



G E B I T Solutions

Die Experten für Java, Anwendungsentwicklung und Requirements Engineering

Combining Sequences and State Machines to Build Complex Test Cases

Dehla Sokenou, GEBIT Solutions

Stephan Weißleder, Humboldt-Universität zu Berlin



Agenda

- Motivation
- State Machines and Sequence Diagrams
- Related Work
- Case Study
- Combination of State Machines and Sequence Diagrams
- Coverage Criteria
- Conclusion and Outlook

Motivation

- Main problem: Test case selection
- Pragmatic approach: Concentrate on **typical sequences** as modeled with UML sequence diagrams
- Problems:
 - Incompleteness, only partly described behavior
 - No state information
 - What are expected values/states?
- Solution: **Combine** sequences with state machines

Testing Based on Sequences

- Advantages
 - Typical behavior, important behavior
 - Object interaction
 - Easy definition of test end criteria, e.g. all sequences
 - Traceability from test cases to specification
- Drawbacks
 - No initial states of participating objects
 - Incomplete specification
 - No test oracle can be derived
 - For each test case, initialization of SUT necessary

Testing Based on State Machines

- Advantages
 - Complete behavior description
 - Initial state is specified
 - State information
 - Easy test oracle derivation
- Drawbacks
 - Definition of test end criteria, test case derivation can result in infinite test cases
 - No difference between important and less important behavior
 - No direct traceability

Related Work

- Sequences with additional initial state and test oracles derivation
 - Contracts
 - State machines
- State machines with additional information about test end and transition selection
 - Coverage criteria
 - Transition selection, e.g. by defining probabilities

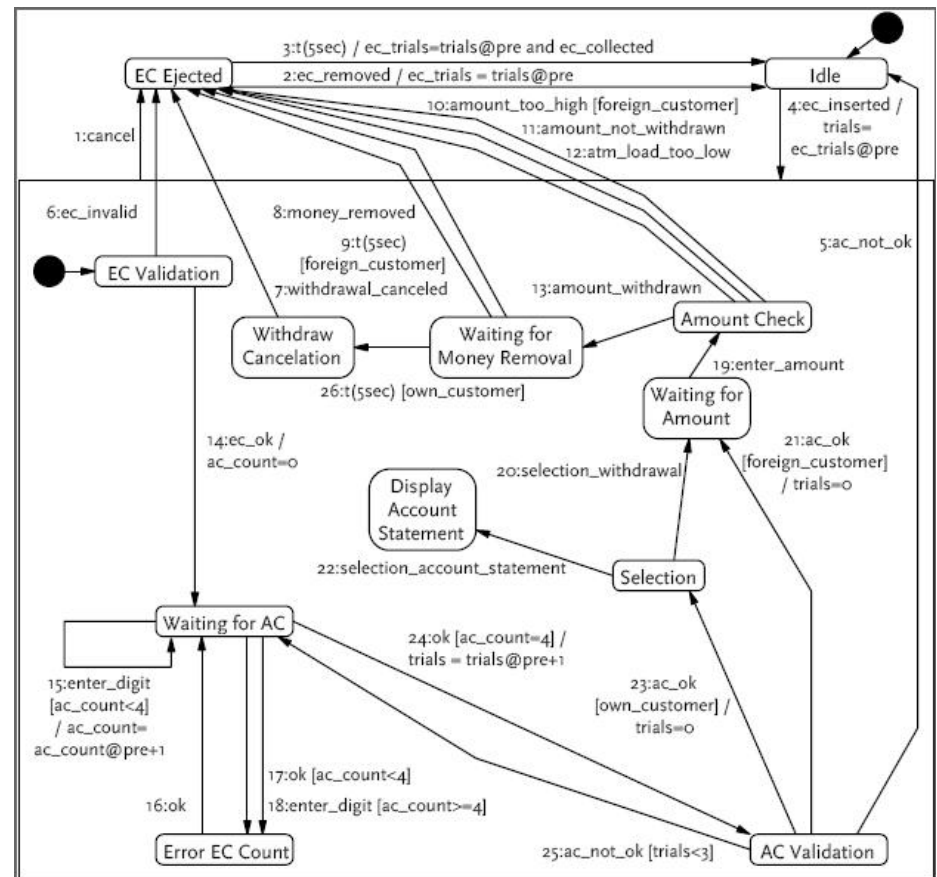
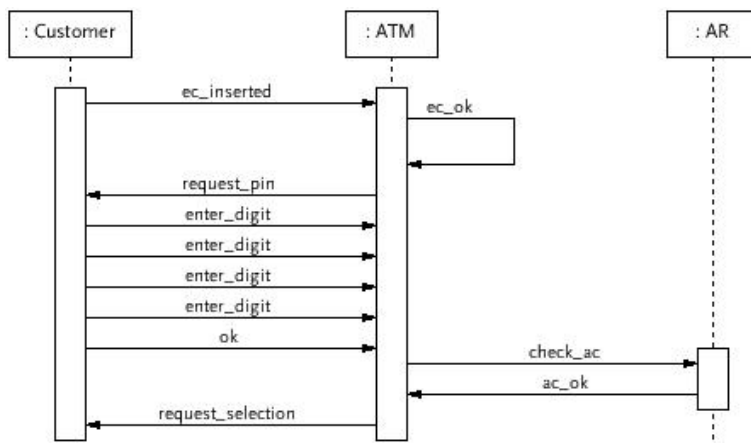
Case Study

- Automatic Teller Machine
 - Original specification with sequences
 - Additional state machine specification for test case derivation
 - Systems: ATMs, ATM central, banks (own, foreign), AR

