State Diagrams

Chapter 5

Objectives

In this chapter we will:

- ❒ **Introduce the terms used with respect to state diagrams**
- ❒ **Discuss the context in which state diagrams are used**
- ❒ **Introduce substates**
- ❒ **Discuss concurrent state diagrams**

Statechart Diagrams

- State diagrams describe the life of an object using three main elements:
  - States of an object
  - Transitions between states
  - Events that trigger the transitions
- A state diagram or statechart specifies a state machine
  - A state machine is described for a class
  - Each object has it’s own state machine

Why Use Statechart Diagrams?

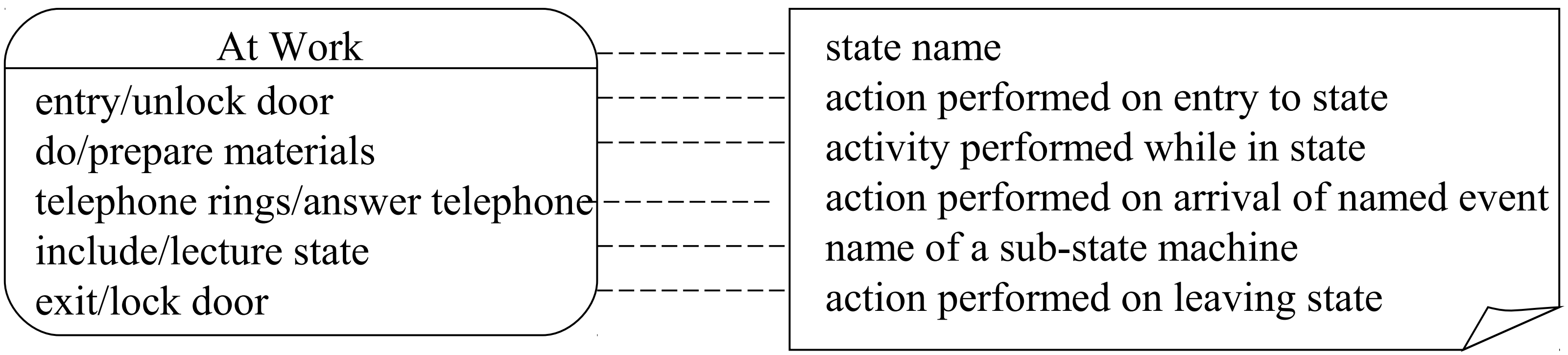
- Statecharts typically are used to describe state-dependent behaviour for an object
  - An object responds differently to the same event depending on what state it is in
  - Usually applied to objects but can be applied to any element that has behaviour
    - Actors, use cases, methods, subsystems, systems
- Statecharts are typically used in conjunction with interaction diagrams (usually sequence diagrams)
  - A statechart describes all events (and states and transitions for a single object)
  - A sequence diagram describes the events for a single interaction across all objects involved

States

- Show what the dependencies are between the state of an object and its reactions to messages or other events
- State
  - is a condition or situation during the life of an object within which it performs some activity, or waits for some events
  - Has a name
  - Has actions -- execute the state
  - Has internal transitions -- transitions cause no change in a state
  - substates -- the nested structure of a state involving disjoint or concurrent substates

States

- For example:

