FSM limitations *(computational power)*

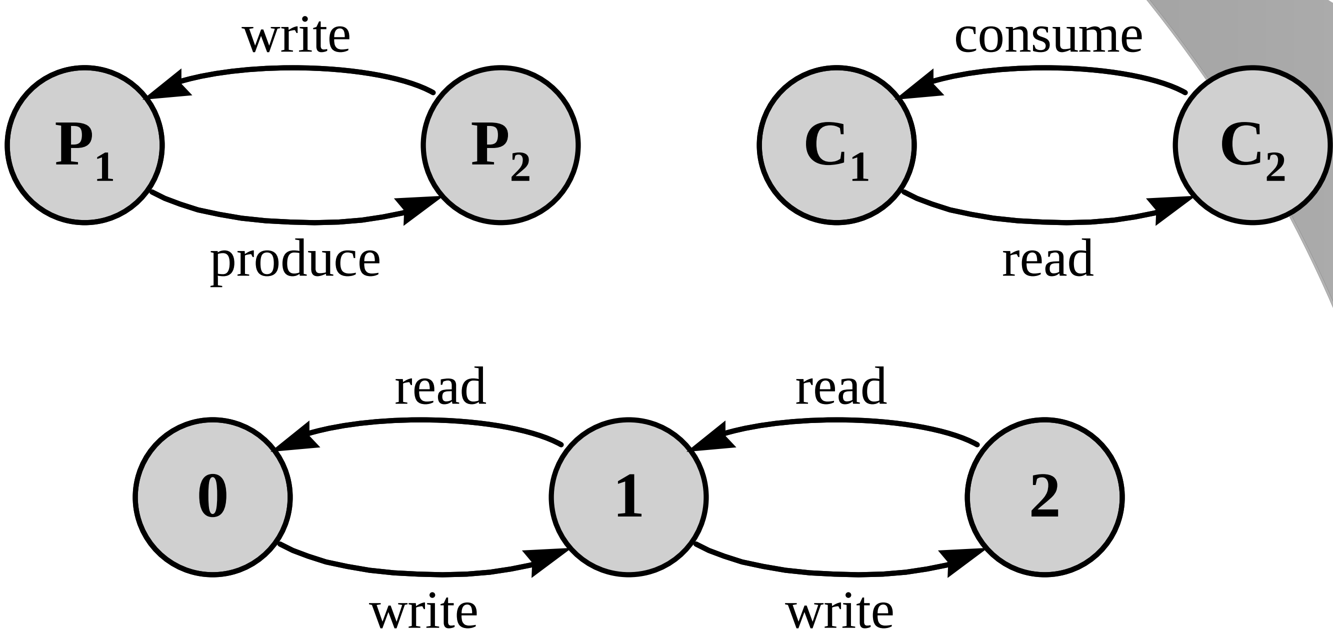
- Consider a car ABS (again) in particular the sub-system that calculates the actual brake pressure required. The range of pressures that can be handled by the servo units is potentially very great.

Therefore defining a state for every possible pressure value that can be required would be unusable!

- Consider modelling the behaviour of a simple 8-bit register - this requires **2^8 distinct states!**

FSM limitations *(state explosion - 1)*

- Consider a typical producer-consumer system modelled using three separate FSMs as follows:



FSM limitations *(state explosion - 2)*

- Integrating parts of one system

The number of states for a composite FSM made up of **n** sub-systems each with $\mathbf{m_i}$ states is $\mathbf{m_1 \times m_2 \times \ldots\ m_n}$. In the case of the producer-consumer example this would mean the following:

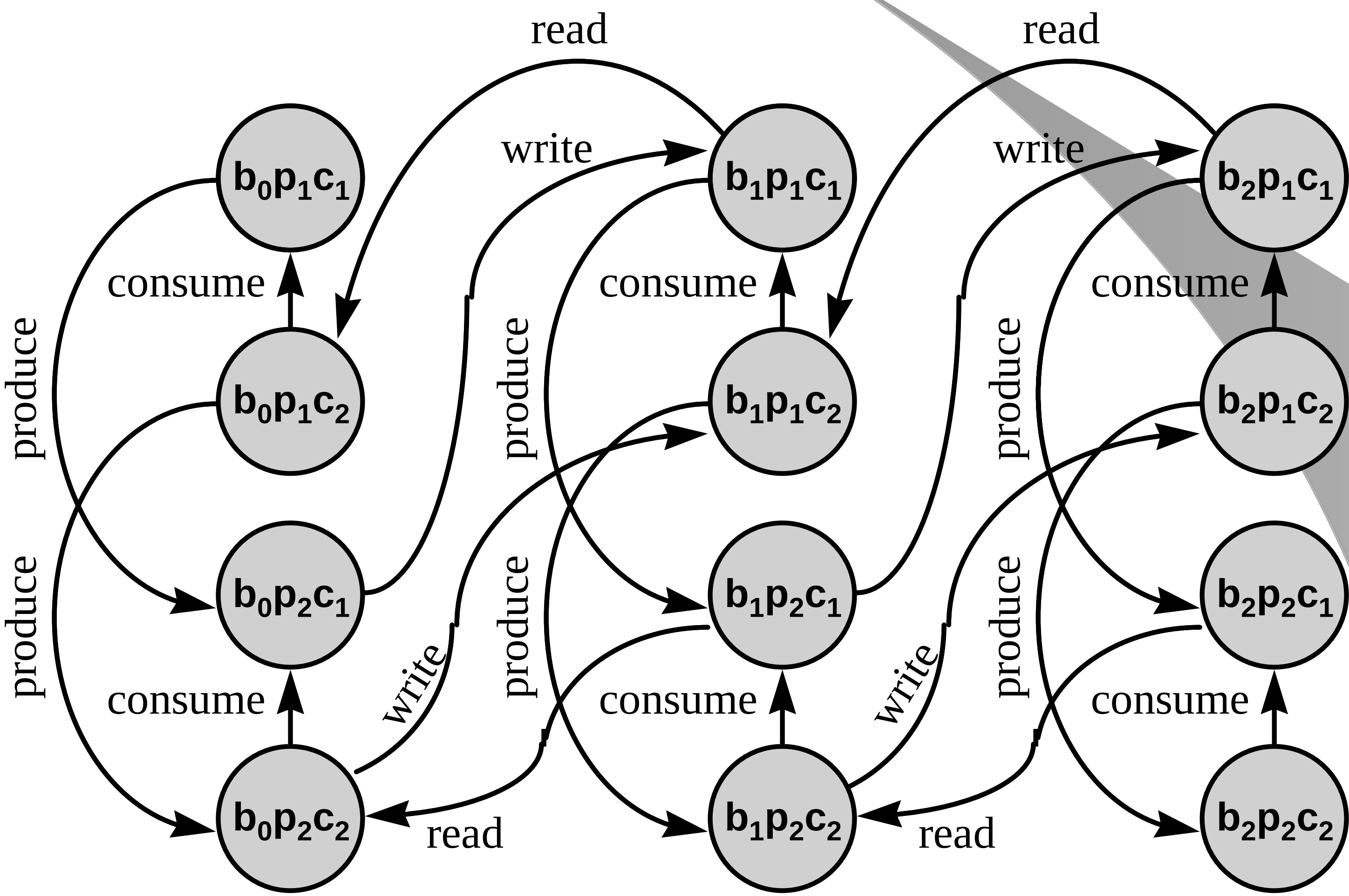
The possible states of P (p_1, p_2) multiplied by the possible states of C (c_1,c_2) multiplied by the possible states of the buffer B - i.e. empty(b_0), one entry(b_1) and two entries(b_2)

$$<b_0,p_1,c_1> \ <b_1,p_1,c_1> \ <b_2,p_1,c_1>$$
$$<b_0,p_1,c_2> \ <b_1,p_1,c_2> \ <b_2,p_1,c_2>$$
$$<b_0,p_2,c_1> \ <b_1,p_2,c_1> \ <b_2,p_2,c_1>$$
$$<b_0,p_2,c_2> \ <b_1,p_2,c_2> \ <b_2,p_2,c_2>$$

