



## Analysis of Non-Functional Service Properties for Transactional Workflow Management

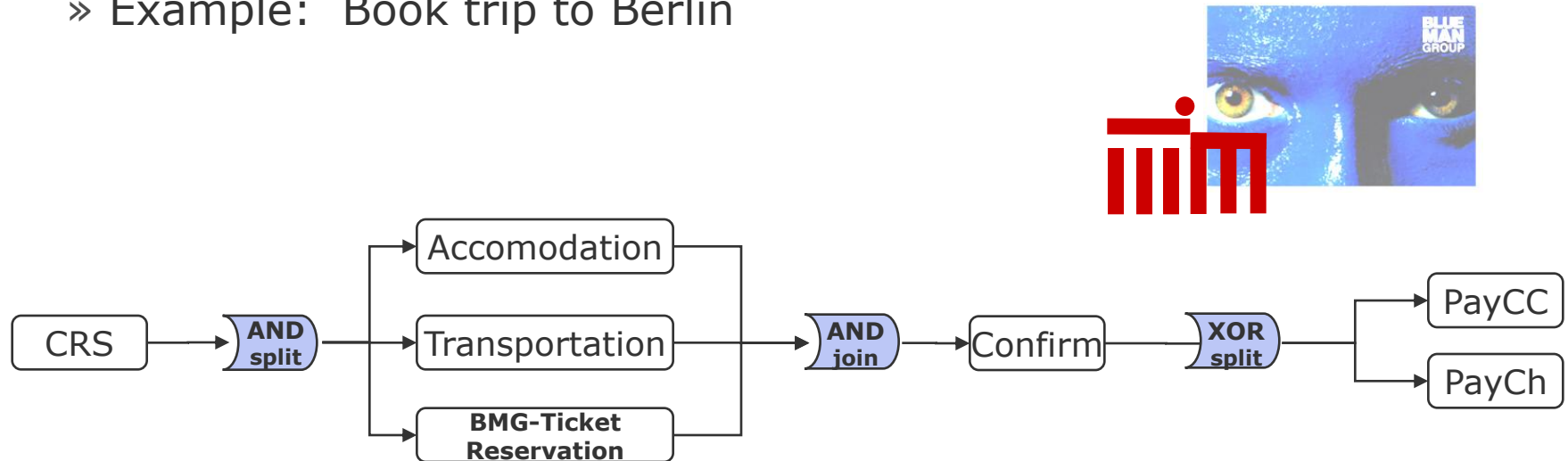
» Katharina Hahn, Heinz F. Schweppe

Institute for Mathematics and Computer Science, Freie Universität Berlin

November 12th 2008

# Motivation

- » Transactional Execution of Composed Services
- » Example: Book trip to Berlin



- » In case of failure:
  - » Specification of failure handling by the designer
  - » *Service binding at runtime* may expose different transactional properties [KuKo07,CJFY06]

Image taken from <http://www.blueman.com>

# Outline

---

- » Motivation
  
- » Introduction
  - » Problem Statement
  - » Transactional Service Model
  - » Workflow Patterns
  
