A Logic for Mailboxes

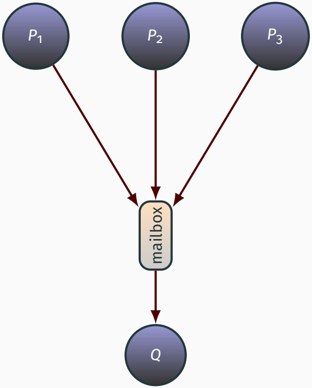
Ornela Dardha and Luca Padovani

BEHAPI
Behavioural Application Program Interfaces

Which message do you read first?

Time	Subject
4:00am	Increase your H-index
5:30am	Review request
6:00am	[ECOOP 2018] Notification

Selective message processing



Context

- **many-to-one** communications
- **unpredictable** message order
- messages **selected** by tag, type, shape, ...

Examples

- **actor model**
  Akka, Pony, Erlang, CAF, ...
- **concurrent objects**
  locks, futures, semaphores, ...

A type system for mailbox interactions

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

Addressing "impure" actors to some extent

> *"We studied 15 large, mature, and actively maintained actor programs written in Scala and found that **80% of them mix the actor model with another concurrency model**"*
>
> *Tasharofi et al. [2013]*

Syntax of the Mailbox Calculus

Asynchronous π-calculus + tagged messages + $\texttt{fail}/\texttt{free}$

Process

$$
\begin{array}{rcl}
P, Q & ::= & \texttt{done} \\
& | & \texttt{X}[\overline{u}] \\
& | & G \\
& | & u\,!\,\texttt{m}[\overline{v}] \\
& | & P \,|\, Q \\
& | & (\nu a)P
\end{array}
$$

Guard

$$
\begin{array}{rcl}
G, H & ::= & \texttt{fail}\ u \\
& | & \texttt{free}\ u\,.\,P \\
& | & u\,?\,\texttt{m}(\overline{x})\,.\,P \\
& | & G + H
\end{array}
$$

A simple example: the lock

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

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

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

Properties

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

Example (non-conformant process)

$$\mathsf{Idle}(lock) \mid lock!\texttt{release}$$

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

Example (conformant but deadlocking process)

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

Mailbox Types

type τ	process	
$?A$	provide one A	
$!A$	consume one A	
$?(A \cdot B)$	consume both A and B	internally ordered
$!(A \cdot B)$	provide both A and B	externally ordered
$?(A + B)$	consume either A or B	externally chosen
$!(A + B)$	provide either A and B	internally chosen
$?\mathbb{1}$	consume nothing	
$!\mathbb{1}$	provide nothing	
$?\mathbb{0}$	throw exception	
$!\mathbb{0}$	—	
$?A^*$	consume some As	externally chosen
$!A^*$	provide some As	internally chosen

Typing Judgments

$$\Gamma \vdash P$$

Intuition

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

Consequence

- all types in Γ are $?\mathbb{1} \iff P$ **breaks even**

Example of typing derivation 1

$$\dfrac{\dfrac{}{u : \,!A \vdash u\,!A} \qquad \dfrac{\dfrac{}{u : \,!B \vdash u\,!B} \qquad \dfrac{\dfrac{\vdots}{u : \,?B \vdash u?B.P}}{u : \,?A \cdot B \vdash u?A.u?B.P}}{u : \,?A \vdash u\,!B \mid u?A.u?B.P}}{u : \,?\mathbb{1} \vdash u\,!A \mid (u\,!B \mid u?A.u?B.P)}$$

Example of typing derivation 2

$$
\dfrac{\dfrac{\dfrac{}{u : !\mathsf{B} \vdash u!\mathsf{B}} \qquad \dfrac{}{u : !\mathsf{A} \vdash u!\mathsf{A}}}{u : !\mathsf{B} \cdot \mathsf{A} \vdash u!\mathsf{B} \mid u!\mathsf{A}} \qquad \dfrac{\dfrac{\vdots}{u : ?\mathsf{B} \vdash u?\mathsf{B}.P}}{u : ?\mathsf{A} \cdot \mathsf{B} \vdash u?\mathsf{A}.u?\mathsf{B}.P}}{u : ?\mathbb{1} \vdash (u!\mathsf{B} \mid u!\mathsf{A}) \mid u?\mathsf{A}.u?\mathsf{B}.P}
$$

Example of typing derivation 2

$$\dfrac{\dfrac{\dfrac{}{u : \,!\mathsf{B} \vdash u!\mathsf{B}} \qquad \dfrac{}{u : \,!\mathsf{A} \vdash u!\mathsf{A}}}{u : \,!\mathsf{A} \cdot \mathsf{B} \vdash u!\mathsf{B} \mid u!\mathsf{A}} \qquad \dfrac{\dfrac{\vdots}{u : ?\mathsf{B} \vdash u?\mathsf{B}.P}}{u : ?\mathsf{A} \cdot \mathsf{B} \vdash u?\mathsf{A}.u?\mathsf{B}.P}}{u : ?\mathbb{1} \vdash (u!\mathsf{B} \mid u!\mathsf{A}) \mid u?\mathsf{A}.u?\mathsf{B}.P}$$

