CauDEr: A Causal-Consistent Reversible Debugger for Erlang

Adrián Palacios
(joint work with Ivan Lanese, Naoki Nishida and Germán Vidal)

Technical University of Valencia

FLOPS 2018

May 11, 2018
Nagoya, Japan

Motivation

Facts about debugging:

- Developers spend 50% of their programming time searching and fixing bugs.
- Global cost of debugging is estimated to be around $312 billions anually.
- Cost of debugging increases with software complexity (size, concurrent).

However, very little research on debugging, especially for concurrent sw

⇒ **Novel approach to debug concurrent message-passing programs**

Motivation

Facts about debugging:

- Developers spend 50% of their programming time searching and fixing bugs.
- Global cost of debugging is estimated to be around \$312 billions anually.
- Cost of debugging increases with software complexity (size, concurrent).

However, very little research on debugging, especially for concurrent sw

⇒ **Novel approach to debug concurrent message-passing programs**

Erlang

Erlang's features

Main features of Erlang:

- integration of functional and concurrent features
- concurrency model based on asynchronous message-passing
- dynamic typing
- hot code loading

These features make it appropiate for distributed, fault-tolerant applications (Facebook, WhatsApp)

Erlang syntax

We consider a subset of Erlang with this syntax:

$$
\begin{array}{rcl}
\mathit{Module} & ::= & \mathrm{module}\ \mathit{Atom} = \mathit{fun}_1, \ldots, \mathit{fun}_n \\
\mathit{fun} & ::= & \mathit{fname} = \mathrm{fun}\ (X_1, \ldots, X_n) \rightarrow \mathit{expr} \\
\mathit{fname} & ::= & \mathit{Atom} / \mathit{Integer} \\
\mathit{lit} & ::= & \mathit{Atom} \mid \mathit{Integer} \mid \mathit{Float} \mid [\,] \\
\mathit{expr} & ::= & \mathit{Var} \mid \mathit{lit} \mid \mathit{fname} \mid [\mathit{expr}_1 | \mathit{expr}_2] \mid \{\mathit{expr}_1, \ldots, \mathit{expr}_n\} \\
 & \mid & \mathrm{call}\ \mathit{expr}\ (\mathit{expr}_1, \ldots, \mathit{expr}_n) \mid \mathrm{apply}\ \mathit{expr}\ (\mathit{expr}_1, \ldots, \mathit{expr}_n) \\
 & \mid & \mathrm{case}\ \mathit{expr}\ \mathrm{of}\ \mathit{clause}_1; \ldots; \mathit{clause}_m\ \mathrm{end} \\
 & \mid & \mathrm{let}\ \mathit{Var} = \mathit{expr}_1\ \mathrm{in}\ \mathit{expr}_2 \mid \mathrm{receive}\ \mathit{clause}_1; \ldots; \mathit{clause}_n\ \mathrm{end} \\
 & \mid & \mathrm{spawn}(\mathit{expr}, [\mathit{expr}_1, \ldots, \mathit{expr}_n]) \mid \mathit{expr}_1\ !\ \mathit{expr}_2 \mid \mathrm{self}() \\
\mathit{clause} & ::= & \mathit{pat}\ \mathrm{when}\ \mathit{expr}_1 \rightarrow \mathit{expr}_2 \\
\mathit{pat} & ::= & \mathit{Var} \mid \mathit{lit} \mid [\mathit{pat}_1 | \mathit{pat}_2] \mid \{\mathit{pat}_1, \ldots, \mathit{pat}_n\}
\end{array}
$$

Erlang syntax

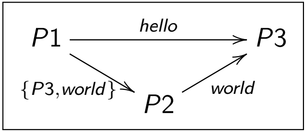
We consider a subset of Erlang with this syntax:

$$
\begin{array}{rcl}
\mathit{Module} & ::= & \mathsf{module}\ \mathit{Atom} = \mathit{fun}_1, \ldots, \mathit{fun}_n \\
\mathit{fun} & ::= & \mathit{fname} = \mathsf{fun}\ (X_1, \ldots, X_n) \rightarrow \mathit{expr} \\
\mathit{fname} & ::= & \mathit{Atom}/\mathit{Integer} \\
\mathit{lit} & ::= & \mathit{Atom} \mid \mathit{Integer} \mid \mathit{Float} \mid [\,] \\
\mathit{expr} & ::= & \mathit{Var} \mid \mathit{lit} \mid \mathit{fname} \mid [\mathit{expr}_1 | \mathit{expr}_2] \mid \{\mathit{expr}_1, \ldots, \mathit{expr}_n\} \\
 & \mid & \mathsf{call}\ \mathit{expr}\ (\mathit{expr}_1, \ldots, \mathit{expr}_n) \mid \mathsf{apply}\ \mathit{expr}\ (\mathit{expr}_1, \ldots, \mathit{expr}_n) \\
 & \mid & \mathsf{case}\ \mathit{expr}\ \mathsf{of}\ \mathit{clause}_1; \ldots; \mathit{clause}_m\ \mathsf{end} \\
 & \mid & \mathsf{let}\ \mathit{Var} = \mathit{expr}_1\ \mathsf{in}\ \mathit{expr}_2 \mid \mathsf{receive}\ \mathit{clause}_1; \ldots; \mathit{clause}_n\ \mathsf{end} \\
 & \mid & \mathsf{spawn}(\mathit{expr}, [\mathit{expr}_1, \ldots, \mathit{expr}_n]) \mid \mathit{expr}_1\ !\ \mathit{expr}_2 \mid \mathsf{self}() \\
\mathit{clause} & ::= & \mathit{pat}\ \mathsf{when}\ \mathit{expr}_1 \rightarrow \mathit{expr}_2 \\
\mathit{pat} & ::= & \mathit{Var} \mid \mathit{lit} \mid [\mathit{pat}_1 | \mathit{pat}_2] \mid \{\mathit{pat}_1, \ldots, \mathit{pat}_n\}
\end{array}
$$