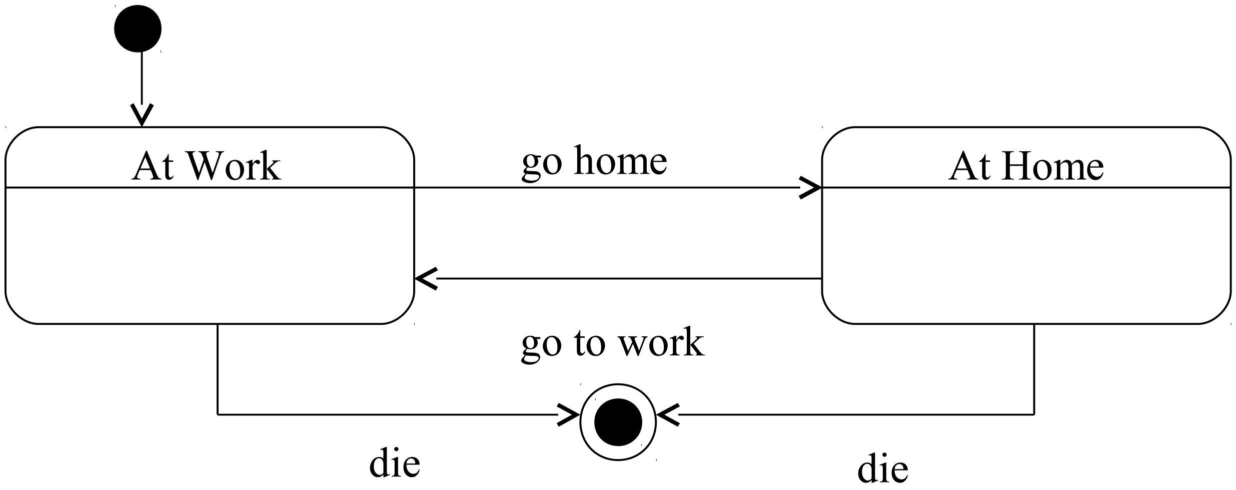


Initial and Final States

- The initial state of a state machine is indicated with a solid circle
  - Known as a pseudo-state
  - A transition from this state will show the first real state
- The final state of a state machine is shown as concentric circles
  - A closed loop state machine does not have a final state; the object lives until the entire system terminates
  - An open loop state machine represents an object that may terminate before the system terminates

Initial and Final States

- An example:



Actions and Activities

- Action
  - is an executable atomic computation
  - includes operation calls, the creation or destruction of another object, or the sending of a signal to an object
  - associated with transitions and during which an action is not interruptible -- e.g., entry, exit
- Activity is associated with states
  - Non-atomic or ongoing computation
  - May run to completion or continue indefinitely
  - Will be terminated by an event that causes a transition from the state in which the activity is defined

Events

- An event signature is described as
  Event-name (comma-separated-parameter-list)
- Events appear in the internal transition compartment of a state or on a transition between states
- An event may be one of four types
  - Signal event
    - Corresponding to the arrival of an asynchronous message or signal
  - Call event
    - Corresponding to the arrival of a procedural call to an operation
  - Time event
  - Change event

Events

- A time event occurs after a specified time has elapsed
  - Event name is specified as keyword after
  - Parameter list is an expression evaluating to a time interval
    - after(10 seconds after state “At Work” is entered)
  - No specified start time implies “since entry to the current state”
    - after(2 seconds)

Events

- A change event occurs whenever a specified condition is met
  - Event name is specified as keyword *when*
  - Parameter list is a boolean expression
  - The event occurs when both of the following conditions are met, irrespective of the order when they happen
    - The expression evaluates to true
    - The object is in the required state
  - For example
    - when (state = At Work)
    - when (date = January 1 2007)

Transitions

A transition is drawn as an arrow between states annotated with a transition string

- The transition string denotes the event and consequent action
- Only one form of arrowhead is used on statecharts
    - The distinction between call events and signal events must be deducted from elsewhere e.g. an interaction diagram

A transition string is described as

- *Event-signature [guard-condition]/action-expression^object.message*
- *If the guard condition is met the transition occurs immediately*

