ECE 09402-2/09504-2
Discrete Event Systems

Lecture 5: **Petri Nets (PNs) – Part I**

Dr. Ying (Gina) Tang
Department of Electrical and Computer Engineering
Rowan University

Finite State Machine vs. Petri Nets

Why Finite State Machine?

- An FSM is a way to describe the behavior of a logic process.
- It encourages designers to think of all possible states and transitions among states based on all possible input conditions.
- Another big point is that designers/developers can discuss with non technical people about the design idea and system behavior.

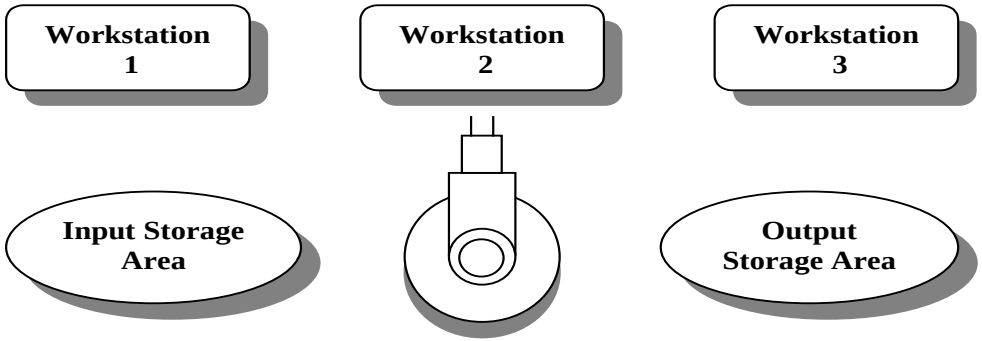
Finite State Machine vs. Petri Nets

- Finite State Machines (FSM) and Petri Nets (PN) are conceptual models to represent the discrete interactions in a system.
  - A FSM represents how sequential events change a system's behavior over time.
  - A PN represents how multiple activities are coordinated.
  - FSM: sequential events; Petri Nets: concurrency.
  - FSM allows the representation of a decision but not the synchronization of parallel activities.
  - FSM can be equivalently represented by a subclass of Petri nets.

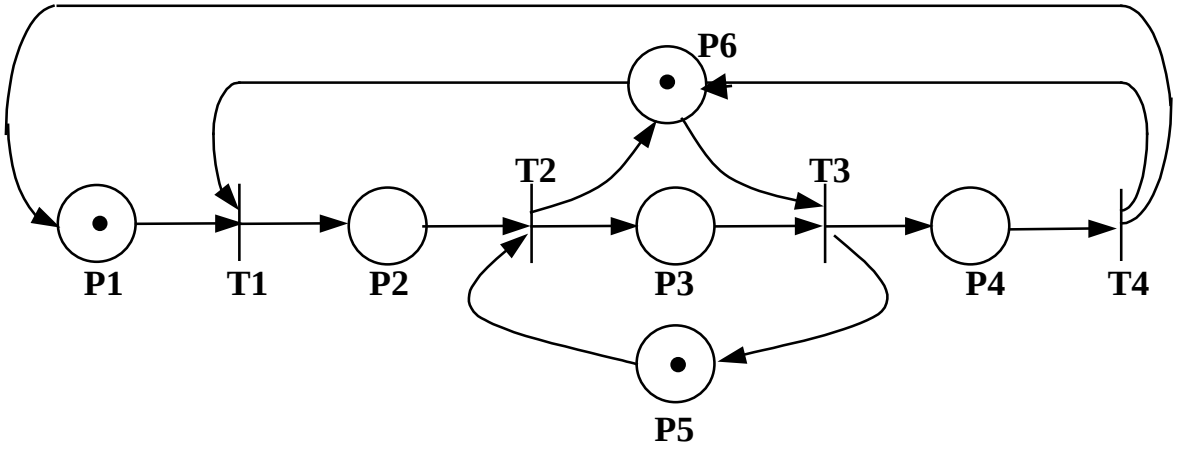
Petri Nets

- A mathematical and graphical modeling method
- A graphical way of modeling concurrent systems by representing the structure and the dynamics of a discrete event system.
- Abstract, formal model of information flow
  - Describe and analyze the flow of information and control in systems involving concurrent activities
- Assess the correctness of systems
  - The system cannot deadlock
  - There cannot be any buffer overflows

Examples of PN Modeling…



A manufacturing system with three workstations and one robot



- P1 -- parts in input storage area
- P2 -- robot with a part in WS1
- P3 -- part being processed in WS2
- P4 -- robot with a part in WS3
- P5 -- capacity of WS2
- P6 -- availability of the robot

