UNIT - II

STATE MODELING

* State model describes the sequences of operations that occur in response to external stimuli
* The state model consists of multiple state diagrams, one for each class with temporal behavior that is important to an application

It has 2 Components

- Events : represent external stimuli
- States : represent values (attributes) of objects.

* The state diagram is a standard computer science concept that relates events & states.

Events

* An event is an occurrence at a point of time

Ex: User depresses left button
Power turned on
Alarm set

* An event happens instantaneously with regard to time scale of an application.
* Events corresponds to verb in the past tense or the onset of some condition
* One event may logically precede or follow another, or the 2 events are unrelated. Events that are casually unrelated are said to be concurrent.
* Events include error conditions as well as normal occurences.
* Examples of error events
  → Motor jammed
  → Transaction aborted
  → Timeout

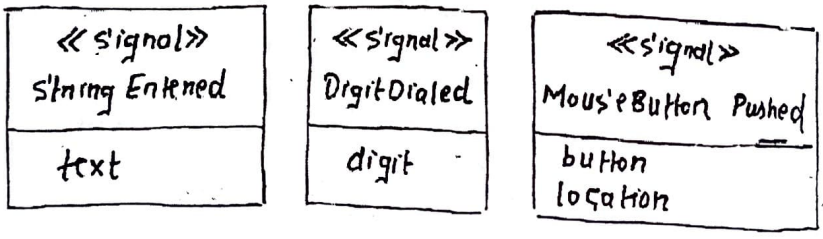
There are several kinds of events. The most common are

1) Signal event
2) Change event
3) Time event

1) Signal Event

* A signal is an explicit one-way transmission of information from one object to another.
* It is different from a subroutine call that returns a value
* An object sending a signal to another object may expect a reply, but the reply is a separate signal under the control of the second object, which may or may not choose to send it.
* A signal event is the event of sending or receiving a signal
* In general we are more concerned about the receipt of a signal because it causes effects in the receiving object
* The difference betn signal & signal event is
  - a signal is a message between objects
  - a signal event is an occurence in time

Ex:



2) Change Event

* A change event is an event that is caused by the satisfaction of a Boolean expression.
* The intent of a change event is that the expression is continuosly tested & whenever the expression changes from false to true the event happens
* UML notation for a change event is keyword when followed by a parenthesized Boolean expression

Ex: when (room temperature < heating set point)
when (room temperature > cooling set point)
when (battery power < lower limit)
when (fire pressure < minimum pressure)

3) Time Event

* Time event is an event caused by the occurence of an absolute time or the elapse of a time interval
* UML notation for an absolute time is the keyword when followed by a parenthesized expression involving time.
* The notation for a time interval is the keyword after followed by a parenthesized expression that evaluates to a time duration

Ex: when (date = jan 1, 2000)
after (10 seconds)

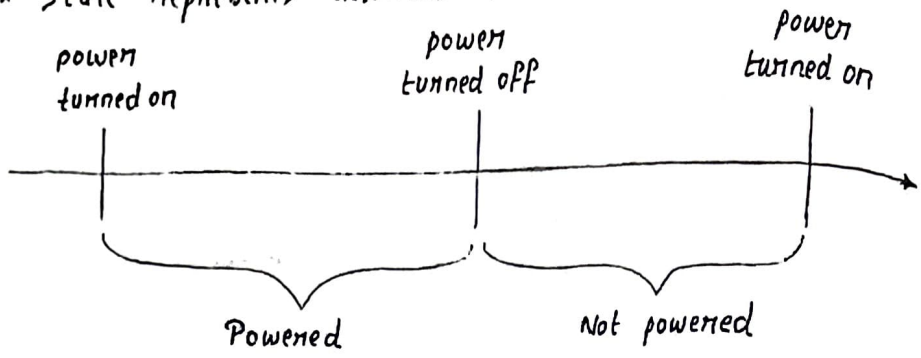
States

* A State is an abstraction of the values & links of an object
* sets of values & links are grouped together into a state according to the gross behavior of objects
* UML notation for state is a rounded box containing an optional state name, list the state name in boldface, center the name near the top of the box, capitalize the first letter
* Ignore attributes that do not affect the behavior of the object.
* The objects in a class have a finite number of possible states
* Each object can be in one state at a time
* A state specifies the response of an object to input events.
* All events are ignored in a state, except those for which behavior is explicitly prescribed.

