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 π-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:

$$N ::= P \ | \ op_n(N_1, \ldots, N_n) \ | \ \mathbf{0}$$

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;
- $\mathbf{0}$ represents the empty system.

Running example: HOπ-calculus

- The syntax of the asynchronous HOπ-calculus is:

$$P, Q ::= a\langle P\rangle \mid a(X) \triangleright P \mid \ (P \mid Q) \ \mid \nu a\,(P) \mid X \mid \mathbf{0}$$

– we separate entities from systems;

– an entity is any HOπ process whose topmost operator is neither a parallel composition nor a restriction nor $\mathbf{0}$.

Running example: HOπ-calculus

- The syntax of the asynchronous HOπ-calculus is:

$$P, Q ::= a\langle P\rangle \mid a(X) \triangleright P \mid \ (P \mid Q) \ \mid \nu a\,(P) \mid X \mid \mathbf{0}$$

  – we separate entities from systems;

  – an entity is any HOπ process whose topmost operator is neither a parallel composition nor a restriction nor $\mathbf{0}$.

- The syntax of systems is defined as:

$$N := P \mid \ (N_1 \mid N_2) \ \mid \nu a\,(N) \mid \mathbf{0}$$

where the operators $\mid$ and $\mathbf{0}$ are required by our framework;

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 $\mathbf{0}$.

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 $\mathbf{0}$.

◦ 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 $\mathbf{0}$.

- 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 HOπ structural rule is:

$$(\textsc{ParC}) \;\; P \mid Q \equiv Q \mid P$$

- exploits contexts of the form $\bullet_1 \mid \bullet_2$ and $\bullet_2 \mid \bullet_1$

The reduction semantics of the forward model

$$(\textsc{Scm-Act})\ \frac{}{P_1 \mid \ldots \mid P_n \rightarrowtail T[Q_1, \ldots, Q_m]} \qquad (\textsc{Eqv})\ \frac{N \equiv N' \quad N \rightarrowtail N_1 \quad N_1 \equiv N_1'}{N' \rightarrowtail N_1'}$$

$$(\textsc{Scm-Opn})\ \frac{N_i \rightarrowtail N_i'}{op_n(N_0, \ldots, N_i, \ldots, N_n) \rightarrowtail op_n(N_0, \ldots, N_i', \ldots, N_n)}$$

$$(\textsc{Par})\ \frac{N \rightarrowtail N'}{N \mid N_1 \rightarrowtail N' \mid N_1}$$

The reduction semantics of the forward model

$$(\textsc{Scm-Act})\ \frac{}{P_1 \mid \ldots \mid P_n \rightarrowtail T[Q_1, \ldots, Q_m]} \qquad (\textsc{Eqv})\ \frac{N \equiv N' \quad N \rightarrowtail N_1 \quad N_1 \equiv N_1'}{N' \rightarrowtail N_1'}$$

$$(\textsc{Scm-Opn})\ \frac{N_i \rightarrowtail N_i'}{op_n(N_0, \ldots, N_i, \ldots, N_n) \rightarrowtail op_n(N_0, \ldots, N_i', \ldots, N_n)}$$

$$(\textsc{Par})\ \frac{N \rightarrowtail N'}{N \mid N_1 \rightarrowtail N' \mid N_1}$$

The reduction semantics of the forward model

$$(\textsc{Scm-Act})\ \frac{}{P_1 \mid \ldots \mid P_n \rightarrowtail T[Q_1, \ldots, Q_m]} \qquad (\textsc{Eqv})\ \frac{N \equiv N' \quad N \rightarrowtail N_1 \quad N_1 \equiv N_1'}{N' \rightarrowtail N_1'}$$

$$(\textsc{Scm-Opn})\ \frac{N_i \rightarrowtail N_i'}{op_n(N_0, \ldots, N_i, \ldots, N_n) \rightarrowtail op_n(N_0, \ldots, N_i', \ldots, N_n)}$$

$$(\textsc{Par})\ \frac{N \rightarrowtail N'}{N \mid N_1 \rightarrowtail N' \mid N_1}$$

