Conformance of Distributed Systems

Maximilian Frey[1], Holger Schlingloff[2,3]

[1] O_2 (Germany) GmbH&Co OHG
Maximilian.Frey@o2.com

[2] Humbold-Universität zu Berlin, Institut für Informatik
[3] Fraunhofer Institut für Rechnerarchitektur und Softwaretechnik
Holger.Schlingloff@FIRST.FhG.de

Contents

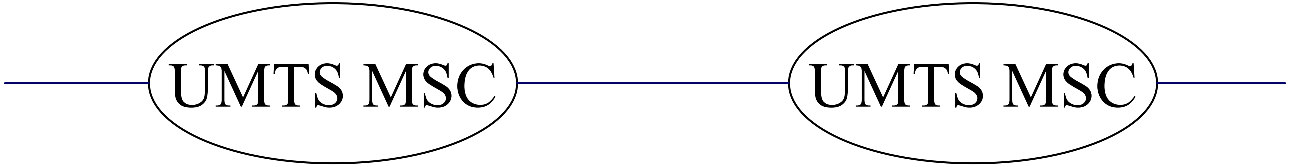
- Motivation: correctness of telecom systems
- A simple UMTS example
- Modelling with extended Petri nets
- Executions, simulations and conformances
- Model checking and test generation
- Summary and further work

Motivation

- Context: test generation from finite state machines
- Distributed systems and test cases
- Partial order approach to model parallelism
- Exponential blow-up while generating global states and test cases
- Conformance of distributed models (Petri nets)

Example

- Architecture:

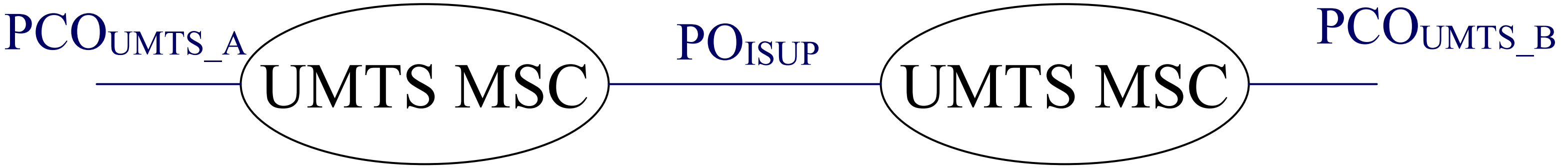


- UMTS layer to test: Call Control. Protocol is reduced to
  - 7 states: NULL, Call present, Call init, Conn req, Active, Release req, Release ind.
  - 5 protocol messages: setup, connect, connect ack, release, release complete
- ISUP protocol is reduced to
  - 4 protocol messages: IAM, CON, REL, RLC

Architecture

- PCOs can be observed and influenced externally
- POs can only be observed
- Input and output functions attached to events
- Asynchronous communication modelled in the net

