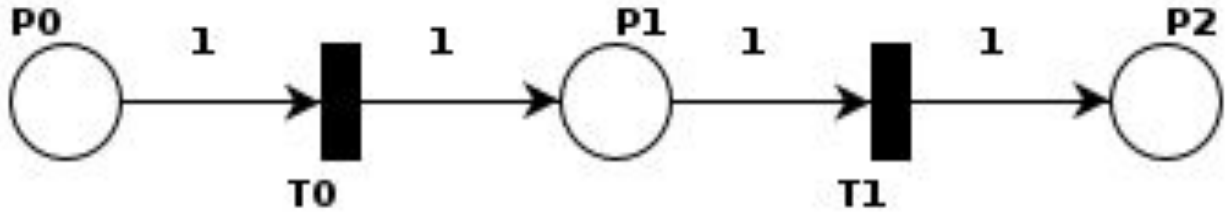
Process modeling in Petri Nets

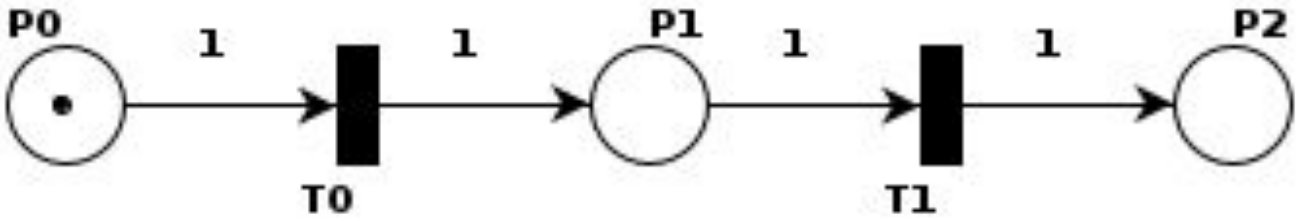
Basics

Rushikesh K Joshi
Computer Science & Engineering
IIT Bombay

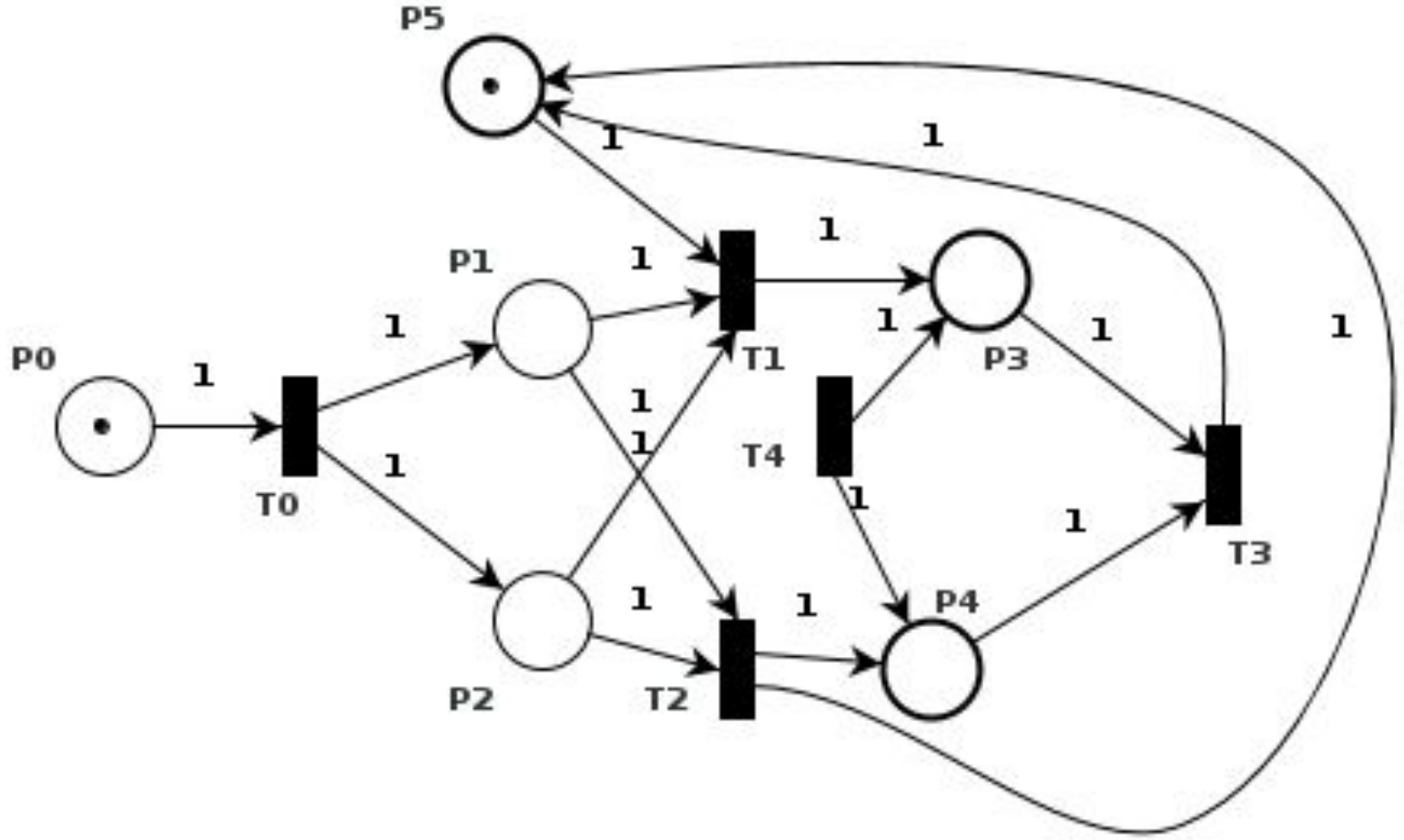
Example Petri Nets



Petri Net with a token



A Complex Petri Net



Circles are called **Places**

Rectangles are called **Transitions**

Tokens can be inserted in a set of places as **initial condition (A Marking)**

Then as per the **rules** of the petri-net language, the tokens move through the Petri Net (Markings change)

This is "**execution**" of our process, that is, state changes.

At a given point of time, **multiple places may have token**s,

The set of places which have tokens in them at a given point of time is called **Marking**

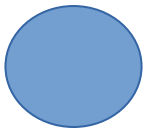
If pre-conditions of transitions are fulfilled (all its input places have tokens), the transitions are called "**enabled**"

One of the triggered transitions can "**fire/trigger**"

Once a transition fires, the **tokens** in the pre-places are **moved out** into output places of that transition

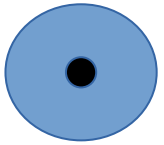
Basic Petri-net Elements

- Places
  - Represent states

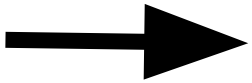


- Transitions
  - Represent actions

- Tokens
  - Part of current state



- Arrows
  - Flow elements



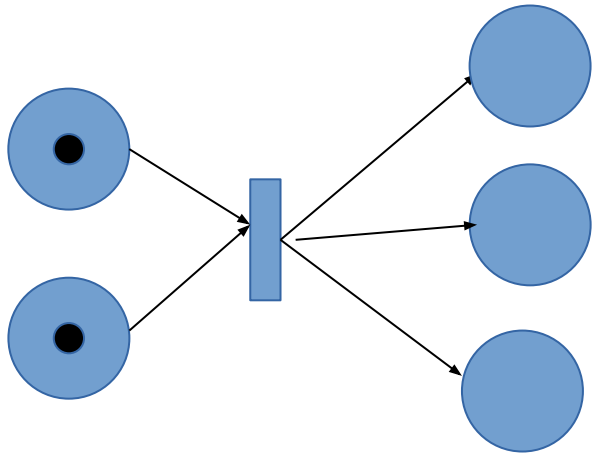
Elementary Petri-Nets

- at most one token per place

Transition Enabling Rule

- each of its preplaces (preconditions) has a token

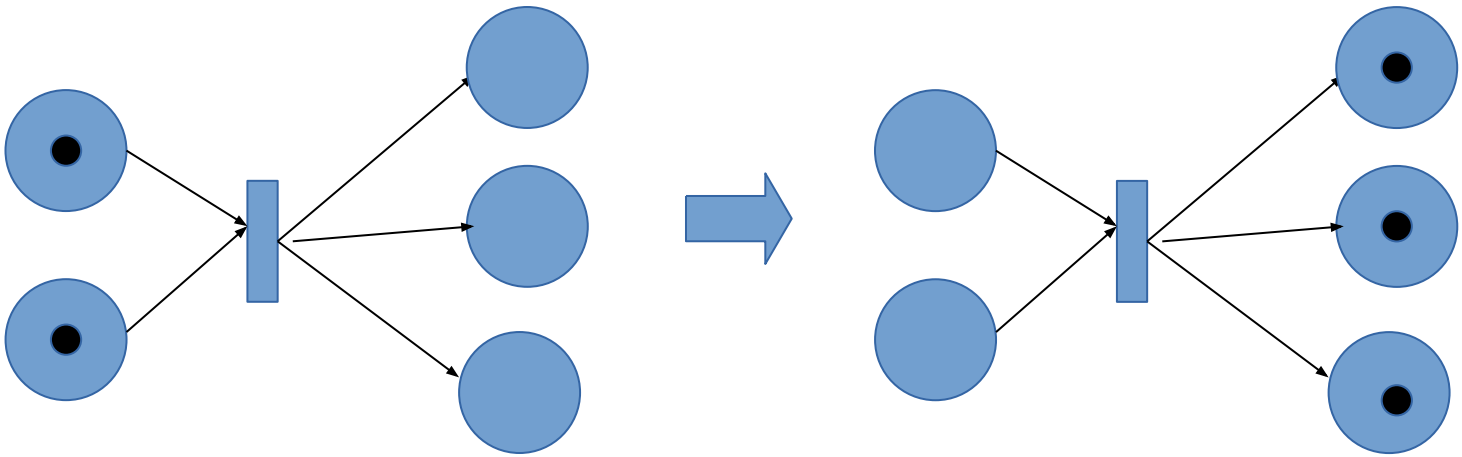
- after it is fired, each of its post places must be able to hold one new token



