COMP 3111
SOFTWARE ENGINEERING

LECTURE 17
SYSTEM ANALYSIS AND DESIGN

STATE MACHINE DIAGRAM

A ***state machine diagram*** describes the behavior **inside** an object.

(What an object does when it receives a message.)

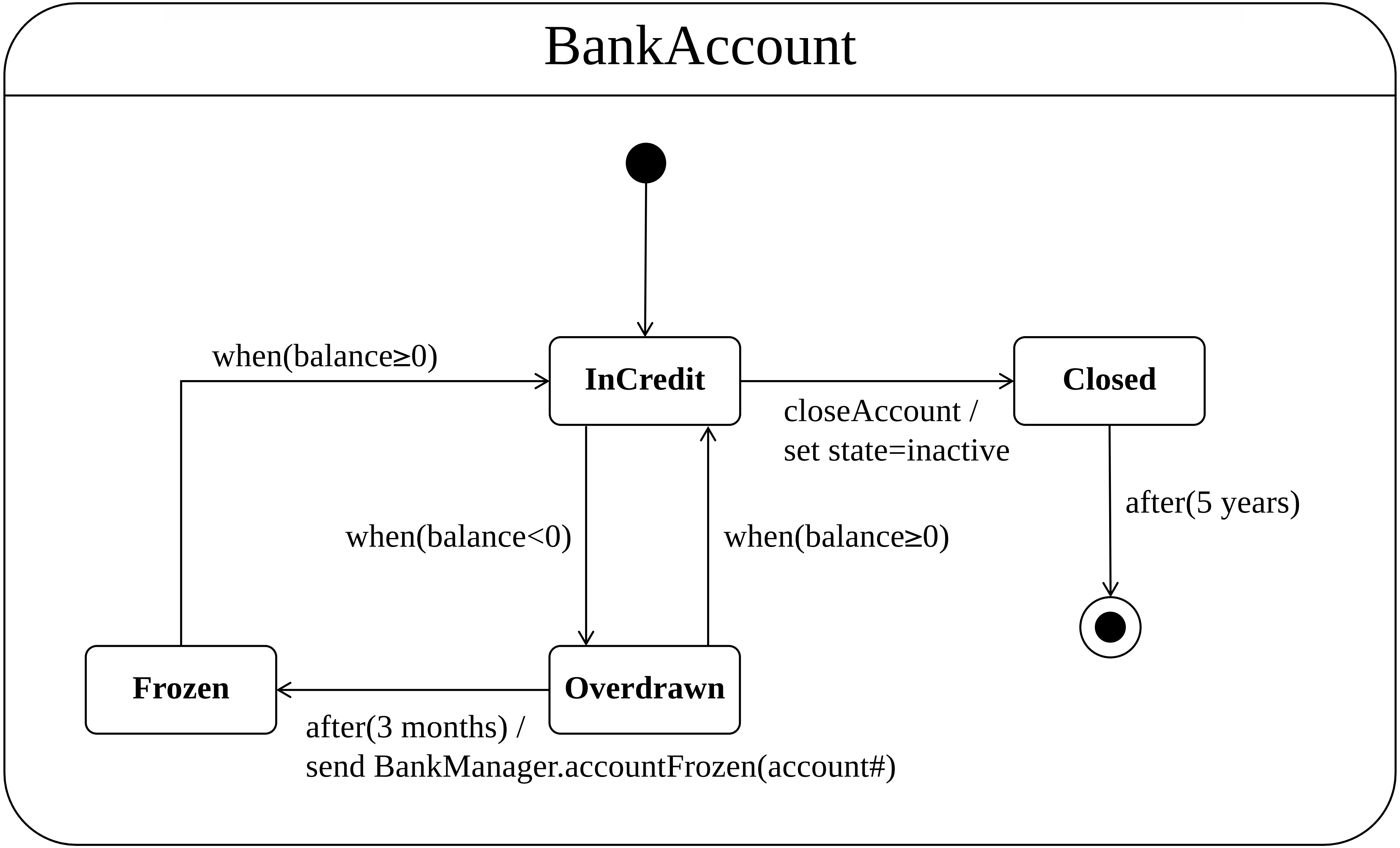
- It is a directed graph that shows:
  - the states of a single object (nodes).
  - the events that cause state changes (arcs).

- It shows all the messages that an object can send and receive.

