# CS-417
# COMPUTER SYSTEMS MODELING

## Spring Semester 2020

### Batch: 2016-17

### (LECTURE # 29)

FAKHRA AFTAB

LECTURER

DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING

NED UNIVERSITY OF ENGINEERING & TECHNOLOGY

# Recap of Lecture # 28

| |
|---|
| Inverse of Petri-Net |
| Petri-Net as Multi-Graph |
| State of Petri-Net |
| Classical Petri-Net |
| Modeling of CS via Petri-Nets (Concurrency, Synchronization, Limited Resources) |

**Chapter # 8 (Cont’d)**

# PETRI NET-BASED PERFORMANCE MODELING

# Mutual Exclusion (Conflict)

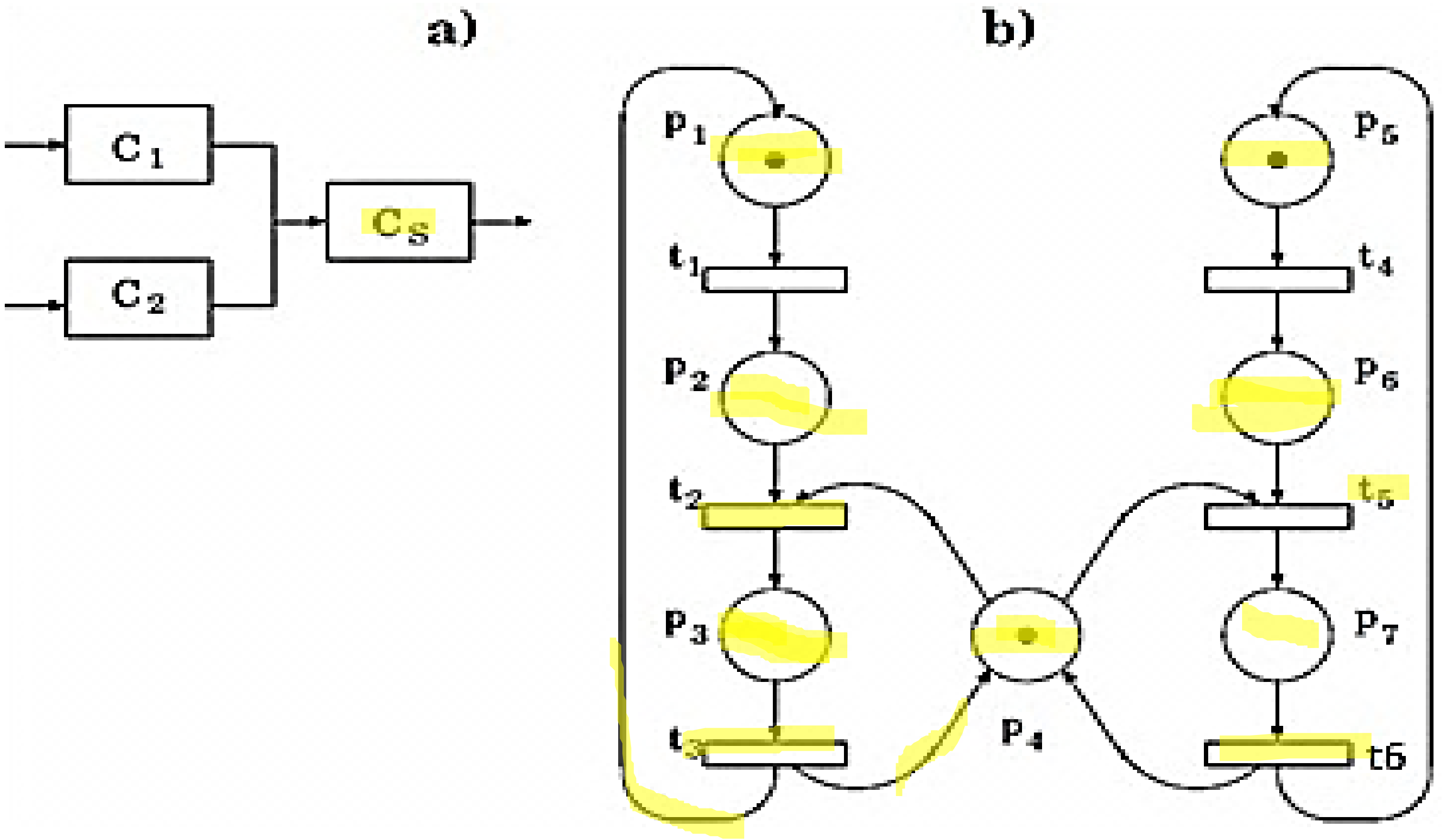


