Modelling and analysis of concurrent systems with Petri nets. Performance evaluation

Javier Campos
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, Spain
jcampos@unizar.es

LSI-UPC

Barcelona, June 2007

Course details

- 15 lectures of 50 minutes
- Topics:
  - Formal models of concurrent systems, Petri nets
  - Qualitative and quantitative (performance) analysis
  - Software performance engineering
- Slides available at:
  - http://webdiis.unizar.es/~jcampos/barcelona07.pdf
- Bibliography:
  - At the end of each lecture
- Orientation:
  - Post-graduate (master/PhD)

Contents

- Introduction to discrete event systems
- Petri nets: definitions, modelling and examples
- Functional properties and analysis techniques
- Time augmented Petri nets
- Performance evaluation with PNs: classic technique
- Structure based performance analysis techniques
  - Bounds
  - Approximations
  - Kronecker algebra-based exact solution
- Software performance engineering with UML and PNs

Modelling and analysis of concurrent systems with Petri nets. Performance evaluation

1. Introduction to discrete event systems

Javier Campos
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, Spain
jcampos@unizar.es

Outline

- Basic concepts
- Formal models

Basic concepts

- ❑ Discrete Event Systems (DES):
  - ❑ Systems whose state variables are seen/considered discrete (they take values in **N** or in a fixed alphabet)
    - ❑ The state space is discrete
    - ❑ Changes of state are due to events
  - ❑ Time is a singular variable
    - ❑ Synchronous systems: a clock –accesible from all nodes of the system– exists ➔ strong synchronization of clocks ➔ total order of events
    - ❑ Asynchronous systems: there is no global time ➔ events are ordered by causal relations ➔ partial order of events

…

Basic concepts

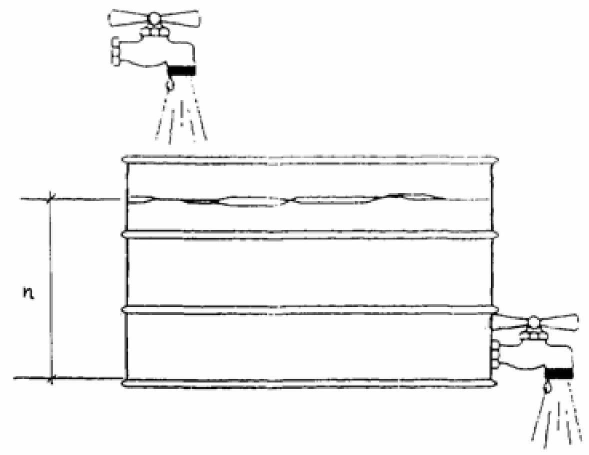
- Discrete Event Systems (cont.):

  …

  - DES appear in several application domains
    - Integrated manufacturing, Protocol engineering, Logistics, Computer architecture, Software engineering…
  - There exist simulation languages for DES with constructors valid to represent:
    - Jobs/activities, resources, duration of activities, logic validation…

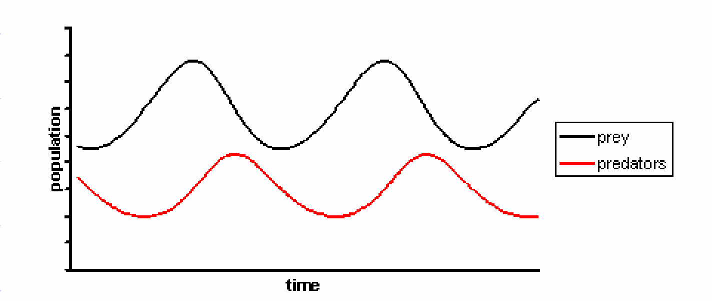
Basic concepts

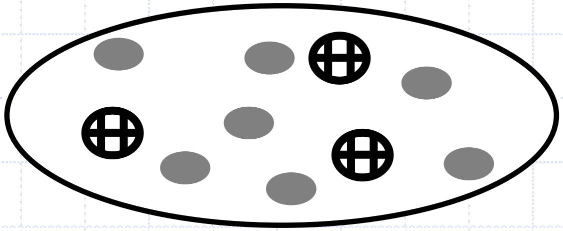
- ❑ Discrete? / continuous?



1) Discrete, $n \in \{0, 1, 2\}$
2) Continuous, $n \in (0, n_{max})$
3) Discrete: molecules
4) ¿...?

Predator/prey problem
Volterra-Lotka equation





Basic concepts

- Models
  - Abstraction of reality
    - Physical model
    - Simulation program
    - Textual/graphic description
    - Formal model
- DES: many complex/paradoxical situations

  ⇒ Interest of formal models

Outline

- □ Basic concepts
- □ Formal models

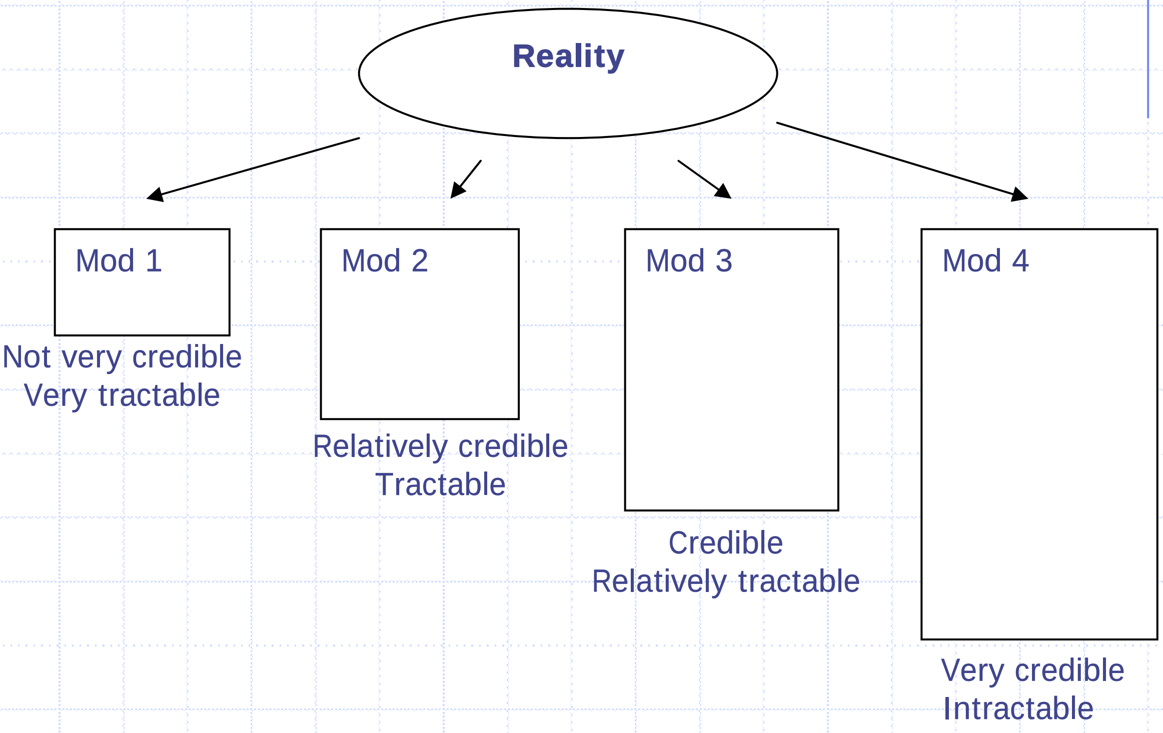
Formal models

- Advantages using formal models
  - Better comprehension (avoid ambiguities and contradictions; identify properties; suggest potential solutions…)
  - Increase the confidence level on the design
  - Help in the correct dimensioning
  - Help in the implementation and documentation
  - Increase re-usability

Need of formal methods is well-accepted in mature engineering domains (vs. emerging)

Formal models

- Formal models: credibility versus tractability



Reality

Mod 1

Not very credible
Very tractable

Mod 2

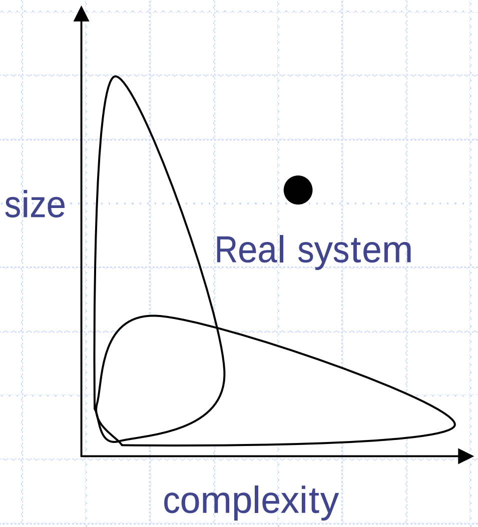
Relatively credible
Tractable

Mod 3

Credible
Relatively tractable

Mod 4

Very credible
Intractable



Formal models

- ❑ Maturity of a scientific/technical discipline
  - ❑ Formalisms
  - ❑ Models (paradigmatic)
  - ❑ Analysis/synthesis techniques
  - ❑ Tools (automated) to build/analyse/implement
  - ❑ Standardization: Norms (ISO, CCITT, ...)
- ❑ First DES problem:
  - ❑ No consensus on a "better formalism"

    (it does not exist a formalism so concise and tractable as differential equations for continuous systems)

Formal models

It does not exist a single formalism…

Life cycle: family of formalisms
(each one adapted for a given phase)

- *Paradigm*:
  - An entire constellation of beliefs, values and techniques, and so on, shared by the members of a given community.
  - A conceptual framework for reducing the chaotic mass to some form of order.
  - The total pattern of perceiving, conceptualizing, acting, validating, and valuing associated with a particular image of reality that prevails in a science or a branch of sience (T. Kuhn).
- *Modelling paradigm*:
  - Conceptual framework allowing to obtain formalisms from some common (few and basic) concepts and principles.
    - Conceptual and operative economy
    - Coherence

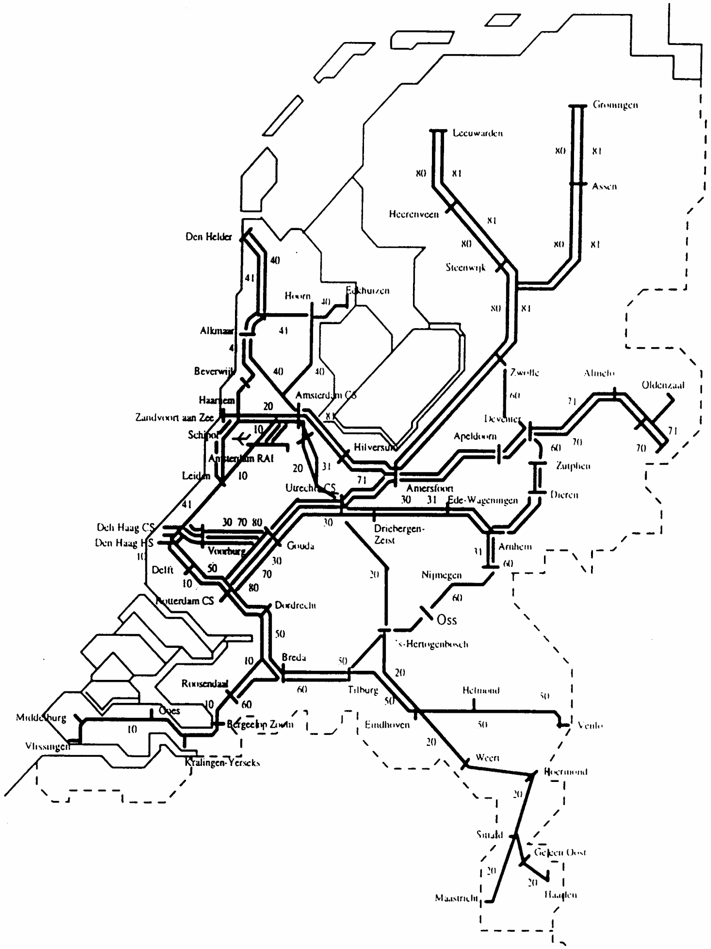
Formal models

- Formalisms for the modelling of DES
  - Sequential
    - Functional (untimed)
      - Regular expresions, grammars…
      - Automata, state diagrams, abstract state machines…

        + Probabilistic/possibilistic extensions…
    - Timed
      - Timed automata (deterministic, probabilistic, possibilistic…)
      - Markov chains…
  - Concurrent
    - Functional (untimed)
      - Product automata
      - Petri nets
      - Process algebras (CCS, CSP…)
    - Timed
      - Queueing networks
      - Conjunctive/disjunctive graphs (PERT, GERT…)
      - Max-plus algebras
      - Timed Petri nets (deterministic, stochastic, fuzzy…)
      - Timed process algebras (deterministic, stochastic, fuzzy…)

Formal models

- ❑ Examples of problems
  - ❑ Nederland intercity train network
    - ❑ Minimum periods
    - ❑ Used periods, flexibility
    - ❑ Efect of mutual waitings between trains (synchronizations)
    - ❑ Critical lines
    - ❑ Fleet and distribution to guarantee minimum period
    - ❑ Optimum lines structure
    - ❑ Dynamics after specific perturbations
    - ❑ Variability of service under stochastic hypothesis



Formal models

- ❑ Main basic modelling approaches (M. Bunge)
  - ❑ Descriptive / analytic (what is?)

    Internal representation:
    - ❑ System: objects + relations
    - ❑ States, events producing changes

      "Process" is not a primitive concept

      Automata, Petri nets, Markov chains, Queueing networks
  - ❑ Constructive / processes-based (how is observed?)

    External representation (I/O)
    - ❑ "Process" is a primitive concept
    - ❑ System: set of processes + synchronization constraints
    - ❑ Structured processes

      Regular expressions, Process algebras

Formal models

- Petri nets (vector addition systems)
    - Duality states and events
        - Place: state variable
        - Transition: state transformer
        - Marking: value of state
    - State equation (but…)
    - Dependency (sequentialization) and independency (parallelism) of events.
      Causal structure
    - True concurrency (versus interleaved sequential observations)
    - Temporal realism (performance, scheduling)
    - Locality (states and actions) → design methodologies (top-down, bottom-up)

Modelling and analysis of concurrent systems with Petri nets. Performance evaluation

2. Petri nets: definitions, modelling and examples

Javier Campos
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, Spain
jcampos@unizar.es

Outline

- Basic concepts
- Definition
- State equation
- Modelling features and examples
- Bibliography

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 and telecommunications systems.

Basic concepts

- Standardisation:
  - 2004-12-02

    Achieved Published Standard status:

    ISO/IEC 15909-1:2004 Software and system engineering - High-level Petri nets - Part 1: Concepts, definitions and graphical notation. Available from ISO, SAI Global and others.
  - 2005-06-23

    New Working Draft of ISO/IEC 15909-2 Software and Systems Engineering - High-level Petri Nets Part 2: Transfer Format submitted for a combined ISO/IEC SC7 WD/CD registration and CD ballot. Comments welcomed - formal or otherwise. [Editor's Announcement | ISO/IEC 15909-2 WD (Version 0.9.0)]

Outline

- ❑ Basic concepts
- ❑ Definition
- ❑ State equation
- ❑ Modelling features and examples
- ❑ Bibliography

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
  - Algorithmic state machines
  - PERTs
  - QNs
  - …

Definition

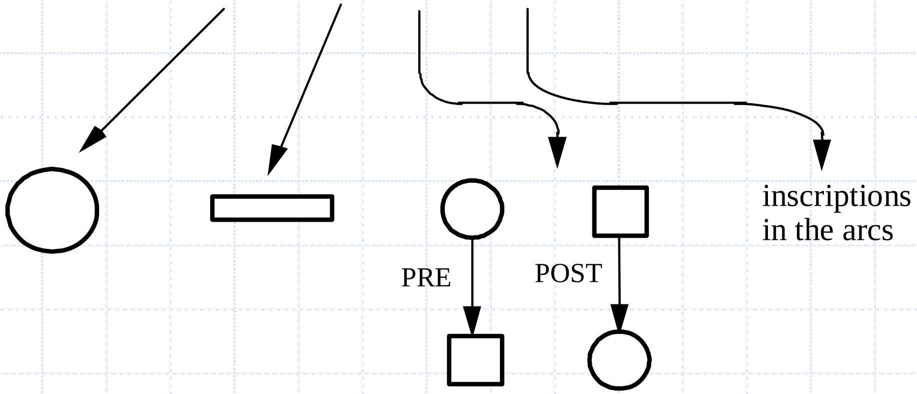
- In Petri Nets: two basic concepts (→ graphical objects)
  - states/data (PLACES)
  - actions/algorithms (TRANSITIONS)

  + weight (labeling) of the arcs

Definition

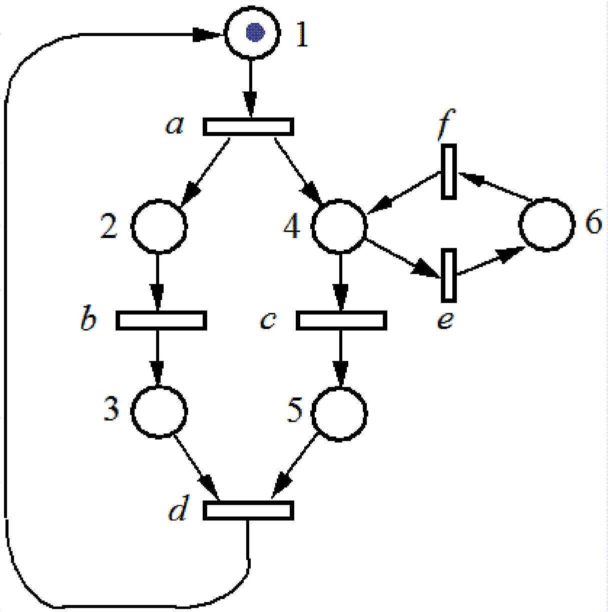
- ❑ Autonomous Petri nets (place/transition nets or P/T nets)
  - ❑ Petri Nets is a bipartite valued graph
    - ❑ Places: states/data (*P*)
    - ❑ Transitions: actions/algorithms (*T*)
    - ❑ Arcs: connecting places and transitions (*F*)
    - ❑ Weights: labeling the arcs (*W*)   ("ordinary nets" → weights = 1)

