Petri Net Modelling

Petri nets

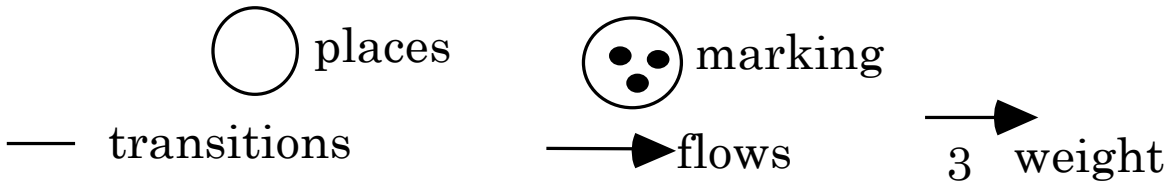
- Formalism for specifying systems that contain parallel or concurrent activities
- Defined by a quadruple (P,T,F,W), Where,

  P: finite set of places

  T: finite set of transitions

  F: flow relation ($F \subseteq \{P \times T\} \cup \{T \times P\}$)

  W: weight function (W: F → N – {0})



Some properties

(1) $P \cap T = \emptyset$

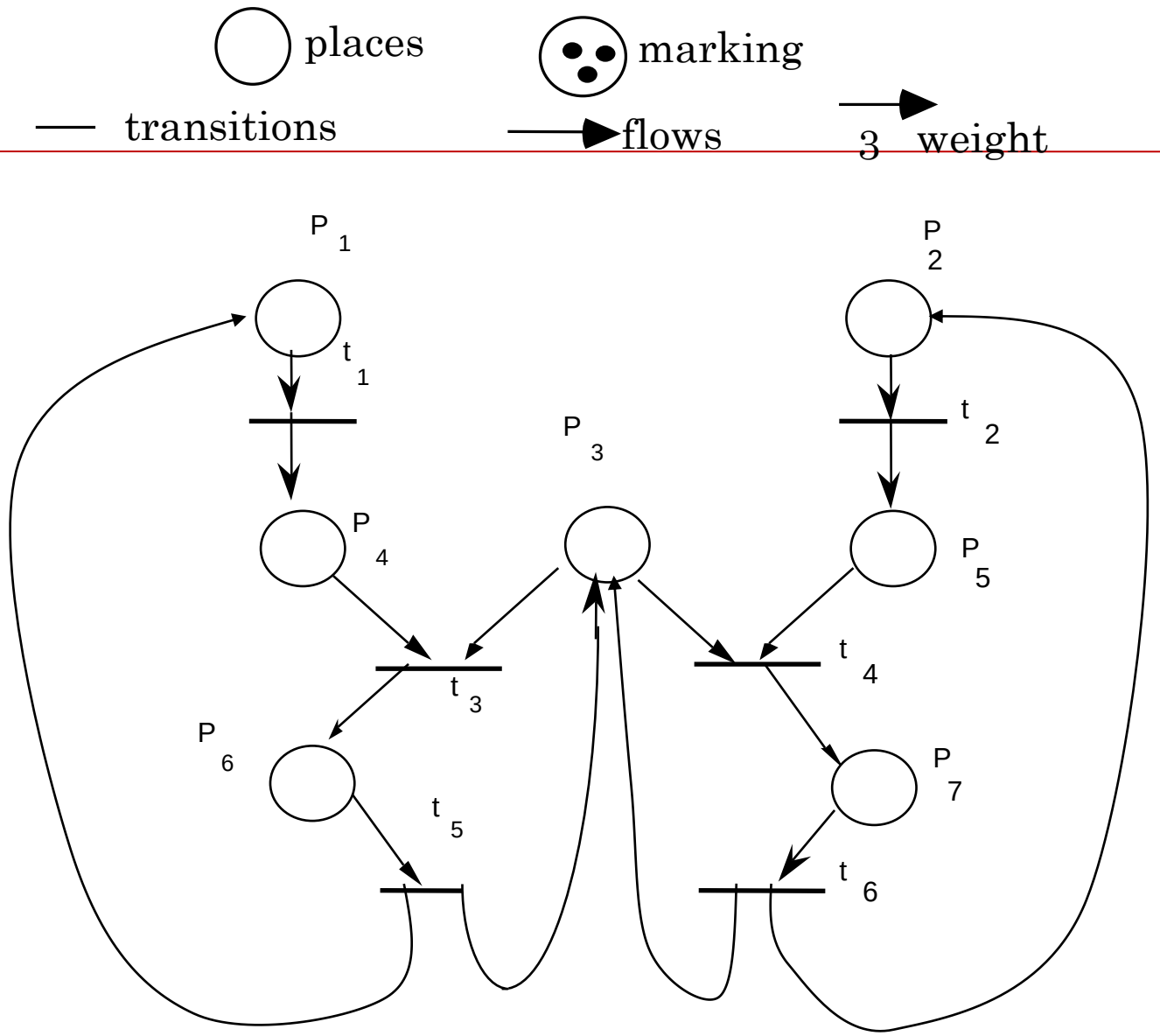
(2) $P \cup T \neq \emptyset$

(3) $F \subseteq (P \times T) \cup (T \times P)$

(4) $W: F \rightarrow N-\{0\}$

Default value of W is 1

(5) State defined by marking: $M: P \rightarrow N$



- Place
- transitions
- input place
- output place
- marking
- weight

Semantics

- Transition t is enabled iff
  - $\forall p \in$ t's input places, M(p) $\geq$ W(<p,t>)
- t fires: produces a new marking M' in places that are either t's input or output places or both
  - if p is an input place: M'(p) = M(p) - W(<p,t>)
  - if p is an output place: M'(p) = M(p) + W(<t,p>)
  - if p is both an input and an output place:

    M'(p) = M(p) - W(<p,t>) + W(<t,p>)

Nondeterminism

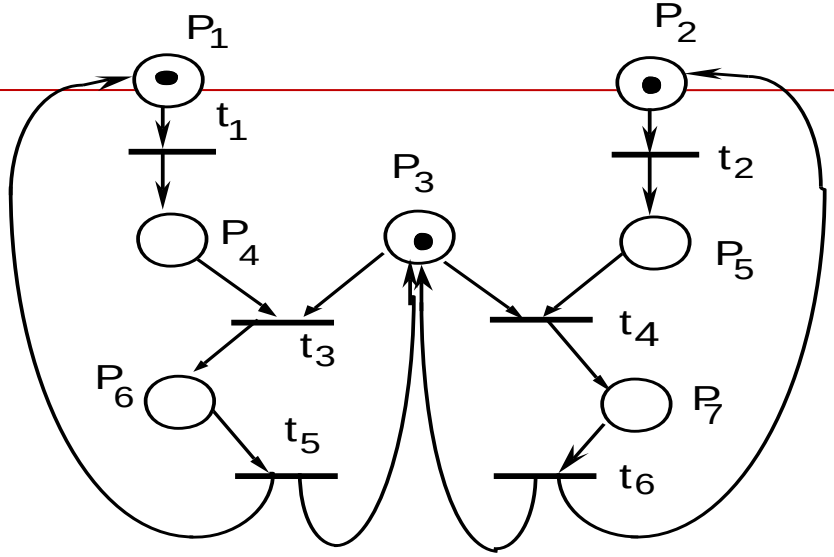
- Any of the enabled transitions may fire
- Model does not specify which fires, nor when it fires

If the token provides input to more than one transition, the network is non-deterministic, and the token at that place may trigger a fire at either of the destination transitions that are enabled.

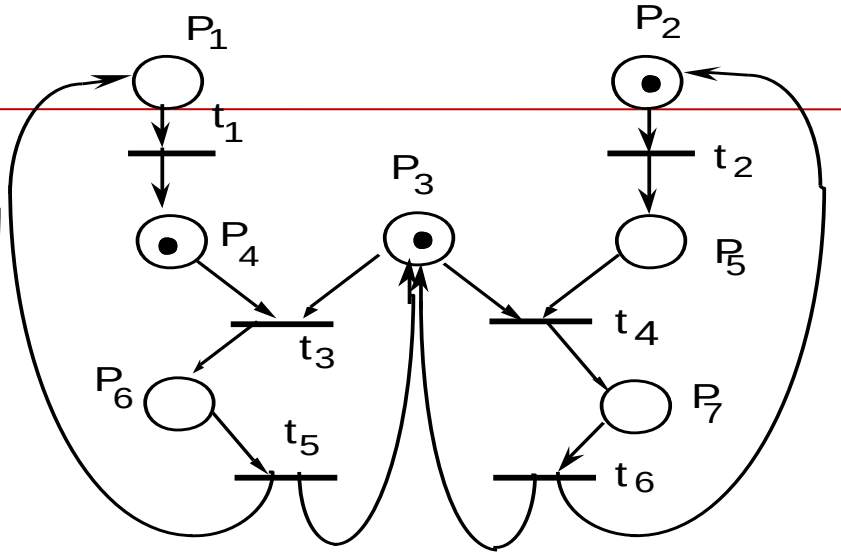
Modeling with Petri nets

- Places represent distributed states
- Transitions represent actions or events that may occur when system is in a certain state
- They can occur as certain conditions hold on the states

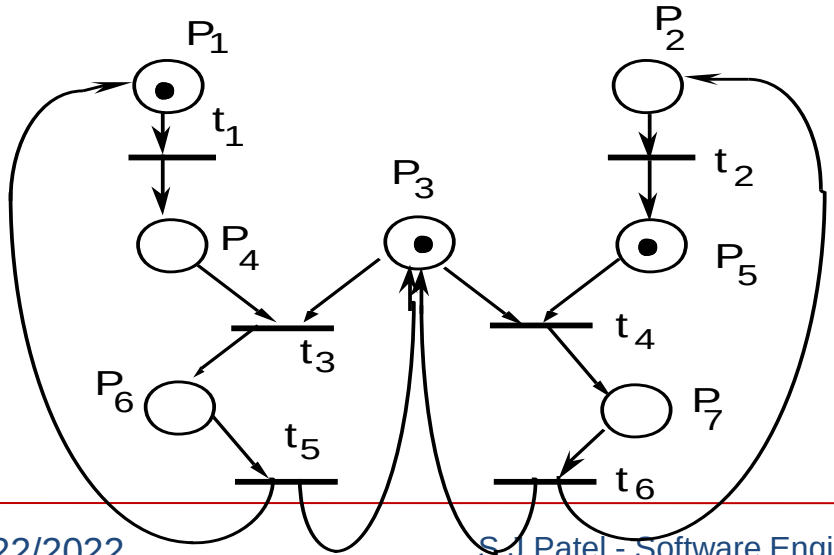
after (a) either (b) or (c) may occur, and then (d)



