Formal Method in Software Engineering (SE-313)

Course Teacher
Assistant Professor
Engr. Mustafa Latif

Introduction to Petri Net

Petri Net

Petri Net Applications

- Software: design, specification, simulation, validation, and implementation
- Manufacturing, production, and scheduling systems
- Sequence controllers (Programmable Logic Controller, PLC)
- Communication protocols and networks

3

Petri Net

Petri Nets

- A Petri net is an abstract, formal model(graphical and mathematical) of information flow.
- Petri nets make it possible to model and visualize behaviors with parallelism, concurrency, synchronization and resource sharing.
- Powerful methods for describing and analyzing the flow of information and control in systems
- Petri nets represent computer systems by
  - providing a means to abstract the basic elements of the system
  - and its informational flow using only four fundamental components.

4

Petri Net

A Petri net is a bipartite directed graph composed of places and transitions. Petri nets are usually represented graphically according to the following conventions:-

- Places are represented by circles,(resources, conditions, buffers, locations)
- transitions by bars,(events, actions)
- input function by arcs directed from places to transitions,
- output function by arcs directed from transitions to places, and
- Tokens are dynamic objects used to track information and appear as solid dots within the circle of a place.
- The state of a Petri net, called the marking, is determined by the distribution of tokens over the places.
-

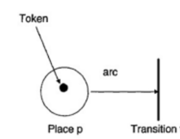


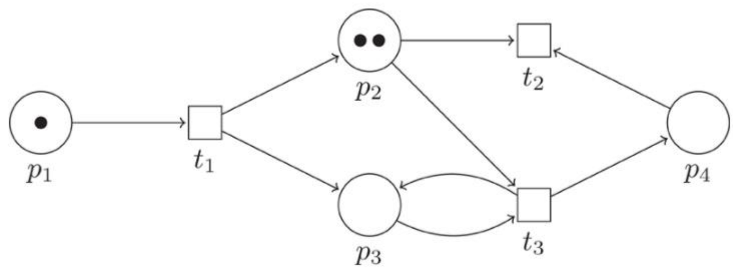
Fig 1: Basic Petri net components

Typical Interpretations of Transitions and Places

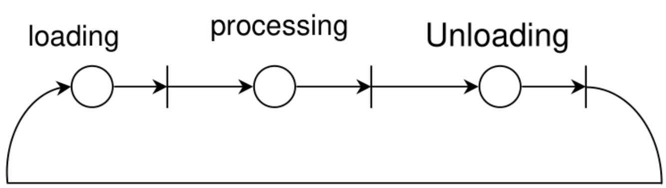
Input Places	**Transitions**	**Output Places**
Precondition	Event	Postcondition
Input Data	Computation Step	Output Data
Input Signal	Signal Processor	Output Signals
Required Resources	Task or Job	Released Resources
Conditions	Logic Clause	Conclusion
Buffers	Process	Buffers

