MiKO-Mikado Koncurrent Objects

Liliana Salvador

lsalvador@ncc.up.pt

University of Porto

Joint work with:

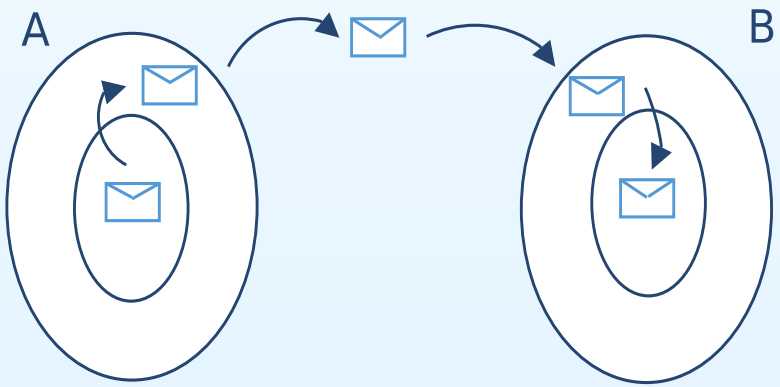
Francisco Martins	fmartins@di.fc.ul.pt	University of Azores
Vasco Vasconcelos	vv@di.fc.ul.pt	University of Lisbon
Luís Lopes	lblopes@ncc.up.pt	University of Porto

Outline

- Domains
- The MiKO Calculus
  - Syntax
  - Operational Semantics
  - Type System
- Examples
- Future/Ongoing work

Domains

- membrane
  - control of migration
  - message handling
    - with networks
    - with contents
- content
  - dinamically interacts with the membrane



The MiKO Calculus

- Based on the Mikado's membrane model
- Instance of a high order TyCO calculus
- Domains form a flat network

Syntax - Membranes/Contents

$$
\begin{array}{rcll}
S, P & ::= & \text{inaction} \mid P \mid Q \mid \text{new } c\ P \mid u?m \mid n\,!\,M & \\
& & \mid X[\tilde{V}] \mid AV \mid \text{def } \{X_i = A_i\}_{i \in I} \text{ in } P & \\
& & \mid \text{in}[P] & \text{(in)} \\
& & \mid \text{out}[a, M] & \text{(out)} \\
& & \mid \text{mkdom}[a, G, P] \text{ in } S & \text{(make domain)}
\end{array}
$$

channels : c domains: d variables : x definition ids: X

$$
\begin{array}{rcll}
A & ::= & (\tilde{x})S & \text{(abstractions)} \\
n & ::= & c \mid a \mid x & \text{(names)} \\
M & ::= & l[\tilde{V}] & \text{(messages)}
\end{array}
\qquad
\begin{array}{rcll}
V & ::= & n \mid A & \text{(values)} \\
u & ::= & x \mid c & \text{(receivers)} \\
m & ::= & \{l_i = A_i\}_{i \in I} & \text{(methods)}
\end{array}
$$

Syntax - Guardians and Networks

Guardians	G	$::=$		
			$m\,\{S\}$	(methods+membrane)
			$\mid$ new $c\;G$	(scope restriction)
Networks	N	$::=$		
			inaction	(inaction)
			$\mid a\{G\}[P]$	(domain)
			$\mid a\,!\,M$	(network message)
			$\mid N \mid N$	(parallel composition)
			$\mid$ new $t\;N$	(scope restriction)

m	$::=$	$\{l_i = A_i\}_{i \in I}$	(methods)
t	$::=$	$c \mid a$	(targets)
M	$::=$	$l[\tilde{V}]$	(messages)

Operational Semantics - Structural Congruence

- networks

  Works up to α-congruence and is the least congruence $\equiv$ on networks plus the next rules:

$$a\{G\}[\mathsf{new}\ c\ P] \equiv \mathsf{new}\ c\ a\{G\}[P] \qquad c \notin \mathrm{fn}(G)$$

$$a\{\widetilde{\mathsf{new}\ c\ m\{S\}}\}[P] \equiv a\{m\ \{\widetilde{\mathsf{new}\ c\ S}\}\}[P] \qquad \tilde{c} \notin \mathrm{fn}(S)$$

$$a\{\widetilde{\mathsf{new}\ c\ m\{S\}}\}[P] \equiv \widetilde{\mathsf{new}\ c\ a\{m\{S\}\}}[P] \qquad \tilde{c} \notin \mathrm{fn}(P)$$