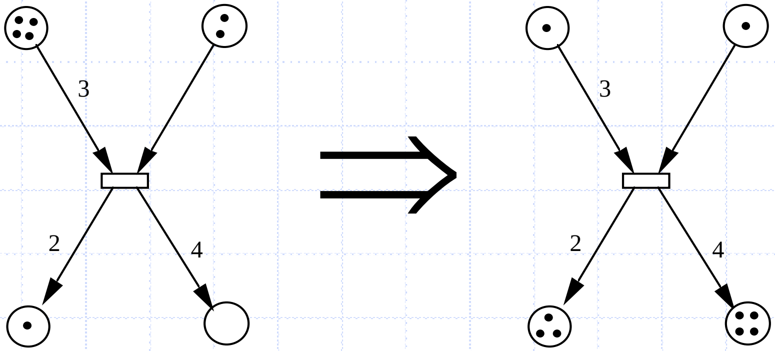
$$N = < P, \ T, \ F, \ W >$$



Definition

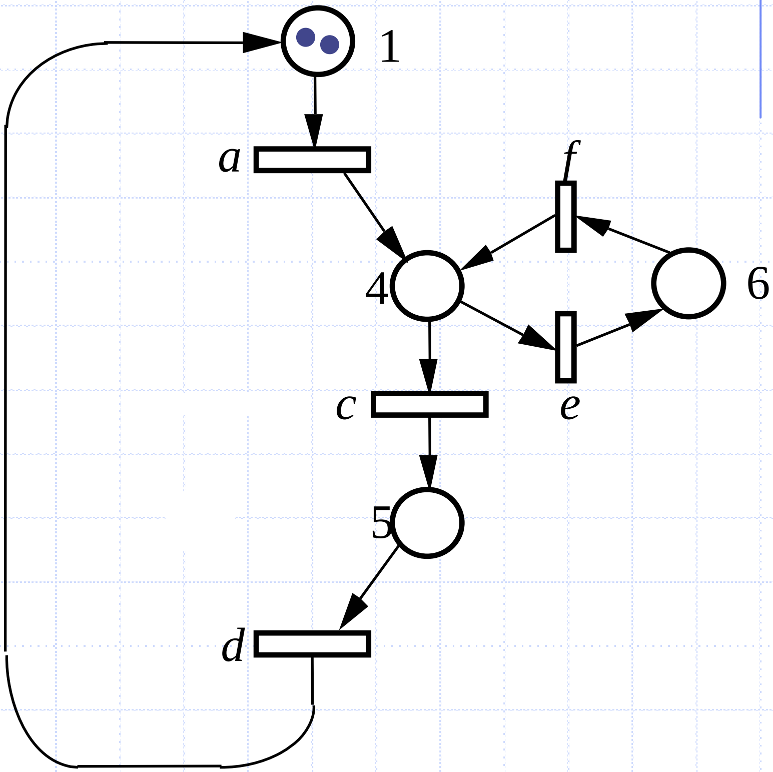
- ❑ Net ➔ Static part
  - ❑ Places : State variables (names)
  - ❑ Transitions: Changes in the state (conditions)
- ❑ Marking ➔ Dynamic part
  - ❑ Marking : State variables (values)
- ❑ Event/Firing
  - ❑ **Enabling**: the pre-condition is verified
  - ❑ **Firing**: change in the marking
    - ❑ the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





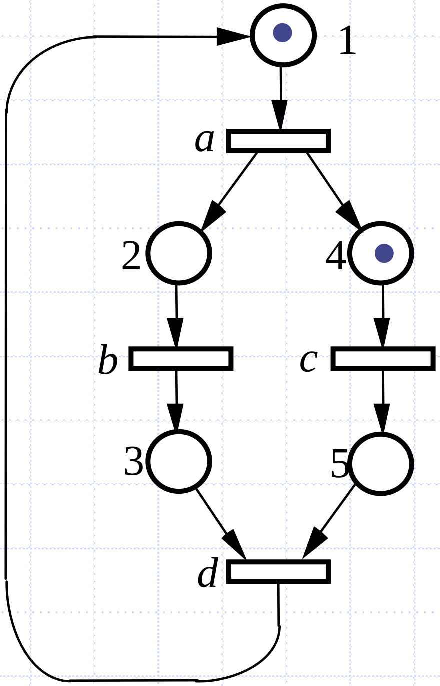
Definition

- ❑ 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



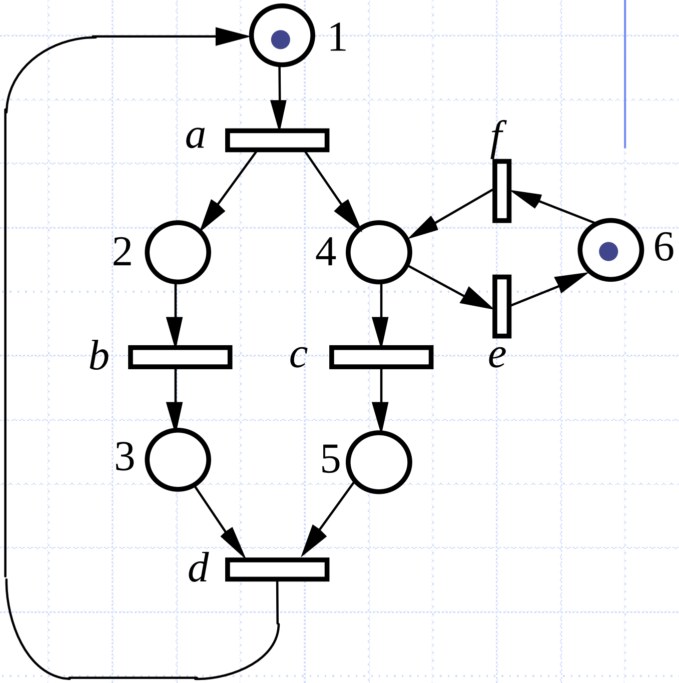
Definition

- 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



Definition

- 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

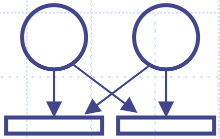


Every outgoing arc from a place is either unique or is a unique incoming arc to a transition.

Definition

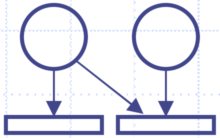
- PN syntactic subclasses (cont.)
  - **Extended free-choice**

    If two places have some common output transition, then they have all their output transitions in common.



  - **Simple** (or asymmetric choice)

    If two places have some common output transition, then one of them has all the output transitions of the other (and possibly more).



And other… (modular subclasses)

Outline

- ❑ Basic concepts
- ❑ Definition
- ❑ State equation
- ❑ Modelling features and examples
- ❑ Bibliography

State equation

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

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

  - ❑ Pre-incidence matrix

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

(→{0,1} for ordinary nets)

  - ❑ Post-incidence matrix

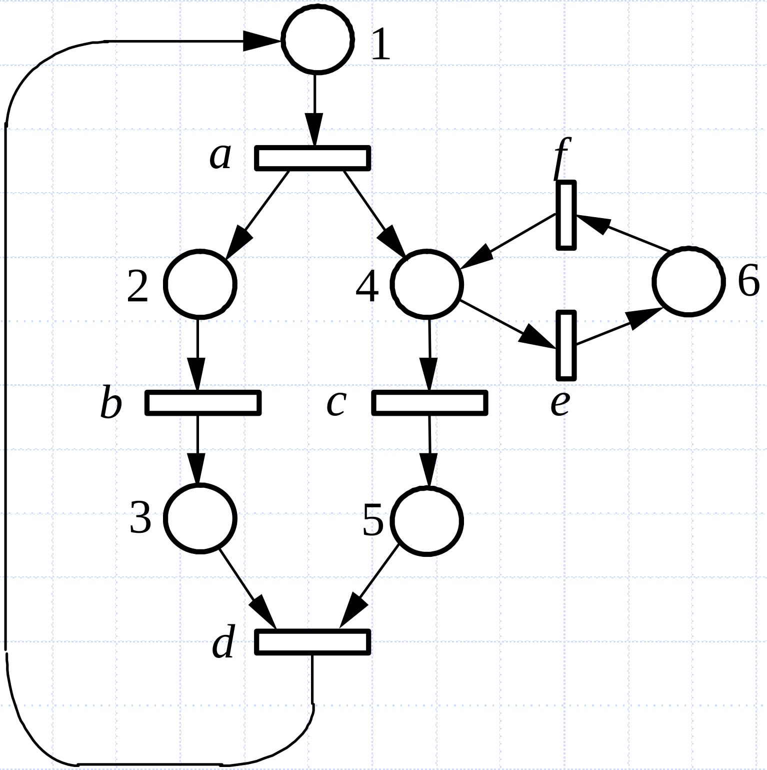
$$\mathbf{Post}(p,t)\colon PxT \rightarrow \mathrm{N}^{+}$$

(→{0,1} for ordinary nets)

  - ❑ Incidence matrix, *C* = Post – Pre

    (marked) Petri Net is finally defined by: $\Sigma = \langle \mathrm{N}, m_0 \rangle$

State equation



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

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

Incidence matrix *C* (= Post – Pre) cannot "see" self loops

State equation

- State equation definition

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

Integrating in one execution (sequence of firing)

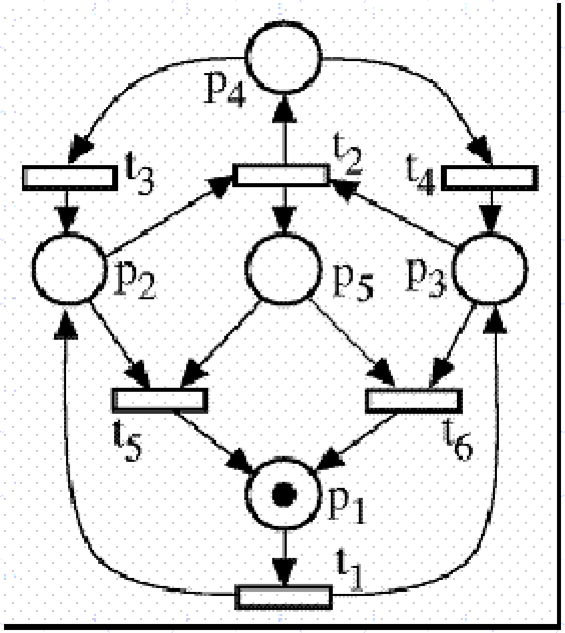
$$m_0\,[\sigma > m(k) \Rightarrow \boxed{m(k) = m_0 + \mathbf{C}\cdot\boldsymbol{\sigma}}$$

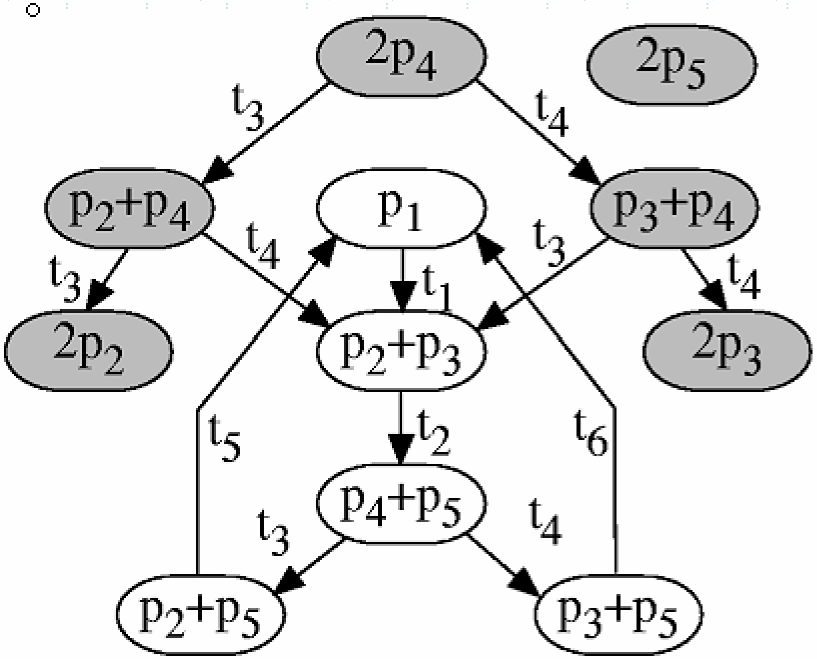
where σ (bold) is the firing counting vector of σ

State equation

Very important: unfortunately…

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





State equation

- Example (of problems): place marking bound

$$\begin{array}{lll} \max & m[p] \\ \text{s.t.} & m \in R(\mathrm{N}, m_0) \end{array} \quad \leq \quad \begin{array}{ll} \max & m[p] \\ \text{s.t.} & m = m_0 + \mathbf{C} \cdot \boldsymbol{\sigma} \\ & (m, \boldsymbol{\sigma}) \in \mathrm{N}^{n+m} \end{array}$$

Problem: spureous solutions ⇒ semidecision

Outline

- ❑ Basic concepts
- ❑ Definition
- ❑ State equation
- ❑ Modelling features and examples
- ❑ Bibliography

Modelling features and examples

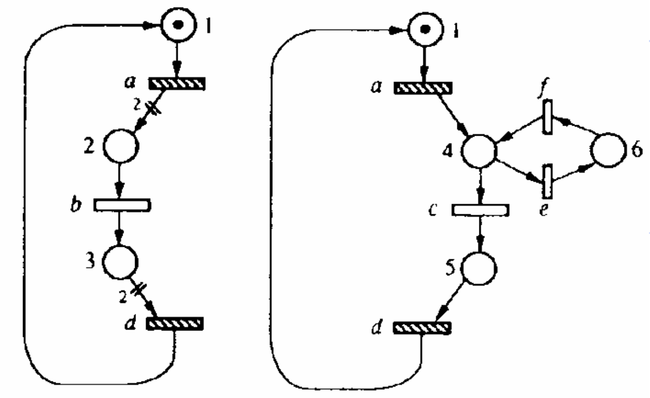
- Modelling expressivity
  - Sequences
  - Conflicts (decisions, iterations)
  - Concurrency and synchronizations

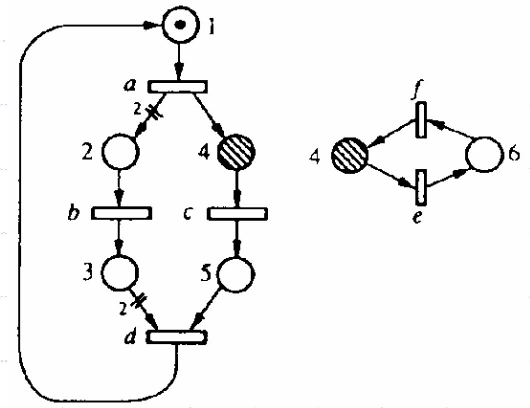
  - Duality places versus transitions

Modelling features and examples

- Design methodologies:
  1. Parallel composition by…

synchronization and fusion

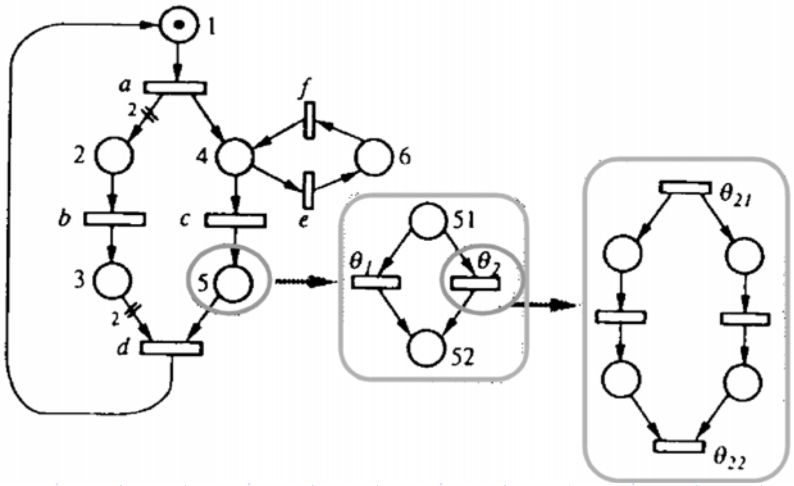




+ bottom-up methodology

Modelling features and examples

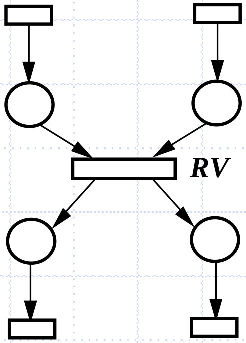
- Design methodologies (cont):
  2. Sequential composition by refinement



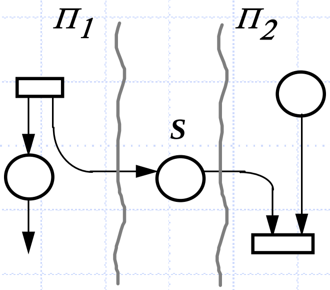
+ top-down methodology

Modelling features and examples

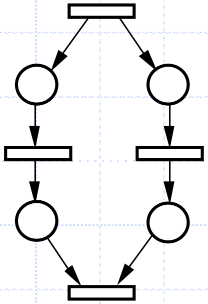
- Design methodologies (cont): typical synchronization schemes



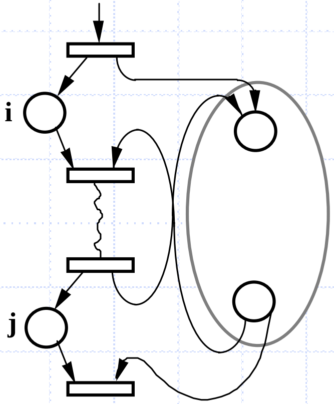
1. Rendezvous, RV



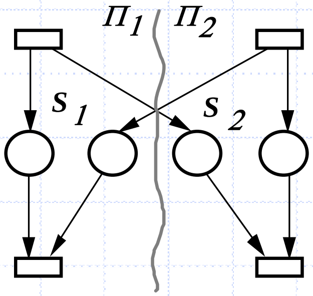
2. Semáforo, S



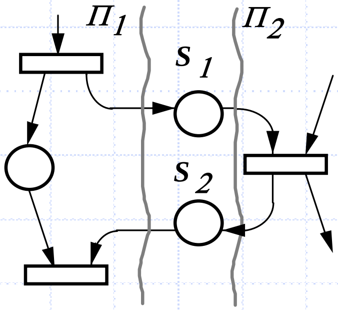
5. Fork-Joint



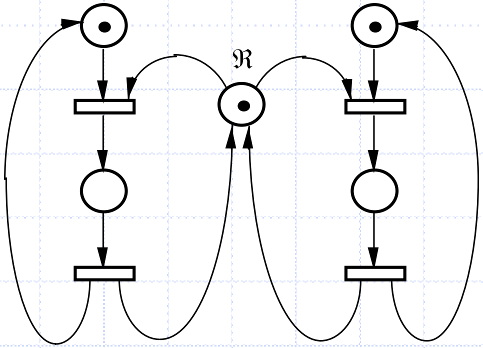
6. Sub programa
(p$_i$,p$_j$ están en mutex)