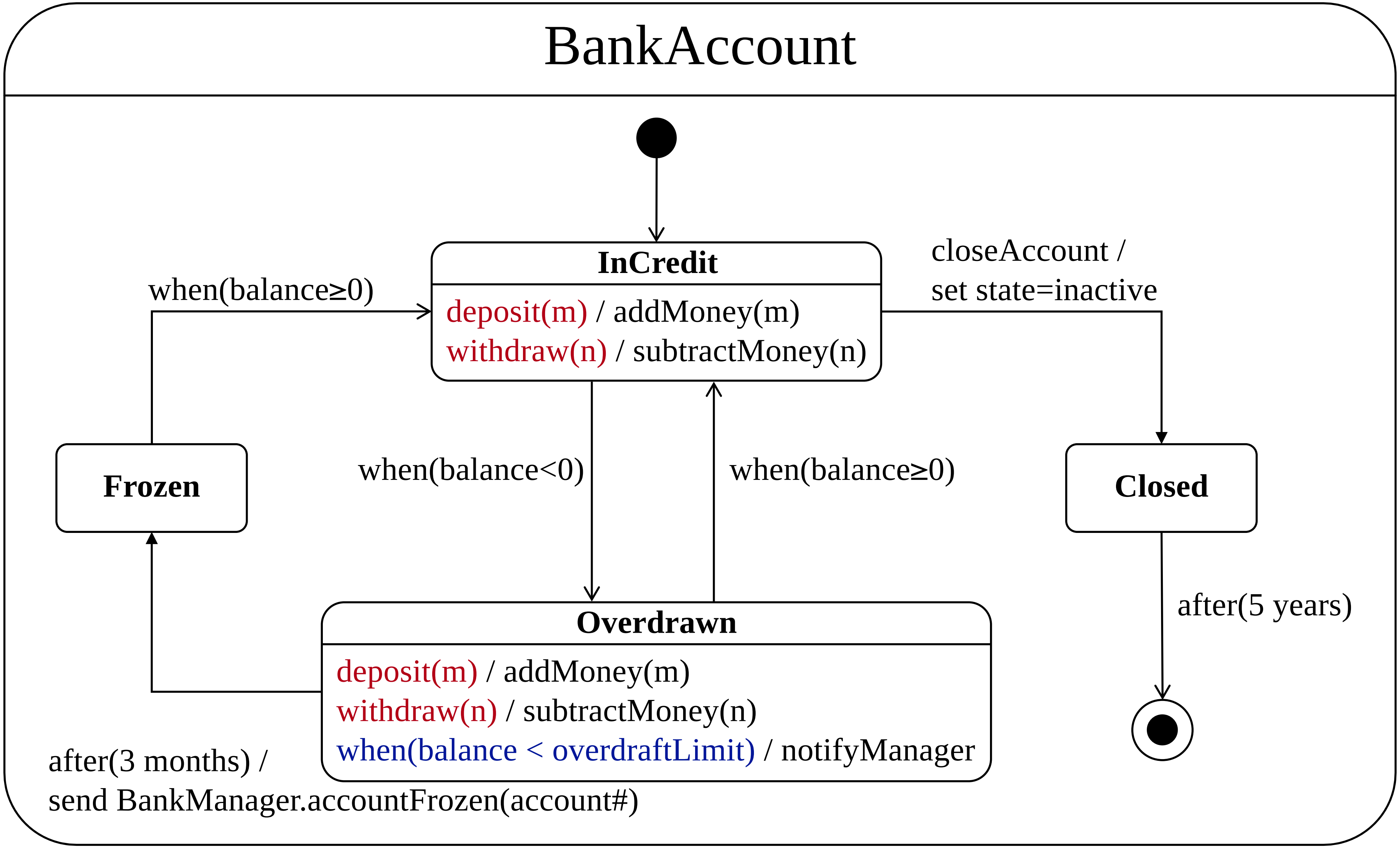
- It describes all the possible states an object can get into during its life time.

- It is drawn for a single class to show the lifetime behavior of a single object.

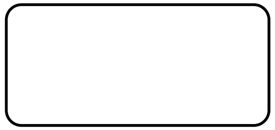
EXAMPLE STATE MACHINE DIAGRAM



EXAMPLE STATE MACHINE DIAGRAM



STATE MACHINE DIAGRAM: STATE



A ***state*** is a time during the life of an object when it satisfies some condition, performs some action or waits for an event.

☞ **A state has duration.**

- A state may be characterized by the:
  - value of one or more attributes of the class (e.g., a BankAccount object can be overdrawn or in credit based on the value of its *balance* attribute).
  - existence of a link to another object (e.g., a BankAccount object may be an individual or joint account based on the existence of links to one or two Customer objects).

STATE MACHINE DIAGRAM: SPECIAL STATES

● **initial state (start state)**

☞ **Each diagram must have at most one initial state.**

◉ **final state (end state)**

☞ **Each diagram can have multiple final states.**

☞ **No initial or final state indicates looping behaviour.**

- States may be named. **InCredit**
- States may be unnamed (called anonymous states) ▭

STATE MACHINE DIAGRAM: TRANSITION

⟶ A ***transition*** is a change of state from an originating state (source state) to a successor state (target state).

☞ **The source and target states may be the same state.**

☞ **A transition takes zero time and cannot be interrupted.**

- Transition adornments (*all are optional*) include:

  event trigger [guard condition] / effect list

  – **event trigger** → an event name plus optional parameters.
    - ➢ An *event trigger* is the (implicit) event that causes a transition to occur.
    - ➢ The event is said to **trigger** the transition; the transition is said to **fire**.

  – **guard condition** → a Boolean expression which must be true for the transition to fire.

  – **effect list** → an atomic procedural expression executed if and when a transition fires.

