COMP 3700: Software Modeling and Design

(State Modeling)

Topics

- State Model
- Events
- States
- Transitions and Conditions
- State Diagrams
- State Diagram Behavior

Topics

- State models specify the behaviour of the system in response to external and internal events
- They show the system's responses to stimuli so are often used for modelling real-time systems
- State machine models show system states as nodes and events as arcs between these nodes. When an event occurs, the system moves from one state to another
- Statecharts are an integral part of the UML

State Model

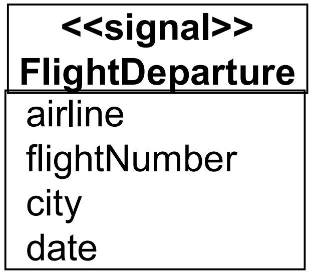
- The state model describes the sequence of operations that occur in response to external stimuli.
- Consists of multiple state diagrams, one for each class with temporal behavior.
- Events represents the stimuli.
- States represent the values of objects.

Events

- **An event is an occurrence at a point in time.**
- **Examples:**
  - ***user depresses left button***
  - ***flight 123 departs from Chicago***
- **Events include error conditions as well as normal occurrences (e.g., *motor jammed, transaction aborted*)**
- **Types of events**
  - **Signal event**
  - **Change event**
  - **Time event**

Events

- A **signal** is an explicit one way transmission of information from one object to another.
- It is different than a subroutine call.
- A **signal event** is the event of sending or receiving a signal



Events

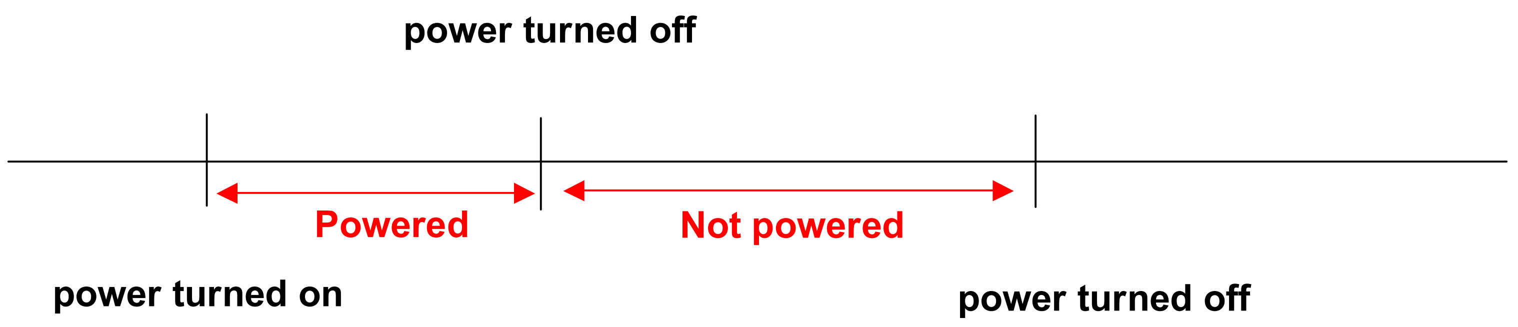
- **A change event is an event that is caused by the satisfaction of a boolean expression represented by a predicate.**
- **Expression is continuously tested – whenever the expression changes from false to true the event happens.**
- **Examples:**
  - **when** (room temperature < heating set point)
  - **when** (battery power < lower limit)

Events

- **A time event is an event caused by the occurrence of an absolute time or the elapse of a time interval.**
- **UML notation –**
  - **when** (date = January 16, 2007)
  - **after** (10 seconds)

States

- A state is an abstraction of the values and links of an object.
- Sets of values and links are grouped into a state according to the gross behavior of an object.
- Example: A stack object can be in one of the following states: *empty*, *non-empty*, and *full*.