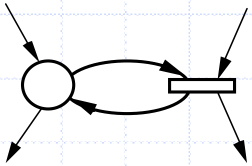
3. RV/Semáforo simétrico



4. RV/Semáforo asimétrico
(master/slave)



7. Recurso compartido (ℜ)

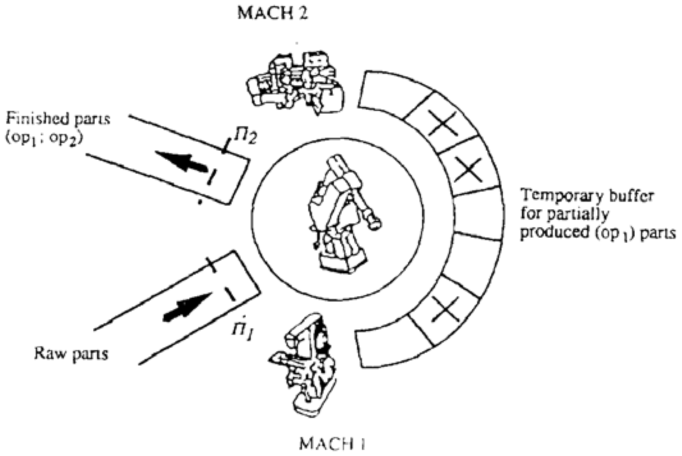


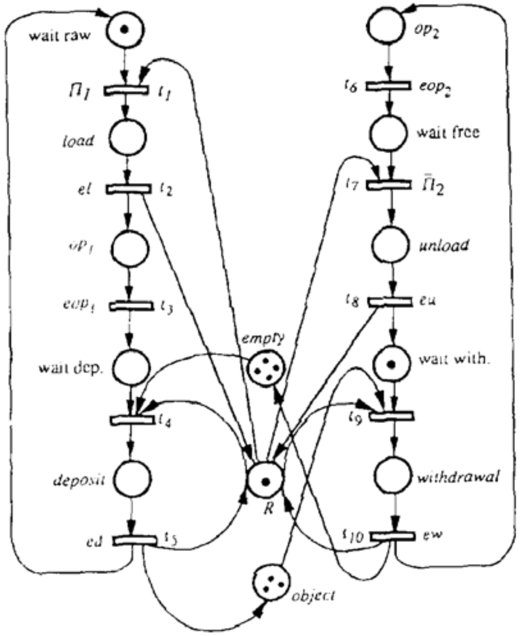
8. Guarda (condición de lectura)

Modelling features and examples

- Modelling example 1:
  Basic manufacturing cell

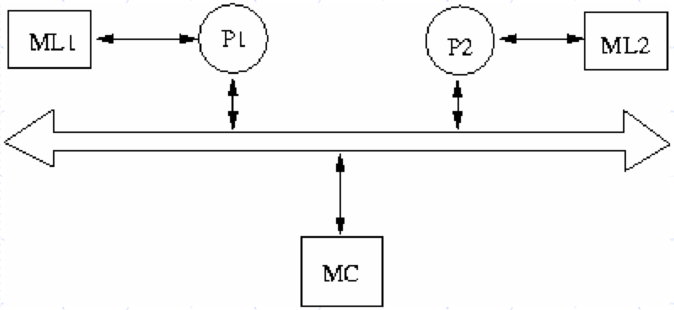
  producer/consumer
  with buffer
  and mutual exclusion

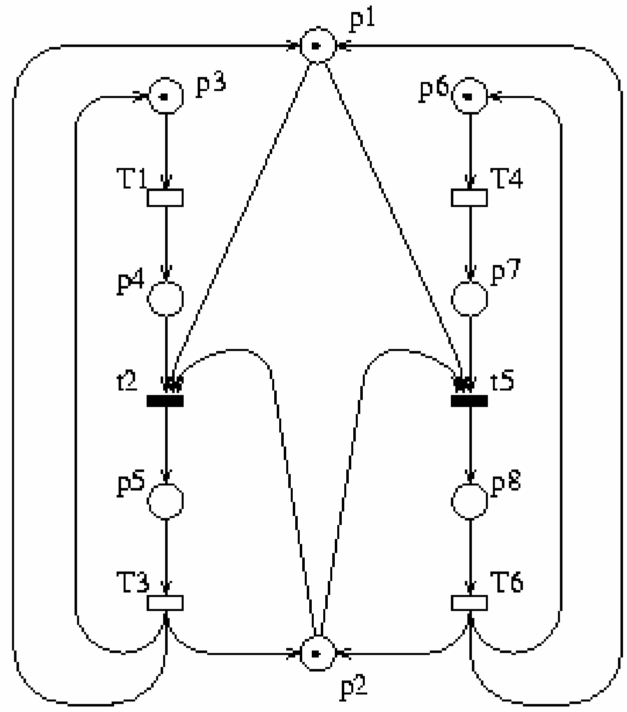

MACH 2
Finished parts (op1 ; op2)
Π2
Temporary buffer for partially produced (op 1) parts
Raw parts
Π1
MACH 1



Modelling features and examples

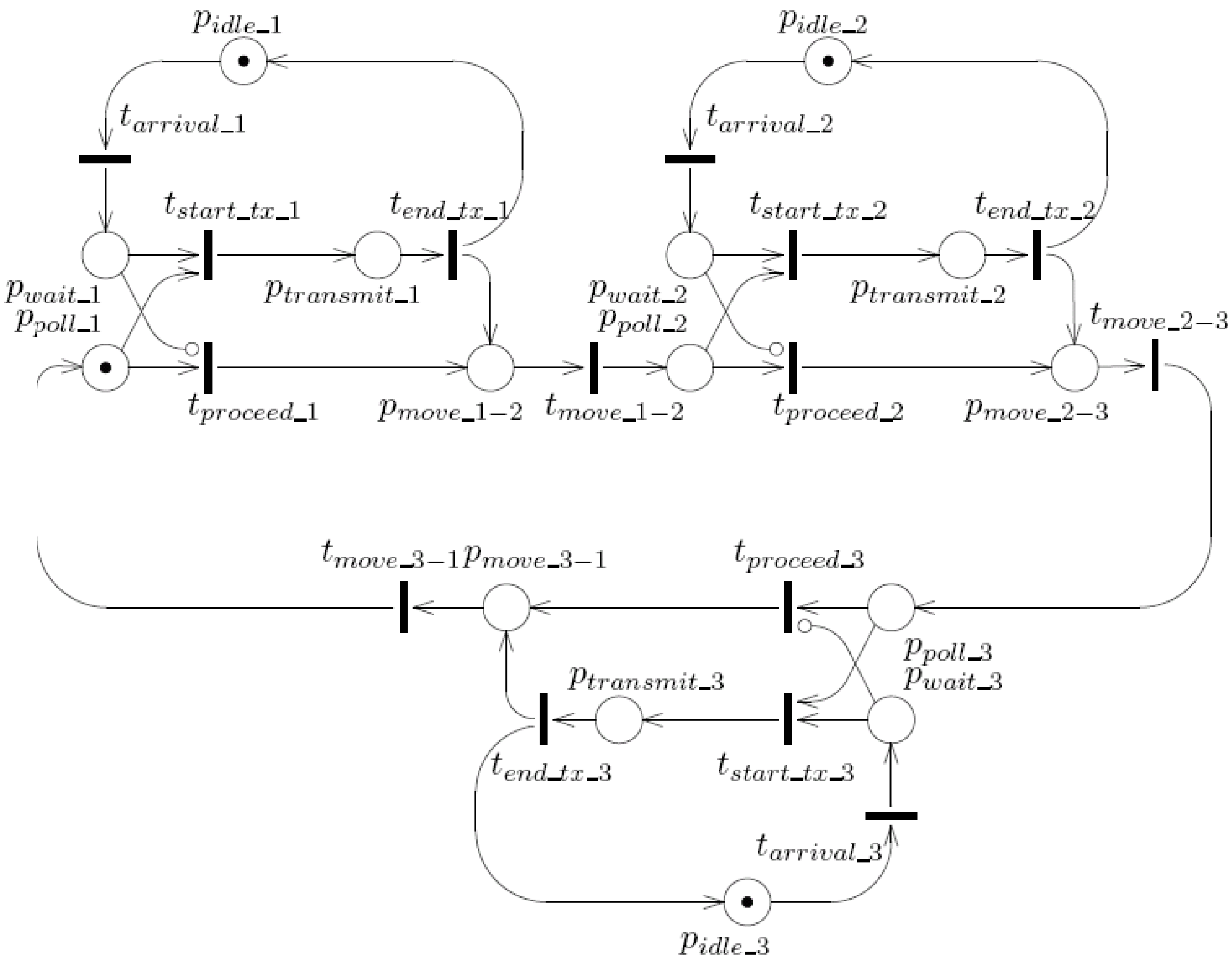
- Modelling example 2: Shared memory multiprocessor
  - two processors with similar behaviour
  - two local memories and one shared common memory





Modelling features and examples

- Modelling example 3: Token ring LAN



Outline

- ❑ Basic concepts
- ❑ Definition
- ❑ State equation
- ❑ Modelling features and examples
- ❑ Bibliography

Bibliography

- ❑ E. Teruel, G. Franceschinis, M. Silva: **Untimed Petri nets**. 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.
  Download here.
- ❑ Website: The Petri Nets World.
  http://www.informatik.uni-hamburg.de/TGI/PetriNets/
- ❑ Website: The Petri Nets Bibliography.
  http://www.informatik.uni-hamburg.de/TGI/pnbib/

Modelling and analysis of concurrent systems with Petri nets. Performance evaluation

3. Functional properties and analysis techniques

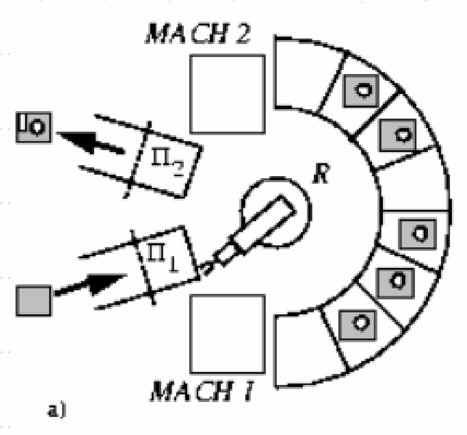
Javier Campos
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, Spain
jcampos@unizar.es

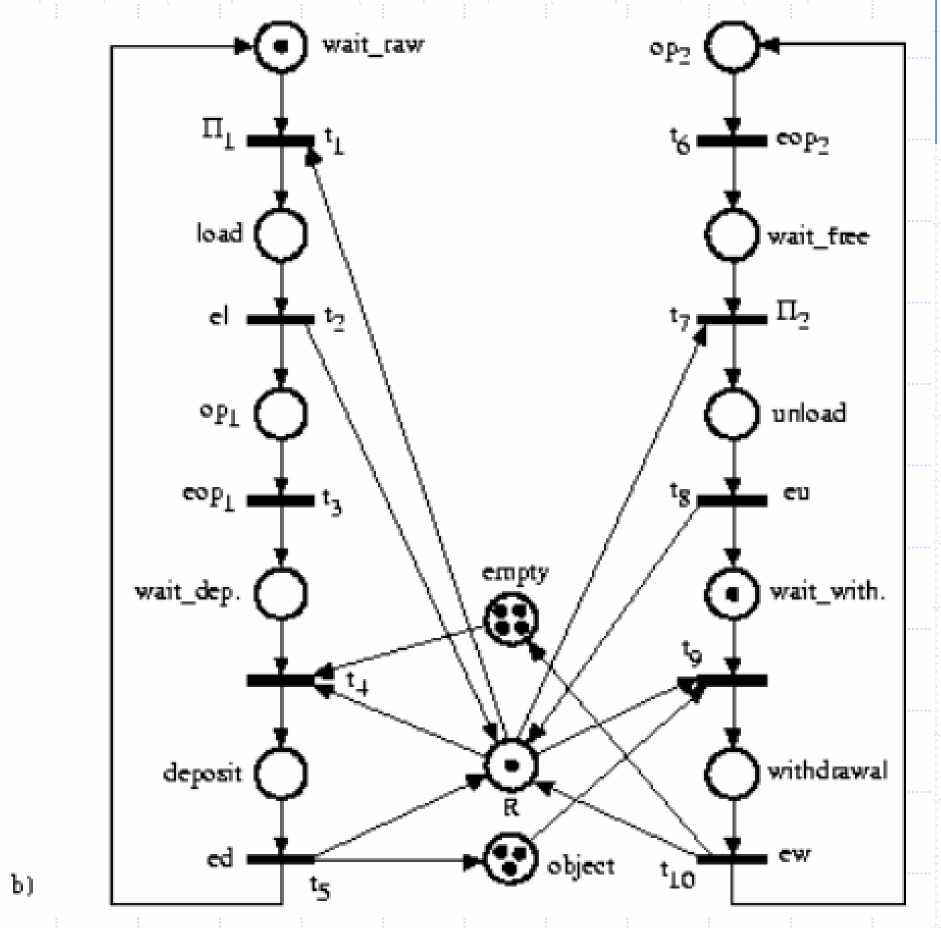
Outline

- Basic properties
- Analysis techniques
- Reachability graph
- Net transformations
- Convex geometry and PNs
- Bibliography

Basic properties

- ❑ Concurrent/parallel systems are difficult to understand
  - ❑ It is easy to make mistakes
  - ❑ Need for easy express properties and proof techniques





Basic properties

- Behavioural properties (for $\mathbf{m}_0$)
  - **Boundedness**: finiteness of the state space, i.e. the marking of all places is bounded

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

  - **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

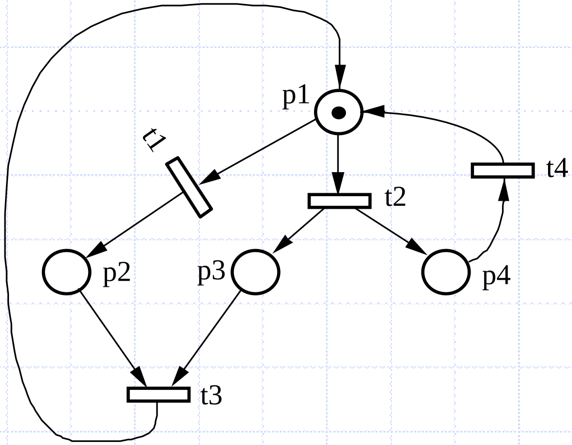
$$\forall t \in T, \forall \mathbf{m} \text{ reachable, } \exists \mathbf{m}', \mathbf{m}\, [\sigma > \mathbf{m}' \text{ such that } \quad \mathbf{m}'\, [t >$$

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

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

Basic properties

- ❑ Boundedness, deadlock, liveness…

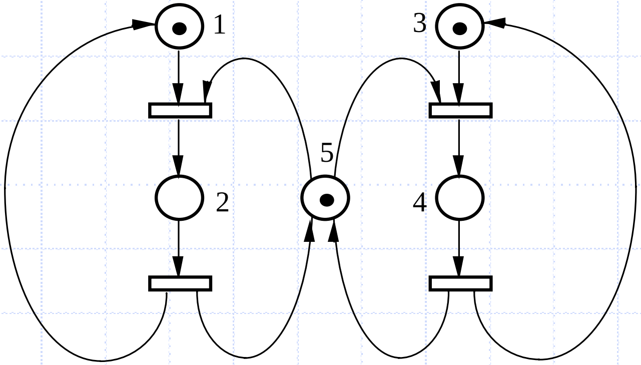


- ❑ Mutual exclusion

  m(p2) + **m**(p4) + **m**(p5) = 1

  ⇒ mutex (p2, p4, p5)

  (i.e., **m**(p2) . **m**(p4) = 0)



Basic properties

- ❑ Structural basic properties:

  ("there exists $\mathbf{m_0}$ …" or "for all $\mathbf{m_0}$ …")

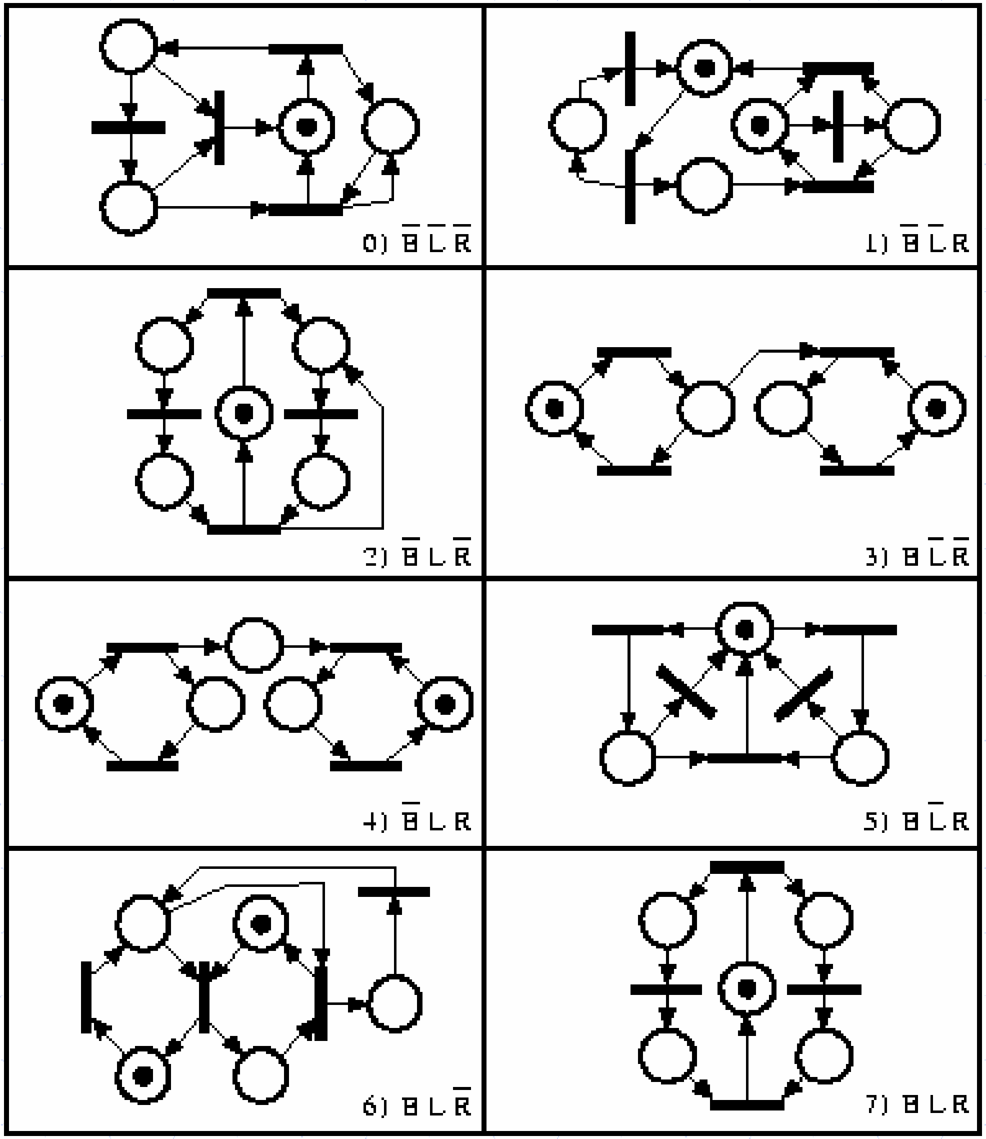
  They are abstractions of behavioural properties

  - ❑ N is **structurally bounded** if for all $\mathbf{m_0}$, $\langle \mathcal{N}, \mathbf{m_0} \rangle$ is bounded

  - ❑ N is **structurally live** if there exists a $\mathbf{m_0}$ for which $\langle \mathcal{N}, \mathbf{m_0} \rangle$ is live

Basic properties

- ❑ Independence of
  - ❑ Liveness
  - ❑ Boundedness
  - ❑ Reversibility


0) $\overline{B}\ \overline{L}\ \overline{R}$
1) $\overline{B}\ \overline{L}\ R$
2) $\overline{B}\ L\ \overline{R}$
3) $B\ \overline{L}\ \overline{R}$
4) $\overline{B}\ L\ R$
5) $B\ \overline{L}\ R$
6) $B\ L\ \overline{R}$
7) $B\ L\ R$