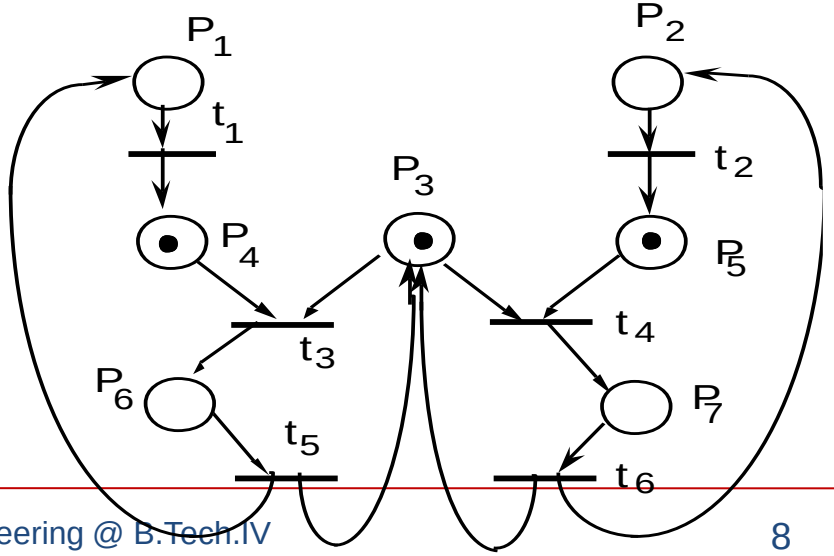
(a)



(b)



(c)

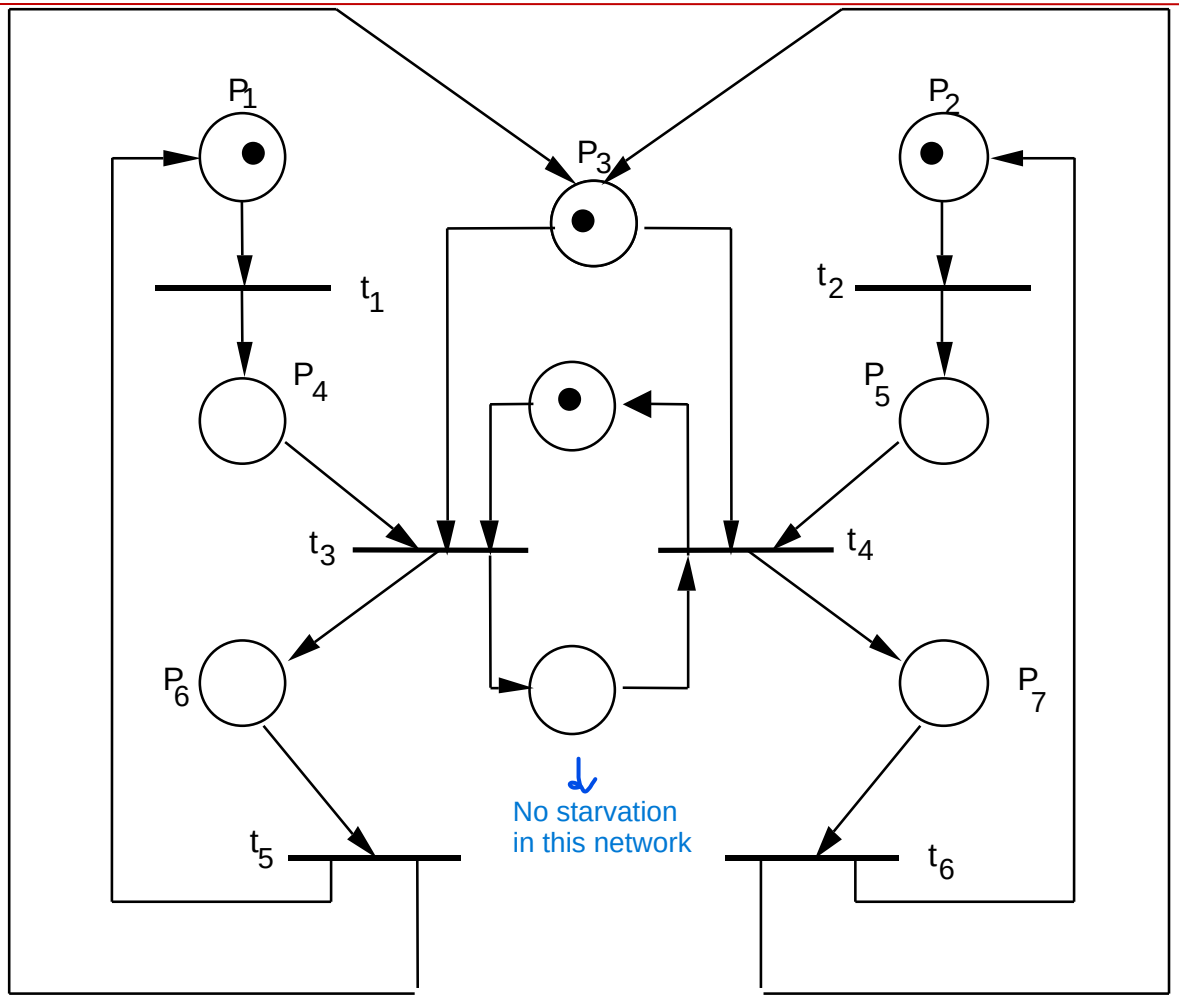


(d)

Common cases

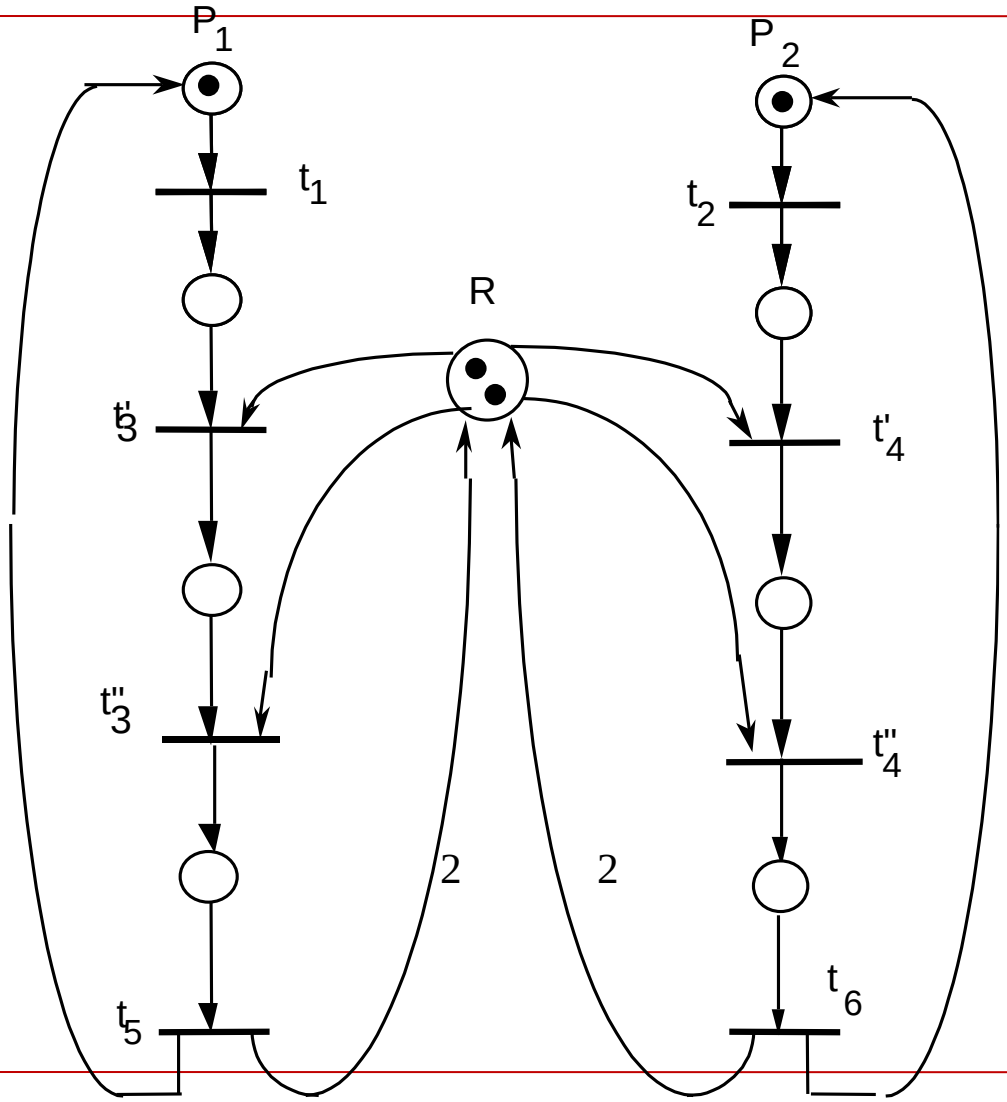
- Concurrency
  - two transitions are enabled to fire in a given state, and the firing of one does not prevent the other from firing
    - see t_1 and t_2 in case (a)
- Conflict
  - two transitions are enabled to fire in a given state, but the firing of one prevents the other from firing
    - see t_3 and t_4 in case (d)
    - place P_3 models a shared resource between two processes
  - no policy exists to resolve conflicts (known as *unfair* scheduling)
- Firing Sequence
  - $<t_1, t_2, t_3, \ldots, t_n>$

