A general approach to derive uncontrolled reversible semantics

Ivan Lanese  Doriana Medić

This work has been partially supported by French ANR project DCore ANR-18-CE25-0007.

CONCUR, September 1-4, 2020
online event
Reversible computing

- attracted interest from the 1970’s

- applied in biochemical modelling, thermodynamics, simulation, robotics, programming, program debugging, etc.

- **sequential system** - forward actions are undone starting from the last executed action

- **concurrent system** - identifying the last action is not immediate

- **causally-consistent reversibility** - an action can be reversed, provided all its consequences have been reversed

- a causal-consistent reversible model satisfies a number of relevant properties (e.g., Loop Lemma)
Aim of this paper

- to explore how to mechanically obtain a causal-consistent reversible extension of a given forward-only model

- the reversible models built by using our approach satisfy the properties related to causally-consistent reversibility

- two case studies: Higher-Order $\pi$-calculus and Core Erlang
  - the obtained reversible models have the same behaviour as the ones in the literature

- we extend reversible Core Erlang to support Core Erlang constructs for remote error handling based on links
The requirements for the forward model

The syntax of the forward model:
The requirements for the forward model

The syntax of the forward model:

- The **lower level** is composed by entities (processes, messages, resources) ranged over by $P, Q$.
  - No restrictions on the syntax of the lower level.
The requirements for the forward model

The syntax of the forward model:

- The lower level is composed by entities (processes, messages, resources) ranged over by $P, Q$.
  - No restrictions on the syntax of the lower level.

- The high-level syntax of the system is defined as:

\[ N ::= P \mid op_n(N_1, \ldots, N_n) \mid 0 \]

where

- $op_n(N_1, \ldots, N_n)$ represent the $n$-ary operators applied on systems $N_1, \ldots, N_n$;
- parallel composition, $N_1 \mid N_2$, is assumed among the operators;
- $0$ represents the empty system.
The syntax of the asynchronous HOπ-calculus is:

\[ P, Q ::= a\langle P \rangle \mid a(X) \triangleright P \mid (P \mid Q) \mid \nu a(P) \mid X \mid 0 \]

- we separate entities from systems;
- an entity is any HOπ process whose topmost operator is neither a parallel composition nor a restriction nor 0.
Running example: HO\(\pi\)-calculus

○ The syntax of the asynchronous HO\(\pi\)-calculus is:

\[
P, Q ::= a\langle P \rangle \ | \ a(X) \triangleright P \ | \ (P \ | \ Q) \ | \ \nu a(P) \ | \ X \ | \ 0
\]

we separate entities from systems;

an entity is any HO\(\pi\) process whose topmost operator is neither a parallel composition nor a restriction nor 0.

○ The syntax of systems is defined as:

\[
N ::= P \ | \ (N_1 \ | \ N_2) \ | \ \nu a(N) \ | \ 0
\]

where the operators \(\mid\) and 0 are required by our framework;
A generic system can be represented as a term

\[ T[P_1, \ldots, P_n] \]

where \( T[\bullet_1, \ldots, \bullet_n] \) is a context with \( n \) numbered holes.

- \( T \) is built from composition operators, possibly including parallel composition and \( 0 \).
**Structural congruence**

- A generic system can be represented as a term

\[ T[P_1, \ldots, P_n] \]

where \( T[\bullet_1, \ldots, \bullet_n] \) is a context with \( n \) numbered holes.

- \( T \) is built from composition operators, possibly including parallel composition and 0.

- Structural congruence is specified by axioms of the form:

\[ T[P_1, \ldots, P_n] \equiv T'[P'_1, \ldots, P'_n] \]

closed under contexts, reflexivity, symmetry and transitivity.
Structural congruence

- A generic system can be represented as a term

\[ T[P_1, \ldots, P_n] \]

where \( T[\bullet_1, \ldots, \bullet_n] \) is a context with \( n \) numbered holes.

- \( T \) is built from composition operators, possibly including parallel composition and \( 0 \).

- Structural congruence is specified by axioms of the form:

\[ T[P_1, \ldots, P_n] \equiv T'[P'_1, \ldots, P'_n] \]

closed under contexts, reflexivity, symmetry and transitivity.

