



# An Operational Semantics for Model Checking Long Running Transactions

Hengbiao Yu, Zhenbang Chen, Ji Wang  
zbchen@nudt.edu.cn

School of Computer, National University of Defense Technology,  
Changsha, China  
National Laboratory for Parallel and Distributed Processing,  
Changsha, China

# Long Running Transactions

- Database
  - Long-lived transactions
  - ACID transactions
- SAGAS
  - 1987, SIGMOD
- Compensation



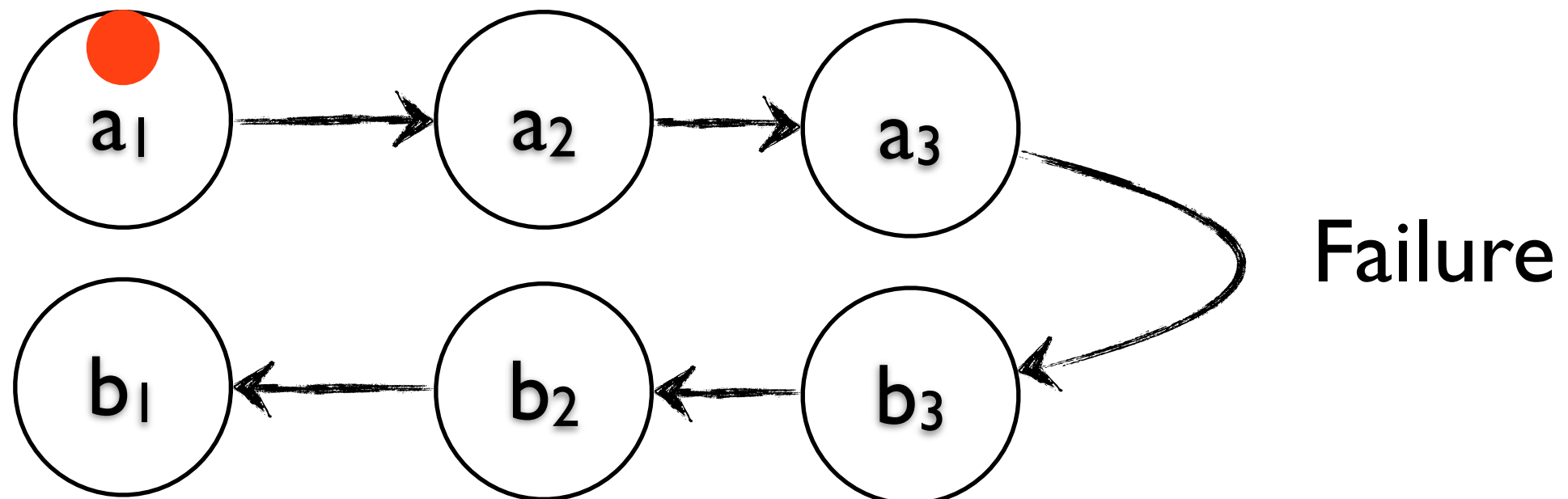


