Petri nets. An introduction

Javier Campos
Universidad de Zaragoza
http://webdiis.unizar.es/~jcampos/

Outline

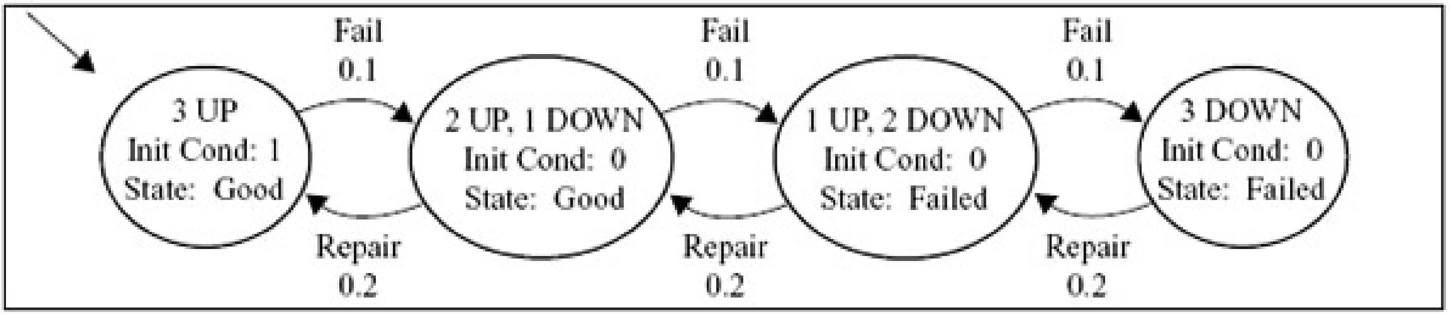
- Basic concepts
- Definitions
- Functional properties and analysis

Basic concepts

- Global versus local models

  A system has three identical components. Each of these components is repairable and fails with the same probability.

  - In a Markov Chain, the circles or states represent **all** the components in that model.



Each state represents the entire system in a particular combination of conditions (global model).

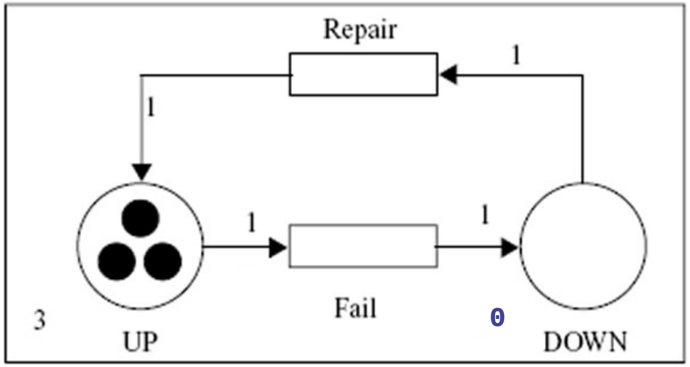
Each node in the graph represents a state = poor abstract level.

Basic concepts

- Global versus local models

  For the same application…

  - The equivalent Petri Net…



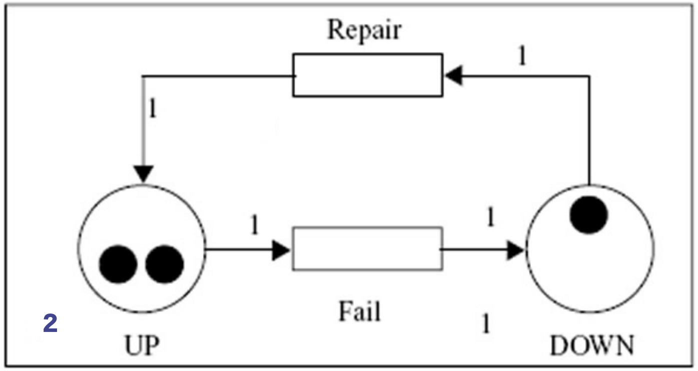
Each component is represented with one or several "state nodes" (called *places*).
The state is distributed. The marking of all the *places* represents the state (local model) = higher level of abstracion.

Basic concepts

- Global versus local models

  For the same application…

  - The equivalent Petri Net…



Each component is represented with one or several "state nodes" (called *places*).

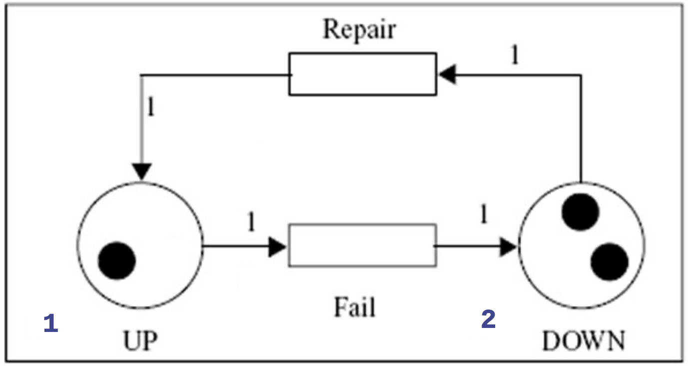
The state is distributed. The *marking* of all the *places* represents the state (local model) = higher level of abstracion.

Basic concepts

- Global versus local models

  For the same application...

  - The equivalent Petri Net...



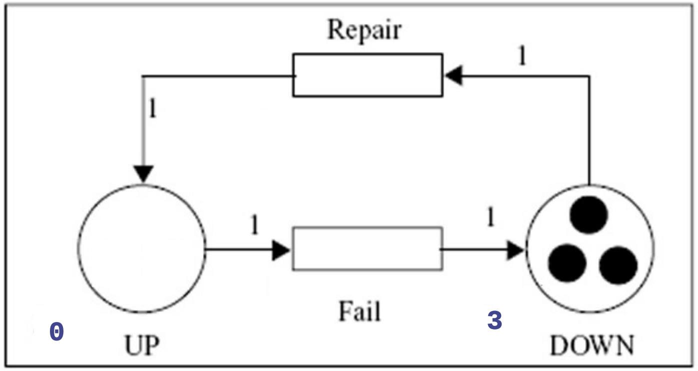
Each component is represented with one or several "state nodes" (called *places*).
The state is distributed. The marking of all the *places* represents the state (local model) = higher level of abstracion.

Basic concepts

- Global versus local models

  For the same application...

  - The equivalent Petri Net...



Each component is represented with one or several "state nodes" (called *places*).
The state is distributed. The marking of all the *places* represents the state (local model) = higher level of abstracion.

Basic concepts

- Petri nets:

  A formal, graphical, executable technique for the specification and analysis of concurrent, discrete-event dynamic systems; a technique undergoing standardisation.

  http://www.petrinets.info/

Basic concepts

- Formal:

  The technique is mathematically defined. Many static and dynamic properties of a PN (and hence a system specified using the technique) may be mathematically proven.