How to avoid starvation



imposes alternation

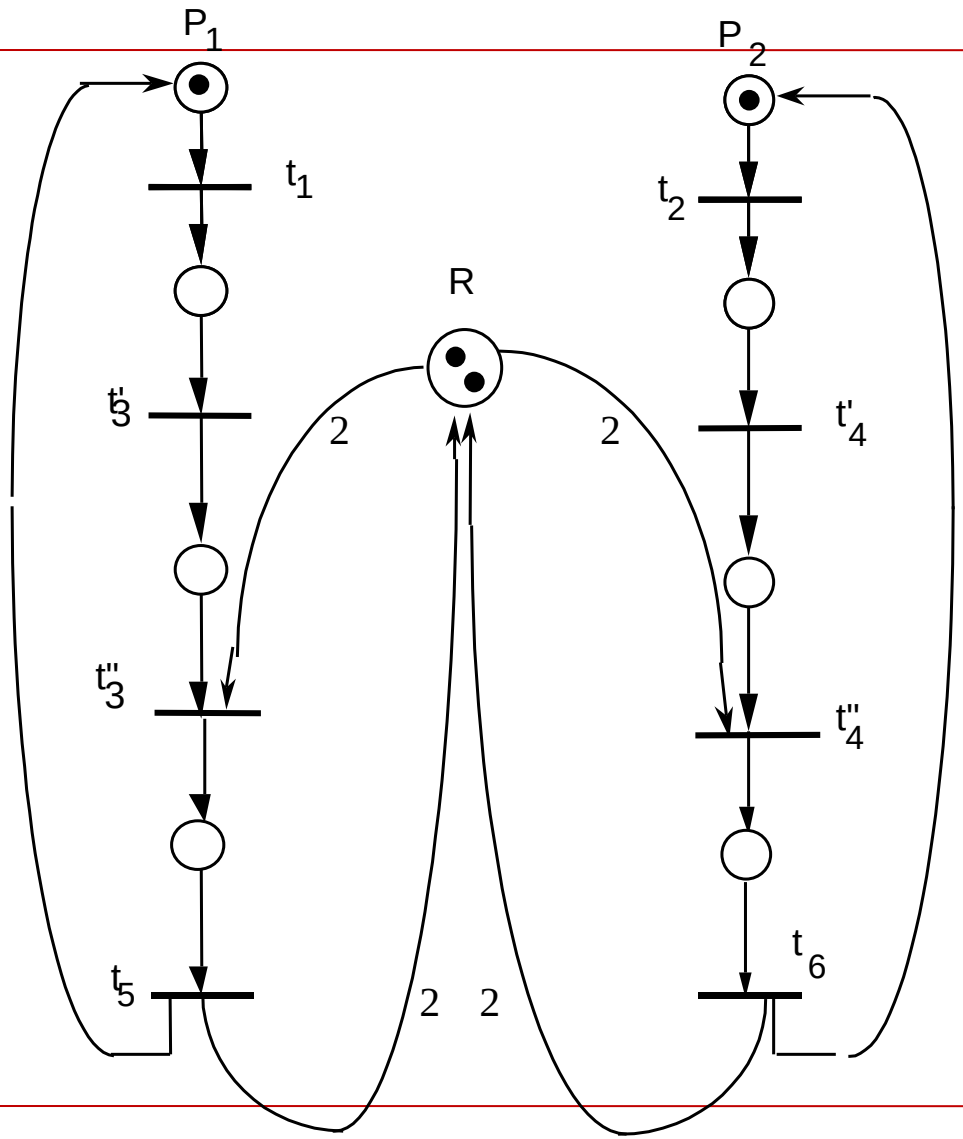
Example



- Is it conflict free?
- Deadlock free? (live)

Deadlock there
Starvation is there
Conflict is there

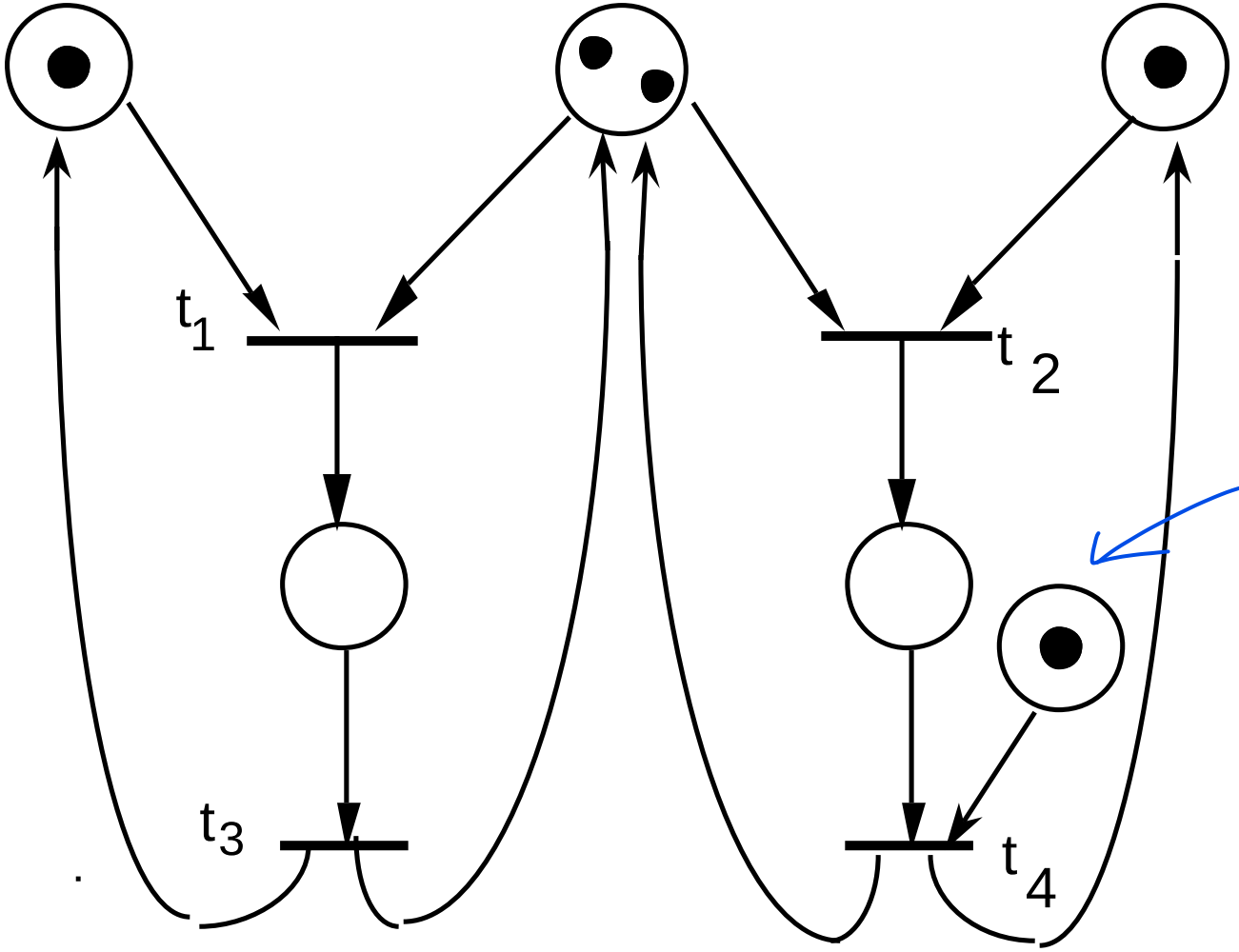
Example



- Is it live?
- Starvation?

Starvation is there
Conflict is there
Deadlock is not there

Example

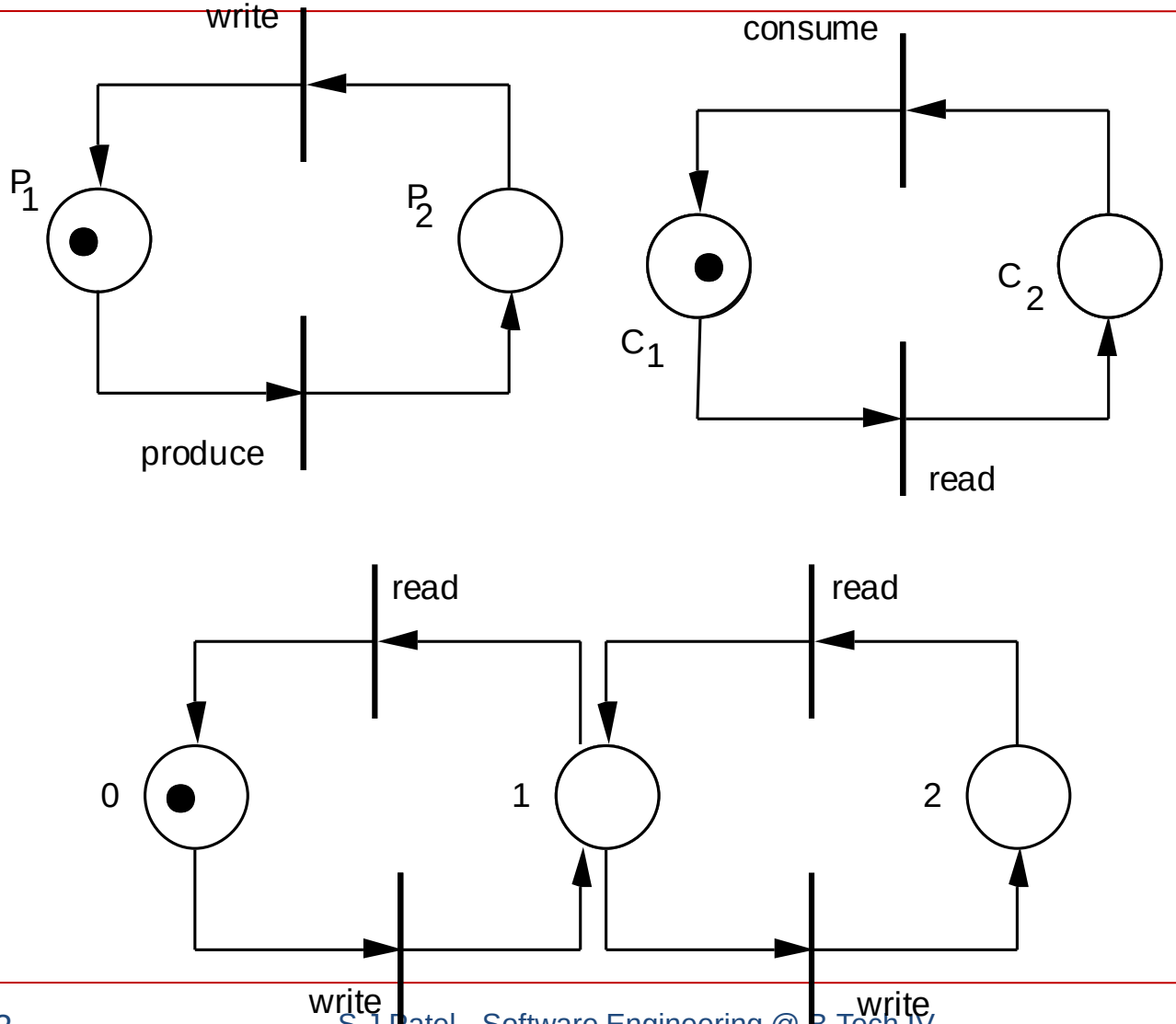


- Is it live?
- Starvation?

