STATECHARTS

CS3213 FSE

Prof. Abhik Roychoudhury

National University of Singapore

Ack: Some figures taken from Statecharts paper by David Harel

WHAT WE DID EARLIER

UML as modeling notation

- System Requirements: Use-cases, Scenarios, Sequence Diagrams
- System structure: Class diagrams
- Discussion on semantics
- System behavior: State diagrams (detailed discussion today!!)

ORGANIZATION

Finite state machines
- Other variants

trigger, action, wait for next trigger from environment

continuous interaction with environment.

Reactive and transformational systems

Statecharts
- Depth (OR-states)
- Orthogonality (AND-states)
- Broadcast communication

READINGS

Statecharts: A visual formalism for complex systems, by David Harel, Science of Computer Programming, 1987

Executable object modeling with statecharts, by David Harel and Eran Gery, IEEE Computer, 1997

Basic understanding of states/transitions is introduced first.

A PUZZLE

A man with a goat, a wolf and a cabbage wants to cross a river.

A boat can carry only 2 of the 4 entities.

Wolf wants to eat the goat.

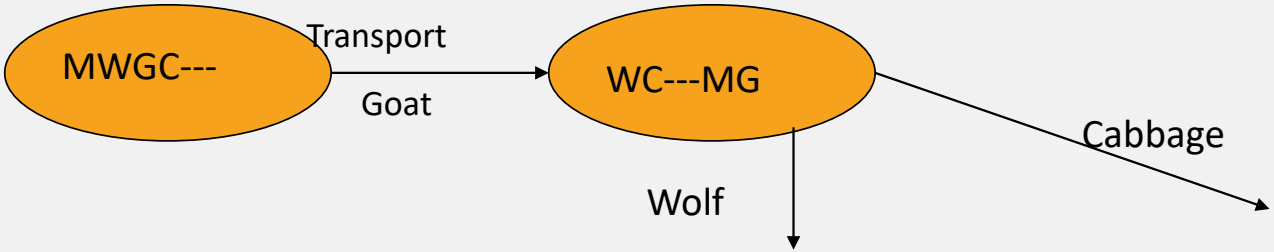
Goat wants to eat the cabbage.

- How to transport all the 4 entities ?

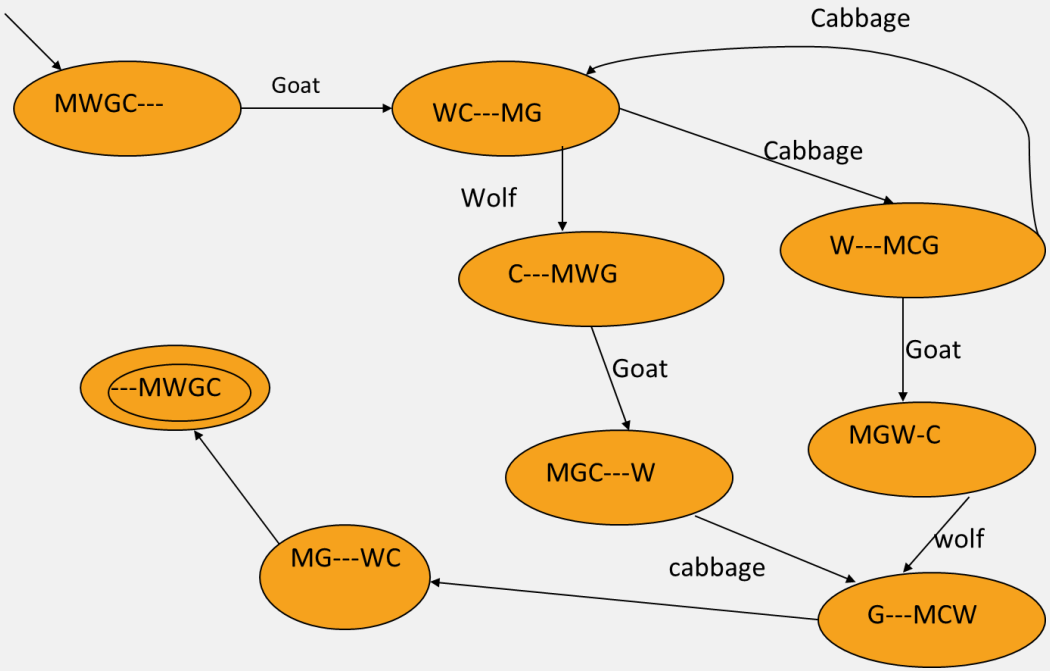
Model the local state of each entity – on which side of the river?

- A global state is a composition of these local states --- transitions of global states form FSM

STATE CHANGE



FINITE STATE MACHINE



MODELING USING FSMS

A solution to our problem is a path from the initial state to a state where all 4 entities are on other side of river.

- Notion of "termination" of the problem.
- Shown as accepting states of FSMs

Minor note:

- Not all cycles in the FSM for this problem have been shown.

FSM - DEFINITION

set of actions

$M = (S, S0, \sum, \rightarrow, F)$

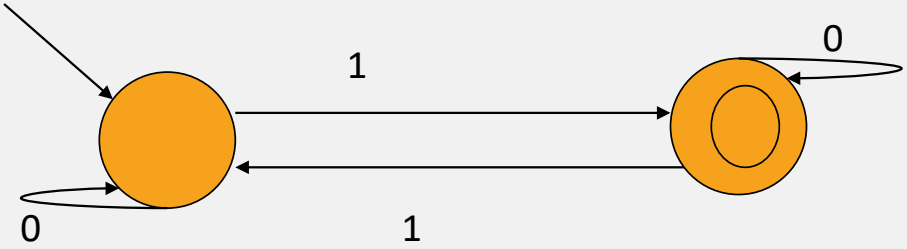
- S is a set of states
- $S0 \subseteq S$ is the set of initial states
- $\rightarrow \subseteq S \times \sum \times S$ is the transition relation
- $F \subseteq S$ is the set of final or accepting states

state action state

The set of strings accepted by M or the language of M

- L(M) = all strings which have a path from an initial state to an accepting state.
- Using finite state machines for recognizing or distinguishing (infinite) set of (finite) strings.

FSM - EXAMPLE



Accepts all binary strings with odd number of 1s

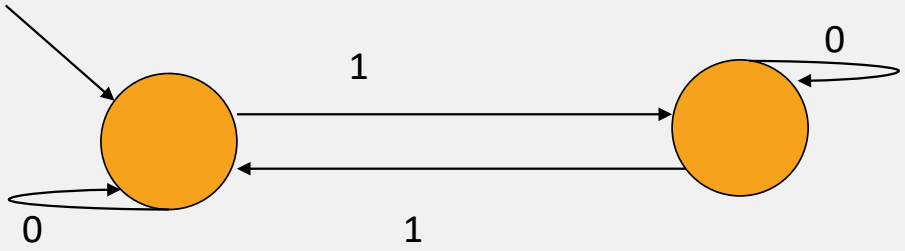
An infinite collection of finite strings

TRANSITION SYSTEMS

Transition systems go one step further where all states are accepting.

- TS = (S, S0, $\sum$, $\rightarrow$)
  - No notion of terminating or accepting states.
  - The alphabet $\sum$ labeling the transitions is also optional.
  - The traces captured by a transition system are obtained by unrolling the graph from the initial state(s).

TS - EXAMPLE



Traces captured by this transition system are

$(0^* 1)^* 0^\omega$

$(0^*1)^\omega$

TRANSFORMATIONAL SYSTEMS

Conventional notion of a terminating program.

- Takes in input.
- Performs computation step.
- Terminates after producing output.

System behavior

- Can be described as a transformation function over the input.

What about controllers ?

- In continuous interaction with the environment.

REACTIVE SYSTEMS

Continuously interacts with its environment.
No notion of system termination.

Interaction with environment is typically asynchronous.

Often consists of a concurrent composition of processes.

Often, its response to environment needs to obey time constraints.

REACTIVE SYSTEM BEHAVIOR

(Infinite) collection of infinite traces.

Traces denote ongoing interaction with environment.

Use state transition systems to describe behavior of a reactive system

- Too much complexity
- Many processes --- concurrency
- Each process has many states --- hierarchy
- What kind of inter-process communication?

