An Abstract Machine for a Higher-Order Distributed Process Calculus

Florence Germain (France Telecom R & D)

Marc Lacoste (France Telecom R & D)

Jean-Bernard Stefani (INRIA)

Motivation

- **Foundations of WAN-based programming :**
  - **A primitive formal model remains elusive**
  - **Existence of various forms of barriers**
  - **Need for some notion of locality**
  - **Various forms of localities in recent distributed process calculi :**
    - **Ambients (Mobile, Safe, Boxed . . .)**
    - **Dπ, Join, Seal, Nomadic Pict**
    - **DiTyCo**
    - **Klaim**

    . . .

Motivation

- **Localities :**
  - **Specific calculi use specific types of localities using dedicated interaction protocols to capture :**
    - **Resource access control**
    - **Process mobility**
    - **Failure modes**

    . . .
  - **A “real” distributed system features combination of localities**
  - **Need for a model to enable programming within the same calculus different forms of locality**

The M-calculus

- **Cells $a(P)[Q]$ = programmable localities**
  - **Membrane**
  - **Content**
- **Communication**
  - **Fully transparent (join) / encoded using mobility (ambients)?**
  - **A trade-off :**
    - **Transparent asynchronous interaction but indirect communication**
    - **Elementary routing steps : a single cell membrane crossing at a time**
    - **Silent routing mechanism : a membrane filters incoming and outgoing messages**

The M-calculus

- **Migration**
  - **A higher-order calculus**
  - **Migration = communication of a passivated process**
- **Dynamic binding features**
- **Key issue : how difficult is the M-calculus to implement?**

Contributions

- A preliminary implementation of the M-calculus
- Specification using a distributed abstract machine (AM)
  - Higher-order features + distribution + mobility can be effectively implemented in a distributed setting
  - The implementation is no more complex than an AM for the Join calculus (JAM)
- A modular AM

Outline

- An Overview of the M-calculus
- The Abstract Machine
- An Implementation
- Towards a MIKADO Core Sofware Framework

Syntax

- **A mix of call-by-value λ-calculus and Join calculus**

$\mathcal{S}$	$::=$	$\epsilon[P] \mid \nu n.\mathcal{S}$	**System**
P	$::=$	$\mathbf{0} \mid V \mid \mathbf{a}(P)[P] \mid (P \mid P) \mid (PP) \mid$	**Process**
		$\nu n.P \mid ([\mu = V]P, P) \mid \langle D \rangle \mid \mathtt{pass}_{\mathtt{a}}\ V$	
V	$::=$	$() \mid u \mid d \mid (V, \ldots, V) \mid \lambda x.P$	**Value**
D	$::=$	$\top \mid J \triangleright P \mid D; D$	**Definition**
J	$::=$	$r\widetilde{x} \mid J \mid J$	**Join-pattern**

Syntax

Different kinds of names

u	$::=$	$a \mid \mathbf{r} \mid d.\mathbf{r}$	**Name**
$\mathbf{r}$	$::=$	$r \mid x$	**Variable resource name**
d	$::=$	$\bar{\mathbf{a}} \mid \underline{\mathbf{a}}$	**Target name**
$\mathbf{a}$	$::=$	$a \mid x$	**Variable cell name**
n	$::=$	$r \mid a$	**Resolved name**
μ	$::=$	$u \mid \_ \mid d.\_ \mid \_.\mathbf{r}$	**Name pattern**

- **Free names** $\mathit{fn}(P)$
- **Defined local names** $\mathit{dln}(P)$
- **Active cells** $\mathit{cells}(P)$

Communication and Migration

- **Asynchronous point-to-point exchange of messages**
- **Messages = applications**
  - **Local messages** $r\widetilde{V}$ **never cross cell boundaries**
  - **Addressed messages** $d.r\widetilde{V}$ **(**$d = \overline{a}|\underline{a}$**) are routed from cell to cell, one boundary at a time**
- **A higher-order calculus :** migration of a cell $a(P)[Q]$ is captured by the communication of **thunks**, resulting from cell passivation using $\texttt{pass}_a$

Structural Congruence

S.NU.PAR
$$\frac{n \notin fn(Q)}{(\nu n.P) \mid Q \equiv \nu n.P \mid Q}$$

S.NU.TOP
$$\frac{}{\epsilon[\nu n.P] \equiv \nu n.\epsilon[P]}$$

S.NU.MEMB
$$\frac{n \notin fn(Q) \ \wedge \ n \neq a}{a(\nu n.P)[Q] \equiv \nu n.a(P)[Q]}$$

S.α
$$\frac{P =_{\alpha} Q}{P \equiv Q}$$

S.NU.CONT
$$\frac{n \notin fn(P) \ \wedge \ n \neq a}{a(P)[\nu n.Q] \equiv \nu n.a(P)[Q]}$$

S.CONTEXT
$$\frac{P \equiv Q}{\mathbf{E}\{P\} \equiv \mathbf{E}\{Q\}}$$

$$\mathbf{E} ::= \ \cdot \ \mid \mathbf{E}V \mid P\mathbf{E} \mid \nu n.\mathbf{E} \mid (\mathbf{E} \mid P) \mid$$
$$a(P)[\mathbf{E}] \mid a(\mathbf{E})[P] \mid \epsilon[\mathbf{E}]$$

Computation

R.BETA

$$\frac{}{(\lambda x.P)V \longrightarrow P\{^V/_x\}}$$

R.COM

$$\frac{\langle D \rangle = \langle D_0 \;;\; r_1 \widetilde{x_1} \mid \ldots \mid r_n \widetilde{x_n} \triangleright P \rangle}{\langle D \rangle \mid r_1 \widetilde{V_1} \mid \ldots \mid r_n \widetilde{V_n} \longrightarrow \langle D \rangle \mid P\{^{\widetilde{V_i}}/_{\widetilde{x_i}}\}}$$