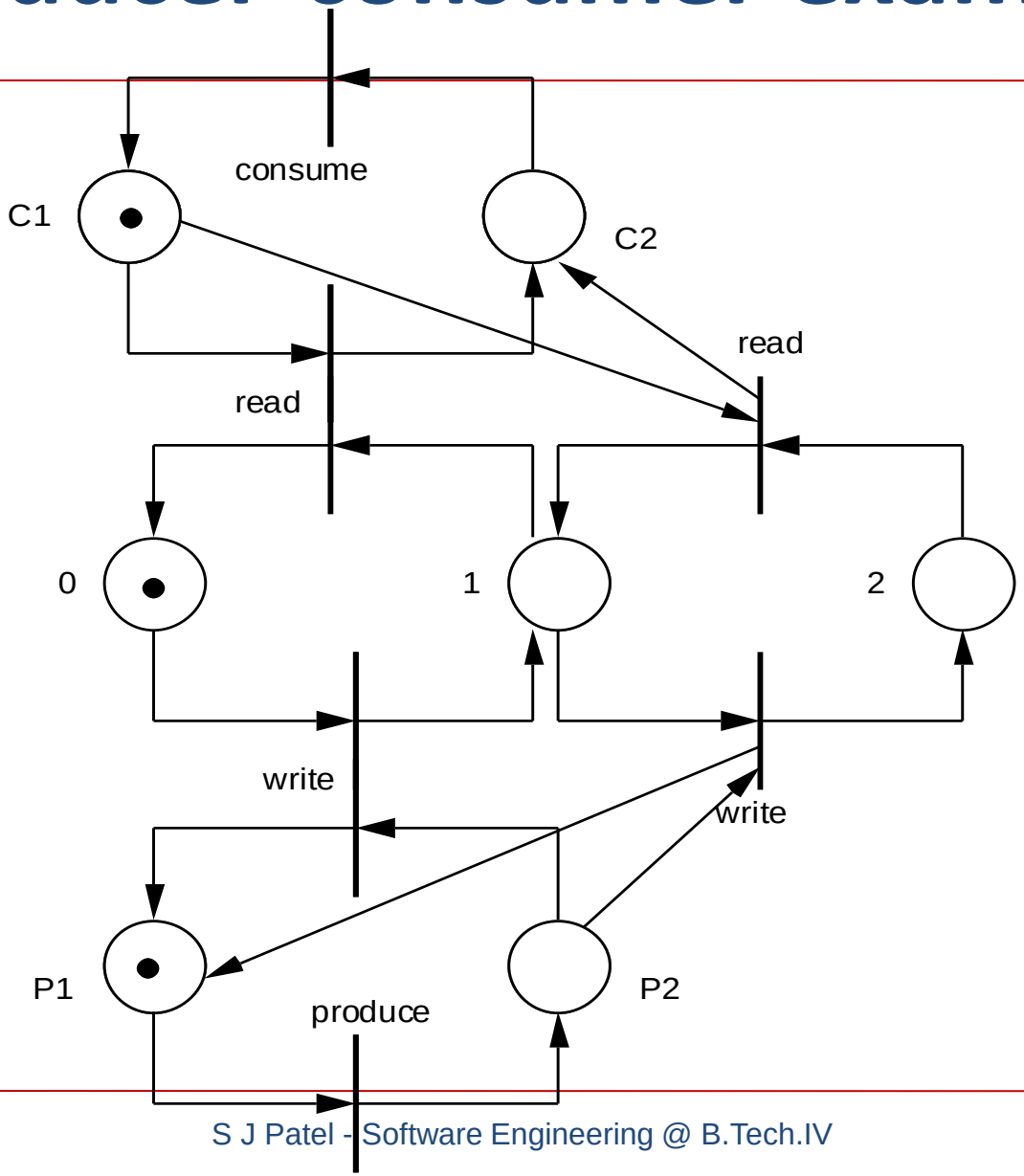
No conflict

This produces deadlock and starvation for right side system

Producer-consumer example



Producer-consumer example



Some Examples

Petri net model for the Traffic signal light.

Case 1: Single signal light

Case 2: Two signal lights that are not allowed to signal green at the same time

Some Examples (cont.)

Petri net model for the Dining Philosopher Problem

Case 1: Single Philosopher

Case 2: Four philosophers seated at round table with four forks. Each philosopher requires two forks at a time in order to eat.

Some Examples (cont.)

Petri net model for the Elevator

Case 1: Consider the elevator installed in the building with three floors. Draw a simple model that only shows movement of the elevator (i.e. upward or downward) in a certain state of the system.

Case 2: Now, draw a model for a person using an elevator.

Some Examples (cont.)

Petri net model for the Computer Program

Model the following program;

```
A=1;
B=2;
C=3;
A=A+1;
C=B+C;
B=A+C;
```

Limitations and extensions

- Control oriented models
- Tokens are anonymous
  - Useful only for analyzing flow of messages within communication network
- Not possible to specify selection policy between different transitions that are enabled
  - Can not prevent starvation for the sample petri net studied
- Timing issues are not handled
  - Required for real time systems
  - Is firing sequence < t1, t2, t3, t5, t4> feasible, if t1,t3, t5 each takes 1 sec. and t2 takes 5 sec. to complete????

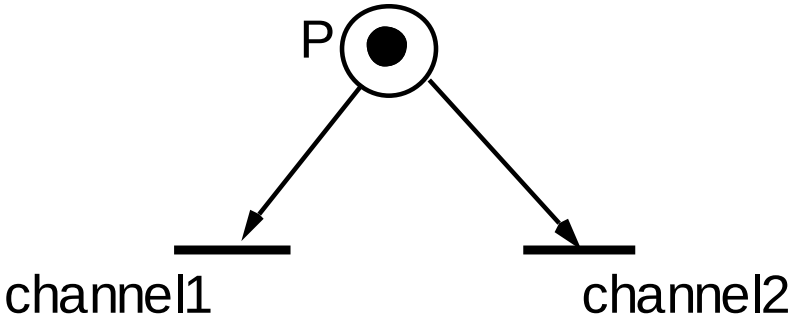
Limitations and extensions

e.g.

Specification of a system in which message is to be forwarded through one of two different channels:

$channel_1$ is selected if message is well-formed;

$channel_2$-the error channel-is selected if message is incorrect



Extension 1: assigning values to tokens

- Tokens are modified to carry a value of an appropriate type
- Transitions have associated predicates and functions
- Predicate refers to values of tokens in input places selected for firing
- Functions define values of tokens produced in output places

Firing rules

- A transition with k input places and h output places is enabled iff,

  There exist k-tuple of tokens-one for each input place- such that the predicate associated with the transition is satisfied by the values of the tokens of the tuple

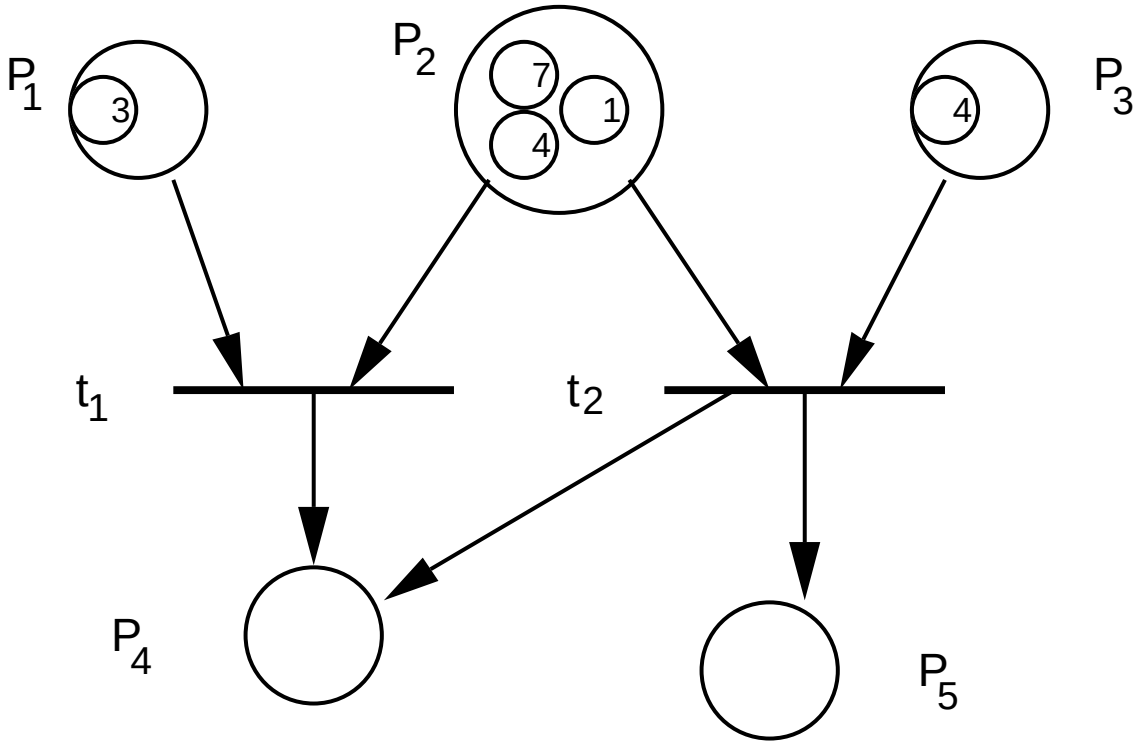
  This tokens are together called ready tuple.

