Croll-π, a reversible calculus with compensations

Ivan Lanese
Computer Science Department
Focus research group
University of Bologna/INRIA
Bologna, Italy

Joint work with Michael Lienhardt (PPS), Claudio Antares Mezzina (Trento), Jean-Bernard Stefani (INRIA) and Alan Schmitt (INRIA)

Roadmap

- The approach
- Croll-π
- Applications
- Conclusions

Roadmap

- The approach
- Croll-π
- Applications
- Conclusions

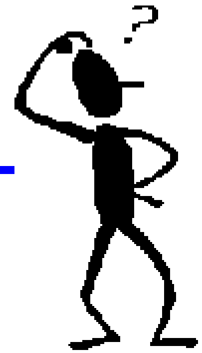
Uncontrolled perfect reversibility

- RCCS, CCSk, ρπ, reversible structures all feature uncontrolled perfect reversibility
  - Uncontrolled since the process can go nondeterminstically backward or forward
  - Perfect since a backward step perfectly undoes a forward step
- Some approaches feature some form of control of reversibility
  - Irreversible actions in RCCS
  - Roll-π roll operator
  - Controller process in CCSk
- Reversibility is still perfect in all these approaches

Drawbacks of perfect reversibility

- With perfect reversibility one goes back to a past state
- From this state the same computation can restart
  - Probably leading to the same rollback
  - Again and again
- The programmer has no tool to avoid this
- We want to provide such a tool
  - Using a compensation mechanism
- We build on top of roll-π

Do you remember roll-π?



- HOπ extended with an explicit roll operator
- The roll has a parameter γ that refers to a past trigger
- The roll undoes all the actions depending on the communication done by the trigger
- Idea: in case of error I undo the computation that caused the error

Compensations

- From database theory and business processes
  - Piece of code to recover from an error
- For us, a piece of code allowing to move
  - From the past state reached by the rollback
  - To a slightly different state
  - Keeping into account the past try
  - Trying to avoid to repeat the same error
- We call croll-π the language integrating compensations into roll-π

Messages with compensations

- Instead of roll-π messages a<P> we use messages with compensations
  - a<P>%0 : try a<P> then stop trying
  - a<P>%b<Q>%0 : try a<P> then b<Q> then stop trying
- If the message with compensation is the target of a rollback, it is replaced by its compensation
- Minimal change to roll-π syntax and semantics
- The expressive power increases considerably

Roadmap

- The approach
- Croll-π
- Applications
- Conclusions

Croll-π syntax

$$
\begin{aligned}
P, Q &::= \mathbf{0} \mid X \mid \nu a.\, P \mid (P \mid Q) \mid a(X) \triangleright_{\gamma} P \mid a\langle P\rangle \div C \mid \mathsf{roll}\ k \mid \mathsf{roll}\ \gamma \\
M, N &::= \mathbf{0} \mid \nu u.\, M \mid (M \mid N) \mid k : P \mid [\mu; k] \mid k \prec (k_1, k_2) \\
C &::= a\langle P\rangle \div \mathbf{0} \mid \mathbf{0} \\
\mu &::= (k_1 : a\langle P\rangle \div C) \mid (k_2 : a(X) \triangleright_{\gamma} Q)
\end{aligned}
$$

- HOπ processes + roll + γ + compensations
- Configurations featuring memories and causal dependencies

Croll-π semantics

$$\frac{\mu = (k_1 : a\langle P\rangle \div C) \mid (k_2 : a(X) \triangleright_\gamma Q_2)}{(k_1 : a\langle P\rangle \div C) \mid (k_2 : a(X) \triangleright_\gamma Q_2) \longrightarrow \nu k.\ (k : Q_2\{^{P,k}/_{X,\gamma}\}) \mid [\mu; k]}$$

$$\frac{k <: N \quad \texttt{complete}(N \mid [\mu; k] \mid (k_r : \mathsf{roll}\ k)) \quad \mu' = \texttt{xtr}(\mu)}{N \mid [\mu; k] \mid (k_r : \mathsf{roll}\ k) \longrightarrow \mu' \mid N \lightning_k}$$