Combination of Sequences and State Machines

Step 1: Matching Transition Sequences

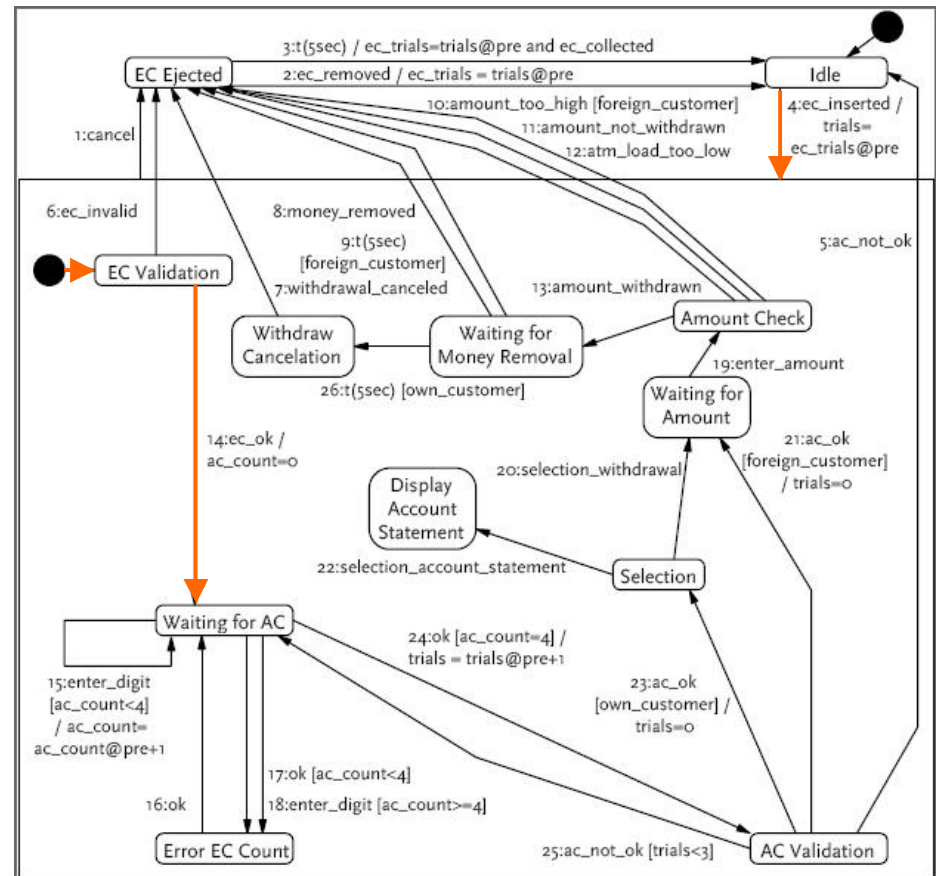
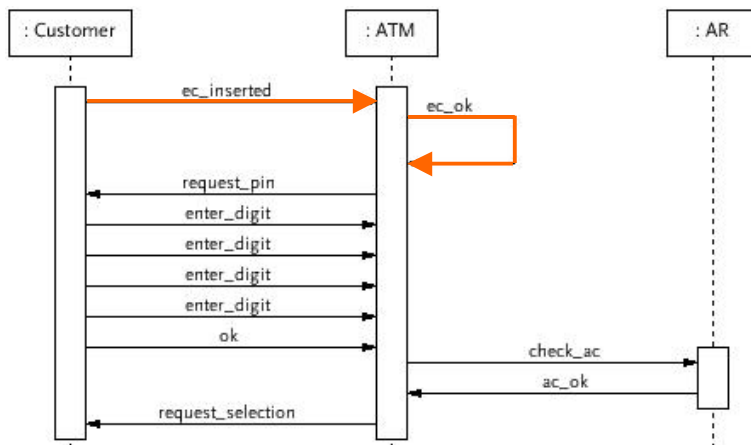
- Retracing sequences in state machines
 - Find matching transition sequences using incoming events in sequence diagram



Combination of Sequences and State Machines

Step 1: Matching Transition Sequences

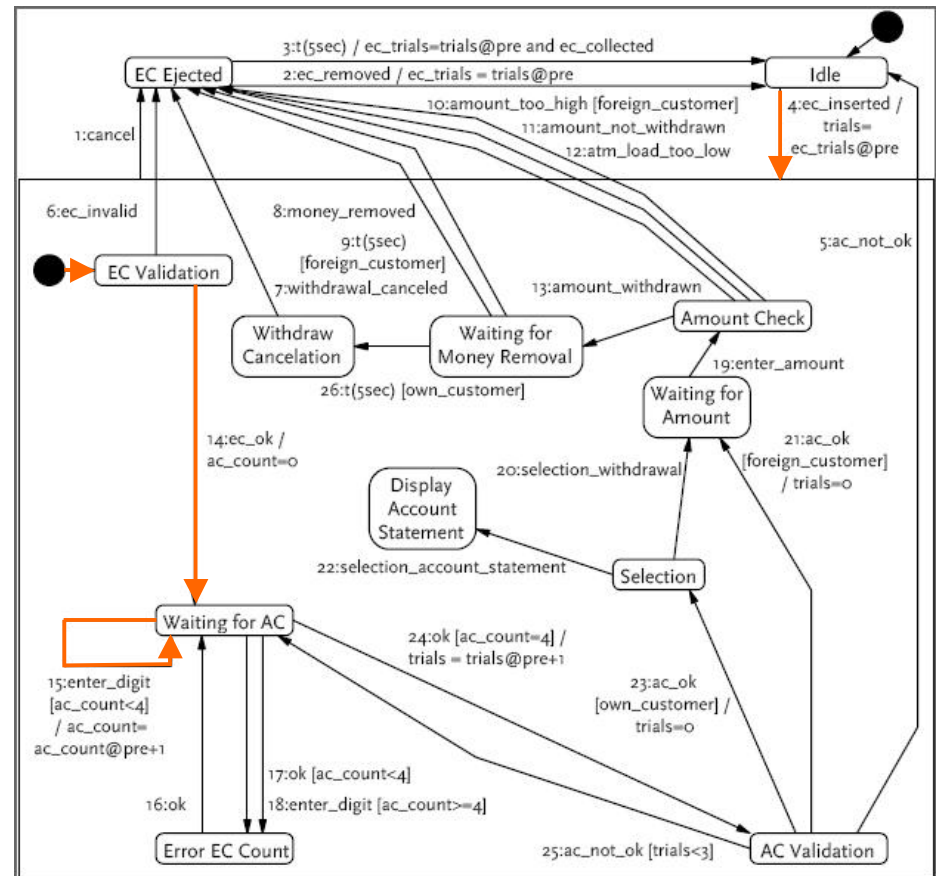
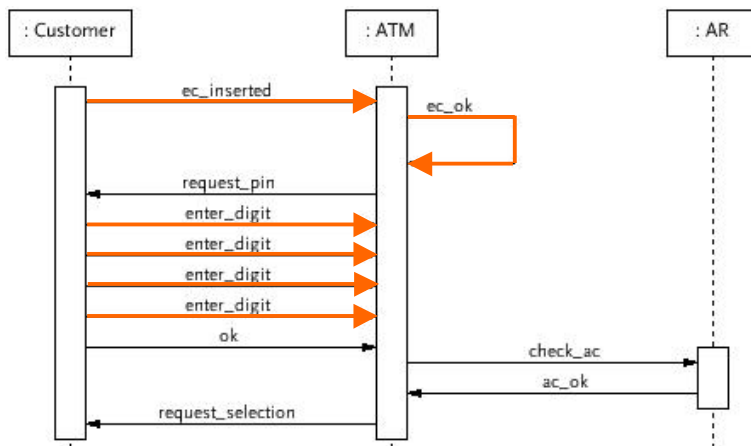
- Retracing sequences in state machines
 - Find matching transition sequences using incoming events in sequence diagram