Hello, World!

$$\begin{aligned} main/0 = \text{fun}\ () \to\ & \text{let}\ P2 = \text{spawn}(echo/0, [\,]) \\ & \text{in let}\ P3 = \text{spawn}(target/0, [\,]) \\ & \text{in let}\ \_ = P3\ !\ hello \\ & \text{in let}\ P2\ !\ \{P3, world\} \end{aligned}$$

$$\begin{aligned} target/0 = \text{fun}\ () \to\ & \text{receive} \\ & \quad A \to \text{receive} \\ & \quad\quad\quad B \to \{A, B\} \\ & \quad\quad \text{end} \\ & \text{end} \end{aligned}$$

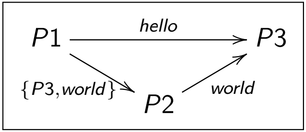


$$\begin{aligned} echo/0 = \text{fun}\ () \to\ & \text{receive} \\ & \quad \{P, M\} \to P\ !\ M \\ & \text{end} \end{aligned}$$

Hello, World!

$$
\begin{array}{rl}
main/0 = \mathrm{fun}\ () \rightarrow & \mathrm{let}\ P2 = \mathrm{spawn}(echo/0, [\,])\\
& \mathrm{in\ let}\ P3 = \mathrm{spawn}(target/0, [\,])\\
& \mathrm{in\ let}\ \_ = P3\ !\ hello\\
& \mathrm{in\ let}\ P2\ !\ \{P3, world\}
\end{array}
$$

$$
\begin{array}{rl}
target/0 = \mathrm{fun}\ () \rightarrow & \mathrm{receive}\\
& \quad A \rightarrow \mathrm{receive}\\
& \qquad\qquad B \rightarrow \{A, B\}\\
& \qquad\quad \mathrm{end}\\
& \mathrm{end}
\end{array}
$$

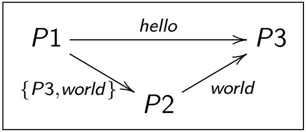


$$
\begin{array}{rl}
echo/0 = \mathrm{fun}\ () \rightarrow & \mathrm{receive}\\
& \quad \{P, M\} \rightarrow P\ !\ M\\
& \mathrm{end}
\end{array}
$$

Hello, World!

$$
\begin{array}{l}
\mathit{main}/0 = \mathrm{fun}\ () \rightarrow \mathrm{let}\ P2 = \mathrm{spawn}(\mathit{echo}/0, [\,])\\
\qquad\qquad\qquad\qquad\ \ \mathrm{in\ let}\ P3 = \mathrm{spawn}(\mathit{target}/0, [\,])\\
\qquad\qquad\qquad\qquad\ \ \mathrm{in\ let}\ \_ = P3\ !\ \mathit{hello}\\
\qquad\qquad\qquad\qquad\ \ \mathrm{in\ let}\ P2\ !\ \{P3, \mathit{world}\}
\end{array}
$$

$$
\begin{array}{l}
\mathit{target}/0 = \mathrm{fun}\ () \rightarrow \mathrm{receive}\\
\qquad\qquad\qquad\qquad\quad A \rightarrow \mathrm{receive}\\
\qquad\qquad\qquad\qquad\qquad\qquad B \rightarrow \{A, B\}\\
\qquad\qquad\qquad\qquad\qquad\quad\ \mathrm{end}\\
\qquad\qquad\qquad\quad\ \mathrm{end}
\end{array}
$$

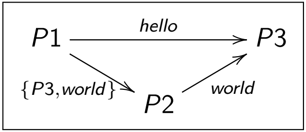


$$
\begin{array}{l}
\mathit{echo}/0 = \mathrm{fun}\ () \rightarrow \mathrm{receive}\\
\qquad\qquad\qquad\qquad\quad \{P, M\} \rightarrow P\ !\ M\\
\qquad\qquad\qquad\quad\ \mathrm{end}
\end{array}
$$

Hello, World!

$$
\begin{array}{lll}
main/0 = \mathrm{fun}\ () \to & \mathrm{let}\ P2 = \mathrm{spawn}(echo/0, [\,]) \\
 & \mathrm{in\ let}\ P3 = \mathrm{spawn}(target/0, [\,]) \\
 & \mathrm{in\ let}\ \_ = P3\ !\ hello \\
 & \mathrm{in\ let}\ P2\ !\ \{P3, world\}
\end{array}
$$

$$
\begin{array}{ll}
target/0 = \mathrm{fun}\ () \to & \mathrm{receive} \\
 & \quad A \to \mathrm{receive} \\
 & \quad\quad\quad B \to \{A, B\} \\
 & \quad\quad\ \mathrm{end} \\
 & \mathrm{end}
\end{array}
$$

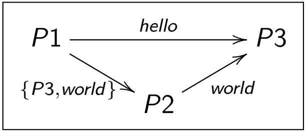


$$
\begin{array}{ll}
echo/0 = \mathrm{fun}\ () \to & \mathrm{receive} \\
 & \quad \{P, M\} \to P\ !\ M \\
 & \mathrm{end}
\end{array}
$$

Hello, World!

$$\begin{aligned} main/0 = \mathrm{fun}\ () \rightarrow\ & \mathrm{let}\ P2 = \mathrm{spawn}(echo/0, [\,])\\ & \mathrm{in\ let}\ P3 = \mathrm{spawn}(target/0, [\,])\\ & \mathrm{in\ let}\ \_ = P3\ !\ hello\\ & \mathrm{in\ let}\ P2\ !\ \{P3, world\} \end{aligned}$$

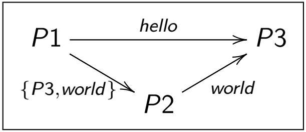
$$\begin{aligned} target/0 = \mathrm{fun}\ () \rightarrow\ & \mathrm{receive}\\ & \quad A \rightarrow \mathrm{receive}\\ & \quad\quad\quad B \rightarrow \{A, B\}\\ & \quad\quad\ \mathrm{end}\\ & \mathrm{end} \end{aligned}$$



$$\begin{aligned} echo/0 = \mathrm{fun}\ () \rightarrow\ & \mathrm{receive}\\ & \quad \{P, M\} \rightarrow P\ !\ M\\ & \mathrm{end} \end{aligned}$$

Hello, World!