Characteristics of PN Modeling…

- The graphical nature of the Petri Nets allows the visualization of the complexity of the system
- Static properties: input, output
- Dynamic properties: execution
- Petri Nets capture the *precedence relations* and *structural interactions* of concurrent and asynchronous events
- State machines: Allow representations of decisions, but not the synchronization of parallel activities.
- There are many variants of the "basic" Petri Nets; some are extensions, some are abbreviations

Petri Nets

Background…

- **In 1962** Carl Adam Petri from University of Darmstadt, Germany, published his research work in his PhD dissertation "Kommunikation mit Automaten"
- **In the early 70s**, major extensions were carried out at MIT through a Project MAC at the Laboratory for Computer Science
- **From 1980 on**, the annual European Workshop on Applications and Theory of Petri Nets has been the forum for the presentation of many results. Selected papers were published in the Springer Verlag series Lecture Notes in Computer Science. =>International Conference on Applications and Theory of Petri Nets
- **In the 90s,** Petri Nets started having wide adoption, first by the Flexible Manufacturing Systems community and now in a wide variety of applications.

Petri Nets

Research History…

- Much research has been conducted.
- Until 1985 it was mainly used by theoreticians.
- Since the 80's, trend increases towards the practical use because of the introduction of high-level Petri nets and the availability of many tools.
- High-level Petri nets are Petri nets extended with
  - color (for the modeling of attributes)
  - time (for performance analysis)
  - hierarchy (for the structuring of models)

Applications of PNs…

- Communication protocols
- Distributed systems
- Distributed database systems
- Flexible manufacturing systems
- Logical controller design
- Multiprocessor memory systems
- Dataflow computing systems
- Fault tolerant systems
- Decision making/expert systems

Petri Nets - Fundamentals

Fundamentals PN = (P, T, I, O, M)

- $P = \{p_1, p_2, \ldots, p_m\}$ is a finite set of places.
- $T = \{t_1, t_2, \ldots, t_n\}$ is a finite set of transitions, $P \cup T \neq \Phi$, $P \cap T = \Phi$
- $I: P \times T \rightarrow N$ is an input function that defines the set of directed arcs from P to T, where N={0, 1, 2…} and I_{ij} indicates the weights of the arc from P to T.
- $O: P \times T \rightarrow N$ is an output function that defines the set of directed arcs from T to P, where N={0, 1, 2…} and O_{ij} indicates the weights of the arc from T to P.
- $M: P \rightarrow \{0, 1, 2, \ldots\}$ is a marking vector whose i^{th} component represents the number of tokens in the i^{th} place. An initial marking is denoted by m_0.

Petri Nets - Fundamentals

Fundamentals PN = (P, T, I, O, M)

- **Petri Nets are bipartite directed multi-graphs consisting of places *P* and transitions *T***
  - Bipartite: two types of nodes
    - Circle node or place
    - Bar node or transition
  - Directed: the arcs that join two nodes are directed.
- **Places are represented by circles.**
- **Transitions are represented by bars (or rectangles)**
- **Arcs can connect places to transitions and transitions to places only**

Fundamentals (cont.)…

- Places represent buffers, channels, geographical locations, conditions or states.
- Transitions represent events, transformations or transportations.

Some typical interpretations of Transitions and Places

Transitions stand for	Input places stand for	Output places stand for
Processor	Buffers	Buffers
Event	Pre-conditions	Post-conditions
Computation step or algorithm	Input data	Output data
Task or job	Resources needed	Resources freed
Clause in logic	Conditions	Conclusions
Signal processor	Input signals	Output signals

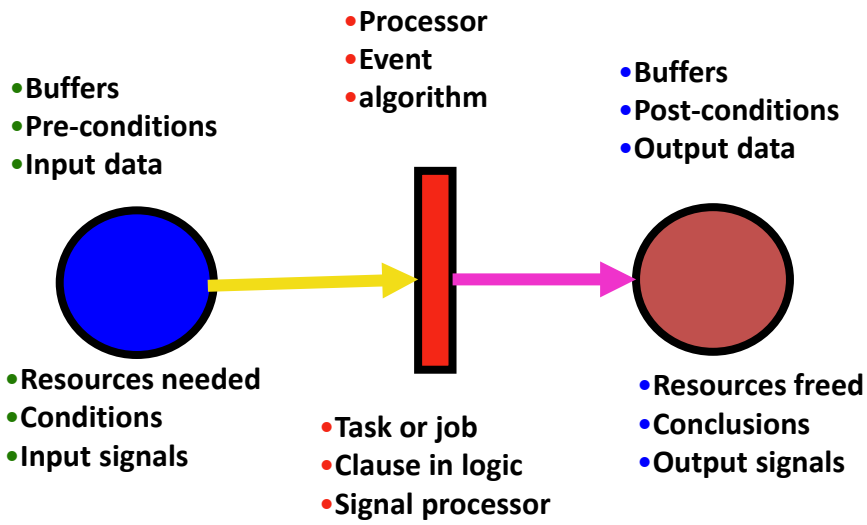
Basics of PNs – Token

- States of a process are modeled by **tokens** in places
- Tokens are depicted graphically by dots (•) and reside in places
- The state of a Petri net is determined by the distribution of tokens over the places.
- The existence of one or more tokens represents the availability of the resource or the fulfillment of the condition, while the absence of tokens indicates that the condition is not satisfied or the resource is unavailable.
- Execution of Petri net are controlled by tokens
- Tokens are indistinguishable (for traditional PN).