- Predicate is evaluated on exactly one token

Firing rules

- When enabled transition fires,
  - The cancellation of all tokens that belong to a ready tuple from the input places
  - The evaluation of h new token values on the basis of the values of the ready tuple by applying the function associated with the transition
  - The production of one token for each output place- the value of the token is computed by the function associated with the transition

Example



Predicate P2 > P1
and function
P4 := P2 + P1
associated with t1

Predicate P3 = P2
and functions
P4 := P3 – P2 and
P5 := P2 + P3 are
associated with t2

The firing of t1 by using <3,7> would produce the value 10 in P4. t2 can then fire using <4, 4>

Example

- *A message is to be forwarded through one of two different channels: Channel1 is selected if the message is well formed; channel2-the "error" channel –is selected if the message is incorrect. The message is well formed if it contains an even number of 1's. (i.e. if it has correct parity)*

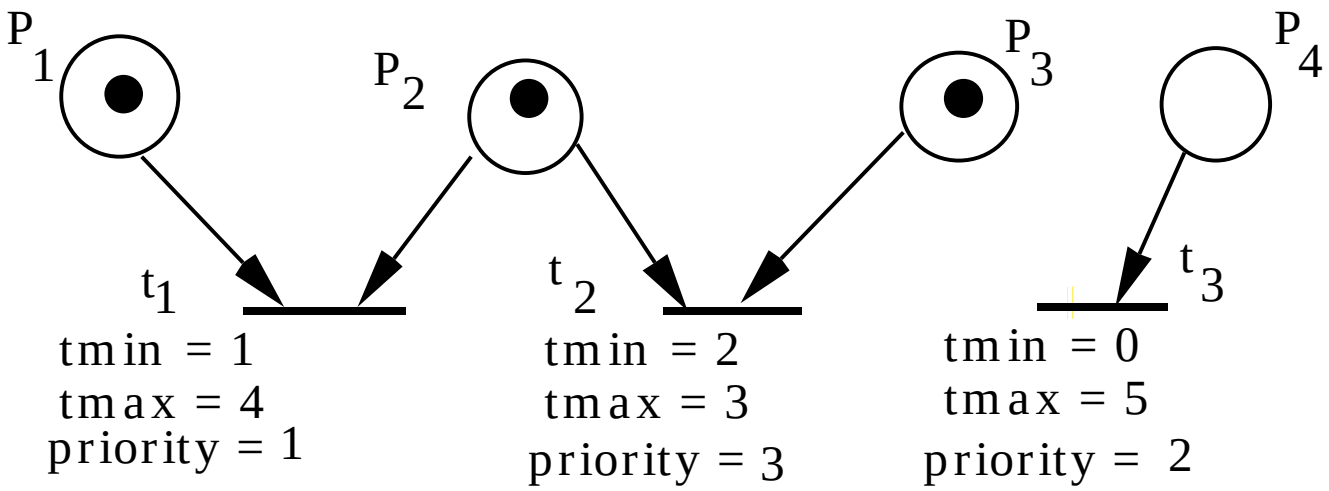
Extension 2 specifying priorities

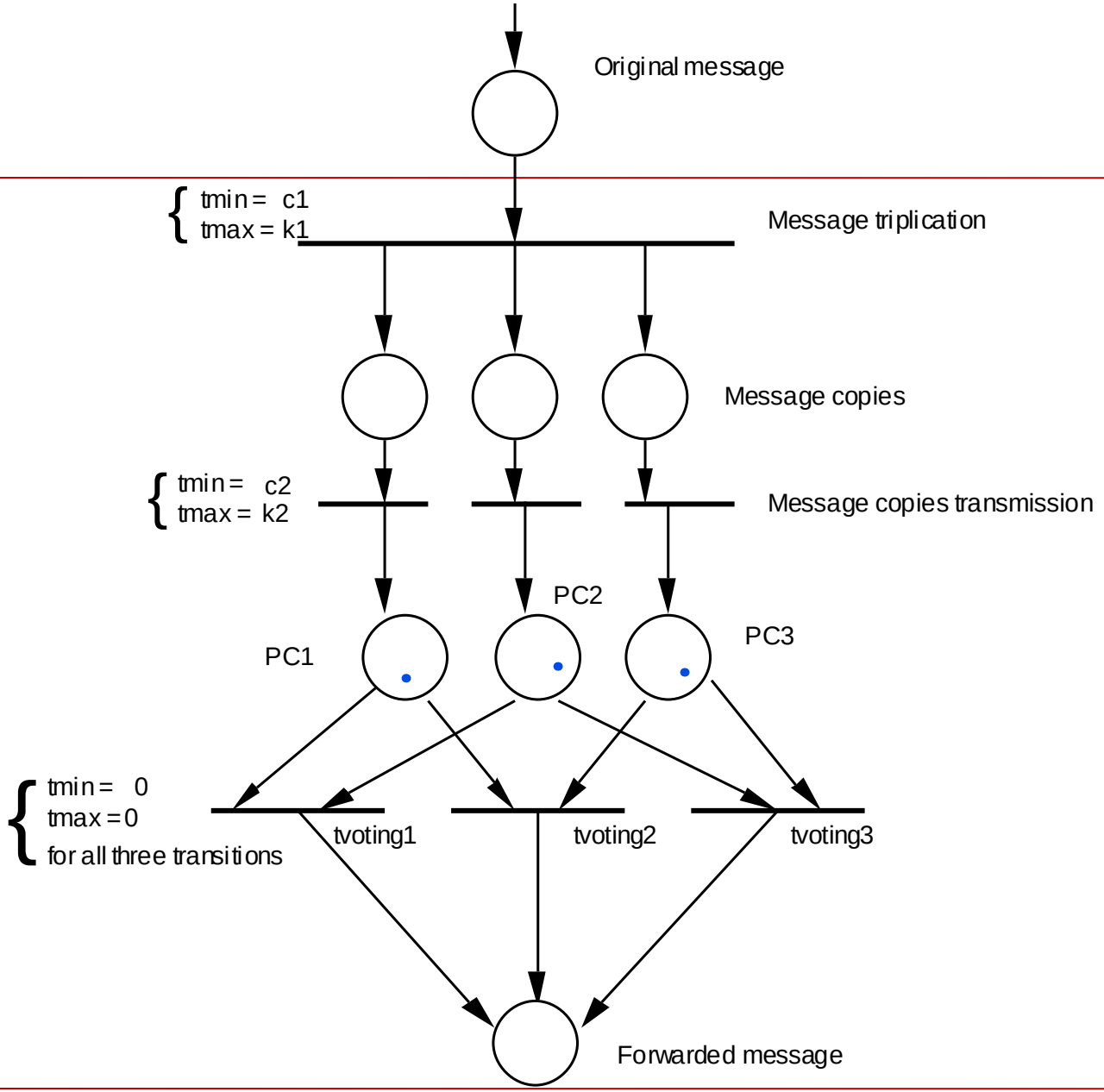
- A priority function pri from transitions to natural numbers:
- pri: T → N
- When several transitions are enabled, only the one with maximum priority are allowed to fire
- Static and Dynamic priority

Extension 3 Timed Petri nets

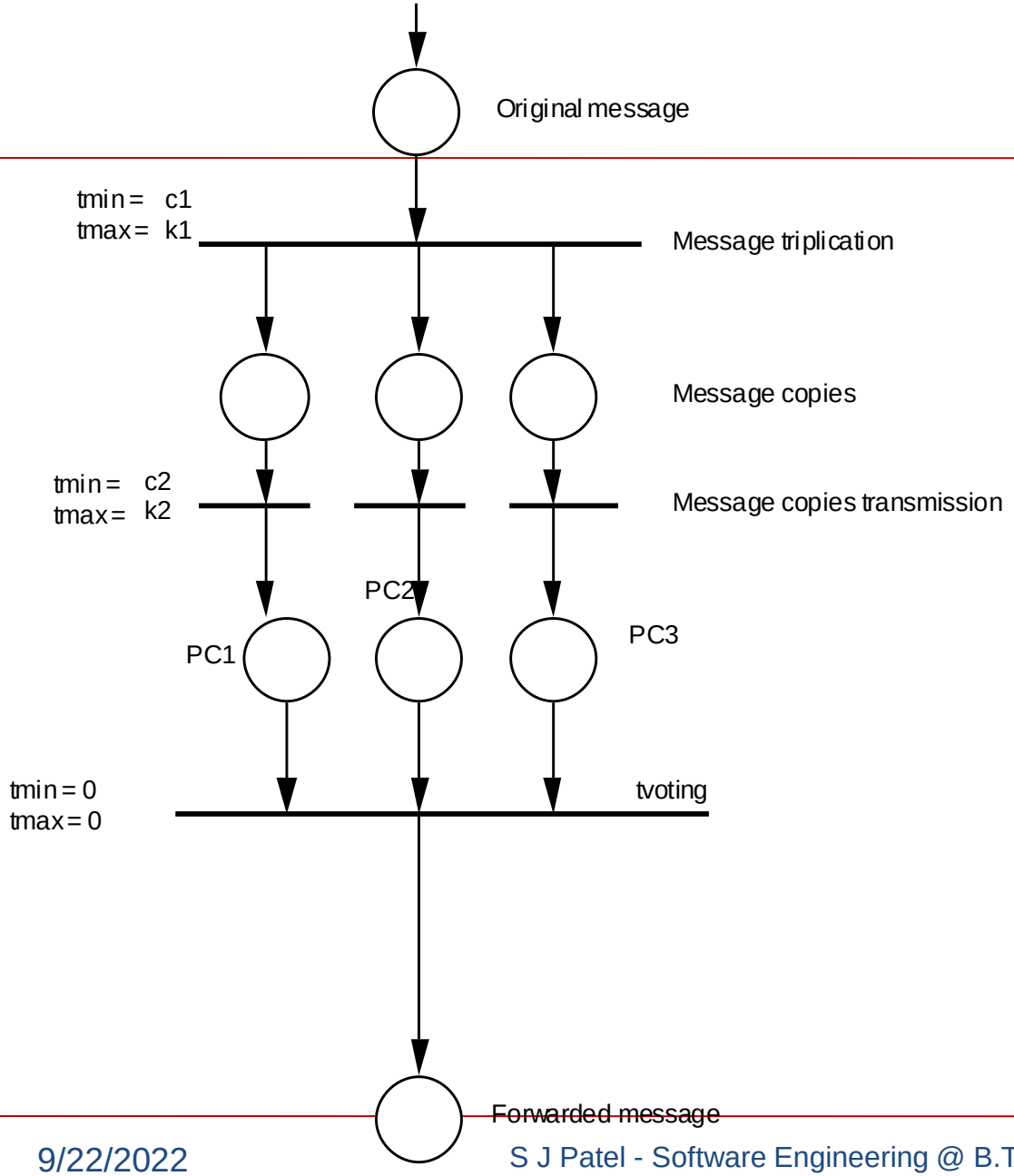
- A pair of constants $<t_{min}, t_{max}>$ is associated with each transition
- Once a transition is enabled, it must wait for at least t_{min} to elapse before it can fire
- If enabled, it *must* fire before t_{max} has elapsed, unless it is disabled by the firing of another transition before t_{max}

Example combining priorities and time





Case 1:
A message must be triplicated. The three copies must be forwarded through three different physical channels. The receiver accepts the message on the basis of a two-out-of-three voting policy



Case 2:

A message must be triplicated. The three copies must be forwarded through three different physical channels. The receiver waits until all three copies are arrived and then perform comparison.

