Aggregation, Composition, and Dependence Relationships Among Classes

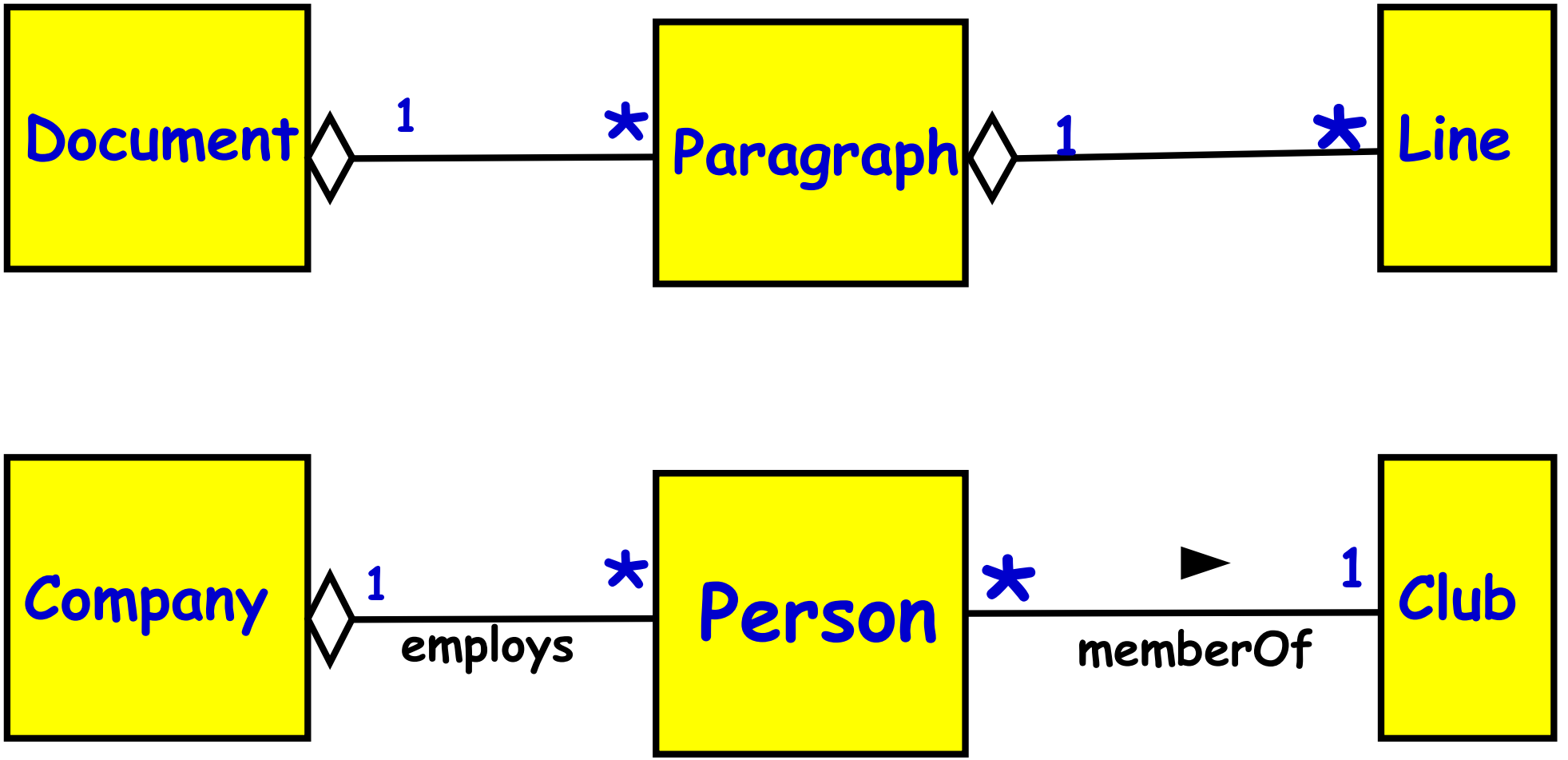
Lect 5—6
14-08-2023

Aggregation Relationship

- Represents whole-part relationship
- Represented by a diamond symbol at the aggregator end.

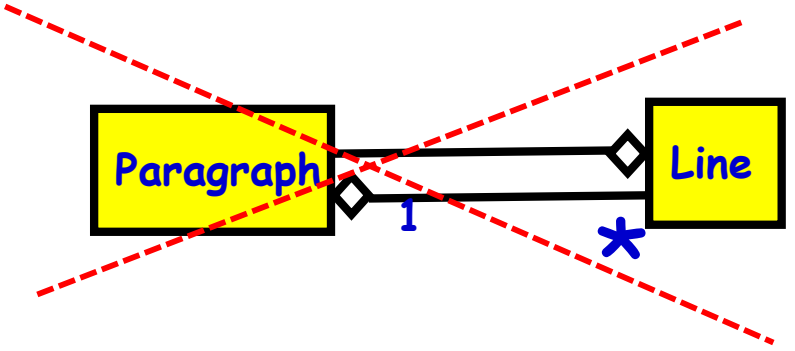
- Often indistinguishable from plain association.
  However:
  - **Aggregate usually creates the components.**
  - **Aggregate usually invokes the same operations on all its components.**
- Usually aggregate is owner of the components:
  - But may share with other objects

Aggregation: Two Examples



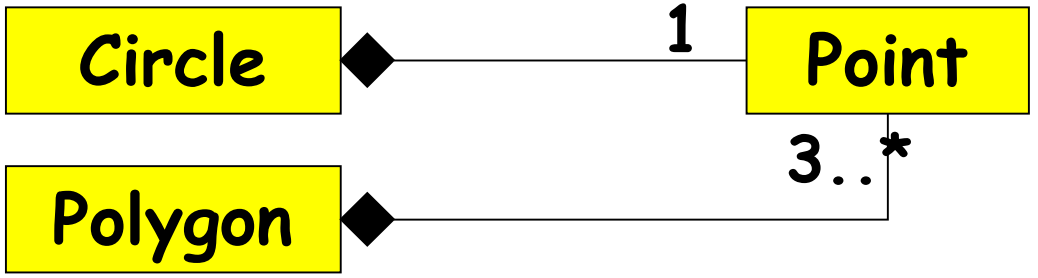
Aggregation cont…

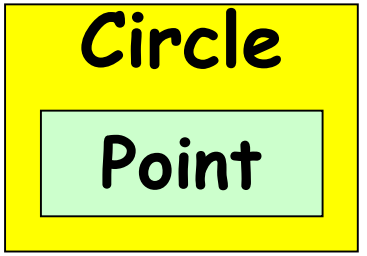
- An aggregate object contains other objects.
- Aggregation limited to tree hierarchy:
  - **No circular inclusion relation.**



Composition

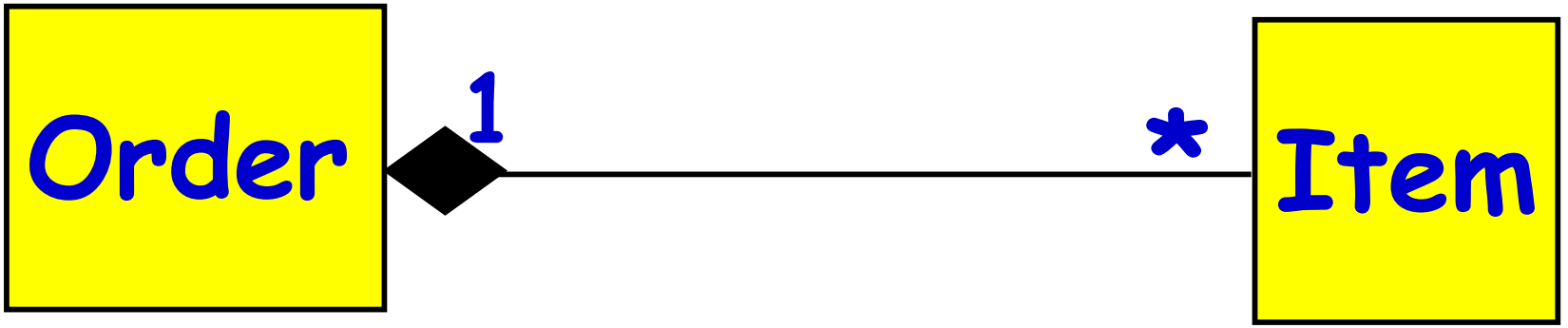
- A stronger form of aggregation
  - The whole is sole owner of its part.
    - A component can belong to only one whole
  - **The life time of the part is dependent upon the whole.**



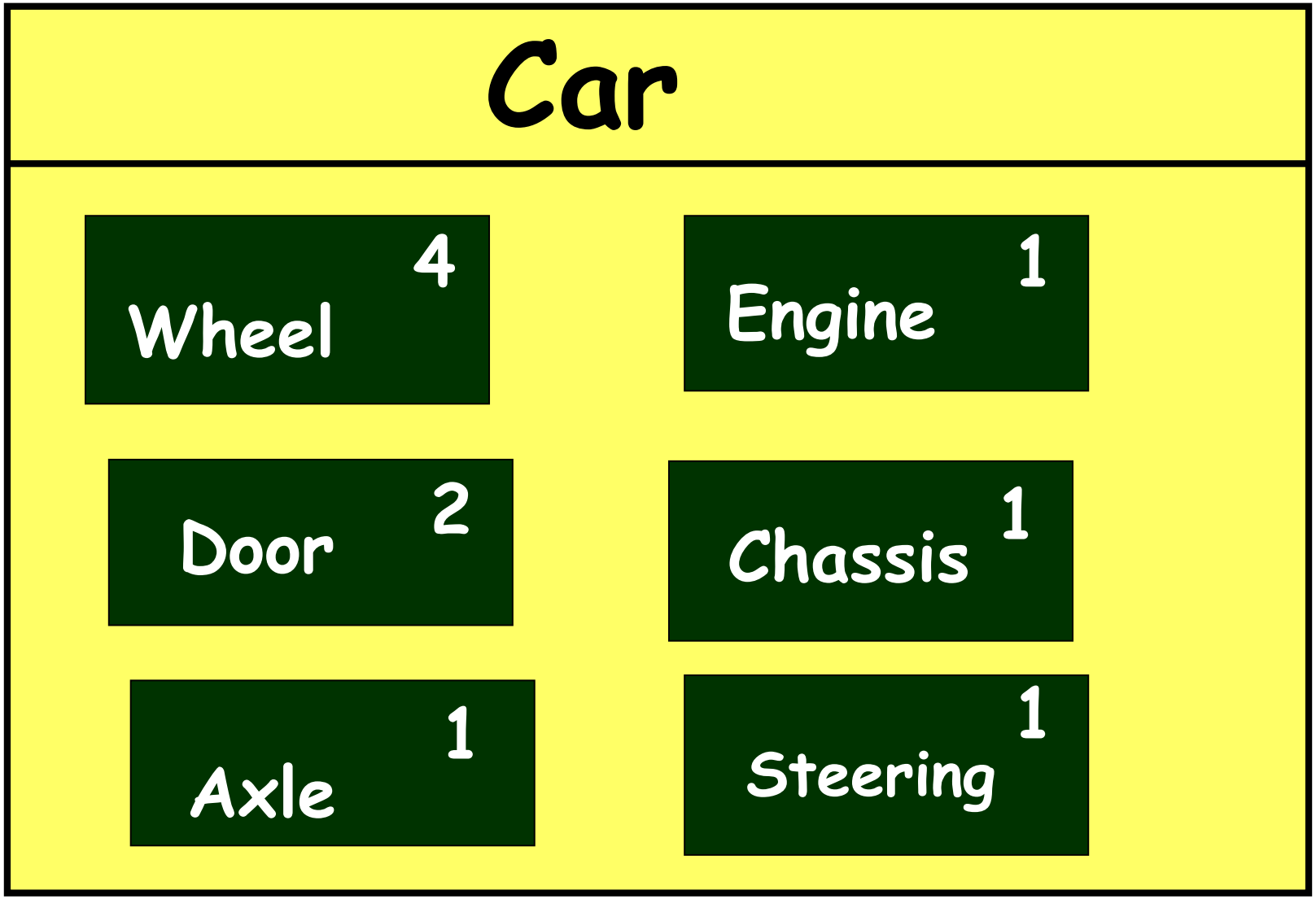


Composition Relationship

- Life of **item** is same as that of **order**

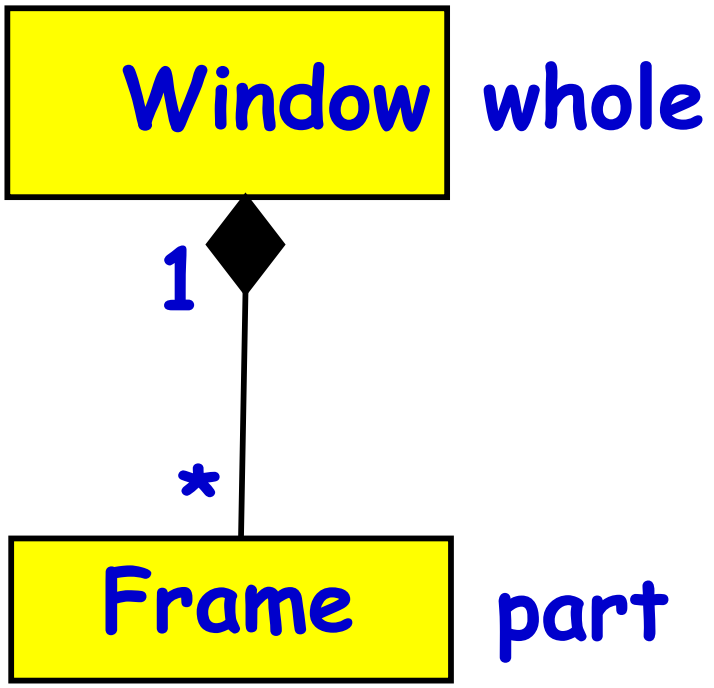


Composition: Alternate Notation



Composition

- An object (component) may be a part of ONLY one composite.
  - Whole is responsible for the creation and disposition of its parts.

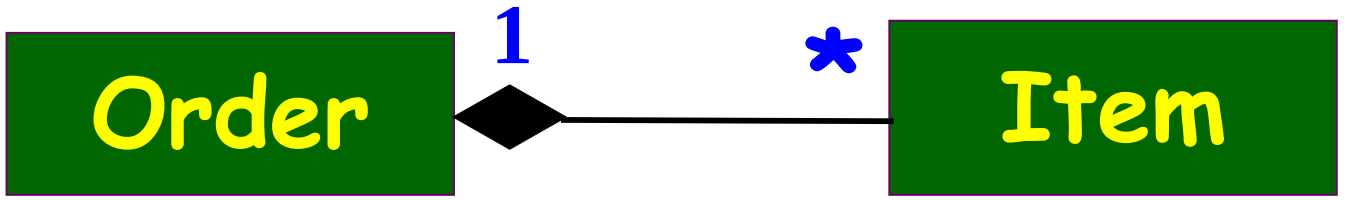


Aggregation vs. Composition

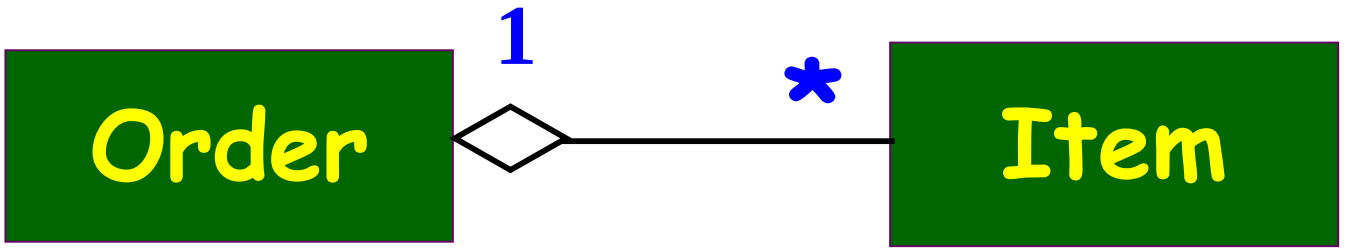
- Composition:
  - Composite and its components have the same life line.
- Aggregation:
  - Lifelines are different.
- Consider an order object:

  - **Aggregation**: If order items can be changed or deleted after placing an order.
  - **Composition**: Otherwise.