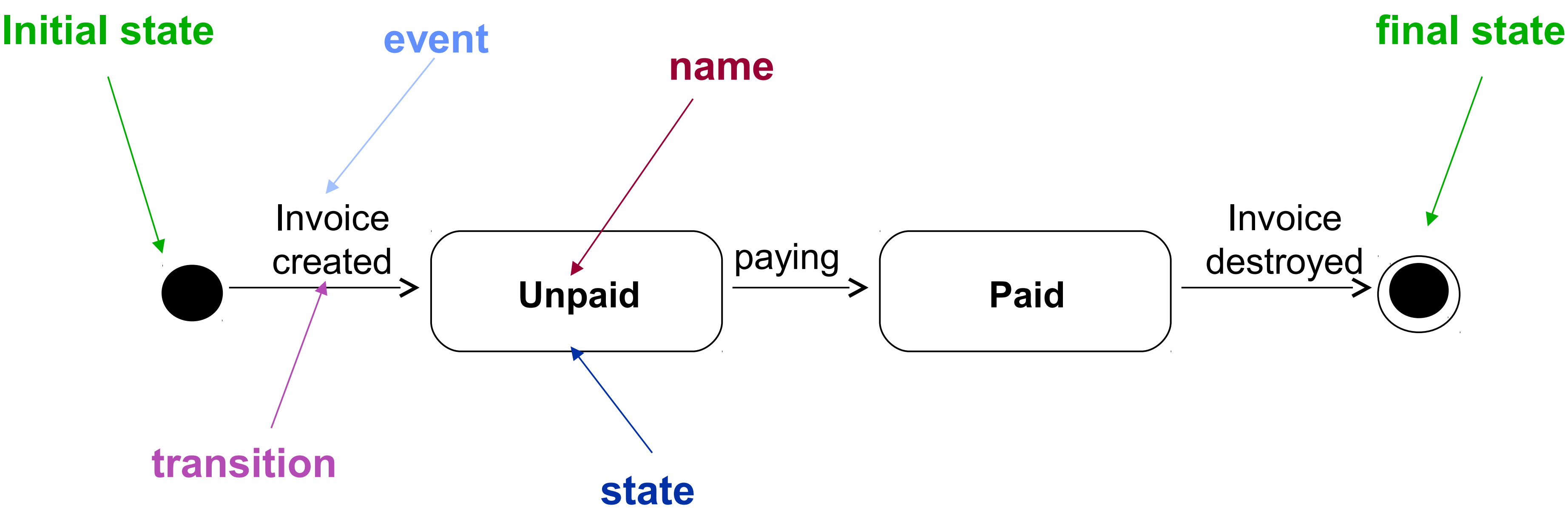
Transitions

- A transition whose string contains neither an event signature nor a guard condition is said to be unlabeled
  - Occurs immediately
  - May still carry an action expression

Transitions

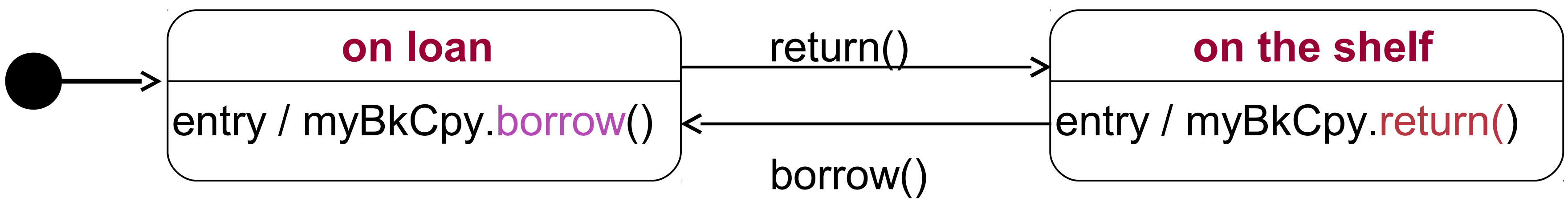
- A transition is triggered when its event occurs
  - If the guard condition is met, the transition is fired
  - If the condition is not met the event is discarded
    - The guard condition is checked only once
- If there is no guard condition, triggering will always cause firing
- Note the distinction between a guard condition and a change event
  - A guard condition is evaluated once, when the associated event occurs
  - A change event occurs whenever its associated condition is met
    - Behaviour is as if the condition were being continually evaluated