Combination of Sequences and State Machines

Step 1: Matching Transition Sequences

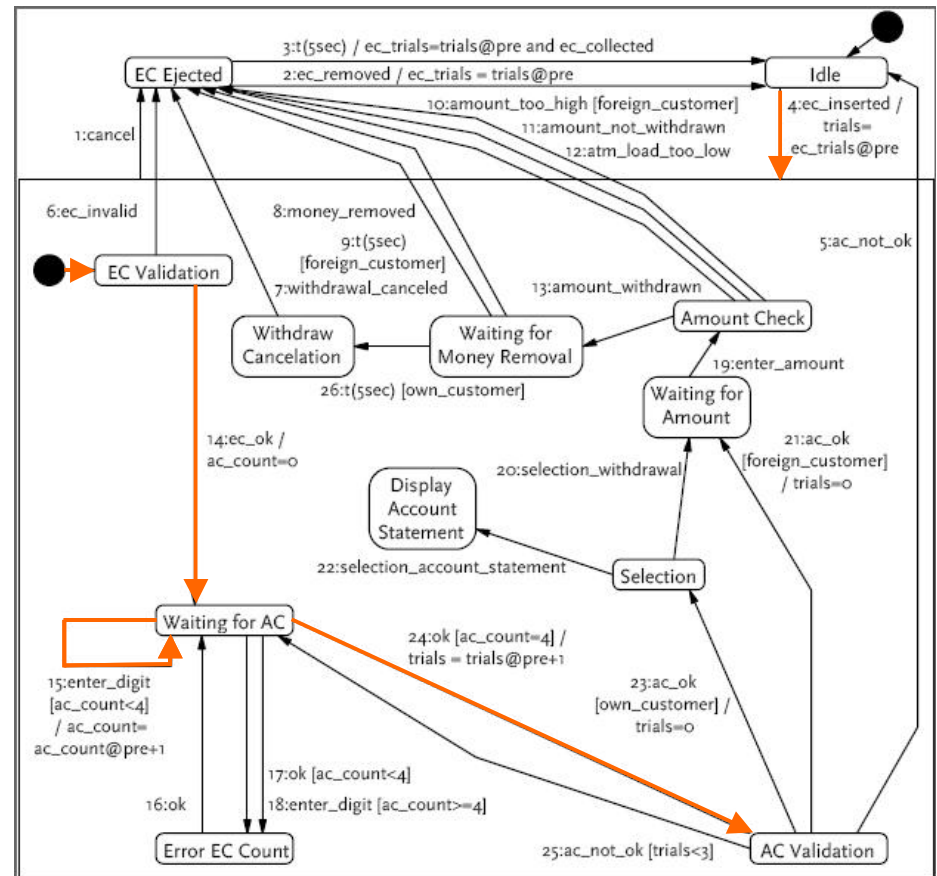
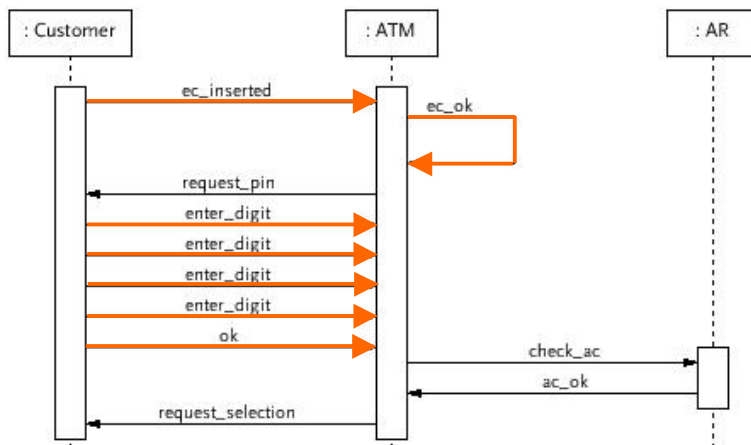
- Retracing sequences in state machines
 - Find matching transition sequences using incoming events in sequence diagram



Combination of Sequences and State Machines

Step 1: Matching Transition Sequences

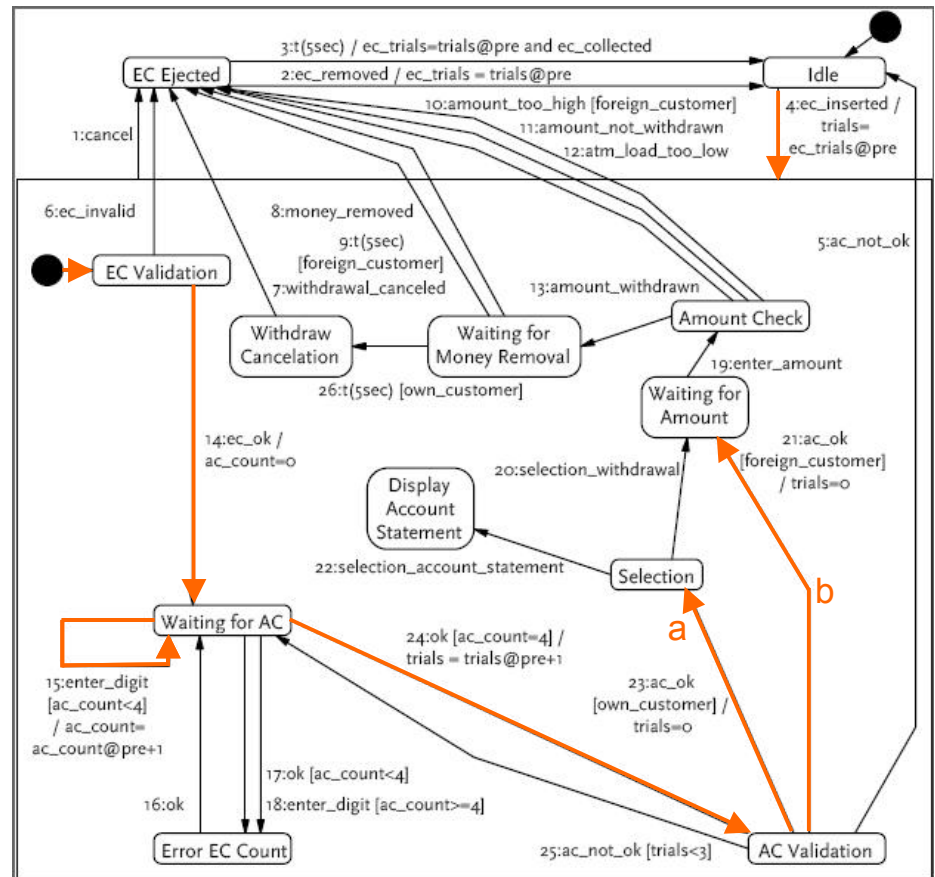
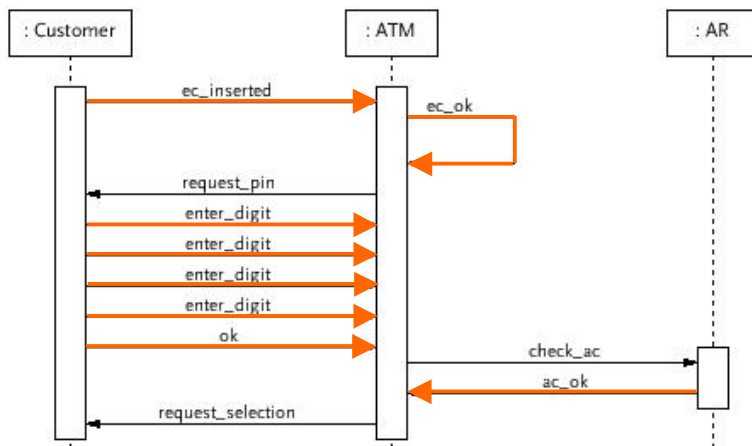
- Retracing sequences in state machines
 - Find matching transition sequences using incoming events in sequence diagram