R.PASSIV

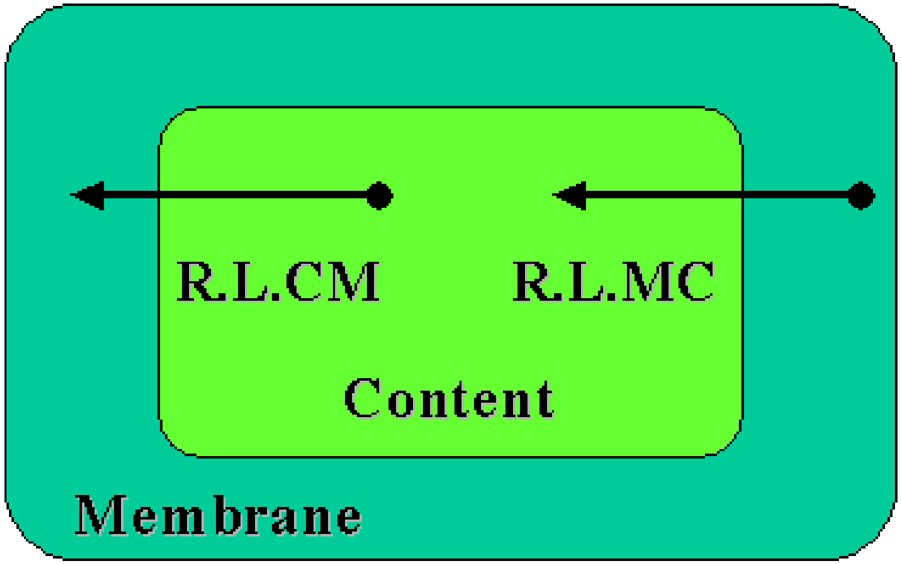
$$\frac{}{a(\mathtt{pass}_a\ V \mid P)[Q] \longrightarrow V(\lambda.P)(\lambda.Q)}$$

R.P.EQUIV

$$\frac{P \equiv P' \qquad P' \longrightarrow Q' \qquad Q' \equiv Q}{P \longrightarrow Q}$$

Routing Local Messages

- Routing between membrane and content of the same cell



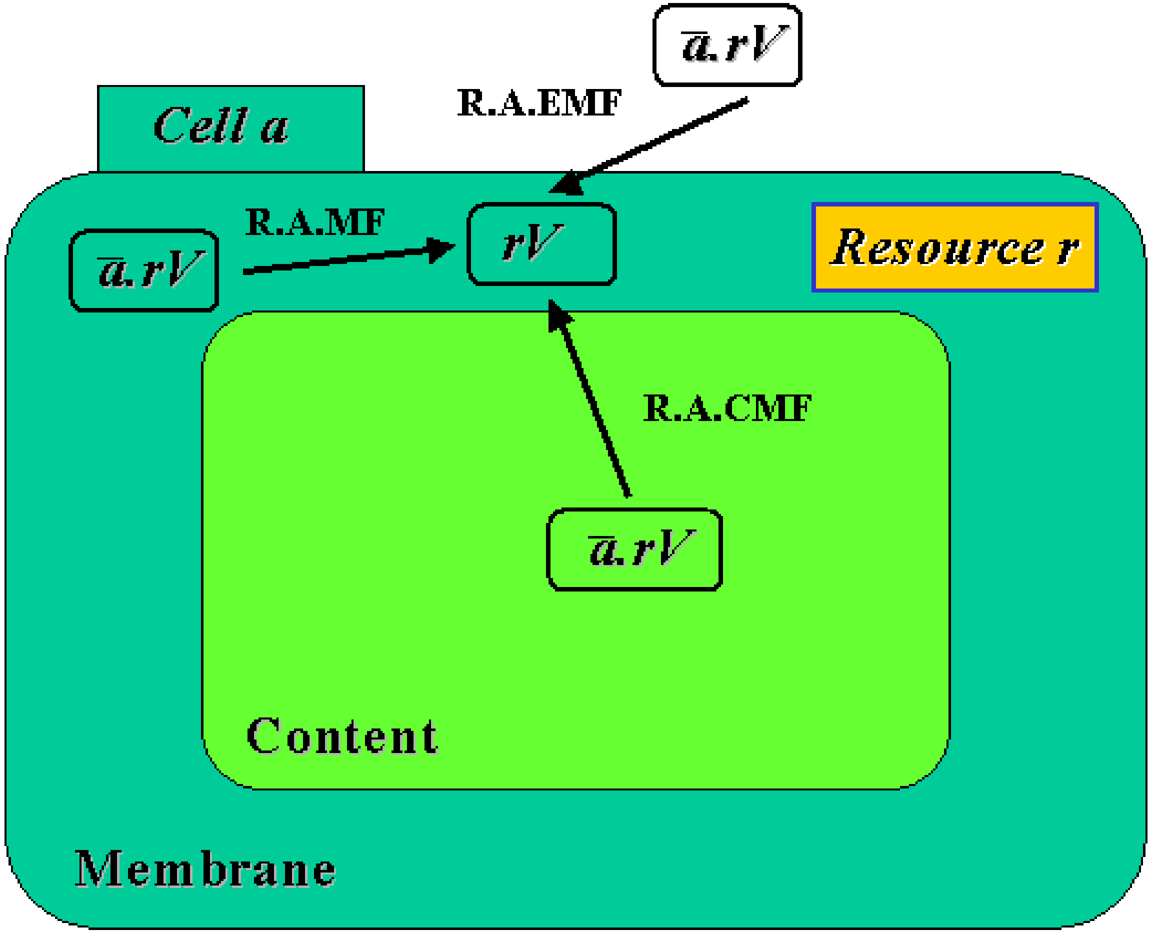
R.L.MC

$$\frac{r \notin dln(P) \qquad r \in dln(Q)}{a(P \mid r\widetilde{V})[Q] \rightarrow a(P)[Q \mid r\widetilde{V}]}$$

R.L.CM

$$\frac{r \in dln(P) \qquad r \notin dln(Q)}{a(P)[Q \mid r\widetilde{V}] \rightarrow a(P \mid r\widetilde{V})[Q]}$$

Routing Addressed Messages



- Conversion into a local message

R.A.MF

$$\frac{r \in dln(P)}{a(P \mid \overline{a}.r\widetilde{V})[Q] \rightarrow a(P \mid r\widetilde{V})[Q]}$$