Composition versus Aggregation



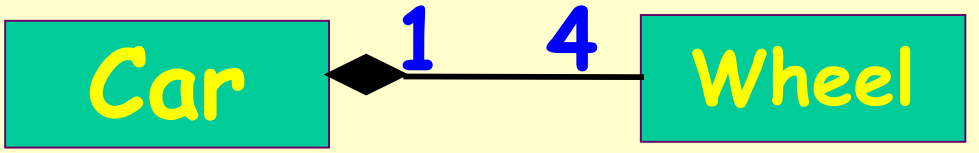
Composition



Aggregation

Implementing Composition…

```
public class Car{
  private Wheel wheels[4];
  public Car (){
          wheels[0] = new Wheel();
          wheels[1] = new Wheel();
          wheels[2] = new Wheel();
          wheels[3] = new Wheel();
      }
  }
```

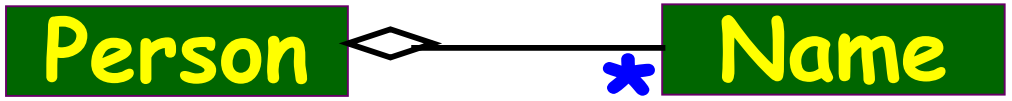



WE NEED TO HIRE THE BEST MARKETING EXPERT WE CAN FIND.
www.dilbert.com scottadams@aol.com
YOUR RÉSUMÉ SAYS YOU'VE WON THE NOBEL PRIZE IN MARKETING, AND FIVE OLYMPIC GOLD MEDALS IN THE MARKETING BIATHLON.
3-30-06 ©2006 Scott Adams, Inc./Dist. by UFS, Inc.
WHAT'S A MARKETING BIATHLON?
YOU SKI UP TO PEOPLE WHO WON'T BUY YOUR CRAP AND YOU SHOOT THEM.

Aggregation: Code Example

- An aggregation relationship is usually represented as a data field in the aggregated class.

```
public class Person {
  /** Other Data fields */
private ArrayList<Name> name =new ArrayList<Name>() ;

}
```



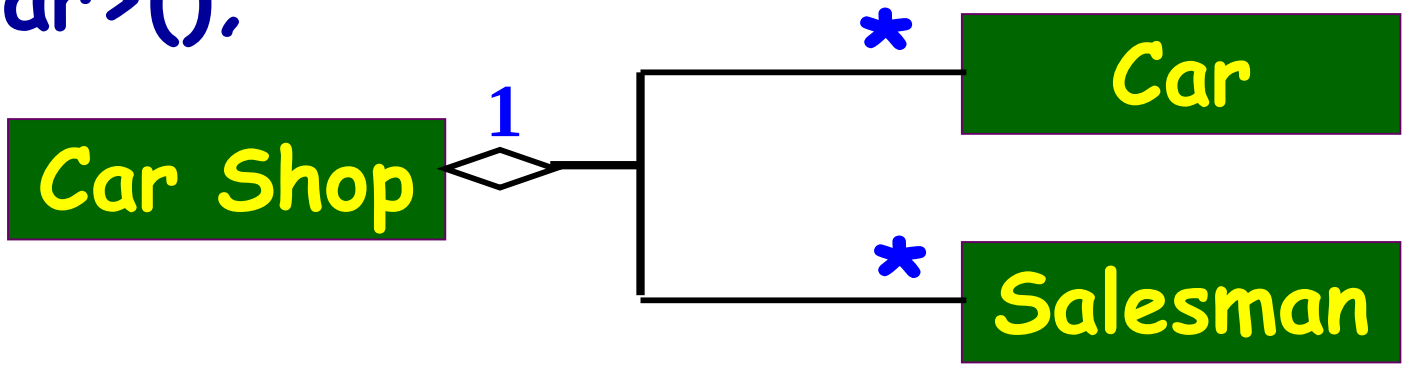
Often Inner Classes are Used

- **If House is used only in the Person class**:
  - Declare it as an inner class in Person.

```
public class Person {
  private Name name;
  private House house;
  ...

  class House {
  ...
  }

}
```

Implementing Aggregation: Ex 1

```
import java.util.ArrayList;

public class CarShop{

  CarCompany company; Manager manager;

  private ArrayList<SalesPerson> people =
  new ArrayList<SalesPerson>();

  private ArrayList<Car> cars = new
  ArrayList<Car>();
}
```



Deciding Whether to Use Composition or Aggregation…

- Use composition if:
  - Lifetime of part is bound with lifetime of composite
  - There is an obvious physical or logical assembly
  - Some properties of composite propagate to parts (e.g., location)
  - Operations applied to composite propagate to parts (e.g., destruction, movement, etc)

Aggregation versus Composition

```
public class A{
    public void operation(B b) {
        b.operation();
    }
}
public class B{
    public void operation() {
    }
}
```

```
public class A {
    private B b;
    public A(B b){
        this.b = b;
    }
    public void operation(){
        b.operation();
    }
}
public class B {
    public void operation() {
    }
}
```

Class Dependency

Any change to the independent class would necessitate a change to the dependent class.

A class may be dependent on another class due to a variety of reasons.

Dependency

- Dependency relationship between two classes X and Y can arise due to a variety of reasons:
  - X has an attribute of class Y
  - X has a template attribute with a parameter of class Y
  - X has a method with an input argument of class Y
  - X has a method with an output argument of class Y
  - X has of a method containing a local variable of class Y
  - Etc.

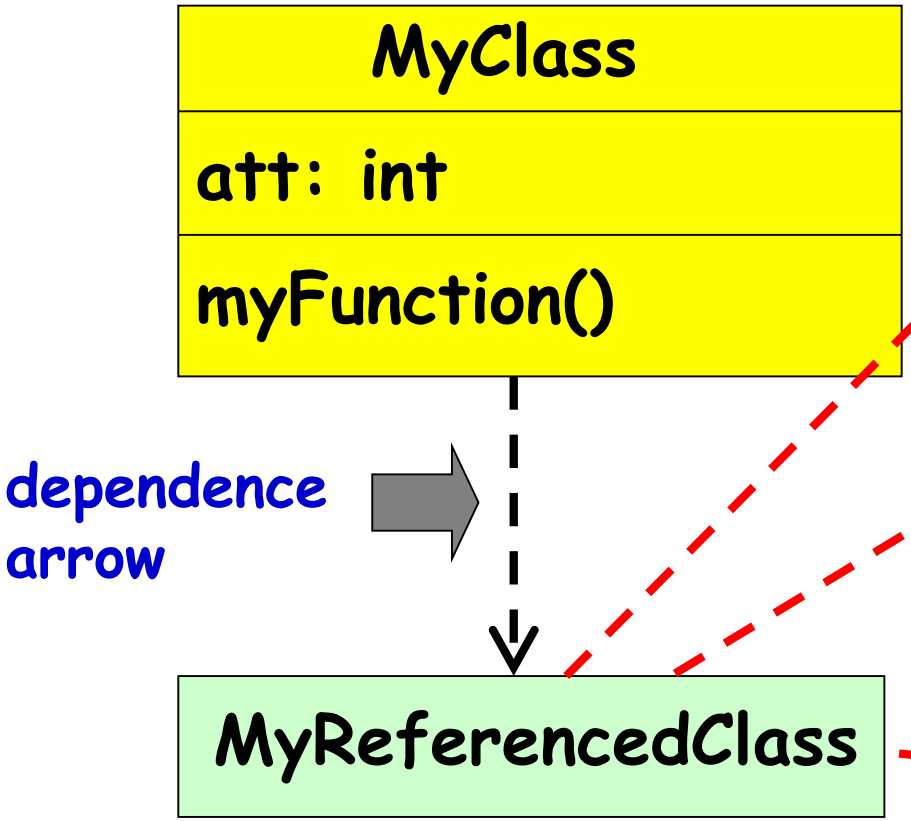
Dependency

- Common Dependences are caused by:
  - Local variables
  - Parameters
  - Return values
- Example:

```
Class A {
   B Foo(B x) {
        B y = new B();
      return y;
   }
}
```

```
Class B {
                ...
                ...
             ...
}
```

Dependence - Examples



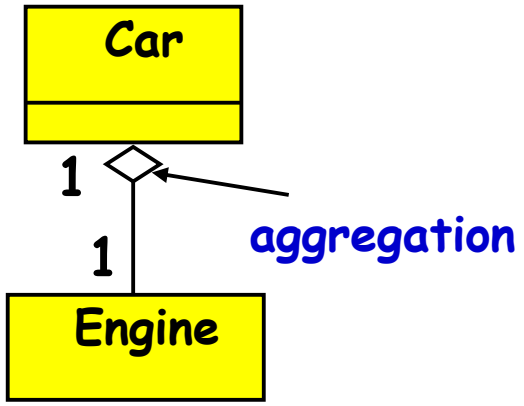

```
class MyDependentClass{

  void myFunction1(
  MyReferencedClass r ) {
  .. }

  MyreferencedClass
  myFunction2( .. ) { .. }

  void myFunction3( .. ){
  MyReferencedClass m .. }

}
```

Association Vs. Aggregation

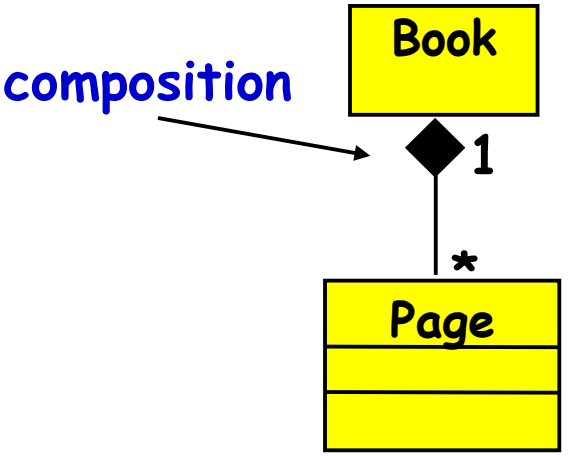
- Is aggregation an association?
- Is composition an aggregation?

Summary of Three Class Relations

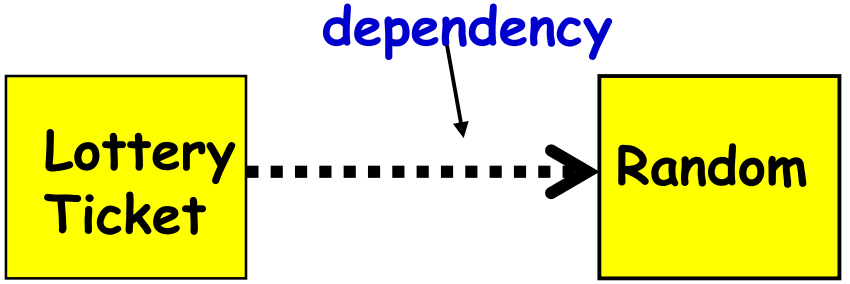
- **aggregation: "is part of"**
  - Symbolized by empty diamond



- **composition: "is made of"**
  - Stronger version of aggregation
  - The parts live and die with the whole
  - Symbolized by a filled diamond



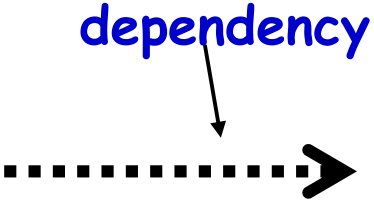
- **dependency: "Depends on"**
  - Represented by dotted arrow.



UML Class Relations: Notation Summary

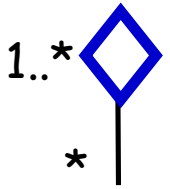
Class Generalization Relationship

dependency

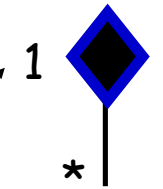


Object Association

Object Aggregation Association

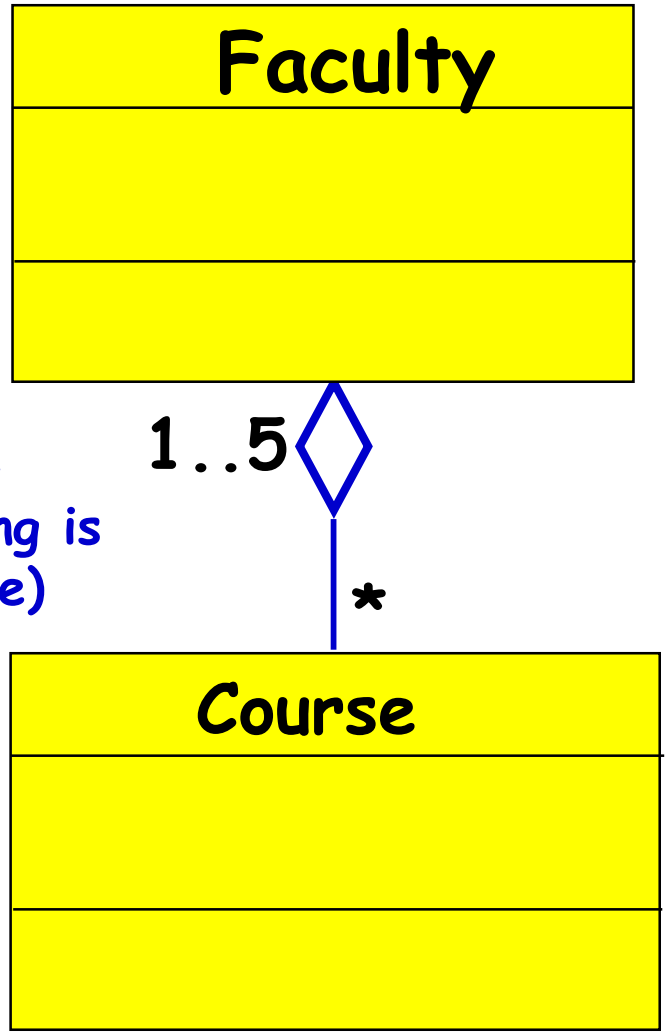


Object Composition Association



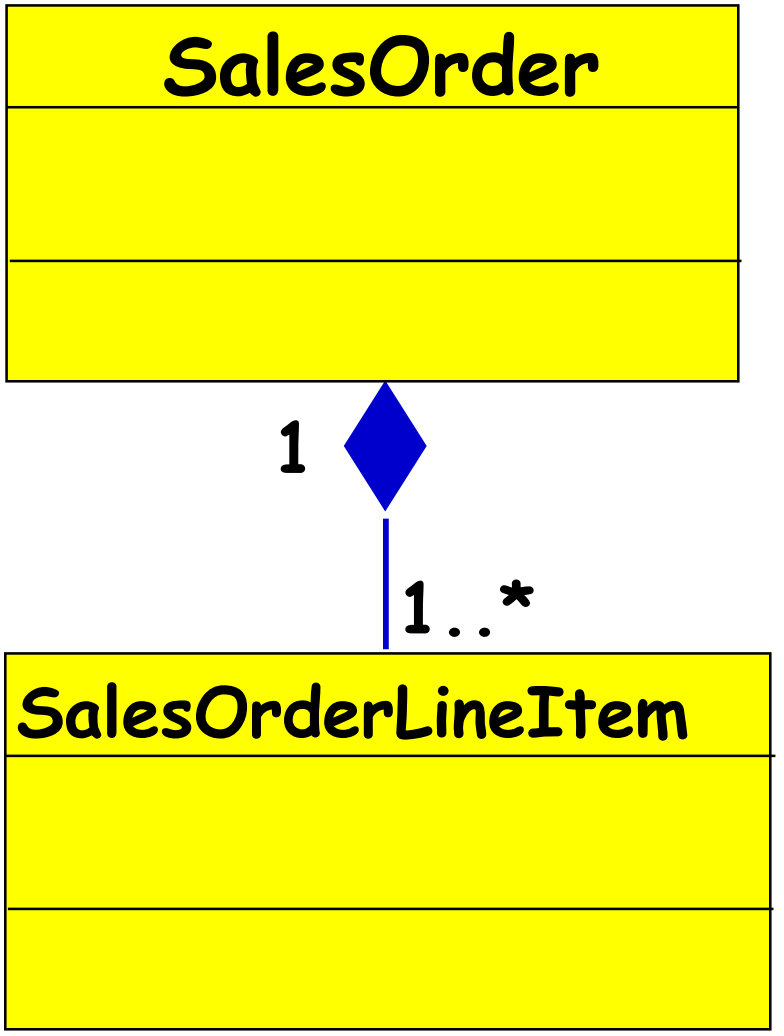
Will always be "1"

