Counterexample Guided Synthesis of Monitors for Realizability Enforcement

Matthias Güdemann

Gwen Salaün

Meriem Ouederni

Inria informatics mathematics

LIG

Choreographies

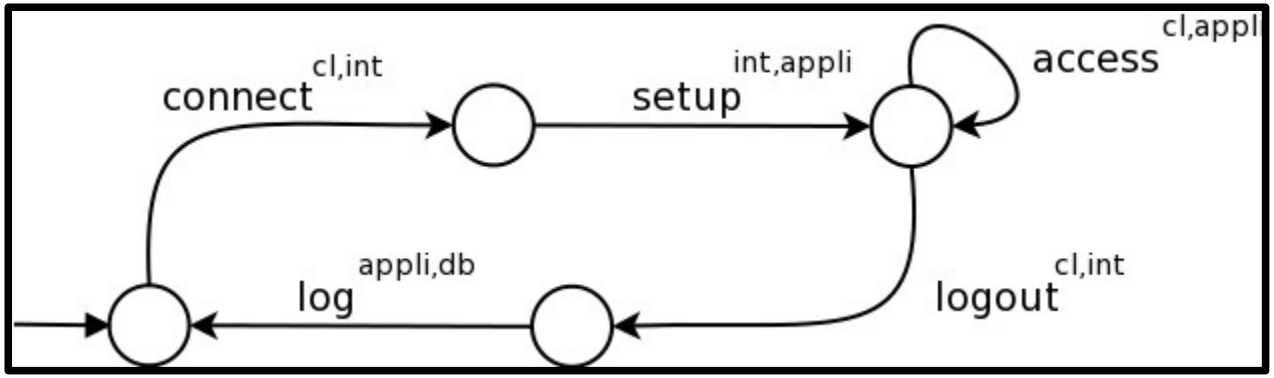
- Global contract specifications
  - Participants, communication, message sequence, choices, synchronizations etc.
- Used e.g.
  - Service Oriented Computing, Web services, Cloud Computing, Business Processes
  - In general: contract specification of interaction-based systems
- Implementation
  - Distributed system
  - Top-down or bottom-up development

Overview

- Choreography Introduction
- Verification of Choreographies
- Enforcing Realizability of Choreographies
- Summary

Example

- Using an application in the cloud
  - Client connects to service interface
  - Interface sets up application
  - Client accesses application
  - Client logs out from interface
  - Application logs events to database

