Controlling Reversibility in $HO\pi$

Ivan Lanese [1] **Claudio Antares Mezzina** [2] 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

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 π-calculus

Syntax

$$
\begin{array}{rcll}
P, Q & ::= & \mathbf{0} & \textit{null process} \\
& | & X & \textit{variable} \\
& | & \nu a.\, P & \textit{new name} \\
& | & (P \mid Q) & \textit{parallel composition} \\
& | & a\langle P \rangle & \textit{message} \\
& | & (a(X) \triangleright P) & \textit{trigger}
\end{array}
$$

$$a \in \mathcal{N}$$

A reversible HOπ:$\rho\pi$

Computation is done by process passing

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

After communication no information about **previous** state of the configuration.

A reversible HOπ:$\rho\pi$

Computation is done by process passing

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

After communication no information about **previous** state of the configuration.

A reversible HOπ:$\rho\pi$

To make HOπ reversible:

- Log each action (message receipt)
- Uniquely identify action participants

A reversible HOπ: $\rho\pi$

Syntax

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

$$\begin{array}{llr}
M, N ::= & & \textit{configurations} \\
& \mathbf{0} & \textit{null configuration} \\
& \mid \nu u.\, M & \textit{restriction} \\
& \mid (M \mid N) & \textit{parallel} \\
& \mid \kappa : P & \textit{thread} \\
& \mid [m; k] & \textit{memory}
\end{array}$$

$$a \in \mathcal{N},\ k \in \mathcal{K},\ u \in \mathcal{N} \cup \mathcal{K}$$

A reversible HOπ: $\rho\pi$

Intuitions:

- $\kappa : P$ thread of computation (process) P uniquely identified by tag κ
- $[m; k]$ log of occurrence k of action m (message receipt)

A reversible HOπ: $\rho\pi$

Reduction rules

FORWARD:

$$\frac{m = (\kappa_1 : a\langle P\rangle) \mid (\kappa_2 : a(X) \triangleright Q)}{(\kappa_1 : a\langle P\rangle) \mid (\kappa_2 : a(X) \triangleright Q) \twoheadrightarrow \nu k.\,(k : Q\{^P/_X\}) \mid [m; k]}$$

BACKWARD: $(k : P) \mid [m; k] \rightsquigarrow m$

A reversible HOπ: $\rho\pi$

Syntactical equivalence

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

$$\tau = a\langle P\rangle \quad \vee \quad \tau = a(X) \triangleright P$$

Rule E.TAGP builds unique identifiers for parallel branches

$\rho\pi$ results

Theorem (Full reversal)

$M \rightarrow^* N$ *if and only if* $N \rightarrow^* M$

$$\rightarrow \; = \; \twoheadrightarrow \cup \rightsquigarrow$$

Roadmap

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

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

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

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

$$(\text{COM})\quad \frac{m = (\kappa_1 : a\langle P\rangle) \mid (\kappa_2 : a(X) \triangleright_\gamma Q)}{(\kappa_1 : a\langle P\rangle) \mid (\kappa_2 : a(X) \triangleright_\gamma Q) \twoheadrightarrow \nu k.\,(k : Q\{^{k,P}/_{\gamma,X}\}) \mid [m;k]}$$

$$(\text{NAIVE})\quad \frac{N \blacktriangleright k \qquad \mathtt{complete}(N \mid [m;k] \mid (\kappa : \mathsf{roll}\ k))}{N \mid [m;k] \mid (\kappa : \mathsf{roll}\ k) \rightsquigarrow m \mid N\lightning_k}$$

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 $[M; k]$

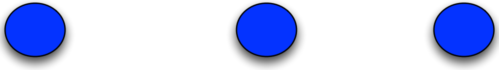
Reduction rules

$$(\textsc{Com})\quad \frac{m = (\kappa_1 : a\langle P\rangle) \mid (\kappa_2 : a(X) \triangleright_{\gamma} Q)}{(\kappa_1 : a\langle P\rangle) \mid (\kappa_2 : a(X) \triangleright_{\gamma} Q) \twoheadrightarrow \nu k.\, (k : Q\{^{k,P}/_{\gamma,X}\}) \mid [m; k]}$$

