Petri Nets Notes

* Difference b/w petrinets & finite state machine is that petri nets allow multiple independant activities in progress at same time (There is always 'current state' that determines next state in state machines)

Structure: Places, Transitions, Edges & Tokens

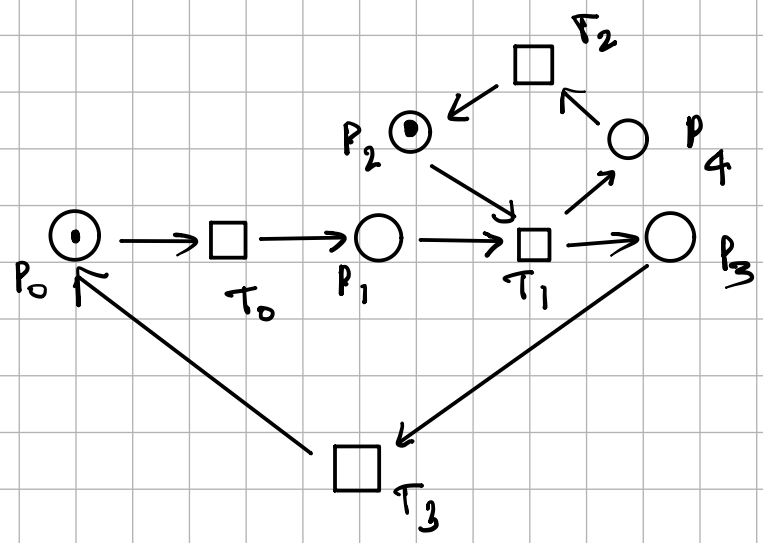
○ <state>

□ <like a button when pressed transitions from one place to another>

→ <directed arrows>

● <represents current state>

Eg:



⇒ Rules for a petrinet:

* The state of a petrinet is represented by tokens at various places. (in eg. P_0, P_2 has tokens)

* There cannot be an edge b/w 2 transitions or places - It is always p - t - p

* firing → a token from one place goes to another via firing of a transition

$P_0 \rightarrow P_1$ when T_0 is fired

○ ⊙ □

→ rules for firing: there should be at least one token in each of the transition's input

during firing one token from each of the inputs and a single token to each of its outputs.

* petrinets can have subsystems (loops in a system)

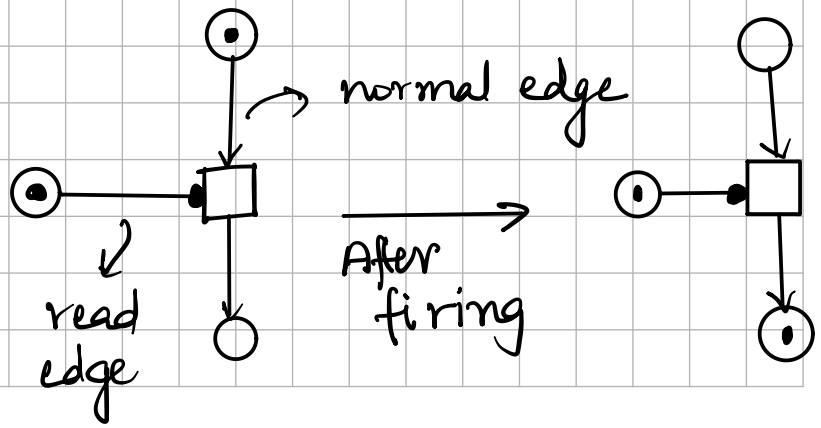
(Note: token from one subsystem goes around in its subsystem only doesn't go out of the loop)

↳ Like in fig. token in P2 can go to P_4 only, not P_3

* During firing of T_1, both P_1 & P_2 have tokens so, after firing $P_1 \rightarrow P_3$, $P_2 \rightarrow P_4$ any one can occur first and next so the next state after firing T_1 is not determined ⇒ This is called a 'non-deterministic' behaviour

⇒ Edge Types :

