On the Semantics of Distributed Compensations with Interruption

Roberto Bruni, Anne Kersten and Ivan Lanese

University of Pisa, IMT Institute for Advanced Studies, University of Bologna/INRIA

July 2, 2010
Service-oriented computing

- modular software solutions using services
- services
  - different functionalities, publicly available
  - combined into larger applications
  - loosely coupled
- open environment leads to unreliability
  - different components
  - communication middleware
- importance of reliable systems for the user
- system should be able to handle unexpected events
Long-running transactions

- ACID transactions (as in databases) not suitable for error handling in service oriented computing
- Generalization: long-running transactions with "non perfect" rollback
- Transaction split into smaller activities
- Compensations to ensure consistency ("undoing" action)
Example for a long-running transaction
Formal approach

- different formal models
  - modeling various compensation policies
  - to prove properties of the system

- communication based approach
  - interaction between communication and error-handling
  - extending name passing calculi
  - c-join, web\(\pi\), dc\(\pi\), . . .

- workflow based approach
  - control flow between components
  - cCSP, StAC, Sagas
Sagas / cCSP

(STEP) \[ X ::= A \div B \mid \text{throw} \]

(PROCESS) \[ P ::= X \mid P; P \mid P\parallel P \]

(SAGA) \[ S ::= \{[P]\} \mid S; S \mid S\parallel S \]
Overview of the semantics

- $A \div B$ (compensation pair): $B$ installed if $A$ is successful, executed in case of a later abort
- compensations of sequential activities are executed in reverse order
- four different policies for the parallel composition

Comparing Two Approaches to Compensable Flow Composition.
In *CONCUR’05*, 2005.
Semantics in the literature

Centralized

1. No interruption

2. Naïve Sagas

3. Interruption

4. Revised Sagas

Distributed

Original cCSP

strategies 1-3 too restrictive

strategy 4 unrealistic
Example

- example term \( \{ (A \div A'; B \div B') | (C \div C'; \text{throw}) \} \)

1. no interruption, forward flow followed by compensations,
   \( S_1 \equiv (AB\|\|C); (B'A'||C') \)

2. no interruption, interleaving of forward flow and compensations,
   \( S_2 \equiv ABB'A'||CC' \)

3. interruption, forward flow followed by compensations,
   \( S_3 \equiv CC' \cup (A\|\|C); (A'||C') \cup (AB\|\|C); (B'A'||C') \)

4. interruption, interleaving of forward flow and compensations,
   \( S_4 \equiv CC' \cup AA'||CC' \cup ABB'A'||CC' \)
Desirable trace

interruption, forward flow and compensations are only interleaved after the error occurred, $S \equiv S_3 \cup (CC'AA') \cup (AB||CC')$; $B'A'$
Our contribution

- define a new semantics (5), as liberal as possible but still realistic
- via encoding into Petri nets
- continuation passing style
- tokens to enable activities and send interrupt in case of an abort
General process

![Diagram showing a general process with inputs I1 and I2, transitions F1, F2, R1, and R2, and a central block P.](image)
Compensation Pair $A \div B$
throw
Sequential Composition
Parallel Composition
Parallel Composition
Parallel Composition
Parallel Composition
Transaction scope $\{P\}$
Expected Behaviour

- **successful computation:**
  - providing a token in $F_1$ we will issue a token in $F_2$
  - providing a token in $R_1$ we will issue a token in $R_2$

- **aborted computation:**
  - providing a token in $F_1$ we will issue a token in $R_2$ and $I_2$

- **interrupted computation:**
  - providing a token in $F_1$ and $I_1$ we will issue a token in $R_2$
Conclusions

- We defined a new realistic semantics for distributed Sagas with interruption.
- Ongoing and future work:
  - Denotational semantics and correspondence proof
  - Definition in Maude
  - Add nested transactions