The language of Statecharts addresses these practical issues !!

VISUAL FORMALISMS

Important/imperative at initial design stages.

Vital for communication.

Formal visual languages can help in:

- Documentation
- Initial analysis.
- Developing correct-by-construction translation to more detailed (non-visual) descriptions.

STATECHARTS

Statecharts =

- FSMs +
- Depth +
- Orthogonality +
- Structured transitions +
- Broadcast communication

Used in the Rhapsody tool.

Included in UML 2.0 as state diagrams.

STATECHARTS

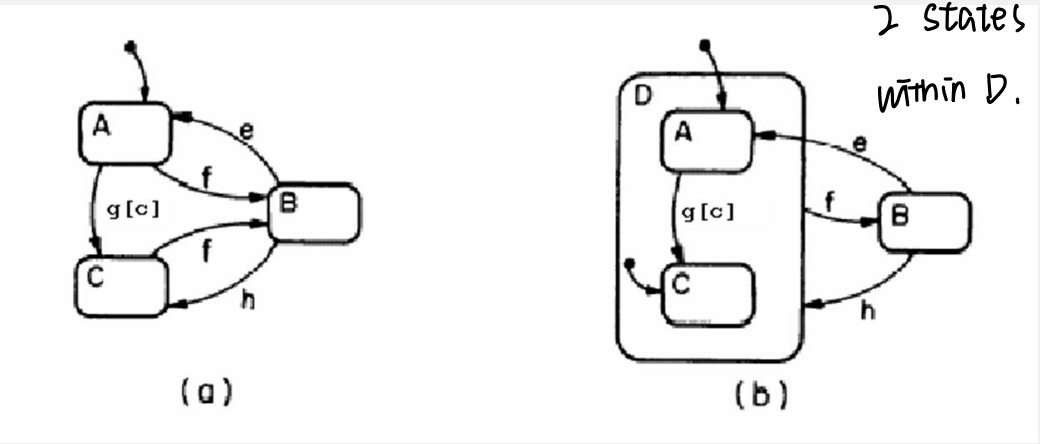
Depth:
- States can have internal structure.
- OR type states

Orthogonality
- Independent states
- Concurrency
- AND type states

Structured transitions
- Succinct descriptions of transition families.

Broadcast communication
- Succinct descriptions of synchronizations

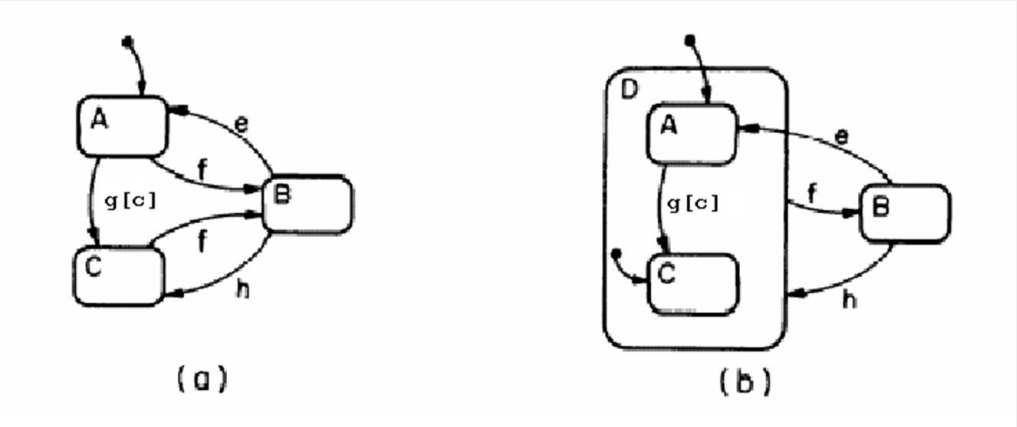
DEPTH: OR STATES



2 states only. D and B.
within D. internal structure

(b) is the statechart representation of the FSM (a).

DEPTH: OR STATES

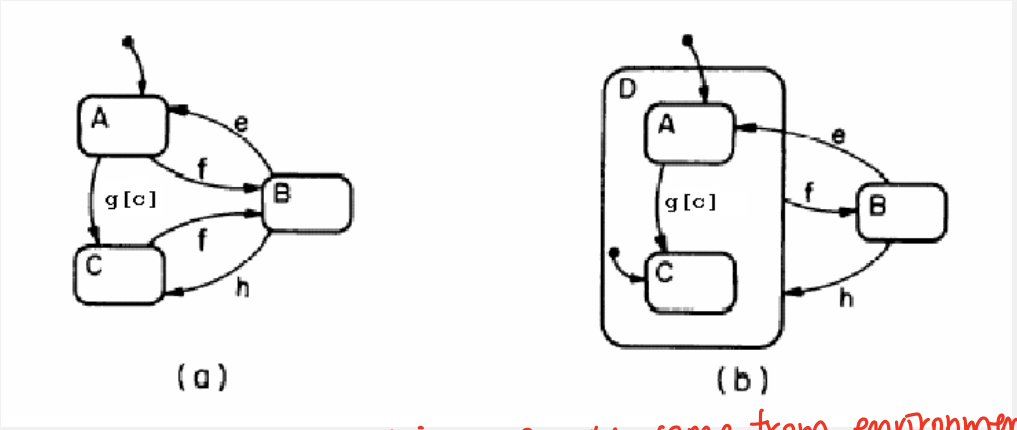


A and C are clustered into a superstate D

A and C are the internal exclusive-or components of the D state.

when in state D. it's actually either state A or state C.

DEPTH: OR STATES



trigger events come from environment.

When g comes, and condition c holds.

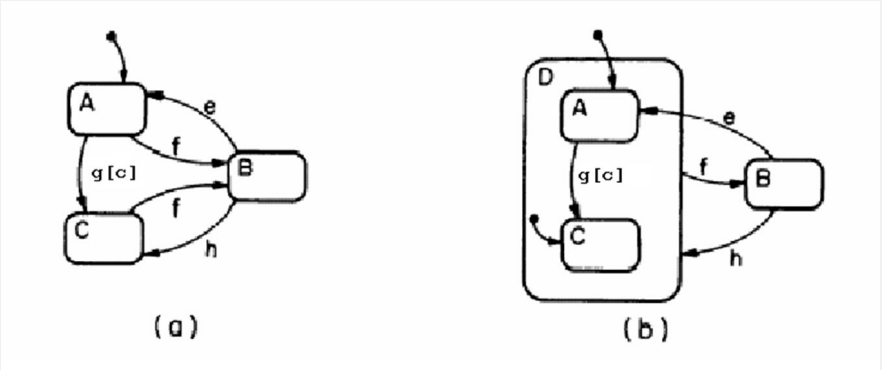
A transits to C

e, f are trigger (external) events.

g [c]: g, a trigger event and c a condition

g comes from environ.

DEPTH: OR STATES

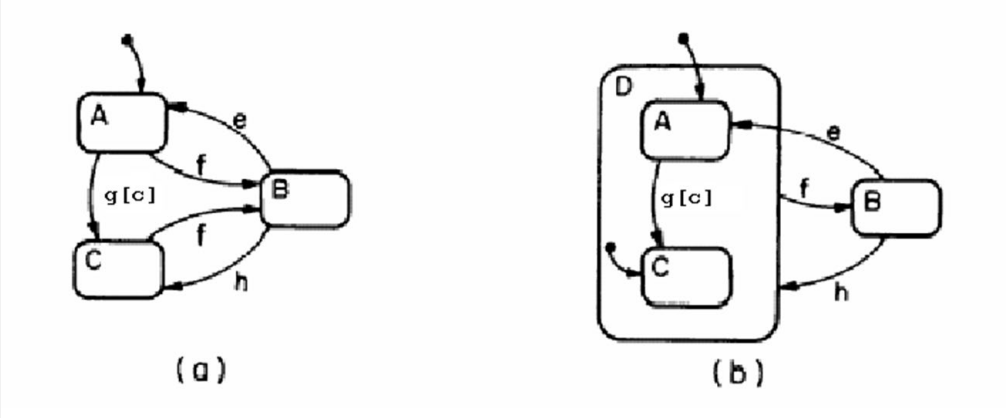


f is a transition from D to B. from superstate to another state.