The above transition in the picture is enabled

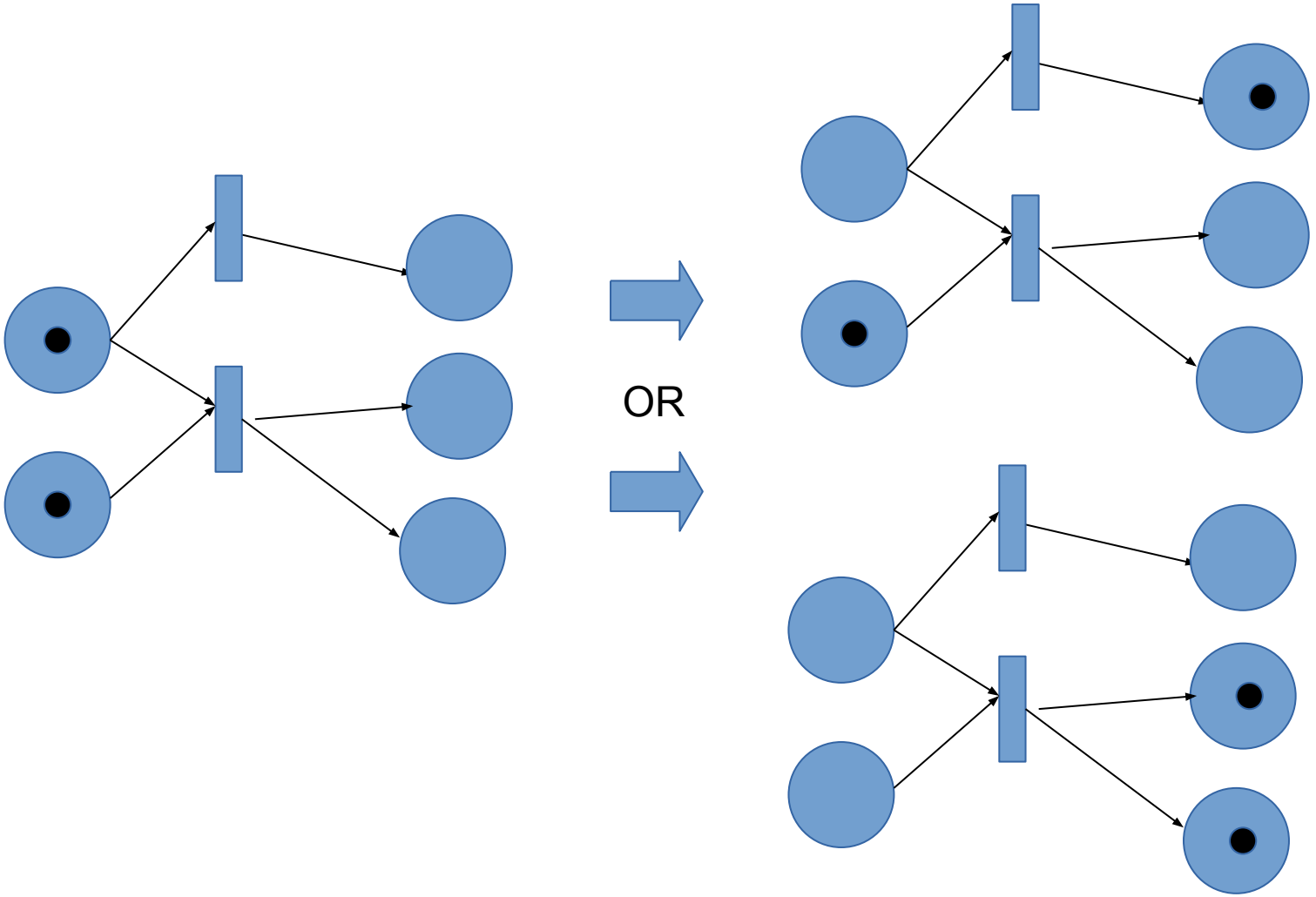
Transition Firing Rule and Example

- Enabled transition **fires**, and it moves the token(s) downstream Into all post places
  - One token is removed from every preplace
  - One token is deposited in every output place

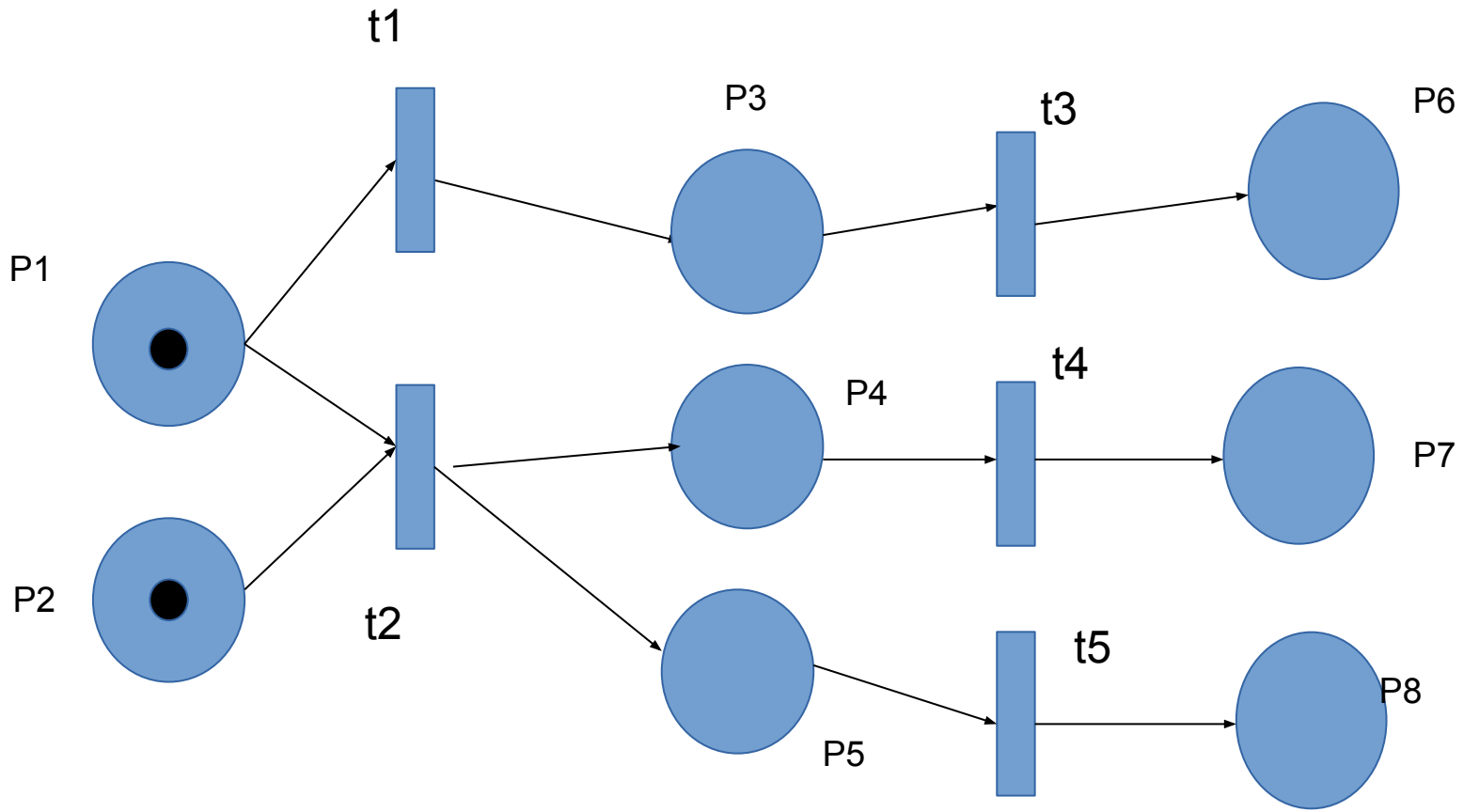


Transition Enabling Rule- multiple transitions may get enabled due to a single marking

- A marking may enable one or more transitions
- Any one of them can be fired

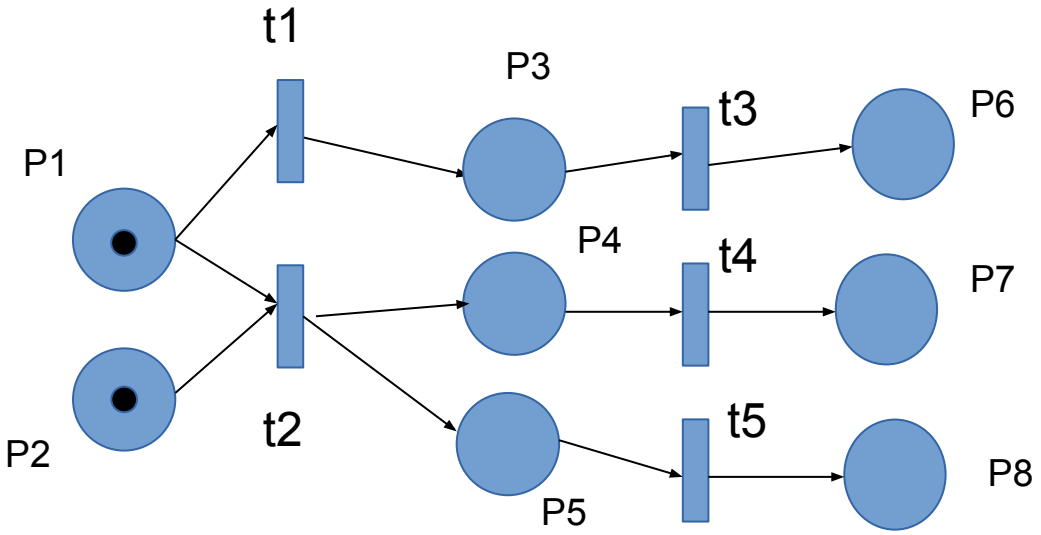


We label the places and transitions
So that we can analyze the state changes



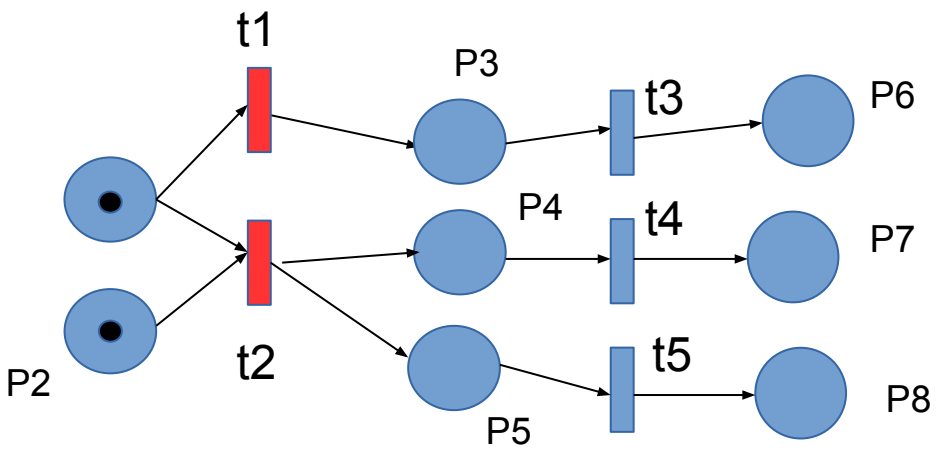
Current Marking = {P1, P2}
Set of enabled transitions = {t1, t2}