R.A.EMF

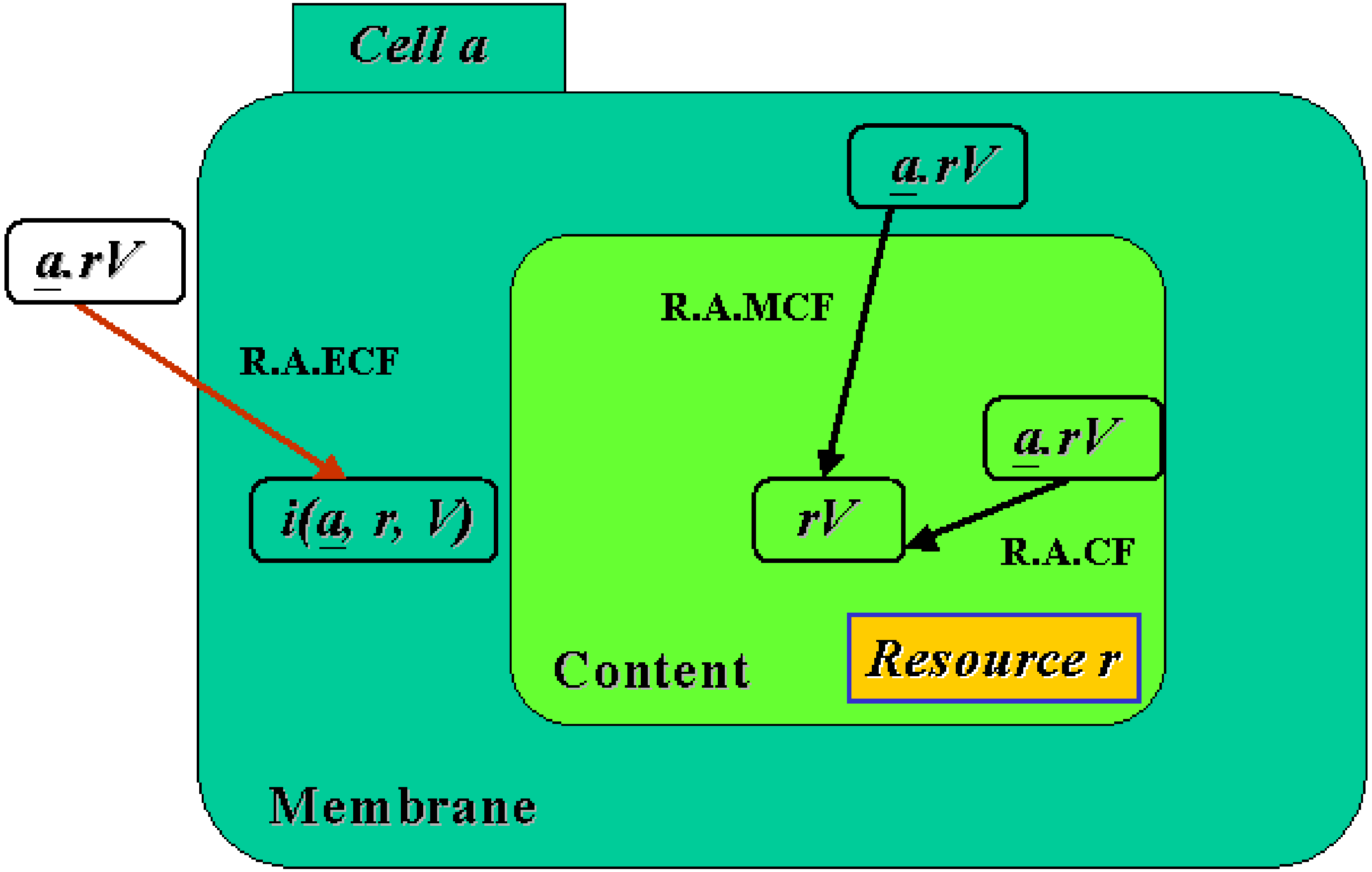
$$\frac{r \in dln(P)}{\overline{a}.r\widetilde{V} \mid a(P)[Q] \rightarrow a(P \mid r\widetilde{V})[Q]}$$

R.A.CMF

$$\frac{r \in dln(P)}{a(P)[Q \mid \overline{a}.r\widetilde{V}] \rightarrow a(P \mid r\widetilde{V})[Q]}$$

Routing Addressed Messages

❒ **Filtering / conversion into a local message**



Routing Addressed Messages

❐ Filtering / conversion into a local message

R.A.CF

$$\frac{r \in \mathit{dln}(Q)}{a(P)[Q \mid \underline{a}.r\widetilde{V}] \rightarrow a(P)[Q \mid r\widetilde{V}]}$$

R.A.MCF

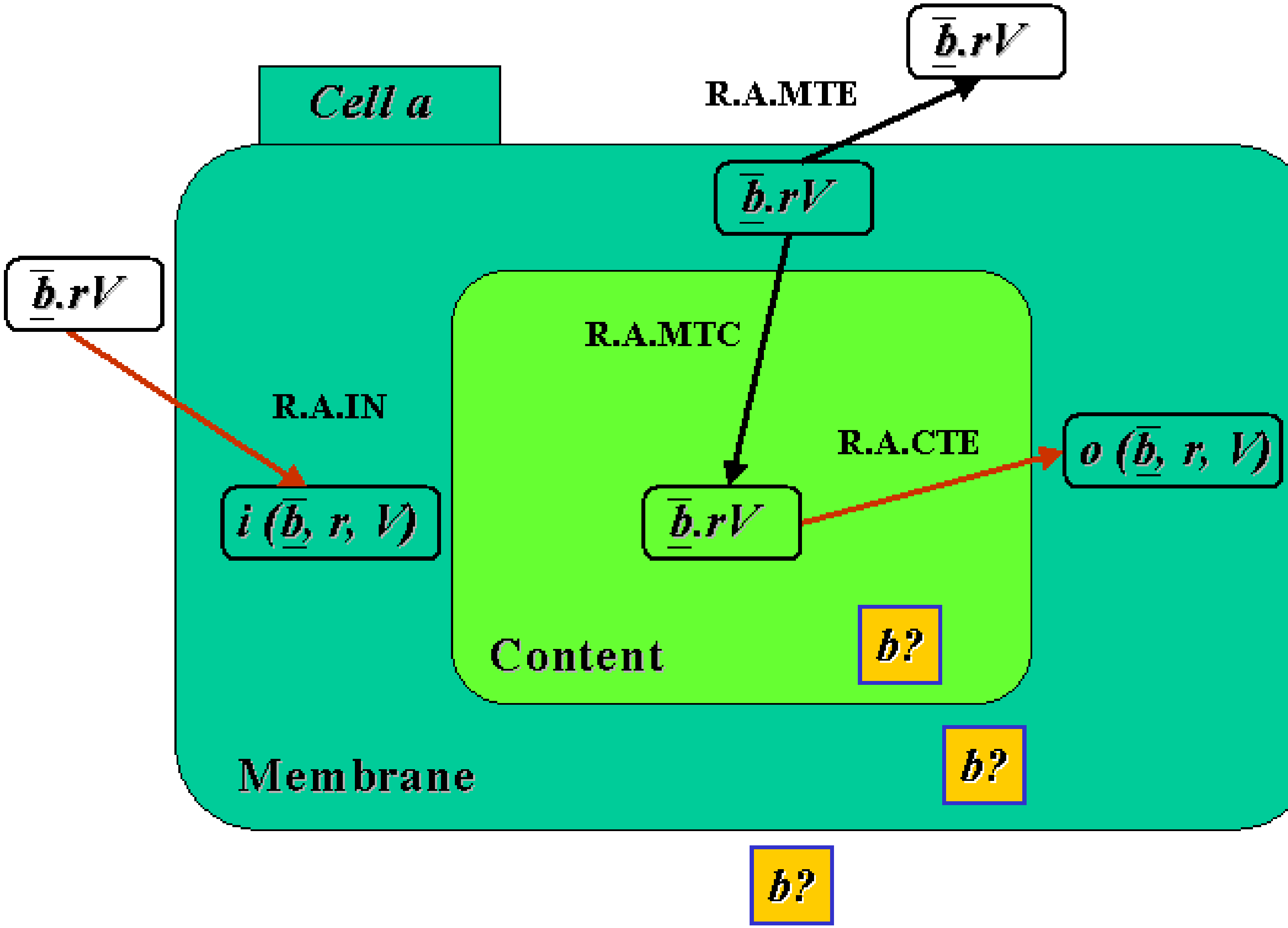
$$\frac{r \in \mathit{dln}(Q)}{a(P \mid \underline{a}.r\widetilde{V})[Q] \rightarrow a(P)[Q \mid r\widetilde{V}]}$$

