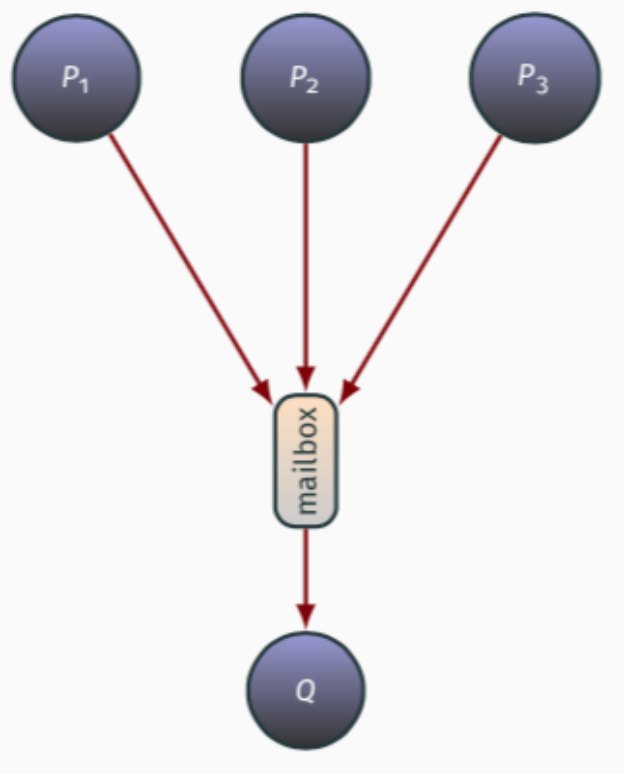
Mailbox Types for Unordered Interactions

Ugo de'Liguoro, Luca Padovani

Università di Torino

UBA - 26th February 2019, Buenos Aires

Static Analysis of Unordered Interactions



A popular communication model...

- **many-to-one** communications
- **selective input**
- used by **actors** (Akka, Erlang, CAF, ...)

...calling for a type system such that

- well-typed processes interact **safely**
- don't receive **unexpected** messages
- don't leave **garbage** behind
- don't **deadlock**

Scala listing from Savina

```
class Account(var balance: Double) extends ScalaActor[AnyRef] {
  private val self = this
  override def process(msg: AnyRef) {
    msg match {
      case dm: DebitMessage =>
        balance += dm.amount
        val sender = dm.sender.asInstanceOf[Account]
        sender.send(ReplyMessage.ONLY)
      case cm: CreditMessage =>
        balance -= cm.amount
        val sender    = cm.sender.asInstanceOf[ScalaActor[AnyRef]]
        val recipient = cm.recipient.asInstanceOf[Account]
        recipient.send(new DebitMessage(self, cm.amount))
        receive {
          case rm: ReplyMessage =>
            sender.send(ReplyMessage.ONLY)
        }
      case _: StopMessage => exit()
      case message =>
        val ex = new IllegalArgumentException("Unsupported message")
        ex.printStackTrace(System.err)
    }
  }
}
```

Key Ideas

1. Types describe **mailboxes** (not processes)
2. Subtyping embodies the **unordered** nature of mailboxes
3. Type judgments express **balance** of *obligations* and *expectations*
4. Well-typed processes **break even**

Syntax of Mailbox Calculus - MC

Asynchronous π-calculus + tagged messages + `fail`/`free`

Process	$P, Q ::=$	`done`	(termination)
		$\mathsf{X}[\overline{u}]$	(invocation)
		G	(guard)
		$u!\mathtt{m}[\overline{v}]$	(message)
		$P \mid Q$	(parallel)
		$(\nu a)P$	(mailbox)
Guard	$G, H ::=$	`fail` u	(exception)
		`free` $u\,.\,P$	(deallocation)
		$u?\mathtt{m}(\overline{x})\,.\,P$	(selective input)
		$G + H$	(external choice)