Aggregation



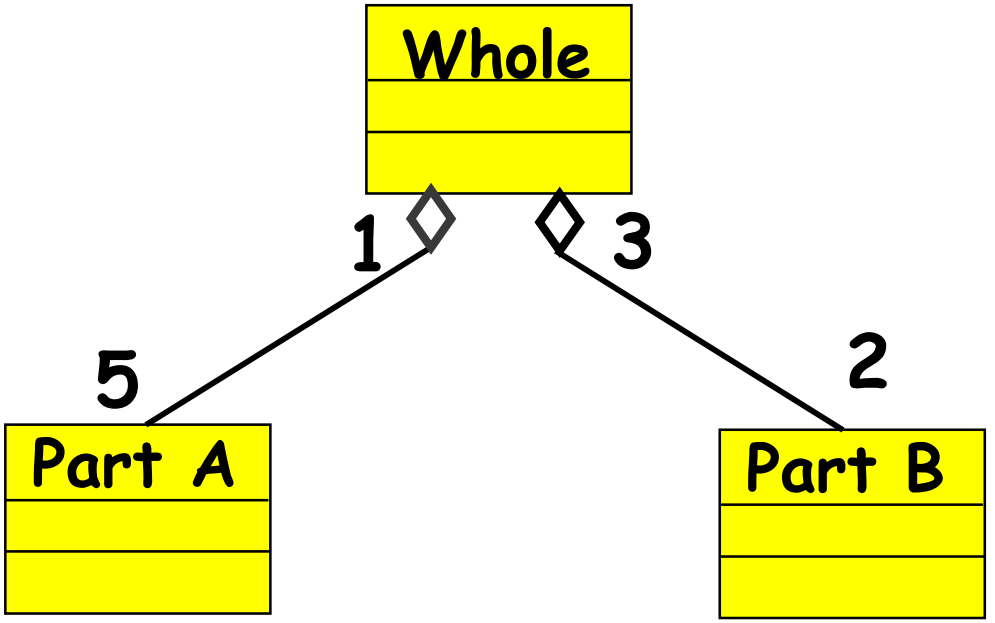
(team-teaching is possible)

Composition



Multiplicity Quiz #1

Class diagram

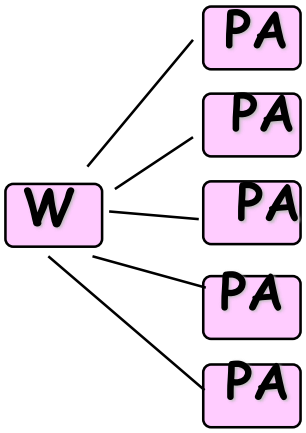


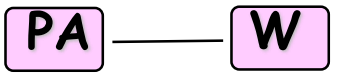
Read

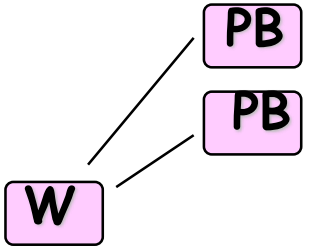
- One Whole is associated with 5 Part A
- One Part A is associated with 1 Whole

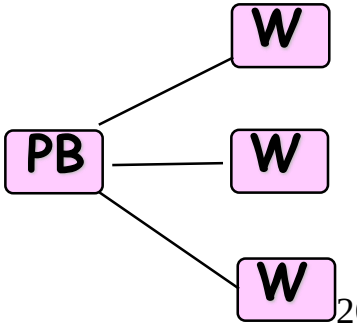
- One Whole is associated with 2 PartB
- One PartB is associated with 3 Whole

Object diagram









Class Relation Hints

- **Composition**
  - B is a permanent part of A
  - A contains B
  - A is a permanent collection of Bs
- **Subclass / Superclass**
  - A is a kind of B
  - A is a specialization of B
  - A behaves like B
- **Association (Collaboration)**
  - A delegates to B
  - A needs help from B
  - A invokes service of B.

Class Diagram Inference Based on Text Analysis (based on Dennis, 2002)

- A common or improper noun implies a class e.g. Book
- A proper noun implies an object (instance of a class): CSE Dept, OOSD, etc.
- An adjective implies an attribute e.g. price of book
- A "doing" verb implies an operation (method)
  - Can also imply a relationship e.g. student issues Book
- A "having" verb implies an aggregation relationship
- An adverb implies an attribute of an operation e.g. fast loading of image…

Identify Class Relations

- Faculty & student
- Hospital & doctor
- Door & Car
- Member & Organization
- People & student
- Department & Faculty
- Employee & Faculty
- Computer Peripheral & Printer
- Account & Savings account

Identify Classes & Relations

- A square is a polygon
- Shyam is a student
- Every student has a name
- 100 paisa is one rupee
- Students live in hostels
- Every student is a member of the library
- A student can renew his borrowed books
- The Department has many students

Identify Classes & Relations

- A country has a capital city
- A dining philosopher uses a fork
- A file is an ordinary file or a directory file
- Files contain records
- A class can have several attributes
- A relation can be association or generalization
- A polygon is composed of an ordered set of points
- A programmer uses a computer language on a project

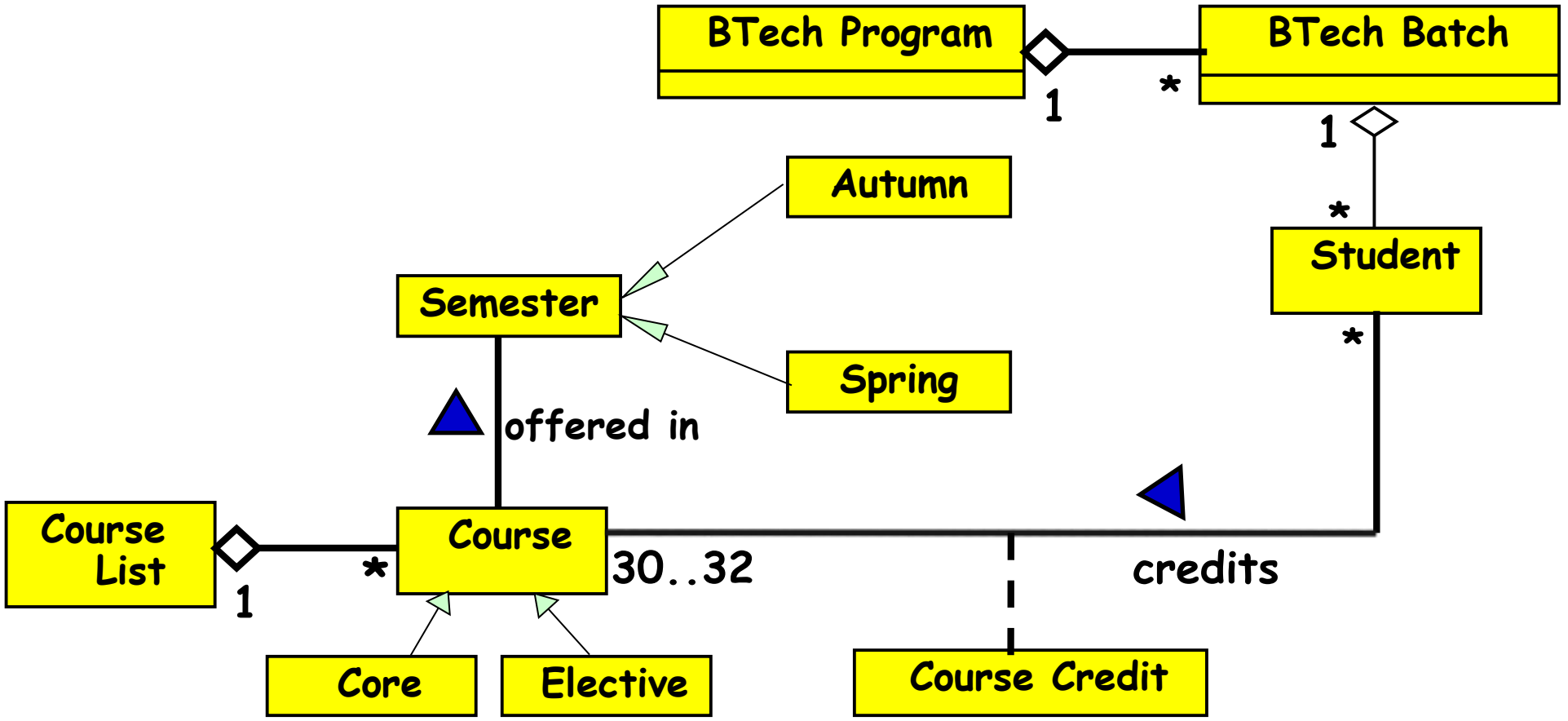

ASOK, I NEED AN INTERN TO TEST-PILOT OUR NEW MOON SHUTTLE PROTOTYPE.
WOULDN'T IT BE WISER TO SEND A MONKEY ON THE FIRST FLIGHT?
YOU'RE THINKING OF THE SECOND FLIGHT.
scottadams@aol.com
www.dilbert.com
12-5-07 © 2007 Scott Adams, Inc./Dist. by UFS, Inc.

- **The B.Tech program of IITKgp Computer Science Department:**
  - **comprises of many B.Tech batches.**

Exercise

- Each B.Tech batch consists of many B.Tech students.
- CSE Department has many listed courses.
  - A course may be offered in either spring or Autumn semesters
  - A course is either listed as an elective course or a core course.
  - Each B.Tech students need to credit between 30 to 32 course offerings.
  - A student might repeat a course if he/she desires

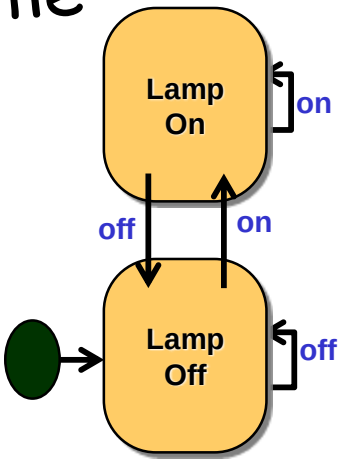
Model Solution



State Machine Diagrams

Stateless vs. Stateful Objects

- **State-independent (modeless):**
  - Type of objects that always respond the same way to an event.



- **State-dependent (modal):**
  - Type of objects that react differently to events depending on its state or mode.

Use state machine diagrams for modeling objects with complex state-dependent behavior.

Stateful Classes

- Give examples of some classes that have non-trivial state models:
  - **Lift controller:** Up, down, standstill,…
  - **Game software controller:** Novice, Moderate, Advanced…
  - **Gui:** Active, Inactive, clicked once, …
  - **Robot controller:** Obstacle, clear, difficult terrain…

- **Controller classes are an important class of statefull examples:**
  - A controller may change its mode depending on sensor inputs and user inputs.

Stateful Classes

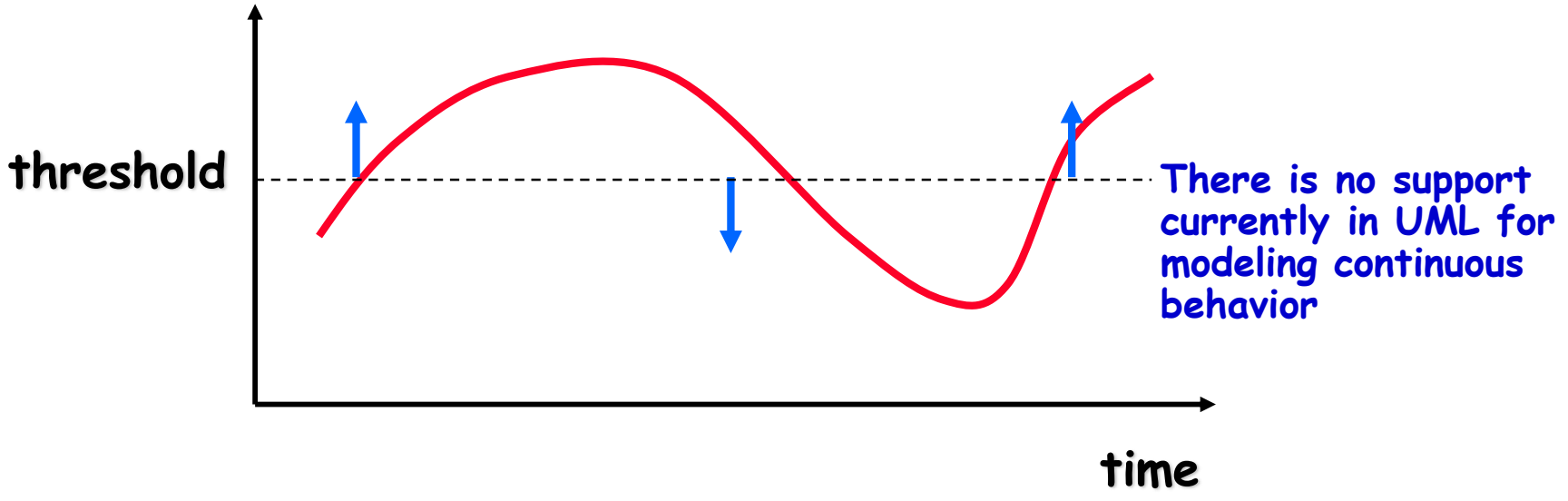
- In a client-server system:
  - **Servers are stateless, clients are stateful.**
- Common stateful objects:
  - **Controllers:**
    - A game controller may put the game in expert, novice or intermediate modes.
  - **Devices:**
    - A Modem object could be dialing, sending, receiving, etc.
  - **Mutators** (objects that change state or role)
    - A RentalVideo is rented, inStore, or overDue

Event-Based Programming

- Traditional programs have single flow of control:
  - Represented using flowchart or activity diagram
- Event-driven systems :
  - **In contrast, depending on an event occurrence, corresponding handler is activated**
  - Programming these using traditional approach is not suitable, and would at the least cause wasteful computations.
  - **Represented using state machines.**

What Kind of Behavior?

- In general, state machines are suitable:
  - **For describing event-driven, discrete behavior**
  - Inappropriate for modeling continuous behavior



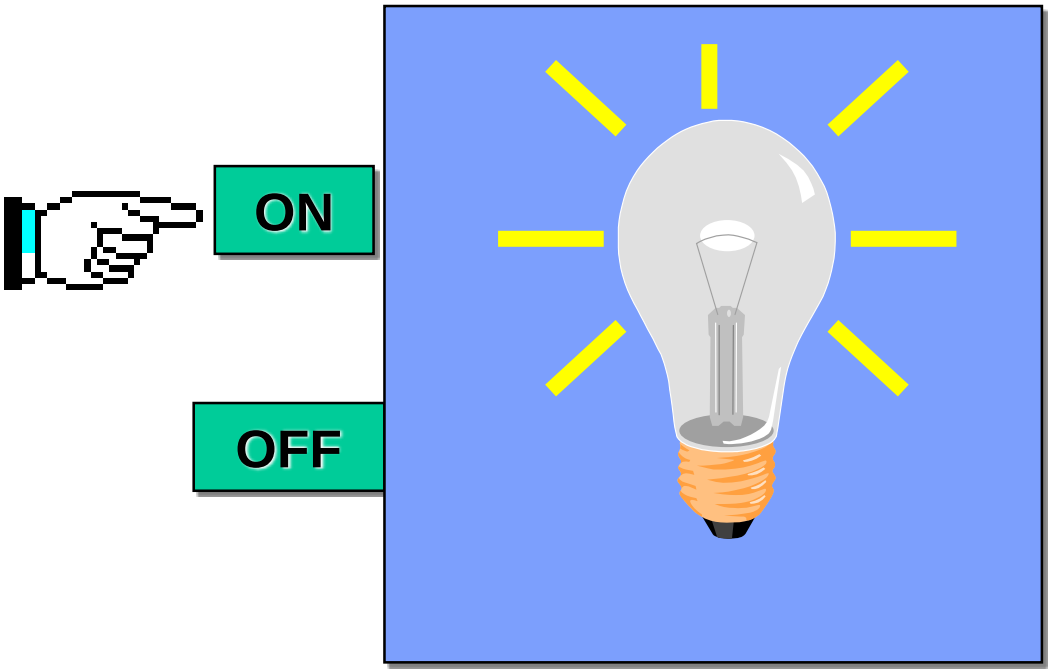
Why Create A State Model?