# Compensation



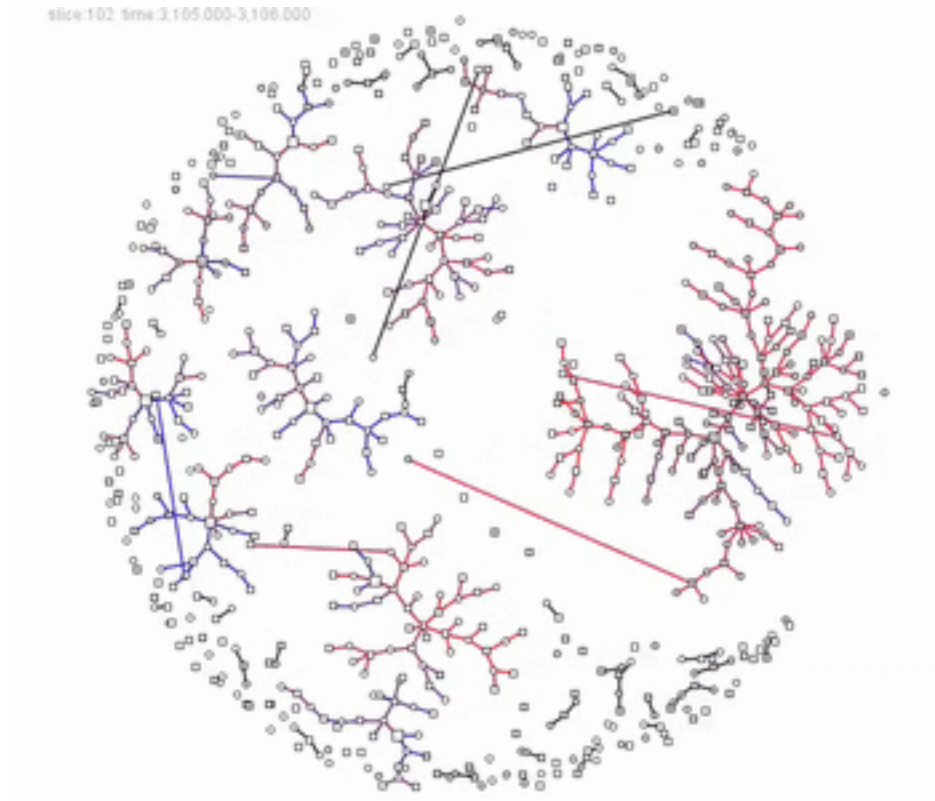
# Compensation-based LRTs

- An activity has its compensation activity
- Use compensations in case of failures
- A relaxed atomicity and consistency



# Service-Oriented Computing

- Services are world wide distributed
- Coordinate to accomplish a task
  - Highly dynamic, not stable



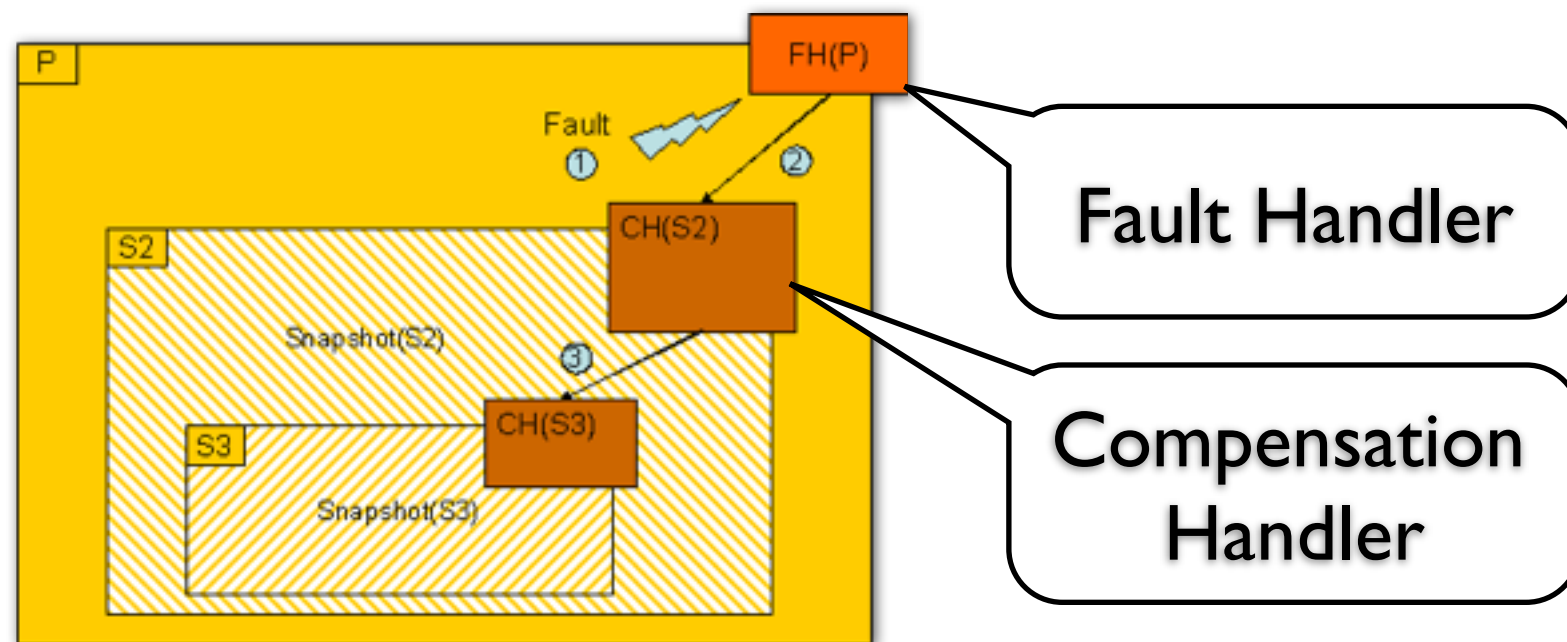
How to ensure consistency  
in case of a failure?