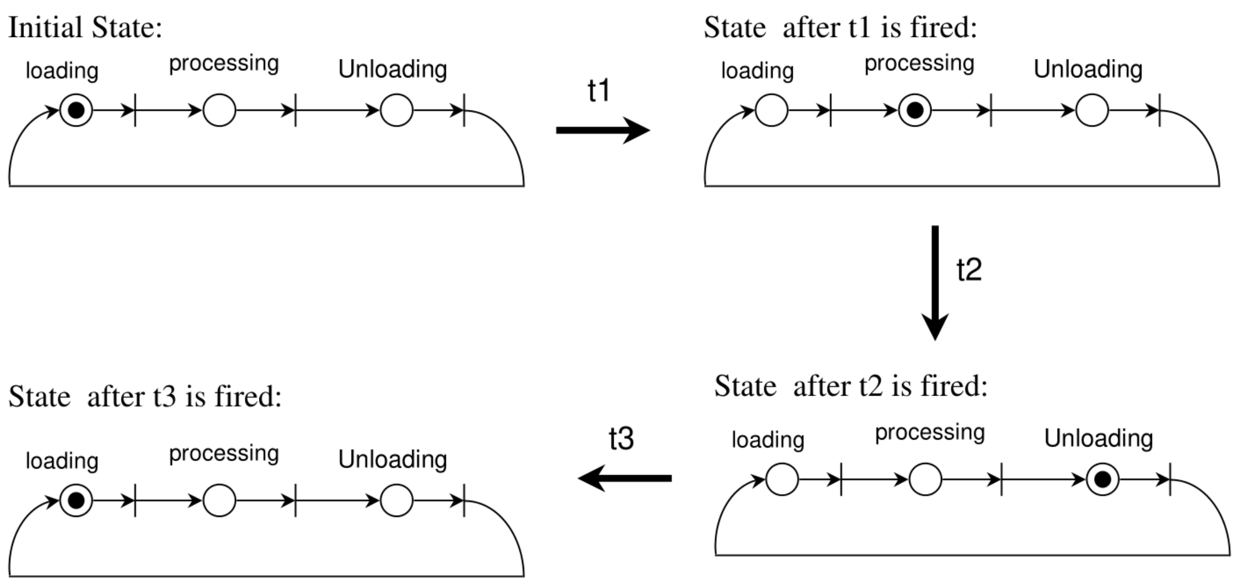
6

- **Example**:



- Enabling Rule:
  - A transition *t* is enabled if every input place contains at least one token
- Firing Rule:
  - Firing an enabled transition
    - removes one token from each input place of the transition
    - adds one token to each output place of the transition

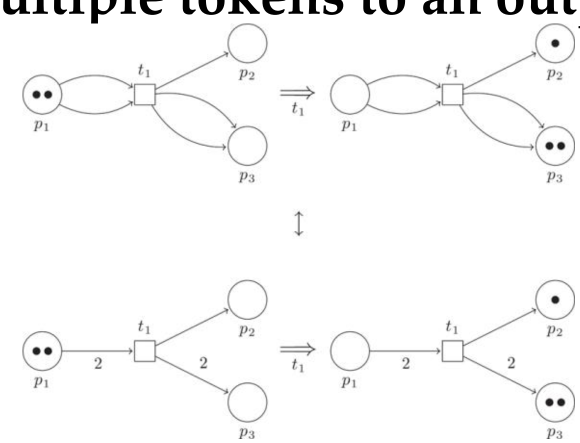




9

Multiplicity of Arcs:

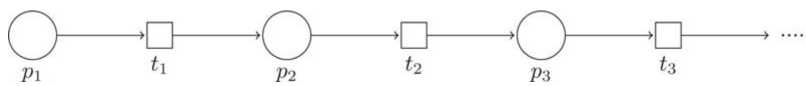
- Firing a transition may require consuming multiple tokens from an input place and may generate multiple tokens to an output place.
-