- **Example:** Sample of \( \text{HO}_\pi \) structural rule is:

\[(\text{PARC}) \quad P \mid Q \equiv Q \mid P\]

- exploits contexts of the form \( \bullet_1 \mid \bullet_2 \) and \( \bullet_2 \mid \bullet_1 \)
The reduction semantics of the forward model

\[
(SCM-\text{Act}) \quad P_1 \mid \ldots \mid P_n \Rightarrow T[Q_1, \ldots, Q_m]
\]

\[
(EQV) \quad N \equiv N' \quad N \Rightarrow N_1 \quad N_1 \equiv N'_1
\]

\[
(SCM-\text{OPN}) \quad N_i \Rightarrow N'_i
\]

\[
\text{op}_n(N_0, \ldots, N_i, \ldots, N_n) \Rightarrow \text{op}_n(N_0, \ldots, N'_i, \ldots, N_n)
\]

\[
(PAR) \quad N \Rightarrow N'
\]

\[
N \mid N_1 \Rightarrow N' \mid N_1
\]
The reduction semantics of the forward model

\[
\begin{align*}
\text{(Scm-Act)} & \quad P_1 \mid \ldots \mid P_n \Rightarrow T[Q_1, \ldots, Q_m] \\
\text{(Eqv)} & \quad N \equiv N' \quad N \Rightarrow N_1 \quad N_1 \equiv N'_1 \\
\text{(Scm-Opn)} & \quad N_i \Rightarrow N_i' \\
\text{(Par)} & \quad N \Rightarrow N' \quad N \mid N_1 \Rightarrow N' \mid N_1
\end{align*}
\]
The reduction semantics of the forward model

\[(\text{Scm-Act})\]

\[
P_1 \mid \ldots \mid P_n \Rightarrow T[Q_1, \ldots, Q_m]
\]

\[(\text{Scm-Opn})\]

\[
N_i \Rightarrow N_i' \\
op_n(N_0, \ldots, N_i, \ldots, N_n) \Rightarrow \nop_n(N_0, \ldots, N_i', \ldots, N_n)
\]

\[(\text{Par})\]

\[
N \Rightarrow N' \\
N \mid N_1 \Rightarrow N' \mid N_1
\]

\[(\text{Eqv})\]

\[
N \equiv N' \quad N \Rightarrow N_1 \quad N_1 \equiv N_1'
\]

\[
N' \Rightarrow N_1'
\]
The reduction semantics of the forward model

\[ (\text{Scm-Act}) \quad \frac{P_1 \mid \ldots \mid P_n \rightarrow T[Q_1, \ldots, Q_m]}{N \equiv N'} \quad N \rightarrow N_1 \quad N_1 \equiv N'_1 \]

\[ (\text{Scm-Opn}) \quad \frac{N_i \rightarrow N'_i}{op_n(N_0, \ldots, N_i, \ldots, N_n) \rightarrow op_n(N_0, \ldots, N'_i, \ldots, N_n)} \]

\[ (\text{Par}) \quad \frac{N \rightarrow N'}{N | N_1 \rightarrow N' | N_1} \]
The reduction semantics of the forward model

\[ \begin{align*}
(\text{SCM-Act}) & \quad \frac{P_1 \mid \ldots \mid P_n \Rightarrow T[Q_1, \ldots, Q_m]}{N \equiv N'} \quad N \Rightarrow N_1 \quad N_1 \equiv N'_1} \\
(\text{SCM-Opn}) & \quad \frac{N_i \Rightarrow N'_i}{\text{op}_n(N_0, \ldots, N_i, \ldots, N_n) \Rightarrow \text{op}_n(N_0, \ldots, N'_i, \ldots, N_n)} \\
(\text{Par}) & \quad \frac{N \Rightarrow N'}{N \mid N_1 \Rightarrow N' \mid N_1}
\end{align*} \]
The reduction semantics of the forward model

\[(\text{SCM-Act})\] \[
P_1 \mid \ldots \mid P_n \Rightarrow T[Q_1, \ldots, Q_m]\]

\[(\text{SCM-Opn})\] \[
N_i \Rightarrow N'_i \quad \Rightarrow \quad \text{op}_n(N_0, \ldots, N_i, \ldots, N_n) \Rightarrow \text{op}_n(N_0, \ldots, N'_i, \ldots, N_n)\]

\[(\text{EQV})\] \[
N \equiv N' \quad N \Rightarrow N_1 \quad N_1 \equiv N'_1\]

\[(\text{PAR})\] \[
N \Rightarrow N' \quad \Rightarrow \quad N \mid N_1 \Rightarrow N' \mid N_1\]
The reduction semantics of the forward model

\[(\text{Scm-Act})\quad \frac{P_1 \ | \ \ldots \ | \ P_n \Rightarrow T[Q_1, \ldots, Q_m]}{N \equiv N'}\]