Reduction Semantics

Tags used to **select** received messages

$$a!\mathtt{m}[\overline{c}] \mid a?\mathtt{m}(\overline{x}).P + G \rightarrow P\{\overline{c}/\overline{x}\}$$

Empty mailboxes are explicitly **deallocated**

$$(\nu a)(\mathtt{free}\ a.P + G) \rightarrow P$$

Failure is relevant only when is the only guard:

$$\mathtt{fail}\ a + G \equiv G$$

Example: Locks

$$\begin{aligned}\mathsf{Idle}(\mathit{lock}) \triangleq\ & \mathtt{free}\ \mathit{lock}.\mathtt{done} \\ & + \mathit{lock}?\mathtt{acquire}(\mathit{user}).(\mathit{user}!\mathtt{reply}[\mathit{lock}] \mid \mathsf{Busy}[\mathit{lock}]) \\ & + \mathit{lock}?\mathtt{release}.\mathtt{fail}\ \mathit{lock}\end{aligned}$$

$$\mathsf{Busy}(\mathit{lock}) \triangleq \mathit{lock}?\mathtt{release}.\mathsf{Idle}[\mathit{lock}]$$

- a lock is either idle or busy
- an idle lock **can** be acquired, but **cannot** be released
- a busy lock **must** be released

Example: Locks (cont.)

$$\mathsf{Idle}(\mathit{lock}) \triangleq \mathtt{free}\ \mathit{lock}.\mathtt{done} + \mathit{lock}?\mathtt{acquire}(\mathit{user}).(\mathit{user}!\mathtt{reply}[\mathit{lock}] \mid \mathsf{Busy}[\mathit{lock}]) + \mathit{lock}?\mathtt{release}.\mathtt{fail}\ \mathit{lock}$$

$$\mathsf{Busy}(\mathit{lock}) \triangleq \mathit{lock}?\mathtt{release}.\mathsf{Idle}[\mathit{lock}]$$

$$\mathsf{User}(\mathit{user}, \mathit{lock}) \triangleq \mathit{lock}!\mathtt{acquire}[\mathit{user}] \mid \mathit{user}?\mathtt{reply}(l).(l!\mathtt{release} \mid \mathtt{free}\ \mathit{user}.\mathtt{done})$$

Example: Locks (cont.)

$$\mathsf{Idle}[\mathit{lock}] \mid \mathsf{User}[\mathit{alice}, \mathit{lock}]$$
$$\downarrow$$
$$(\cdots + \mathit{lock}?\mathtt{acquire}(\mathit{user}).(\cdots) + \cdots) \mid (\mathit{lock}!\mathtt{acquire}[\mathit{alice}] \mid \cdots)$$
$$\downarrow$$
$$\mathit{alice}!\mathtt{reply}[\mathit{lock}] \mid \mathsf{Busy}[\mathit{lock}] \mid \mathit{alice}?\mathtt{reply}(l).(\cdots)$$
$$\downarrow$$
$$\mathit{lock}?\mathtt{release}.\mathsf{Idle}[\mathit{lock}] \mid \mathit{lock}!\mathtt{release} \mid \mathtt{free}\ \mathit{alice}.\mathtt{done}$$
$$\downarrow$$
$$\mathsf{Idle}[\mathit{lock}] \mid \mathtt{free}\ \mathit{alice}.\mathtt{done}$$

hence

$$(\nu\ \mathit{lock}\ \mathit{alice})(\mathsf{Idle}[\mathit{lock}] \mid \mathsf{User}[\mathit{alice}, \mathit{lock}]) \rightarrow^*$$
$$(\nu\ \mathit{lock}\ \mathit{alice})(\mathsf{Idle}[\mathit{lock}] \mid \mathtt{free}\ \mathit{alice}.\mathtt{done}) \rightarrow$$
$$(\nu\ \mathit{lock}\ \mathit{alice})(\mathsf{Idle}[\mathit{lock}] \mid \mathtt{done}) \equiv (\nu\ \mathit{lock})(\mathsf{Idle}[\mathit{lock}])$$

