

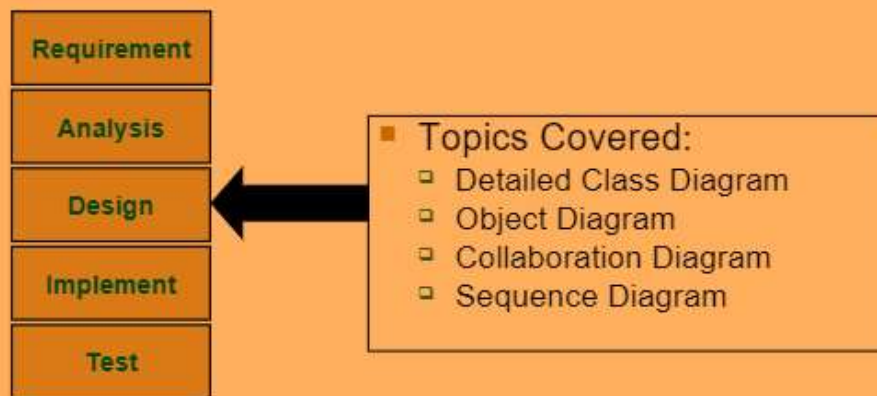
Lecture 8

Thursday, March 9, 2023 3:36 PM


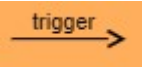
Overview of This Lecture

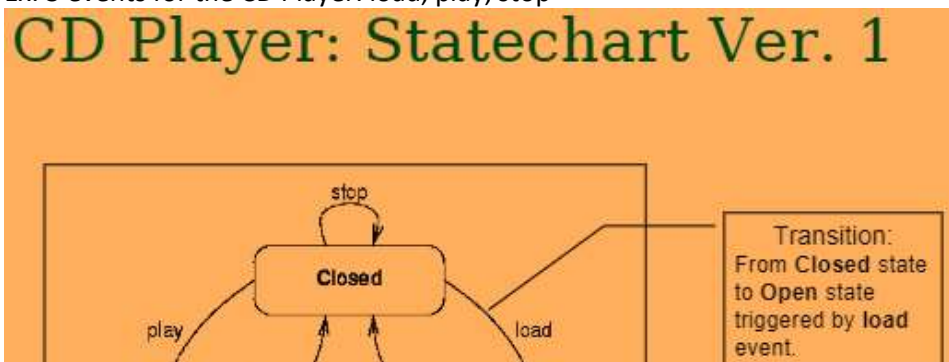
- State diagrams (a.k.a. statecharts):
 - State-dependent behaviors, events, transitions
 - Initial and final states, guard conditions
 - Actions, activities, composite states, history states

Where are we now?

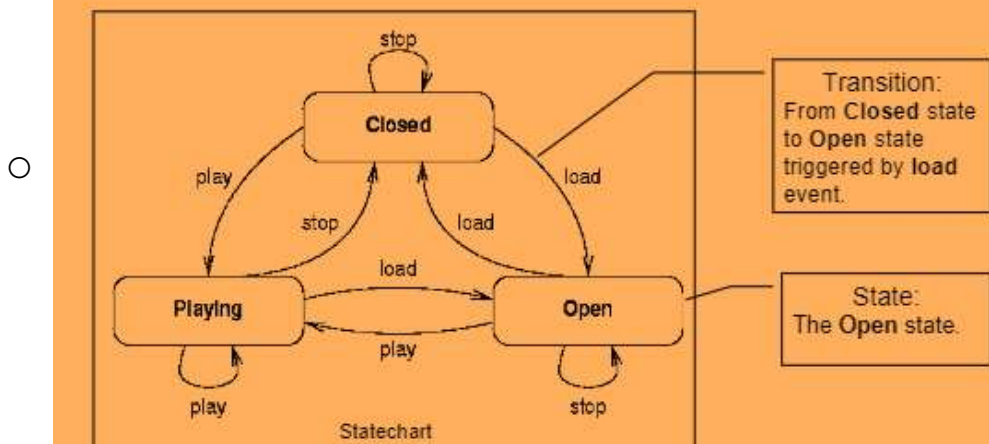


- Object Behavior
 - Behavior of object is defined by its reaction to messages at any point in execution
 - Some object behaviors can be studied by object diagram or interaction diagram
 - Not sufficient tho bc
 - Cant model all possible scenarios, like only specific sequence of messages is studied
 - Only model some legal (possible) states – show how an object behaves in certain interactions
 - Need to know abt illegal/impossible states to plan for them
- Specifying Behavior in UML
 - Diff notation needed to summarize overall behavior of objects
 - UML defines statecharts aka state diagrams, made just for this
 - Complement the interaction diagrams for understanding the dynamic behavior of the sys
 - Interaction diagram: models some inter-object messages with well-defined order during a short duration
 - Statechart: models entire lifetime of single object, specifying all possible sequences of messages and responses
- State-dependent Behavior
 - Objects respond differently to same stimulus/events at diff times

- Behavior can be modelled by defining a state machine
 - It has a set of *states*: an object can be in one state at any time, the state it is in determines how it responds to events
 - It has a set of *transitions*: an event can cause the object to move from one state to another
- EX: CD Player
 - Behavior of simple CD player:
 - Drawer to hold the CD
 - Ctrl interface w/ 3 buttons:
 - Load button: drawer will open if it was shut, will shut if it was open
 - Stop button: player will stop playing (if no CD, than no effect)
 - Play button: CD will be played, if drawer is open, than drawer shuts before playing starts
- UML Statechart Semantics
 - Statechart defines the behavior of instances of a class
 - Shows:
 - Possible states of an object
 - Events it can detect
 - Response to those events
 - Object is in one active state at a time:
 - Events may be received at any time, can trigger a transition to the next active state
 - If an Event only causes a loop on a state, the transition is known as self-transition
- Statechart: Deciding the States
 - ID separate states: Informal principle: states S1 and S2 are separate if an object in state S1 responds differently to at least one event from the way it responds to that event in state S2
 - Syntax: states: rounding rectangles with the name of the state (
 - 
 -)
 - Ex: 3 states for the CD player: closed, player, open
 - Events are usually external stimulus
 - Ex: messages that can be sent to an object
 - *Internal stimulus* will be covered later in the lecture
 - Events usually cause an object to change state
 - Moving from one state to another known as transition
 - Events that cause transitions are known as trigger
 - Events can optionally carry data (like message parameter)
 - Syntax: arrow w/ trigger attached
 - 
 - Ex: 3 events for the CD Player: load, play, stop