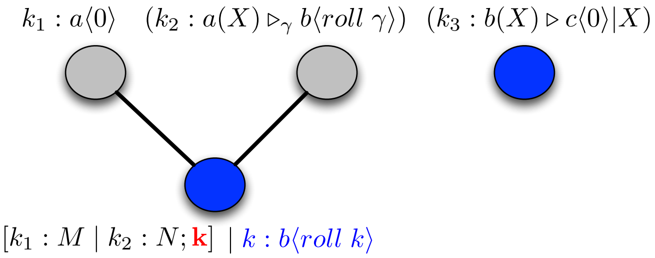
$$(\textsc{Naive})\quad \frac{N \blacktriangleright k \qquad \mathtt{complete}(N \mid [m; k] \mid (\kappa : \mathsf{roll}\ k))}{N \mid [m; k] \mid (\kappa : \mathsf{roll}\ k) \rightsquigarrow m \mid N \lightning_k}$$

Controlling Rollback

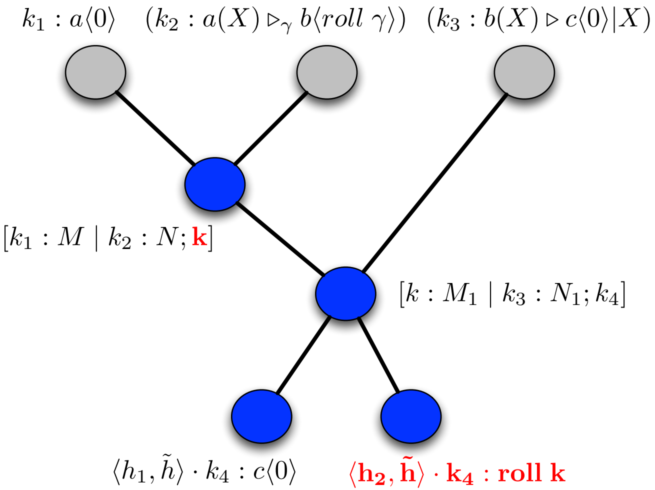
$k_1 : a\langle 0\rangle \quad (k_2 : a(X) \triangleright_\gamma b\langle roll\ \gamma\rangle) \quad (k_3 : b(X) \triangleright c\langle 0\rangle | X)$