State Diagrams notation

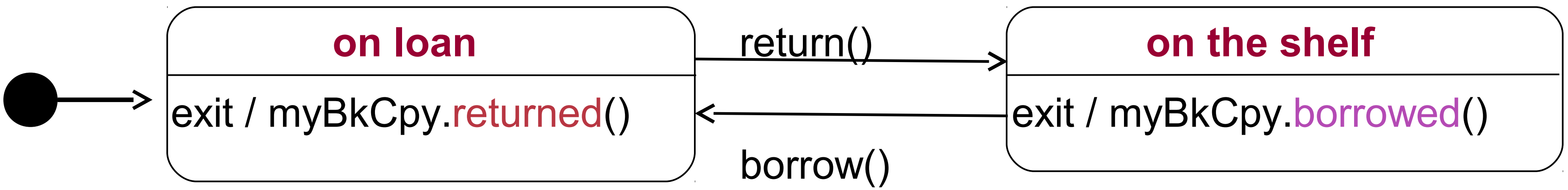


State Diagram *Example*

This shows the state of an object myBkCpy from a BookCopy class

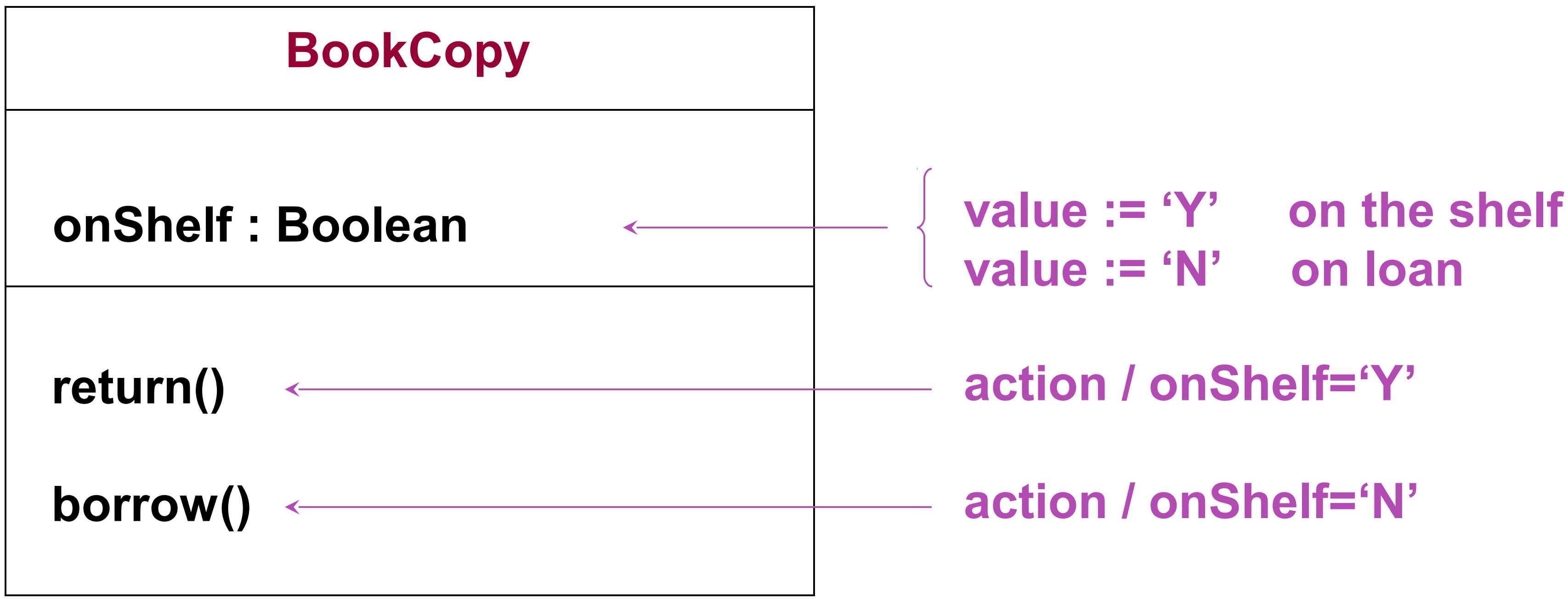


Entry action : any action that is marked as linked to the entry action is executed whenever the given state is entered via a transition