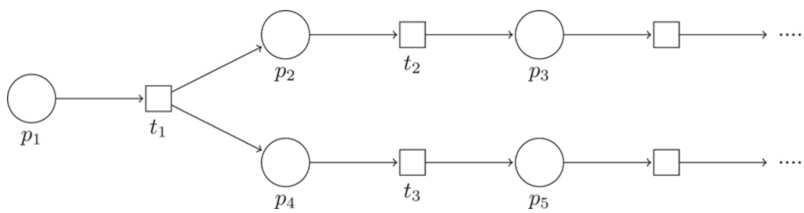
10

Common Petri Net Structures

- Sequential Execution



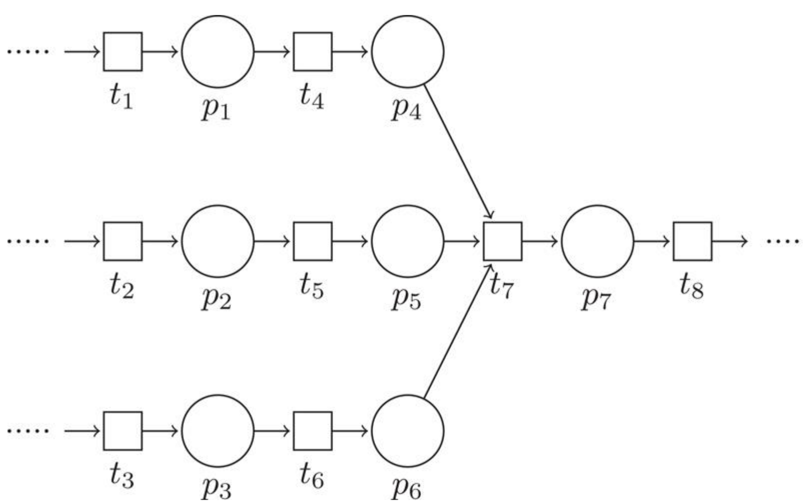
- Concurrent Execution



11

Common Petri Net Structures

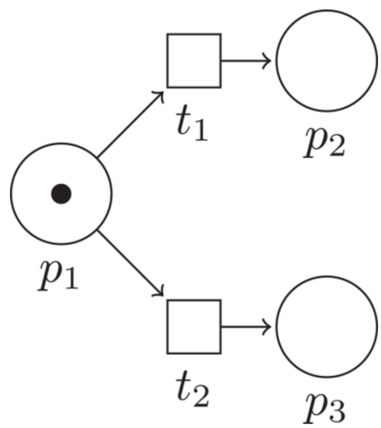
- Synchronization



12

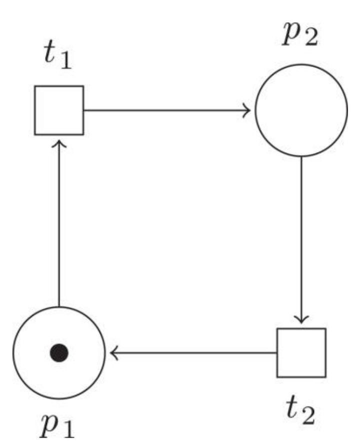
Common Petri Net Structures

- **Nondeterminism**



Common Petri Net Structures

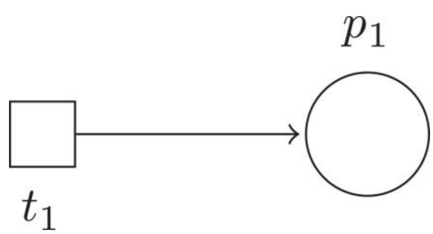
- **Loop**



Common Petri Net Structures

Source

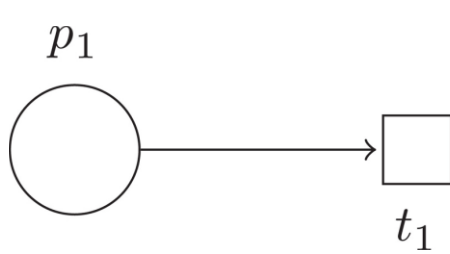
- A source is a transition that has no input place but does have an output place.
- This can be used to model an infinite source of resources entering into the system.



15

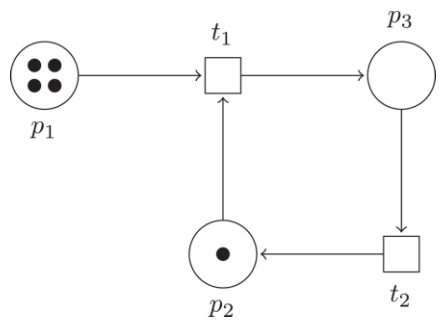
Common Petri Net Structures

- Consumer
- The opposite of a source is a consumer. A consumer is a transition that has an input place with no output places.
- This consumes the tokens in the input place for the transition.

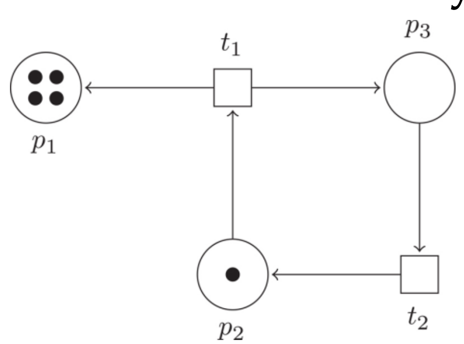


16

Common Petri Net Structures

- Control
- Modeling some kind of fixed resource can be accomplished rather easily. In this case, a control place is initialized with the available resources.
- That place can be incorporated into the Petri net as a limiter.
- It can also be used as a break point for the loop.
- 

