rangle \\
 & & \qquad\qquad \mid\ \mathrm{ret}?(x)\ \ \mathrm{print}!\langle x\rangle \\
\mathbf{Cell}(n) & \Leftarrow & (\mathbf{new}\, \mathrm{s})\ \ \mathrm{s}!\langle n\rangle \\
 & & \mid\ *\,\mathrm{g}?(y)\qquad \mathrm{s}?(x)\ \ (\mathrm{s}!\langle x\rangle \mid\ \mathbf{goto}\, y.\, \mathrm{ret}!\langle x\rangle) \\
 & & \mid\ *\,\mathrm{p}?(y, v)\ \ \mathrm{s}?(x)\ \ (\mathrm{s}!\langle v\rangle \mid\ \mathbf{goto}\, y.\, \mathrm{ack}!\langle\rangle)
\end{array}$$

System Evolution



Migration



Reduction Semantics

$M \to N$: in one reduction step system M can evolve to N

- Migration:

$$\text{(move)} \quad l[\![\mathbf{goto}\, k.\, P]\!] \to k[\![P]\!]$$

- Local Communication:

$$\text{(com)} \quad l[\![a!\langle V\rangle\, P]\!] \;\mid\; l[\![a?(X)\, Q]\!] \quad\to\quad l[\![P]\!] \;\mid\; l[\![(Q[{}^{V}\!/\!{}_{X}])]\!]$$

- Name testing:

$$\begin{array}{lll} \text{(eq)} \quad l[\![\mathsf{if}\ u = u\ \mathsf{then}\ P\ \mathsf{else}\ Q]\!] & \to & l[\![P]\!] \\ \phantom{\text{(eq)}} \quad l[\![\mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q]\!] & \to & l[\![P]\!] \ \text{ if } u \neq v \end{array}$$

Reduction Semantics _{making arrangements}

- Name exporting:

$$\text{(newexp)} \quad l[\![(\mathbf{new}\, e)\ P]\!] \quad\to\quad (\mathbf{new}\, e@l)\ l[\![P]\!]$$

- agent splitting:

$$\text{(split)} \quad l[\![P \mid Q]\!] \quad\to\quad l[\![P]\!] \mid l[\![Q]\!]$$

- ignoring details structural equivalence:

$$\frac{M \equiv N,\ M \to M',\ M' \equiv N'}{N \to N'}$$

Structural Equivalence

$$
\begin{array}{lrcl}
\text{(grb1)} & (\mathbf{new}\, e@l)\ 0 & \equiv & 0 \\
\text{(grb2)} & l[\![\mathbf{stop}]\!] & \equiv & 0 \\
\text{(newstr)} & M \mid (\mathbf{new}\, e@l)\ N & \equiv & (\mathbf{new}\, e@l)\ (M \mid N)\ \text{ if } e \notin \mathsf{fn}(M) \\
\text{(repstr)} & *P & \equiv & P \mid *P \\
\text{(parstr)} & M \mid 0 & \equiv & N \\
 & M \mid N & \equiv & N \mid M \\
 & (M \mid N) \mid R & \equiv & M \mid (N \mid R)
\end{array}
$$

A refined cell in Dπ

The ack and reply channels generated on the fly.

$$
\begin{array}{rcl}
\mathbf{System} & \Leftarrow & l[\![\mathbf{Cell}(v)]\!] \mid h[\![\mathbf{User}]\!] \\
\mathbf{User} & \Leftarrow & (\mathbf{new}\, a, r)\ \ l.\,\mathrm{p}!\langle a@h, 0\rangle \\
 & & \quad \mid \mathrm{a}?()\ (l.\,\mathrm{g}!\langle r@h\rangle \ \mid\ \mathrm{r}?(x)\ \ \mathrm{print}!x) \\
\mathbf{Cell}(n) & \Leftarrow & (\mathbf{new}\,\mathrm{s})\ \ \mathrm{s}!\langle n\rangle \\
 & & \mid *\,\mathrm{g}?(z@y) \quad \mathrm{s}?(x)\ \ (\mathrm{s}!\langle x\rangle \mid y.z!\langle x\rangle) \\
 & & \mid *\,\mathrm{p}?(z@y, v)\ \mathrm{s}?(x)\ \ (\mathrm{s}!\langle v\rangle \mid y.z!\langle\rangle)
\end{array}
$$

Notation: calling a method at a location: $l.a!\langle V\rangle$ shorthand for $\mathbf{goto}\, l.\, a!\langle V\rangle\ \mathbf{stop}$

A Cell Server in Dπ

The Server determines site of requested cells.

