



Process Evolution with Atomicity Consistency

SEAMS 2007

Chunyang Ye¹, **S.C. Cheung**¹, W.K. Chan²

¹Hong Kong University of Science and Technology

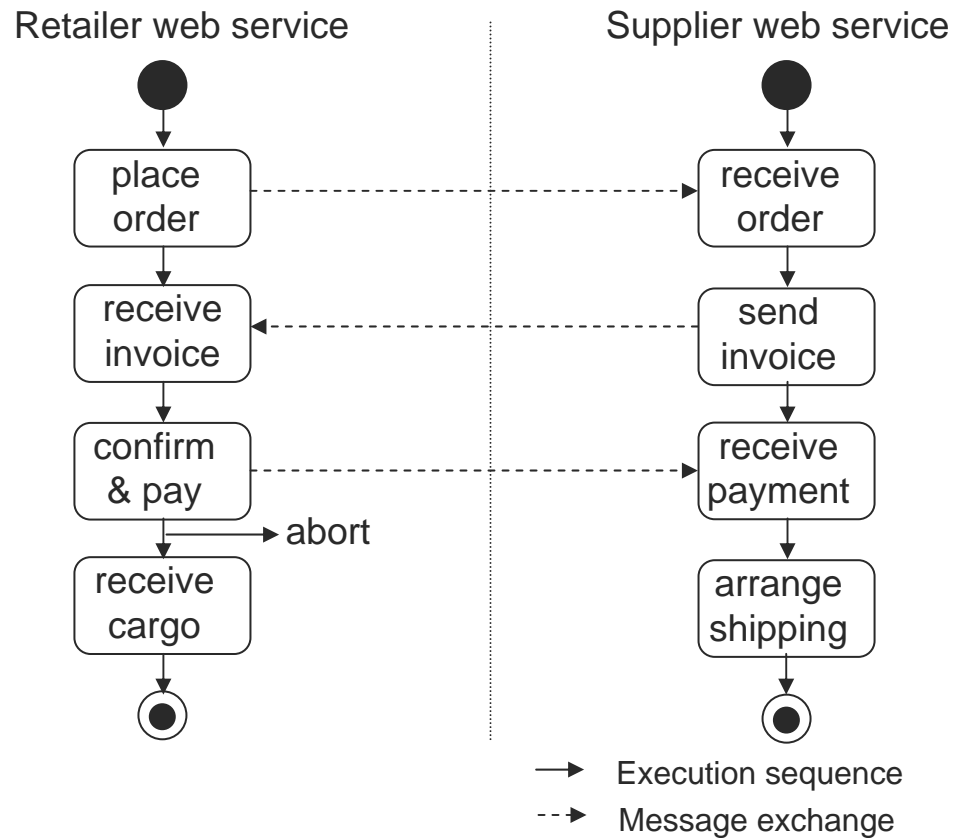
²City University of Hong Kong

Outline

- Introduction
- Motivation
- Process evolution
 - Evolution scenarios
 - Principles for evolution
- Discussions
- Conclusion

