Lecture 29
Chapter # 8 (contd.)
PETRI NET-BASED PERFORMANCE MODELING

Mutual Exclusion (Conflict)

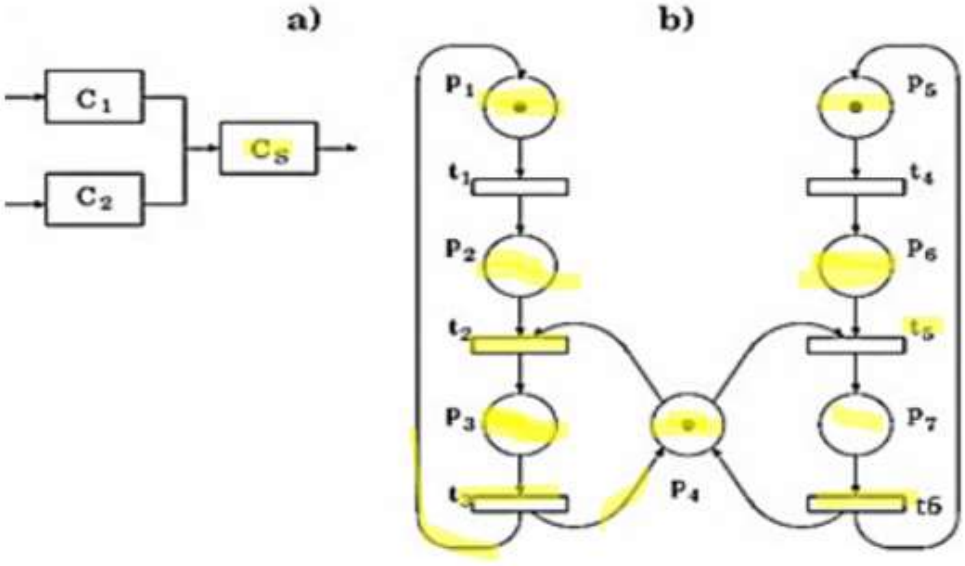


Fig 16: The Mutual Exclusion Problem

Fig a here represents two parallel processes & they both are executing but they may ask or request for a critical section or a critical resource but only one process should enter in the critical section & while he is performing some operation at that particular time the other process should be block & once the first process has completed the execution in the critical section the block process or the waiting process should enter in the critical session/region immediately.
Fig b represents this situation by means of a petri net;
P_1 & P_5 is representing the working of C_1 & C_2, P_1 & P_5 are actually showing that the two processes C_1 & C_2 are right now working.
Here P_2 & P_6 represents that C_1 & C_2 may request access to critical section.
Place P_4 with the single token determines the availability of resource C_s i.e. the critical section.
P_4 actually prevents P_3 & P_7 to be marked at the same time. P_3 represents that the critical section is busy with C_1 & P_7 represents that the critical section is busy with C_2.
Suppose C_1 wants critical section; C_1 is executing, when it is executing normally it will remain in P_1 but when it wants to use critical section; it has got one token & transition t_1 is being enabled & the P_2 will represent that it's now asking for critical section & when it will be moved to P_3 it means that is busy with the critical section but it is only possible when there's a token in P_4. So, single token in P_4 represents that only one process can enter the critical section at any one time. Transition t_2 can only be enabled when there is a token in both P_2 & P_4. & when t_2 is enable the C_1 will enter critical section being represented at P_3.
In the meantime if C_2 wants to enter in the critical section it will go to place P_6 but in order to enable t_5 we need tokens in both P_4 & P_6 which is not possible right now it will only be possible once the process will exit from the critical section.& when it will exit it will again go to its normal operation plus it will deposit a token in t_4 again.
So, C_2 will wait at P_6.
The firing of t_3 & t_6 models the release of the common critical resource.

Logical Conditions

• It is often desirable to *model logical conditions.*
• e.g., to only fire a transition when there are more than *n* tokens in a place.

