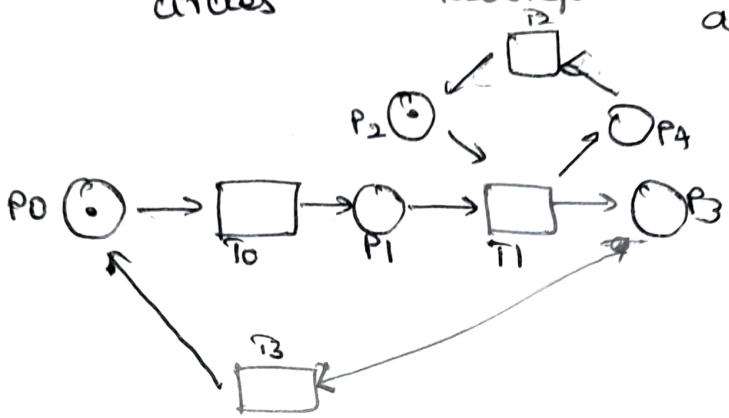
Petri-nets

* It consists of four elements:

(i) Places (ii) Transitions (iii) Edges (iv) Tokens

↓ circles ↓ Rectangles ↓ Directed arrows ↓ Solid circles

Eg:



4 places: P0 → P4 3 transitions: T0 → T3. Edges connect places to transitions and vice versa

P0 and P2 have a single token represented by dot

⇒ Petri net may contain cycles. Here it has 2 cycles.

(P0, T0, P1, T1, P3, T3) and (P2, T1, P4, T2)

*Tokens represent the state of the petri nets. There are rules by which petri nets change its state.

"A petri net" changes from one state to the next state when a transition fires. We define input, output for edges in the petri-net. (In(T0) = P0, Out(T0) = P1 Out(T1) = P4, P3)

Firing rules for a transition:

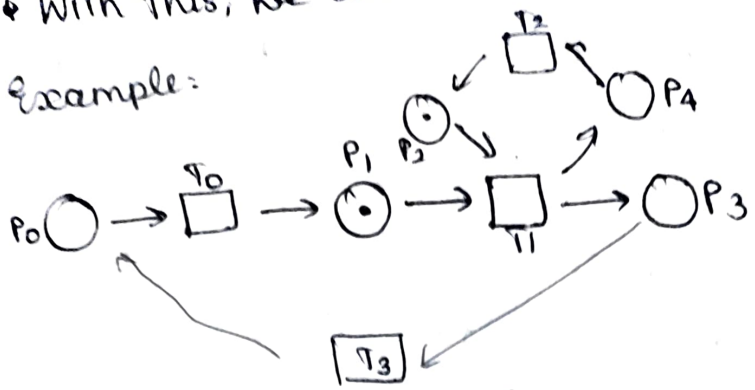
* A transition is able to fire when there is atleast one token on each of the inputs of transition

* When a transition happens, it removes one token from input and would place it on output

. Every transition will get active at a point. If it is not eligible we will disable it.

• With this, we can model non-deterministic behaviour.

Example:



(Next state of previous example

(T_1 could not do as P_1 did not have token)

• From this state, we can fire T_1 (as both P_1, P_2 has tokens)

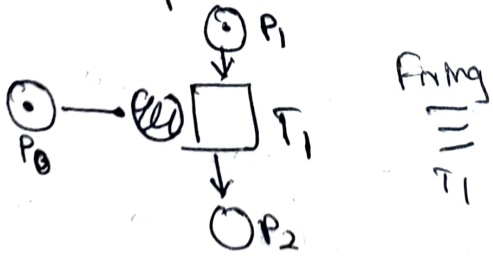
→ Once you have two transition states ready to fire, you can go in any order. After this state, we will reach the first state or second state. (T_2 after T_3) (T_3 after T_2), (T_3 after T_0 after T_2)

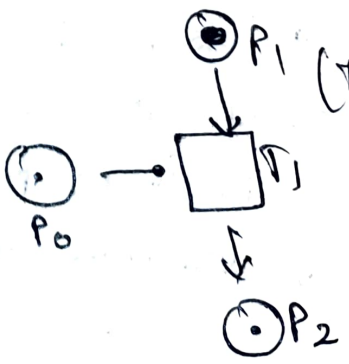
Extended edge types:

• We will see the extensions to the petri net structure, two of it are read edges, inhibitor edge

* Read edges (arrow) can be drawn only from place to transitions unlike the normal one.