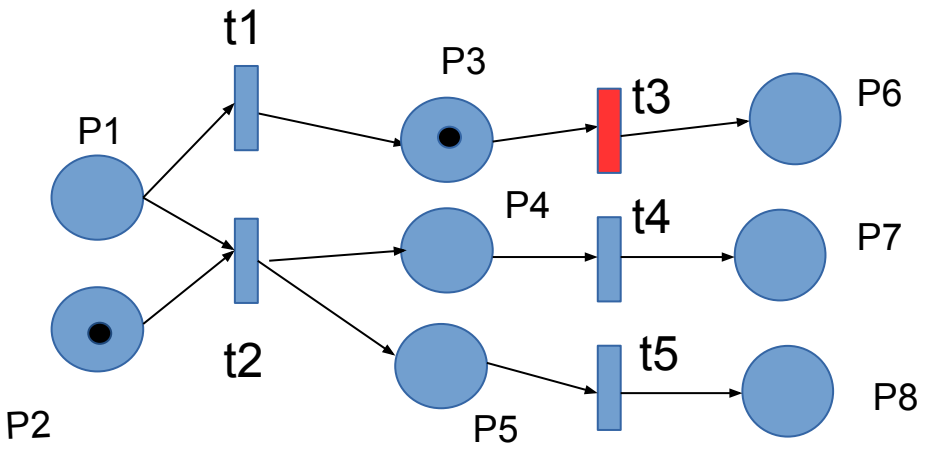
Firing Sequence (Trace) is sequence of transition firings
We can have many traces in a given marked net, since a marking can enable more than one transitions

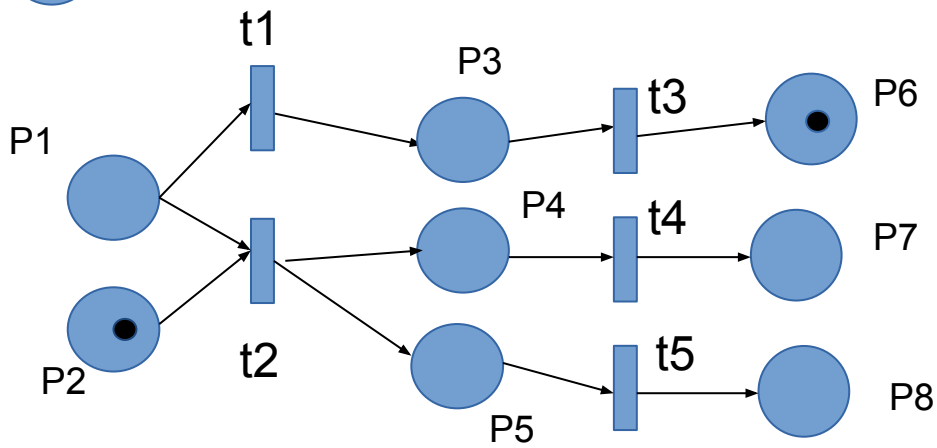


- Here, the following firing sequences (traces) are possible. Note that only one transition fires at a time
  - t1 --> t3 (trace 1)
  - t2 --> t4 -->t5 (trace 2)
  - t2 --> t5 -->t4 (trace 3)

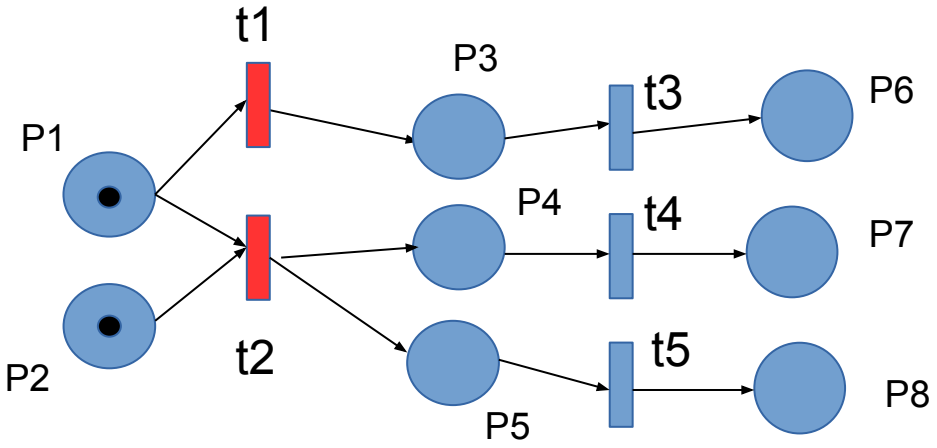
Trace 1 : t1 -> t3
Through states (markings):
{P1,P2}->{P3,P2}->{P6,P2}

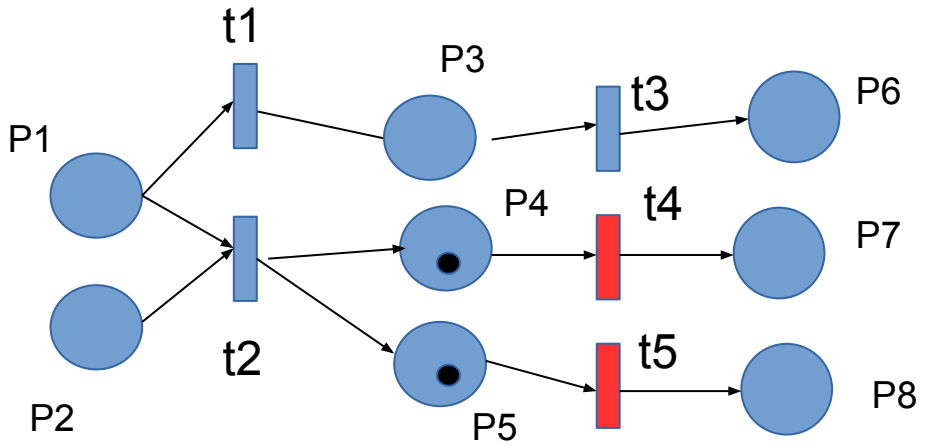


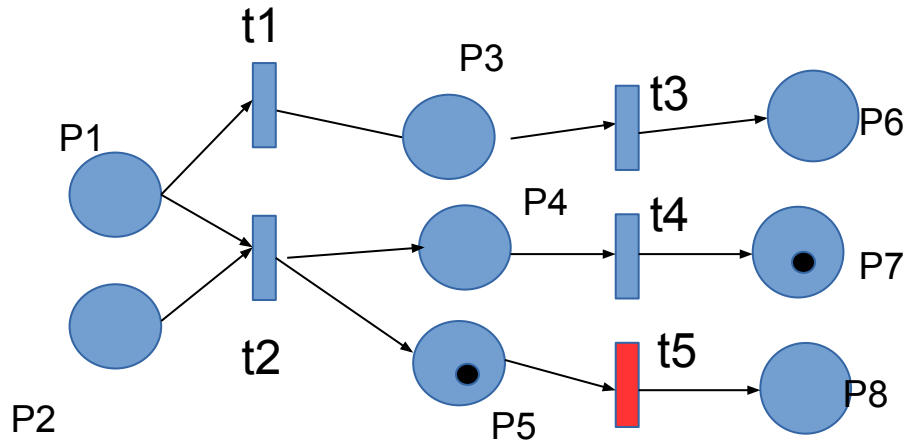




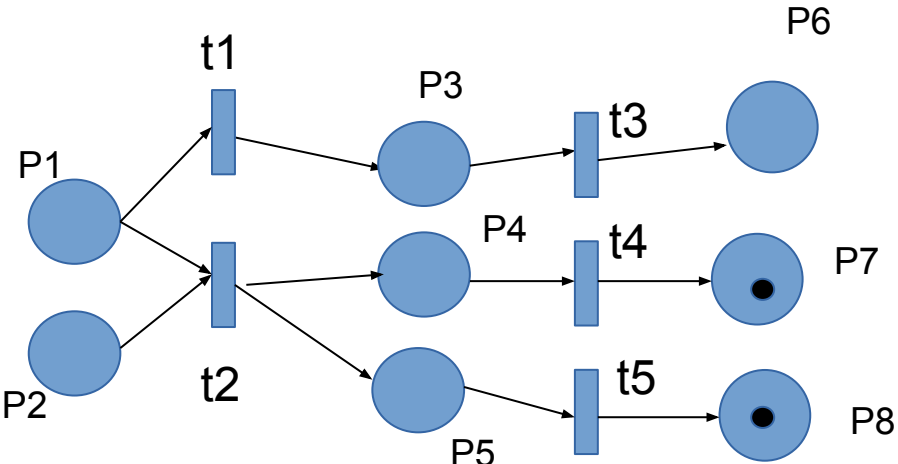
Enabled transitions are shown in red



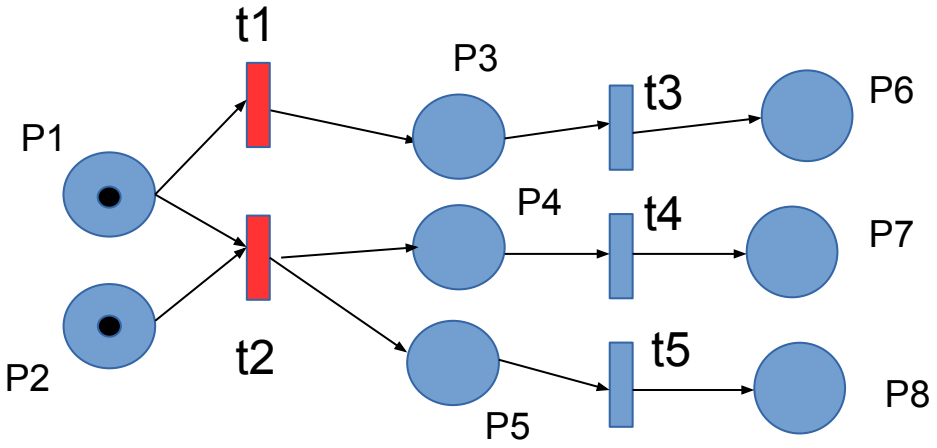


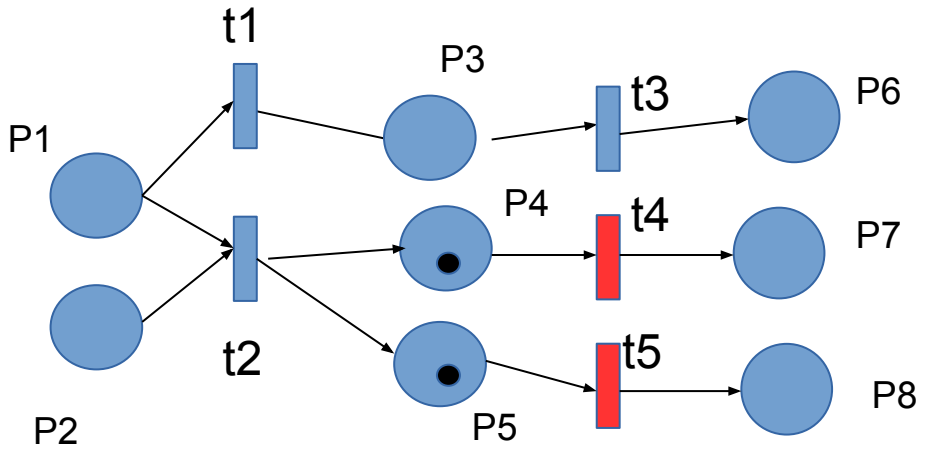


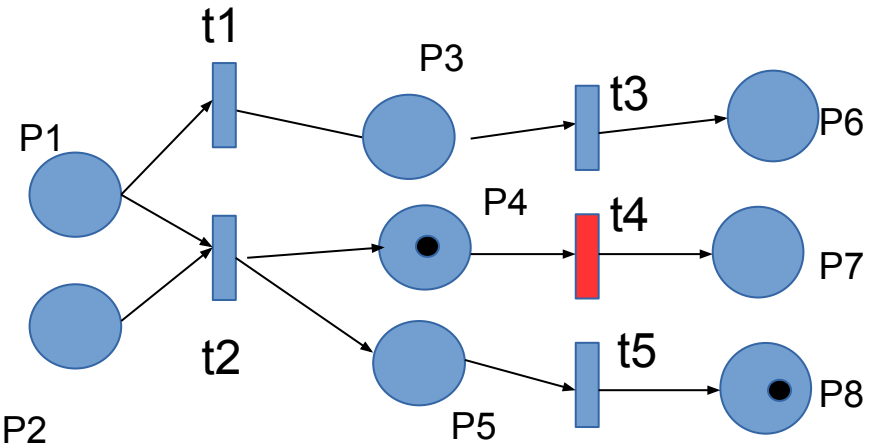
Trace 2 : t2 -> t4 -> t5
Through states (markings):
{P1,P2}->{P4,P5}->{P7,P5}
->{P7,P8}