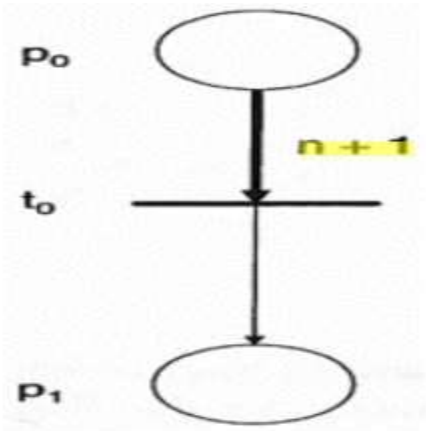
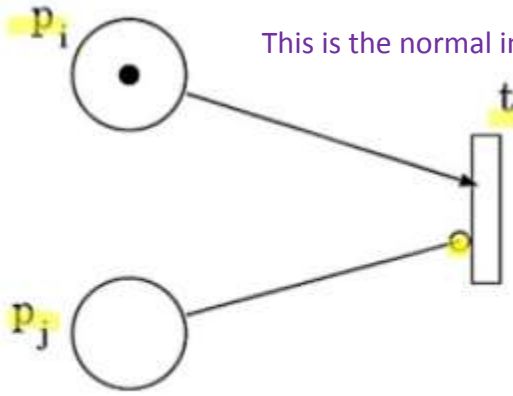


Fig 17: Petri net component to test condition greater than M

It means that transition t_0 will only be enabled when there are more than n tokens in P_0.
Let's suppose P_0 consist of n-1 tokens in this particular situation transition will not be enabled or it won't be able to fire. It can only fire when there are either n+1 or more than that tokens.
So, we can use this petri net component to test some conditions like something>m then only transition will fire etc.

Inhibitor Arcs

• The inhibition function usually represented by *circle-headed arcs*.
• Modifies the enabling rules so that the transition fires only if p_j does not contain tokens.



This is the normal input function arc.

This is the inhibitor arc because it is represented with circle head. By means of inhibitor arc we can modify the firing rules or transition enabling rules.
If we had this inhibitor arc & there is no token in that particular place (P_j) only at that particular time the transition can fire.

Fig 18: Inhibitor Arc

• To test for the condition of equal to some value but not greater than the value, we can use an inhibitor of arity n + 1 to block a transition if there are more than n tokens in place 0.

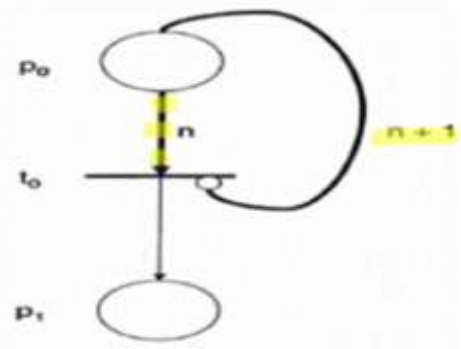


Fig 19: Petri net component to test condition equal but not greater than M

Suppose if there are n tokens then transition can fire but I want to make sure that the value is not greater than n; so, for that I am using an inhibitor arc & I am labeling it as n+1. I will actually block a transition if there are more than n tokens in place 0 & transition can fire if there are exactly n tokens at P_0. It means that inhibitor arcs are also useful for blocking various transitions.

• If we wish to test for *less than n* items and remove the items, we could use the Petri net shown in Fig 21.

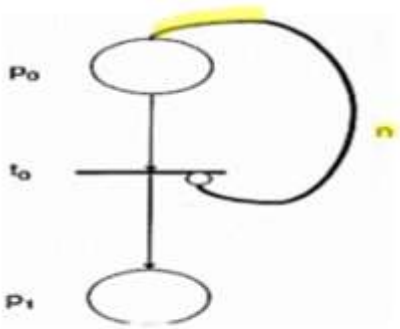


Fig 20

Modeling *conflict* and concurrency

➢ An initial marking, μ = (1,1,0,0,0), results in transitions t_1 and t_2 being enabled, the condition of concurrent transitions.

➢ If t1 fires first, then we have two transitions enabled, t_2 and t_3. This then depicts a *conflict*.