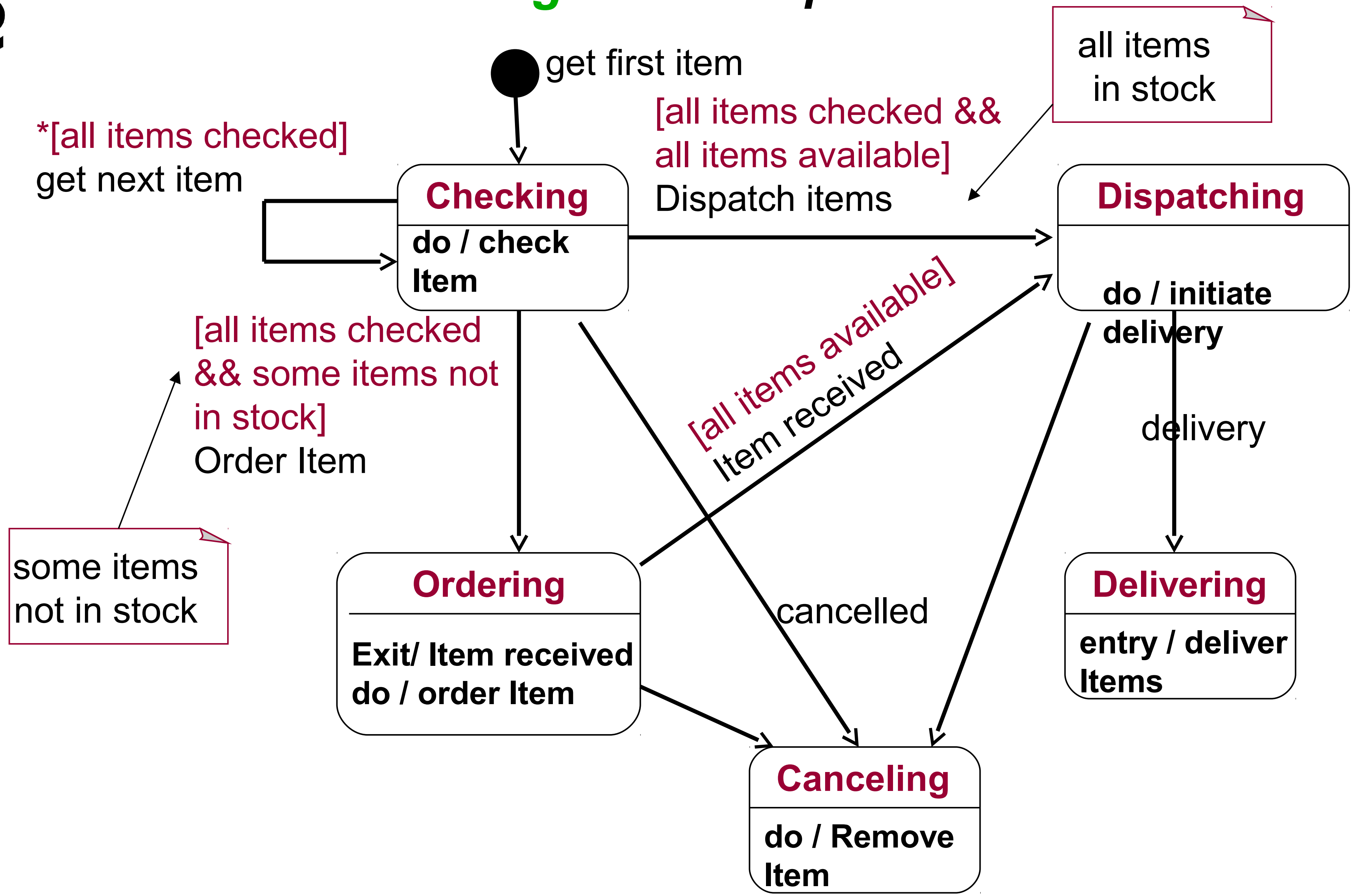


Exit action : any action that is marked as linked to the exit action is executed whenever the state is left via a transition