From any D-state (A or C) there is an f-move to B.

(b). simplify (a).

DEPTH: OR STATES



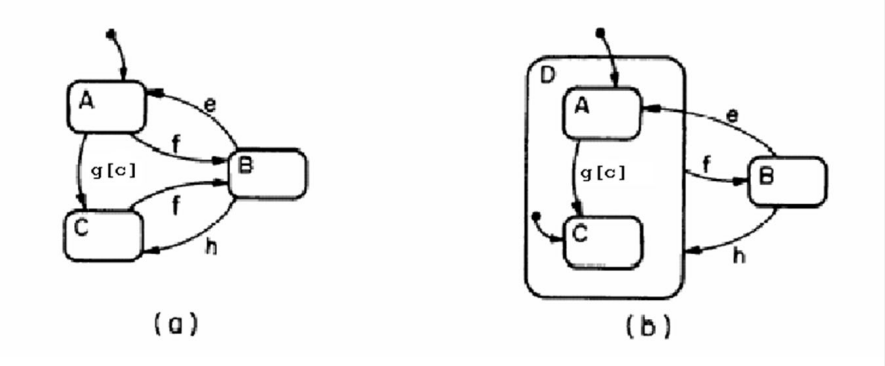
h is transition from B to D (A or C).

The actual state entered is the default entry state; the state C

outgoing transitions, f: from both A and C.

incoming transitions, h: to either A or C

DEPTH: OR-STATES

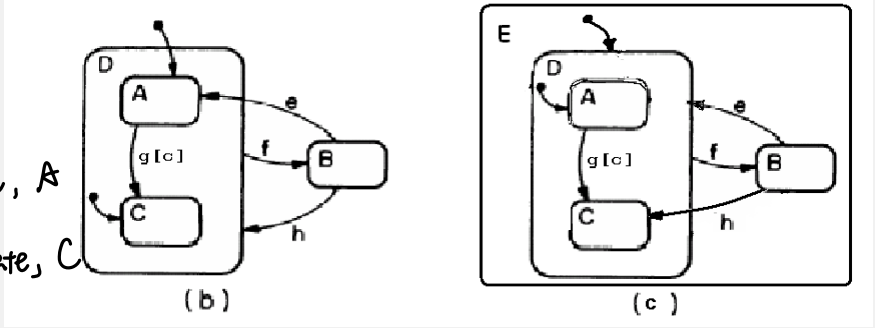


D is the initial state.

The actual initial state within D is not the default state C.

Instead, it is A.

DEPTH: OR STATES



① (b). A.
(c). default entry state, A

② (b). default entry state, C
(c). C.

③ A

E → D → A.

Which state will transition e yield in (b) and (c)?

Which state will transition h yield in (b) and (c)?

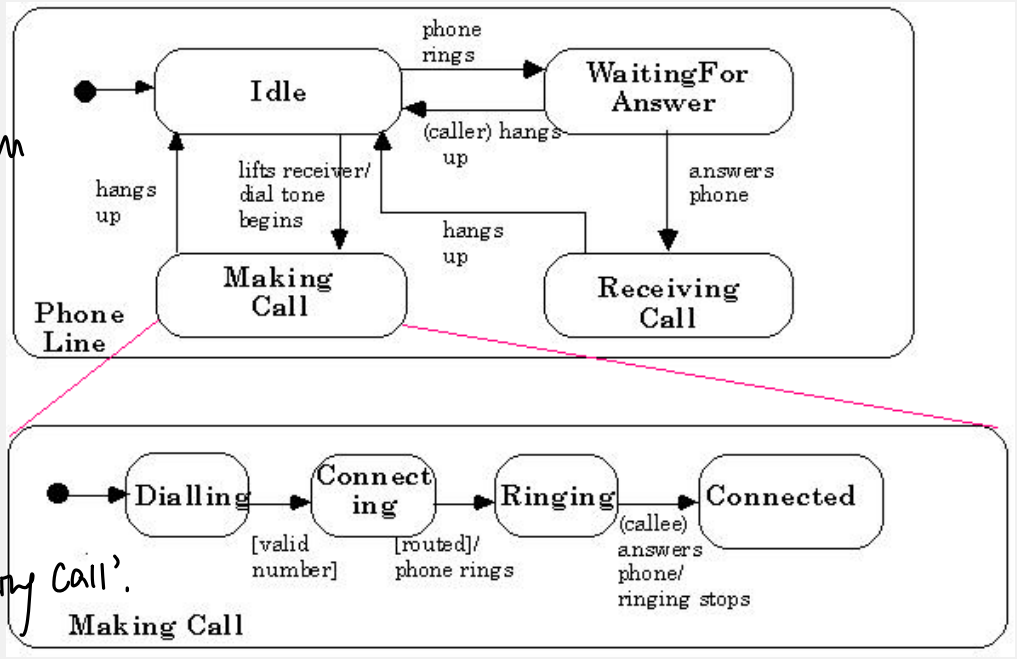
What is the default state for superstate E in (c)? Hierarchically!

two levels.

A CONCRETE EXAMPLE OF OR-CHART

4 superstates SM inside superstate 'phone Line'.

lower level SM of superstate 'making Call'.



OR-STATE: IN A NUTSHELL

Superstate active : only one substate active

An OR-state can contain other states as its internal substates (hierarchical internal structure);

A super OR-state is active, if and only if one of its immediate substates is active (exclusive or);

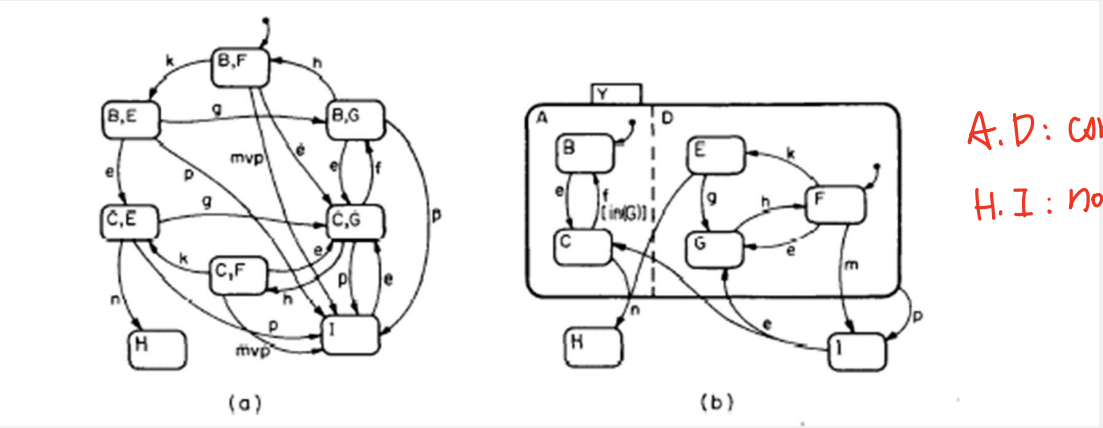
When the control enters a (super) OR-state, its default substate is entered and becomes active;

When the control leaves a (super) OR-state, all its substates become inactive!

More issues: history, priority, …

ORTHOGONALITY: AND-STATES

concurrent collection of states.

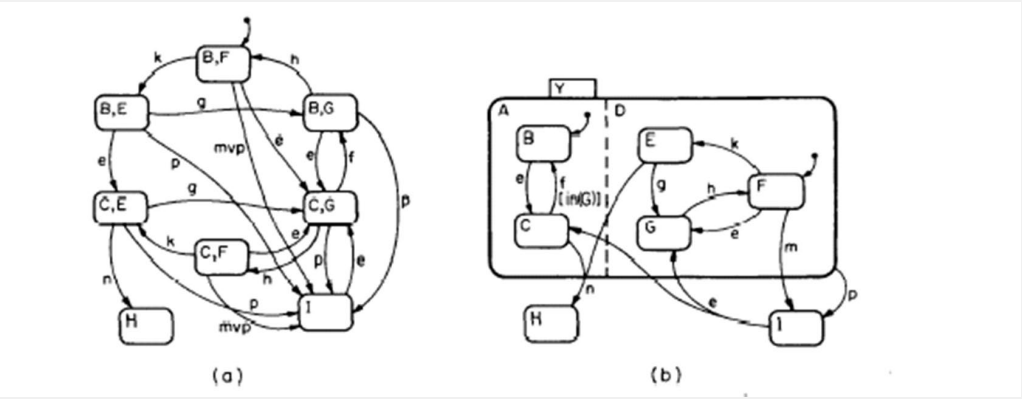


A.D: concurrent.