Common Petri Net Structures

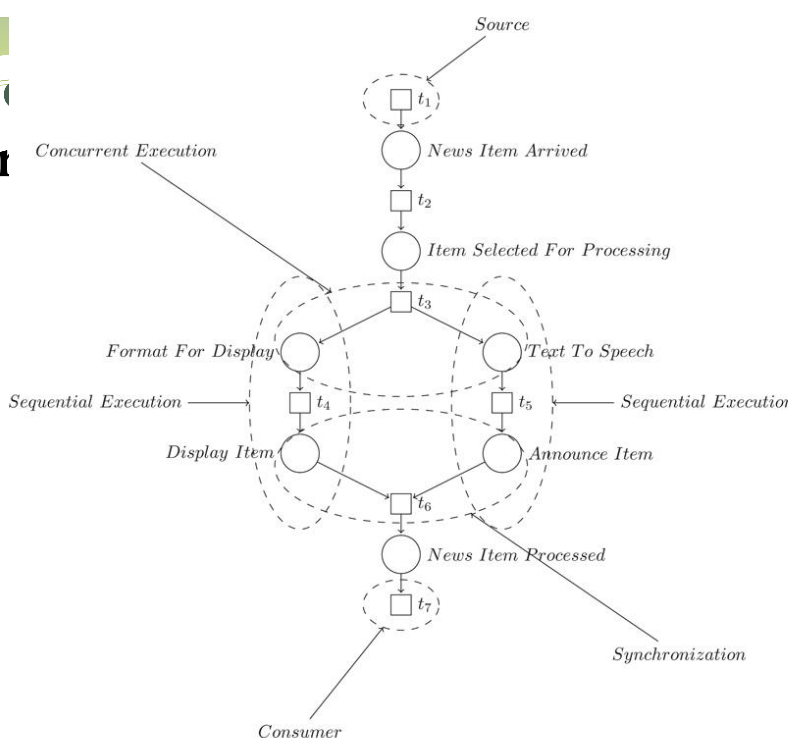
- Accumulator
- The opposite of a control is an accumulator. Rather than a place that holds tokens which are to be consumed, an accumulator is a place where tokens are created and never consumed.
- It effectively holds a count of how many times t1 has been fired.
- 

-

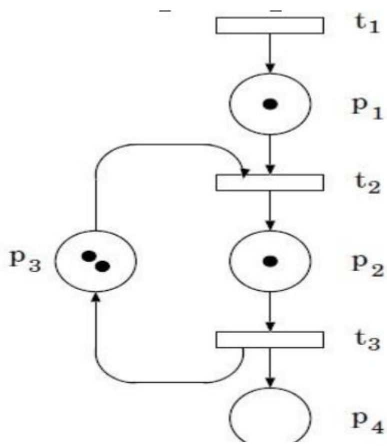
- **Example**
- **A Petri net representing a general announcement program that displays news items on a monitor and speaks the news item aloud over a loudspeaker. It is easy to analyze this program by recognizing the various structures and substructures within it.**

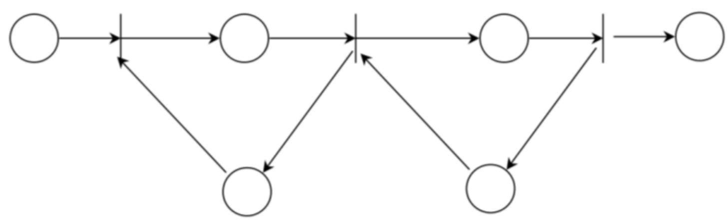
Comm

- **Synchr**



Buffer (Queue)