\[(\text{Eqv})\quad \frac{N \Rightarrow N_1}{N' \Rightarrow N_1'}\]

\[(\text{Scm-Opn})\quad \frac{N_i \Rightarrow N_i'}{op_n(N_0, \ldots, N_i, \ldots, N_n) \Rightarrow op_n(N_0, \ldots, N_i', \ldots, N_n)}\]

\[(\text{Par})\quad \frac{N \Rightarrow N'}{N \ | \ N_1 \Rightarrow N' \ | \ N_1}\]
The communication rule \((\text{Act})\) of \(\text{HO}^\pi\)-calculus, is defined as:

\[
\begin{align*}
(\text{Act}) & \quad a\langle Q \rangle \mid a(X) \triangleright P \iff P\{Q/X\}
\end{align*}
\]
Example: the communication rule of $\text{HO}^\pi$

- The communication rule ($\text{Act}$) of $\text{HO}^\pi$-calculus, is defined as:

$$
\text{(Act)} \quad a\langle Q \rangle | a(X) \triangleright P \leftrightarrow P\{Q/X\}
$$

- The number of entities in the resulting process may vary:

$$
a\langle b\langle P \rangle | b(Y) \triangleright Y | c\langle Q \rangle \rangle | a(X) \triangleright X \leftrightarrow b\langle P \rangle | b(Y) \triangleright Y | c\langle Q \rangle
$$

- the resulting process has three entities $b\langle P \rangle$, $b(Y) \triangleright Y$ and $c\langle Q \rangle$, composed using a context $T = \bullet_1 | \bullet_2 | \bullet_3$. 
Definition of the reversible system

- The syntax of configurations $R$ is as:

$$ R ::= k : P \mid op_n(R_1, \ldots, R_n) \mid 0 \mid [R; C] \quad C ::= T[k_1 : \bullet_1, \ldots, k_m : \bullet_m] $$

where

- $k$ denotes a key identifying each entity of a system;
- $op_n$ are the same as in the forward system;
- $T$ is a context composed of operators $op_n$ and $0$;
- $\bullet_i$ are numbered holes, to be filled by the processes with keys $k_i$;
- $[R; C]$ is a memory ($R$ is the configuration which gave rise to the forward step and $C$ is the context of the resulting configuration).
Example: the syntax of the reversible HOπ-calculus is defined as:

\[ R ::= k : P \mid (R_1 \mid R_2) \mid \nu a (R) \mid 0 \mid [R; C] \]

where \( P \) are the entities as in the underlying calculus.
- **Example:** the syntax of the reversible HOπ-calculus is defined as:

\[
R ::= k : P | (R_1 | R_2) | \nu a (R) | 0 | [R; C]
\]

where \( P \) are the entities as in the underlying calculus.

- **Structural congruence** is specified by axioms:

\[
T[k_1 : P_1, \ldots, k_n : P_n] \equiv T'[k_1 : P'_1, \ldots, k_n : P'_n]
\]

- one structural rule for each structural rule of the original semantics;
- the context \( T \) is the same as in the forward-only semantics;
- entities are labelled with keys and keys on both sides are the same.
The uncontrolled reversible semantics

- The forward rules of the uncontrolled reversible semantics:

\[ \text{(F-Scm-Act)} \]

\[
P_1 \mid \ldots \mid P_n \mapsto T[Q_1, \ldots, Q_m] \quad j_1, \ldots, j_m \text{ are fresh keys}
\]

\[
k_1 : P_1 \mid \ldots \mid k_n : P_n \mapsto T[j_1 : Q_1, \ldots, j_m : Q_m] \mid [k_1 : P_1 \mid \ldots \mid k_n : P_n; T[j_1 : \bullet_1, \ldots, j_m : \bullet_m]]
\]

\[ \text{(F-Scm-Opn)} \]

\[
R_i \mapsto R'_i \quad (\text{key}(R'_i) \setminus \text{key}(R_i)) \cap (\text{key}(R_0, \ldots, R_{i-1}, R_{i+1}, \ldots, R_n) = \emptyset
\]

\[
op_n(R_0, \ldots, R_i, \ldots, R_n) \mapsto \op_n(R_0, \ldots, R'_i, \ldots, R_n)
\]

\[ \text{(F-Eqv)} \]

\[
R \equiv R' \quad R \mapsto R_1 \quad R_1 \equiv R'_1
\]

\[
R' \mapsto R'_1
\]
The uncontrolled reversible semantics

- The forward rules of the uncontrolled reversible semantics:

  \[ F-\text{Scm-Act} \]
  \[
P_1 \mid \ldots \mid P_n \rightarrow T[Q_1, \ldots, Q_m] \quad j_1, \ldots, j_m \text{ are fresh keys}
  \]
  \[
k_1 : P_1 \mid \ldots \mid k_n : P_n \rightarrow T[j_1 : Q_1, \ldots, j_m : Q_m] | k_1 : P_1 \mid \ldots \mid k_n : P_n; T[j_1 : \bullet_1, \ldots, j_m : \bullet_m]
  \]

  \[ F-\text{Scm-Opn} \]
  \[
  R_i \rightarrow R_i' \quad (\text{key}(R_i') \setminus \text{key}(R_i)) \cap (\text{key}(R_0, \ldots, R_{i-1}, R_{i+1}, \ldots, R_n) = \emptyset
  \]
  \[
op_n(R_0, \ldots, R_i, \ldots, R_n) \rightarrow \op_n(R_0, \ldots, R_i', \ldots, R_n)
  \]

  \[ F-\text{Eqv} \]
  \[
  R \equiv R' \quad R \rightarrow R_1 \quad R_1 \equiv R_1'
  \]
  \[
  R' \rightarrow R_1'
  \]
The uncontrolled reversible semantics

- The forward rules of the uncontrolled reversible semantics:

\[(F-Scm-Act)\]

\[
P_1 \mid \ldots \mid P_n \Rightarrow T[Q_1, \ldots, Q_m] \quad j_1, \ldots, j_m \text{ are fresh keys}
\]

\[
k_1 : P_1 \mid \ldots \mid k_n : P_n \Rightarrow T[j_1 : Q_1, \ldots, j_m : Q_m] \mid
\]

\[
[k_1 : P_1 \mid \ldots \mid k_n : P_n; T[j_1 : \bullet_1, \ldots, j_m : \bullet_m]]
\]

\[(F-Scm-Opn)\]

\[
R_i \Rightarrow R'_i \quad (\text{key}(R'_i) \setminus \text{key}(R_i)) \cap (\text{key}(R_0, \ldots, R_{i-1}, R_{i+1}, \ldots, R_n) = \emptyset
\]

\[
op_n(R_0, \ldots, R_i, \ldots, R_n) \Rightarrow \op_n(R_0, \ldots, R'_i, \ldots, R_n)
\]

\[(F-Eqv)\]

\[
R \equiv R' \quad R \Rightarrow R_1 \quad R_1 \equiv R'_1
\]

\[
R' \Rightarrow R'_1
\]
The uncontrolled reversible semantics

- The forward rules of the uncontrolled reversible semantics:

\[
(F\text{-SCM-Act}) \quad P_1 \mid \ldots \mid P_n \rightarrow T[Q_1, \ldots, Q_m] \quad j_1, \ldots, j_m \text{ are fresh keys}
\]

\[
\quad k_1 : P_1 \mid \ldots \mid k_n : P_n \rightarrow T[j_1 : Q_1, \ldots, j_m : Q_m] |
\quad [k_1 : P_1 \mid \ldots \mid k_n : P_n; T[j_1 : \bullet_1, \ldots, j_m : \bullet_m]]
\]

\[
(F\text{-SCM-OPN}) \quad R_i \rightarrow R_i' \quad (\text{key}(R_i') \setminus \text{key}(R_i)) \cap (\text{key}(R_0, \ldots, R_{i-1}, R_{i+1}, \ldots, R_n) = \emptyset
\]

\[
\quad \text{op}_n(R_0, \ldots, R_i, \ldots, R_n) \rightarrow \text{op}_n(R_0, \ldots, R_i', \ldots, R_n)
\]

\[
(F\text{-EQV}) \quad R \equiv R' \quad R \rightarrow R_1 \quad R_1 \equiv R_1'
\quad R' \rightarrow R_1'
\]
The uncontrolled reversible semantics

- The forward rules of the uncontrolled reversible semantics:

\[
\text{(F-SCM-Act)} \quad \begin{array}{c}
\frac{P_1 \mid \ldots \mid P_n \Rightarrow T[Q_1, \ldots, Q_m]}{k_1 : P_1 \mid \ldots \mid k_n : P_n \Rightarrow T[j_1 : Q_1, \ldots, j_m : Q_m]} \\
\quad \quad \text{\(j_1, \ldots, j_m\) are fresh keys}
\end{array}
\]

\[
\text{(F-SCM-Act)} \quad \begin{array}{c}
k_1 : P_1 \mid \ldots \mid k_n : P_n \Rightarrow T[j_1 : Q_1, \ldots, j_m : Q_m] \\
\quad \quad \text{\([k_1 : P_1 \mid \ldots \mid k_n : P_n; T[j_1 : \bullet_1, \ldots, j_m : \bullet_m]]\)}
\end{array}
\]

