VERIFICA DEI PROGRAMMI CONCORRENTI
VPC 19-20

Formalismi: le reti di Petri
(versione ridotta per le lezioni on-line)

Prof.ssa Susanna Donatelli

Università di Torino

www.di.unito.it

susi@di.unito.it

Reference material books:

Chapter 2

Untimed Petri Nets

2.1 Introduction

Typical discrete event dynamic systems (DEDS) exhibit parallel evolutions which lead to complex behaviours due to the presence of synchronisation and resource sharing phenomena. *Petri nets (PN)* are a mathematical formalism which is well suited for modelling concurrent DEDS: it has been satisfactorily applied to fields such as communication networks, computer systems, discrete part manufacturing systems, etc. Net models are often regarded as self documented specifications, because their graphical nature facilitates the communication among designers and users. The mathematical foundations of the formalism allow both correctness (i.e., logical) and efficiency (i.e., performance) analysis. Moreover, these models can be (automatically) implemented using a variety of techniques from hardware to software, and can be used for monitoring purposes once the system is readily working. In other words, they can be used all along in the life cycle of a system.

Rather than a single formalism, PN are a family of them, ranging from low to high level, each of them best suited for different purposes. In any case, they can represent very complex behaviours despite the simplicity of the actual model, consisting of a few objects, relations, and rules. More precisely, a PN model of a dynamic system consists of two parts:

1. A *net structure*, an inscribed bipartite directed graph, that represents the static part of the system. The two kinds of nodes are called places and transitions, pictorially represented as circles and boxes, respectively. The places correspond to the state variables of the system and the transitions to their transformers. The fact that they are represented at the same level is one of the nice features of PN compared to other formalisms. The inscriptions may be very different, leading to various families of nets. If the inscriptions are simply natural numbers associated with the arcs, named weights or multiplicities, *Place/Transition (P/T) nets* are obtained. In this case, the weights permit the modelling of bulk services and arrivals.

Notes of the EU-sponsored Jaca MATCH school

First topic: formalisms

1. Check the kind of system to analyze.
2. Choose formalisms, methods and tools.
3. Express system properties.
4. Model the system.
5. Apply methods.
6. Obtain verification results.
7. Analyze results.
8. Identify errors.
9. Suggest correction.

Concurrent Systems

- Involve several computation agents.
- Interaction through global, common variables or through message exchange (memoria condivisa vs scambio di messaggi)
- Global state or distributed state
- May involve remote components.
- May interact with users (Reactive).
- May involve hardware components (Embedded).

Problems in modeling concurrent systems

- Representing concurrency:
  - Allow one transition at a time, or
  - Allow coinciding transitions.
- Granularity of transitions.
  - Assignments and checks?
  - Application of methods?
- Global (all the system) or local (one thread at a time) states.

Formalisms considered

- *Petri nets* (reti di Petri).
- *Process algebra.* (algebra dei processi)
- *LTL* (Logica temporale lineare)
- *CTL* (Logica temporale branching)
- Language of guarded commands (nusmv modelling language)
- *Timed automata* (automi temporizzati o tempificati)

Specifying the *system* or its *properties*?

Petri nets

Formalism to describe

Discrete Events Dynamic Systems (DEDS)

Dynamic: the system is described through its evolution

Event: what cause a change of state

Discrete: system state described by discrete variables (or variables that are considered discrete (discretization). A discrete variable takes its value over natural numbers or over finite sets of element

Type of systems which are easily modelled with Petri Nets

- FMS (sistemi flessibili di produzione).
- Distributed algorithms of various sorts (per esempio i dining philosophers, e vari algoritmi di mutua esclusione)
- Control system (per esempio di un ascensore).
- Workflows
- Protocols.
- Any finite state automata

Petri nets - applets

- GreatSPN editor
- www.di.unito.it/~greatspn/index.html (contiene riferimento al sito github della nuova versione)
- www.di.unito.it/~amparore/mc4cslta/editor.html

- Give a look at the site *http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/java/*

Petri Nets (PN) definition

Petri nets + initial state = PN system

Definition 1: a Petri Net N is a 4-tuple

$$N = (P, T, F, W)$$

where

- P, set of *places* and T, set of *transitions*, are finite and non empty set and $P \cap T = \Phi$
- The *flow* relation $F \subset PxT \cup TxP$
- The *weight* function W: F --> N^+