* States often correspond to
  - verbs with a suffix 'ing': waiting, dialing
  - or the duration of some condition: powered, below freezing

Events v/s States

* Events represents points in time
* State represents intervals of time



* A State corresponds to the interval betn two events received by an object
* The state of an object depends on past events
* Both events & states depend on the level of abstraction

States may be characterized in various ways. as follows

State : Alarm Ringing

Description : alarm on watch is ringing to indicate target time

Event Sequence that produces the state

SetAlarm (targetTime)
any sequence not including CleanAlarm
when (CurrentTime = targetTime)

Condition that characterizes the state

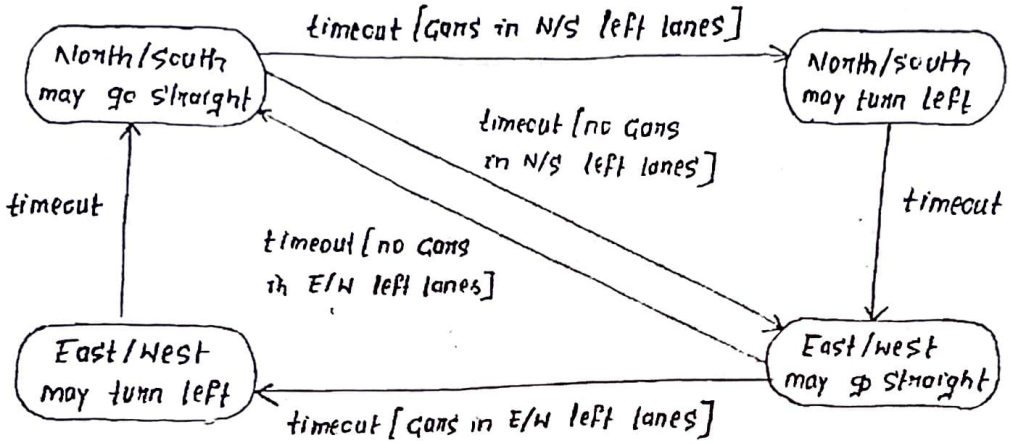
alarm = on, alarm set to targetTime, targetTime ≤ CurrentTime ≤ targetTime + 20 seconds & no button has been pushed since targetTime

Events accepted in the state

event	response	next state
when (CurrentTime = targetTime + 20)	resetAlarm	normal
buttonPushed (any button)	resetAlarm	normal

Transitions & Conditions

- A transition is an instantaneous change from one state to another
- The transition is said to fire upon the change from the source state to target state
- The origin & target of a transition usually are different states, but sometimes may be the same
- A transition fires when its events occurs
- A guard condition is checked only once, at the time the event occurs, & the transition fires if the condition is true.
- A guard condition is a Boolean expression that must be true in order for a transition to occur