R.A.ECF

$$\frac{r \in \mathit{dln}(Q)}{\underline{a}.r\widetilde{V} \mid a(P)[Q] \rightarrow P \mid \mathbf{i}(\underline{a}, r, \widetilde{V}))[Q]}$$

Routing Addressed Messages

❒ Routing to other cells



Routing Addressed Messages

❒ Routing to other cells

R.A.IN

$$\frac{b \in \mathit{cells}(P) \cup \mathit{cells}(Q) \qquad b \neq a}{\overline{\underline{b}}.r\widetilde{V} \mid a(P)[Q] \rightarrow a(P \mid \mathbf{i}(\overline{\underline{b}}, r, \widetilde{V}))[Q]}$$

R.A.MTC

$$\frac{b \in \mathit{cells}(Q) \setminus \mathit{cells}(P) \qquad b \neq a}{a(P \mid \overline{\underline{b}}.r\widetilde{V})[Q] \rightarrow a(P)[Q \mid \overline{\underline{b}}.r\widetilde{V}]}$$

R.A.MTE

$$\frac{b \notin \mathit{cells}(P) \cup \mathit{cells}(Q) \qquad b \neq a}{a(P \mid \overline{\underline{b}}.r\widetilde{V})[Q] \rightarrow a(P)[Q] \mid \overline{\underline{b}}.r\widetilde{V}}$$

R.A.CTE

$$\frac{b \notin \mathit{cells}(Q) \qquad b \neq a}{a(P)[Q \mid \overline{\underline{b}}.r\widetilde{V}] \rightarrow a(P \mid \mathbf{o}(\overline{\underline{b}}, r, \widetilde{V}))[Q]}$$

Programming in the M-calculus

- **Transparent communications :** a simple forwarder

$$Fwd = \langle \mathbf{i}(d,r,v) \triangleright d.r\ v\ ;\ \ \mathbf{o}(d,r,v) \triangleright d.r\ v \rangle$$

- **Cell mobility :**
  - **Consider the following cell :**

$$Q^m(a) \quad = \quad a(Fwd \mid \langle \mathsf{go}\ u \triangleright Go(a,u) \rangle)[Q]$$
$$Go(a,u) \quad = \quad \mathtt{pass}_a\ \lambda p\, q.(\overline{u}.\mathsf{enter}\ \lambda.a(p())[q()])$$

  - **Cell passivation :** $(\mathsf{go}\ u)$
  - **Cell reception :** $Enter(u)$

$$Enter(u) \quad = \quad \langle \mathsf{enter}\ f \triangleright \mathtt{pass}_u\ \lambda p\, q.u(p())[q() \mid f()] \rangle$$

Outline

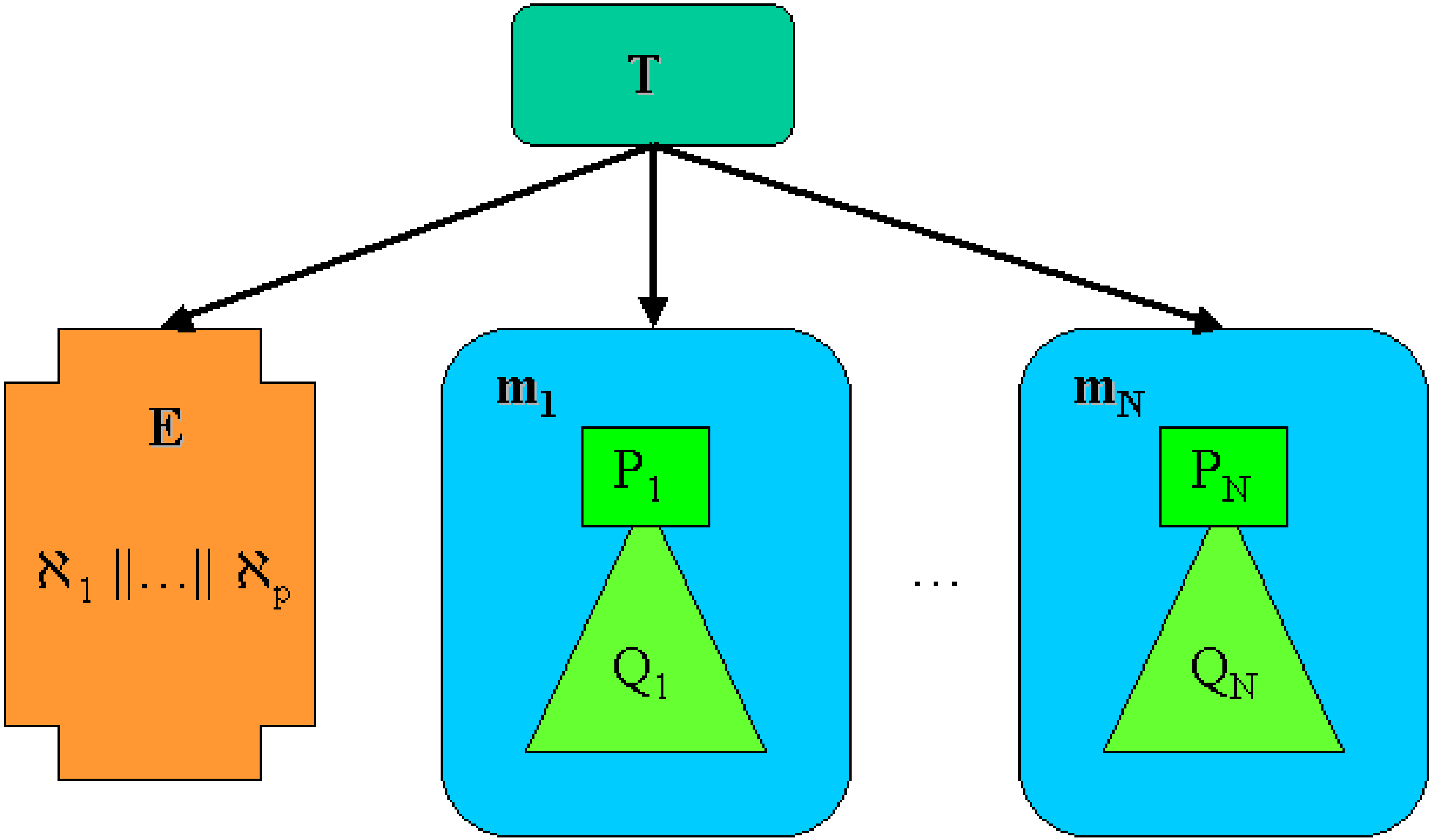
- An Overview of the M-calculus
- ➨ The Abstract Machine
- An Implementation
- Towards a MIKADO Core Software Framework