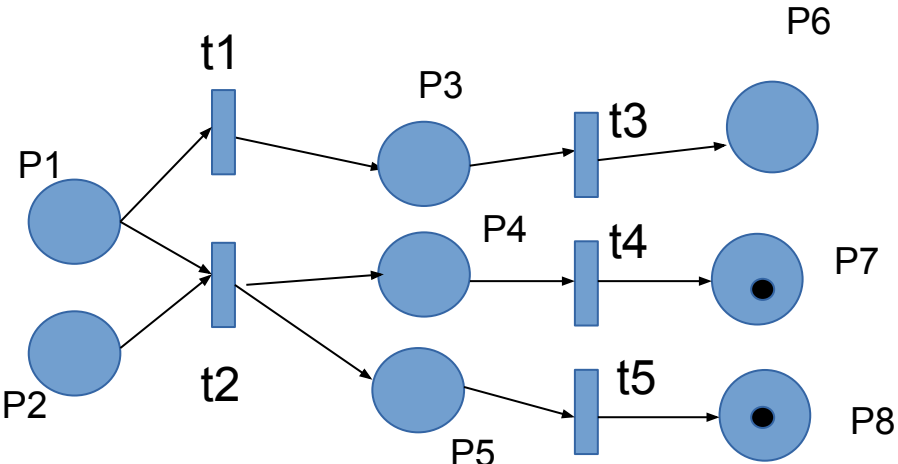
Enabled transitions are shown in red





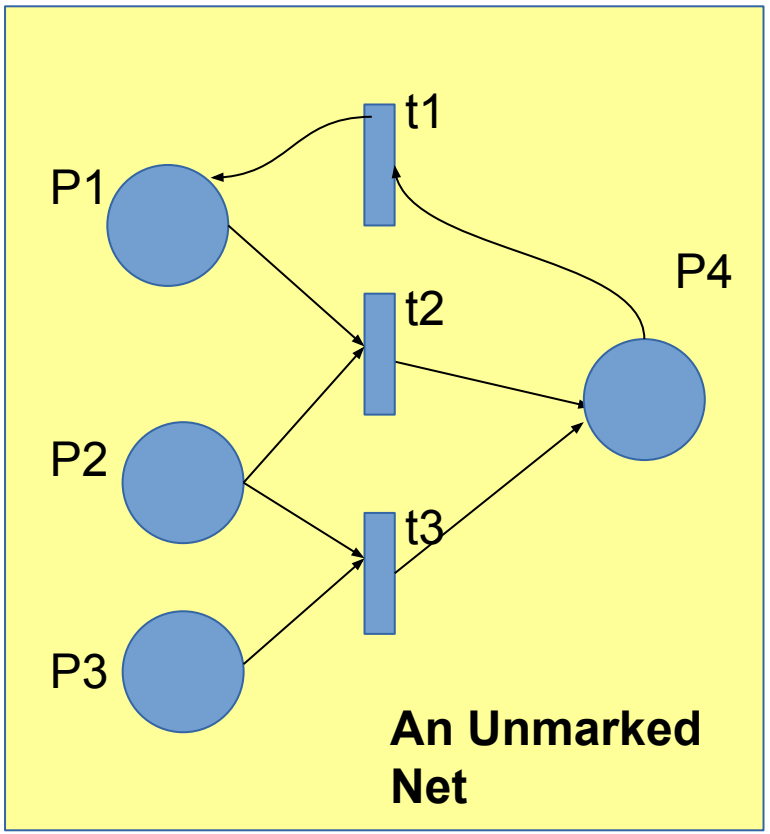


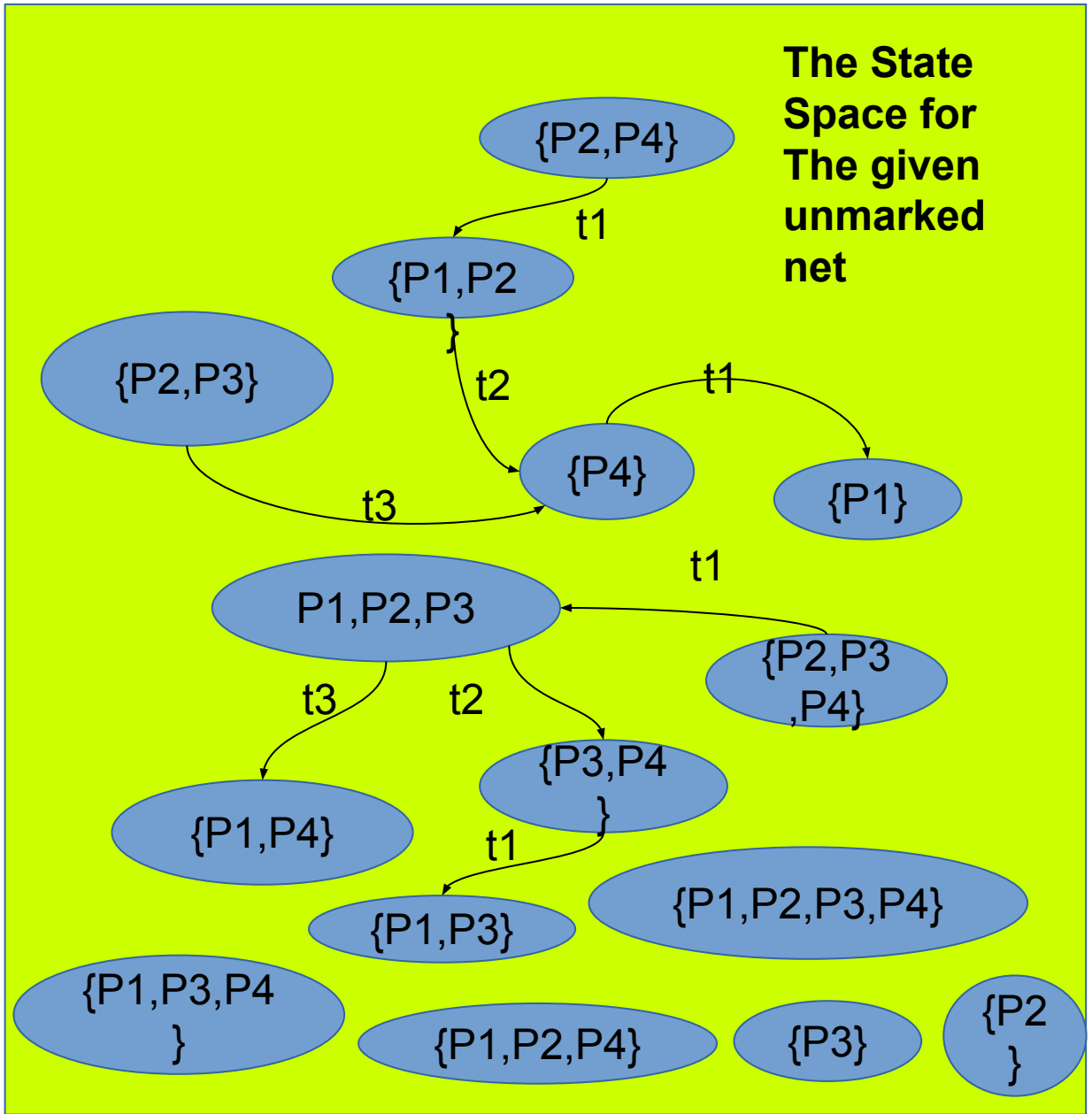
Trace 3 : t2 -> t5 -> t4
Through states (markings):
{P1,P2}->{P4,P5}->{P4,P8}
->{P7,P8}



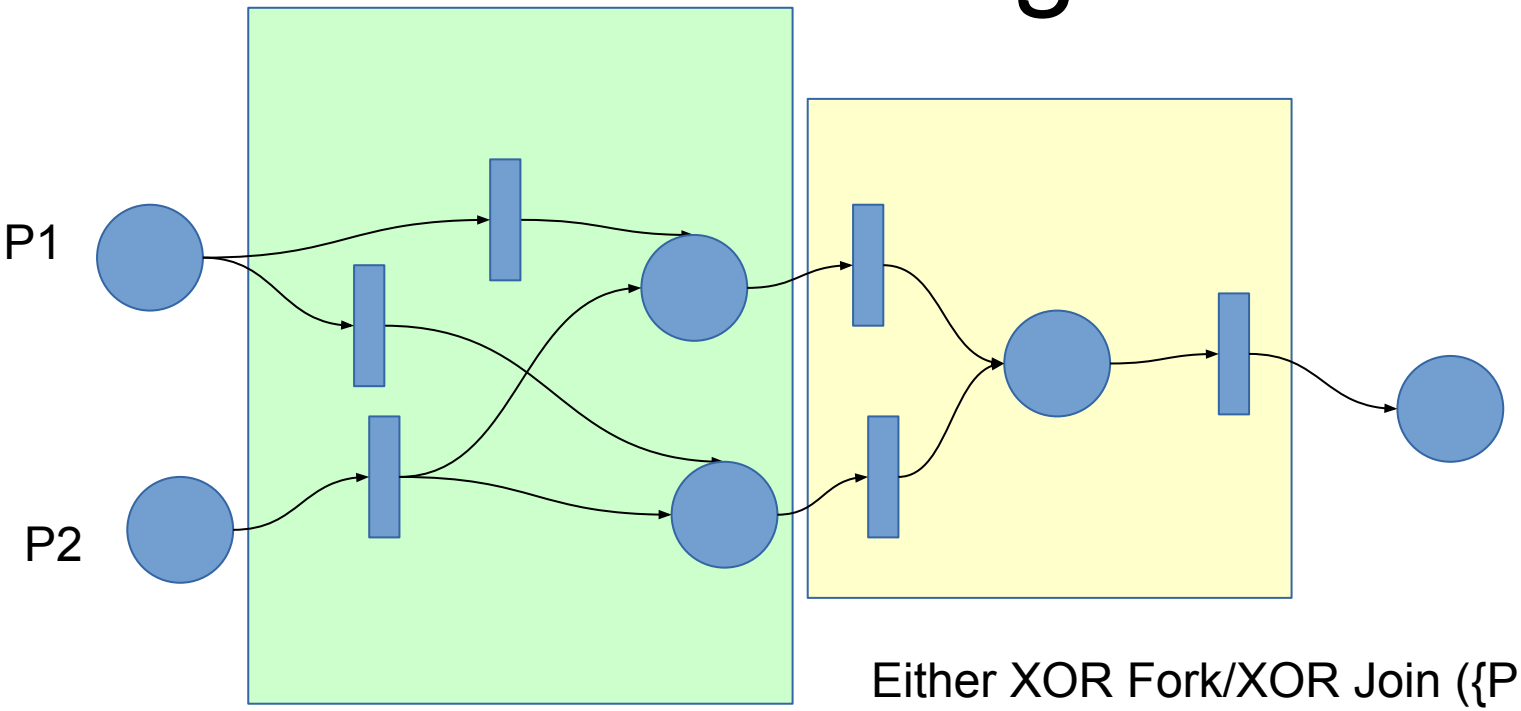
Enabled transitions are shown in red

State Space: State transition diagram
set of all possible markings, and the transitions through them