- » Transactional Composition of Services
  
- » Atomicity Guarantees
  
- » Effects of Transactional Properties
  
- » Conclusion

# Introduction

## » Problem Statement

### » Transactional execution of composite services

- » "Correct" execution
- » Flexible failure handling

### » Automatically react to different failure situations

### » Integration of

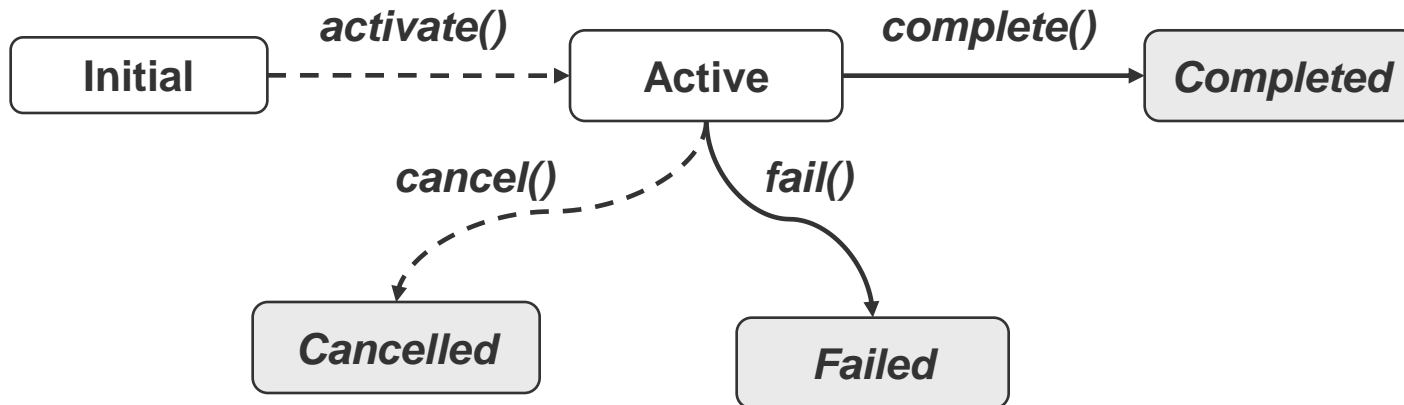
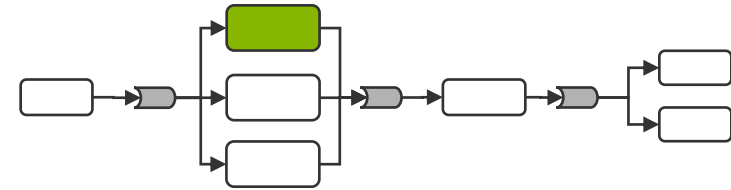
- » Workflow Management
- » Transactional Execution
- » Mostly: separate concerns
  - » BPEL engines
  - » WS-Coordination, WS-Tx
- » Automate transactional support

```
<faultHandlers>
  <catch faultName=„HotelOutOfRoomException“
    faultVariable=„HotelOfRoomVariable“>
    ...
  </catch>
  <catchAll>...</catchAll>
</faultHandlers>
```



# Transactional Service Model

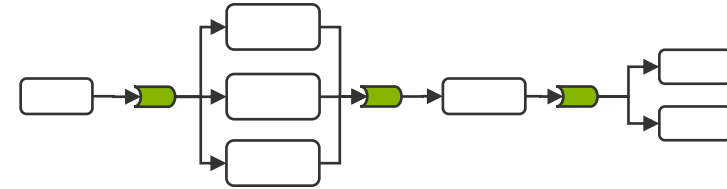
- » Formally modelling transactional services
  - » Event-based algebra [GRGH07]
  - » Service as a state-model
  - » Transitions:
    - » Internal: fail(), complete()
    - » External: activate(), cancel(), compensate()



# Workflow Patterns

## » Control Flow Patterns [\[ABEW00\]](#)

» A function that, given a set of services, returns the control flow



» SEQUENCE (sequential/serial routing)

» AND-split/join (parallel split/join, fork/synchronization)

» XOR (exclusive choice)

» N-OUT-OF-M

» ...

» Expose different execution semantics

# Outline

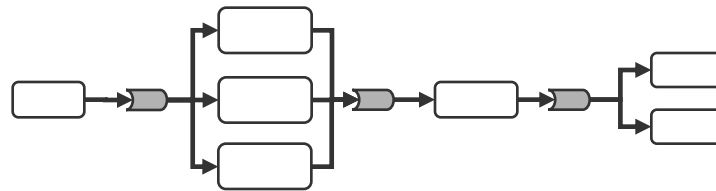
---

- » Motivation
  
- » Introduction
  - » Problem Statement
  - » Transactional Service Model
  - » Workflow Patterns
  
- » Composition of Services
  - » Transactional Composition
  - » Transactional Service Properties
  
- » Atomicity Guarantees
  
- » Effects of Transactional Properties
  
- » Conclusion

# Transactional Composition

- » Set of services
- » Foreach service, specify external transitions
- » Normal execution dependencies:
  - » Completion of  $s_1$  triggers activation of  $s_2$