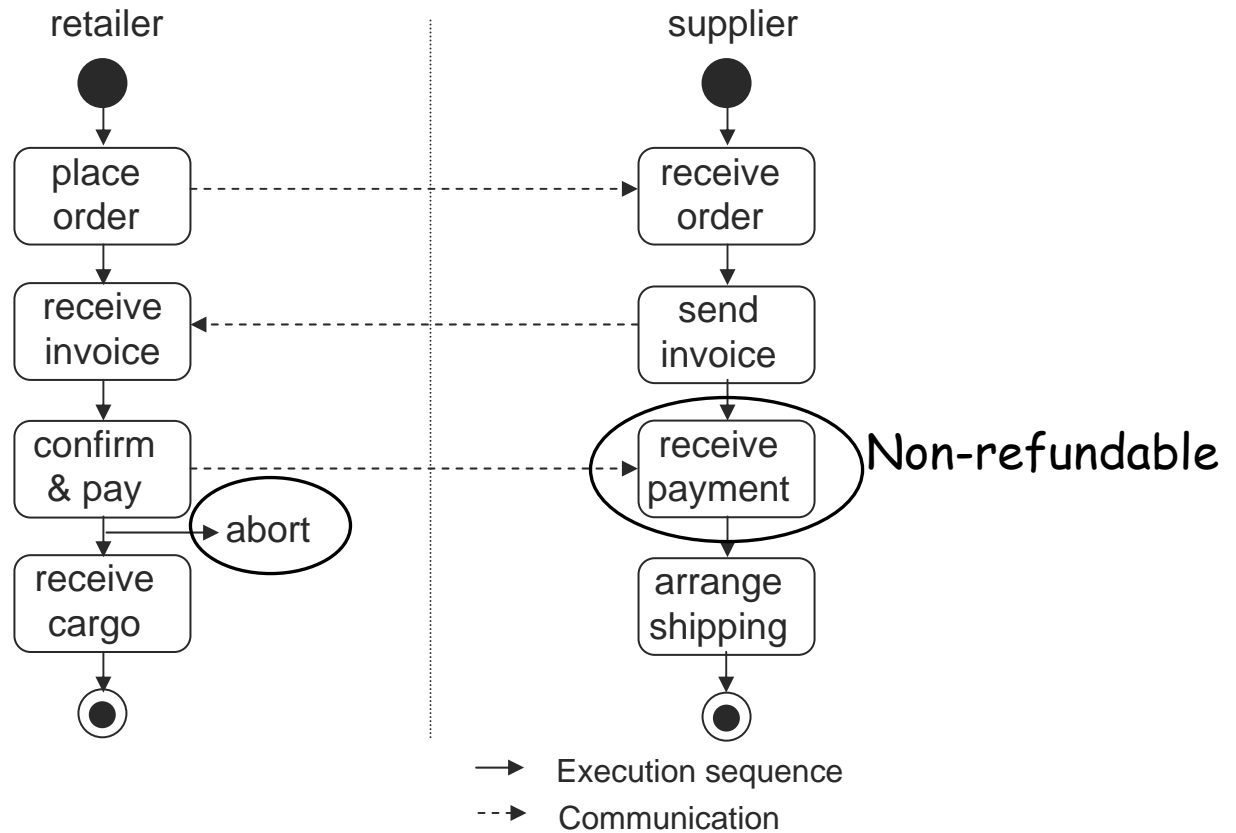
Introduction

■ Service composition



Introduction

- Value loss can occur

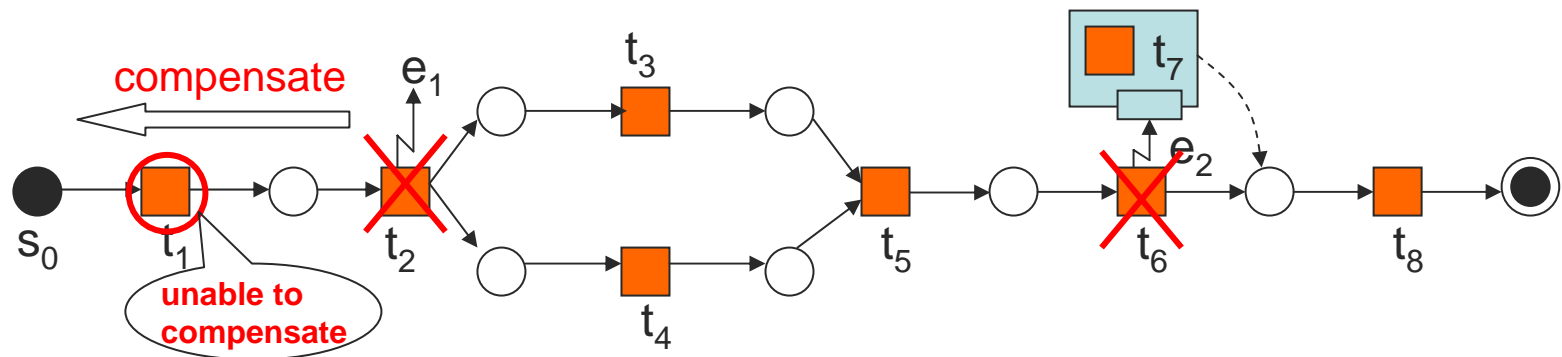


Introduction

- Atomicity atop service composition is desirable
 - Help avoid value loss
 - Difficult to enforce conventional database transaction
 - One alternative way to achieve atomicity is through the concept of ***atomicity sphere***

Introduction – Atomicity

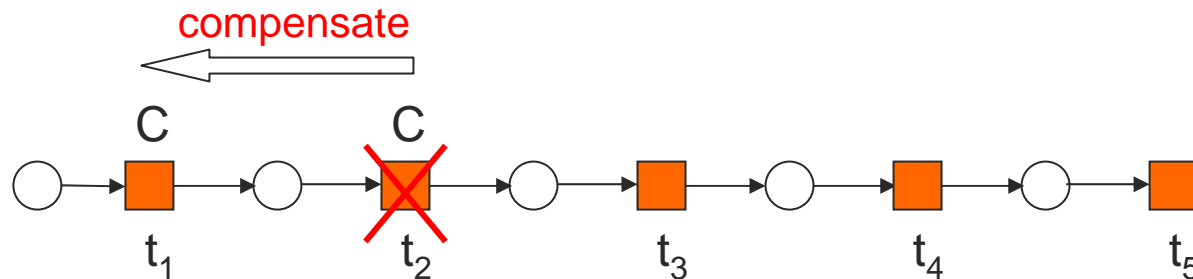
- A process satisfying atomicity sphere^[1]
 - Could either terminate successfully
 - Or, rewind to a state as if the process has not executed in case of failure



[1] C. Hagen and G. Alonso. Exception handling in workflow management systems. *TSE*, vol.26, no.10, Oct. 2000, pp.943-958.