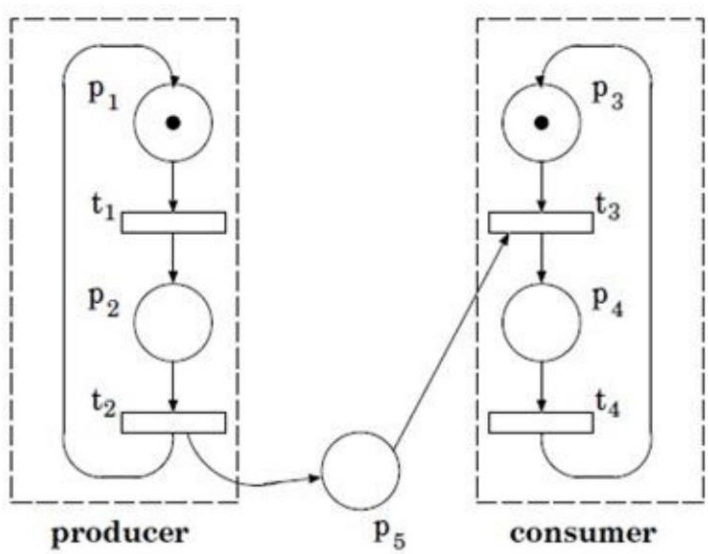
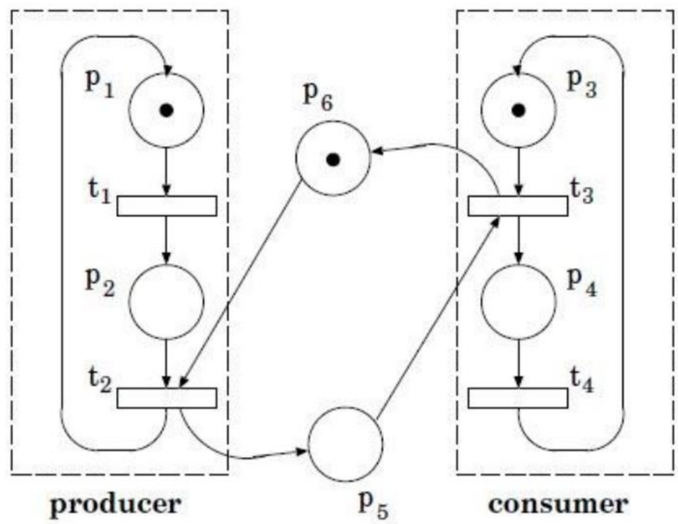


Fig 15: The producer/consumer problem with unbounded buffer



: The producer/consumer problem with finite buffer

Resource Sharing

Mutual Exclusion (Conflict)

- Places *p*1 and *p*5 represent C1 and C2 working independently;
- *p*2 and *p*6 represent C1 and C2 requesting access to C*s*;
- Place *p*4 determines availability of resource C*s*; prevents *p*3 & *p*7 to be marked at the same time;
- *p*3 and *p*7 represent C*s* busy with C1 and C2 respectively.
- Firing of *t*3 or *t*6 models the release of the common resource.

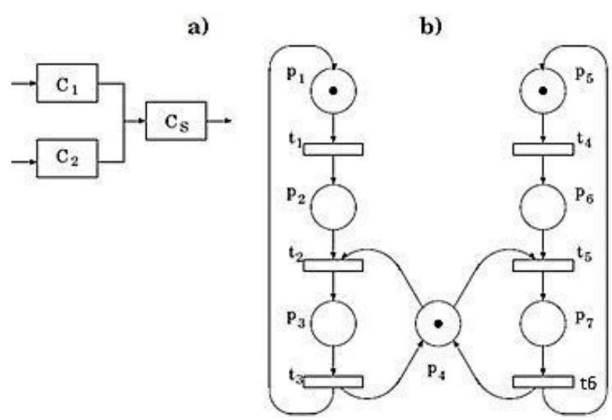
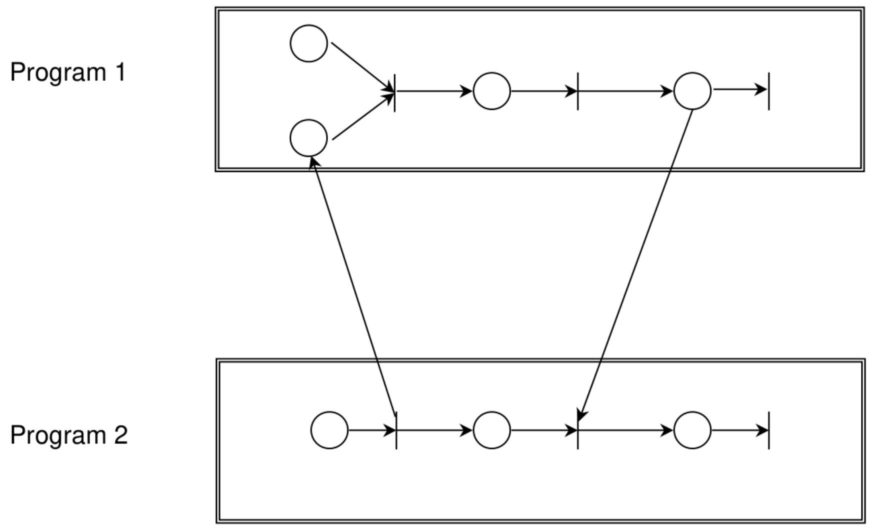