In response to a request

- a new location, c, is created

- appropriate code is initiated there

- location of new cell is transmitted to client

$$
\begin{array}{rcl}
\mathbf{System} & \Leftarrow & \mathrm{s}[\![\mathbf{S}]\!] \mid h_1[\![\mathbf{User_1}]\!] \mid h_2[\![\mathbf{User_2}]\!] \\
\mathbf{S} & \Leftarrow & *\,\mathrm{req}?(z@y)\ \ (\mathbf{new}\,\mathrm{c}) \\
 & & \quad y.z!\langle \mathrm{c}\rangle \mid \mathbf{goto}\,\mathrm{c}.\ \mathbf{Cell}(0) \\
\mathbf{User_i} & \Leftarrow & (\mathbf{new}\, r)\ \ \mathrm{s}\,.\,\mathrm{req}!\langle r@h_i\rangle \mid r?(z)\ \ \mathbf{User_i}(z)
\end{array}
$$

Behaviour of Cell Server in Dπ

$$\boxed{\mathrm{s}[\![S]\!]\ \mid\ h_1[\![U_1]\!]\ \mid\ h_2[\![U_2]\!]}$$

Request from h_1 is serviced:

$$\boxed{\mathrm{s}[\![S]\!]\ \mid\ \mathrm{c}_1[\![\mathbf{Cell}]\!]\ \mid\ h_1[\![U_1]\!]\ \mid\ h_2[\![U_2]\!]}$$

Request from h_2 is serviced:

$$\boxed{\mathrm{s}[\![S]\!]\ \mid\ \mathrm{c}_1[\![\mathbf{Cell}]\!]\ \mid\ h_1[\![U_1]\!]\ \mid\ h_2[\![U_2]\!]\ \mid\ \mathrm{c}_2[\![\mathbf{Cell}]\!]}$$

- location c_1 chosen by s, private to S, U_1
- location c_2 private chosen by s, to S, U_2
- methods g, p available at locations c_1, c_2

A Local Cell Server

User determines site of requested cells.

- the server is sent an address for the new cell by the client

- the server initialises new version of cell code at this address

- the client is informed of new local method names

$$
\begin{array}{rcl}
\mathbf{System} & \Longleftarrow & \mathrm{s}[\![\mathbf{S}]\!] \mid h_1[\![\mathbf{User_1}]\!] \mid h_2[\![\mathbf{User_2}]\!] \\
\mathbf{S} & \Longleftarrow & *\,\mathrm{req}?(z@y)\ \ \mathbf{goto}\, y.\,(\mathbf{new}\, p, g) \\
 & & \qquad\qquad \mathbf{Cell}_{p,\,g} \mid z!\langle p, g\rangle \\
\mathbf{User_i} & \Longleftarrow & (\mathbf{new}\,\mathrm{c})\ \ \mathbf{goto}\,\mathrm{c}.\,(\mathbf{new}\,\mathrm{ack})\ \ \mathrm{s}\,.\,\mathrm{req}!\langle \mathrm{c}, \mathrm{ack}\rangle \\
 & & \qquad\qquad\qquad \mid \mathrm{ack}?(p, g)\ \ \mathbf{User_i}(p, g)
\end{array}
$$

Behaviour of local Cell Server in $\mathrm{D}\pi$

$$\boxed{\mathrm{s}[\![S]\!] \mid h_1[\![U_1]\!] \mid h_2[\![U_2]\!]}$$

Request from h_1 is serviced:

$$\boxed{\mathrm{s}[\![S]\!] \mid \mathrm{c}_1[\![\mathbf{Cell}]\!] \mid h_1[\![U_1]\!] \mid h_2[\![U_2]\!]}$$

Request from h_2 is serviced:

$$\boxed{\mathrm{s}[\![S]\!] \mid \mathrm{c}_1[\![\mathbf{Cell}]\!] \mid h_1[\![U_1]\!] \mid h_2[\![U_2]\!] \mid \mathrm{c}_2[\![\mathbf{Cell}]\!]}$$

- location c_1 chosen by h_1, private to S, U_1
- location c_2 chosen by h_2, private to S, U_2
- methods g_1, p_1 available at location c_1, private to S, U_1
- methods g_2, p_2 available c_2, private to S, U_2

Remote Channel Creation

In $\mathrm{D}\pi$ new channels can only be created at the *current* location.