- membranes/processes

  Works up to α-congruence and is the least congruence $\equiv$ on membranes/processes plus the standard rules of TyCO

Operational Semantics - Reduction Rules

- Reduction relation based on evaluation contexts

$$C \quad ::= \quad [\cdot] \mid (C \mid N) \mid \text{new } n\ C \mid \text{def } D \text{ in } C$$

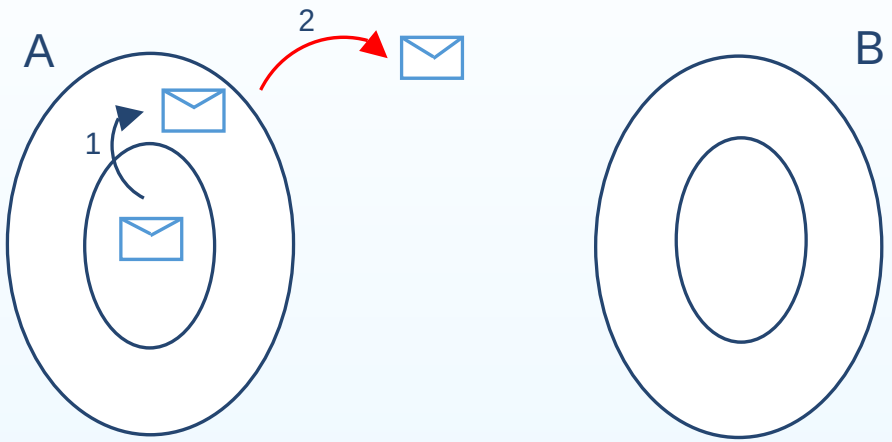
- Domain A sends a message to domain B (four steps of communication)
  - content $\rightarrow$ membrane
  - membrane $\rightarrow$ network
  - network $\rightarrow$ membrane
  - membrane $\rightarrow$ content

content $\longrightarrow$ membrane

$$a\{\widetilde{\text{new } c}\,\{l_i = A_i\}_{i\in I}\{S\}\}[C[a\,!\,l_j[\tilde{V}]]] \longrightarrow$$

$$a\{\widetilde{\text{new } c}\,\{l_i = A_i\}_{i\in I}\{A_j\tilde{V} \mid S\}\}[C[\text{inaction}]] \quad (1)$$