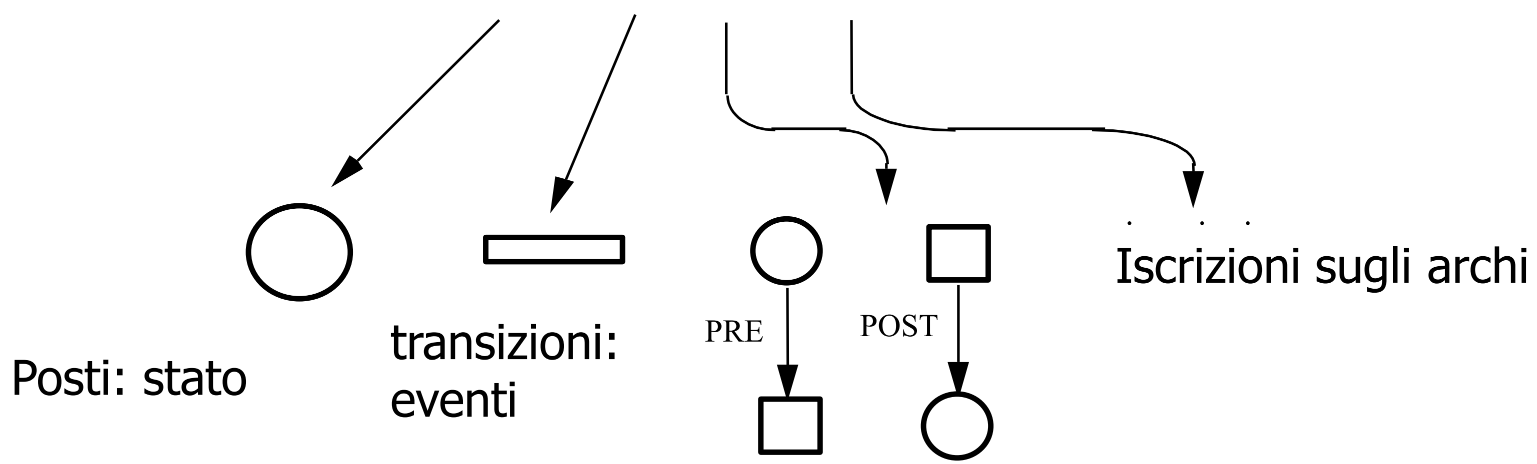
Petri Nets (PN) definition

- Places: state variables
- Transitions: change of state
- Marking: evaluation of the state variables

Petri Nets (PN) definition

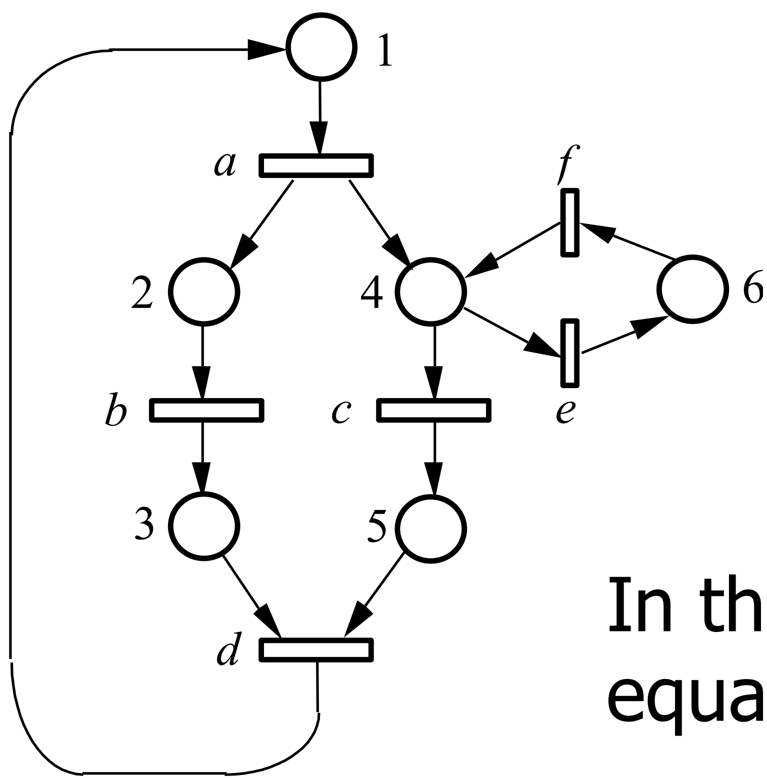
Petri nets have an easy visualization as bipartite graph

$$N = < P, \quad T, \quad F, \quad W >$$



Pre e post sono definiti rispetto alle transizioni

A first example of a PN



Any choice for names and transitions: it helps if names are distinct

In the example W is equal to the constant 1

Petri Nets (PN) definition in matrix form

Definition 2: a Petri Net N is a 4-tuple N = (P, T, Pre, Post)

where:

- P, set of *places,* and T, set of *transitions*, are finite and non empty set and $P \cap T = \Phi$
- The *Pre*-function Pre: PxT --> N
  - Pre(p,t) = W(p,t) if $(p,t) \in F$
  - = 0 if $(p,t) \notin F$

Input of the transition

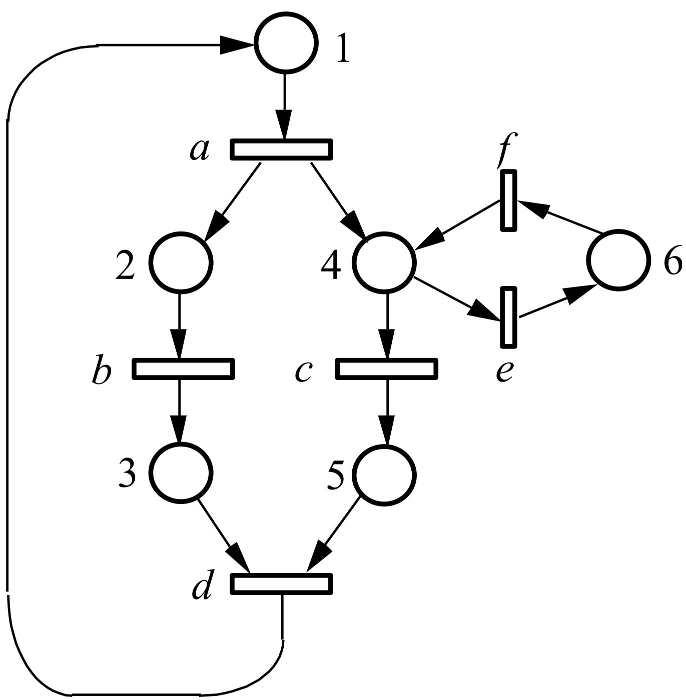
- The *Post*-function Post: PxT --> N
  - Post(p,t) = W(t,p) if $(t,p) \in F$
  - = 0 if $(t,p) \notin F$

Output of the transition

Alternative definition as vectors:

- $Pre \in N^{PxT}$
- $Post \in N^{PxT}$

A PN in matrix form



$$\mathbf{Pre} = \begin{array}{c} \\ p1 \\ p2 \\ p3 \\ p4 \\ p5 \\ p6 \end{array} \begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

$$\mathbf{Post} = \begin{array}{c} \\ p1 \\ p2 \\ p3 \\ p4 \\ p5 \\ p6 \end{array} \begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

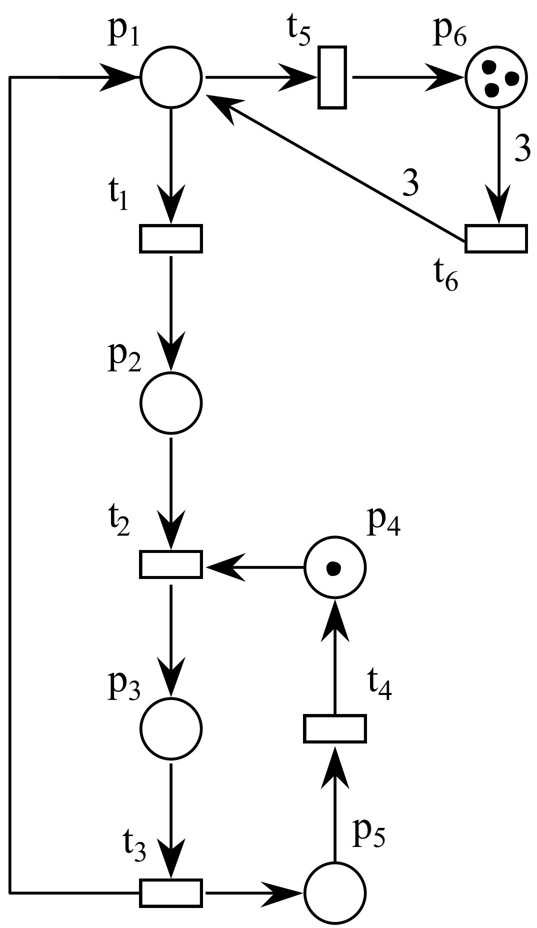
Petri Nets (PN) definition in matrix form

Based on the matrix representation of bipartite graph with weighted arcs:

- P: rows
- T: columns
- How many matrix do I need?
  1. one for Pre and one for Post?
  2. can I use a single one?

  incidence matrix C:PxT --> Z, C = Post- Pre

Another example



$$\mathbf{Pre} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

$$\mathbf{Post} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 3 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\mathbf{C} =$$

Marking

Petri nets + initial state = PN system

Definition: the *marking* (marcatura, stato) of a Petri Net N = (P, T, F, W) is a function

$$m: P \dashrightarrow N$$

Definition: the *marking* of a Petri Net N = (P, T, F, W) is a vector $m \in N^P$

Graphical representation: black dots (*tokens*) in places

m(p) = n is read as "there are n tokens in place p"

PN system

Petri nets + initial state = PN system

Definition: a *PN system* is a pair S = (N,m0) where

- N=(P, T, F, W) is a PN
- m0 is a marking *(initi*al marking)

Note: PN have a notion of "composite state": the state of the PN system is the union of the states of the single places

PN evolution

The evolution of the system is due to the *firing* of transitions

The firing of a transition change the marking in a formally defined manner

A transition can fire only if it is *enabled*

Definition: $t \in T$ is enabled in marking m iff

$m \geq Pre[-,t]$ (also written as $Pre[P,t]$)

$\forall p : (p,t) \in F , \quad W(p,t) \leq m(p)$

Definition: $t \in T$ enabled in marking m can fire, and its firing produce the marking m', with

$$m' = m + C[P,t]$$

State equation

$$m' = m + Post[P,t] - Pre[P,t]$$

PN and concurrency structures

Fork: a task Tk activates two of more tasks Tk_1, ..., Tk_n.

Join: two or more tasks synchronize into a single task

PN and concurrency structures

Choice (distribution): in a given (local) state there is a choice between executing event e_1 or event e_2 or …..event e_n