Introduction

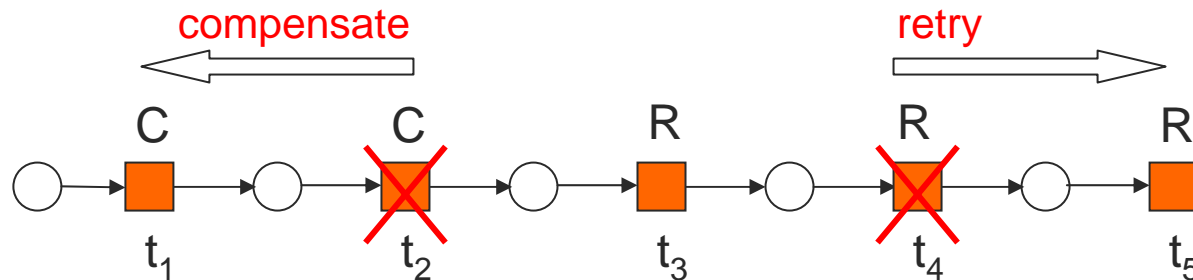
- To check atomicity
 - Tasks have two properties: compensability and retriability
 - A *compensable task* is one that can be undone one way or another in case the process fails or is canceled.



C: compensable NC: non-compensable
R: retrievable NR: non-retriable

Introduction

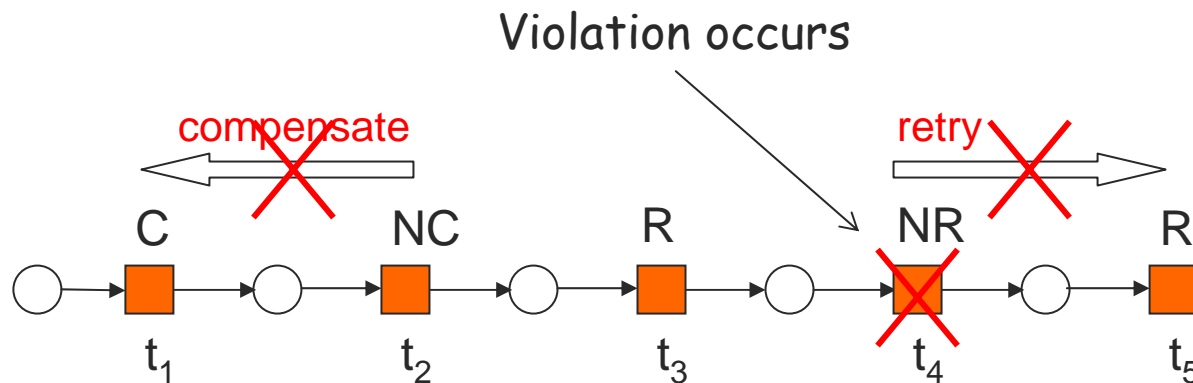
- To check atomicity
 - Tasks have two properties: compensability and retriability
 - A *compensable task* is one that can be undone one way or another in case the process fails or is canceled.
 - A *retriable task* is one that can finally succeed by retrying itself a finite number of times in face of failure.



C: compensable NC: non-compensable
R: retriable NR: non-retriable

Introduction

- Atomicity violation occurs:
 - a non-retriable (NR) task is executed after some non-compensable (NC) task.



C: compensable NC: non-compensable
R: retrievable NR: non-retriable

Introduction

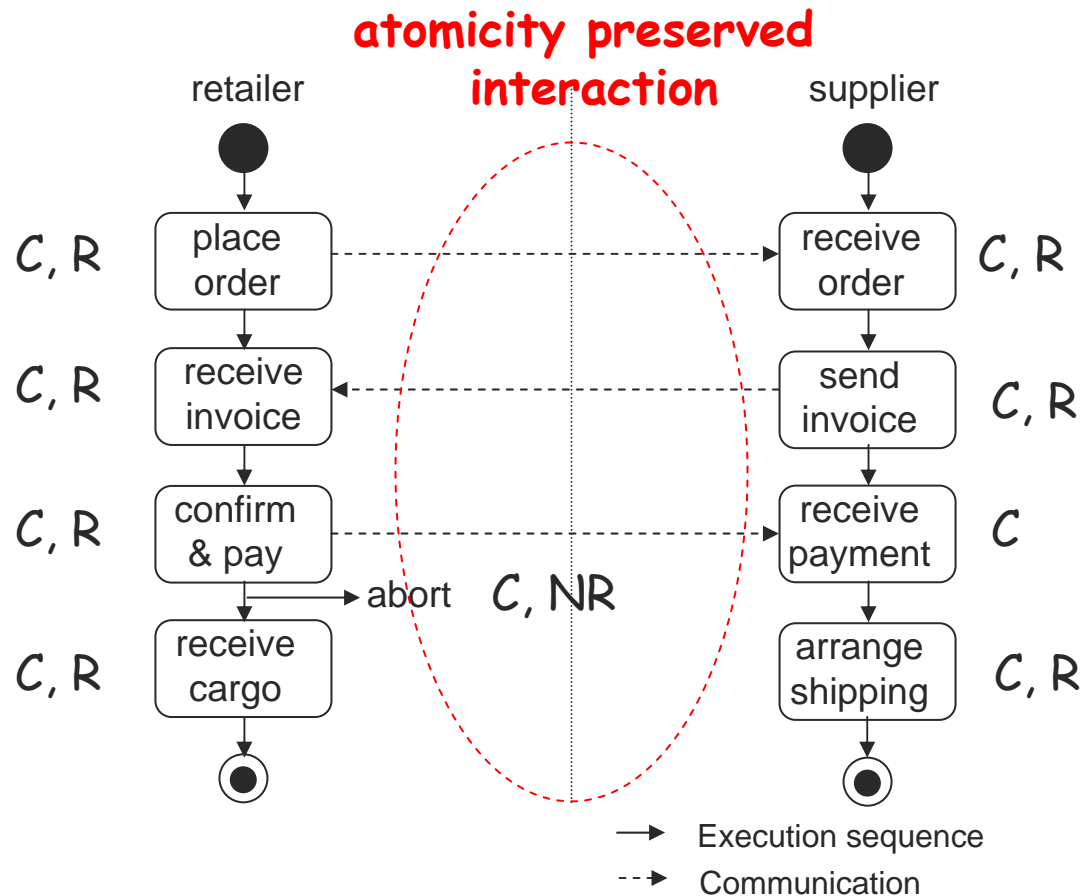
- Checking atomicity in service composition
 - Difficult due to only restricted views are exposed.
 - Our previous work^{[2][3]}
 - Address this problem by projecting atomicity information into the public views.
 - Use the public views to check atomicity instead of the original processes.