Petri Nets - Fundamentals

Basics of PNs --Arc

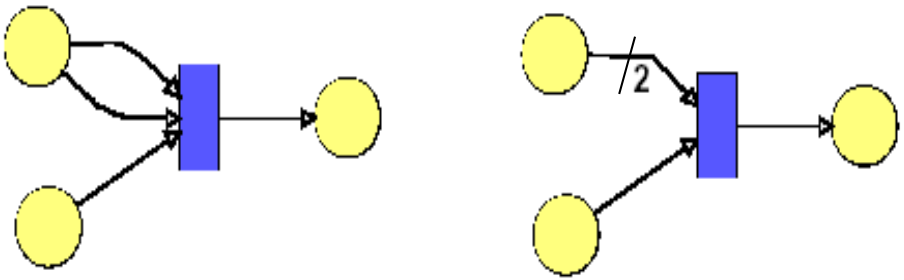
- **An arc from a place to a transition means that**
  - The process needs the resource in the buffer
  - The event has this precondition in order to occur
  - The algorithm requires the input data
  - The task needs this resource
  - This is a condition for the logic clause



- **An arc from a transition to a place means that**
  - The process generates a resource in the output buffer
  - The event generates this condition when it occurs
  - The algorithm generates the output data
  - The task produces this resource
  - This is a conclusion of the logic clause

Basics of PNs –Arc Weight

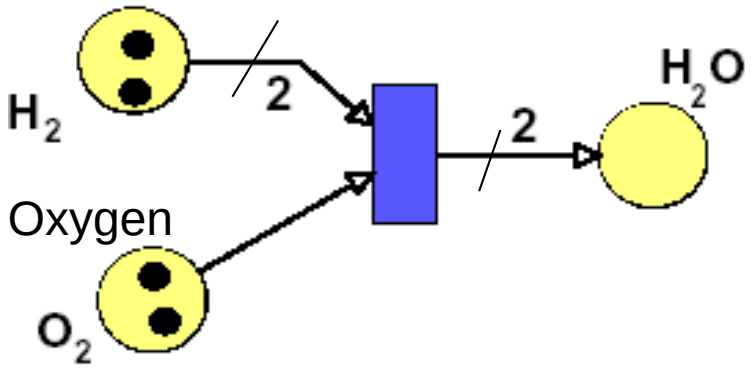
- Petri Nets are, in general, multi-graphs. This means that there can be a number of arcs from a place to a transition or a transition to a place.
- The number of arcs is indicated by an integer annotation on the arc.
- It is called the multiplicity of that arc or ***Arc Weighting***.
- In an ordinary Petri Net, all arcs have multiplicity of 1.
- Ordinary Petri Nets are the basic model.
- Normally, arcs are labeled with their weights (positive integers). Labels for unity weights are normally omitted.