CLAM : Design Principles

- **Aim: demonstrate the implementability of the M-calculus**
- **Infrastructure requirements :**
  - **A direct refinement of the M-calculus**
    - **CLAM (Cellular Abstract Machine)**
  - **Implementability in a distributed setting :**
    - **Logical and physical distribution**
    - **Distributed interaction by asynchronous message passing between physical sites**
    - **Locality for reduction rules**
  - **Modularity : clear separation**
    - **Functional evaluation engine vs. concurrency / distribution management**
    - **Routing and name management vs. computation : *a distributed lookup service guarantees routing determinacy (unicity of cell names property)***

Overview

❒ Distribution model



Overview

- **Lookup service :**
  - **In which location is a resource defined?**
  - **On which machine does a location reside?**
- **Locations**
  - **Membrane and ether locations**
  - **Structure : execution engine + functional evaluator**

Machine and Location State

- **Values**
  - $\text{reify}_{\text{control}}$
  - $\text{reify}_{\text{content}}$
  - **Internal names**
- **Location state** l **:** $\langle \alpha, \mathcal{D}, \mathcal{H}, \mathcal{R}, \mathcal{E}, \mathcal{L} \rangle$
- **Machine state** $\langle m : N : O : L \rangle$
- **Reduction rules**

$$m : N : O : L\{l \xmapsto[l_e:\mathcal{L}]{\mathcal{D}:\mathcal{H}} \langle \alpha, \mathcal{R} \rangle, \ldots\} \parallel E\{m \to m' \mapsto \mathcal{M}\} \longrightarrow$$

$$m : N' : O' : L'\{l \xmapsto[l_e:\mathcal{L}']{\mathcal{D}':\mathcal{H}'} \langle \alpha, \mathcal{R}' \rangle, \ldots\} \parallel E\{m \to m' \mapsto \mathcal{M} :: msg\}$$

Computation

PRL

$$\frac{}{m : N : O : L\{l \overset{-}{\underset{-}{\longmapsto}} \langle \alpha, (\rho, (P \mid Q))\rangle\} \longrightarrow m : N : O : L\{l \overset{-}{\underset{-}{\longmapsto}} \langle \alpha, (\rho, P) :: \mathcal{R} :: (\rho, Q)\rangle\}}$$

NEW

$$\frac{\rho' = \rho\{n \mapsto i\} \quad i = \langle m, i_{loc}\rangle \quad i_{loc} \notin N}{m : N : O : L\{l \overset{-}{\underset{-}{\longmapsto}} \langle \alpha, (\rho, (\nu\ n)\ P) :: \mathcal{R}\rangle\} \longrightarrow m : N \oplus \{i_{loc}\} : O : L\{l \overset{-}{\underset{-}{\longmapsto}} \langle \alpha, (\rho', P) :: \mathcal{R}\rangle\}}$$

Computation

R.BETA $$\frac{}{(\lambda x.P)V \longrightarrow P\{^V/_x\}}$$

EVAL $$\frac{\mathbb{S}_l = [\![\, E \,]\!]_{load} \qquad \mathbb{S}_l \Longrightarrow_\beta \mathbb{S}'_l \qquad P = [\![\, \mathbb{S}'_l \,]\!]_{unload}}{m : N : O : L\{l \overset{-}{\underset{-}{\longmapsto}} \langle \alpha, (\rho, E) :: \mathcal{R} \rangle\} \longrightarrow m : N : O : L\{l \overset{-}{\underset{-}{\longmapsto}} \langle \alpha, \mathcal{R} :: (\rho, P) \rangle\}}$$

Definitions

$$\rho' = \rho \restriction \bigcup_{J_i \triangleright P_i \in D} \mathit{fn}(P_i) \setminus \mathit{rn}(J_i) \quad \mathcal{D}' = \mathcal{D} \oplus (\rho', D)$$

DEF
$$\frac{\mathcal{H}' = \mathcal{H}\ \{\rho(r) \mapsto \bullet\}_{r \in \mathit{dn}(D)} \qquad O' = O\ \{\rho(r) \mapsto^{+} \langle l, 0 \rangle\}_{r \in \mathit{dn}(D)}}{m : N : O : L\{l \xmapsto[-]{\mathcal{D}:\mathcal{H}} \langle \alpha, (\rho, \langle D \rangle) :: \mathcal{R} \rangle\} \longrightarrow m : N : O' : L\{l \xmapsto[-]{\mathcal{D}':\mathcal{H}'} \langle \alpha, \mathcal{R} \rangle\}}$$

Local Communication

R.COM

$$\frac{\langle D\rangle = \langle D_0 \,;\, r_1\widetilde{x_1} \mid \ldots \mid r_n\widetilde{x_n} \triangleright P\rangle}{\langle D\rangle \mid r_1\widetilde{V_1} \mid \ldots \mid r_n\widetilde{V_n} \longrightarrow \langle D\rangle \mid P\{^{\widetilde{V_i}}/_{\widetilde{x_i}}\}}$$

