# Statechart Diagrams

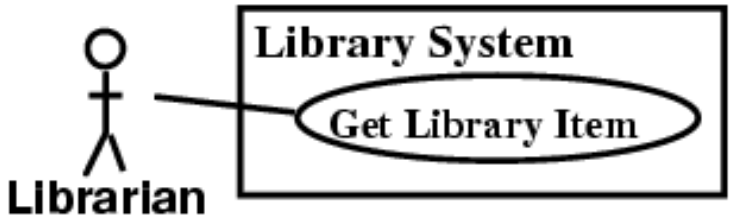
- Purpose:
  - to specify the behavior of the instances of a given class in response to external stimuli formally
- UML notation:
  - a directed graph of states connected by transitions
- Origins:
  - finite state machines (formal setbased notation)
  - state transition diagrams (formal graphical notation)
  - Harel's statecharts (formal graphical notation )

# NO FORMAL SPECIFICATION OF CLASS BEHAVIOR

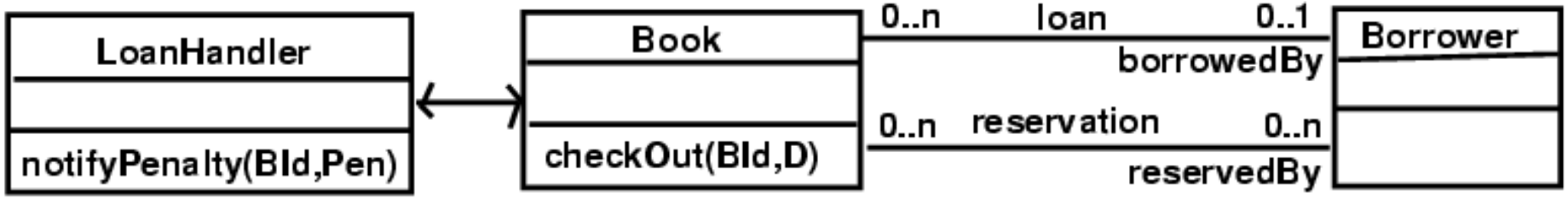
- The library borrower goes to the librarian and asks for a book. The librarian swipes the borrower card and the isbn number of the book using the barcode reader. If the borrower has not exceeded his/her quota and the book has not been reserved the borrower gets the book.

# NO FORMAL SPECIFICATION OF CLASS BEHAVIOUR SO FAR

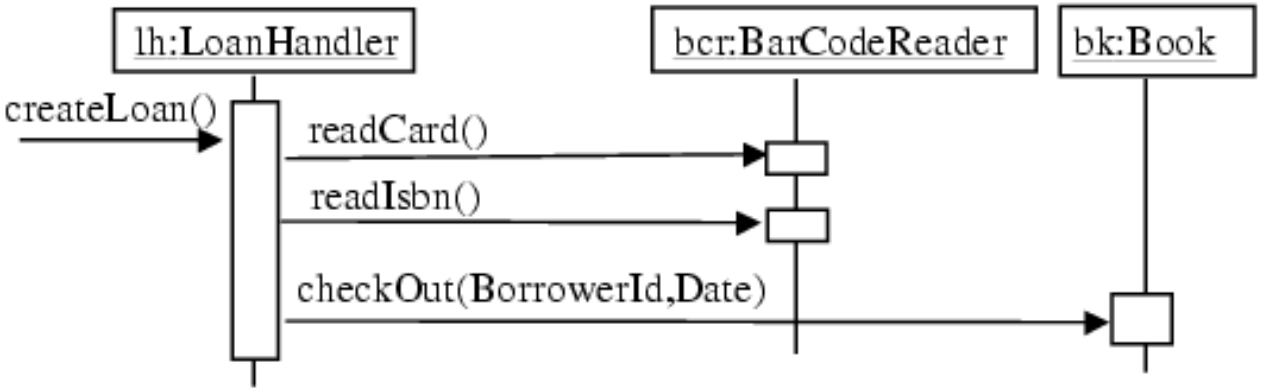
## *Use Case Diagram/Description:*




## *Design Class Diagram:*




## *Sequence Diagram:*



# What is State?

- A state is a condition during the life of an object or an interaction during which:
  - it satisfies some condition
  - performs some action
  - waits for some event
- An object remains in a state for a finite time

