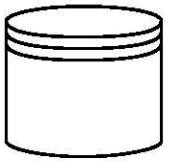
Execution Semantics for Data in BPMN

- Start semantics of an activity A with respect to external data (if available)
  - If A is (in terms of control flow) to be executed, input sets will be tested
  - In case that an InputSet has all data in the required conditions, the activity started with this set

Data Semantics in BPMN

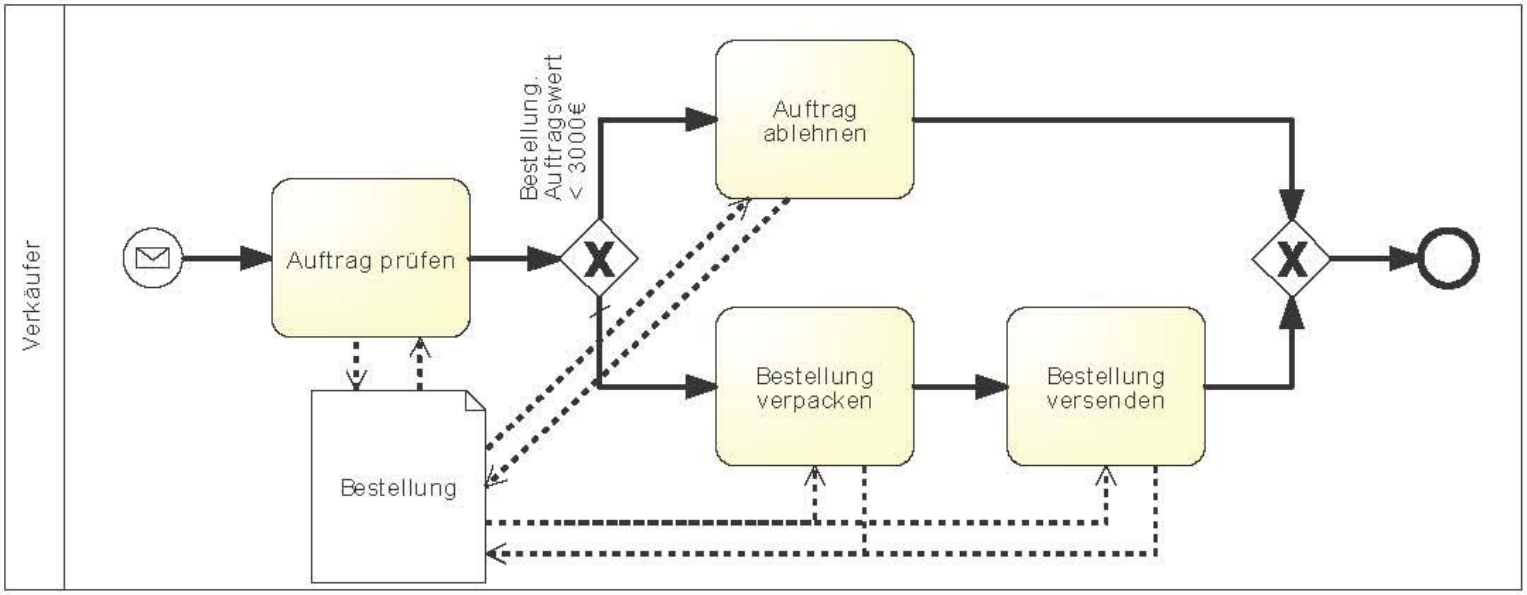
- A data object is bound to a process
  - Data objects are transient during process execution, but can be explicitly persisted, through writing to a data store
  - Terminating a process instance leads to the loss of not-persisted data objects



- Data Input
  - Input data can be "valid" or "not available"
    - Valid: Data objects exist at process start
  - Definable in BPMN:
    - Acitivities can only start with valid data input
    - Activities can start anyway

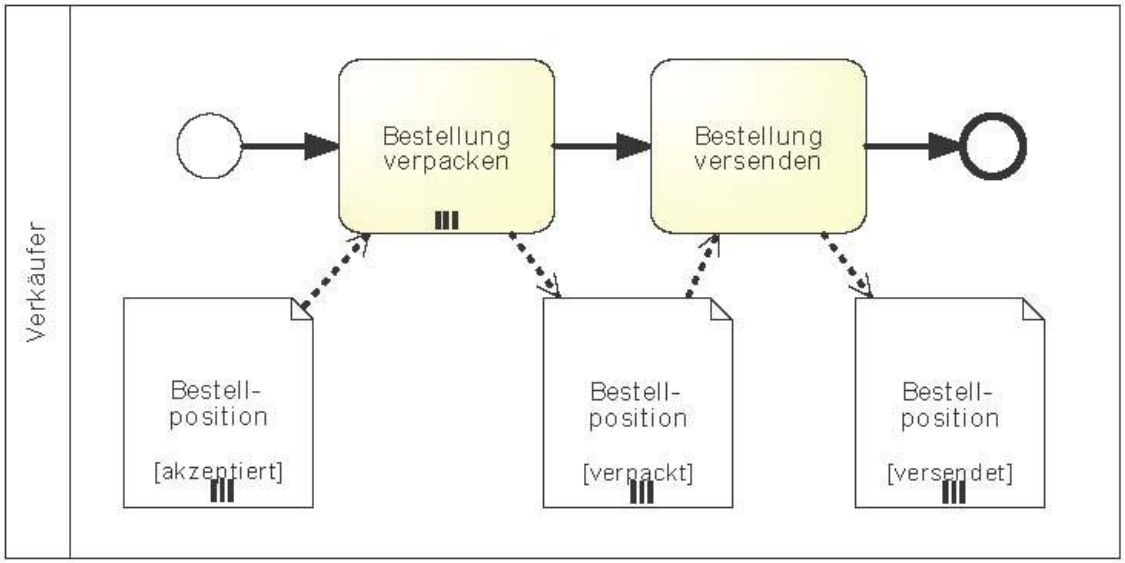
Data-based Decisions

- Decisions at XOR or OR gateways are based on information from data objects
  - Transaction amount of the request
  - Truth value of a specific attribute
- Evaluation of the decision parameters in BPMN
  - "Proposal": XMLSchema (XSD) and XPATH