H.I: not depict concurr-ent composition.

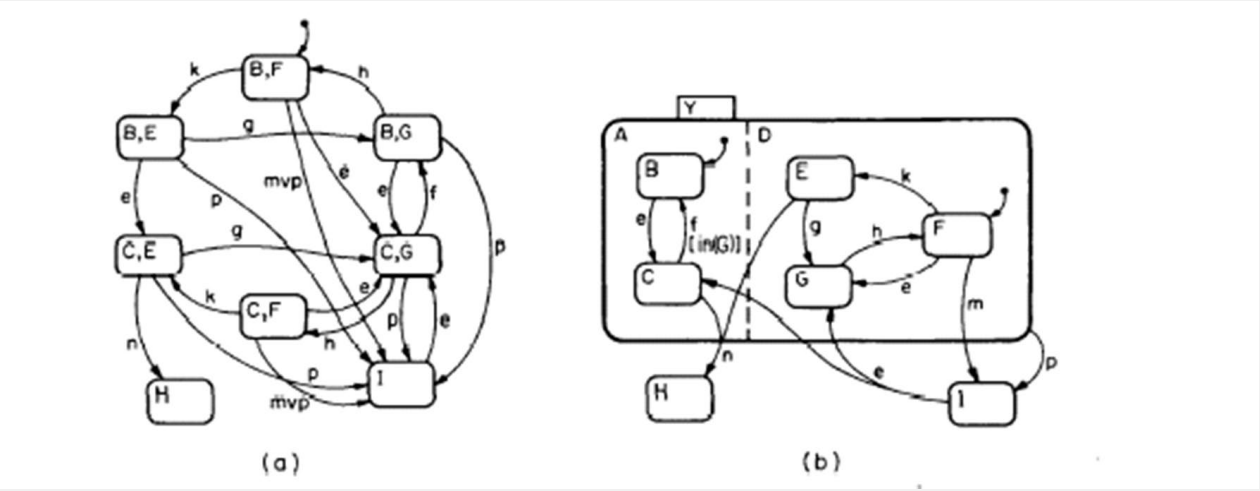
(b) is the statechart representation of the FSM (a).

ORTHOGONALITY: AND-STATES



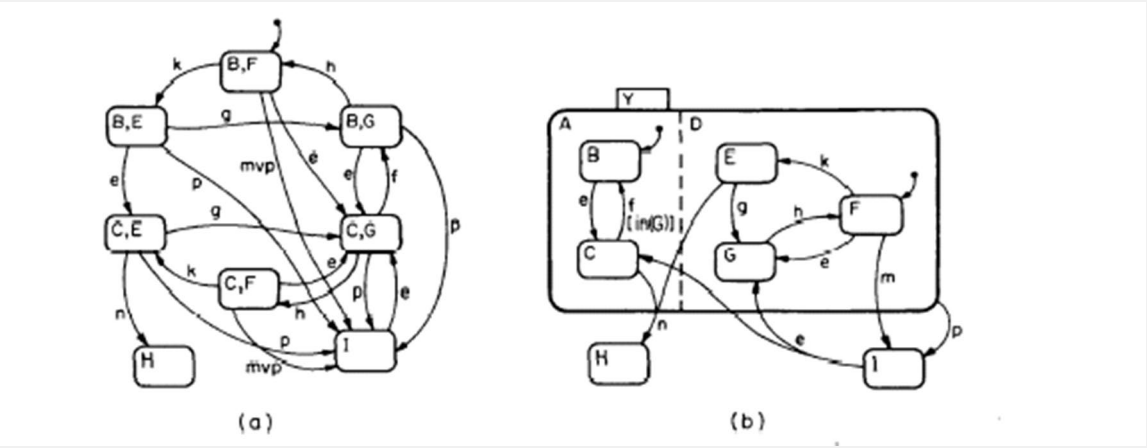
Y is an AND state.
It has two orthogonal components A and D.
A is an OR state with components B and C.
D is an OR state with components E, F and G.

ORTHOGONALITY: AND-STATES



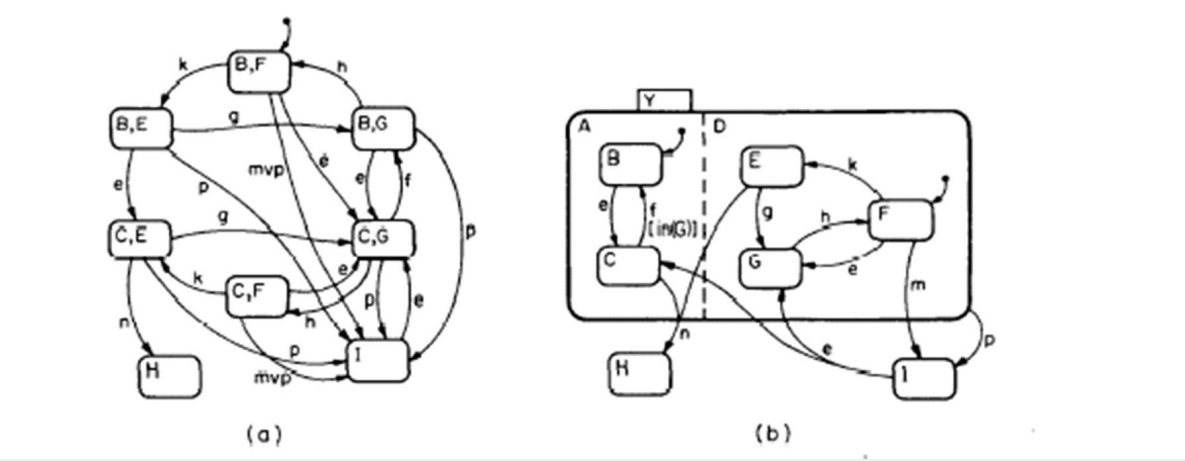
Y is an AND state.
It has two orthogonal components A and D.
a state of Y is composed of a state of A and a state of D
What is the default initial state of Y?

ORTHOGONALITY: AND-STATES



Y is an AND state.
It has two orthogonal components A and D.
a state of Y is composed of a state of A and a state of D
What is the default initial state of Y? (B,F)

ORTHOGONALITY: AND-STATES

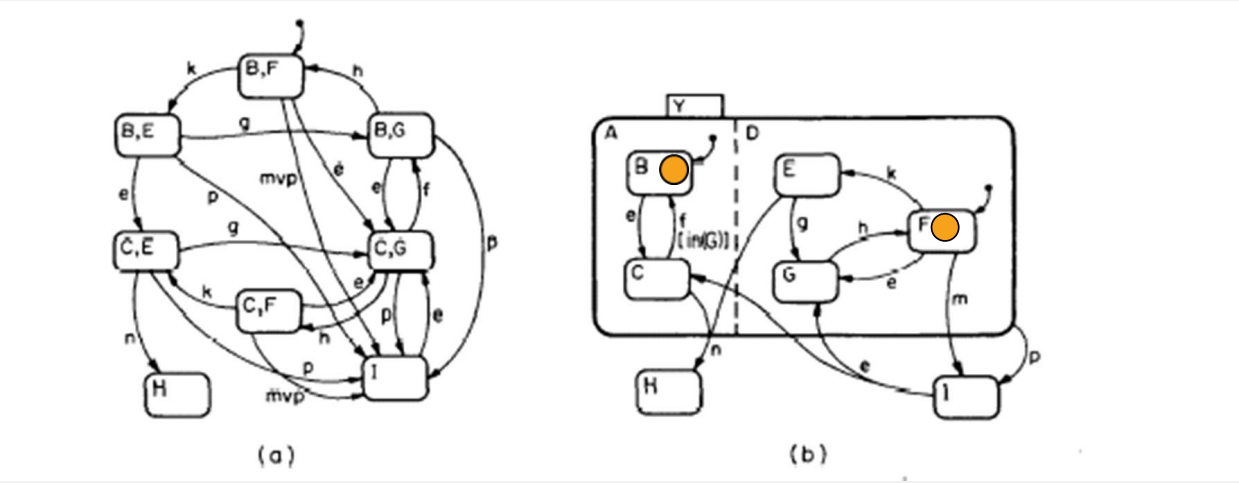


f belongs to only A.

e belongs to both A and D.