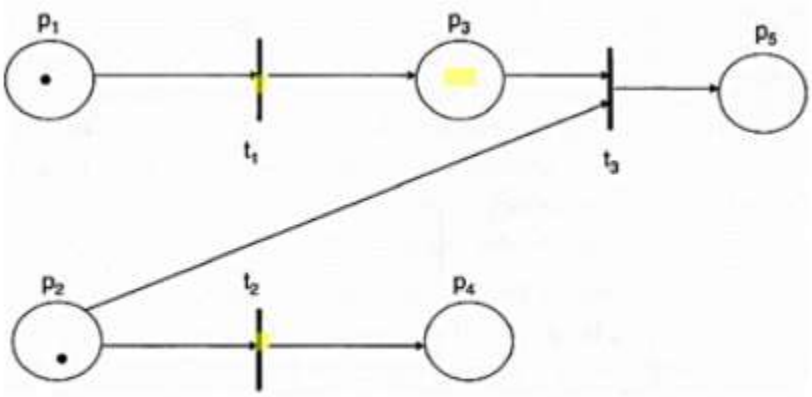


Fig 21: Petri net modeling *conflict* and *concurrency*

Reachability in Petri-Nets

• A Petri net state, μ, reachable from another state, μ', if there is an integer number of intermediate steps from μ' to μ.

• e.g., μ_0 = (3, 0, 0, 0), and a target state μ' = (1,0,0,1).

• We can reach this target state in three firings of our net.

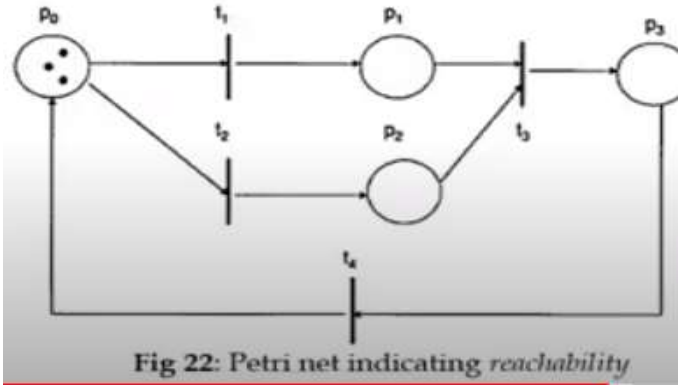


Fig 22: Petri net indicating *reachability*

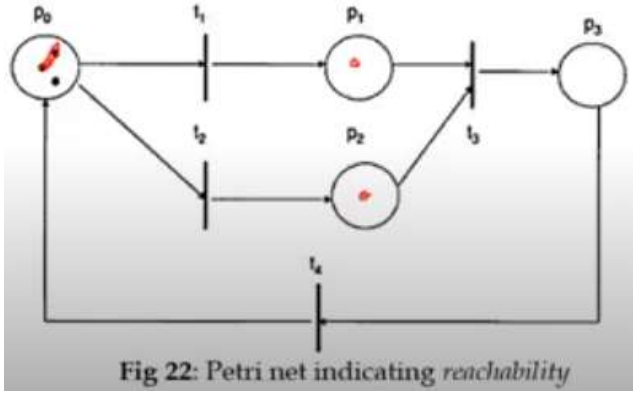


Fig 22: Petri net indicating *reachability*

The transitions t_1 & t_2 fired. P_1 & P_2 both got a token; now this transition t_3 is crucial cause in order to enable t_3 we need one token each from P_1 & P_2. Since it has a single output function to P3 only a single token can be placed over here. It means that one token will be absorb by t_3 & only one will stay in P_3 because it is not a multi-arc function it is just a single arc input function.