- The backward rules are symmetric w.r.t. the forward ones.

\[
\text{(F-SCM-Opn)} \quad \begin{array}{c}
R_i \rightarrow R'_i \\
(\text{key}(R'_i) \setminus \text{key}(R_i)) \cap (\text{key}(R_0, \ldots, R_{i-1}, R_{i+1}, \ldots, R_n) = \emptyset
\quad \quad \text{op}_n(R_0, \ldots, R_i, \ldots, R_n) \rightarrow \text{op}_n(R_0, \ldots, R'_i, \ldots, R_n)
\end{array}
\]

\[
\text{(F-Eqv)} \quad \begin{array}{c}
R \equiv R' \\
R \Rightarrow R_1 \\
R_1 \equiv R'_1
\end{array}
\]

\[
\begin{array}{c}
\frac{R \equiv R' \quad R \Rightarrow R_1 \quad R_1 \equiv R'_1}{R' \Rightarrow R'_1}
\end{array}
\]
The uncontrolled reversible semantics

- The forward rules of the uncontrolled reversible semantics:

\[(F-Scm-Act)\]

\[P_1 | \ldots | P_n \implies T[Q_1, \ldots, Q_m] \quad j_1, \ldots, j_m \text{ are fresh keys}\]

\[k_1 : P_1 | \ldots | k_n : P_n \implies T[j_1 : Q_1, \ldots, j_m : Q_m] | \]

\[\left[ k_1 : P_1 | \ldots | k_n : P_n; T[j_1 : \bullet_1, \ldots, j_m : \bullet_m] \right]\]

\[(F-Scm-Opn)\]

\[R_i \rightarrow R'_i \quad (\text{key}(R'_i) \setminus \text{key}(R_i)) \cap (\text{key}(R_0, \ldots, R_{i-1}, R_{i+1}, \ldots, R_n) = \emptyset\]

\[\text{op}_n(R_0, \ldots, R_i, \ldots, R_n) \rightarrow \text{op}_n(R_0, \ldots, R'_i, \ldots, R_n)\]

\[(F-Eqv)\]

\[R \equiv R' \quad R \rightarrow R_1 \quad R_1 \equiv R'_1\]

\[R' \rightarrow R'_1\]
The uncontrolled reversible semantics

- The forward rules of the uncontrolled reversible semantics:

\[
(F-\text{SCM-Act}) \quad P_1 \mid \ldots \mid P_n \rightarrow T[Q_1, \ldots, Q_m] \quad j_1, \ldots, j_m \text{ are fresh keys}
\]

\[
k_1 : P_1 \mid \ldots \mid k_n : P_n \rightarrow T[j_1 : Q_1, \ldots, j_m : Q_m] |
\]

\[
[k_1 : P_1 \mid \ldots \mid k_n : P_n; T[j_1 : \bullet_1, \ldots, j_m : \bullet_m]]
\]

\[
(F-\text{SCM-OPN}) \quad R_i \rightarrow R_i' \quad (\text{key}(R_i') \setminus \text{key}(R_i)) \cap (\text{key}(R_0, \ldots, R_{i-1}, R_{i+1}, \ldots, R_n) = \emptyset
\]

\[
op_n(R_0, \ldots, R_i, \ldots, R_n) \rightarrow \op_n(R_0, \ldots, R_i', \ldots, R_n)
\]

\[
(F-\text{EQV}) \quad R \equiv R' \quad R \rightarrow R_1 \quad R_1 \equiv R_1'
\]

\[
R' \rightarrow R_1'
\]
The uncontrolled reversible semantics

- The forward rules of the uncontrolled reversible semantics:

\[(F\text{-}SCM\text{-}Act)\]

\[
\frac{P_1 | \ldots | P_n \Rightarrow T[Q_1, \ldots, Q_m]}{k_1 : P_1 | \ldots | k_n : P_n \Rightarrow T[j_1 : Q_1, \ldots, j_m : Q_m] | [k_1 : P_1 | \ldots | k_n : P_n; T[j_1 : \bullet_1, \ldots, j_m : \bullet_m]]}
\]

\[(F\text{-}SCM\text{-}Opn)\]

\[
\frac{R_i \Rightarrow R'_i \quad \left(\text{key}(R'_i) \setminus \text{key}(R_i)\right) \cap \left(\text{key}(R_0, \ldots, R_{i-1}, R_{i+1}, \ldots, R_n) = \emptyset \right)}{op_n(R_0, \ldots, R_i, \ldots, R_n) \Rightarrow op_n(R_0, \ldots, R'_i, \ldots, R_n)}
\]

\[(F\text{-}Eqv)\]

\[
\frac{R \equiv R' \quad R \Rightarrow R_1 \quad R_1 \equiv R'_1}{R' \Rightarrow R'_1}
\]

