Controlling Reversibility in $HO\pi$

Ivan Lanese\(^1\)  \hspace{1cm} Claudio Antares Mezzina\(^2\)  \hspace{1cm} Alan Schmitt\(^2\)
Jean-Bernard Stefani\(^2\)

Focus Team, University of Bologna/INRIA, Italy
INRIA Grenoble-Rhône-Alpes, France

September 8, 2011

Concur 2011
Roadmap

1. Reversibility

2. A small reversible language

3. Controlling Rollback
The case for reversibility

What if we could undo every action?

Language support for Recovery Oriented Computing (ROC)
Undo manifestations

- Application undo
- System undo (ROC)
- Logs and checkpoints
- Transaction rollback
- Distributed rollback-recovery
The case for reversibility

Enabling defeasible partial agreements

- Systematic rollback-recovery
- Different transaction models
The case for reversibility

Enabling causality tracking

- Debugging
- Diagnosis
- Simulation
- Fault isolation
Roadmap

1 Reversibility

2 A small reversible language

3 Controlling Rollback
Higher-order $\pi$-calculus

Syntax

\[
P, Q ::= 0 \quad \text{null process} \\
| \ X \quad \text{variable} \\
| \nu a. P \quad \text{new name} \\
| \ (P | Q) \quad \text{parallel composition} \\
| \ a\langle P\rangle \quad \text{message} \\
| \ (a(X) \triangleright P) \quad \text{trigger} \\
\]

\[a \in \mathcal{N}\]
A reversible $\text{HO}\pi: \rho\pi$

Computation is done by process passing

$$a\langle P \rangle \mid a(X) \triangleright X \rightarrow P$$

After communication no information about \textit{previous} state of the configuration.
A reversible $\text{HO}_\pi: \rho_\pi$

Computation is done by process passing

$$a\langle P \rangle | a(X) \triangleright X \rightarrow P$$

After communication no information about previous state of the configuration.
A reversible $\text{HO}^\pi: \rho^\pi$

To make $\text{HO}^\pi$ reversible:

- Log each action (message receipt)
- Uniquely identify action participants
A reversible \( \text{HO}_\pi \): \( \rho \pi \)

**Syntax**

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

\[
M, N ::= \text{configurations}
\]

\[
\begin{align*}
0 \\
\mid \nu u. M \\
\mid (M \mid N) \\
\mid \kappa : P \\
\mid [m; k]
\end{align*}
\]

\( a \in \mathcal{N}, \ k \in \mathcal{K}, \ u \in \mathcal{N} \cup \mathcal{K} \)
A reversible HO\(\pi\): \(\rho\pi\)

Intuitions:

- \(\kappa : P\) thread of computation (process) \(P\) uniquely identified by tag \(\kappa\)

- \([m; k]\) log of occurrence \(k\) of action \(m\) (message receipt)
A reversible HO$\pi$: $\rho\pi$

Reduction rules

**Forward:**

\[
m = (\kappa_1 : a\langle P \rangle) \mid (\kappa_2 : a(X) \triangleright Q)
\]

\[
(\kappa_1 : a\langle P \rangle) \mid (\kappa_2 : a(X) \triangleright Q) \twoheadrightarrow \nu k. (k : Q\{P/x\}) \mid [m; k]
\]

**Backward:**

\[
(k : P) \mid [m; k] \rightsquigarrow m
\]
Syntactical equivalence

(E.TAGP) \[ k : \prod_{i=1}^{n} \tau_i \equiv \nu \tilde{h}. \prod_{i=1}^{n} (\langle h_i, \tilde{h} \rangle \cdot k : \tau_i) \]
\[ \tilde{h} = \{ h_1, \ldots, h_n \} \]

\[ \tau = a\langle P \rangle \quad \lor \quad \tau = a(X) \triangleright P \]

Rule E.TAGP builds unique identifiers for parallel branches
Theorem (Full reversal)

\[ M \rightarrow^* N \text{ if and only if } N \rightarrow^* M \]
Roadmap

1. Reversibility

2. A small reversible language

3. Controlling Rollback
In $\rho\pi$ back tracking is **uncontrolled**

- **When do we get back?**
  - normal execution should go **forward**
  - enable backward execution with a specific primitive

- **How far do we get back?**
  - each action is logged and uniquely identified
  - $[M; k]$ as **checkpoint**: roll $k$
In $\rho_\pi$ back tracking is **uncontrolled**

- When do we get back?
  - normal execution should go **forward**
  - enable backward execution with a specific primitive

- How far do we get back?
  - each action is logged and **uniquely identified**
  - $[M; k]$ as **checkpoint**: roll $k$
In $\rho\pi$ back tracking is **uncontrolled**

- When do we get back?
  - normal execution should go **forward**
  - enable backward execution with a specific primitive

- How far do we get back?
  - each action is logged and **uniquely identified**
  - $[M; k]$ as **checkpoint**: roll $k$
How the roll primitive should work?