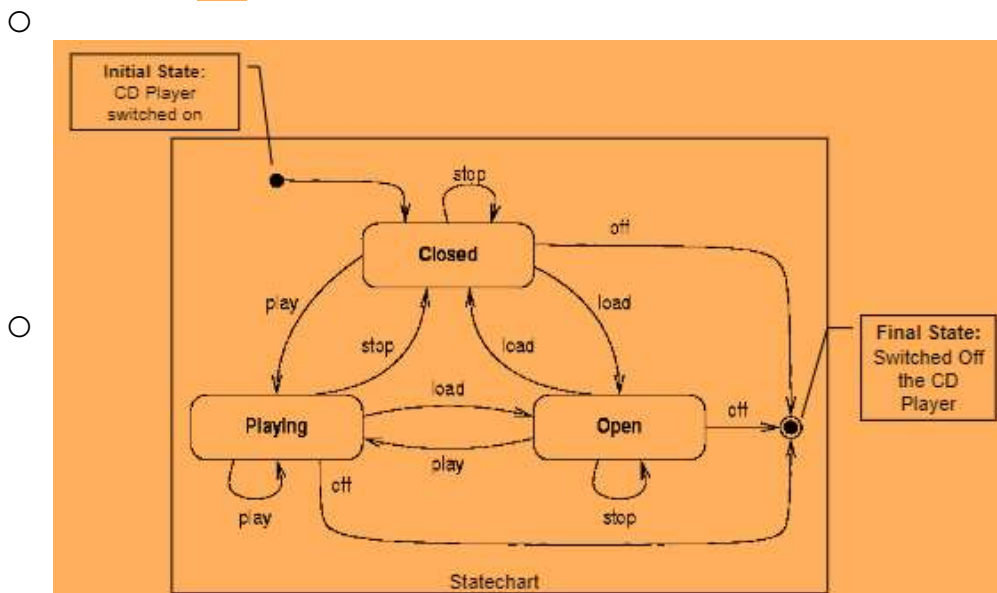


CD Player: Statechart Ver. 1



- Initial and Final States

- To specify the 1st active state when an object is made/initialized: Use Initial
 - A transition leading from the initial state indicates the first active state
 - No event should be written on a transition from an initial state
 - Syntax: a black disc
 -
- To model destruction of an object: use final state:
 - Rep the end of flow (ex the object don't exist any longer)
 - May correspond to actual destruction for software object
 - Syntax: a circled black disc:
 -

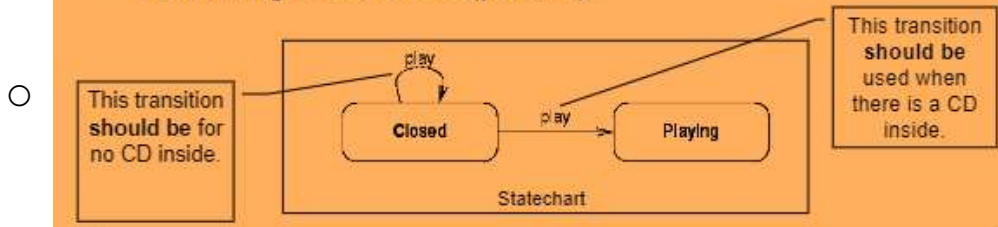


- Non-Deterministic System

- When 2/more transitions leading from a state share the same trigger
 - No way to distinguish btwn them
 - Rando choose one of them
 - Same history of events may end up in a diff state
 - Known as non-deterministic sys

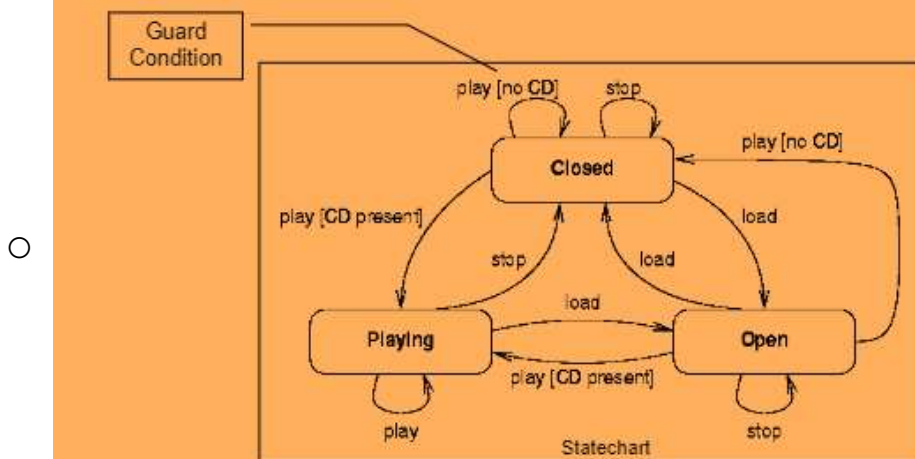
- Can be removed by adding info to distinguish btwn transitions w/ the same trigger
- CD Player: Non Deterministic Ex:
 - Irl, a cd player won't play if no CD inside
 - Chart no. 2 don't describe this
 - Correct behavior:
 - When play even detected, the CD player should remain in the Closed state if there is no CD in

■ Resulting Statechart (partial):



- Guard Condition
 - Tho 2 transitions meant for diff scenarios, the statechart don't distinguish among the 2, resulting in a non-det behavior
 - Non-det can be removed if we can be removed if can specify the condition that determines the correct transition to use
 - Syntax: trigger[guard_condition]
 - Semantics:
 - If a transition has a guard condition, it can only fire if that transition is evaluated to true;
 - If all guard conditions are false & there's not unguarded transition, the event will be ignored
 - Usually only 1 condition is true (determinism)

CD Player: Statechart Ver. 3

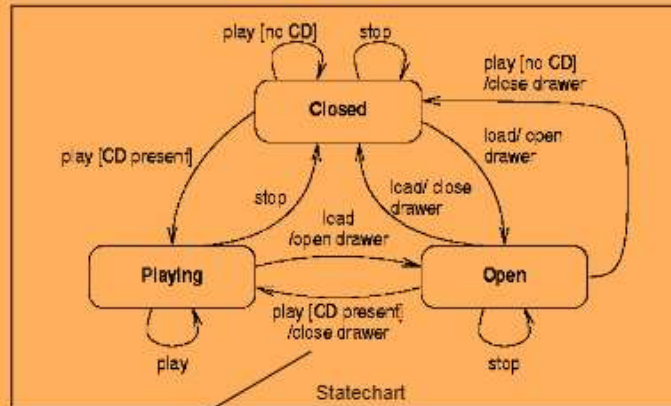


■ Note that guard conditions are also added to transitions leading from the **Open** state. What do they represent?