[2] C.Y. Ye, S.C. Cheung, and W.K. Chan. Publishing and composition of atomicity-equivalent services for B2B collaboration. *ICSE06*.

[3] C.Y. Ye, S.C. Cheung, W.K. Chan, and C. Xu. Local analysis of atomicity sphere for B2B collaboration. *FSE06*.

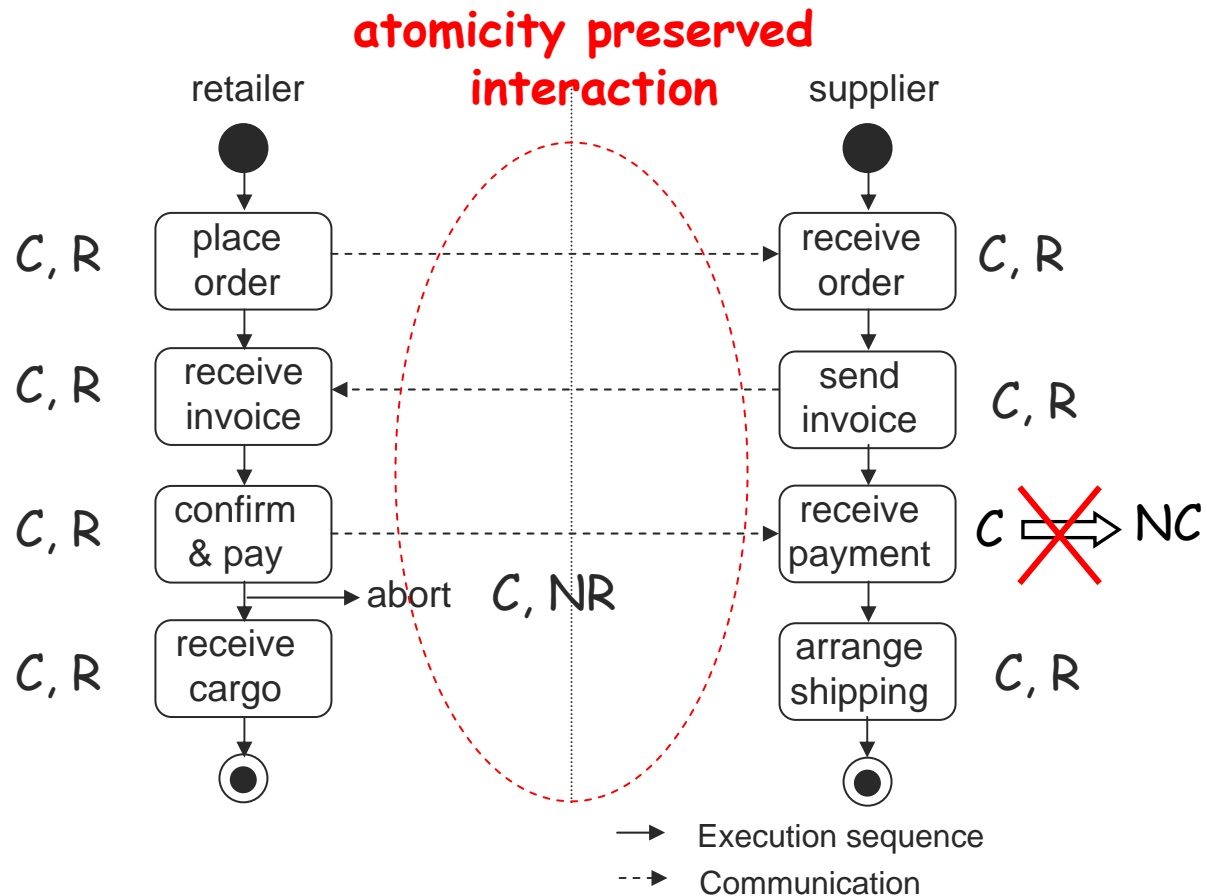
Motivation

- Our previous work assumes processes do not **change**



Motivation

- Our previous work assumes processes do not **evolve**

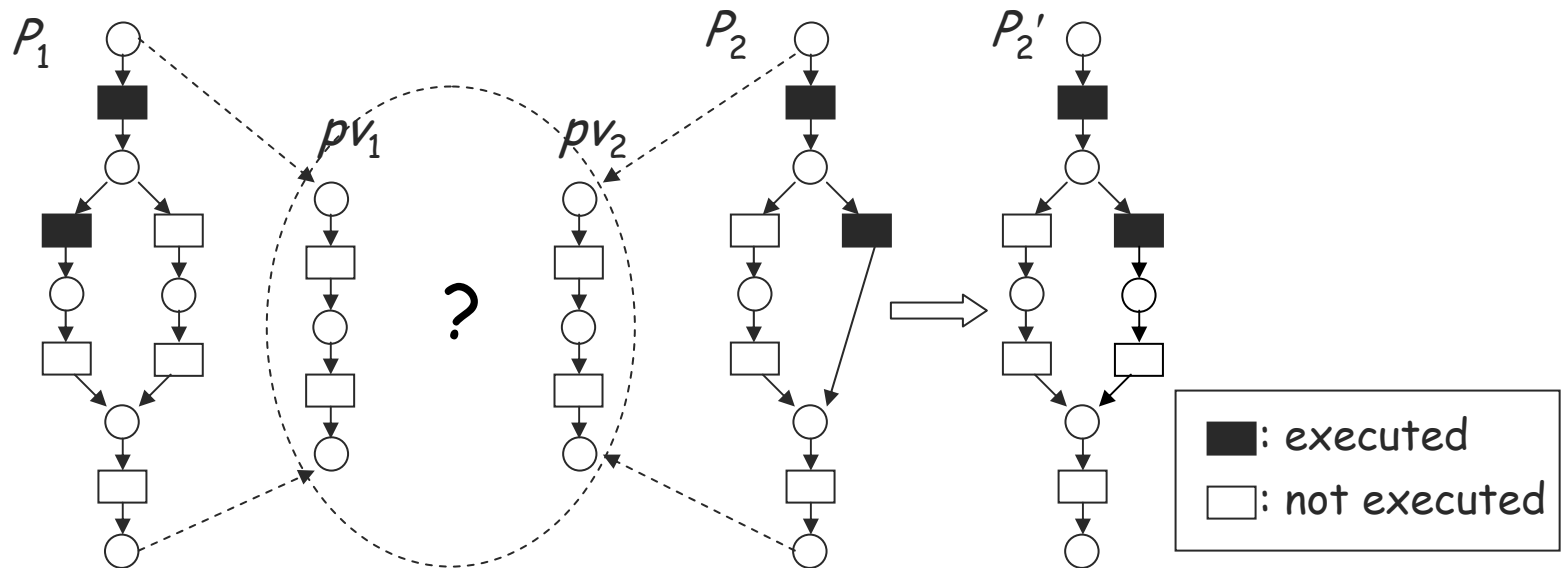


Motivation