Outline

- ❑ Basic properties
- ❑ Analysis techniques
- ❑ Reachability graph
- ❑ Net transformations
- ❑ Convex geometry and PNs
- ❑ Bibliography

Analysis techniques

- Analysis techniques for the computation of functional properties
  - Enumerative
    - Exahustive exploration of the state space, thus based on reachability graph
    - Only valid for bounded systems
    - Conclusions are valid only for a given $\mathbf{m}_0$
    - For unbounded systems: coverability graph
      - Lost of part of information of state space thus we cannot conclude about some of the properties

Analysis techniques

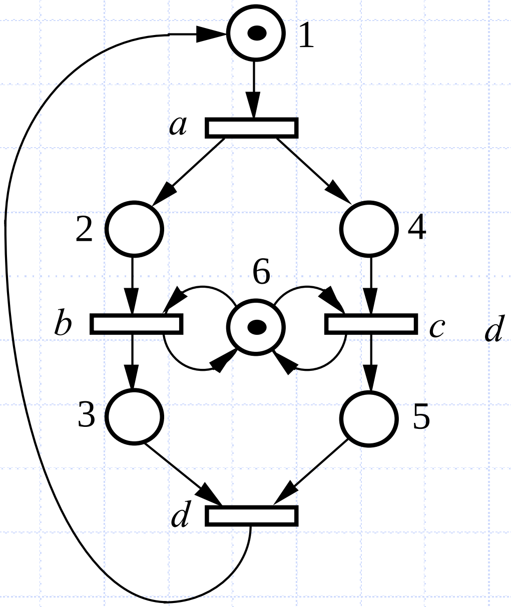
- Analysis techniques for the computation of functional properties (cont.)
  - Reduction/transformation of the model
    - $\langle \mathcal{N}^{i}, \mathbf{m}_0^{i} \rangle \rightarrow \langle \mathcal{N}^{i+1}, \mathbf{m}_0^{i+1} \rangle$
    - Rules that preserve the property under study and simplify the model for the analysis of such property
  - Structural
    - Based on the structure of the model, considering $\mathbf{m}_0$ as a parameter
    - Make use of relation between structure and behaviour using techniques coming from…
      - Convex geometry / linear programming (invariants)
      - Graph theory (siphons, traps, handles, bridges…)

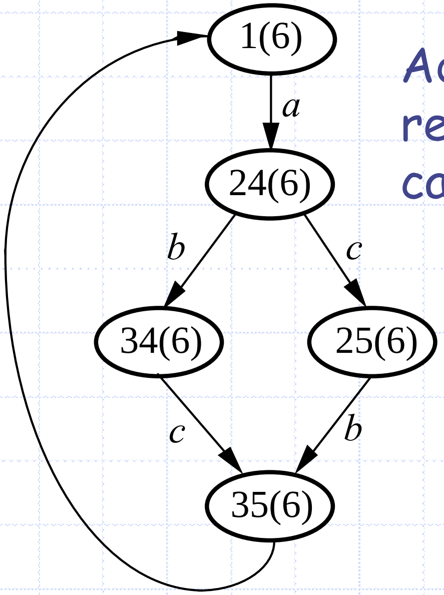
Outline

- ❑ Basic properties
- ❑ Analysis techniques
- ❑ Reachability graph
- ❑ Net transformations
- ❑ Convex geometry and PNs
- ❑ Bibliography

Reachability graph

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



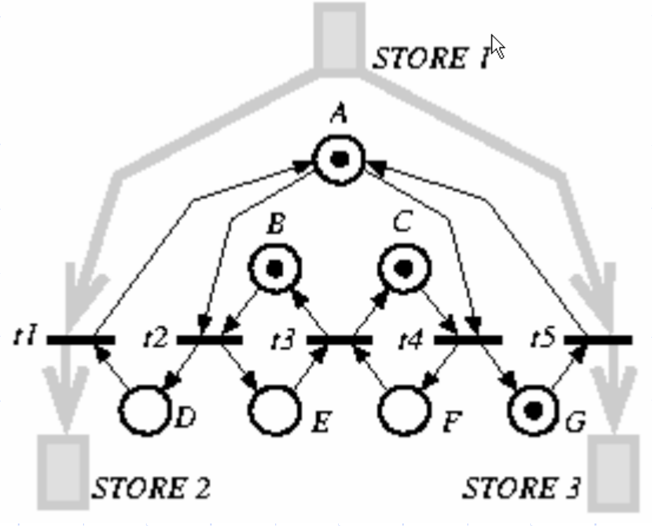


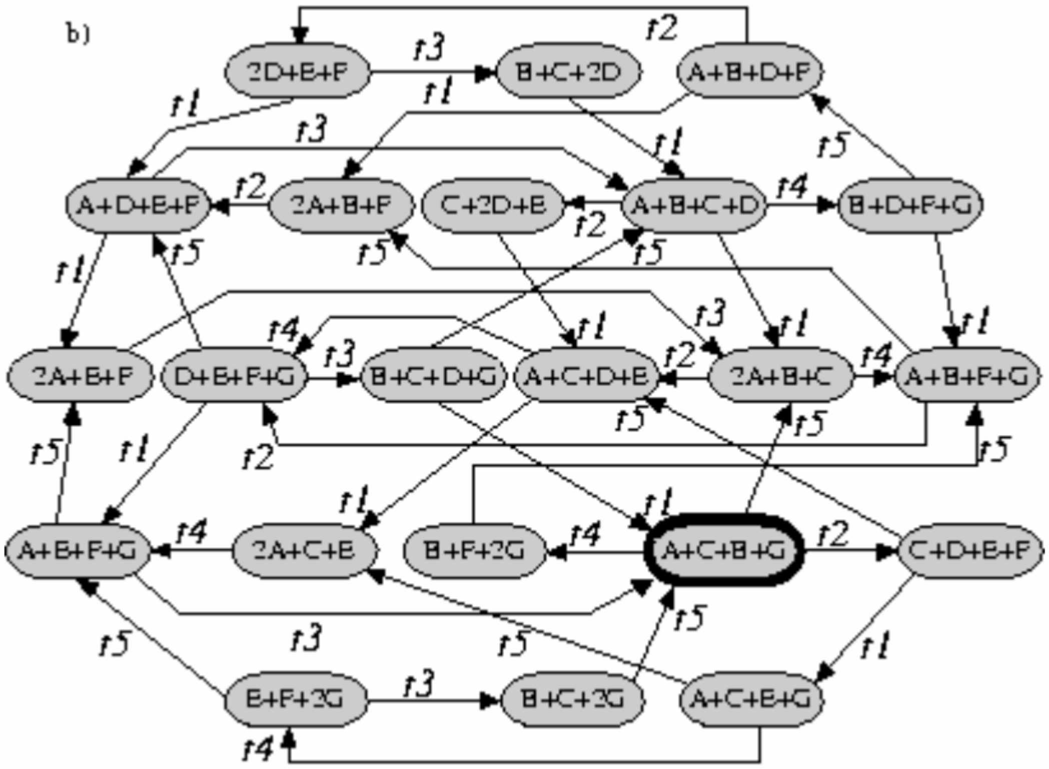
Adding place 6 does not modify reachability graph but *b* and *c* cannot fire simultaneously.

reachability graph

Reachability graph

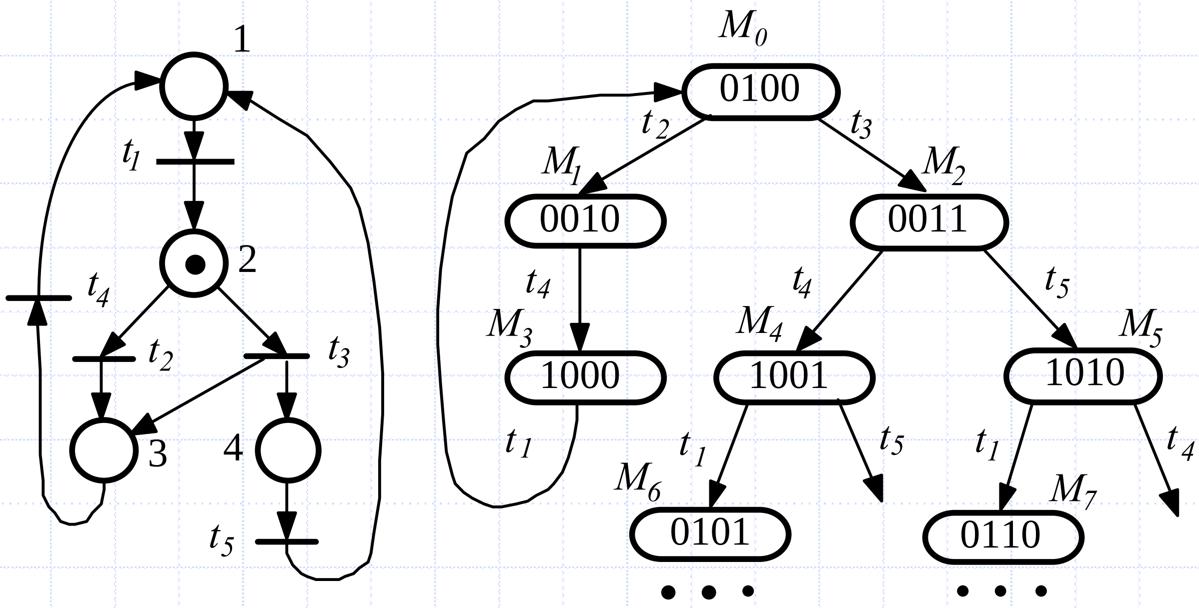
- ❑ Example of "easy" solution of a conflict with a regulation net





Reachability graph

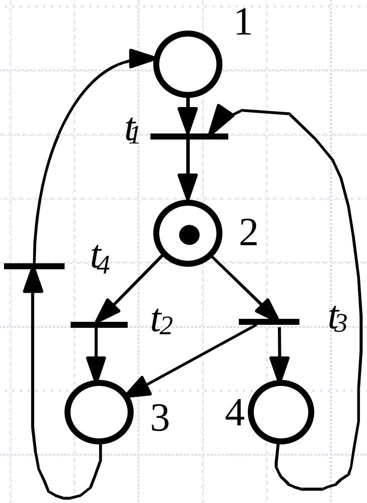
- ❑ Bounded system ⇔ finite reachability graph

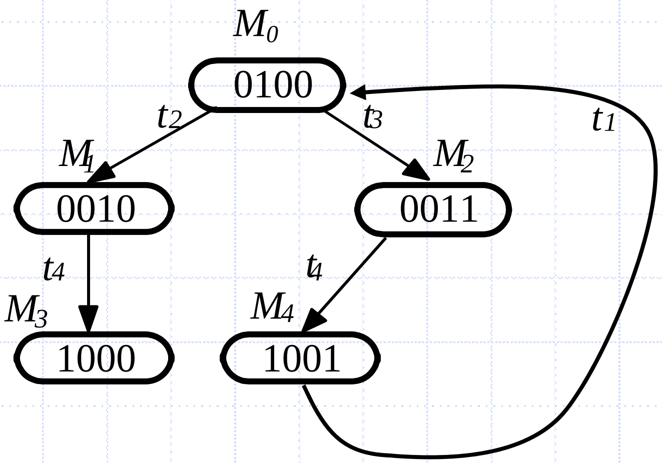


unbounded system

Reachability graph

- ❑ Deadlock exists ⇔ There exists a terminal node in the RG

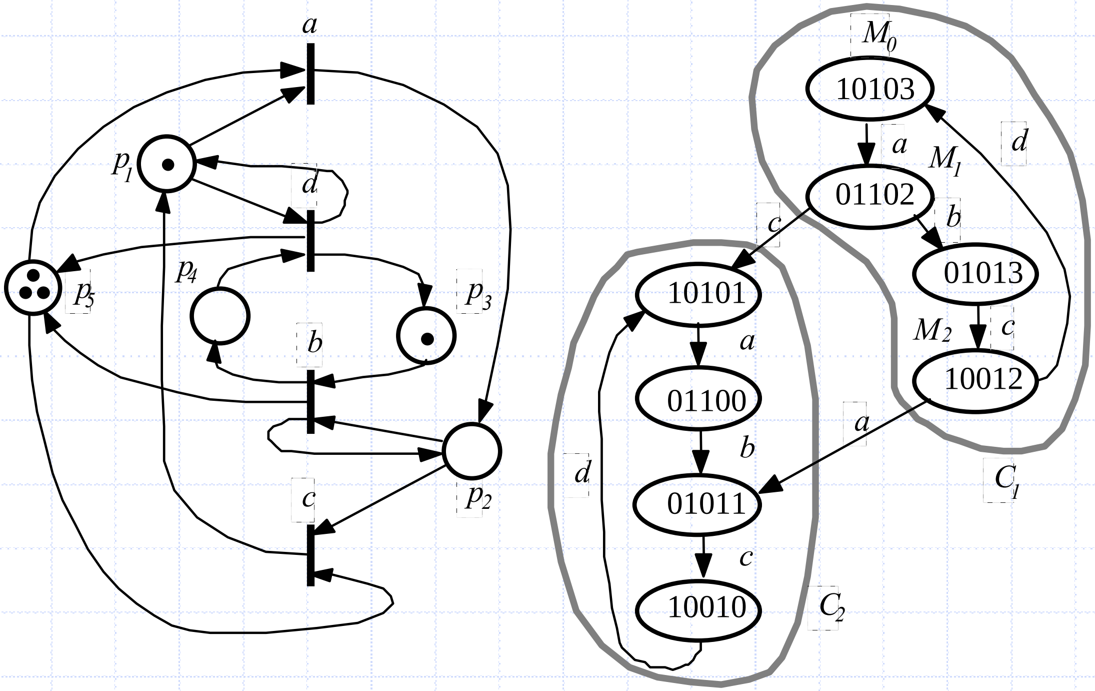




M_3 is a deadlock

Reachability graph

- 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

Outline

- ❑ Basic properties
- ❑ Analysis techniques
- ❑ Reachability graph
- ❑ Net transformations
- ❑ Convex geometry and PNs
- ❑ Bibliography

Net transformations

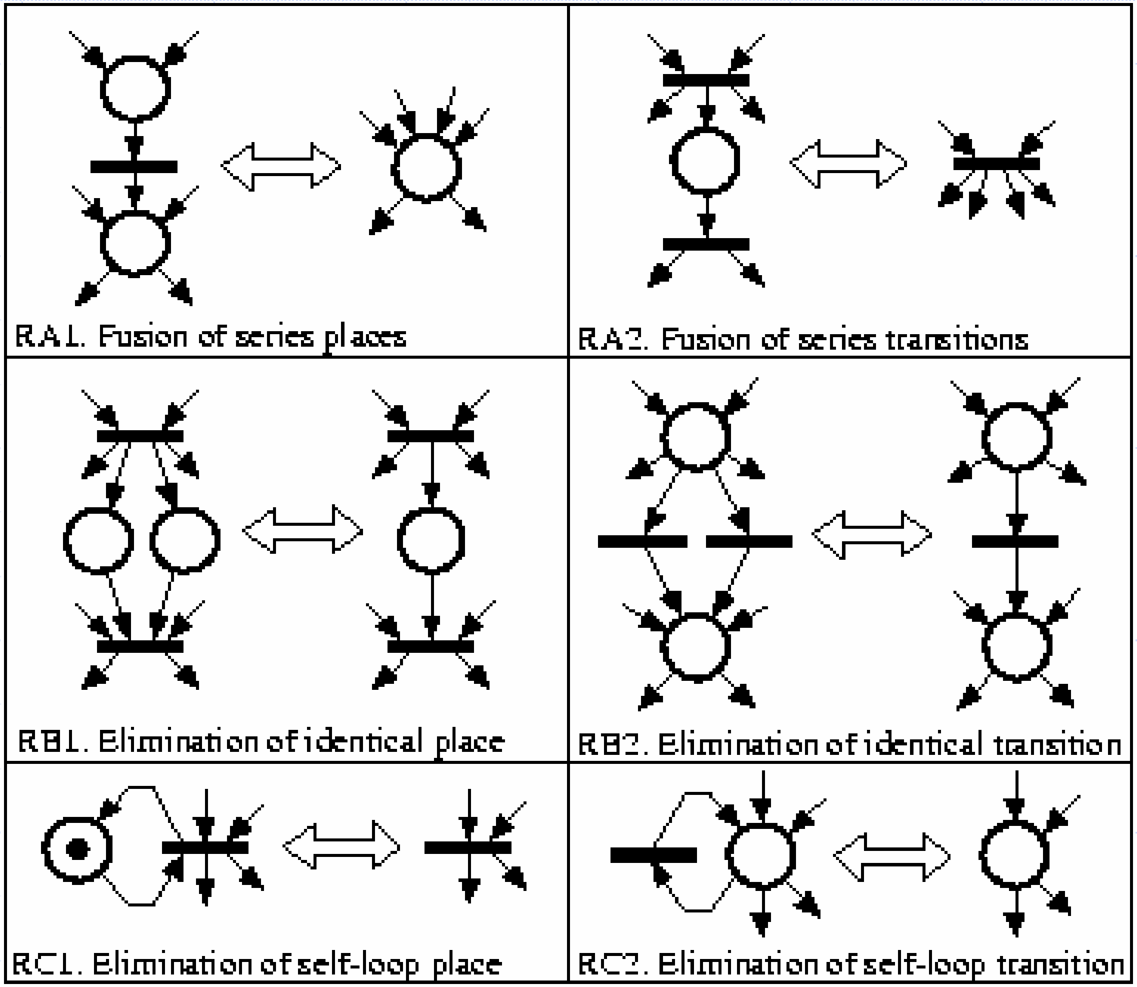
- Kit(s) of reduction rules
  - Rule:
    - Preconditions on the structure
    - Preconditions on the marking
    - Change of structure
    - Change of marking
  - Application of the rule:
    - If preconditions hold then apply changes
  - Problems:
    - For a given kit of rules, there exist irreducible systems
    - Trade-off:

      kit reduction power versus kit application complexity
  - Observation:

    for some net subclasses (for instance live and bounded free choice nets) there exist complete kits of reduction rules