**Fig 16:** The Mutual Exclusion Problem

# 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.

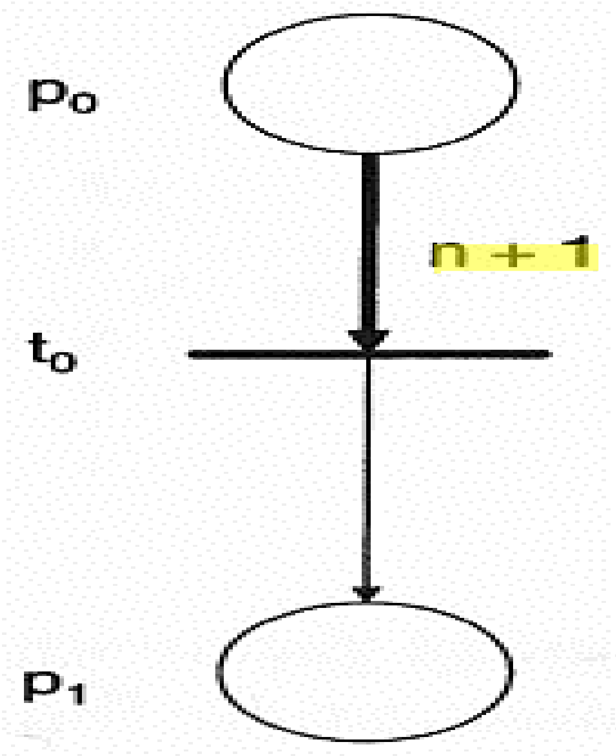


**Fig 17**: Petri net component to test condition greater than M

# 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.

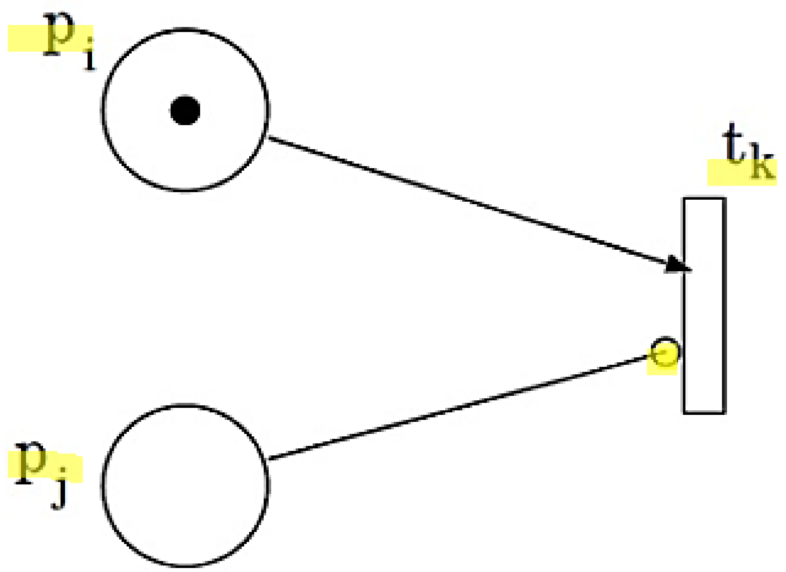


**Fig 18:** Inhibitor Arc

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

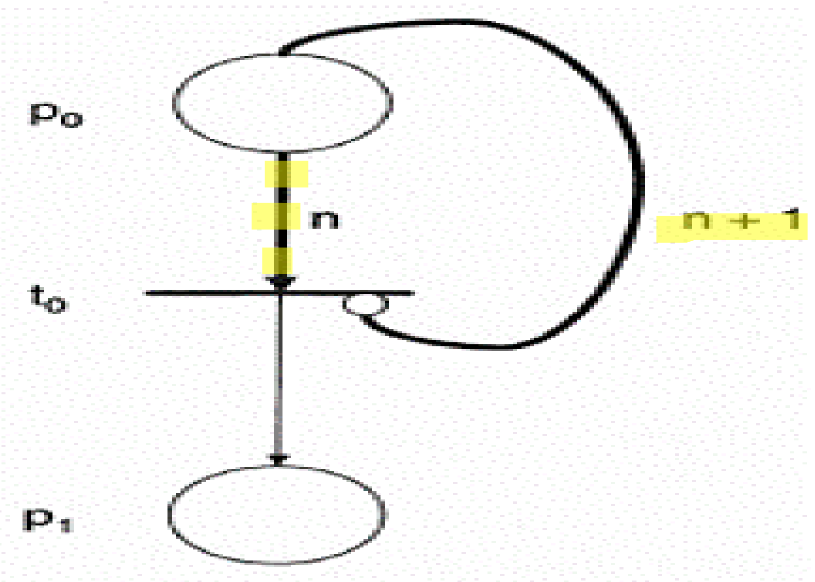