Combination of Sequences and State Machines

Step 1: Matching Transition Sequences

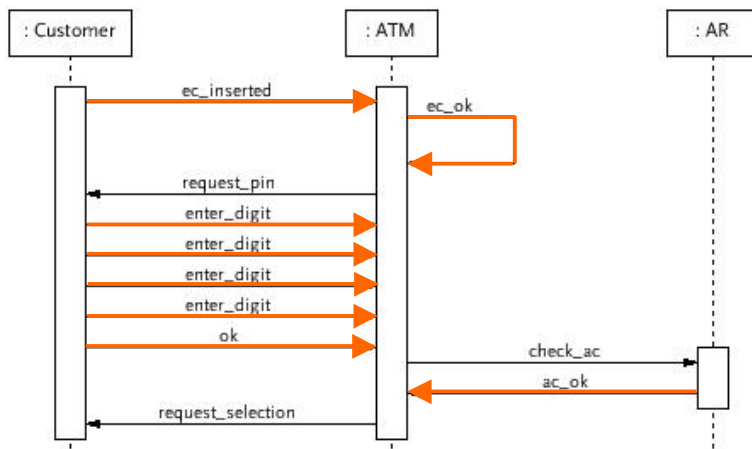
- Retracing sequences in state machines
 - Find matching transition sequences using incoming events in sequence diagram



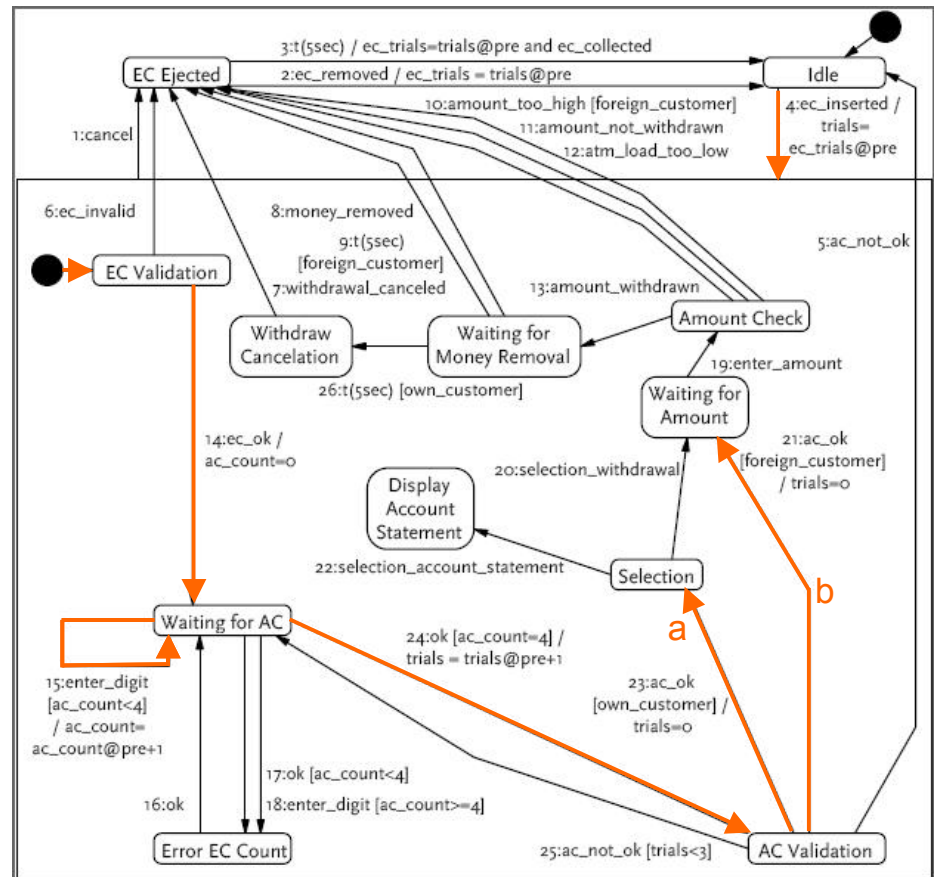
Combination of Sequences and State Machines

Step 1: Matching Transition Sequences

- Retracing sequences in state machines
 - Find matching transition sequences using incoming events in sequence diagram



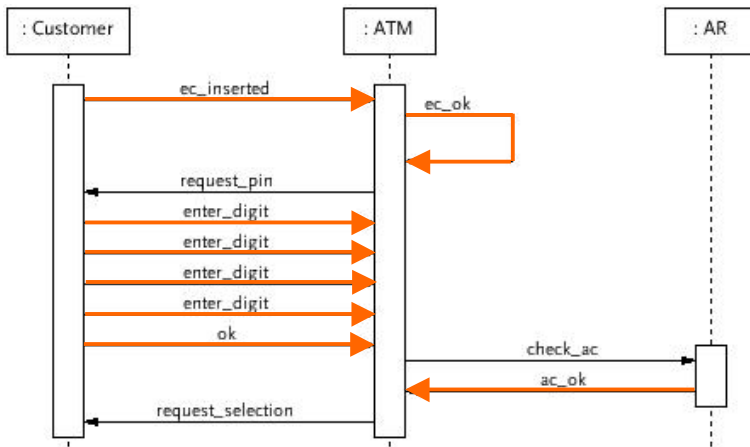
Transition sequences:



Combination of Sequences and State Machines

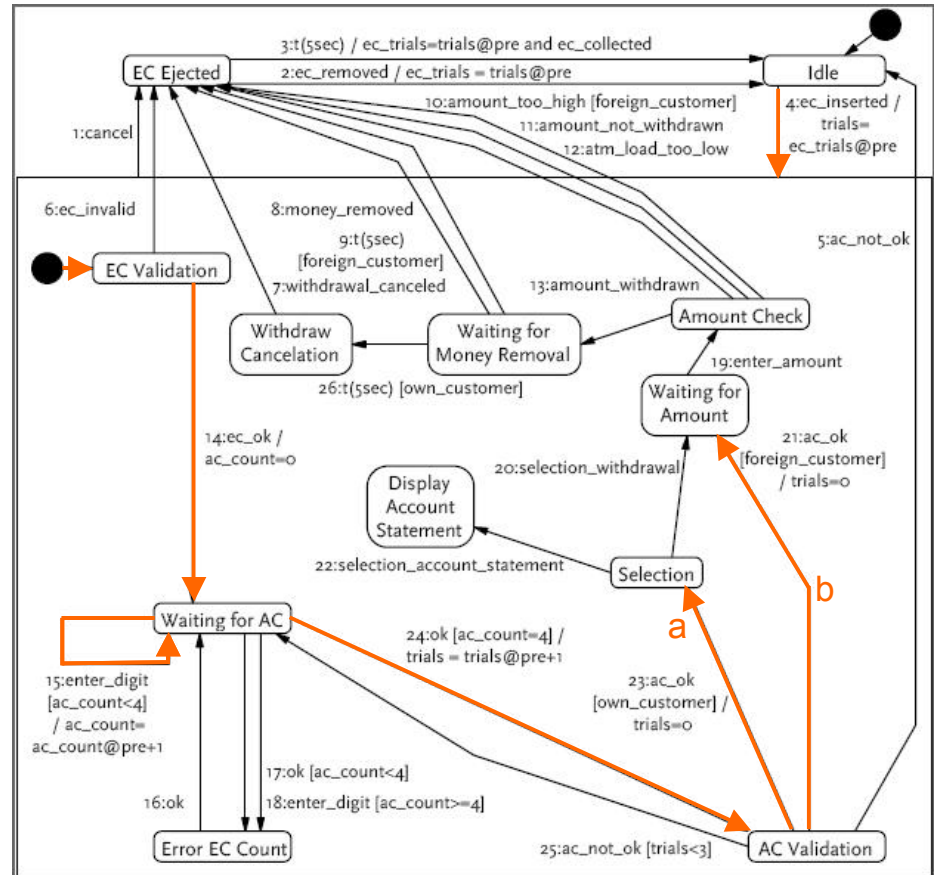
Step 1: Matching Transition Sequences

- Retracing sequences in state machines
 - Find matching transition sequences using incoming events in sequence diagram



Transition sequences:

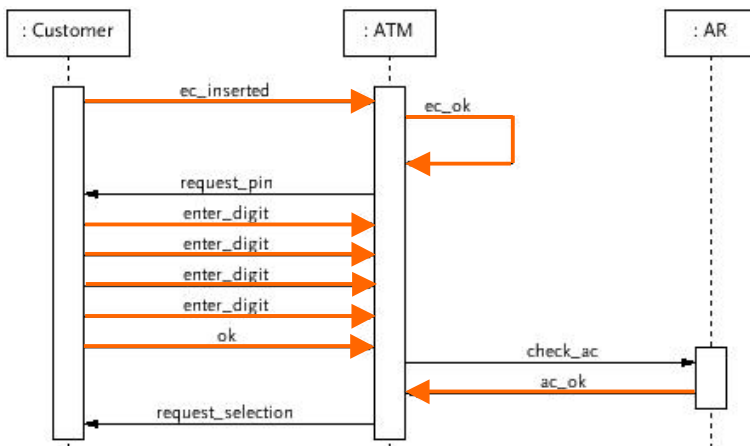
- 4, 14, 15, 15, 15, 15, 24, 23



Combination of Sequences and State Machines

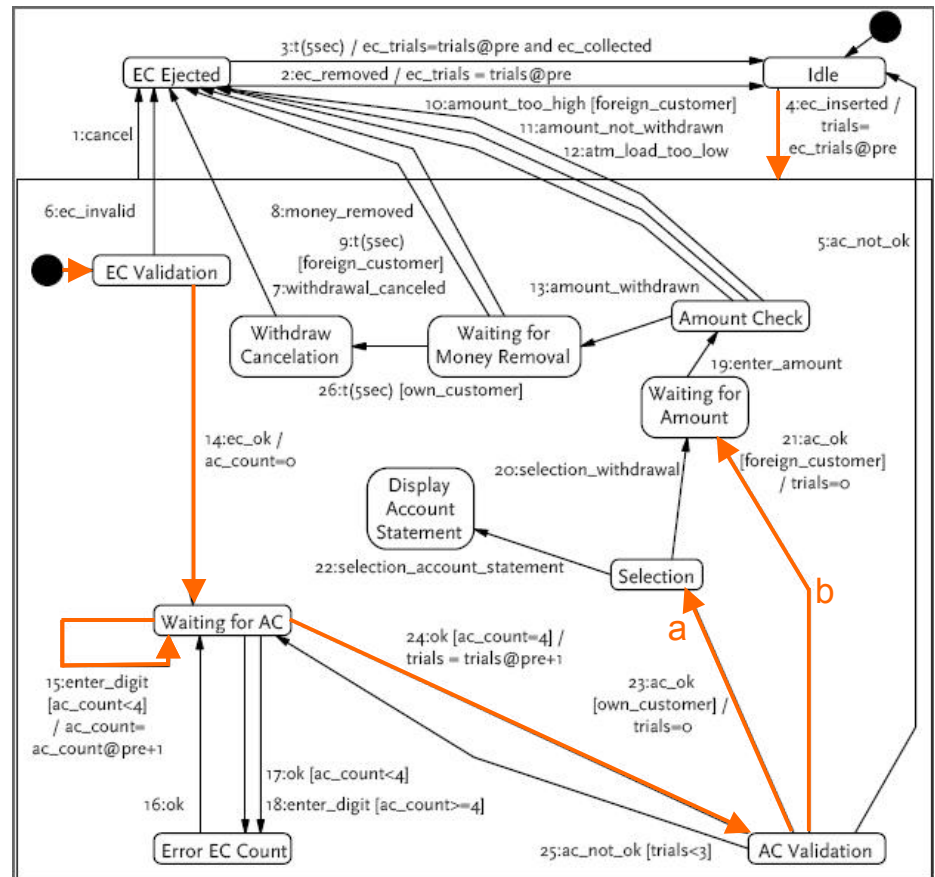
Step 1: Matching Transition Sequences

- Retracing sequences in state machines
 - Find matching transition sequences using incoming events in sequence diagram