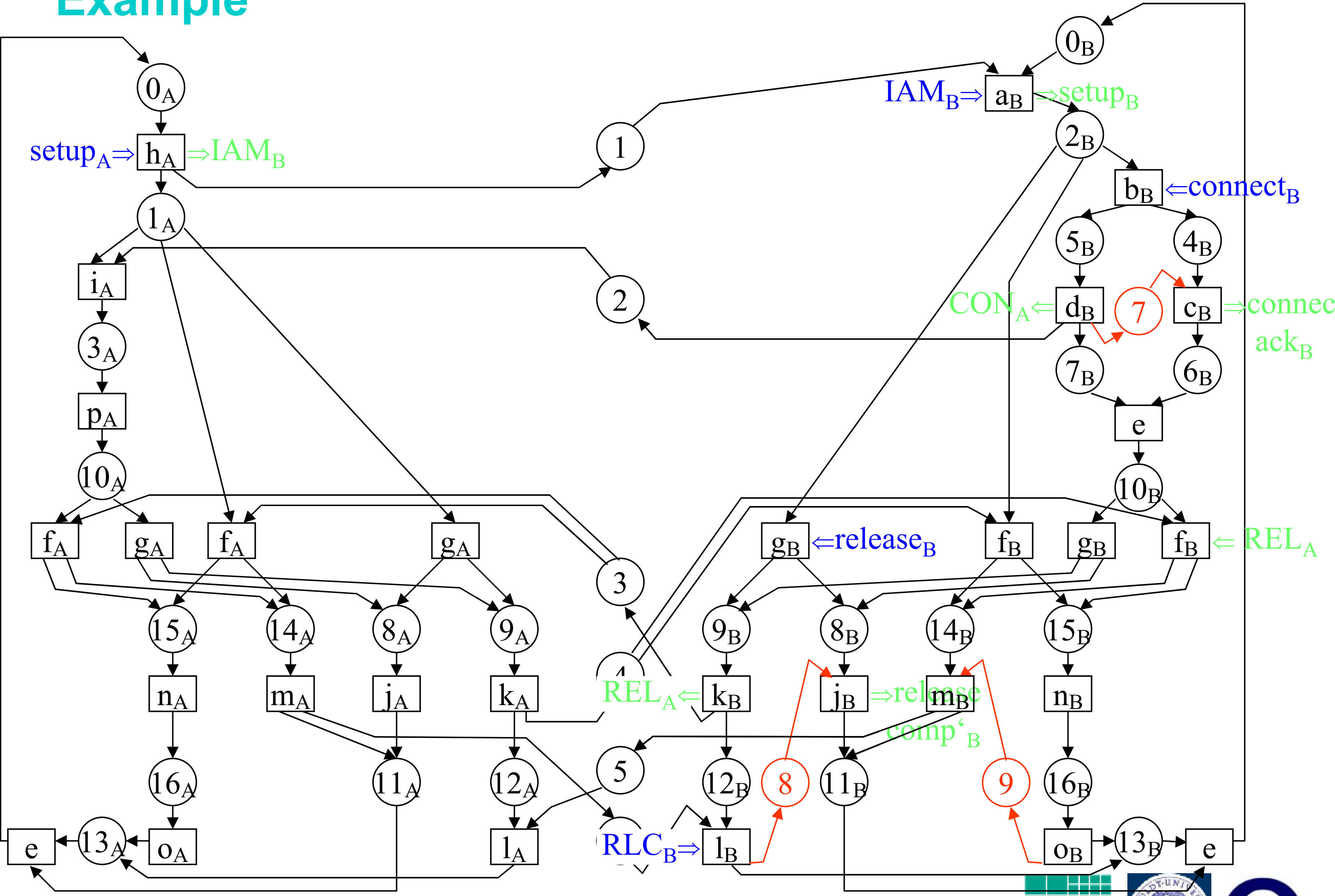
Example:


$\text{PCO}_{\text{UMTS\_A}}$
UMTS MSC
PO_{ISUP}
UMTS MSC
$\text{PCO}_{\text{UMTS\_B}}$

Modelling

- Petri nets: a widely used model to describe distributed systems
- Extensions to Petri nets:
  - Definitions of PCOs as sets of input and output symbols
  - Each event has at most one input and output symbol
    - Outputs and inputs at a PO are represented by edges
    - For each event with input at a PO there exists at least one event with output at the PO connected to it and vice versa
    - Inputs and outputs at PCOs are not represented by an edge
    - Alternatives are deterministic