Basic concepts

- Graphical:

  The technique belongs to a branch of mathematics called graph theory.

  A PN may be represented graphically as well as mathematically.

  The ability to visualise structure and behaviour of a PN promotes understanding of the modelled system.

  Software tools exist which support graphical construction and visualisation.

Basic concepts

- Executable:

  A PN may be executed and the dynamic behaviour observed graphically.

  PN practitioners regard this as a key strength of the PN technique, both as a rich feedback mechanism during model construction and as an aid in communicating the behaviour of the model to other practioners and lay-persons.

  Software tools exist which automate execution.

Basic concepts

- Specification:

  System requirements expressed and verified (by formal analysis) using the technique constitute a formal system specification.

Basic concepts

- Analysis:
  A specification in the form of a PN model may be formally analysed, to verify that static and dynamic system requirements are met.

  Methods available are based on Occurrence graphs (state spaces), Invariants and Timed PN. The inclusion of timing enables performance analysis.

  Modelling is an iterative process. At each iteration analysis may uncover errors in the model or shortcomings in the specification. In response the PN is modified and re-analysed. Eventually a mathematically correct and consistent model and specification is achieved.

  Software tools exist which support and automate analysis.

Basic concepts

- Concurrent:

  The representation of multiple independent dynamic entities within a system is supported naturally by the technique, making it highly suitable for capturing systems which exhibit concurrency, e.g., multi-agent systems, distributed databases, client-server networks and modern telecommunications systems.

Basic concepts

- Discrete-event dynamic system:

  A system which may change state over time, based on current state and state-transition rules, and where each state is separated from its neighbour by a step rather than a continuum of intermediate infinitesimal states.

  Often falling into this classification are information systems, operating systems, networking protocols, banking systems, business processes, telecommunications systems, population systems, chemical networks, many biological systems…

Basic concepts

- Standardisation:

  - 2004-12-02

    Achieved Published Standard status:

    ISO/IEC 15909-1:2004 Systems and software engineering - High-level Petri nets - Part 1: Concepts, definitions and graphical notation. Available from ISO.

  - 2011-02-14

    Achieved Published Standard status:

    ISO/IEC 15909-2:2011 Systems and software engineering - High-level Petri nets - Part 2: Transfer format. Available from ISO.

Definition

- Graphical representations

  Useful to inform about model structure

a picture is better than a thousand words

- Continuous systems:
  - Circuits diagrams
  - Block diagrams
  - Bond graphs
  - ...
- Discrete event systems:
  - State diagrams →
    → Markov chains
  - Algorithmic state machines
  - PERTs
  - QNs
  - ...

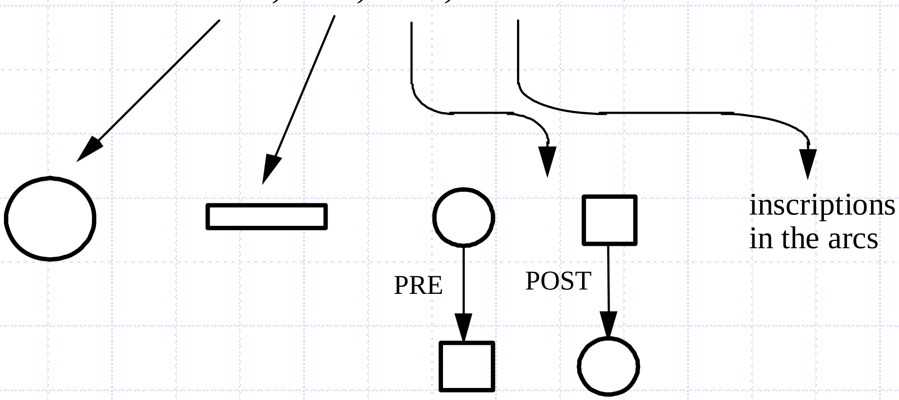
Definition

- In Petri Nets: two basic concepts ($\rightarrow$ graphical objects)
  - states/data (PLACES)
  - actions/algorithms (TRANSITIONS)
  - + weight (labeling) of the arcs

Definitions

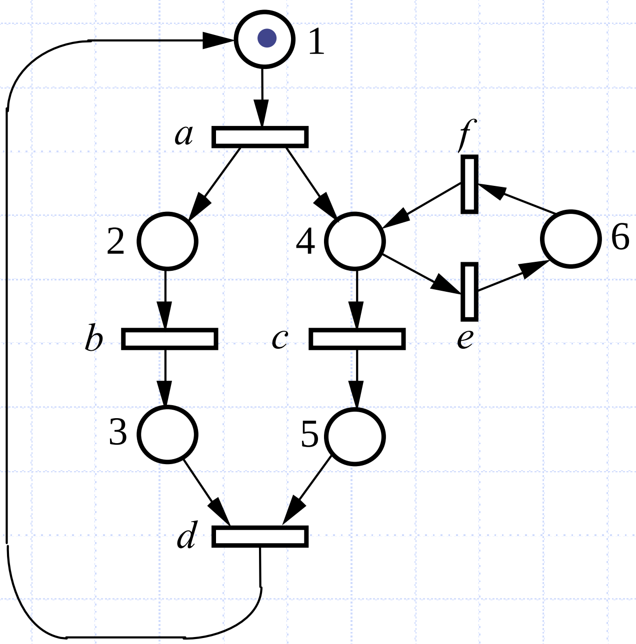
- Autonomous Petri nets (place/transition nets or P/T nets)
  - Petri Nets is a bipartite valued graph
    - Places: states/data (P)
    - Transitions: actions/algorithms (T)
    - Arcs: connecting places and transitions (F)
    - Weights: labeling the arcs (W)

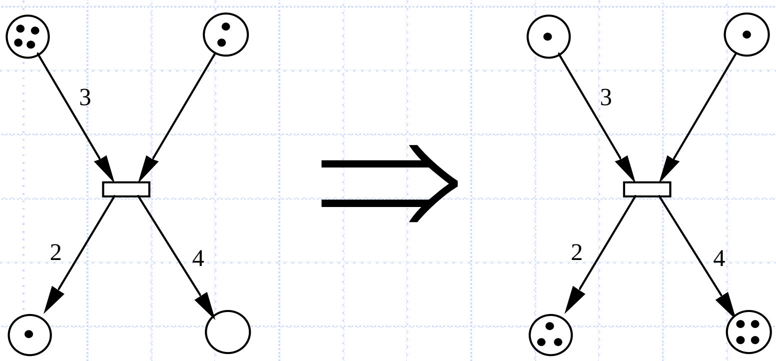
$$N = < P, \quad T, \quad F, \quad W >$$