- One way to handle process evolution
 - Service providers re-publish their public views.
 - Re-conduct the atomicity checking using the updated public views.
 - However, this strategy has limitations:
 - May need to delay evolution until collaboration completes
 - Aborting ongoing collaboration may cause value loss

Motivation

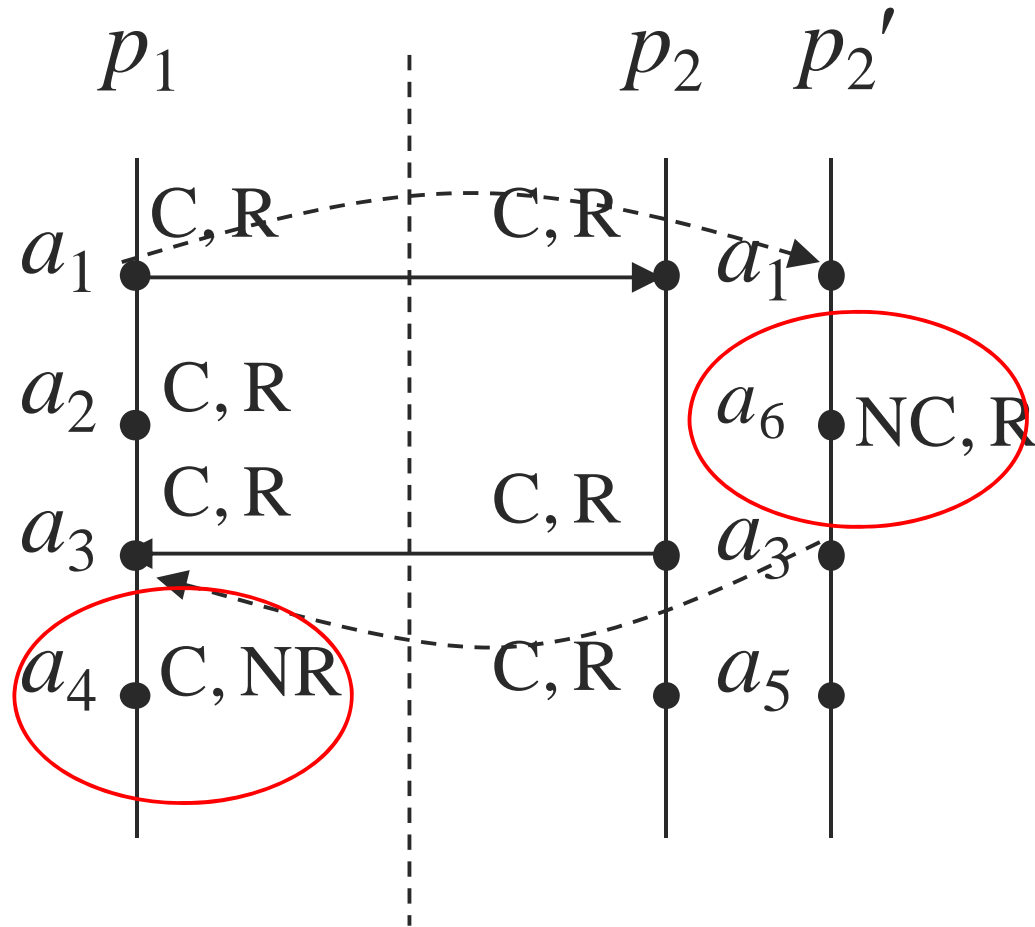
- Can a process evolve independently without affecting atomicity?
- Can the decision be made with local information?