Fig 17: The mutual exclusion problem

Communication



Petri Net

Example

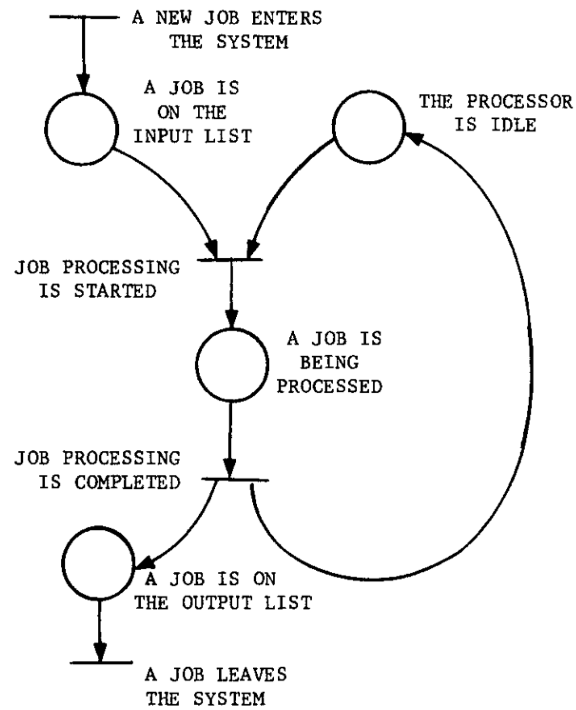
Consider for example the following description of a computer system:

- **Jobs appear and are put on an input list. When the processor is free, and there is a job on the input list, the processor starts to process the job.**
- **When the job is complete, it is placed on an output list, and if there are more jobs on the input list, the processor continues with another job; otherwise it waits for another job.**

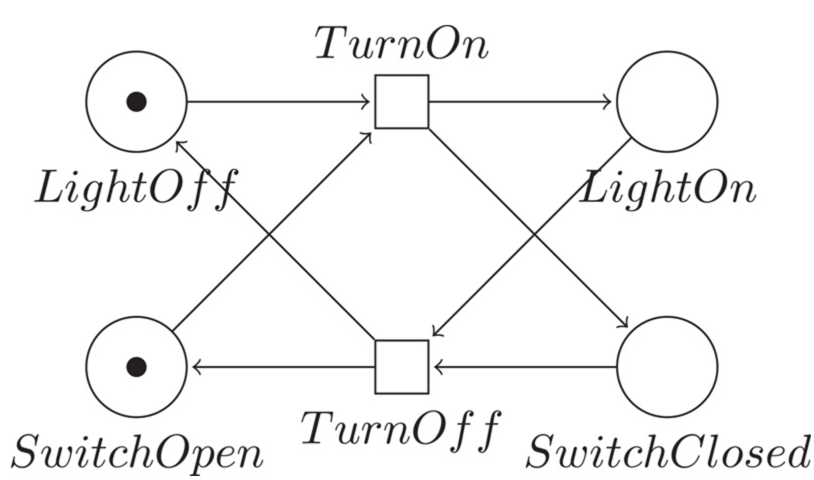
Petri Net

Petri Net Example

- conditions:
  - The processor is idle;
  - A job is on the input list;
  - A job is being processed;
  - A job is on the output list;
- Events:
  - A new job enters the system;
  - Job processing is started;
  - Job processing is completed;
  - A job leaves the system.



