ECPS 205
EECS 227

Cyber-Physical System Design

* office hour: Wed 2pm - 3pm

→ IT is being embedded into growing range of physical devices linked together through networks.

→ Trends.
a) proliferation (rapid increase)
b) integration at scale
c) Biological evolution (slow)

increasing autonomy → direct world access

* Embedded systems
They are information processing systems embedded into a larger product

* CPS
= Embedded System + Physical environment

PES
MCCI Physical engineered systems whose operation are monitored, coordinated controlled & integrated by computer and communication core.

Characteristics of CPS

a) Dependability ✓

* Reliability . R(t)
* Maintainability . M(t)
* Availability @ t
* Safety : no harm
* Security

b) Trust worthy computing.

→ Computing = HW + SW + people

c) efficiency. ✓
d) system resp.
e) Real - time constraints ✓

CPS are reactive systems

↳ interact with environment

executes at pace dictated by environment

f) need to be dedicated system.

challenges

a) Societal challenges

b) Scientific.
1. computation and abstraction
2. System and network support

c) integration with legacy devices.

→ Model (imitation).
Model specifies what a system does

Design
How a system does what it does. This includes optimisation.

Analysis
Why a system does what it does

Clear idea of Model

A model is an artifact that imitates the system process or artifact of interest

Modeling Recipe.

1. State the problem. PROBLEM??
Simple language to describe the problem.

2. Model the physical process. ODE.
what process it wants to do
- Basic observation
- Simplified representation of real systems

3. Characterise the problem. Parameters
- isolate parameters
fixed, adjustable parameter.
- identify quantities that characterize the physical process
- understand
how physical process may interact with computation.

4. Devise a control algorithm.
How would you control the process
To specify req on latencies, delays, sampling rates.

5. Select model of computation.
MOC is a set of allowable instruction used in computation.

6. Specify hardware.
Select hardware that is capable of withstanding the environment
Calculate worst case execution time measurement. ↳ longest path

7. Simulate. Cyber twin
- use platform based design to seperate application logic and architecture specific hardware

8. Construct.
plan to construct so that you can co-iterate between simulation & testing.

9. Synthesize software
- correct by construction

10. Verify, Validate, test.
- test each subsystem & component independently

works on smooth motion,
* linearity, time invariance, continuity

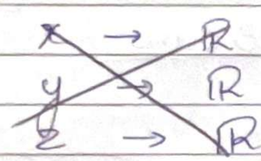
Modeling of continuous dynamics

→ Motion of mechanical parts may be modelled using differential equation.

→ Six degree of rotation {for modelling helicopter tail}

x y z → position
pitch yaw roll → orientation

Continuous time - signals



$x: \mathbb{R} \to \mathbb{R}$
$y: \mathbb{R} \to \mathbb{R}$
$z: \mathbb{R} \to \mathbb{R}$

range

time

$X: \mathbb{R} \to \mathbb{R}^3$

→ Velocity $\dot{x}(t) = \frac{d}{dt} x(t)$

→ accelaration

$$\ddot{x}(t) = \frac{d^2}{dt^2} \dot{x}(t)$$

$$x(t) = \underbrace{x(0)}_{\text{initial position}} + t\,\dot{x}(0) + \frac{1}{M} \int_0^t \int_0^T F(\alpha)\, d\alpha\, dt$$

Ordinary differential equations

discrete

Collision of mechanical parts are modelled by instantaneous events (hybrid system modelling)

Orientation : $\theta : R \to R^3$
Angular Velocity : $\dot{\theta} : R \to R^3$
Angular accelaration : $\ddot{\theta} : R \to R^3$

Torque : $R \to R^3$ Nm

$$\theta(t) = \begin{bmatrix} \theta_x(t) \\ \theta_y(t) \\ \theta_z(t) \end{bmatrix} = \begin{bmatrix} \text{roll} \\ \text{yaw} \\ \text{pitch} \end{bmatrix}$$

Angular version of force is torque. Nm

$$T(t) = \frac{d}{dt} \left(I(t)\, \dot{\theta}(t) \right)$$

Moment of inertia ~~tensor~~ tensor

$$T(t) = I \ddot{\theta}(t)$$

$$\theta(t) = \theta(0) + t\,\dot{\theta}(0) + \frac{1}{t} \int_0^t \int_0^T T(\alpha)\, d\alpha\, dt$$

yaw dynamics.

$$T_y(t) = I_{yy}\, \ddot{\theta}_y(t)$$

gives the input and output relation

$$\theta_y(t) = \theta_y(0) + \frac{1}{I_{yy}} \int_0^t T_y(T)\, dT$$

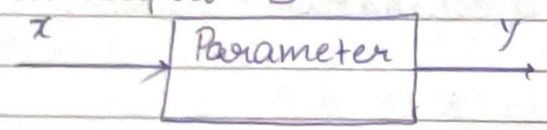
Model order reduction

Reducing the dimensions that are considered.

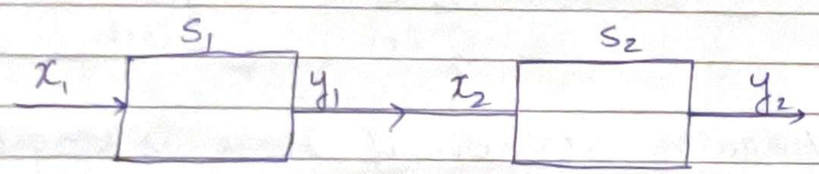
Tail rotar applies torque to counterbalance the top rotar.

Actor model of systems

* System is a function that accepts ip signal & yeilds output. S

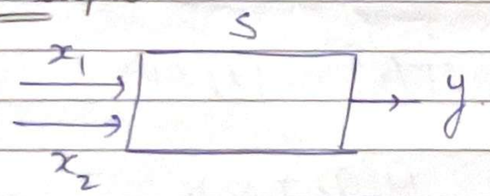


$X = Y = (\mathbb{R} \to \mathbb{R})$



$\forall t \in \mathbb{R} \quad y_1(t) = x_2(t)$.

Multiple inputs



$S : (\mathbb{R} \to \mathbb{R})^2$

Discrete (distinct) Systems

seperate

→ A discrete system operates in a sequence of discrete steps and is said to have discrete dynamics.

→ Continuous systems evolve smoothly, while discrete systems evolve abruptly.

→ Pure signal → {absent, present}

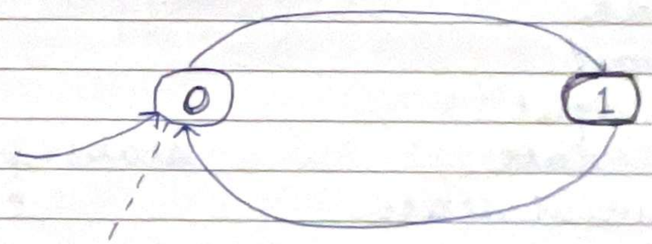
This might have been a Question

Reaction: System wakes up & acts when the signal is only present.

State Space encoding of past and can have impact on current and future inputs

→ State is the summary of the past.

up ∧ - down/1 ; - - → output valuation

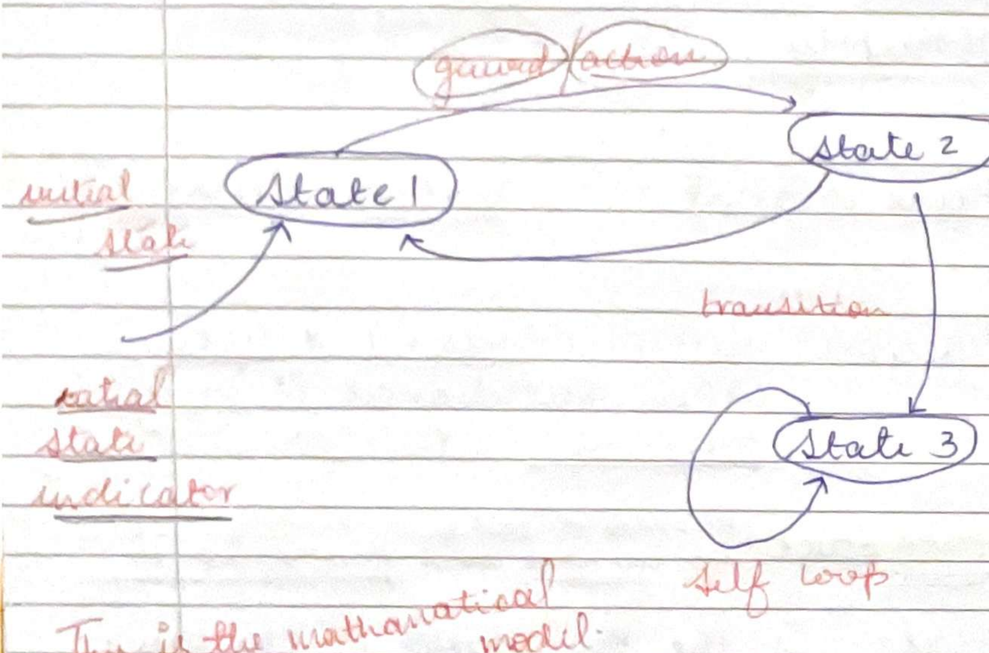


initial state

down ∧ - up /0

Discrete models with finite state spaces are called FSM

FSM : Finite State Machine

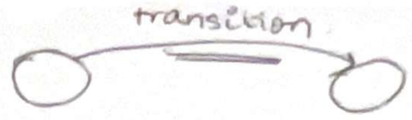


This is the mathematical model.

→ FSM has 5 tuples

(a) state — $\{0 \ldots M\}$
(b) input
(c) output
(d) update — states × ip → state × op
(e) initial state — = 0

eg: garage counter

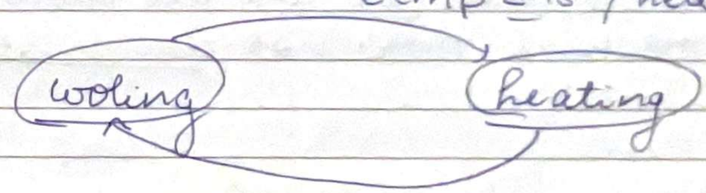


examples of gaurd for pure signals.

true		Transition is always enabled
p_1		Transition is enabled if p_1 is present
$-p_1$		Transition is enabled if p_1 is absent
Both	$p_1 \wedge p_2$	" if both are present
either	$p_1 \vee p_2$	" if either are present
	$p_1 \wedge -p_2$	" if p_1 present, p_2 absent

→ Thermostat

ip : temp : $\mathbb{R}$
op : heat on, heat off : pure



2 conditions to turn on or off to prevent chattering

Which means heat would turn on & off rapidly close to set points.

→ FSM can be

a) Event triggered ✓

b) time triggered ✓✓

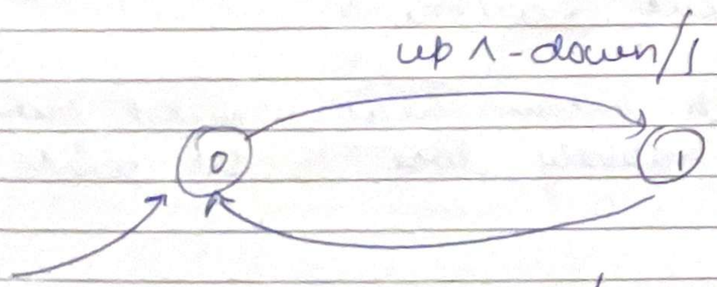
q_1 — q_1 pres q_2, q_3 absent.
q_1, q_2 — q_1, q_2 pres q_3 absent.
$q_3 = 1$ — q_3 pres with value 1

Stuttering Reaction

Stuttering reaction
- the ip and op are all present and the machine does not change state.