Long running transactions



# Compensation-based Programming in SOC

- Industrial service orchestration languages
  - Compensation based fault handling
  - Very flexible recovery mechanisms



WS-BPEL 2.0, OASIS Standard, 11 April 2007

# Compensation-based Programming in SOC

- Industrial service orchestration languages
  - Compensation based fault handling
  - Very flexible recovery mechanisms
- Formal Languages
  - cCSP, StAC, SAGAs Calculi, *etc.*
  - Theoretical foundations

# Compensating CSP

- A CSP variant for modeling LRTs
  - Proposed by Butler *et al.*, 2005
- Two types of processes
  - Standard & Compensable
- Composition operators
  - Choice, sequence, interleaving, *etc.*



# Extended cCSP

$$P ::= a \mid P;P \mid \boxed{P \sqcap P \mid P \square P \mid P \parallel_{\bar{X}} P \mid P \setminus X \mid P[R]} \mid P \triangleright P \mid [PP] \mid$$

$$\text{skip} \mid \text{throw} \mid \text{yield} \mid \boxed{\mu p.F(p)}$$

$$PP ::= P \div P \mid \boxed{PP;PP \mid PP \sqcap PP \mid PP \square PP \mid PP \parallel_{\bar{X}} PP \mid PP \boxtimes PP \mid}$$

$$\boxed{PP \setminus X \mid PP[R]} \mid \text{skipp} \mid \text{throww} \mid \text{yieldd} \mid \boxed{\mu pp.FF(pp)}$$

- Enabling the modeling of non-determinism, deadlock and livelock
- Stable-failures semantics

Zhenbang Chen and Zhiming Liu. An Extended cCSP with Stable Failures Semantics. 7th International Colloquium on Theoretical Aspects of Computing (ICTAC'10), LNCS 6255, 2010.

# Extended cCSP

$$P ::= a \mid P;P \mid P \sqcap P \mid P \sqcup P \mid P \parallel P \mid P \setminus X \mid P[R] \mid P \triangleright P \mid [PP] \mid \text{skip} \mid \text{throw} \mid \text{yield} \mid \mu p.F(p)$$

$$PP ::= P \div P \mid PP;PP \mid PP \sqcap PP \mid PP \sqcup PP \mid PP \parallel PP \mid PP \boxtimes PP \mid PP \setminus X \mid PP[R] \mid \text{skipp} \mid \text{throww} \mid \text{yieldd} \mid \mu pp.FF(pp)$$

- Failure-divergence semantics, refinement

Zhenbang Chen, Zhiming Liu and Ji Wang. Failure-Divergence Refinement of Compensating Communicating Processes. **17th International Symposium on Formal Methods (FM'11)**, LNCS 6664, 2011.

Zhenbang Chen, Zhiming Liu and Ji Wang. Failure-Divergence Semantics and Refinement of Long Running Transactions. **Theoretical Computer Science (TCS)**, Vol 455, pp:31-65, 2012

# Current Issue

- No operational semantics for extended cCSP
  - There is one for cCSP
- No tool support for modeling and verification
  - Animating
  - Model checking
  - ...