Properties

Definition

P is *mailbox conformant* if $P \not\rightarrow^* \mathcal{C}[\texttt{fail}\ a]$

Exampe: non-conformant process

$$\mathsf{Idle}[lock] \mid lock\,!\texttt{release}$$

Definition

P is *deadlock free* if $P \rightarrow^* Q \not\rightarrow$ implies $Q \equiv \texttt{done}$

Example: conformant but deadlocking process

$$\begin{aligned}&\mathsf{Idle}[lock] \mid lock\,!\texttt{acquire}[user] \mid lock\,!\texttt{acquire}[user] \\ \mid\ & user?\texttt{reply}(l_1).user?\texttt{reply}(l_2).(l_1\,!\texttt{release} \mid l_2\,!\texttt{release})\end{aligned}$$

Syntax of Mailbox Types

$$
\begin{array}{rrcl}
\text{type} & \tau & ::= & \dagger E \\
\text{capability} & \dagger & ::= & ? \mid ! \\
\text{pattern} & E & ::= & \mathbb{0} \mid \mathbb{1} \mid \mathtt{m}[\overline{\tau}] \mid E + F \mid E \cdot F \mid E^*
\end{array}
$$

Capabilities

- $?$ = mailbox with **negative** balance (used for **inputs**)
- $!$ = mailbox with **positive** balance (used for **outputs**)

Patterns

- **commutative Kleene algebra** over message types $\mathtt{m}[\overline{\tau}]$
- describe the content of the mailbox

Pattern Semantics

$[\![E]\!]$ = set of *configurations* i.e. multisets of messages $\mathtt{m}[\overline{\tau}]$:

- $[\![\mathbb{0}]\!] = \varnothing$
- $[\![\mathbb{1}]\!] = \text{unit} = \{\langle\rangle\}$, where $\langle\rangle$ is the empty multiset
- $[\![\mathtt{m}[\overline{\tau}]]\!] = \{\langle \mathtt{m}[\overline{\tau}]\rangle\}$
- $[\![E + F]\!] = [\![E]\!] \cup [\![F]\!]$
- $[\![E \cdot F]\!] = [\![E]\!] \uplus [\![F]\!] = \{M \uplus N \mid M \in [\![E]\!], N \in [\![F]\!]\}$
- $[\![E^*]\!] = \text{fix}(\Phi_E) \quad \text{where} \quad \Phi_E(X) = \text{unit} \cup ([\![E]\!] \uplus X)$

Pattern Semantics (cont.)

$\mathbb{0}$	unreliable mailbox
$\mathbb{1}$	empty mailbox
$\mathtt{A} + \mathtt{B}$	mailbox containing either $\mathtt{A}$ or $\mathtt{B}$, not both
$\mathtt{A} \cdot \mathtt{B}$	mailbox containing both $\mathtt{A}$ and $\mathtt{B}$
$\mathtt{A}^*$	mailbox containing arbitrarily many (possibly zero) $\mathtt{A}$

in particular

$$[\![\mathtt{A} + \mathtt{B}]\!] = \{\langle \mathtt{A} \rangle, \langle \mathtt{B} \rangle\} \qquad [\![\mathtt{A} \cdot \mathtt{B}]\!] = \{\langle \mathtt{A}, \mathtt{B} \rangle\}$$

and

$$[\![\mathtt{A}^*]\!] = \{\langle\rangle, \langle \mathtt{A} \rangle, \langle \mathtt{A}, \mathtt{A} \rangle, \cdots\}$$

Type Semantics

$\sigma = !E$ represents an **obligation**:

$u : !E \vdash P$ P may store in u *some* configuration in $[\![E]\!]$

$\tau = ?E$ represents an **expectation**:

$u : ?E \vdash P$ P expects from u *any* configuration in $[\![E]\!]$

Typing Judgments

$$\Gamma \vdash P \quad \text{where} \quad \Gamma = \{u_1 : \tau_1, \ldots, u_n : \tau_n\}$$

Intuition

- Γ = messages **produced** by P − messages **consumed** by P

Consequences

- all mailboxes in Γ are **empty** $\iff$ P **breaks even**
- types in Γ are **preserved** by reductions

Type semantics: examples

judgement	behavior
$u : ?(\mathtt{A} \cdot \mathtt{B}) \vdash P$	P expects from u both $\mathtt{A}$ and $\mathtt{B}$
$u : !(\mathtt{A} \cdot \mathtt{B}) \vdash P$	P stores in u both $\mathtt{A}$ and $\mathtt{B}$
$u : ?(\mathtt{A} + \mathtt{B}) \vdash P$	P expects from u either $\mathtt{A}$ or $\mathtt{B}$
$u : !(\mathtt{A} + \mathtt{B}) \vdash P$	P stores in u either $\mathtt{A}$ or $\mathtt{B}$

Type semantics: examples (cont.)

judgement	behavior
$u : ?\mathbb{1} \vdash P$	P deallocates u
$u : !\mathbb{1} \vdash P$	P discards u
$u : ?\mathbb{0} \vdash P$	P fails
$u : !\mathbb{0} \vdash P$	—

Subtyping

$\sigma \leqslant \tau \iff$ any mailbox satisfying σ can be used where τ is expected

$$\frac{u : \tau \vdash P \quad \sigma \leqslant \tau}{u : \sigma \vdash P}$$

let $E \sqsubseteq F$ (roughly) mean $[\![E]\!] \subseteq [\![F]\!]$, then

$$\frac{E \sqsubseteq F}{!F \leqslant !E} \qquad \frac{E \sqsubseteq F}{?E \leqslant ?F}$$

Examples:

$$!(\mathtt{A} + \mathtt{B}) \leqslant !\mathtt{A} \qquad ?\mathtt{A} \leqslant ?(\mathtt{A} + \mathtt{B})$$

Typing Rules: subtyping

Given

$$\frac{}{u : !\mathbb{1}, \Gamma \leqslant \Gamma} \qquad \frac{\sigma \leqslant \tau}{u : \sigma, \Gamma \leqslant u : \tau, \Gamma}$$

from which we have a derived rule of restricted weakening:

$$\frac{\sigma \leqslant !\mathbb{1}}{u : \sigma, \Gamma \leqslant \Gamma}$$

then we have the **subtyping** rule

$$\frac{\Delta \vdash P \qquad \Gamma \leqslant \Delta}{\Gamma \vdash P}$$

Typing Rules: `done`, `fail`

$$\frac{}{\vdash \texttt{done}}$$

$$\frac{}{u : ?\mathbb{0}, \Gamma \vdash \texttt{fail}\ u}$$

Typing Rules: `free` and new

$$\frac{\Gamma \vdash P}{u : ?\mathbb{1}, \Gamma \vdash \texttt{free}\ u\,.\,P}$$

$$\frac{u : ?\mathbb{1}, \Gamma \vdash P}{\Gamma \vdash (\nu u)P}$$

Typing Rules: output and input

$$\frac{}{u : !\mathtt{m}[\overline{\tau}],\ \overline{v} : \overline{\tau} \vdash u!\mathtt{m}[\overline{v}]}$$

$$\frac{u : ?E,\ \overline{x} : \overline{\tau},\ \Gamma \vdash P}{u : ?(\mathtt{m}[\overline{\tau}] \cdot E),\ \Gamma \vdash u?\mathtt{m}(\overline{x}).P}$$

Typing Rules: branching

$$\frac{u : ?E, \Gamma \vdash G \quad u : ?F, \Gamma \vdash H}{u : ?(E + F), \Gamma \vdash G + H}$$

with side condition that:

$E + F$ is a sum of $\mathbb{0}$, $\mathbb{1}$ or of patterns $\mathtt{M} \cdot E'$ (normal form)

Typing Rules: branching (cont.)