Collection: event e_1, e_2 , …..and e_n lead to the same local state

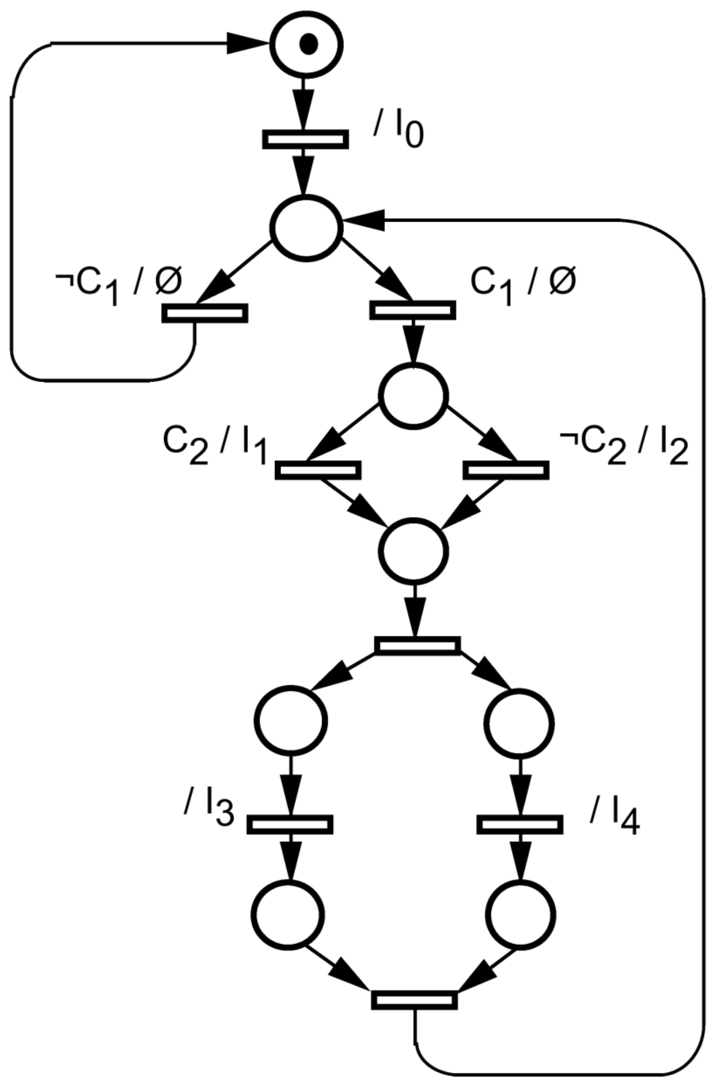
PN and concurrency structures

An event causing another event

Two concurrent events

PN and concurrency structures

Flow-chart




```
loop
    I0
    while C1 do
        if C2
            then I1
            else I2
        par_begin
            I3
            I4
        par_end
    end
end
```

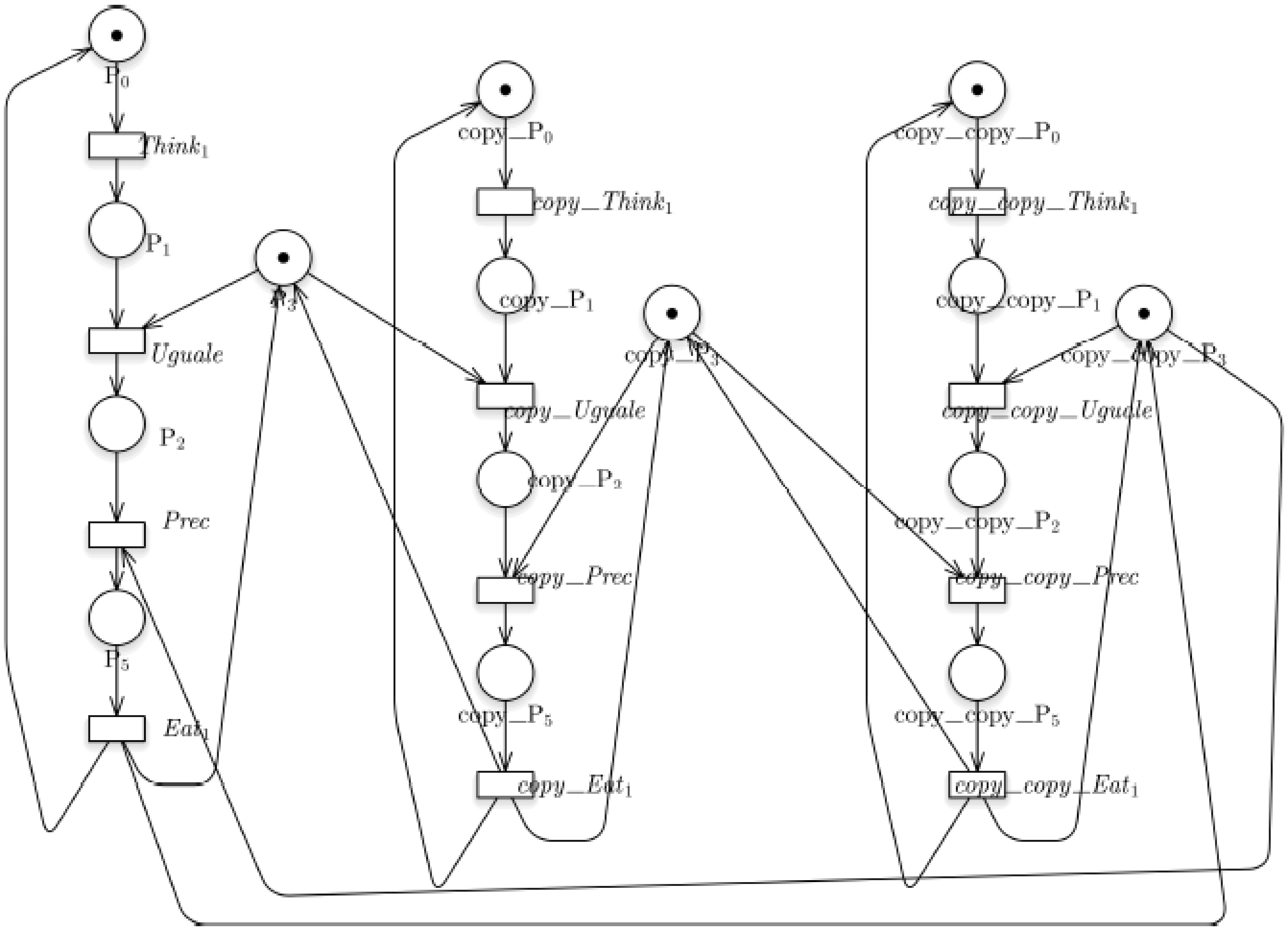
I 5 filosofi (da S.O.)

