CHAPTER 2B: STATE MODELING

- The state model describes the sequence of operations that occur in response to external stimuli, as opposed to what the operations do, what they operate on, or how they are implemented.
- The state model consists of multiple state diagrams, one for each class with temporal behavior that is important to an application.
- The state diagram is a standard computer science concept (a graphical representation of finite state machines) that relates events and states.
  - Events represent external stimuli.
  - States represent values of objects.

EVENTS

- **Definition:** An event is an occurrence at a point in time.
  Ex: User depresses left button.
  Flight 123 departs from Chicago.
- Events often correspond to verbs in the past tense or to the onset of some condition.
- By definition, an event happens instantaneously w.r.t time scale of an application. Of course, nothing is really instantaneous; an event is simply an occurrence that an application considers atomic and fleeting.
- The time at which an event occurs is an implicit attribute of the event.
- One event may logically precede or follow another, or the two events maybe unrelated. Two events that are casually unrelated are said to be concurrent; they have no effect on each other.
- Events include error conditions as well as normal occurrences.
  E.g. motor jammed, transaction aborted and time-out are typical error events.
- **Kinds of events:**
  - Signal event
  - Change event
  - Time event

Signal event

- A signal is an explicit one-way transmission of information from one object to another. 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.
- Difference between *signal* and *signal event*:
  - A signal is a message between objects.
  - A signal event is an occurrence in time.
- Every signal transmission is a unique occurrence, but we group them into *signal classes* and give each signal class a name to indicate common structure and behavior.

E.g. UA Flight 123 departs from Chicago on Jan 10, 1991 is an instance of signal class *FlightDeparture*.

- The UML notation :
  - Use the keyword *signal* in guillemots (<< >>) above the signal class name in the top section of a box.
  - The second section lists the signal attributes.

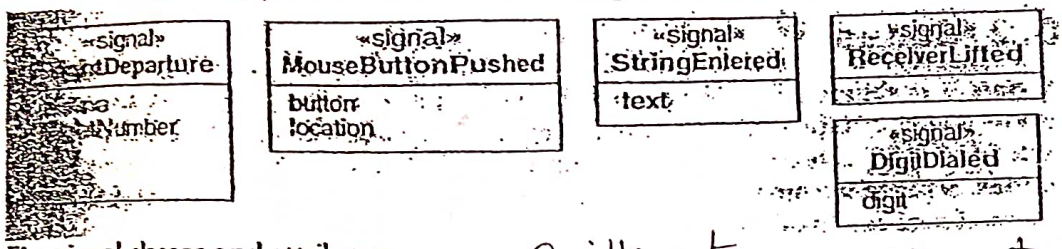


Fig: signal classes and attributes

guillemots guillemots (<<>>)

guillemots(<<>>) cla

Change event

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

E.g.

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

Time event

- A time event is an event caused by the occurrence of an absolute time or the elapse of a time interval.
- UML notation:
  - For an absolute time, it is the keyword *when* followed by a parenthesized expression involving time.
  - For a time interval, it is the keyword *after* followed by a parenthesized expression that evaluates to a time duration.

E.g.

when (date=January 1, 2000) .
after (10 seconds).

STATES

- Definition: A state is an abstraction of the values and links of an object. Sets of values and links are grouped together into a state according to the gross behavior of objects.

- States often correspond to verbs with a suffix "ing" (*waiting, dialing*) or the duration of some condition (*powered, BelowFreezing*).
- **UML notation:**
  - A rounded box containing an optional state name.
  - Our convention is to list the state name in boldface, center the name near the top of the box and capitalize the first letter.

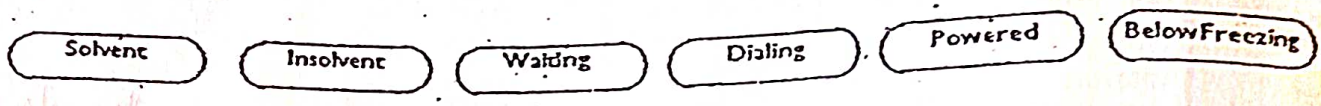
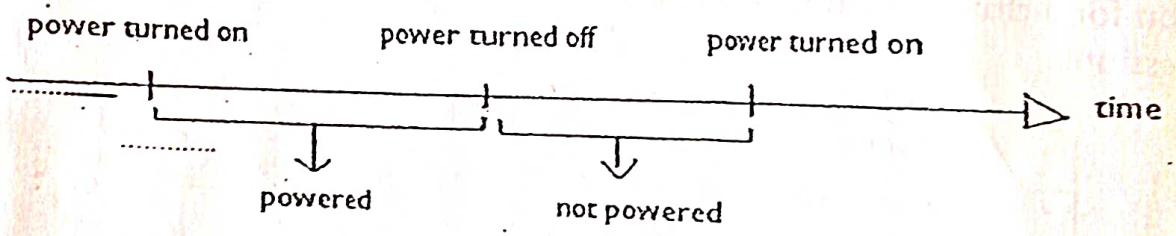