Transition sequences:

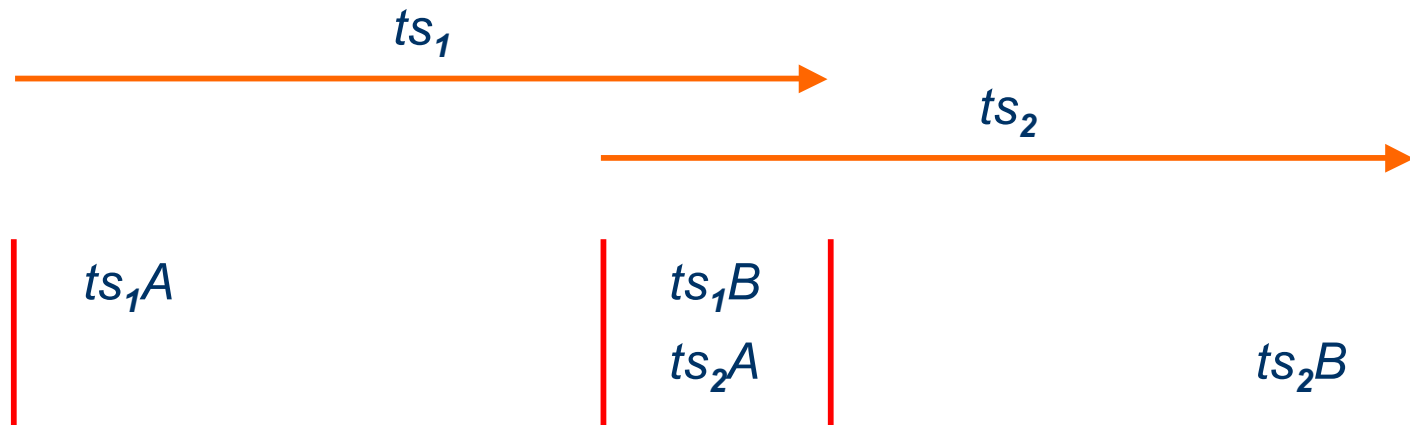
- 4, 14, 15, 15, 15, 15, 24, 23
- 4, 14, 15, 15, 15, 15, 24, 21



Combination of Sequences and State Machines

Step 2: Transition Sequences Overlap

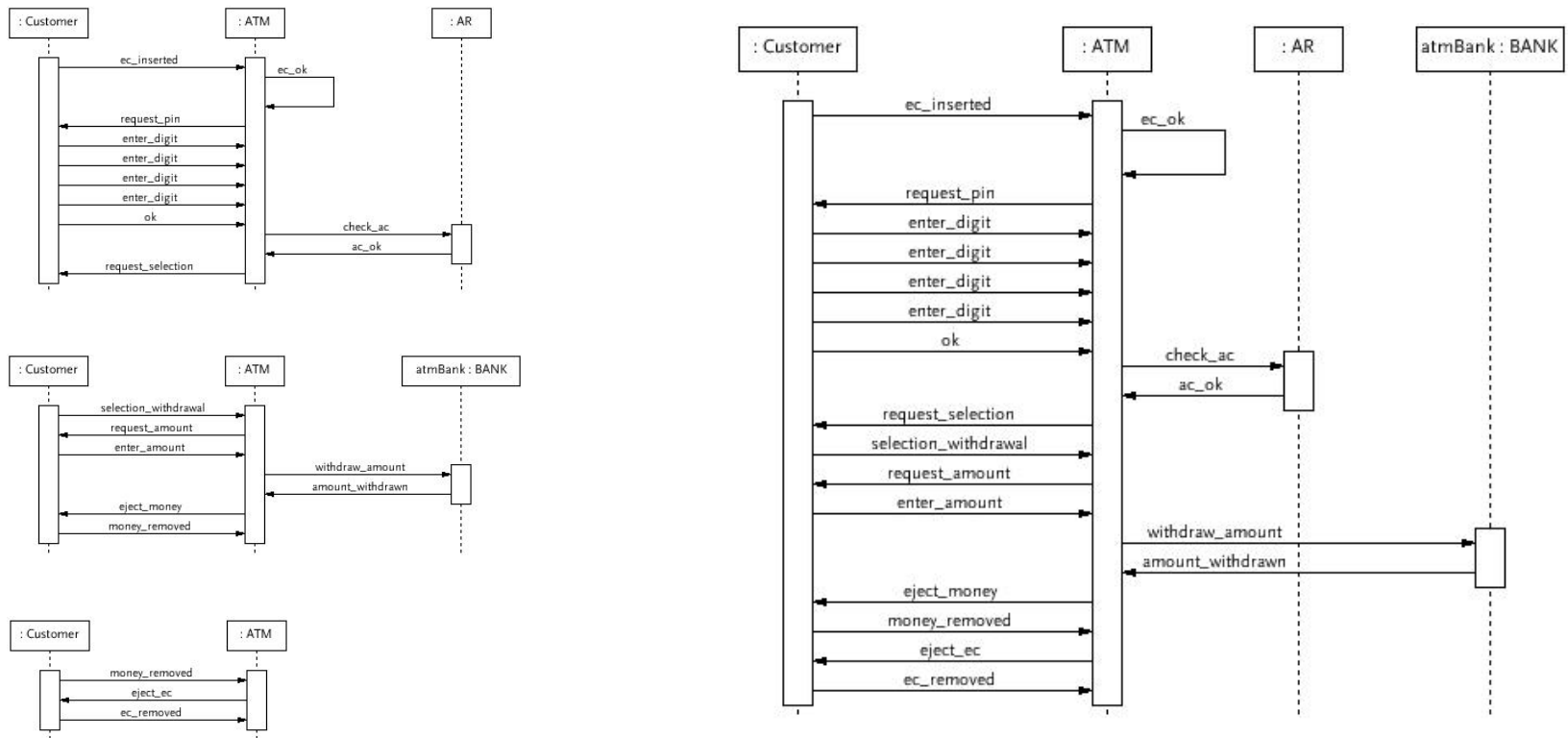
- A sequence s_2 starts with same transition sequence ts_2 as another sequence s_1 ends with (ts_1) → Transition sequence overlap
 - Sequence diagrams can be concatenated
 - Minimum overlap: 1 transition
 - Maximum overlap: $\min(\# \text{ transitions}(ts_1), \# \text{ transitions}(ts_2)) - 1$
 - Maximum overlap must be validated in future case studies