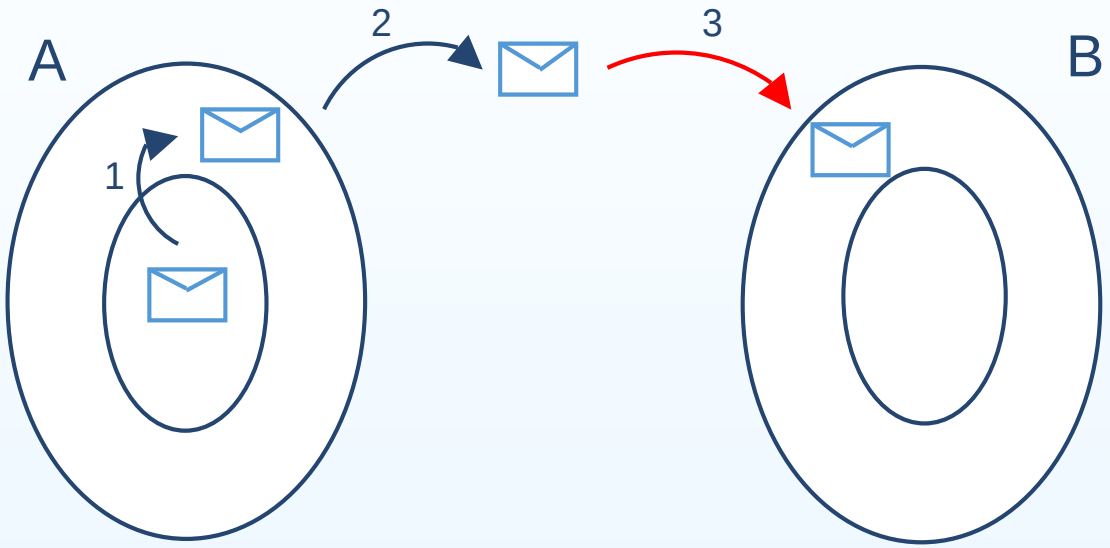
membrane $\longrightarrow$ network



$$a\{\widetilde{\text{new } c}\; m\{C[\text{out}[b, M]]\}\}[P] \longrightarrow$$

$$b\,!\,l_j[\tilde{V}] \mid a\{\widetilde{\text{new } c}\; m\{C[\text{inaction}]\}\}[P] \quad (2)$$

network $\longrightarrow$ membrane



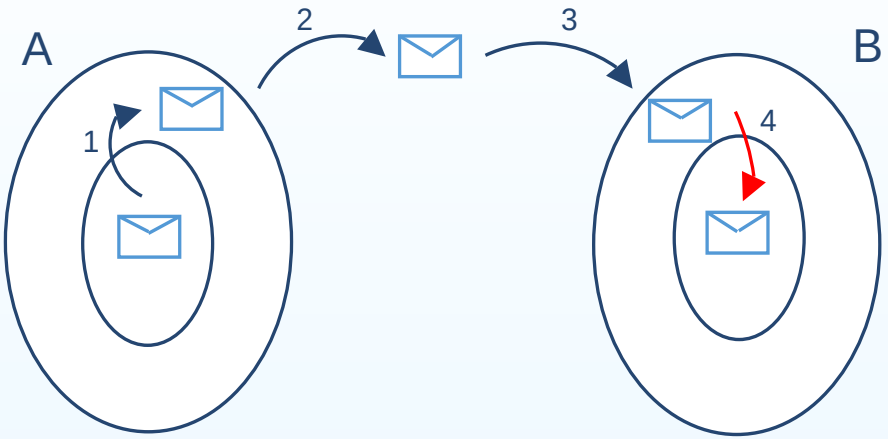
$$a\,!\,l_j[\tilde{V}] \mid a\{\widetilde{\text{new } c}\,\{l_i = A_i\}_{i\in I}\{S\}\}[P] \longrightarrow$$

$$\text{if } \{\Gamma \vdash \tilde{V} : \tilde{\alpha} \wedge \Gamma \vdash A_i : \tilde{\alpha} \longrightarrow \Diamond \wedge j \in I\}$$

$$\text{then } \{a\{\widetilde{\text{new } c}\,\{l_i = A_i\}_{i\in I}\{A_j\tilde{V} \mid S\}\}[P]\}$$

$$\text{else } \{a\{\widetilde{\text{new } c}\,\{l_i = A_i\}_{i\in I}\{S\}\}[P]\} \quad (3)$$

membrane $\longrightarrow$ content



$$a\{\widetilde{\text{new } c}\; m\{C[\text{in}[P]]\}\}[Q] \longrightarrow$$

$$a\{\widetilde{\text{new } c}\; m\{C[\text{inaction}]\}\}[P \mid Q]$$

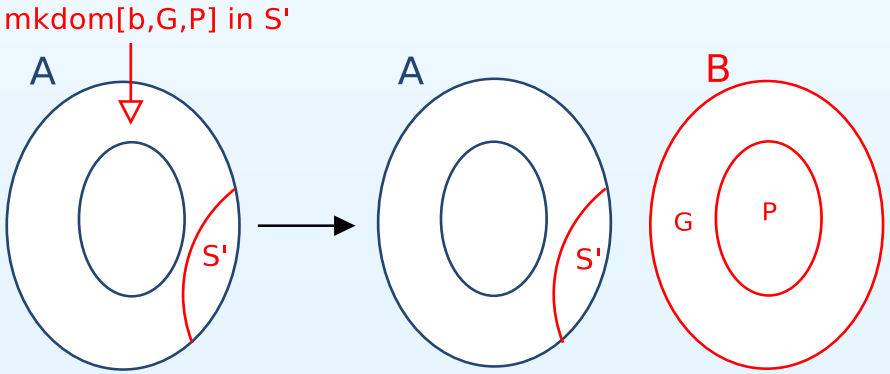
$$\text{if } \operatorname{fn}(P) \cap \tilde{c} = \emptyset \quad (4)$$