$$depNrm(s_1, s_2) := dep(s_1.complete(), s_2.activate())$$



# Transactional Composition

## » Transactional execution dependencies

### » What happens in case of failure?

- » Execute alternatives

$$depAlt(s_1, s_2) := dep(s_1.fail(), s_2.activate())$$

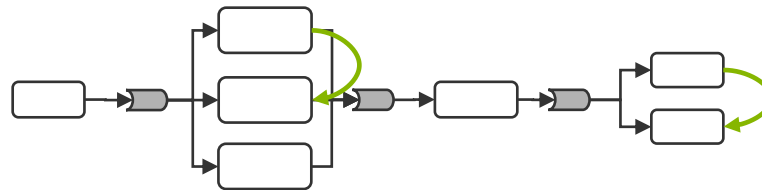
- » Cancel services

$$depCLn(s_1, s_2) := dep(s_1.fail(), s_2.cancel())$$

- » Compensate services

$$depCps(s_1, s_2) := dep(s_1.fail(), s_2.compensate())$$

$$\vee dep(s_1.compensate(), s_2.compensate())$$



## » Transactional dependencies rely on

- » Workflow pattern but also
- » Transactional properties of service

# Transactional Service Properties

## » Transactional Service Properties of Service S

» Originally in flexible transactions

» Compensatability ( $S.comp = \{0,1\}$ ):

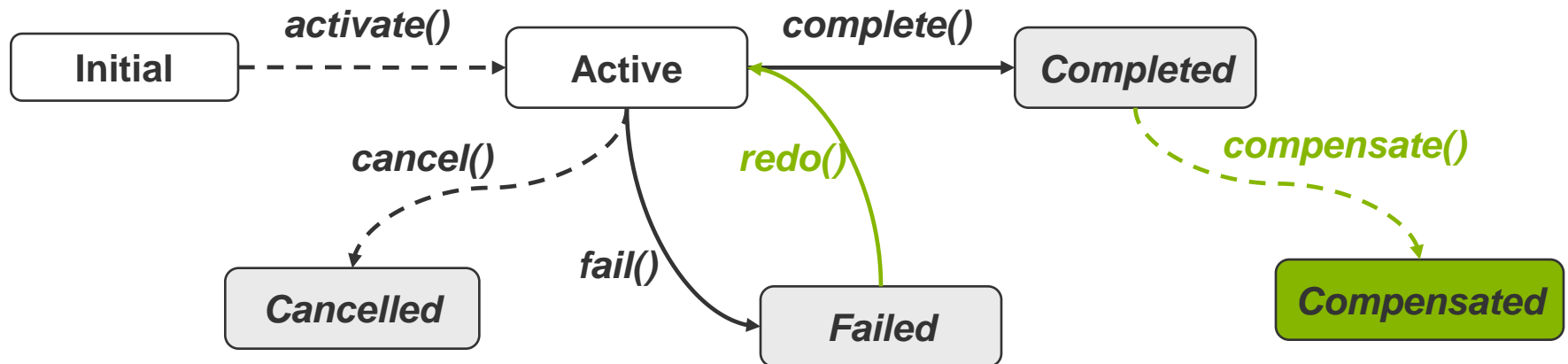
» If there is a service  $c$  which semantically undoes the effects of  $s$

» Modelled through *compensate()* and *Compensated*

» Redoability ( $S.redo = \{0,1\}$ ):

»  $S$  will definitely complete, if its activation is repeated  $n$  times

» Modelled through *redo()*



# Transactional Service Properties

- » Consistent Closure ( $S.consCompl = \{0,1\}$ ):
  - » Does service need compensation in case of failure?
    - » Printing/LogIn service
    - » Reservation/Registration
  - » Modelled through omitting service in backward-recovery
  
- » Service properties are denoted as triple:

$$P_S = (s.comp, s.consCompl, s.redo)$$

- » E.g.  $S$  with  $P_S = (0,1,1)$ 
  - » is not compensatable, needs consistent closure, is redoable

# Outline

---

- » Motivation
  
- » Introduction
  
- » Composition of Services
  - » Transactional Composition
  - » Transactional Service Properties
  
- » Atomicity Guarantees
  
- » Effects of Transactional Properties
  
- » Conclusion

# Atomicity-Guarantees

## » Transactional Model

» What exactly is "correct" execution?