Lock's mailbox

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

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

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

$$\texttt{?release} \cdot \texttt{acquire}[\cdots]^*$$

Lock's mailbox

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

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

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

$$?\mathtt{release} \cdot \mathtt{acquire}[\cdots]^*$$

$$?\mathtt{acquire}^* = ?\mathbb{1} + \mathtt{acquire} \cdot \mathtt{acquire}^* + \mathtt{release} \cdot \mathbb{0}$$

Properties of Well-Typed Processes

Theorem (conformance)
If $\Gamma \vdash P$, then P is mailbox conformant

Lemma (type preservation)
If $\Gamma \vdash P$ and $P \to Q$, then $\Gamma \vdash Q$

Remark
Types in Γ are **preserved**, also when the type of a mailbox **isn't**

This process is **mailbox conformant** but **deadlocks**

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

On deadlocks and mailbox dependencies

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

Strategy

1. collect **mailbox dependencies** in a graph φ

$$\Gamma \vdash P :: \varphi$$

2. make sure the graph has **no cycles**

Properties of well-typed processes, strengthened

Theorem (deadlock freedom)
If $\emptyset \vdash P :: \varphi$, then P is deadlock free

Theorem (fair termination)
If $\emptyset \vdash P :: \varphi$ for P finitely unfolding, then $P \rightarrow^* Q$ implies $Q \rightarrow^* \mathtt{done}$

Corollary (garbage freedom)
*Closed, well-typed, finitely-unfolding processes **leave no garbage***

Concluding Remarks

An interpretation of first-order mailbox types into μMALL

E	$\widehat{!E}$	$\widehat{?E}$
$\mathbb{0}$	0	$\top$
$\mathbb{1}$	1	$\bot$
$\mathtt{m}$	$\mathtt{m}$	$\mathtt{m}^\bot$
$E + F$	$\widehat{!E} \oplus \widehat{!F}$	$\widehat{?E} \mathbin{\&} \widehat{?F}$
$E \cdot F$	$\widehat{!E} \otimes \widehat{!F}$	$\widehat{?E} \parr \widehat{?F}$

- $\sigma \leq \tau$ iff $\vdash \widehat{\tau}^\bot, \widehat{\sigma}$ derivable in (one sided) μMALL
- typing rules for mailbox calculus $\sim$ inference rules for μMALL

Unresolved issues

- interpretation of **higher-order** mailbox types
- relationship between **reduction** and **cut elimination**

An interpretation of first-order mailbox types into μMALL

E	$\widehat{!E}$	$\widehat{?E}$
$\mathbb{0}$	0	$\top$
$\mathbb{1}$	1	$\bot$
$\mathtt{m}$	$\mathtt{m}$	$\mathtt{m}^\bot$
$E + F$	$\widehat{!E} \oplus \widehat{!F}$	$\widehat{?E} \mathbin{\&} \widehat{?F}$
$E \cdot F$	$\widehat{!E} \otimes \widehat{!F}$	$\widehat{?E} \parr \widehat{?F}$
E^*	$\mu X.1 \oplus \widehat{!E} \oplus (X \otimes X)$	$\nu X.\bot \mathbin{\&} \widehat{?E} \mathbin{\&} (X \parr X)$

- $\sigma \leq \tau$ iff $\vdash \widehat{\tau}^\bot, \widehat{\sigma}$ derivable in (one sided) μMALL
- typing rules for mailbox calculus $\sim$ inference rules for μMALL

Unresolved issues

- interpretation of **higher-order** mailbox types
- relationship between **reduction** and **cut elimination**

Wrap up

- mailbox calculus $\sim$ actors with **first-class/multiple** mailboxes
- mailbox types $\sim$ descriptions of **unordered** mailboxes

In the paper **[De'Liguoro and Padovani, 2018]**

- more examples (actors using **futures**, master-workers)
- encoding of **binary sessions** with **joins** and **forks**

Proof-of-concept implementation available

- subtyping can be as complex as validity of Presburger formulas
- potentially **lots** of type annotations, **Newtonian program analysis** to the rescue [Esparza et al., 2010]

References

Ugo De'Liguoro and Luca Padovani. Mailbox Types for Unordered Interactions. Technical report, Università di Torino, 2018. URL `https://arxiv.org/abs/1801.04167`.

Javier Esparza, Stefan Kiefer, and Michael Luttenberger. Newtonian program analysis. *J. ACM*, 57(6):33:1–33:47, November 2010. ISSN 0004-5411.

Samira Tasharofi, Peter Dinges, and Ralph E. Johnson. Why do scala developers mix the actor model with other concurrency models? In *Proceedings of ECOOP'13*, LNCS 7920, pages 302–326. Springer, 2013.