Petri Nets - Fundamentals

a) Before Reaction

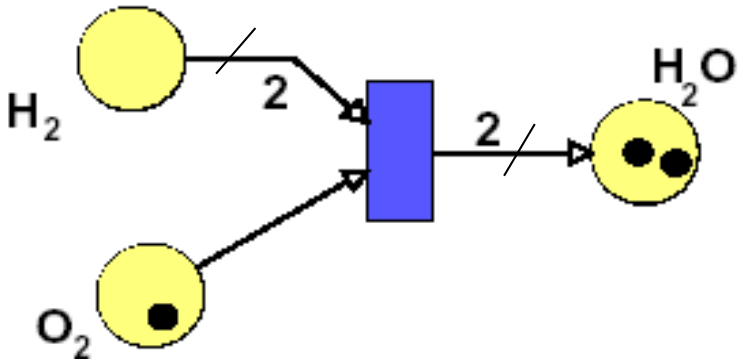
Hydrogen



Representation of the chemical reaction

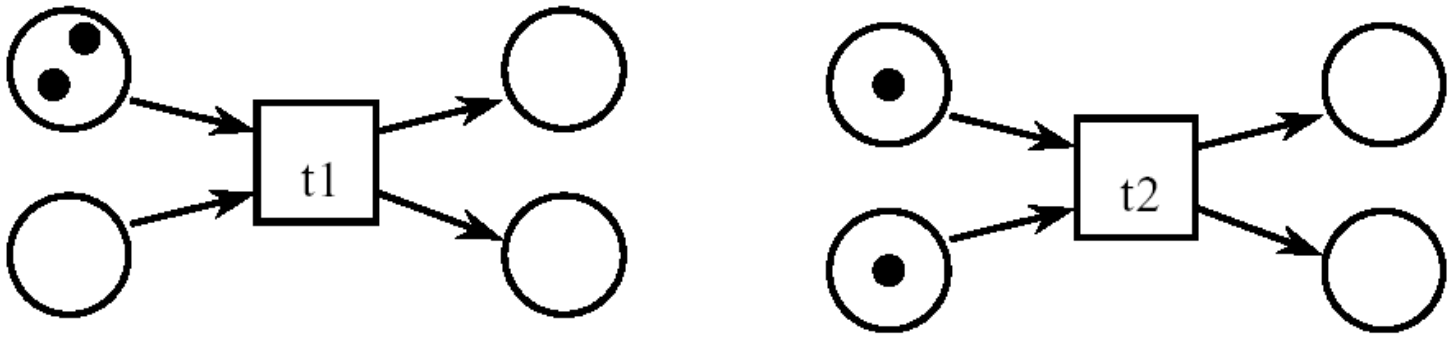
$2H_2 + O_2 \rightarrow 2H_2O$

b) After Reaction



Dynamics of PNs – Transition Fire

- A system's dynamic behavior can be represented by the **movement of tokens**.
- **Tokens are moved by firing of transitions**.
- A transition **must be enabled in order to fire**.
- A transition is ***enabled*** if ***all*** its input places contain at least the number of tokens specified by the arc weighting.



Transition t1 is not enabled, transition t2 is enabled.

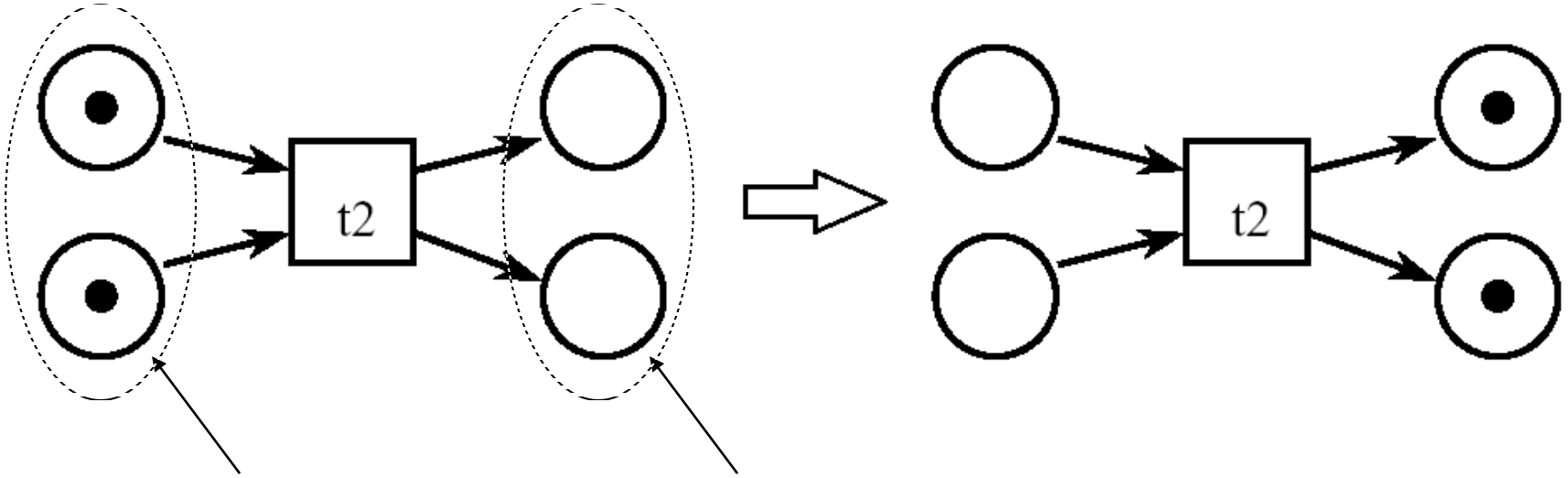
Petri Nets - Fundamentals

Fundamentals PN = (P, T, I, O, M)