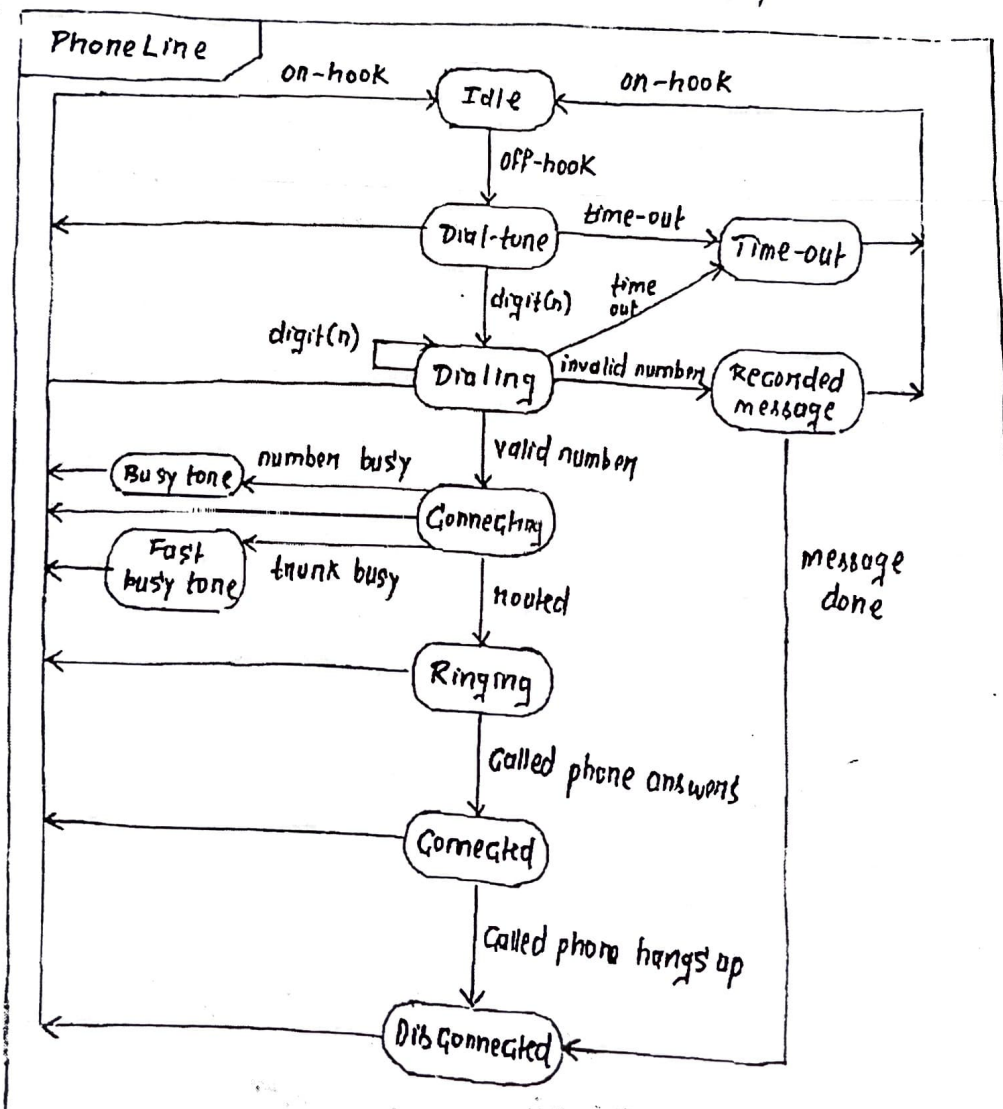
Guard Condition V/S Change event

Guard condition	Change event
A guard condition is checked only once	A change event is checked continuously
UML notation for a transition is a line	may include event label in italics
followed by guard condition in square brackets	from the origin state to the target state an arrowhead points to the target state

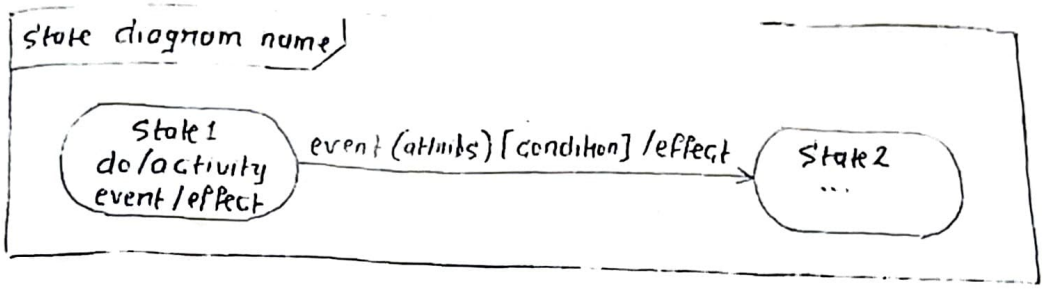
State Diagram

- A state diagram is a graph whose nodes are states & whose directed arcs are transitions betn states
- A state diagram specifies the state sequence caused by event sequences
- state names must be unique within the scope of a state diagram
- UML notation for a state diagram is a rectangle with its name in small pentagonal tag in the upper left corner
- The constituent states & transitions lie within the rectangle
- If more than one transition leaves a state, then the first event to occur causes the corresponding transition to fire.
- If an event occurs & no transition matches it, then the event is ignored.
- If more than one transition matches an event, only one transition will fire, but the choice is nondeterministic