Dynamic Evolution Operators

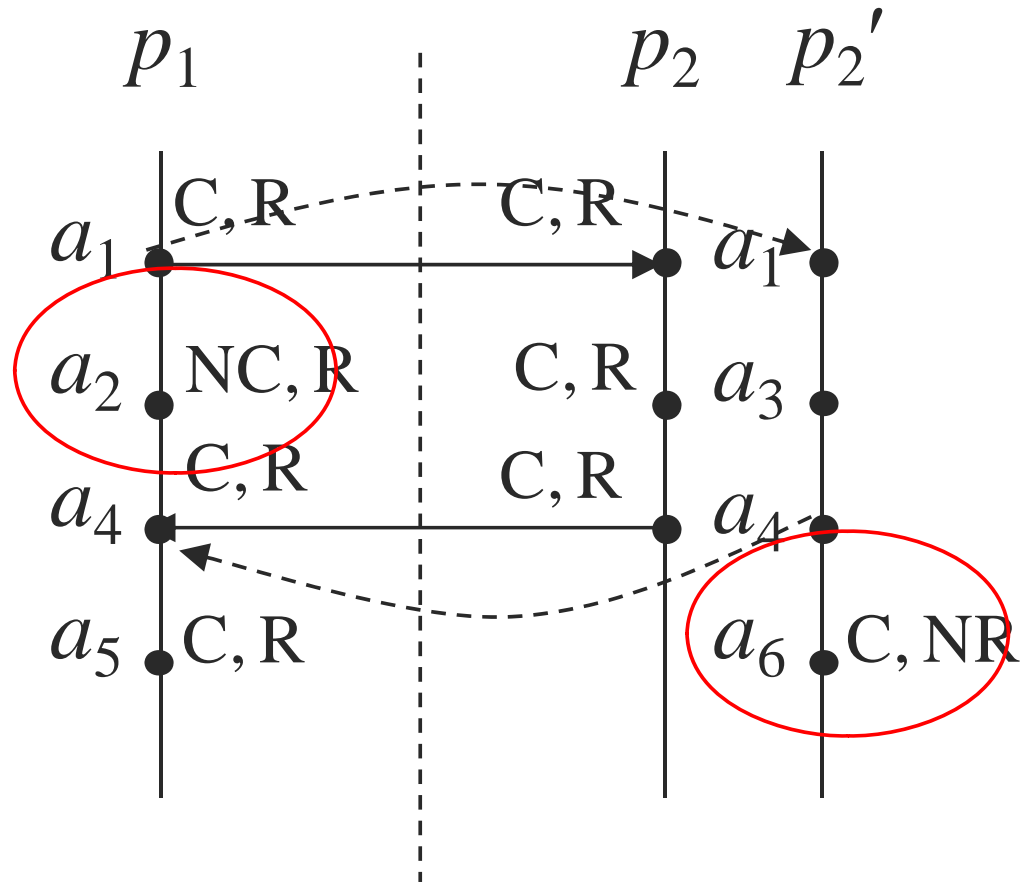
- Two basic operators
 - Insert an action
 - Remove an action
- Other operators
 - E.g., Change the property, relocate an action
 - Could be simulated by the two basic operators