- Tackle complexity
- Document:
  - For review, explaining to others, etc.
- **Generate code automatically**
- Generate test cases

Finite State Automaton

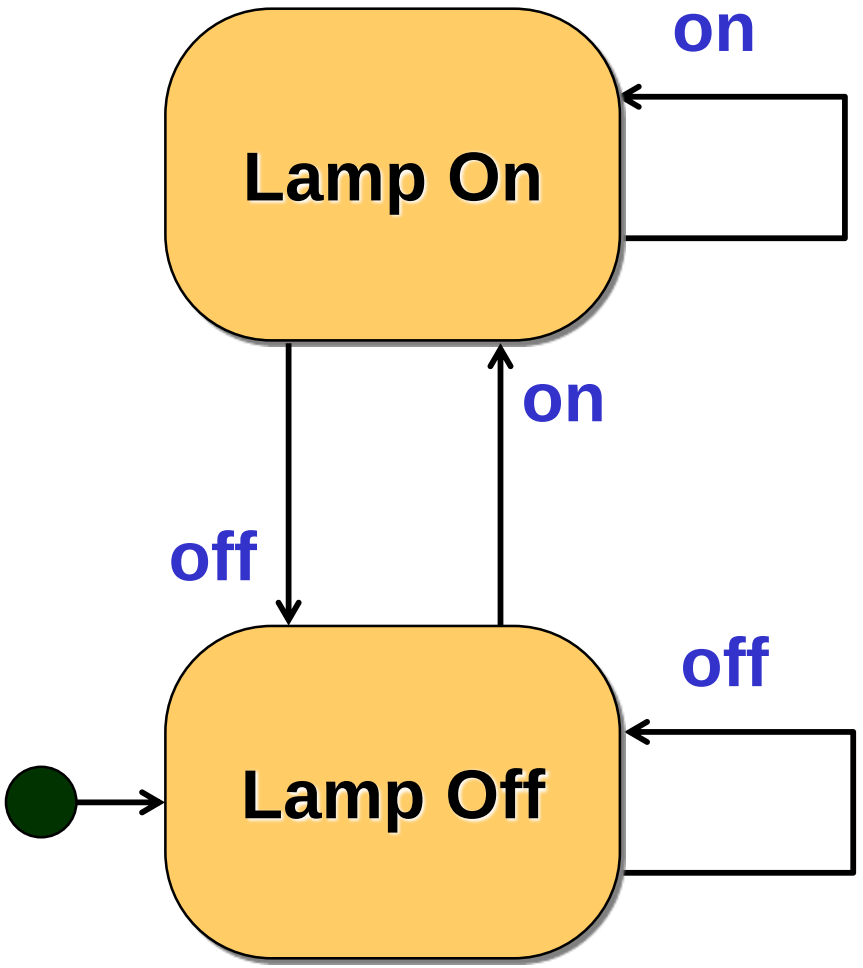
- A machine whose output behavior is not only a direct consequence of the current input,
  - But past history of its inputs

- **Characterized by an internal state which captures its past history.**



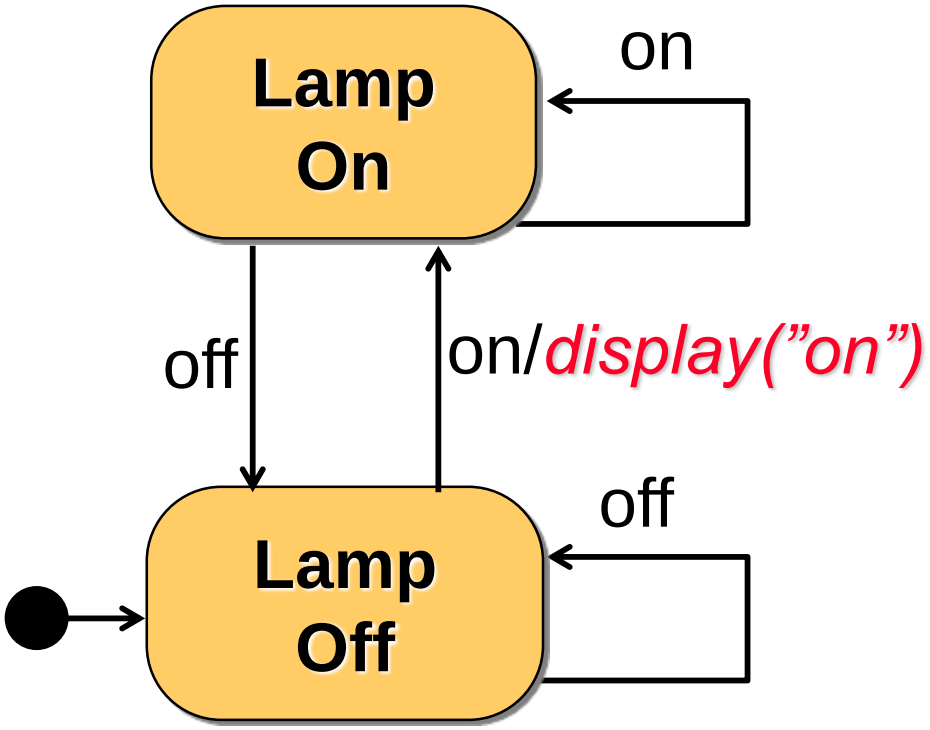
Basic State Machine Diagram

- Graphical representation of automata behavior…

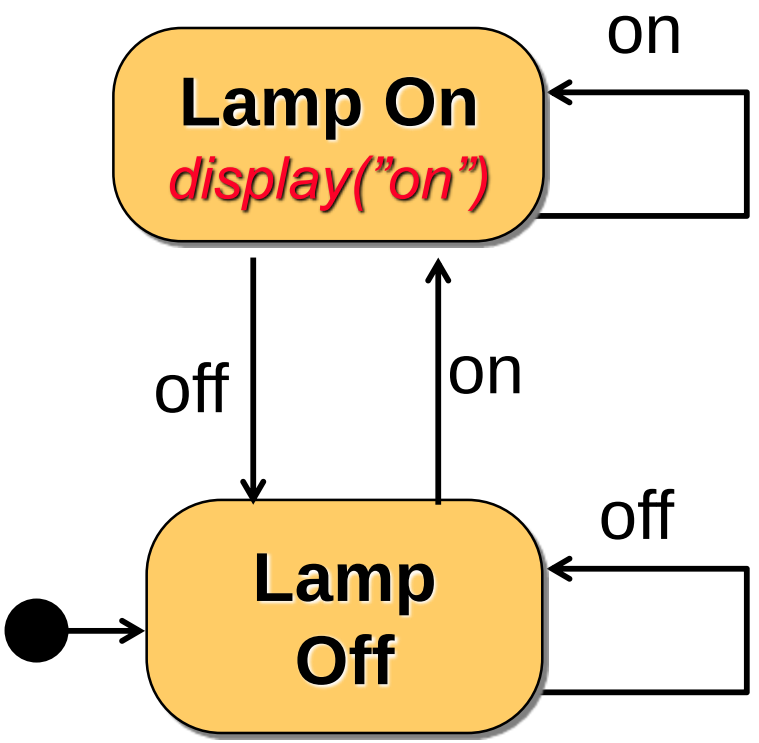


Outputs and Actions

- Automaton generates outputs during transition.
  - Alternate representations.



Mealy automaton

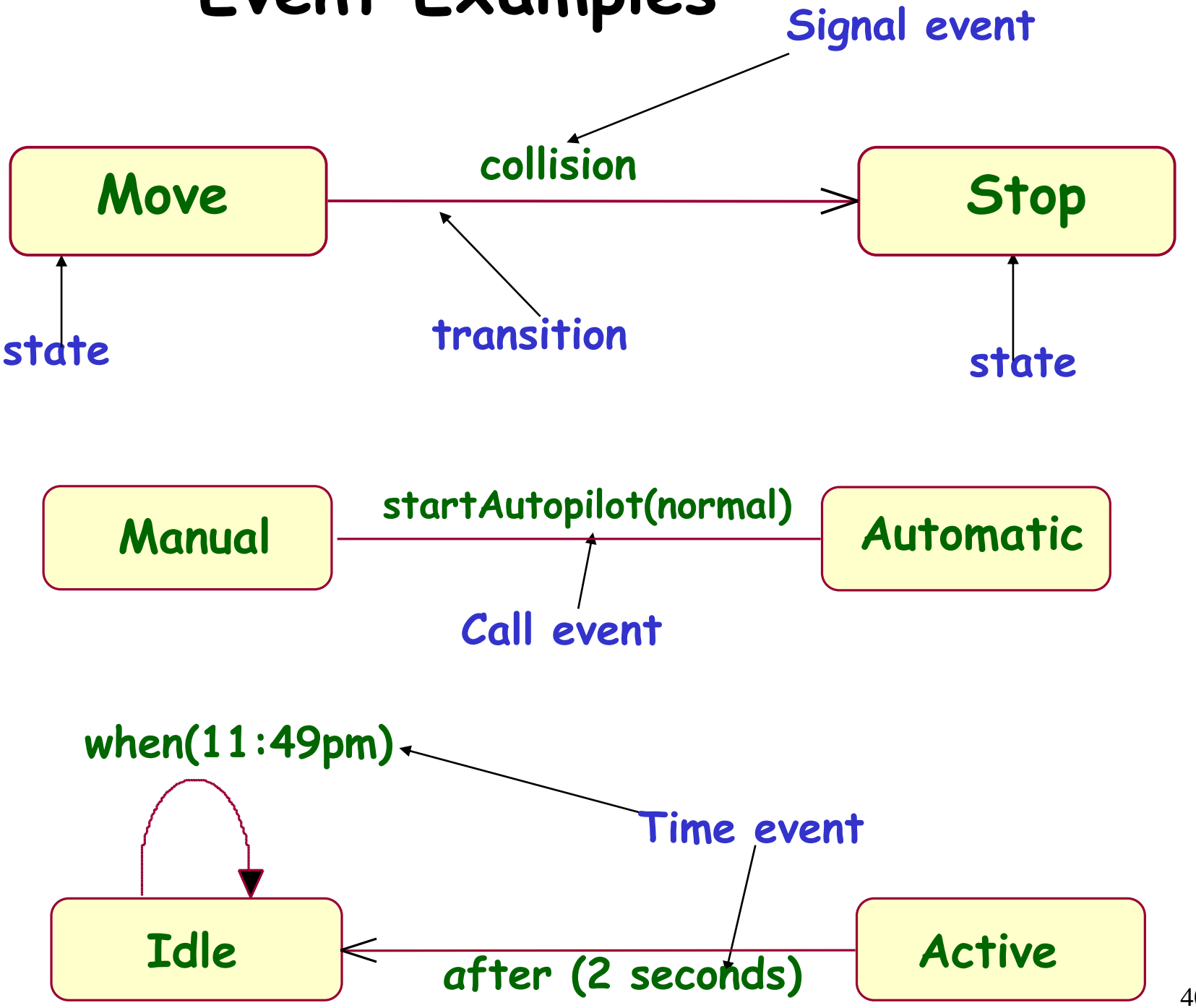


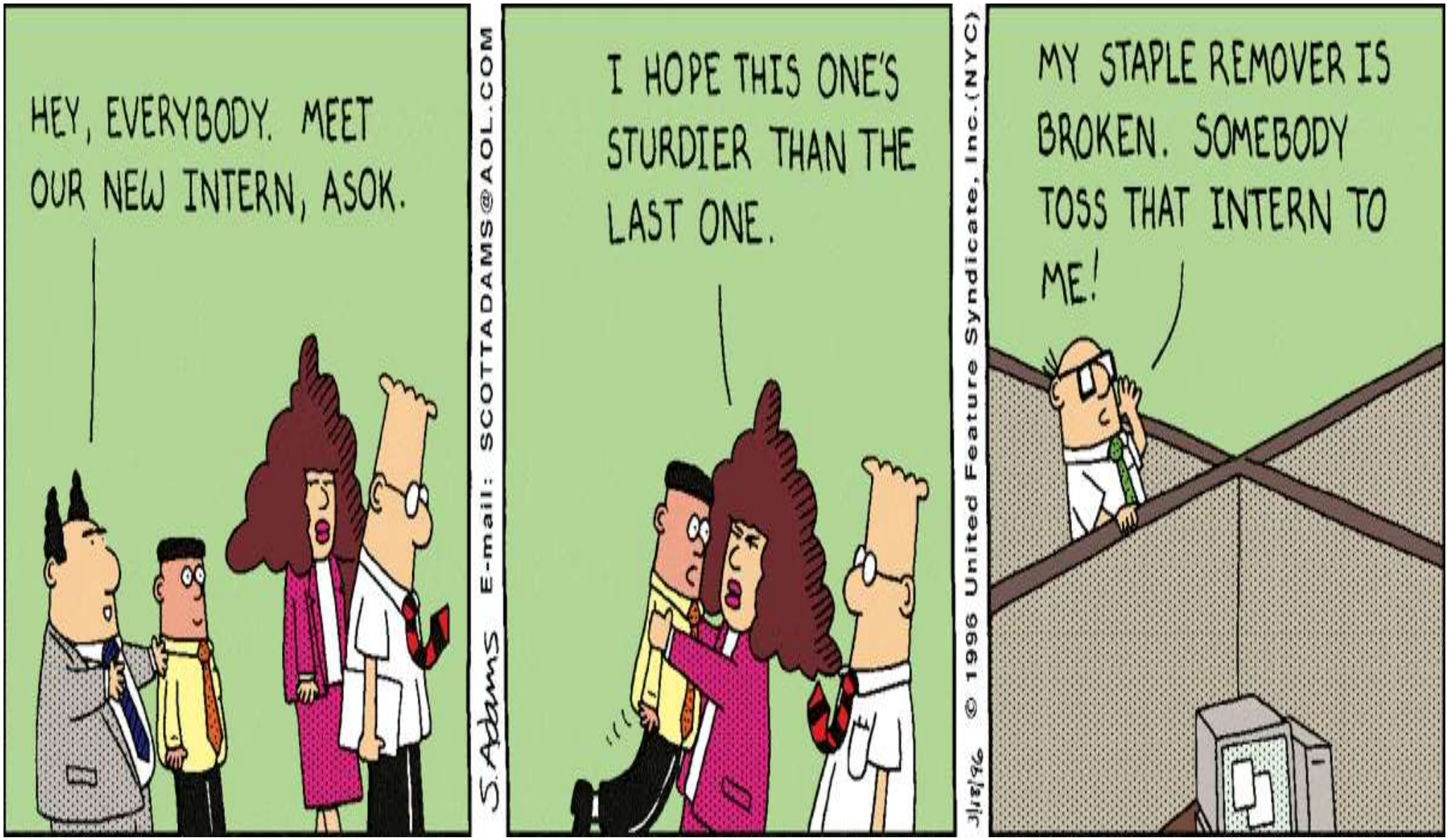
Moore automaton

Event-Driven Behavior

- **Event types:**
  - **Object interactions:**
    - synchronous object operation invocation (call event)
    - asynchronous signal reception (signal event)
  - **Occurrence of time instants (time event)**
    - interval expiry
    - calendar/clock time
  - **Change in value of some entity (change event)**
- Event Instance = an instance of an event (type)
  - occurs at a particular time instant and has no duration