Definitions

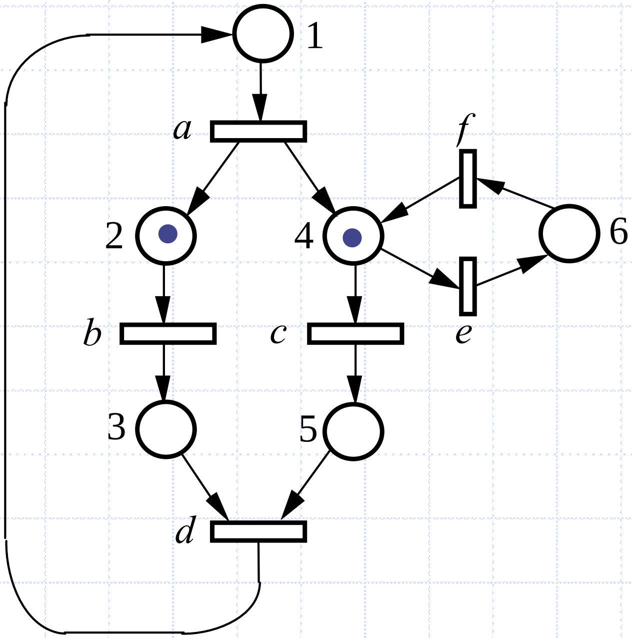
- Net ➔ Static part
  - Places : State variables (names)
  - Transitions: Changes in the state (conditions)
- Marking ➔ Dynamic part
  - Marking : State variables (values)
- Event/Firing
  - **Enabling**: the pre-condition is verified
  - **Firing**: change in the marking
  - the pre-condition "consumes" tokens
  - the post-condition "produces" tokens





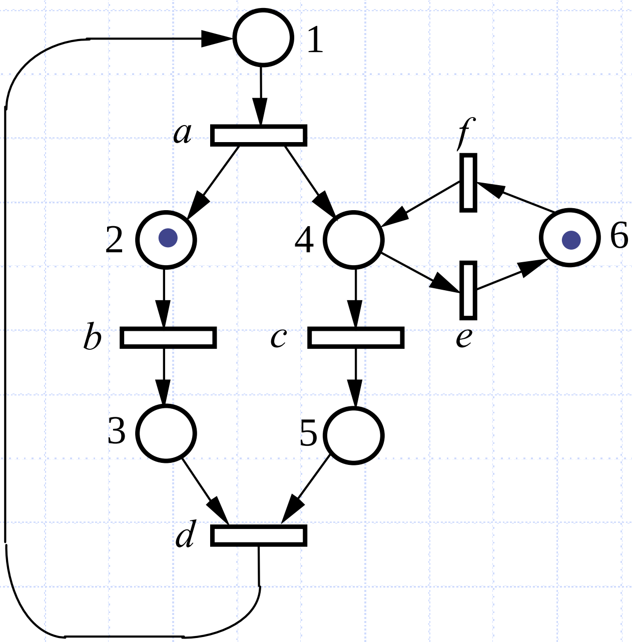
Definitions

- Net ➔ Static part
  - Places : State variables (names)
  - Transitions: Changes in the state (conditions)
- Marking ➔ Dynamic part
  - Marking : State variables (values)
- Event/Firing
  - **Enabling**: the pre-condition is verified
  - **Firing**: change in the marking
  - the pre-condition "consumes" tokens
  - the post-condition "produces" tokens



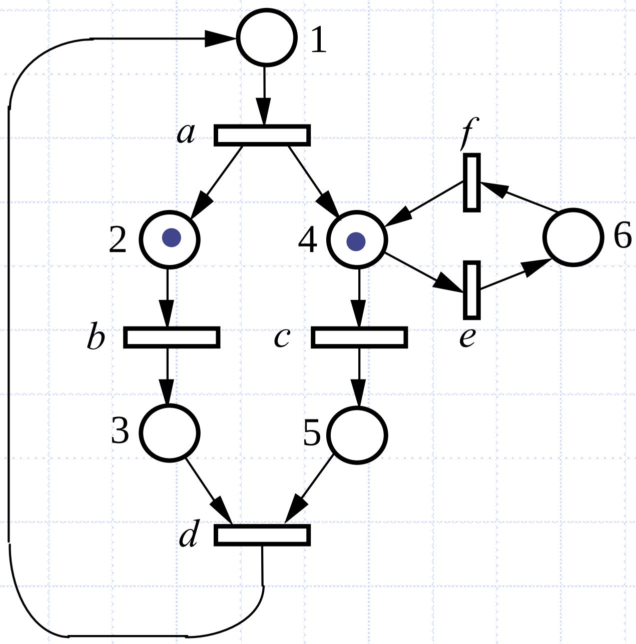
Definitions

- Net ➔ Static part
  - Places : State variables (names)
  - Transitions: Changes in the state (conditions)
- Marking ➔ Dynamic part
  - Marking : State variables (values)
- Event/Firing
  - **Enabling**: the pre-condition is verified
  - **Firing**: change in the marking
  - the pre-condition "consumes" tokens
  - the post-condition "produces" tokens



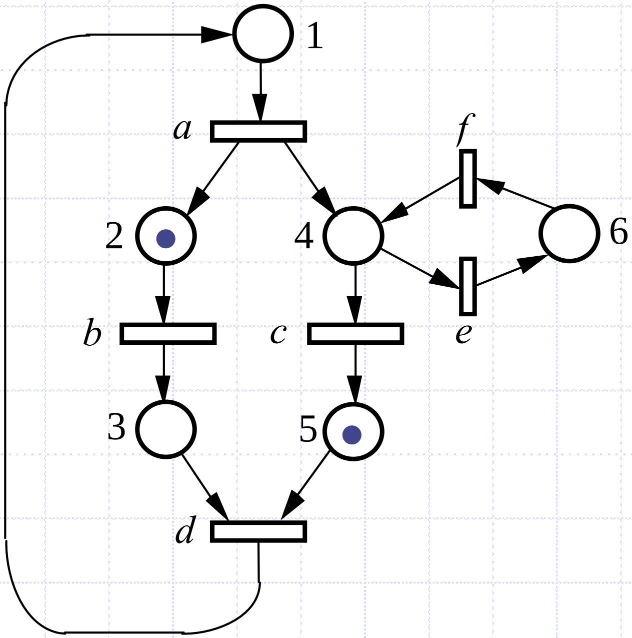
Definitions

- Net ➔ Static part
  - Places : State variables (names)
  - Transitions: Changes in the state (conditions)
- Marking ➔ Dynamic part
  - Marking : State variables (values)
- Event/Firing
  - **Enabling**: the pre-condition is verified
  - **Firing**: change in the marking
  - the pre-condition "consumes" tokens
  - the post-condition "produces" tokens