The same Petri-net for different purposes with different initial markings

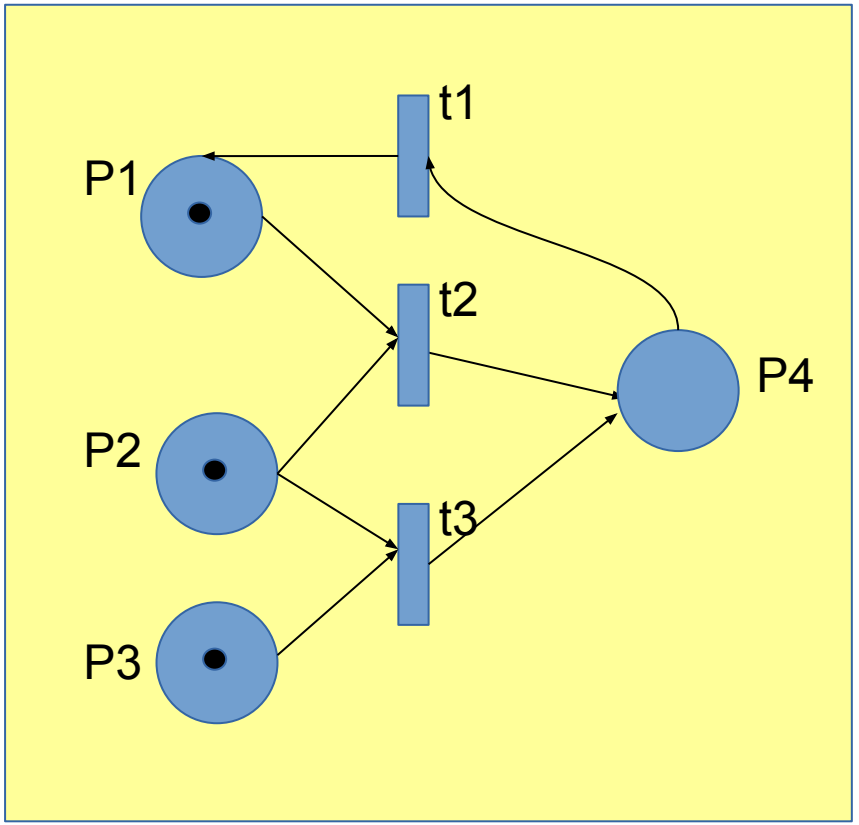


The helper part of the net

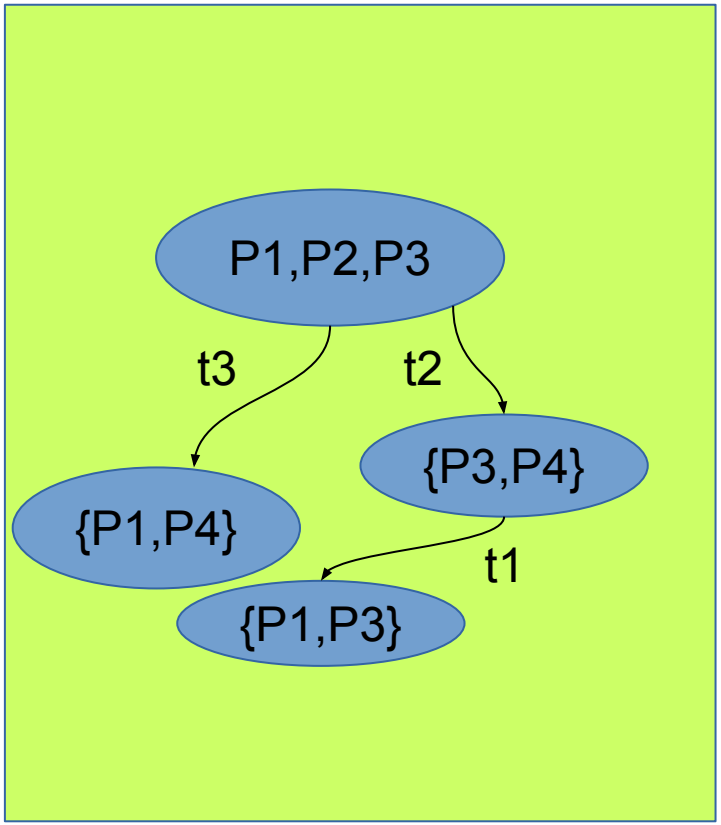
Either XOR Fork/XOR Join ({P1} as initial)
Or AND Fork/Multi-join

In Multi-join, multiple tokens get collected one after the other ({P2} as initial)

Reachability Graph
= state space reachable from a given initial marking

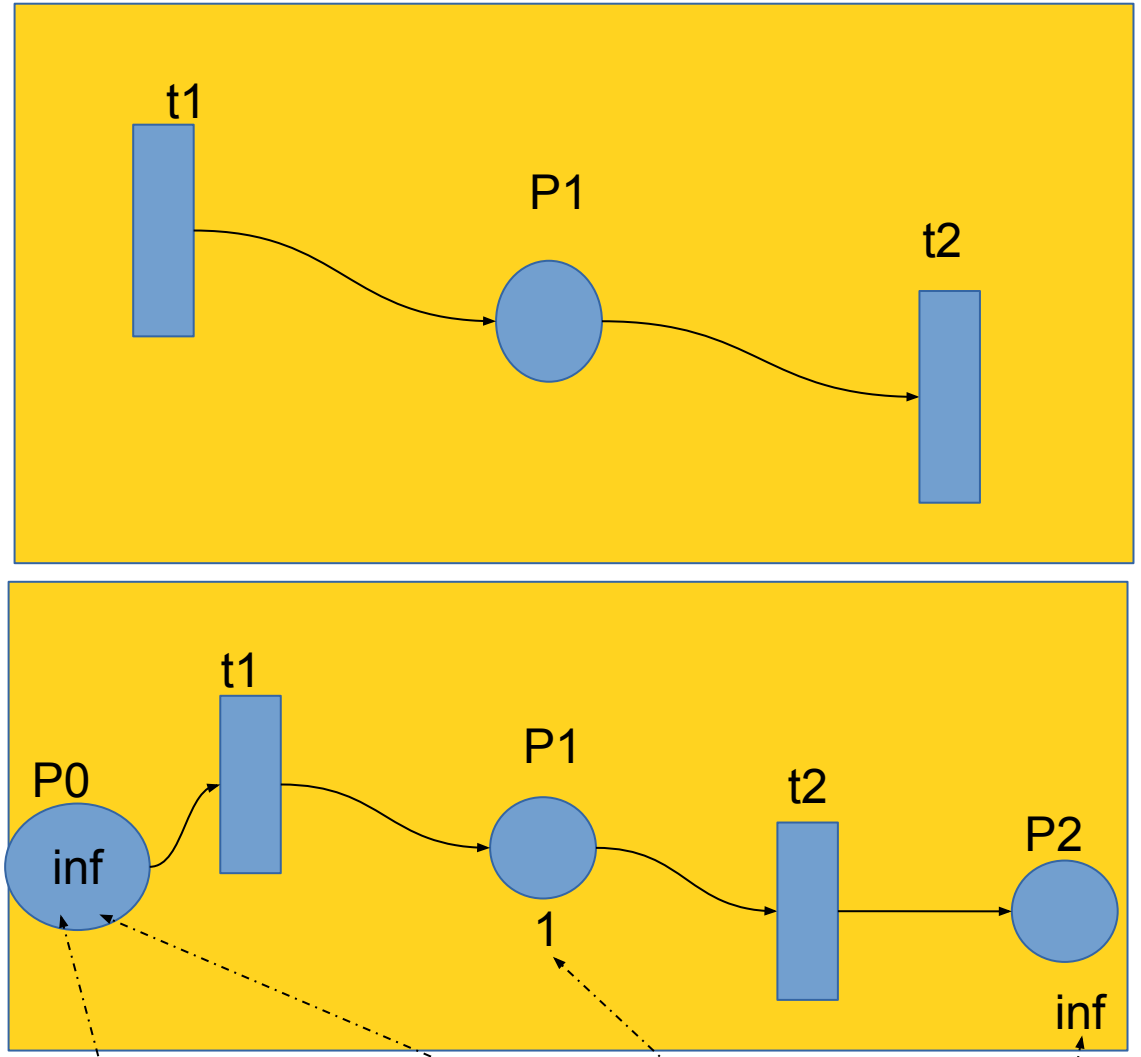


A Net with initial marking as {P1,P2,P3}



Reachability Graph
for the given net with given initial marking P1,P2,P3}

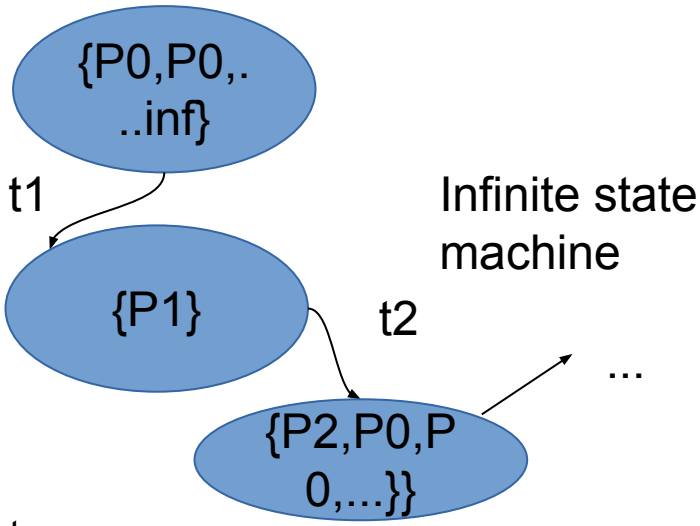
Infinitely enabled net



P0 has inf tokens

The number written inside represents those many (initial) tokens in the given state of the machine.