Combination of Sequences and State Machines

Step 3: Combined Transition Sequences

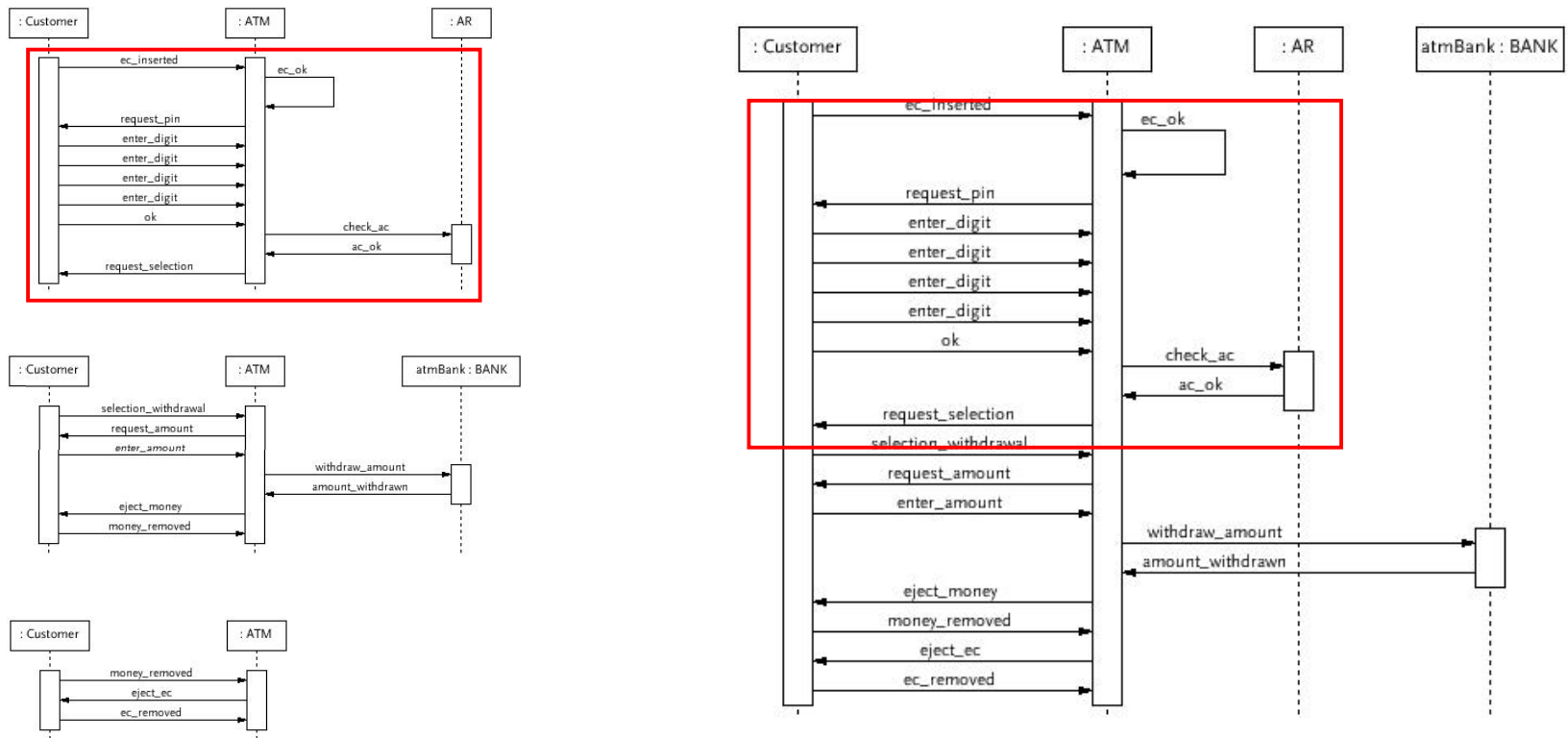
- Transition sequences are combined using overlap



Combination of Sequences and State Machines

Step 3: Combined Transition Sequences

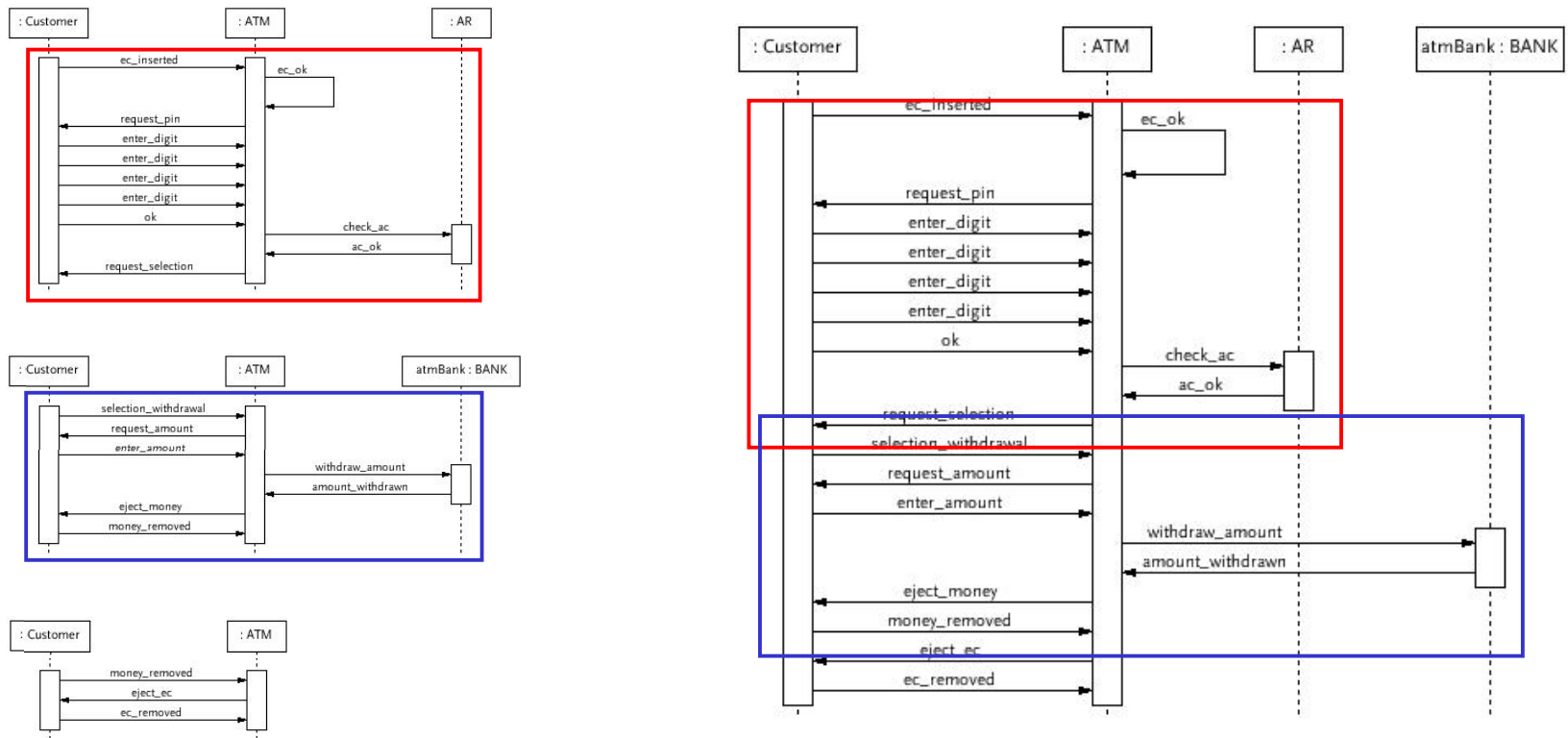
- Transition sequences are combined using overlap