Fig: states

- In defining states, we ignore attributes that do not affect the behavior of the object.
- The objects in a class have a finite number of possible states – one or possibly some large number. Each object can only be in one state at a time.
- A state specifies the response of an object to input events.
- Event v/s state:



  - Events represent points of time.
  - States represent intervals of time.
- You can characterize a state in various ways, as fig shows for the state **Alarm ringing** on a watch.

State: *AlarmRinging*

Description: alarm on watch is ringing to indicate target time

Event sequence that produces the state:

setAlarm (*targetTime*)
any sequence not including *clearAlarm*
when (*currentTime* = *targetTime*)

Condition that characterizes the state:

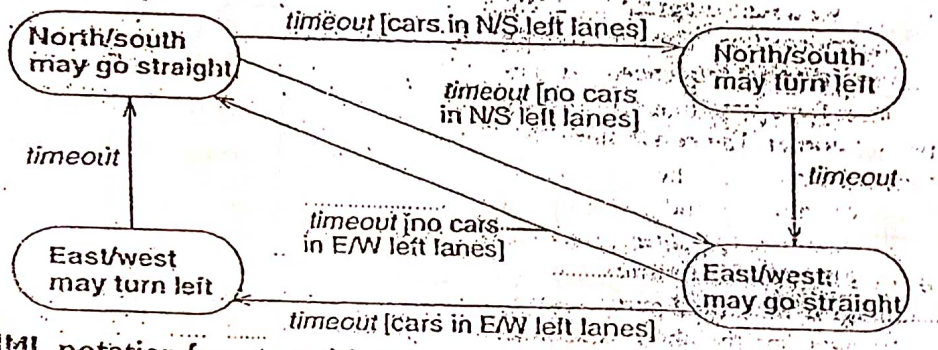
alarm = on, alarm set to *targetTime*, *targetTime* ≤ *currentTime* ≤ *targetTime* + 20 seconds, and no button has been pushed since *targetTime*

Events accepted in the state:

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

TRANSITIONS AND CONDITIONS

- **Definition:** A **transition** is an instantaneous change from one state to another.
  E.g. when a called phone is answered, the phone line transitions from the *Ringing* state to the *connected* state.
- The transition is said to **fire** upon the change from the source state to the target state.
- The transition fires when its events occur. The choice of next state depends on both the original state and the event received.
- A **guard condition** is a Boolean expression that must be true in order for a transition to occur.
  Ex: A traffic light at an intersection may change only if a road has cars waiting.
- A **guarded transition** fires when its event occurs, but only if the guard condition is true.
  Ex: "when you go out in the morning (event), if the temperature is below freezing (condition), then put on your gloves (next state)".
- Fig shows guarded transitions for traffic lights at an intersection.



UML notation for a transition:

- Use a line from origin state to the target state. An arrowhead points to the target state.
- Line may have several line segments.
- An event may label the transition and be followed by an optional guard condition in square brackets.
- By convention, we usually confine line segments to a rectilinear grid. We italicize the event name and show he condition in normal font.

STATE DIAGRAMS

Definition: A **state diagram** is a graph whose nodes are states and whose directed arcs are transitions between states. It specifies the state sequences caused by event sequences.

State names must be unique within the scope of a state diagram.

All objects in a class execute the state diagram for that class, which models their common behavior.

We can implement state diagrams by direct interpretation or by converting the semantics into equivalent programming code.

- The state model consists of multiple state diagrams, one for each class with important temporal behavior.
  The individual state diagrams interact by passing events and through the side effects of guard conditions.