» Successful execution specified by termination states (ATS)

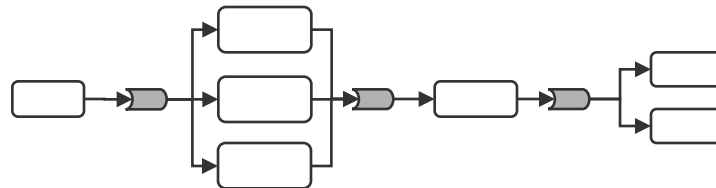
» Several atomicity guarantees in databases

» Strict, Semantic, Semi-Atomicity [[Gar83](#),[ZNBB94](#)]

» Semi-Atomicity for composite service with ATS and introduced properties

» Either all services belonging to one valid execution path to an ATS are completed and all services not included on that path which demand consistent closure are not completed

» Or no service demanding consistent closure is completed



# Outline

---

- » Motivation
- » Introduction
- » Composition of Services
- » Atomicity Guarantees
- » Effects of Transactional Properties on Composite Service
  - » Generally
  - » Applied:
    - » AND-pattern
    - » XOR-pattern
- » Conclusion

# General Effects

- » How to support semi-atomicity of a workflow?
  - » Try to avoid blocking situations
  
- » Effects of transactional properties on workflow patterns
  - » **Transactional pattern property**
    - » Analyze whole workflow
    - » Generalize patterns and services as workflow elements
    - » Workflow properties
      - » Compensatability (cCompensatability), consistent closure, redoability
  - » **Execution type and order**
    - » Consider failure model
    - » Change control flow to guarantee semi-atomicity
    - » Coordinated ("worst case")
      - » I.e., an atomic sub-transaction using WS-Tx
    - » Independent
      - » Sequential
      - » In any order
      - » Parallel
  - » **Recovery mechanism**
    - » Which services have to be compensated in case of backward recovery?

# Effects on AND-pattern

## » Pattern Properties

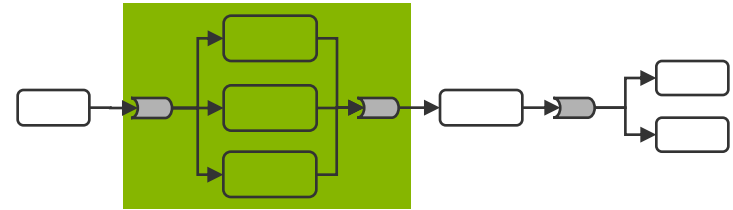
$$WP_{AND}(S).comp = 1 \Leftrightarrow \forall s \in S : s.comp = 1$$

$$WP_{AND}(S).consCompl = 1 \Leftrightarrow \exists s \in S : s.consCompl = 1$$

$$WP_{AND}(S).redo = 1 \Leftrightarrow \forall s \in S : s.redo = 1$$

$$WP_{AND}(S).cComp = 1 \Leftrightarrow \forall s \in S : s.cComp = 1 \Leftrightarrow$$

$$\forall s \in S : s.comp = 1 \vee s.consCompl = 0$$



## » Recovery of Pattern

- » Compensate all services which need consistent completion

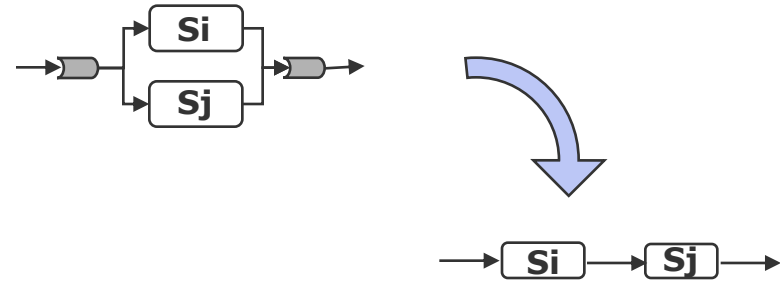
$$\forall s \in S, s.consCompl = 1 : s.compensate()$$

# Effects on AND-pattern

## » Execution Type and Order

### » Sequential $S_i \rightarrow S_j$

1.  $P_{S_i} = (*, *, 0) \wedge P_{S_j} = (0, 1, 1)$
2.  $P_{S_i} = (*, 0, 0) \wedge P_{S_j} = (0, 1, *)$
3.  $P_{S_i} = (1, *, 0) \wedge P_{S_j} = (0, 1, *)$