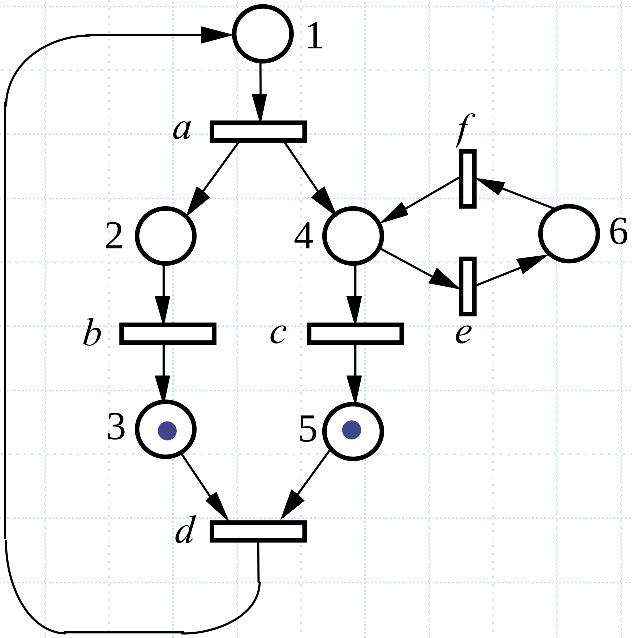
Definitions

- Net ➔ Static part
  - Places : State variables (names)
  - Transitions: Changes in the state (conditions)
- Marking ➔ Dynamic part
  - Marking : State variables (values)
- Event/Firing
  - **Enabling**: the pre-condition is verified
  - **Firing**: change in the marking
  - the pre-condition "consumes" tokens
  - the post-condition "produces" tokens



Definitions

- Net → Static part
  - Places : State variables (names)
  - Transitions: Changes in the state (conditions)
- Marking → Dynamic part
  - Marking : State variables (values)
- Event/Firing
  - **Enabling**: the pre-condition is verified
  - **Firing**: change in the marking
  - the pre-condition "consumes" tokens
  - the post-condition "produces" tokens



Definitions

- PN and its algebraic representation based on state equation
  - Linear representation of PNs, the structure:

$$\mathcal{N} = < P, T, \mathbf{Pre}, \mathbf{Post} >$$

  - Pre-incidence matrix

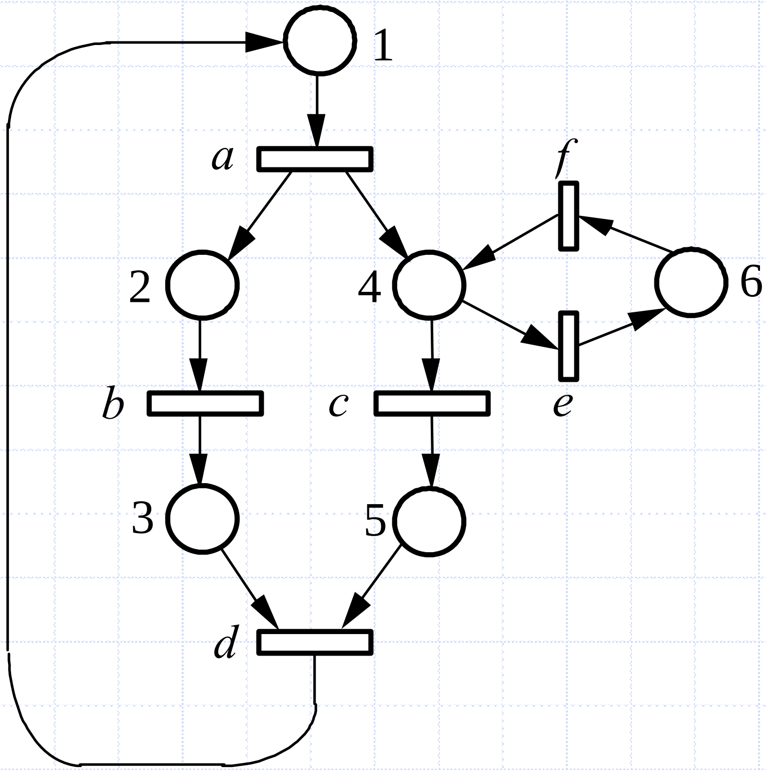
$$\mathbf{Pre}(p,t): PxT \rightarrow \mathbb{N}^{+}$$

  - Post-incidence matrix

$$\mathbf{Post}(p,t): PxT \rightarrow \mathbb{N}^{+}$$

  - Incidence matrix, C = Post – Pre
    (marked) Petri Net is finally defined by: $\Sigma = \langle \mathcal{N}, m_0 \rangle$

Definitions



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

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

Incidence matrix C (= Post - Pre) ➔ cannot "see" self loops

Definitions

- State equation definition

$$m(k)\,[t > m(k+1) \Leftrightarrow \begin{array}{l} m(k+1) = m(k) + \mathbf{C}(t) = \\ = m(k) + \mathbf{Post}(t) - \mathbf{Pre}(t) \geq 0 \end{array}$$

Integrating in one execution (sequence of firing)

$$m_0\ [\sigma > m(k) \Rightarrow \boxed{m(k) = m_0 + \mathbf{C} \cdot \boldsymbol{\sigma}}$$

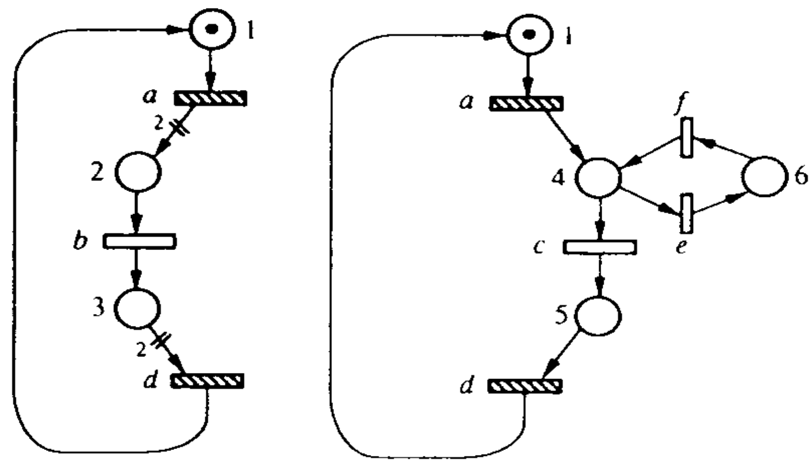
where $\boldsymbol{\sigma}$ (bold) is the firing counting vector of σ