Vedi modello dei filosofi nella distribuzione di GreatSPN

Per accedere alla libreria dei modelli:

- attivate l'interfaccia grafica di GreatSPN
- create un progetto (se non ne avete già uno aperto)
- cliccate sull'icona ``add a new page page to the active project"
- scegliete ``add a library model"
- selezionate il modello dei filosofi (attenzione, ce ne sono due, uno colorato e uno con le reti P/T, che è quello da usare in questa fase)

I 3 filosofi (rete costruita a lezione)



PN evolution through a firing sequence

Definition: $\sigma = [t1,..,tk]$, with $ti \in T$, is a *firing sequence* in marking m, and we write $m\ [\sigma >m'$ iff $\exists$ a set of marking $\{m_0,.., m_k\}$: $\forall i \in [1..k]$, $m_{i-1}[ti>m_i$

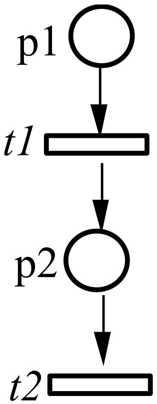
and we say that m' is reachable from m through σ.

Language of a PN

Definition: Given a P/T system $S=(N, m_0)$, the language L(S) is defined as

$L(S) = \{\sigma = (t1,..,tk)$, s.t. σ is a firing sequence for S in $m_0\}$

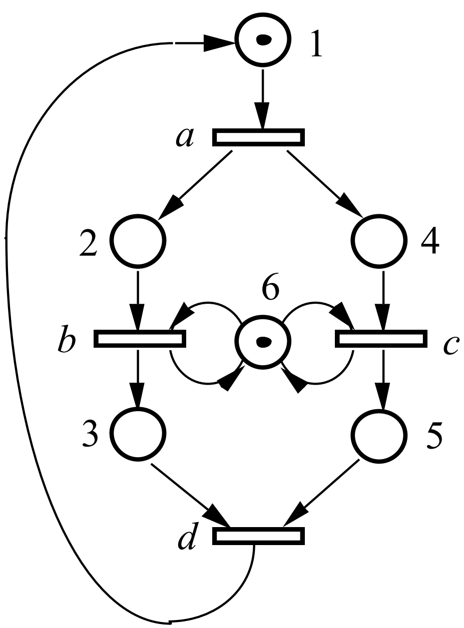
Example with m0= 2•p1, L(N, m0) ={t1, t1t2, …, t1t1t2t2, t1t2t1t2,..}



State space of a PN system

Definition: the reachability set of a PN system $S=(N,m_0)$, $RS(S)$ or $RS(N,m_0)$, or $RS_N(m_0)$ is the set of all marking reachable from m0 through a firing sequence of L(S)

$$RS_N(m_0) = \{ m: \exists \sigma \in L(N,m_0) \text{ s.t. } m_0 [\sigma > m \}$$



$RS_N(m_0) =$

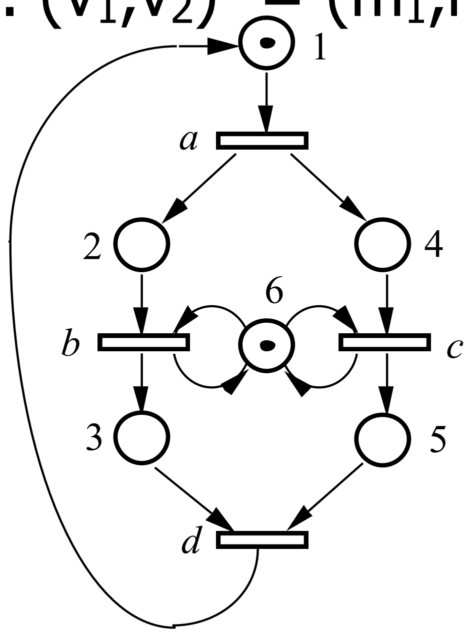
{ p1+p6,
p2+p4+p6,
p3+p4+p6,
p2+p5+p6,
p3+p5+p6 }

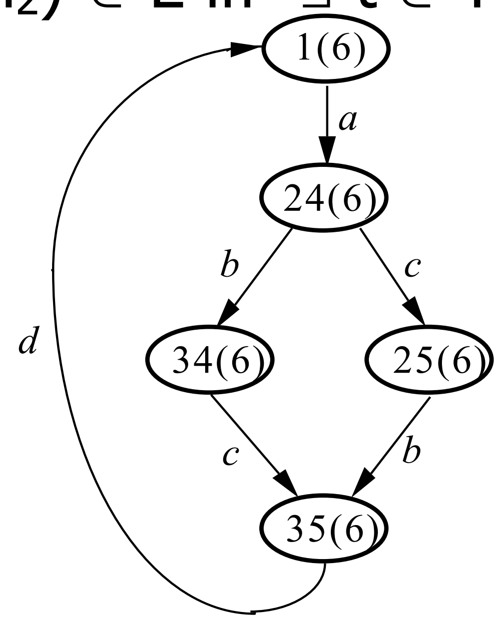
State space of a PN system

Definition: the reachability graph of a PN system $S=(N,m_0)$, $RG(S)$ or $RG(N,m_0)$, or $RG_N(m_0)$ is the direct graph defined as follows:

$RG_N(m_0) = (V,E)$, where

1. $V=RS_N(m_0)$
2. $(v_1,v_2) = (m_1,m_2) \in E$ iff $\exists\, t \in T$ s.t. $m_1[t>m_2$