Creating a new domain

$$a\{\widetilde{\text{new } c}\; m\{C[\text{mkdom}[b, G, P] \text{ in } S']\}\}[Q] \rightarrow$$

$$\text{new } b\ (b\{G\}[P] \mid a\{\widetilde{\text{new } c}\; m\{C[S']\}\}[Q])$$

$$b \notin \text{fn}(Q), b \notin \text{fn}(C[\text{inaction}]) \quad (5)$$



Type System

- Syntax of types

$$\begin{array}{rcl} \tau & ::= & \gamma \mid \alpha \\ \gamma & ::= & \{l_i : \alpha_i\}_{i \in I} \\ \alpha & ::= & \gamma \mid \alpha \to \Diamond \end{array}$$

- Environments
  - networks: Γ

$$n : \gamma \qquad \text{names to types}$$

  - membranes/processes: Δ

$$\begin{array}{rr} n : \gamma & \text{names to types} \\ x : \alpha & \text{variables to types} \\ \mathcal{X} : \tilde{\alpha} & \text{def. ids to types} \end{array}$$

Type System - Cont.

- Judgements
  - networks

$$\Gamma \vdash N : \diamond$$

  - membranes/processes

$$\Delta \vdash S : \diamond$$

  - methods and messages

$$\Delta \vdash m : \gamma \qquad \Delta \vdash M : \gamma$$

  - values

$$\Delta \vdash V : \alpha$$

Typing Networks

- domain

$$\frac{\Gamma \vdash a : \gamma \quad \Gamma, \tilde{c} : \tilde{\gamma'} \vdash m : \gamma \quad \Gamma, \tilde{c} : \tilde{\gamma} \vdash S : \diamond \quad \Gamma \vdash P : \diamond}{\Gamma \vdash a\{\widetilde{\mathsf{new}\ c}\ m\{S\}\}[P] : \diamond}$$

- message in the network

$$\Gamma \vdash a\,!\,M : \diamond$$

Typing Guardians and Membranes

- methods

$$\frac{\Delta \vdash A_i : \alpha_i \to \Diamond}{\Delta \vdash \{l_i = A_i\}_{i \in I} : \{l_i : \alpha_i \to \Diamond\}_{i \in I}}$$

- in

$$\frac{\Delta \vdash P : \Diamond}{\Delta \vdash \mathsf{in}[P] : \Diamond}$$

- out

$$\Delta \vdash \mathsf{out}[a, M] : \Diamond$$

Typing new domains and messages

- new domain

$$\Delta, a : \gamma, \tilde{c} : \tilde{\gamma'} \vdash m : \gamma$$

$$\Delta, a : \gamma, \tilde{c} : \tilde{\gamma'} \vdash S : \diamond$$

$$\frac{\Delta, a : \gamma \vdash P : \diamond \quad \Delta, a : \gamma \vdash S' : \diamond}{\Delta \vdash \mathsf{mkdom}[a, \widetilde{\mathsf{new}\ c}\ m\{S\}, P]\ \mathsf{in}\ S' : \diamond}$$

- message

$$\frac{\Delta \vdash \tilde{V} : \tilde{\alpha_j}, \quad j \in I}{\Delta \vdash l_j[\tilde{V}] : \{l_i : \tilde{\alpha_i} \to \diamond\}_{i,j \in I}}$$

Example

```
a{ {
    enter(x)  = in [x []]
    exit(x,y) = out[x, enter[y]]
   }
   {
    inaction
   }
 } [ a! exit[b,P] ]
```

```
a{ {
    enter(x)  = in [x []]
    exit(x,y) = out[x, enter[y]]
   }
   {
    out[b, enter[P]]
   }
 }[inaction]
```