Very important: unfortunately...

$$m(k) = m_0 + \mathbf{C} \cdot \boldsymbol{\sigma} \geq 0,\ \ \boldsymbol{\sigma} \geq 0 \not\Rightarrow m_0\ [\sigma > m(k)$$

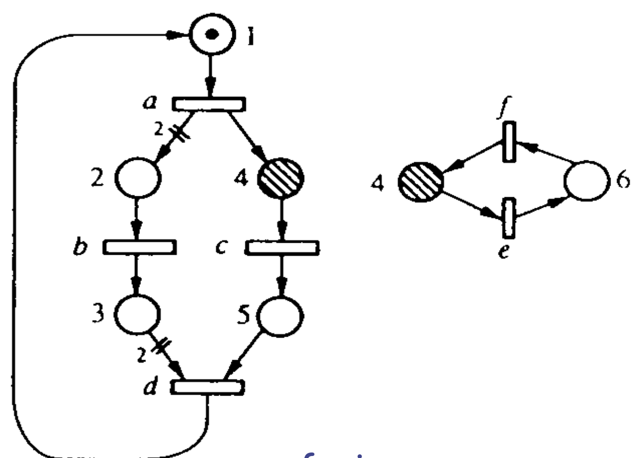
Definitions

- Design methodologies:
  1. Parallel composition by…



synchronization

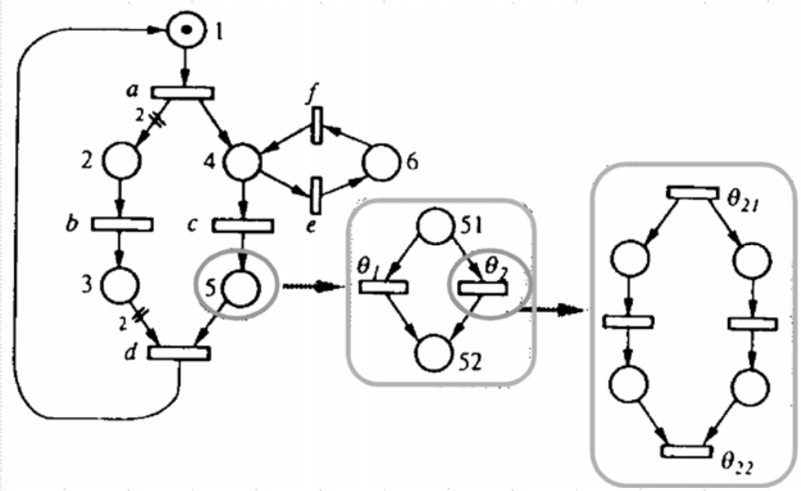
+ bottom-up methodology



fusion

Definitions

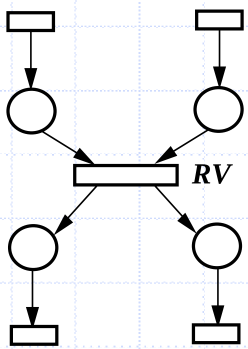
- Design methodologies (cont):
  2. Sequential composition by refinement



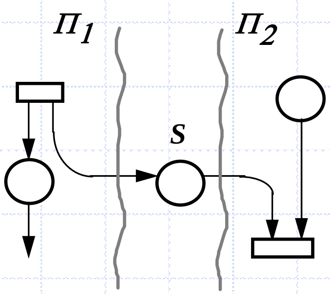
+ top-down methodology

Definitions

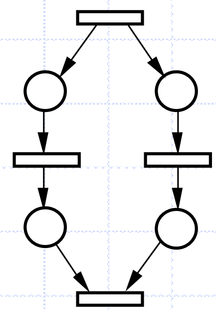
- Design methodologies (cont): typical synchronization schemes



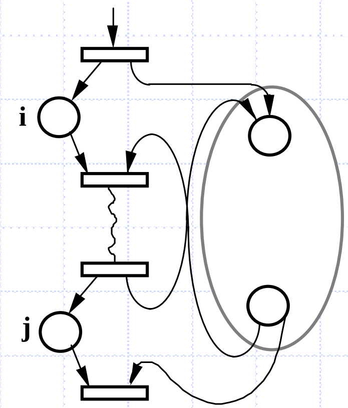
1. Rendezvous, RV



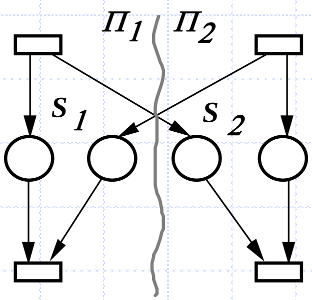
2. Semáforo, S



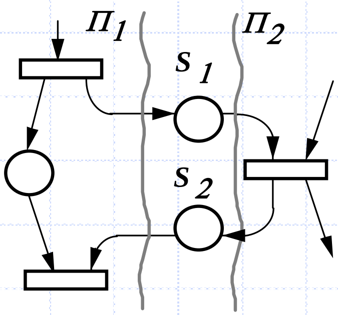
5. Fork-Join



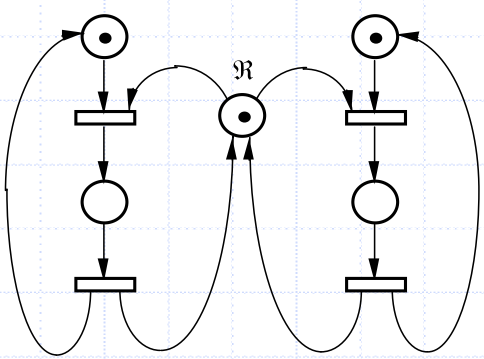
6. Sub programa
(p_i ,p_j están en mutex)