- Actions
 - A state can have actions triggered
 - 3 ways action can be triggered:
 - Event
 - Entering a state

- Leaving a state
- Action triggered by an event takes place in response to that event:
- Syntax: Trigger[Guard_Condition]/action
- Ex: CD drawer will be closed when play is detected in the Open state
-

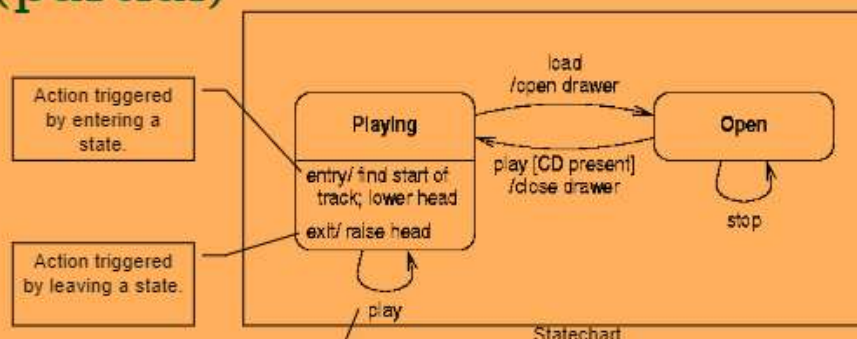
CD Player: Statechart Ver. 4



Action triggered by Event

- Action: Entering/Exiting a State
 - Action can be triggered as soon as the state is entered
 - Useful to capture actions that must be performed regardless the transition used to arrive at the state
 - Syntax: write in a compartment in the state box: entry/action
 - Similarly, action can be performed just b4 leaving a state
 - Syntax: write in a compartment in the state box: exit/action

CD Player: Statechart (partial)

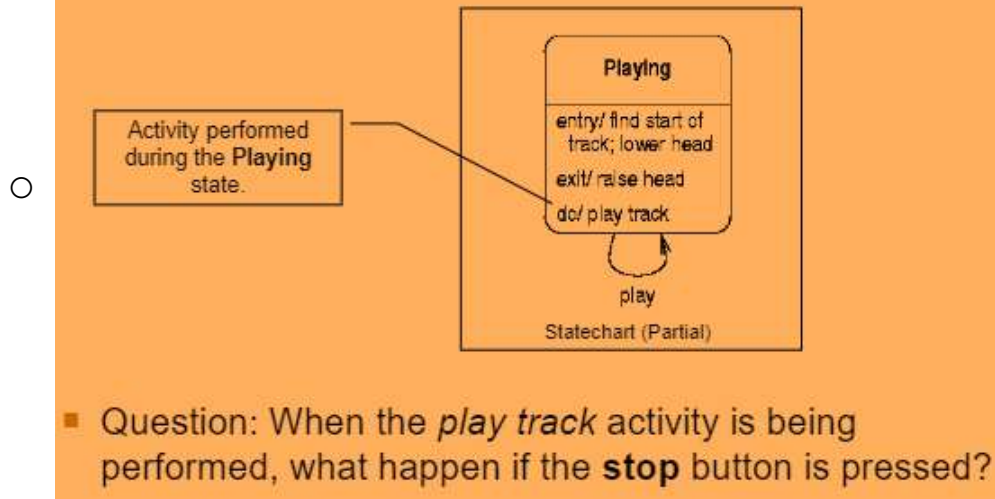


Question:
What happened when the **play** button is pressed during CD playing?

- Action and Activity
 - Properties of Actions:
 - Short, self-contained processing

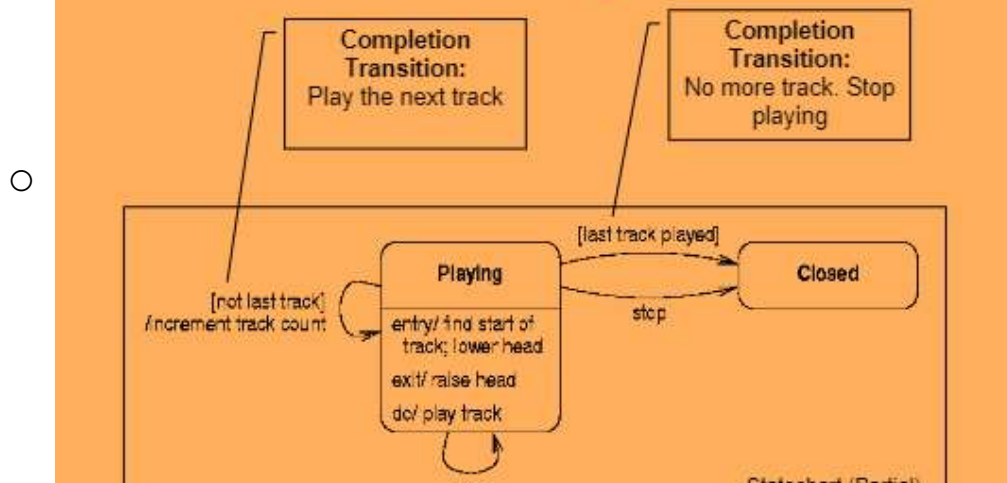
- Finished "instantaneously"
- Cannot be interrupted by events
- An "action" that doesn't conform to the above is termed as activity instead
- Property of activities:
 - Performed during a state;
 - Carry out for an extended period of time
 - Can be interrupted by events
- Syntax: Write in a compartment of the state box: do/activity

CD Player: Activity Example



- Complete Transition
 - Along w/ interrupted by events, some activities will come to an end of their own accord
 - If an activity completes uninterrupted, then it can trigger a completion transition, these are transitions w/out event labels
 - Multiple completion transitions can be distinguished by guard conditions
 - Ex: When play track activity is completed:
 - If its last track, go to closed state
 - If it's not last track, play next track

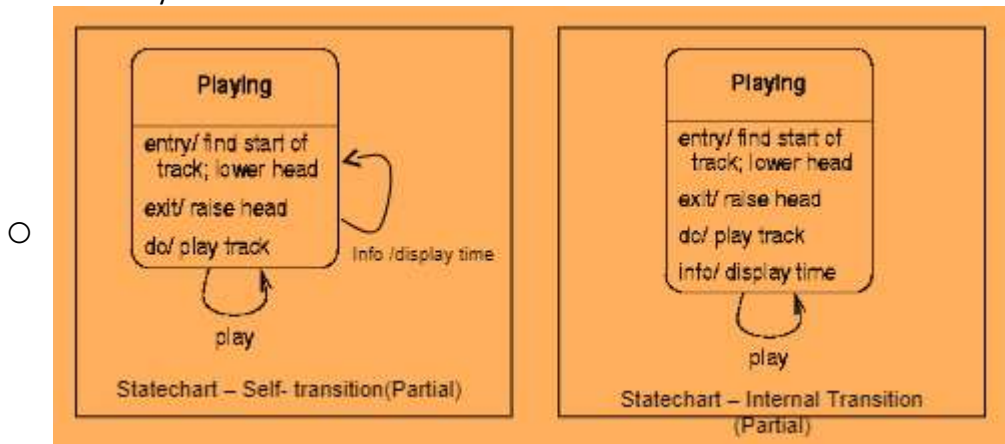
CD Player: Completion Transition Example