Net transformations

- A basic kit of reduction rules



RA1. Fusion of series places

RA2. Fusion of series transitions

RB1. Elimination of identical place

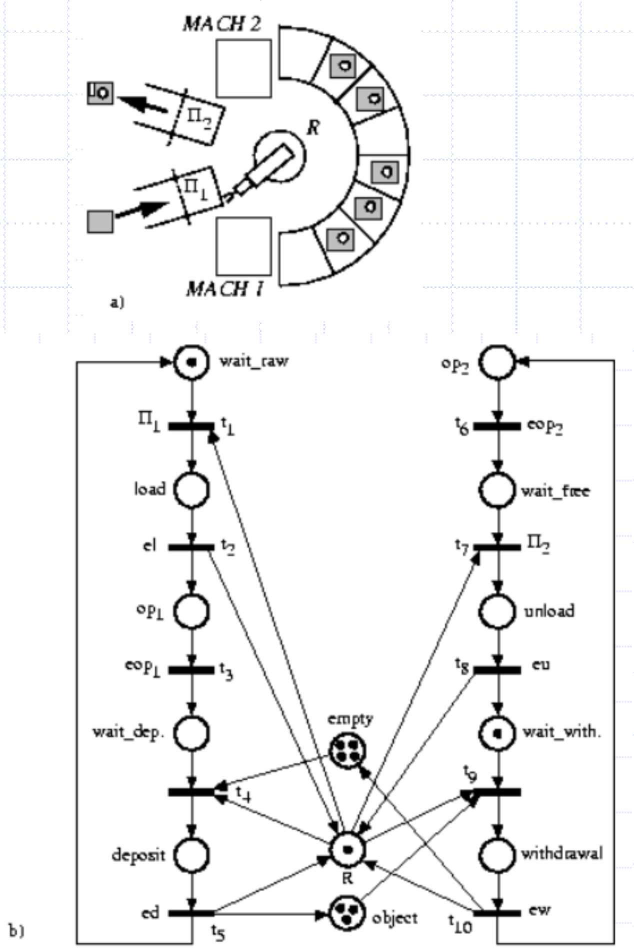
RB2. Elimination of identical transition

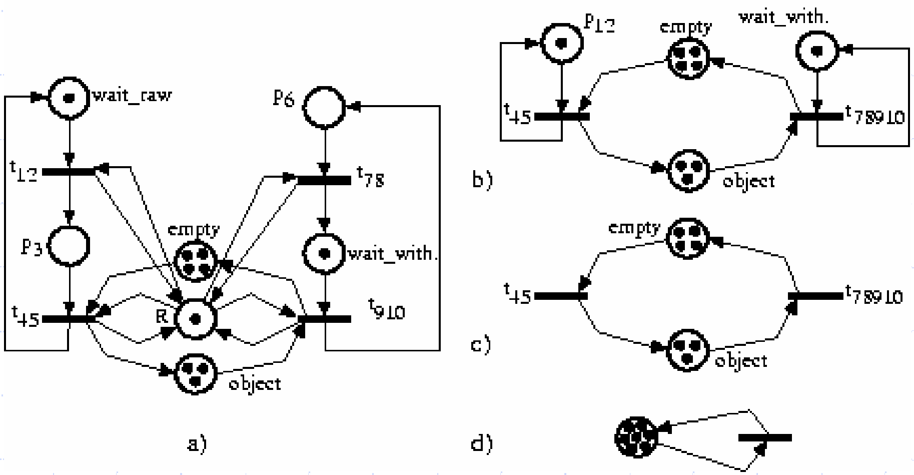
RC1. Elimination of self-loop place

RC2. Elimination of self-loop transition

Net transformations

- ❑ Example: a manufacturing cell



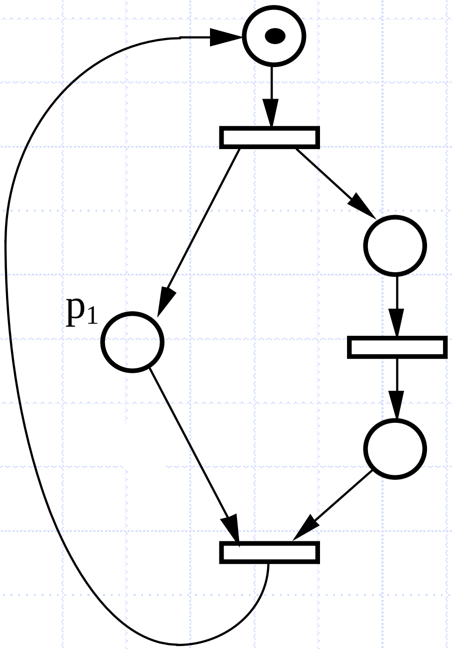


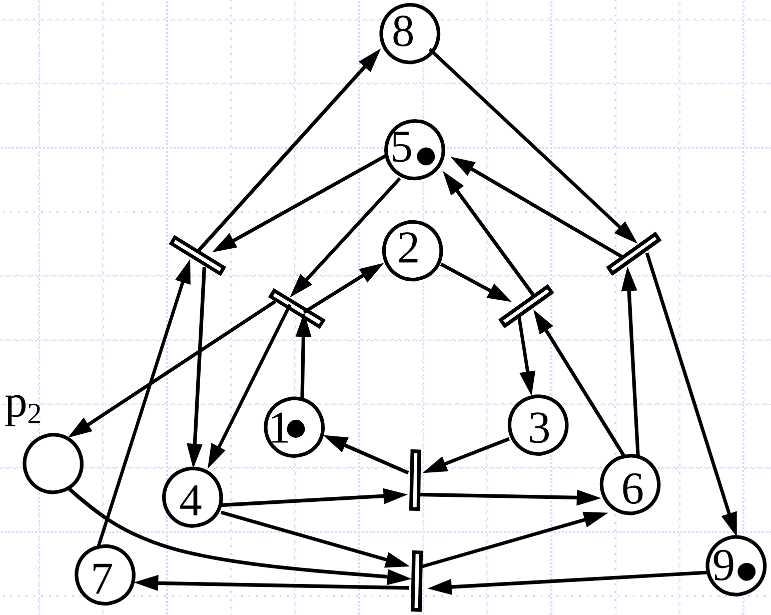
Net transformations

- Implicit places:
  - A place is implicit in $\langle \mathcal{N},\mathbf{m}_0\rangle$ if never is the unique constraint for the firing of its output transitions
  - Therefore: elimination of an implicit place does not change the set of firable sequences
  - Then: elimination of implicit places preserves liveness and synchronic properties (distance, fairness…)

Net transformations

- Implicit places (cont.):



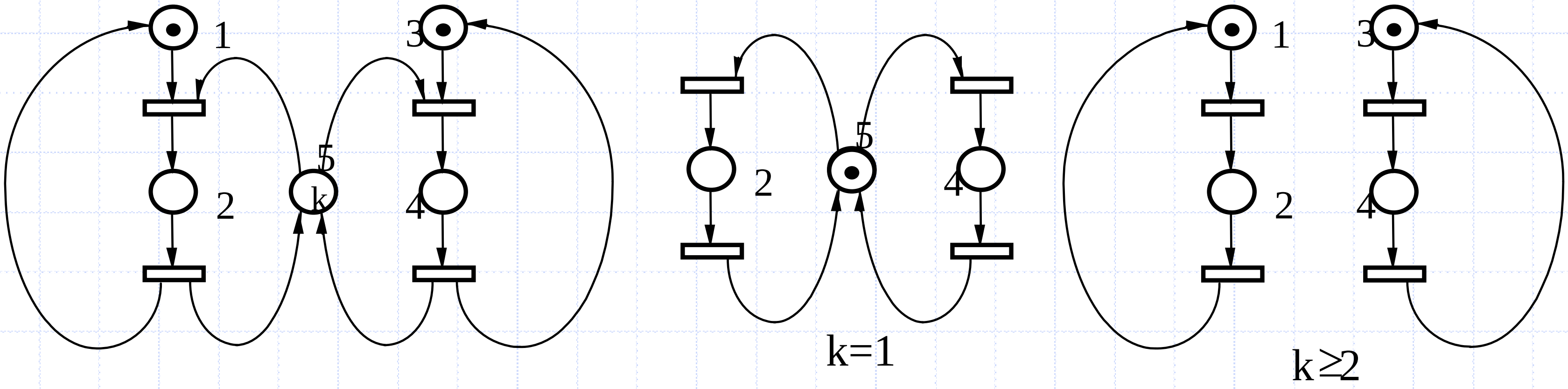


p_1 and p_2 are implicit for $\mathbf{m_0}$

p_2 is not structurally implicit

Net transformations

- Implicit places (cont.):
  - Place *p* is **structurally implicit** in *N* if
    for all initial marking of the other places,
    an initial marking of *p* can be defined such that *p* is implicit
  - An struct. implicit place may be implicit or not



Net transformations

- ❑ Implicit places (cont.):
  - ❑ Property: a place p is structurally implicit if and only if $\exists\ \mathbf{y} \geq 0, \mathbf{y}(p)=0$ such that $\mathbf{y}^T C \leq C(p)$.
  - ❑ Property: if p is structurally implicit and $\mathbf{m}_0(p) \geq \mathbf{y}^T\mathbf{m}_0$, then p is implicit.

Outline

- ❑ Basic properties
- ❑ Analysis techniques
- ❑ Reachability graph
- ❑ Net transformations
- ❑ Convex geometry and PNs
- ❑ Bibliography

Convex geometry and PNs

- ❑ 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: $\mathbf{y} \geq 0,\ \mathbf{y}^T.C = \mathbf{0}$

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

Convex geometry and PNs

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

$$\text{for all } \mathbf{m} \in RS(N, \mathbf{m}_0) \text{ and for all } \mathbf{m}_0 \Rightarrow$$
$$\Rightarrow \mathbf{y}^T.\ \mathbf{m} = \mathbf{y}^T.\ \mathbf{m}_0.$$

Proof: if $\mathbf{m} \in RS(N, \mathbf{m}_0)$ then $\mathbf{m} = \mathbf{m}_0 + C.\sigma$, and pre-multiplying by $\mathbf{y}^T$:

$$\mathbf{y}^T.\ \mathbf{m} = \mathbf{y}^T.\ \mathbf{m}_0 + \mathbf{y}^T.C.\sigma = \mathbf{y}^T.\mathbf{m}_0$$

P-semiflows ➔ Conservation of tokens

Convex geometry and PNs

- ❑ Properties (cont.):

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

  $$\mathbf{m}\,[\sigma > \mathbf{m}$$

  Proof: if $\mathbf{x}$ is a *T*-semiflow, $\mathbf{m} = \mathbf{m}_0 + C.\mathbf{x} = \mathbf{m}_0$

  T-semiflows ➔ Repetitivity of the marking

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

Convex geometry and PNs

- Definitions:
  - ❑ $\mathcal{N}$ is conservative $\Leftrightarrow \exists\, \mathbf{y} > 0,\ \mathbf{y}^T.\mathcal{C} = \mathbf{0}$
  - ❑ $\mathcal{N}$ is structurally bounded $\Leftrightarrow \exists\, \mathbf{y} \geq \mathbf{1},\ \mathbf{y}^T.\mathcal{C} \leq \mathbf{0}$

    (computable in polynomial time)

- Properties: pre-multiplying by **y** the state equation
  - ❑ $\mathcal{N}$ conservative $\Rightarrow \mathbf{y}^T.\ \mathbf{m} = \mathbf{y}^T.\ \mathbf{m}_0$

    (token conservation)
  - ❑ $\mathcal{N}$ structurally bounded $\Rightarrow \mathbf{y}^T.\ \mathbf{m} \leq \mathbf{y}^T.\ \mathbf{m}_0$

    (tokens limitation)

Convex geometry and PNs

- Definitions:
  - $\mathcal{N}$ is consistent $\Leftrightarrow \exists\, \mathbf{x} > 0,\ \mathcal{C}.\mathbf{x} = \mathbf{0}$
  - $\mathcal{N}$ is structurally repetitive $\Leftrightarrow \exists\, \mathbf{x} \geq \mathbf{1},\ \mathcal{C}.\mathbf{x} \geq \mathbf{0}$
- Properties:
  - $\langle \mathcal{N},\mathbf{m}_0 \rangle$ 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

Convex geometry and PNs

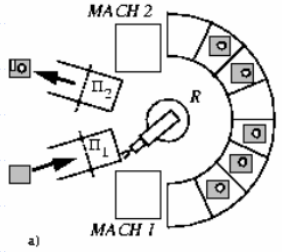
- ❑ Example: Producer/consumer with buffer in mutex

$m_{wait\_raw} + m_{load} + m_{op1} + m_{wait\_dep} + m_{deposit} = 1$ [1]

$m_{deposit} + m_{object} + m_{withdrawal} + m_{empty} = 7$ [2]

$m_{op2} + m_{wait\_free} + m_{unload} + m_{wait\_with} + m_{withdrawal} = 1$ [3]

$m_{R} + m_{load} + m_{deposit} + m_{unload} + m_{withdrawal} = 1$ [4]



For instance, from [1]:

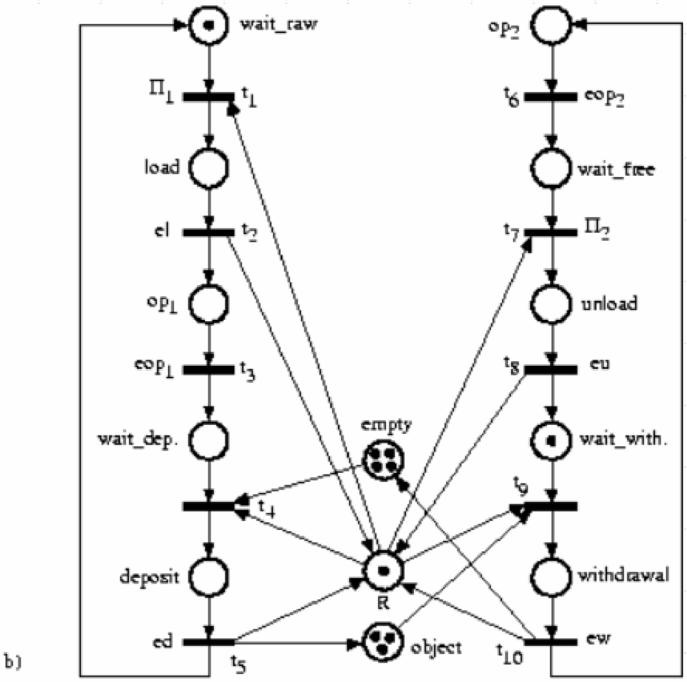
$m_{wait\_raw} \leq 1 \Leftrightarrow$ p_{wait_raw} is 1-bounded

$(m_{wait\_raw} = 0)$ OR $(m_{load} = 0)$

$\Rightarrow$ p_{wait_raw} and p_{load} are in MUTEX

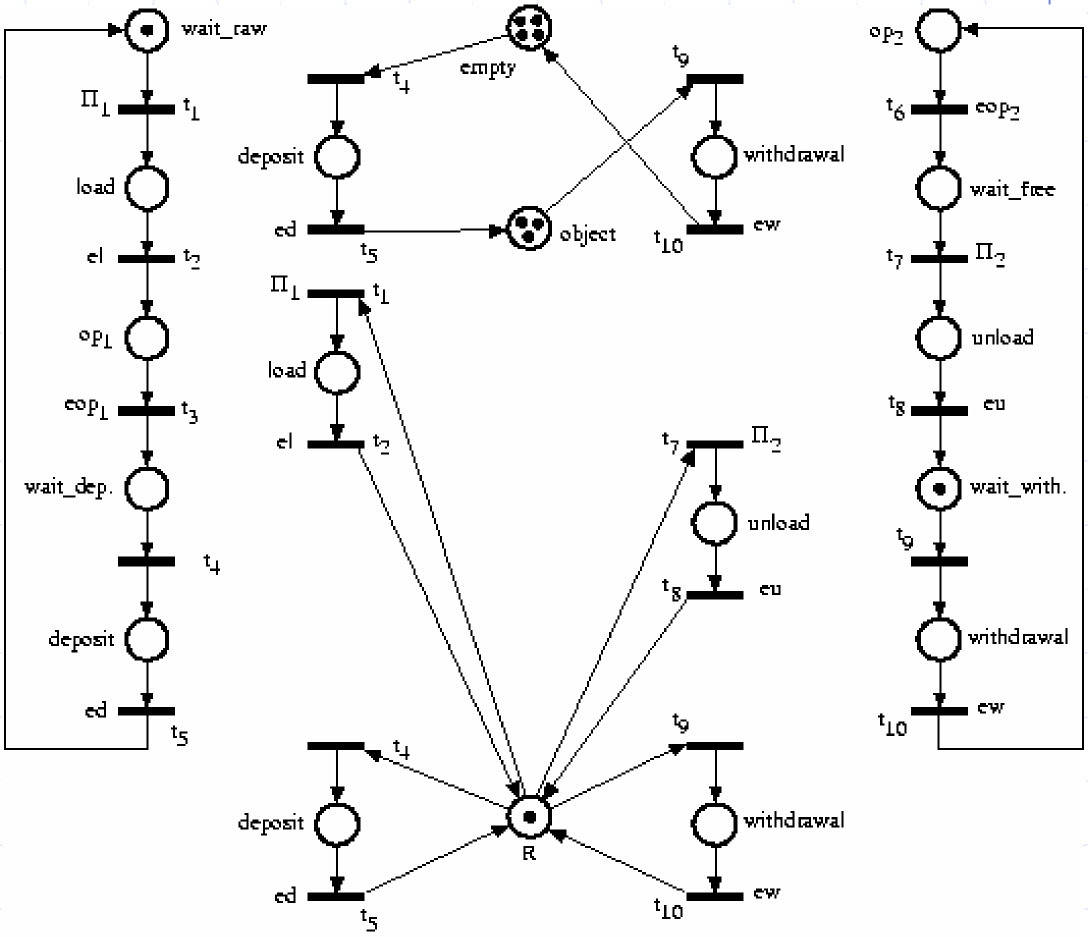
- ❑ Non-negative invariants $\Rightarrow$

$\Rightarrow$ provide a decomposed view of the original model



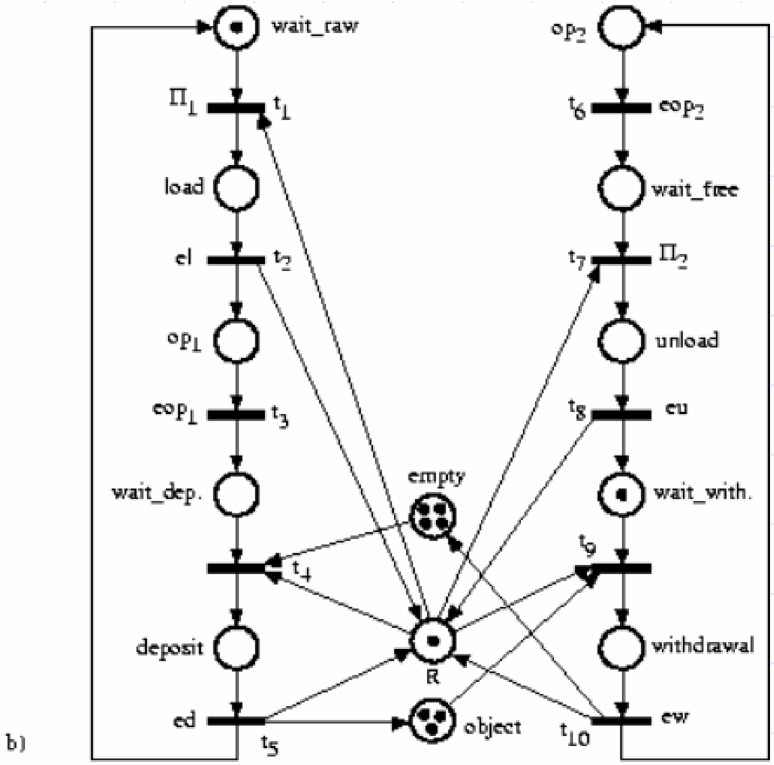
Convex geometry and PNs

- ❑ Applications of decomposed view of the model
  - ❑ Partial analysis
  - ❑ Implementation of the model