The reduction semantics of the forward model

$$(\textsc{Scm-Act})\ \frac{}{P_1 \mid \ldots \mid P_n \rightarrowtail T[Q_1,\ldots,Q_m]} \qquad (\textsc{Eqv})\ \frac{N \equiv N' \quad N \rightarrowtail N_1 \quad N_1 \equiv N_1'}{N' \rightarrowtail N_1'}$$

$$(\textsc{Scm-Opn})\ \frac{N_i \rightarrowtail N_i'}{op_n(N_0,\ldots,N_i,\ldots,N_n) \rightarrowtail op_n(N_0,\ldots,N_i',\ldots,N_n)}$$

$$(\textsc{Par})\ \frac{N \rightarrowtail N'}{N \mid N_1 \rightarrowtail N' \mid N_1}$$

The reduction semantics of the forward model

$$(\textsc{Scm-Act})\ \frac{}{P_1 \mid \ldots \mid P_n \rightarrowtail T[Q_1, \ldots, Q_m]} \qquad (\textsc{Eqv})\ \frac{N \equiv N' \quad N \rightarrowtail N_1 \quad N_1 \equiv N_1'}{N' \rightarrowtail N_1'}$$

$$(\textsc{Scm-Opn})\ \frac{N_i \rightarrowtail N_i'}{op_n(N_0, \ldots, N_i, \ldots, N_n) \rightarrowtail op_n(N_0, \ldots, N_i', \ldots, N_n)}$$

$$(\textsc{Par})\ \frac{N \rightarrowtail N'}{N \mid N_1 \rightarrowtail N' \mid N_1}$$

The reduction semantics of the forward model

$$(\textsc{Scm-Act})\ \frac{}{P_1 \mid \ldots \mid P_n \rightarrowtail T[Q_1, \ldots, Q_m]} \qquad (\textsc{Eqv})\ \frac{N \equiv N' \quad N \rightarrowtail N_1 \quad N_1 \equiv N_1'}{N' \rightarrowtail N_1'}$$

$$(\textsc{Scm-Opn})\ \frac{N_i \rightarrowtail N_i'}{op_n(N_0, \ldots, N_i, \ldots, N_n) \rightarrowtail op_n(N_0, \ldots, N_i', \ldots, N_n)}$$

$$(\textsc{Par})\ \frac{N \rightarrowtail N'}{N \mid N_1 \rightarrowtail N' \mid N_1}$$

The reduction semantics of the forward model

$$(\textsc{Scm-Act})\ \frac{}{P_1 \mid \ldots \mid P_n \rightarrowtail T[Q_1, \ldots, Q_m]} \qquad (\textsc{Eqv})\ \frac{N \equiv N' \quad N \rightarrowtail N_1 \quad N_1 \equiv N_1'}{N' \rightarrowtail N_1'}$$

$$(\textsc{Scm-Opn})\ \frac{N_i \rightarrowtail N_i'}{op_n(N_0, \ldots, N_i, \ldots, N_n) \rightarrowtail op_n(N_0, \ldots, N_i', \ldots, N_n)}$$

$$(\textsc{Par})\ \frac{N \rightarrowtail N'}{N \mid N_1 \rightarrowtail N' \mid N_1}$$

Example: the communication rule of HOπ

- The communication rule $(\textsc{Act})$ of HOπ-calculus, is defined as:

$$(\textsc{Act}) \; \frac{}{a\langle Q\rangle \mid a(X) \triangleright P \rightarrowtail P\{Q/X\}}$$

Example: the communication rule of HOπ

- The communication rule (ACT) of HOπ-calculus, is defined as:

$$(\text{ACT})\ \frac{}{a\langle Q\rangle \mid a(X) \triangleright P \rightarrowtail P\{Q/X\}}$$

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

$$a\langle b\langle P\rangle \mid b(Y) \triangleright Y \mid c\langle Q\rangle\rangle \mid a(X) \triangleright X \rightarrowtail b\langle P\rangle \mid b(Y) \triangleright Y \mid c\langle Q\rangle$$

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

Definition of the reversible system

- The syntax of configurations R is as:

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

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 $\mathbf{0}$;
- $\bullet_i$ 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:

$$R ::= k : P \mid (R_1 \mid R_2) \mid \nu a\,(R) \mid \mathbf{0} \mid [R; C]$$

where P are the entities as in the underlying calculus.