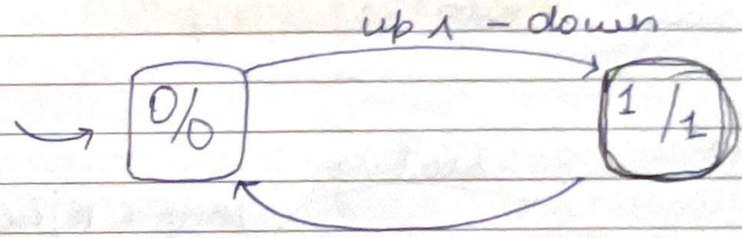
Mealy FSM → output in a transision.

Mealy machine are characterised by producing outputs when a transition is taken.



Moore FSM → output in a state

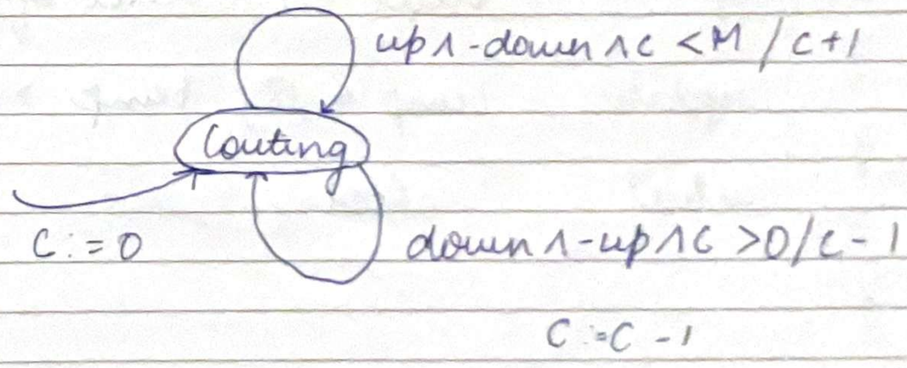
- Moore machine produces outputs when the machine is in a state, rather than when a transition is taken



ip : ups, down : pure.

transition as a variable

- An extended state machine solves this problem by augmenting the FSM model with variables that may be read and written as a part of taking a transition between states



FSM → The Variable is the number of states you have.

__/__/__

n - ~~discrete~~ states

m - variables

p - possible value.

$n \times p^m$

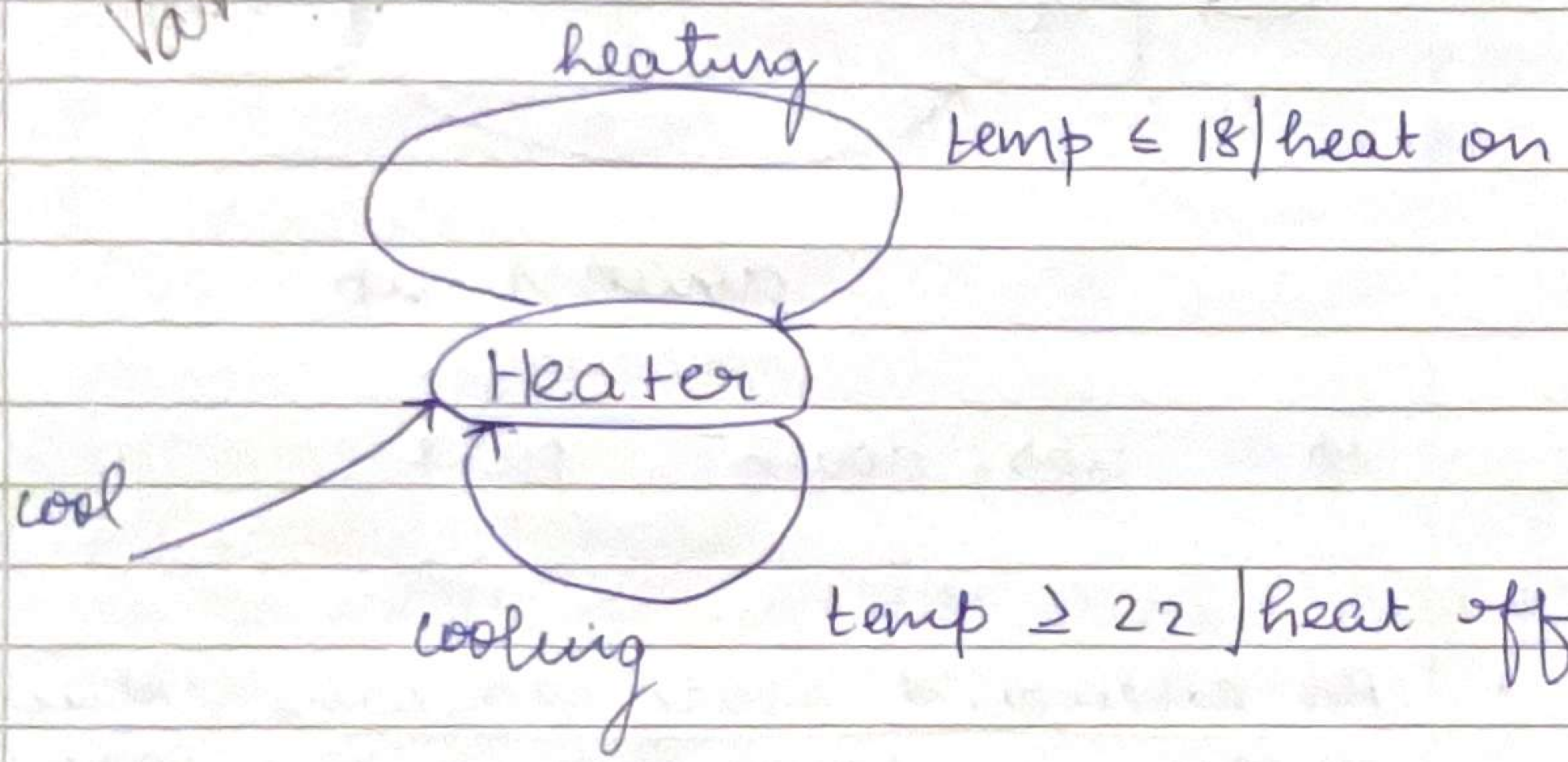
n - number of states   states = $n p^m$

m - number of variable

p - possible values

$|states| = n p^m$

Var ??



state : { heating , cooling }

ip : temp → ℝ

op : heat on, heat off

update : temp ≤ 18, temp ≥ 22

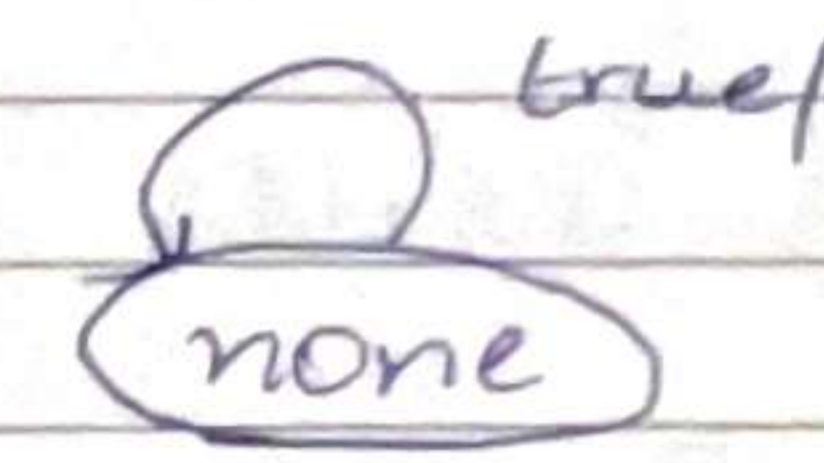
initial : cool

Q. ~~sing~~ signal FSM, how are there 3×61 + ~~6~~ 5 states & why exactly do we need the pending state for

__/__/__

→ Determinism: In every state, for all input values exactly 1 transition is enabled

non - determinism



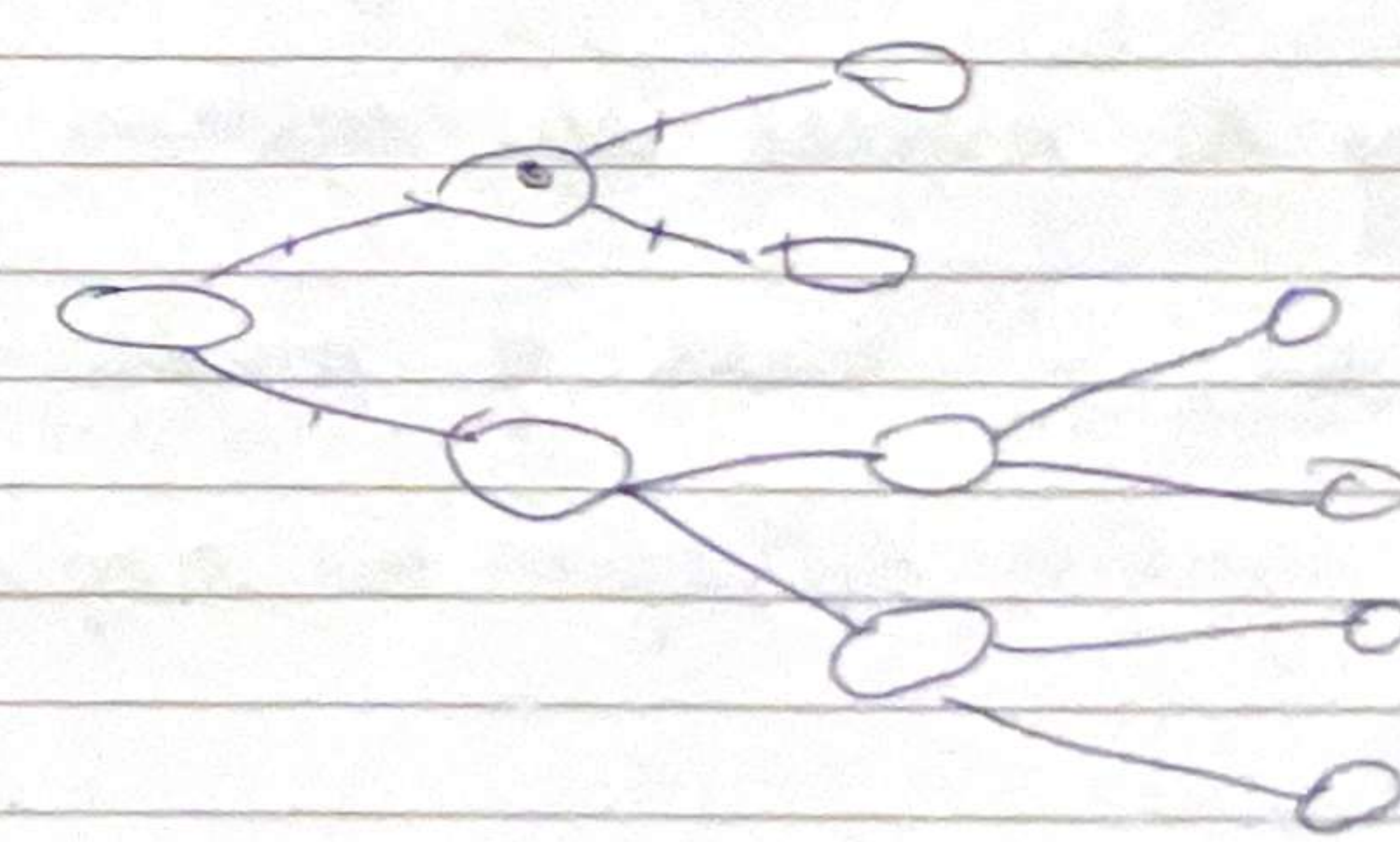
⟹ non - deterministic FSM has 5 tuples where the update ~~func~~ is replaced by a function

{ possible updates : states × inputs ⟶ $2^{States \times Op}$ }

☆ computational tree

A graphical representation of all possible traces

☆ A trace : is the record of inputs, states and outputs in a behavior



* ND - FSM

They are used to modal the environment.