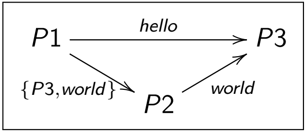
```
main/0 = fun () → let P2 = spawn(echo/0, [ ])
                  in let P3 = spawn(target/0, [ ])
                  in let _ = P3 ! hello
                  in let P2 ! {P3, world}

 target/0 = fun () → receive
                         A → receive
                                 B → {A, B}
                             end
                     end

  echo/0 = fun () → receive
                         {P, M} → P ! M
                     end
```



Hello, World!

$$\begin{aligned}
main/0 = \text{fun } () \to\ & \text{let } P2 = \text{spawn}(echo/0, [\,]) \\
& \text{in let } P3 = \text{spawn}(target/0, [\,]) \\
& \text{in let } \_ = P3 \,!\, hello \\
& \text{in let } P2 \,!\, \{P3, world\}
\end{aligned}$$

$$\begin{aligned}
target/0 = \text{fun } () \to\ & \text{receive} \\
& \quad A \to \text{receive} \\
& \qquad\quad B \to \{A, B\} \\
& \qquad\ \text{end} \\
& \text{end}
\end{aligned}$$

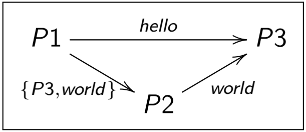


$$\begin{aligned}
echo/0 = \text{fun } () \to\ & \text{receive} \\
& \quad \{P, M\} \to P \,!\, M \\
& \text{end}
\end{aligned}$$

Hello, World!

$$
\begin{array}{rl}
\mathit{main}/0 = \mathrm{fun}\ () \to & \mathrm{let}\ P2 = \mathrm{spawn}(\mathit{echo}/0, [\,])\\
& \mathrm{in\ let}\ P3 = \mathrm{spawn}(\mathit{target}/0, [\,])\\
& \mathrm{in\ let}\ \_ = P3\ !\ \mathit{hello}\\
& \mathrm{in\ let}\ P2\ !\ \{P3, \mathit{world}\}
\end{array}
$$

$$
\begin{array}{rl}
\mathit{target}/0 = \mathrm{fun}\ () \to & \mathrm{receive}\\
& \quad A \to \mathrm{receive}\\
& \qquad\quad B \to \{A, B\}\\
& \qquad \mathrm{end}\\
& \mathrm{end}
\end{array}
$$

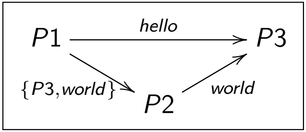


$$
\begin{array}{rl}
\mathit{echo}/0 = \mathrm{fun}\ () \to & \mathrm{receive}\\
& \quad \{P, M\} \to P\ !\ M\\
& \mathrm{end}
\end{array}
$$

Hello, World!

$$\begin{aligned} main/0 = \text{fun } () \to {} & \text{let } P2 = \text{spawn}(echo/0, [\,]) \\ & \text{in let } P3 = \text{spawn}(target/0, [\,]) \\ & \text{in let } \_ = P3 \,!\, hello \\ & \text{in let } P2 \,!\, \{P3, world\} \end{aligned}$$

$$\begin{aligned} target/0 = \text{fun } () \to {} & \text{receive} \\ & \quad A \to \text{receive} \\ & \qquad\quad B \to \{A, B\} \\ & \qquad\ \ \text{end} \\ & \text{end} \end{aligned}$$

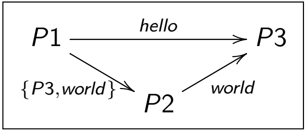


$$\begin{aligned} echo/0 = \text{fun } () \to {} & \text{receive} \\ & \quad \{P, M\} \to P \,!\, M \\ & \text{end} \end{aligned}$$

Hello, World!

$$
\begin{array}{ll}
main/0 = \text{fun } () \rightarrow & \text{let } P2 = \text{spawn}(echo/0, [\,]) \\
 & \text{in let } P3 = \text{spawn}(target/0, [\,]) \\
 & \text{in let } \_ = P3 \mathbin{!} hello \\
 & \text{in let } P2 \mathbin{!} \{P3, world\}
\end{array}
$$

$$
\begin{array}{l}
target/0 = \text{fun } () \rightarrow \text{receive} \\
\qquad\qquad A \rightarrow \text{receive} \\
\qquad\qquad\qquad B \rightarrow {\color{red}\{A, B\}} \\
\qquad\qquad\quad \text{end} \\
\qquad\quad \text{end}
\end{array}
$$



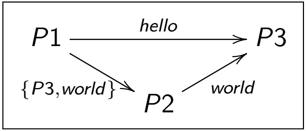
$$
\begin{array}{l}
echo/0 = \text{fun } () \rightarrow \text{receive} \\
\qquad\qquad \{P, M\} \rightarrow P \mathbin{!} M \\
\qquad\quad \text{end}
\end{array}
$$

{hello,world}

Hello, World!