**Fig 19:** Petri net component to test condition equal but not greater than M

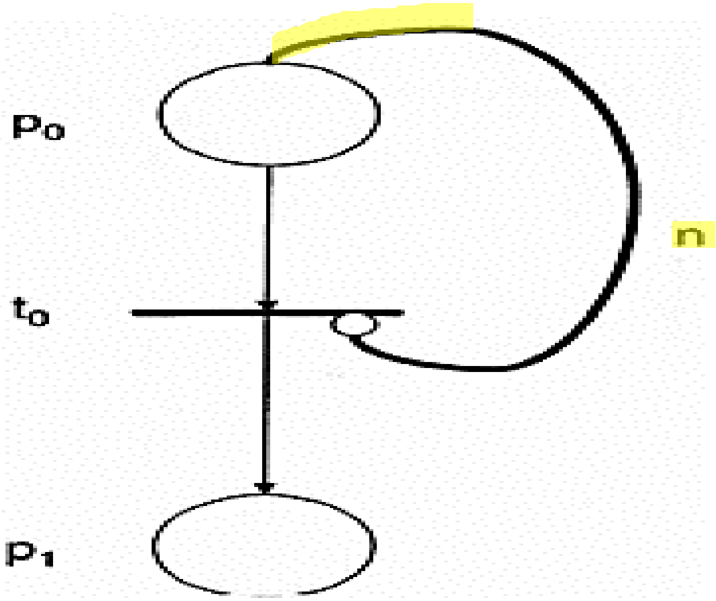


**Fig 20**

# 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 $t_1$ fires first, then we have two transitions enabled, $t_2$ and $t_3$. This then depicts a *conflict*.

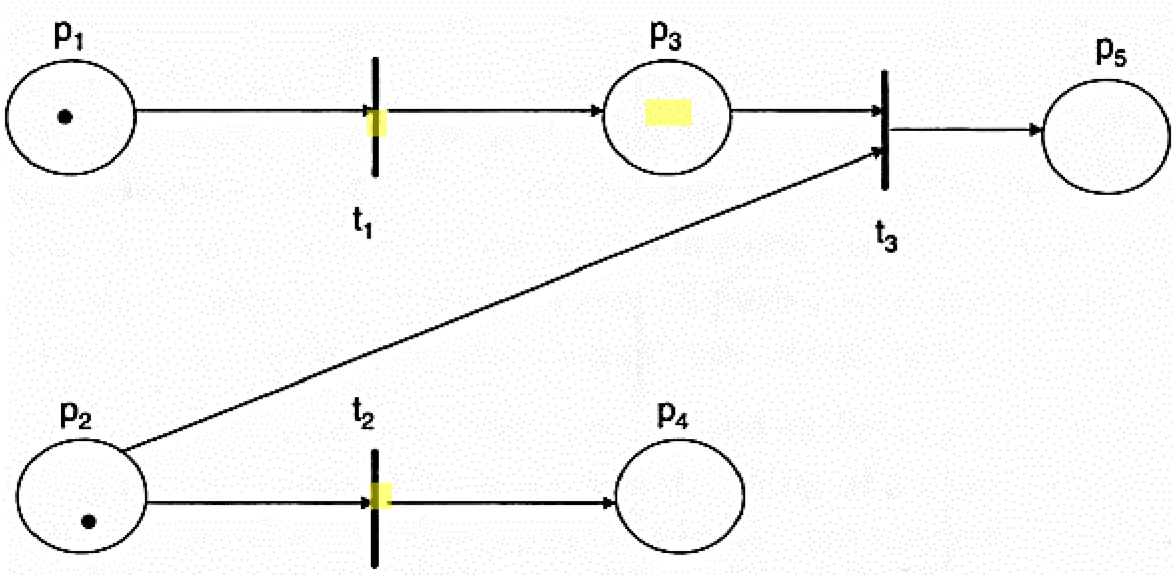


**Fig 21:** Petri net modeling *conflict* and *concurrency*

# *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., $\mu_0 = (3, 0, 0, 0)$, and a target state $\mu' = (1,0,0,1)$.
- We can reach this target state in three firings of our net.