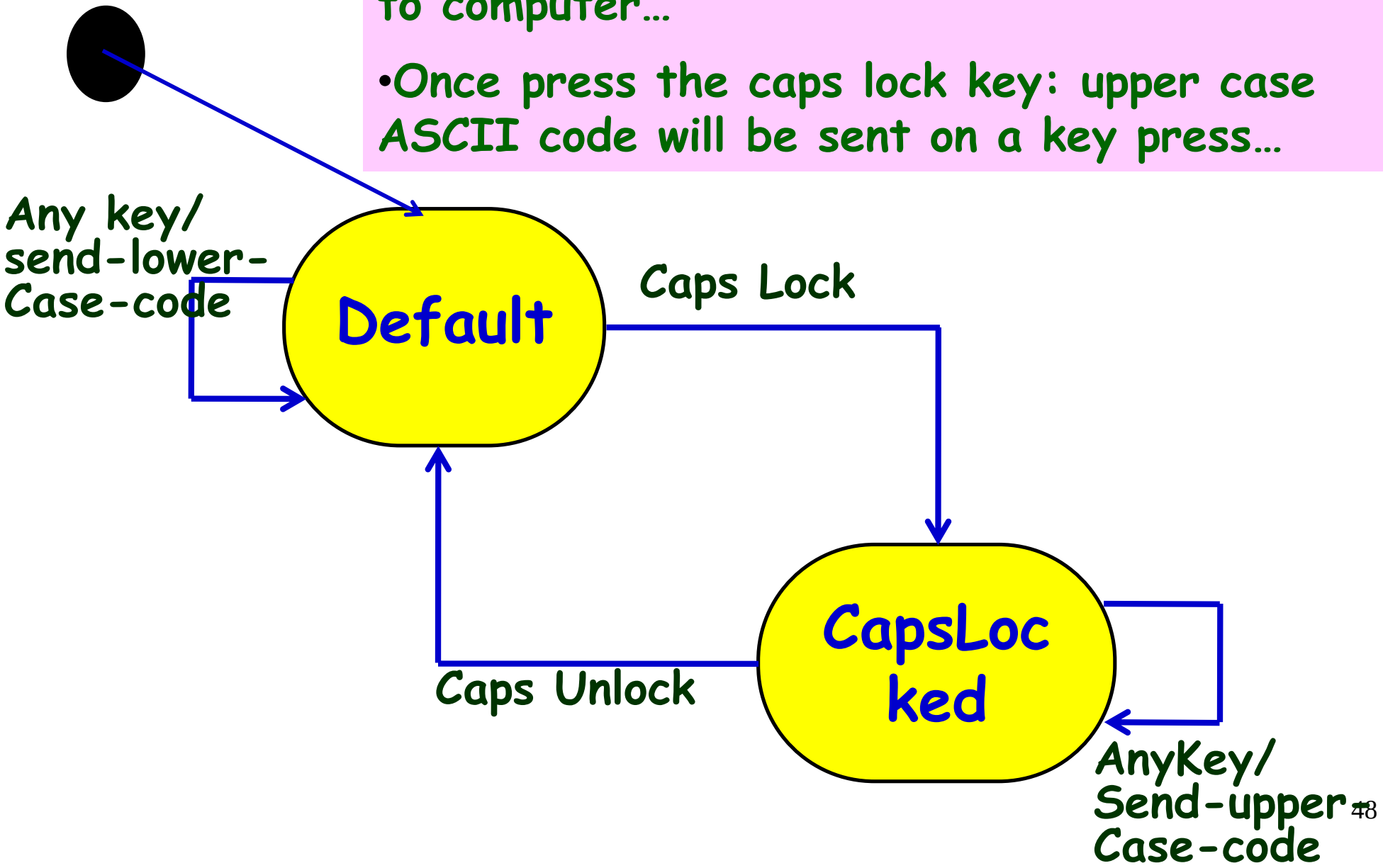
Event Examples




HEY, EVERYBODY. MEET OUR NEW INTERN, ASOK.
S. Adams
E-mail: SCOTTADAMS@AOL.COM
I HOPE THIS ONE'S STURDIER THAN THE LAST ONE.
3/18/96
© 1996 United Feature Syndicate, Inc. (NYC)
MY STAPLE REMOVER IS BROKEN. SOMEBODY TOSS THAT INTERN TO ME!

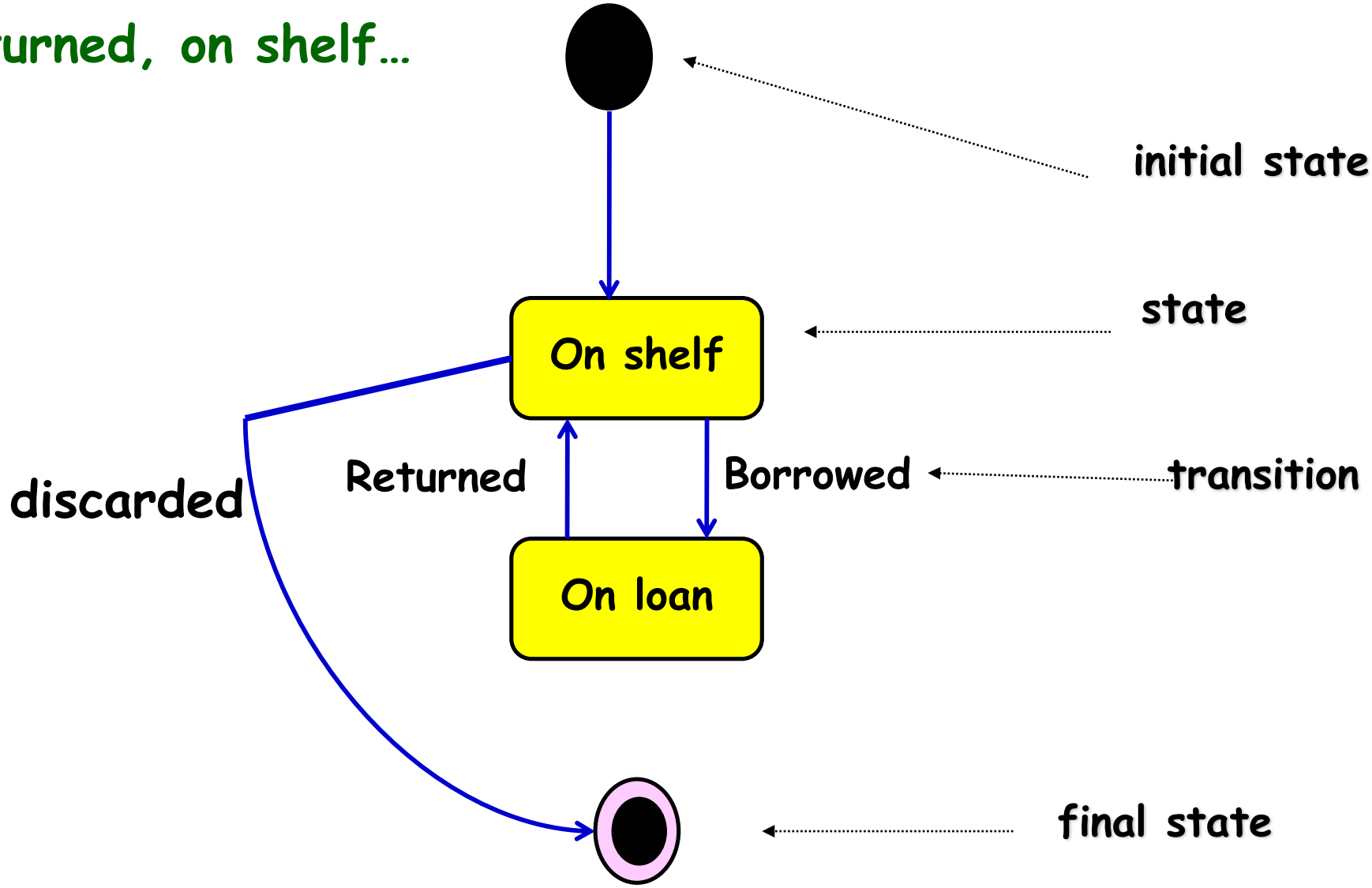
Exercise 0: Draw State Machine Diagram of a Keyboard?

- Press any key: lower case ASCII code is sent to computer…
- Once press the caps lock key: upper case ASCII code will be sent on a key press…



Exercise 1: State Machine Diagram of a Library Book

- A library book to start with, is present in a shelf…
- When borrowed out, it is not on shelf…
- Returned, on shelf…



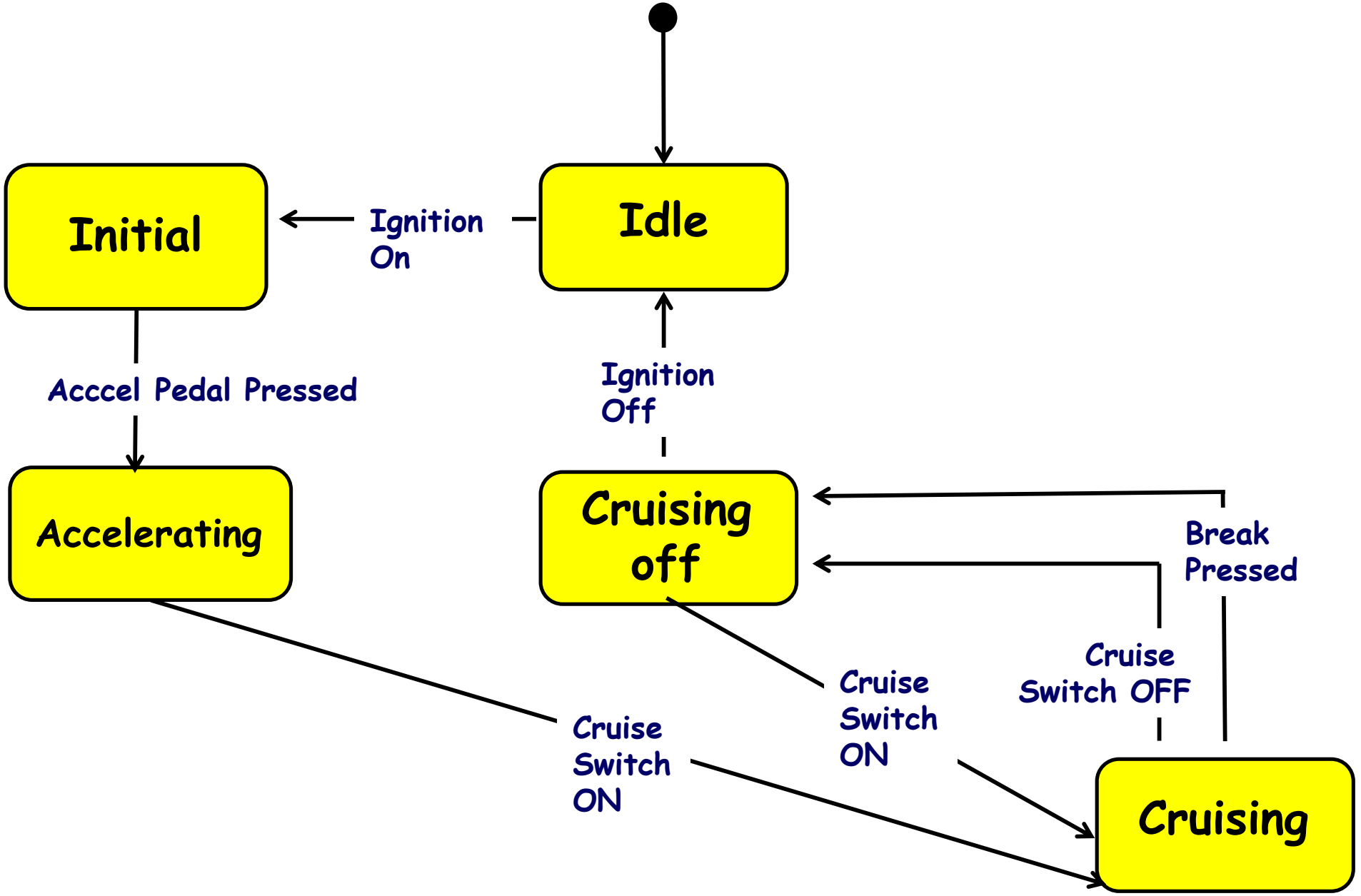
Exercise 2: Construct State Model

- A car is in idle mode when ignition is off:
  - Changes to initial mode when ignition is keyed ON.
- The car accelerates when the acceleration pedal is pressed.

Cruise Controller

- While accelerating, the car goes into a cruise mode, as soon as cruise switch is set to ON.
- Cruise mode is turned off either when brake is applied or the cruise switch is turned off
  - Cruise mode can be resumed by setting cruise switch to ON.
  - When ignition is turned off the car goes to idle mode.