Convex geometry and PNs

- ❑ Absence of deadlock
  - ❑ if $m_{load} + m_{op1} + m_{deposit} + m_{op2} + m_{unload} + m_{withdrawal} \geq 1$
    then $(t_2 + t_3 + t_5 + t_6 + t_8 + t_{10})$ is firable
    else
      if $m_{wait\_raw} + m_{wait\_free} \geq 1$
        then $(t_1 + t_7)$ is firable
        else $(t_4 + t_9)$ is firable



Convex geometry and PNs

- ❑ Reversibility (with $\mathbf{m}_0$(empty)=7 and $\mathbf{m}_0$(object)=0):

  (Lyapunov-like proof technique

  potential function: $V(\mathbf{m}) = \mathbf{W}^T.\mathbf{m}$ with $\mathbf{W}(p)=0 \Leftrightarrow \mathbf{m}_0(p)>0$)

  - ❑ if $\mathbf{m}_{load} + \mathbf{m}_{op1} + \mathbf{m}_{deposit} + \mathbf{m}_{op2} + \mathbf{m}_{unload} + \mathbf{m}_{withdrawal} \geq 1$

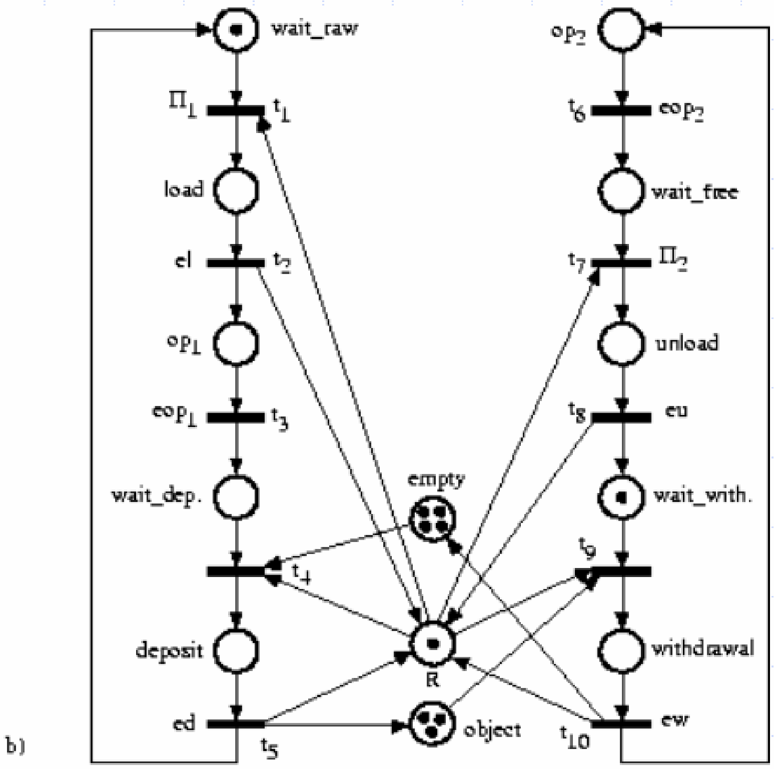
    then *$V(\mathbf{m})$ may decrease*

    else if $\mathbf{m}_{wait\_raw} + \mathbf{m}_{wait\_free} \geq 1$

    then *$V(\mathbf{m})$ may decrease*

    else *$V(\mathbf{m})$ may decrease **OR***

    t_1 is the unique firable transition ($\Leftrightarrow \mathbf{m}_0$)



Convex geometry and PNs

- ❑ Liveness
  - ❑ $\sigma = t_1, t_2, t_3, t_4, t_5, t_9, t_{10}, t_6, t_7, t_8$ is firable
  - ❑ The net is reversible

    Then it is live
- ❑ Fairness
  - ❑ *C* has a unique left annuller

    $\mathbf{x} = (1,1,1,1,1,1,1,1,1,1)^T$

    for all scheduling: all components work!

Convex geometry and PNs

- ❑ Linear programming and PNs
  - ❑ Example: structural marking bound of a place

[LPP]

$$\begin{aligned} \max \quad & m[p] \\ \text{s.t.} \quad & m = m_0 + \mathbf{C} \cdot \boldsymbol{\sigma} \\ & (m, \boldsymbol{\sigma}) \in \mathrm{N}^{n+m} \end{aligned}$$

- ❑ Polinomial time (on the net struct. size) computation
- ❑ Other properties can be analyzed: synchronic properties, dead transitions, mutex, etc.

Convex geometry and PNs

- ❑ General comments
  - ❑ Advantages:
    - ❑ Efficient computation
    - ❑ Analysis independent of initial marking ($\mathbf{m}_0$ is only a parameter)
  - ❑ Problems:
    - ❑ Only necessary or sufficient conditions are obtained (in general)
    - ❑ The heart of the matter is that σ (vector) does not exactly represent σ (sequence)

Outline

- ❑ Basic properties
- ❑ Analysis techniques
- ❑ Reachability graph
- ❑ Net transformations
- ❑ Convex geometry and PNs
- ❑ Bibliography

Bibliography

- J.M. Colom, E. Teruel, M. Silva: **Logical properties of P/T systems and their analysis.** 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. Download here.
- M. Silva, E. Teruel, J.M. Colom: **Linear algebraic and linear programming techniques for the analysis of net systems.** *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. Download here.

Modelling and analysis of concurrent systems with Petri nets. Performance evaluation

4. Time augmented Petri nets

Javier Campos
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, Spain
jcampos@unizar.es

Outline

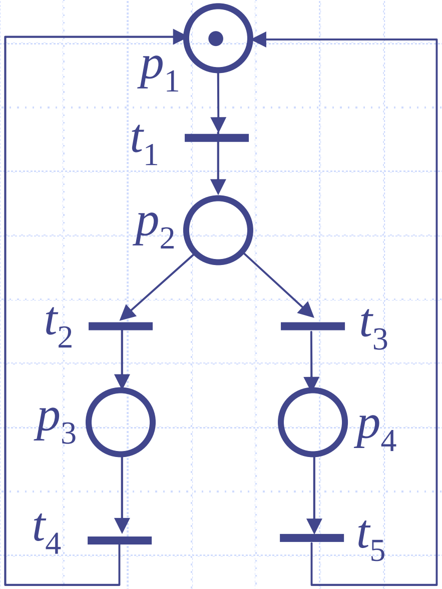
- Introduction
- Interpreted graphs
- Interpreted Petri nets
- Bibliography

Introduction

- **Formalism**: conceptual framework suited for a given purpose
- **Life cycle**: all phases, from preliminary design, detailed design, implementation, tuning...
- Different goals in each phase →
  → different formalisms
- Family of formalisms: PARADIGM

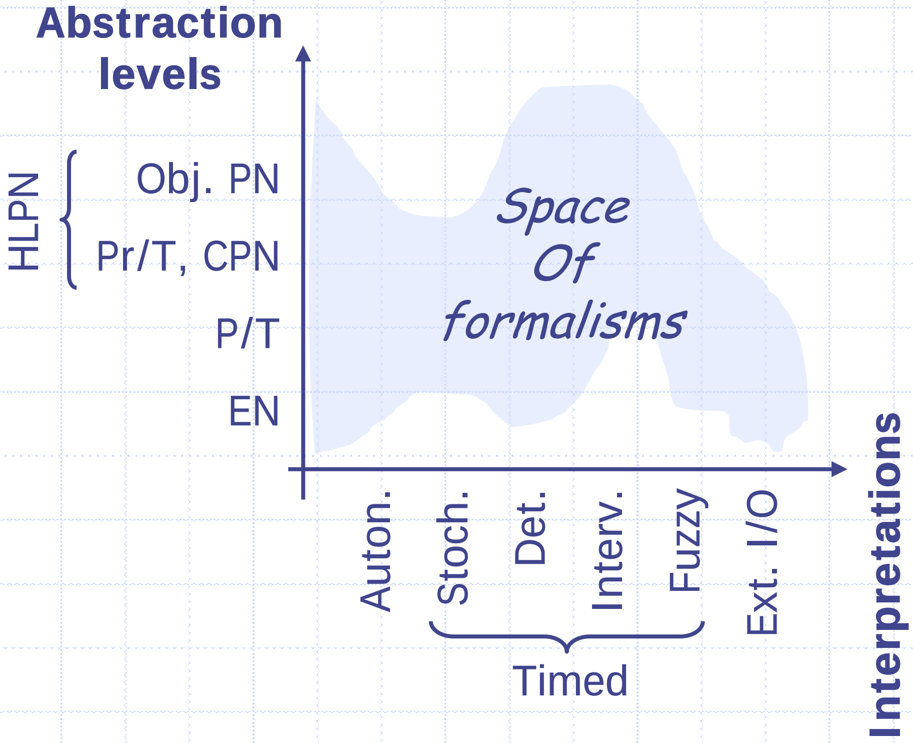
Introduction

- Why time augmenting the formalism?
  - Autonomous Petri nets
    - **Non-determinism** with respect to
      - **Which** enabled transition will fire?
      - **When** will it fire?
      - duration of activities and
      - routing
    - Not valid for performance evaluation (quantitative analysis: throughput, response time, average marking)



Introduction

- ❑ Formalism suitable for system life cycle. Two characteristics:
  - ❑ Different and interrelated abstraction levels
  - ❑ Different interpretations



Outline

- Introduction
- Interpreted graphs
- Interpreted Petri nets
- Bibliography

Interpreted graphs

- Interpreted graphs as formalisms for Discrete-Event Dynamic Systems

Basic initial idea:

Formalism = **graph** (precedence relations…) +
+ **interpretation** (meaning,
control…)

Interpreted graphs

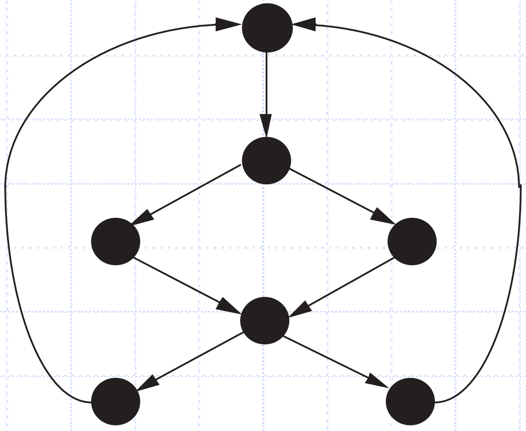
- □ Graphs as sequential formalisms

  (Valued) binary relations over a finite set (states, locations…) are represented as (valued) directed graphs

  - □ Vertices (entities)
  - □ Arcs (relations)

  Matricial representations

  - □ Adjacency (vertex-vertex)
  - □ Incidence (vertex-arc)



Graph

Interpreted graphs

- Interpretation
  - Just the “meaning” of mathematical entities
    - Example: locations and connections (static)

      typical problem: traveling salesman
  - Wider sense: meaning and external control of evolution
    - Meaning of entities
    - Connection of the model with the outside (the effect of the “rest of the world”)

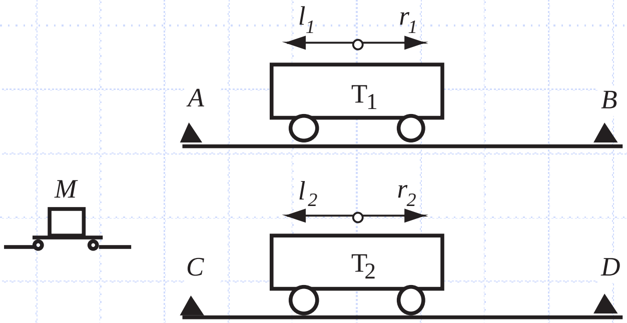
      events and external conditions

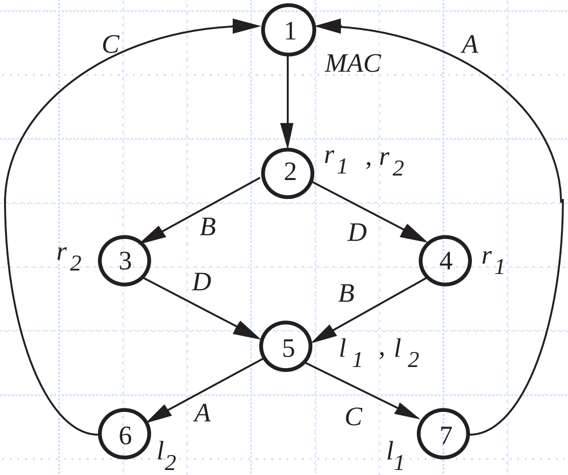
      what happened? When did it happen?

Interpreted graphs

- ❑ Example (interpretation 1): state diagram
  - ❑ Vertices: global states (possible values of unique state variable)
  - ❑ Arcs: transitions between states

  Sequentiel system (Moore like)





- ❑ conditions & input events → transitions
- ❑ output → states

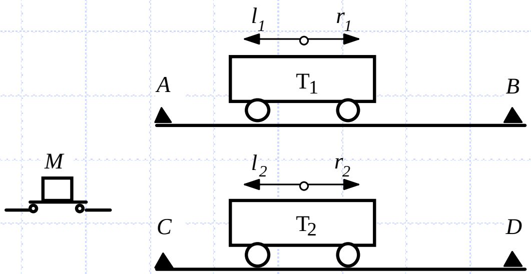
System evolution depends on the outside world through events and conditions represented with the input variables.

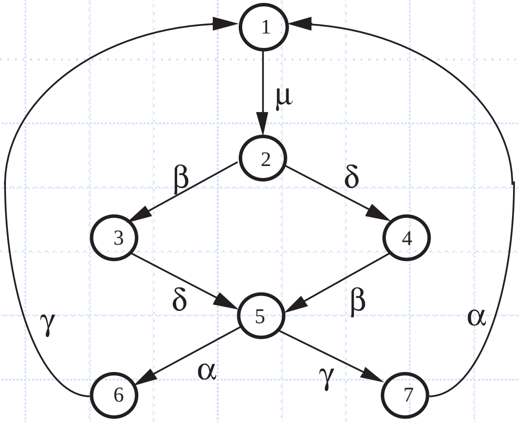
Interpreted graphs

- Other example (interpretation 2): Continuous Time Markov Chain

  State diagrams + “speed” of transitions

  - Vertices: global states (= state diagrams)
  - Arcs: transition rates between states

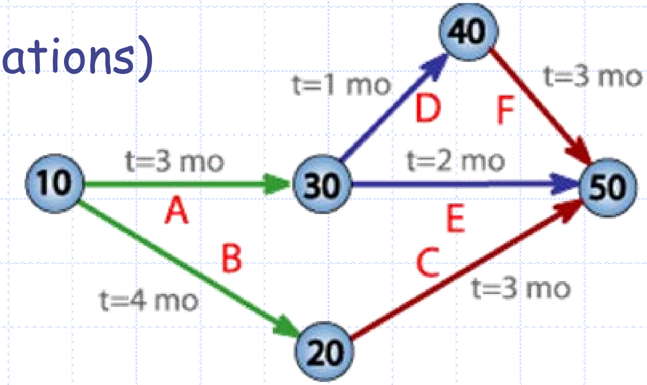




- system evolution depends on “outside” time
- events depend on time

Interpreted graphs

- ❑ Examples of formalisms for parallel behaviour
  - ❑ PERT (Program Evaluation and Review Technique)
    - ❑ Vertices = events
    - ❑ Arcs = activities (labelled with durations)
    - ❑ Special characteristics:
      - ❑ AND/AND logic (different from state diagrams or Markov chains)
      - ❑ Acyclic
      - ❑ Only one execution each time
      - ❑ Evolution depends on "outside" time (min, max, or average)
      - ❑ Distributed state of the system

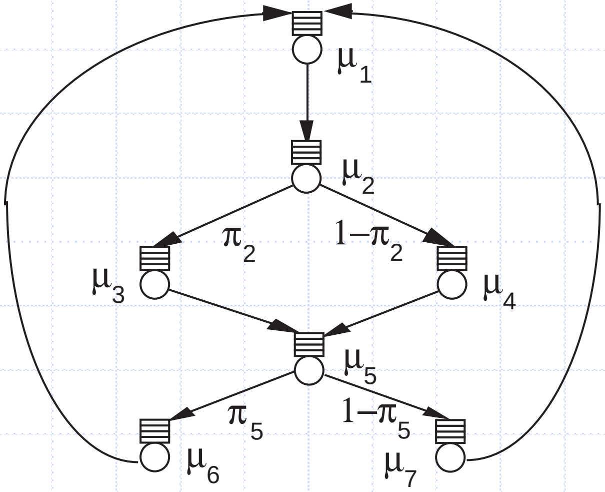


Typical problem: Critical Path Method

computation of shortest time to complete the project

Interpreted graphs

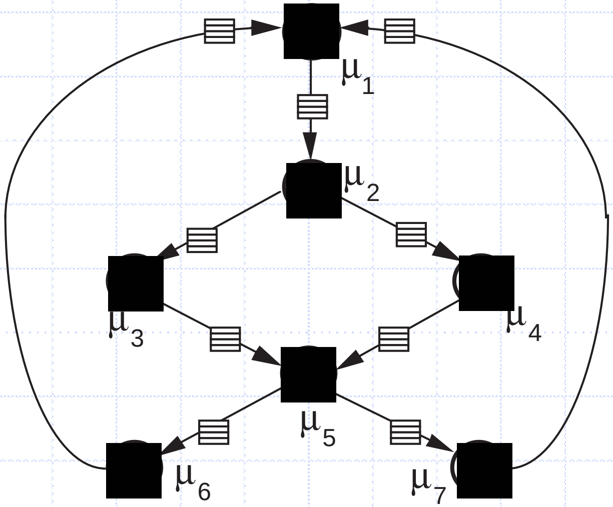
- ❑ Gordon-Newell queueing networks
  - ❑ Vertices = stations+queues
  - ❑ Arcs = routing of jobs
  - ❑ Special characteristics:
    - ❑ No synchronizations
    - ❑ Parallel evolution of jobs
    - ❑ OR/OR logic (identity of job is preserved)
    - ❑ Distributed state of the system



Typical problems: performance queries (mean queue lengths, throughput, etc)

Interpreted graphs

- Fork-Join queueing networks
  - Vertices = stations
  - Arcs = queues
  - Special characteristics:
    - No decisions
    - Only forks and joins
    - AND/AND logic (jobs are created and destroyed)
    - Distributed state of the system



Typical problems: performance queries (mean queue lengths, throughput, etc)

Outline

- ❑ Introduction
- ❑ Interpreted graphs
- ❑ Interpreted Petri nets
- ❑ Bibliography

Interpreted Petri nets

Abstract formalism ↔ Reality

- Generic meaning:
  - Place = state variable
  - Marking = value of variable
  - Transition = transformation of state
  - Firing = event that produces transformation
- Particular meanings (annotations):
  - Place (and marking)
    - State of subsystem S_i
    - Condition C_j is true
    - Resource R_k is available
    - Stock of parts in a store…
  - Transition (and firing)
    - Subsystem S_i evolves
    - End of activity Aj
    - A customer arrives
    - A fail occurs…