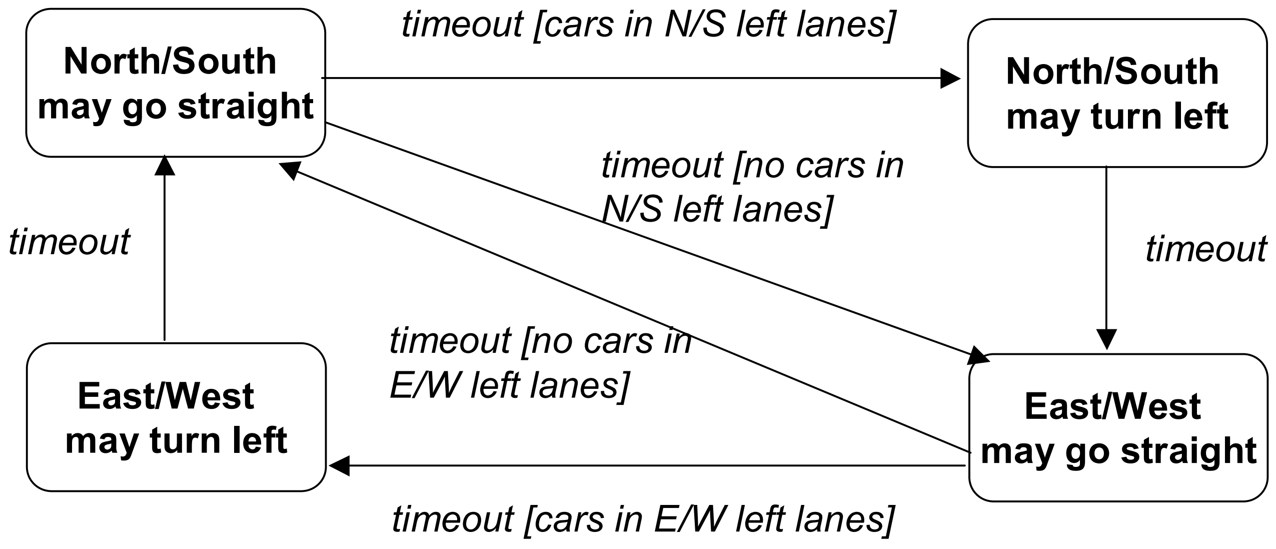


Transitions and Conditions

- **A transition is a relation between two states.**
- **Transition depicts an instantaneous change from one state to another.**
- **Example:**
  - ***When a called phone is answered, the phone line transitions from*** *Ringing state to the Connected state.*
- **The transition is said to fire upon the change from the source state to the target state.**

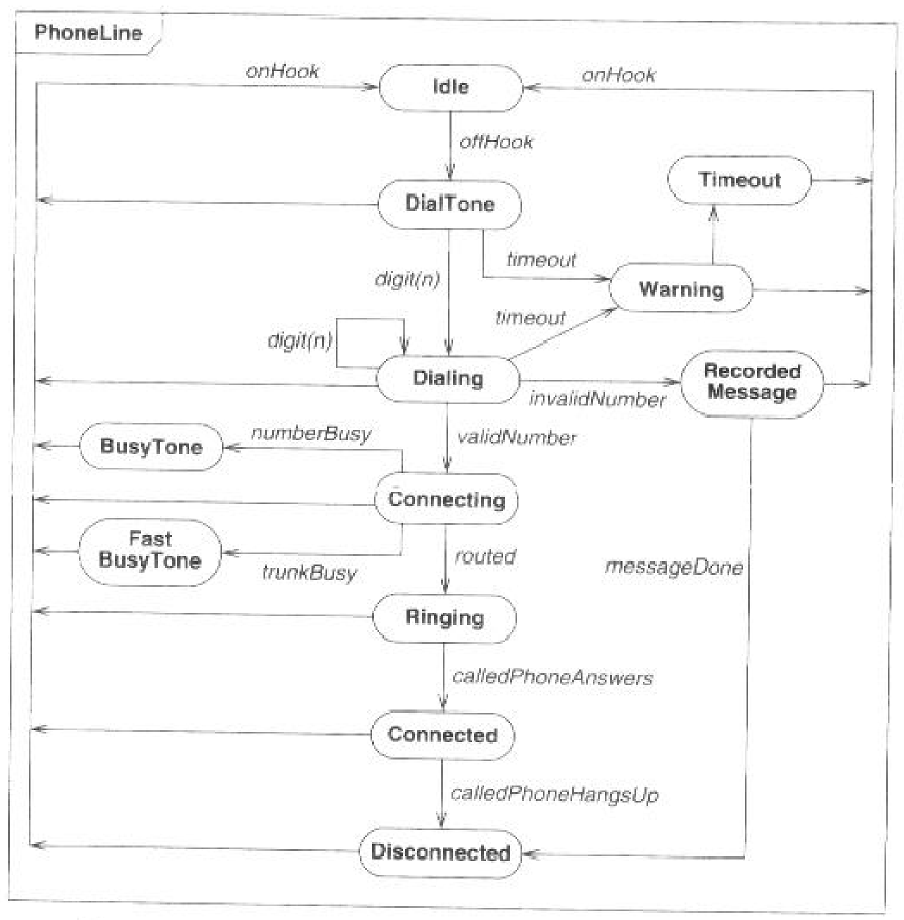
Transitions and Conditions

- A **guard condition** is a boolean condition that must be true in order for a transition to occur.


North/South may go straight
timeout [cars in N/S left lanes]
North/South may turn left
timeout [no cars in N/S left lanes]
timeout
timeout
timeout [no cars in E/W left lanes]
East/West may turn left
East/West may go straight
timeout [cars in E/W left lanes]

State Diagrams – (Phone Line Example)

- A **state diagram** is a graph whose nodes are states and whose arcs are transitions between states.
- A full description of an object must specify what the object does in response to events.