12 states in all

FSM limitations *(state explosion - 3)*

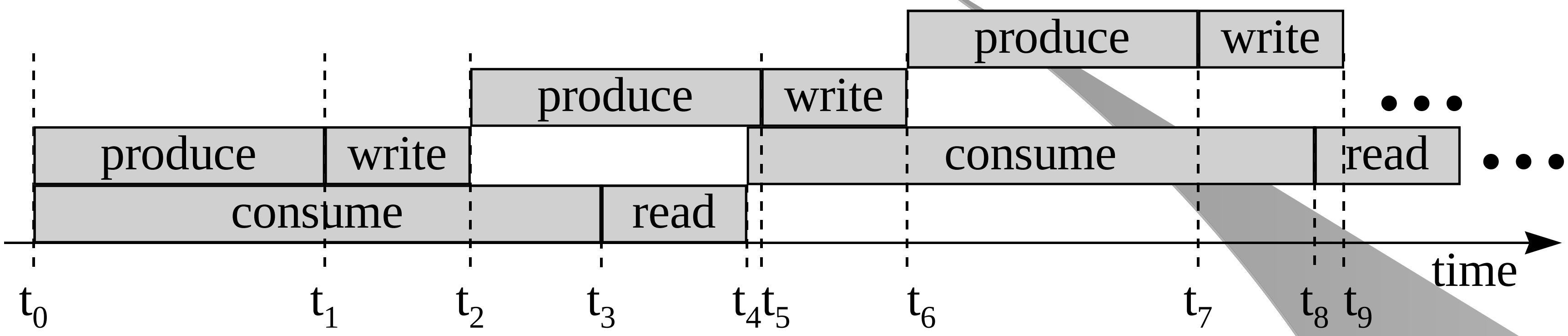
The resulting integrated FSM:



FSM limitations *(modelling timing -1)*

- An FSM is a “snapshot” of a system at a particular instant in time
- An FSM assumes one action (transition) per instant in time
- Asynchronous (i.e. potentially concurrent) actions can be of different duration *(eg. p = t; c = 2t; r = w = t/4)*
- This aspect (concurrent action timing) of system operation is not clearly captured by FSMs

FSM limitations *(modelling timing -2)*



t_0: $<b_0,p_1,c_2>$	t_5: $<b_0,p_2,c_2>$
t_1: $<b_0,p_2,c_2>$	t_6: $<b_1,p_1,c_2>$
t_2: $<b_1,p_1,c_2>$	t_7: $<b_1,p_2,c_2>$
t_3: $<b_1,p_1,c_1>$	t_8: $<b_1,p_2,c_1>$
t_4: $<b_0,p_1,c_2>$	t_9: $<b_1,p_1,c_1>$

The states at times t_0 and t_4 are the same $<b_0,p_1,c_2>$ as are also the states at times t_1 and t_5 $<b_0,p_2,c_2>$, states t_2 and t_6 $<b_1,p_1,c_2>$, states t_3 and t_9 $<b_1,p_1,c_1>$. However, looking at the graph it is clear that they do not refer to the same actions. This fact is not captured adequately by the FSM.

Petri Net (PN) components

- Graphical formalism of system behaviour specification
- Constituents:
  - finite set of *places* **P**
  - finite set of *transitions* **T**
  - finite set of *directed arcs* **A**
    (these connect either places to transitions or vice-versa)
- PN marking - imposing a state on the PN.
- Tokens - used to mark a PN

Petri Nets specs

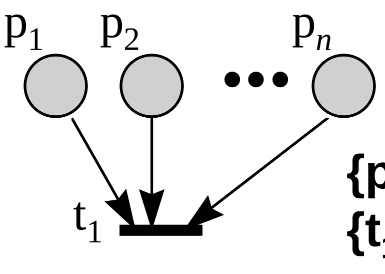
- Referred to as: **PN**
- To model: **Asyncronous systems**
- Type: **Formalised notation**
- Popularity: **Medium**
- Notation: **Relatively simple and intuitive**

Place symbol   Transition symbol   Connecting arc symbol

PN “evolution” rules (1)

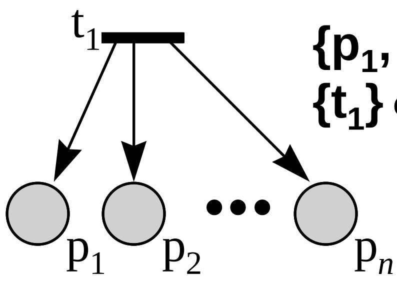
- PN evolution implies movement of tokens
- A PN does not change shape as it evolves
- A transition may have one or more input or output places

- Input places $\{I\} \subset \{P\}$



$\{p_1, p_2, \ldots p_n\} \in I$
$\{t_1\} \in T$

- Output places $\{O\} \subset \{P\}$



$\{p_1, p_2, \ldots p_n\} \in O$
$\{t_1\} \in T$

Dept. of Computer Science and A. I., University of Malta

PN "evolution" rules (2)

- Transition enabled $\Rightarrow$

  At least **one** token in **each** ot transition's imput places.