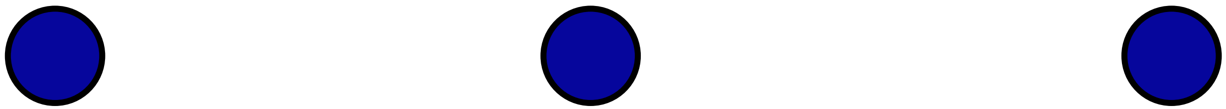
$$\frac{}{k : P \mid Q \longrightarrow \nu k_1, k_2.\ k \prec (k_1, k_2) \mid k_1 : P \mid k_2 : Q}$$

$$\frac{}{k : \nu a.\ P \longrightarrow \nu a.\ k : P}$$

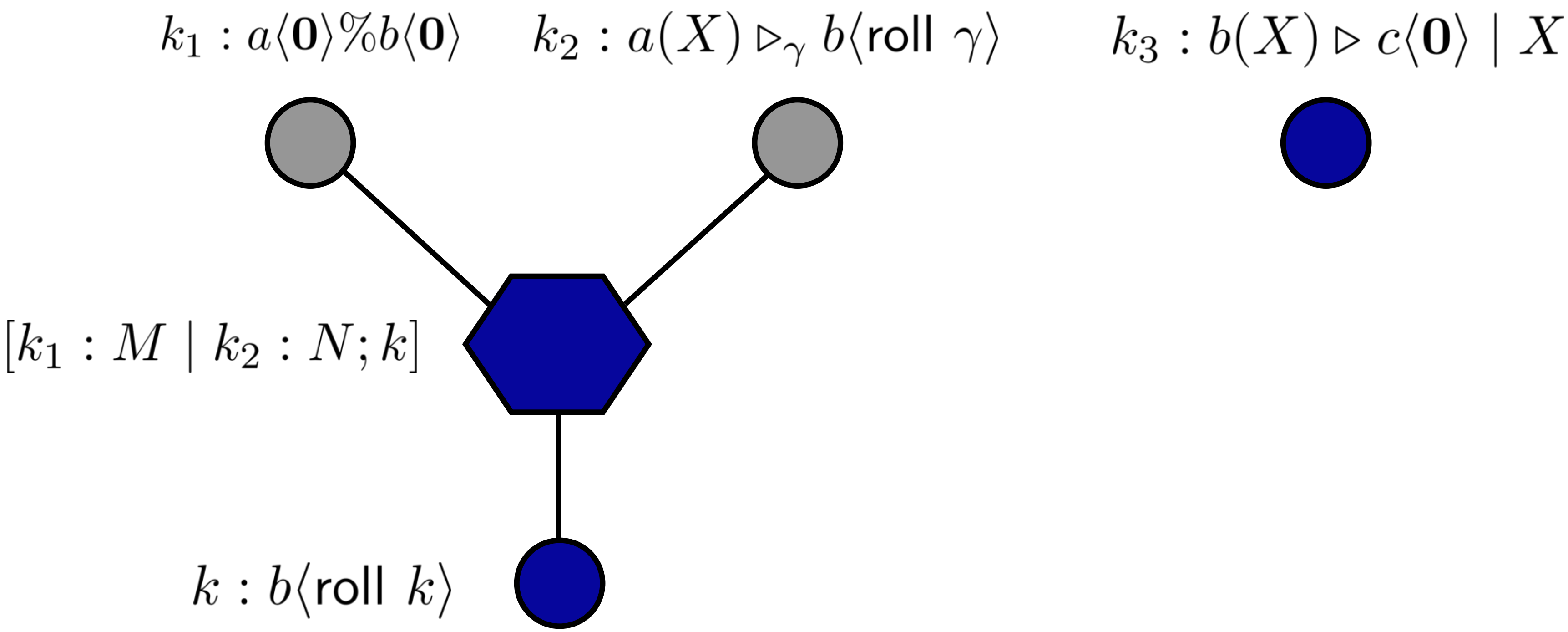
- Instead of restoring the initial configuration we replace the message with its compensation
- xtr(a<P>%C)=C