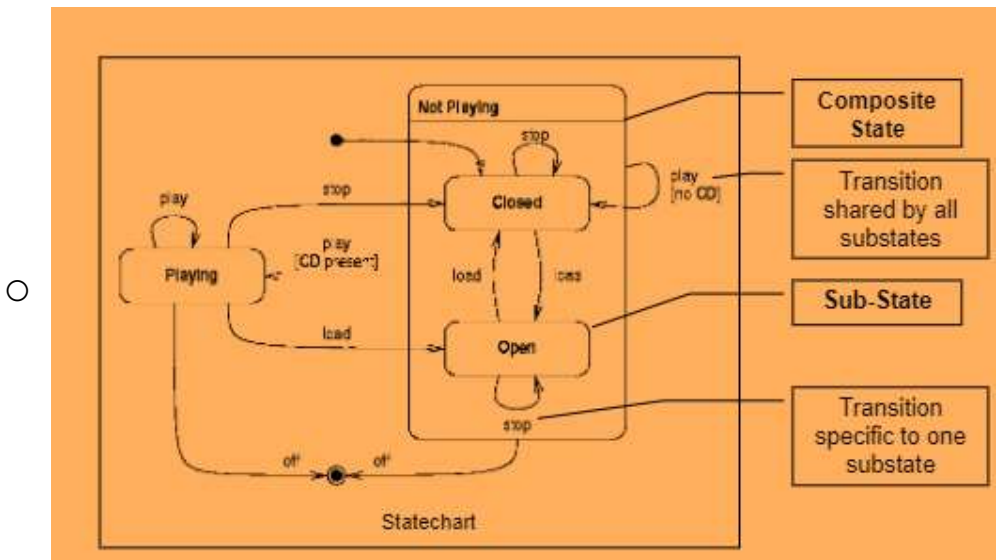


- Internal Transition
 - There is need for transitions that leave the object in the same state but don't trigger the entry and exit actions
 - Self-transition not appropriate (why?)
 - Use Internal Transition which doesn't trigger the entry and exit actions
 - Syntax: write in a compartment of the state box: event/action
 - Ex: supposed we add an info button for the CD player to display the remaining playing time for current track

- CD Player: Internal Transition
 - 2 diff ways to model info button



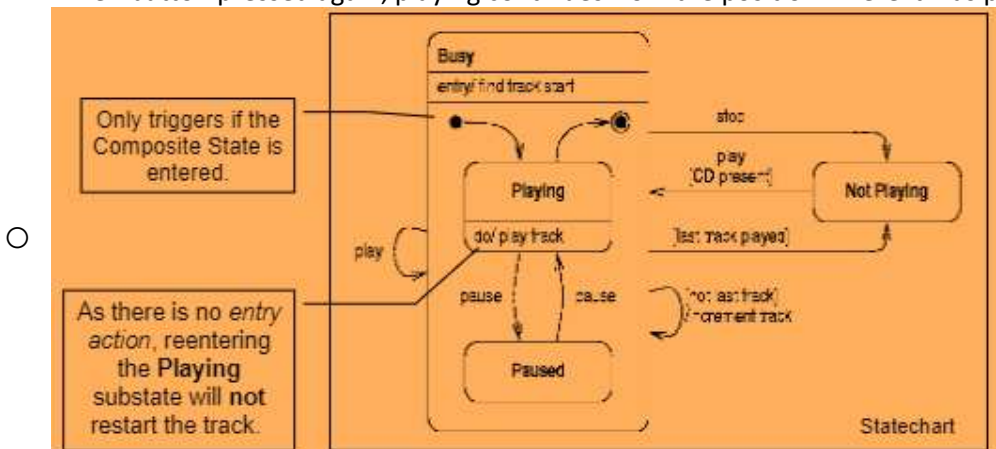
- If an "info" button is a self transition action, then it will trigger the entry and exit actions, which isn't correct
- Composite State
 - States that share similar behavior can be grouped into a Composite State to simplify the statechart
 - Composite state;
 - Syntax like simple state
 - Has a number of substates
 - When composite state is active, only one of the substate must be active
 - Transitions can lead away from a composite state as well as any of its substates
 - Ex: CD Player: response to the **play** event is similar during the **Open** or **Closed** states
- CD Player: Composite State



- Composite State: Additional Property
 - A composite state is like a simple state:
 - can have entry/exit actions
 - can have extended activity
 - Composite state also like a mini-statechart
 - Can have an Initial state to indicate the default substate if a transition terminates at boundary of a composite state
 - Can have a final state, which is triggered when ongoing activity within the state has finished
 - Transitions:
 - Leading away from a composite state apply to all substates
 - Arrive at composite state go to the default state
 - Can cross composite state boundaries

- CD Player: **Pause** button

- Pressing pause button causes playing to be interrupted
- When button pressed again, playing continues from the position where it was paused



- History State

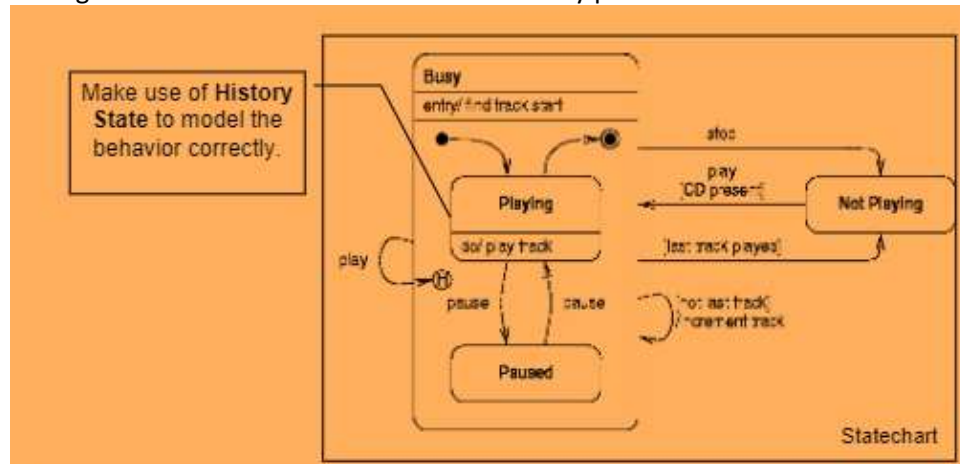
- When composite state entered, it begins @ the initial state/ directly transits into one of the substates
- Sometimes, useful to re-enter a composite state at a pt which it was left
 - Re-enter the last active substate
- Indicate using the History State
 - Transitions arriving at the History State activate the last active substate

- If there no last active substate, a default state can be indicated by an unlabeled transition from History State
- Syntax: a circled H