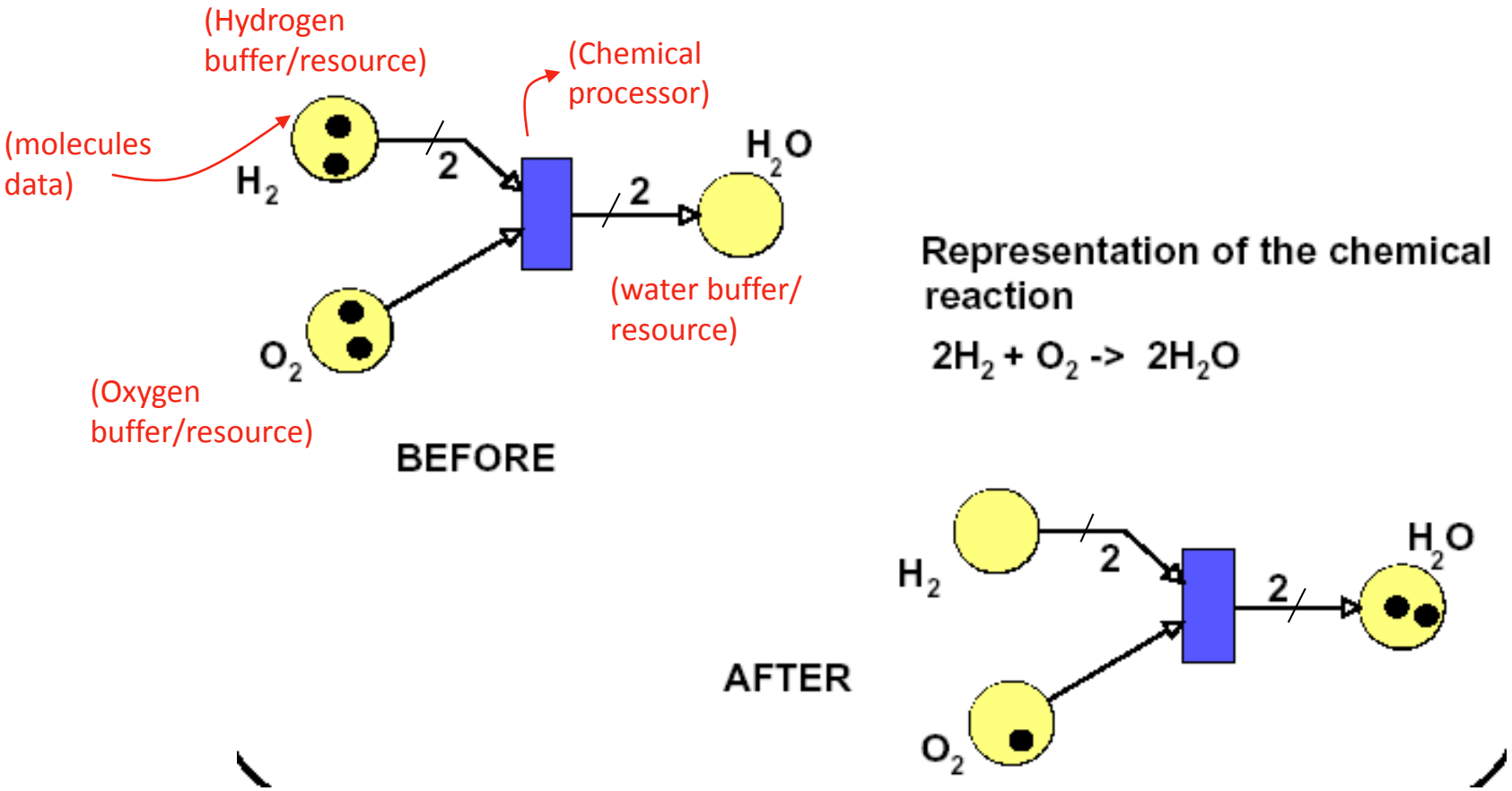
- $P = \{p_1, p_2, \ldots, p_m\}$ is a finite set of places.
- $T = \{t_1, t_2, \ldots, t_n\}$ is a finite set of transitions, $P \cup T \neq \Phi$, $P \cap T = \Phi$
- $I: P \times T \rightarrow N$ is an input function that defines the set of directed arcs from P to T, where N={0, 1, 2…} and I_{ij} indicates the weights of the arc from P to T.
- $O: P \times T \rightarrow N$ is an output function that defines the set of directed arcs from T to P, where N={0, 1, 2…} and O_{ij} indicates the weights of the arc from T to P.
- $M: P \rightarrow \{0, 1, 2, \ldots\}$ is a marking vector whose i^{th} component represents the number of tokens in the i^{th} place. An initial marking is denoted by m_0.

Petri Nets - Fundamentals

- An enabled transition may **fire**.
- Firing corresponds to:
  - **consuming** tokens from the input places and
  - **producing** tokens for the output places.



Petri Nets - Fundamentals



Note: From this example, we know that the firing does not guarantee the conservation of tokens.

Dynamics of PNs – Transition Rules

- A transition t is enabled if each input place contains at least one token.
- An enabled transition may or may not fire.
- Firing an enabled transition t means removing one token from each input place of t and adding one token to each output place of t.
- The firing of a transition has zero duration.
- The firing of a sink transition (having only input places) only consumes tokens.
- The firing of a source transition (having only output places) only produces tokens.

Petri Nets - Fundamentals

Formal Description of Execution:

Enabling: t is enabled at marking m

if $\forall\ p \in P,\ m(p) \geq I(p, t)$.