An expensive implementation of *remote* channel creation:

$$h[\![(\mathbf{new}\, k)\ \ \mathbf{goto}\, k.\,(\mathbf{new}\, a, b)\ \ \mathbf{goto}\, h.\, P]\!]$$

- the continuation thread P is transmitted twice through the network.

More efficient:

$$h[\![T(h)]\!]$$

where

$$
\begin{array}{rcl}
T(h) & \Longleftarrow & (\mathbf{new}\, r, k)\ \ \mathbf{goto}\, k.\,(\mathbf{new}\, a, b) \\
 & & \qquad\qquad \mathbf{goto}\, h.\, r!\langle (a, b)@k\rangle \\
 & & \qquad\qquad \mid r?((z, x)@y)\ P[k,a,b/z,x,y]
\end{array}
$$

Routed Forwarding

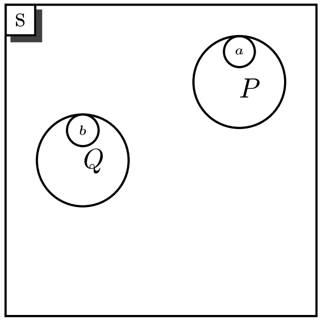
Write a forwarding program $\mathbf{Fwd}(\mathrm{in}\,@h, \mathrm{s}\,@d)$ which sets up a communication link between channel in, local to h and s, local to d.

$$h[\![*\,\mathrm{in}?(x)\,\mathbf{goto}\, d.\,\mathrm{s}!\langle x\rangle]\!]$$

Assuming no direct connection from h to d but instead a function *route*(d) which returns the neighbour nearest to d:

$$
\begin{array}{rcl}
\mathbf{Fwd}(h[\mathrm{in}], d[\mathrm{s}]) & \Longleftarrow & \mathsf{if}\ h = d\ \mathsf{then}\ \ *\,\mathrm{in}?(x)\,\mathrm{s}!\langle x\rangle \\
 & & \mathsf{else} \\
 & & \quad \mathsf{let}\ n \leftarrow \mathit{route}(d) \\
 & & \quad \mathsf{in}\ \ (\mathbf{new}\, c@n)\ \ \mathbf{goto}\, n.\,\mathbf{Fwd}(n[c], d[s]) \\
 & & \qquad\qquad\qquad \mid *\,\mathrm{in}?(x)\,\mathbf{goto}\, n.\,\mathrm{c}!\langle x\rangle
\end{array}
$$

Nomadic Pict



Independent agents are **named** - $a, b, \ldots$

Communication is local **within/between agents**

Channels **owned** by agents

Sites have a flat structure

Nomadic Pict

Two level architecture:

locations: physical machines/runtime systems

agents: **named** autonomous threads

Two extra constructs:

$\mathsf{agent}\, a = P \mathrel{\mathsf{in}} Q$ - start a new agent named a with body P in parallel with Q

$\mathsf{iflocal}\langle a \rangle\, c!\langle v \rangle \rightarrow P \mathrel{\mathsf{else}} Q$ - if a is local send v to a on channel c (of agent a) and continue with P, else continue with Q.

Nomadic Pict - Example

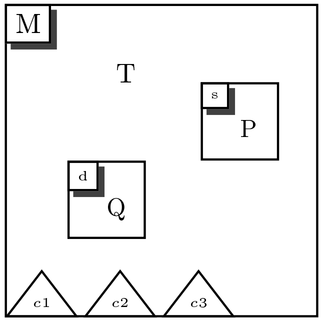
$$\begin{aligned} l[\![\mathsf{agent}\, \mathrm{s} \quad &= \quad \mathsf{getApp}?(cl, ack, k) \\ &\quad\ \ \mathsf{agent}\, a = \mathsf{goto}\ k. \\ &\qquad\qquad \mathsf{iflocal}\langle cl \rangle\, ack!\langle a \rangle \ \mid A]\!] \end{aligned}$$

Server s running at location l.