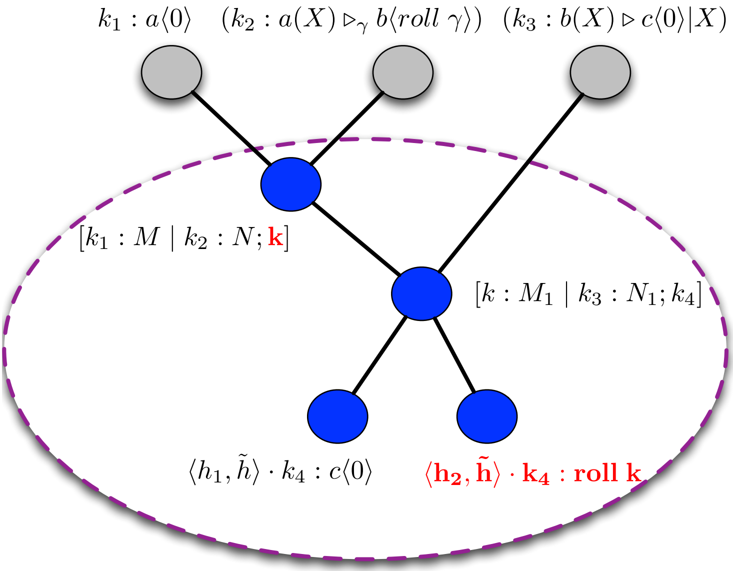
Controlling Rollback



Controlling Rollback

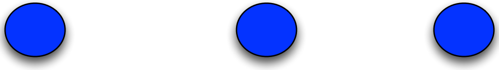


Controlling Rollback

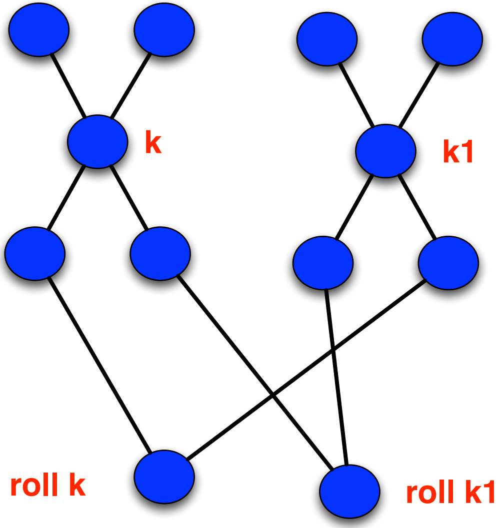


Controlling Rollback

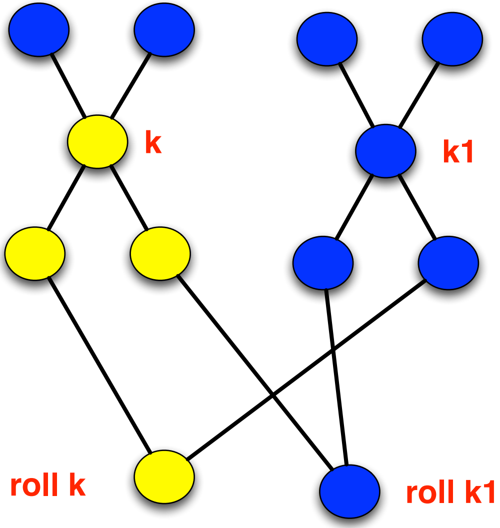
$$k_1 : a\langle 0\rangle \quad (k_2 : a(X) \triangleright_\gamma b\langle roll\ \gamma\rangle) \quad (k_3 : b(X) \triangleright c\langle 0\rangle | X)$$



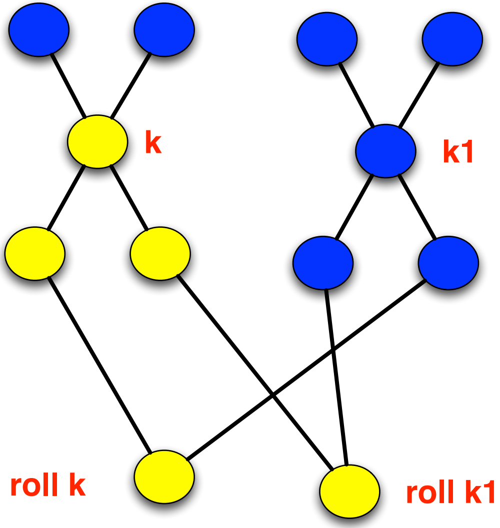
Concurrent rolls



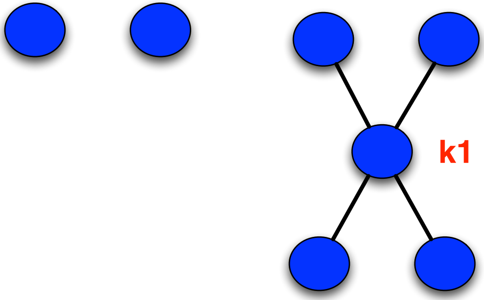
Concurrent rolls



Concurrent rolls



Concurrent rolls



Problems

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

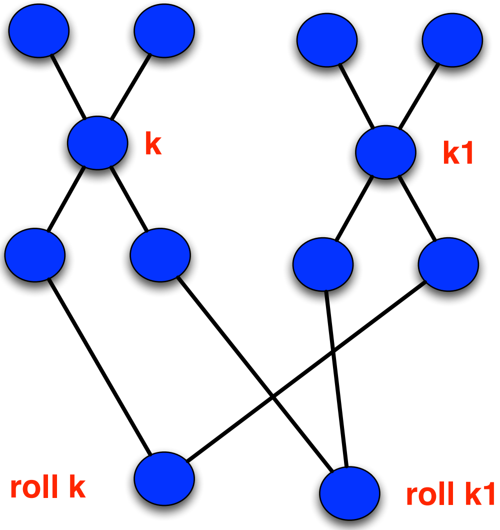
Controlling Rollback

Reduction rules: HL semantics

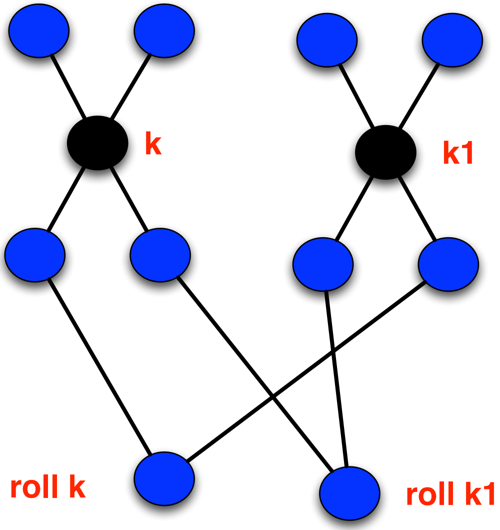
$$(\textsc{Start}) \quad (\kappa_1 : \mathsf{roll}\ k) \mid [m;k] \rightsquigarrow (\kappa_1 : \mathsf{roll}\ k) \mid [m;k]^{\bullet}$$