Interpreted Petri nets

Interpretation

(relation with the environment)

⇓

Constraints over the evolution

(imposed by the environment)

⇓

Reduction of non-determinism

- Synchronization with signals (from the environment)
- Time constraints

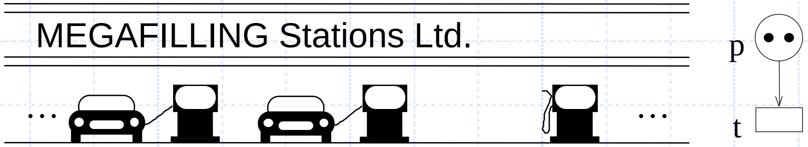
- Typical interpretations:
  - Marking diagrams (and Grafcet)
  - Timed interpretations (time augmented Petri nets)

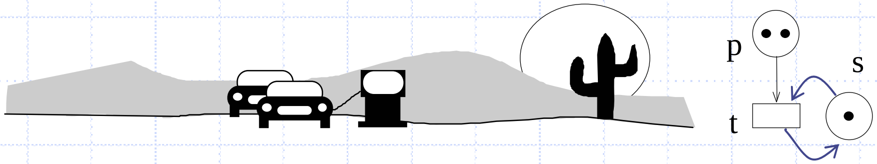
Interpreted Petri nets

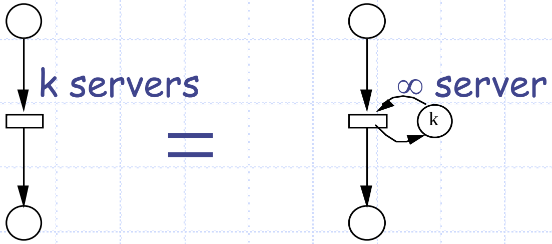
- Timed interpretations
  - Specification of activities and servers
    - sensibilization → start of activity
    - firing → end of activity
      } delay
    - transition → service station (# servers)

Specification of:

- delay
- # servers (multi-sensibilization: single, multiple, or infinite)

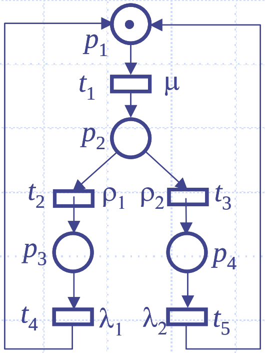




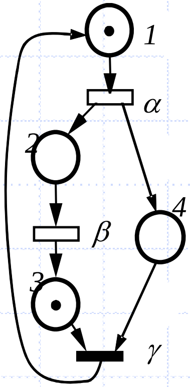


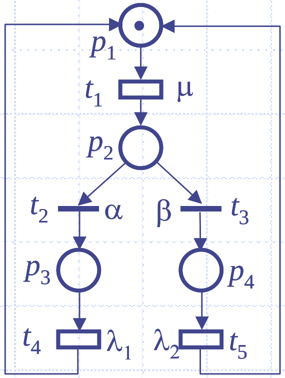
Interpreted Petri nets

- ❑ Specification of resolution of conflicts
  - ❑ race policy (race between timed enabled transitions)
  - ❑ preselection (random or deterministic choice)



- ❑ Immediate transitions
  - ❑ Modelling of synchronizations or routing
  - ❑ Zero delay ⇒ higher priority in case of conflict



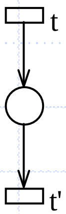


Interpreted Petri nets

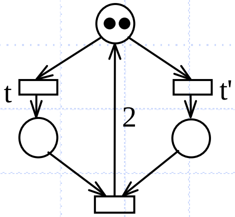
- Reduction of the non-determinism
  - Define duration of activities
    (elapsed time from enabling to firing of a transitions)
    - Constant → *Timed* Petri nets (TPN, Ramchandani, 1974)
    - Interval → *Time* Petri nets (TPN, Merlin and Faber, 1976)
    - Random (exponentially distrib.) → *Stochastic* Petri nets (SPN, Symons, 1978; Natkin, 1980; Molloy, 1981)
    - Random or immediate → *Generalized Stochastic* Petri nets (GSPN, Ajmone Marsan, Balbo, Conte, 1984)
  - Define server semantics (single/multiple/infinite)
  - Define routing at conflicts
    - Race between stochastically timed transitions
    - Preselection (probabilistic or deterministic choice)

Interpreted Petri nets

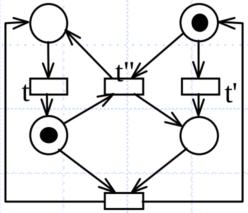
- Interpretation and logic properties
  - An interpretation restricts possible behaviour
    - Some reachable markings are not reachable anymore
    - Analysis of qualitative properties of the autonomous model can be non conclusive



unbounded?



total deadlock?



live?

- In general, a marking does not define a state
- In a SPN:
  - The same reachable makings than autonomous model (support of r.v. = [0,∞) and race policy gives positive probabilities to all possible outcomes of conflicts)
  - A marking does define a state (memoryless property)

Outline

- ❑ Introduction
- ❑ Interpreted graphs
- ❑ Interpreted Petri nets
- ❑ Bibliography

Bibliography

- M. Silva, E. Teruel: **A systems theory perspective of discrete event dynamic systems: the Petri net paradigm**. *Procs. of the Symposium on Discrete Events and Manufacturing Systems, CESA '96 IMACS Multiconference*, P.Borne, J.C. Gentina, E.Craye, S.El Khattabi (ed.), pp. 1-12, Lille, France, July 1996.
Download here.
- M. Silva, E. Teruel: **DEDS along their life-cycle: interpreted extension of Petri nets**. *Procs. of IEEE International Conference on Systems, Man and Cybernetics (SMC'98)*, San Diego, USA, 1998.
Download here.

Modelling and analysis of concurrent systems with Petri nets. Performance evaluation

5. Performance evaluation with PNs: classic technique

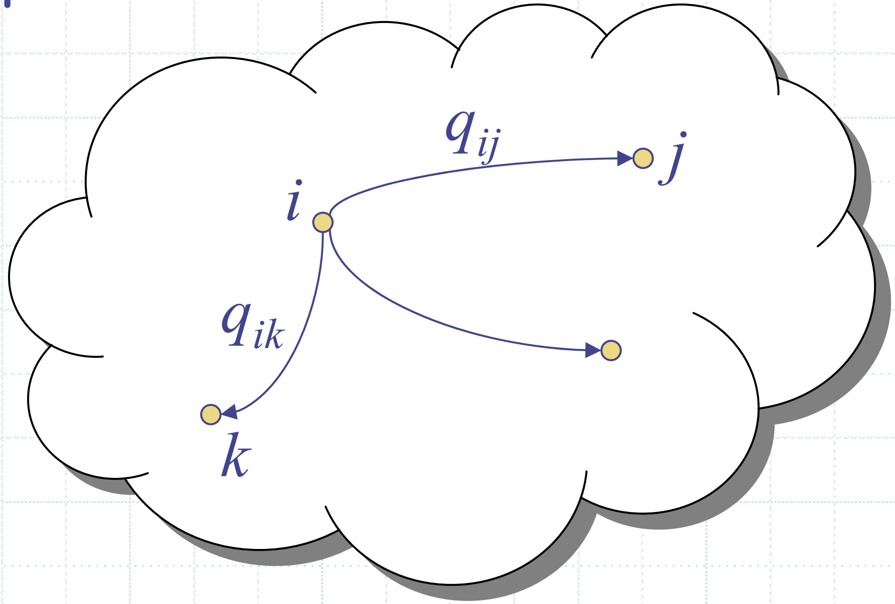
Javier Campos
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, Spain
jcampos@unizar.es

Outline

- ❑ Continuous time Markov chains
- ❑ Stochastic Petri nets
- ❑ CTMC-based exact analysis
- ❑ Bibliography

Continuous time Markov chains

- ❑ Stochastic process
  - ❑ discrete state space
  - ❑ continuous time



- ❑ q_{ij} is the **transition rate** from state i to state j

Continuous time Markov chains

- Formally:
  - A CTMC is a stochastic process $\{X(t) \mid t \geq 0, t \in IR\}$ s.t. for all $t_0,\ldots,t_{n-1},t_n,t \in IR$, $0 \leq t_0 < \ldots < t_{n-1} < t_n < t$, for all $n \in IN$

$$P(X(t)=x \mid X(t_n)=x_n, X(t_{n-1})=x_{n-1},\ldots, X(t_0)=x_0) =$$
$$= P(X(t)=x \mid X(t_n)=x_n)$$

  - Alternative (equivalent) definition: $\{X(t) \mid t \geq 0, t \in IR\}$ s.t. for all $t,s \geq 0$

$$P(X(t+s)=x \mid X(t)=x_t, X(u), 0 \leq u \leq t) =$$
$$= P(X(t+s)=x \mid X(t)=x_t)$$

Continuous time Markov chains

- Homogeneity
  - We are considering discrete state (sample) space, then we denote

$$p_{ij}(t,s) = P(X(t+s)=j \mid X(t)=i), \text{ for } s > 0.$$

  - A CTMC is called (time-)homogeneous if

$$p_{ij}(t,s) = p_{ij}(s) \quad \text{for all } t \geq 0$$

Continuous time Markov chains

- Time spent in a state:
  - Markov property and time homogeneity imply that if at time *t* the process is in state *j*, the time remaining in state *j* is independent of the time already spent in state *j* : memoryless property.

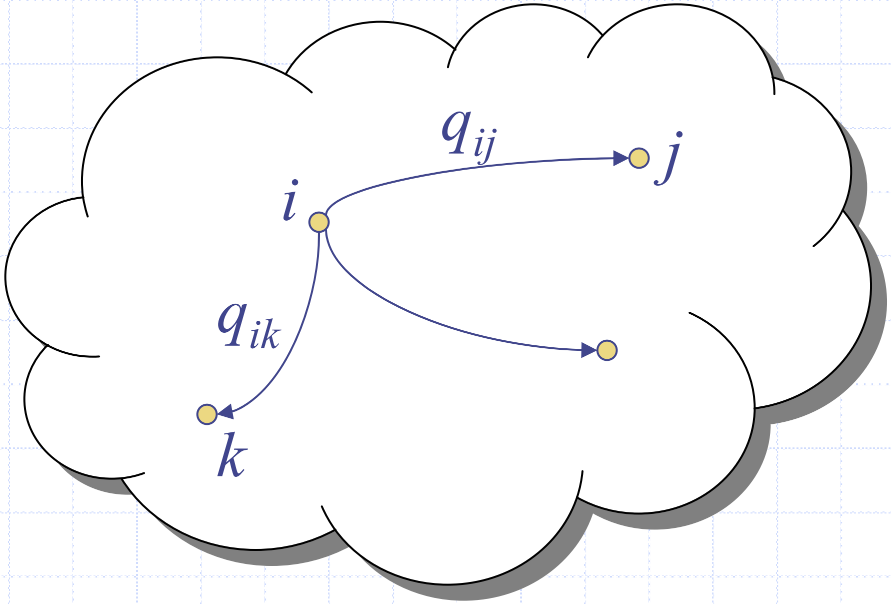
$$\begin{aligned}
P(S > t + s | S > t) &= P(X_{t+u} = j, 0 \leq u \leq s | X_u = j, 0 \leq u \leq t \\
&\quad \text{where S} = \text{time spent in state } j \\
&\quad \text{state } j \text{ entered at time } 0 \\
&= P(X_{t+u} = j, 0 \leq u \leq s | X_t = j) \text{ by MP} \\
&= P(X_u = j, 0 \leq u \leq s | X_0 = j) \text{ by T.H.} \\
&= P(S > s)
\end{aligned}$$

$\Rightarrow$ time spent in state *j* is exponentially distributed.

Continuous time Markov chains

- Transition rates:
  - In time-homogeneous CTMC, $p_{ij}(s)$ is the probability of jumping from i to j during an interval time of duration s.
  - Therefore, we define the instantaneous transition rate from state i to state j as:

$$q_{ij} = \lim_{\Delta t \to 0} \frac{p_{ij}(\Delta t)}{\Delta t}$$



  - And the exit rate from state i as $-q_{ii}$

$$q_{ii} = -\sum_{j \neq i} q_{ij} = \lim_{\Delta t \to 0} \frac{p_{ii}(\Delta t) - 1}{\Delta t}$$

  - $Q = [q_{ij}]$ is called **infinitesimal generator matrix** (*Q matrix*)

Continuous time Markov chains

- Steady-state distribution
  - Kolmogorov differential equation:

    Denote the distribution at instant t: $\pi_i(t) = P(X(t)=i)$

    And denote in matrix form: $P(t) = [p_{ij}(t)]$

    Then $\pi(t) = \pi(u)P(t-u)$, for $u < t$

    (we omit vector transposition to simplify notation)

    Substituting $u = t-\Delta t$ and substracting $\pi(t-\Delta t)$:

    $\pi(t) - \pi(t-\Delta t) = \pi(t-\Delta t)\ [P(\Delta t) - I]$, with I the identity matrix

    Dividing by Δt and taking the limit

    $$\frac{d}{dt}\pi(t) = \pi(t) \lim_{\Delta t \to 0} \frac{P(\Delta t) - I}{\Delta t}$$

    Then, by definition of $Q = [q_{ij}]$, we obtain the **Kolmogorov differential equation**

    $$\frac{d}{dt}\pi(t) = \pi(t)\, Q$$

Continuous time Markov chains

- Since also $\pi(t)\mathbf{1}^T = 1$, with $\mathbf{1} = (1,1,\ldots,1)$
  If the following limit exists

  $$\lim_{t\to\infty} \pi(t)$$

  then taking the limit of Kolmogorov differential equation we get the equations for the steady-state probabilities:

$$\boxed{\begin{array}{l} \pi\, Q = \mathbf{0} \\ \pi\, \mathbf{1}^T = 1 \end{array}}$$

(*balance* equations)

(*normalizing* equation)

Outline

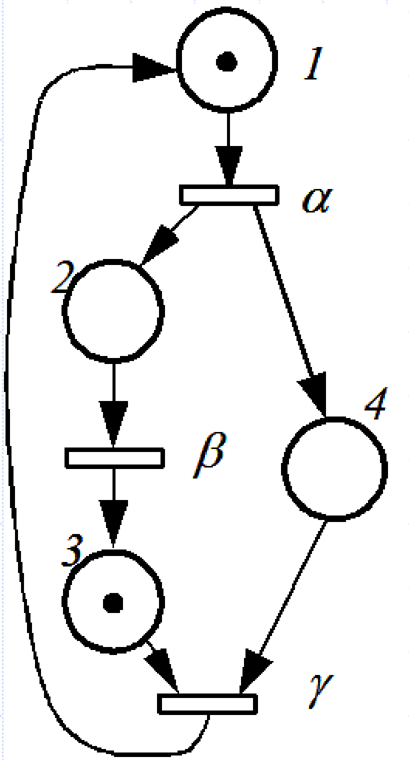
- ❑ Continuous time Markov chains
- ❑ Stochastic Petri nets
- ❑ CTMC-based exact analysis
- ❑ Bibliography

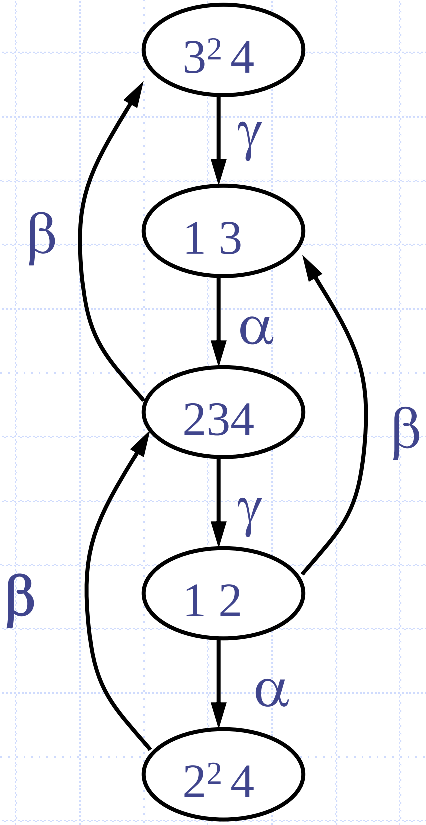
Stochastic Petri nets

- Time interpretation of Petri nets:
  - Duration of activities: exponentially distributed random variables
  - Single server semantics at each transition
  - Conflicts resolution: race policy

The reachability graph of the SPN is isomorphic to a Continuous Time Markov Chain

Stochastic Petri nets





The reachability graph of the SPN is isomorphic to a Continuous Time Markov Chain

Stochastic Petri nets

- The CTMC associated with a (bounded) SPN is obtained:
  - The state space $S = \{s_i\}$ of the CTMC is equal to the reachability set $RS(m_0)$ of the underlying PN ($m_i \leftrightarrow s_i$)
  - The transition rate from state s_i (corresponding to marking m_i) to state s_j (m_j) is obtained as the sum of the service rates of transitions enabled in m_i whose firing leads to marking m_j.
- If transitions have single-server semantics and marking independent rates, the components of Q are:

$$q_{ij} = \begin{cases} \displaystyle\sum_{T_k \in e_j(m_i)} w_k, & \text{si} \quad i \neq j \\ -q_i, & \text{si} \quad i = j \end{cases}$$

where

$$q_i = \sum_{T_k \in e(m_i)} w_k$$

$$e_j(m_i) = \{T_h \mid T_h \in e(m_i) \wedge m_i \xrightarrow{T_h} m_j\}$$

Outline

- ❑ Continuous time Markov chains
- ❑ Stochastic Petri nets
- ❑ CTMC-based exact analysis
- ❑ Bibliography

CTMC-based exact analysis

- Let $\pi(m_i,\tau)$ be the probability for the SPN to be at the state m_i at instant τ.