STATE MACHINE DIAGRAM: TRANSITION (CONT'D)

event trigger

- The parameters of an event are available within effects specified on the transition or within activities initiated in the target state.

guard condition

- The condition is written in terms of parameters of the triggering event and attributes and links of the object.
- A guarded transition only fires when its event occurs and if the condition is true.

effect list

- An effect list may contain more than one action, which may include call, send, and other kinds of actions.
- An effect list is written in terms of operations, attributes and links of the source object and parameters of the triggering event.

STATE MACHINE DIAGRAM: EVENT

An *event* is something that happens at an instantaneous point in time.

Event types

call event – the receipt of a *synchronous call* from an object (a request that an operation be performed)

change event – a specified Boolean condition becoming true → when(balance < 0)

time event – absolute time → when(date=07/03/2009)
– elapsed time → after(10 seconds)

signal event – the receipt of an *asynchronous communication* from an object

STATE MACHINE DIAGRAM: EVENT TYPES

call event

- The event trigger specifies the operation and the parameters of the event trigger are the parameters of the operation.

change event

- A change event occurs whenever the value of a designated condition (expression) changes from *false* to *true.*
- All values in the Boolean expression must be attributes of the state machine's object; there are no parameters of the event.
- *A change event implies a continuous test for the condition.*

  Note: A change event is not the same as a guard since a guard is passive (evaluated only when it is encountered) while a change event is active (evaluated continuously).

signal event

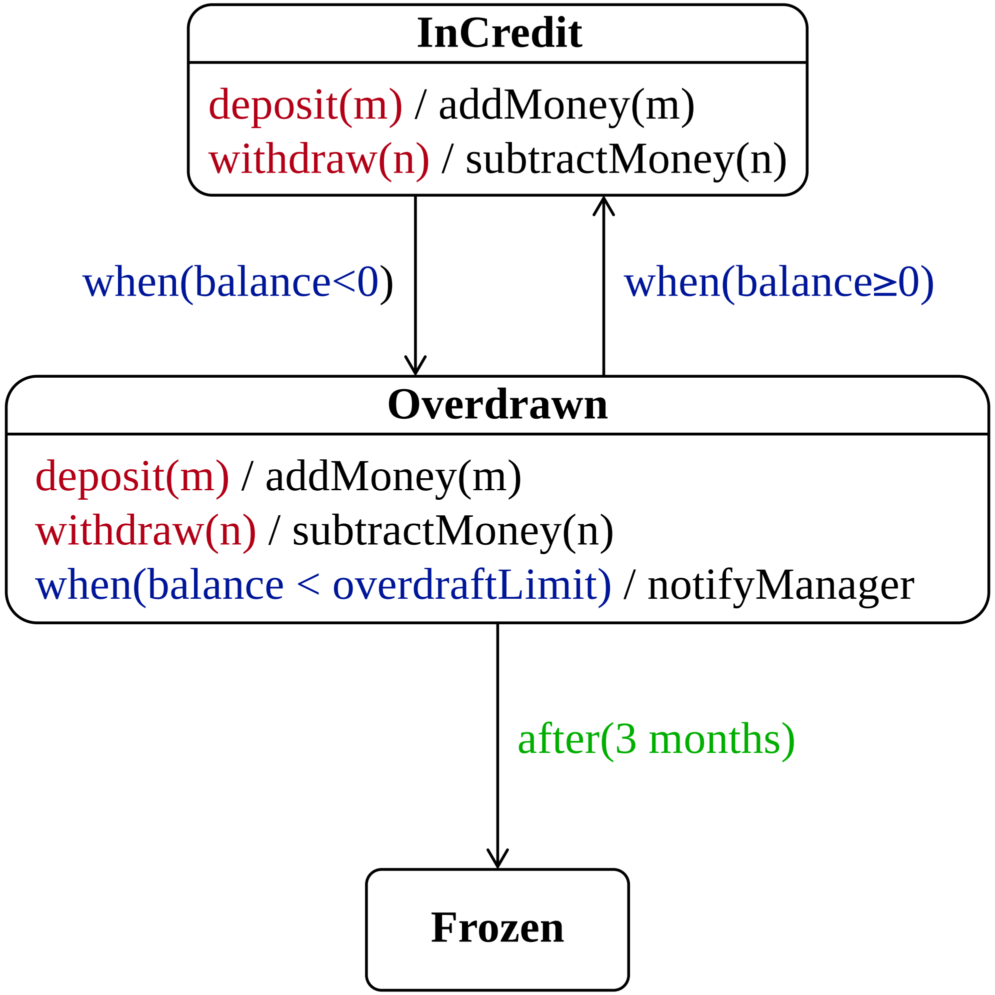
- A signal event is modeled as a stereotyped class («signal») and designates that a package of information is sent asynchronously between objects.
- It is most often used in real-time systems specification.

STATE MACHINE DIAGRAM: EVENT TYPE EXAMPLES

Call event

Change event

Time event



STATE MACHINE DIAGRAM:
EVENT / TRANSITION MECHANICS

- Events are processed one at a time.