Example



additional implementation design decisions

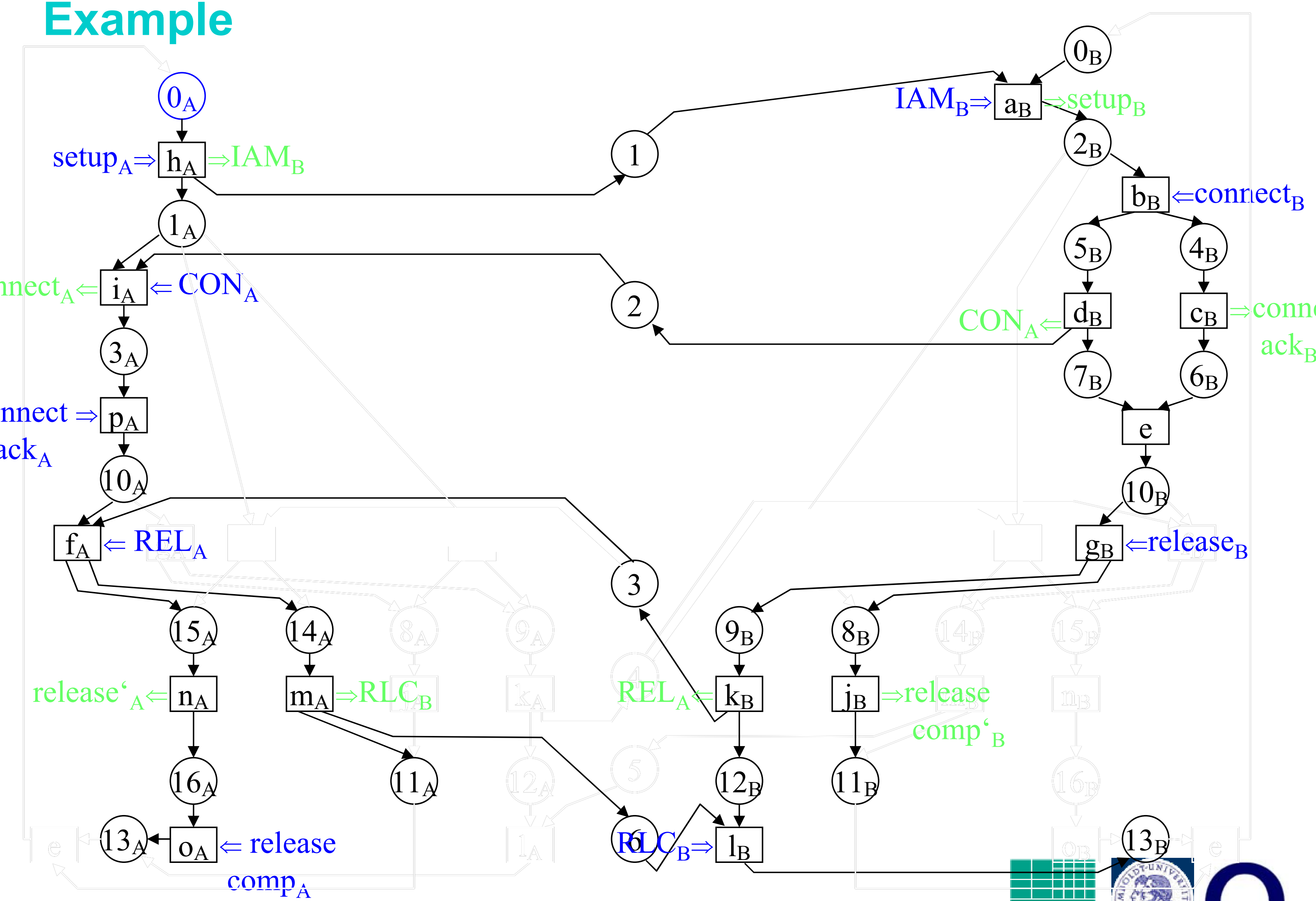
Conformance

- Conditions on the Petri net: loop-free, deterministic, strongly connected
- Similar to language containment of Mazurkiewicz traces
- Structural relation between nets
- Initial marking irrelevant
- Execution of a net starting at a place with an admissible input sequence

Execution of a net

- Constructed from a sequence of input symbols for each PCO
- Execution of a net starting at a condition according to a certain input:
  - (Extended) causal net: acyclic, no choice
  - Unfolding of the original net
  - All input symbols must be consumed