- **Example:** the syntax of the reversible HOπ-calculus is defined as:

$$R ::= k : P \mid (R_1 \mid R_2) \mid \nu a\,(R) \mid \mathbf{0} \mid [R; C]$$

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:

$$(\text{F-Scm-Act})\ \frac{P_1 \mid \ldots \mid P_n \rightarrowtail T[Q_1, \ldots, Q_m] \qquad j_1, \ldots, j_m \text{ are fresh keys}}{k_1 : P_1 \mid \ldots \mid k_n : P_n \twoheadrightarrow T[j_1 : Q_1, \ldots, j_m : Q_m] \mid [k_1 : P_1 \mid \ldots \mid k_n : P_n ; T[j_1 : \bullet_1, \ldots, j_m : \bullet_m]]}$$

$$(\text{F-Scm-Opn})\ \frac{R_i \twoheadrightarrow R_i' \quad (\mathtt{key}(R_i') \setminus \mathtt{key}(R_i)) \cap (\mathtt{key}(R_0, \ldots, R_{i-1}, R_{i+1}, \ldots, R_n) = \emptyset}{op_n(R_0, \ldots, R_i, \ldots, R_n) \twoheadrightarrow op_n(R_0, \ldots, R_i', \ldots, R_n)}$$

$$(\text{F-Eqv})\ \frac{R \equiv R' \quad R \twoheadrightarrow R_1 \quad R_1 \equiv R_1'}{R' \twoheadrightarrow R_1'}$$

The uncontrolled reversible semantics

- The forward rules of the uncontrolled reversible semantics:

$$(\text{F-Scm-Act})\ \frac{P_1 \mid \ldots \mid P_n \rightarrowtail T[Q_1, \ldots, Q_m] \qquad j_1, \ldots, j_m \text{ are fresh keys}}{k_1 : P_1 \mid \ldots \mid k_n : P_n \twoheadrightarrow T[j_1 : Q_1, \ldots, j_m : Q_m] \mid [k_1 : P_1 \mid \ldots \mid k_n : P_n; T[j_1 : \bullet_1, \ldots, j_m : \bullet_m]]}$$

$$(\text{F-Scm-Opn})\ \frac{R_i \twoheadrightarrow R_i' \quad (\mathtt{key}(R_i') \setminus \mathtt{key}(R_i)) \cap (\mathtt{key}(R_0, \ldots, R_{i-1}, R_{i+1}, \ldots, R_n) = \emptyset}{op_n(R_0, \ldots, R_i, \ldots, R_n) \twoheadrightarrow op_n(R_0, \ldots, R_i', \ldots, R_n)}$$

$$(\text{F-Eqv})\ \frac{R \equiv R' \quad R \twoheadrightarrow R_1 \quad R_1 \equiv R_1'}{R' \twoheadrightarrow R_1'}$$

The uncontrolled reversible semantics

- The forward rules of the uncontrolled reversible semantics:

$$\text{(F-SCM-ACT)}\ \frac{P_1 \mid \ldots \mid P_n \rightarrowtail T[Q_1, \ldots, Q_m] \qquad j_1, \ldots, j_m \text{ are fresh keys}}{k_1 : P_1 \mid \ldots \mid k_n : P_n \twoheadrightarrow T[j_1 : Q_1, \ldots, j_m : Q_m] \mid [k_1 : P_1 \mid \ldots \mid k_n : P_n ; T[j_1 : \bullet_1, \ldots, j_m : \bullet_m]]}$$

$$\text{(F-SCM-OPN)}\ \frac{R_i \twoheadrightarrow R_i' \quad (\mathtt{key}(R_i') \setminus \mathtt{key}(R_i)) \cap (\mathtt{key}(R_0, \ldots, R_{i-1}, R_{i+1}, \ldots, R_n) = \emptyset}{op_n(R_0, \ldots, R_i, \ldots, R_n) \twoheadrightarrow op_n(R_0, \ldots, R_i', \ldots, R_n)}$$

$$\text{(F-EQV)}\ \frac{R \equiv R' \quad R \twoheadrightarrow R_1 \quad R_1 \equiv R_1'}{R' \twoheadrightarrow R_1'}$$