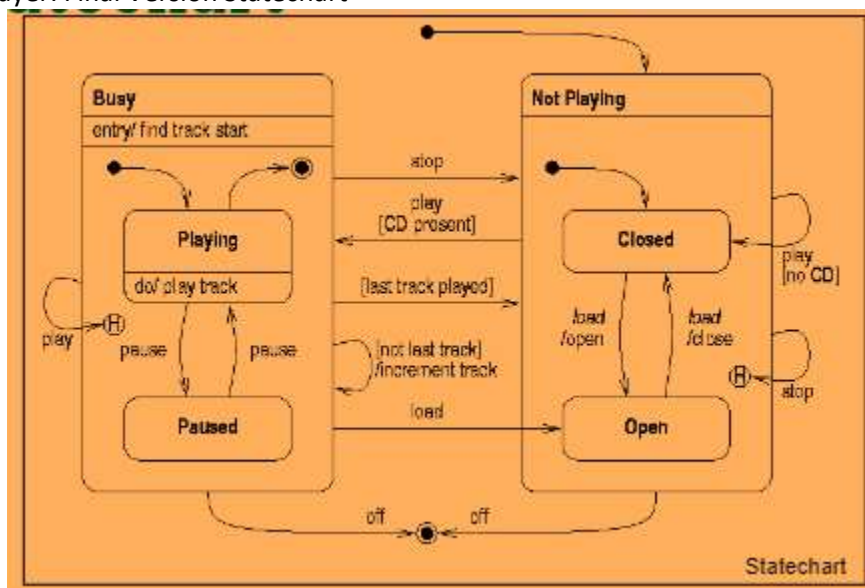


- Cd Player: History State

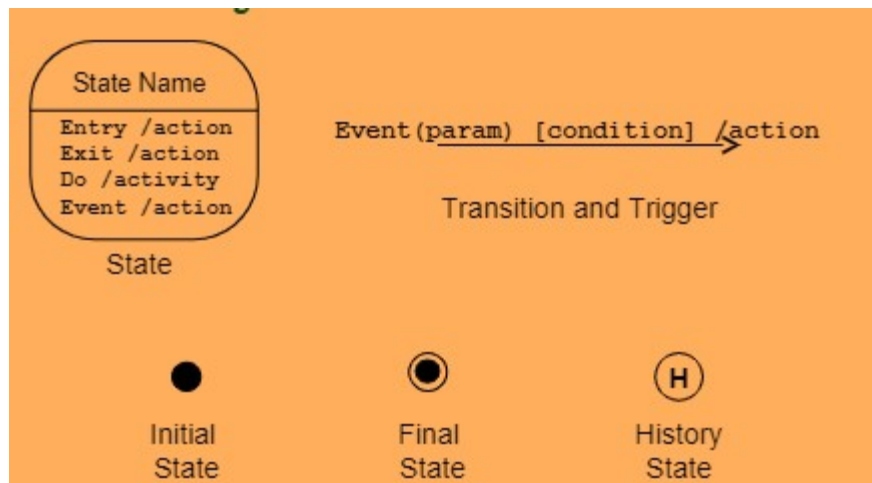
- Suppose re-pressing the play button:
 - During playing: restart and play the current track
 - During Paused: restart the current track but stay paused



- CD Player: Final Version Statechart



- Statechart Notation Summary



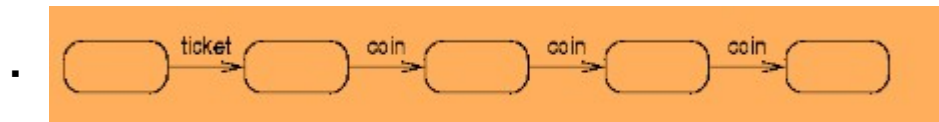
- Steps for Constructing Statechart
 1. Id objects that have complex behavior
 2. Determine the initial and final states of the object
 3. Id the events that affect the entity
 4. Working from the initial state, trace the impact of events and id the intermediate state
 5. Id any entry and exit actions on the states
 6. Expand states into a composite state if necessary
 7. Check which actions in the state are supported by operation(method)
 8. Refine the class if necessary

- Making a statechart

- How info from interaction diagrams can be used to derive statecharts?
- Can be hard to id all necessary states
- Statecharts can be developed incrementally:
 - Consider individual sequences of events received by an object
 - These might be specified on interaction diagrams
 - Start w/ a statechart for one interaction
 - Add states as required by additional interactions

- Ex: Ticket Machine

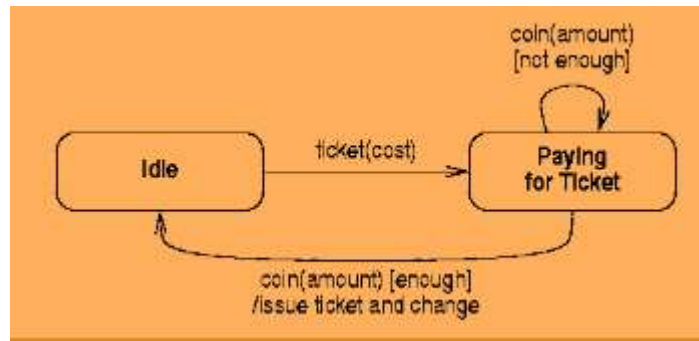
- Consider a ticket machine, 2 events:
 - Select a ticket type
 - Enter a coin
- Basic interactino is to select a ticket and then enter coins
- 'linear' statechart



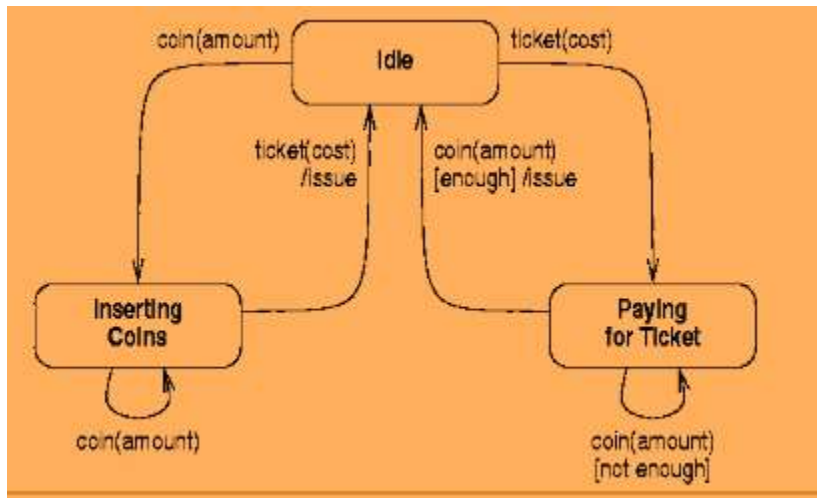
- Problems:
 - Defines only 1 transaction, but ticket machine able to carry out repeated transactions
 - Shows only 3 coins, but this numb should be arbitrary