# STATECHART DIAGRAM: The basic semantics

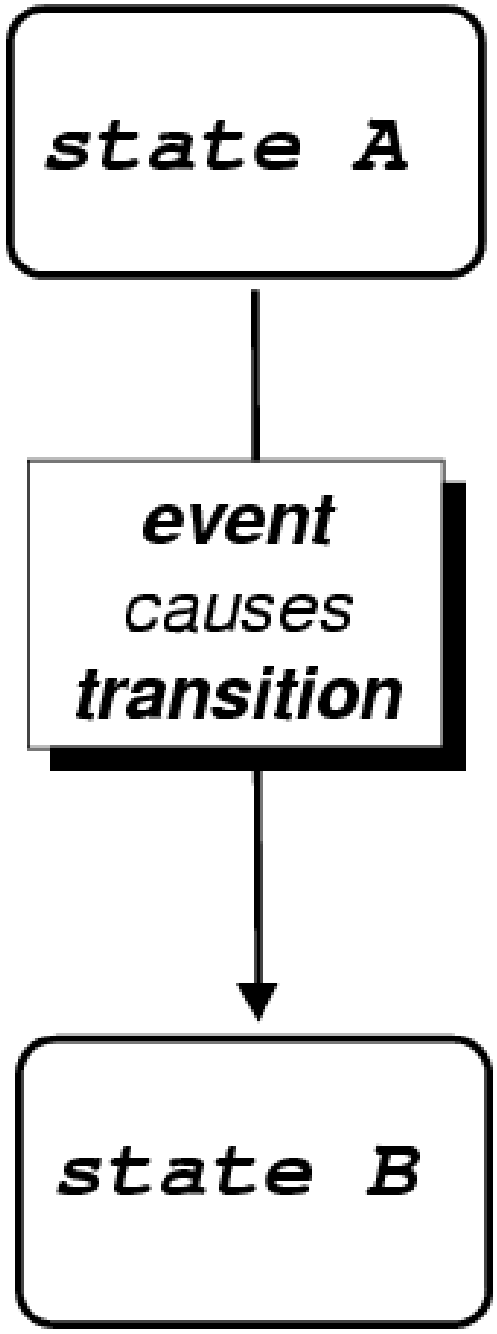
**event:** an occurrence of

- » a change in the truthvalue of a condition
- » a receipt of a call for an operation
- » the end of a designated period of time

**state:** a finite (notinstantaneous) period in the life of an object during which this object

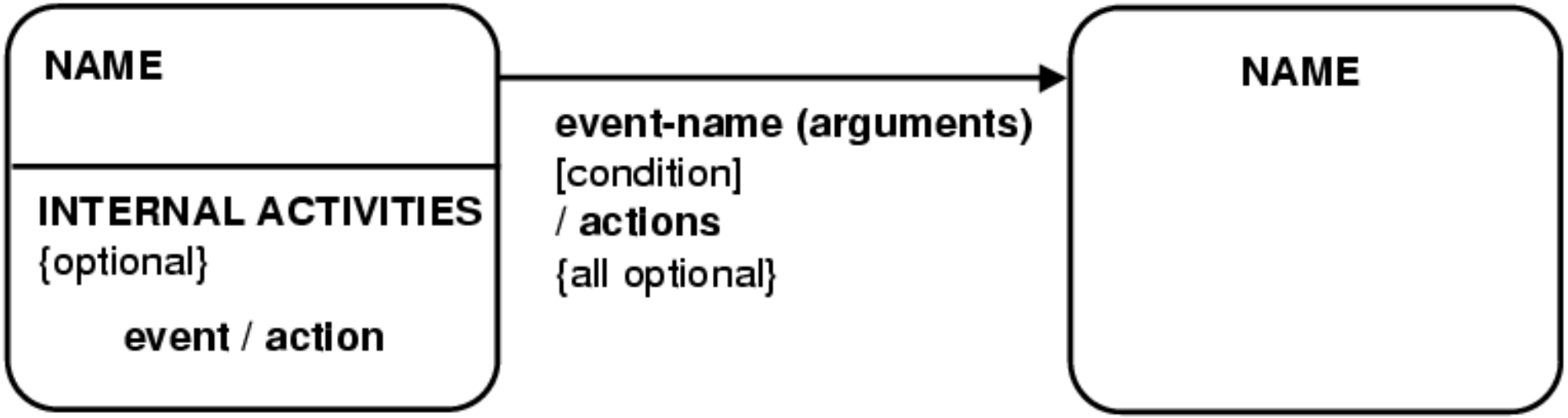
- » satisfies some condition; or
- » performs some action; or
- » waits for some event to occur