The uncontrolled reversible semantics

- The forward rules of the uncontrolled reversible semantics:

$$\text{(F-SCM-ACT)}\ \frac{P_1 \mid \ldots \mid P_n \rightarrowtail T[Q_1, \ldots, Q_m] \qquad j_1, \ldots, j_m \text{ are fresh keys}}{k_1 : P_1 \mid \ldots \mid k_n : P_n \twoheadrightarrow T[j_1 : Q_1, \ldots, j_m : Q_m] \mid [k_1 : P_1 \mid \ldots \mid k_n : P_n; T[j_1 : \bullet_1, \ldots, j_m : \bullet_m]]}$$

$$\text{(F-SCM-OPN)}\ \frac{R_i \twoheadrightarrow R_i' \quad (\mathtt{key}(R_i') \setminus \mathtt{key}(R_i)) \cap (\mathtt{key}(R_0, \ldots, R_{i-1}, R_{i+1}, \ldots, R_n) = \emptyset}{op_n(R_0, \ldots, R_i, \ldots, R_n) \twoheadrightarrow op_n(R_0, \ldots, R_i', \ldots, R_n)}$$

$$\text{(F-EQV)}\ \frac{R \equiv R' \quad R \twoheadrightarrow R_1 \quad R_1 \equiv R_1'}{R' \twoheadrightarrow R_1'}$$

The uncontrolled reversible semantics

- The forward rules of the uncontrolled reversible semantics:

$$(\text{F-Scm-Act}) \quad \frac{P_1 \mid \ldots \mid P_n \rightarrowtail T[Q_1, \ldots, Q_m] \qquad j_1, \ldots, j_m \text{ are fresh keys}}{\begin{array}{l} k_1 : P_1 \mid \ldots \mid k_n : P_n \twoheadrightarrow T[j_1 : Q_1, \ldots, j_m : Q_m] \mid \\ \qquad\qquad [k_1 : P_1 \mid \ldots \mid k_n : P_n; T[j_1 : \bullet_1, \ldots, j_m : \bullet_m]] \end{array}}$$

$$(\text{F-Scm-Opn}) \quad \frac{R_i \twoheadrightarrow R_i' \quad (\mathtt{key}(R_i') \setminus \mathtt{key}(R_i)) \cap (\mathtt{key}(R_0, \ldots, R_{i-1}, R_{i+1}, \ldots, R_n) = \emptyset}{op_n(R_0, \ldots, R_i, \ldots, R_n) \twoheadrightarrow op_n(R_0, \ldots, R_i', \ldots, R_n)}$$

$$(\text{F-Eqv}) \quad \frac{R \equiv R' \quad R \twoheadrightarrow R_1 \quad R_1 \equiv R_1'}{R' \twoheadrightarrow R_1'}$$

The uncontrolled reversible semantics

- The forward rules of the uncontrolled reversible semantics:

$$(\text{F-Scm-Act})\ \frac{P_1 \mid \ldots \mid P_n \rightarrowtail T[Q_1, \ldots, Q_m] \qquad j_1, \ldots, j_m \text{ are fresh keys}}{\begin{array}{l} k_1 : P_1 \mid \ldots \mid k_n : P_n \twoheadrightarrow T[j_1 : Q_1, \ldots, j_m : Q_m] \mid \\ \qquad\qquad\qquad\qquad [k_1 : P_1 \mid \ldots \mid k_n : P_n; T[j_1 : \bullet_1, \ldots, j_m : \bullet_m]] \end{array}}$$

$$(\text{F-Scm-Opn})\ \frac{R_i \twoheadrightarrow R_i' \quad (\mathtt{key}(R_i') \setminus \mathtt{key}(R_i)) \cap (\mathtt{key}(R_0, \ldots, R_{i-1}, R_{i+1}, \ldots, R_n) = \emptyset}{op_n(R_0, \ldots, R_i, \ldots, R_n) \twoheadrightarrow op_n(R_0, \ldots, R_i', \ldots, R_n)}$$

$$(\text{F-Eqv})\ \frac{R \equiv R' \quad R \twoheadrightarrow R_1 \quad R_1 \equiv R_1'}{R' \twoheadrightarrow R_1'}$$