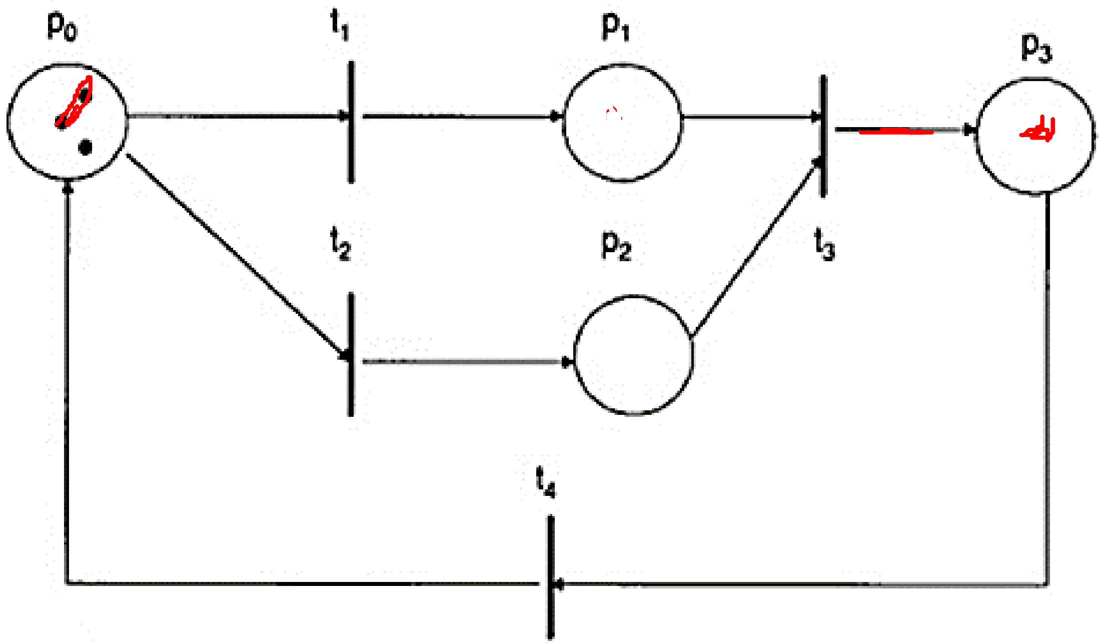


**Fig 22**: Petri net indicating *reachability*

# *Reversibility* in Petri-Nets

- It is the property where, given some initial state, we can return back to this state, $\mu$, in finite time.
- In Fig 23, $\mu_o$, is not reversible, since we cannot get back to this state in a finite number of steps.

***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.
- For example, Fig 23 is a three-bounded net.

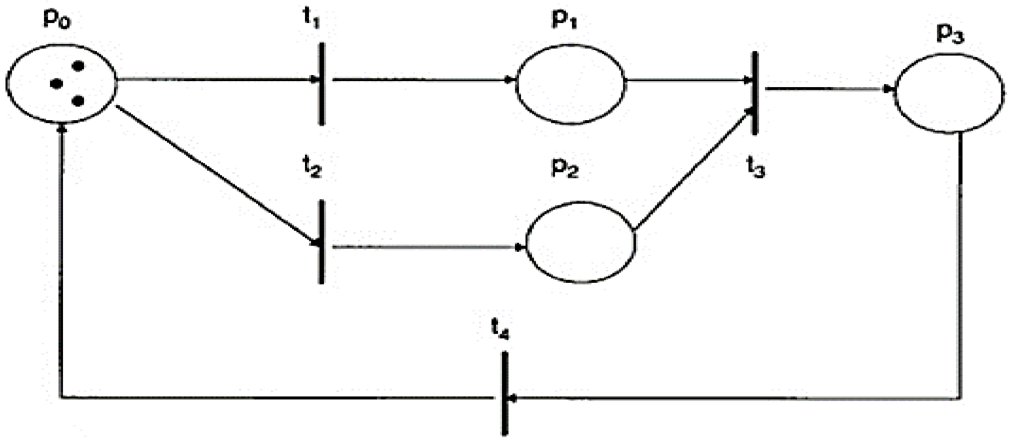


**Fig 23**: Petri net

# Example Problem

Consider the Petri-Net in Fig 23 with the initial state as $\mu' = (0,1,0,2)$.
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 ?