Croll-π example

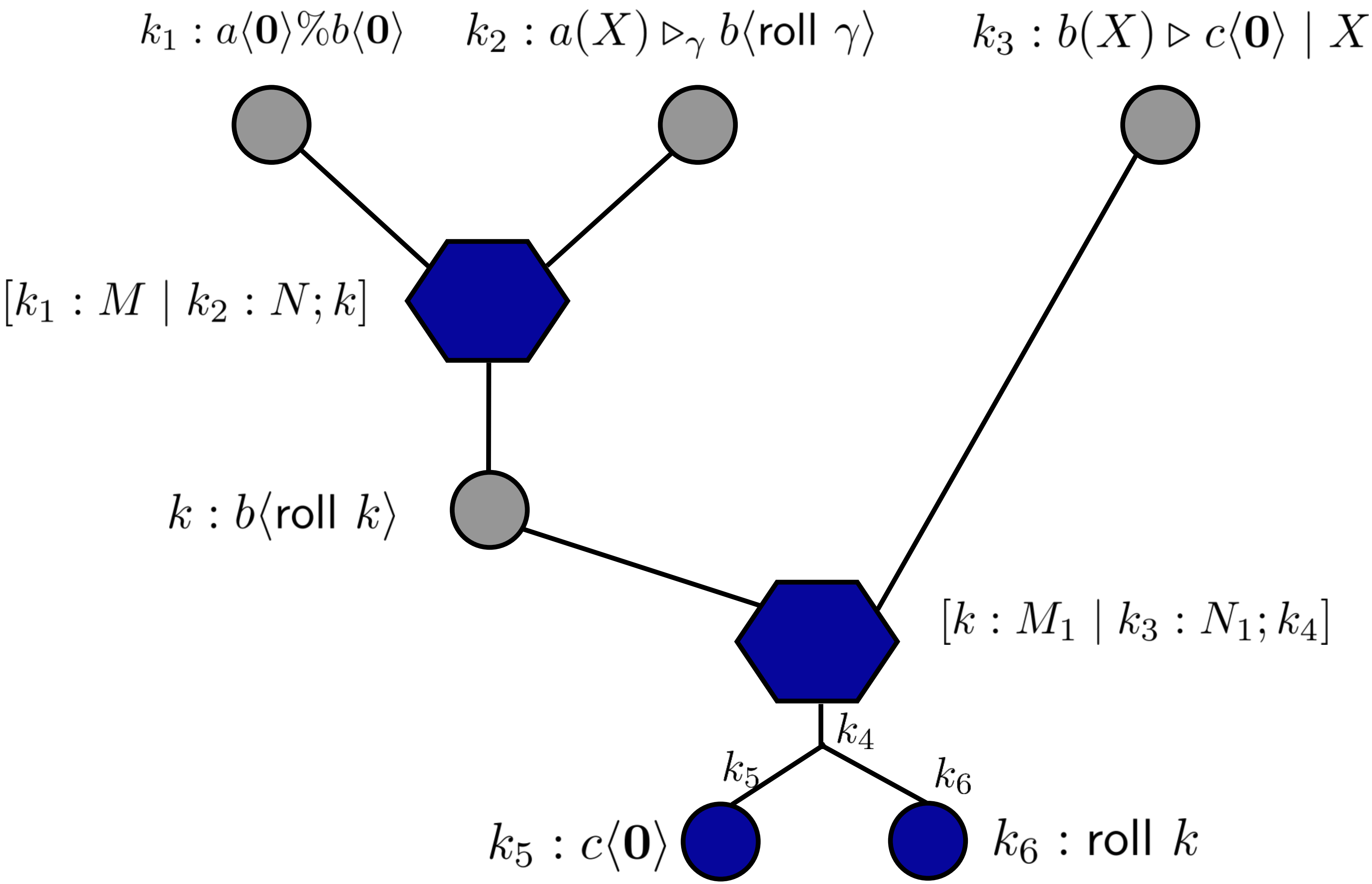
$k_1 : a\langle \mathbf{0}\rangle \% b\langle \mathbf{0}\rangle$ $\quad$ $k_2 : a(X) \triangleright_\gamma b\langle \mathsf{roll}\ \gamma\rangle$ $\quad$ $k_3 : b(X) \triangleright c\langle \mathbf{0}\rangle \mid X$