# What we have done in this paper

- An operational semantics for the extended cCSP
- Model checking problem w.r.t. regular properties
- A prototype tool built on PAT
  - Modeling, animating and model checking



# Operational Semantics

$$P ::= a \mid P;P \mid P \sqcap P \mid P \square P \mid P \parallel_X P \mid P \setminus X \mid P[R] \mid P \triangleright P \mid [PP] \mid \text{skip} \mid \text{throw} \mid \text{yield} \mid \mu p.F(p)$$

$$PP ::= P \div P \mid PP;PP \mid PP \sqcap PP \mid PP \square PP \mid PP \parallel_X PP \mid PP \boxtimes PP \mid PP \setminus X \mid PP[R] \mid \text{skipp} \mid \text{throww} \mid \text{yieldd} \mid \mu pp.FF(pp)$$

## Compensation Pair

$$\boxed{\frac{P \xrightarrow{e} P'}{P \div Q \xrightarrow{e} P' \div Q} \quad (e \in \Sigma^\tau) \quad \frac{P \xrightarrow{\check{}} 0}{P \div Q \xrightarrow{\check{}} Q} \quad \frac{P \xrightarrow{\omega} 0}{P \div Q \xrightarrow{\omega} \text{skip}} \quad (\omega \in \{!, ?\})}$$

## Example

$$a_1 \div b_1 \xrightarrow{a_1} \text{skip} \div b_1 \xrightarrow{\check{}} b_1$$

# Operational Semantics

$$P ::= a \mid P;P \mid P \sqcap P \mid P \sqcup P \mid P \parallel_X P \mid P \setminus X \mid P[R] \mid P \triangleright P \mid [PP] \mid \text{skip} \mid \text{throw} \mid \text{yield} \mid \mu p.\bar{F}(p)$$

$$PP ::= P \div P \mid PP;PP \mid PP \sqcap PP \mid PP \sqcup PP \mid PP \parallel_X PP \mid PP \boxtimes PP \mid PP \setminus X \mid PP[R] \mid \text{skipp} \mid \text{throww} \mid \text{yieldd} \mid \mu pp.FF(pp)$$

## Sequential Composition

$$\frac{PP \xrightarrow{\checkmark} P \wedge QQ \xrightarrow{e} QQ'}{PP;QQ \xrightarrow{e} \langle QQ', P \rangle} \quad (e \in \Sigma^\tau)$$

$$\frac{PP \xrightarrow{\checkmark} P \wedge QQ \xrightarrow{\omega} Q}{PP;QQ \xrightarrow{\omega} Q;P} \quad (\omega \in \Omega)$$

## Example

$$\begin{array}{c} a_1 \div b_1 ; a_2 \div b_2 \\ \downarrow a_1 \\ \text{skip} \div b_1 ; a_2 \div b_2 \\ \downarrow a_2 \\ \langle \text{skip} \div b_2 , b_1 \rangle \end{array}$$

# Operational Semantics

$$P ::= a \mid P;P \mid P \sqcap P \mid P \sqcup P \mid P \parallel_X P \mid P \setminus X \mid P[R] \mid P \triangleright P \mid [PP] \mid \text{skip} \mid \text{throw} \mid \text{yield} \mid \mu p.\bar{F}(p)$$

$$PP ::= P \div P \mid PP;PP \mid PP \sqcap PP \mid PP \sqcup PP \mid PP \parallel_X PP \mid PP \boxtimes PP \mid PP \setminus X \mid PP[R] \mid \text{skipp} \mid \text{throww} \mid \text{yieldd} \mid \mu pp.FF(pp)$$

## Nested Configuration

$$\boxed{\frac{QQ \xrightarrow{e} QQ'}{\langle QQ, P \rangle \xrightarrow{e} \langle QQ', P \rangle} \quad (e \in \Sigma^\tau) \quad \frac{QQ \xrightarrow{\omega} Q}{\langle QQ, P \rangle \xrightarrow{\omega} Q; P} \quad (\omega \in \Omega)}$$

## Example

$$a_1 \div b_1 ; a_2 \div b_2 \xrightarrow{a_1} \text{skip} \div b_1 ; a_2 \div b_2 \xrightarrow{a_2} \langle \text{skip} \div b_2, b_1 \rangle \xrightarrow{\checkmark} b_2 ; b_1$$