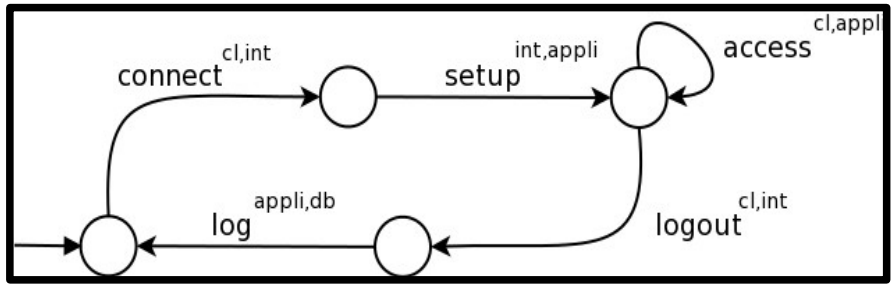


Example

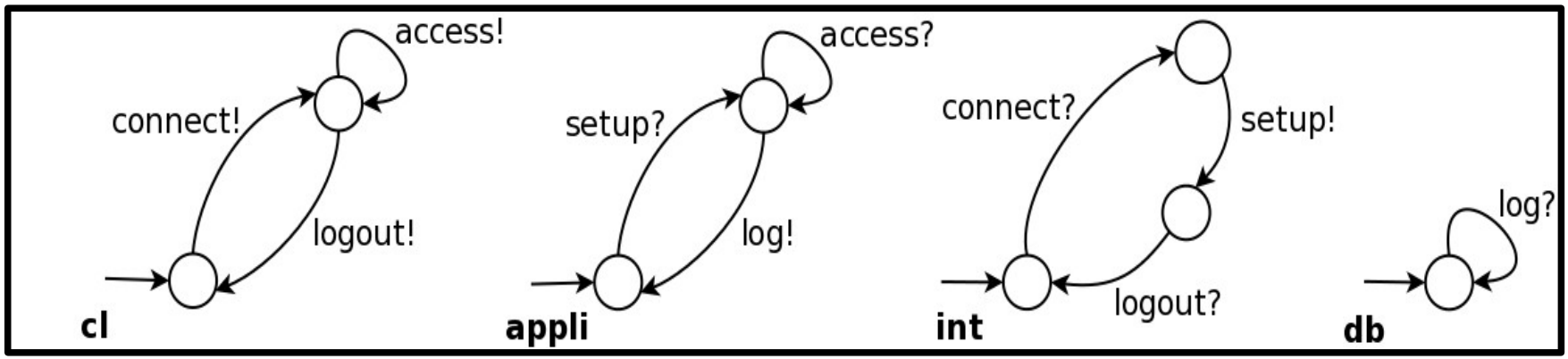
- Distributed implementation
  - Projection onto single peers
  - Parallel composition of peers
  - Message exchange via asynchronous communication (FIFO Buffers)

Example

- Distributed implementation

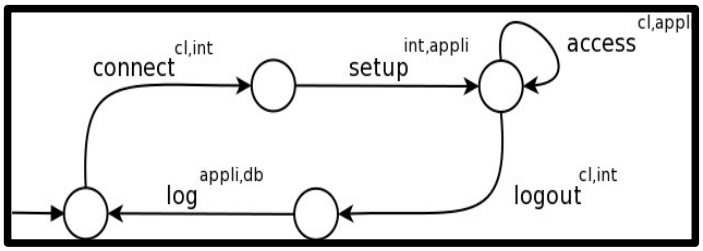


Projection: relabeling and reduction with CADP

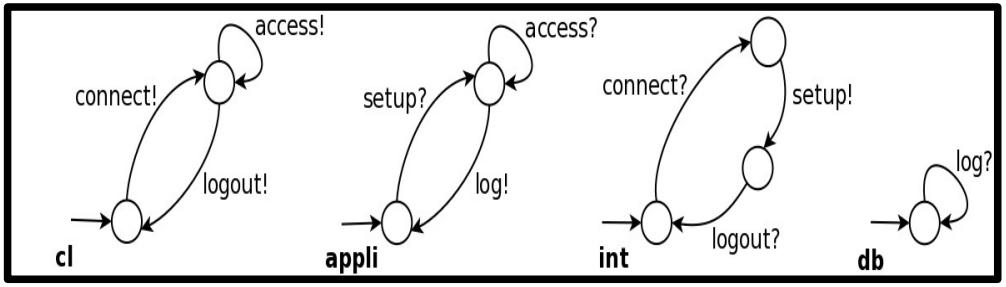


Realizability

Is the behavior of the choreography equivalent to the distributed implementation?

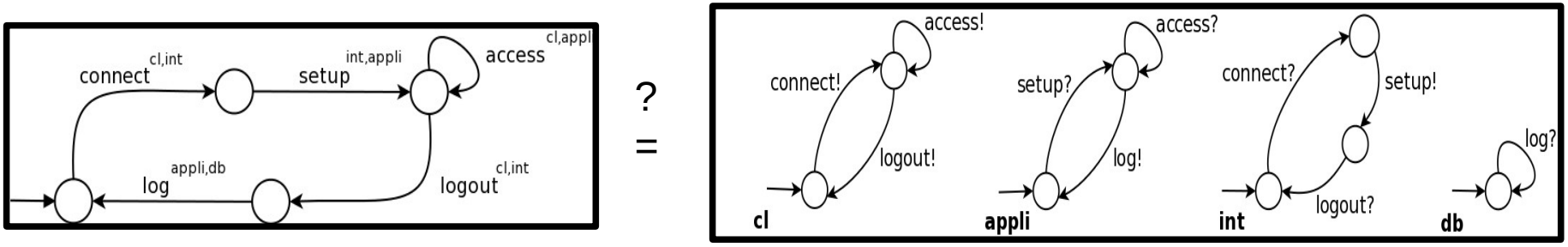


?
=



Realizability

Is the behavior of the choreography equivalent to the distributed implementation?