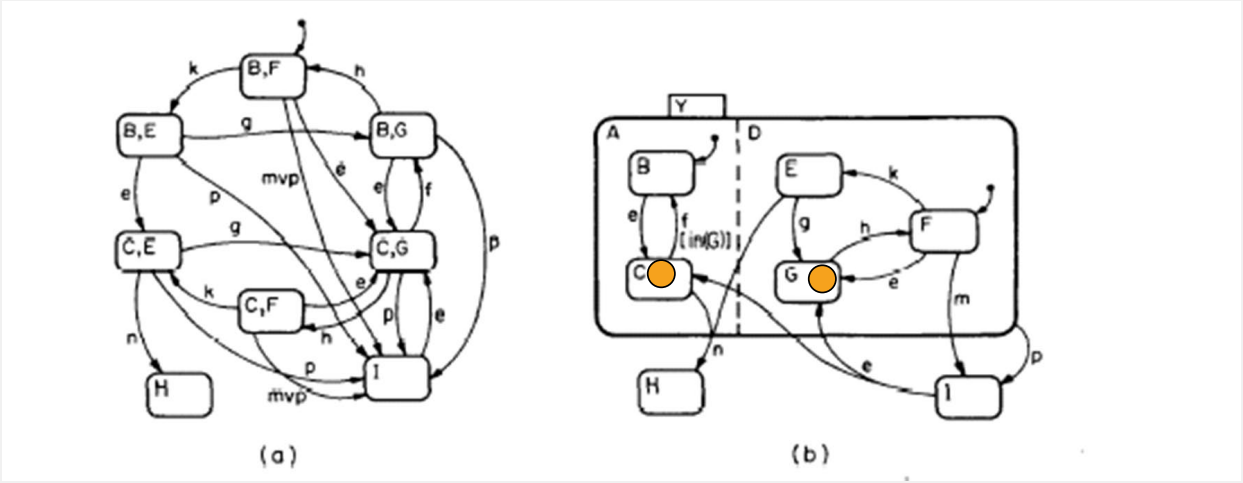
From (B,F) there is a simultaneous e-move to (C,G)

ORTHOGONALITY: AND-STATES



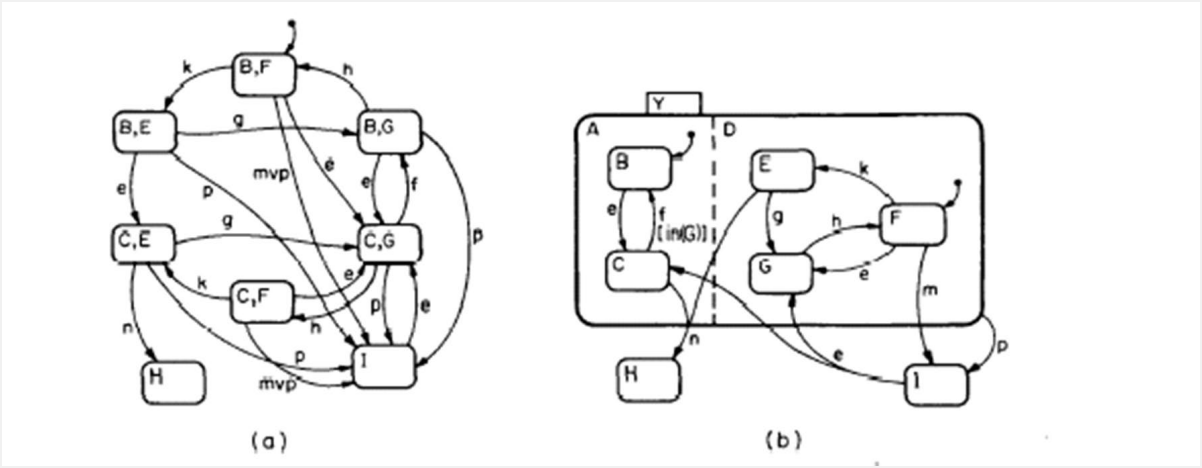
f belongs to only A.
e belongs to both A and D.
From (B,F) there is a simultaneous e-move to (C,G)

ORTHOGONALITY: AND-STATES



e belongs to both A and D.
From (B,F) there is a simultaneous e-move to (C,G)

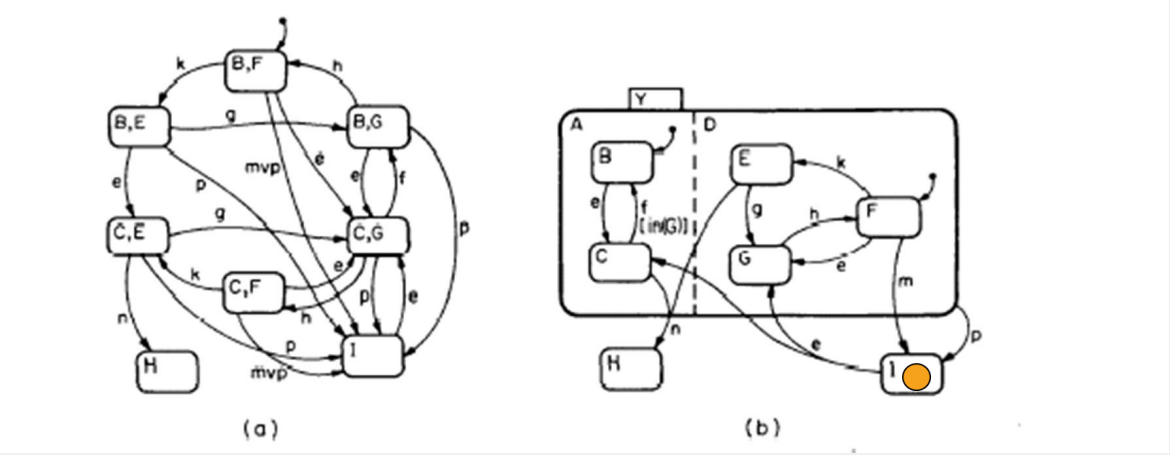
ORTHOGONALITY: AND-STATES



- From every Y state (how many?) there is a p-move to I

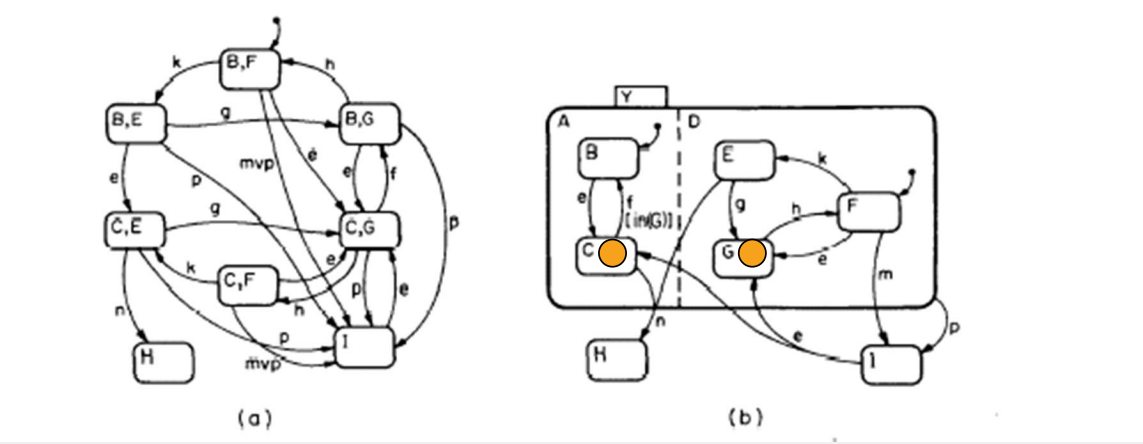
2x3=6.