Is this correct?

$$\frac{\dfrac{u : ?\mathtt{C},\ \Gamma \vdash P}{u : ?(\mathtt{A} \cdot \mathtt{C}),\ \Gamma \vdash u?\mathtt{A}.P} \qquad \dfrac{u : ?\mathtt{A},\ \Gamma \vdash Q}{u : ?(\mathtt{B} \cdot \mathtt{A}),\ \Gamma \vdash u?\mathtt{B}.Q}}{u : ?(\mathtt{A} \cdot \mathtt{C} + \mathtt{B} \cdot \mathtt{A}),\ \Gamma \vdash u?\mathtt{A}.P + u?\mathtt{B}.Q}$$

Typing Rules: branching (cont.)

Is this correct?

$$\frac{u : ?\mathtt{C},\ \Gamma \vdash P}{u : ?(\mathtt{A} \cdot \mathtt{C}),\ \Gamma \vdash u?\mathtt{A}.P} \qquad \frac{u : ?\mathtt{A},\ \Gamma \vdash Q}{u : ?(\mathtt{B} \cdot \mathtt{A}),\ \Gamma \vdash u?\mathtt{B}.Q}$$

$$\frac{}{u : ?(\mathtt{A} \cdot \mathtt{C} + \mathtt{B} \cdot \mathtt{A}),\ \Gamma \vdash u?\mathtt{A}.P + u?\mathtt{B}.Q}$$

NO: the type $\mathtt{C}$ of u in P does not describe u after receiving $\mathtt{A}$ in $u?\mathtt{A}.P + u?\mathtt{B}.Q$; indeed

$$(\mathtt{A} \cdot \mathtt{C} + \mathtt{B} \cdot \mathtt{A})/\mathtt{A} = \mathtt{C} + \mathtt{B} \neq \mathtt{C}$$

This is fixed by taking the equivalent pattern

$$\mathtt{A} \cdot (\mathtt{B} + \mathtt{C}) + \mathtt{B} \cdot \mathtt{A}$$

Typing Rules: parallel composition

$$\frac{u : !E \vdash P \qquad u : !F \vdash Q}{u : !(E \cdot F) \vdash P \mid Q}$$

$$\frac{u : !E \vdash P \qquad u : ?(E \cdot F) \vdash Q}{u : ?F \vdash P \mid Q}$$

but parallel of input actions is **undefined** (a mailbox has a unique owner). In general

$$\frac{\Gamma \vdash P \qquad \Delta \vdash Q}{\Gamma \| \Delta \vdash P \mid Q}$$

where $\Gamma \| \Delta$, acting pointwise on name typings, must be defined

Typing locks

$$\begin{aligned}\mathsf{Idle}(\mathit{lock}) \triangleq\ & \mathtt{free}\ \mathit{lock}.\mathtt{done} \\ & + \mathit{lock}?\mathtt{acquire}(\mathit{user}).(\mathit{user}!\mathtt{reply}[\mathit{lock}] \mid \mathsf{Busy}[\mathit{lock}]) \\ & + \mathit{lock}?\mathtt{release}.\mathtt{fail}\ \mathit{lock}\end{aligned}$$

$$\mathsf{Busy}(\mathit{lock}) \triangleq \mathit{lock}?\mathtt{release}.\mathsf{Idle}[\mathit{lock}]$$

Example: mailbox type of an idle lock

$$?\mathtt{acquire}[!\mathtt{reply}[!\mathtt{release}]]^*$$

Example: mailbox type of a busy lock

$$?(\mathtt{release} \cdot \mathtt{acquire}[!\mathtt{reply}[!\mathtt{release}]]^*)$$