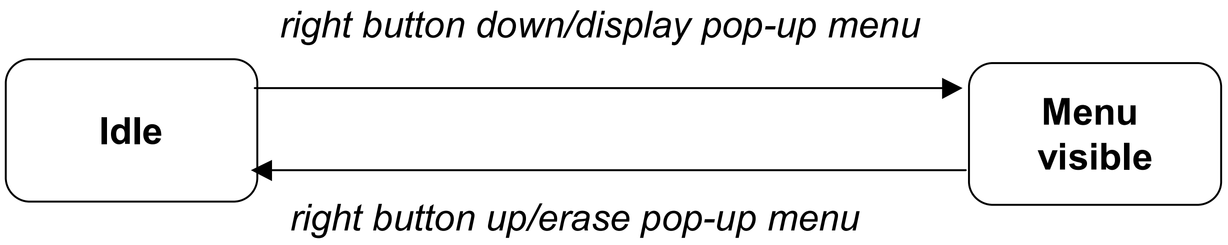


Completeness and Consistency Analysis of UML Finite Statecharts

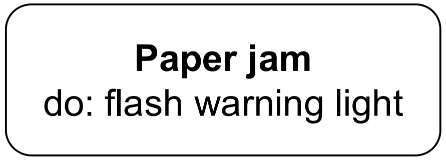
- **States and Transitions**
  - *Completeness* requires that in each basic state, for all possible events, there must be a transition defined.
  - *Completeness* requires that each state is targeted by (at least one) transition. Note that initial states have an incoming transition from the initial pseudo-state.
  - *Consistency* of the specification requires that in each state, only a single transition is triggered by a given event.
- **Guards**
  - *Completeness* requires that in each basic state, considering also inherited transitions, guards of transitions triggered by the same event form a tautology.
  - *Consistency* is checked by the following criterion: If there are two or more transitions that are originating from the same state and triggered by the same event, then their guards could not be true at the same time.

Activities

- An **activity** is the actual behavior invoked or executed in response to an event.


right button down/display pop-up menu
Idle
Menu visible
right button up/erase pop-up menu

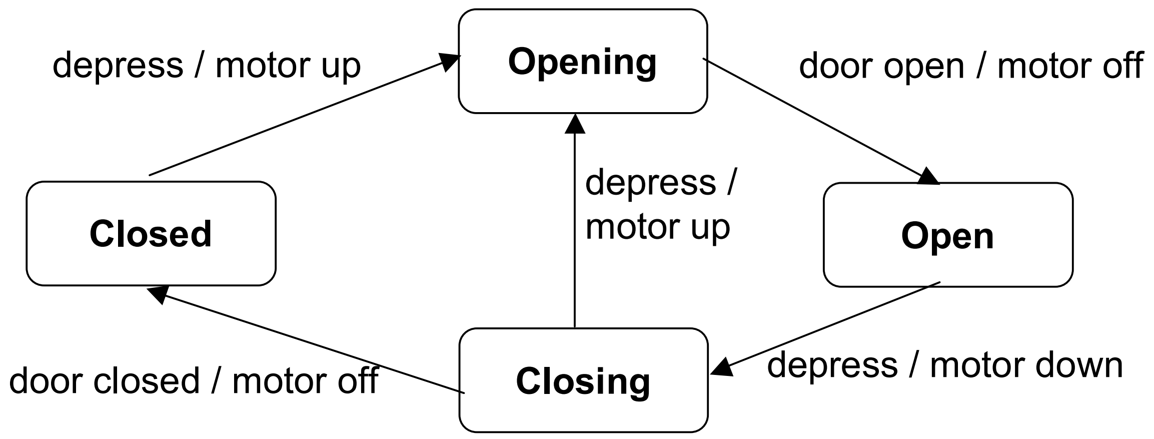
- **Do-Activities** are those activities that continue for an extended time.



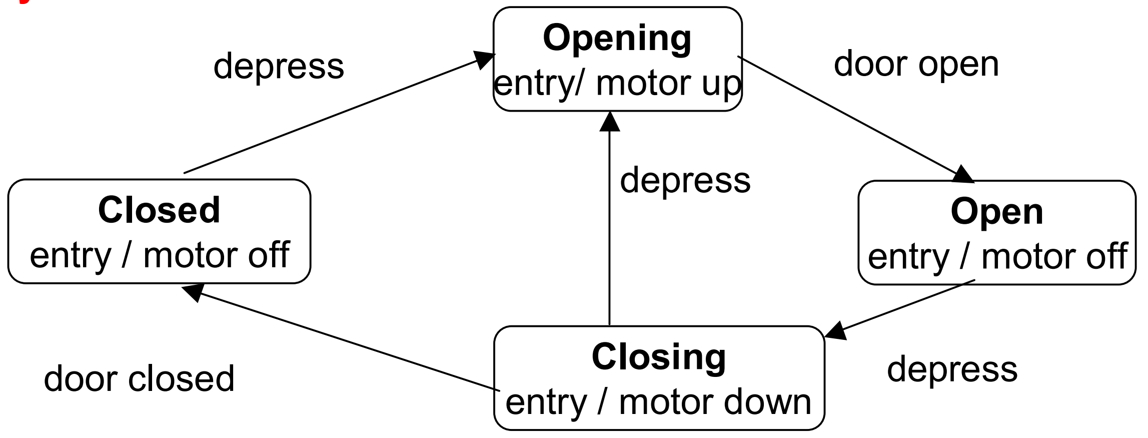
Example

- **Example: Garage door opener – The user generates *depress* events with a push button to open and close the door. Each event reverses the direction of the door, but for safety reasons the door must open fully before it can be closed. The control generates *motor-up* and *motor-down* activities for the motor. The motor generates *door open* and *door closed* events when the motion has been completed. [Assume that the door is initially closed]**