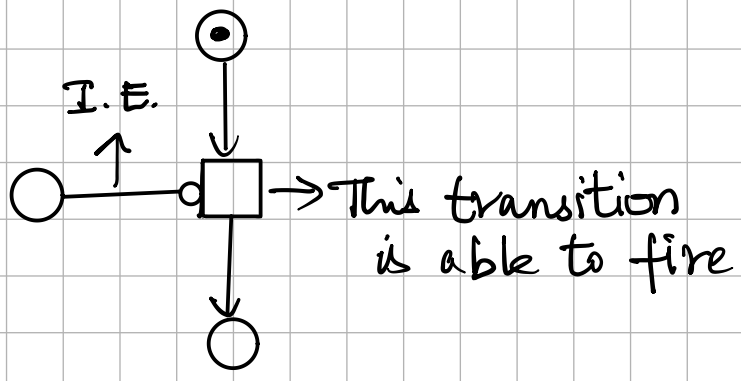
① Read edges:



* Read edges just 'reads' if there is token and doesn't take the token and produce in output

② Inhibitor Edges :

If token is 'NOT' there then ONLY the transition can occur.

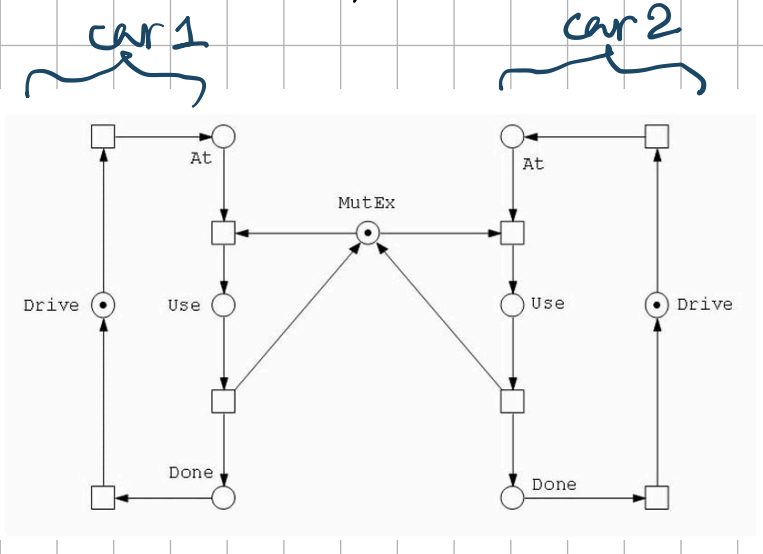


* Concept of Mutual Exclusion :

Sometimes we want a system where activities must be mutually exclude each other

In example of 2 cars at an intersection only one car can cross the intersection at a time, else system faces problems - here accidents

This is modelled here,



There is also 'fairness' in this, either cars can go first, not that car 1 always goes first in this model.

When the token is 'at intersection' of car 1, if token fires it removes the token from 'MutEx' making car 2 not cross the intersection

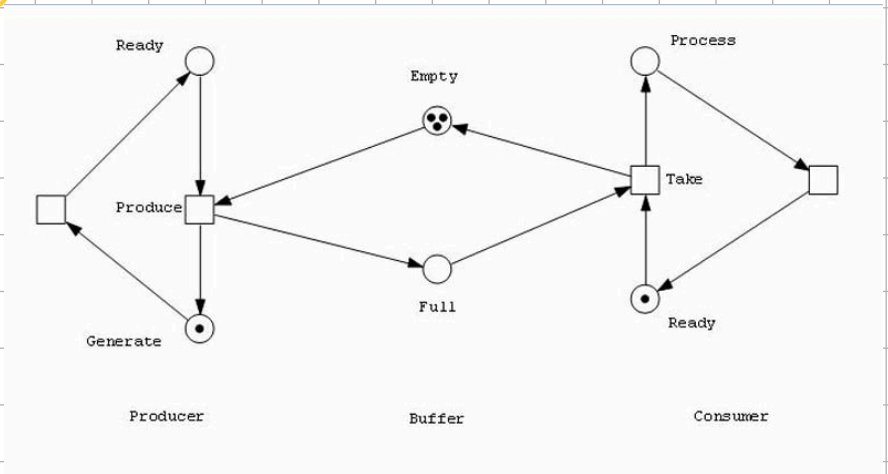
* Concept of Resource allocation :

There can also be multiple tokens in a place at a time, these can be seen in system where is multiple units of resource to allocate

Example is a consumer-producer system → here the total no. of available units can be limited

The producer can produce units as long as the 'Empty' place is not having token.

(Overflow is managed)



The consumer can take units as long as there is at least one token in 'Full' place

(Underflow is managed)