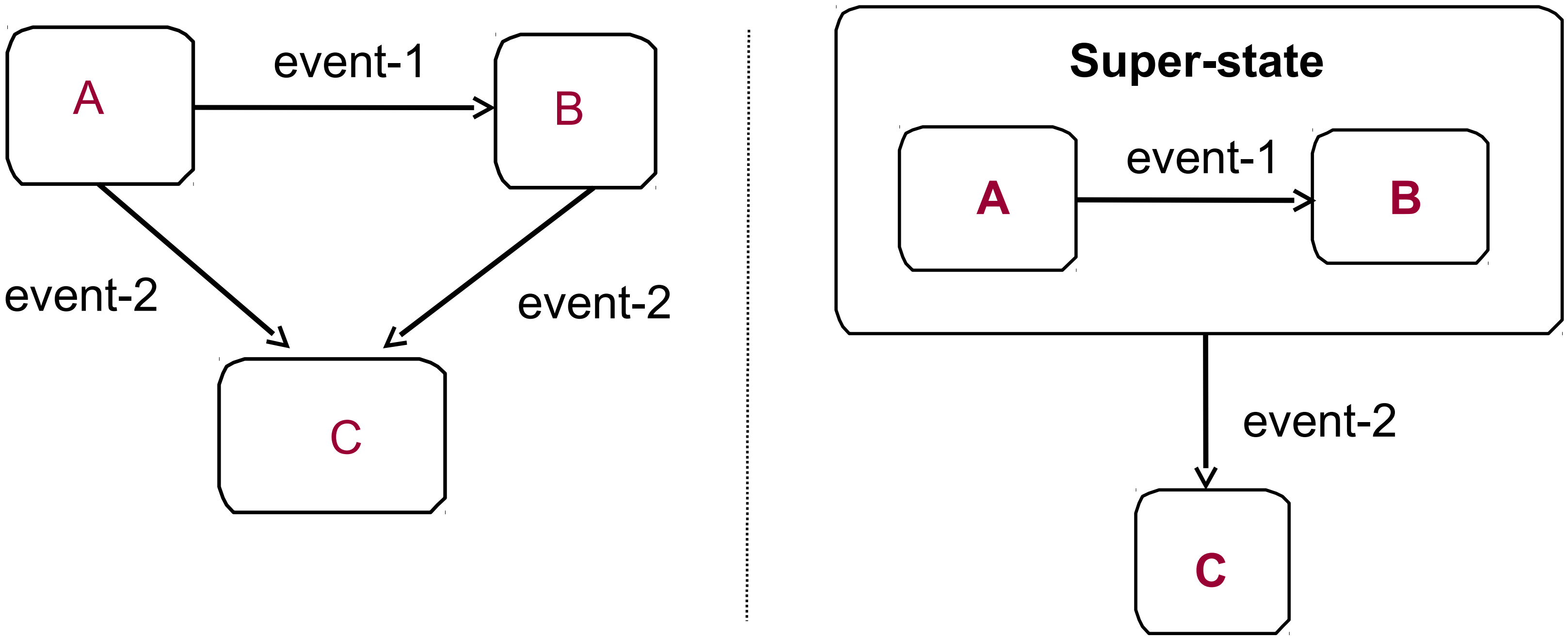
A Class of BookCopy



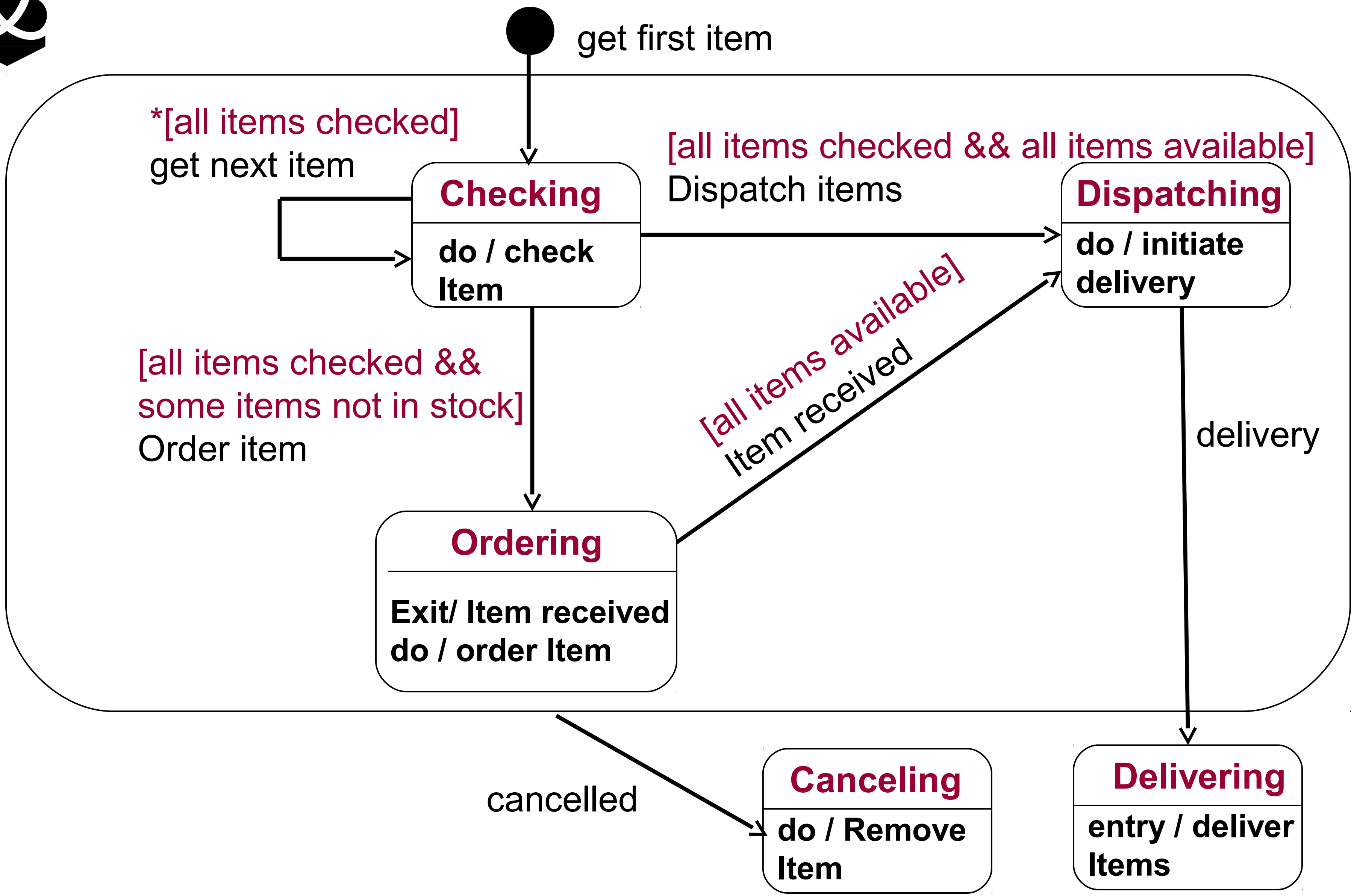
State Diagram Example



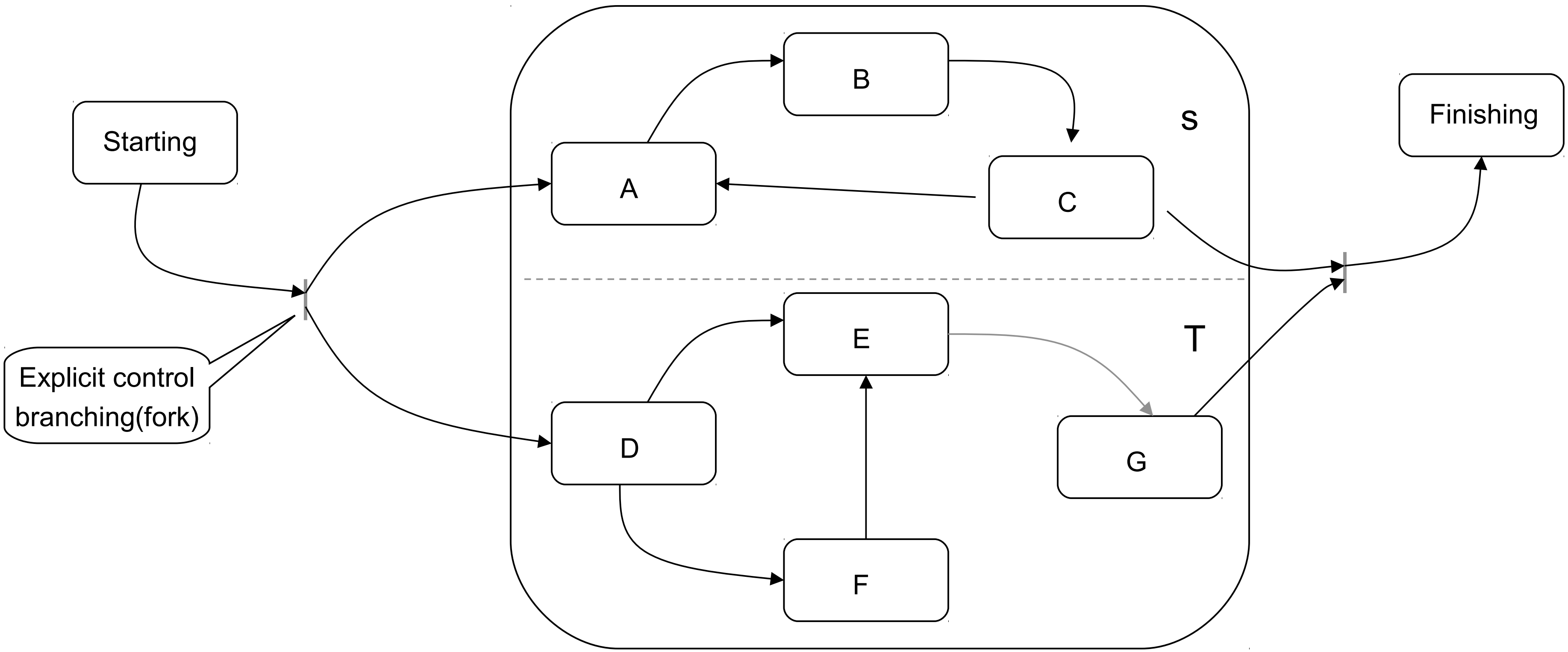
State Diagram - Nested States



State Diagram Example including substates



Concurrent State models



- **Orthogonal Components and Concurrency**
  - **shown separated by dashed line**
    - **supports concurrency**
- **Objects must be in only one state from each of the orthogonal components**

Concurrent State Models

Three different ways for orthogonal components to communicate:

- **Broadcast Events**
- **Propogated Events**
- **IN operators**

Concurrent State Models

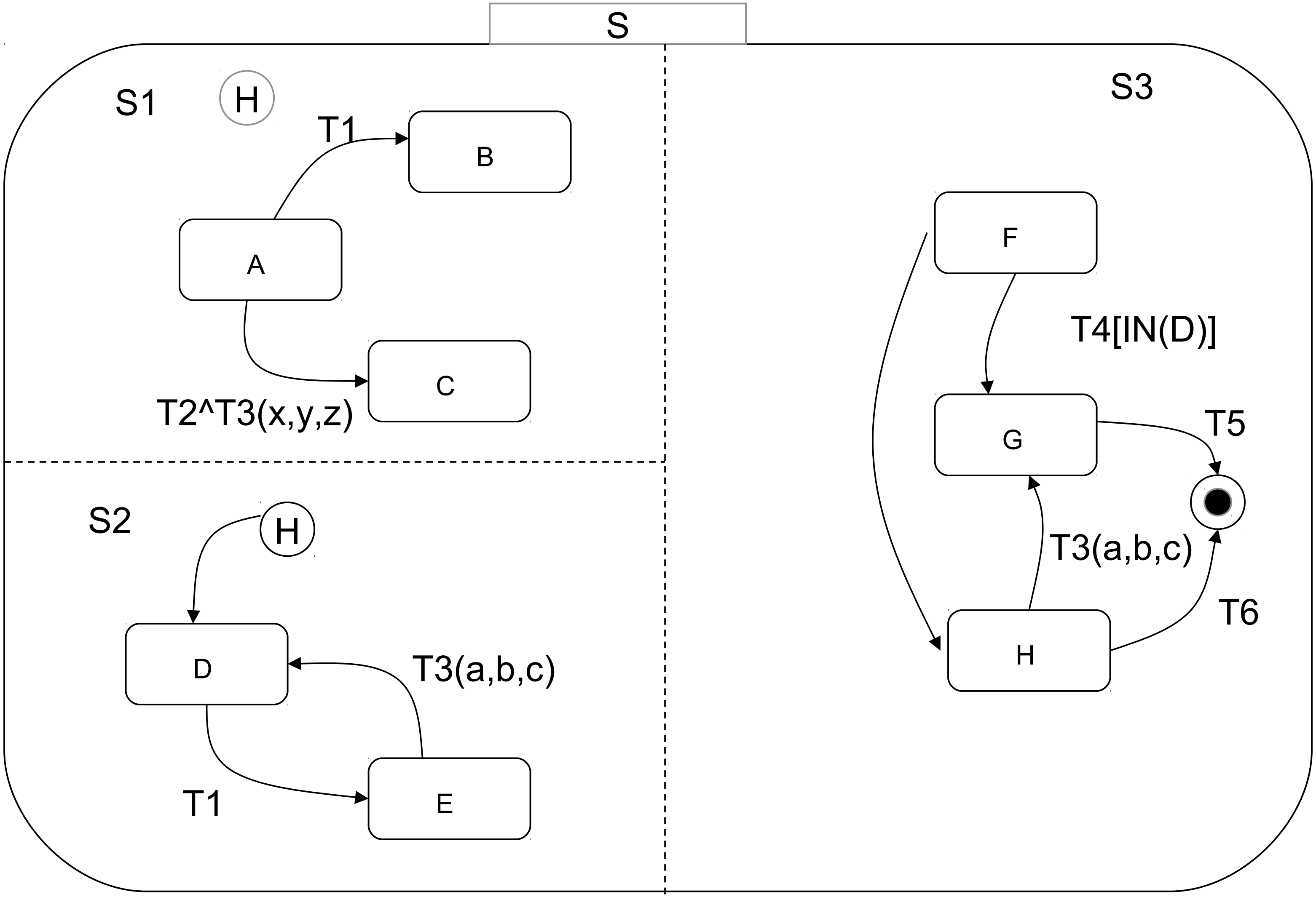
- Broadcast events
  - events that more than one orthogonal component accepts
  - For example an event T1 is sent to all active orthogonal components it need not be acted on by all components
    - what happens if component S1 is in state A, S2 is in state E and S3 is in state G when a T1 event occurs?
    - what happens if S1 is in state A and S2 is in state D when the event T1 occurs?

Concurrent State Models

- Propagated events are indicated with the caret following the event name (and optional parameters and guard)

- IN operators are used as a guard on transition T4. This allows the S3 component to take the transition T4 only if S2 is currently in state D

Concurrent State Models



Summary

In this chapter we have:

- Introduced the terms used with respect to state diagrams
- Discussed the context in which state diagrams are used
- Introduced substates
- Discussed concurrent state diagrams