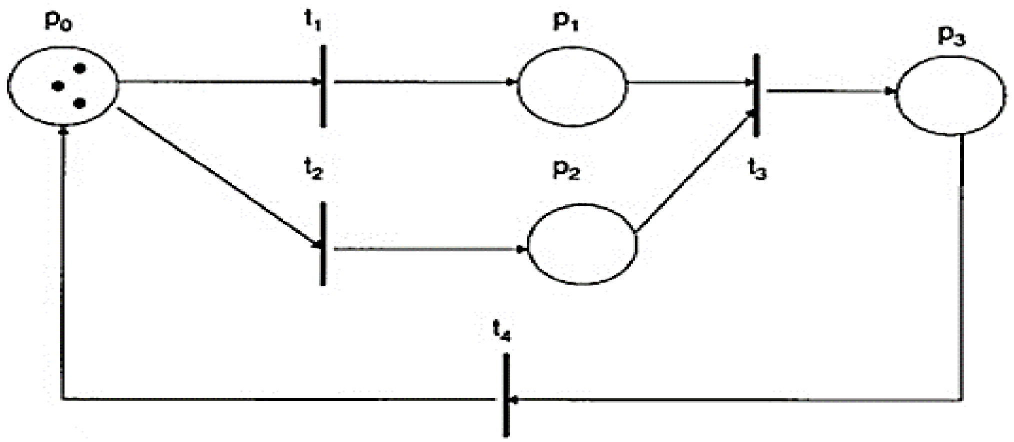


**Fig 23**: Petri net

# Deadlocked Petri-net

- A Petri net is deadlocked if there are no transitions in the net that are enabled.
- Initial marking, $\mu_0$ **= (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.

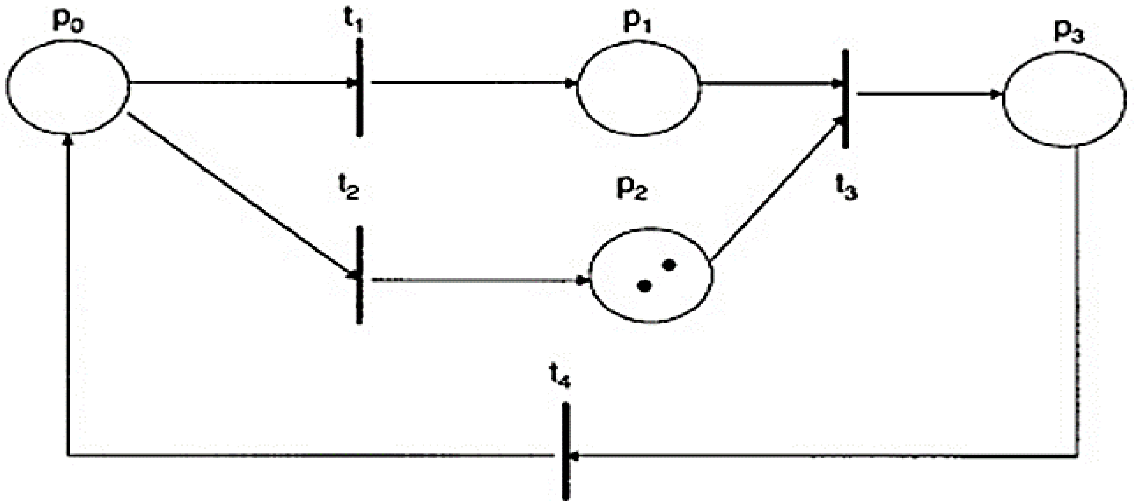


**Fig 24:** Deadlocked Petri-Net

# Properties of Petri Nets

## LIVENESS

- A transition is *live* if it is potentially firable in any marking of $R(M1)$.
- 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 $R(M1)$.
- A PN is safe if each place is safe.

# Properties of Petri Nets (Cont'd)

**<u>BOUNDEDNESS</u>**

- A simple generalization of safeness.
- A PN is $k$-bounded if each place is $k$-bounded.

**<u>CONSERVATION</u>**

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