$$(\textsc{Roll}) \quad \frac{N \blacktriangleright k \qquad \mathtt{complete}(N \mid [m;k])}{N \mid [m;k]^{\bullet} \rightsquigarrow m \mid N \lightning_k}$$

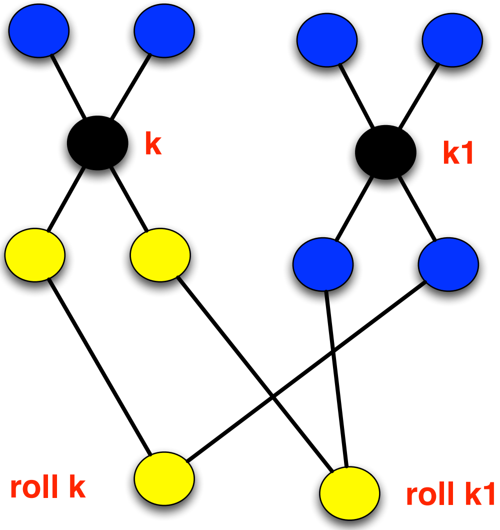
Concurrent rolls



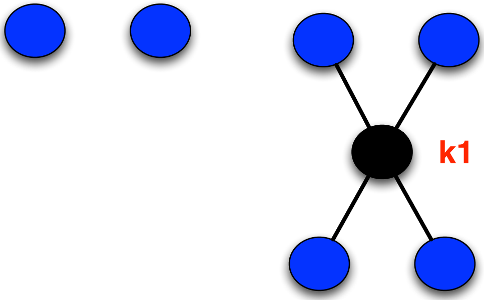
Concurrent rolls



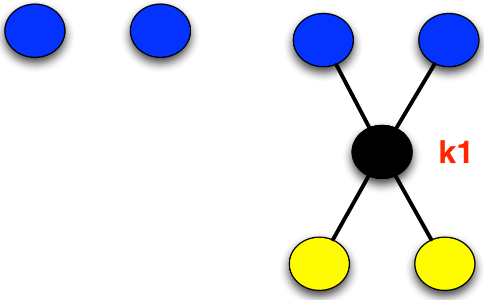
Concurrent rolls



Concurrent rolls



Concurrent rolls



Concurrent rolls

Properties

Soundness of backward reductions

if $M \rightsquigarrow^* M'$ then $M' \twoheadrightarrow^* M$, with M and M' unmarked

Completeness of backward reductions

if $\phi(M) \rightsquigarrow^*_T N$ then $M \rightsquigarrow^*_{HL} M'$ with $M' = \phi(N)$.

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

High Level Semantics

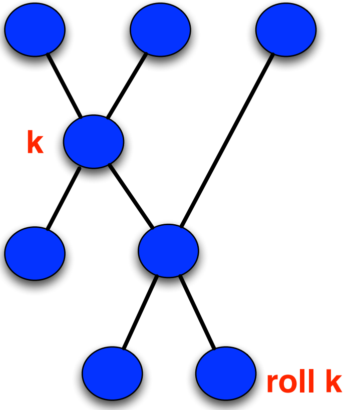
- **sound** and **complete** with respect to $\rho\pi$
- good as specification semantics
- relies on global checks and atomic steps

far away from a real implementation

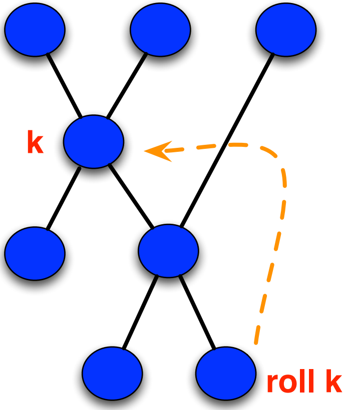
Low Level Implementation

- 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)