- The backward rules are symmetric w.r.t. the forward ones.
Example: consider system $R = k_1 : a(P_1 \mid P_2) \mid k_2 : a(X) \triangleright X$. 
• **Example:** consider system $R = k_1 : a\langle P_1 | P_2 \rangle | k_2 : a(X) \triangleright X$.

  • **forward step:**

    \[
    k_1 : a\langle P_1 | P_2 \rangle | k_2 : a(X) \triangleright X \rightarrow
    \]
○ **Example:** consider system \( R = k_1 : a(P_1 | P_2) | k_2 : a(X) \triangleright X \).

- **forward step:**

\[
k_1 : a(P_1 | P_2) | k_2 : a(X) \triangleright X \implies j_1 : P_1 | j_2 : P_2
\]
Example: consider system $R = k_1 : a\langle P_1 \mid P_2 \rangle \mid k_2 : a(X) \triangleright X$.

- forward step:

$$k_1 : a\langle P_1 \mid P_2 \rangle \mid k_2 : a(X) \triangleright X \rightarrow j_1 : P_1 \mid j_2 : P_2 \mid [R; T[j_1 : \cdot_1, j_2 : \cdot_2]]$$
Example: consider system $R = k_1 : a(P_1 \mid P_2) \mid k_2 : a(X) \triangleright X$.

- forward step:

$$k_1 : a(P_1 \mid P_2) \mid k_2 : a(X) \triangleright X \rightarrow j_1 : P_1 \mid j_2 : P_2 \mid [R ; T[j_1 : \bullet_1 , j_2 : \bullet_2]]$$

where $T = j_1 : \bullet_1 \mid j_2 : \bullet_2$.
- **Example**: consider system $R = k_1 : a(P_1 | P_2) | k_2 : a(X) ▽ X$.

  - **forward step**:

    $k_1 : a(P_1 | P_2) | k_2 : a(X) ▽ X \rightarrow j_1 : P_1 | j_2 : P_2 | [R; T[j_1 : \bullet_1, j_2 : \bullet_2]]$

    where $T = j_1 : \bullet_1 | j_2 : \bullet_2$.

  - **backward step**:

    $j_1 : P_1 | j_2 : P_2 | [R; T[j_1 : \bullet_1, j_2 : \bullet_2]] \rightsquigarrow$
Example: consider system \( R = k_1 : a\langle P_1 | P_2 \rangle | k_2 : a(X) \triangleright X \).

- **forward step:**

\[
k_1 : a\langle P_1 | P_2 \rangle | k_2 : a(X) \triangleright X \rightarrow j_1 : P_1 | j_2 : P_2 | [R; T[j_1 : \bullet_1, j_2 : \bullet_2]]
\]

where \( T = j_1 : \bullet_1 | j_2 : \bullet_2 \).

- **backward step:**

\[
j_1 : P_1 | j_2 : P_2 | [R; T[j_1 : \bullet_1, j_2 : \bullet_2]] \rightsquigarrow k_1 : a\langle P_1 | P_2 \rangle | k_2 : a(X) \triangleright X
\]
Concurrent transitions

- We extract a notion of concurrency from the reversible syntax.

- Transitions $t$ of a system $R$ are defined as:

  $$ t : R \xrightarrow{\mu} R' $$

  where $\mu$ is the memory created by the transition, if it is forward, or consumed by it, if it is backward.

- The function $\text{key}(\cdot)$ computes the set of keys of a given system.

Definition (Concurrent transitions)

Two coinitial transitions $t' : R \xrightarrow{\mu'} R'$ and $t'' : R \xrightarrow{\mu''} R''$ are concurrent, written $t' \sim_c t''$, if $\text{key}(\mu') \cap \text{key}(\mu'') = \emptyset$. Coinitial transitions which are not concurrent are in conflict.
Properties

- Properties which our framework satisfies are:

  - Loop Lemma [1] - every action can be undone;
  - Parabolic Lemma [1] - each reversible computation can be rearranged as a backward computation, followed by a forward computation;
  - Causal Consistency [1] - the correct history and causality information is stored;
  - Causal Safety [2] - an action cannot be reversed until all actions caused by it have been reversed;
  - Causal Liveness [2] - actions can be reversed in any order consistent with Causal Safety.