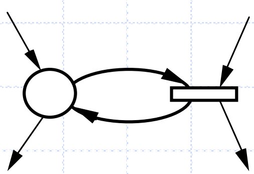
3. RV/Semáforo simétrico



4. RV/Semáforo asimétrico
(master/slave)



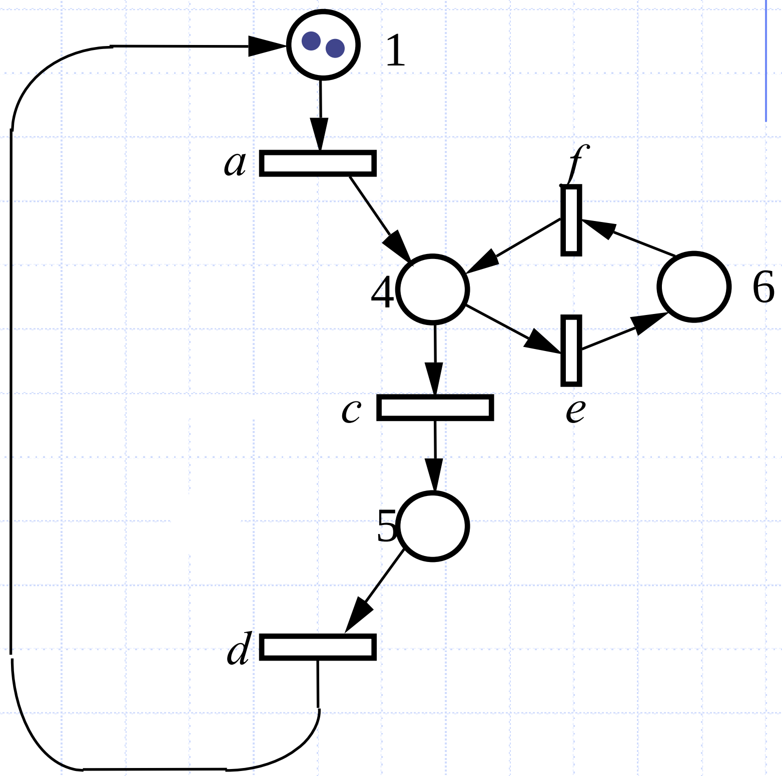
7. Recurso compartido ($\Re$)



8. Guarda (condición
de lectura)

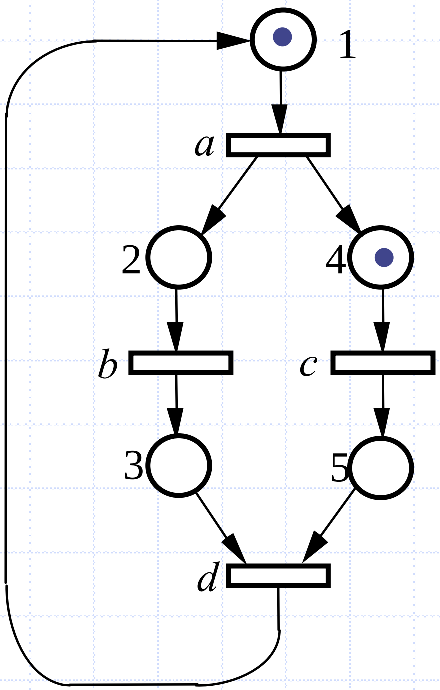
Definitions

- PN syntactic subclasses
  - **State machines**
    - Subclass of ordinary PN (arc weights = 1)
    - Neither synchronizations nor structural parallelism allowed
    - Model systems with a finite number of states
    - Their analysis and synthesis theory is well-known



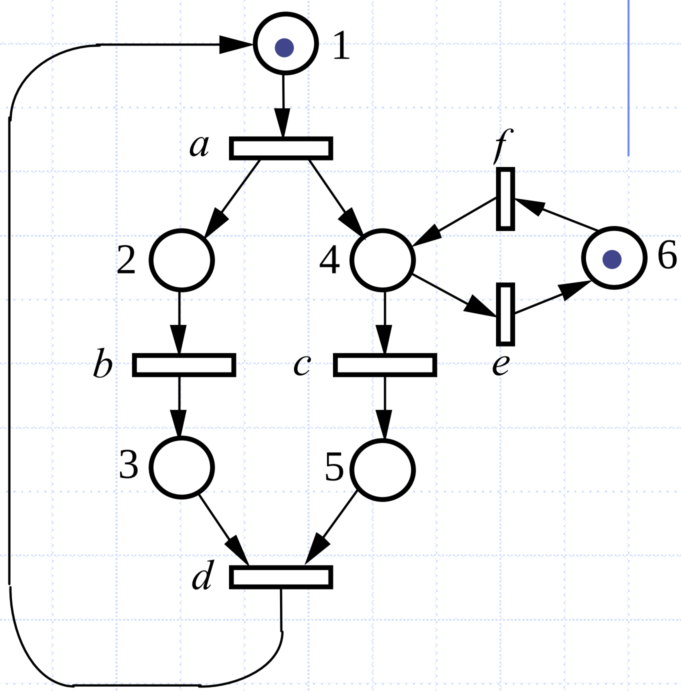
Definitions

- PN syntactic subclasses (cont.)
  - **Marked Graphs**
    - Subclass of ordinary PN (arc weights = 1)
    - Allow synchronizations and parallelism but not allow decisions
    - No conflicts present
    - Allow the modeling of infinite number of states
    - Their analysis and synthesis theory is well-known



Definitions

- PN syntactic subclasses (cont.)
  - **Free-Choice nets**
    - Subclass of ordinary PN (arc weights = 1)
    - Allow synchronizations, parallelism and choices
    - Choices and synchronizations cannot be present in the same transition
    - Their analysis and synthesis theory is well-known



- There are other syntactic subclasses…

Functional properties and analysis

- Functional basic properties
  - **Boundedness**: finiteness of the state space, i.e. the marking of all places is bounded

$$\forall p \in P \quad \exists k \in N \ \text{ such that } \ \mathbf{m}(p) \leq k$$

  - **Safeness** = 1-boundedness (binary marking)
  - **Mutual Exclusion**: two or more places cannot be marked simultaneously (problem of shared resources)
  - **Deadlock**: situation where there is no transition enabled
  - **Liveness**: infinite potential activity of all transitions

$$\forall t \in T, \forall \mathbf{m} \text{ reachable}, \ \exists \mathbf{m}', \mathbf{m}\, [\sigma > \mathbf{m}' \ \text{ such that } \ \mathbf{m}'\, [t >$$

  - **Home state**: a marking that can be recovered from every reachable marking
  - **Reversibility**: recovering of the initial marking

$$\forall \mathbf{m} \text{ reachable}, \ \exists \sigma \ \text{ such that } \ \mathbf{m}\, [\sigma > \mathbf{m}_0$$

Functional properties and analysis

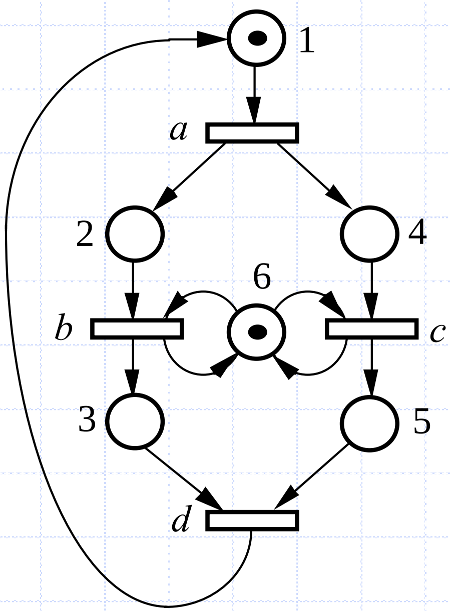
- Structural basic properties:
  - N is **structurally bounded** if for all m_0, $\langle N, m_0 \rangle$ is bounded
  - N is **structurally live** if there exists a m_0 for which $\langle N, m_0 \rangle$ is live