* for fixed inputs

1) Deterministic system gives a single behavior

2) ND system gives a set of behavior

* but if the behavior are probabilities than the model is a **Stochastic Model**

FSM provides:

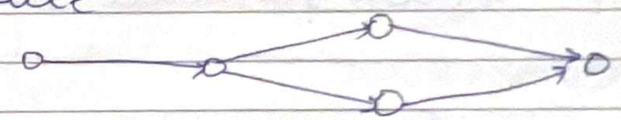
1) A way to keep the system
2) Way to model the system
3) What it must & must not do.
4) If system satisfies the requirement.

Hybrid Automata Technique to model — Hybrid systems —

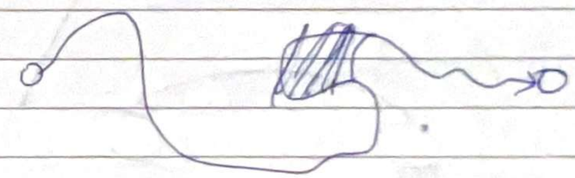
Hybrid Models

CPS commonly combines both discrete & continuous dynamics to give HYBRID MODEL

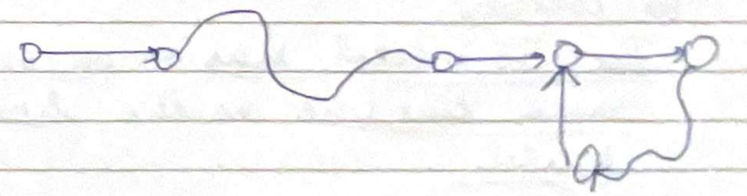
Discrete



Continuous System



Hybrid

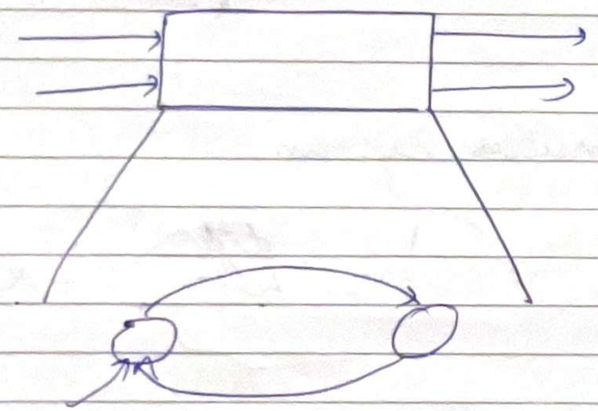


* Hybrid System arise where we have a digital controller for physical 'plant'

eg: thermostat,

* They also exist in multi-agent systems.

→ Actor model for state machine.



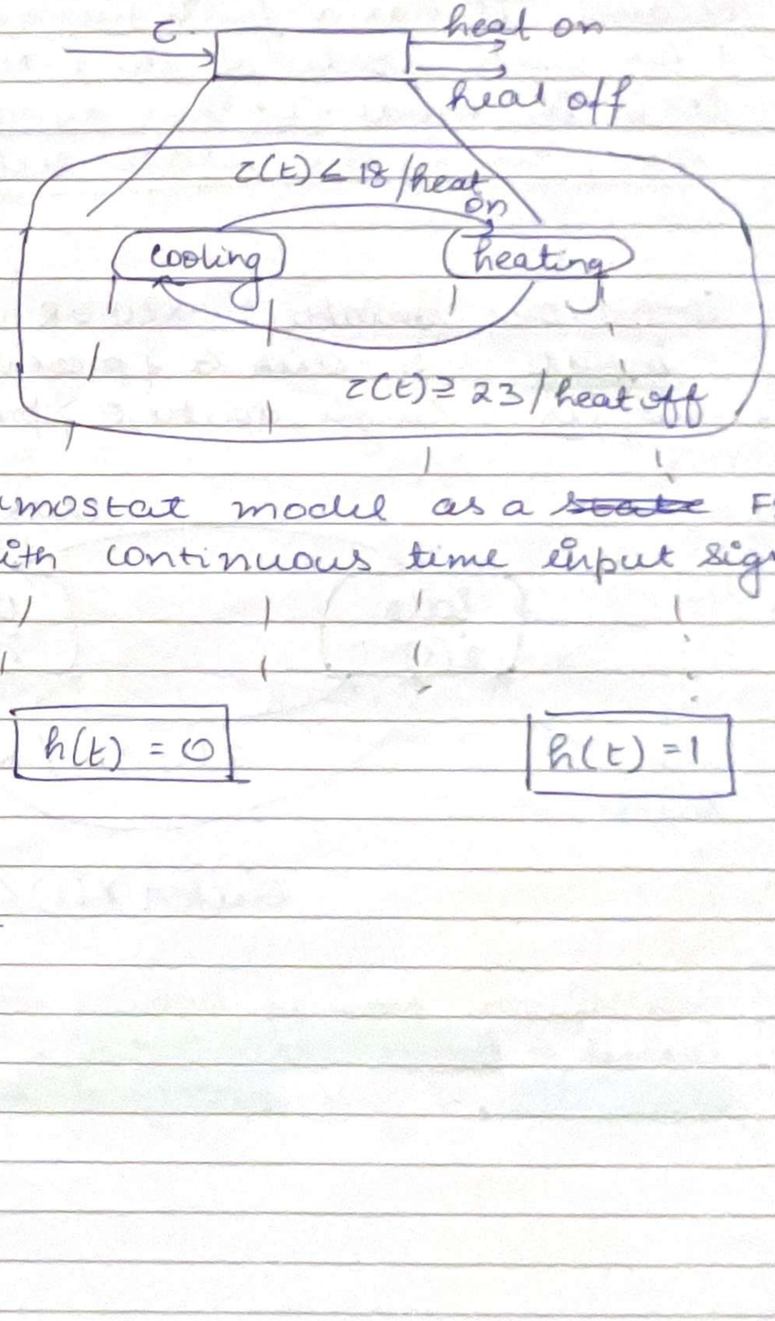
* for both state space model & actor model to coexist,
The SS model has to be on the same time line as the time-based model.

→ Make it a time based system!

Refinement: state, input, output to a time based system

eg: Heater

τ = temp



Thermostat model as a ~~state~~ FSM with continuous time input signal

output is also continuous time.

$h(t) = 0$

$h(t) = 1$

Modal Model — Hybrid System

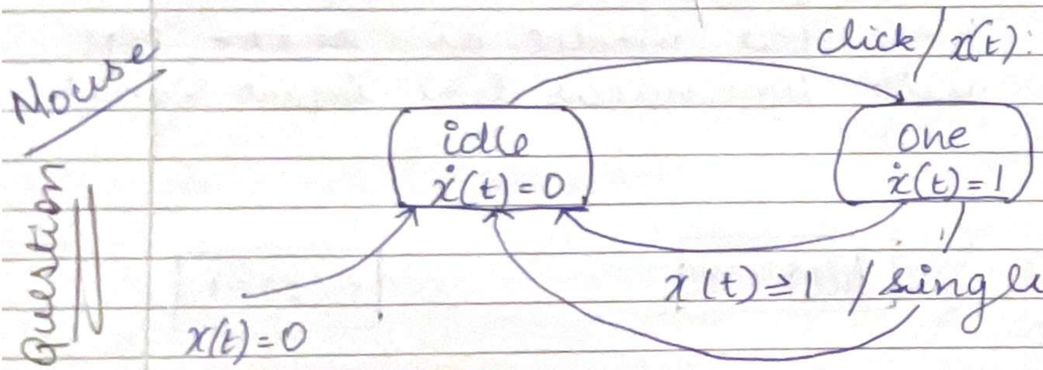
* Hybrid system is called modal model because it has a finite number of modes 1 for each state of the FSM & when it is in mode. It has dynamics specified by the state refinment.

eg:

Continuous variable : $x(t) \in R$

inputs : click $\in$ {present, absent}

outputs : single, double, $\in$ {present, absent}

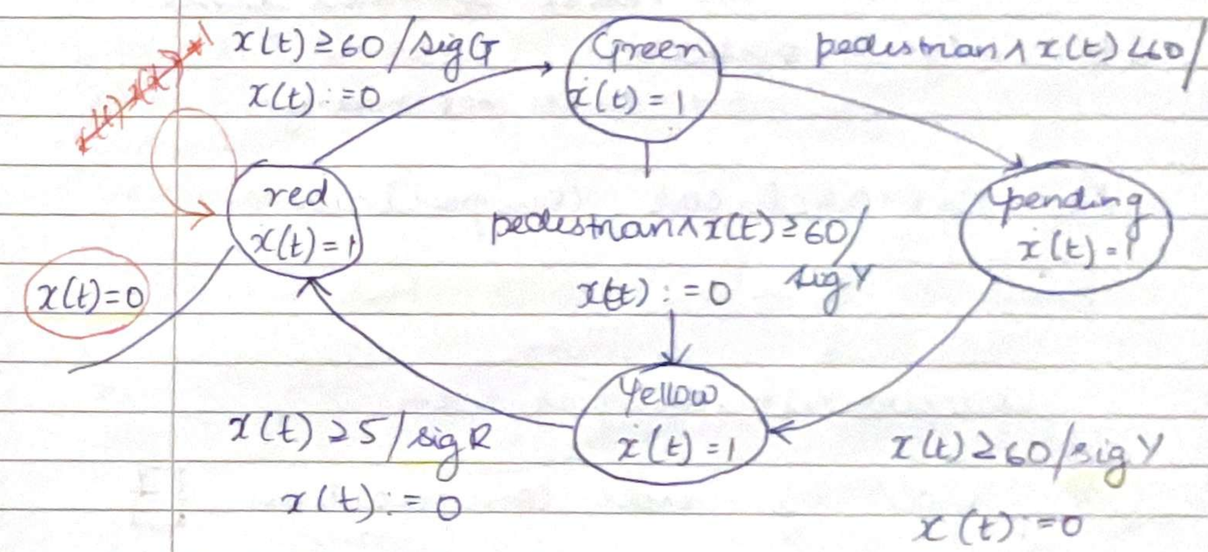


ESM
Variable states in FSM
& output based
Modal
change it to time based.

orm of hybrid system is
d automation, where the
just passage of time.

Hybrid system or modal model has infinite states

Continuous variable x(t) : R.
input : pedestrian
output : sigR, sigG, sigY



Lecture-5

→ Composition of state Machine

1) Concurrent Composition.

2 or more state machine react either simultaneously or independently simultaneously and instantaneously

1.a) Synchronous
↳ react at same time.

