# UML State Modeling

Alex X. Liu

## Motivation and Overview

Some objects in a system have complex *temporal behaviors*, which must be carefully designed

- E.g., modern interactive and distributed applications
  - Typically comprise multiple active objects
  - Use locking primitives to synchronize threads
- E.g., embedded systems where software controls devices
  - Devices run "in parallel" with controller
  - Communicate with one another by signaling

Design problems:

- e.g., race conditions and synchronization anomalies
- e.g., lost or unexpected signals, deadlock

**Issue:** How to design to prevent these problems

## Potential problems in such systems

Controller enters a state in which it is no longer *receptive* to signals from its environment

- Signals may arrive but have no effect
- Controller may prevent issuing of signals
  - E.g., greying out of buttons in a graphical dialog box
  - E.g., my former car Chrylser 300, battery in trunk and the trunk is electronically controlled.

Controller enters a state in which it is receptive to some signals, but not those that are being offered by the environment

Controller expects some peer to be in a state that is ready to receive a signal, sends the signal, but the peer isn't ready

## Problems (continued)

The bad news: most of these problems *cannot* be reliably detected and fixed via testing

- Some are "race conditions"
  - Depend on how the various actors are scheduled by OS
  - Difficult to reproduce
- Very difficult to simulate all possible interactions with an environment
  - Often we test our programs under lots of assumptions about how they will be used
  - These assumptions often turn out to be naive

## Current state of the practice…

Relies on "designing out" these problems rather than trying to uncover and reproduce them after the fact

Aided by *finite state modeling and analysis* of software architectures

- Model each entity in the system as a communicating finite state machine
- Simulate interactions between state machines, looking for flaws

*Model checking:* Attempts to exhaustively analyze a system specified in this fashion

## Finite-state models

Describe temporal/behavioral view of a system

Specify *control:*

- Sequence operations in response to stimuli
- Distinguish *states, events,* and *transitions*
- Especially useful during design

UML state diagrams

## Key terms

*Event:* occurrence at a point in time
- instantaneous
- often corresponds to verb in past tense
  - e.g., alarm set, powered on
- or onset of a condition
  - e.g., paper tray becomes empty, temperature drops below freezing

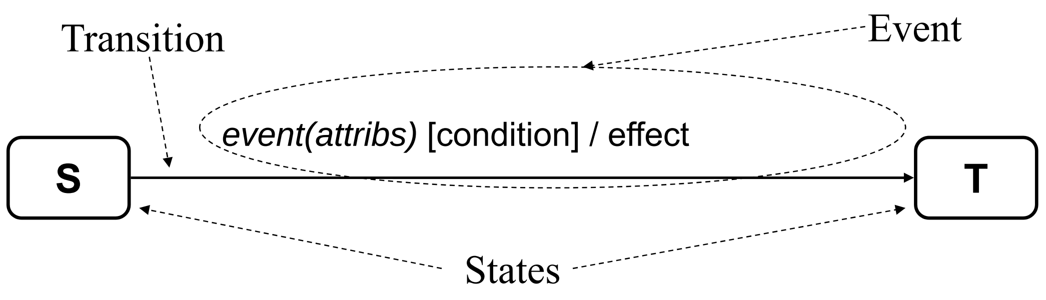
*State:* behavioral condition that persists in time
- often corresponds to verbs with suffix of "-ing"
  - e.g., *Boiling, Waiting, Dialing*
- in OO terms: **an abstraction of values of attributes and configuration of objects**

*Transition:* instantaneous change in state
- triggered by an event

## State Diagrams

Graphical state-modeling notation:
- States: labeled roundtangles
- Transitions: directed arcs, labeled by triggering event, guard condition, and/or effects

Example:



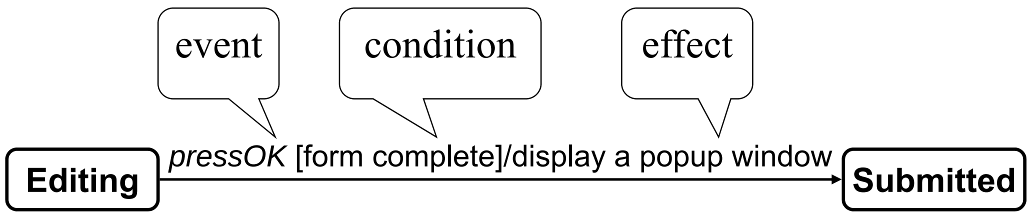
## Enabling and firing of transitions

**Transition is:**

- *enabled* when source state is *active* and guard condition *satisfied*
- *fires* when enabled and the triggering event occurs