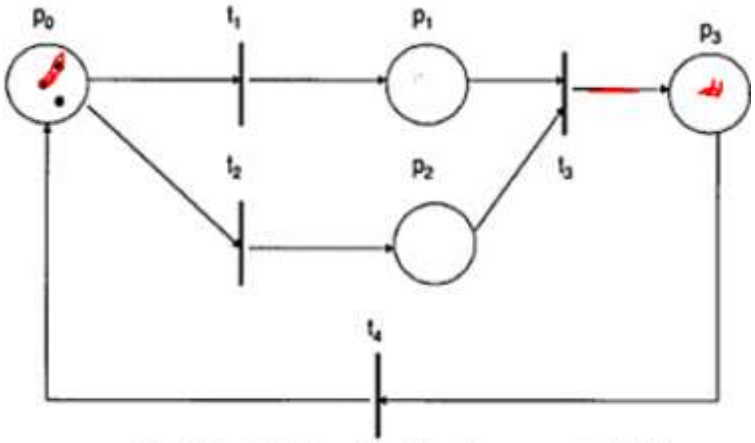


Fig 22: Petri net indicating *reachability*

Right now, t_3 is enabled as token are present in both P_1 & P_2. When t_3 will fire; a token will be stored in P_3 & tokens in P_1 & P_2 will be removed.
We received the target state in 3 firings.

Reversibility in Petri-Nets

• It is the property where, given some initial state, we can return back to this state, μ, in finite time.
• In Fig 23, μ_o, is not reversible, since we cannot get back to this state in a finite number of steps. As one of the tokens will be absorbed in transition t_3.

K-bounded Petri-Net:

• A Petri net defined to be k-place bounded if for all places, there are k or less tokens in each place for all possible states of the network.

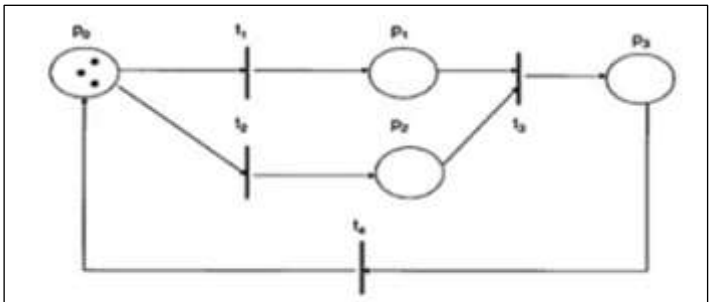


Fig 23: Petri net

• For example, Fig 23 is a three-bounded net. Because in all the possible state of this network the maximum number of tokens would be three.

Example Problem

Consider the Petri-Net in Fig 23 with the initial state as μ' = (0,1,0,2).
Is it possible to return to this state after every few transitions?
If is that so, provide the number of transitions. Is the state reversible?

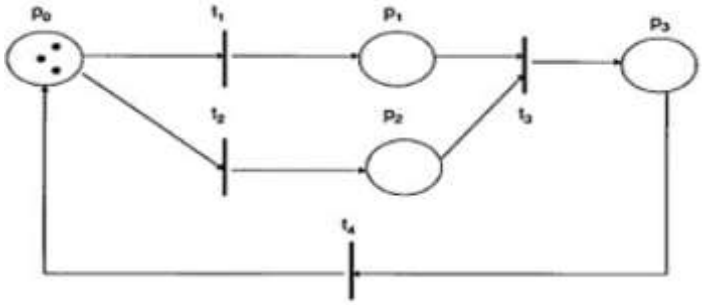
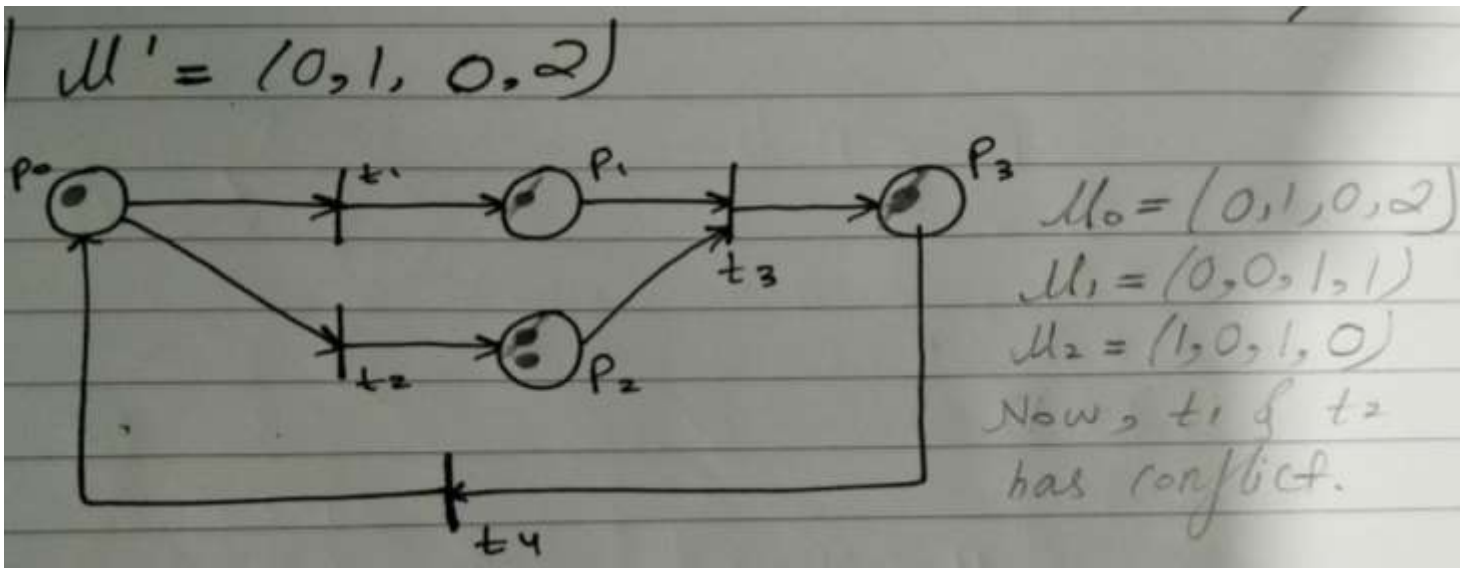