1.b) Asynchronous.
↳ react seperately.

2) Hierarchical composition.

Concurrent composition

* Side - by - side composition 
* Cascade composition 
* Feedback composition 

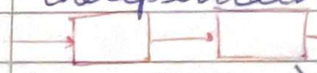
Lecture - 5

Compositions.

concurrent, sequential
cascade, 1 of 1 with the other

Concurrent S & I — Hirarchical.

* State machine react together or independently.

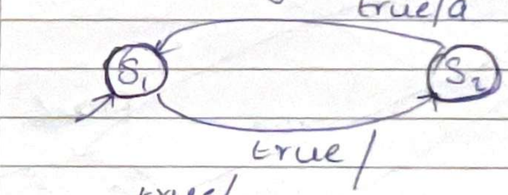


(Synchronous, Asynch).
- side - by - side
- Cascade
- feedback

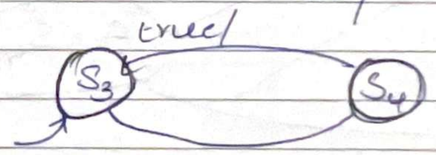
} can be

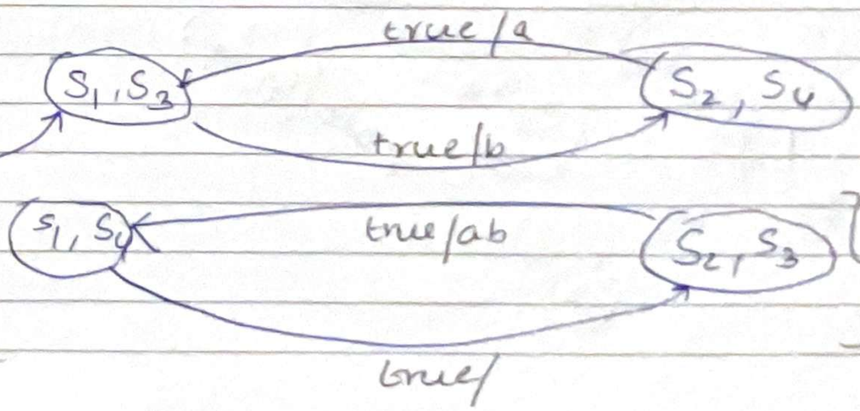
→ Side by side

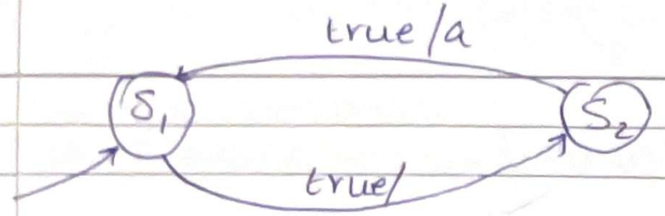
eg:

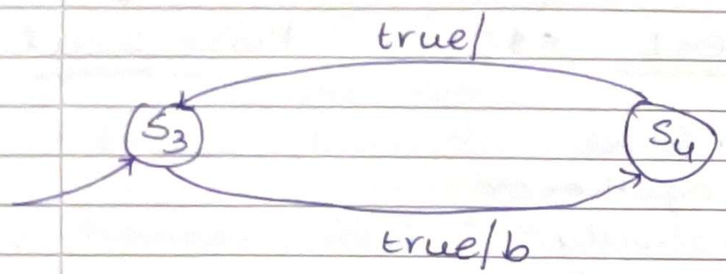


Both A, B work at the same time
Synchronous comb

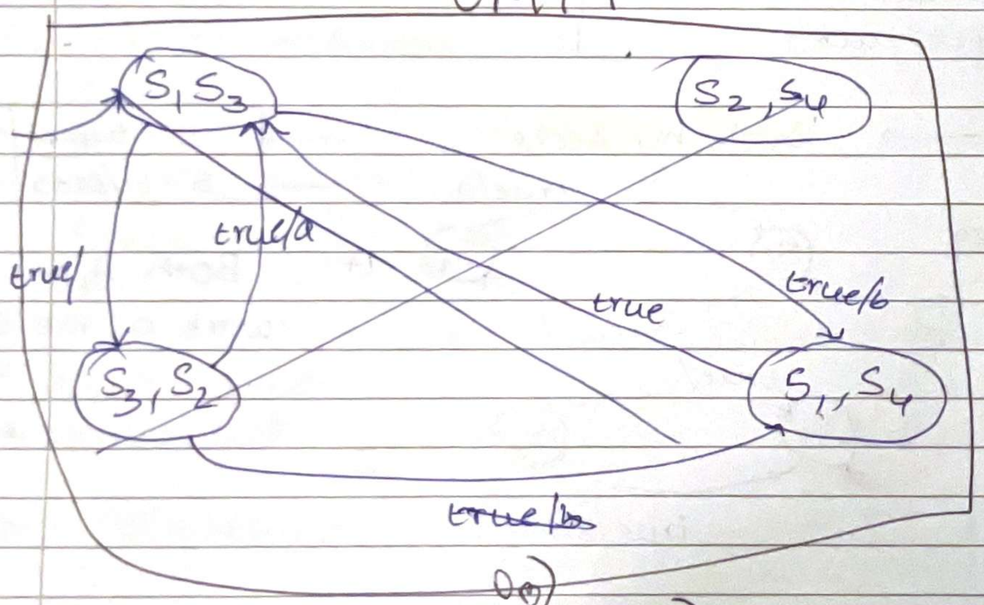








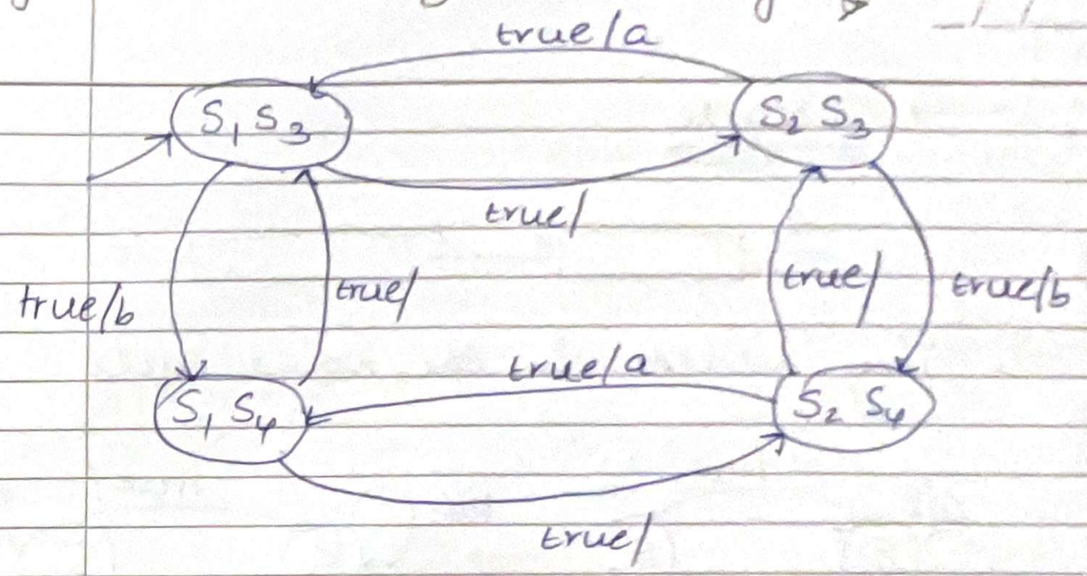
OMIT



$(S_A, S_B)(i_A, i_B) = (S'_A, S'_B)(O_A, O_B)$

$(S'_A, O_A) = \text{update}(S_A, i_A)$

Asynchronous using interleaving semantics



* All States are reachable.

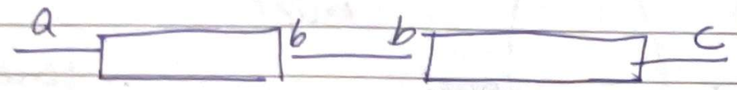
Stuttering: Reaction has no output is produced & the state is not changed.

{ Synchronous with stuttering ≠ Asynchronous }

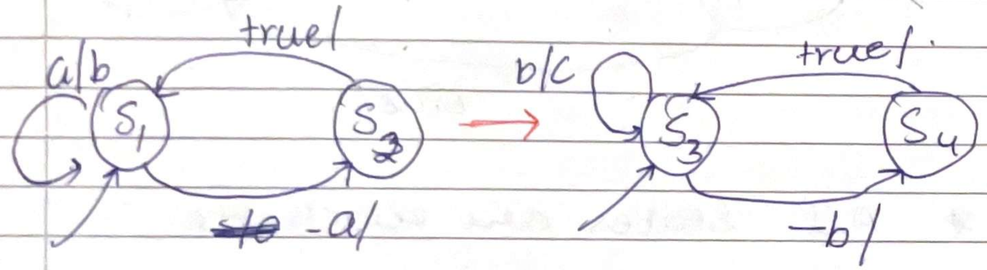
$(S_A(n+1), y_A(n)) = \text{update}_A(S_A(n), x_A(n))$