**Example below:**

- enabled when current state is Editing and the form is complete
- fires when the user presses the “OK” button
- effect is the display of a popup window



## Kinds of events

**Signal event:**

- occurrence of a *signal*
  - an explicit one-way transmission of information
  - may be parameterized
    - E.g., *stringEntered(“Foo”)*
- Sending of a signal by one object is a distinct event from its reception by another
- Signal vs. signal event
- Example:
  - flight departures,
  - button pressed,
  - string entered,
  - digit dialed

**Change event:**

- Event caused by satisfaction of a Boolean expression
- Intent: Expression continually tested; when changes from false to true, event happens
- Notation: ***when****(bool-expr)*
- Example:
  - When (temperature < 0)
  - When (tirepressure < minimum pressure)
  - When (time=November 20, 2007)

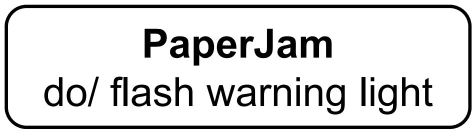
**Time event:**

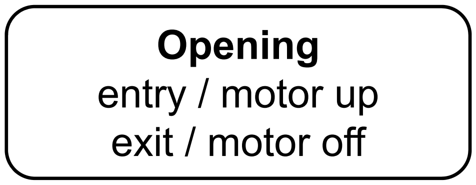
- Example:
  - ***after*** *(10 seconds)*

## Activities

**Often useful to specify an *activity* that is performed within a given state**

- **E.g., while in PaperJam state, the warning light should be flashing**
- **E.g., on entry into the Opening state, the motor should be switched on**
- **E.g., upon exit of the Opening state, the motor should be switched off**