Exercise

- Use the PN extension to describe a message dispatcher that works along the following lines: The dispatcher receives messages from two different channels and then checks the parity of each message. If the parity is wrong, it sends a NACK through reply channel (there is one such channel for each input channel); if the parity is right it places the received message into a buffer. The buffer may store 10 messages. When the buffer is full the dispatcher sends the whole content of the buffer to a processing unit through another channel. No message can be placed into a full buffer.

Case study

- An n elevator system to be installed in a building with m floors
- Natural language specs contain several ambiguities
- Formal specification using PNs removes ambiguities
- Specification will be provided in a stepwise fashion
- Will use modules, each encapsulating fragments of PNs which describe certain system components

Informal Specs

- An n-elevator system for an m-floored apartment
  - each elevator has a set of m buttons one for each floor
    - each button lights up when pressed and cause the elevator to visit the corresponding floor
    - Lights switched off when the elevator visits the floor
  - each floor other than the ground floor and top floor has two buttons; one to request an up elevator and one to request a down elevator
    - These buttons light up when pressed.
    - The lights switch off when the elevator visits the floor and either moving in desired direction or has no outstanding requests.
    - In latter case if both floor buttons are pressed only one is cancelled. The algorithm to decide which to service first should minimize the waiting time for both the requests

Informal Specs

- when no requests to service, the elevators remains at the final destination with its doors closed and await for further requests
- all requests must be serviced eventually
  - requests from elevators for floors
  - requests from floors for elevators
- emergency button within the elevator

From informal specs…

- Focus on point 2

1. *Each floor other than ground floor and top floor has two buttons, one to request a up elevator and one to request a down elevator.*
2. *These buttons light up when pressed. The lights switch off when the elevator visits the floor and either is moving in the desired direction or has no outstanding requests.*
3. *In latter case, if both floor-request buttons are pressed, only one is cancelled. The algorithm to decide which to service first should minimize the waiting time for both requests.*

From informal specs…

- each floor other than the ground floor and top floor has two buttons….
  - What could be the possible interpretations???

From informal specs…

"The illumination (in the floor buttons) is cancelled when the elevator visits the floor and is either moving in the desired direction, or has no outstanding requests..."

- 2 different interpretations (case of up call)
  - switch off as the elevator arrives at the floor from below (obvious restrictions for 1st and last floor)
  - switch off after the elevator starts moving up

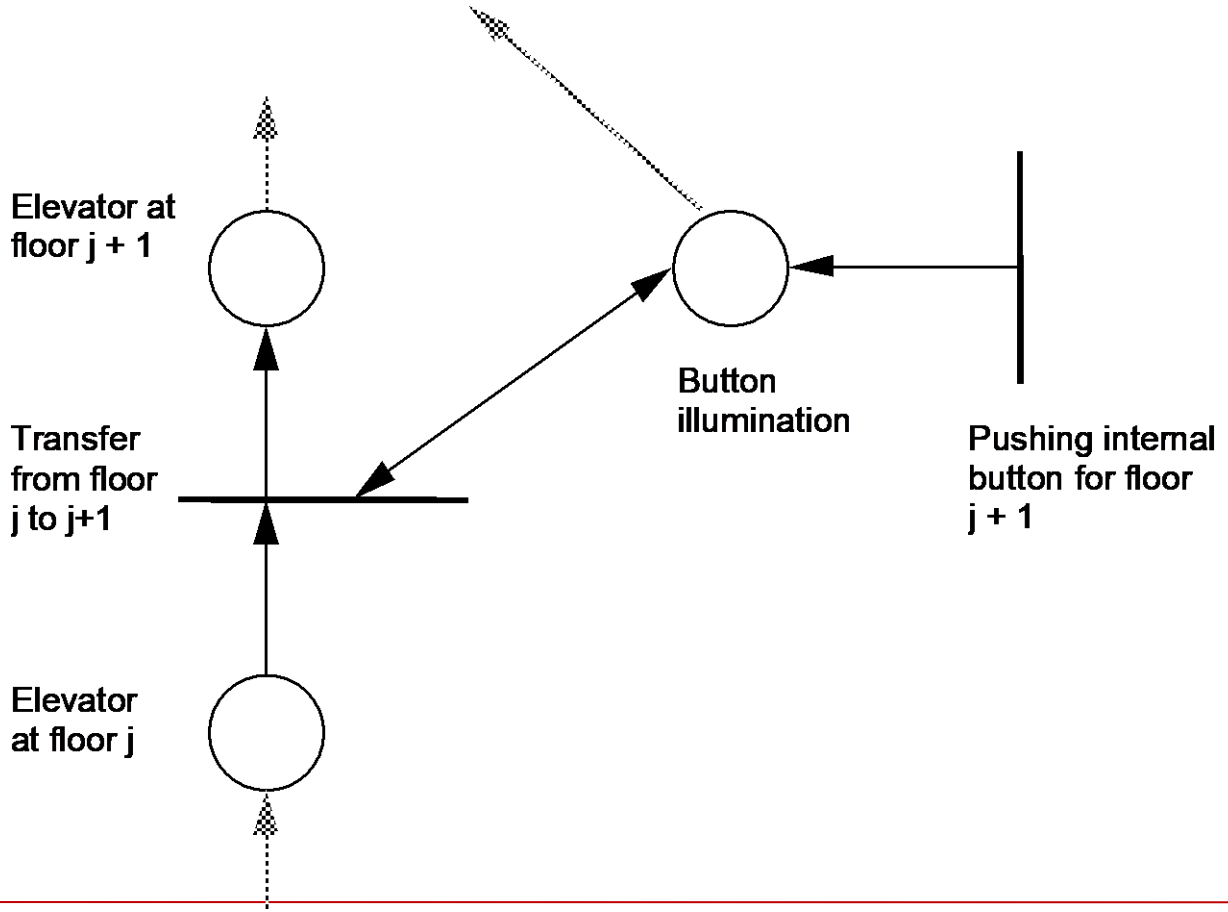
…more analysis of informal specs

"The algorithm to decide which to service first should minimize the waiting time for both requests."

what does this mean?

- in no other way can you satisfy either request in a shorter time
  - but minimizing for one may require longer for the other
- the sum of both is minimal
  - why the sum?

Initial sketch of movement



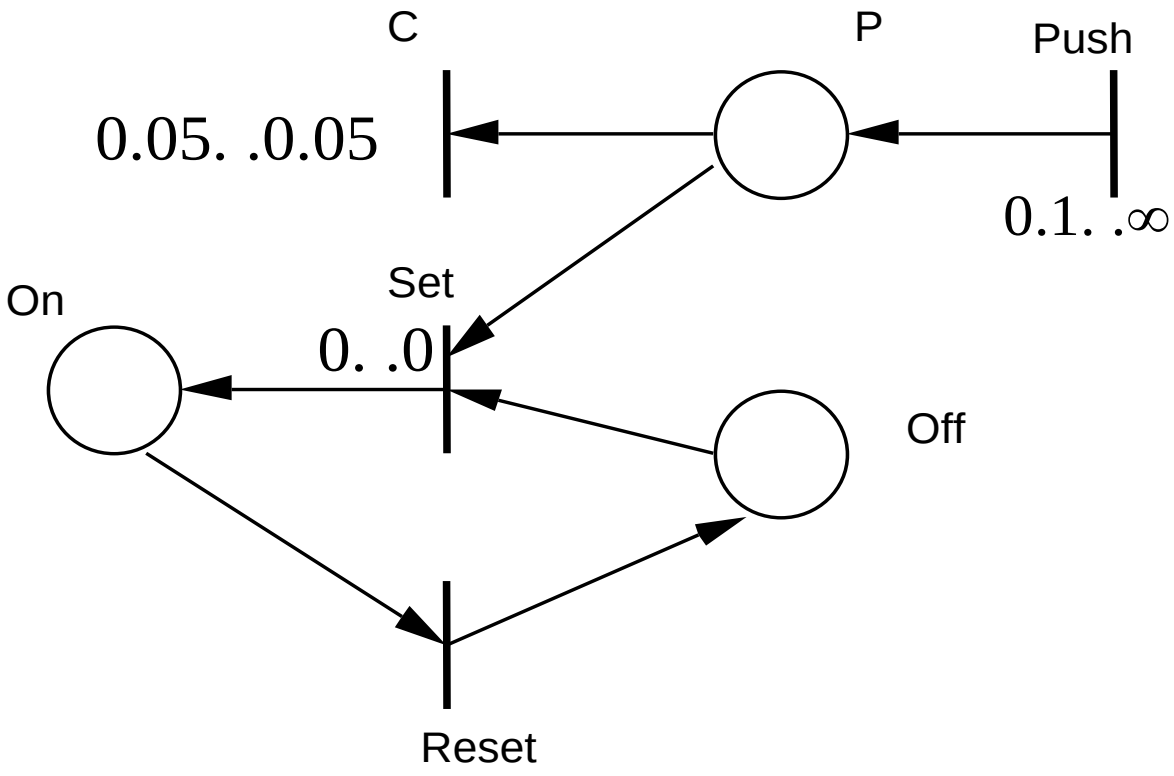
Points to be noted for the above scheme

- Limitations of the above scheme
  - incomplete specs…internal/external buttons
  - difficult to generalize when m, n are large nos
  - at some places the specs are wrong, too
- Modular design approach to specs…
  - n spec modules of type ELEVATOR
    - ELEVATOR_POSITION submodule
    - ELEVATOR_BUTTONS submodule
      - m further submodules of type BUTTON