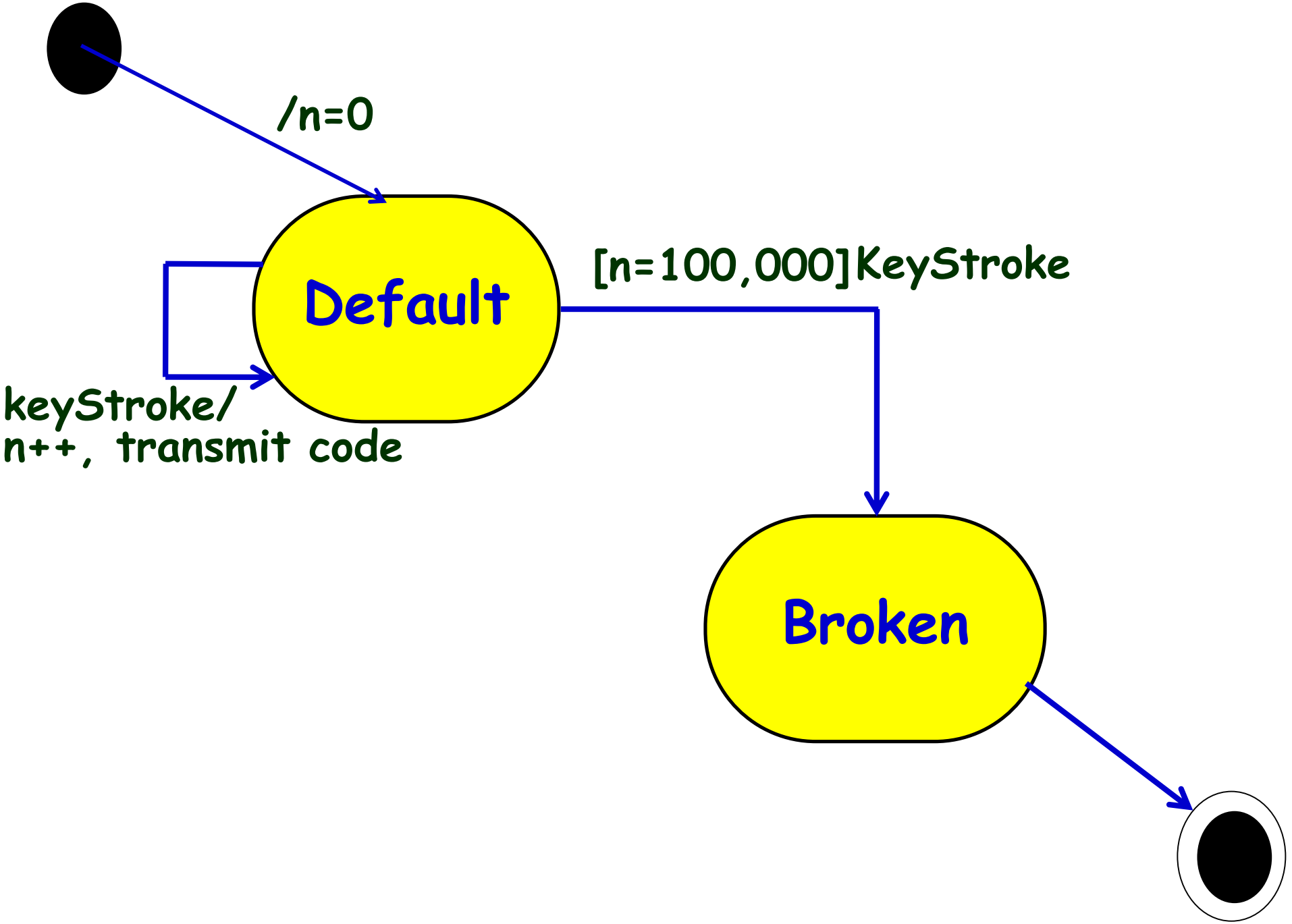
Cruise Controller



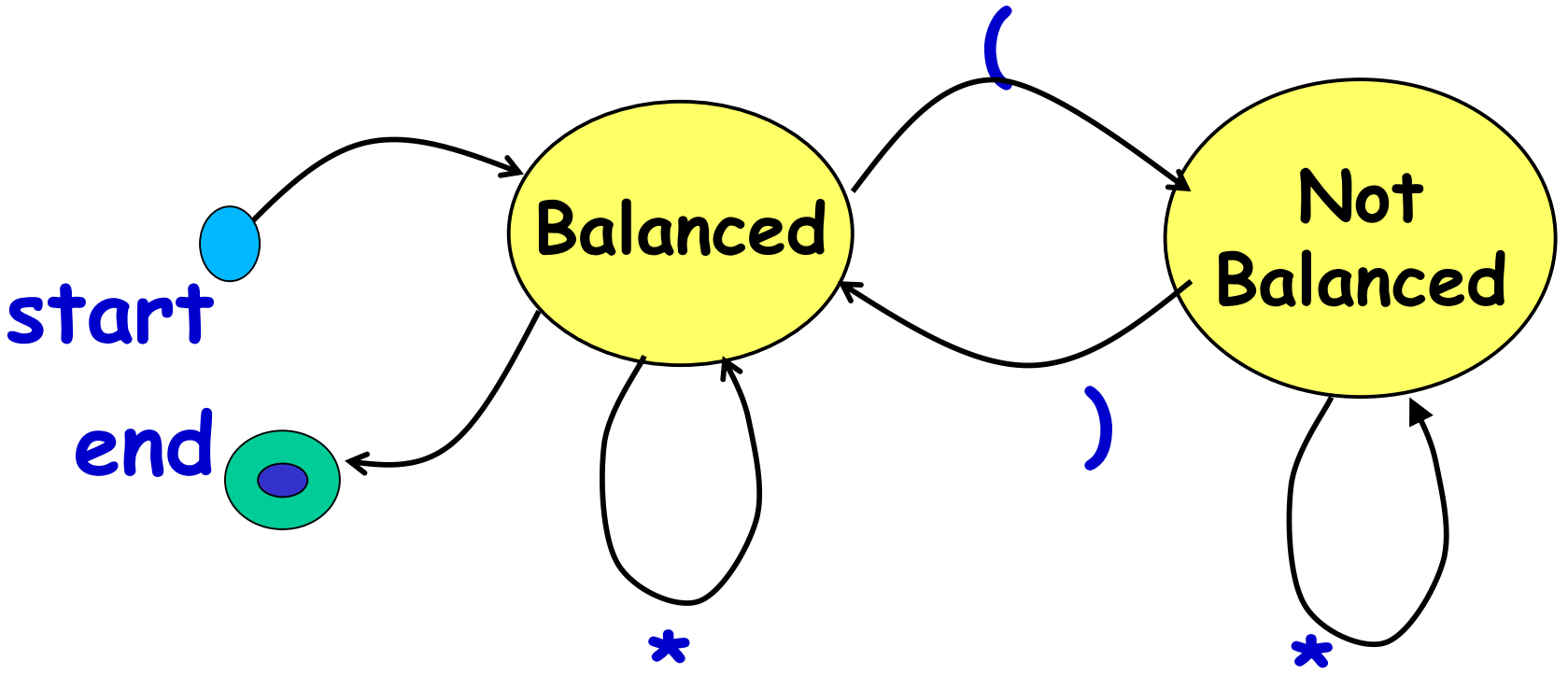
Exercise 3

- Model a keyboard using UML state machine diagram:
  - Transmits key code on each key stroke.
  - Breaks down after entering 100,000 key strokes.

Solution

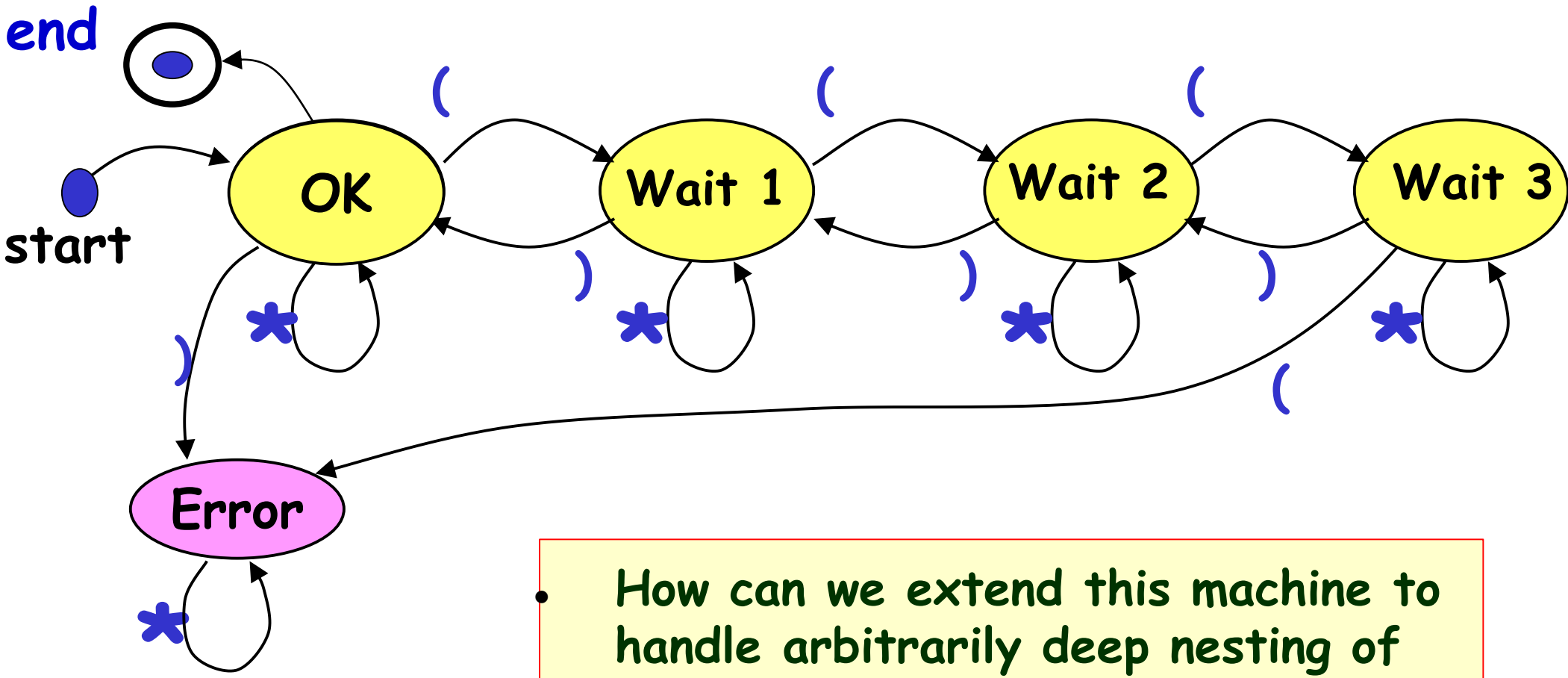

/n=0
Default
[n=100,000]KeyStroke
keyStroke/
n++, transmit code
Broken

Exercise 4: Draw State Machine: GUI Accepts only Balanced Parentheses



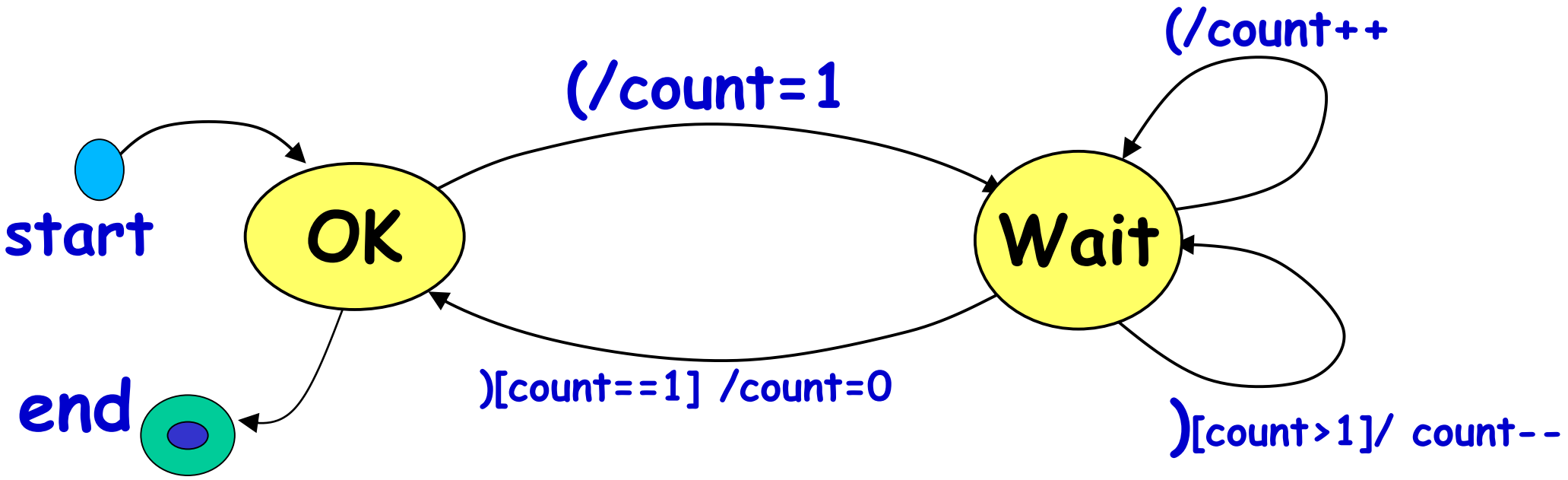
- Inputs are any characters
- No nesting of parentheses
- No "output" other than any state change

Example 5: Draw State Machine: GUI Accepts only upto 3 Nested parentheses



- How can we extend this machine to handle arbitrarily deep nesting of parentheses?

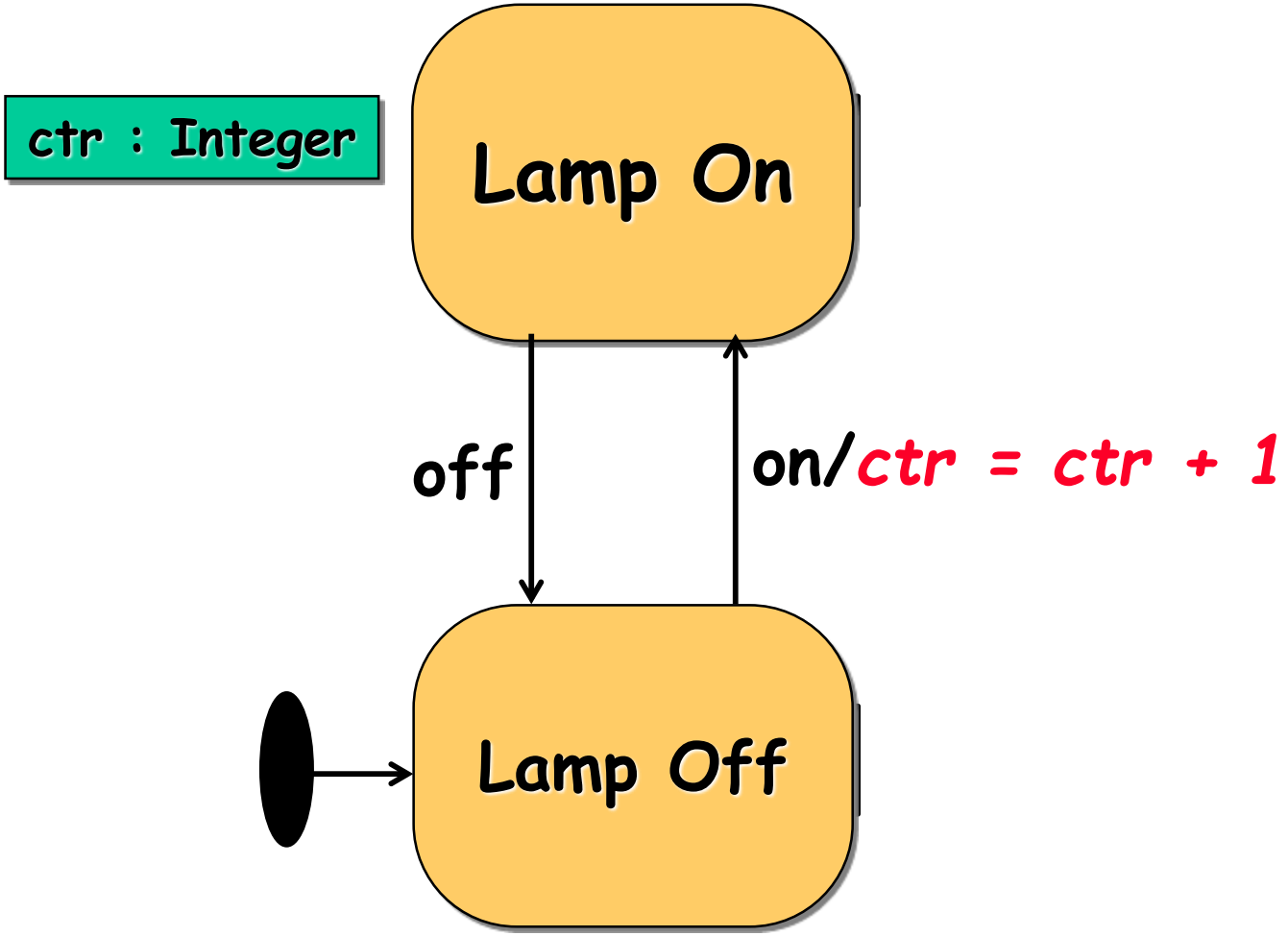
How to Model Nested parentheses?



- *A state machine, but not just a state machine --- an EFSM*

Extended State Machines

- Addition of variables ("extended")



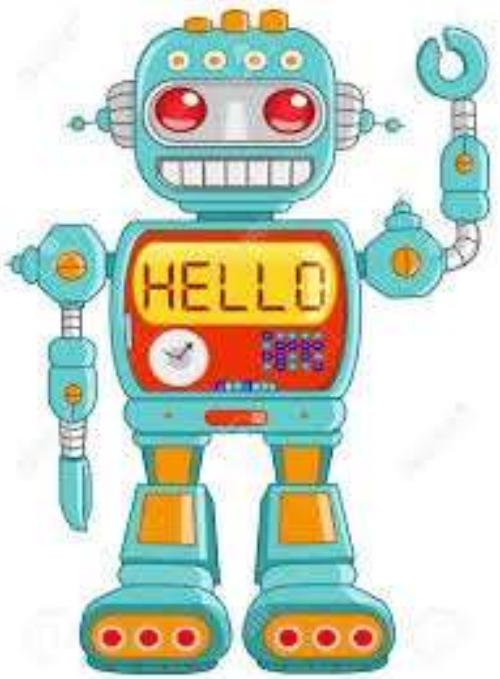
UML State Machine Model

State Chart Diagram

- FSMs suffer from a few severe shortcomings:
  - **What are the shortcomings of FSM?**

- State chart is based on the work of **David Harel** [1990]:
  - Overcomes important shortcomings of FSM
  - Extends FSM in 2 major ways: **Concurrent states** and **hierarchy**.