$$
\begin{array}{l}
\mathit{main}/0 = \mathrm{fun}\ () \rightarrow \mathrm{let}\ P2 = \mathrm{spawn}(\mathit{echo}/0, [\,])\\
\qquad\qquad\qquad\qquad \mathrm{in\ let}\ P3 = \mathrm{spawn}(\mathit{target}/0, [\,])\\
\qquad\qquad\qquad\qquad \mathrm{in\ let}\ \_ = P3\ !\ \mathit{hello}\\
\qquad\qquad\qquad\qquad \mathrm{in\ let}\ P2\ !\ \{P3, \mathit{world}\}
\end{array}
$$

$$
\begin{array}{l}
\mathit{target}/0 = \mathrm{fun}\ () \rightarrow \mathrm{receive}\\
\qquad\qquad\qquad\qquad\quad A \rightarrow \mathrm{receive}\\
\qquad\qquad\qquad\qquad\qquad\qquad B \rightarrow \{A, B\}\\
\qquad\qquad\qquad\qquad\qquad\quad \mathrm{end}\\
\qquad\qquad\qquad\qquad \mathrm{end}
\end{array}
$$



$$
\begin{array}{l}
\mathit{echo}/0 = \mathrm{fun}\ () \rightarrow \mathrm{receive}\\
\qquad\qquad\qquad\qquad\quad \{P, M\} \rightarrow P\ !\ M\\
\qquad\qquad\qquad\qquad \mathrm{end}
\end{array}
$$

{hello,world},{world,hello}

Semantics

Standard Semantics

Definition (process)

A process is a triple $\langle p, (\theta, e), q \rangle$ where

- p is the pid of the process
- (θ, e) is the control of the state
- q is the process' local mailbox

Definition (system)

A system is denoted by $\Gamma; \Pi$, where

- Γ is the global mailbox of the system (models the network)
- Π is a pool of processes

We often use $\Gamma; \langle p, (\theta, e), q \rangle \& \Pi$

$$(\mathit{Seq})\quad \frac{\theta, e \xrightarrow{\tau} \theta', e'}{\Gamma; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta', e'), q\rangle \mid \Pi}$$

$$(\mathit{Send})\quad \frac{\theta, e \xrightarrow{\mathsf{send}(p'', v)} \theta', e'}{\Gamma; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma \cup (p'', v); \langle p, (\theta', e'), q\rangle \mid \Pi}$$

$$(\mathit{Receive})\quad \frac{\theta, e \xrightarrow{\mathsf{rec}(\kappa, \overline{cl_n})} \theta', e' \quad \mathsf{matchrec}(\overline{cl_n}, q) = (\theta_i, e_i, v)}{\Gamma; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta'\theta_i, e'\{\kappa \mapsto e_i\}), q \backslash\!\!\backslash v\rangle \mid \Pi}$$

$$(\mathit{Spawn})\quad \frac{\theta, e \xrightarrow{\mathsf{spawn}(\kappa, a/n, [\overline{v_n}])} \theta', e' \quad p' \text{ is a fresh pid}}{\begin{array}{r}\Gamma; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta', e'\{\kappa \mapsto p'\}), q\rangle \\ \mid \langle p', (id, \mathsf{apply}\ a/n\ (\overline{v_n})), [\,]\rangle \mid \Pi\end{array}}$$

$$(\mathit{Self})\quad \frac{\theta, e \xrightarrow{\mathsf{self}(\kappa)} \theta', e'}{\Gamma; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta', e'\{\kappa \mapsto p\}), q\rangle \mid \Pi}$$

$$(\mathit{Sched})\quad \frac{}{\Gamma \cup \{(p, v)\}; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta, e), v : q\rangle \mid \Pi}$$

(Seq)

$$\frac{\theta, e \xrightarrow{\tau} \theta', e'}{\Gamma; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta', e'), q\rangle \mid \Pi}$$

(Send)

$$\frac{\theta, e \xrightarrow{\mathsf{send}(p'', v)} \theta', e'}{\Gamma; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma \cup (p'', v); \langle p, (\theta', e'), q\rangle \mid \Pi}$$

(Receive)

$$\frac{\theta, e \xrightarrow{\mathsf{rec}(\kappa, \overline{cl_n})} \theta', e' \quad \mathsf{matchrec}(\overline{cl_n}, q) = (\theta_i, e_i, v)}{\Gamma; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta'\theta_i, e'\{\kappa \mapsto e_i\}), q \backslash\!\!\backslash v\rangle \mid \Pi}$$

(Spawn)

$$\frac{\theta, e \xrightarrow{\mathsf{spawn}(\kappa, a/n, [\overline{v_n}])} \theta', e' \quad p' \text{ is a fresh pid}}{\begin{array}{r}\Gamma; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta', e'\{\kappa \mapsto p'\}), q\rangle \\ \mid \langle p', (id, \mathsf{apply}\ a/n\ (\overline{v_n})), [\,]\rangle \mid \Pi\end{array}}$$

(Self)

$$\frac{\theta, e \xrightarrow{\mathsf{self}(\kappa)} \theta', e'}{\Gamma; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta', e'\{\kappa \mapsto p\}), q\rangle \mid \Pi}$$