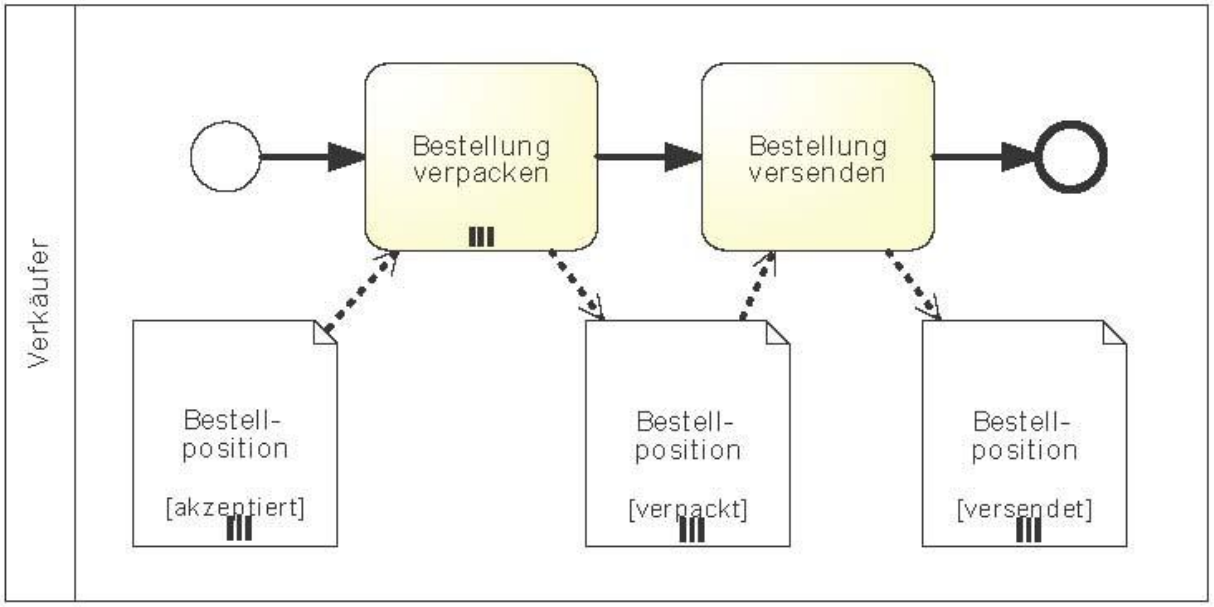
Multiple Instances

- Approach
  - Use of MI-activities in order to process data objects
- Process objects individually, concurrently
  - Data objects defined as a list, and MI activity
  - Each activity instance processes a data object of the list
- Example: "pack order"



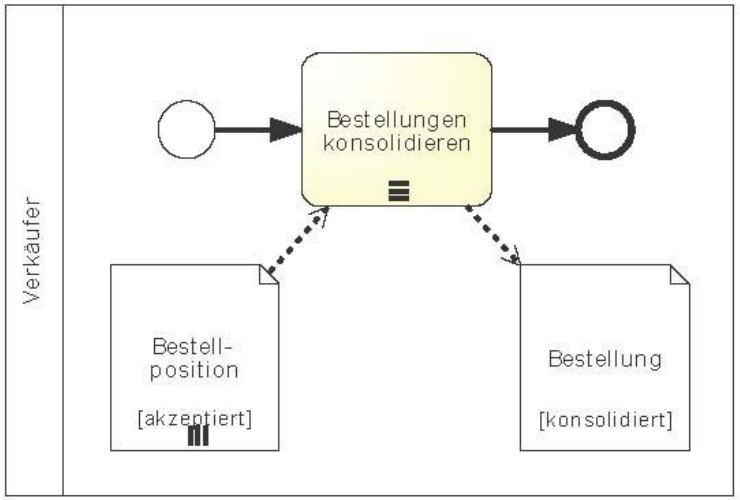
Multiple Instances

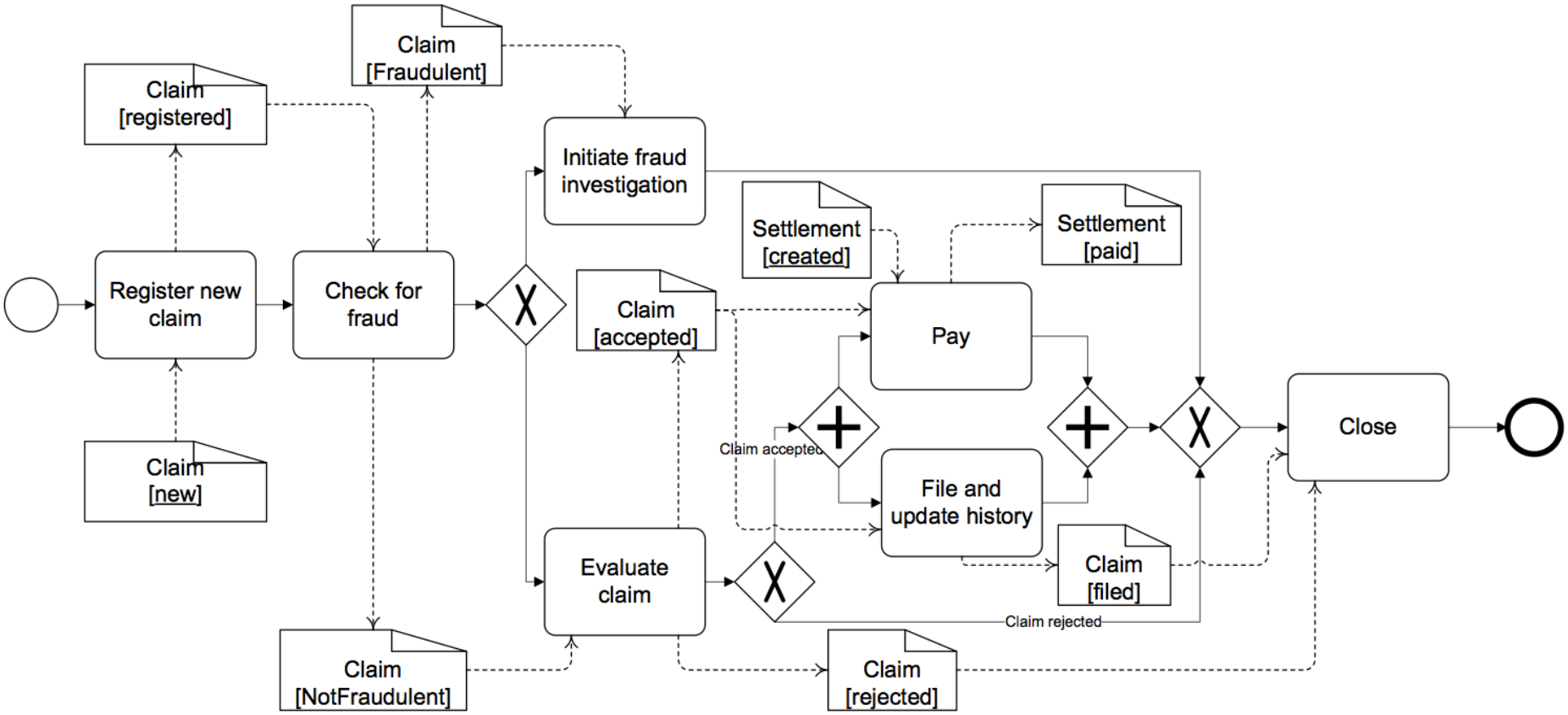
- Edit object list as a whole
  - Data object list is processed by an activity
- Example: "Send order"



Multiple Instances

- Process data objects separately, but sequentially
  - Data objects defined as a list, and MI activity
  - Each activity instance processes a data object of the list
    - Sequential processing
  - After completion of all activity instances there is a new, atomic data object
- Example: "Consolidate Order"




Claim [Fraudulent]
Claim [registered]
Initiate fraud investigation
Settlement [created]
Settlement [paid]
Register new claim
Check for fraud
Claim [accepted]
Pay
Claim [new]
Claim accepted
File and update history
Close
Evaluate claim
Claim [filed]
Claim rejected
Claim [NotFraudulent]
Claim [rejected]

Data Anomalies

- Too Restrictive Preconditions: This is the general case of data anomalies. It occurs when an activity, from the control flow only perspective, is ready to execute but the on the data side data object(s) are not in the expected state.
- Implicit Routing: Alternative branches can be taken in the process where each branch requires the data objects to be in certain states. However, while the branching condition does not impose these restrictions, they only become apparent as the activities have corresponding preconditions. In our example, this could occur when a claim is fraudulent and still the lower branch leading to “evaluate claim” could be chosen.