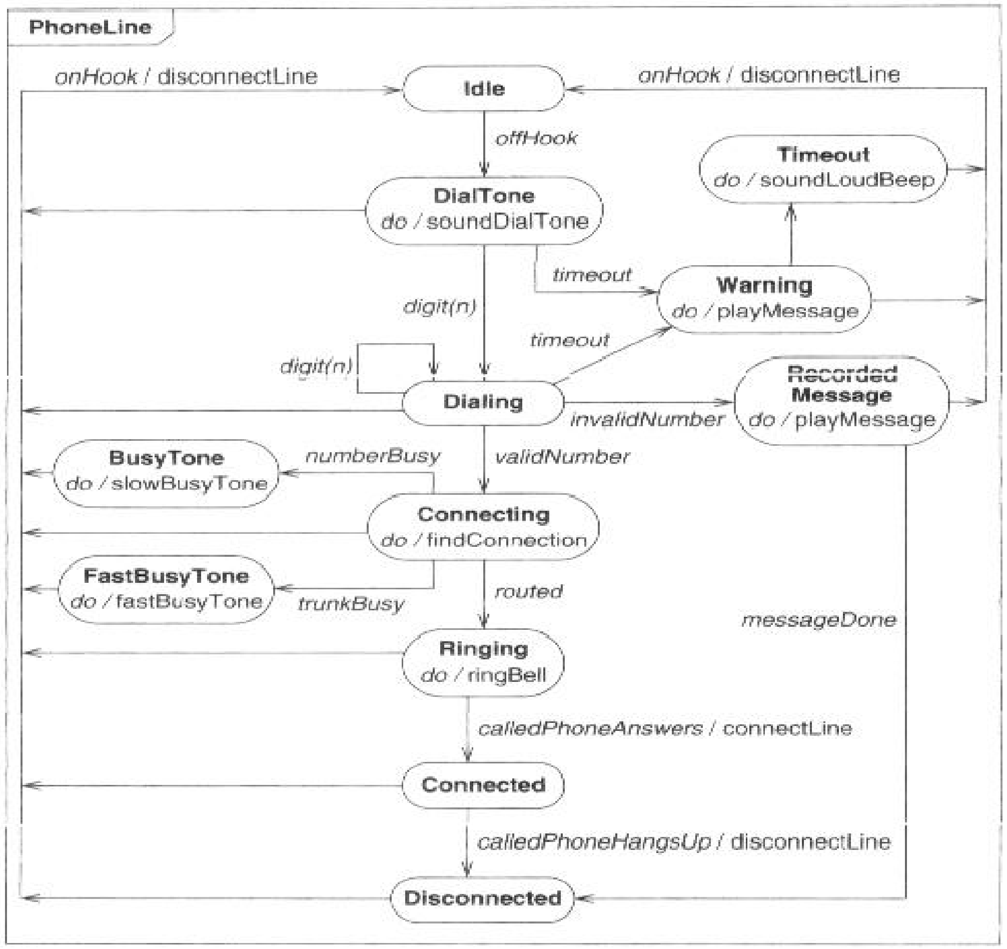
Solution



Alternatively :