Capacity of P1 is 1,
capacity of P0,
Capacity of P2 is inf
Inf is default capacity in classical nets



A Problem: Rules of one masters program

- It's 2 semester program
- At most 10 courses
- Minimum 8 courses
- Per semester max 5 courses
- One R&D project can be taken as one course
- One masters project can be taken in place of 3 courses.
- MTP 1 in sem 1: counted as 3 courses, MTP 2 in sem 2: counted as 2 courses
- 1 seminar in 1^{st} sem is must
- Backlogs of sem1 can be taken in sem2
- If any backlogs remain, one more semester is granted

Simplified Version of the Problem

- 5 courses
- 2 semester
- 2 courses per semester: total exactly 4 courses
- A course cannot be taken twice

- Make your net to represent this system
- Try to reduce the no. of transitions
- Model with: One transition representing one course

Vending machine Problem

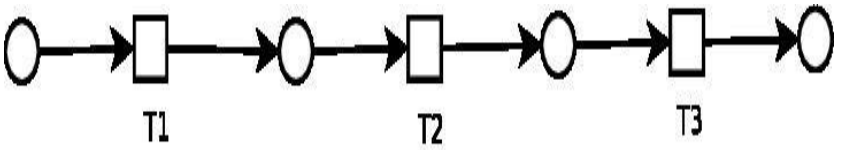
- Insert Rs. 25: currency Rs. 5, Rs. 10 coins accepted e.g. 5+5+5+5+5, 10+5+10 etc. (design as small a machine as possible)
- Choose the drink
- Pick it up
- The machine is ready for the next task
- Time out of 1 minute, the machine returns all the coins inserted and resets to initial state
- Invalid coin is rejected, and all the coins are returned, the machine resets with a spoken message of invalid coin.

Workflow Nets

Workflow Nets

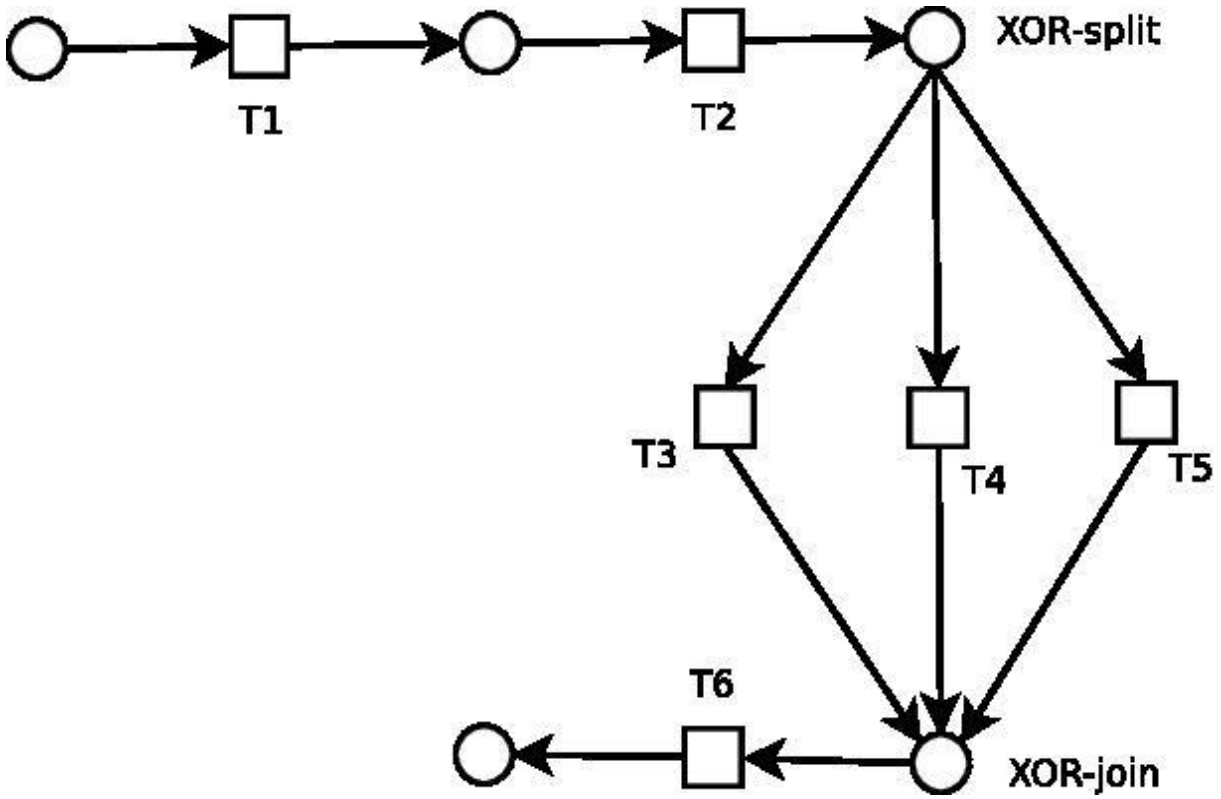
- Unique source place
- Unique sink place
- Connected
- Unique initial marking, unique terminal marking
- Well-formed – every transition is reachable, every marking is reachable, every marking terminates