Data Anomalies

- Implicit Constraints on the Execution Order: The example has shown that although the control flow allowed concurrency between the pay and file and update activities, the data flow demanded that pay cannot start after file and update history has already completed. Therefore, this data flow constraint could be interpreted as constraint on the execution order that would normally need to be reflected in the control flow.

Resolving Data Anomalies

- Too restrictive precondition
  - Loosen the precondition
- Implicit routing
  - Explicit routing conditions
- Implicit constraints on execution condition
  - Execute the updating activity after all reading activities
  - Condition: only one updating activity

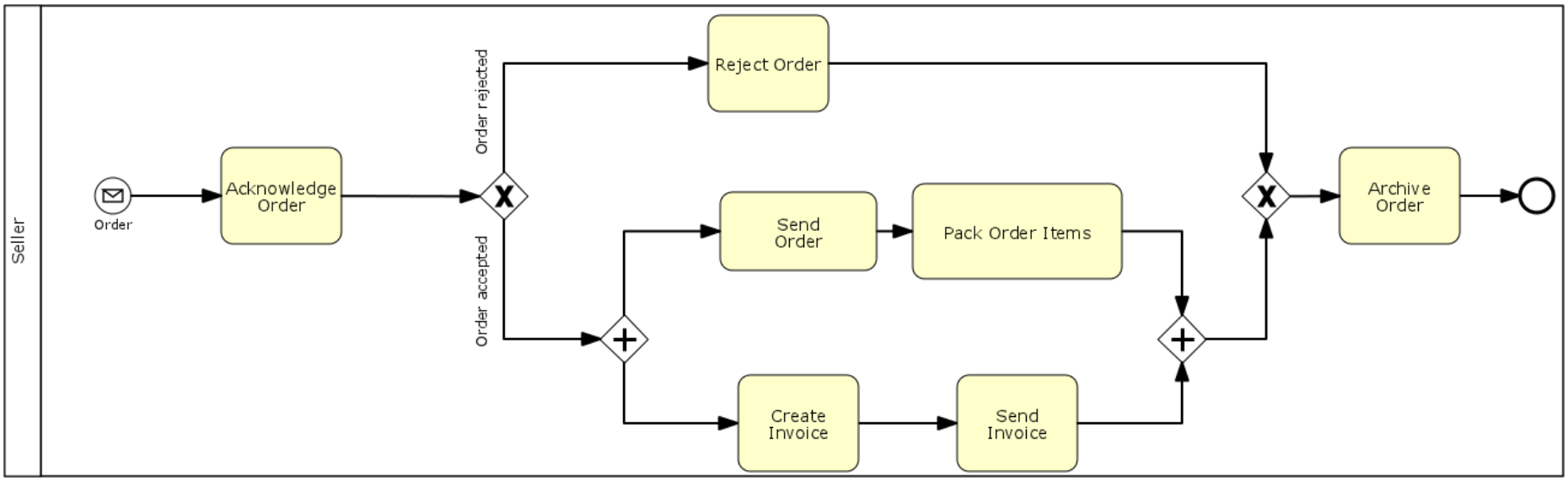
Consistency: Processes - Data Objects

- Motivation
- Notations about relationships between
  - Process model and life cycle (state transition diagram)
  - Multiple state transition diagrams
- Detect and correction of modeling errors
  - Errors in process models
  - Errors in behavioral descriptions of data objects