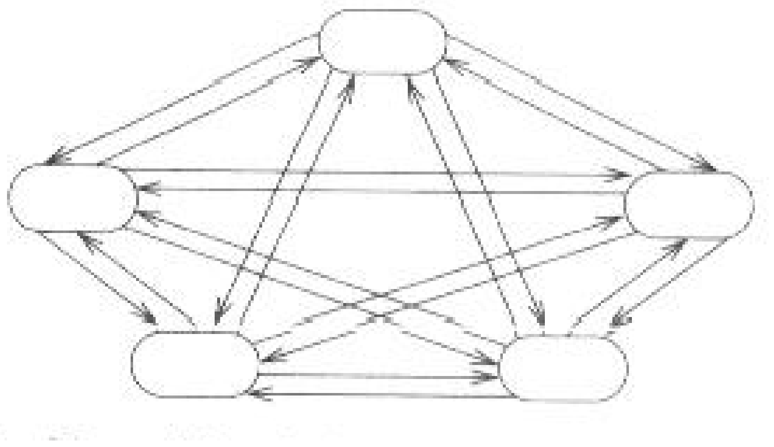


Phone Line Example with Activities



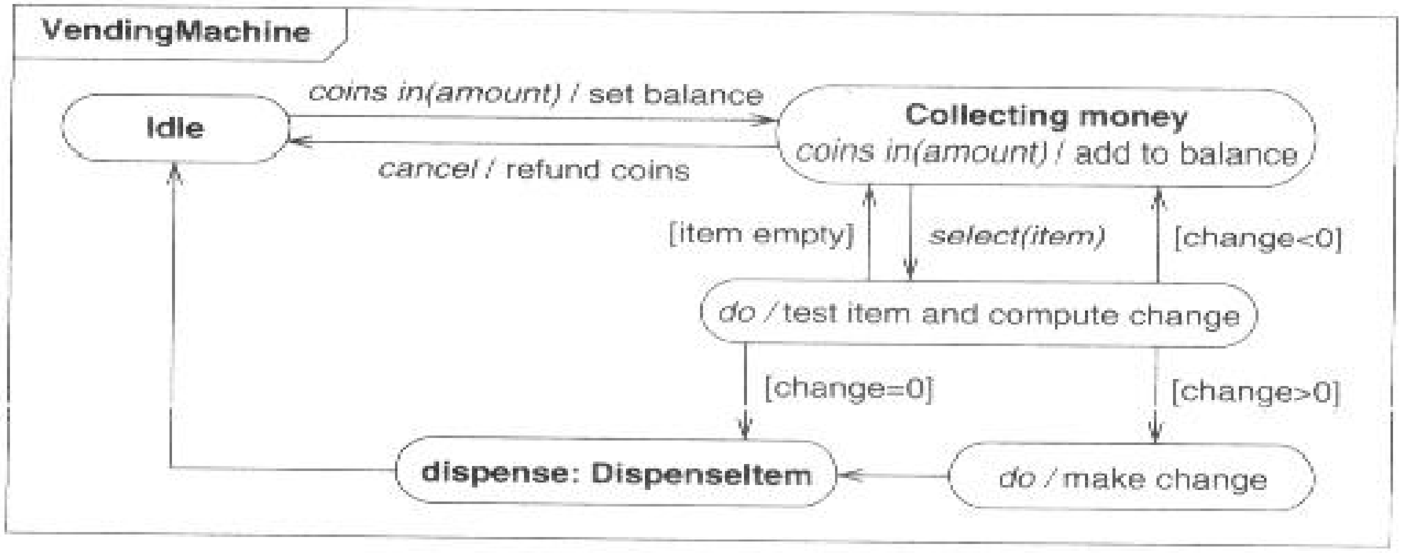
Nested State Diagrams

- **Problems with Flat State Diagrams**
  - **Impractical for large problems**
  - **Consider *n* independent boolean attributes – the flat state diagram would require 2^n states.**
  - **Solution: Partition the state into *n* independent state diagrams – 2n states are required**



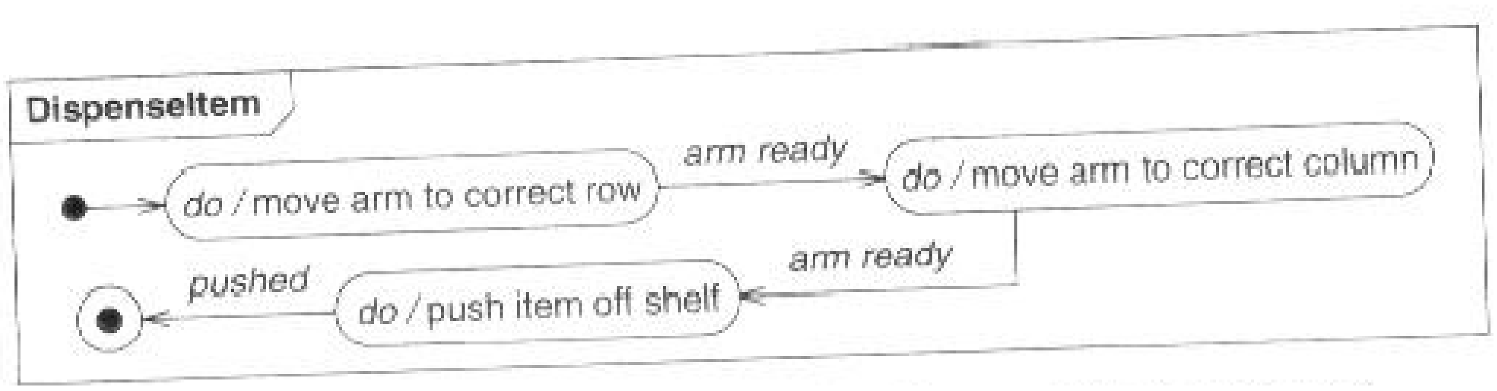
Expanding States

- **Expanding states:** One way to organize is to have a high-level diagram with subdiagrams expanding certain states.
  - Similar to macro substitution