Example - cont

```
b!enter[P] | b{ {
                  enter(x)  = in[x []]
                  exit(x,y) = out[x, enter[y]]
                }
                {
                  inaction
                }
              } [inaction]
```

```
b { {
      enter(x)  = in[x []]
      exit(x,y) = out [x, enter[y]]
    }
    {
      in[P]
    }
  } [inaction]
```

Example

```
b { {
      enter(x)  = in [x []]
      exit(x,y) = out [x, enter[y]]
    }
    {
      inaction
    }
  } [P]
```

Future/Ongoing Work

- add the origin domain to outgoing messages
- extend the type system to cope with recursive types
- work on a type safety result

Structural Congruence

The structural congruence works up to α-congruence and is the least congruence $\equiv$ on networks plus the next rules:

- Networks

$$\mathsf{new}\ t\ N \mid R \equiv \mathsf{new}\ t\ (N \mid R) \qquad n \notin \mathrm{fn}(R) \quad \text{(N-SRC)}$$

$$a\{G\}[\mathsf{new}\ c\ P] \equiv \mathsf{new}\ c\ a\{G\}[P] \qquad c \notin \mathrm{fn}(G) \quad \text{(N-PSR)}$$

$$a\{\widetilde{\mathsf{new}\ c}\ m\{S\}\}[P] \equiv a\{m\ \{\widetilde{\mathsf{new}\ c}\ S\}\}[P] \qquad \tilde{c} \notin \mathrm{fn}(S) \quad \text{(N-MSR)}$$

$$a\{\widetilde{\mathsf{new}\ c}\ m\{S\}\}[P] \equiv \widetilde{\mathsf{new}\ c}\ a\{m\{S\}\}[P] \qquad \tilde{c} \notin \mathrm{fn}(P) \quad \text{(N-GSR)}$$

Structural Congruence

The structural congruence works up to α-congruence and is the least congruence $\equiv$ on controllers and processes plus the next rules:

- Membranes/Processes

$$\text{new } c\ P \mid Q \equiv \text{ new } c\ (P \mid Q) c \notin \mathrm{fn}(Q) \qquad \text{(M-SRPAR)}$$

$$\text{def } D \text{ in new } c\ P \equiv \text{ new } c \text{ def } D \text{ in } P c \notin \mathrm{fn}(D) \qquad \text{(M-SRDEF)}$$

$$\text{def } D \text{ in } P \mid Q \equiv \text{ def } D \text{ in } (P \mid Q)\ \mathrm{fId}(D) \cap \mathrm{fId}(Q) = \emptyset \qquad \text{(M-DEFPAR)}$$

Reduction Rules

- Networks

$$a\,!\,l_j[\tilde{V}] \mid a\{\widetilde{\mathsf{new}\ c}\,\{l_i = A_i\}_{i\in I}\{S\}\}[P] \rightarrow$$

$$\text{if } \{\Gamma \vdash \tilde{V} : \tilde{\alpha} \wedge \Gamma \vdash A_i : \tilde{\alpha} \rightarrow \Diamond \wedge j \in I\}$$

$$\text{then } \{a\{\widetilde{\mathsf{new}\ c}\,\{l_i = A_i\}_{i\in I}\{A_j\tilde{V} \mid S\}\}[P]\}$$

$$\text{else } \{a\{\widetilde{\mathsf{new}\ c}\,\{l_i = A_i\}_{i\in I}\{S\}\}[P]\} \quad \text{(N-COM)}$$

$$a\{\widetilde{\mathsf{new}\ c}\,m\{C[\mathsf{out}[b, M]l_j[\tilde{V}]]\}\}[P] \rightarrow$$

$$b\,!\,l_j[a, \tilde{V}] \mid a\{\widetilde{\mathsf{new}\ c}\,m\{C[\mathsf{inaction}]\}\}[P]$$

(N-OUT)