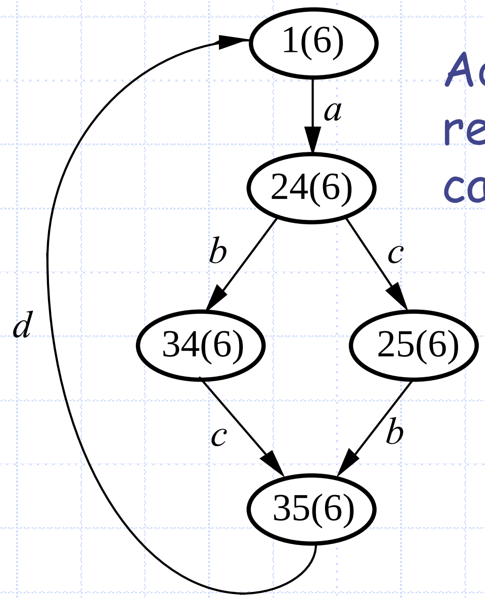
Functional properties and analysis

- Analysis techniques (for the computation of functional properties)
  - Enumerative: based on reachability graph
  - Structural: based on the structure of the model, considering m_0 as a parameter
  - Reduction/transformation: rules that preserve a given property and simplify the model

Functional properties and analysis

- Enumerative analysis: exhaustive sequential enumeration of reachable states
  - Problem 1: state explosion problem
  - Problem 2: lost of information about concurrent behaviour



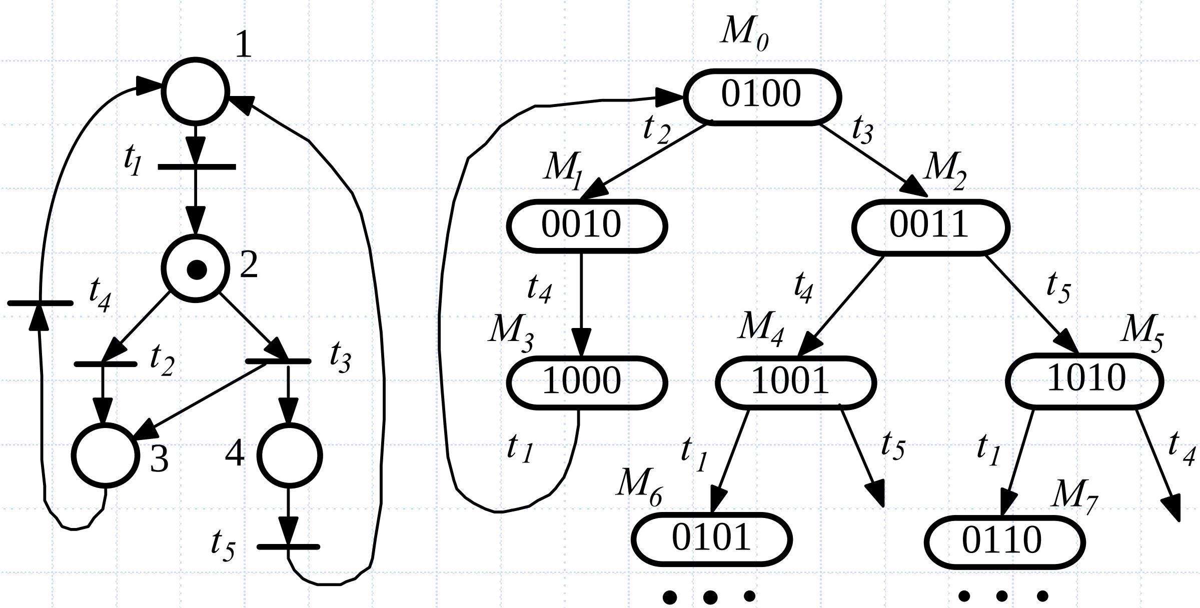


Adding place 6 does not modify reachability graph but *b* and *c* cannot fire simultaneously.

reachability graph

Functional properties and analysis

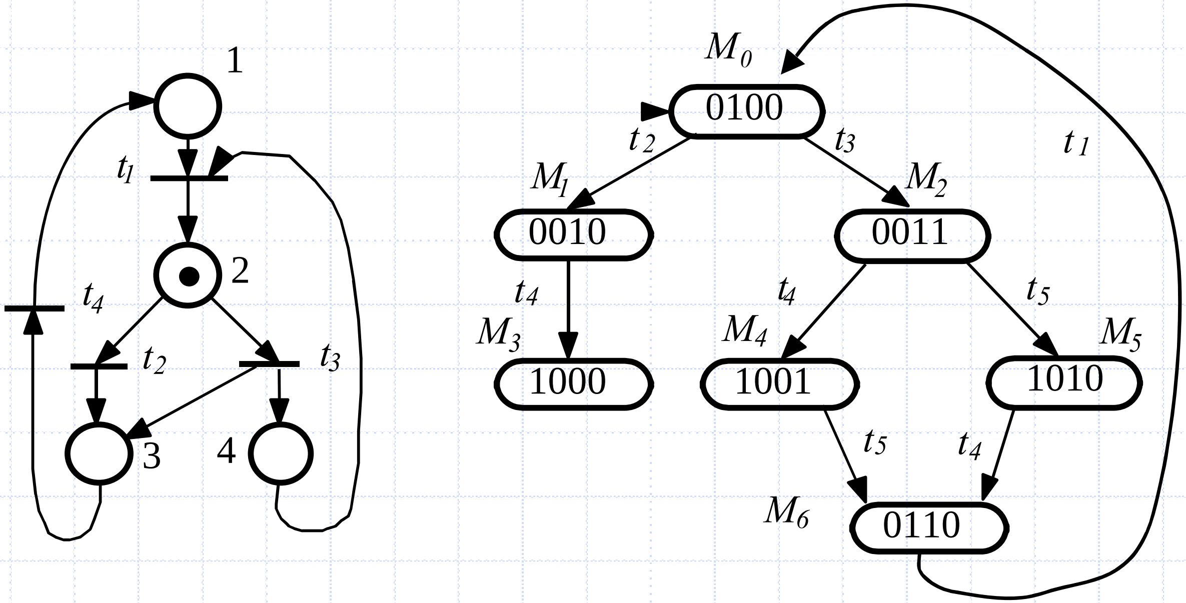
- Enumerative analysis (cont.):
  - Bounded system ⇔ finite reachability graph