Workflow Patterns: SEQUENCE

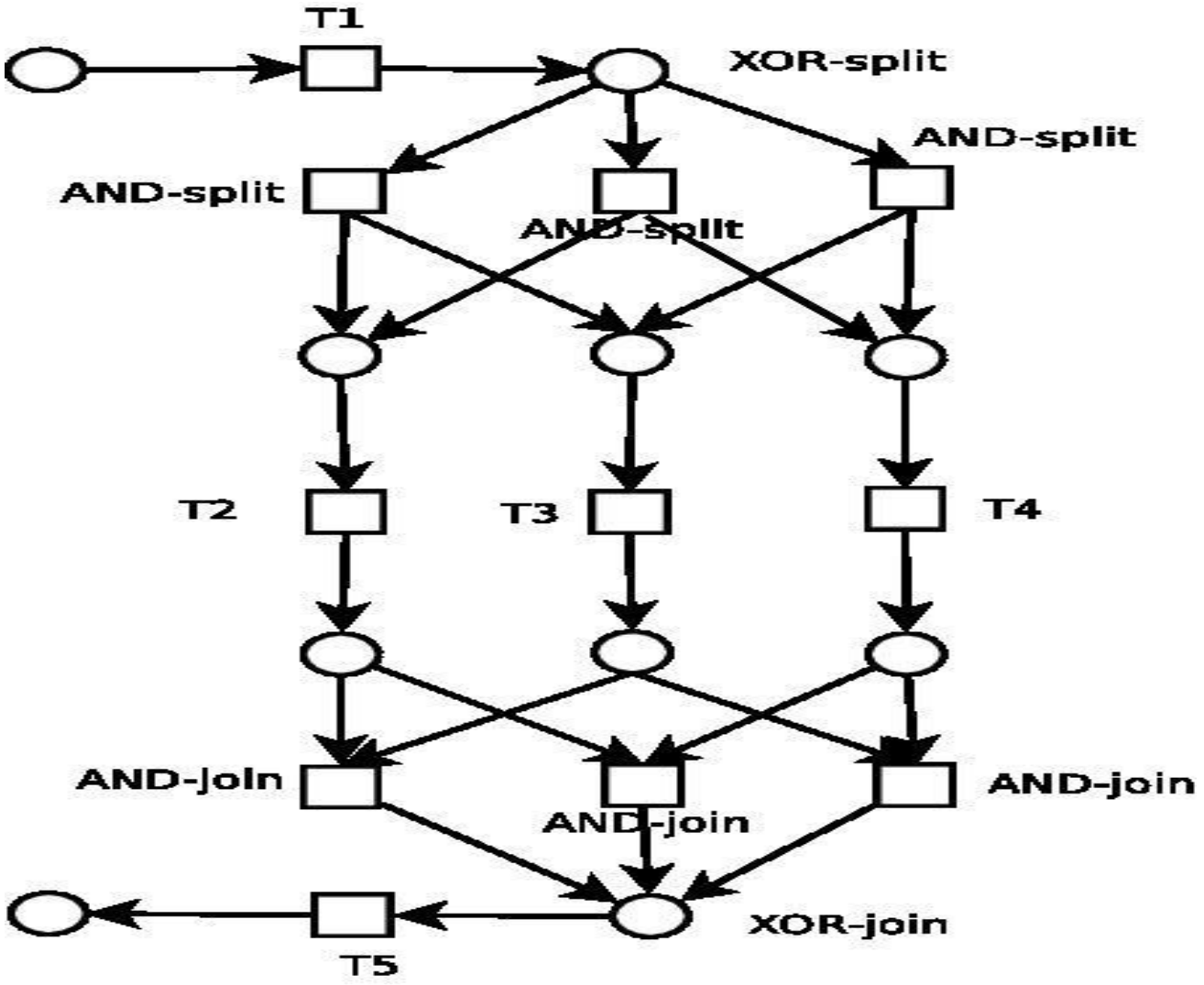


Sequence

Workflow Patterns: XOR

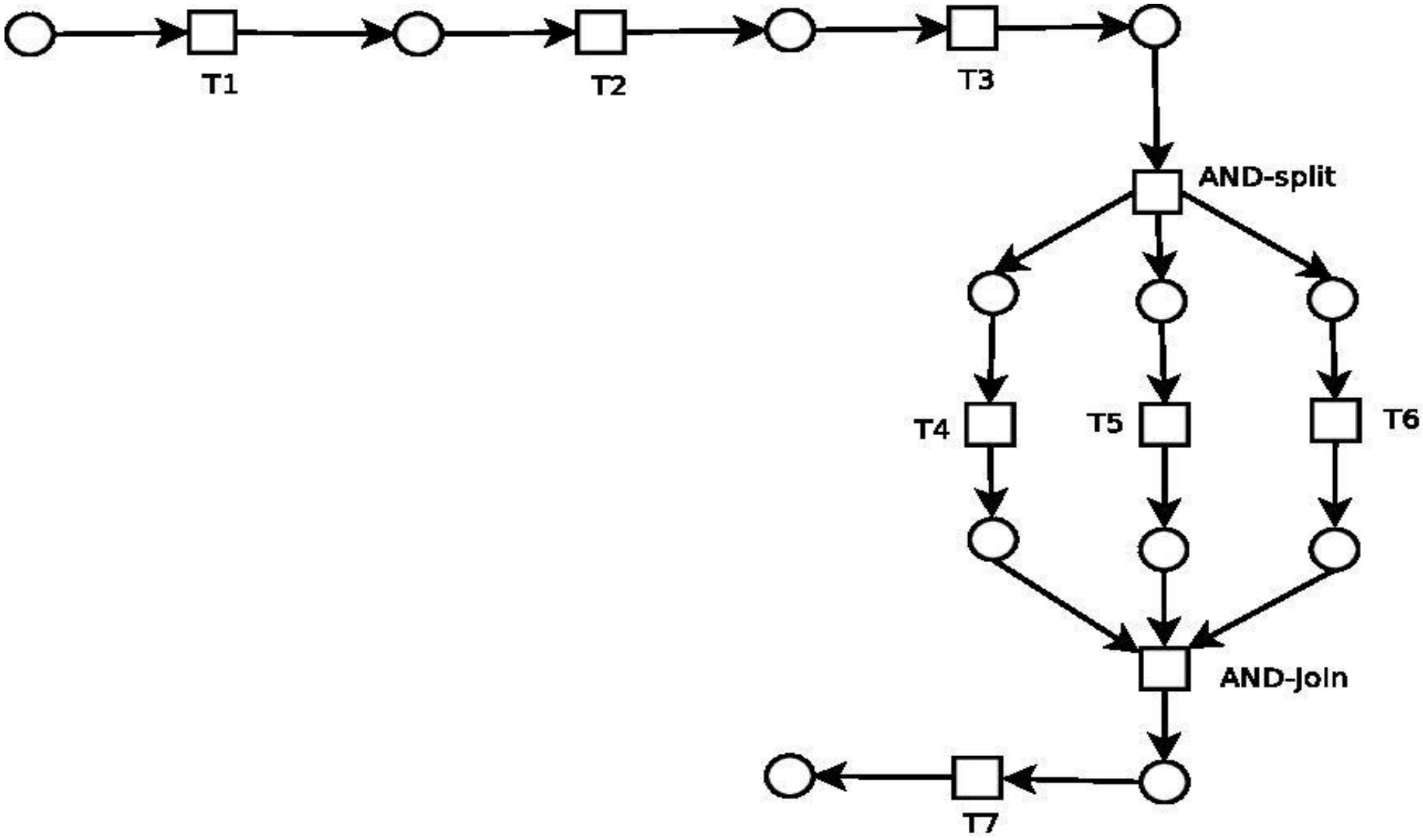


Workflow Patterns: 2/3 Choice



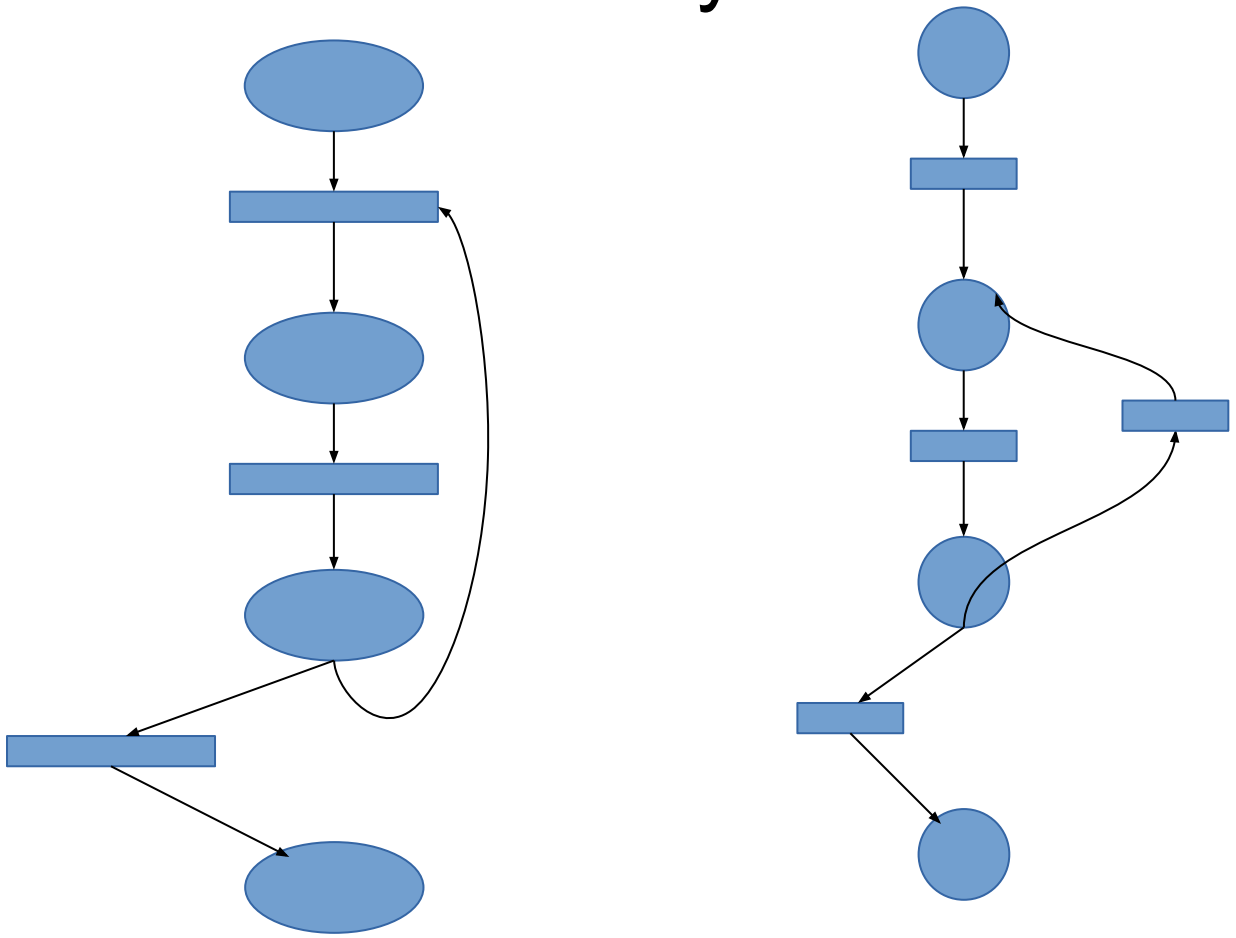
2/3 split-join

Workflow Patterns: AND

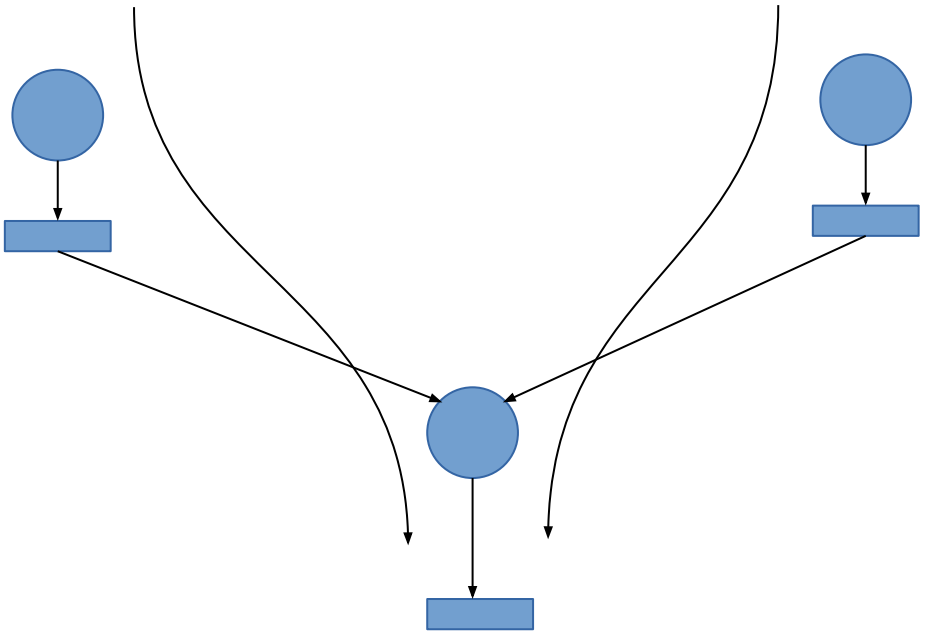


Workflow Patterns: Iteration

which one is correct? Which one is incorrect? Why?



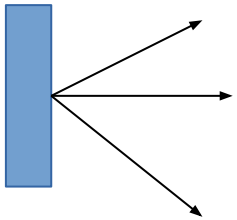
Workflow Patterns: Multi-merge



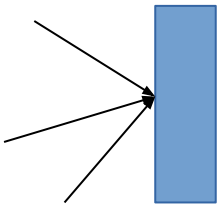
Tokens can arrive both ways, and they are all sent down

Split/Join

- Parallel split

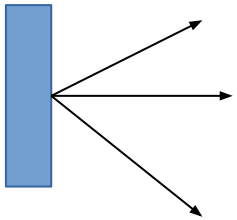


- Synchronization (parallel merge/AND join)

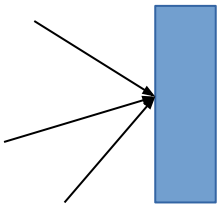


Split/Join

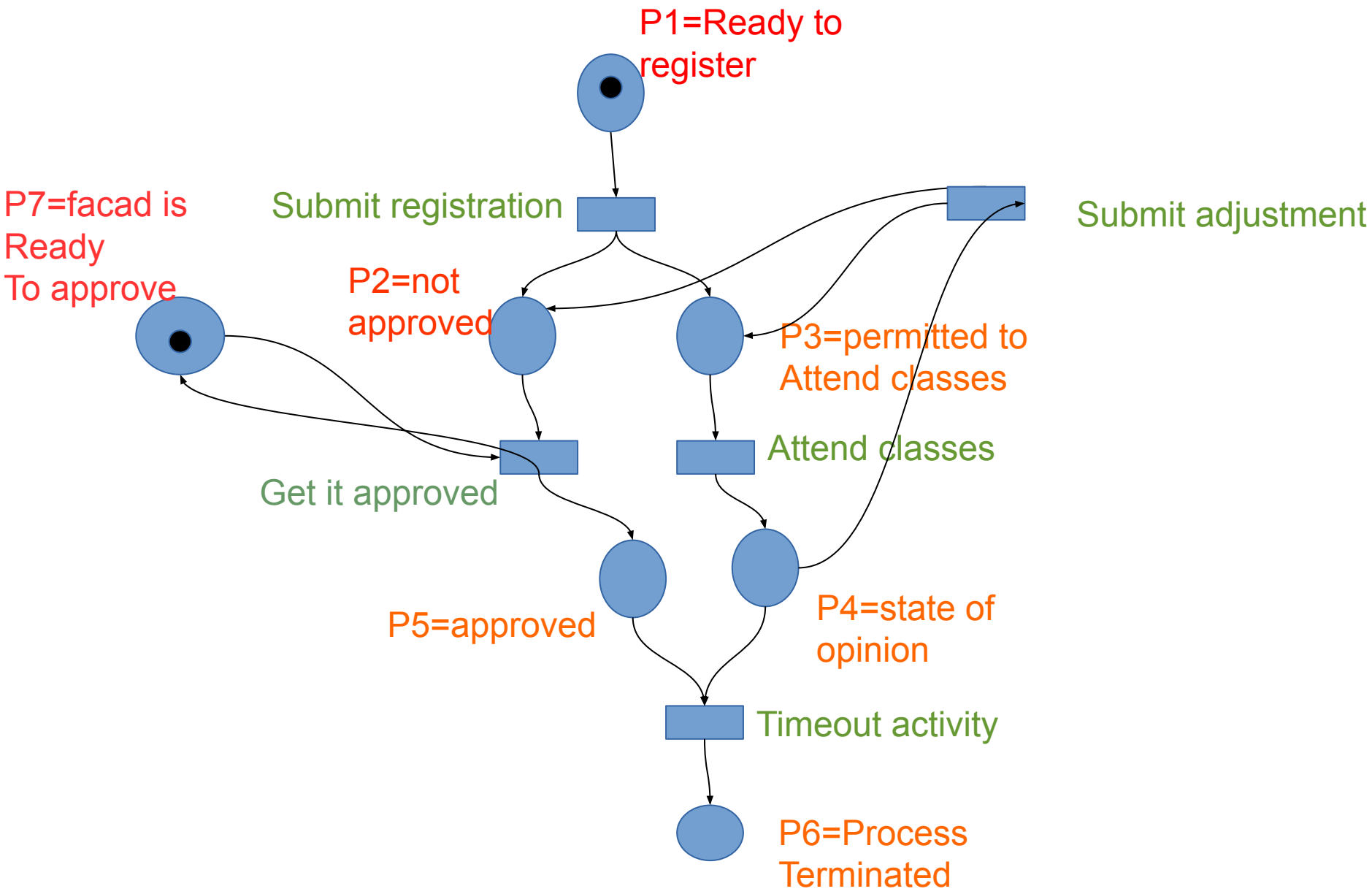
- Parallel split



- Synchronization (parallel merge/AND join)