  ☞ **If the event does not trigger any transition it is ignored.**

- Only one transition within a state machine diagram may fire.
  - If two transitions can fire, then the choice may be nondeterministic (i.e., a race condition) → *This is probably a specification error!*

  ☞ **All state transitions must correspond to different events.**

- There are two ways to transition out of a state:
  - **automatic** - when the activity of the state completes.

    ☞ **Transitions without labels fire immediately when the state activity, if any, completes.**

  - **non-automatic** - caused by an event.

STATE MACHINE DIAGRAM: ACTIONS & ACTIVITIES

action - instantaneous and cannot be interrupted

☞ Processing for transitions between states and entry/exit of a state.

activity - takes time to complete and can be interrupted

☞ Processing that occurs while in a state.

- Possible forms of state behavior:

  no behavior → wait until an event occurs that exits the state.

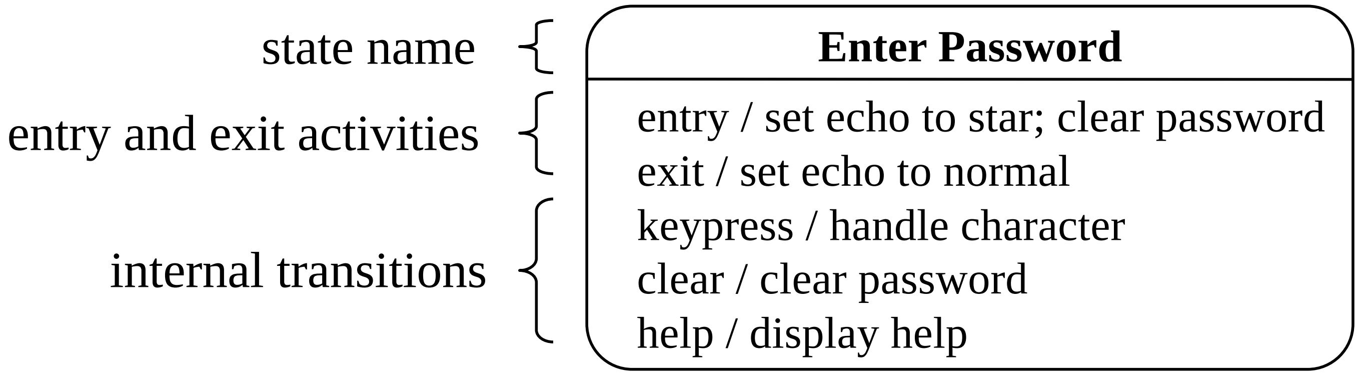
  event trigger [guard condition] / effect list → when *event trigger* occurs and *guard condition* is true, *effect list* is performed.

  do / activity → ongoing execution of behaviour (e.g., operations).

  entry / activity → performed every time the state is entered.

  exit / activity → performed every time the state is exited.

STATE MACHINE DIAGRAM:
ACTIONS & ACTIVITIES EXAMPLE



state name

entry and exit activities

internal transitions

Enter Password

entry / set echo to star; clear password
exit / set echo to normal
keypress / handle character
clear / clear password
help / display help

ASU COURSE REGISTRATION (REVISED)

At the beginning of each term, students may request a course catalogue containing a list of course offerings needed for the term. Information about each course, such as instructor, department, and prerequisites are included to help students make informed decisions.

The new system will allow students to select four course sections for the coming term. In addition, each student will indicate two alternative choices in case a course section becomes filled or is canceled. No course section will have more than forty students or fewer than ten students. A course section with fewer than ten students will be canceled. Once the registration process is completed for a student, the registration system sends information to the billing system so the student can be billed for the term.

Professors must be able to access the online system to indicate which courses they will be teaching, and to see which students signed up for their course offerings.

For each term, during the registration period that students can change their schedule. Students must be able to access the system during this time to add or drop courses.

ASU STATE MACHINE DIAGRAM: SECTION CLASS

- To construct a state machine diagram we ask the following questions:
  - What states can the class be in?
  - What determines the state that the class is in?
  - To what events (messages) does each state respond and what happens when the event occurs?

The solution for the Section class state machine diagram is included in the solution to the lecture exercise.

COMPOSITE STATE MACHINE DIAGRAM

A ***composite state machine diagram*** contains one or more nested state machine diagrams.

sequential composite state machine diagram

- An object is in exactly one state in one of the (nested) state machine diagrams.

  ☞ This corresponds to an or-condition on all state machine diagrams.

- It is used to abstract/generalize states.