Combination of Sequences and State Machines

Step 3: Combined Transition Sequences

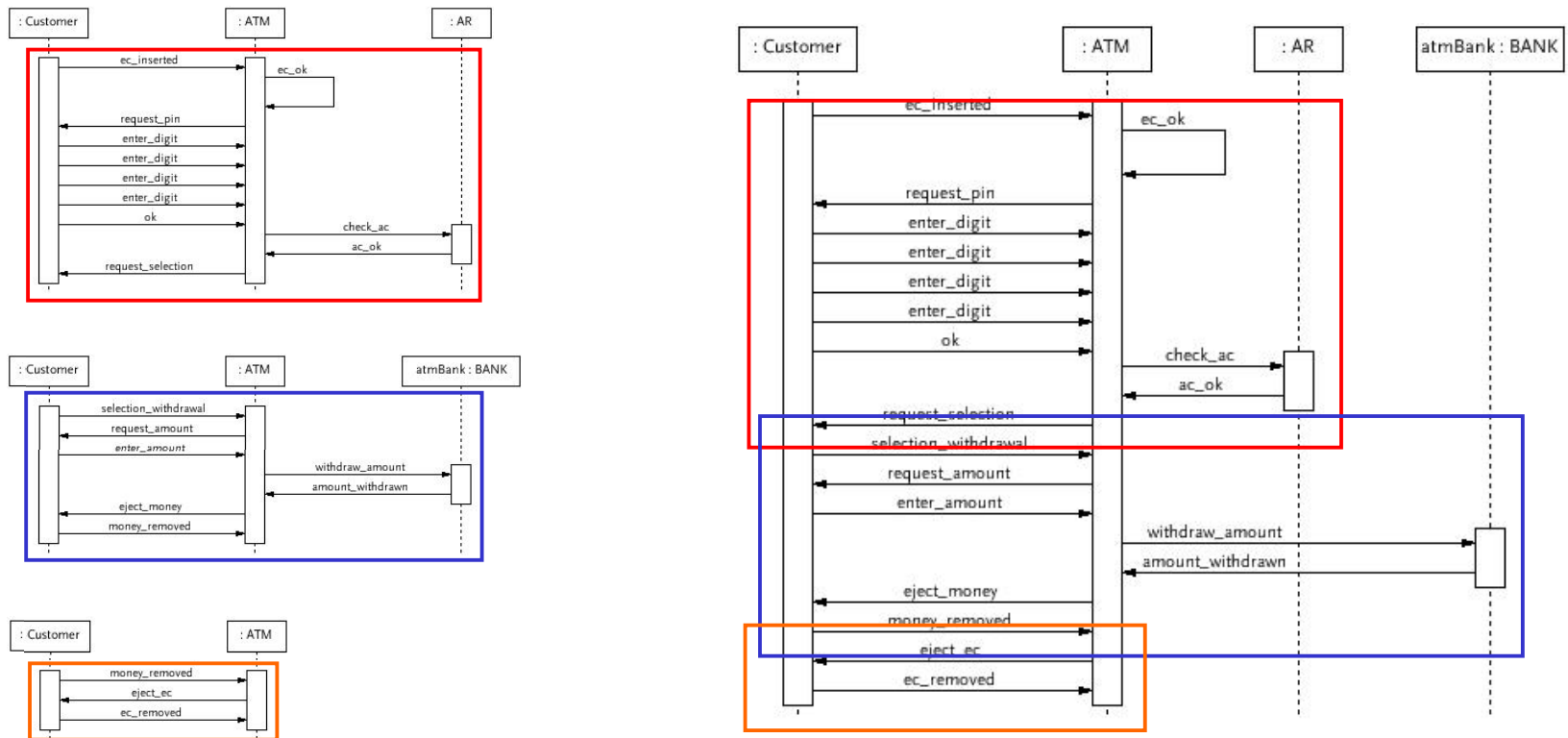
- Transition sequences are combined using overlap



Combination of Sequences and State Machines

Step 3: Combined Transition Sequences

- Transition sequences are combined using overlap



Coverage Criteria

- We propose new coverage criteria for combination of sequences diagrams and state machines
 - *All-context-sequences*: Coverage of matching transition sequences
 - *All-sequences-pairs*: Concatenation of all pairs of sequences
 - *All-n-sequences*: Concatenation of all n-tupels of sequences
 - *All-sequence-paths*: All sequences of concatenation of any length

Summary

- Test cases are transition sequences
 - Test cases are derived from sequence diagrams where the sequence is used as main test input
 - State machines are used to retrace sequences from sequence diagrams, test cases will not necessarily cover complete state machine
 - A test case is a (combined) transition sequence
- Integration in ParTeG is on the way
 - At the moment, ParTeG's test cases generation is based on state machines using state machine coverage criteria
 - ParTeG already considers OCL guards in state machines
 - Sequence diagrams have to be integrated as well as coverage criteria for sequences / transition sequences

Discussion

- Combined transition sequences can reduce test effort
 - Instead of a high number of small test cases only a small set of more complex test cases is used
 - Special initialization sequences can often be avoided
 - Complex test cases can find errors that cannot be found using sequence diagrams alone, e.g. when concatenating sequences to new loops
- More complex test cases can increase effort of error finding
 - If a complex test case fails, some sequences are not executed anymore even if they will not fail when executed separately
 - Errors can be masked

Outlook

- Additional case studies are needed to clear open points
 - What is the maximum overlap to consider?
 - What is minimum number of complex test cases?
 - What is maximum number of sequences to consider for one complex test case?
- Evaluation of test suite derived using the proposed coverage criteria
- Tool support is needed

ParTeG

Partition Test Generator

<http://parteg.sourceforge.net/>