Ex



Basic State Diagram Notation



~~State diagram Behavior~~

Activity

* An activity is behavior that can be executed in response to an event
* An activity can be performed upon a transition, upon the entry to or exit from a state or upon some event within a state

State diagram Behavior

Activity effects

→ An effect is a reference to a behavior that is executed in response to an event

→ An activity is the actual behavior that can be invoked by any number of effects

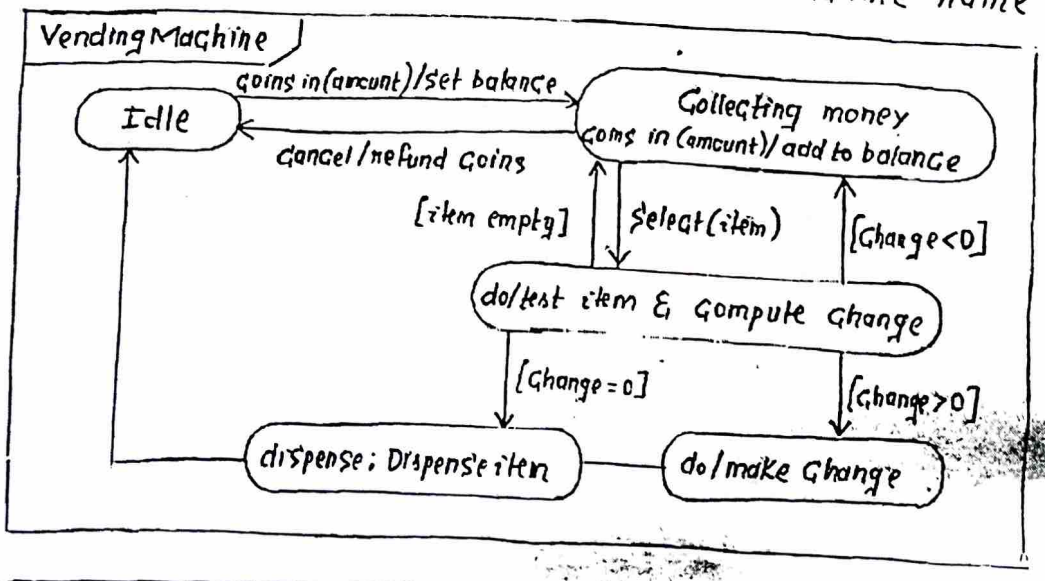
Ex: disconnectPhoneLine might be an activity that executed in response to an onHook event

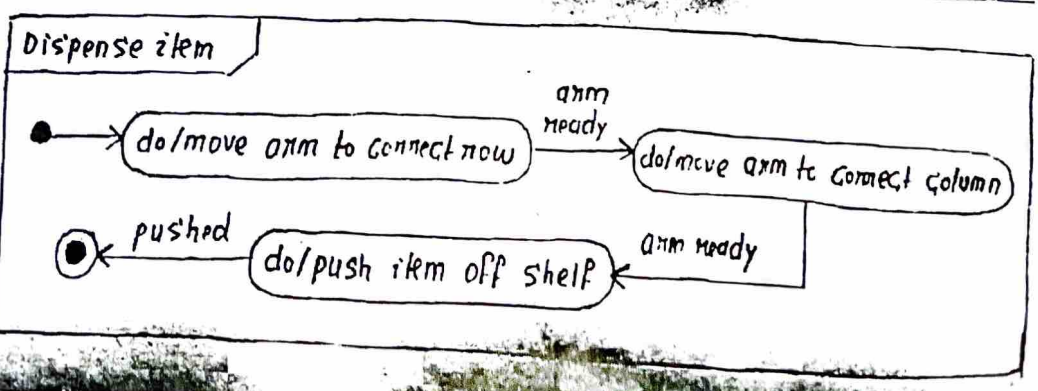
Advanced State Diagrams

* Conventional state Diagroms are helpful for describing simple problem, but for complex system need additional concepts
* Additional Concepts are
  - Nested state diagram
  - Nested state
  - Signal generalization
  - Concurrency