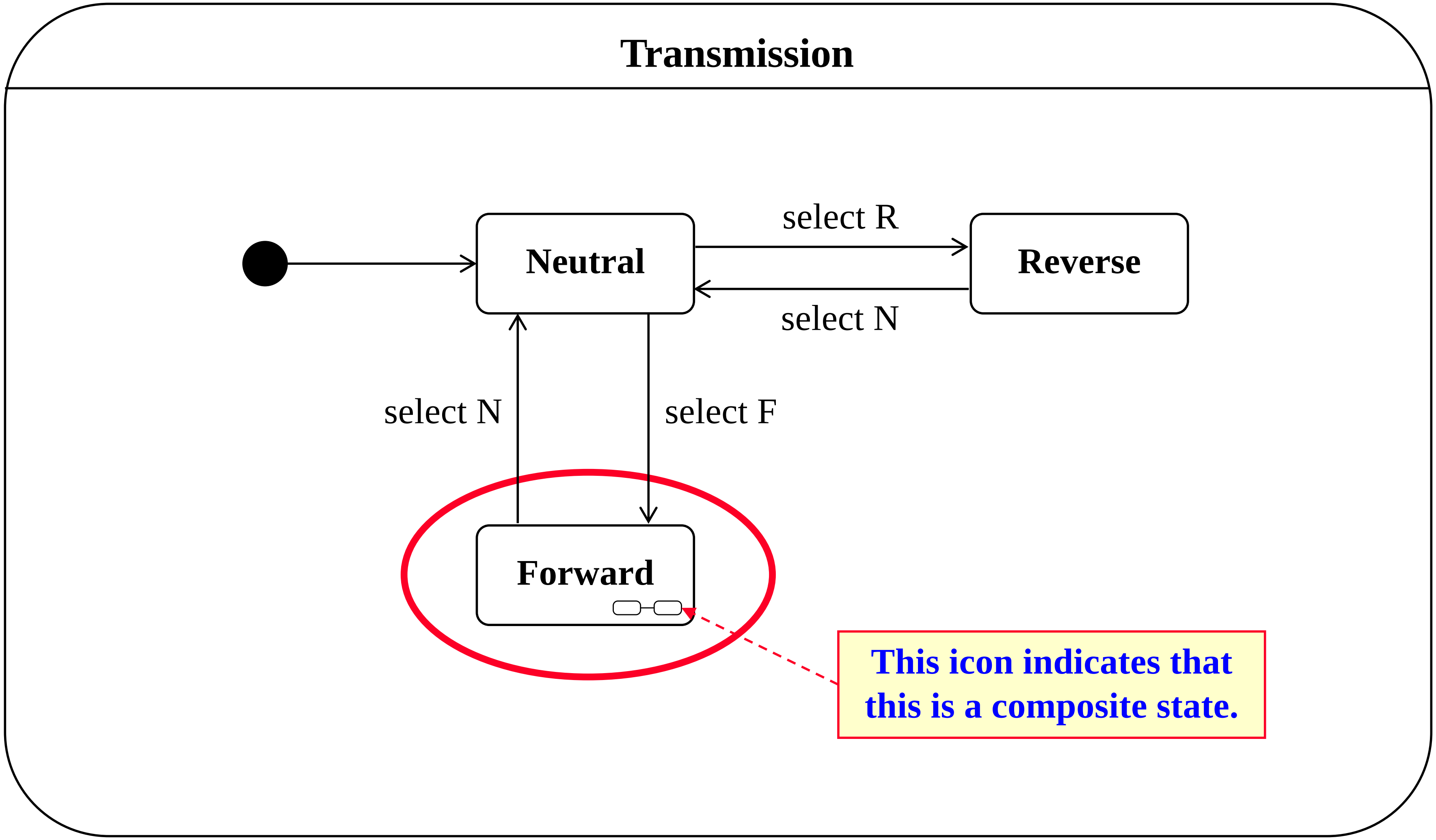
concurrent composite state machine diagram

- An object is in exactly one state in each of the regions of the state machine diagrams.

  ☞ This corresponds to an and-condition on all regions.