### » Coordinate $S_i$ and $S_j$

$$P_{S_i} = P_{S_j} = (0, 1, 0)$$

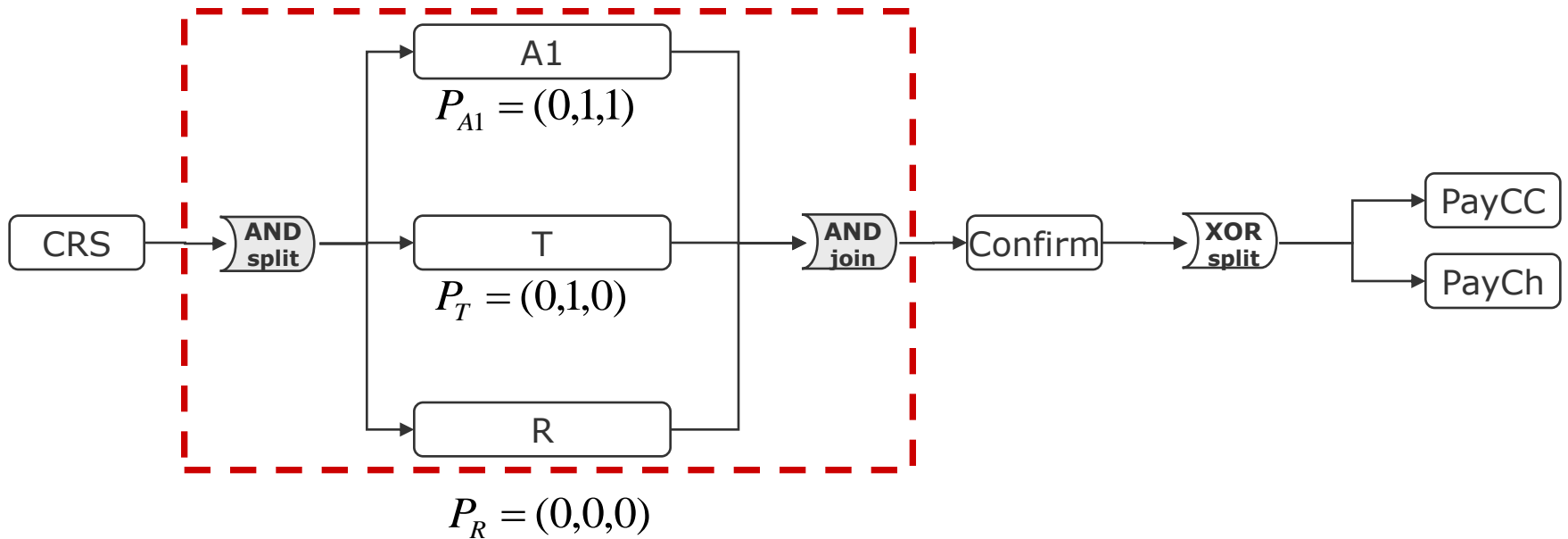
### » Parallel

- » In any other case

$$P_S = (s.comp, s.consCompl, s.redo)$$

# Analysis: AND-pattern

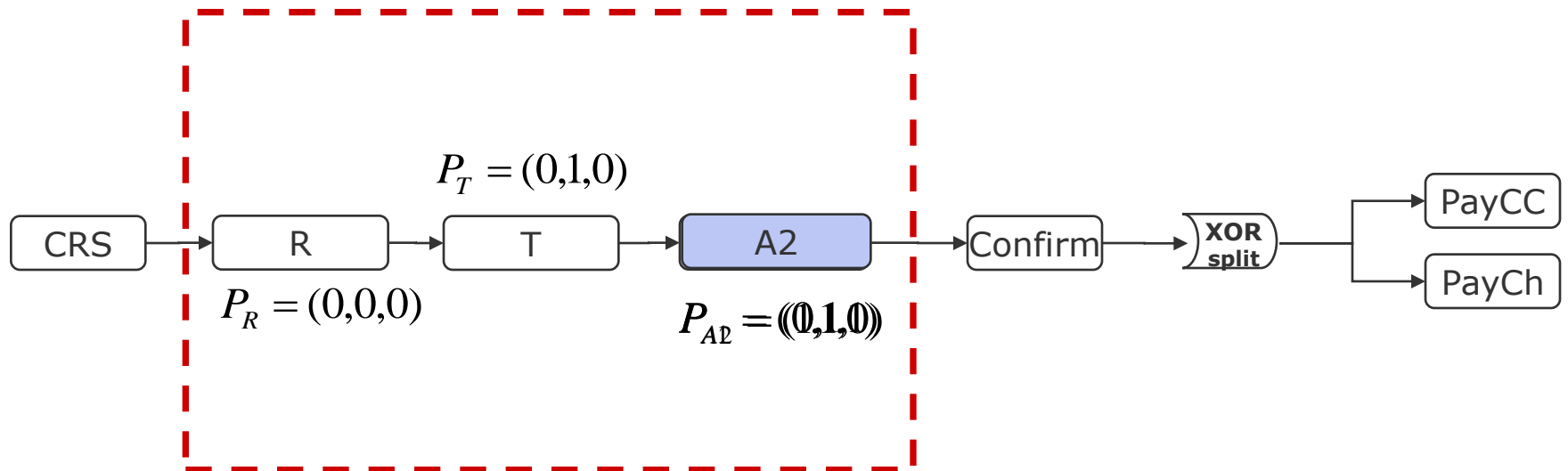
» Example



$$P_S = (s.comp, s.consCompl, s.redo)$$

# Analysis: AND-pattern

- » Example
  - » Order R and T, and T and A1
  - » Dynamically bind A2 with different properties