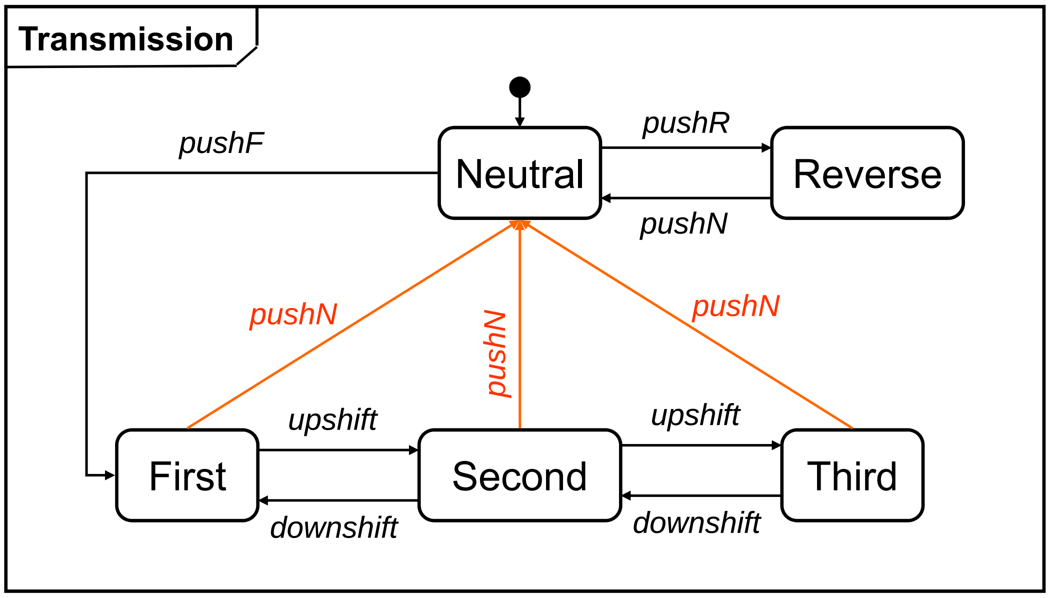




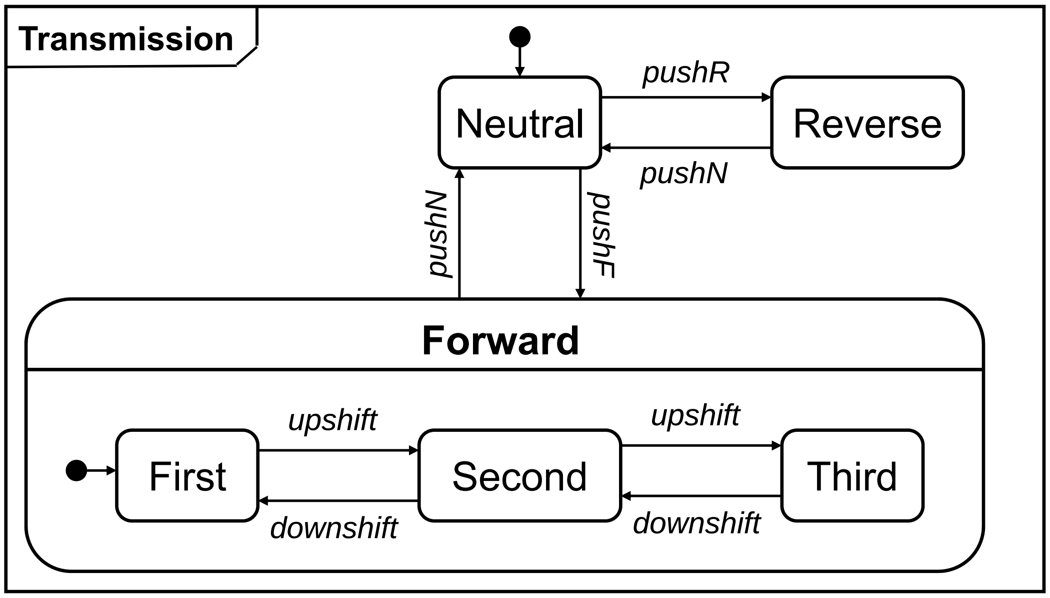
## Two sources of state explosion

- **Multiple transitions with same triggering event, guard condition, and effect but different source and/or target states**
  - **Solution: state generalization**
- **State explosion due to concurrency and/or orthogonality**
  - **Solution: parallel composition**

## Problem: Multiple similar transitions



## Solution: State generalization



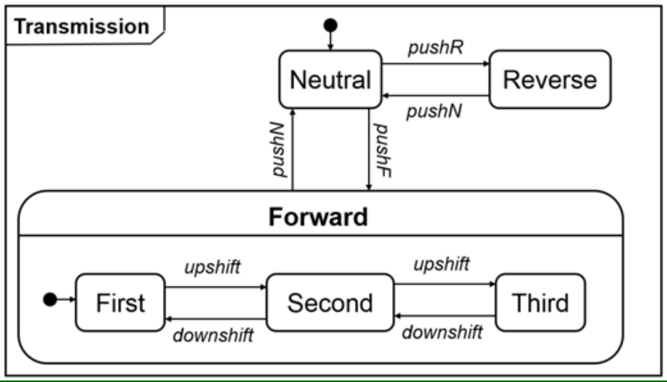
## State generalization

Introduces an abstract “super state”:

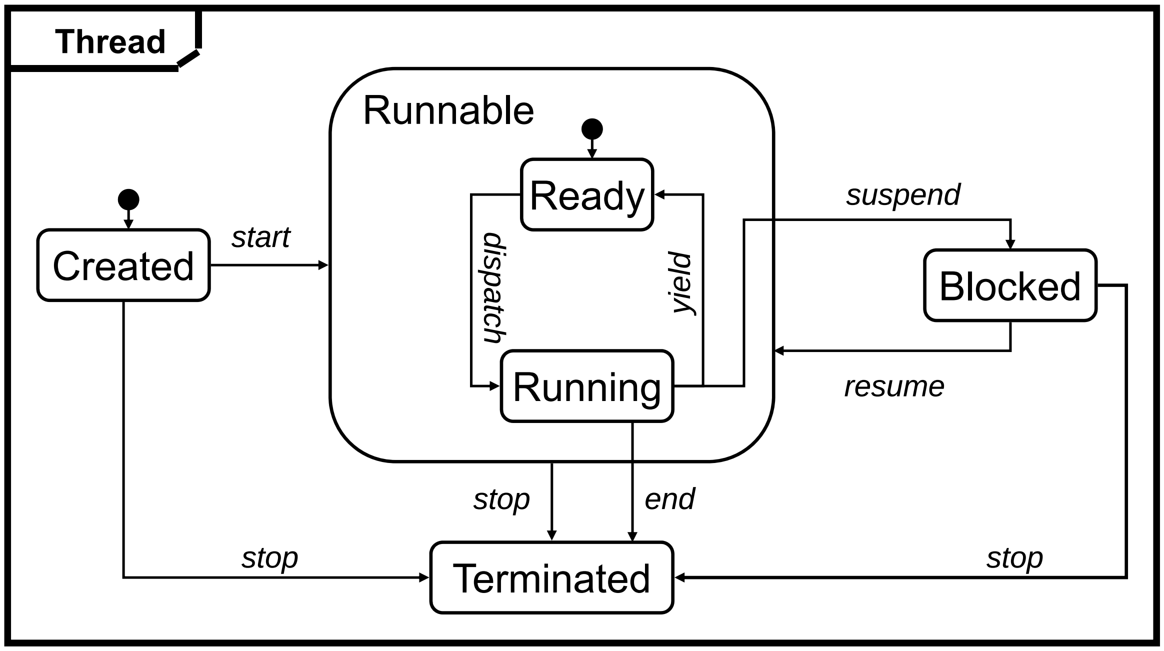
- decomposes into multiple substates
- when super state is active, exactly one of its substates is active