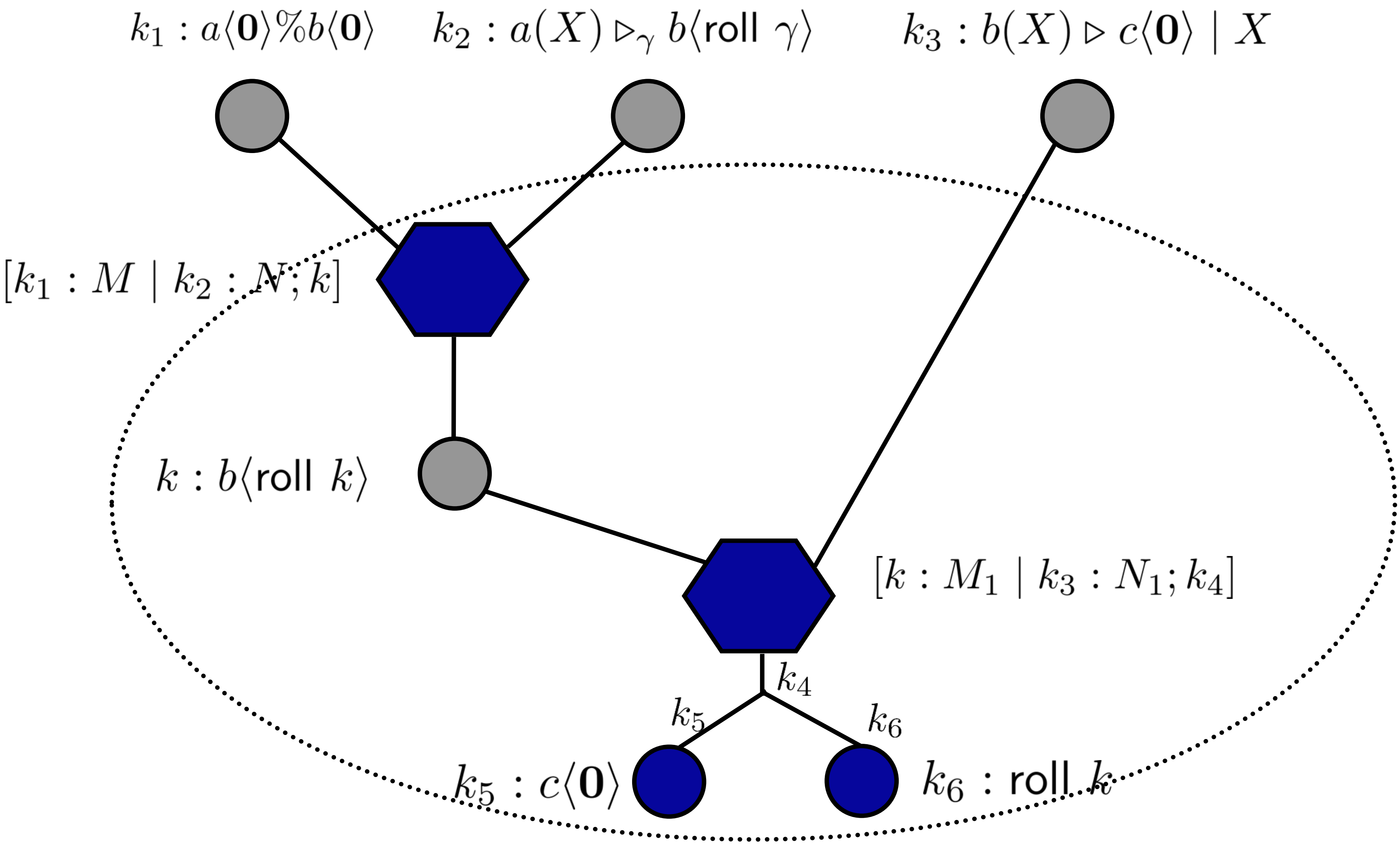
Croll-π example



Croll-π example

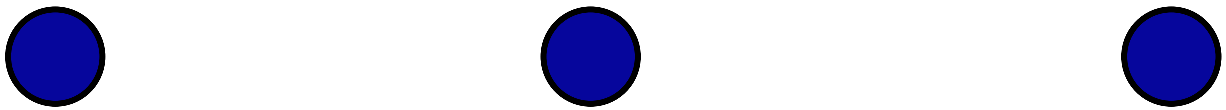


Croll-π example



Croll-π example

$k_1 : b\langle \mathbf{0} \rangle$ $\quad$ $k_2 : a(X) \triangleright_\gamma b\langle \mathsf{roll}\ \gamma \rangle$ $\quad$ $k_3 : b(X) \triangleright c\langle \mathbf{0} \rangle \mid X$



Roadmap

- The approach
- Croll-π
- Applications
- Conclusions

And now?