Sample state diagram

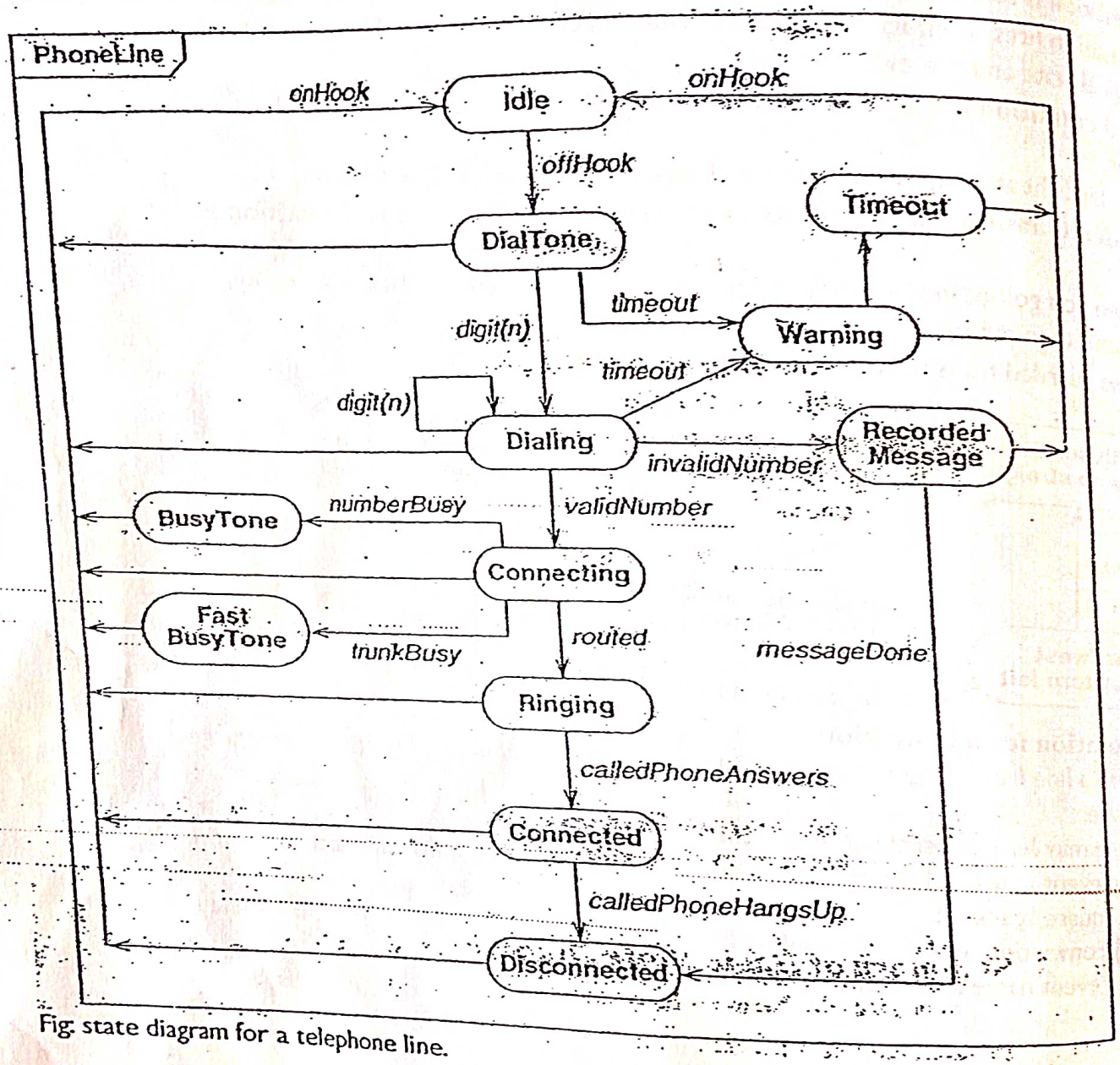


Fig: state diagram for a telephone line.

One-shot state diagrams

- State diagrams can represent continuous loops or one-shot life cycles.
  Diagram for the phone line is a continuous loop.
  One-shot state diagrams represent objects with finite lives and have initial and final states.
  - The initial state is entered on creation of an object.
  - Entry of the final state implies destruction of the object.