The uncontrolled reversible semantics

- The forward rules of the uncontrolled reversible semantics:

$$\text{(F-SCM-ACT)}\ \frac{P_1 \mid \ldots \mid P_n \rightarrowtail T[Q_1, \ldots, Q_m] \qquad j_1, \ldots, j_m \text{ are fresh keys}}{k_1 : P_1 \mid \ldots \mid k_n : P_n \twoheadrightarrow T[j_1 : Q_1, \ldots, j_m : Q_m] \mid [k_1 : P_1 \mid \ldots \mid k_n : P_n; T[j_1 : \bullet_1, \ldots, j_m : \bullet_m]]}$$

$$\text{(F-SCM-OPN)}\ \frac{R_i \twoheadrightarrow R_i' \quad (\mathtt{key}(R_i') \setminus \mathtt{key}(R_i)) \cap (\mathtt{key}(R_0, \ldots, R_{i-1}, R_{i+1}, \ldots, R_n) = \emptyset}{op_n(R_0, \ldots, R_i, \ldots, R_n) \twoheadrightarrow op_n(R_0, \ldots, R_i', \ldots, R_n)}$$

$$\text{(F-EQV)}\ \frac{R \equiv R' \quad R \twoheadrightarrow R_1 \quad R_1 \equiv R_1'}{R' \twoheadrightarrow R_1'}$$

The uncontrolled reversible semantics

- The forward rules of the uncontrolled reversible semantics:

$$\text{(F-SCM-ACT)}\ \frac{P_1 \mid \ldots \mid P_n \rightarrowtail T[Q_1, \ldots, Q_m] \qquad j_1, \ldots, j_m \text{ are fresh keys}}{\begin{array}{l} k_1 : P_1 \mid \ldots \mid k_n : P_n \twoheadrightarrow T[j_1 : Q_1, \ldots, j_m : Q_m] \mid \\ \qquad\qquad\qquad\qquad [k_1 : P_1 \mid \ldots \mid k_n : P_n; T[j_1 : \bullet_1, \ldots, j_m : \bullet_m]] \end{array}}$$

$$\text{(F-SCM-OPN)}\ \frac{R_i \twoheadrightarrow R_i' \quad (\mathtt{key}(R_i') \setminus \mathtt{key}(R_i)) \cap (\mathtt{key}(R_0, \ldots, R_{i-1}, R_{i+1}, \ldots, R_n) = \emptyset}{op_n(R_0, \ldots, R_i, \ldots, R_n) \twoheadrightarrow op_n(R_0, \ldots, R_i', \ldots, R_n)}$$

$$\text{(F-EQV)}\ \frac{R \equiv R' \quad R \twoheadrightarrow R_1 \quad R_1 \equiv R_1'}{R' \twoheadrightarrow R_1'}$$

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

- **Example:** consider system $R = k_1 : a\langle P_1 \mid P_2 \rangle \mid k_2 : a(X) \triangleright X$.

- **Example:** consider system $R = k_1 : a\langle P_1 \mid P_2 \rangle \mid k_2 : a(X) \triangleright X$.

- forward step:

$$k_1 : a\langle P_1 \mid P_2 \rangle \mid k_2 : a(X) \triangleright X \twoheadrightarrow$$

- **Example:** consider system $R = k_1 : a\langle P_1 \mid P_2 \rangle \mid k_2 : a(X) \triangleright X$.

- forward step:

$$k_1 : a\langle P_1 \mid P_2 \rangle \mid k_2 : a(X) \triangleright X \twoheadrightarrow j_1 : P_1 \mid j_2 : P_2 \mid$$

- **Example:** consider system $R = k_1 : a\langle P_1 \mid P_2 \rangle \mid k_2 : a(X) \triangleright X$.

- forward step:

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

- **Example:** consider system $R = k_1 : a\langle P_1 \mid P_2 \rangle \mid k_2 : a(X) \triangleright X$.

- forward step:

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

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

- **Example:** consider system $R = k_1 : a\langle P_1 \mid P_2\rangle \mid k_2 : a(X) \triangleright X$.

- forward step:

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

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

- backward step:

$$j_1 : P_1 \mid j_2 : P_2 \mid [R; T[j_1 : \bullet_1, j_2 : \bullet_2]] \rightsquigarrow$$

- **Example:** consider system $R = k_1 : a\langle P_1 \mid P_2\rangle \mid k_2 : a(X) \triangleright X$.

- forward step:

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

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

- backward step:

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

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 μ is the memory created by the transition, if it is forward, or consumed by it, if it is backward.