Reduction Rules

$$a\{\widetilde{\mathsf{new}\ c}\ m\{C[\mathsf{in}[P]]\}\}[Q] \rightarrow$$
$$a\{\widetilde{\mathsf{new}\ c}\ m\{C[\mathsf{inaction}]\}\}[P \mid Q]$$
$$\text{if } \mathrm{fn}(P) \cap \tilde{c} = \emptyset \qquad \text{(N-IN)}$$

$$a\{\widetilde{\mathsf{new}\ c}\ m\{C[\mathsf{mkdom}[b, G, P] \text{ in } S' \mid S'']\}\}[Q] \rightarrow$$
$$\mathsf{new}\ b\ (b\{G\}[P] \mid a\{\widetilde{\mathsf{new}\ c}\ m\{C[S']\}\}[Q])$$
$$b \notin \mathrm{fn}(Q), b \notin \mathrm{fn}(C[\mathsf{inaction}]) \qquad \text{(N-MKD)}$$

Reduction Rules

$$a\{\widetilde{\mathsf{new}\ c}\,\{l_i = A_i\}_{i\in I}\{S\}\}[C[a\,!\,l_j[\tilde{V}]]] \to a\{\widetilde{\mathsf{new}\ c}\,\{l_i = A_i\}_{i\in I}\{A_j\tilde{V} \mid S\}\}[C[\mathsf{inaction}]] \qquad \text{(N-SEND)}$$

$$\frac{S \to S'}{a\{\widetilde{\mathsf{new}\ c}\ m\{S\}\}[P] \to a\{\widetilde{\mathsf{new}\ c}\ m\{S'\}\}[P]} \qquad \text{(N-MEMB)}$$

Reduction Rules

$$\frac{P \to P'}{a\{G\}[P] \to a\{G\}[P']} \quad \text{(N-PROC)}$$

$$\frac{N \to N'}{N \mid R \to N' \mid R} \quad \text{(N-PAR)}$$

$$\frac{N \to N'}{\text{new } n\ N \to \text{new } n\ N'} \quad \text{(N-SRESC)}$$

$$\frac{N \equiv R \quad N \to N'}{R \to N'} \quad \text{(N-STRC)}$$

Reduction Rules

- Membranes/Processes

$$c?\{l_i = A_i\}_{i\in I} \mid c!\, l_j[\tilde{V}] \rightarrow A_j\tilde{V} \qquad \text{(M-COM)}$$

$$\text{def } \{X_i = A_i\}_{i\in I} \text{ in } X_j[\tilde{V}] \mid T \rightarrow \text{def } \{X_i = A_i\}_{i\in I} \text{ in } A_j\tilde{V} \mid T \qquad \text{(M-CALL)}$$

$$((\tilde{x})P)_j\tilde{V} \rightarrow P[\tilde{V}/\tilde{x_j}] \qquad \text{(M-SUBS)}$$

$$\frac{P \rightarrow P'}{P \mid Q \rightarrow P' \mid Q} \qquad \text{(M-PAR)}$$

$$\frac{P \rightarrow P'}{\text{new } n\ P \rightarrow \text{new } n\ P'} \qquad \text{(M-SRESC)}$$

$$\frac{P \rightarrow P'}{\text{def } D \text{ in } P \rightarrow \text{def } D \text{ in } P'} \qquad \text{(M-DEF)}$$

$$\frac{P \equiv Q \quad P \rightarrow P'}{Q \rightarrow P'} \qquad \text{(M-STRC)}$$

Type System

- Networks

$$\Gamma \vdash \mathsf{inaction} : \diamond \qquad \text{(TN-INAC)}$$

$$\frac{\Gamma \vdash a : \gamma \quad \Gamma, \tilde{c} : \tilde{\gamma} \vdash m : \gamma \quad \Gamma, \tilde{c} : \tilde{\gamma} \vdash S : \diamond \quad \Gamma \vdash P : \diamond}{\Gamma \vdash a\{\widetilde{\mathsf{new}\ c}\ m\{S\}\}[P] : \diamond} \qquad \text{(TN-NET)}$$

$$\Gamma \vdash a\,!\,M : \diamond \qquad \text{(TN-MSG)}$$

Type System - Networks

$$\frac{\Gamma \vdash N : \diamond \quad \Gamma \vdash R : \diamond}{\Gamma \vdash N \mid R : \diamond} \qquad \text{(TN-PAR)}$$

$$\frac{\Gamma, t : \gamma \vdash N : \diamond}{\Gamma \vdash \mathsf{new}\ t\ N : \diamond} \qquad \text{(TN-RESC)}$$