COM

$$\frac{\begin{array}{c}(\rho, D) \in \mathcal{D} \quad D = \ldots; J \triangleright P; \ldots \\ J = r_1\widetilde{x_1} \mid \ldots \mid r_q\widetilde{x_q} \quad \left[\mathcal{H}(\rho(r_i)) = (\rho_i, \widetilde{V_i}) :: \mathcal{V}_i\right]_{1\le i\le q} \\ \rho' = [\rho \,\cup \bigcup_{1\le i\le q} \rho_i]\{\widetilde{x_i} \mapsto \widetilde{V_i}\} \qquad \mathcal{H}' = \left[\mathcal{H}\;\{\rho(r_i) \mapsto \mathcal{V}_i\}\right]_{1\le i\le q}\end{array}}{\begin{array}{c}m : N : O : L\{l \xrightarrow[\_]{\mathcal{D}:\mathcal{H}} \langle\alpha, \mathcal{R}\rangle\} \longrightarrow \\ m : N : O : L\{l \xrightarrow[\_]{\mathcal{D}:\mathcal{H}'} \langle\alpha, \mathcal{R} :: (\rho', P)\rangle\}\end{array}}$$

Cell Creation

$$\textbf{CELL}\quad \frac{\begin{array}{c} l' = \langle m, l'_{loc} \rangle \quad l'' = \langle m, l''_{loc} \rangle \\ l'_{loc} \notin N \quad l''_{loc} \notin N \quad \mathcal{L}'_0 = \mathcal{L}_0 \cup \{l'\} \\ \rho' = closure(\rho, P) \quad \rho'' = closure(\rho, Q) \end{array}}{\begin{array}{c} m : N : O : L\{l_0 \xmapsto[l:\mathcal{L}_0]{-} \langle \alpha_0, \mathcal{R}_0 \rangle \,, \\ l \xmapsto[-]{-} \langle \sharp, (\rho, b(P)[Q]) :: \mathcal{R} \rangle\} \longrightarrow \\ m : N \oplus \{l'_{loc}, l''_{loc}\} : O \oplus \{l' \mapsto \langle m, 0 \rangle \,, l'' \mapsto \langle m, 0 \rangle\} : \\ L\{l_0 \xmapsto[l:\mathcal{L}'_0]{-} \langle \alpha_0, \mathcal{R}_0 \rangle \,, l \xmapsto[-]{-} \langle \sharp, \mathcal{R} \rangle \,, \\ l' \xmapsto[l'':\emptyset]{\emptyset:\emptyset} \langle \rho(b), (\rho', P) \rangle \,, l'' \xmapsto[-]{\emptyset:\emptyset} \langle \sharp, (\rho'', Q) \rangle\} \end{array}}$$

Message Routing : Principles

- **A distributed lookup service**
  - **A distributed database**
  - **Update policies**
    - **Consistent views (atomic broadcast)**
    - **Temporarily inconsistent views (forwarders)**

Definition 1 **The update policy is sound iff the only consequence on message routing of temporarily inconsistent views will be delayed message emission or reception, assuming reliable inter-machine communications**

  - **A lookup service per machine for definitions and locations**
  - **A routing algorithm R* :**
    - **Forwarders**
    - **Piggyback routing information updates on communication messages**
    - **Timestamps to capture the instant of migration (cells, definitions)**
    - ➡ **Avoid communication inconsistencies due to stale routing information**

Routing Local Messages

R.L.MC $$\frac{r \notin \mathit{dln}(P) \qquad r \in \mathit{dln}(Q)}{a(P \mid r\widetilde{V})[Q] \longrightarrow a(P)[Q \mid r\widetilde{V}]}$$

M.L.MC $$\frac{l' \Subset O(\rho(r)) \quad l \not\Subset O(\rho(r))}{\begin{array}{c} m : N : O : L\{l \xmapsto[l':\mathcal{L}]{-} \left\langle \alpha, (\rho, r\widetilde{V}) :: \mathcal{R} \right\rangle, l' \xmapsto[-]{-} \langle \sharp, \mathcal{R}' \rangle\} \longrightarrow \\ m : N : O : L\{l \xmapsto[l':\mathcal{L}]{-} \langle \alpha, \mathcal{R} \rangle, l' \xmapsto[-]{-} \left\langle \sharp, \mathcal{R}' :: (\rho, r\widetilde{V}) \right\rangle\} \end{array}}$$

Routing Addressed Messages

❐ Intra-machine communication

R.A.MTE
$$\frac{b \notin \mathit{cells}(P) \cup \mathit{cells}(Q) \qquad b \neq a}{a(P \mid \overline{\underline{b}}.r\widetilde{V})[Q] \longrightarrow a(P)[Q] \mid \overline{\underline{b}}.r\widetilde{V}}$$

M.A.MTE.L
$$\frac{siblings(l, l') \quad O(\rho(b)) = \langle k, t \rangle \qquad k \neq l \quad \neg\, desc(k, l)}{\begin{array}{c} m : N : O : L\{l \longmapsto \left\langle \alpha, (\rho, \overline{\underline{b}}.r\widetilde{V}) :: \mathcal{R} \right\rangle, l' \longmapsto \langle \sharp, \mathcal{R}' \rangle\} \longrightarrow \\ m : N : O : L\{l \longmapsto \langle \alpha, \mathcal{R} \rangle, l' \longmapsto \left\langle \sharp, \mathcal{R}' :: (\rho, \overline{\underline{b}}.r\widetilde{V}) \right\rangle\} \end{array}}$$

Routing Addressed Messages

Routing Addressed Messages

- **Inter-machine communication : message send**

R.A.MTE
$$\frac{b \notin \mathit{cells}(P) \cup \mathit{cells}(Q) \qquad b \neq a}{a(P \mid \underline{\overline{b}}.r\widetilde{V})[Q] \longrightarrow a(P)[Q] \mid \underline{\overline{b}}.r\widetilde{V}}$$

Routing Addressed Messages

$$\frac{\begin{array}{c} l_\top \in \top.\mathcal{L} \quad O(\rho(b)) = \langle l'', t'' \rangle \\ O(l'') = \langle m', t' \rangle \quad m' \neq m \\ O' = O\{\forall l \in loc\_id(\widetilde{V}) \textbf{ where} \\ O(l) = \langle m, t_l \rangle, l \mapsto \langle m', t_l + 1 \rangle\} \\ O'' = O'\{\forall r \in rsc\_id(\widetilde{V}) \textbf{ where} \\ O'(\rho(r)) = \langle l_r, t_r \rangle, \rho(r) \mapsto \langle \daleth, t_r + 1 \rangle\} \\ O''' = O''\{\daleth \mapsto \langle m', 0 \rangle\} \quad \daleth = \langle m, \daleth_{loc} \rangle \quad \daleth_{loc} \notin N \end{array}}{\begin{array}{c} m : N : O : L\{l_\top \overset{-}{\underset{-}{\longmapsto}} \left\langle \alpha, (\rho, \underline{\overline{b}}.r\widetilde{V}) :: \mathcal{R} \right\rangle\} \parallel \\ E\{m \rightarrow m' \mapsto \mathcal{M}\} \longrightarrow \\ m : N \oplus \{\daleth_{loc}\} : O' : L\{l_\top \overset{-}{\underset{-}{\longmapsto}} \langle \alpha, \mathcal{R} \rangle\} \parallel \\ E\{m \rightarrow m' \mapsto \mathcal{M} :: msg(\underline{\overline{b}}.r, \widetilde{V}, \rho, O''')\} \end{array}} \quad \textbf{M.A.MTE.R}$$