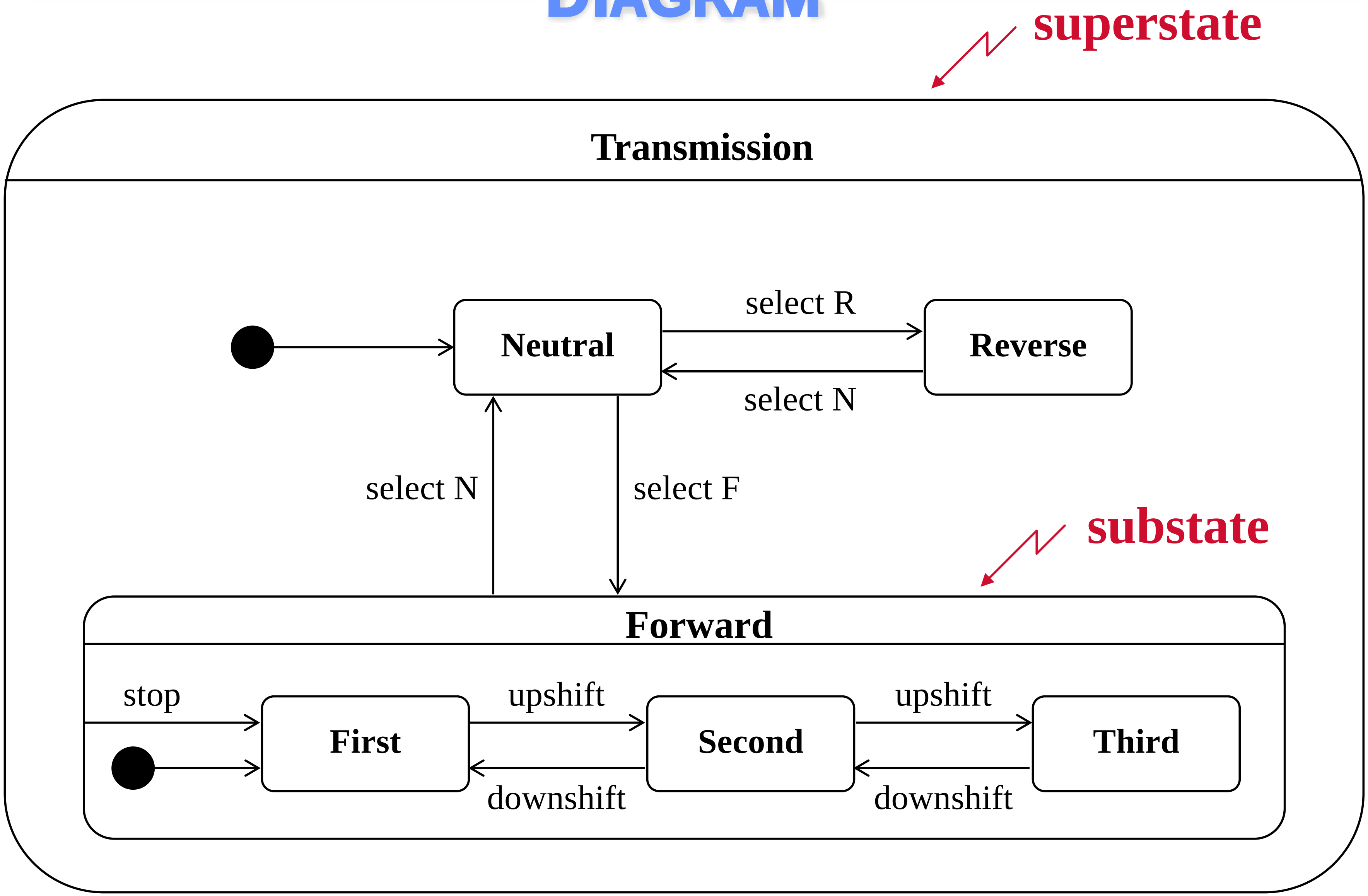
- It is used to show multi-threading behavior.

SEQUENTIAL COMPOSITE STATE MACHINE DIAGRAM



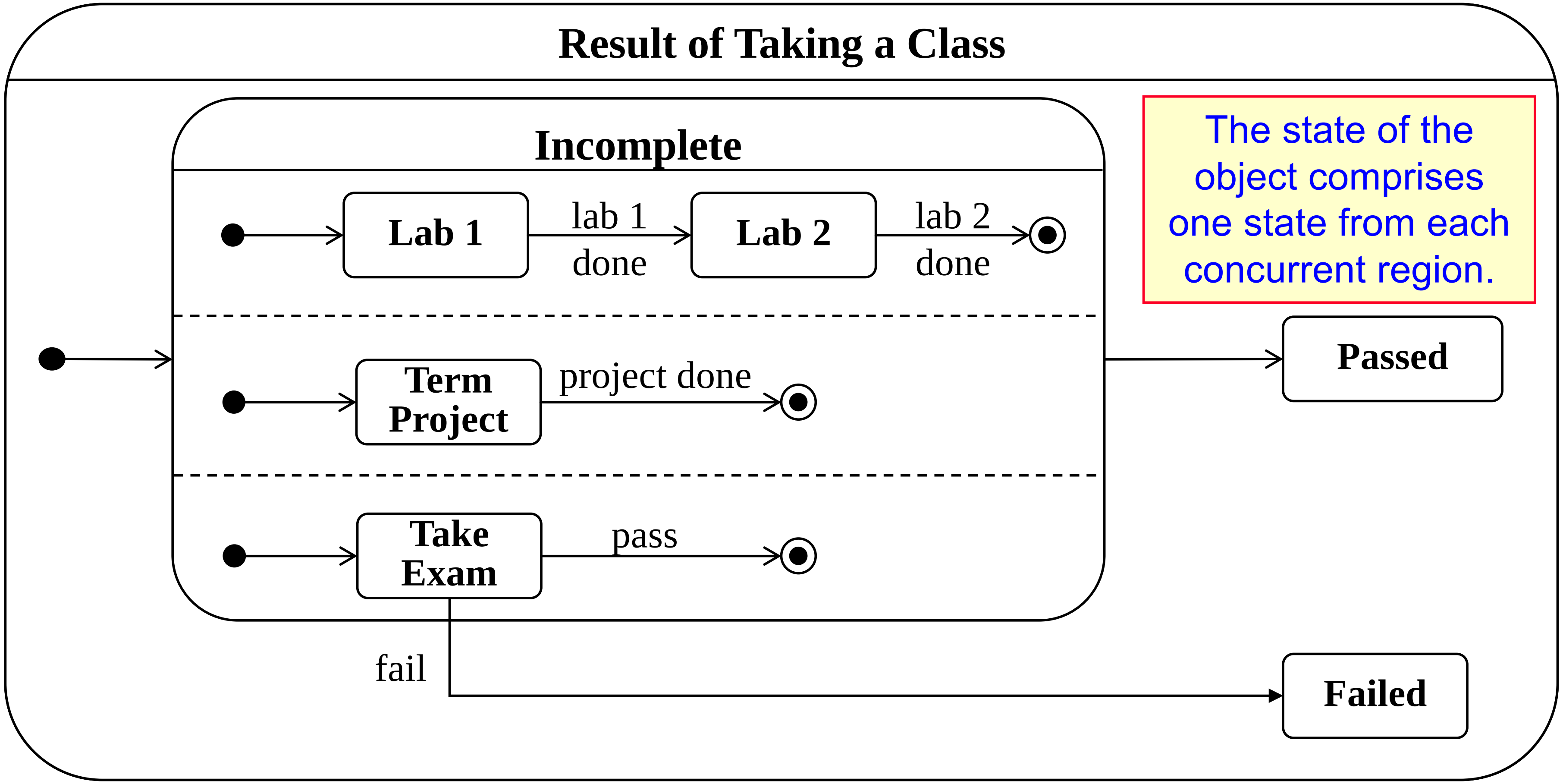
This icon indicates that this is a composite state.