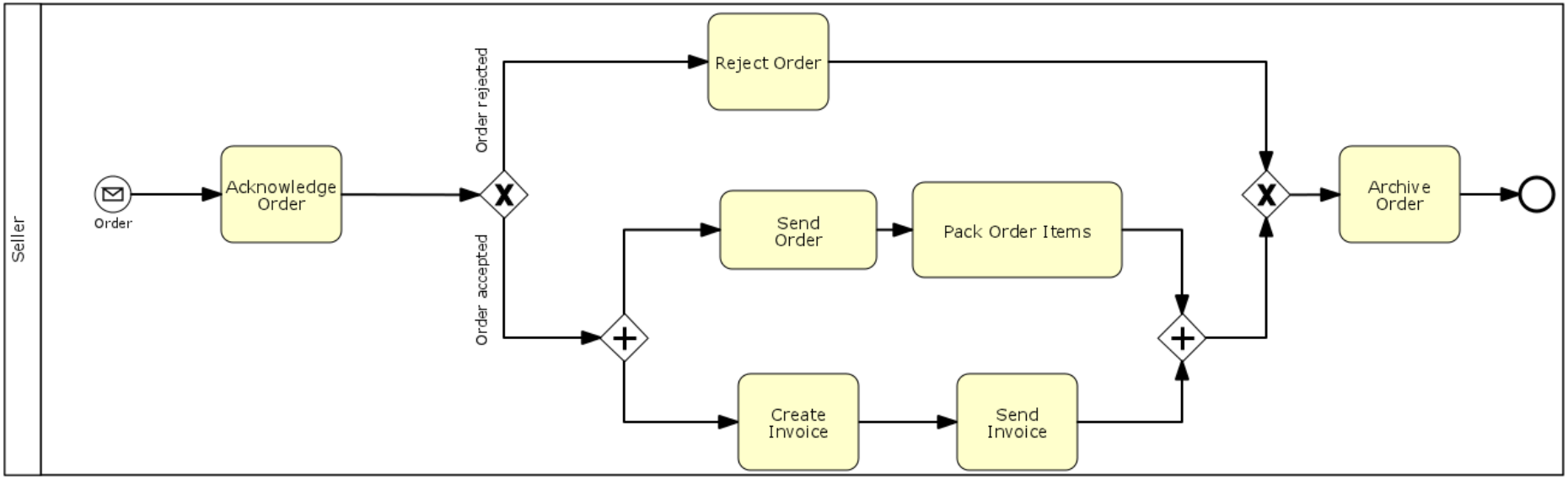
Motivation for consistency checking

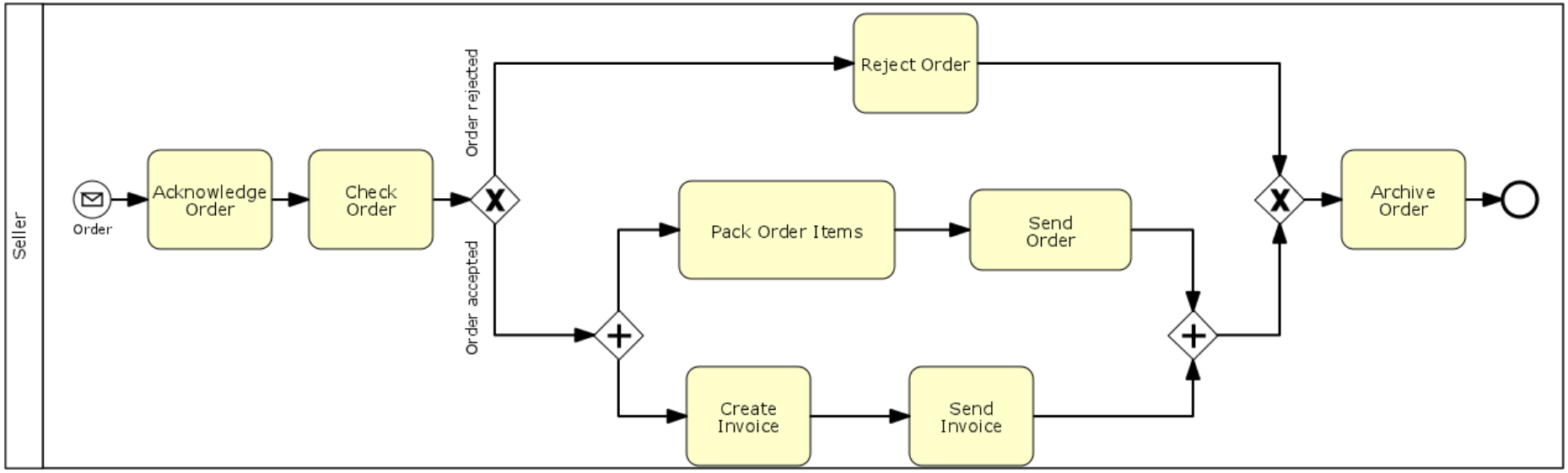
- Verifying that the process considers all data dependencies (deadlock avoidance)
- Comparison of the processes and related state transition diagrams
- Basis for compliance checking
  - Comparison of the current process against "best practice" models
  - "Best practice" models are often state-transition diagrams
  - Consistency between actual process and best-practice state transition diagram is needed

Motivation for consistency checking



Motivation for consistency checking



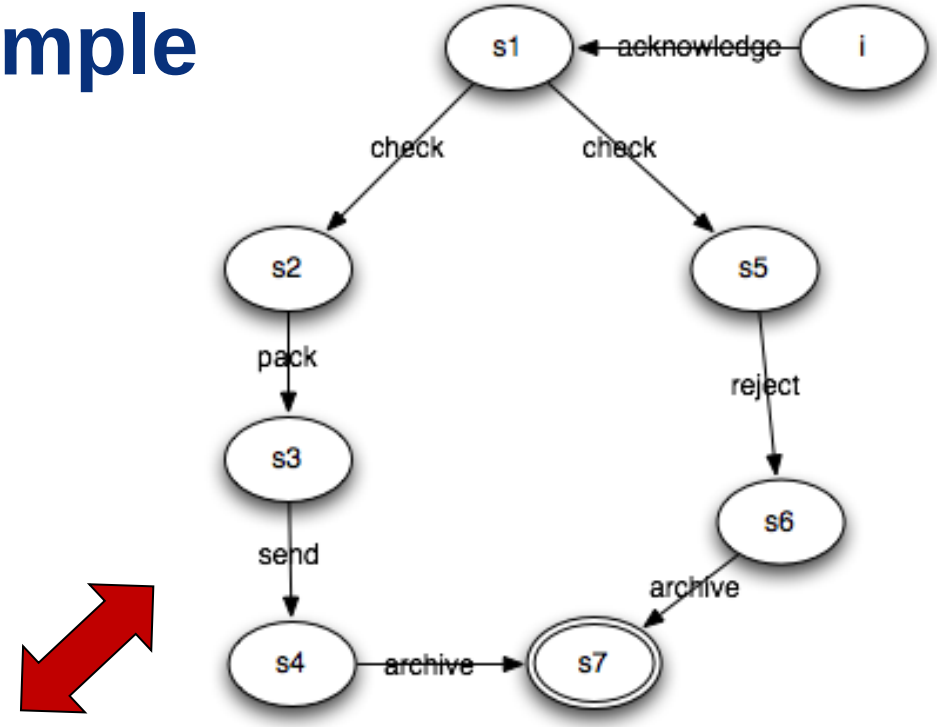


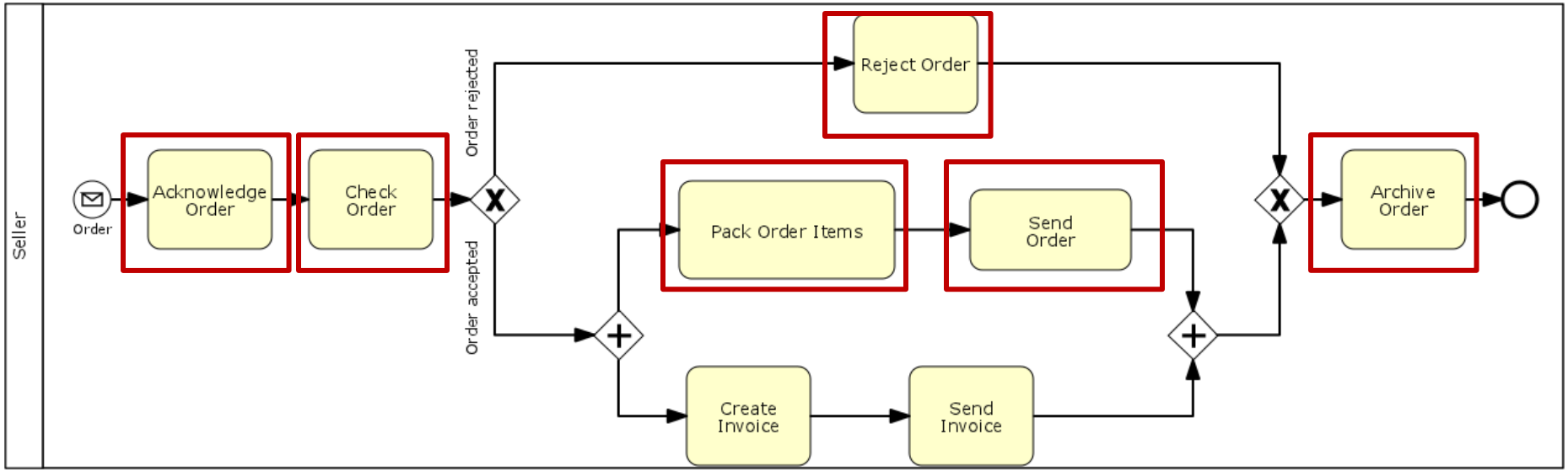
Consistency Checking

- Consistency checking by means of transformation of the process model into a state transition diagram
- Rules regarding
  - Object creation
  - State transition (change)
  - Object consumption
  - Final state
  - Process input
  - Process output
- Authors
  - IBM Research: Wahler und Küster, since 2006