→ Read edges won't remove the token from where it reads represented by →



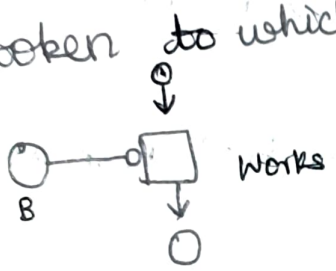


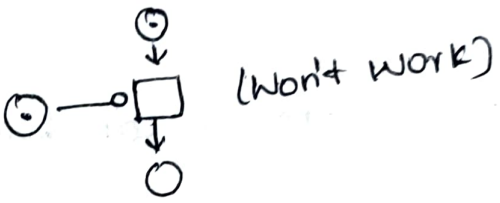
(Token is removed from place but not from place)

Inhibitor edge: (Opposite of normal (read edges)

* The logical operation is and (presence of token) in all the input places of a transition.

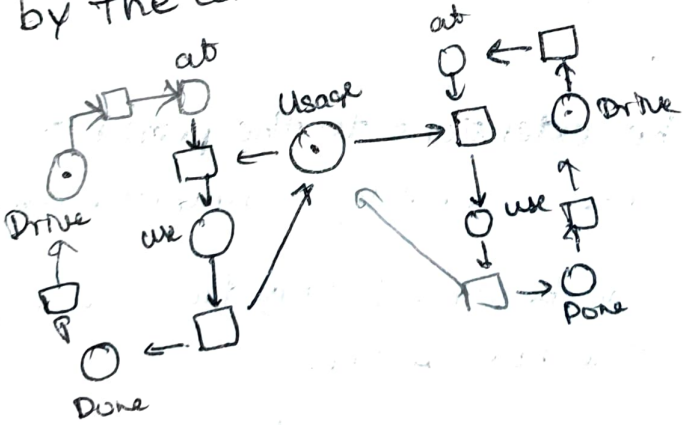
* In inhibitor edges, the not logic is used i.e presence of a token inhibits the firing of the token to which it is connected. (—o)





Eg: Mutual exclusion:

* Two concurrent process cannot take place simultaneously. So we declare two petri nets connected by the conditional statement.



Resource Allocation:

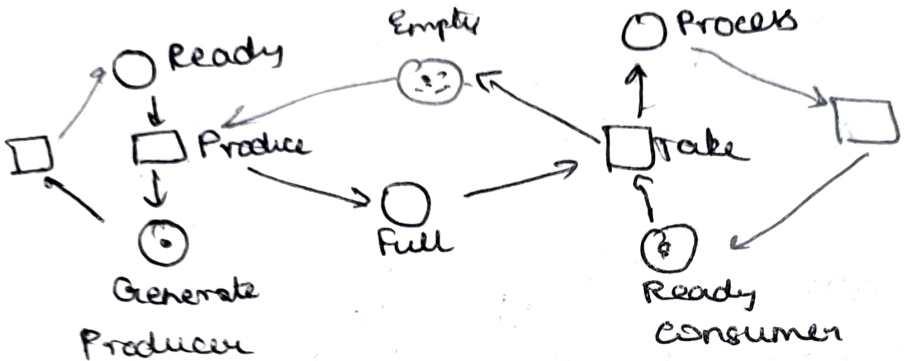
* It is possible that a single place may contain multiple tokens at one time.

→ In these kinds of problems, place represents the number of available units of the resource.

* If there is no token then you need to wait till you get to have a unit in the place.

* The number of units in a system can be variable

Eg: Classical consumer-producer problem



When it is empty, producer will work

When it is full, consum
will start to work