No! Counterexample: `connect, access`

Overview

- Choreography Introduction
- Verification of Choreographies
- Enforcing Realizability of Choreographies
- Outlook

Verifying Choreographies

- Model-Checking
  - Verification with EVALUATOR
  - Specification of temporal logic formulas
  - Reachability, liveness, deadlocks etc.

Verifying Choreographies

- Realizability
  - Top-down development approach
  - Equivalence check with BISIMULATOR
- Conformance
  - Analogous to realizability
  - Bottom-up approach (no projection required)
- Synchronizability
  - Equivalence check of synchronous and asynchronous system
  - Synchronizable systems are bounded

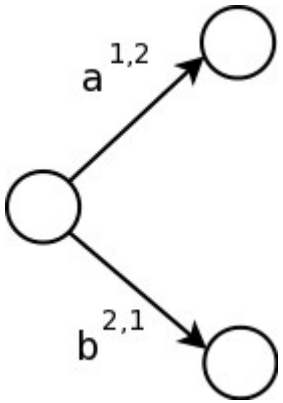
Verifying Choreographies

A non-faulty choreography is realizable if

- It is synchronizable and
- the behavior of the synchronous distributed system is equivalent to the global contract of the choreography (Fu et al. [POPL12])

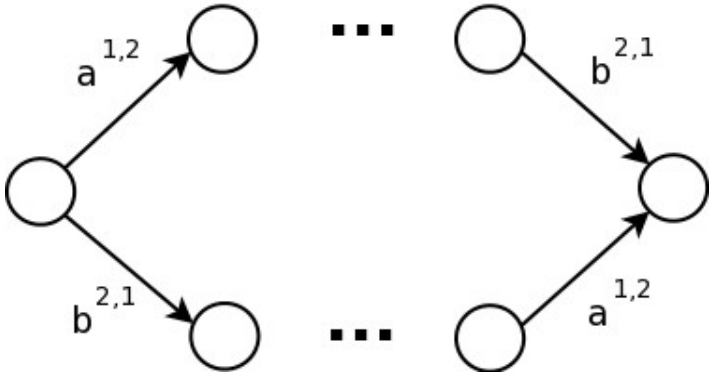
Non-faulty Choreographies

potential problem at selection:



Non-faulty Choreographies

potential problem at selection:

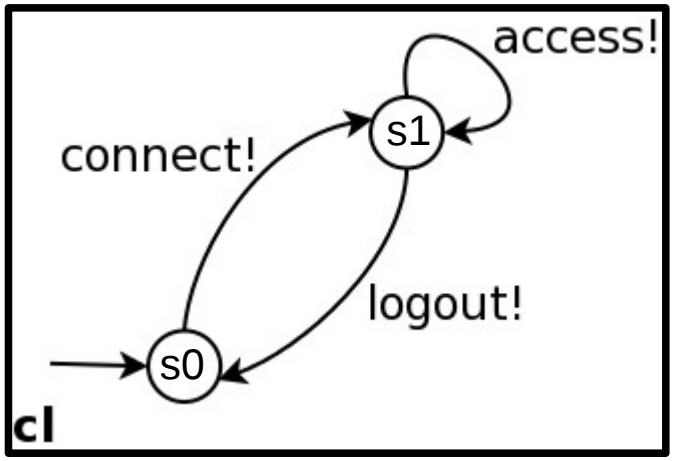


(Possibly) no problem if confluent!