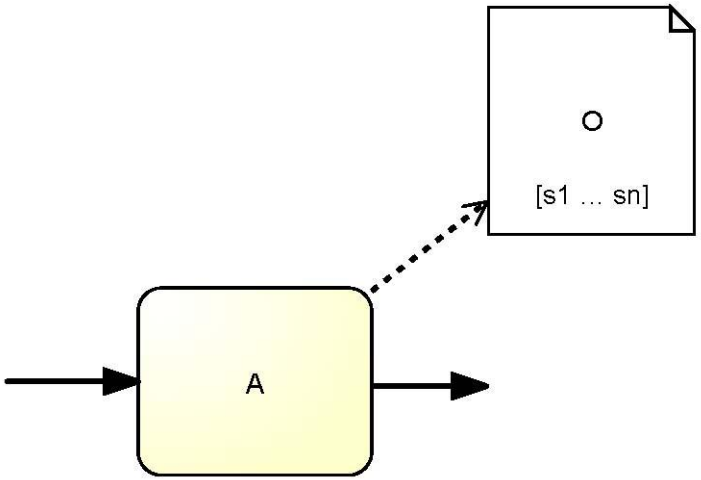
Transformation Example

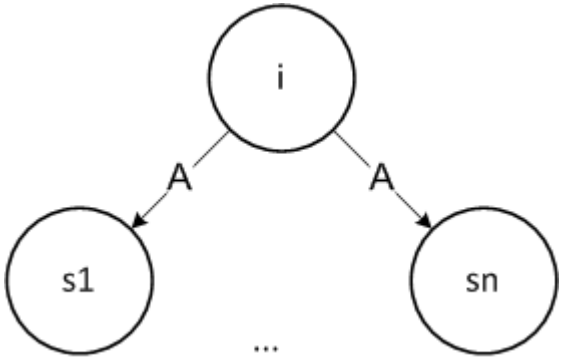
- Derive the data objet life cycle from the process





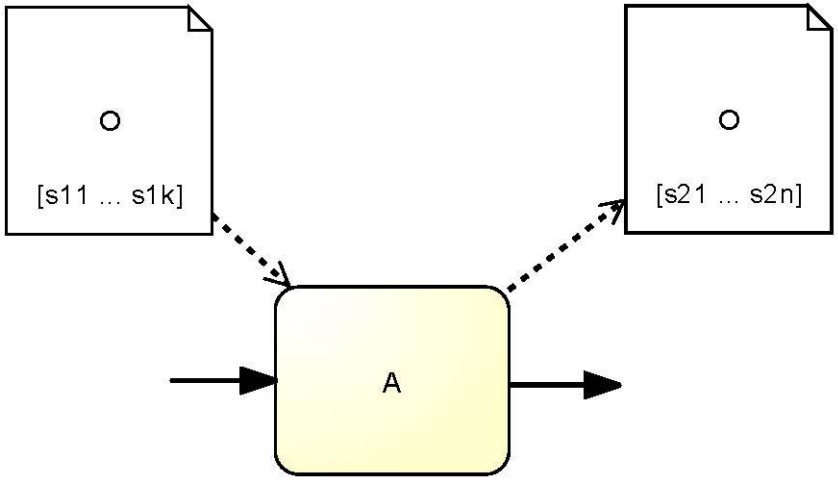
Rule 1: Object creation

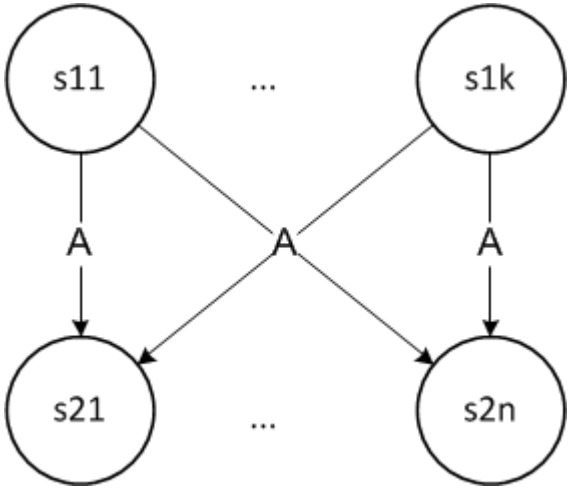




- Task A creates object O in one of *n* possible states
- Adding transitions with the label of the task on the current state

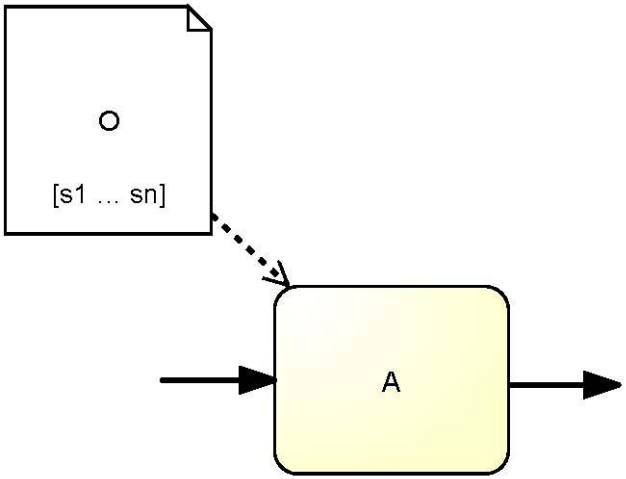
Rule 2: State change

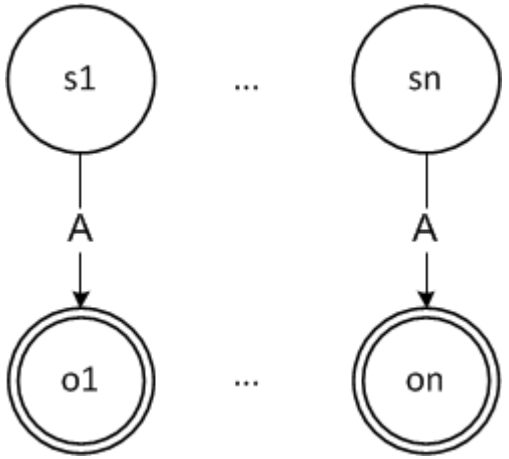




- Data object O can assume k different input and n different output states
- Adding transitions with the label of the task of each input state to each output state