Potential problems



Insert a NC action

Potential problems

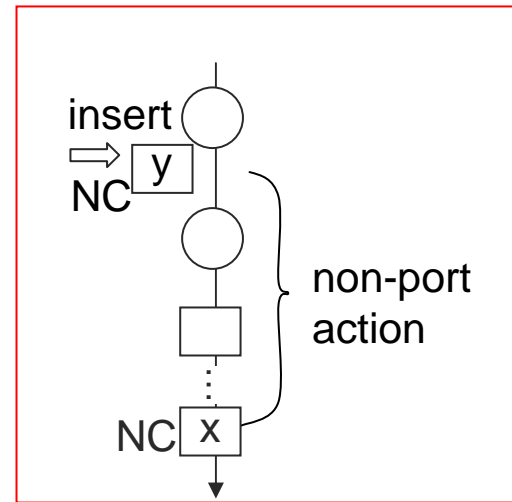
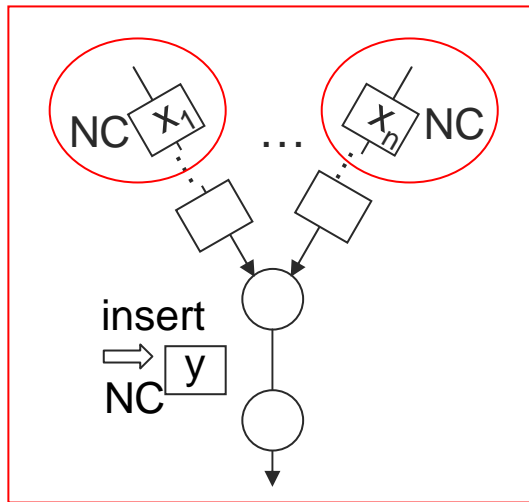


Insert a NR action

Underlying Principles of Evolution

- An NC task can be inserted if it is/will be executed after an NC task
- An NR task can be inserted if it isn't or won't be executed after an NC task
- A message exchange (or port action) can be skipped if its removal does not introduce a new NC-NR pair.

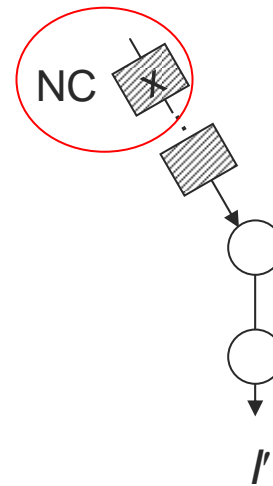
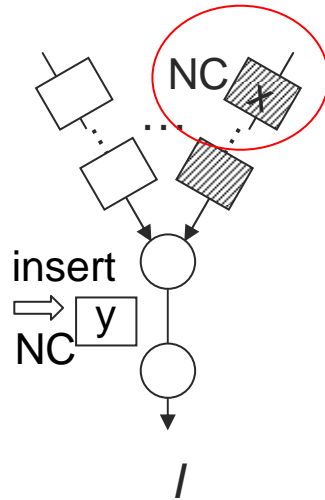
Situation 1



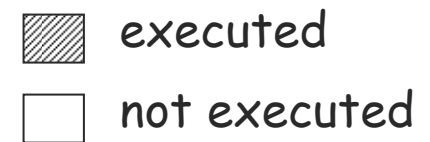
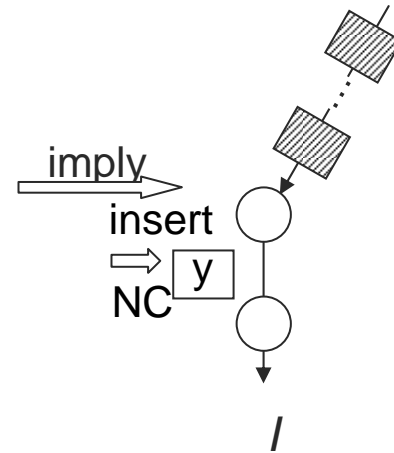
Insert a NC action