**transition:** a response to an event received by an object which is in a certain state

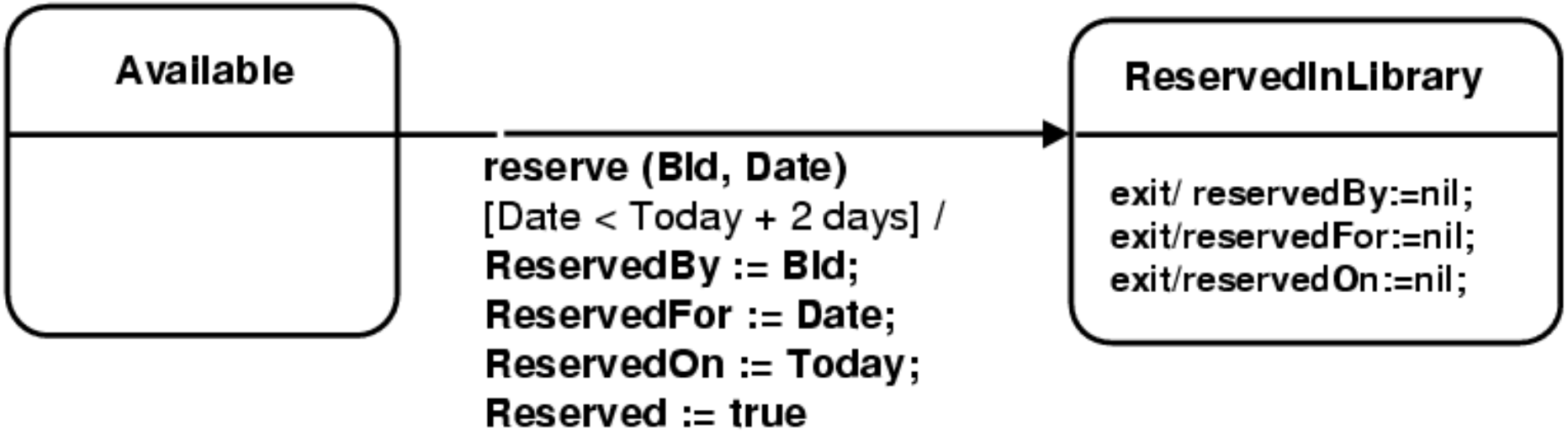

state A
event
causes
transition
state B

- UML provides a graphical notation for representing these concepts which is introduced in the next slide