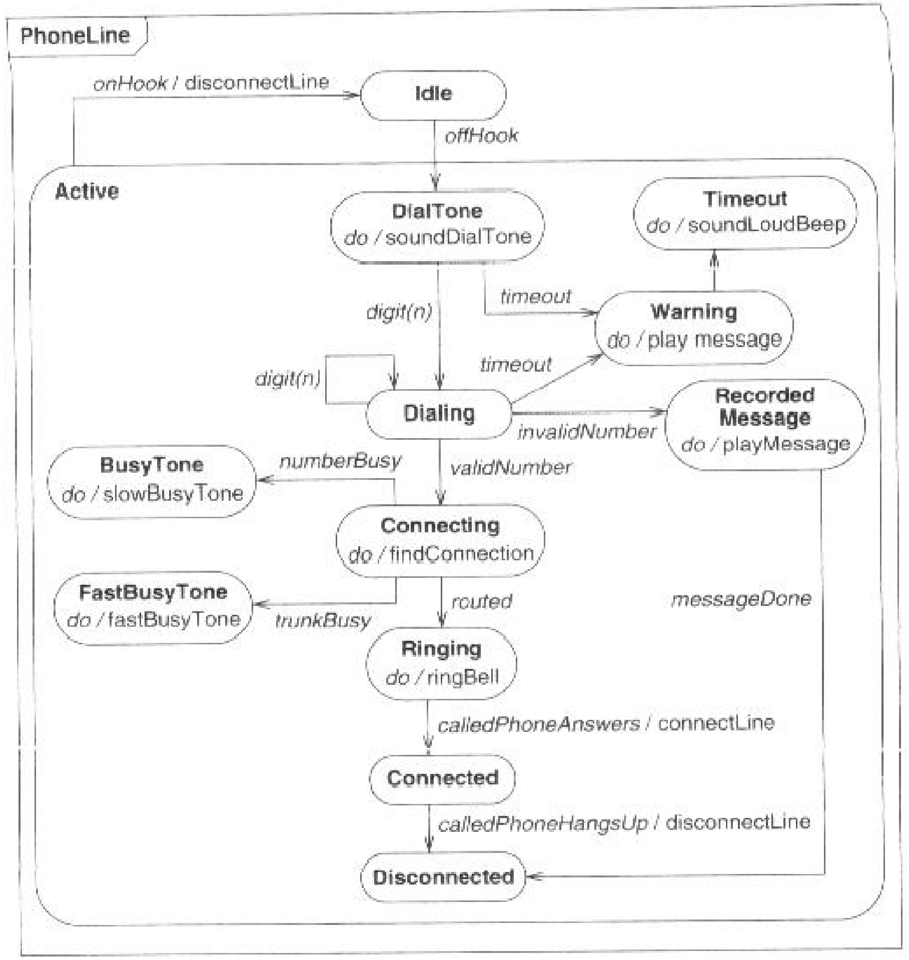
Expanding States

- **Expanding states:** One way to organize is to have a high-level diagram with subdiagrams expanding certain states.
  - Similar to macro substitution