SEQUENTIAL COMPOSITE STATE MACHINE DIAGRAM



CONCURRENT COMPOSITE STATE MACHINE DIAGRAM

- Concurrency arises when an object can be partitioned into sub-sets of attributes or links, each with its own state machine diagram.



The state of the object comprises one state from each concurrent region.

COMPOSITE STATE MACHINE DIAGRAM: TRANSITIONS

- A transition to the boundary ≡ a transition to the initial state.

  ☞ **The entry activity of all regions entered are performed.**

- There may be transitions directly into a composite state region.

- A transition from the boundary ≡ a transition from the composite state.

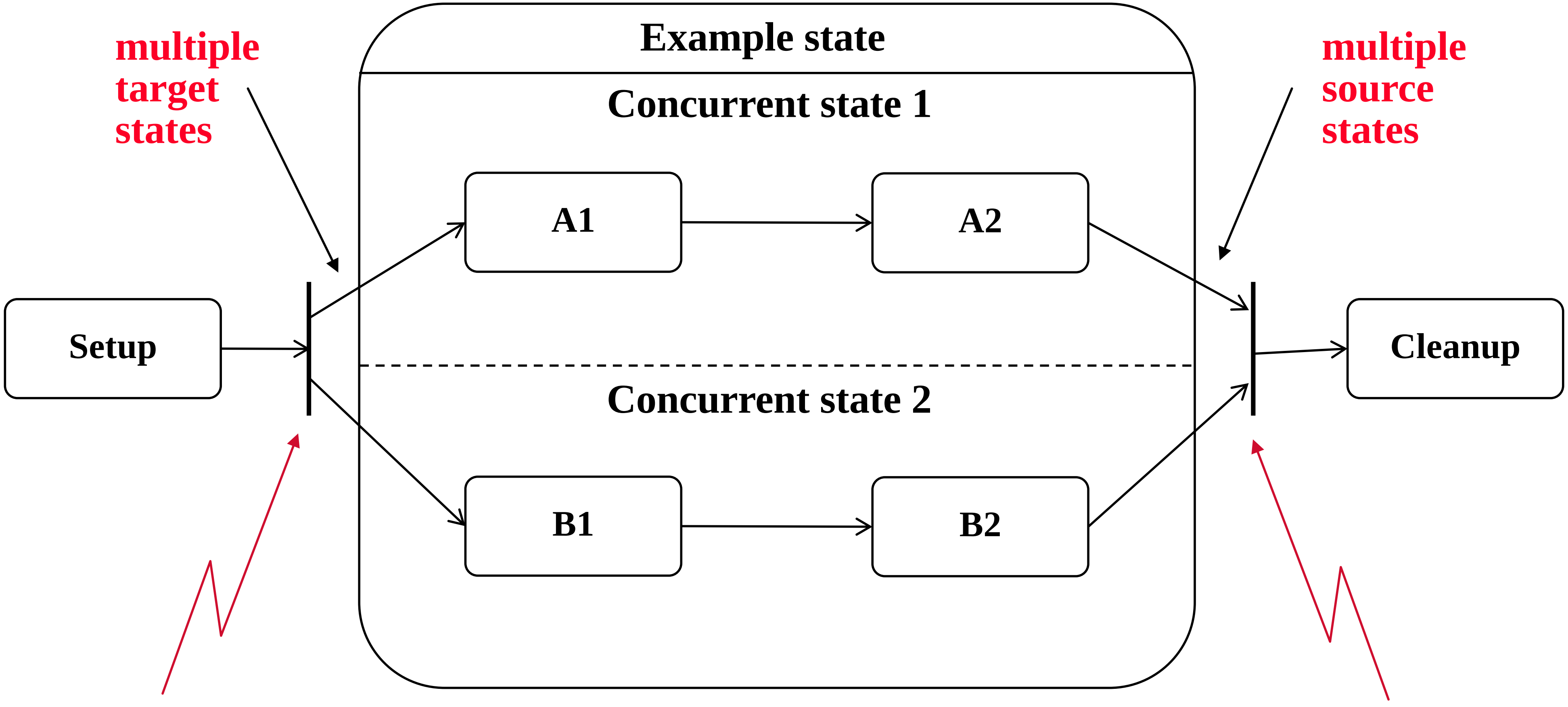
  ☞ **The exit activity of all regions exited are performed.**

- There may be transitions directly from within a composite state region to an outside state.