Submachine

* One way to organize a model is by having high level diagram with sub diagrams expanding certain state. This is like a macro substitution in programming language
* A submachine is a state diagram that may be invoked as part of another state diagram
* UML notation for invoking a submachine is to list a local state name followed by colon & submachine name

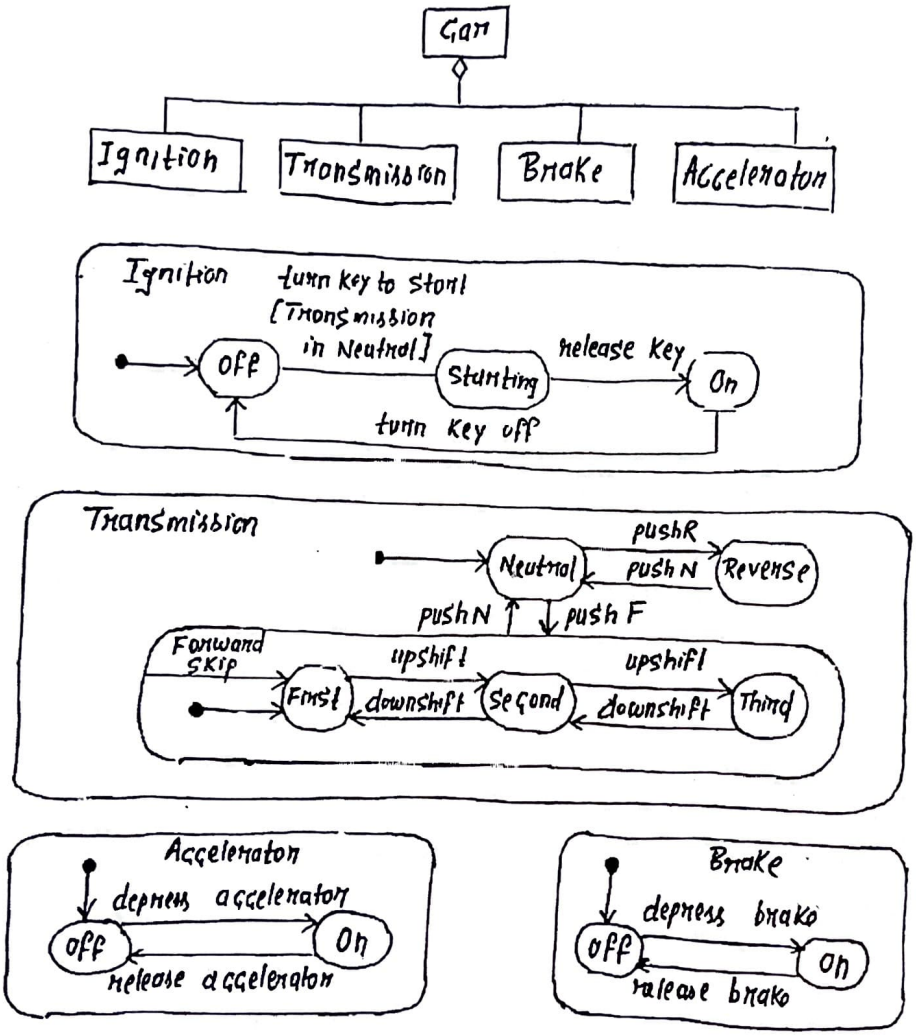




Concurrency

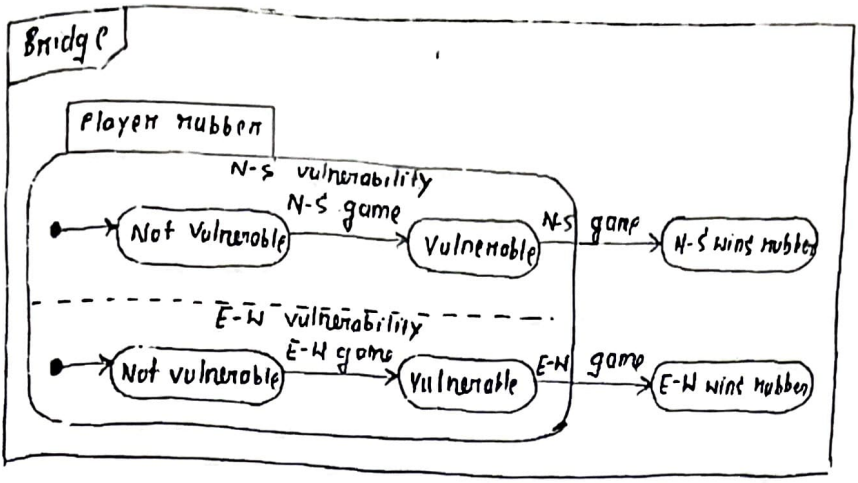
- The state model implicitly supports concurrency among objects.
- In general, objects are autonomous entities that can act & change state independent of one another. However objects need not be completely independent & may be subject to shared constraints that cause some correspondence among their state changes

1. Aggregation Concurrency



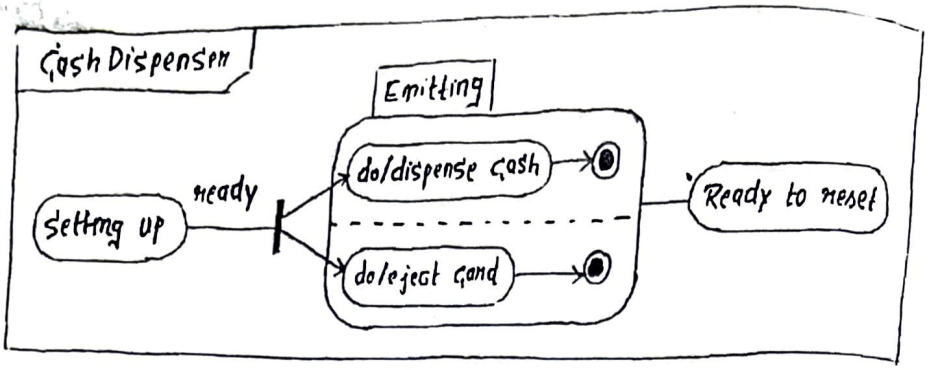
- A state diagram for an assembly is a collection of state diagrams, one for each part. The aggregate state corresponds to the combined states of all the parts.
- Aggregation is the "and-relationship". The aggregate state is one state from 1st diagram & a state from the 2nd diagram & a state from each other diagram.

2. Concurrency within an object



* You can partition some objects into subsets of attributes or links, each of which has its own subdiagram. The state of the object comprises one state from each subdiagram. The subdiagram need not be independent; the same event can cause transitions in more than one subdiagram.
* The UML shows concurrency within an object by partitioning the composite state into regions with dotted lines. You should place the name of the composite state in a separate tab so that it does not become confused with the concurrent regions
* The above fig. shows the state diagram for the play of a bridge rubber. when a side wins a game, it becomes "vulnerable"; the first side to win 2 games wins the rubber.
* During the play of the rubber, the state of the rubber consists of one state from each subdiagram. When the playing rubber composite state is entered, both regions are initially in their respective default states Not vulnerable.
* Each region can independently advance to state vulnerable when its side wins a game.
* When one side wins a second game, a transition occurs to the corresponding wins rubber state. This transition terminates both concurrent regions, because they are part of the same composite state Playing rubber & are active only when the top level state Diagram is in that state.

3. Synchronization of Concurrent activities



* Sometimes one object must perform two (or more) activities concurrently. The object does not synchronize the internal steps of the activities but must complete both activities before it can progress to its next state.
* Ex: a cash dispensing machine dispenses cash & returns the user's card at the end of a transaction.
* The machine must not reset itself until the user takes both the cash & the card, but the user may take them in either order or even simultaneously.
* The order in which they are taken is irrelevant, only the fact that both of them have been taken. This is an example of splitting control into concurrent activities & later merging them