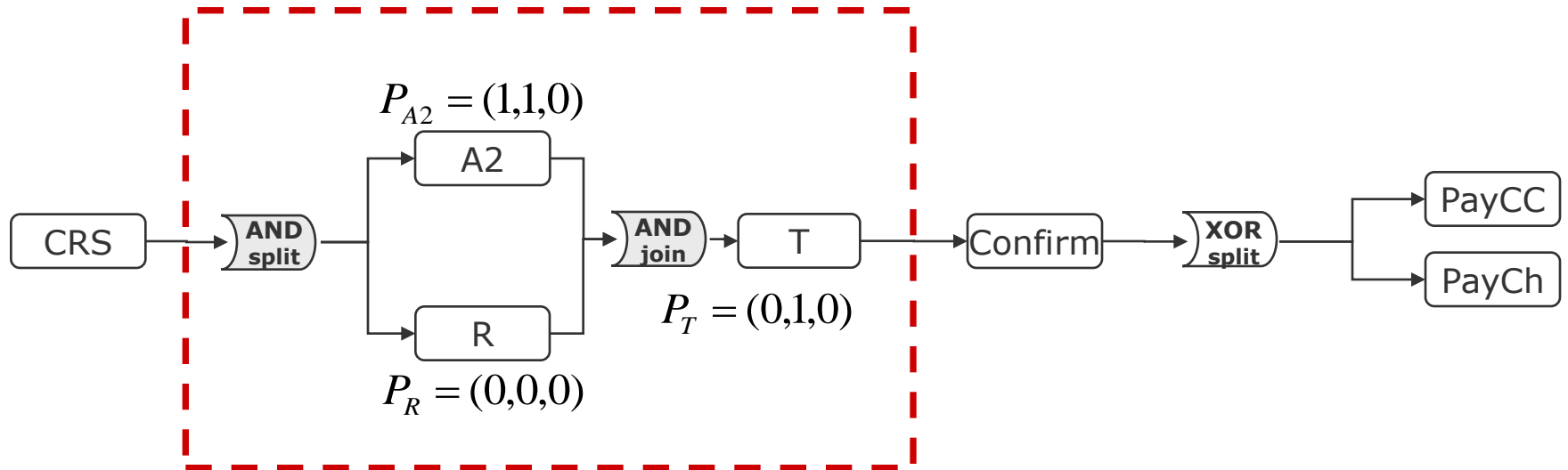
$$P_S = (s.comp, s.consCompl, s.redo)$$



# Analysis: AND-pattern

## » Example

» Parallelize A2 and R, followed by T

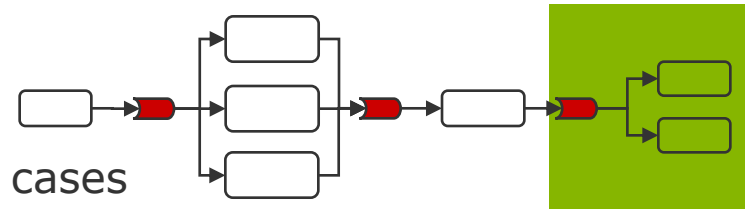


$$P_S = (s.comp, s.consCompl, s.redo)$$

## Effects on XOR-pattern

### » Pattern Properties

- » Cannot definitely be pre-determined for all cases
- » All depend on the executed path but redoability
  - » Redoable: As soon as one service is redoable