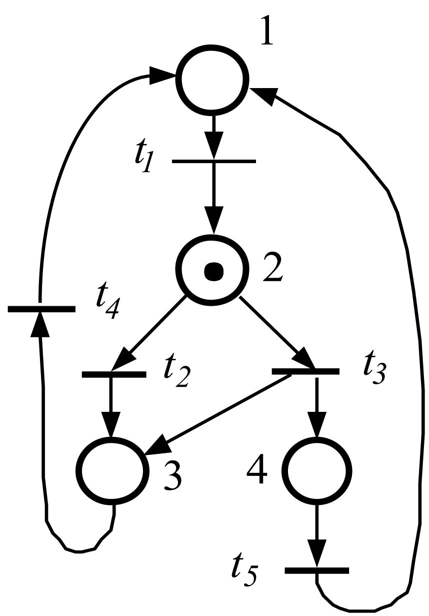


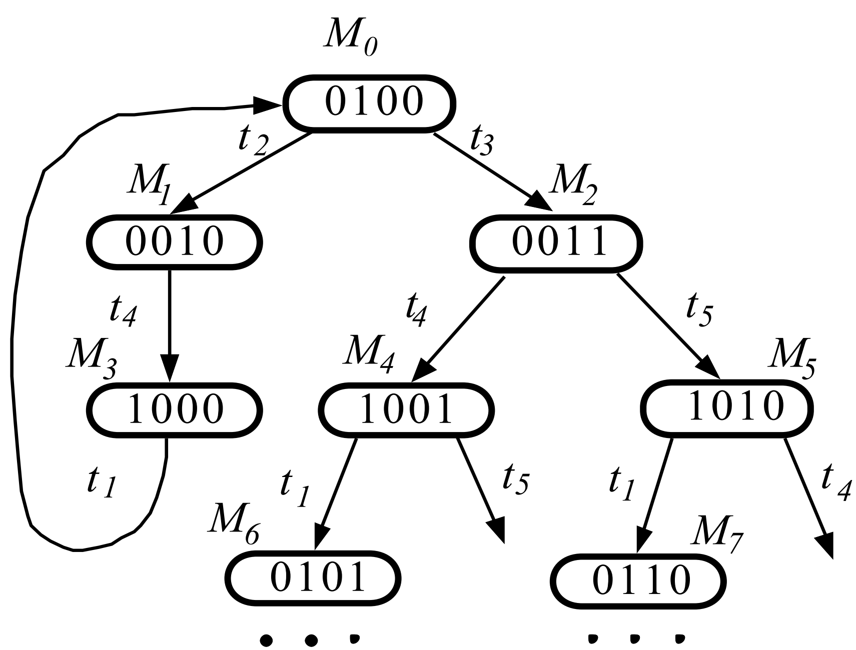


State space of a PN system - some basic properties

Def.: A system is finite iff the RG is finite

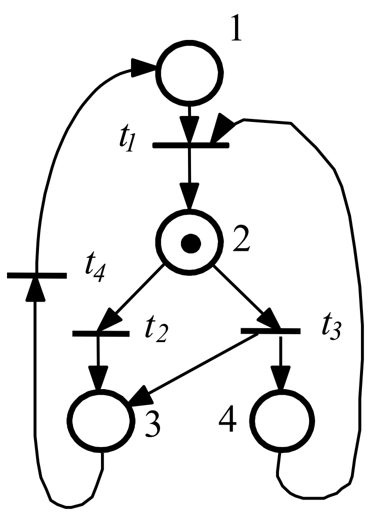
The PN system below is not finite





State space of a PN system - some basic properties

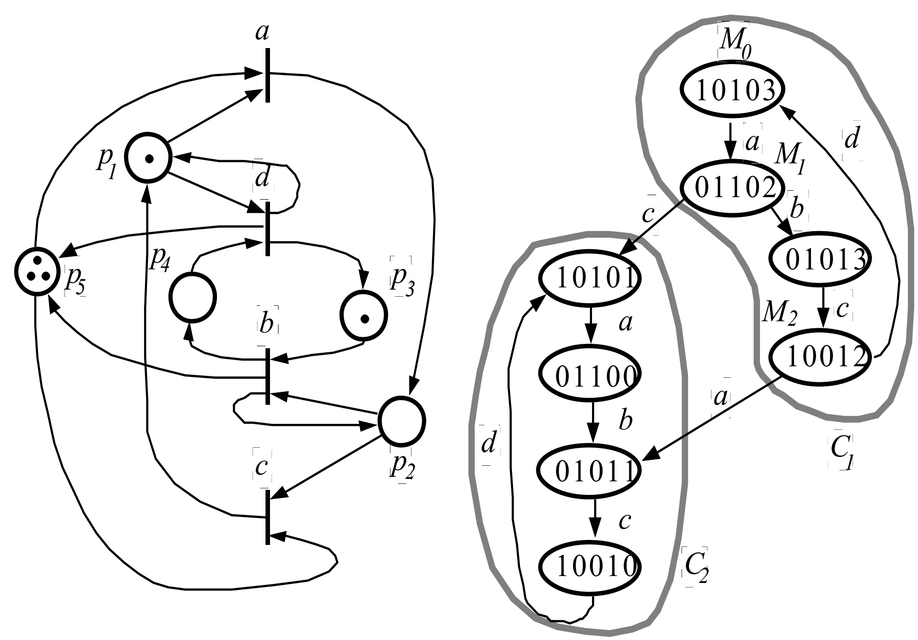
- A system exhibits <u>absence of deadlock</u> iff it does not exist a reachable state that does not enable at least a transition (all reachable states enable at least a transition)
- The PN system below has a deadlock



State space of a PN system - some basic properties

A PN system is live if, for all reachable states m and for all transitions t, it is possible to reach a state in which t is enabled