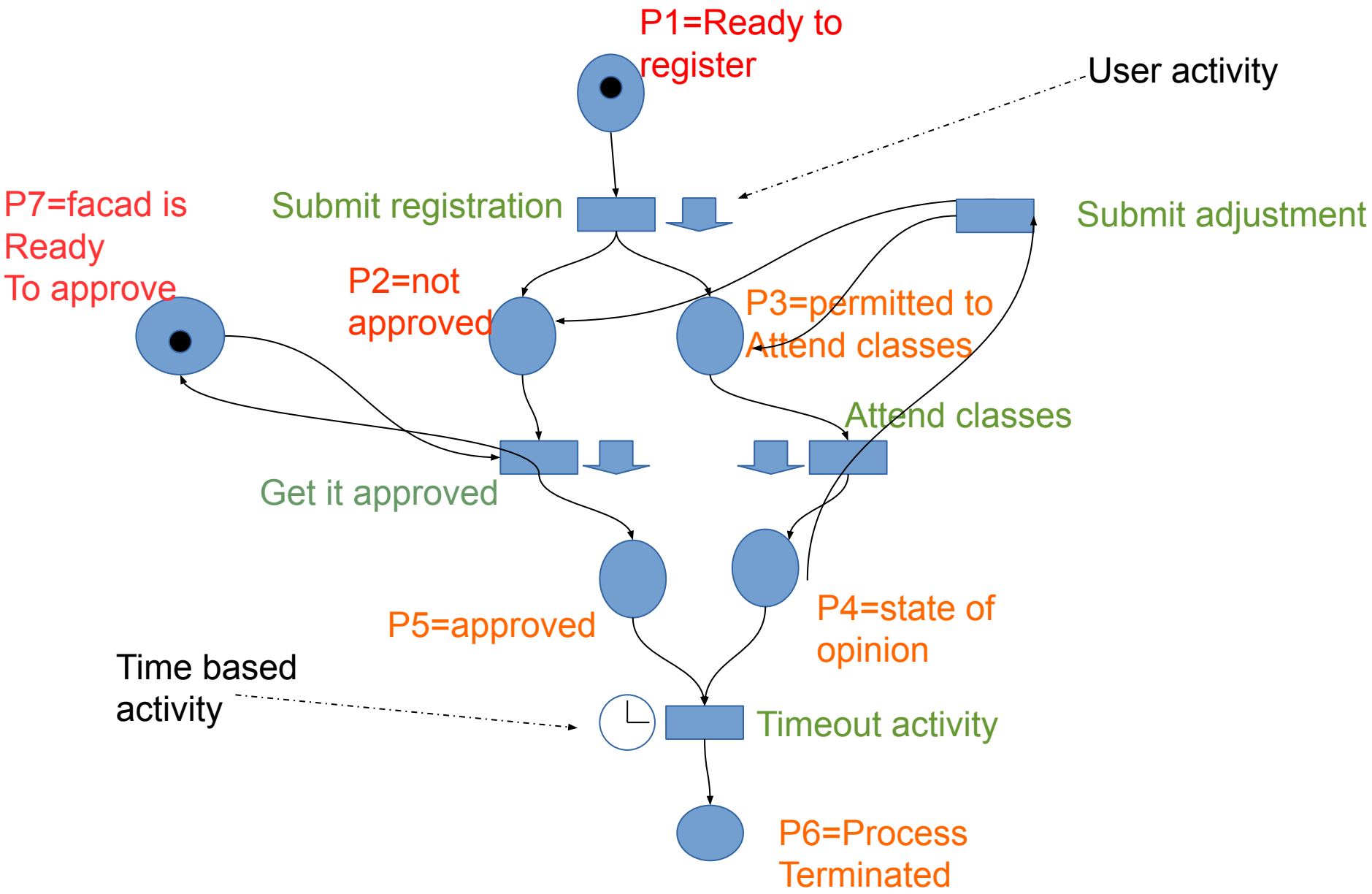
Roles as tokens in places



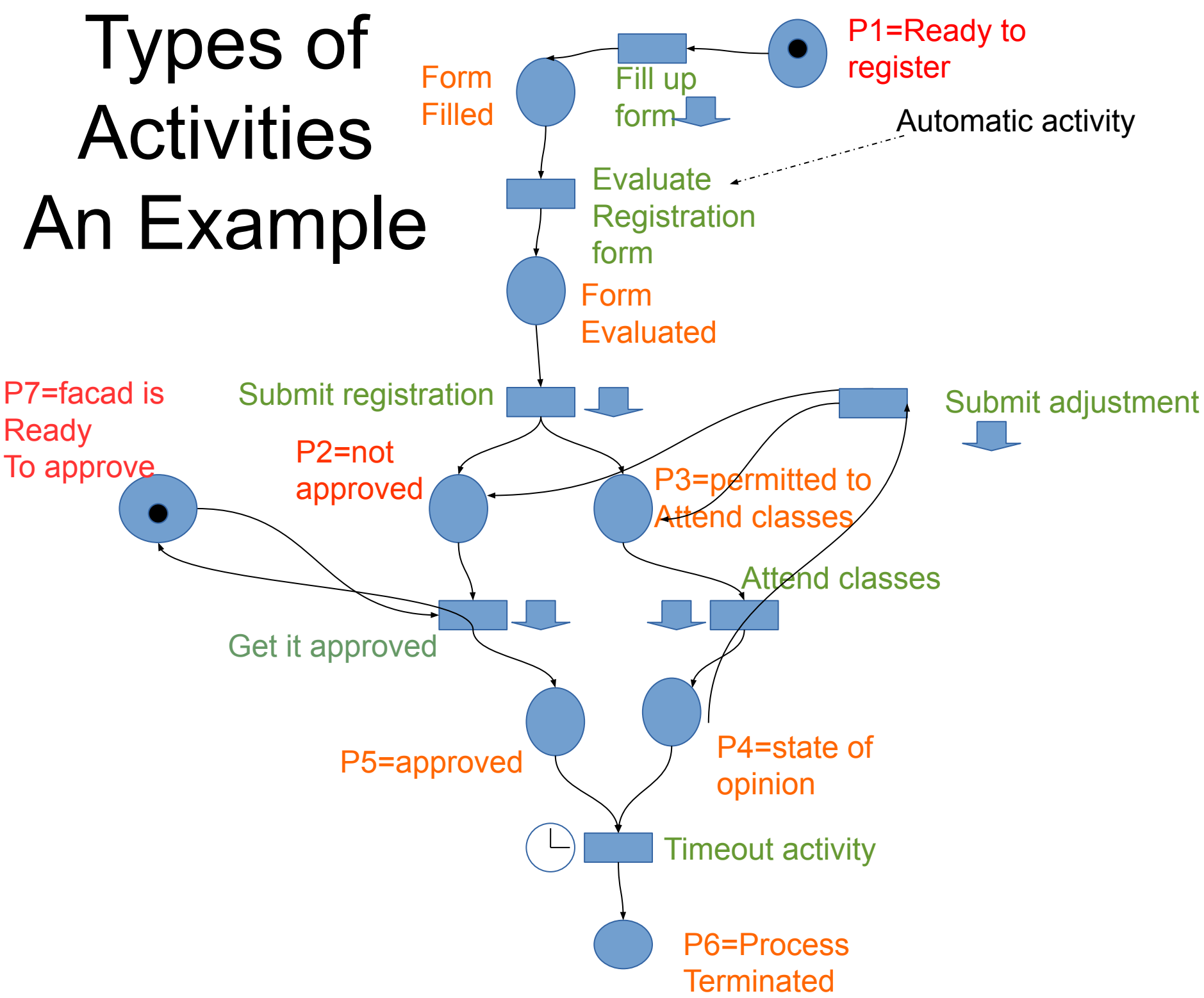
Types of activities

- Automatic Activity
  - Computer can execute it fully
    - (when enabled, it is executed automatically such as by an algorithm, script task etc.)
- User Activity
  - A human being executes it
    - (though enable, it is done manually)
- Message Activity
  - An external message triggers the task instance
    - (though enabled, it requires a message to trigger it)
- Time Triggered Activity
  - Task needs to be trigged at a particular time, or after a certain period of timeout
    - (though enabled, time has to trigger it)

Types of Activities
An Example



Types of Activities An Example



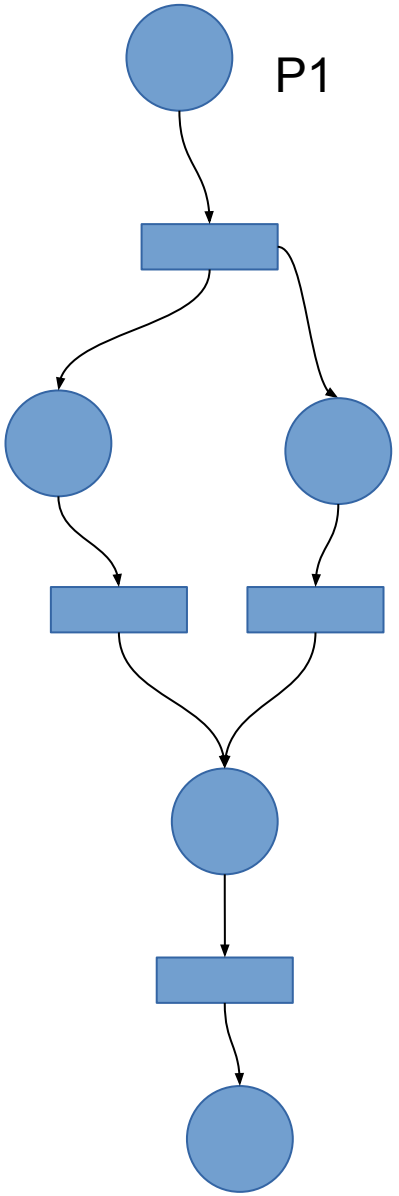
Problem

- Re-engineer the above net to reflect the process that we are actually following
- Make sure all traces that we actually take are in
- Also make sure the traces that we donot take are not in
- i.e. the aim is to get the exact model that is necessary and sufficient to represent the present process.

Classical Petri Nets

- A place can contain 0 or more tokens (more than 1)
  - The state then needs to mention the count of tokens held in places
    - e.g. {1 p1, 2 p3, 4 p4}
    - This state is different from {2 P1, 2 P3, 4 p4}

Places with multiple tokens



Build a state space

Initial marking: {P1}