Receives requests at channel getApp from client cl

- spawns new applet called a
- new applet migrates to k
- name of applet a is sent to cl via ack
- code of applet A starts running (at k)

Seals



Sites have **sub-sites** - **dependent** sons/daughters

Communication parent $\leftrightarrow$ offspring and offspring $\leftrightarrow$ offspring

Channels **owned by parent**

Migration of offspring **controled** by parent

Seal - Reduction semantics

Communication - via channels:

offspring - parent

$$c^{\mathsf{son}}(x)?.P \quad | \quad \mathsf{son}[\![c^{\uparrow}!\langle v\rangle\,.Q \mid R]\!] \;\rightarrow\; P\{v/x\} \quad | \quad \mathsf{son}[\![Q \mid R]\!]$$

Migration - via **channels**:

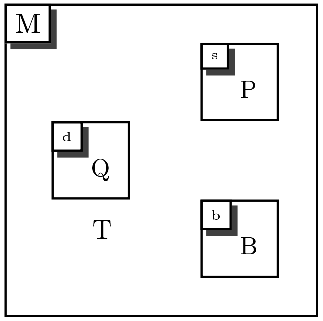
offspring - parent

$$c^{\mathsf{son}}\{\mathsf{tom}\}.P \quad | \quad \mathsf{son}[\![c^{\uparrow}\langle \mathsf{b}\rangle\,.Q \;\mid\; \mathsf{b}[\![R]\!] \;\mid\; S]\!]$$

$$\rightarrow$$

$$P \quad | \quad \mathsf{son}[\![Q \mid S]\!] \quad | \quad \mathsf{tom}[\![R]\!]$$

Ambients



Sites have **sub-sites** - **autonomous** offspring

No Channels - communication internal to sites via β-reduction

Migration of offspring **independent** of parent

Ambients Reduction semantics

Enter:

$$n[\![\mathsf{in}\ m.P \;\mid\; Q]\!] \quad | \quad m[\![R]\!] \;\rightarrow\; m[\![\,n[\![P \mid Q]\!] \quad | \quad R]\!]$$

Exit:

$$m[\![n[\![\,!\langle m\rangle.\,.P \mid Q]\!] \;\mid\; R]\!] \;\rightarrow\; n[\![P \mid Q]\!] \quad | \quad m[\![R]\!]$$

Open:

$$\mathsf{open}\ n.P \quad | \quad n[\![Q]\!] \;\rightarrow\; P \quad | \quad Q$$

Communication:

$$(x).P \quad | \quad \langle M\rangle \;\rightarrow\; P\{M/x\}$$

Note: all autonomous moves

Ambients - Firewall Access Example

Password k - known to agent

Firewall w - secret

$$\begin{aligned}
\mathsf{Firewall} &= (\mathbf{new}\,w)\;\; k[\![\mathsf{in}\ k.\ \mathsf{in}\ \ w]\!] \;\mid\; w[\![\mathsf{open}\ k.P]\!] \\
\mathsf{Agent} &= k[\![\mathsf{open}\ k.Q]\!]
\end{aligned}$$

$$\begin{aligned}
A \;\mid\; F \quad &\rightarrow^* \quad (\mathbf{new}\,w)\;\; k[\![k[\![\mathsf{in}\ w]\!] \;\mid\; \underline{\mathsf{open}\ k}.Q]\!] \quad | \quad F \\
&\rightarrow^* \quad (\mathbf{new}\,w)\;\; k[\![\underline{\mathsf{in}\ w} \;\mid\; Q]\!] \quad | \quad F \\
&\rightarrow^* \quad (\mathbf{new}\,w)\;\; w[\![k[\![Q]\!] \;\mid\; \underline{\mathsf{open}\ k}.P]\!] \\
&\rightarrow^* \quad (\mathbf{new}\,w)\;\; w[\![Q \mid P]\!]
\end{aligned}$$

Types in Located Calculi

DPI: Sites offer services - types of form $[s_1 : \mathrm{T}_1, \ldots, s_n : \mathrm{T}_n]$

Ensuring that incoming agents only use services to which they have been granted access - Resource Access Control

Seal: Sites offer services as above

Ensuring that offspring conform to the interface determined by their parents

Enforcing security properties

Ambients: Control mobility of autonomous agents many proposals

Type of ambient controls activity within ambient

eg when is an ambient really a channel ?

Behavioural Equivalences in Located Calculi

Nomadic Pict: *Translocating equivalences*

Co-inductive equivalence parameterised on set of agent names which are *movable* by the environment.

Application: Implementation of location independent agent communication.

$$\langle a@?\rangle\, c!\langle v\rangle$$

Ambients: Contextual Equivalences

$$(\mathbf{new}\, n)\;\; n[\![P]\!] \simeq \mathbf{stop}$$

if $n \notin fn(P)$

$R \simeq T$ means that $C[R]$ and CT have the same *observables*, in all contexts $C[\,]$.