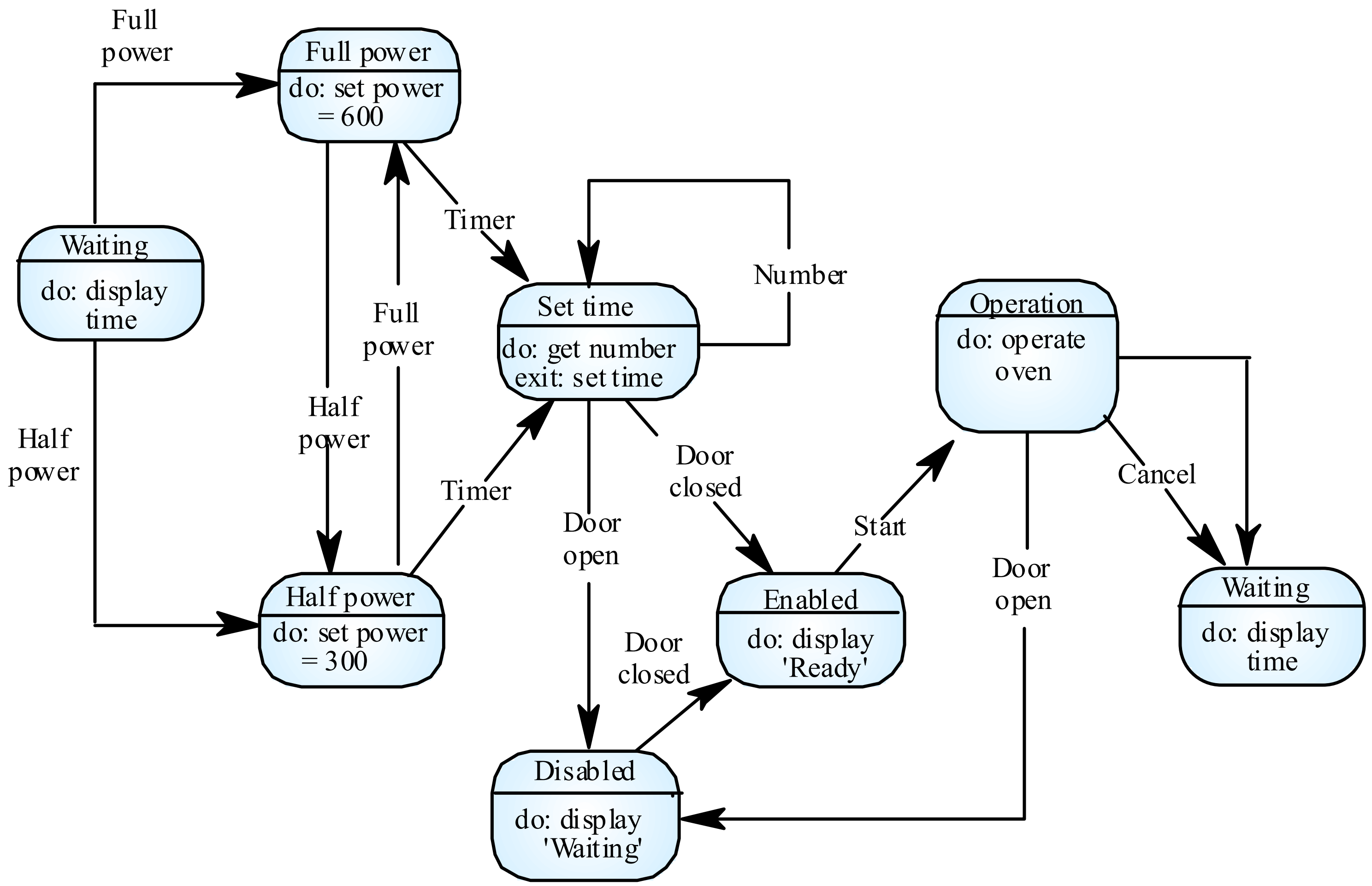


Nested States

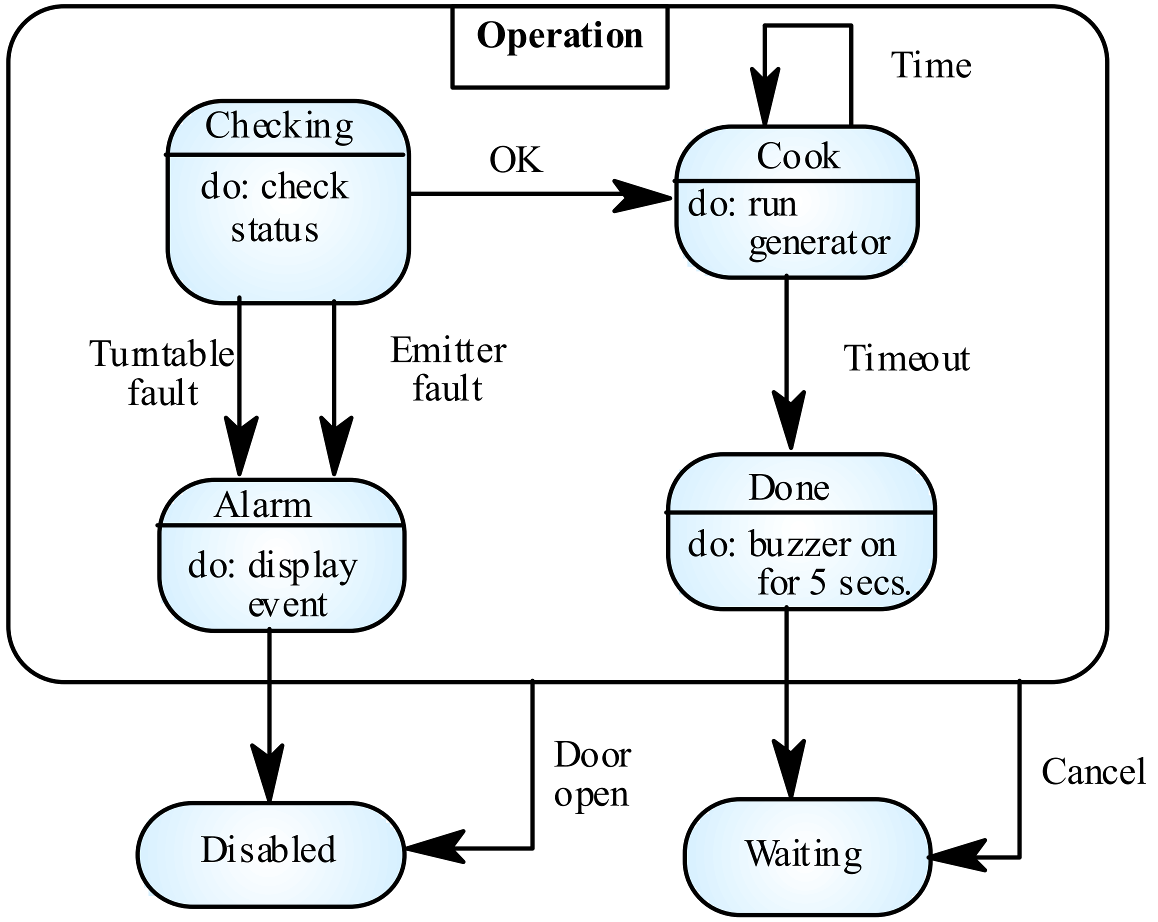
- **Nested states:** Structure states more deeply than just replacing a state with a submachine.
- ***Active*** **is a composite state with regard to nested states** ***DialTone, TimeOut, etc.***



Nested States

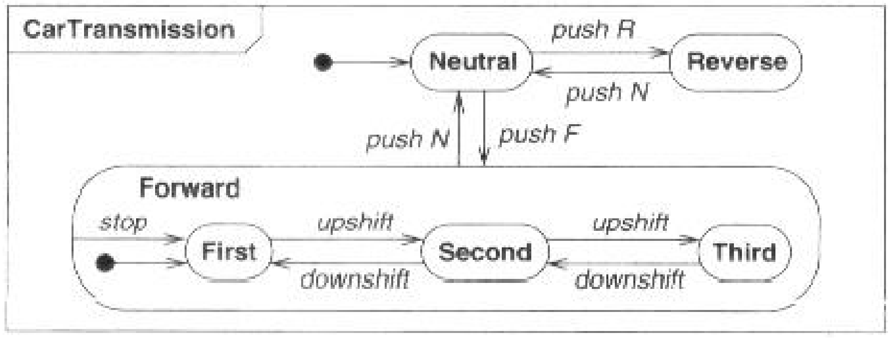


Nested States



Nested States

- **Nested states:** You can nest states to arbitrary depth.
- A nested state receives the outgoing transitions of its composite state.