(Sched)

$$\frac{}{\Gamma \cup \{(p, v)\}; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta, e), v : q\rangle \mid \Pi}$$

$$(\mathit{Seq}) \qquad \frac{\theta, e \xrightarrow{\tau} \theta', e'}{\Gamma; \langle p, (\theta, e), q \rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta', e'), q \rangle \mid \Pi}$$

$$(\mathit{Send}) \qquad \frac{\theta, e \xrightarrow{\mathsf{send}(p'', v)} \theta', e'}{\Gamma; \langle p, (\theta, e), q \rangle \mid \Pi \hookrightarrow \Gamma \cup (p'', v); \langle p, (\theta', e'), q \rangle \mid \Pi}$$

$$(\mathit{Receive}) \qquad \frac{\theta, e \xrightarrow{\mathsf{rec}(\kappa, \overline{cl_n})} \theta', e' \quad \mathsf{matchrec}(\overline{cl_n}, q) = (\theta_i, e_i, v)}{\Gamma; \langle p, (\theta, e), q \rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta'\theta_i, e'\{\kappa \mapsto e_i\}), q \backslash\!\!\backslash v \rangle \mid \Pi}$$

$$(\mathit{Spawn}) \qquad \frac{\theta, e \xrightarrow{\mathsf{spawn}(\kappa, a/n, [\overline{v_n}])} \theta', e' \quad p' \text{ is a fresh pid}}{\begin{array}{r} \Gamma; \langle p, (\theta, e), q \rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta', e'\{\kappa \mapsto p'\}), q \rangle \\ \mid \langle p', (id, \mathsf{apply}\ a/n\ (\overline{v_n})), [\,] \rangle \mid \Pi \end{array}}$$

$$(\mathit{Self}) \qquad \frac{\theta, e \xrightarrow{\mathsf{self}(\kappa)} \theta', e'}{\Gamma; \langle p, (\theta, e), q \rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta', e'\{\kappa \mapsto p\}), q \rangle \mid \Pi}$$

$$(\mathit{Sched}) \qquad \frac{}{\Gamma \cup \{(p, v)\}; \langle p, (\theta, e), q \rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta, e), v : q \rangle \mid \Pi}$$

$$(\mathit{Seq})\qquad \frac{\theta, e \xrightarrow{\tau} \theta', e'}{\Gamma; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta', e'), q\rangle \mid \Pi}$$

$$(\mathit{Send})\qquad \frac{\theta, e \xrightarrow{\mathsf{send}(p'',v)} \theta', e'}{\Gamma; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma \cup (p'', v); \langle p, (\theta', e'), q\rangle \mid \Pi}$$

$$(\mathit{Receive})\qquad \frac{\theta, e \xrightarrow{\mathsf{rec}(\kappa,\overline{cl_n})} \theta', e' \quad \mathsf{matchrec}(\overline{cl_n}, q) = (\theta_i, e_i, v)}{\Gamma; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta'\theta_i, e'\{\kappa \mapsto e_i\}), q \backslash\!\!\backslash v\rangle \mid \Pi}$$

$$(\mathit{Spawn})\qquad \frac{\theta, e \xrightarrow{\mathsf{spawn}(\kappa,a/n,[\overline{v_n}])} \theta', e' \quad p' \text{ is a fresh pid}}{\begin{array}{r}\Gamma; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta', e'\{\kappa \mapsto p'\}), q\rangle \\ \mid \langle p', (id, \mathsf{apply}\ a/n\ (\overline{v_n})), [\,]\rangle \mid \Pi\end{array}}$$

$$(\mathit{Self})\qquad \frac{\theta, e \xrightarrow{\mathsf{self}(\kappa)} \theta', e'}{\Gamma; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta', e'\{\kappa \mapsto p\}), q\rangle \mid \Pi}$$

$$(\mathit{Sched})\qquad \frac{}{\Gamma \cup \{(p, v)\}; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta, e), v : q\rangle \mid \Pi}$$

$$(\mathit{Seq}) \qquad \dfrac{\theta, e \xrightarrow{\tau} \theta', e'}{\Gamma; \langle p, (\theta, e), q \rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta', e'), q \rangle \mid \Pi}$$

$$(\mathit{Send}) \qquad \dfrac{\theta, e \xrightarrow{\mathsf{send}(p'', v)} \theta', e'}{\Gamma; \langle p, (\theta, e), q \rangle \mid \Pi \hookrightarrow \Gamma \cup (p'', v); \langle p, (\theta', e'), q \rangle \mid \Pi}$$

$$(\mathit{Receive}) \qquad \dfrac{\theta, e \xrightarrow{\mathsf{rec}(\kappa, \overline{cl_n})} \theta', e' \quad \mathsf{matchrec}(\overline{cl_n}, q) = (\theta_i, e_i, v)}{\Gamma; \langle p, (\theta, e), q \rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta'\theta_i, e'\{\kappa \mapsto e_i\}), q \backslash\!\backslash v \rangle \mid \Pi}$$

$$(\mathit{Spawn}) \qquad \dfrac{\theta, e \xrightarrow{\mathsf{spawn}(\kappa, a/n, [\overline{v_n}])} \theta', e' \quad p' \text{ is a fresh pid}}{\begin{array}{r}\Gamma; \langle p, (\theta, e), q \rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta', e'\{\kappa \mapsto p'\}), q \rangle \\ \mid \langle p', (id, \mathsf{apply}\ a/n\ (\overline{v_n})), [\,] \rangle \mid \Pi\end{array}}$$

$$(\mathit{Self}) \qquad \dfrac{\theta, e \xrightarrow{\mathsf{self}(\kappa)} \theta', e'}{\Gamma; \langle p, (\theta, e), q \rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta', e'\{\kappa \mapsto p\}), q \rangle \mid \Pi}$$

$$(\mathit{Sched}) \qquad \dfrac{}{\Gamma \cup \{(p, v)\}; \langle p, (\theta, e), q \rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta, e), v : q \rangle \mid \Pi}$$

$$(\mathit{Seq})\quad \frac{\theta, e \xrightarrow{\tau} \theta', e'}{\Gamma; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta', e'), q\rangle \mid \Pi}$$

$$(\mathit{Send})\quad \frac{\theta, e \xrightarrow{\mathsf{send}(p'',v)} \theta', e'}{\Gamma; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma \cup (p'', v); \langle p, (\theta', e'), q\rangle \mid \Pi}$$