- Verification using XTL (extensible temporal logic)
- Analysis of choices and interleavings

Transformation to Lotos NT

- Simple choice encoding in LNT

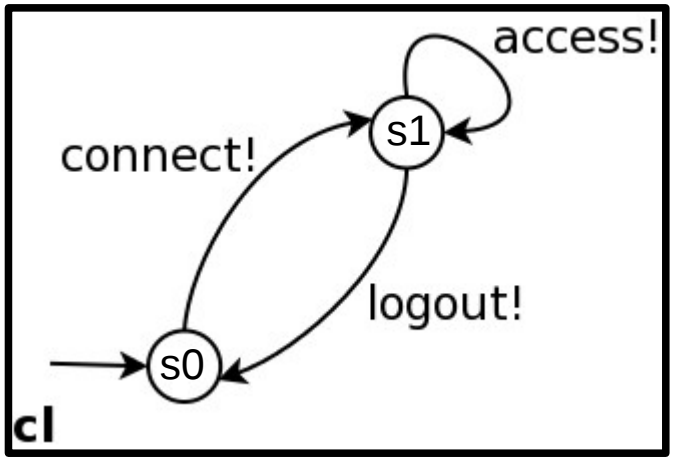



```
process s0[...]
  connect; s1[];
end process

process s1[...]
  select
    access; s1[];
  []
    logout; s0[];
  end select
end process
```

Transformation to Lotos NT

- Simple choice encoding in LNT




```
process s0[...]
  connect; s1[];
end process

process s1[...]
  select
    access; s1[];
  []
    logout; s0[];
  end select
end process
```

  - Transformation to BCG / SVL (REDUCTOR, smart composition)
  - Analysis with various CADP tools

Overview

- Choreography Introduction
- Verification of Choreographies
- Enforcing Realizability of Choreographies
- Summary

Enforcing Realizability

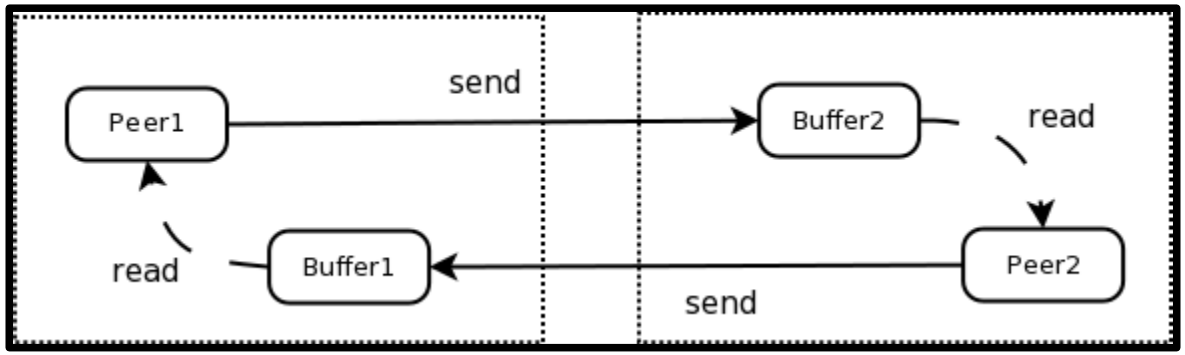
- What if a choreography is not realizable?

- Formal analysis benefits:
  - CADP will provide counterexamples
  - Explanation why the system is not realizable

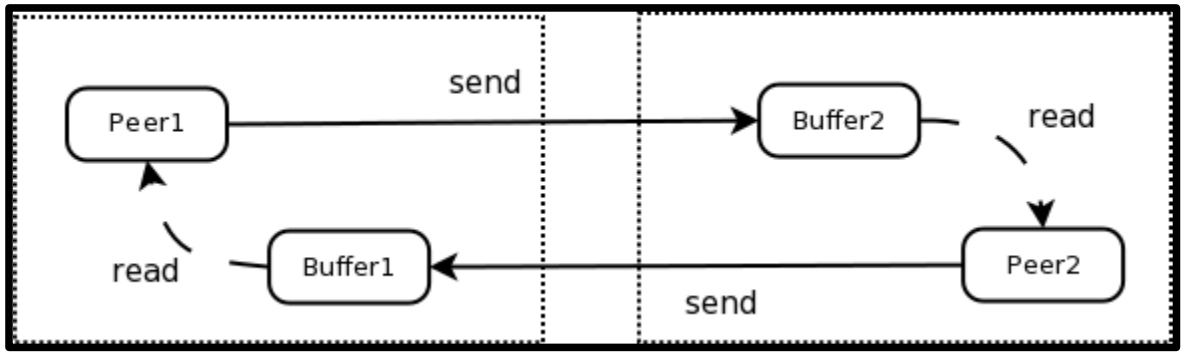
Enforcing Realizability

- What if a choreography is not realizable?
- Formal analysis benefits:
  - CADP will provide counterexamples
  - Explanation why the system is not realizable
- Our solution:
  distributed monitors which control send messages without changing the original peers