unbounded system

Functional properties and analysis

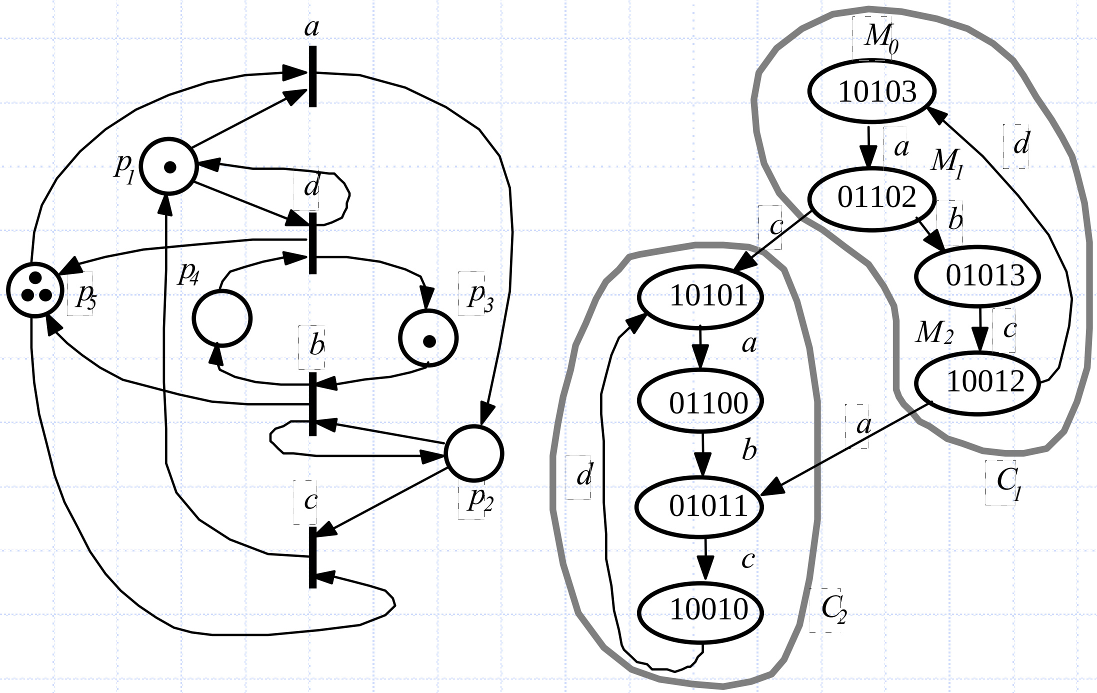
- Enumerative analysis (cont.):
  - Deadlock exists ⇔ There exists a terminal node in the RG



M_3 is a deadlock

Functional properties and analysis

- Enumerative analysis (cont.):
  - Live net ⇔ in all the strongly connected components of the RG all transitions can be fired
  - Reversible net ⇔ there is only one strongly connected component in the RG



live and non-reversible system

Functional properties and analysis

- Structural analysis:
  - Based either on convex geometry (linear algebra and linear programming), or
  - Based on graph theory

  → We concentrate on first approach.

- Definitions:

  P-semiflow: $y \geq 0,\ \ y^{T}.C = 0$

  T-semiflow: $x \geq 0,\ \ C.x = 0$

Functional properties and analysis

- Properties:
  1. If y is a P-semiflow, then the next token conservation law holds (or P-invariant):

     $$\text{for all } m \in RS(N, m_0) \text{ and for all } m_0 \Rightarrow$$
     $$\Rightarrow y^T.\ m = y^T.\ m_0.$$

     Proof: if $m \in RS(N, m_0)$ then $m = m_0 + C.\sigma$, and pre-multiplying by y^T:

     $$y^T.\ m = y^T.\ m_0 + y^T.C.\sigma = y^T.m_0$$

  P-semiflows ➔ Conservation of tokens

Functional properties and analysis

- Properties (cont.):
  2. If m is a reachable marking in N, σ a fireable sequence with $\sigma = x$, *and* x a *T*-semiflow, the next property follows (or *T*-invariant):

     $$m\,[\sigma> m$$

     Proof: if is a *T*-semiflow, $m = m_0 + C.x = m_0$

  T-semiflows ➔ Repetitivity of the marking

- *P* and *T*-semiflows can be computed using algorithms based in Convex Geometry (linear algebra and linear programming)

Functional properties and analysis

- Definitions:
  - N is conservative $\Leftrightarrow \exists y > 0,\ y^T.C = 0$
  - N is structurally bounded $\Leftrightarrow \exists\ y \geq 1, y^T.C \leq 0$ (computable in polynomial time)

- Properties: pre-multiplying by y the state equation
  - N conservative $\Rightarrow y^T.\ m = y^T.\ m_0$ (token conservation)
  - N structurally bounded $\Rightarrow y^T.\ m \leq y^T.\ m_0$ (tokens limitation)

Functional properties and analysis

- Definitions:
  - $\mathcal{N}$ is consistent $\Leftrightarrow \exists\, x > 0,\ C.x = 0$
  - $\mathcal{N}$ is structurally repetitive $\Leftrightarrow \exists\, x \geq 1,\ C.x \geq 0$
- Properties:
  - $\langle \mathcal{N}, m_0 \rangle$ repetitive $\Rightarrow$ $\mathcal{N}$ structurally repetitive
  - $\mathcal{N}$ structurally live $\Rightarrow$ $\mathcal{N}$ structurally repetitive
  - $\mathcal{N}$ structurally live and structurally bounded $\Rightarrow$ structurally repetitive and structurally bounded $\Leftrightarrow$ consistent and conservative

Reading material

- Untimed Petri nets, by E. Teruel, G. Franceschinis, M. Silva. In *Performance Models for Discrete Event Systems with Synchronizations: Formalisms and Analysis Techniques*, G. Balbo & M. Silva (ed.), Chapter 2, pp. 27-75, Zaragoza, Spain, Editorial KRONOS, September 1998.
- Logical properties of P/T systems and their analysis, by J.M. Colom, E. Teruel, M. Silva. In *Performance Models for Discrete Event Systems with Synchronizations: Formalisms and Analysis Techniques*, G. Balbo & M. Silva (ed.), Chapter 6, pp. 185-232, Zaragoza, Spain, Editorial KRONOS, September 1998.
- Linear algebraic and linear programming techniques for the analysis of net systems, by M. Silva, E. Teruel, J.M. Colom. *Lecture Notes in Computer Science, Lectures in Petri Nets. I: Basic Models*, G. Rozenberg and W. Reisig (ed.), vol. 1491, pp. 309-373, Berlin, Springer-Verlag, 1998.