Routing Addressed Messages

❒ **Inter-machine communication : message receive**

R.A.ECF
$$\frac{r \in \mathit{dln}(Q)}{\underline{a}.r\widetilde{V} \mid a(P)[Q] \longrightarrow P \mid \mathbf{i}(\underline{a}, r, \widetilde{V}))[Q]}$$

M.A.ECF.R
$$\frac{\begin{array}{c} l_\top \in \top.\mathcal{L} \quad O_0(\rho(b)) = \langle k, t\rangle \\ O(k) = \langle m, t'\rangle \quad O' = merge(O, O_0) \end{array}}{\begin{array}{c} m : N : O : L\{l_\top \overset{-}{\underset{-}{\longmapsto}} \langle \alpha, \mathcal{R}\rangle\} \parallel \\ E\{m' \to m \mapsto msg(\underline{b}.r, \widetilde{V}, \rho, O_0) :: \mathcal{M}\} \longrightarrow \\ m : N : O' : L\{l_\top \overset{-}{\underset{-}{\longmapsto}} \left\langle \alpha, \mathcal{R} :: (\rho, \mathbf{i}\ (\underline{b}, r, \widetilde{V}))\right\rangle\} \parallel \\ E\{m` \to m \mapsto \mathcal{M}\} \end{array}}$$

Cell Mobility

❒ Cell passivation

R.PASSIV

$$\frac{}{a(\mathsf{pass}_a\, V \mid P)[Q] \rightarrow V(\lambda.P)(\lambda.Q)}$$

PASS

$$\frac{\begin{array}{c}\rho(a) = \alpha \quad l \in \mathcal{L}_0 \quad \mathcal{L}_0' = \mathcal{L}_0 \setminus \{l\} \\ \langle \mathsf{reify}_{\mathsf{control}}(\mathcal{D}, \mathcal{H}, \mathcal{R}), \mathsf{reify}_{\mathsf{content}}(l_e, \mathcal{L}), L' \rangle = \\ extract(l, \mathcal{D}, \mathcal{H}, \mathcal{R}, l_e, \mathcal{L}, L)\end{array}}{\begin{array}{c} m : N : O : L\{l \xmapsto[l_e:\mathcal{L}]{\mathcal{D}:\mathcal{H}} \langle \alpha, (\rho, \mathsf{pass}_a\, V) :: \mathcal{R} \rangle, \\ l_0 \xmapsto[l':\mathcal{L}_0]{-} \langle \alpha_0, \mathcal{R}_0 \rangle, l' \xmapsto[-]{-} \langle \sharp, \mathcal{R}' \rangle\} \longrightarrow \\ m : N : O : L'\{l_0 \xmapsto[l':\mathcal{L}_0']{-} \langle \alpha_0, \mathcal{R}_0 \rangle, \\ l' \xmapsto[-]{-} \langle \sharp, \mathcal{R}' :: (\rho, V\, \mathsf{reify}_{\mathsf{control}}(\mathcal{D}, \mathcal{H}, \mathcal{R})\ \mathsf{reify}_{\mathsf{content}}(l_e, \mathcal{L})) \rangle\} \end{array}}$$

Cell Mobility

Cell activation

$$\frac{\begin{array}{c}\langle \mathcal{D}', \mathcal{H}', \mathcal{R}' \rangle = \\ insert_{flat}(\mathsf{reify}_{\mathsf{control}}(\mathcal{D}_m, \mathcal{H}_m, \mathcal{R}_m), \mathcal{D}, \mathcal{H}, \mathcal{R}) \\ O' = O\{\forall D \in \mathcal{D}_m \;\; \forall r \in dn(D) \textbf{ where} \\ O(\rho(r)) = \langle \beth, t_r \rangle, \rho(r) \mapsto \langle l, t_r \rangle\}\end{array}}{\begin{array}{c} m : N : O : L\{l \overset{\mathcal{D}:\mathcal{H}}{\underset{-}{\longmapsto}} \langle \alpha, (\rho, \mathsf{reify}_{\mathsf{control}}(\mathcal{D}_m, \mathcal{H}_m, \mathcal{R}_m)\ ()) :: \mathcal{R} \rangle\} \\ \longrightarrow \\ m : N : O' : L\{l \overset{\mathcal{D}':\mathcal{H}'}{\underset{-}{\longmapsto}} \langle \alpha, \mathcal{R}' \rangle\}\end{array}} \quad \textbf{CONTROL}$$

A Few Invariants

Invariant 1 **On each machine, locations are organized in a tree**

Invariant 2 **If a machine m hosts a location l, then the name of l is contained in the lookup service of m:**

$$\forall (m, l) \in (\textsc{MName} \times \textsc{LocNm}) \;\; l \in \mathit{dom}(m).L \Longrightarrow$$
$$\exists\, t \in \textsc{TimeStamp} \;\; \{l \mapsto \langle m, t\rangle\} \subset m.O$$

Invariant 3 **1. When a message targeted at a name a is received on a machine m, either there exists on m :**

(i) a location defininig a

(ii) a forwarder towards another machine

2. When a passivated cell passivated is sent from machines m towards m', then a forwarder $m \to m'$ is created locally

Additional Desirable Properties

Conjecture 1 **[Soundness] The routing algorithm R* is a sound update policy for the lookup service** $(O_m)_{m \in \textbf{MNAME}}$

Conjecture 2 **[Correctness]** $\forall m \in \textbf{MNAME}\ \forall (l, r) \in (\textbf{LOCNM} \times \textbf{REF}$ **such that** $l \in \mathit{dom}(m).L$ **and** $l.\mathcal{R} = \ldots :: (\rho, P) :: \ldots$ **and** $r \in \mathit{dom}(\rho)$:

$$l \Subset m.O(\rho(r)) \implies r \in \mathit{dln}(P)$$

Outline

- An Overview of the M-calculus
- The Abstract Machine
- ➨ An Implementation
- Towards a MIKADO Core Software Framework