Naive implementation of roll $k$:

1. **collect** all the processes caused by $k$
2. **delete** them
3. **restore** the content of memory $[M; k]$

Reduction rules

\[
\begin{align*}
\text{(COM)} & \quad m = (\kappa_1 : a(P)) \mid (\kappa_2 : a(X) \triangleright_\gamma Q) \\
& \quad (\kappa_1 : a(P)) \mid (\kappa_2 : a(X) \triangleright_\gamma Q) \rightarrow \nu k. (k : Q^{k,P/\gamma,X}) \mid [m; k]
\end{align*}
\]

\[
\begin{align*}
\text{(NAIVE)} & \quad N \triangleright k \quad \text{complete}(N \mid [m; k] \mid (\kappa : \text{roll } k)) \\
& \quad N \mid [m; k] \mid (\kappa : \text{roll } k) \leadsto m \mid N \downarrow_k
\end{align*}
\]
How the roll primitive should work?

Naive implementation of roll $k$:

1. **collect** all the processes caused by $k$
2. **delete** them
3. **restore** the content of memory $[M; k]$}

**Reduction rules**

\[(\text{COM})\]

\[m = (\kappa_1 : a\langle P \rangle) \mid (\kappa_2 : a(X) \triangleright_\gamma Q)
\]

\[(\kappa_1 : a\langle P \rangle) \mid (\kappa_2 : a(X) \triangleright_\gamma Q) \rightarrow \nu k. (k : Q^{k,P/\gamma,x}) \mid [m; k]\]

\[(\text{NAIVE})\]

\[N \triangleright k \quad \text{complete}(N \mid [m; k] \mid (\kappa : \text{roll } k))\]

\[N \mid [m; k] \mid (\kappa : \text{roll } k) \leadsto m \mid N \frac{1}{k}\]
$k_1 : a\langle 0 \rangle$ (\(k_2 : a(X) \triangleright_\gamma b \langle \text{roll } \gamma \rangle\)) (\(k_3 : b(X) \triangleright c\langle 0 \rangle | X\))
Controlling Rollback

\[ k_1 : a\langle 0 \rangle \quad (k_2 : a(X) \triangleright b\langle \text{roll } \gamma \rangle) \quad (k_3 : b(X) \triangleright c\langle 0 \rangle | X) \]

\[ [k_1 : M \mid k_2 : N; k] \mid k : b\langle \text{roll } k \rangle \]
Controlling Rollback

\[ k_1 : a \langle 0 \rangle \quad (k_2 : a(X) \triangleright_\gamma b \langle \text{roll } \gamma \rangle) \quad (k_3 : b(X) \triangleright c \langle 0 \rangle | X) \]

\[ [k_1 : M \mid k_2 : N ; k] \]

\[ [k : M_1 \mid k_3 : N_1 ; k_4] \]

\[ \langle h_1, \tilde{h} \rangle \cdot k_4 : c \langle 0 \rangle \quad \langle h_2, \tilde{h} \rangle \cdot k_4 : \text{roll } k \]
Controlling Rollback

\[ k_1 : a\langle 0 \rangle \quad (k_2 : a(X) \triangleright \gamma b\langle \text{roll } \gamma \rangle) \quad (k_3 : b(X) \triangleright c\langle 0 \rangle | X) \]

\[ [k_1 : M \mid k_2 : N; k] \]

\[ [k : M_1 \mid k_3 : N_1; k_4] \]

\[ \langle h_1, \tilde{h} \rangle \cdot k_4 : c\langle 0 \rangle \quad \langle h_2, \tilde{h} \rangle \cdot k_4 : \text{roll } k \]
Controlling Rollback

\[ k_1 : a\langle 0 \rangle \quad (k_2 : a(X) \triangleright_{\gamma} b\langle \text{roll} \gamma \rangle) \quad (k_3 : b(X) \triangleright c\langle 0 \rangle | X) \]
Concurrent rolls

\[ \text{roll } k \quad \text{roll } k_1 \]

Mezzina (INRIA)
Concurrent rolls

Mezzina (INRIA)
Controlling Reversibility in $HO\pi$

September 8, 2011 17 / 31
Concurrent rolls

\begin{align*}
\text{roll } k & \quad \text{roll } k1 \\
\text{roll } k1 & \quad \text{roll } k
\end{align*}
Concurrent rolls

\[ k1 \]
Problem with concurrent **rolls**

- naive semantics is not unsound with respect to $\rho\pi$
- naive semantics is *insufficiently permissive* with respect to $\rho\pi$
Why a better semantics?