$(S_B(n+1), y_B(n)) = \text{update}_B(S_B(n), x_B(n))$

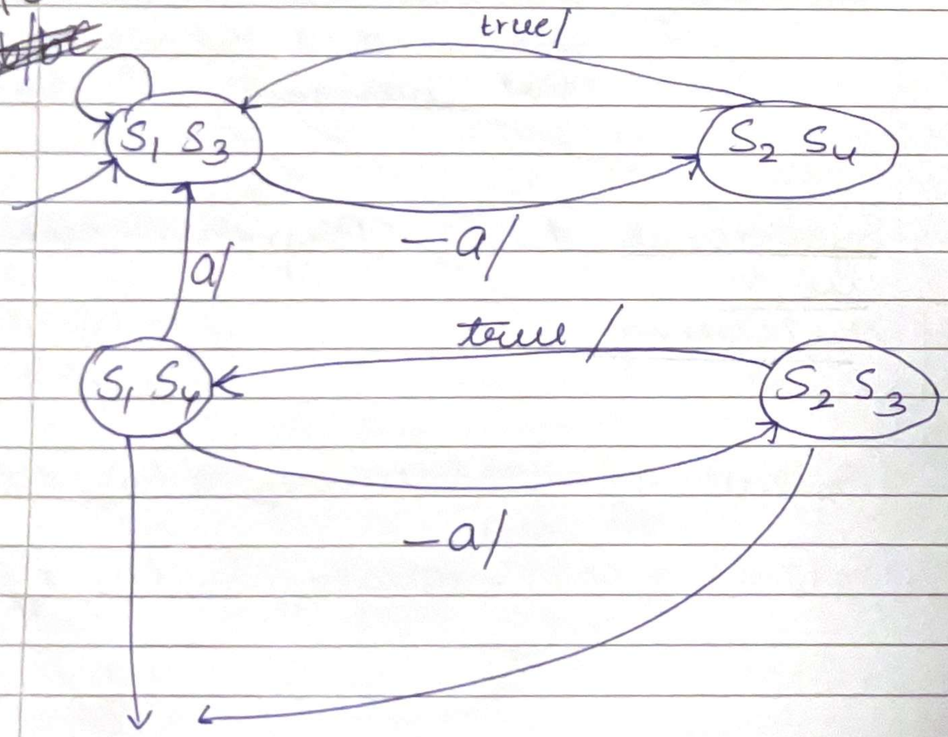
77. → Cascade



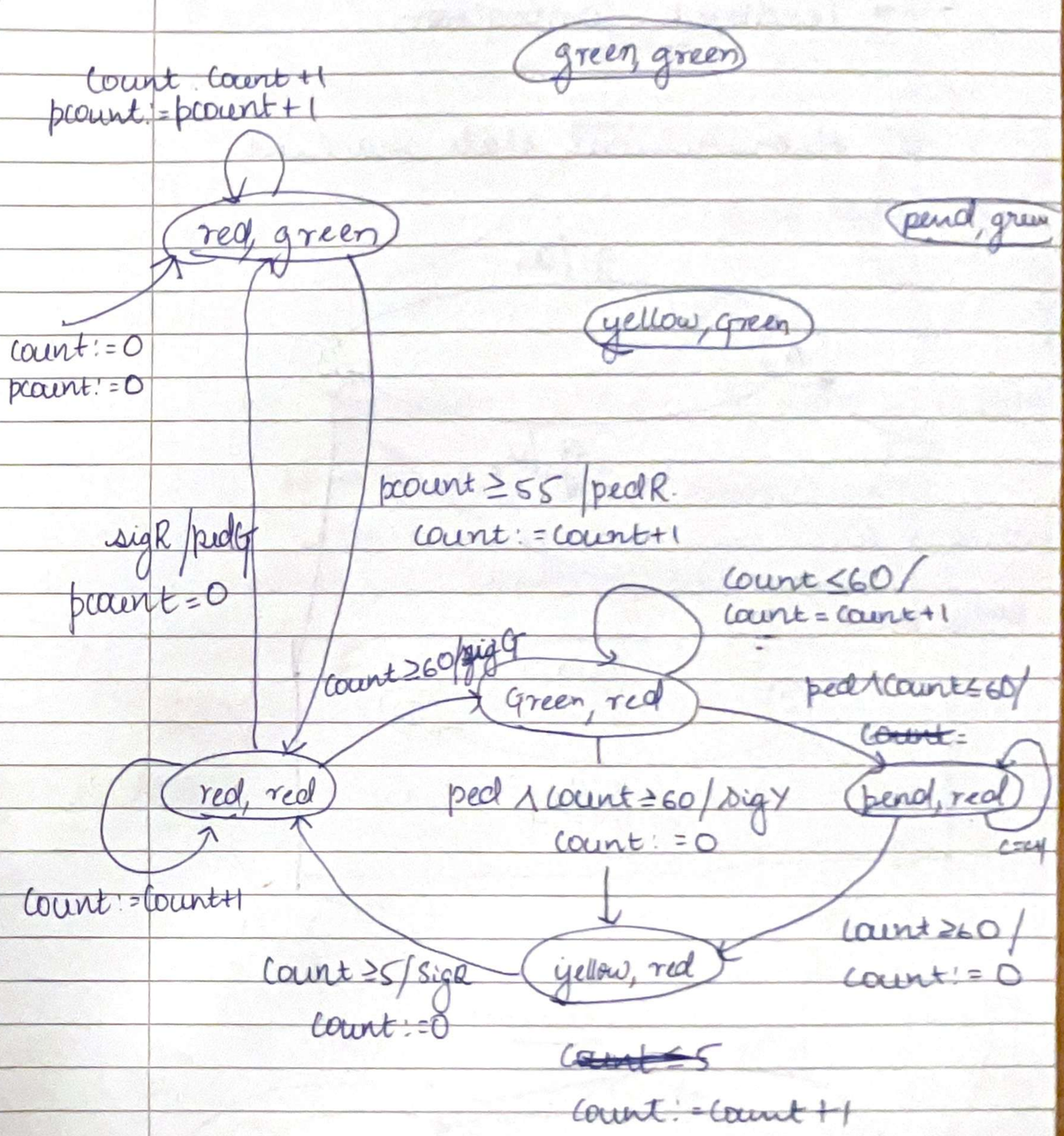
They react in the same cycle.



a/c



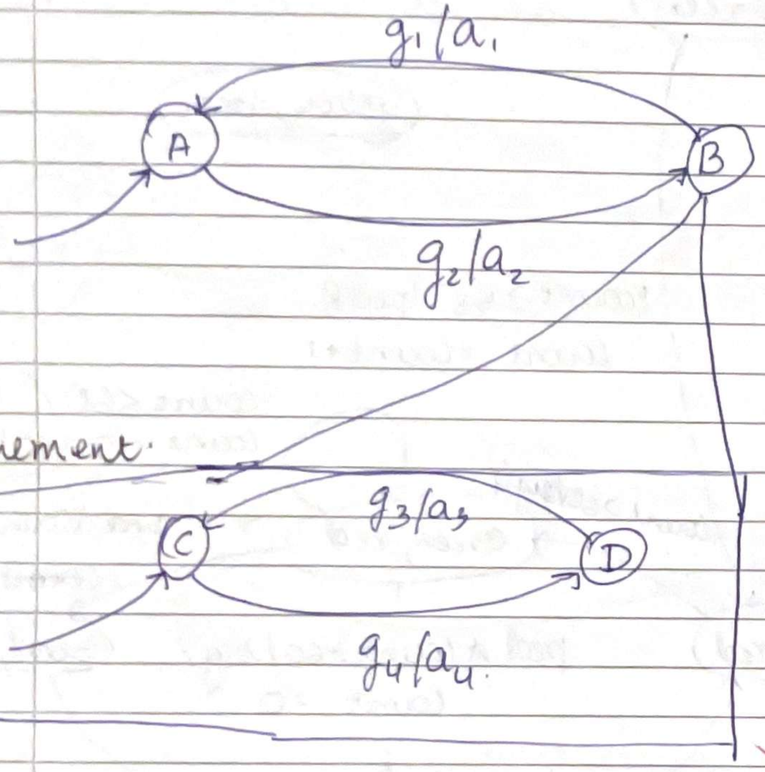
unreachable states



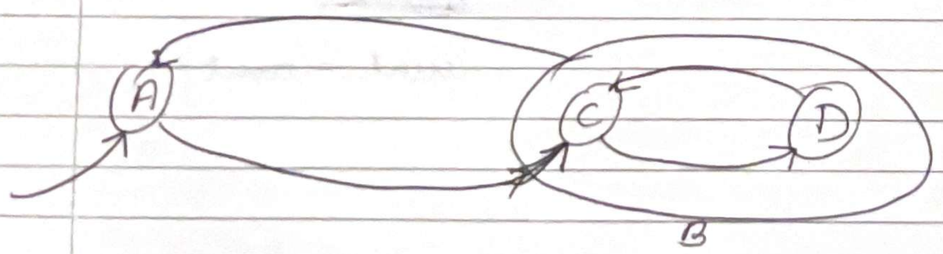
Volatile : do not optimise
reading & writing has side effect → allowed

→ Feedback composition

→ Hierarchical state machine.

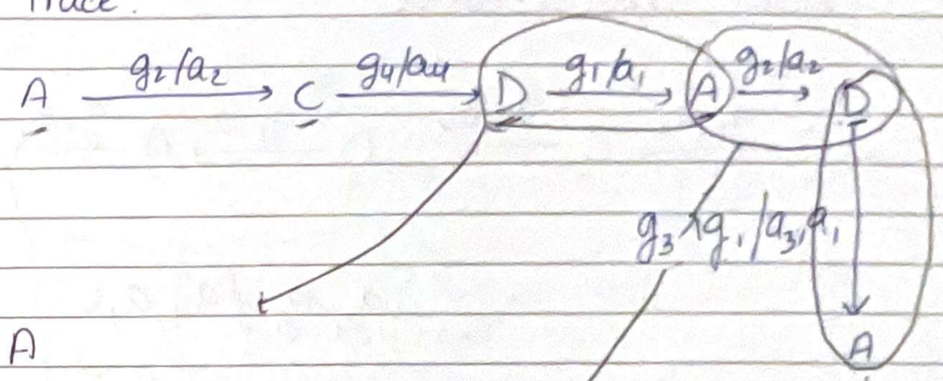


Visualise



Refinement : hierarchy.

Trace:



B → A

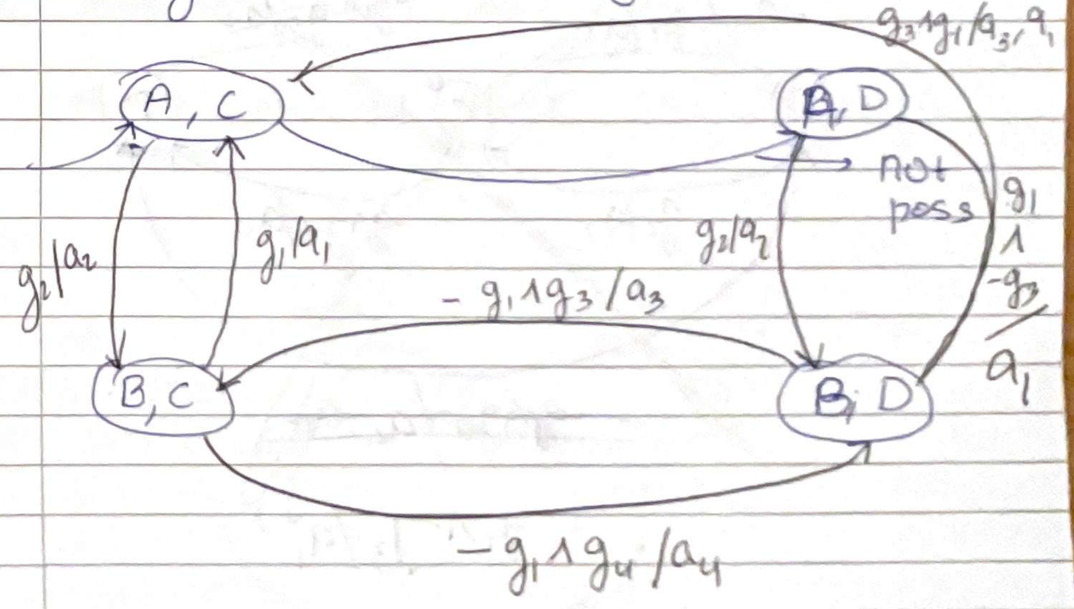
D was the last location, so went back to D

A → B (HISTORY TRANSITIONS)

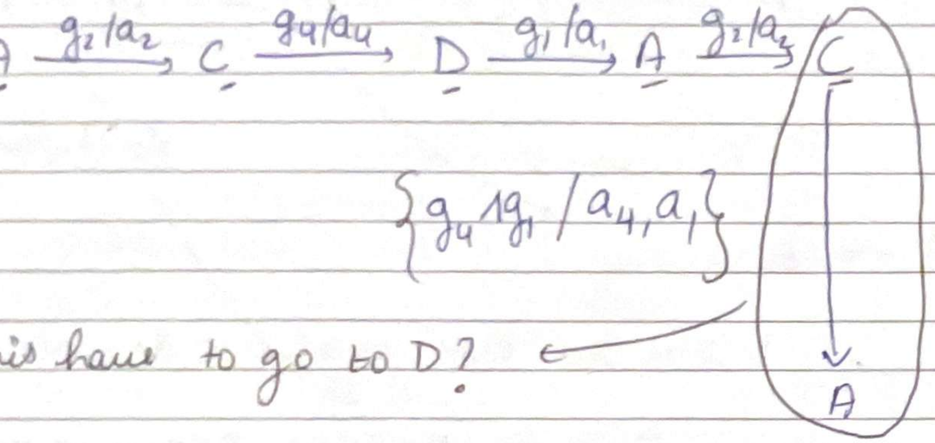
When state with refinement is left, it is essential to remember the state of refinement