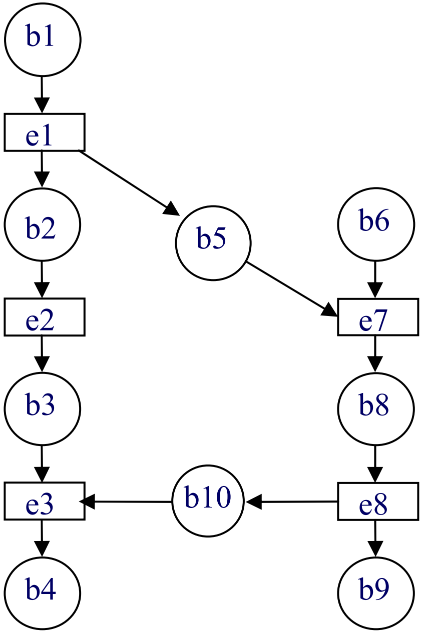
Example

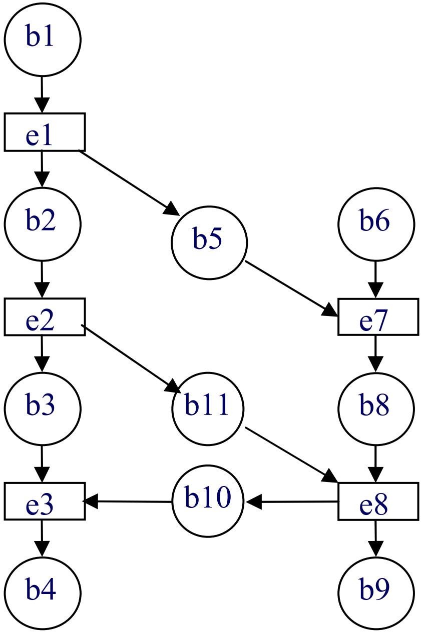


0_A
setup$_A$⇒
h_A
⇒IAM$_B$
1_A
nnect$_A$⇐
i_A
⇐ CON$_A$
3_A
nnect ⇒
ack$_A$
p_A
10_A
f_A
⇐ REL$_A$
15_A
14_A
release'$_A$⇐
n_A
m_A
⇒RLC$_B$
16_A
11_A
13_A
o_A
⇐ release
comp$_A$
1
2
3
6
0_B
IAM$_B$⇒
a_B
⇒setup$_B$
2_B
b_B
⇐connect$_B$
5_B
4_B
CON$_A$⇐
d_B
c_B
⇒conn
ack$_B$
7_B
6_B
e
10_B
g_B
⇐release$_B$
9_B
8_B
REL$_A$⇐
k_B
j_B
⇒release
comp'$_B$
12_B
11_B
RLC$_B$⇒
l_B
13_B