Abstract semantics should
- be as liberal as possible
- not unduly restrict implementations

An implementation of naive:
- will reach same backward states
- hard to characterize
- artificial, do not correspond to any policy
Controlling Rollback

Reduction rules: HL semantics

\[(\text{START}) \quad (\kappa_1 : \text{roll } k) \mid [m; k] \leadsto (\kappa_1 : \text{roll } k) \mid [m; k]^* \]

\[(\text{ROLL}) \quad \frac{N \triangleright k \quad \text{complete}(N \mid [m; k])}{N \mid [m; k]^* \leadsto m \mid N \nLeftarrow k} \]
Concurrent rolls

\[ k \]

\[ k_1 \]

\[ \text{roll } k \]

\[ \text{roll } k_1 \]
Concurrent rolls
Concurrent rolls

Mezzina (INRIA)

Controlling Reversibility in $HO\pi$

September 8, 2011 21 / 31
Concurrent rolls
Concurrent rolls
Concurrent rolls
Properties

Soundness of backward reductions

\[ M \rightsquigarrow^* M' \text{ then } M' \rightarrow^* M, \text{ with } M \text{ and } M' \text{ unmarked} \]

Completeness of backward reductions

\[ \phi(M) \rightsquigarrow_T^* N \text{ then } M \rightsquigarrow_{HL}^* M' \text{ with } M' = \phi(N). \]

- \( \phi(.) : HL \rightarrow \rho\pi \)
- \( T \) is a set of keys
- \( \rightsquigarrow_T \) is a \( \rho\pi \) reduction restricted to keys caused by \( T \)
High Level Semantics

- **sound** and **complete** with respect to $\rho\pi$
- good as specification semantics
- relies on global checks and atomic steps

far away from a real implementation
Low Level Implementation

- local checks
- use of asynchronous notifications
- rollback in two phases:
  1. top-down visit to gather all the causally dependent processes
  2. bottom-up visit to undo communications (one by one)
LL example

\[ \text{roll } k \]

Mezzina (INRIA)  Controlling Reversibility in $HO\pi$  September 8, 2011  25 / 31
LL example

roll k
k
roll k
LL example

roll k

Mezzina (INRIA)  Controlling Reversibility in $HO_\pi$  September 8, 2011  25 / 31
LL example
LL example
LL example
Properties

Correspondence HL vs LL

\[ M_{HL} \approx^c_{LL} M \]

Full Abstraction

\[ j(M)_{LL} \approx^c_{LL} j(N) \iff M_{HL} \approx^c_{HL} N \]

- \( M, N \) consistent \( HL \) configuration
- \( j(\cdot) \) injection from \( HL \) to \( LL \)
Conclusions

We presented a fine grained rollback primitive for $HO\pi$

- built on top of $\rho\pi$
- sound and complete with respect to $\rho\pi$

We presented a distributed semantics

- closer to an implementation
- **fully abstract** with respect to the high level semantics
Ongoing and Future works

- encode roll$_\pi$ into $HO\pi$ (✓)
- refine and relate the LL semantics to checkpoint and rollback schemes
  - localities, failures ...
- mix **dynamic compensations** and controlled rollback
Thank you!

Questions?
Low Level

**LL Reduction Rules 1/2**

**START** \( (\kappa_1 : \text{roll } k) | [m; k] \leadsto_{LL} (\kappa_1 : \text{roll } k) | [m; k]^\bullet | \text{rl } k \)

**SPAN** \( \text{rl } \kappa_1 | [\kappa_1 : P | M; k]^\circ \leadsto_{LL} [\kappa_1 : P | M; k]^\circ | \text{rl } k \)

**BRANCH** \( \frac{\langle h_i, \tilde{h} \rangle \cdot k \text{ occurs in } M}{\text{rl } k | M \leadsto_{LL} \prod_{h_i \in \tilde{h}} \text{rl } \langle h_i, \tilde{h} \rangle \cdot k | M} \)
Low Level

LL Reduction Rules 2/2

(U_P) \quad \text{rl } \kappa_1 \mid (\kappa_1 : P) \rightsquigarrow_{LL} [\kappa_1 : P] \quad \text{(STOP)} \quad [m; k]^\circ \mid [k : P] \rightsquigarrow_{LL} m

(GB) \quad \nu k. \prod_{i \in I} \text{rl } \kappa_i \equiv 0 \quad \kappa_i = k \lor \kappa_i = \langle h, \tilde{u} \rangle \cdot k

(TAGPFR) \quad [k : \prod_{i=1}^{n} \tau_i] \equiv_{LL} \nu \tilde{h}. \prod_{i=1}^{n} [\langle h_i, \tilde{h} \rangle \cdot k : \tau_i] \quad \tilde{h} = \{h_1, \ldots, h_n\} \quad n \geq 2