- Refining the Statechart

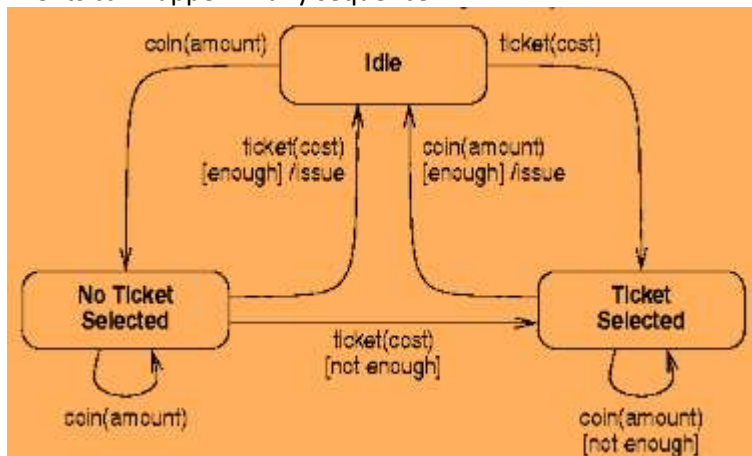
- Improved by adding 'loops'
 - Numb of coins entered will vary: entry will cont. Until ticket paid for
 - Whole transaction can be repeated
- State 'Idle': no transaction is in progress



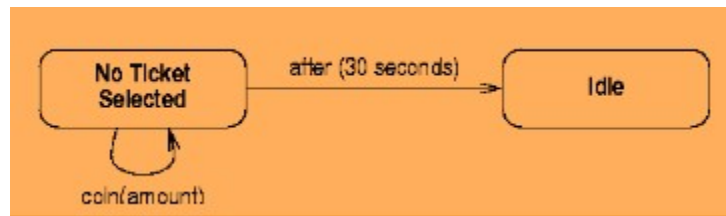
- Adding another interaction
 - Suppose the req allow user to enter coin b4 selecting a ticket
 - A 'coin' transition from the 'Idle' state is needed to handle this event
 - Transition can't go to the 'Paying for Ticket' state as the ticket isn't yet selected
 - So new state 'Inserting coins' is required
 - Statechart built up step-by-step
- Adding a second Interaction
 - If all coins entered b4 ticket selected:



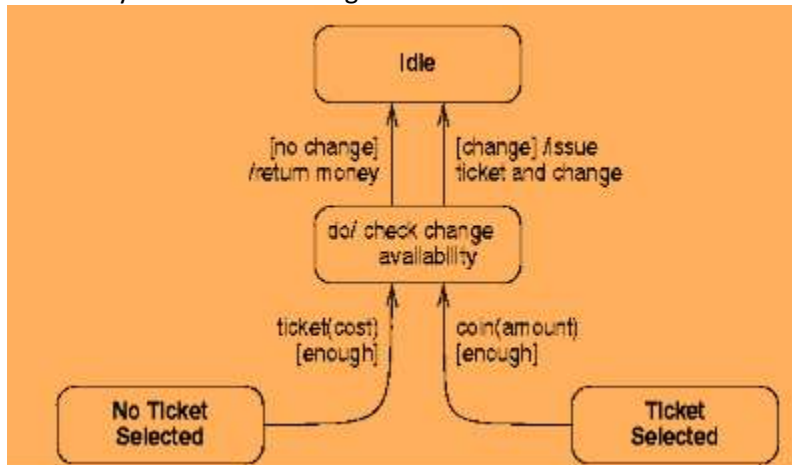
- Integrating the Interactions
 - Suppose the req allow user to enter some coins b4 selecting a ticket and the rest after that
 - Events can happen in any sequence:



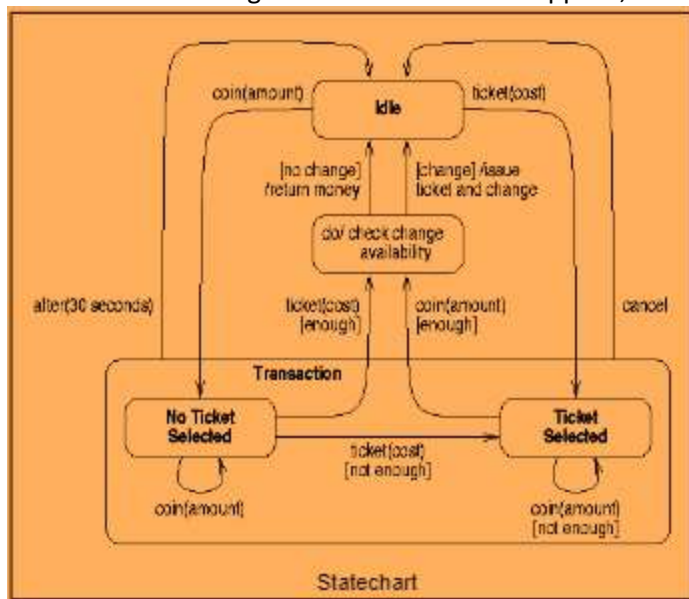
- Time Events
 - If ticket machine times out after 30 sec
 - Need to fire a transition that's not triggered by a user-generated event
 - UML defines time events to handle these cases
 - Ex: transition will fire 30 sec after 'no ticket selected' state is entered



- Activity States
 - Both 'no ticket selected' and 'ticket selected' should check if the machine is able to return any change that is req
 - Efficient solution: activity states
 - Activity states define periods of time when the object is carrying out internal processing
 - Unlike norm activities, these cant be interrupted by external events
 - Only completion transitions leading from them will make them leave the current state
 - Useful for simplifying the structure of complex statecharts
- Returning change
 - Note that activity as a property of state isn't same as activity states
 - Us activity state to calc change



- Ticket Machine Statechart
 - Incorporates 'Pressing Cancel' in middle of a transaction
 - Composite state 'Transaction' reduces numb of statechart transitions
 - Still room for refining: when 'cancel' event happens, machine should give back money



- Compare btwn statecharts and activity diagrams

- Many similar btwn both, activity diagrams are state diagrams extended w/ some extra notation
- Main diff btwn activity diagrams and state diagrams is that activity diagrams don't usually include events
- Also, activity intended to proceed w/out getting stuck, but state diagrams it is acceptable for object never to reach some of its potential state

Where are we now?