- Transition "firing" $\Rightarrow$

  Only enabled transitions can fire.

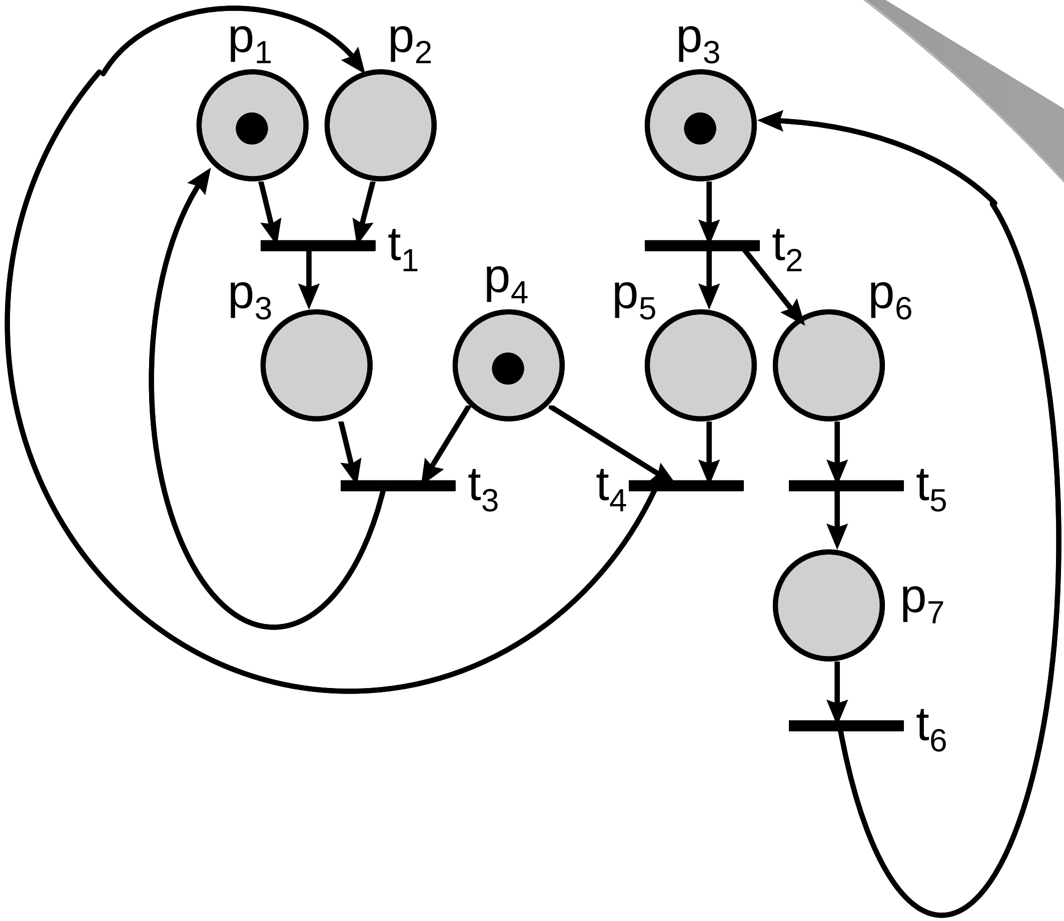
- PN evolution $\Rightarrow$

  Every firing leads to a new PN (i.e. system) state. This is to PN evolution.

- Evolution result *(apart from new state)* $\Rightarrow$

  One token is removed from each input place and **one** is inserted into each output place.

PN example



Some PN modelled situations (1)

- **Firing sequence**

  The string of transitions $\mathbf{<t_1, t_2, \ldots, t_n>}$ where t_1 is enabled in the PN's initial marking.

  A possibility in the previous example: $\mathbf{<t_2, t_4, t_1>}$

- **Nondeterminism**

  The possibility of more than one evolution path given an initial marking.

  Possibilities in the previous example: $\mathbf{<t_2, t_4, t_1>}$
  or: $\mathbf{<t_2, t_5, t_6, t_2, t_5, t_6, t_2, \ldots>}$

Some PN modelled situations (2)

- **Deadlock**

  In terms of a system modelled using a PN, deadlock is the situation in which no transitions are enabled within a given marking. The PN stops evolving.