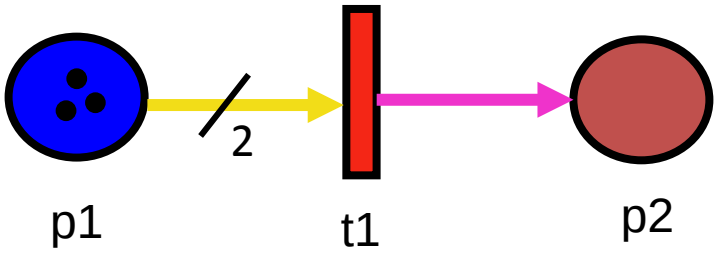
Note: ***If I(p, t)=0, the above equation holds regardless of m(p).***
If t has k input places, k such relations must hold simultaneously.

Firing: An enabled transition t at m can fire,
yielding a new marking m' such that
$\forall\ p \in P,\ m'(p) = m(p) - I(p, t) + O(p, t)$.

APPLICABLE TO ALL PETRI NETS

Petri Nets - Fundamentals

PN Examples:



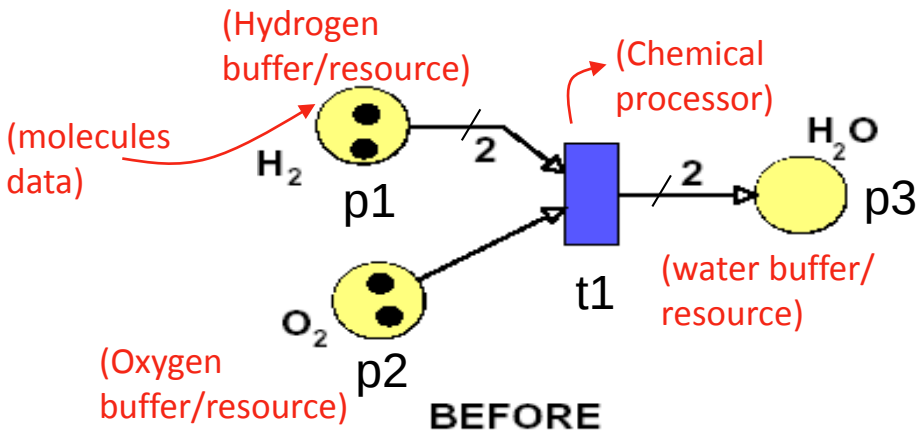
$$I = \begin{array}{c} \\ p1 \\ p2 \end{array} \begin{array}{c} t1 \\ \begin{pmatrix} 2 \\ 0 \end{pmatrix} \end{array} \qquad O = \begin{array}{c} \\ p1 \\ p2 \end{array} \begin{array}{c} t1 \\ \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{array}$$

Initially m(p1)=3, m(p2)=0.

$m(p1) \geq I(p1, t1)=2$; $m(p2) \geq I(p2, t1)=0$ => t1 is enable

After t1 fires, m'(p1)=3-2+0=1; m'(p2)=0-0+1=1. So new marking is (1, 1)

Petri Nets - Fundamentals

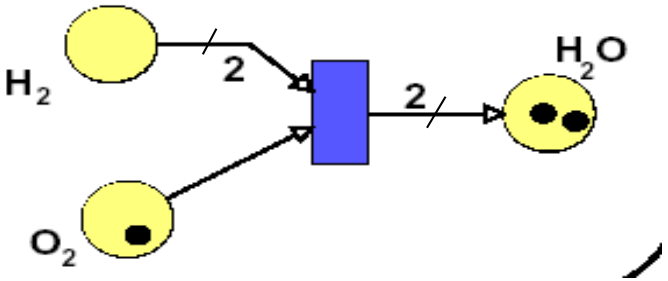


Representation of the chemical reaction

$2H_2 + O_2 \rightarrow 2H_2O$

$$I = \begin{array}{c} \\ p1 \\ p2 \\ p3 \end{array} \begin{array}{c} t1 \\ \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix} \end{array} \qquad O = \begin{array}{c} \\ p1 \\ p2 \\ p3 \end{array} \begin{array}{c} t1 \\ \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix} \end{array}$$