- A transition may have multiple source and target states.

  ☞ **This represents a synchronization or splitting of control.**

CONCURRENT COMPOSITE STATE MACHINE DIAGRAM: SYNCHRONIZATION



multiple target states

multiple source states

synchronization bar (fork)

synchronization bar (join)

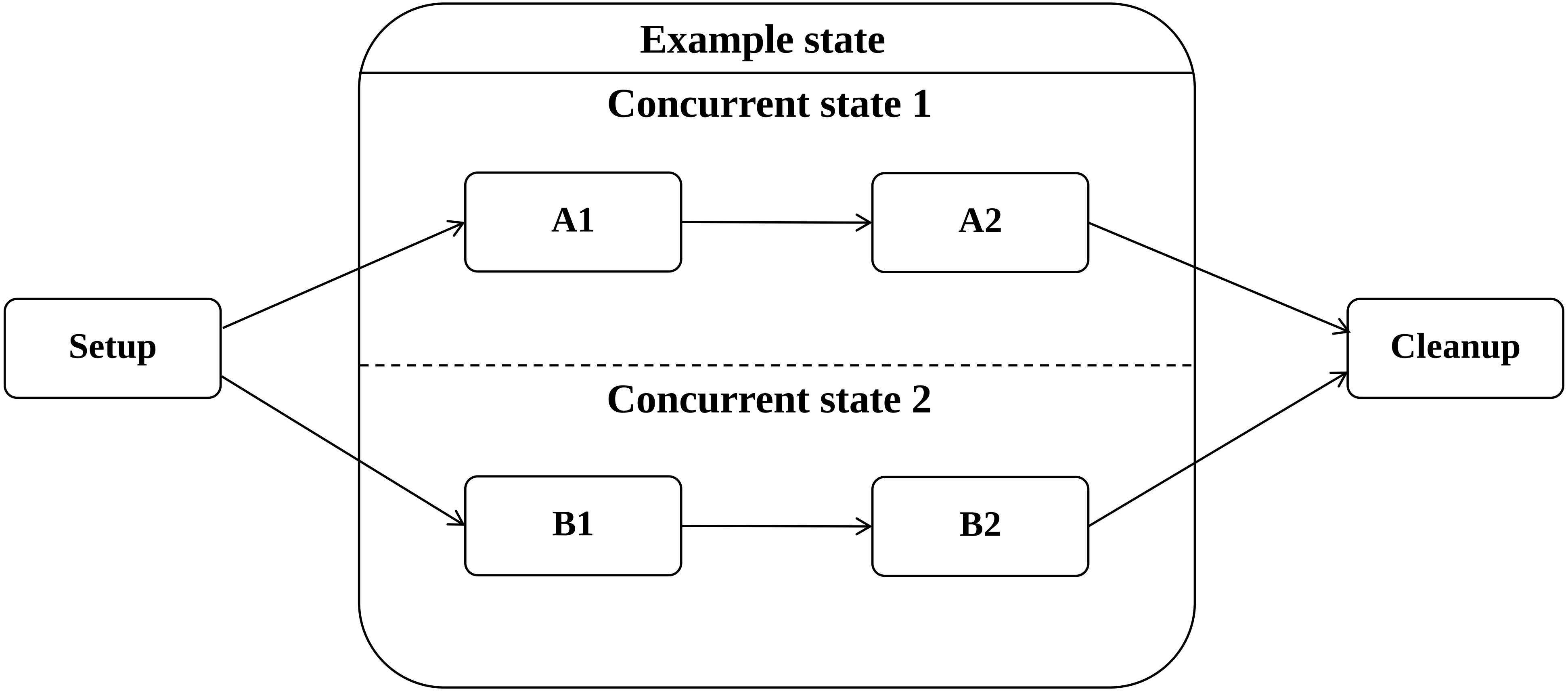
CONCURRENT COMPOSITE STATE MACHINE DIAGRAM: SYNCHRONIZATION (CONT'D)

- For multiple target states, after the transition out of the source state fires, *all* of the target states are enabled (e.g., after the transition out of Setup fires, then A1 and B1 are enabled).

- For multiple source states, after the transitions out of *all* of the source states fire, the target state is enabled (e.g., after A2 and B2 complete, then Cleanup is enabled).

CONCURRENT COMPOSITE STATE MACHINE DIAGRAM: SYNCHRONIZATION (CONT'D)

Is there a difference in execution between this state machine diagram and the previous one?



WHEN TO USE A STATE MACHINE DIAGRAM

- A state machine diagram is good at describing the behavior of an object across several use cases.

☞ **It is not necessary to produce a state machine diagram for every class.**

- A state machine diagram should be used only for classes with **significant** dynamic behaviour.