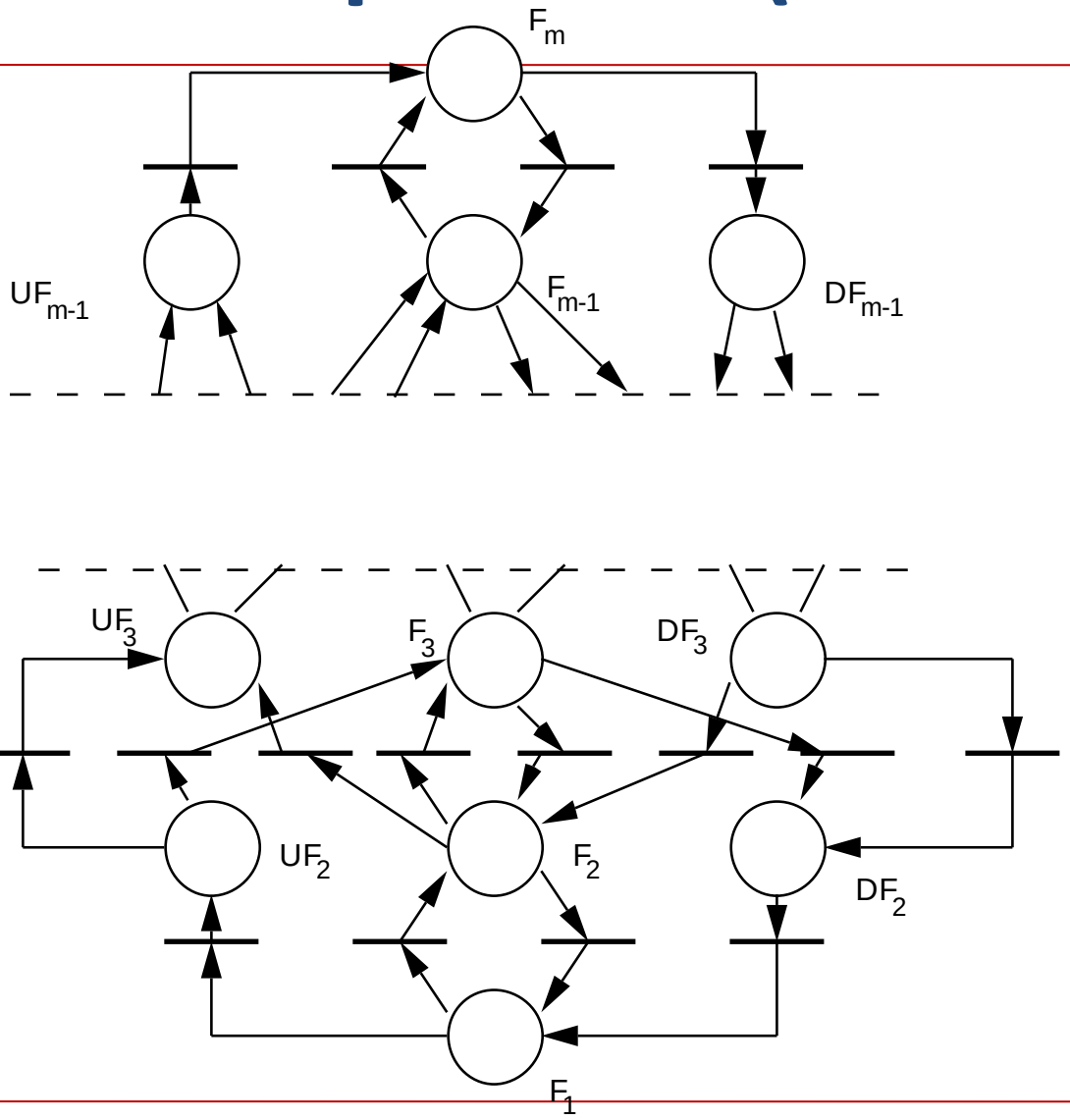
Points to be noted for the above scheme (contd…)

- m spec modules of type FLOOR
  - two further submodules of type BUTTON

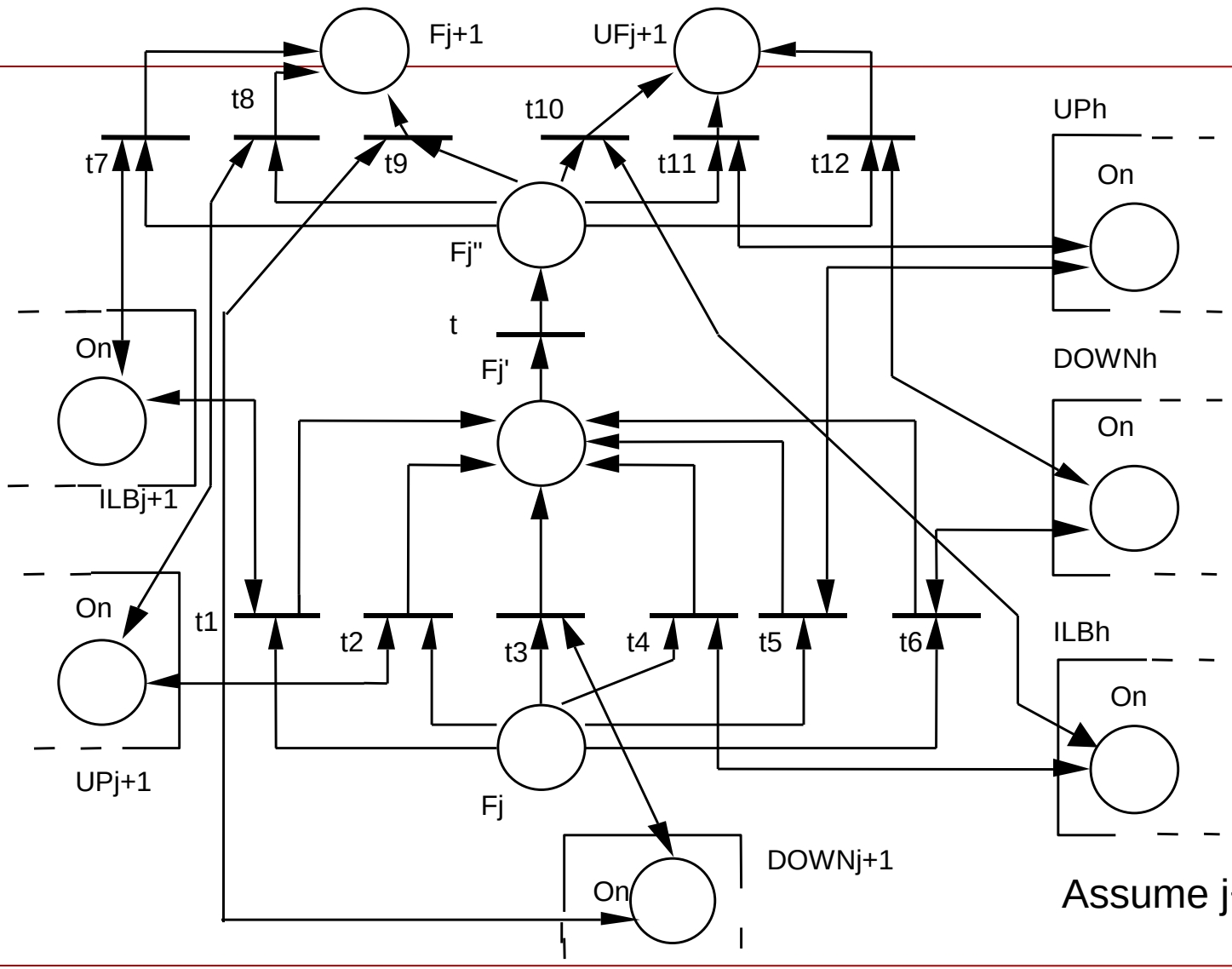
Button module



Elevator position (sketch)