AFTER



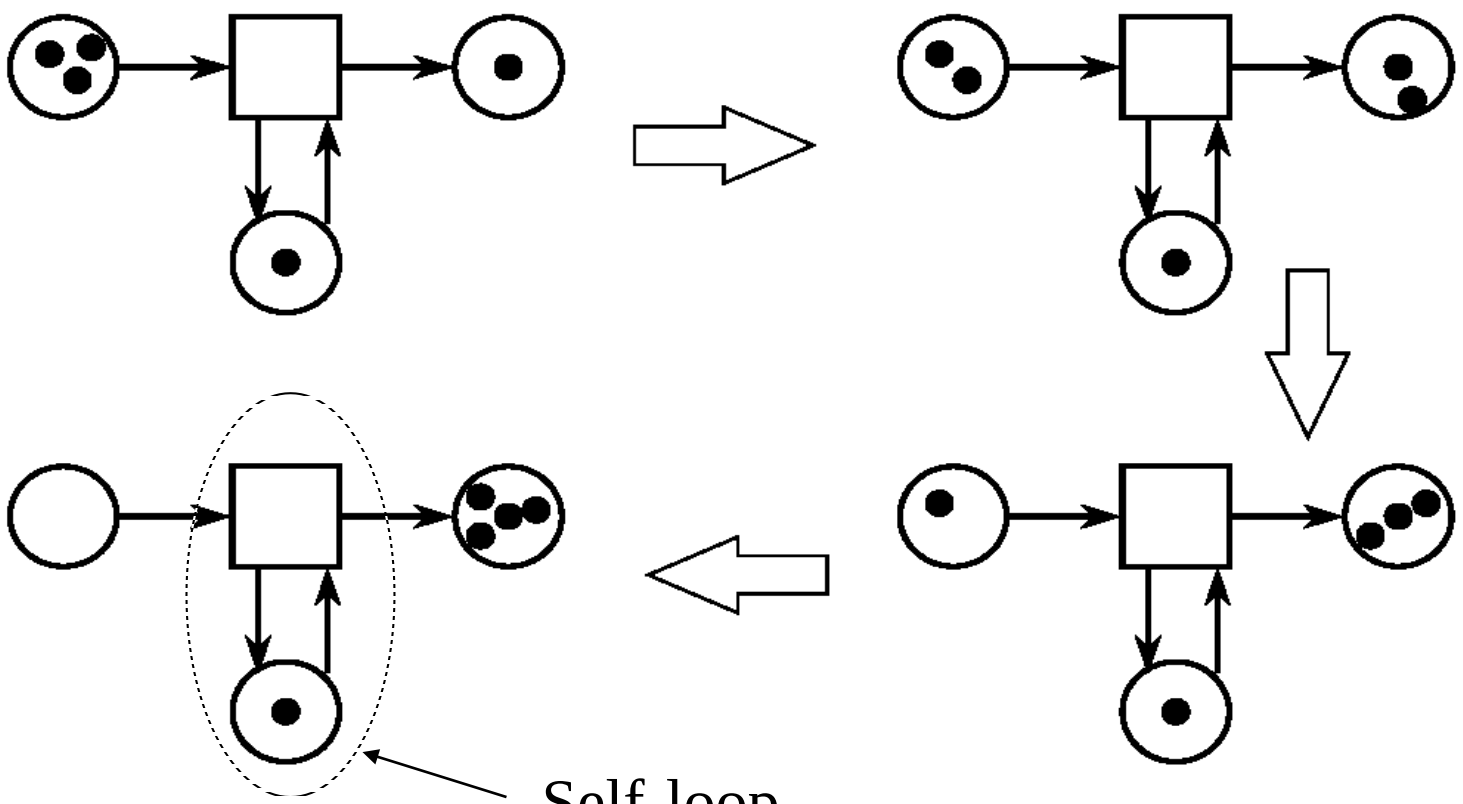
m(p1)=2; m(p2)=2; m(p3)=0

After firing t1: m'(p1)=m(p1)-I(p1,t1)+O(p1,t1)=2-2+0=0;

m'(p2)=m(p2)-I(p2,t1)+O(p2,t1)=2-1+0=1;