Summary

- Statechart
 - ▣ State-dependent behaviors, events, transitions
 - ▣ Initial and final states, guard conditions
 - ▣ Actions, activities, composite states, history states

○

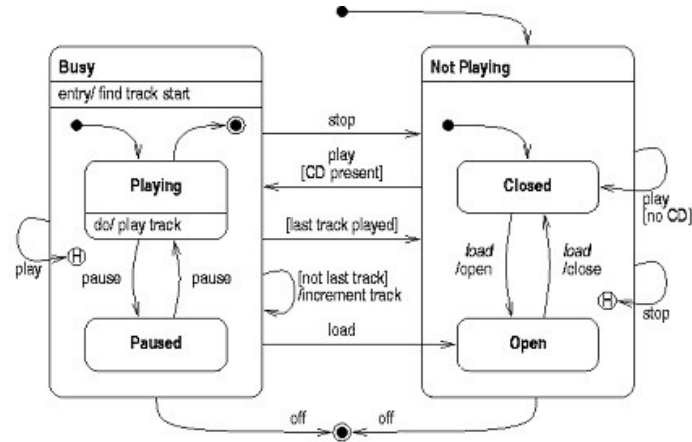
L8 Assessment

Friday, March 10, 2023 9:52 AM

Question 1

10 Points

(Exercise 10.1 from [Priestley; 2004]) What state is the CD player specified in the below figure after the following sequences of events are detected? Assume that a CD is always present when tested for.

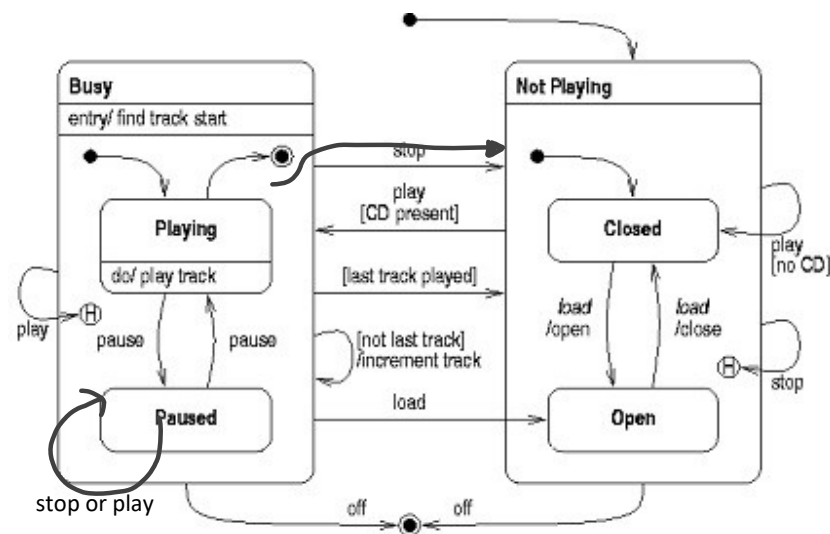


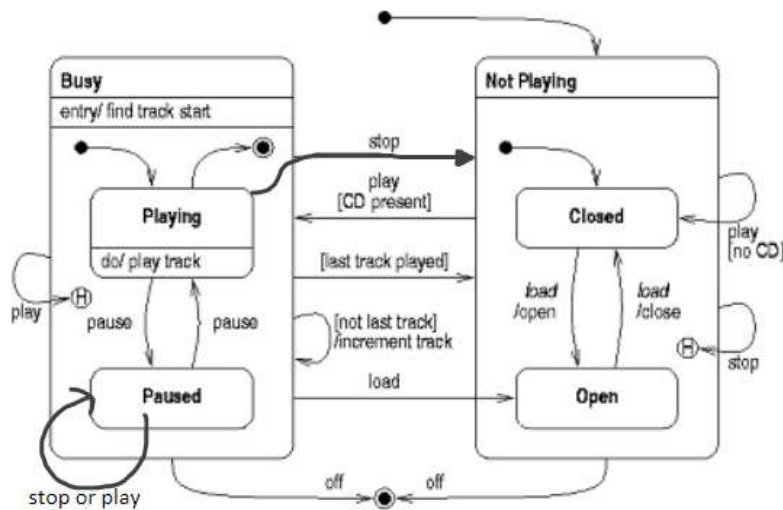
- (a) Initialize, load. → Open
- (b) Initialize, load, play, stop. → Not Playing
- (c) Initialize, load, play, pause, play. → History State
- (d) Initialize, play, stop, load. → Open
- (e) Initialize, load, pause, play. → Closed

Question 2

10 Points

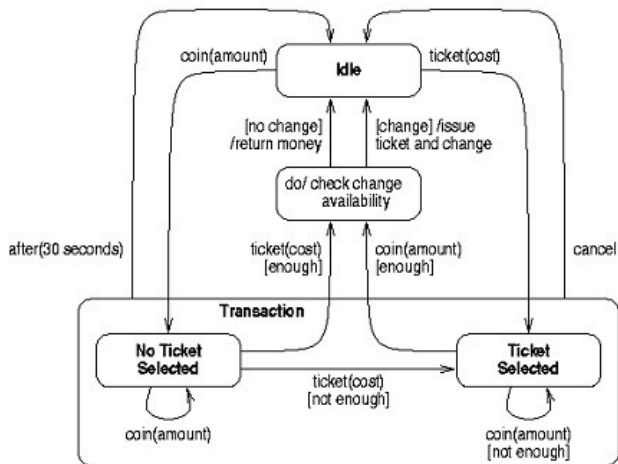
(Exercise 10.3 from [Priestley; 2004]) Draw a revised version of the statechart from the above figure modeling the requirement that the CD player should remain paused even when 'stop' and then 'play' are pressed.