- We have to test the expressive power of croll-π
- Can we programme interesting applications exploiting rollback and compensations?
- Compensations allow to specify what to do after a rollback
- Croll-π is strictly more expressive than roll-π

Messages with compensations are robust

- We can encode different idioms:
  - General compensations: not only messages
  - Finite retry: try n times
  - Endless retry: try forever (same as roll-π)
  - Triggers with compensations: we can attach compensations to triggers instead of to messages

Triggers with compensations

$$a(X) \rhd_\gamma Q \div C$$

- Compensation replaces trigger instead of message

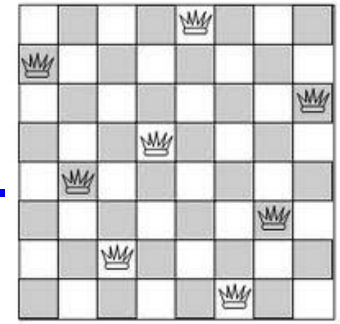
$$(\!|a(X) \rhd_\gamma Q \div C|\!)_{tc} = \nu c, d.\, \overline{c} \div \overline{d} \div \mathbf{0} \mid (c \rhd_\gamma a(X) \rhd (\!|Q|\!)_{tc}) \mid (d \rhd (\!|C|\!)_{tc})$$

- Messages with compensations and triggers with compensations can coexist
- Makes the framework more symmetric

What can we model?

- Interesting problems
  - State space exploration with backtracking: 8 queens problem
  - Error handling scenario: Automotive case study from Sensoria project
- Can we recover/improve existing techniques?
  - Interacting transactions from Hennessy et al. [CONCUR 2010]
  - Software transactional memories
    - » Different approaches
    - » We started from the process calculus account by Acciai et al. [ESOP'07]

Eight queens problem



- Queens are interacting processes
- Queen i tries the 8 positions on column i then fails by notifying the previous queen
- If it succeeds it notifies the next queen, and waits for failures from her

$$
\begin{aligned}
Q_i \;\triangleq\;& c_1(\mathbf{x_1}) \triangleright \ldots c_{i-1}(\mathbf{x_{i-1}}) \triangleright\; p_i\langle i,1\rangle \div \ldots \div p_i\langle i,8\rangle \div f_i\langle 0\rangle \div 0 \\
& \mid p_i(\mathbf{x}) \triangleright_{\gamma_i} \texttt{if}\ err(\mathbf{x_1},\mathbf{x})\ \texttt{then}\ \mathsf{roll}\ \gamma_i\ \texttt{else}\ \ldots\ \texttt{if}\ err(\mathbf{x_{i-1}},\mathbf{x})\ \texttt{then}\ \mathsf{roll}\ \gamma_i \\
& \texttt{else}\ !c_i\langle \mathbf{x}\rangle \div 0 \mid f_{i+1}(y) \triangleright \mathsf{roll}\ \gamma_i
\end{aligned}
$$

$$
err((x_1,x_2),(y_1,y_2)) \;\triangleq\; (x_1 = y_1 \vee x_2 = y_2 \vee |x_1 - y_1| = |x_2 - y_2|)
$$

- We also have a more concurrent solution

Interacting transactions

- Allow a transaction to interact with the environment
- In case of rollback the effects on the environment are undone

$$\frac{}{\overline{a}|a.P \longrightarrow P} \qquad \frac{k \notin \mathtt{fn}(R)}{[\![P \triangleright_k Q]\!] | R \longrightarrow [\![P | R \triangleright_k Q | R]\!]}$$

$$\frac{}{[\![P | \mathsf{co}\ k \triangleright_k Q]\!] \longrightarrow P} \qquad \frac{}{[\![P \triangleright_k Q]\!] \longrightarrow Q}$$

- Parts of the environment moved inside the transaction to interact with it
  - Also saved in the compensation

Modeling interacting transactions

- A transaction is a computation originated by a trigger labeled by γ
- Roll γ used as an abort
- Commit by disabling roll γ

$$(\!|[\![P \triangleright_l Q]\!]|\!)_t = \nu a, c.\, \overline{a} \div \overline{c} \div \mathbf{0} \mid a \triangleright_\gamma (\nu l.\, (\!|P|\!)_t \mid l\langle \textsf{roll } \gamma \rangle \mid l(X) \triangleright X) \mid c \triangleright (\!|Q|\!)_t$$

- More precise than Hennessy
- Only parts of the environment that interacted with the transaction are actually rollbacked

Roadmap

- The approach
- Croll-π
- Applications
- Conclusions

Summary

- A framework supporting controlled rollback and compensations
- Some interesting applications
  - 8 queens problem
  - Interacting transactions
  - Software transactional memories

Future work

- Is this enough for programming recoverable systems?
  - Some good hints, but more work is needed
- A known limitation
  - We have no information on what caused the roll
  - The compensation does not depend on the reason of the failure
  - A mechanism to keep this information would be helpful
- Studying the behavioral theory
  - To reason about the encodings
  - To have an axiomatic theory
  - History information makes things more complex

Finally

Thanks!

Questions?