# STATECHART DIAGRAM: The basic notation

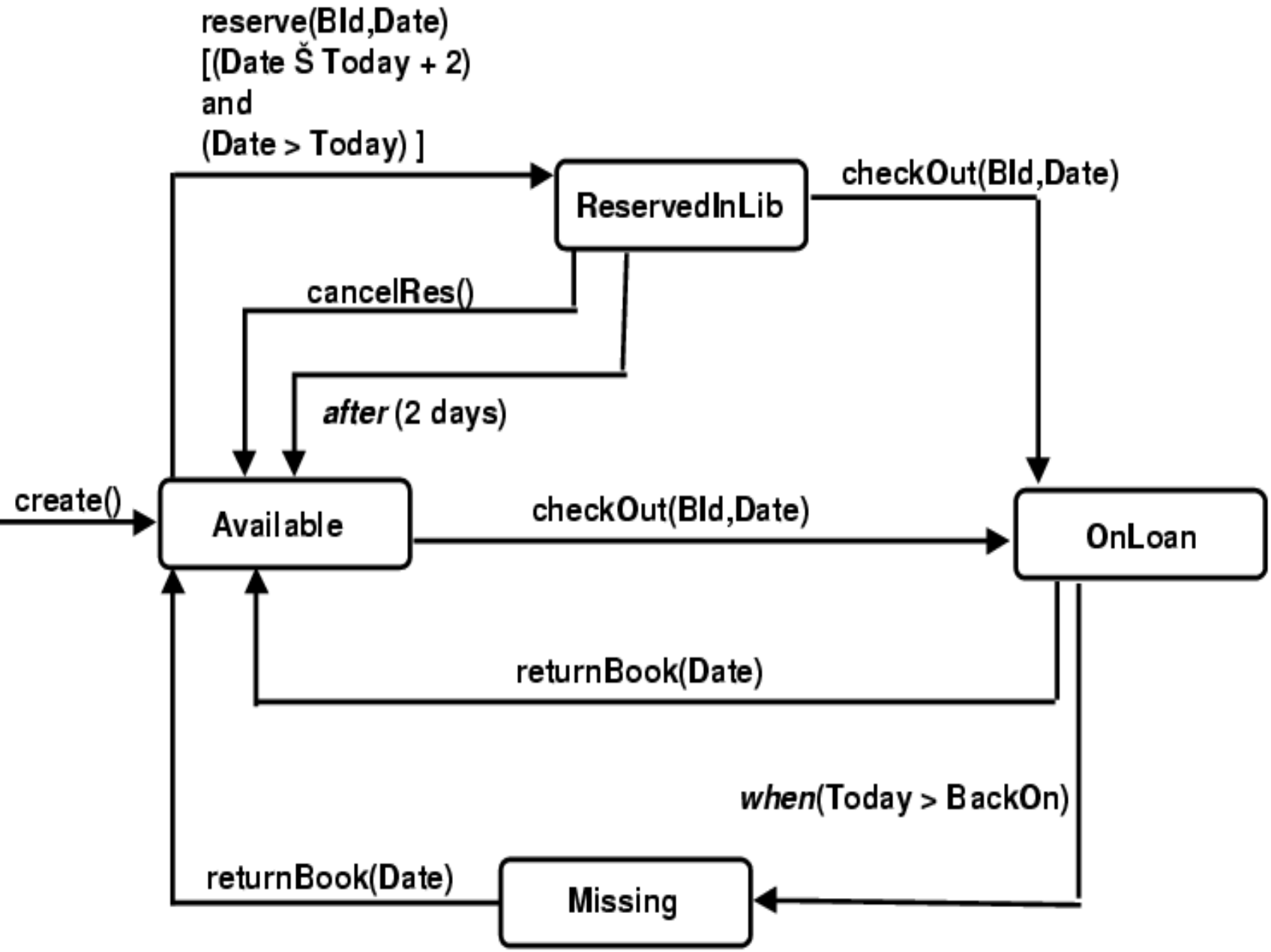



**example:** two states of a library book



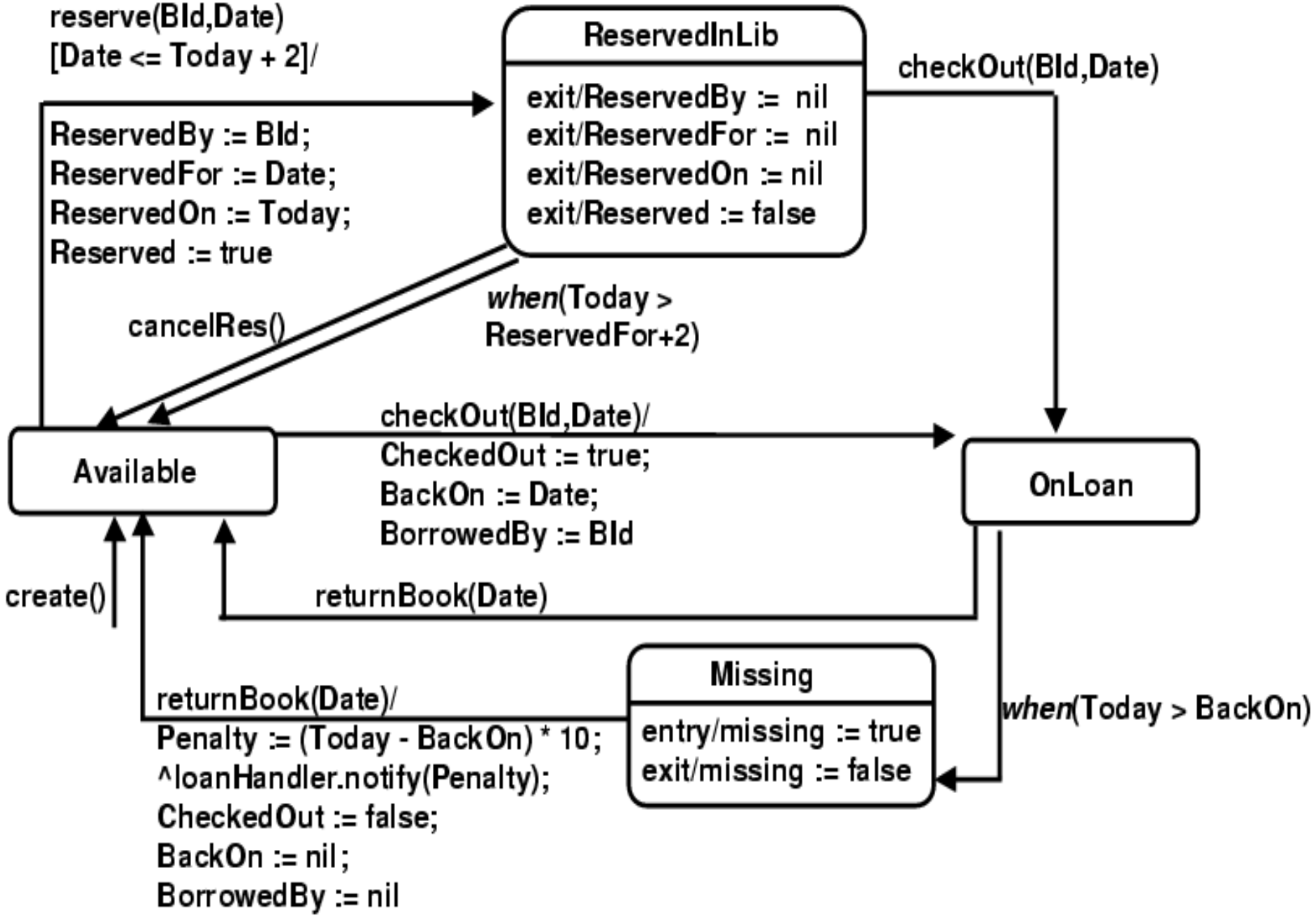
The next slide gives an example of specifying the behaviour of a library book using a statechart diagram