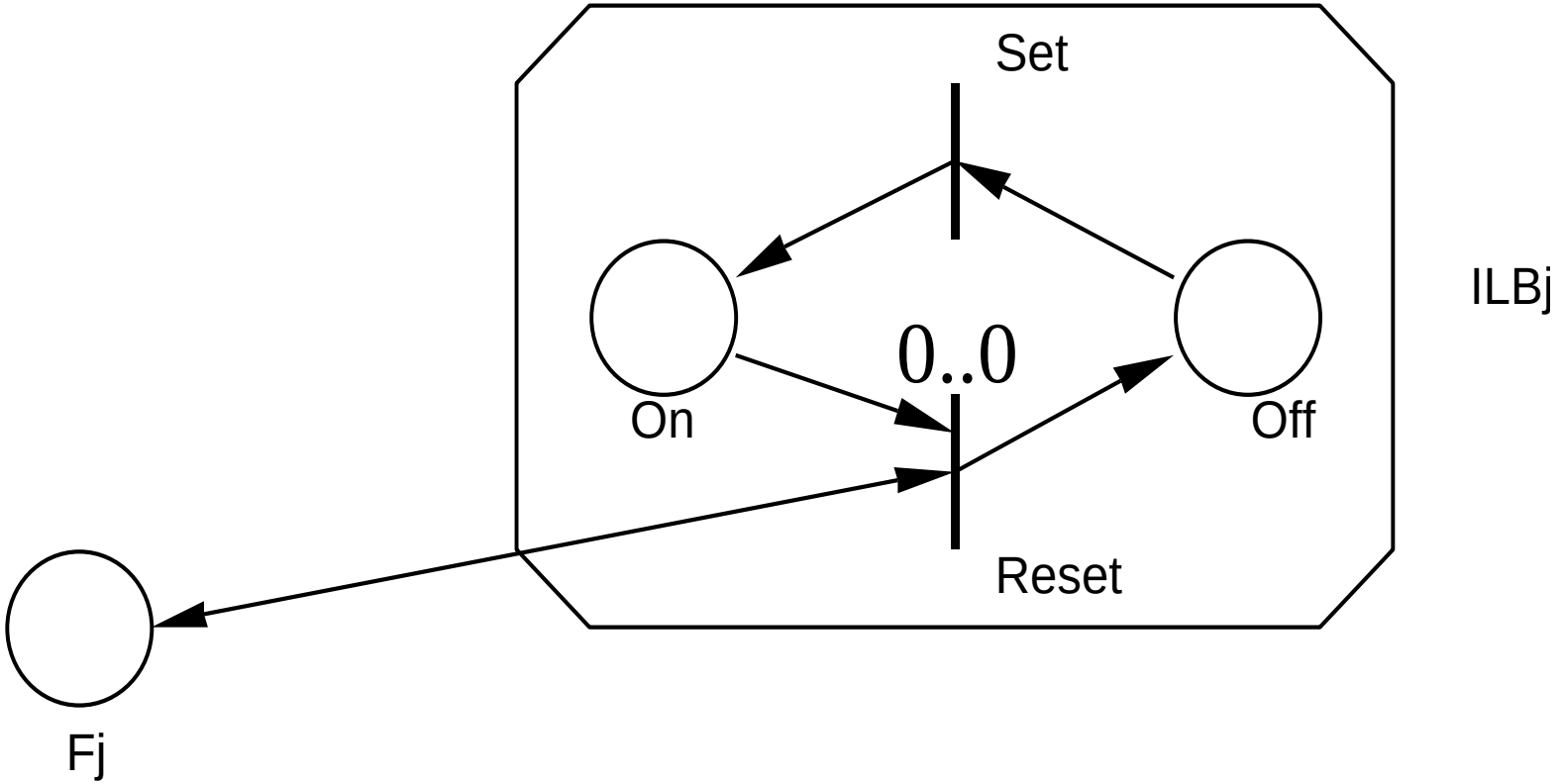


More precise description of elevator position

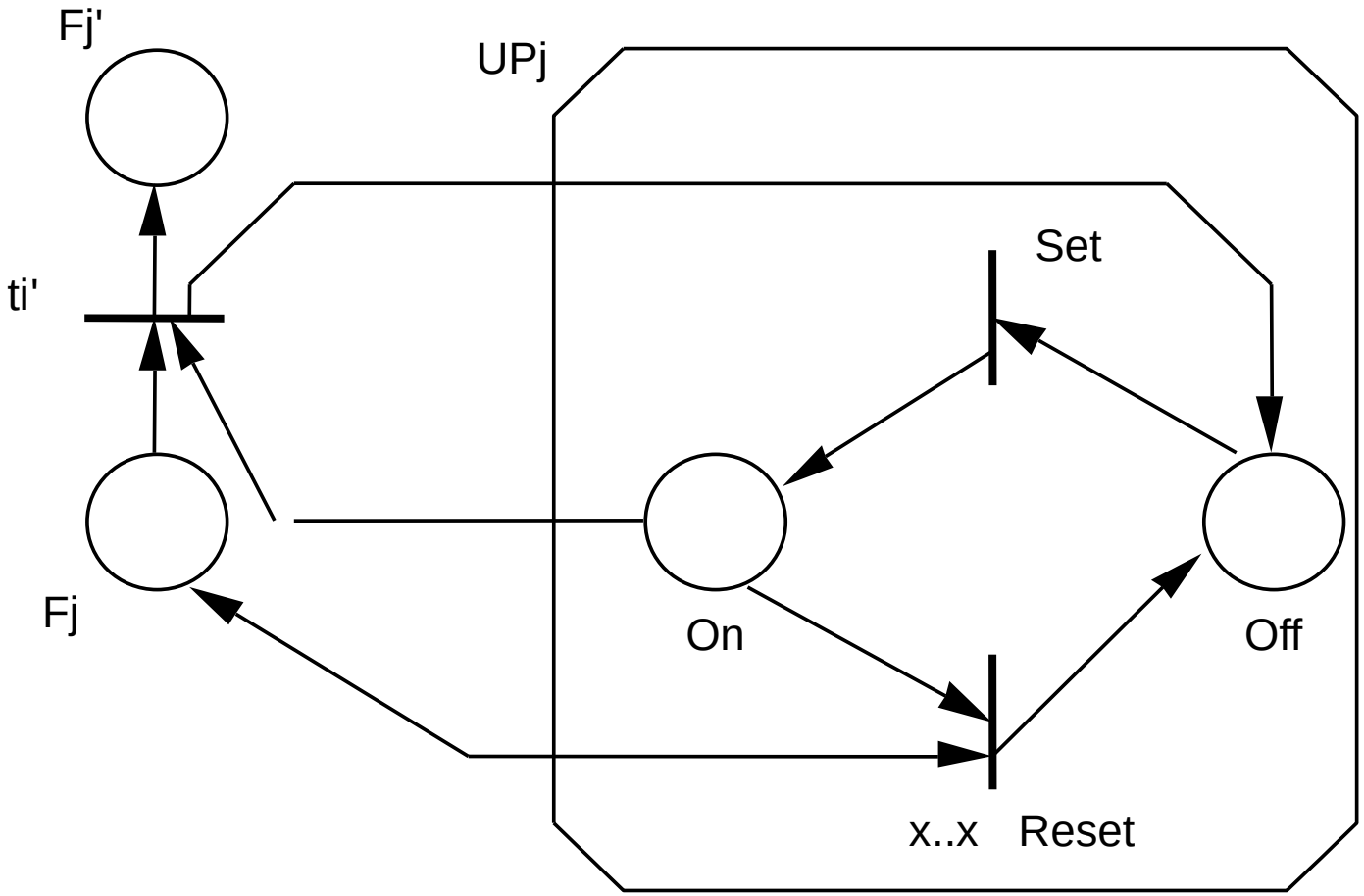


Assume j+1<h≤m

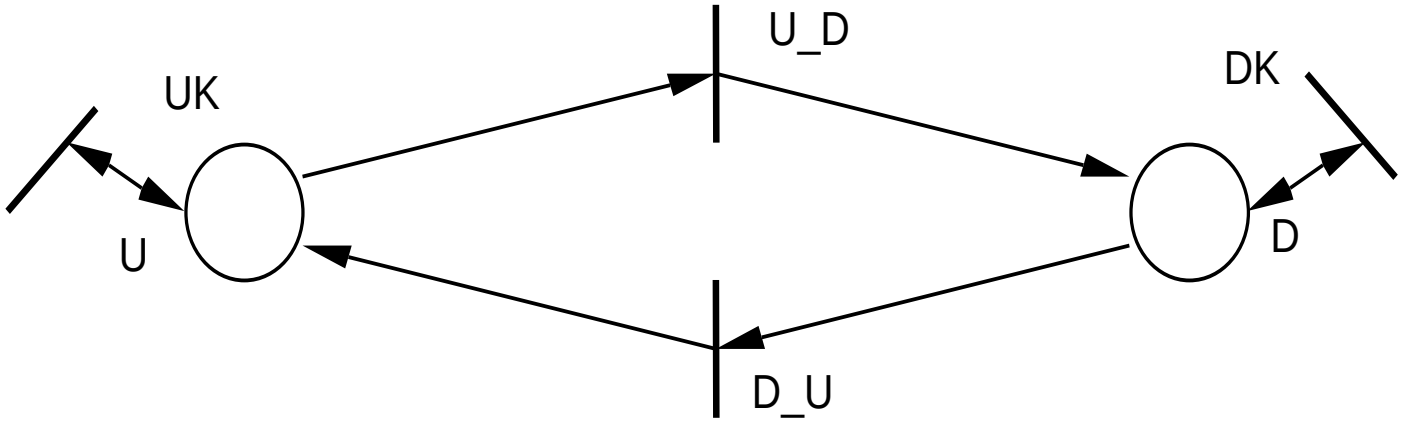
Switch internal button off



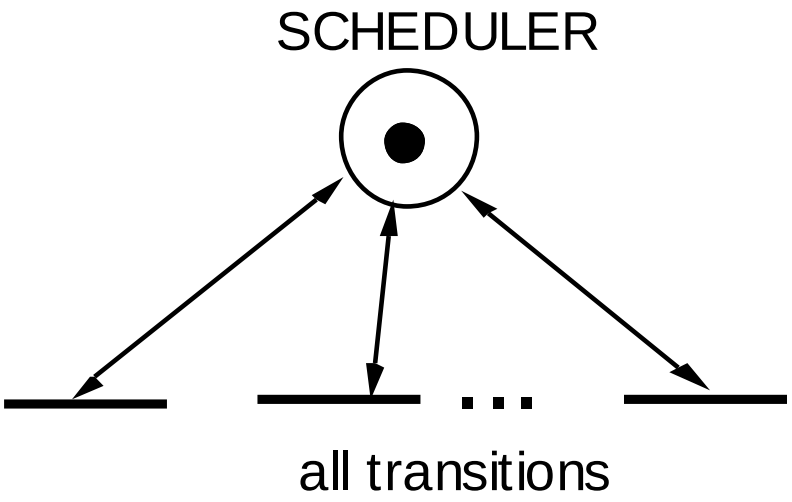
Switch external button off



Specifying policy



A general scheduler



Tutorial questions

1. Consider an initial marking where an elevator is at floor 1 and all of the internal and external buttons are off. Now assume that somebody enters the elevator and pushes the internal button 2.

   Simulate the movement of the elevator with respect to the internal call using transition sequence. (start with transition PUSH of BUTTON module of floor 2 i.e. ILB_2)

2. Now, consider an initial marking where an elevator is at floor 3 and the internal button of floor 6 and an external button of floor 10 are on.

   Simulate the movement of the elevator with respect to the above calls using appropriate transition sequences

Tutorial questions…

3. Use the PN to model office organizations and automation. You should describe all relevant activities that occur in an office (producing documents, computing invoices and salaries etc.), the data such activities involve, the way the activities operate on the data, and the precedence relations among different activities (e.g. one can not mail a letter if it has not been written). You do not need to produce a long list of activities and data stores. Rather, you should focus attention on a few of them and analyze them in some depth.

Tutorial questions…