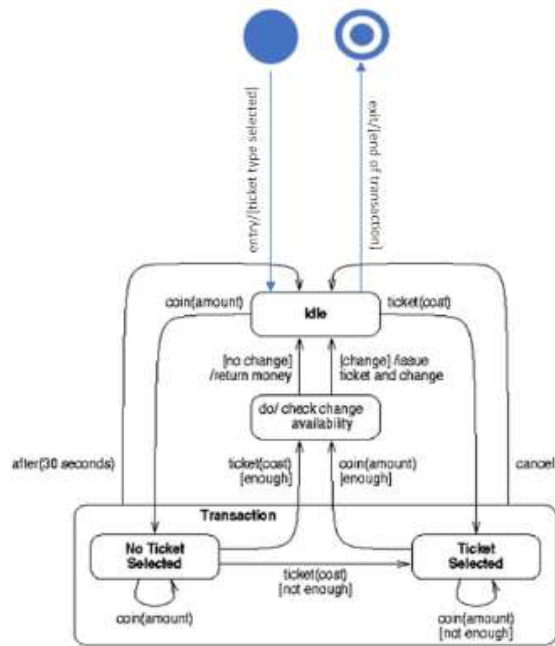




Question 3
10 Points

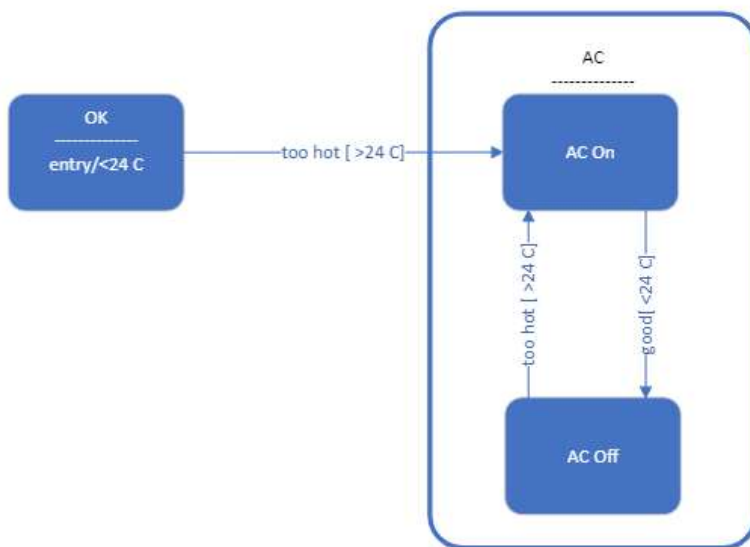
(Exercise 10.6 from [Priestley; 2004]) In the ticketing machine example described in Lecture Notes, suppose that the ticket selection buttons are deactivated once a ticket type has been selected, and only reactivated at the end of the transaction. In addition, suppose that once enough money has been entered to pay for the required ticket, the coin entry slot is closed, and only reopened once any ticket and change have been issued. Use entry and exit actions to show this behavior explicitly on the below state-chart:





Question 4 10 Points

Let us consider the automatic air conditioning system. Initially, the temperature is Ok, i.e., under 24 Celsius degrees. If it is too hot, then the air conditioner automatically changes into another state in order to turn on the air conditioning system. Once the temperature is below 24 Celsius degrees, it goes into a state responsible to turn off the air conditioning system. Draw a state diagram modeling the behavior of the automatic air conditioning system.

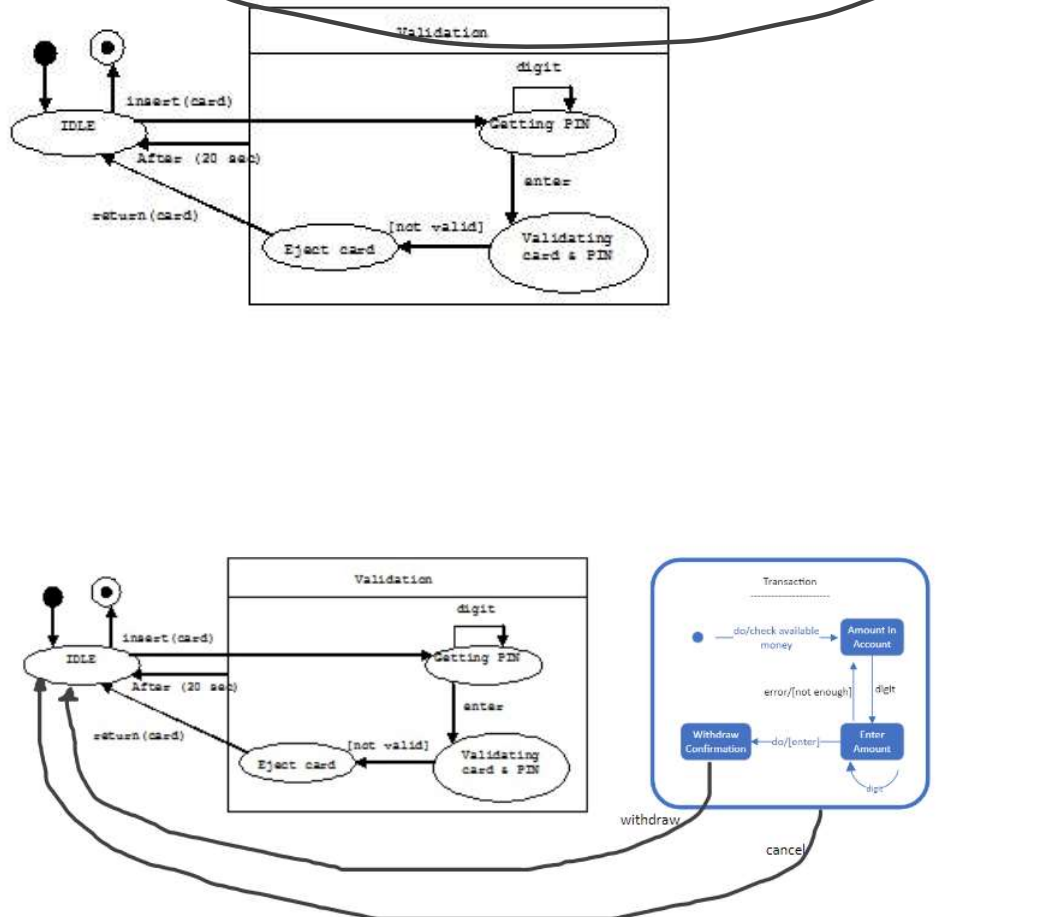


Question 5 10 Points



Question 5 10 Points

Let us consider the behaviour specification of a simple state-chart for an ATM for banking transactions. It has a slot to hold the bank card for reading and money. Also, it has an interface with a screen to display messages (e.g., options representing the digits for keying the Personal Identification Number (PIN), a 'clear', 'cancel', and 'enter'. For simplicity, we shall consider only the validation of the card, namely inserting card, PIN, and their validation. A partial state-chart for composite validation state is shown below. Continue this state-chart by specifying a transaction of money withdrawal for the case when 'Validating card & PIN' is valid.



Question 6 10 Points

What is the difference between the interaction diagrams and state-charts?

- Interactions diagrams specify the behavior of more objects than state-charts can do;
- Interactions diagrams are the same thing as class diagrams, whereas state-charts do not specify all the possible behaviours of objects;
- Interactions diagrams show behaviour in particular interactions, whereas state-charts summarize the overall behaviour of objects;
- They are basically the same thing;
- None of the above.

Question 7 10 Points

How can we eliminate the non-deterministic transitions from state-charts?

This is not possible;

By adding guard conditions to transitions such that all of them are evaluated to true;

By adding guard conditions to transitions such that at most one of them is evaluated to true;

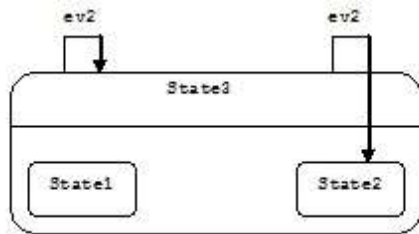
By adding the same guard condition to all transitions from a given state;

None of the above.

Question 8

10 Points

What is true about the below state-chart?



The transition labeled with ev1 is incorrect;

The transition labeled with ev2 is incorrect;

There is no self-transition of State1;

There is a transition from State1 to State2;

None of the above.