Properties

Properties which our framework satisfies are:

- **Loop Lemma** [1] - every action can be undone;

Properties

- Properties which our framework satisfies are:
  
  - **Loop Lemma** [1] - every action can be undone;
  
  - **Parabolic Lemma** [1] - each reversible computation can be rearranged as a backward computation, followed by a forward computation;

Properties

- Properties which our framework satisfies are:
  
  - **Loop Lemma** [1] - every action can be undone;
  
  - **Parabolic Lemma** [1] - each reversible computation can be rearranged as a backward computation, followed by a forward computation;
  
  - **Causal Consistency** [1] - the correct history and causality information is stored;
  
Properties which our framework satisfies are:

- **Loop Lemma** [1] - every action can be undone;

- **Parabolic Lemma** [1] - each reversible computation can be rearranged as a backward computation, followed by a forward computation;

- **Causal Consistency** [1] - the correct history and causality information is stored;

- **Causal Safety** [2] - an action cannot be reversed until all actions caused by it have been reversed;

Properties

- Properties which our framework satisfies are:
  - **Loop Lemma** [1] - every action can be undone;
  - **Parabolic Lemma** [1] - each reversible computation can be rearranged as a backward computation, followed by a forward computation;
  - **Causal Consistency** [1] - the correct history and causality information is stored;
  - **Causal Safety** [2] - an action cannot be reversed until all actions caused by it have been reversed;
  - **Causal Liveness** [2] - actions can be reversed in any order consistent with Causal Safety.

Case study: Core Erlang

- A **Core Erlang** system [1] is defined as:

\[ E := \langle p, \theta, e \rangle | \]

where

- \( \langle p, \theta, e \rangle \) - a process with a pid \( p \) evaluating expression \( e \) in environment \( \theta \);
Case study: Core Erlang

- A Core Erlang system [1] is defined as:

\[ E := \langle p, \theta, e \rangle \mid (p, p', v) \mid \]

where

- \( \langle p, \theta, e \rangle \) - a process with a pid \( p \) evaluating expression \( e \) in environment \( \theta \);
- \((p, p', v)\) - a message carrying value \( v \) sent by the process with pid \( p \) to the one with pid \( p' \).

Case study: Core Erlang

- A **Core Erlang** system [1] is defined as:

\[ E := \langle p, \theta, e \rangle \mid (p, p', v) \mid (E_1 \mid E_2) \]

where

- \( \langle p, \theta, e \rangle \) - a process with a pid \( p \) evaluating expression \( e \) in environment \( \theta \);
- \((p, p', v)\) - a message carrying value \( v \) sent by the process with pid \( p \) to the one with pid \( p' \).

Case study: Core Erlang

- A Core Erlang system [1] is defined as:

\[ E := \langle p, \theta, e \rangle \mid (p, p', v) \mid (E_1 \mid E_2) \]

where

- \( \langle p, \theta, e \rangle \) - a process with a pid \( p \) evaluating expression \( e \) in environment \( \theta \);
- \( (p, p', v) \) - a message carrying value \( v \) sent by the process with pid \( p \) to the one with pid \( p' \).

- A reversible Core Erlang configuration, is defined by the following grammar:

\[ R ::= k : \langle p, \theta, e \rangle \mid k : (p, p', v) \mid (R_1 \mid R_2) \mid [R; C] \]

Example: sample instances for rules (Send), (F-Send) and (B-Send)

- Rule $(\text{SEND})$ of Core Erlang semantics:

\[
\langle p, \theta, p'!5 \rangle \leftrightarrow \langle p, \theta, 5 \rangle \mid (p, p', 5)
\]
Example: sample instances for rules (Send), (F-Send) and (B-Send)

- **Rule (Send)** of Core Erlang semantics:
  \[
  \langle p, \theta, p'!5 \rangle \leftrightarrow \langle p, \theta, 5 \rangle | (p, p', 5)
  \]

- **Forward** rule (F-Send) of the reversible semantics for Erlang:
  \[
  k : \langle p, \theta, p'!5 \rangle \rightarrow
  \]

- **Backward** rule (B-Send) of the reversible semantics for Erlang:
Example: sample instances for rules (Send), (F-Send) and (B-Send)