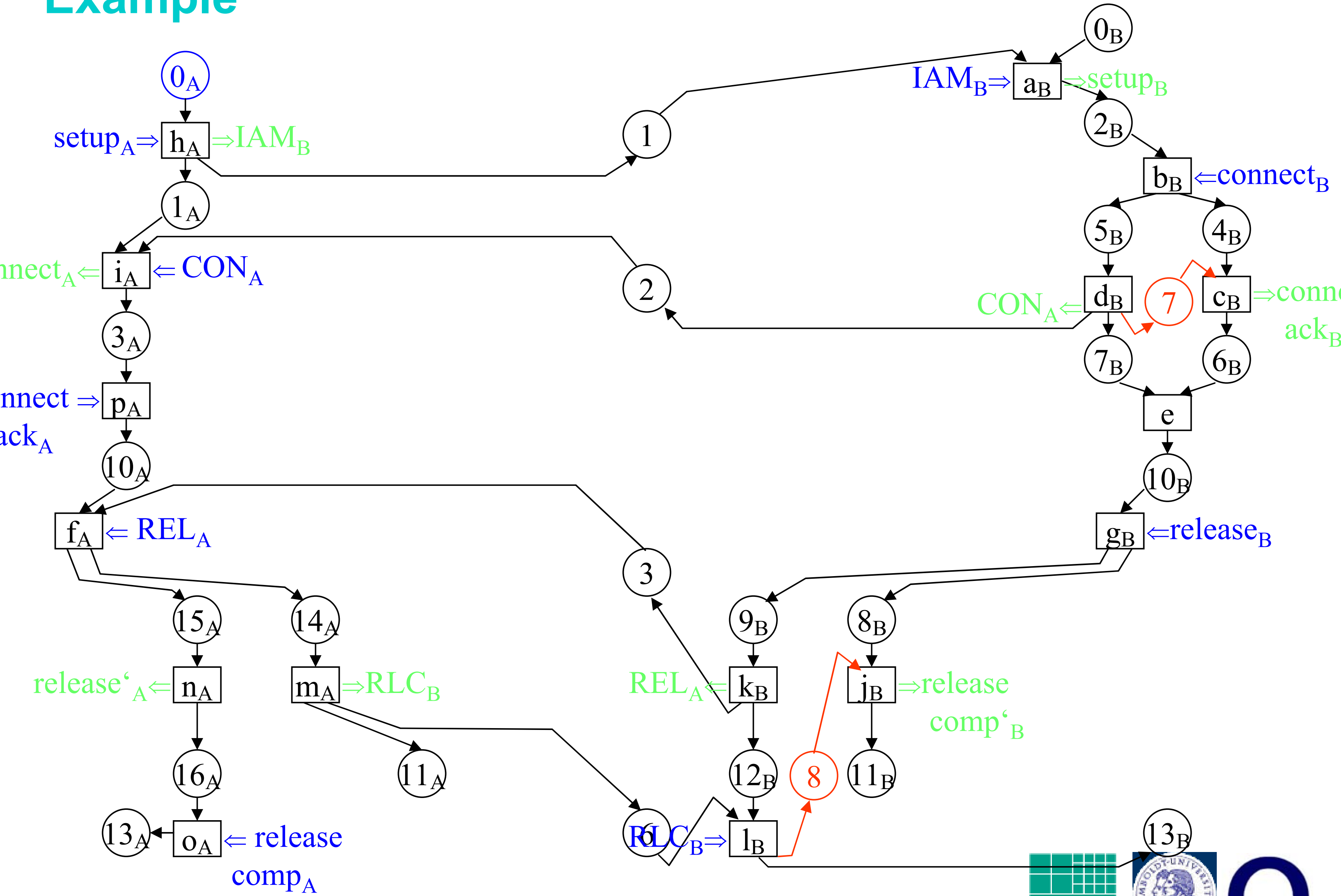
Weak and Strong Simulation

- Relation between two executions (of implementation and specification)
- Strong simulation: causal relation is preserved
- Weak simulation: additional causal dependencies





Example



implementation

Conformance Relation

- Condition b_I in an implementation is (*weakly) simulating* condition b_S in a specification if for all executions K_I and K_S of implementation and specification $K_I \in$ is (weakly) simulating K_S
- I *(weakly) conforms to* S if
  - $\forall\ b_S\ \exists\ b_I$: b_I is (weakly) simulating b_S, and vice versa
- Structural relation between specification and implementation
- No reference to an initial marking
- Implementation may have fewer parallelism, but no deadlocks, different I/O behaviours or actions, state faults etc.

Comparison

- Conformance on sequential extended Petri nets is equivalent to conformance on FSMs
- General case not equivalent

Model Checking and Test Generation

- Model Checking of conformance by partition refinement:
  - Put two conditions into the same equivalence class iff
    - they are already in the same class, and
    - the same inputs lead to the same outputs, and
    - liveness is preserved
- Generation of test cases from an execution of the specification:
  - Starting from a condition and admissible input sequences
  - Extend the initial marking stepwise as necessary

Conclusions

- Defined a new correctness notion based on partial order semantics
- Structural property
- Extension of Petri nets by I/O on PCOs and POs, resp.
- Comparison of weak and strong conformance
- Model checking and test generation algorithms

Further work:

- Compare to other correctness definitions and formalisms
- Implementation