$$(\mathit{Receive})\quad \frac{\theta, e \xrightarrow{\mathsf{rec}(\kappa,\overline{cl_n})} \theta', e' \qquad \mathsf{matchrec}(\overline{cl_n}, q) = (\theta_i, e_i, v)}{\Gamma; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta'\theta_i, e'\{\kappa \mapsto e_i\}), q \backslash\!\!\backslash v\rangle \mid \Pi}$$

$$(\mathit{Spawn})\quad \frac{\theta, e \xrightarrow{\mathsf{spawn}(\kappa, a/n, [\overline{v_n}])} \theta', e' \qquad p' \text{ is a fresh pid}}{\begin{array}{r}\Gamma; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta', e'\{\kappa \mapsto p'\}), q\rangle \\ \mid \langle p', (\mathit{id}, \mathsf{apply}\ a/n\ (\overline{v_n})), [\,]\rangle \mid \Pi\end{array}}$$

$$(\mathit{Self})\quad \frac{\theta, e \xrightarrow{\mathsf{self}(\kappa)} \theta', e'}{\Gamma; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta', e'\{\kappa \mapsto p\}), q\rangle \mid \Pi}$$

$$(\mathit{Sched})\quad \frac{}{\Gamma \cup \{(p, v)\}; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta, e), v : q\rangle \mid \Pi}$$

$$(\mathit{Seq}) \quad \frac{\theta, e \xrightarrow{\tau} \theta', e'}{\Gamma; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta', e'), q\rangle \mid \Pi}$$

$$(\mathit{Send}) \quad \frac{\theta, e \xrightarrow{\mathsf{send}(p'',v)} \theta', e'}{\Gamma; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma \cup (p'', v); \langle p, (\theta', e'), q\rangle \mid \Pi}$$

$$(\mathit{Receive}) \quad \frac{\theta, e \xrightarrow{\mathsf{rec}(\kappa,\overline{cl_n})} \theta', e' \quad \mathsf{matchrec}(\overline{cl_n}, q) = (\theta_i, e_i, v)}{\Gamma; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta'\theta_i, e'\{\kappa \mapsto e_i\}), q \backslash\!\backslash v\rangle \mid \Pi}$$

$$(\mathit{Spawn}) \quad \frac{\theta, e \xrightarrow{\mathsf{spawn}(\kappa,a/n,[\overline{v_n}])} \theta', e' \quad p' \text{ is a fresh pid}}{\begin{array}{r}\Gamma; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta', e'\{\kappa \mapsto p'\}), q\rangle \\ \mid \langle p', (id, \mathsf{apply}\ a/n\ (\overline{v_n})), [\,]\rangle \mid \Pi\end{array}}$$

$$(\mathit{Self}) \quad \frac{\theta, e \xrightarrow{\mathsf{self}(\kappa)} \theta', e'}{\Gamma; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta', e'\{\kappa \mapsto p\}), q\rangle \mid \Pi}$$

$$(\mathit{Sched}) \quad \frac{}{\Gamma \cup \{(p, v)\}; \langle p, (\theta, e), q\rangle \mid \Pi \hookrightarrow \Gamma; \langle p, (\theta, e), v : q\rangle \mid \Pi}$$

Reversible Trace Semantics

$$(\mathit{Seq})\quad \frac{\theta, e \xrightarrow{\tau} \theta', e'}{\Gamma; \langle h, p, (\theta, e), q\rangle \& \Pi \rightharpoonup \Gamma; \langle \tau(\theta, e) : h, p, (\theta', e'), q\rangle \& \Pi}$$

$$(\mathit{Self})\quad \frac{\theta, e \xrightarrow{\mathsf{self}(y)} \theta', e'}{\Gamma; \langle h, p, (\theta, e), q\rangle \& \Pi \rightharpoonup \Gamma; \langle \mathsf{self}(\theta, e) : h, p, (\theta', e'\{y \mapsto p\}), q\rangle \& \Pi}$$

$$(\mathit{Spawn})\quad \frac{\theta, e \xrightarrow{\mathsf{spawn}(y, a/n, [e_1, \ldots, e_n])} \theta', e' \quad p' \text{ is a fresh pid}}{\begin{array}{ll}\Gamma; \langle h, p, (\theta, e), q\rangle \& \Pi \rightharpoonup & \Gamma; \langle \mathsf{spawn}(\theta, e, p') : h, p, (\theta', e'\{y \mapsto p'\}), q\rangle \\ & \& \langle [\,], p', (\theta, \mathsf{apply}\ a/n\ (e_1, \ldots, e_n)), [\,]\rangle \& \Pi\end{array}}$$

$$(\mathit{Send})\quad \frac{\theta, e \overset{\mathsf{send}(p'',v)}{\longrightarrow} \theta', e' \;\; \text{and} \;\; \lambda \text{ is a fresh identifier}}{\Gamma; \langle h, p, (\theta, e), q\rangle \& \Pi \rightharpoonup \Gamma \cup (p'', \{v, \lambda\}); \langle \mathsf{send}(\theta, e, \{v, \lambda\}) : h, p, (\theta', e'), q\rangle \& \Pi}$$

$$(\mathit{Receive})\quad \frac{\theta, e \overset{\mathsf{rec}(y,\overline{cl_n})}{\longrightarrow} \theta', e' \quad \mathsf{matchrec}(\overline{cl_n}, q) = (\theta_i, e_i, \{v, \lambda\})}{\Gamma; \langle h, p, (\theta, e), q\rangle \& \Pi \rightharpoonup \Gamma; \langle \begin{array}{l} \mathsf{rec}(\theta, e, \{v, \lambda\}, q, q \backslash\!\backslash \{v, k\}) : h, \\ p, (\theta'\theta_i, e'\{y \mapsto e_i\}), q \backslash\!\backslash \{v, k\} \end{array} \rangle \& \Pi}$$

$$(\mathit{Sched})\quad \frac{}{\Gamma \cup \{(p, \{v, \lambda\})\}; \langle h, p, (\theta, e), q\rangle \& \Pi \rightharpoonup \Gamma; \langle h, p, (\theta, e), \{v, \lambda\} : q\rangle \& \Pi}$$