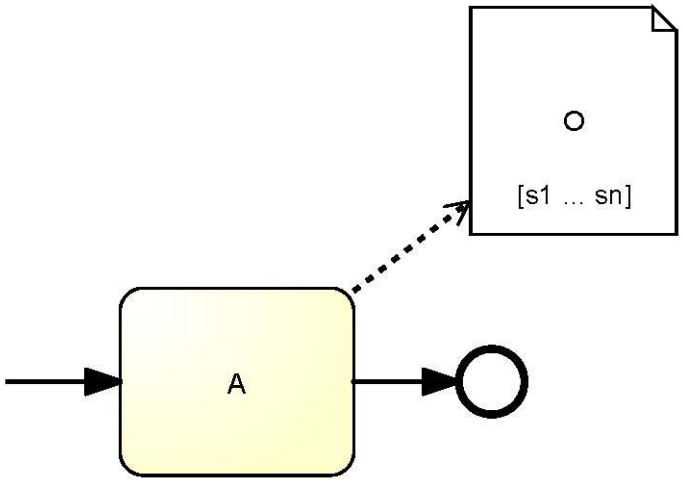
Rule 3: Object consumption

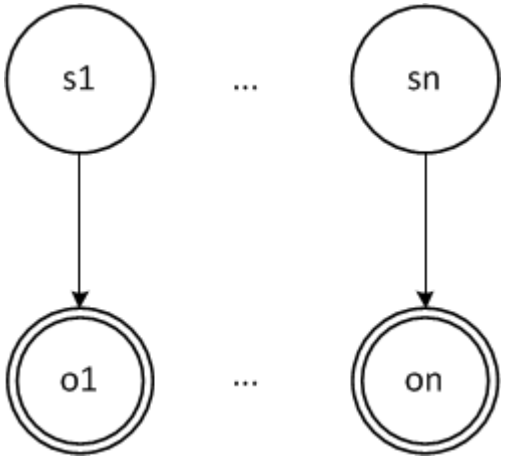




- Task A consumes object O when it is present in any of the n possible states
- Adding transitions with the label of the task to final states

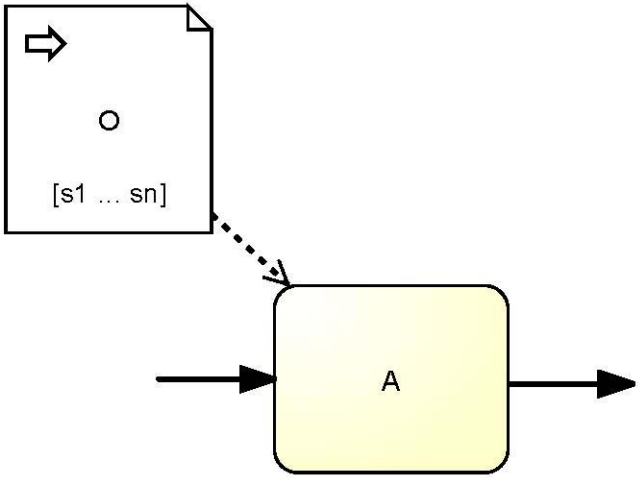
Rule 4: Final state

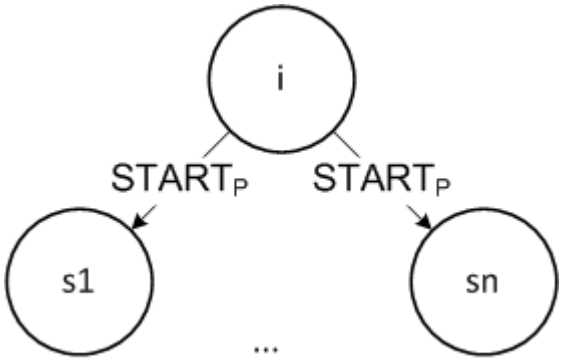




- Object O can be generated in n possible states by A
- Adding unlabeled transitions to final states

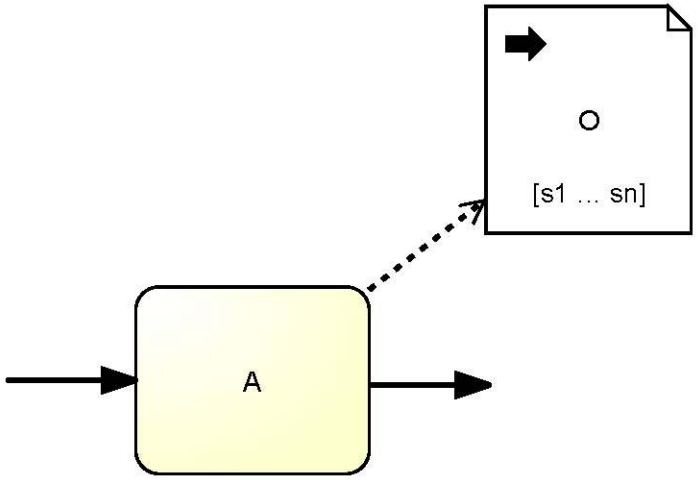
Rule 5: Process Input

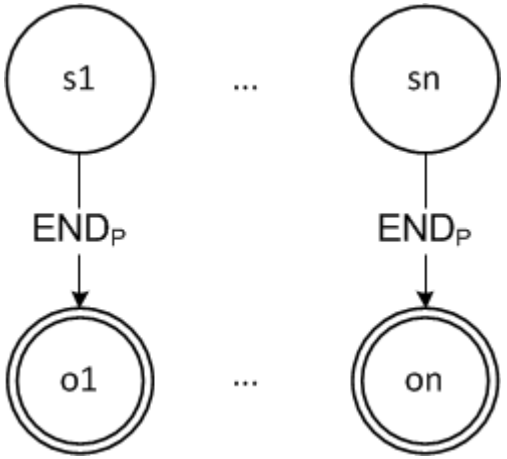




- Object O is an external Input for process P
- Adding transitions with the label STARTp from the initial state

Rule 6: Process Output





- Object O is an external output from process P
- Adding transitions with the label ENDp to the final states

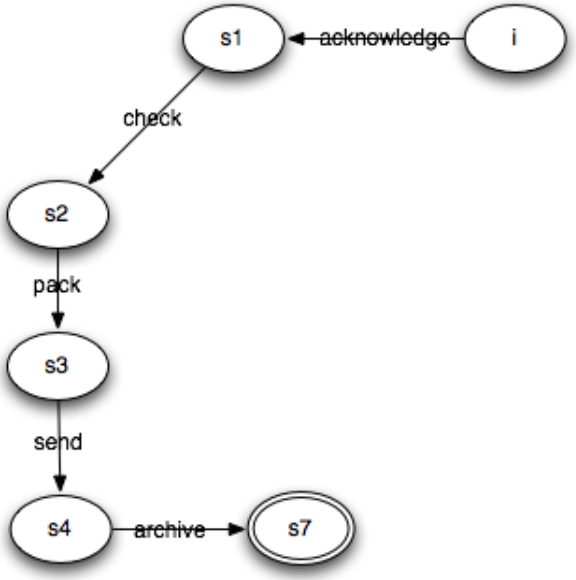
Consistency: Processes - Data Objects

- Idea
  - Define metrics to determine consistency of data objects and processes
- Object Lifecycle Conformance
  - A process fits to a data object, if the process requires only those state changes of the object that are defined in the life cycle of the object.
- Object Lifecycle Coverage
  - A process covers the behavior of a data object, if it allows all possible state transitions of the object.