- The function $\mathtt{key}(\cdot)$ 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 $t' \smile_c t''$, if $\mathtt{key}(\mu') \cap \mathtt{key}(\mu'') = \emptyset$. Coinitial transitions which are not concurrent are in **conflict**.

Properties

- Properties which our framework satisfies are:

[1] V. Danos and J. Krivine: Reversible communicating systems. In: CONCUR 2004.
[2] I. Lanese, I. C. C. Phillips and I. Ulidowski: An axiomatic approach to reversible computation. In FOSSACS 2020.

Properties

- Properties which our framework satisfies are:
  - Loop Lemma [1] - every action can be undone;

[1] V. Danos and J. Krivine: Reversible communicating systems. In: CONCUR 2004.
[2] I. Lanese, I. C. C. Phillips and I. Ulidowski: An axiomatic approach to reversible computation. In FOSSACS 2020.

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;

[1] V. Danos and J. Krivine: Reversible communicating systems. In: CONCUR 2004.
[2] I. Lanese, I. C. C. Phillips and I. Ulidowski: An axiomatic approach to reversible computation. In FOSSACS 2020.

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;

[1] V. Danos and J. Krivine: Reversible communicating systems. In: CONCUR 2004.
[2] I. Lanese, I. C. C. Phillips and I. Ulidowski: An axiomatic approach to reversible computation. In FOSSACS 2020.

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;

[1] V. Danos and J. Krivine: Reversible communicating systems. In: CONCUR 2004.
[2] I. Lanese, I. C. C. Phillips and I. Ulidowski: An axiomatic approach to reversible computation. In FOSSACS 2020.

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.

[1] V. Danos and J. Krivine: Reversible communicating systems. In: CONCUR 2004.
[2] I. Lanese, I. C. C. Phillips and I. Ulidowski: An axiomatic approach to reversible computation. In FOSSACS 2020.

Case study: Core Erlang

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

$$E := \langle p, \theta, e \rangle \mid$$

where

- $\langle p, \theta, e \rangle$ - a process with a pid p evaluating expression e in environment θ;

[1] I. Lanese, A. Palacios and G. Vidal (2019): Causal-consistent replay debugging for message passing programs. In: Technical report, DSIC, Universitat Politecnica de Valencia

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 θ;
- (p, p', v) - a message carrying value v sent by the process with pid p to the one with pid p'.

[1] I. Lanese, A. Palacios and G. Vidal (2019): Causal-consistent replay debugging for message passing programs. In: Technical report, DSIC, Universitat Politecnica de Valencia

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 θ;
- (p, p', v) - a message carrying value v sent by the process with pid p to the one with pid p'.

[1] I. Lanese, A. Palacios and G. Vidal (2019): Causal-consistent replay debugging for message passing programs. In: Technical report, DSIC, Universitat Politecnica de Valencia

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 θ;
- (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:

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

[1] I. Lanese, A. Palacios and G. Vidal (2019): Causal-consistent replay debugging for message passing programs. In: Technical report, DSIC, Universitat Politecnica de Valencia

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

- Rule $(\textsc{Send})$ of Core Erlang semantics:

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

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

- Rule $(\textsc{Send})$ of Core Erlang semantics:

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

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

$k : \langle p, \theta, p'!5\rangle \twoheadrightarrow$

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

- Rule $(\textsc{Send})$ of Core Erlang semantics:

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

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

$$k : \langle p, \theta, p'!5\rangle \twoheadrightarrow k_1 : \langle p, \theta, 5\rangle \mid k_2 : (p, p', 5) \mid$$

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

- Rule $(\textsc{Send})$ of Core Erlang semantics:

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

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

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

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

- Rule $(\textsc{Send})$ of Core Erlang semantics:

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

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

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

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

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

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

- Rule $(\textsc{Send})$ of Core Erlang semantics:

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

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

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

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

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

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.

[1] I. Lanese, A. Palacios and G. Vidal (2019): Causal-consistent replay debugging for message passing programs. In: Technical report, DSIC, Universitat Politecnica de Valencia

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

[1] I. Lanese, A. Palacios and G. Vidal (2019): Causal-consistent replay debugging for message passing programs. In: Technical report, DSIC, Universitat Politecnica de Valencia

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 HOπ-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?