# Operational Semantics

$$P ::= a \mid P;P \mid P \sqcap P \mid P \sqcup P \mid P \parallel_X P \mid P \setminus X \mid P[R] \mid P \triangleright P \mid [PP] \mid \text{skip} \mid \text{throw} \mid \text{yield} \mid \mu p.F(p)$$

$$PP ::= P \div P \mid PP;PP \mid PP \sqcap PP \mid PP \sqcup PP \mid PP \parallel_X PP \mid PP \boxtimes PP \mid PP \setminus X \mid PP[R] \mid \text{skipp} \mid \text{throww} \mid \text{yieldd} \mid \mu pp.FF(pp)$$

Configuration Discussion

$$\frac{PP \xrightarrow{\checkmark} P \wedge QQ \xrightarrow{e} QQ'}{PP;QQ \xrightarrow{e} \langle QQ', P \rangle} \quad (e \in \Sigma^\tau)$$

$$a_1 \div b_1 ; (\text{skip} \div b_2 ; a_3 \div b_3) \xrightarrow{a_1} \text{skip} \div b_1 ; (\text{skip} \div b_2 ; a_3 \div b_3)$$

$$\text{skip} \div b_2 ; a_3 \div b_3 \xrightarrow{a_3} \langle \text{skip} \div b_3 , b_2 \rangle \quad \text{No rule exists in the before semantics}$$

$$\text{skip} \div b_1 ; (\text{skip} \div b_2 ; a_3 \div b_3) \xrightarrow{a_3} \langle \langle \text{skip} \div b_3 , b_2 \rangle , b_1 \rangle \xrightarrow{\checkmark} b_3 ; b_2 ; b_1$$



# Operational Semantics

$$P ::= a \mid P;P \mid P \sqcap P \mid P \sqcup P \mid P \parallel_X P \mid P \setminus X \mid P[R] \mid P \triangleright P \mid [PP] \mid \text{skip} \mid \text{throw} \mid \text{yield} \mid \mu p.F(p)$$

$$PP ::= P \div P \mid PP;PP \mid PP \sqcap PP \mid PP \sqcup PP \mid PP \parallel_X PP \mid PP \boxtimes PP \mid PP \setminus X \mid PP[R] \mid \text{skipp} \mid \text{throww} \mid \text{yieldd} \mid \mu pp.FF(pp)$$

## Parallel Composition

$$\boxed{\begin{array}{l} \frac{PP \xrightarrow{e} PP' \wedge QQ \xrightarrow{e} QQ'}{PP \parallel_X QQ \xrightarrow{e} PP' \parallel_X QQ'} \quad (e \in X) \quad \frac{PP \xrightarrow{\omega_1} P \wedge QQ \xrightarrow{\omega_2} Q}{PP \parallel_X QQ \xrightarrow{\omega_1 \parallel \omega_2} P \parallel_X Q} \quad (\omega_1, \omega_2 \in \Omega) \end{array}}$$

## Example

$$(a_1 \div b_1 \parallel_{\{a_1, a_2\}} a_2 \div b_2) \quad \text{deadlock}$$

# Operational Semantics

$$P ::= a \mid P;P \mid P \sqcap P \mid P \sqcup P \mid P \parallel_X P \mid P \setminus X \mid P[R] \mid P \triangleright P \mid [PP] \mid \text{skip} \mid \text{throw} \mid \text{yield} \mid \mu p.F(p)$$

$$PP ::= P \div P \mid PP;PP \mid PP \sqcap PP \mid PP \sqcup PP \mid PP \parallel_X PP \mid PP \boxtimes PP \mid PP \setminus X \mid PP[R] \mid \text{skipp} \mid \text{throww} \mid \text{yieldd} \mid \mu pp.FF(pp)$$

## Parallel Composition

$$\boxed{\begin{array}{l} \frac{PP \xrightarrow{e} PP' \wedge QQ \xrightarrow{e} QQ'}{PP \parallel_X QQ \xrightarrow{e} PP' \parallel_X QQ'} \quad (e \in X) \quad \frac{PP \xrightarrow{\omega_1} P \wedge QQ \xrightarrow{\omega_2} Q}{PP \parallel_X QQ \xrightarrow{\omega_1 \parallel \omega_2} P \parallel_X Q} \quad (\omega_1, \omega_2 \in \Omega) \end{array}}$$