Causal-Consistent Debugging Semantics

Rollbacks

Processes in rollback mode are annotated using $\lfloor\ \rfloor_{\Psi}$ where Ψ is the requested set of rollbacks

- s: one causal-consistent backward step
- $\lambda^{\Uparrow}$: a backward derivation up to the sending of λ
- $\lambda^{\Downarrow}$: a backward derivation up to the delivery of λ
- λ^{rec}: a backward derivation up to the receive of λ
- sp_p: a backward derivation up to the spawning of p
- sp: a backward derivation up to the creation of the process
- X: a backward derivation up to the introduction of variable X

Undoing the sending of a message

$$(\overline{\mathit{Send1}}) \quad \begin{array}{l} \Gamma \cup \{(p', \{v, \lambda\})\}; \\ \lfloor \langle p, \mathsf{send}(\theta, e, p', \{v, \lambda\}) : h, (\theta', e'), q \rangle \rfloor_{\Psi} \mid \Pi \\ \qquad\qquad \leftharpoondown \Gamma; \lfloor \langle p, h, (\theta, e), q \rangle \rfloor_{\Psi \setminus \{\lambda^{\Uparrow}\}} \mid \Pi \end{array}$$

$$(\overline{\mathit{Send2}}) \quad \begin{array}{l} \Gamma; \lfloor \langle p, \mathsf{send}(\theta, e, p', \{v, \lambda\}) : h, (\theta', e'), q \rangle \rfloor_{\Psi} \\ \quad \mid \lfloor \langle p', h', (\theta'', e''), q' \rangle \rfloor_{\Psi'} \mid \Pi \\ \qquad\qquad \leftharpoondown \Gamma; \lfloor \langle p, \mathsf{send}(\theta, e, p', \{v, \lambda\}) : h, (\theta', e'), q \rangle \rfloor_{\Psi} \mid \\ \qquad\qquad\qquad\qquad \lfloor \langle p', h', (\theta'', e''), q' \rangle \rfloor_{\Psi' \cup \{\lambda^{\Downarrow}\}} \mid \Pi \end{array}$$

$$(\overline{\mathit{Sched}}) \quad \begin{array}{l} \Gamma; \lfloor \langle p, h, (\theta, e), \{v, \lambda\} : q \rangle \rfloor_{\Psi} \mid \Pi \\ \leftharpoondown \Gamma \cup (p, \{v, \lambda\}); \lfloor \langle p, h, (\theta, e), q \rangle \rfloor_{\Psi \setminus \{\lambda^{\Downarrow}\}} \mid \Pi \end{array}$$

Undoing the sending of a message

$(\overline{\mathit{Send1}})$

$$\begin{array}{l} \Gamma \cup \{(p', \{v, \lambda\})\}; \\ \lfloor \langle p, \mathsf{send}(\theta, e, p', \{v, \lambda\}) : h, (\theta', e'), q \rangle \rfloor_{\Psi} \mid \Pi \\ \qquad \leftharpoondown \Gamma; \lfloor \langle p, h, (\theta, e), q \rangle \rfloor_{\Psi \setminus \{\lambda^{\Uparrow}\}} \mid \Pi \end{array}$$

$(\overline{\mathit{Send2}})$

$$\begin{array}{l} \Gamma; \lfloor \langle p, \mathsf{send}(\theta, e, p', \{v, \lambda\}) : h, (\theta', e'), q \rangle \rfloor_{\Psi} \\ \quad \mid \lfloor \langle p', h', (\theta'', e''), q' \rangle \rfloor_{\Psi'} \mid \Pi \\ \qquad \leftharpoondown \Gamma; \lfloor \langle p, \mathsf{send}(\theta, e, p', \{v, \lambda\}) : h, (\theta', e'), q \rangle \rfloor_{\Psi} \mid \\ \qquad\qquad \lfloor \langle p', h', (\theta'', e''), q' \rangle \rfloor_{\Psi' \cup \{\lambda^{\Downarrow}\}} \mid \Pi \end{array}$$

$(\overline{\mathit{Sched}})$

$$\begin{array}{l} \Gamma; \lfloor \langle p, h, (\theta, e), \{v, \lambda\} : q \rangle \rfloor_{\Psi} \mid \Pi \\ \leftharpoondown \Gamma \cup (p, \{v, \lambda\}); \lfloor \langle p, h, (\theta, e), q \rangle \rfloor_{\Psi \setminus \{\lambda^{\Downarrow}\}} \mid \Pi \end{array}$$

Undoing the sending of a message

$(\overline{\mathit{Send1}})$

$$\begin{array}{l} \Gamma \cup \{(p', \{v, \lambda\})\}; \\ \lfloor \langle p, \mathsf{send}(\theta, e, p', \{v, \lambda\}) : h, (\theta', e'), q \rangle \rfloor_{\Psi} \mid \Pi \\ \qquad \leftharpoondown \Gamma; \lfloor \langle p, h, (\theta, e), q \rangle \rfloor_{\Psi \setminus \{\lambda^{\Uparrow}\}} \mid \Pi \end{array}$$