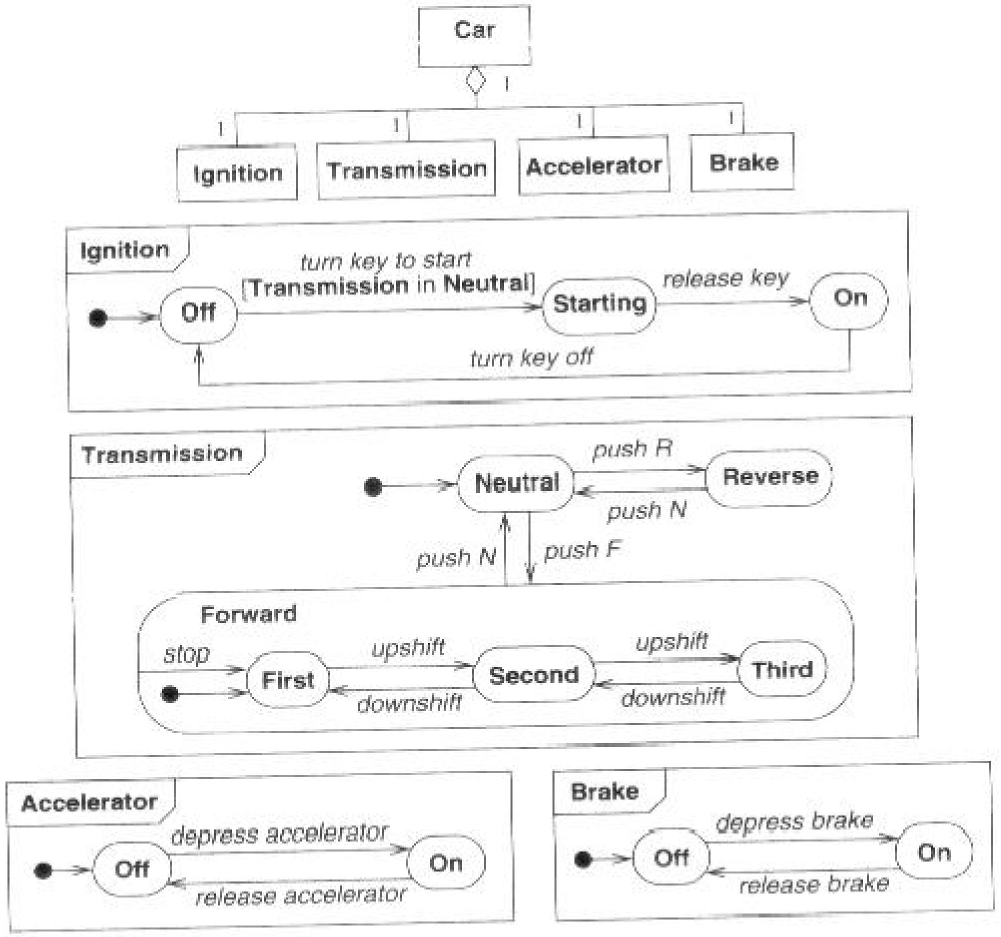


Concurrency

- The state model implicitly supports concurrency.
- Objects are autonomous entities that can act and change state independent of another.
- Two types of concurrency:
  - Aggregation Concurrency –
    - A state diagram for an assembly is a collection of state diagrams, one for each part.
    - The aggregate state is one state from each diagram.
  - Concurrency within an Object is based on partitioning objects into subsets of attributes, each one of which has its own state diagrams.

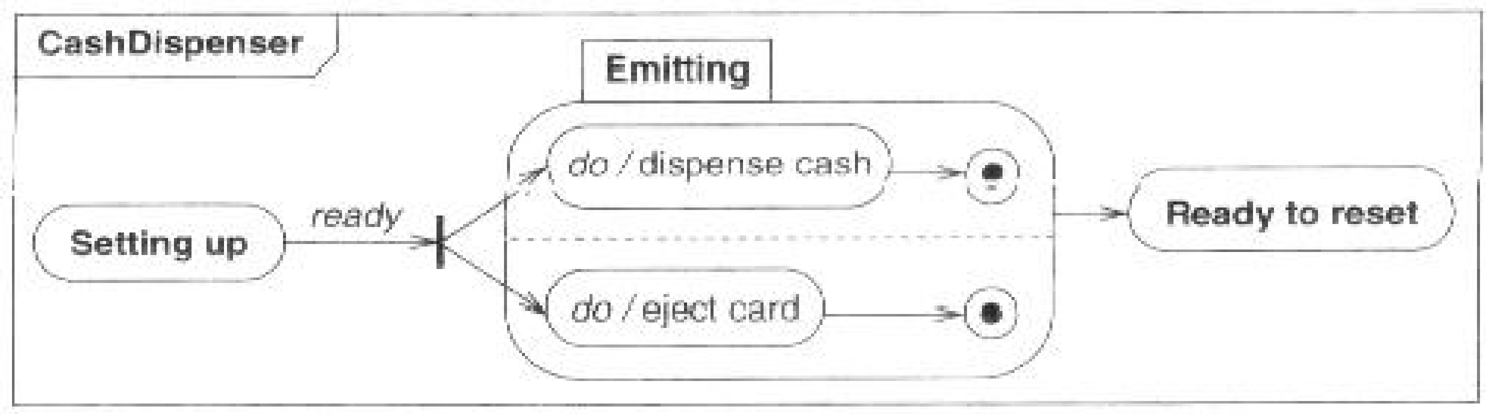
Aggregation Concurrency

- Car is an aggregation of *Ignition*, *Transmission*, *Accelerator*, and *Brake* components.
- Transition for one object may depend the state of another.



Concurrency within an Object

- Dashed arrow indicates concurrently executing state machines.
- A single event may cause transitions in more than one state machine.
- A transition that forks indicates splitting of control into two concurrent parts.


CashDispenser
Emitting
Setting up
ready
do / dispense cash
do / eject card
Ready to reset