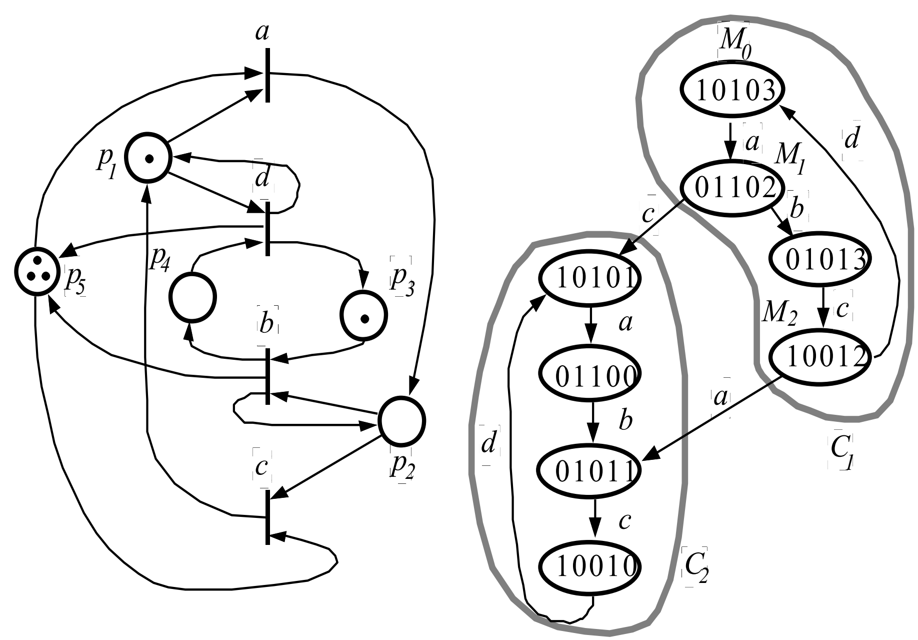
The PN system below is live, because in each BSCC of the RG it is possible to fire all transitions



State space of a PN system - some basic properties

A PN <u>system is reversible</u> if, for all reachable states m, it exists a firing sequence, firable in m, that leads to the initial marking

The PN system below is not reversible (there are two SCC)



Other Petri nets classes

We distinguish subclasses (restriction of the basic PN formalism) and superclasses (extensions)

Example of subclasses: state machines, marked graphs (no choice), free choice, ordinary nets

Example of superclasses: nets with inhibitor arcs, nets with priorities, colored nets

Subclass --> same enabling and firing rule

Superclass --> modified enabling and/or firing rule

Subclass --> more analysis techniques, less expressive power

Superclass --> (usually) less analysis techniques, more expressive power

Petri nets subclasses - ordinary vs. weighted

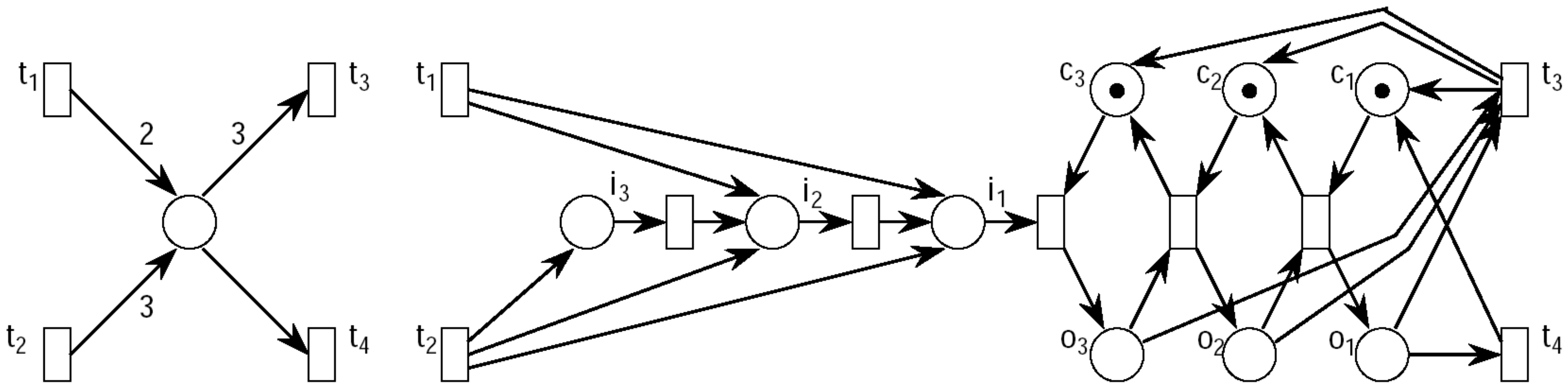


Figure 2.8: Ordinary implementation of a weighted net.

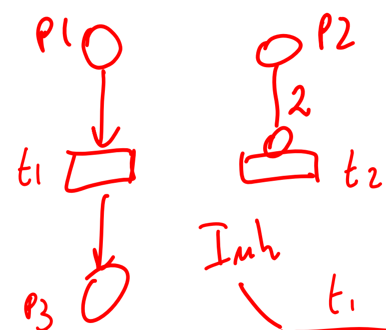
Superclass: PN with inhibitor arcs

Definition: a Petri Net N with inhibitor arcs is a 5-tuple

$$N = (P, T, Pre, Post, Inh)$$

where:

- P, set of places, and T, set of transitions, are finite and non empty set and $P \cap T = \Phi$
- Pre is the *Pre*-function, Pre: PxT --> N
- Post is the *Post*-function, Post: PxT --> N
- *Inh* is the Inhibitor-function, Inh: PxT --> $N^{+} \cup \infty$