$(\overline{\mathit{Send2}})$

$$\begin{array}{l} \Gamma; \lfloor \langle p, \mathsf{send}(\theta, e, p', \{v, \lambda\}) : h, (\theta', e'), q \rangle \rfloor_{\Psi} \\ \quad \mid \lfloor \langle p', h', (\theta'', e''), q' \rangle \rfloor_{\Psi'} \mid \Pi \\ \qquad \leftharpoondown \Gamma; \lfloor \langle p, \mathsf{send}(\theta, e, p', \{v, \lambda\}) : h, (\theta', e'), q \rangle \rfloor_{\Psi} \mid \\ \qquad\qquad \lfloor \langle p', h', (\theta'', e''), q' \rangle \rfloor_{\Psi' \cup \{\lambda^{\Downarrow}\}} \mid \Pi \end{array}$$

$(\overline{\mathit{Sched}})$

$$\begin{array}{l} \Gamma; \lfloor \langle p, h, (\theta, e), \{v, \lambda\} : q \rangle \rfloor_{\Psi} \mid \Pi \\ \leftharpoondown \Gamma \cup (p, \{v, \lambda\}); \lfloor \langle p, h, (\theta, e), q \rangle \rfloor_{\Psi \setminus \{\lambda^{\Downarrow}\}} \mid \Pi \end{array}$$

Undoing the spawning of a process

$$(\overline{\mathit{Spawn1}}) \qquad \begin{array}{l} \Gamma; \lfloor\langle p, \mathsf{spawn}(\theta, e, p'') : h, (\theta', e'), q\rangle\rfloor_{\Psi} \\ \qquad\qquad \mid \lfloor\langle p'', [\,], (\theta'', e''), [\,]\rangle\rfloor_{\Psi'} \mid \Pi \\ \leftharpoondown \Gamma; \lfloor\langle p, h, (\theta, e), q\rangle\rfloor_{\Psi \setminus \{\mathsf{sp}_{p''}\}} \mid \Pi \end{array}$$

$$(\overline{\mathit{Spawn2}}) \qquad \begin{array}{l} \Gamma; \lfloor\langle p, \mathsf{spawn}(\theta, e, p'') : h, (\theta, e), q\rangle\rfloor_{\Psi} \\ \qquad\qquad \mid \lfloor\langle p'', h'', (\theta'', e''), q''\rangle\rfloor_{\Psi'} \mid \Pi \\ \leftharpoondown \Gamma; \lfloor\langle p, \mathsf{spawn}(\theta, e, p'') : h, (\theta, e), q\rangle\rfloor_{\Psi} \\ \qquad\qquad \mid \lfloor\langle p'', h'', (\theta'', e''), q''\rangle\rfloor_{\Psi' \cup \{\mathsf{sp}\}} \mid \Pi \end{array}$$

More details in the paper...

Undoing the spawning of a process

$$(\overline{Spawn1}) \quad \begin{array}{l} \Gamma; \lfloor \langle p, \mathsf{spawn}(\theta, e, p'') : h, (\theta', e'), q \rangle \rfloor_{\Psi} \\ \qquad\qquad \mid \lfloor \langle p'', [\,], (\theta'', e''), [\,] \rangle \rfloor_{\Psi'} \mid \Pi \\ \leftharpoondown \Gamma; \lfloor \langle p, h, (\theta, e), q \rangle \rfloor_{\Psi \setminus \{\mathsf{sp}_{p''}\}} \mid \Pi \end{array}$$

$$(\overline{Spawn2}) \quad \begin{array}{l} \Gamma; \lfloor \langle p, \mathsf{spawn}(\theta, e, p'') : h, (\theta, e), q \rangle \rfloor_{\Psi} \\ \qquad\qquad \mid \lfloor \langle p'', h'', (\theta'', e''), q'' \rangle \rfloor_{\Psi'} \mid \Pi \\ \leftharpoondown \Gamma; \lfloor \langle p, \mathsf{spawn}(\theta, e, p'') : h, (\theta, e), q \rangle \rfloor_{\Psi} \\ \qquad\qquad \mid \lfloor \langle p'', h'', (\theta'', e''), q'' \rangle \rfloor_{\Psi' \cup \{\mathsf{sp}\}} \mid \Pi \end{array}$$

More details in the paper. . .

Undoing the spawning of a process

$$(\overline{\mathit{Spawn1}}) \quad \begin{aligned} &\Gamma; \lfloor \langle p, \mathsf{spawn}(\theta, e, p''):h, (\theta', e'), q \rangle \rfloor_{\Psi} \\ &\qquad\qquad \mid \lfloor \langle p'', [\,], (\theta'', e''), [\,] \rangle \rfloor_{\Psi'} \mid \Pi \\ &\leftharpoondown \Gamma; \lfloor \langle p, h, (\theta, e), q \rangle \rfloor_{\Psi \setminus \{\mathsf{sp}_{p''}\}} \mid \Pi \end{aligned}$$

$$(\overline{\mathit{Spawn2}}) \quad \begin{aligned} &\Gamma; \lfloor \langle p, \mathsf{spawn}(\theta, e, p''):h, (\theta, e), q \rangle \rfloor_{\Psi} \\ &\qquad\qquad \mid \lfloor \langle p'', h'', (\theta'', e''), q'' \rangle \rfloor_{\Psi'} \mid \Pi \\ &\leftharpoondown \Gamma; \lfloor \langle p, \mathsf{spawn}(\theta, e, p''):h, (\theta, e), q \rangle \rfloor_{\Psi} \\ &\qquad\qquad \mid \lfloor \langle p'', h'', (\theta'', e''), q'' \rangle \rfloor_{\Psi' \cup \{\mathsf{sp}\}} \mid \Pi \end{aligned}$$

More details in the paper. . .

Demo: CauDEr

https://github.com/mistupv/cauder

Conclusions

We have

- designed a causal-consistent reversible semantics for Erlang
- developed CauDEr: a causal-consistent debugger for Erlang

Ongoing work:

- causal-consistent replay
- redesign GUI

Thanks for your attention!

https://github.com/mistupv/cauder