Robot: State Variables

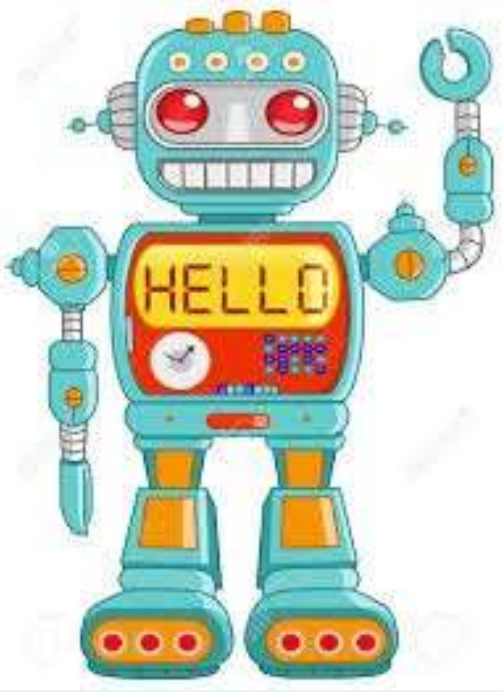


- Power: On, OFF
- Movement: Walk, Run
- Direction: Forward, Backward, left, Right
- Left hand: Raised, Down
- Right hand: Raised, down
- Head: Straight, turned left, turned right
- Headlight: On, Off
- Turn: Left, Right, Straight

How many states in the state machine model?

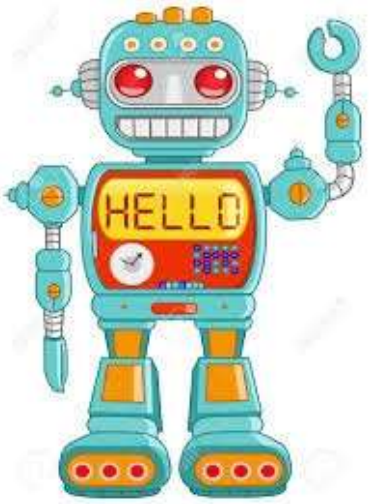
FSM: exponential rise in number of states with state variables

Event	State
turnOn	Activated
turnOff	Deactivated (Idle)
stop	Stopped
walk	Walking
run	Running
raiseLeftArm	LeftArmRaised
lowerLeftArm	LeftArmLowered
lowerLeftArm	LeftArmLowered
raiseRightArm	RightArmRaised
lowerRightArm	RightArmLowered
turnHead	HeadTurned(direction)
speak	Talking(text)



State Chart Diagram Cont…

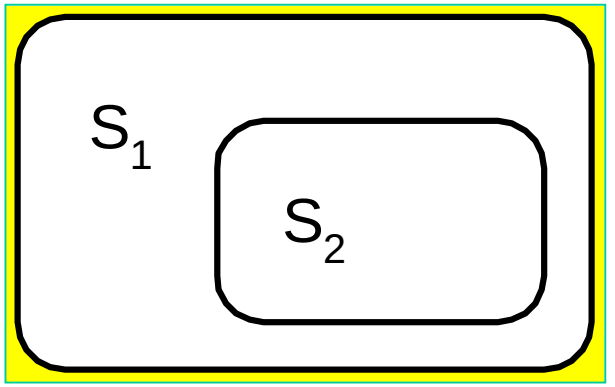
- State chart avoids two problems of FSM:
  - **State explosion**
  - **Lack of support for representing concurrent states**



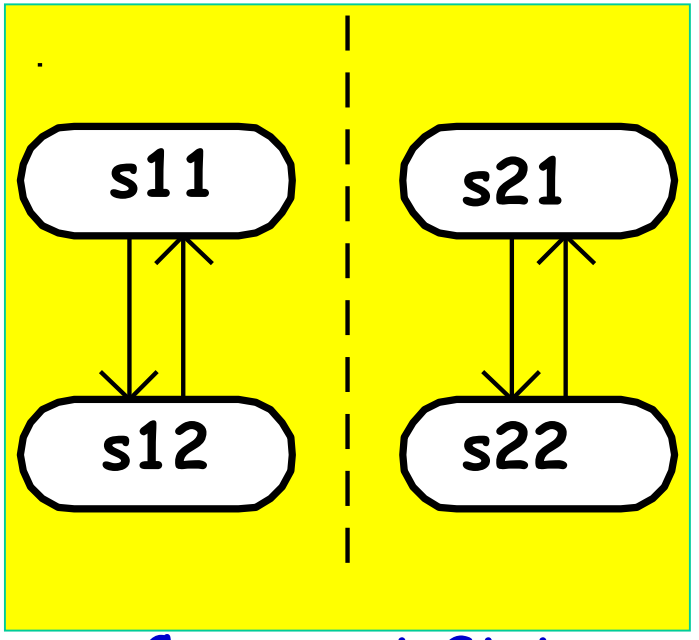
- A hierarchical state model:
  - Makes use of **composite states** --- OR and AND states.

Features of State Charts

- Two major features introduced :
  - **Nested** states
  - **Concurrent** states
- Several other features have also been added:
  - History state
  - Broadcast messages
  - Actions on state entry and exit



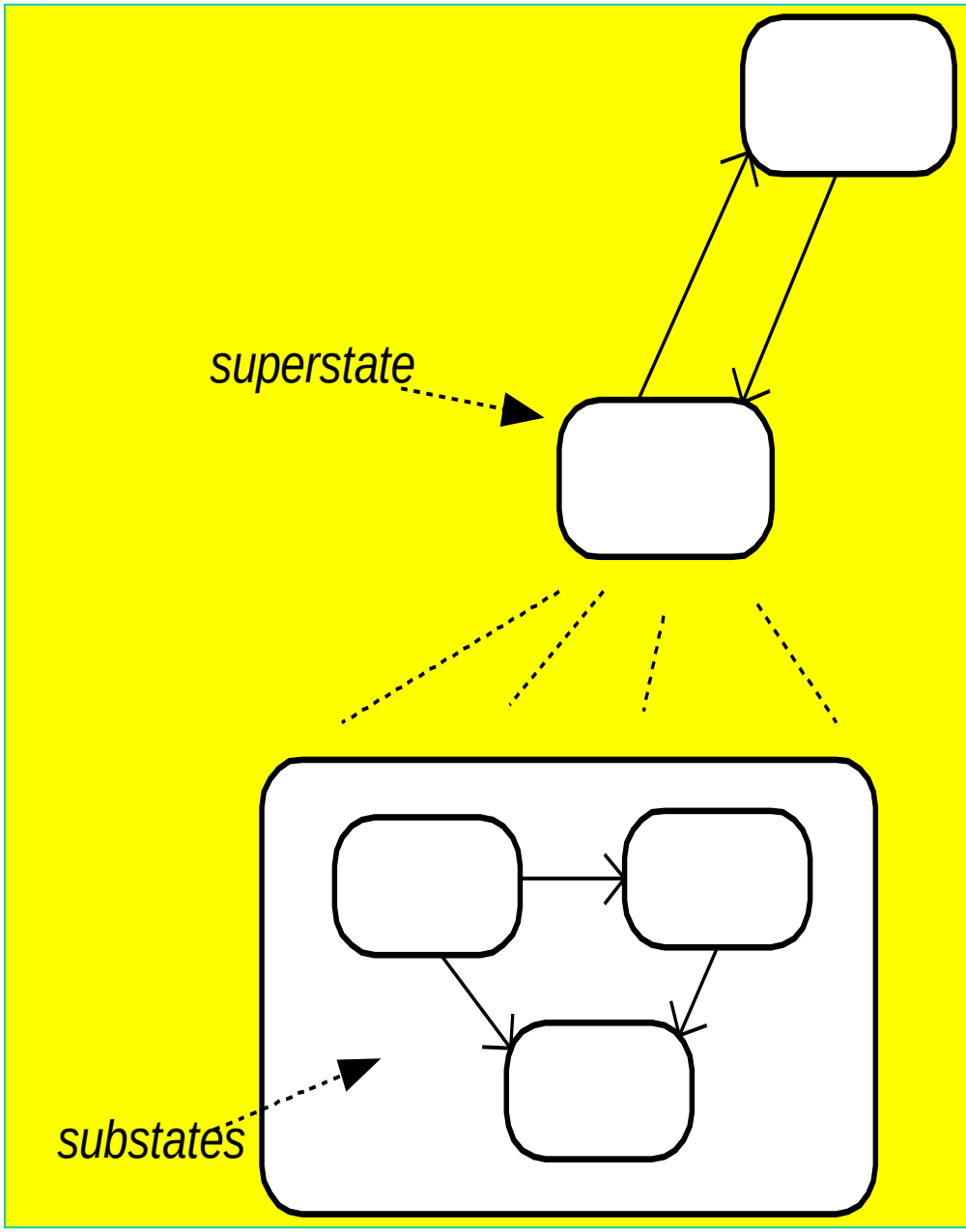
Nested State



Concurrent State

Nested State Diagrams

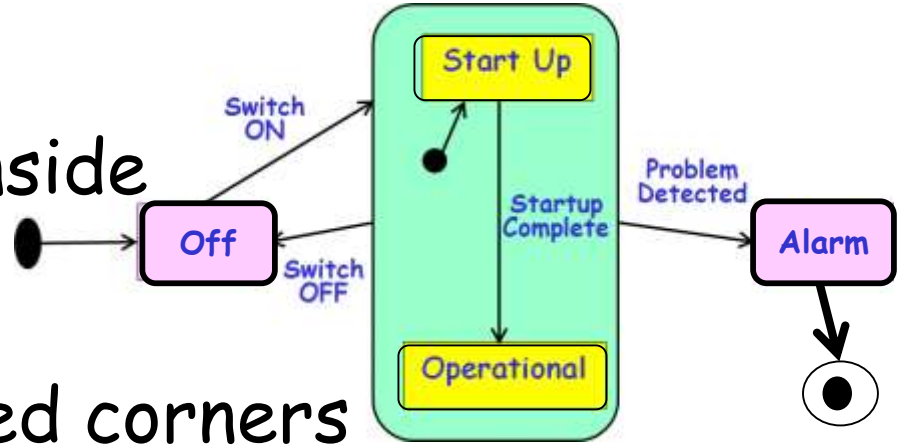
- Hierarchical organization is a classic way to control complexity:
  - of programs
  - of documentation
  - of objects
  - …
- Why not state diagrams?
  - **superstates**
  - **substates**



State Chart Diagram

Cont…

- Basic elements of state chart diagram:
  - **Initial State:** A filled circle
  - **Final State:** A filled circle inside a larger circle
  - **State:** Rectangle with rounded corners
  - **Transitions:** Arrow between states, also boolean logic condition (guard)



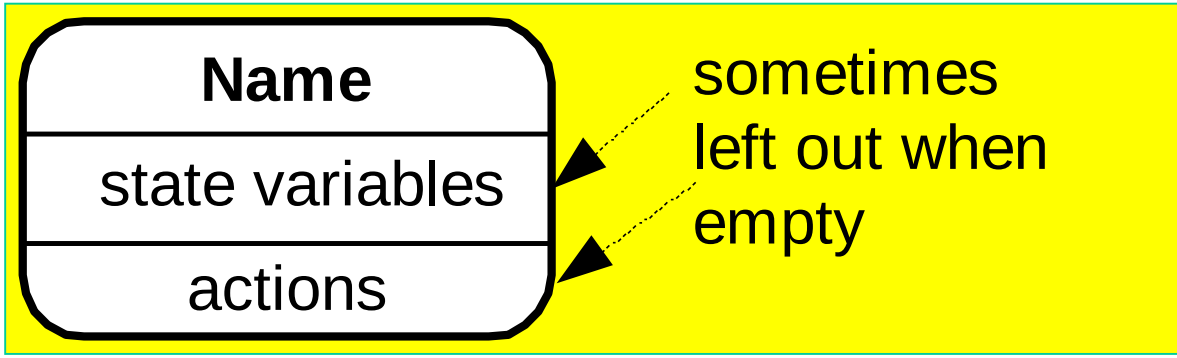
- **State chart in UML is called state machine:**
  - **As it not only models state behavior but also generates code…**

State Machine Diagram

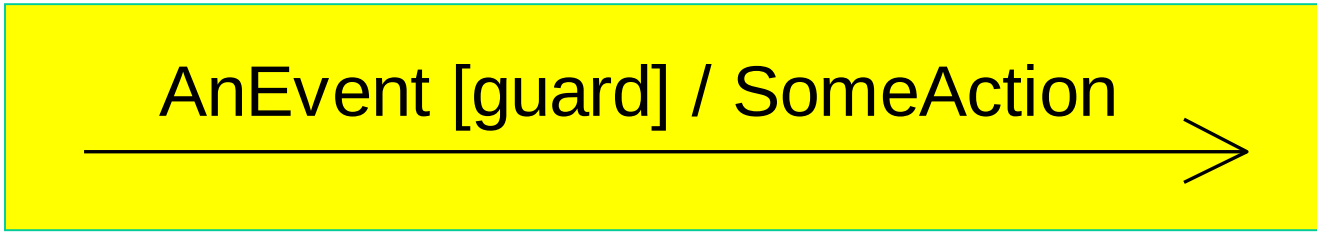
- State machine is the code that implements a model and runs on a computer.
- In contrast, a state chart is a description of a state machine,

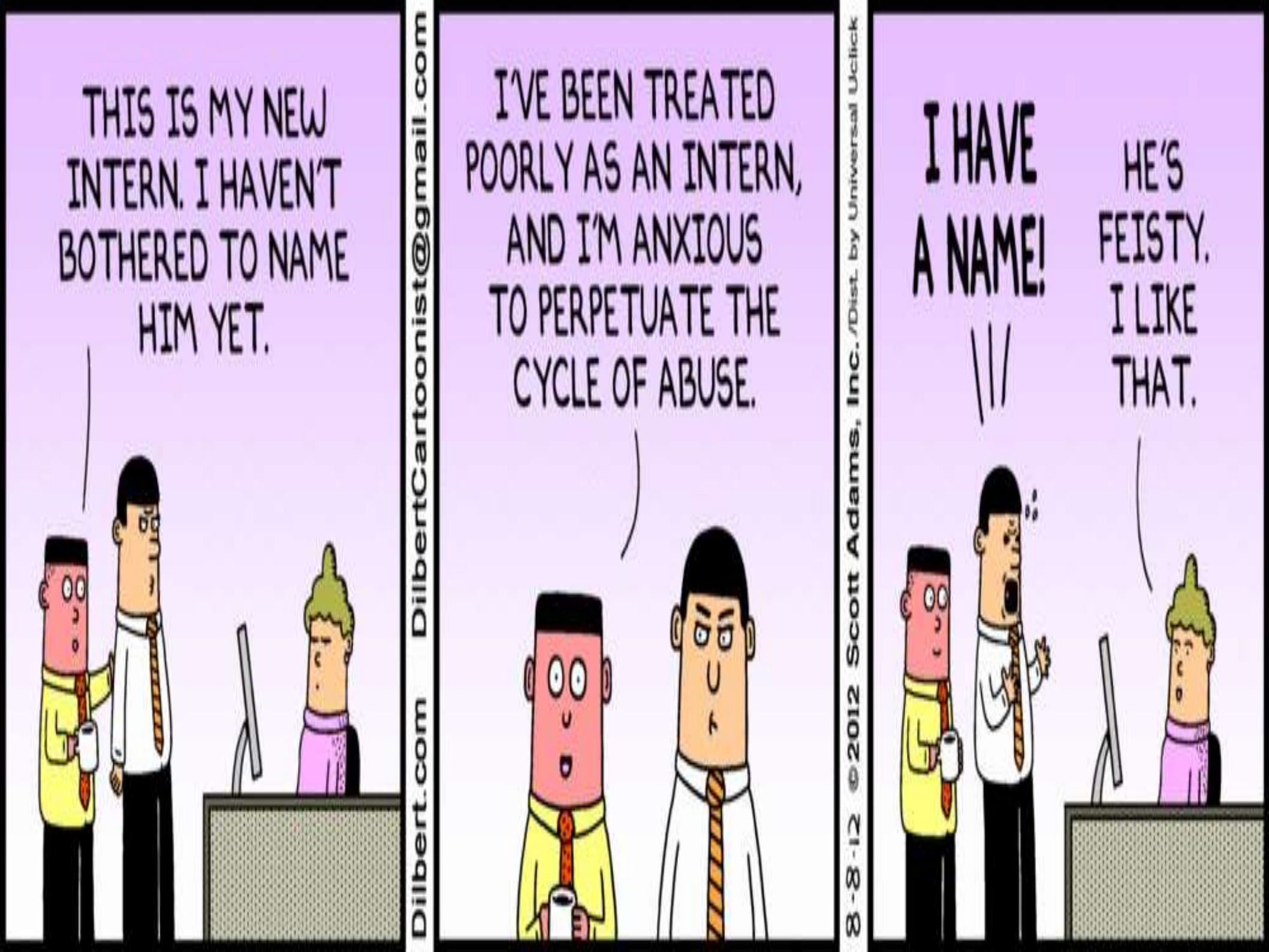
UML State Machine Diagram: Syntax

- A state is drawn with a round box, with three compartments for:
  - name
  - state variables
  - actions performed