Inh	t_1	t_2
p_1	∞	∞
p_2	∞	2
p_3	∞	∞

Def: a transition t is enabled in m if

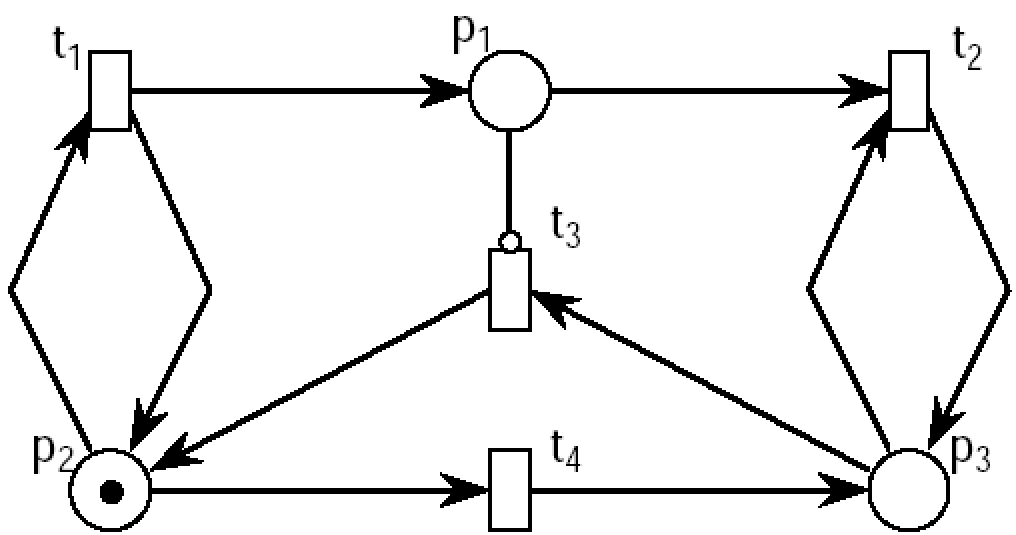
$$m \geq Pre[-,t] \quad \text{and} \quad \underbrace{m < Inh[-,t]}_{\text{modifica}}$$

Definition: the firing of $t \in T$ in m produce the marking m', with

$$m' = m + C[P,t]$$

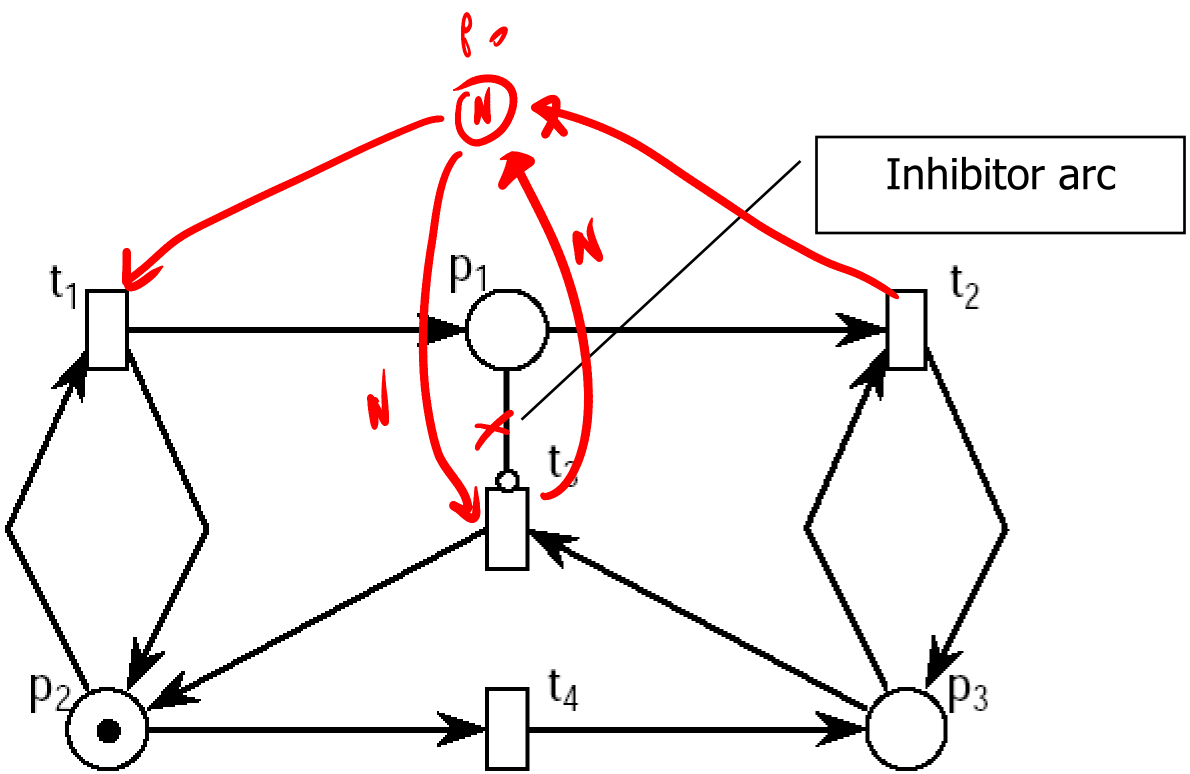
Superclass: example of PN with inhibitor arcs

With inhibitor arc



Superclass: example of PN with inhibitor arcs

Example of the lazy lad (scapolo pigro): he prepares a number of dishes, and then eats everything from the fridge until it is empty. Then he starts cooking again



Superclass: PN with priorities

Definition: a Petri Net N with priorities is a 5-tuple

$$N = (P, T, Pre, Post, Pri)$$

where:

- P,T, Pre and Post as usual
- *Pri* is the priority function, Pri:T --> N

Def: a transition t *has concession* in m if

$$m \geq Pre[-,t]$$

Def: a transition t *is enabled* in m if

$$t \text{ has concession and , } \forall t' \text{ with concession in m, } Pri(t) \geq Pri(t')$$

Note: firing unchanged

Note: PN and local enabling

Superclass: example of PN with priorities

The lazy lad with priorities