~~A~~ D → C → A
(SIMULTANIOUS TRANSITION)

Flattening (with History transitions)

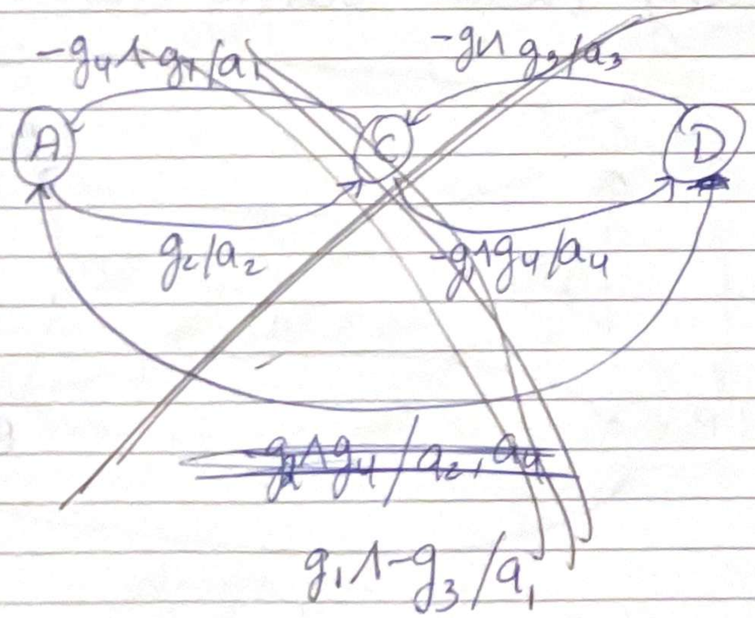


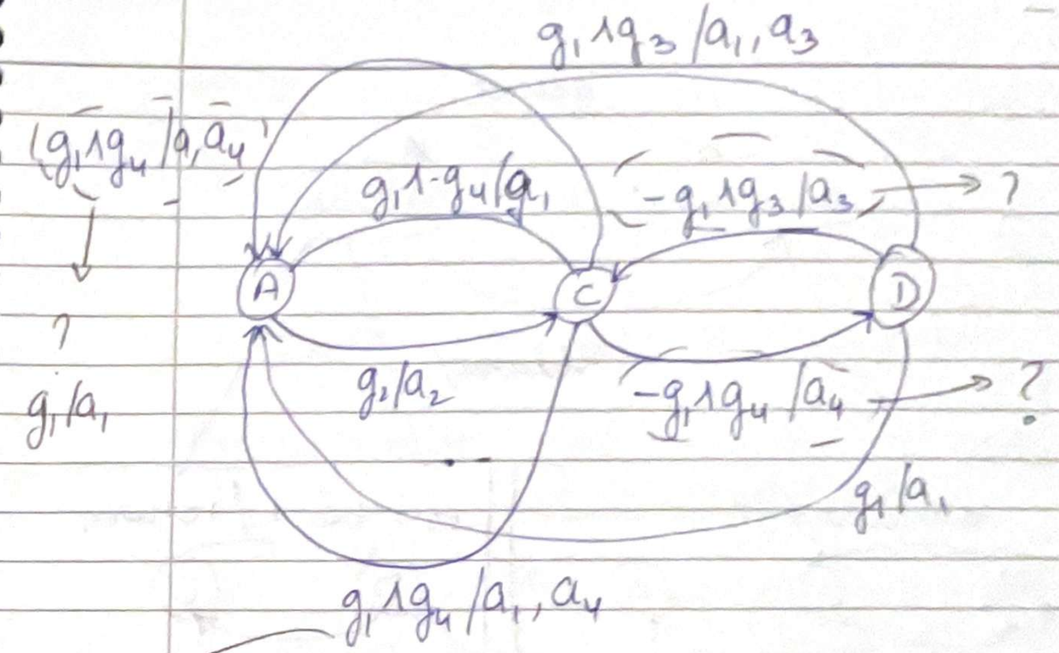
Reset transitions



does this have to go to D?

Reset: When refinment is left, you ~~leave~~ forget the refinment state





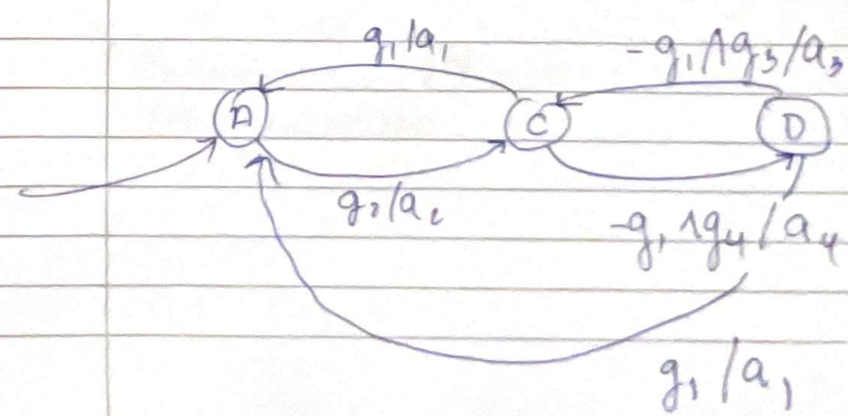
* fewer states as result is allowed

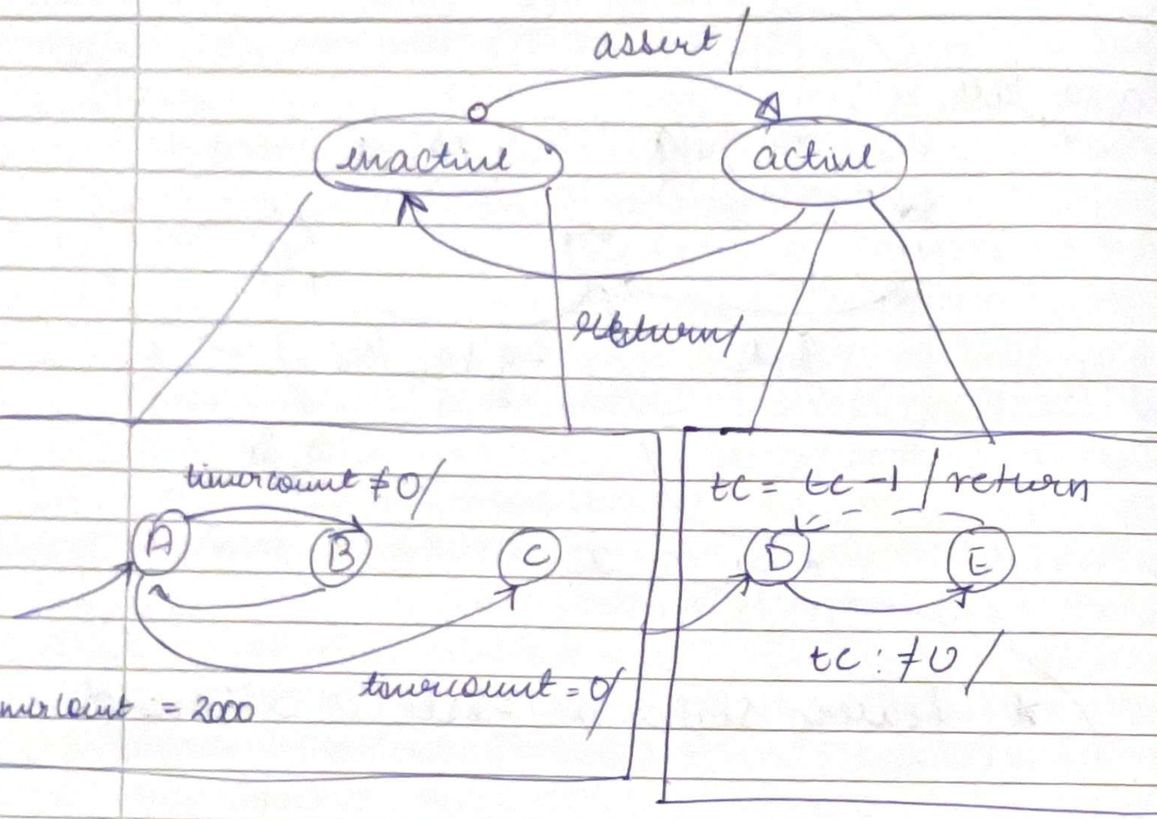
→ Preemptive transitions

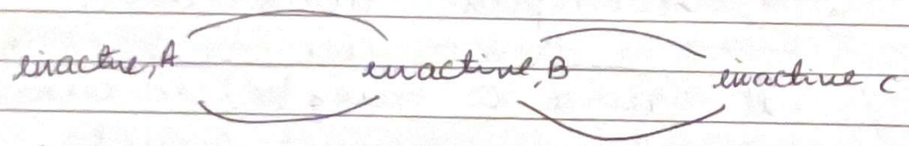
→ 2 jumps are not possible ← if gaurd is true before current state refinment reacts,

then current state should not react.

g1 1 g4 / a1, a4







A D B D C D

A E B E C E

→ Lecture 6

Model Based of CPS

1) Concurrent models of computation.

Concurrent: parts of a system operate at the same time

No order of computation

Semantics governed by 3 rules

* These are collectively called model of computation.

Rule 1: What constitutes a component?

Rule 2: Concurrency mechanism

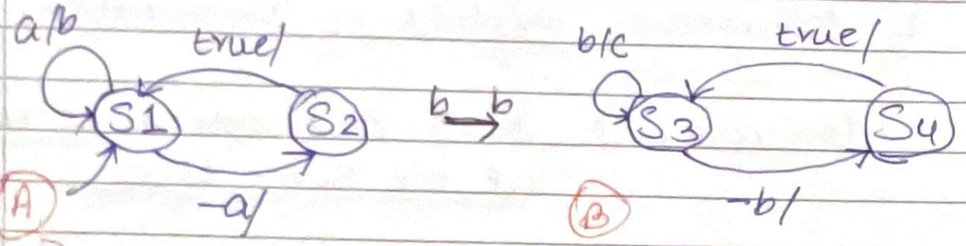
Rule 3: Communication mechanism.