ORTHOGONALITY: AND-STATES



From every Y state (6) there is a p-move to I
From I there is an e-move to the Y-state (?, ?) (C, G).

ORTHOGONALITY: AND-STATES

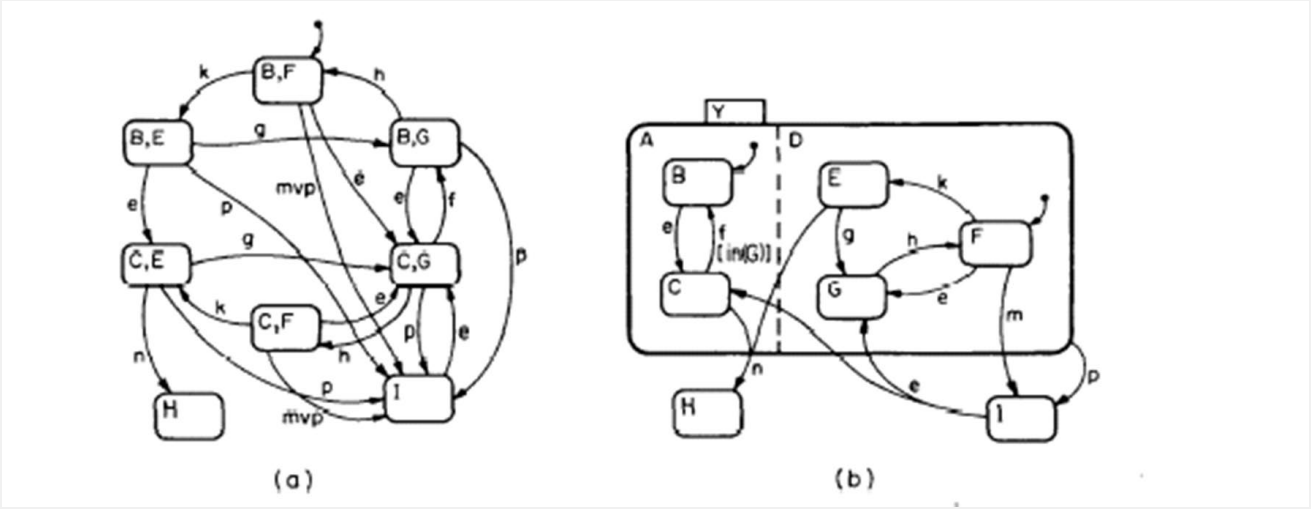


From I there is an e-move to the Y-state (C, G)

What if there is an e-arrow from I to just the surface of Y ?

default - entry state (B, F).

ORTHOGONALITY: AND-STATES



For each (?, F) state there is an m-move to I

Note the [in G] condition attached to the f-move from C (state reference!).

f-move from C to B only when D is in G.

AND-STATE: IN A NUTSHELL

An AND-state is composed of several independent (OR-)states that run in parallel (concurrency);

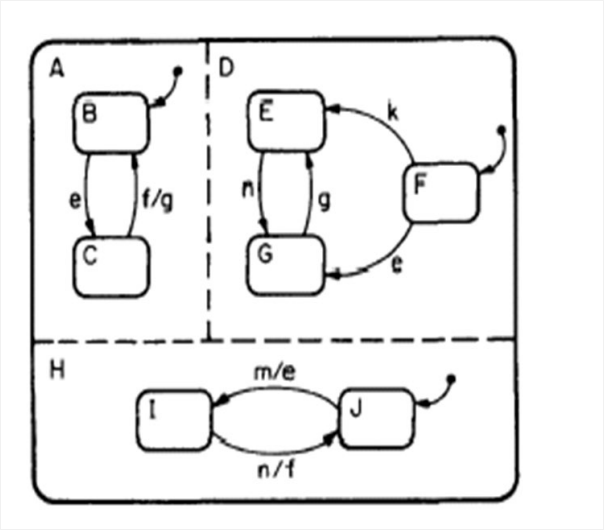
An *active state* of an AND-state comprises a state of each concurrent component, i.e., $(s_1, s_2, \ldots, s_n)$;

When the control enters (leaves) an AND-state, it simultaneously enters (leaves) all its components;

An AND-state can even occur inside an OR-state

between states, from state to environ : not a single step.

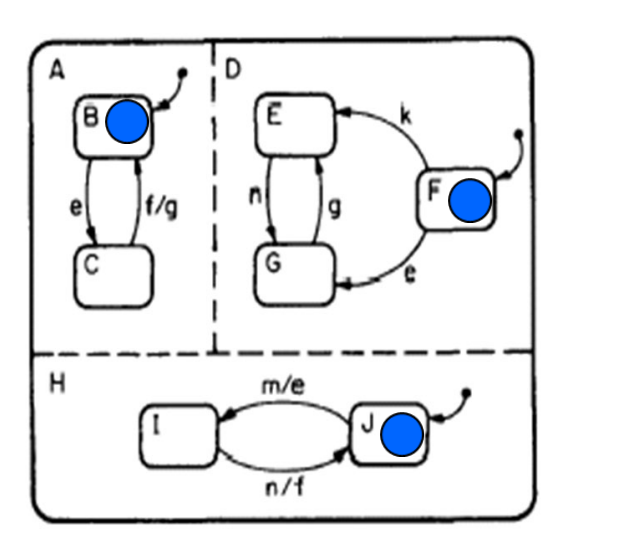
BROADCAST COMMUNICATION



A transition has a trigger and an action (output!)

But the output of a transition can be inputs for other orthogonal components!

BROADCAST COMMUNICATION



Start configuration (B, F, J)

m/e: m is the trigger event, while e is the action (output!)

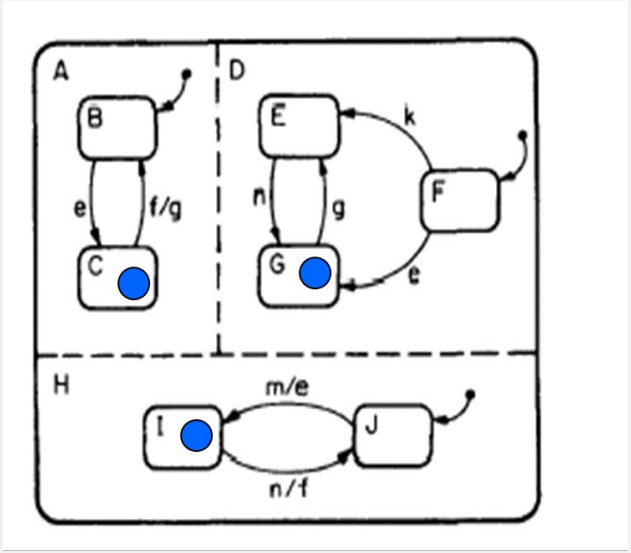
Suppose m (external event) occurs.

BROADCAST COMMUNICATION

Start configuration (B, F, J)

m occurs

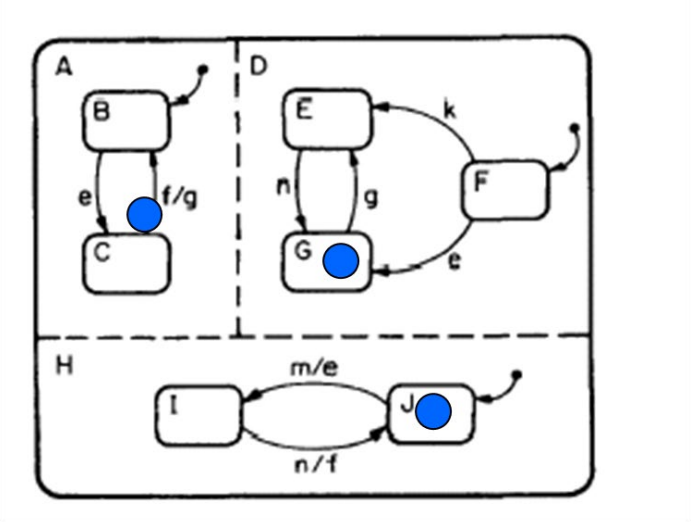
Final configuration (C, G, I)



Suppose event n comes,

What happen now?