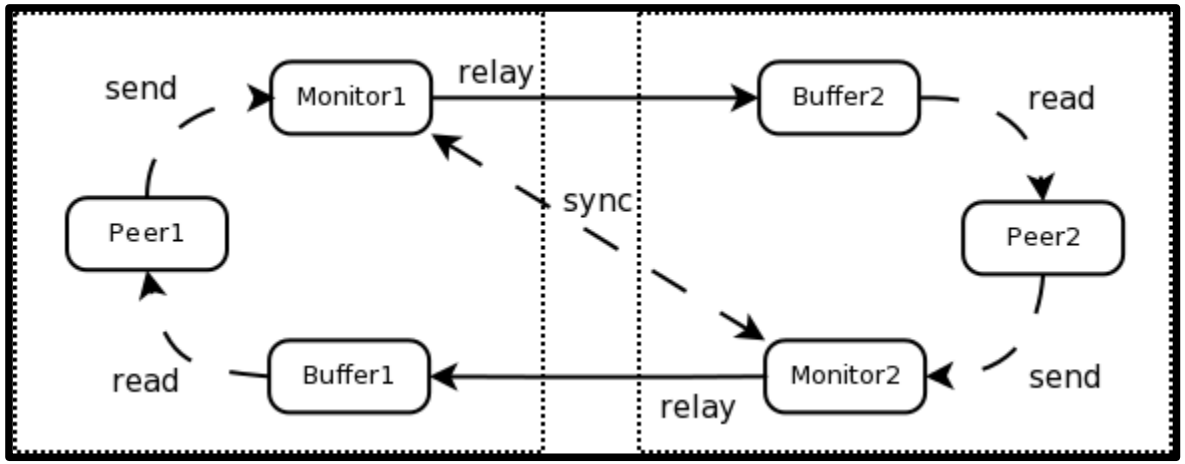
Original System



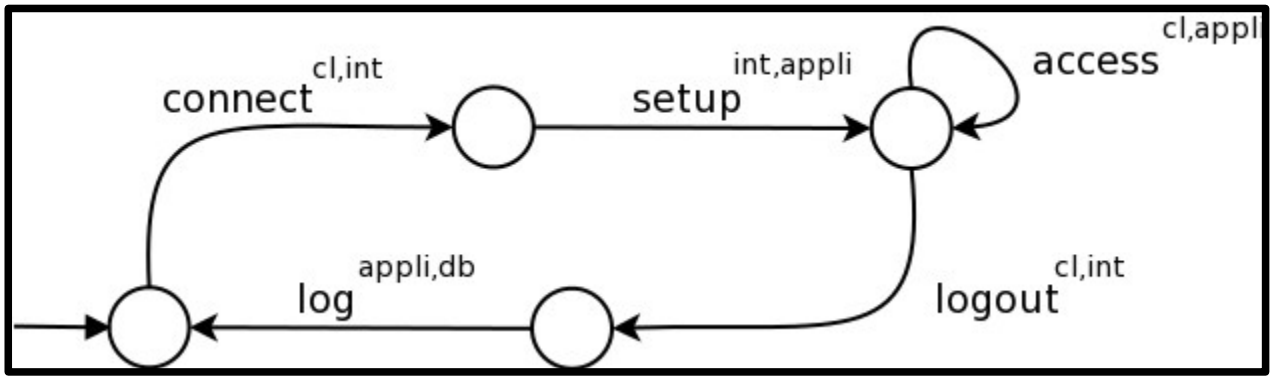
Monitored System



Local, distributed Monitors