Fig 23: Petri net



So 3 firings.

Deadlocked Petri-net

- A Petri net is deadlocked if there are no transitions in the net that are enabled.
- Initial marking, μ_0 = (0,0,2,0).
- This marking results in no transitions being enabled.
- Conversely, a Petri net is considered *live* if there are any transitions enabled.

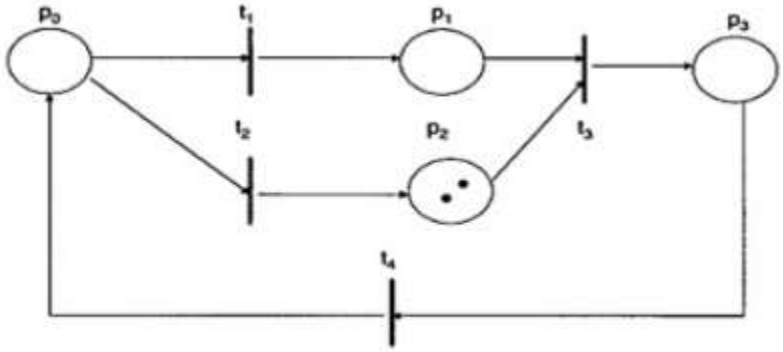


Fig 24: Deadlocked Petri-Net

Properties of Petri Nets

LIVENESS

- A transition is *live* if it is potentially firable in any marking of R(M1).

R(M1); it's actually representing the reachability set of a petri net. & reachability set is generated by means of reachability tree & the initial marking M1 will be considered as the root of reachability tree & by properly identifying all the nodes of the tree. The generation of a reachability tree involves a finite number of steps even if the petri net is unbounded.

- A transition is *dead* in M if it is not potentially firable; if the PN enters marking M the dead transition cannot fire any more.

SAFENESS

• A place is safe if the token count does not exceed 1 in any marking of R(M1).

If every place consists of single token then petri net is said to be safe however, it is not possible always.

• A PN is safe if each place is safe.

BOUNDEDNESS

• A simple generalization of safeness.

• A PN is k-bounded if each place is k-bounded.

It's a better property if we have already defined boundedness or we've defined the maximum number of resources already in the system then each place would have maximum these *k* number of tokens or resources.

If we limit the number of tokens to 1, we say that it's a safe PN & if we bound it to certain number *k* then we can say that PN follows boundedness property.

CONSERVATION

• A PN is strictly conservative if the total number of tokens is constant in each marking of R(M1).