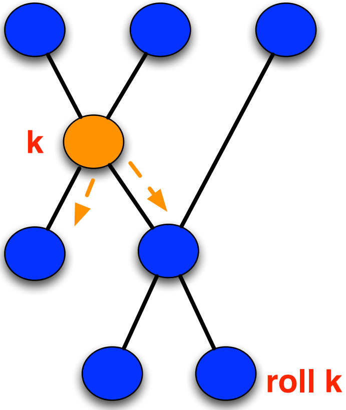
LL example



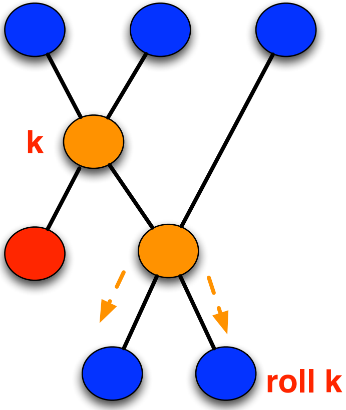
LL example



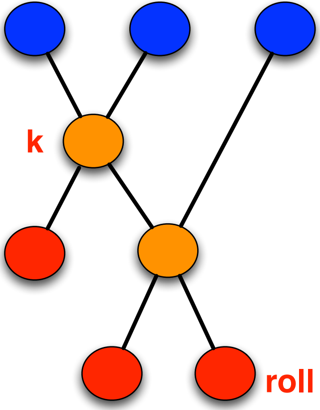
LL example



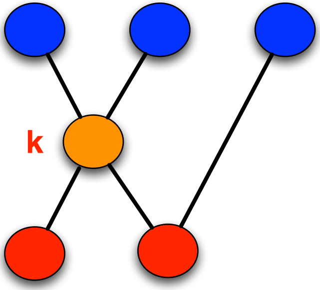
LL example



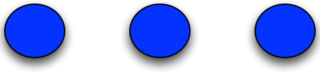
LL example



LL example



LL example



Properties

Correspondence HL vs LL

$$M \;_{HL}{\approx}^{c}_{LL}\; M$$

Full Abstraction

$$j(M) \;_{LL}{\approx}^{c}_{LL}\; j(N) \text{ iff } M \;_{HL}{\approx}^{c}_{HL}\; N$$

- 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 rollπ into $HO\pi$ ($\checkmark$)
- refine and relate the LL semantics to checkpoint and rollback schemes
  - localities, failures ...
- mix **dynamic compensations** and controlled rollback

Thank you!

Questions?

LL Reduction Rules 1/2

$$(\textsc{Start}) \quad (\kappa_1 : \mathsf{roll}\ k) \mid [m;k] \rightsquigarrow_{LL} (\kappa_1 : \mathsf{roll}\ k) \mid [m;k]^{\bullet} \mid \mathsf{rl}\ k$$

$$(\textsc{Span}) \quad \mathsf{rl}\ \kappa_1 \mid [\kappa_1 : P \mid M;k]^{\circ} \rightsquigarrow_{LL} [\lfloor \kappa_1 : P \rfloor \mid M;k]^{\circ} \mid \mathsf{rl}\ k$$

$$(\textsc{Branch}) \quad \frac{\langle h_i, \tilde{h} \rangle \cdot k \text{ occurs in } M}{\mathsf{rl}\ k \mid M \rightsquigarrow_{LL} \prod_{h_i \in \tilde{h}} \mathsf{rl}\ \langle h_i, \tilde{h} \rangle \cdot k \mid M}$$

Low Level

LL Redution Rules 2/2

$$(\textsc{Up}) \quad \mathsf{rl}\ \kappa_1 \mid (\kappa_1 : P) \rightsquigarrow_{LL} \lfloor \kappa_1 : P \rfloor \qquad (\textsc{Stop}) \quad [m;k]^\circ \mid \lfloor k : P \rfloor \rightsquigarrow_{LL} m$$

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

$$(\textsc{TagPFr})\ \lfloor k : \prod_{i=1}^{n} \tau_i \rfloor \equiv_{LL} \nu \tilde{h}.\ \prod_{i=1}^{n} \lfloor \langle h_i, \tilde{h} \rangle \cdot k : \tau_i) \rfloor \ \ \tilde{h} = \{h_1, \ldots, h_n\} \ \ n \geq 2$$