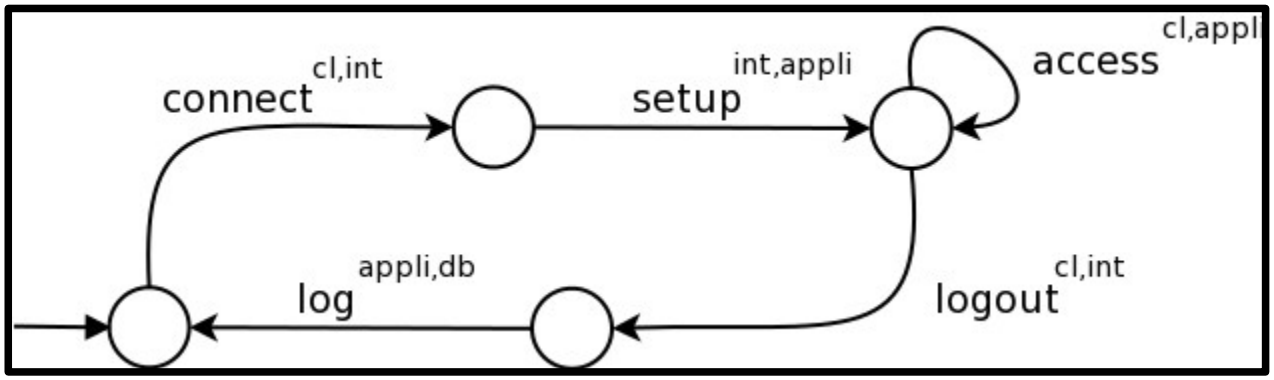


Extended Choreography

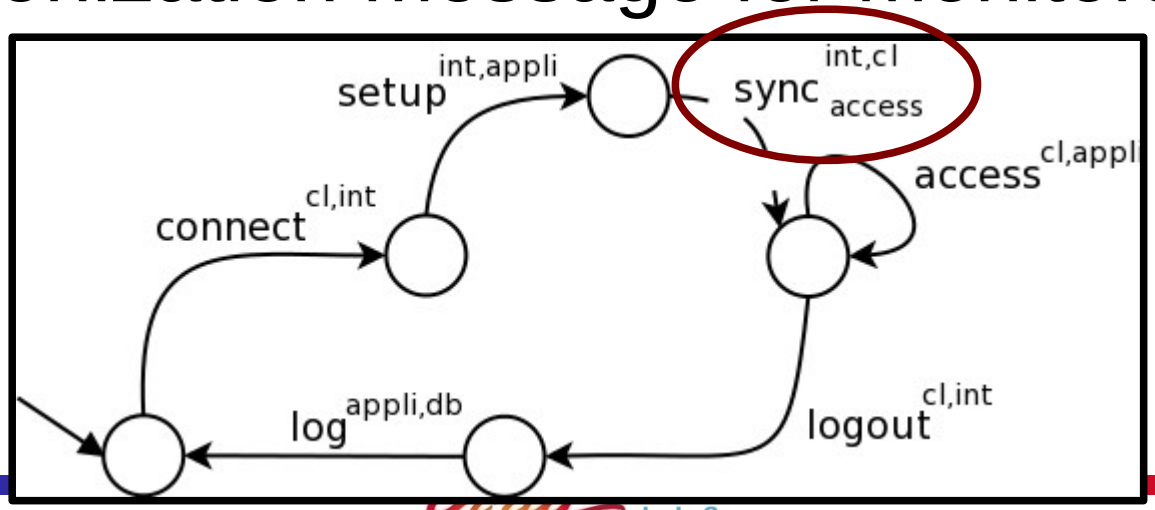


- Counterexample: `connect, access`