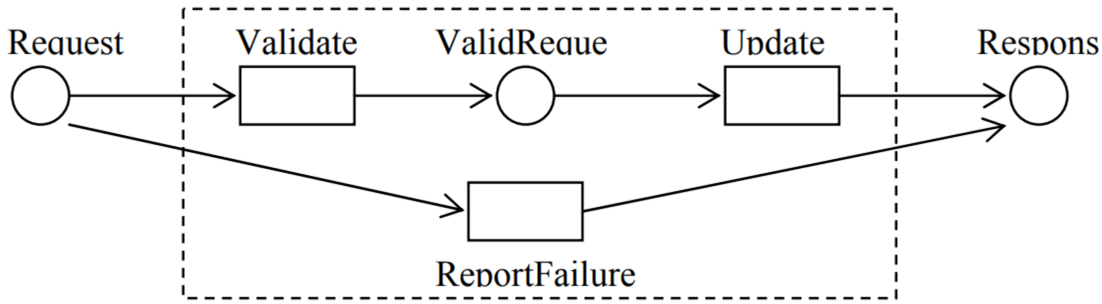
Example Model Light bulb and switch circuit using petri Net.



28

Petri Net

Petri Net Example



Model of Transactions

Petri Net

Mathematical Description of Petri Nets

A graphical depiction of a Petri net allows one to understand it structurally and intuitively.

However, a picture doesn't help answer a number of questions that are of interest when specifying a system, including such questions as

- Can the system reach a specific state from the current state?
- What states of the system are accessible?
- Is the system alive?

To answer these questions, one must adopt a mathematical model of a Petri net.

Petri Net

Mathematical Description of Petri Nets

A Petri net N is a tuple N = {P, T, I, O, M_0}, where

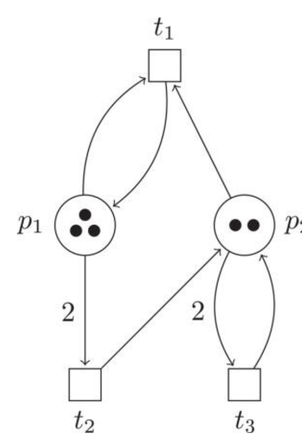
- **P is a finite set of places, graphically represented by circles.**
- **T is a finite set of transitions, graphically represented by boxes.**
- **I: P × T → ({0, 1, 2, …}) is the pre-incidence function representing input arcs.**
- **O: T × P → ({0, 1, 2, …}) is the post-incidence function representing output arcs.**
- **M_0: P → N is the initial marking representing the initial distribution of tokens.**

Places P and transitions T are disjoint (P ∩ T = Ø).

Petri Net

Mathematical Description of Petri Nets

Example



$$P = \{p_1, p_2\}$$

$$T = \{t_1, t_2, t_3\}$$

$$\mathrm{I}(p_1,\ t_1) = 1\ \mathrm{I}(p_1,\ t_2) = 2\ \mathrm{I}(p_1,\ t_3) = 0$$

$$\mathrm{I}(p_2,\ t_1) = 1\ \mathrm{I}(p_2,\ t_2) = 0\ \mathrm{I}(p_2,\ t_3) = 2$$

Petri Net

Mathematical Description of Petri Nets

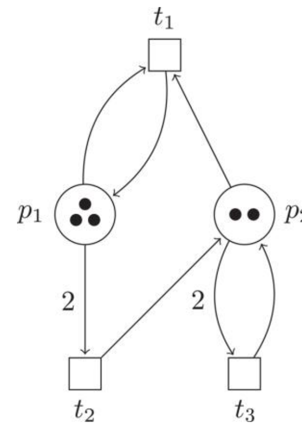
Example

$$O(t_1,\ p_1) = 1 \quad O(t_1,\ p_2) = 0$$

$$O(t_2,\ p_1) = 0 \quad O(t_2,\ p_2) = 1$$

$$O(t_3,\ p_1) = 0 \quad O(t_3,\ p_2) = 1$$

$$M_0 \ = \ (3,\ 2)$$



33

Petri Net

Extensions of Petri Nets

- Deterministic Timed Petri Nets
  - Deterministic time delays with transitions
- Stochastic Timed Petri Nets
  - Stochastic time delays with transitions
- Color Petri Nets
  - Tokens with different colors
- Hybrid Nets
  - Combine object-oriented concept into Petri nets

34