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 "en-route".
- interface to environment:
  * receptionists
  * external actors

SYNCHRONOUS VS ASYNCHRONOUS COMMUNICATION

- Π-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 Π-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.

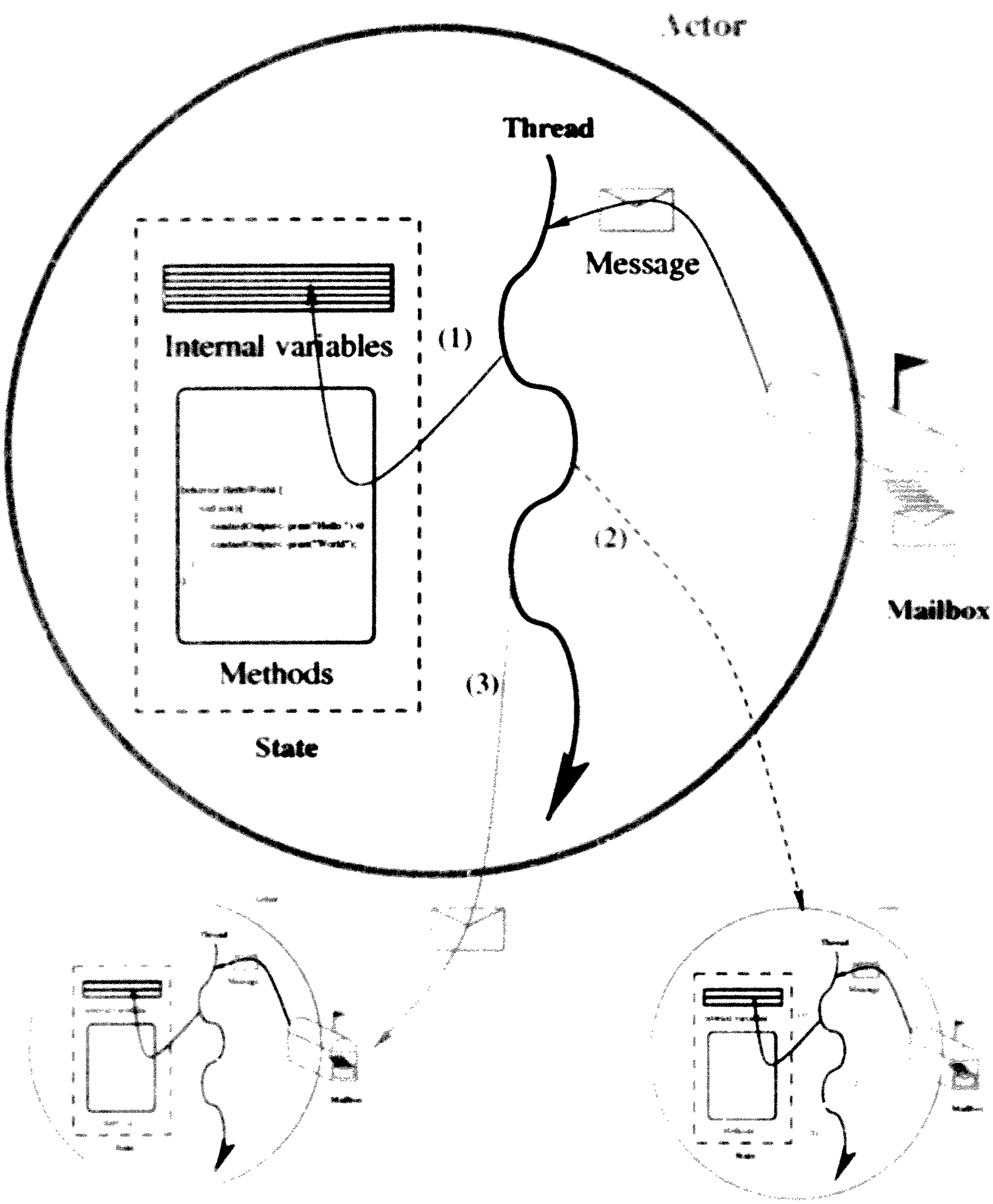
Fairness is very useful for reasoning about equivalences of actor programs but can be hard/expensive to guarantee; in particular when distributio and failures are considered.

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
- Nomadic Pict '99
- JOCAML '99

Actor (Agent) Model



AGHA, MASON, SMITH & TALCOTT

1. Extend a functional language (λ-Calculus + ifs + pairs) 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.

λ-CALCULUS

SYNTAX

$$
\begin{aligned}
e ::= \; & v \\
| \; & \lambda v . e \\
| \; & (e \; e)
\end{aligned}
$$

value
function abstracti
applicati

EXAMPLE

$((\lambda x . x) \; 5)$

5

$x \{5/x\}$ ← π

$[5/x] \, x$ ⇐ ?

$pr(x, y)$ returns a pair containing x & y.

$ispr(x)$ returns t if x is a pair; f otherwise

$1st(pr(x, y)) = x$ 1st returns the first value of a pair

$2nd(pr(x, y)) = y$ 2nd 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

$$b5 = rec(\lambda y.\lambda x.seq(send(x,5), ready(y)))$$

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

$$send(new(b5), a)$$

A SINK

$$sink = rec(\lambda b.\lambda m.ready(b))$$

an actor that disregards all messages.

REFERENCE CELL IN ACTOR LAMBDA


```
cell = rec (λb.λc.λm.
   if (get?(m),
         seq (send (cust(m), c),
                ready (b(c))),
         if (set? (m),
                ready (b(contents(m))),
                ready (b(c)))))
```

Using the cell:

```
let a = new(cell(0)) in
     seq ( send (a, mkset(7)),
           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 Bcell to notify a customer when the cell value is updated (such as in the π-calculus cell example).