- **Livelock**

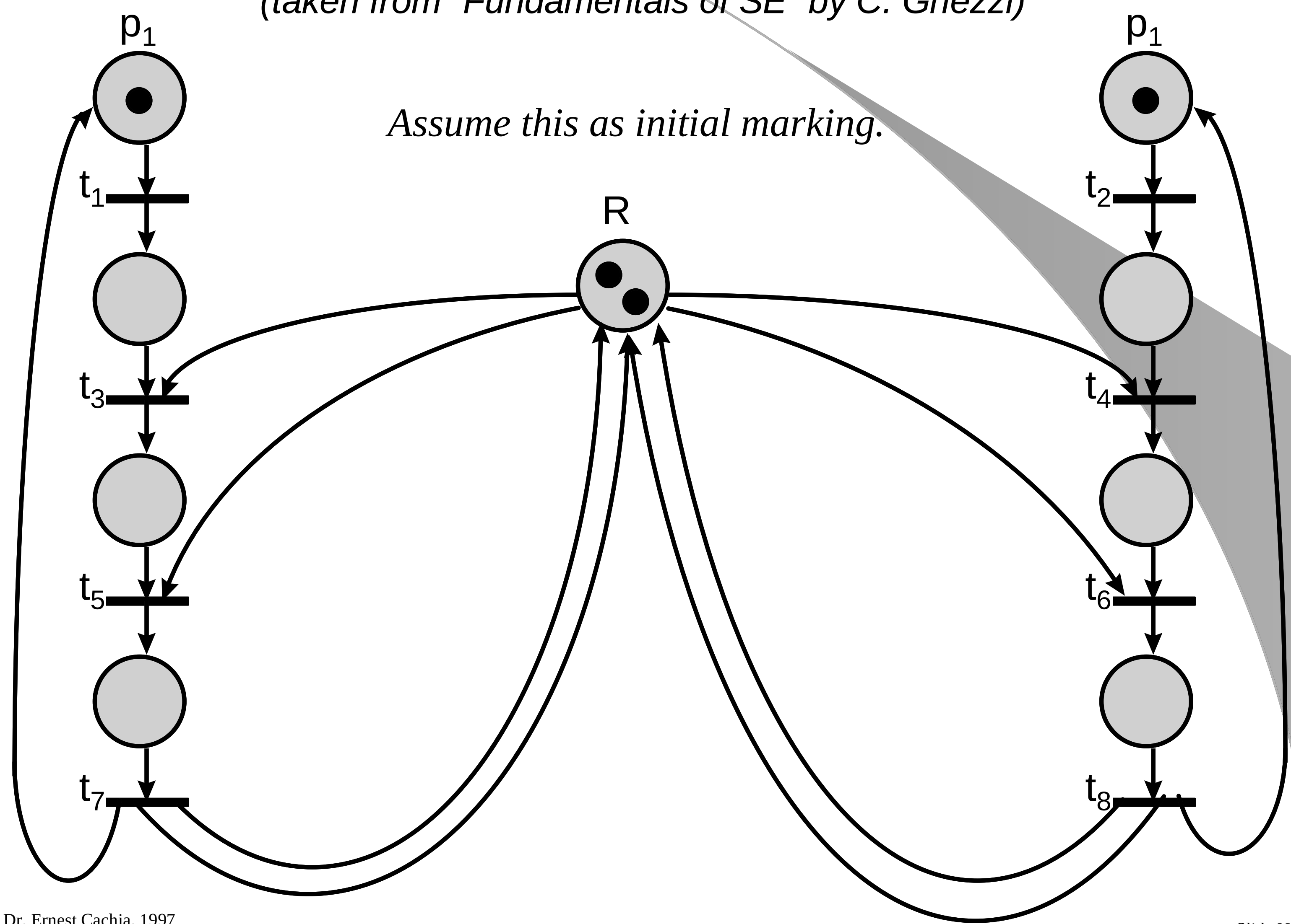
  A situation in which deadlock can **never** occur (i.e. the PN will never reach a "conclusive" state).

- **Starvation**

  A situation in which part of a system can never proceed due to another part using a resource it needs.

Deadlock example

(taken from "Fundamentals of SE" by C. Ghezzi)



Deadlock example analysis

- Normal system progress:

  The PN evolves according to firing sequence:

  <t_1, t_3, t_5 , t_7,...> or

  <t_2, t_4, t_6 , t_8,...>

- Deadlock situation:

  The PN evolves according to firing sequence:

  <t_1, t_3, t_2 , t_4, d> or

  <t_2, t_4, t_1 , t_3, d>