BROADCAST COMMUNICATION

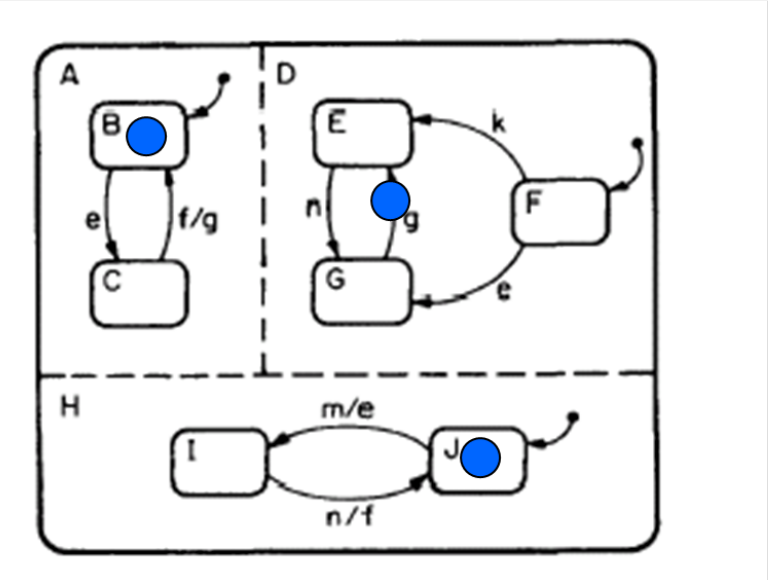


Now suppose event n comes,

What happen? Transition $I \xrightarrow{n/f} J$ is fired, f is generated,

which fires transition $C \xrightarrow{f/g} B$

BROADCAST COMMUNICATION

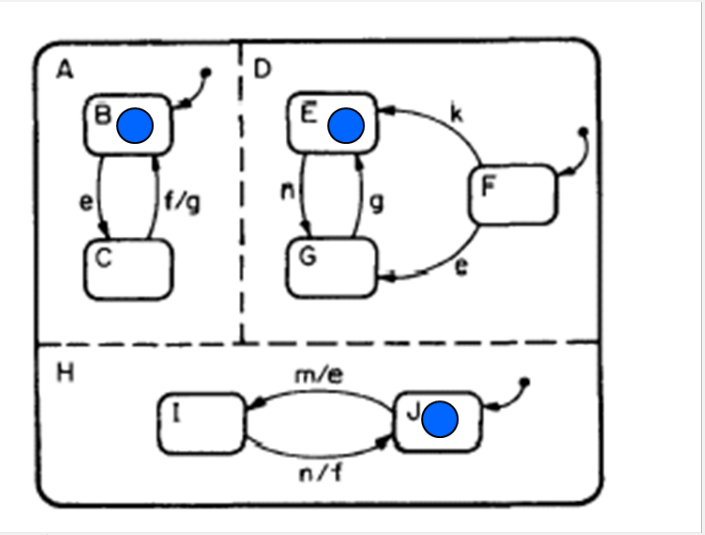


Now suppose event n comes,

What happen? Transition $I \xrightarrow{n/f} J$ is fired, f is generated,

which fires transition $C \xrightarrow{f/g} B$, which again fires $G \xrightarrow{g} E$

BROADCAST COMMUNICATION



Now suppose event n comes,

Transition $I \xrightarrow{n/f} J$ is fired, f is generated,

which fires transition $C \xrightarrow{f/g} B$, which again fires $G \xrightarrow{g} E$

Finally yielding (B,E,J)

WHAT ARE THE TRIGGERS/ACTIONS ?

Method call

- Method_name(parameters)

Or, Event

- Event_name(parameters)

Is there a difference?

- Lots, in terms of semantics
- A method call involves a transfer of control
  - If there are nested method calls, they can cause further transfer of control
- An event will be lodged in a system queue
  - It will be removed by the recipient later.

EVENTS AND METHOD CALLS

Event based communication

- Inherently asynchronous
  - Designer does not worry about controlling all interaction sequences (this is taken care of by the system queue)

Method call based communication

- Synchronous, involving transfer of control
- Involves close control by the designer over interaction sequences ---
  - getting closer to code level

MOST GENERAL FORM OF …

… **annotation for a transition**

- *Trigger[condition]/Action*

Trigger is event expression or method invocation

Condition is like a branch condition on data variables

Action is a program

- Sequence of event generation or method invocation or even code in a programming language.

SUMMARY

Practical Use of Statecharts in Modeling Object-based systems

- Use statecharts to describe behavior of classes (of active objects)
- Class Associations given by class diagrams.
- Contains code in the actions for realistic designs

A realistic approach for modeling control systems as well.

CS3213 FSE COURSE AT A GLANCE

- Software Engineering
  - Requirements -> Modeling -> Coding -> Testing
  - Requirements: Elicitation and Representation, Checking without a single LoC
  - Modeling: UML Notations, and variants such as Statecharts, Architecture modeling
  - Coding: Major part of the module and the project, Repair.
  - Testing: Validation concepts, Debugging.

EXAMPLE MODELING VIA STATECHARTS

Prof. Abhik Roychoudhury

National University of Singapore

EXAMPLE AIR TRAFFIC CONTROL SYSTEM

NASA CTAS

- Automation tools for managing large volume arrival air traffic in large airports.
- Final Approach Spacing Tool
  - Determine speed and trajectory of incoming aircrafts on their final approach.
  - Master controller updates weather info. to "clients"
    - controllers using inputs to compute aircraft trajectories.

  - **Model the Weather update subsystem from Requirements Document.**
  - **1-page requirements document available in Luminus.**
  - **Discussed in past classes.**

WEATHER UPDATE CONTROLLER

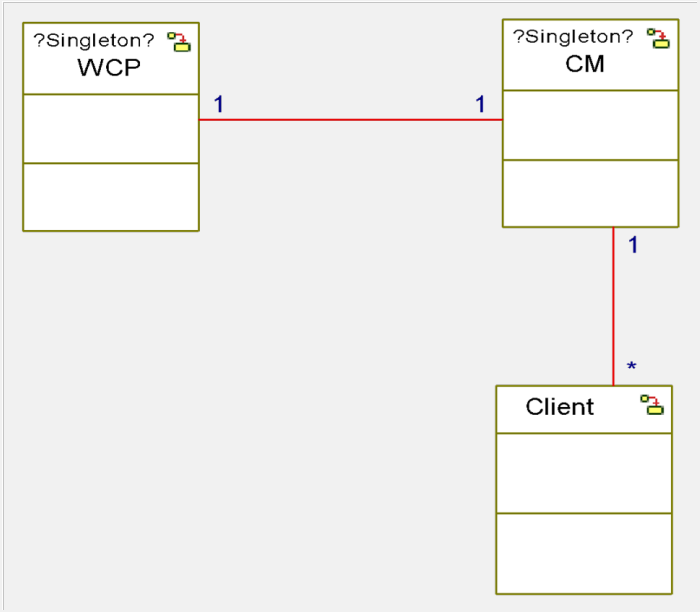
Part of the *Center TRACON Automation System (CTAS)* by NASA

- manage high volume of arrival air traffic at large airports
- http://ctas.arc.nasa.gov