Subject reduction

$$\dfrac{u : !\mathtt{m} \vdash u!\mathtt{m} \qquad \dfrac{\dfrac{\vdash \mathtt{done}}{u : ?\mathbb{1} \vdash \mathtt{free}\ u.\mathtt{done}}}{u : ?(\mathtt{m} \cdot \mathbb{1}) \vdash u?\mathtt{m}.\mathtt{free}\ u.\mathtt{done}}}{u : ?\mathbb{1} \vdash u!\mathtt{m} \mid (u?\mathtt{m}.\mathtt{free}\ u.\mathtt{done})}$$

Now

$$u!\mathtt{m} \mid (u?\mathtt{m}.\mathtt{free}\ u.\mathtt{done}) \rightarrow \mathtt{free}\ u.\mathtt{done} \not\rightarrow$$

and clearly $u : ?\mathbb{1} \vdash \mathtt{free}\ u.\mathtt{done}$ is derivable

Subject reduction (cont.)

Actually

$$(\nu u)(u!\mathtt{m} \mid (u?\mathtt{m}.\mathtt{free}\ u.\mathtt{done})) \rightarrow (\nu u)(\mathtt{free}\ u.\mathtt{done}) \rightarrow \mathtt{done}$$

in which case by rule [new] we have

$$\frac{u : ?\mathbb{1} \vdash u!\mathtt{m} \mid (u?\mathtt{m}.\mathtt{free}\ u.\mathtt{done})}{\vdash (\nu u)(u!\mathtt{m} \mid (u?\mathtt{m}.\mathtt{free}\ u.\mathtt{done}))}$$

and both $\vdash (\nu u)(\mathtt{free}\ u.\mathtt{done})$ and $\vdash \mathtt{done}$ are derivable

Subject Reduction and Conformance

Theorem

If $\Gamma \vdash P$ *and* $P \to Q$, *then* $\Gamma \vdash Q$

With other behavioral type systems (e.g. session types) you just have

$$\text{if} \quad \Gamma \vdash P \quad \text{and} \quad P \to Q \quad \text{then} \quad \Delta \vdash Q \quad \text{for some } \Delta$$

A closed P is **conformant** if $P \twoheadrightarrow^* Q \equiv (\nu u)(\texttt{fail}\ u \mid \cdots)$

Corollary

If $\vdash P$ *then* P *is conformant*

Deadlock

The term

$$(\nu u)(\nu v)(u?\mathtt{m}.\mathtt{free}\ u.v!\mathtt{m} \mid v?\mathtt{m}.\mathtt{free}\ v.u!\mathtt{m}) \nrightarrow$$

is typable and hence conformant, but **deadlocked**

Definition (mailbox dependency)

There is a **dependency** between mailboxes u and v if either

- v occurs in the continuation of a process blocked on u
- v occurs in a message stored in u

Dependency Graphs

Dependency graphs are unordered graphs plus local names:

$$\varphi ::= \varnothing \mid \{u, v\} \mid \varphi \sqcup \varphi \mid (\nu u)\varphi$$

Rules are extended to derive judgments

$$\Gamma \vdash P :: \varphi \quad \text{where } \varphi \text{ is acyclic}$$

Theorem

If $\Gamma \vdash P :: \varphi$ then P is conformant and deadlock free

Conclusion and research directions

Achievements:

- a notion of type inspired to regular language theory
- a simple type assignment system for an asynchronous π-calculus where types are invariant properties of names (mailboxes)
- the system is decidable and there is a type reconstruction algorithm (actually we have a tool)
- the system ensures safety and (some) liveness properties

Future work:

- find a mathematical model of types and typing judgments
- relate the system to logic (e.g. linear logic) if it makes sense
- scale the type system to concrete programming languages and libraries, like Java and Scala Akka library

References

- Paper: U. de'Liguoro, L. Padovani, *Mailbox Types for Unordered Interactions*, ECOOP 2018

- Tool: L. Padovani, *MC*^{2} - *Mailbox Calculus Checker*, `http://www.di.unito.it/~padovani/Software/MCC/`