(a) Synchronous model of computation

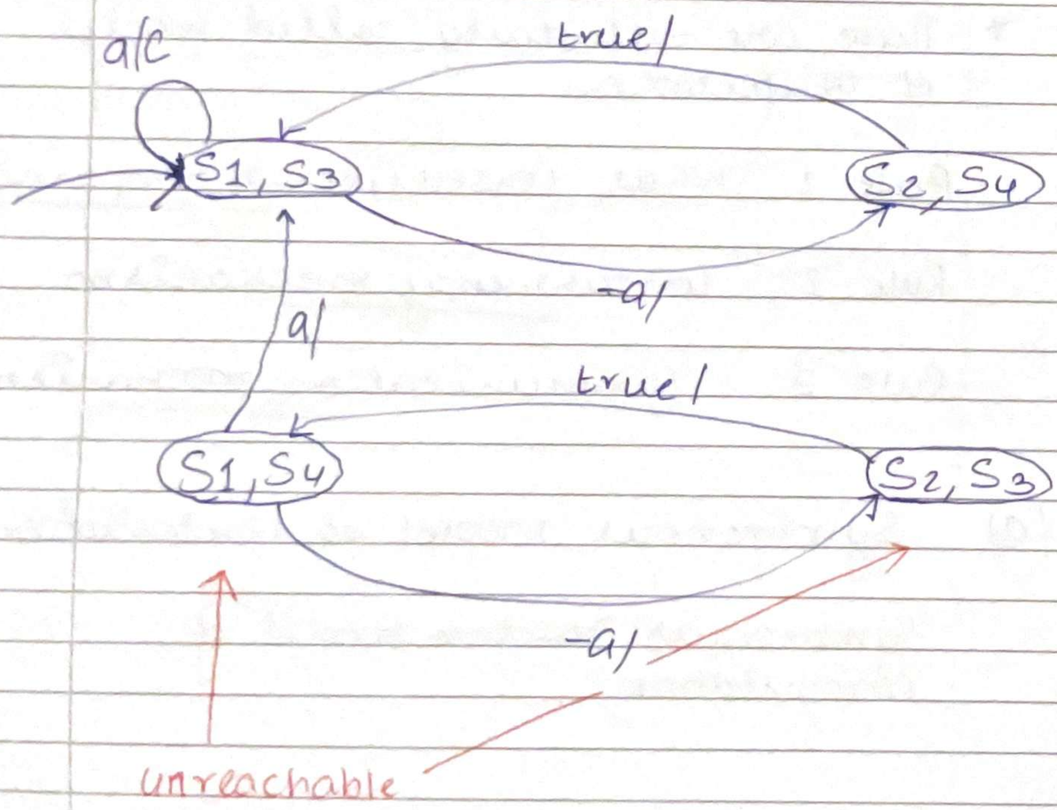
Synchronous reactive model of computation.

eg: Consider a cascade compositions (Synchronous)

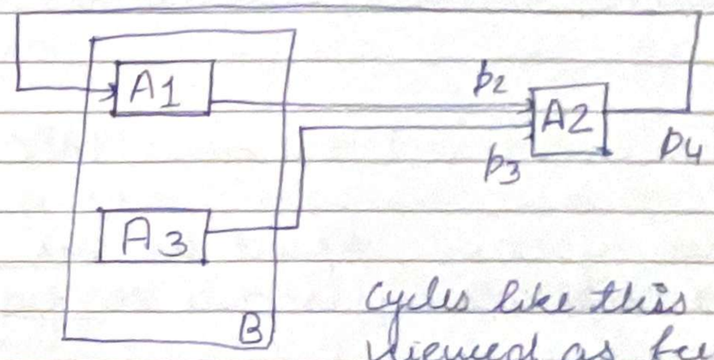
i/p : a : pure
o/p : b : pure



Don't think machine A reacts before B if it did, the unreachable states would not be unreachable.



Feedback composition



Cycles like this can be viewed as feedback composition

MOC

1) Synchronus Reactive MOC (Timed)

SR timed signals global clock

- Discreate System, where Signals absent all the time
- They become present at the tick of the global ~~clk~~ clock

→ execution of model → seq of global reaction

→ at each global reaction, actors react instantly and simultaniously

SR language & tools.

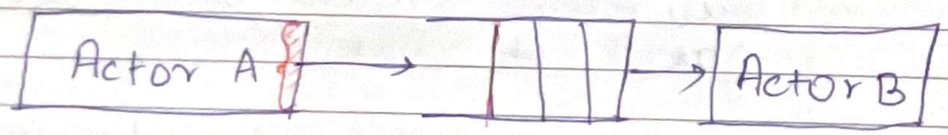
* Lustre, Signals, Statechart, Simulink.

(B) Synchronus Dataflow MOC.

* When reaction depend on one-another the dependency is due to flow of data

Rather than sychrony of events.

→ MoC, where data dependency are the key constraints on reactions is called a dataflow model of computation.



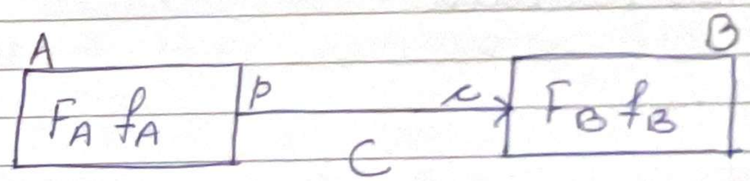
SDMoC

untimed dependence of data

An actor can fire whenever it has enough data (tokens) in its input buffers. It then produces some data on its outputs buffers

* no of tokens consumed = no of tok created

then, schedule the firing to get a useful execution.



q_A & q_B : no of firings for A & B

~~P_B & P_C~~ P_C & C_c : tok consumed & produced

Then system is balanced, if for all connections C.

$$q_A P_C = q_B C_c$$

$q_A P_c = q_B C_c$

(E) Timed model of computation.

- Similar to SR MoC since there is a clock.
- But computation takes time instead of being simultaneous.
- Simulink supports (timed model) of computation.

Discrete Event Systems.

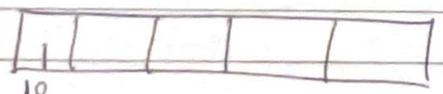
- Used to build simulations systems.

To execute DE

1) use an empty queue - a list of events sorted by time stamp.

used for complex systems!

event Queue



Discrete Event Simulations

* Describes a dynamic systems.

Continuous - time systems

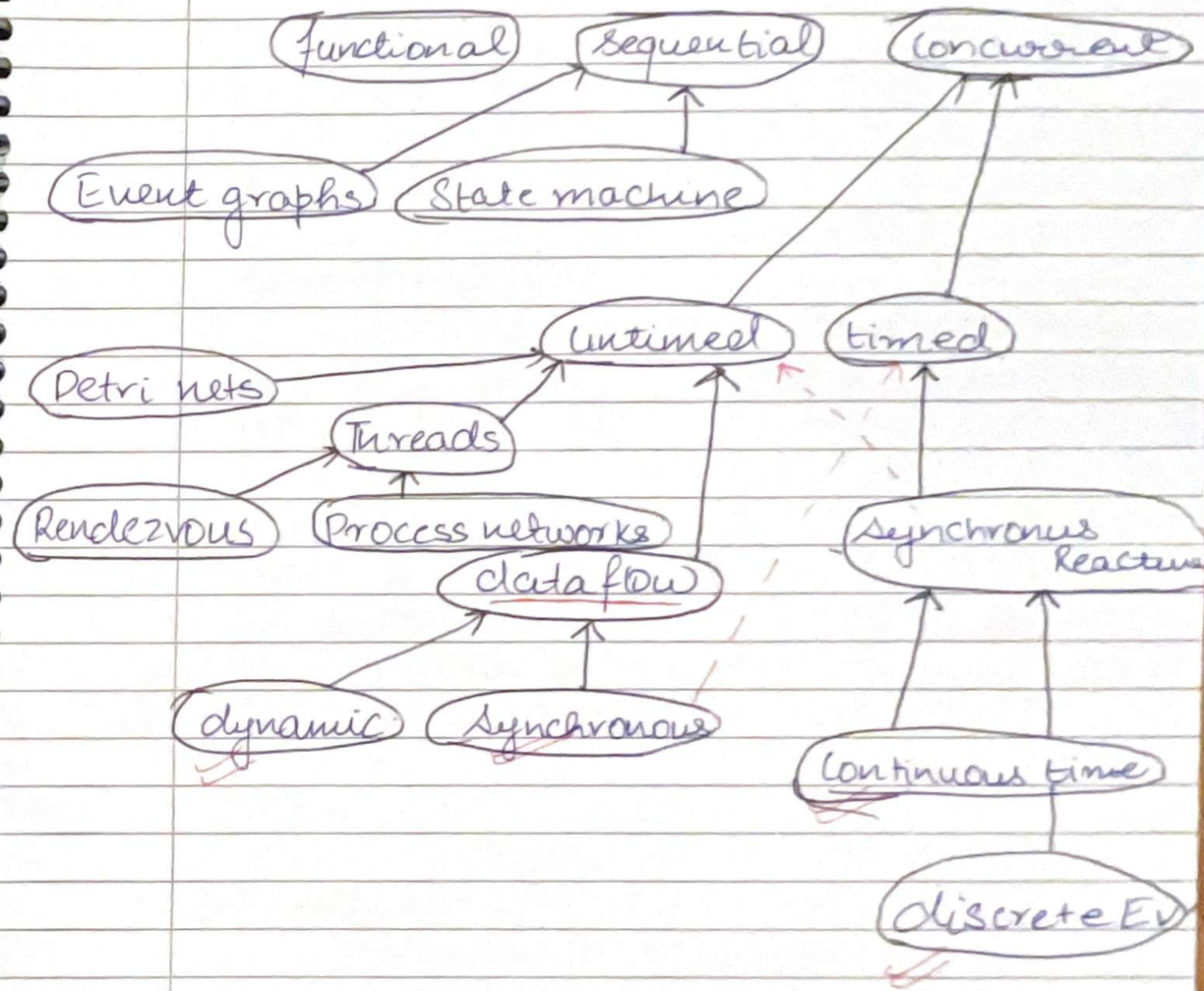
→ A model can be described by interconnection of actors

The communication between actors happens via continuous - time signal

1) Key diff b/w SR & Sync dataflow?

2) does SR not have dependencies, since it is also in cascade

Model of computations:



Embedded Hardware

* Timing, memory usage, power consumption and physical failures.

* Reasons for considering HW
  - Real - time
  - efficiency
  - security
  - realiability

* Sensors.
  - Processing physical data starts with Capturing of the data.

eg: Magnetometers.

(Hall effect)

Accelerometer.
- plate fixed to a platform & one attached by a spring and damper.

uses: navigation, orientation

$F_2(t) = -Cx(t)$ $\quad F_1(t) = K(p - x(t))$ $\quad M\ddot{x}(t)$

* Spring mass damper

- Newtons law $F = ma$.

- F could be graviational pull.
The force is balanced by restoring force of spring.

* $F_1(t) = K(p - x(t))$ — position of mass
spring rest position — force due to spring extension.

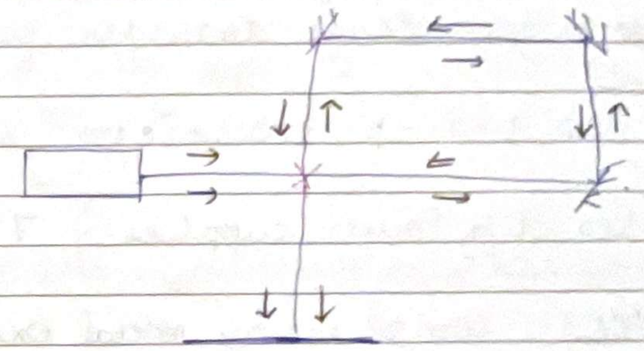
* $F_2(t) = -C\dot{x}(t)$
viscous dampeing.
VD const

→ disadvantage of accelerometers.

= seperating tilt from accelaration
= integrating twice to get position.
= vibrations.
= non-linearties. in spring or damper

Gyroscopes Sagnac effect.

* Optical gyros : Sagnac effect.



laser sent around, in loop, opp direction and the interference is calculated.

rotation of loop causes change in interference.

Inertial navigation System

* combination of.

