Open Distributed Systems

- Addition of new components.
- Replacement of existing components.
- Changes in interconnections.

Actor Configurations

model open system components:

- set of individually named actors.
- messages "on-route".
- interface to environment:
  - receptionists
  - external actors
Synchronous vs Asynchronous Communication

- Ti-Calculus (and other process algebras such as CCS, CSP) take synchronous communication as a primitive.

- Actors assume asynchronous communication is more primitive.
COMMUNICATION MEDIUM

- In Pi-calculus, channels are explicitly modelled. Multiple processes can share a channel, potentially causing interference.

- In the actor model, the communication medium is not explicit. Actors (active objects) are first-class, history-sensitive entities with an explicit identity used for communication.
FAIRNESS

The actor model theory assumes fair computations:

1. message delivery is guaranteed.
2. individual actor computations are guaranteed to progress.

Fairness is very useful for reasoning about equivalences of actor programs but can be hard/expensive to guarantee; in particular when distribution and failures are considered.
Programming Languages Influenced by π-Calculus and Actors.

- Scheme '75
- Act1 '87
- Acore '87
- Rosette '89
- Obliq '94
- Erlang '93
- ABCL '90
- SALSA '99
- Amber '86
- Facile '89
- CML '91
- Pict '94
- Nonadic Pict '99
- JOCAML '99
Actor (Agent) Model

Actor

Thread

Message

Internal variables

Methods

State

Mailbox
1. Extend a functional language (\(\lambda\)-calculus\) with actor primitives

2. Define an operational semantics for actor configurations.

3. Study various notions of equivalence of actor expressions and configurations.

4. Assume fairness:
   - guaranteed message delivery.
   - individual actor progress.
\textbf{\LaTeX-calculus}

\textbf{Syntax}

\[ e ::= v \mid \lambda v. e \mid (e e) \]

\textbf{Example}

\[ (\lambda x. x \ 5) \]

\[ 5 \]

\[ \pi \]

\[ x \{5/x\} \]

\[ [5/x]x \]
\( \text{pr} (x, y) \) returns a pair containing \( x \) & \( y \).

\( \text{ispr} (z) \) returns \( \text{T} \) if \( z \) is a pair; \( \text{F} \) otherwise.

\( 1st \left( \text{pr} (x, y) \right) = x \)  \( 1^{st} \) returns
The first value of \( x \).

\( 2nd \left( \text{pr} (x, y) \right) = y \)  \( 2^{nd} \) returns
The second value.
Actor Primitives

send(a, v)

sends value v to actor a.

new(b)

creates a new actor with behavior b, and returns the identity/name of the newly created actor.

ready(b)

becomes ready to receive a new message with behavior b.
**ACTOR LANGUAGE EXAMPLE**

\[ b_5 = \text{rec}(x.y. a_. \text{seg}(\text{send}(x, y), \text{ready}(y))) \]

receives an actor name \( x \) and sends the number 5 to that actor, then it becomes ready to process new messages with the same behavior \( y \).

**SAMPLE USAGE**

\[ \text{send}(\text{new}(b_5), \text{a}) \]

**A SINK**

\[ \text{sink} = \text{rec}(2b. a.m. \text{ready}(b)) \]

an actor that disregards all messages.
cell = rec (A b. A c. A m,
    if (get?(m),
        seg (send (cust(m), c),
            ready (b(c))),
        if (set?(m),
            ready (b(contains(m))),
            ready (b(c))))))

Using the cell:

let u = new(cell(b)) in
    seg (send (a, mkset(1)),
        send (a, mkSet(2)),
        send (a, mkget(c)));
Exercises

1. Write get, cust, set?, contents, mkset, mkget to complete the reference cell example in the AMST actor language.

2. Modify B-cell to notify a customer when the cell value is updated (such as in the TT-calculus cell example).