Outbound transition incident on superstate abbreviates set of transitions, one from each substate

Inbound transition incident on superstate enters substate that is distinguished as the start state



## Example: Lifecycle of a thread



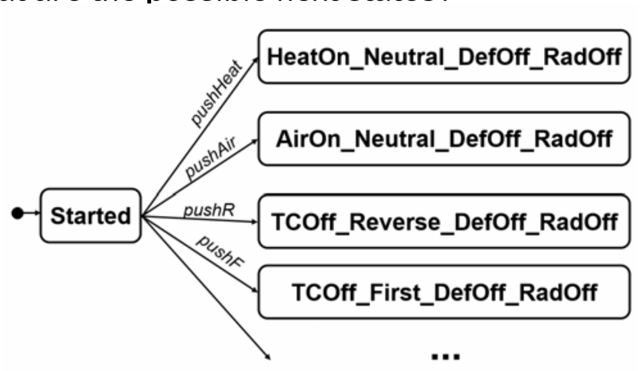
## Problem: Composite behaviors

Consider an automobile with multiple options:

- Automatic transmission
- Temperature control (heating/air)
- Rear-window defroster
- Stereo system

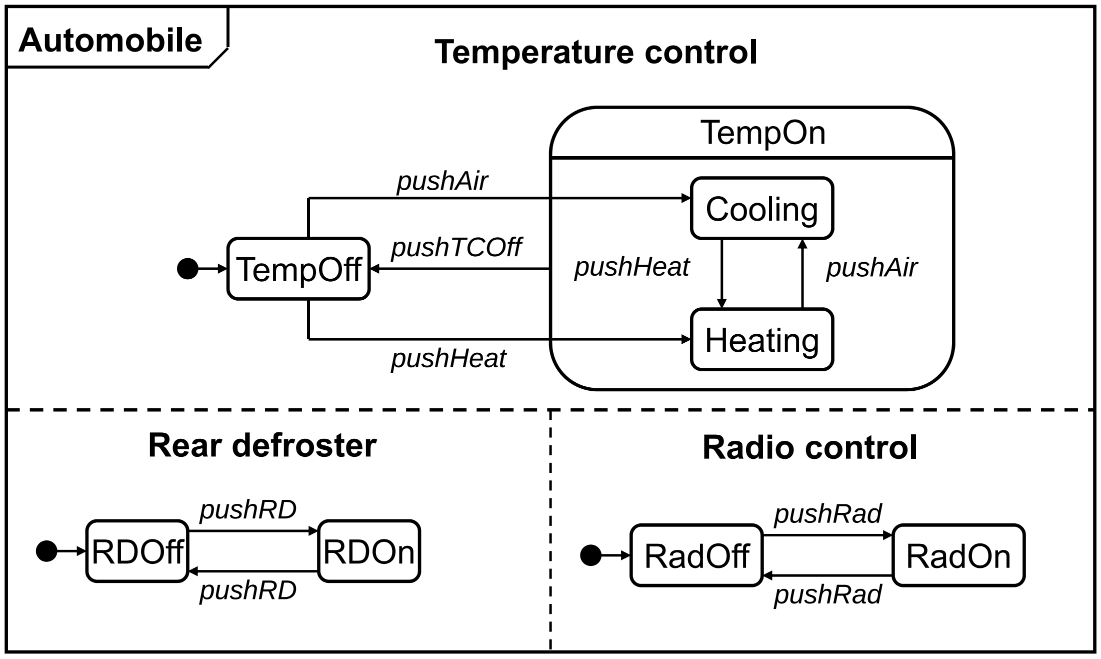
Suppose we wish to construct a state diagram for the autmobile:

- Assume car starts with transmission in neutral and temp control, rear defroster, and stereo are all off
- What are the possible next states?



Started
pushHeat
pushAir
pushR
pushF
HeatOn_Neutral_DefOff_RadOff
AirOn_Neutral_DefOff_RadOff
TCOff_Reverse_DefOff_RadOff
TCOff_First_DefOff_RadOff
...

## Solution: Parallel Composition



## Reading Assignment

- **Rumbaugh book Chapter 5 “State Modeling”**