1) GPS
2) 3-axis gyroscope → for orientation.
3) 3-axis accelorameter — for positon after correction for orientation

Typical drift

0.6 nautical miles per/hr
tenths of digree per/hr. } depends on application

Image Sensor

* Charge Coupled Devices (CCD)

→ Works on charge transfer to next pixel cell

→ Use p-n-p transistor

→ Req diff Power supplies { 7V to 10V }

CMOS - Complimentry metal oxide semiconduct

photosite - pixels.

* Charge to voltage conversion is carried out in the pixel it self

* Requires single supply voltage
  Typical voltage : 3.3V to 5V.

CCD consumes more power than CMOS. because CCD compute charge to voltage conversion pixel by pixel and sent through the vertical shift register.

needs more ports timers, ADC

Diff ~~=~~ power for diff timing clock

Signals

→ It is a mapping

from time domain to a value domain

$S : D_T \rightarrow D_V$

Continuous or discreate time domain

Continuous or discreate value domain

Discretization of time

* digital computers require discrete sequences of ~~time~~ physical value

→ Sampling

$$e'_k(t) = \sum_{k=1,3,5\ldots}^{k} \frac{4}{\pi k} \sin\left(\frac{2\pi t}{41k}\right)$$

with higher K - Value, the more accurate it gets to being a square wave.

Linear transformation

* A transformation T_r of signals is linear iff

$$T_r(c_1 + c_2) = T_r(c_1) + T_r(c_2)$$

→ Aliasing

The misidentification of signal frequency introducing distortion or error.

- Reconstruction is impossible.

How frequently do you need to sample

Nyquist Criterion

aliasing can be avoided if we restrict sampling freq to less than half of the sampling rate.

$$f_s > 2f_N \quad \text{or} \quad P_S < \tfrac{1}{2} P_N$$

f_N → Nyquist freq

anti aliasing filter is needed to remove high frequencies

→ Resolution

* Resolution (in bits): number of bits produced

Resolution (in volts): difference between 2 input voltages causing the output voltage to be incremented by 1

$$Q = \frac{V_{FSR}}{n}$$

V_{FSR} → diff b/w largest and smallest val

n → no of voltage intervals.

$$Q = \frac{V}{n}$$

→ Quantisation Noise

It is the effect of representing an analog (continuous signal with a discrete number (digital signal)

The rounding error is referred to as quantisation noise

Signal to noise ratio.

$$SNR_{db} = 20 \log_{10} \frac{\text{effective sig Vol}}{\text{effective noise Vol}}$$

↑ SNR is better

Hardware, Processing, Actuators.

* $E = \int P(t)\,dt$.

Power consumption

1) designing power supply
2) diminishing of interconnect.

→ Hardware, Processor, Actuating

→ energy or power, the more important

$E = \int P(t)\,dt$ $\quad E = \int P(t)dt$

Minimising power

1) designing power supply
2) Short term cooling

Minimising energy

1) restricted availability of energy
2) cooling
3) thermal effects

* Power density continues to get worse
* Problem: Increasing performance requirements of mobile phones

30% power to RF modem & amplifier
20% application processor
20% memories.
10% colour display and backlighting

Power distribution in portable systems

- Graphics
- Media
- Radio
- Application

→ Static and dynamic power consumption

static power consumption is caused by leakage current

dynamic power consumption caused by charging capacitors, when logic levels are switched

$$P = \alpha C_L V_{dd}^2 f$$

- α : Switching activity
- C_L : load capacitance
- V_{dd} : supply voltage
- f : clock freq

leakage current becoming more imp due to smaller devices.

→ How to make systems more energy eff

Power consumption of CMOS circuits $\{P = \alpha C_L V_{dd}^2 f\}$ reducing V_{dd} reduces P ~~dramatically~~ Quadratically.

Delay for CMOS circuits

$$\tau = k C_L \frac{V_{dd}}{(V_{dd} - V_t)^2}$$

V_t : threshold voltage

$V_t < V_{dd}$

Basic equations

- → Power : $P \sim V_{dd}^2$
- → Max clock freq : $f \sim V_{dd}$
- → ~~ee~~ energy to run a prg : $E = P \times t$
- → Time to run a prg : $t \sim 1/f$

Why multicore.

* Acc to Pollack's rule

2X Power = (1.4X) Performance.

* multicore - potential ~~=~~ near-linear performance speed up.

→ Dual core.

Vol = 15% less
Freq = 15% less.
Perf = ~ 18

→ Multi

* power efficient
* Better power & thermal managment

Fine - grain powermanagment.

Critical task - core.
Freq = f @ Vdd.
TPT = 1
Power = 1

non - critical

freq = f/2 @ 0.7Vdd.
TPT = 0.5 Power = 0.25.

→ ASIC - Application Specific circuits

* less flexibility
* ~~less~~ high design time
* higher cost
* need to be sold in ↑ Quantity
* made for high speed or high energy saving

eg: VIP for car mirrors.
Video platform.
Telephony W-CDMA

→ Dark silicon.

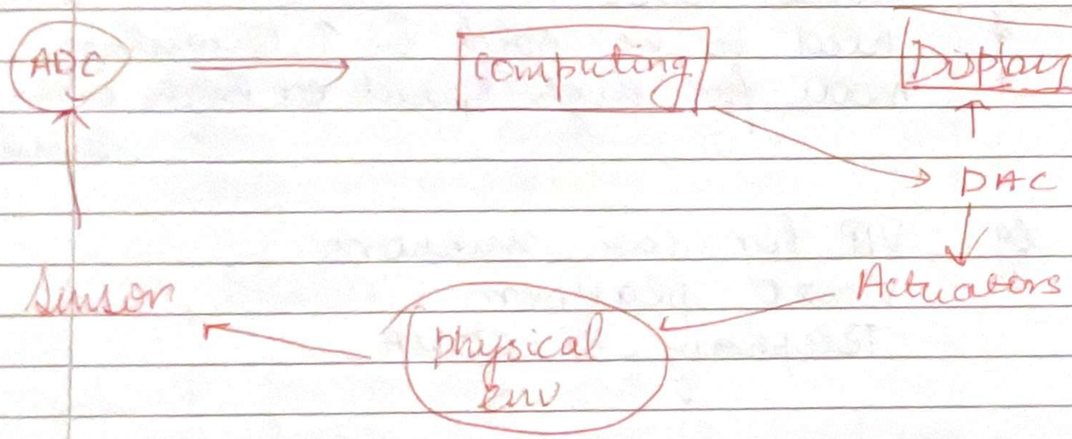
* Not all silicon can be powered at the same time, due to current, power or temperature constraints
* Can not use all of them at high speeds

Dynamic power managment

Run: operational

IDLE: a SW routin may stop CPU when not in use.

SLEEP: shutdown of chip activity.



→ Multitasking, OS, kernels, scheduling

Multitasking

* layers of abstraction

— Concurrent model of comp.

dataflow, Synchronous

Multitasking

threads, message passing,

Processor

ISR, pipe, core

* abstraction: hide details and provide platform above.

OS and Microkernels

Microkernel: A small, custom OS

uses:

- scheduling of threads
  - creating, time of activation
- Synchronisation
- input and output
- Memory managment
- file system
- networking
- Security

A unit of execution within a process

* Threads are sequential procedure that share memory
* Process are imperative programs with their own memory space.
  ↳ OR program in execution

Uses of Concurrency:

* Reacting to external events. (ISR)
* exception handling (software interrupts)
* Simultaneous running program. (multitasking).

* without OS, multithreading is achieved via interrupts

* Pthread — API.
  ↳ Real time & not.
  ↳ lib of C procedures.

* Threads may or may not run when created.
* can be suspended
* can have priority

→ Thread Scheduling

* A scheduling decision is made based on
  1) Assignment → What processor ✓
  2) Order → What order ✓
  3) Timing → What time ✓
* decision maybe at design time or run time.

- fully static scheduler
  ↳ at design time
- static order
  ↳ offline, gets order before exe.
- static assignment
  assignment at design
  scheduling at run time.
- fully dynamic scheduler

Threads are widely non-deterministic

⇒ Algorithm

1) Rate monotonic scheduling

- shorter period → high priority
  larger " → less "

→ Periods have deadlines.
→ process needs to be completed before that
→ deadline can be missed

2) Earliest deadline first
(Non-Repeating tasks)

→ 2 tasks with same deadline, the order does not matter.

→ minimises maximum lateness
{compared to all other ordering}

→ Priority for deadline

early DL → High

→ Data Driven Modeling

* find relationship between system's input and output
* NO explicit knowledge of behaviour
* ML & statistical model represent ODM

classification & Regression

↳ distinct
eg gender

↳ continuous
eg Battery

Types of predictions

Supervised → labelled data ✓
target value ✓
prediction close to target

unsupervised → NO idea of target values
nothing to predict
reward → explaining features

Semi supervised learning →
some data - labelled
data have unknown target value

_/_/_

* Bayes theorem → Naive

$$P(C/x) = \frac{P(x/C) \cdot P(C)}{P(x)}$$

likelihood

feature/attribute

* used in text classification or with multiple classes.

Markov Model.

1st order: Prob → to a state, given a state

2nd order.

Prob of n, depends on $n-1$ & $n-2$.

$$P(w_n/w_{n-1}, w_{n-2}, \dots, w_1) \approx P(w_n/w_{n-1})$$

Tree modeling

* Supervised learning.

_/_/_

Multivariate Linear Regression

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \quad + \theta_n x_n$$

_/_/_

→ Decision tree → ID 3 Algorithm

* Supervised learning → used for classification

* split the population into 2 or more sets

Advantages

1) easy to understand

2) useful in data exploration

3) less data cleaning required

4) Data type is not a constraint.

→ Random forest.

* Both Regg & classification

→ Support Vector Machine classifications

* It creates a margin, rather than a single line.

*

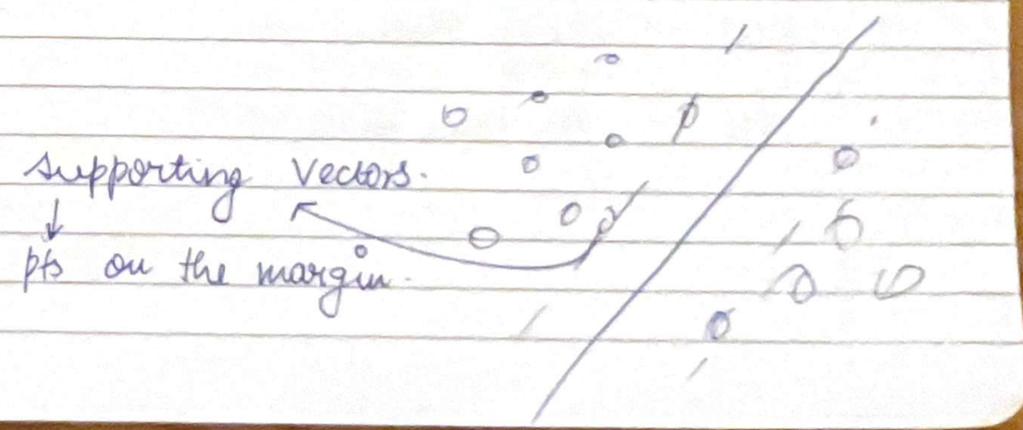


Image classification

Ip: N images, k different classes (Training)

Learning: To learn about each class
learning a model.

Evaluation: Quality check by prediction

Artificial Neural Network

* Ability to handle large no of features
* more prediction power
* high fault tolerance
* more weights to more desireble feature.

→ CNN → Images → due to more features
→ KNN → speech, Machine → audio
→ NN → Act, Hand feature

MEA