Object Lifecycle Conformance (OLC)

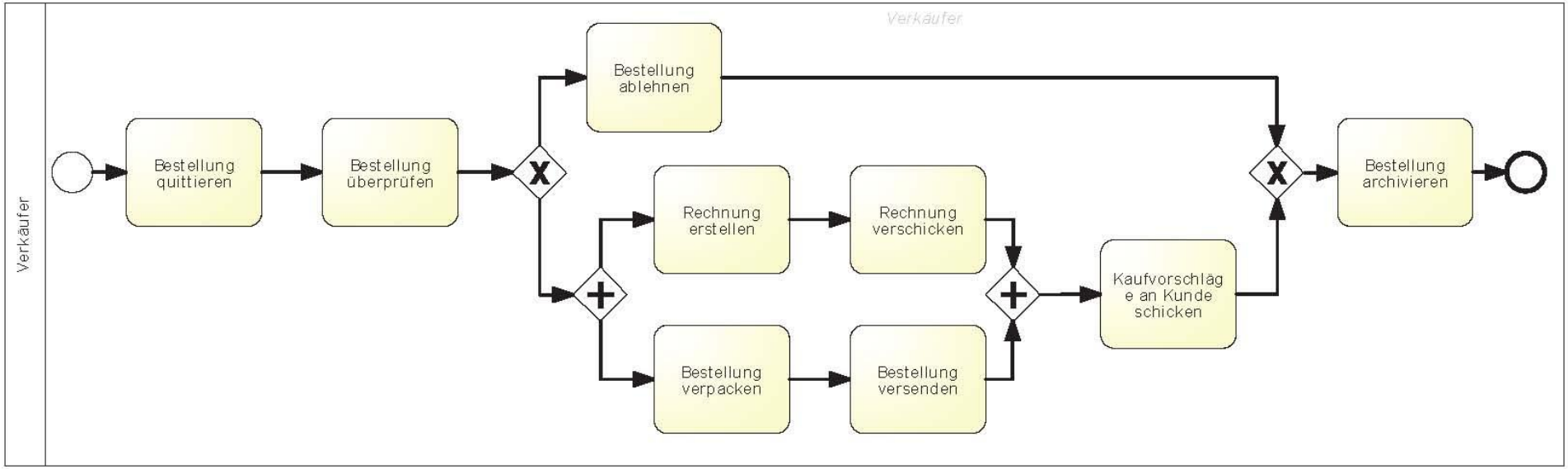
- Definition: The process P meets object-lifecycle-conformance for object o, if:
  - For each state transition of o in P, a corresponding state transition exists into the behavioral description of o
  - Start states and final states of o in P are also found in the behavioral description of o

Example 1: OLC



OLC not fulfilled

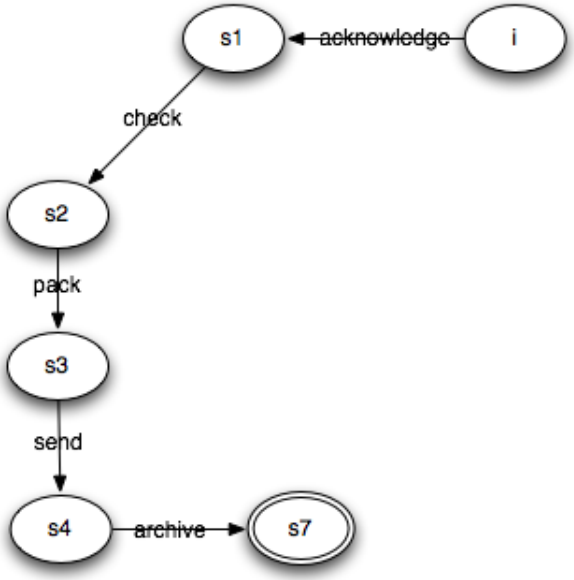
- State transition from "check" to "reject" for data object "order" does not exist in the behavioral description of the object



Object Lifecycle Coverage (OLCC)

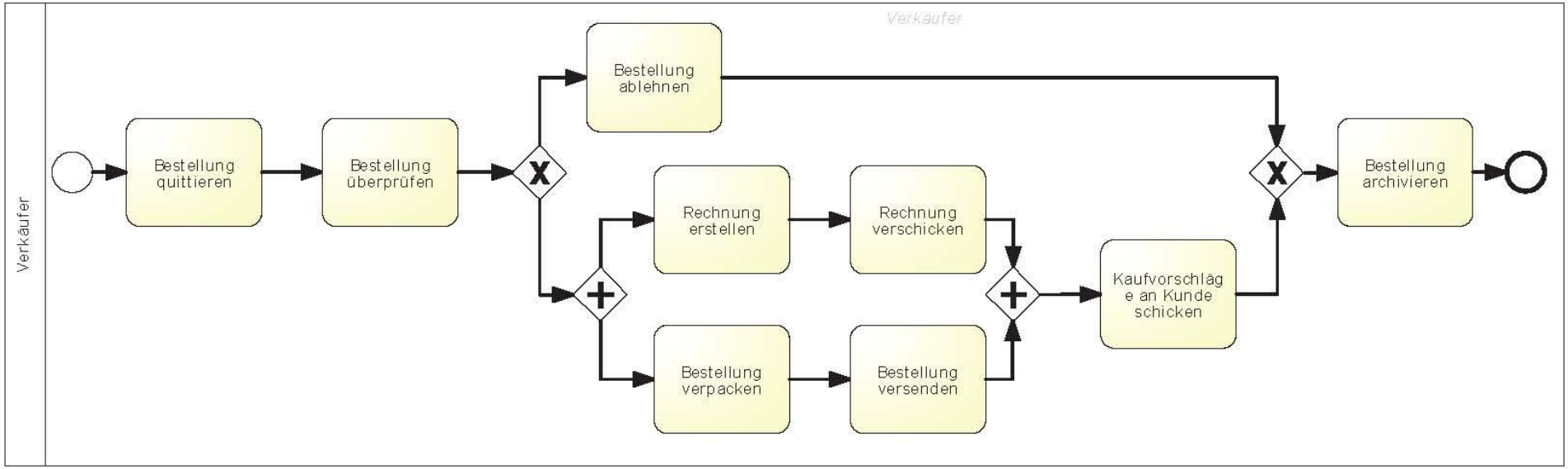
- Definition: The process P meets object-lifecycle-coverage for data object o, if:
  - For each state transition of o, there is a corresponding activity in P that triggers this state transition
  - For each initial state in the behavioral description of o, a counterpart state exists in P
  - For each final state in the behavioral description of o, a counterpart state exists in P