# Example: A statechart for a library book (basic states and transitions)



- The library book statechart diagram may be expanded to specify:
  - the actions that are triggered when the transitions fire
  - the internal actions which are triggered by events which occur while the book is in each of its states but do not cause any state transition.

# Example: A statechart for a library book (actions)



# Composite states

- A state can be decomposed into
  - a set of concurrent substates known as **and** decoposition
  - or a set of mutually exclusive substates known as or decomposition
- A state which is decomposed into substates is called composite state

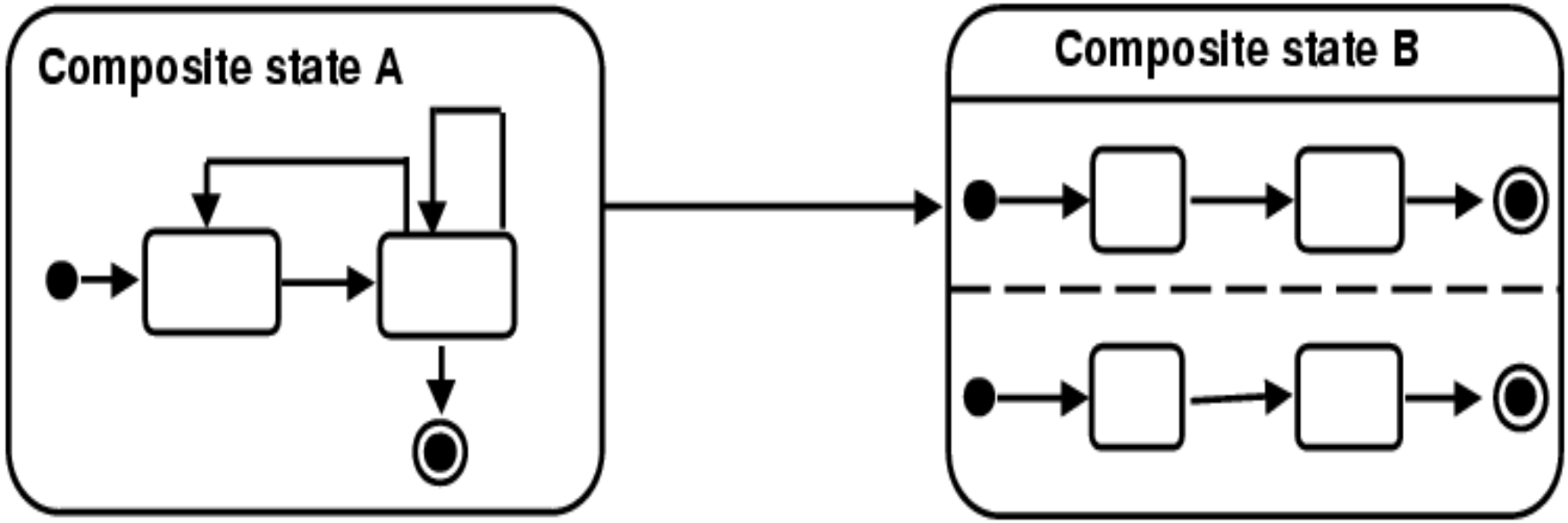
# AND / OR Decomposition

- Substates may also be decomposed in exactly the same way
- The meaning of the ordecomposition is that when the composite state is 'active' the object might be in exactly one of its substates
- The meaning of the anddecomposition is that when the object is in the composite state it is concurrently in all the concurrent substates