Extended Choreography



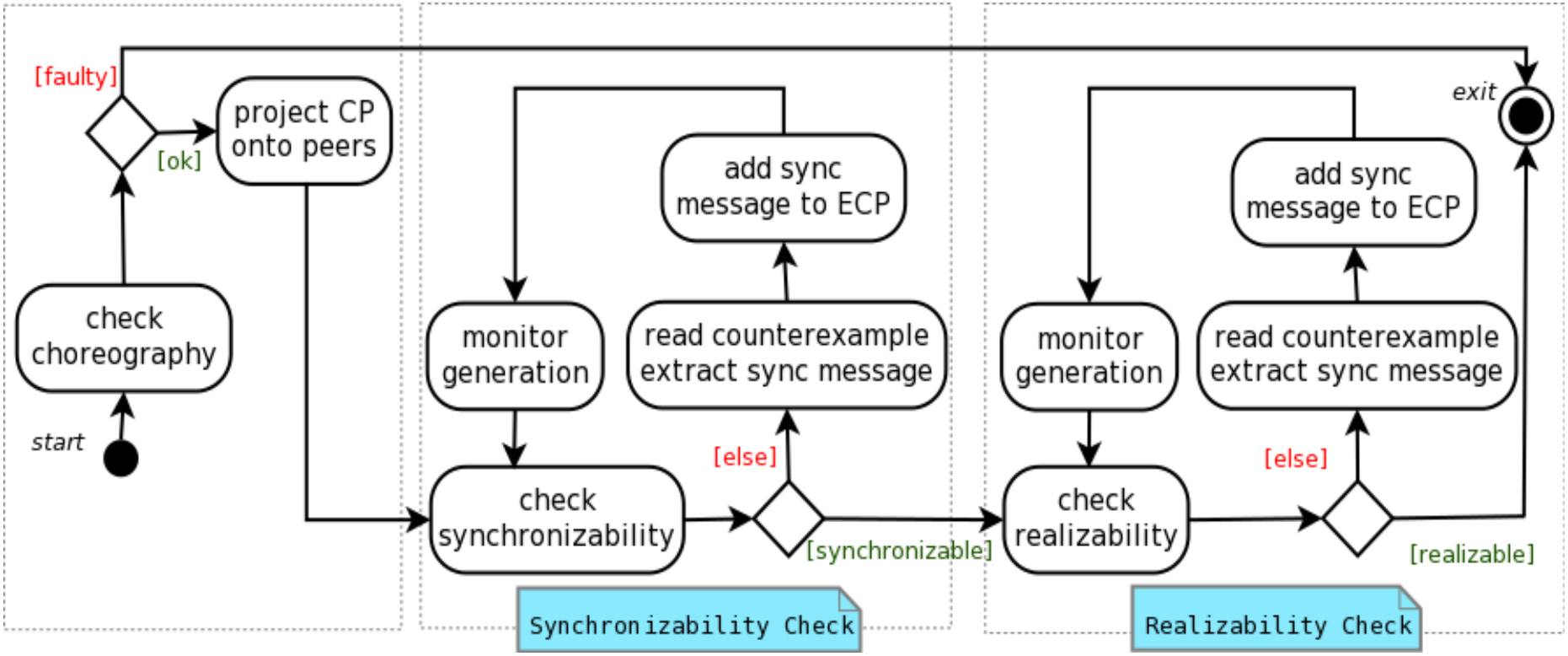
- Counterexample: `connect, access`
- Synchronization message for monitors