Example 1– OLCC

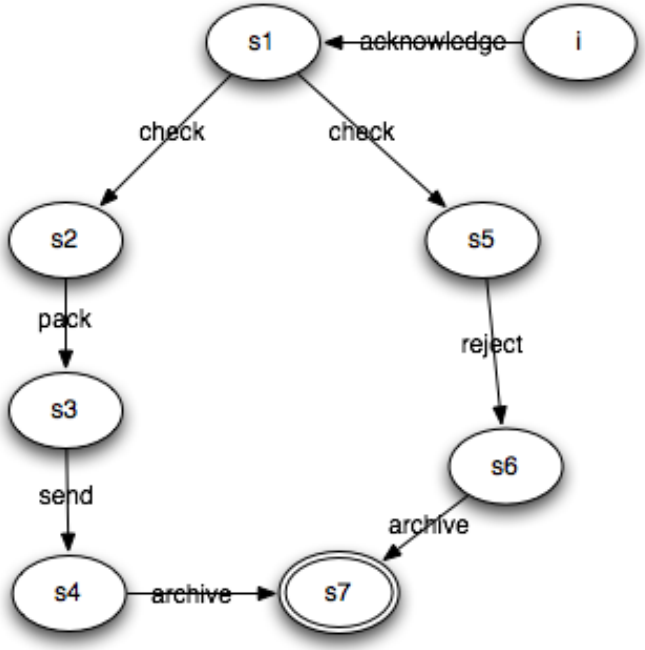


OLCC fulfilled

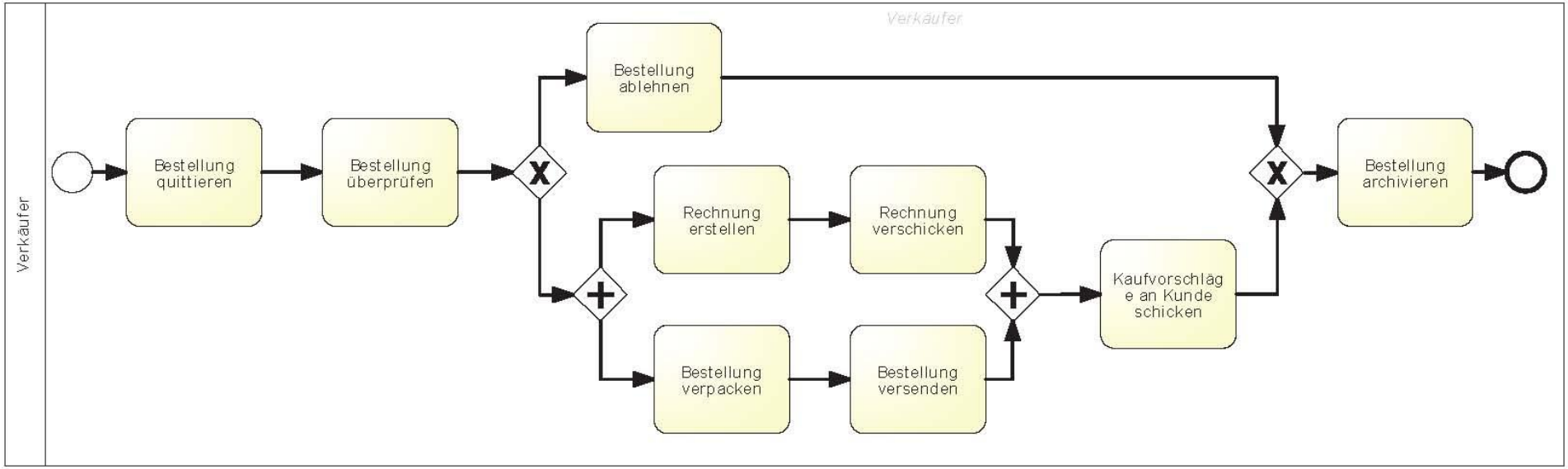
- The behavior of the object is covered by the process in its entirety



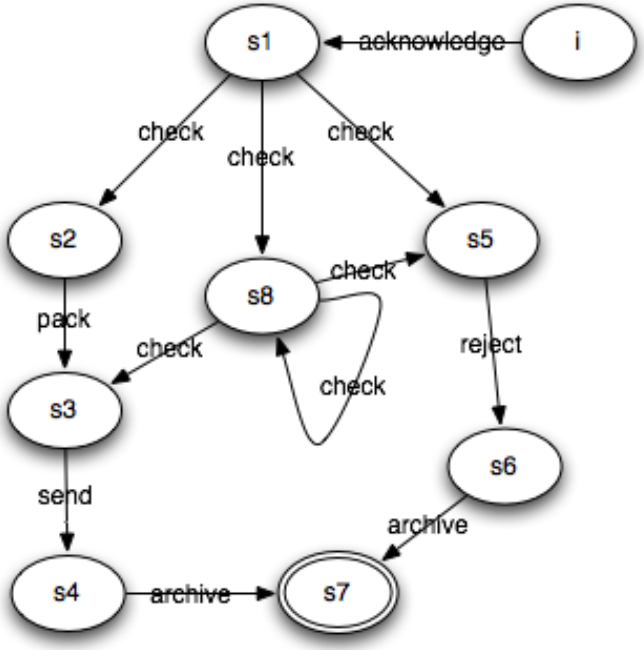
Example 2



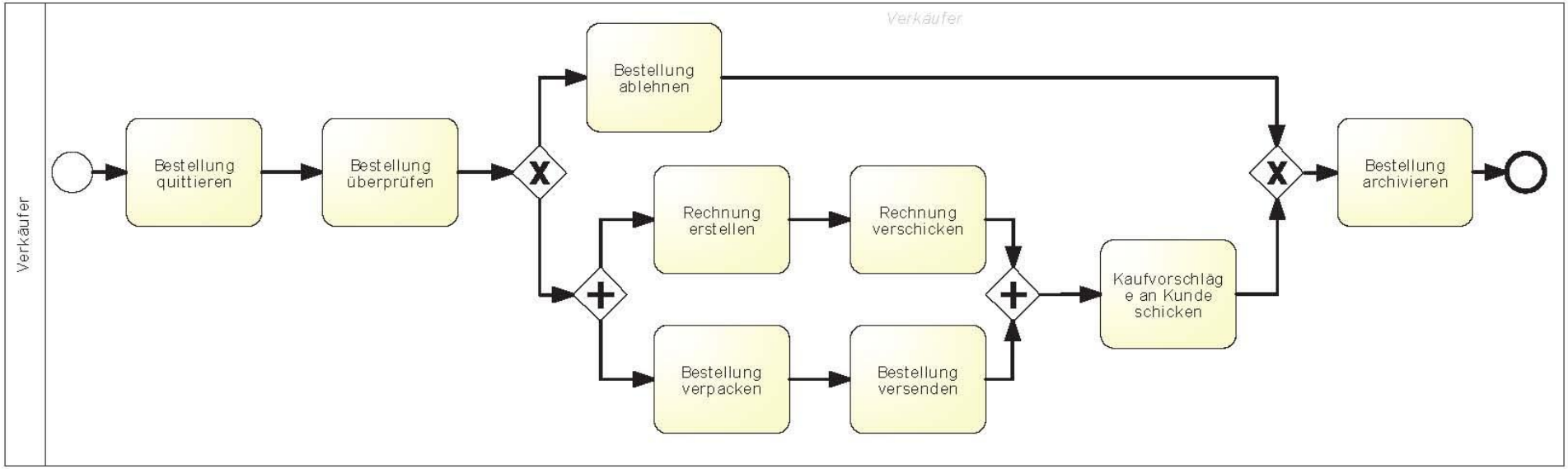
- OLC fulfilled
  - Only allowed state transitions of the object are required in the process
- OLCC fulfilled
  - The behavior of the object is covered by the process in its entirety



Example 3



- OLC fulfilled
  - Only allowed state transitions of the object are required in the process
- OLCC not fulfilled
  - Object "can do much more" than is covered by the process



OLC/OLCC Use Cases

- By OLC and OLCC, errors and inconsistencies are discovered during modeling time
  - In the process: Missing or incorrect order of activities
  - The object: Error in the behavioral description of objects
- Generation of process fragments from behavioral descriptions
- Expansion conceptual processes in order to get implemented processes