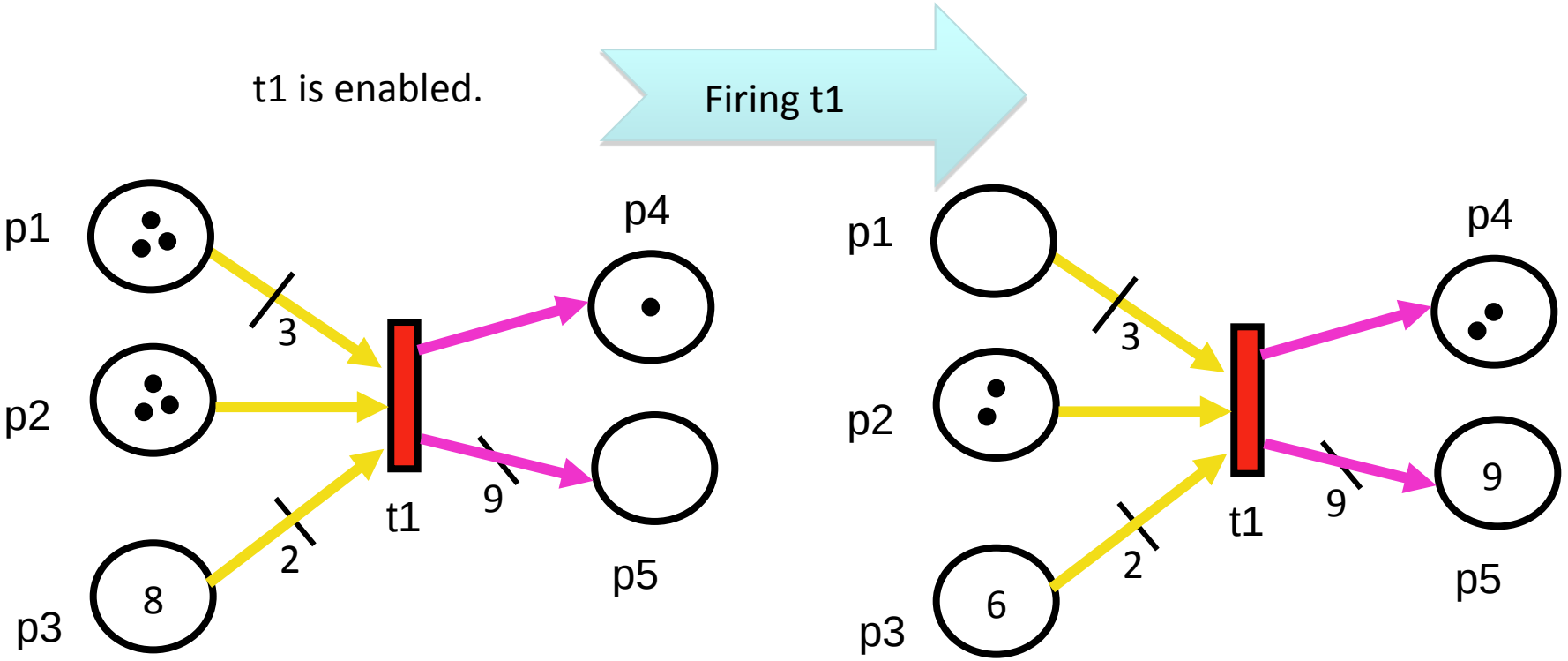
m'(p3)=m(p3)-I(p3,t1)+O(p3,t1)=0-0+2=2.

Petri Nets - Fundamentals



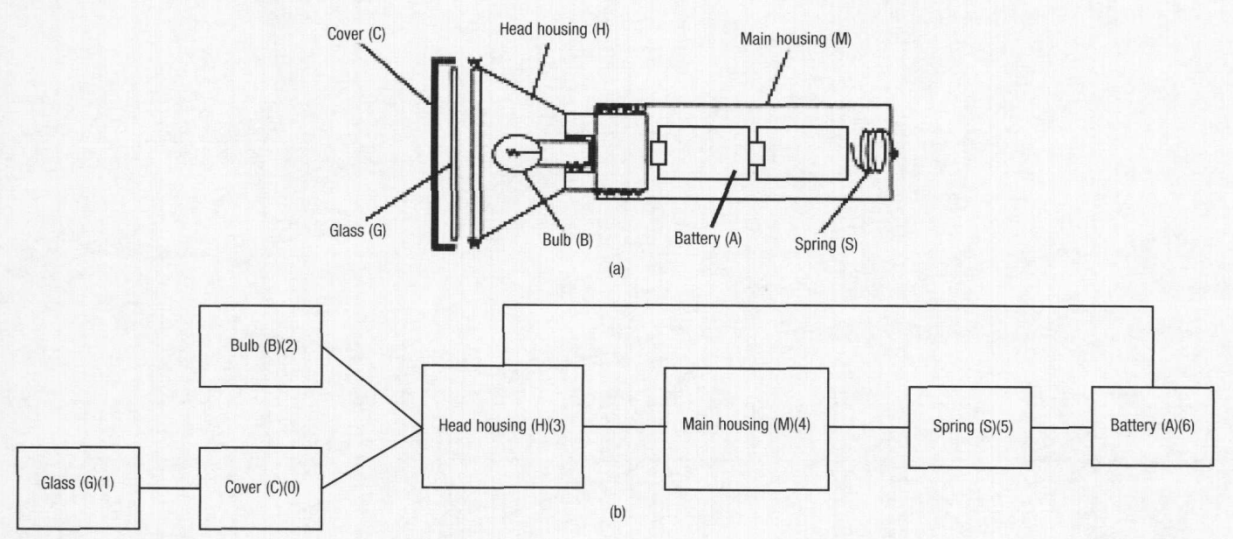
- A pair of a place and a transition t is called a self-loop if p is both an input and output place of t.
- A Petri net is said to be pure if it has no self-loop.

Petri Nets - Fundamentals



Petri Nets - Fundamentals

- **Exercise:**
  - Model the disassembly the handlight using a PN.



Petri Nets - Fundamentals

- **Exercise:**
  - Model the traffic intersection using a PN.