Type System

- Membranes/Processes

$$\Delta \vdash \text{inaction} : \diamond \qquad \text{(TM-INAC)}$$

$$\frac{\Delta \vdash P : \diamond \quad \Delta \vdash Q : \diamond}{\Delta \vdash P \mid Q : \diamond} \qquad \text{(TM-PAR)}$$

$$\frac{\Delta, c : \gamma \vdash P : \diamond}{\Delta \vdash \text{new } c\ P : \diamond} \qquad \text{(TM-RESC)}$$

$$\frac{\Delta \vdash u : \gamma \quad \Delta \vdash m : \gamma}{\Delta \vdash u?m : \diamond} \qquad \text{(TM-OBJ)}$$

$$\frac{\Delta \vdash A_i : \alpha_i \to \diamond}{\Delta \vdash \{l_i = A_i\}_{i \in I} : \{l_i : \alpha_i \to \diamond\}_{i \in I}} \qquad \text{(TM-METH)}$$

Type System - Membrane/Processes

$$\frac{\Delta \vdash n : \gamma \quad \Delta \vdash M : \gamma}{\Delta \vdash n\,!\,M : \diamond} \qquad \text{(TM-OUTP)}$$

$$\frac{\Delta \vdash \tilde{V} : \tilde{\alpha} \quad \Delta \vdash X : \tilde{\alpha} \to \diamond}{\Delta \vdash X[\tilde{V}] : \diamond} \qquad \text{(TM-INST)}$$

$$\frac{\Delta, X_i : \alpha_i \vdash A_i : \alpha_i \quad \Delta, X_i : \alpha_i \vdash P : \diamond}{\Delta \vdash \mathsf{def}\ \{X_i = A_i\}_{i \in I}\ \mathsf{in}\ P : \diamond} \qquad \text{(TM-DEF)}$$

$$\frac{\Delta \vdash A : \tilde{\alpha} \to \diamond \quad \Delta \vdash \tilde{V} : \tilde{\alpha}}{\Delta \vdash A\tilde{V} : \diamond} \qquad \text{(TM-APP)}$$

Type System - Membranes/Processes

$$\frac{\Delta \vdash P : \diamond}{\Delta \vdash \mathsf{in}[P] : \diamond} \quad \text{(TM-IN)}$$

$$\Delta \vdash \mathsf{out}[a, M] : \diamond \quad \text{(TM-OUT)}$$

$$\frac{\Delta, a : \gamma, \tilde{c} : \tilde{\gamma'} \vdash m : \gamma \qquad \Delta, a : \gamma, \tilde{c} : \tilde{\gamma'} \vdash S : \diamond \qquad \Delta, a : \gamma \vdash P : \diamond \quad \Delta, a : \gamma \vdash S' : \diamond}{\Delta \vdash \mathsf{mkdom}[a, \widetilde{\mathsf{new}\ c}\ m\{S\}, P]\ \mathsf{in}\ S' : \diamond} \quad \text{(TM-MKD)}$$

Type System - Messages

- Messages

$$\frac{\Delta \vdash \tilde{V} : \tilde{\alpha}_j, j \in I}{\Delta \vdash l_j[\tilde{V}] : \{l_i : \tilde{\alpha}_i \to \diamond\}_{i,j \in I}} \quad \text{(T-MSG)}$$

- Values

$$\Delta, x : \alpha \vdash x : \alpha \quad \text{(TV-VAR)}$$

$$\frac{\Delta, \tilde{x} : \tilde{\alpha} \vdash S : \diamond}{\Delta \vdash (\tilde{x})S : \diamond} \quad \text{(TV-ABS)}$$