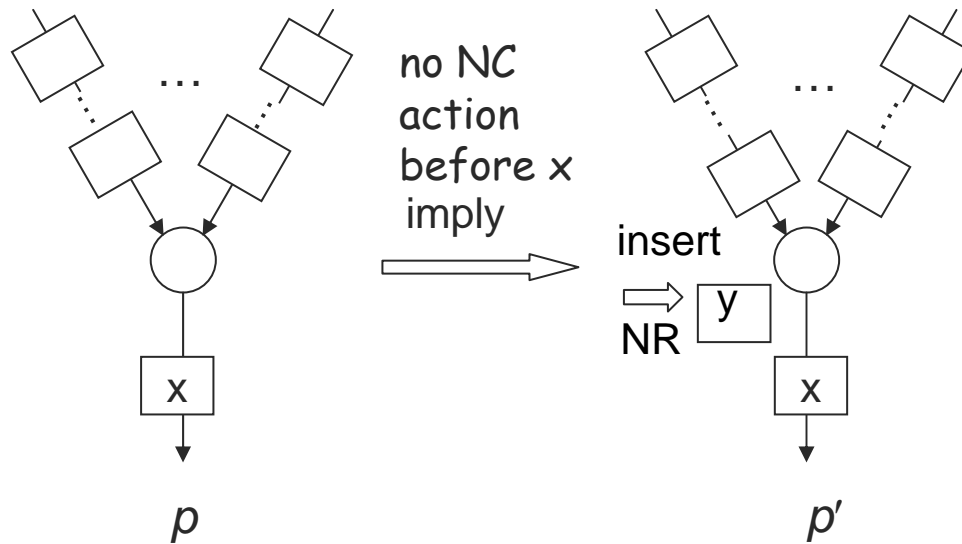
Situation 2



Insert a NC action



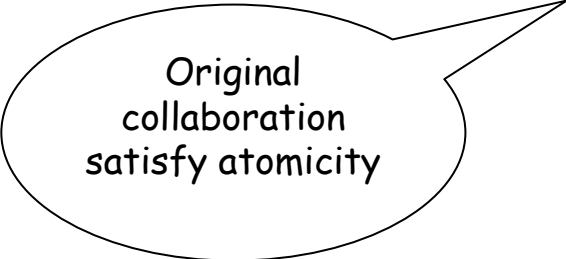
Situation 3



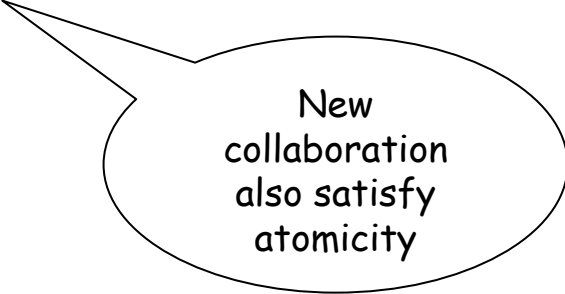
Correctness

- Let p be a process, and pv be its atomicity-equivalent public view.
- Suppose p' is the resultant process of inserting a new action into p satisfying the principles, then for any process q ,

if $\varphi(pv \parallel q) = \text{true}$, then $\varphi(p' \parallel q) = \text{true}$.

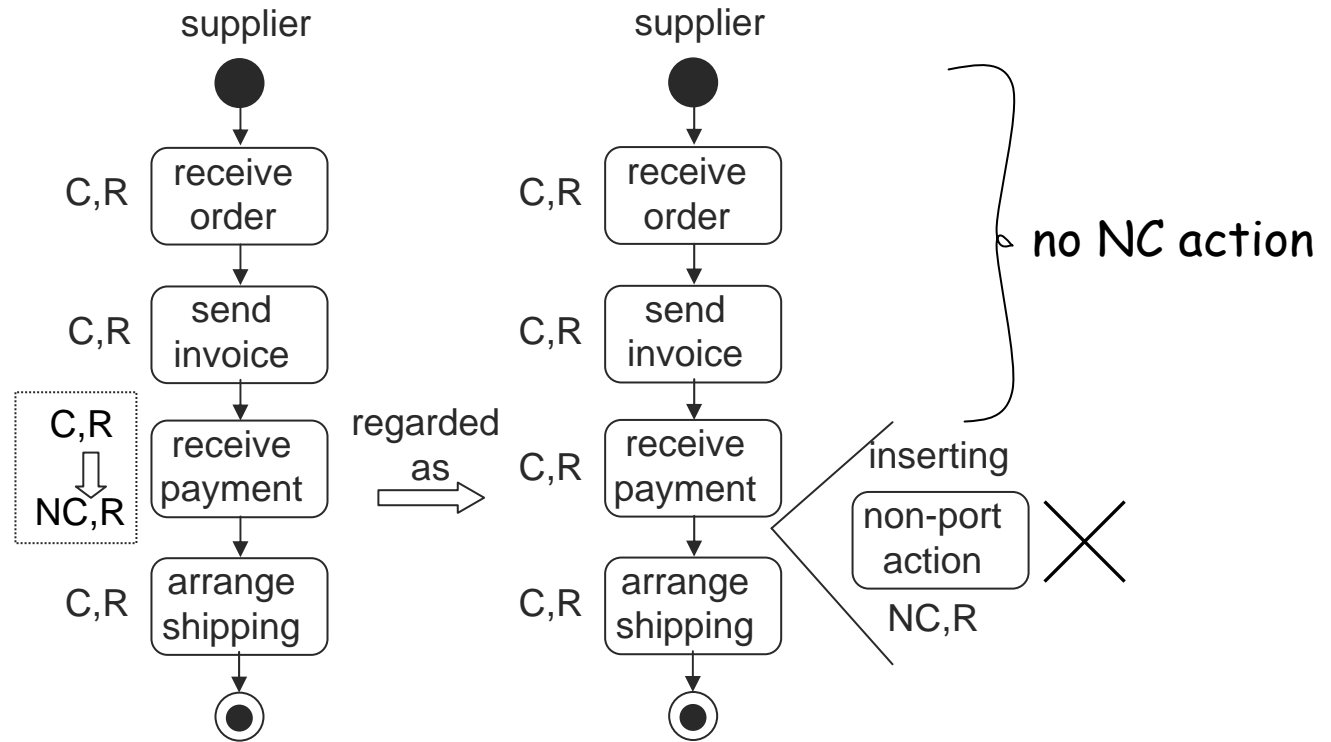


Original
collaboration
satisfy atomicity



New
collaboration
also satisfy
atomicity

Application



Discussions

- Related work
 - From the perspective of behavior consistency, not address atomicity
 - [Aalst et al. 02][Basten et al. 01][Casati et al. 98][Ellis et al. 95]
- Limitations
 - Completeness?
 - Missing operators
 - Missing situations

Conclusion and future work

- Evolution principles are analyzed
- Situations of these principles are discussed
 - To prevent atomicity violation in dynamic evolution
- Future work
 - Study the completeness issue

Thank you!