- Examples:

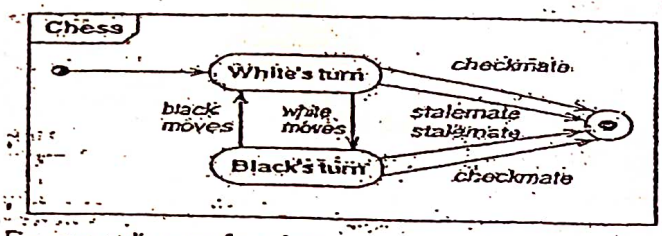


Fig: state diagram for chess game: one-shot diagrams represent objects with finite lives.

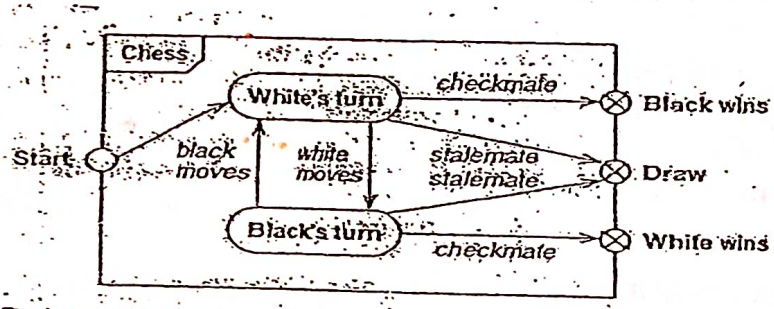
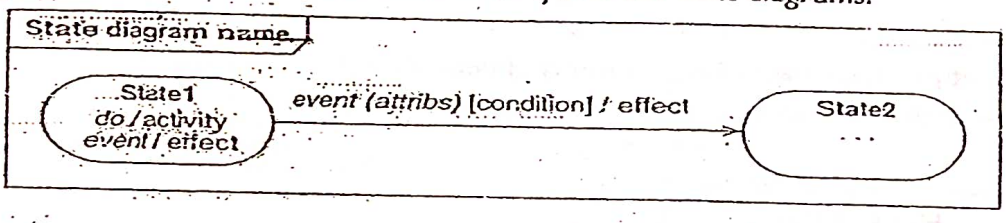


Fig: state diagram for chess game: you can also show one-shot diagrams by using entry and exit points.

Summary of Basic state diagram notation

- Figure below summarizes the basic UML syntax for state diagrams.

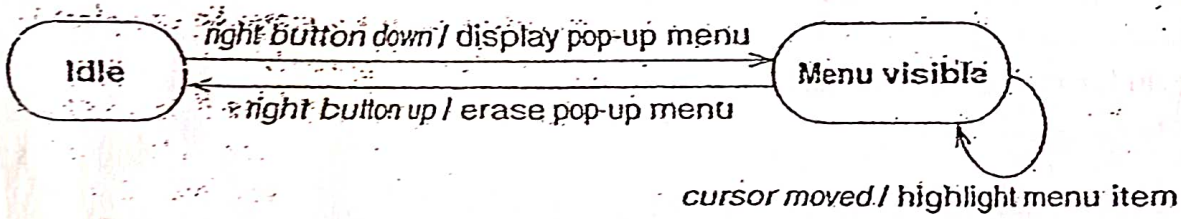


- State: drawn as round box containing an optional name. A special notation is available for initial states(a solid circle) and final states(a bull's eye or encircled 'x').
- Transition: drawn as a line from the origin state to the target state. An arrowhead points to the target state.
- Event: a signal event is shown as a label on a transition and maybe followed by parenthesized attributes.
  A change event is shown with the keyword *when* followed by a parenthesized Boolean expression.
  A time event is shown with the keyword *when* followed by a parenthesized expression involving time or the keyword *after* followed by a parenthesized expression that evaluates to a time duration.
- State diagram: enclosed in a rectangular frame with the diagram name in a small pentagonal tag in the upper left corner.
- Effects: can be attached to a transition or state and are listed after a slash ("/"). Multiple effects are separated with a comma and are performed concurrently.

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 is executed in response to an onHook even for fig(state diagram for telephone line).
- An activity may be performed upon a transition, upon the entry to or exit from a state, or upon some other event within a state.
- The notation for an activity is a slash ("/") and the name (description) of the activity, following the event that causes it. The keyword *do* is reserved for indicating an ongoing activity and may not be used as an event name.
- Fig shows the state diagram for a pop-up menu on a workstation.