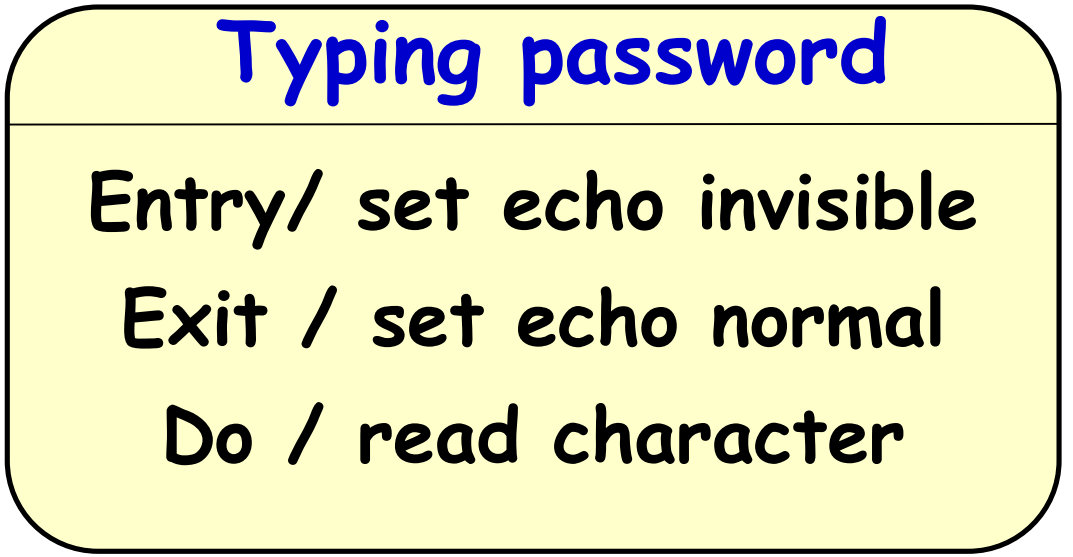
- A transition is drawn with a labeled arrow,
  - event causing the transaction
  - guard condition
  - Action to perform



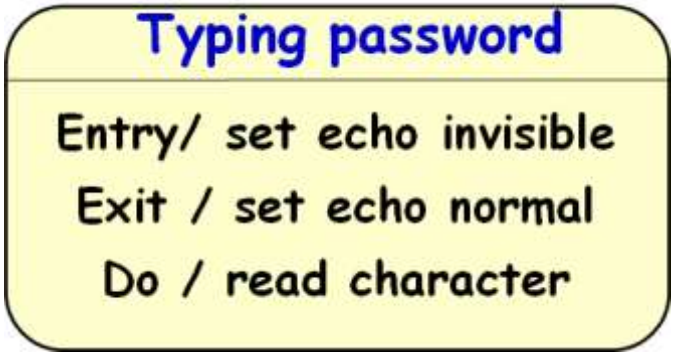

THIS IS MY NEW INTERN. I HAVEN'T BOTHERED TO NAME HIM YET.
I'VE BEEN TREATED POORLY AS AN INTERN, AND I'M ANXIOUS TO PERPETUATE THE CYCLE OF ABUSE.
I HAVE A NAME!
HE'S FEISTY. I LIKE THAT.
Dilbert.com DilbertCartoonist@gmail.com
8-8-12 ©2012 Scott Adams, Inc./Dist. by Universal Uclick

Syntax of UML State machine

- **State:** Rectangle with rounded corners
- Name tab
- Action label:
  - Entry
  - Exit
  - Do



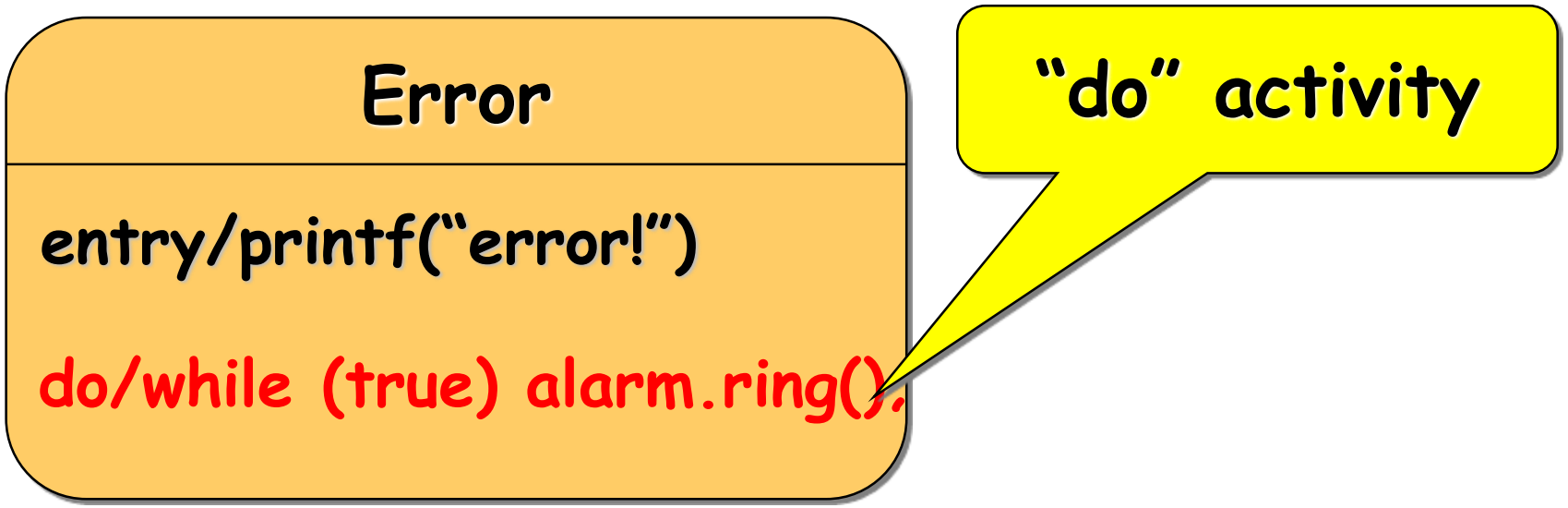
Predefined Action Labels



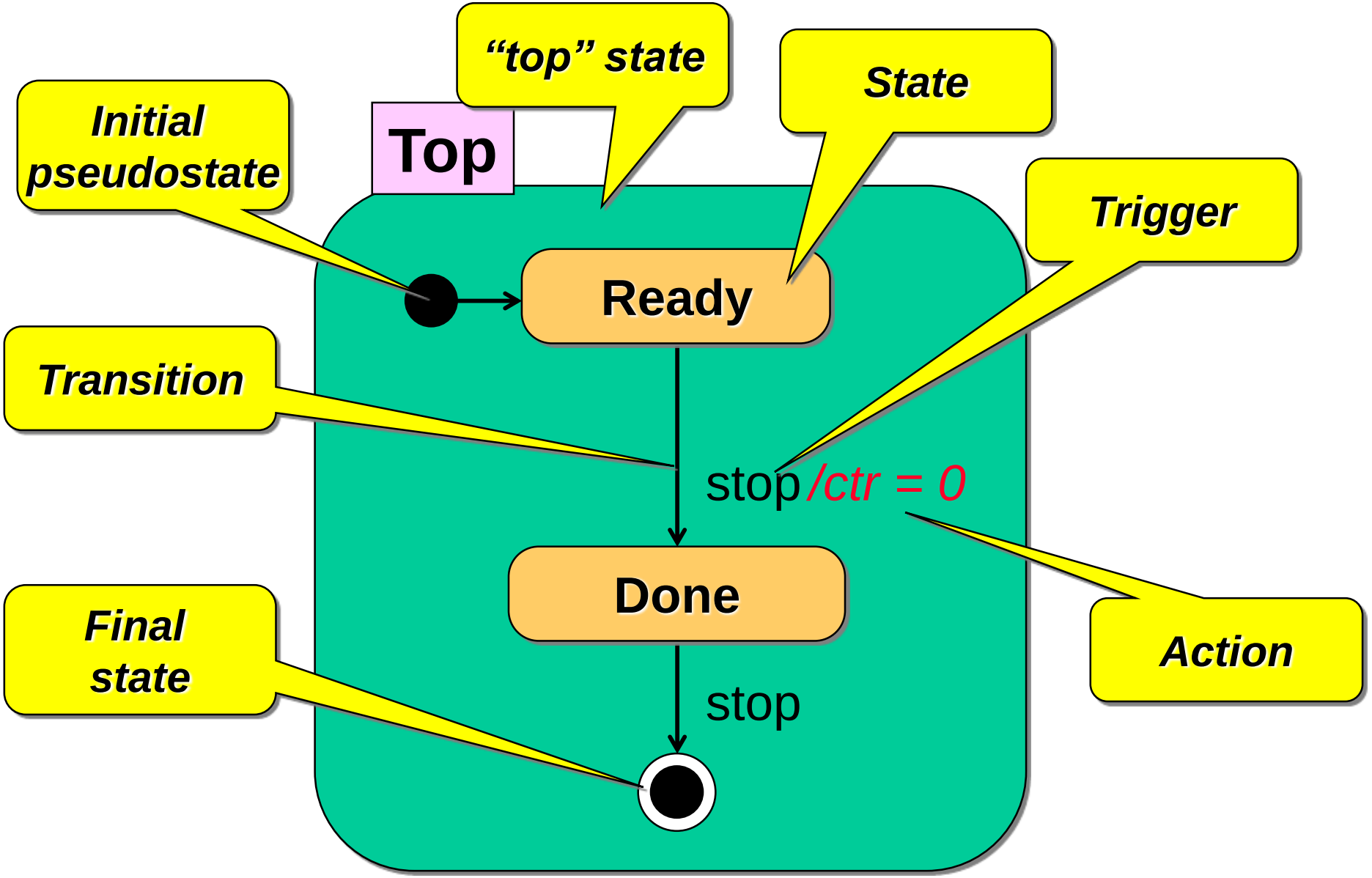
- **"entry/"**
  - Identifies an action to be performed upon entry to the state
- **"exit/"**
  - Identifies an action to be performed upon exit from the state (exit action)
- **"do/"**
  - Identifies an ongoing activity ("do activity") that is performed as long as the modeled element is in the state or until the computation specified by the action expression is completed

"Do" Activities

- The thread executes until:
  - The action completes or
  - The state is exited through an outgoing transition

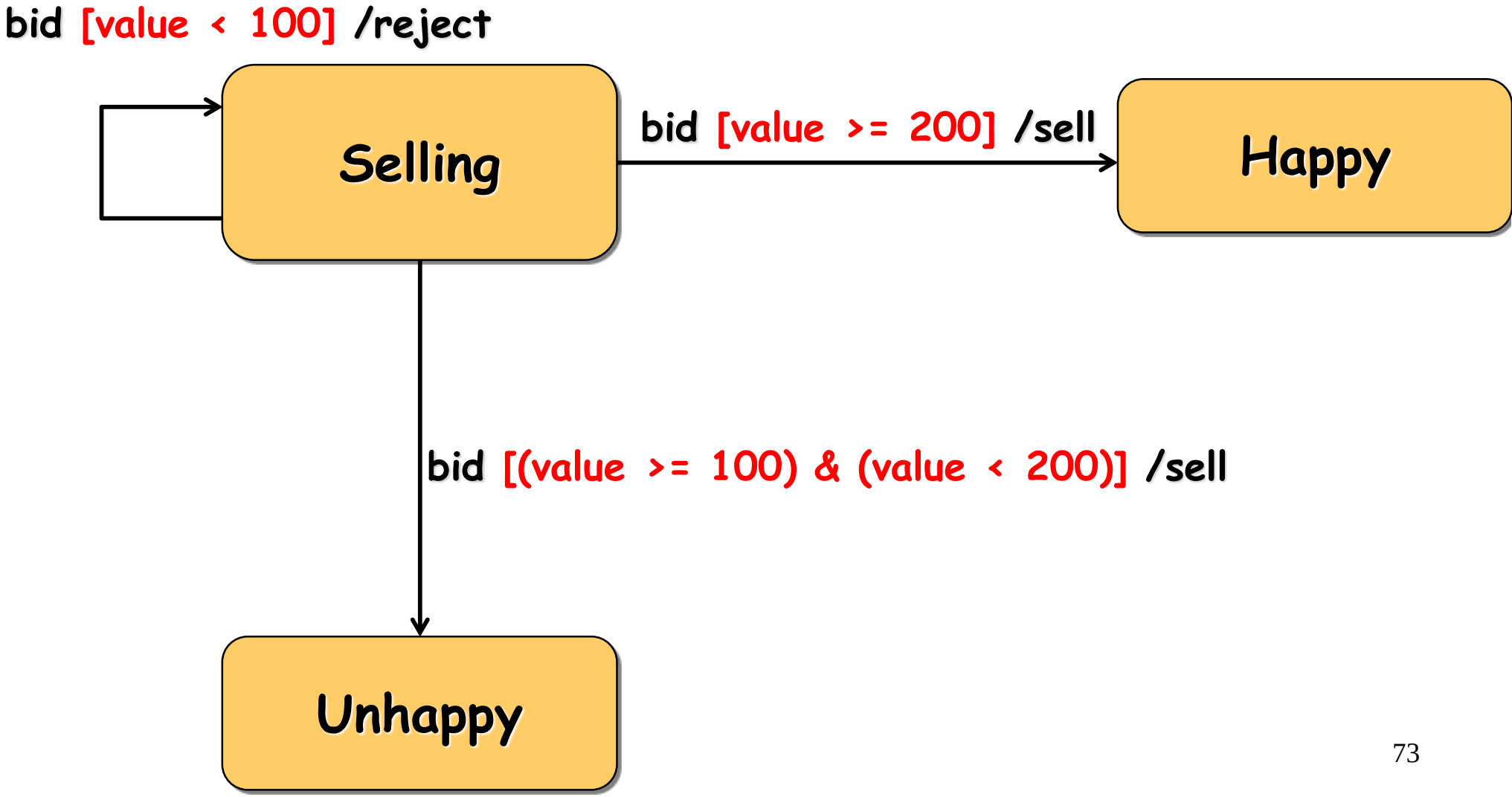


Basic UML State Model Syntax



Guards

- Boolean predicates to indicate Conditional execution of transitions



Eliminating Duplicated Transitions

- Duplicate transitions usually exist when some transition can happen from every state:
  - "error"
  - "quit"
  - "abort"
- These duplicates can be combined into a single transition:
  - **A transition from a superstate holds for all of its substates!**