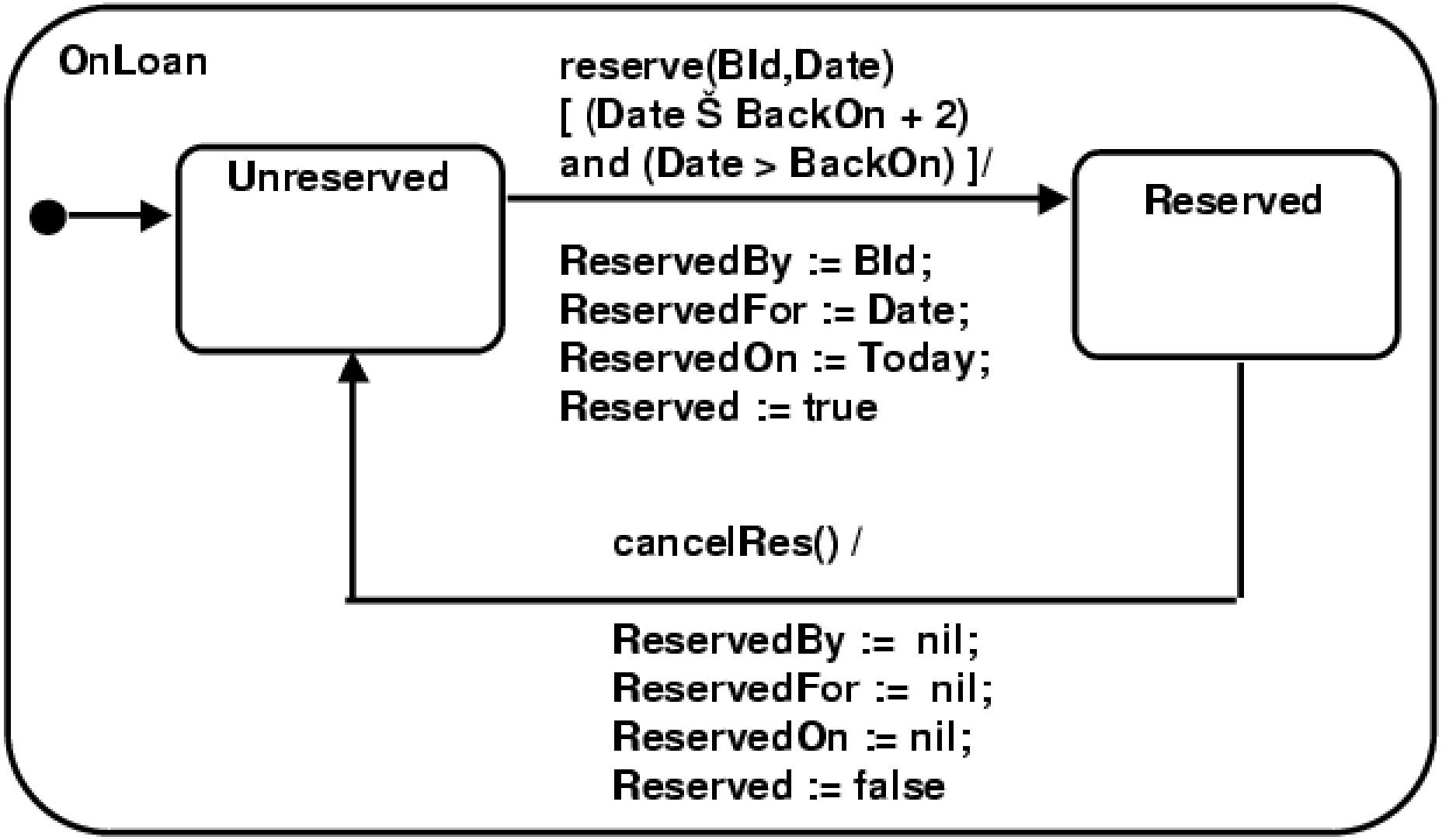
# STATECHART DIAGRAM: Composite states

notation for **or-decomposition**

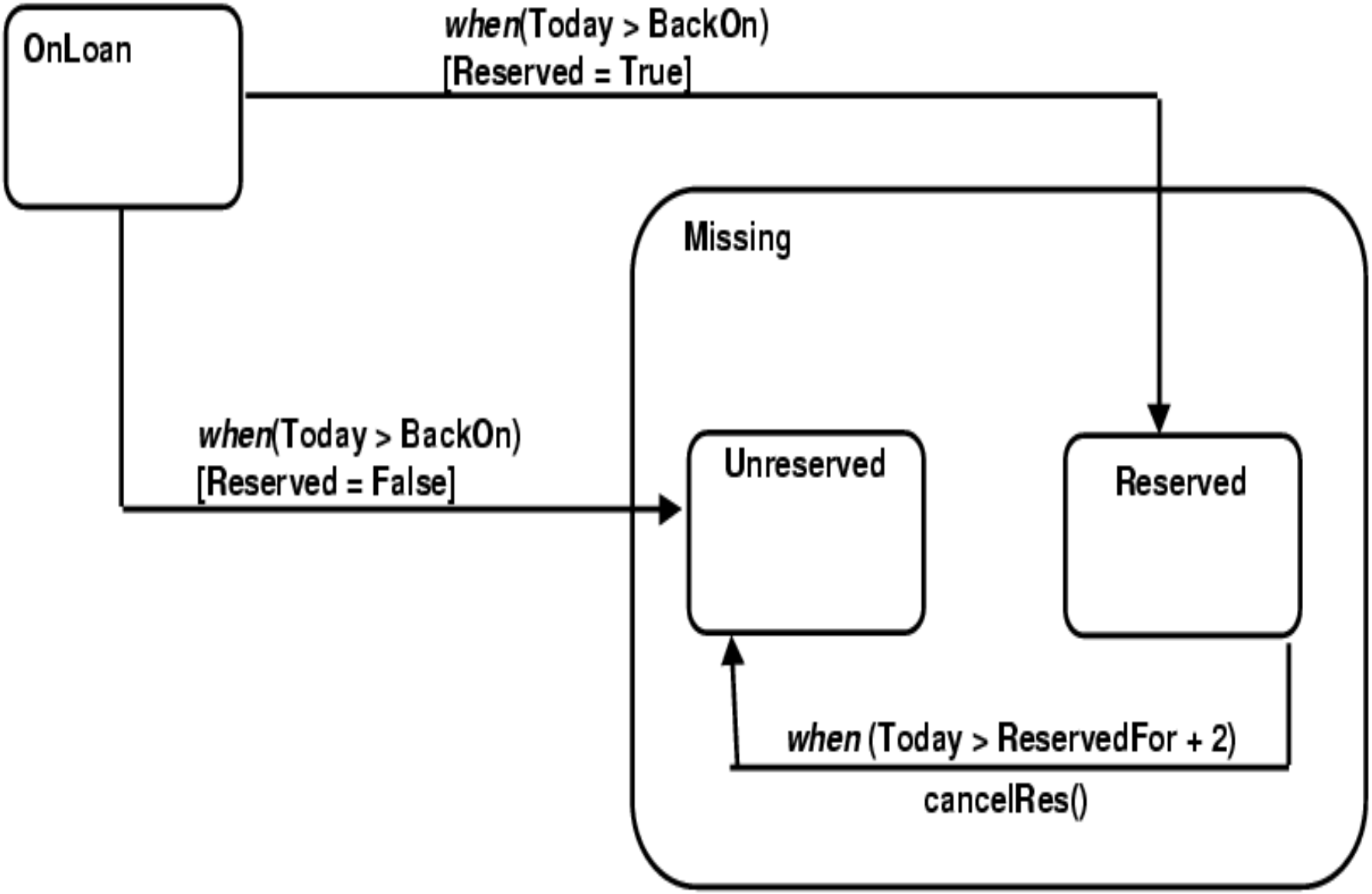
**notation for and- decomposition**



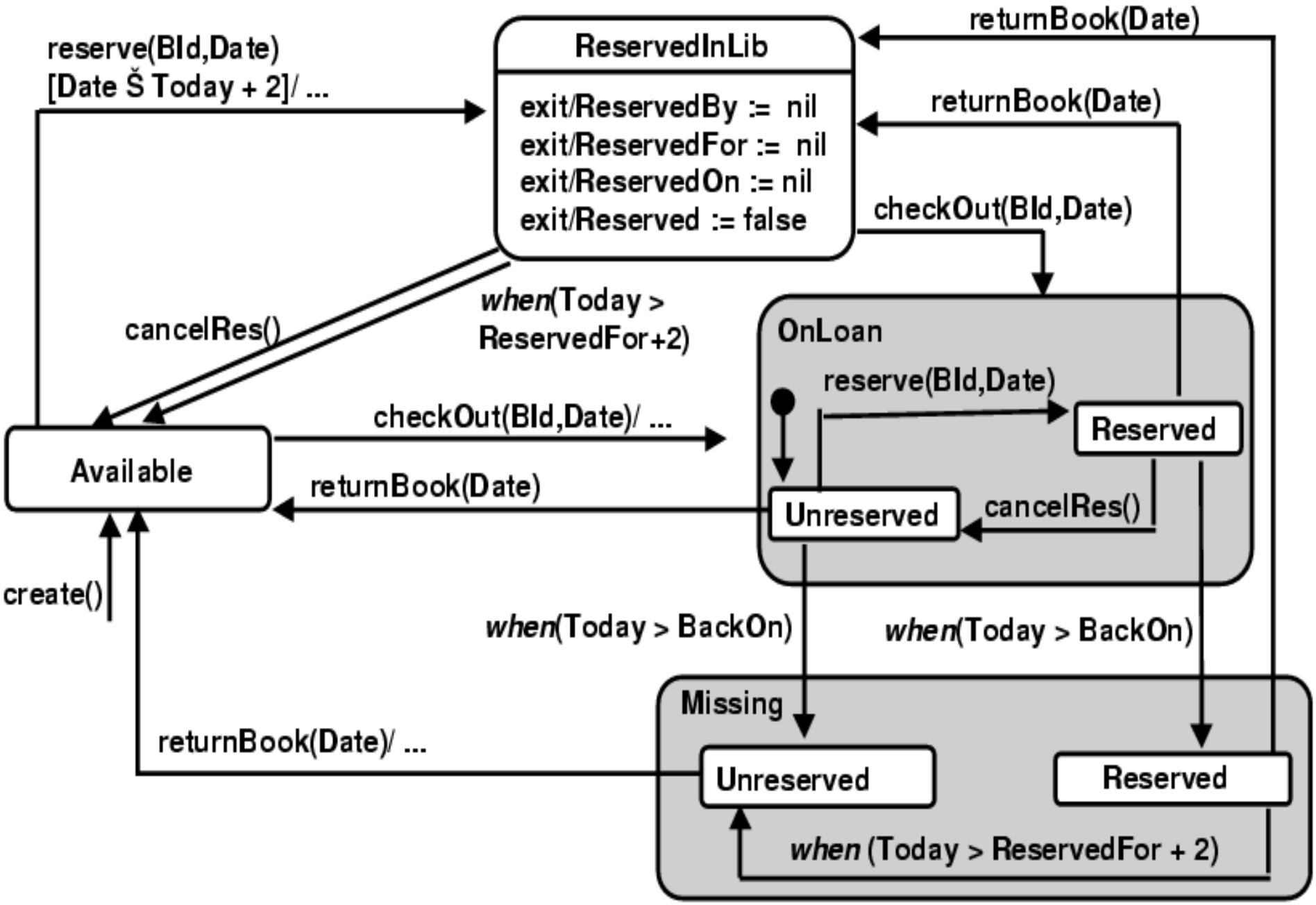
# Example of OR-decomposition



# Example of OR-decomposition



# A statechart for a library book (revised)



# AND Decomposition

# Role of statecharts in system development

- The statechart diagrams:
  - formally specify the behaviour of objects
  - increase understanding of classes
- describe what happens when stimuli and other kind of events occur within a system and its environment
- provide abstract and partial descriptions of the actual code

# Consistency between Statecharts and the Design class diagrams

- Statecharts must be checked against the design class diagramm to ensure that:
  - the messages which label transitions correspond to operations of the relevant class
  - the attributes and associations which are referenced by transition and state actions have been defined for the relevant class
  - the messages which are sent to other objects correspond to operations in the definition of the classes of these objects

# Conclusions

- Modelling class behaviour :
  - is an essential part of design and the necessary final step before detailed coding
  - requires an understanding of the different states in which the classes might be and the transitions between these states
  - can be expressed using the Statechart notation of UML
  - can be used to check other artefacts of system design