Do-Activities

- A do-activity is an activity that continues for an extended time. By definition, a do-activity can only occur within a state and cannot be attached to a transition.
  E.g.

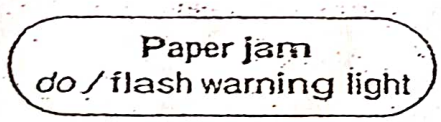


Fig do-activity for a copy machine

- The notation "do/" denotes a do-activity that maybe performed for all or part of the duration that an object is in a state.
- A do-activity maybe interrupted by an event that is received during its execution.

Entry and Exit activities

- As an alternative to showing activities on transition, you can bind activities to entry or to exit from a state.
  E.g.
  Fig below shows the control of a garage door opener.

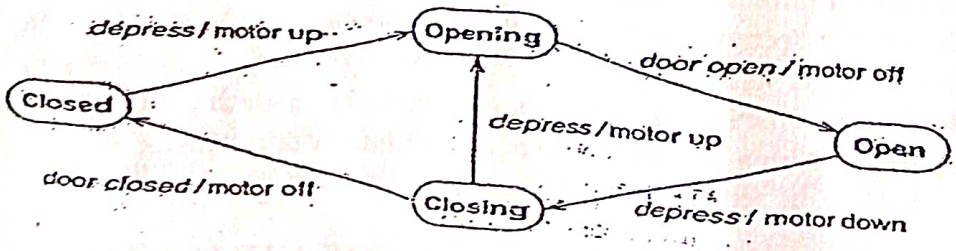
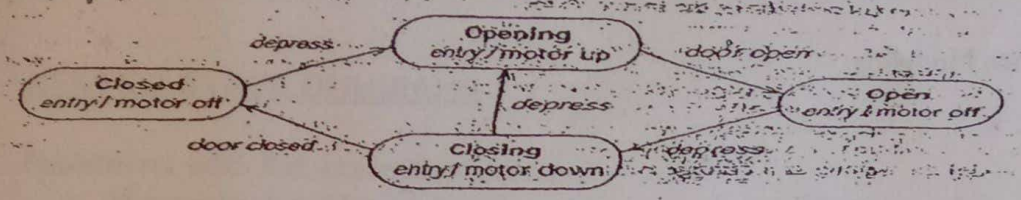
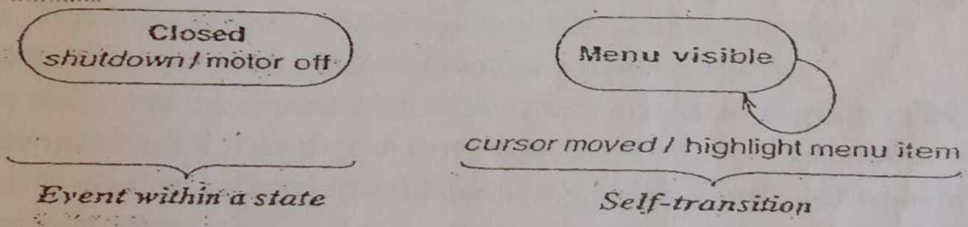


Fig below shows the same model using activities on entry to states



- An entry activity is shown inside the state box following the keyword *entry* and a "/" character.
- Whenever the state is entered by an incoming transition, the entry activity is performed.
- An entry activity is equivalent to attaching the activity to every incoming transition.
  An exit activity is shown inside the state box following the keyword *exit* and a "/" character.
- Whenever the state is exited by any outgoing transition, the exit activity is performed first.
- If a state has multiple activities, they are performed in the following order –
  - Activities on the incoming transition.
  - Entry activities.
  - Do-activities.
  - Exit activities.
  - Activities on outgoing transition.
- As shown in below figure, there is a difference between an event within a state and a self transition; only the self transition causes the entry and exit activities to be executed.



Complete transition

Unlabeled transitions are called **complete transitions** because they are triggered by the completion of activity in the source state.

Sending signals

An object can perform the activity of sending a signal to another object. A system of objects interacts by exchanging signals.

The activity "send target.S(attributes)" sends signal S with the given attributes to the target object(s).

Ex: PhoneLine sends a connect (phone number) signal to the switcher when a complete phone number has been dialed.

If an object can receive signals from more than one object, the order in which concurrent signals are received may affect the final state; this is called a **race condition.**