### » Execution Type and Order

- » Static, given by the pattern
- » Only one at a time

### » Recovery if cCompensatable

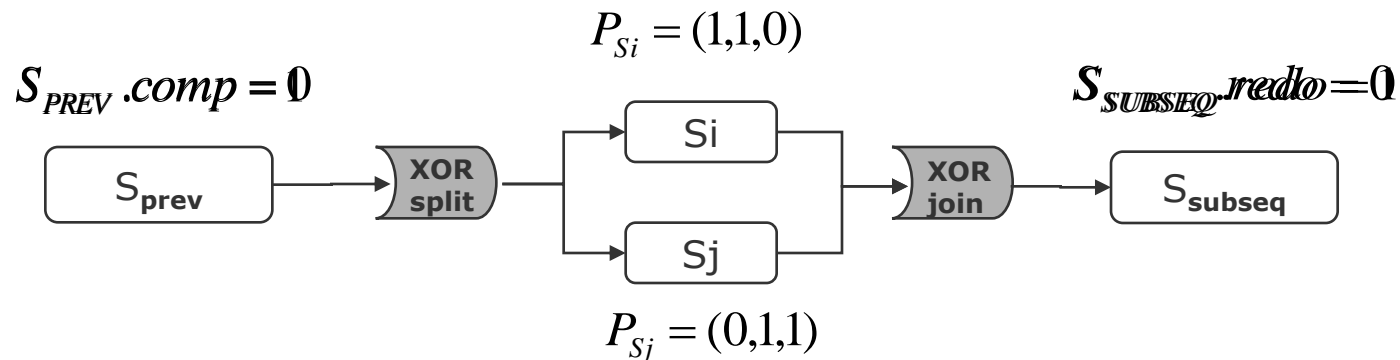
- » Depends on executed service

### » But, if choices are equal alternatives

- » Selection based on transactional properties
- » And surrounding workflow

# Analysis: XOR-pattern

- » Example
- » Service selection
  - » If subsequent workflow is not redoable, failure might occur
  - » Choose compensatable alternative





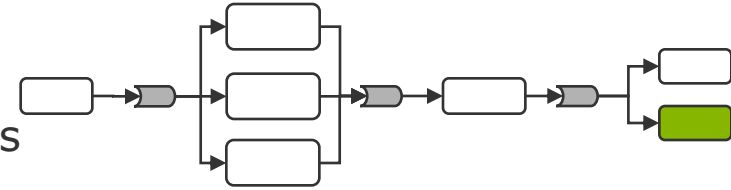
# Outline

---

- » Motivation
- » Introduction
- » Composition of Services
- » Atomicity Guarantees
- » Effects of Transactional Properties on Composite Service
  - » Generally
  - » Applied:
    - » AND-pattern
    - » XOR-pattern
- » Conclusion

## Conclusion

- » Problem
  - » Transactional execution of composite services



- » Model for "correct" workflow execution
  - » Transactional Composition of Services
  - » Transactional Service Properties
  - » Workflow Patterns
  - » Semi-Atomicity
- » Analysis
  - » Of properties
  - » And patterns
- » Automated workflow adaptations
  - » To ensure semi-atomic execution
  - » Failure handling
  - » In the presence of dynamic service binding

## References

### [ABEW00]

Aalst et al.: Workflow Modeling Using Proclets. In *Proceedings of 7th Intl. Conference on Cooperative Information Systems (COOPIS) 2000*, pages 198-209.

### [GRGH07]

Gaaloul et al.: Verifying composite service transactional behavior using event calculus. In *Proceedings of 14th Intl. Conference on Cooperative Information Systems (COOPIS) 2007*.

### [CJFY06]

Chakraborty et al.: Toward Distributed Service Discovery in Pervasive Computing Environments. In *IEEE Transactions on Mobile Computing* 2006.

### [KuKo07]

Küster, König-Ries: Semantic Service Discovery with DIANE Service Descriptions. In *Proceedings of Intl. Workshop on Service Composition*, Silicon Valley, USA, November 2007.

### [Gar83]

Garcia-Molina: Using semantic knowledge for transaction processing in a distributed database. In *ACM Transactions on Database Systems* 8(2):186-213, 1983.

### [ZNBB94]

Zhang et al.: Ensuring Relaxed Atomicity for Flexible Transactions in Multidatabase Systems. In *SIGMOD Records* 23(2), 1994