C-VM : a Centralized Implementation

- **Requirements**
  - Portability and extensibility
  - Reasonable complexity
  - Code migration between machines
  - Interactive evaluation of M-calculus expressions
- **Structure of the run-time**
  - A compiler (OCaml) : M-calculus code $\longrightarrow$ byte-code
  - A byte-code interpreter (Java)

The C-VM byte-code

❒ Design

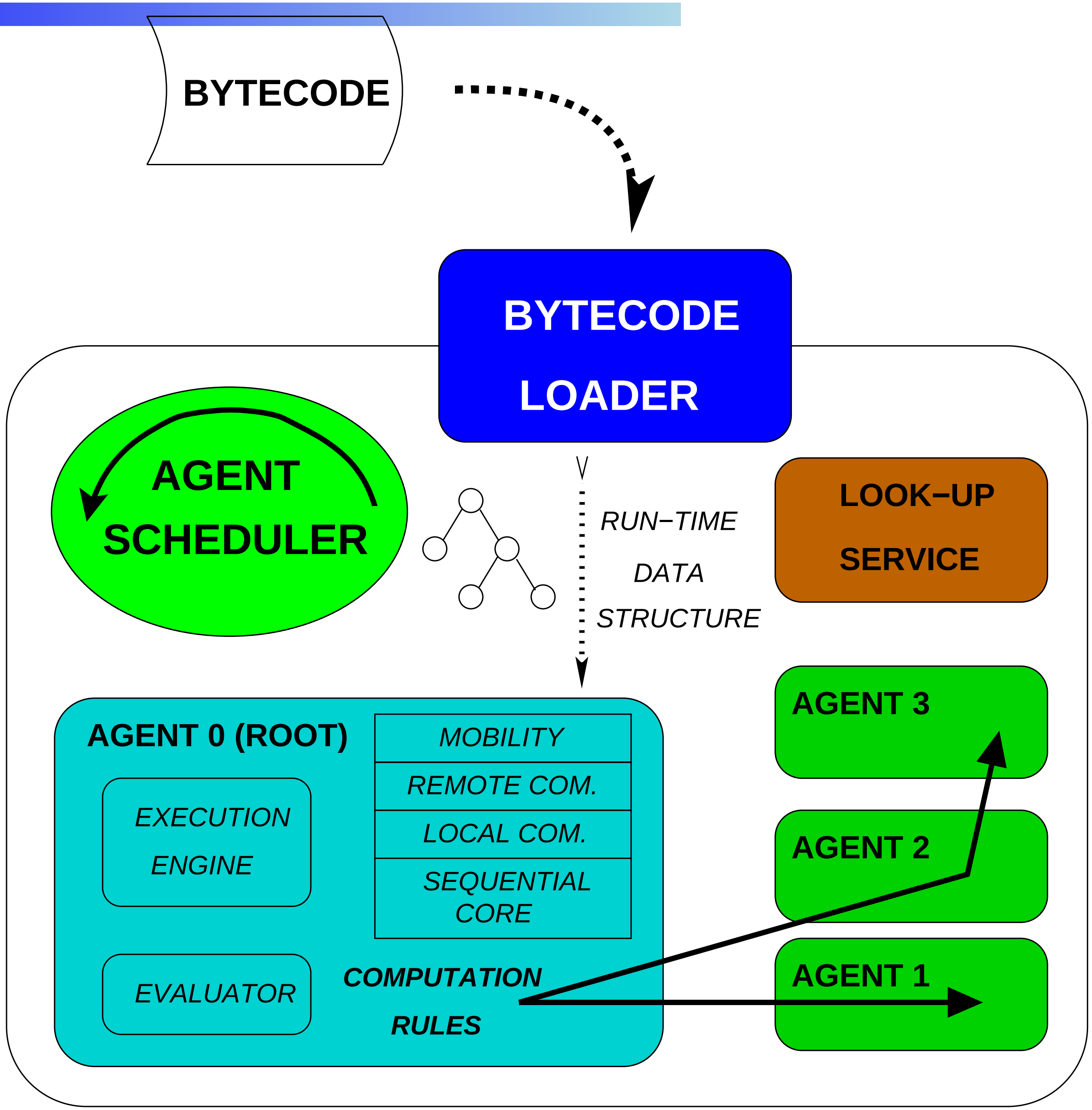
- Few opcodes
- Close to the calculus syntax
  - ➢ Easier proofs of correctness
  - ➢ Reflection of the nesting of process terms
  - ➨ Possible optimizations for VM internal data structures manipulations

❒ Organization

- Language constructs compiled to instructions
- An instruction = series of blocks
- Sample instructions

...	DOM	@(a)	@(P)	@(Q)	...	APP	@(P)	@(Q)	...

Structure of the VM



Summary

- **Conclusion**
  - **The M-calculus**
    - **A new distributed process calculus**
    - **Higher-order extension of the Join calculus with programmable localities, process mobility and dynamic binding**
  - **The CLAM**
    - **Supporting distributed abstract machine**
    - **No more complex than the JAM**
- **Related AM:** Nomadic Pict, PAN, DiTyCO, JAM
- **Current implementation efforts:**
  - **A distributed run-time**
  - **More efficient implementation**

Outline

- An Overview of the M-calculus
- The Abstract Machine
- An Implementation
- ➡ Towards a MIKADO Core Software Framework

Towards a Core Software Framework (CSF)

- **Aim:** a generic framework (CSF) to build execution support for domain-based calculi
- **Output of the WP3 Meeting** (September, Firenze)
  - **Requirements:**
    - **Genericity: a middleware allowing the implementation of existing platforms (PAN, Klava, DiTyCO, JAM, CLAM)**
    - **Portability: CSF = a set of Java interfaces and classes**
  - **Structure:**
    - **Computational Node Abstraction**
    - **Node Topology Management**
    - **Communication Protocols**
    - **Process Mobility**

Computational Node Abstraction

```
interface NodeIdentifier {
}

interface Node {
    public NodeIdentifier getNodeIdentifier();
    public Object getImplementation();
}

interface NodeIdentifierFactory {
    public NodeIdentifier newNodeIdentifier();
}
```

Topology Management

```
interface TopologyManager {
    public NodeIdentifier [] getSubNodes();
    public void addSubNode(NodeIdentifier id);
    public void removeSubNode(NodeIdentifier id);
    pblic Node getSubNode(NodeIdentifier id);

}
```

Communication Protocols

```
interface Identifier {
    public NamingContext getContext();
    public Object resolve();
    public void unexport();
    public Object bind();
    public byte [] encode();
}

interface NamingContext {
    public Identifier export(Object obj);
    public Identifier decode (byte [] data);
}

+ Sessions, Protocols, Connections, Marshallers, ...
```

- **To be discussed to produce deliverable D3.1.0...**