- Rule \(\text{SEND}\) of Core Erlang semantics:

\[
\langle p, \theta, p'!5 \rangle \leftrightarrow \langle p, \theta, 5 \rangle | (p, p', 5)
\]

- Forward rule \(\text{F-SEND}\) of the reversible semantics for Erlang:

\[
k : \langle p, \theta, p'!5 \rangle \rightarrow k_1 : \langle p, \theta, 5 \rangle | k_2 : (p, p', 5) |
\]
Example: sample instances for rules (Send), (F-Send) and (B-Send)

- Rule (Send) of Core Erlang semantics:

  \[ \langle p, \theta, p'!5 \rangle \rightarrow \langle p, \theta, 5 \rangle \mid (p, p', 5) \]

- Forward rule (F-Send) of the reversible semantics for Erlang:

  \[ k : \langle p, \theta, p'!5 \rangle \rightarrow k_1 : \langle p, \theta, 5 \rangle \mid k_2 : (p, p', 5) \mid [k : \langle p, \theta, p'!5 \rangle; k_1 : \bullet_1 \mid k_2 : \bullet_2] \]
Example: sample instances for rules (Send), (F-Send) and (B-Send)

- **Rule (Send)** of Core Erlang semantics:

  \[
  \langle p, \theta, p'!5 \rangle \hookrightarrow \langle p, \theta, 5 \rangle \mid (p, p', 5)
  \]

- **Forward rule (F-Send)** of the reversible semantics for Erlang:

  \[
  k : \langle p, \theta, p'!5 \rangle \rightarrow k_1 : \langle p, \theta, 5 \rangle \mid k_2 : (p, p', 5) \mid [k : \langle p, \theta, p'!5 \rangle; k_1 : \bullet_1 \mid k_2 : \bullet_2]
  \]

- **Backward rule (B-Send)** of the reversible semantics for Erlang:

  \[
  k_1 : \langle p, \theta, 5 \rangle \mid k_2 : (p, p', 5) \mid [k : \langle p, \theta, p'!5 \rangle; k_1 : \bullet_1 \mid k_2 : \bullet_2] \rightarrow
  \]

Ivan Lanese and Doriana Medić - A general approach to derive uncontrolled reversible semantics 15
Example: sample instances for rules (Send), (F-Send) and (B-Send)

- **Rule** \((\text{Send})\) of Core Erlang semantics:

\[
\langle p, \theta, p'!5 \rangle \hookrightarrow \langle p, \theta, 5 \rangle \mid (p, p', 5)
\]

- **Forward rule** \((\text{F-Send})\) of the reversible semantics for Erlang:

\[
k : \langle p, \theta, p'!5 \rangle \rightarrow k_1 : \langle p, \theta, 5 \rangle \mid k_2 : (p, p', 5) \mid [k : \langle p, \theta, p'!5 \rangle; k_1 : \bullet_1 \mid k_2 : \bullet_2]
\]

- **Backward rule** \((\text{B-Send})\) of the reversible semantics for Erlang:

\[
k_1 : \langle p, \theta, 5 \rangle \mid k_2 : (p, p', 5) \mid [k : \langle p, \theta, p'!5 \rangle; k_1 : \bullet_1 \mid k_2 : \bullet_2] \leadsto k : \langle p, \theta, p'!5 \rangle
\]
Correspondence between two reversible semantics for Erlang:

- We prove the following results relating our reversible semantics and the one in [1]:

**Theorem (Causal correspondence)**

Two coinitial transitions $t_1$ and $t_2$ of our reversible Core Erlang semantics are in conflict according to [1] iff they are in conflict according to our definition.

**Theorem**

The reversible semantics of Erlang in [1] and our reversible semantics of Erlang are strong back and forth barbed bisimilar.

A general idea about links and their role in Erlang

- original semantics of reversible Core Erlang [1] did not cover error handling;
- remote error handling is the key aspect of the Erlang language;
- it is based on links along which errors are propagated;
- links can be created by constructs such as `link()` and `spawn_link()`;
- our approach can deal with the remote error handling mechanism based on links

Conclusion and future work:

○ To summarise:
  • a fully automatic method to extend a given forward model to a reversible one;
  • case studies: reversible extensions of HO\(\pi\)-calculus and Core Erlang;
  • the obtained reversible semantics are equivalent to the ones in the literature;
  • we tackled Core Erlang constructs for remote error handling based on links.

○ Future work:
  • addition of control mechanisms such as irreversible actions, rollback operators or energy potentials;
  • adaptation of our approach to handle further forward models (shared memory models);
  • extension of CauDEr implementation;
Thank you for attention 😊

Questions?