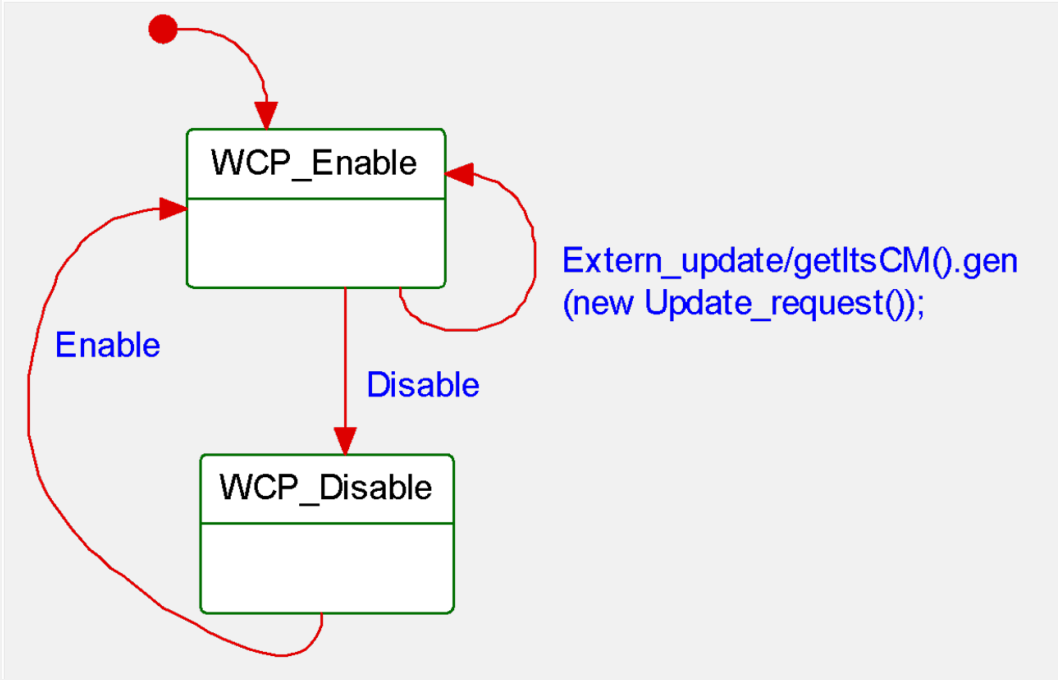
Control weather updating to all weather-aware clients

- A weather control panel (WCP)
- Many weather-aware clients
- A communication manager (CM)

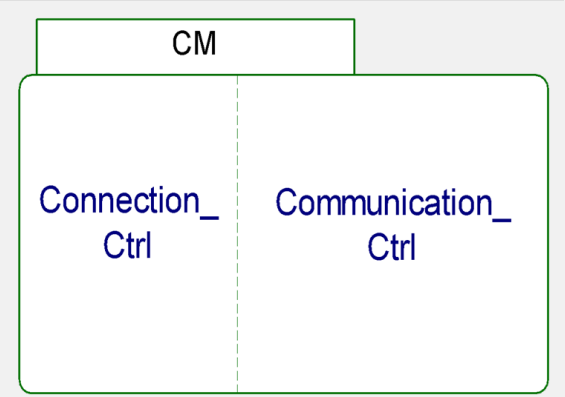
CLASS DIAGRAM



STATECHART FOR WCP



STATECHART FOR CM

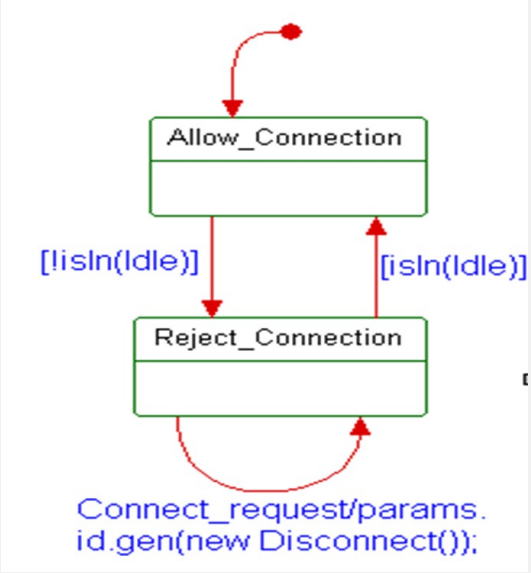


AND - States.

Connection_Ctrl: make sure a client can get initialized only when CM is neither initializing any other client nor performing an update

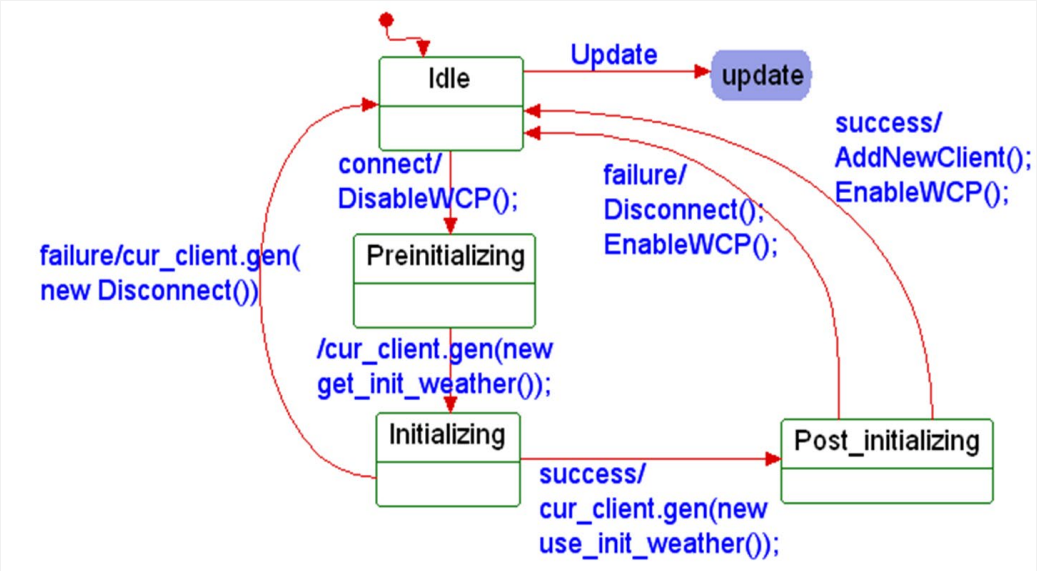
Communication_Ctrl: control initialization and update process

CONNECTION_CTRL

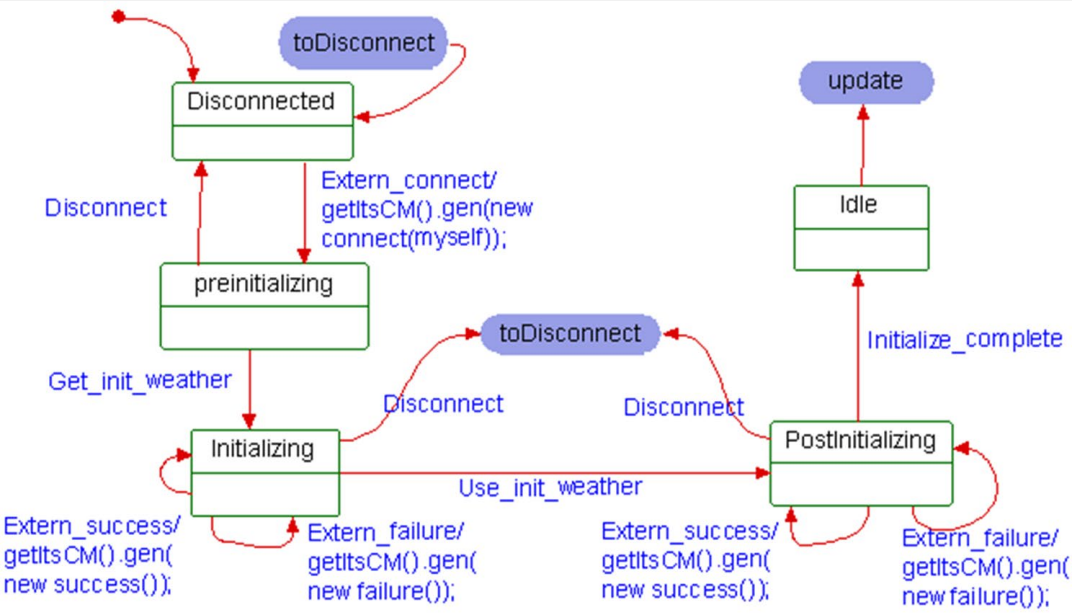


isIn(s): test whether the object (CM in this example) is currently in state s.

COMMUNICATION_CTRL (INITIALIZATION)



CLIENT (INITIALIZATION)



EXERCISE

- Construct a sample Sequence Diagram
  - (a) c1 connects to the CM successfully.
  - (b) c2 connects to the CM successfully.
  - (c) After that, a weather update request is sent by the WCP.
  - (d) Both c1 and c2 report success in getting new weather information.
  - (e) In the meantime, c3 tries to connect to CM. The connection request fails since the weather update is in progress.
  - (f) c1 successfully uses the new weather information, but c2 fails, which causes both of them disconnected from the CM.