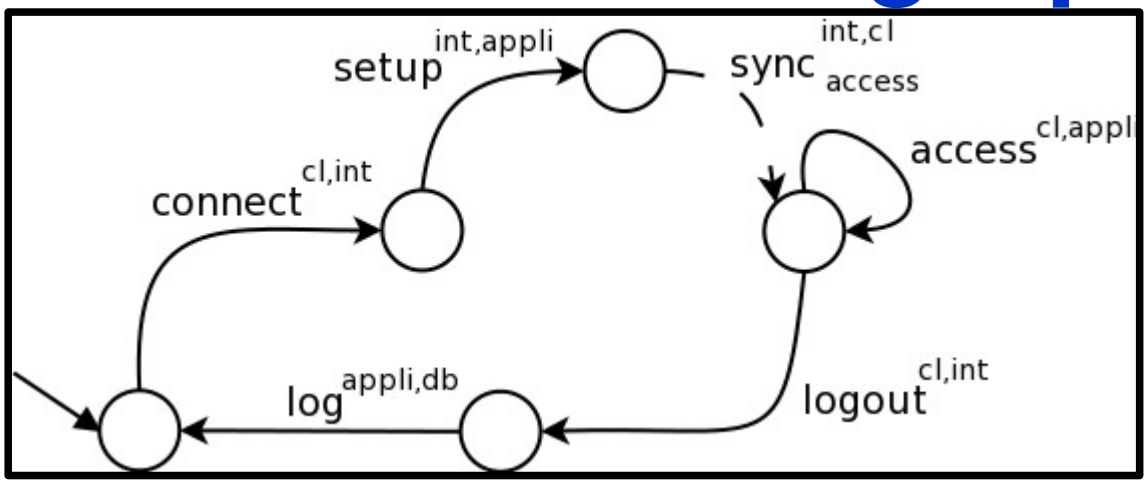
Monitor Generation

- Similar to projection
- Local reception via message renaming
- Most permissive construction (peer continues)

Iterative Approach

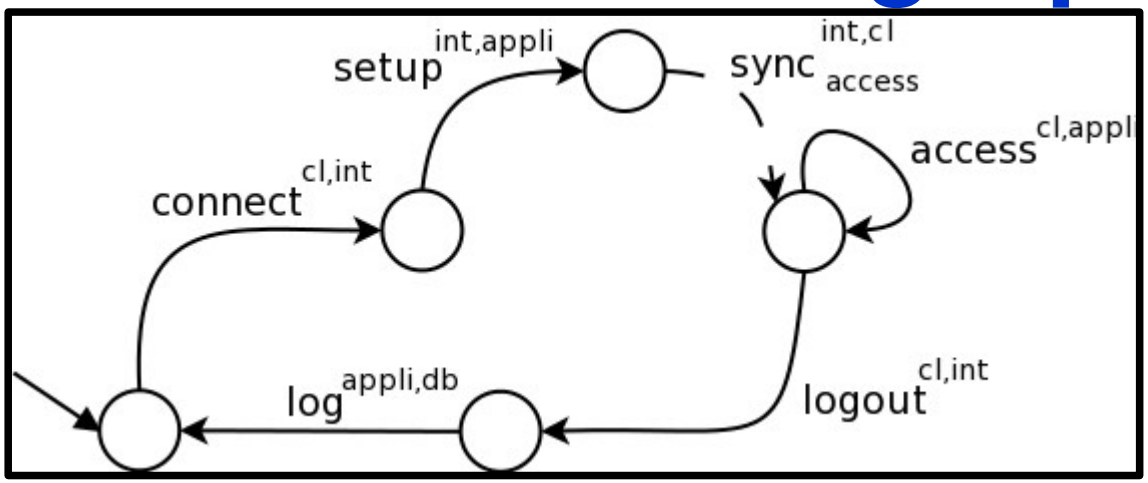


Extended Choreography



- Counterexample: `connect, setup, log`

Extended Choreography



- Counterexample: `connect, setup, log`

Extended Choreography

- Counterexample: `connect, setup, logout, connect`

Repairability Results

example	|peers|	$\|T\|/\|S\|$	|sync|	parallel composition	time max / total
cp0121	3	12 / 8	0	355 / 931	- / 54s
cp0016	3	4 / 3	1	121 / 337	46s / 1m 31s
cp0063	4	5 / 4	3	337 / 988	58s / 3m 54s
cp0153	3	29 / 16	5	15,182 / 59,033	53s / 7m 03s
cp0031	7	11 / 11	6	158,741 / 853,559	5m 47s / 19m 31s
cp0032	9	11 / 12	5	105,598 / 856,617	25m 53s / 1h 25m 10s

Overview

- Choreography Introduction
- Verification of Choreographies
- Enforcing Realizability of Choreographies
- Summary

Summary

- Contributions:
  - Realizability enforcement, i.e., repairability using non-intrusive distributed monitors
  - Fully automatized, prototypical implementation
  - Minimal number of additional messages
  - Identifies all problematic messages

Summary

- Contributions:
  - Realizability enforcement, i.e., repairability using non-intrusive distributed monitors
  - Fully automatized, prototypical implementation
  - Minimal number of additional messages
  - Identifies all problematic messages
- Outlook
  - Larger set of repairable choreographies
  - Full support for higher level formalisms (e.g. Chor, BPEL4Chor, BPMN 2.0 choreographies)