## Example

$$(a_1 \div b_1 \parallel a_1 \div b_2) \xrightarrow{a_1} (\text{skip} \div b_1 \parallel \text{skip} \div b_2) \xrightarrow{\checkmark} b_1 \parallel b_2$$

$\{a_1, a_2\} \qquad \qquad \qquad \{a_1, a_2\} \qquad \qquad \qquad \{a_1, a_2\}$

# Correspondence with FD Semantics

- A method for deriving FD semantics from operational semantics
- Inspired by the method for CSP

## Theorem 1

For a standard process  $P$ , the derived model according to the operational semantics is equal to the FD model of  $P$ .

Basic idea of proof: induce the structures of processes

# Correspondence with FD Semantics

- A method for deriving FD semantics from operational semantics
- Inspired by the method for CSP

## Theorem 2

For a compensable process  $PP$ , the derived model according to the operational semantics is equal to the FD model of  $PP$ .



# Model Checking Problem

- General model checking problem with respect to regular properties

## Theorem 3

Given a standard process  $P$  of the extended cCSP and an FSM  $R$ , the language inclusion problem  $L(T(P)) \subseteq L(R)$  is undecidable.

Basic idea of proof: reduction from the halting problem of Minsky 2-counter machine

# Tool Implementation

- Built on Process Analysis Toolkit (PAT)
  - Extended cCSP parser
  - Operational semantics Implementation
- Features
  - Modeling environment, animating
  - LTL model checking
  - Refinement checking (only standard process)

# Tool Screenshots

The screenshot displays the 'Verifier (Debug Model) - TravelAgency.ccsp' window. The 'Assertions' list on the left contains eight items, with the seventh assertion, 'GBP() |= [] (cancelAir R ! letter)', selected and highlighted in red. Below the list, the 'The selected assertion' section shows the same formula. The 'Options' section includes 'Admissible Behavior' set to 'All' and 'Verification Engine' set to 'First Witness Trace using Depth First Search'. The 'Output' pane at the bottom shows a 'Verification Result' indicating the assertion is 'NOT valid' and provides a counterexample trace: <init -> reqTravel -> checkCredit -> reqCar -> bookAir -> reqHotel -> [int\_choice] -> invalid -> [int\_choice] -> hasCar -> [int\_choice] -> noAir -> [int\_choice] -> noRoom -> DataOperation -> CCSPTAU -> cancelCar -> letter>. Verification statistics show 17 visited states, 90 total transitions, and a time used of 0.1070332s. To the right of the verifier window, the 'InteractionPanel' and 'EventTrace' are visible. The 'EventTrace' table lists a sequence of events from 0 to 14, including 'init', 'reqTravel', 'checkCredit', 'bookAir', '[int\_choice]', 'valid', 'reqCar', '[int\_choice]', 'reqHotel', 'payment', '[int\_choice]', '[int\_choice]', 'noRoom', and 'hasCar'.

Source...	Enabled Event

Source...	Event	Target ...
0	init	1
1	reqTravel	2
2	checkCredit	3
3	bookAir	4
4	[int_choice]	5
5	[int_choice]	6
6	valid	7
7	reqCar	8
8	[int_choice]	9
9	reqHotel	10
10	payment	11
11	[int_choice]	12
12	[int_choice]	13
13	noRoom	14
14	hasCar	15

<http://rcos.iist.unu.edu/~zbchen/LRT.html>

# Conclusion

- An operational semantics for extended cCSP
  - Correspondence with FD semantics
- Model checking problem discussion
- A tool for modeling and verifying LRTs
  - Animating, LTL model checking
  - Refinement checking



# Next Step

- Refinement checking of compensable processes
- Model checking Algorithms
- Applications
  - Business process verification: BPMN
  - Service orchestration verification: WS-BPEL
  - Recovery-oriented Computing

# Web Services and Formal Methods (WS-FM 2013)

August 29-30, 2013, Beijing, China

10th International Workshop on Web Services and Formal Methods  
Formal Aspects of Service-Oriented and Cloud Computing

# End

# Thank you!