- The Kolmogorov differential equation for the associated CTMC is:

$$\frac{d\pi(m_i,\tau)}{d\tau} = \sum_{T_k \in T} q_{kj}\pi(m_k,\tau)$$

in matrix form: $$\frac{d\pi(\tau)}{d\tau} = \pi(\tau)Q$$

and its solution can be expressed as: $$\pi(\tau) = \pi(0)e^{Q\tau}$$

where $\pi(0)$ is the initial probability distribution

(usually $\pi_i(0)$ = 1 if m_i = m_0 and $\pi_i(0)$ = 0 otherwise)

CTMC-based exact analysis

- The steady-state "solution" of an SPN is based on the study of the probability distribution of the set of reachable markings

$$\pi = (\pi_1, \ldots, \pi_{|RS|})$$

- The limit behaviour of that distribution

$$\pi = \lim_{\tau \to \infty} \pi(\tau)$$

is computed by solving the following system of linear equations

$$\begin{cases} \pi\, Q = \mathbf{0} \\ \pi\, \mathbf{1}^{\mathrm{T}} = 1 \end{cases}$$

where $\mathbf{0}$ and $\mathbf{1}^{\mathrm{T}}$ are vectors of the size of π with all the components equal to 0 and 1 respectively

CTMC-based exact analysis

- The steady-state distribution π is used for the computation of performance indices of interest
- Performance indices can be expressed from *reward functions* defined over the markings of the SPN, the average reward is computed as average value of the reward of the steady-state distribution

$$R = \sum_{m_i \in RS(m_0)} r(m_i)\pi_i$$

where $r(m)$ represents a given reward function

CTMC-based exact analysis

- To compute the probability of a given condition $\Gamma(m)$ in the SPN
  - First, we define the reward function:

$$r(m) = \begin{cases} 1, & \text{if } \Gamma(m) = \mathit{true} \\ 0, & \text{otherwise} \end{cases}$$

  - Then, the desired probability is computed as:

$$P\{\Gamma\} = \sum_{m_i \in RS(m_0)} r(m_i)\pi_i = \sum_{m_i \in A} \pi_i$$

where

$$A = \{m_i \in RS(m_0) \mid \Gamma(m_i) = \mathit{true}\}$$

CTMC-based exact analysis

- Example: mean number of tokens at place p_j
  - The reward function is

$$r(m) = n \quad \text{if and only if} \quad m(p_j) = n$$

  - Then the average marking of place:

$$\overline{\mu}(p_j) = \sum_{m_i \in RS(m_0)} r(m_i)\pi_i = \sum_{\mathrm{n}>0} n\, P\{A(j,n)\}$$

where $A(j,n) = \{m_i \in RS(m_0) : m_i(p_j) = n\}$ and the sum is constrained to *n* ≤ *k* if place is *k*-bounded

CTMC-based exact analysis

- Other example: throughput of transition T_j (average number of firings per time unit)
  - A transition can fire only if it is enabled, thus the reward function is

$$r(m) = \begin{cases} w_j, & \text{si } T_j \in e(m) \\ 0, & \text{en otro caso} \end{cases}$$

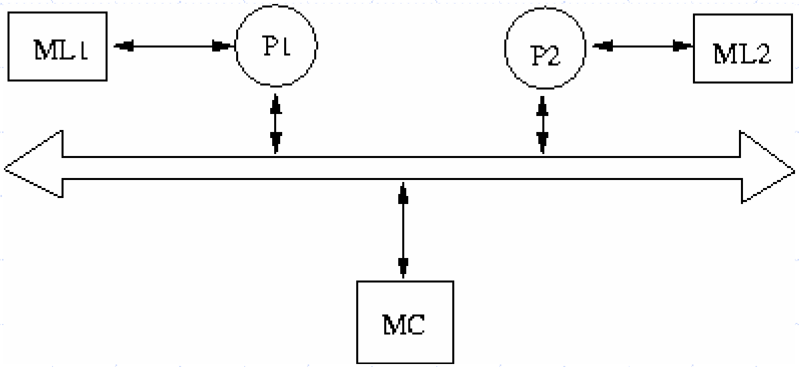
  - Then the throughput of T_j is

$$\chi_j = \sum_{m_i \in RS(m_0)} r(m_i)\pi_i = \sum_{m_i \in A_j} w_j \pi_i$$

where $A_j = \{m_i \in RS(m_0) : T_j \in e(m_i)\}$

CTMC-based exact analysis

- ❑ Shared memory multiprocessor



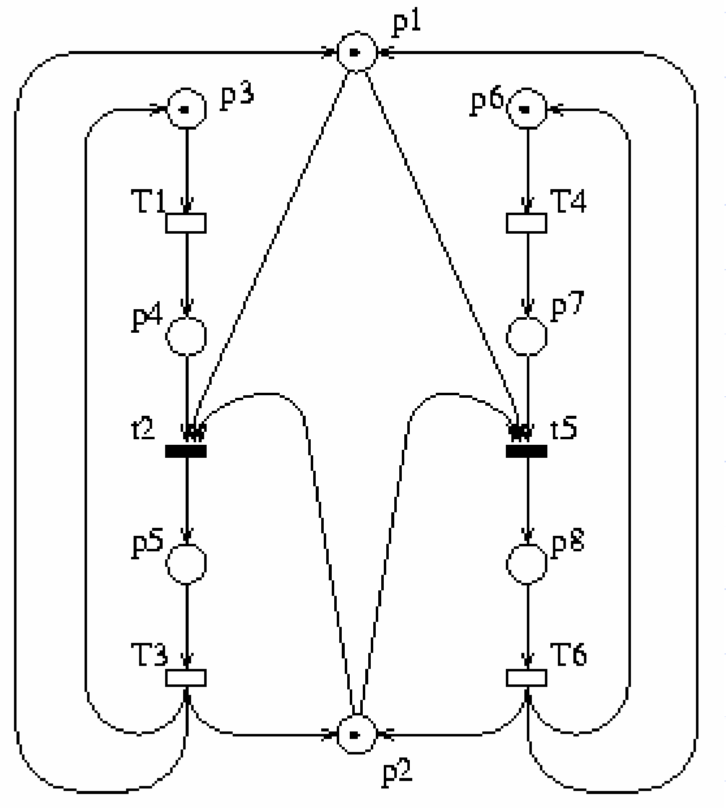
Both processors behave in a similar way:

- ❑ A cyclic sequence of: local activity, then
- ❑ an access request to the shared memory, and then
- ❑ accessing the shared memory

CTMC-based exact analysis

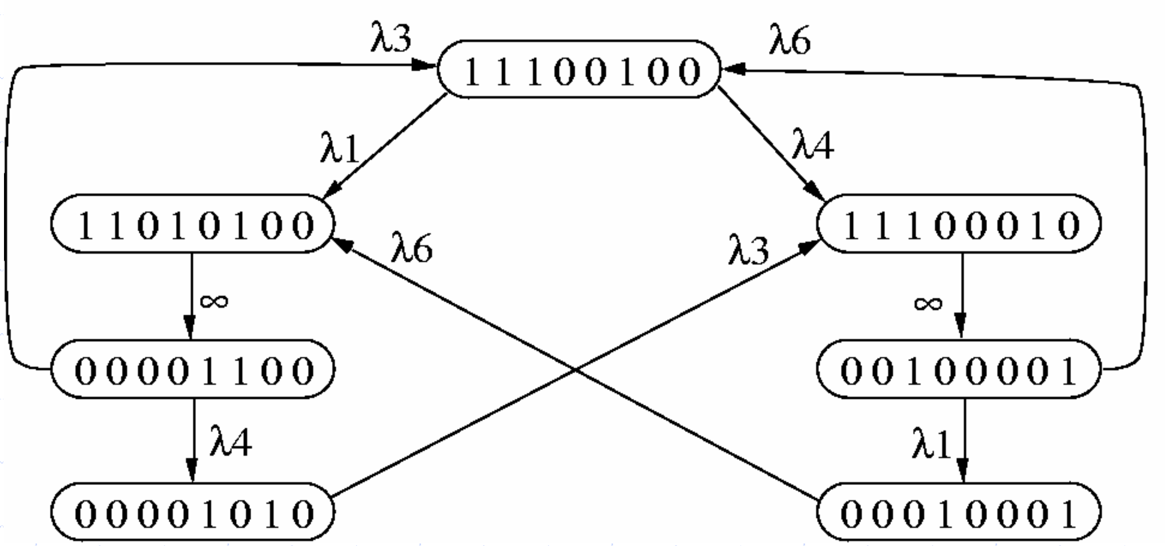
- ❑ All transitions have exponentially distributed durations, except for t2 and t5,
  access request to the shared memory (immediate)

  → GSPN



CTMC-based exact analysis

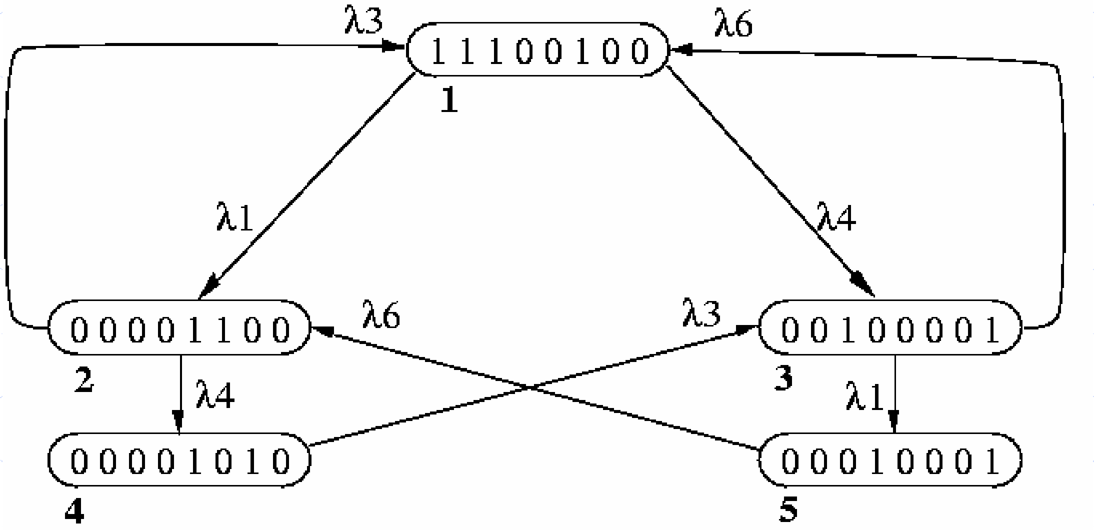
- Reachability graph



It is not isomorphic to a Continuous Time Markov Chain (infinite rates are not allowed in CTMCs)

CTMC-based exact analysis

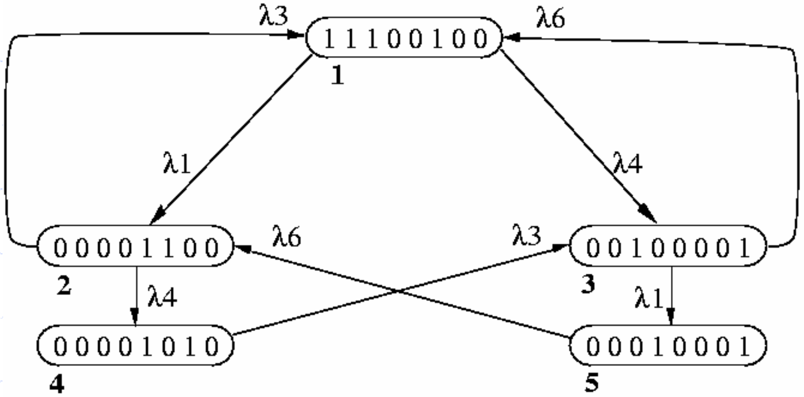
- Tangible reachability graph



It is isomorphic to a Continuous Time Markov Chain

CTMC-based exact analysis

- ❑ Infinitesimal generator matrix of the CTMC



$$\begin{bmatrix} -(\lambda_1+\lambda_4) & \lambda_1 & \lambda_4 & 0 & 0 \\ \lambda_3 & -(\lambda_3+\lambda_4) & 0 & \lambda_4 & 0 \\ \lambda_6 & 0 & -(\lambda_1+\lambda_6) & 0 & \lambda_1 \\ 0 & 0 & \lambda_3 & -\lambda_3 & 0 \\ 0 & \lambda_6 & 0 & 0 & -\lambda_6 \end{bmatrix}$$

CTMC-based exact analysis

- The stationary distribution can be computed (steady state probability of each state)

$$(\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \boldsymbol{\pi}_3, \boldsymbol{\pi}_4, \boldsymbol{\pi}_5) \cdot \begin{bmatrix} -(\lambda_1+\lambda_4) & \lambda_1 & \lambda_4 & 0 & 0 \\ \lambda_3 & -(\lambda_3+\lambda_4) & 0 & \lambda_4 & 0 \\ \lambda_6 & 0 & -(\lambda_1+\lambda_6) & 0 & \lambda_1 \\ 0 & 0 & \lambda_3 & -\lambda_3 & 0 \\ 0 & \lambda_6 & 0 & 0 & -\lambda_6 \end{bmatrix} = \mathbf{0}$$

$$\boldsymbol{\pi}_1 + \boldsymbol{\pi}_2 + \boldsymbol{\pi}_3 + \boldsymbol{\pi}_4 + \boldsymbol{\pi}_5 = 1$$

- And from here, compute, for instance, *utilization rate of shared memory*
  - In this case, it is equal to the steady-state probability of the unique state with p_2 (shared memory is free) marked

$$\bar{\mu}[p_2] = \pi_1$$

CTMC-based exact analysis

- Other example, *processing power*
  - Average number of processors effectively (locally) working
  - We define the reward function

$$r_P(m) = m[p_3] + m[p_6]$$

  - Then:

$$P = \sum_{m_i \in RS(m_0)} r_P(m_i)\pi_i = 2\pi_1 + \pi_2 + \pi_3$$

Outline

- □ Continuous time Markov chains
- □ Stochastic Petri nets
- □ CTMC-based exact analysis
- □ Bibliography

Bibliography

- J. Campos: **Evaluación de Prestaciones de Sistemas Concurrentes Modelados con Redes de Petri**. *Actas de la XI Escuela de Verano de Informática de la Universidad de Castilla-La Mancha*, pp. 141-156, Universidad de Castilla-La Mancha, Albacete, Spain, Departamento de Informática, In Spanish. July 2001.
  Download here.
- M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, G. Franceschinis: **Modelling with Generalized Stochastic Petri Nets**. Wiley Series in Parallel Computing, John Wiley and Sons, 1995 (out of print).
  Download here (a revised version).

Modelling and analysis of concurrent systems with Petri nets. Performance evaluation

6.1. Structure based performance analysis techniques: Bounds

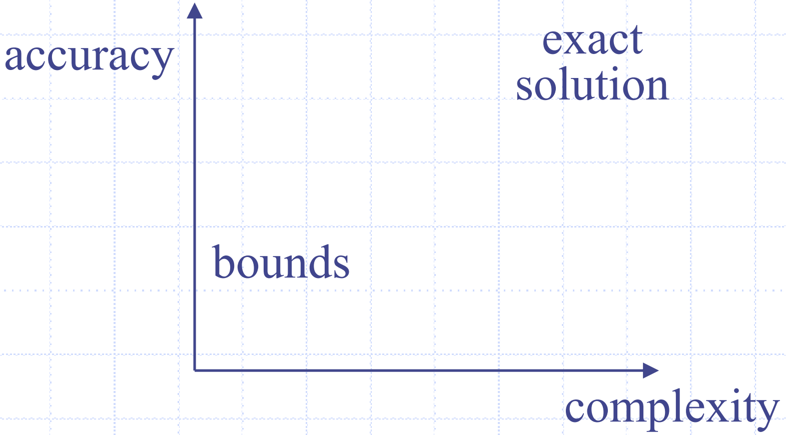
Javier Campos
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, Spain
jcampos@unizar.es

Outline

- Preliminary comments
- Introducing the ideas: Marked Graphs case
- Generalization: use of visit ratios
- Improvements of the bounds
- A general linear programming statement
- Bibliography

Preliminary comments

- Interest of bounding techniques
  - preliminary phases of design
    - many parameters are not known accurately
    - quick evaluation and rejection of those clearly bad



Preliminary comments

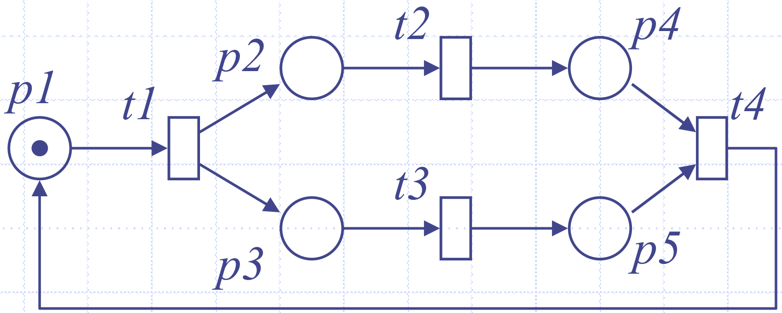
- □ Net-driven solution technique
  - □ stressing the intimate relationship between qualitative and quantitative aspects of PN's
  - □ structure theory of net models

→ efficient computation techniques

Outline

- □ Preliminary comments
- □ Introducing the ideas: Marked Graphs case
- □ Generalization: use of visit ratios
- □ Improvements of the bounds
- □ A general linear programming statement
- □ Bibliography

Introducing ideas: Marked Graph case



generally distributed service times (random variables X_i with mean $\bar{\mathbf{s}}[t_j]$)

we assume **infinite-server semantics**

exact cycle time (random variable): $X = X_1 + \max\{X_2, X_3\} + X_4$

average cycle time: $\Gamma = \bar{\mathbf{s}}[t_1] + \mathrm{E}[\max\{X_2, X_3\}] + \bar{\mathbf{s}}[t_4]$

but (non-negative variables): $X_2, X_3 \le \max\{X_2, X_3\} \le X_2 + X_3$

therefore:

$$\boxed{\bar{\mathbf{s}}[t_1] + \max\{\bar{\mathbf{s}}[t_2], \bar{\mathbf{s}}[t_3]\} + \bar{\mathbf{s}}[t_4] \le \Gamma \le \bar{\mathbf{s}}[t_1] + \bar{\mathbf{s}}[t_2] + \bar{\mathbf{s}}[t_3] + \bar{\mathbf{s}}[t_4]}$$

Introducing ideas: Marked Graph case

Thus, the lower bound for the average cycle time is computed looking for the slowest circuit

$$\Gamma \geq \max_{\substack{C \in \{\text{circuits} \\ \text{of the net}\}}} \left(\sum_{t_i \in C} \bar{\mathbf{s}}[t_i] \Big/ \#\text{tokens in } C \right)$$

Interpretation:

an MG may be built synchronising circuits, so we look for the bottleneck

Introducing ideas: Marked Graph case

- Computation:

$$\begin{aligned}\Gamma \geq\ & \text{maximum}\ \mathbf{y}\cdot\mathbf{Pre}\cdot\bar{\mathbf{s}}\\ & \text{subject to}\ \mathbf{y}\cdot\mathbf{C}=\mathbf{0}\\ & \qquad\qquad \mathbf{y}\cdot\mathbf{m_0}=1\\ & \qquad\qquad \mathbf{y}\geq\mathbf{0}\end{aligned}$$

($\bar{\mathbf{s}}$ is the vector of average service times)

(the proof of this comes later for a more general case)

solving a linear programming problem (**polynomial complexity** on the net size)

Introducing ideas: Marked Graph case

- Even if naïf, the bounds are tight!
- Lower bound for the average cycle time

$$\max\{\bar{\mathbf{s}}[t_2],\bar{\mathbf{s}}[t_3]\} \leq \mathrm{E}[\max\{X_2, X_3\}]$$

  - it is exact for deterministic timing
  - it cannot be improved using only mean values of r.v. (it is reached in a limit case for a family of random variables with arbitrary means and variances)

Introducing ideas: Marked Graph case

$$X_{\mu,\sigma}(\alpha) = \begin{cases} \mu\alpha & \text{with probability } 1-\varepsilon \\ \mu\left(\alpha + \dfrac{1-\alpha}{\varepsilon}\right) & \text{with probability } \varepsilon \end{cases} \qquad \varepsilon = \frac{\mu^2(1-\alpha)^2}{\mu^2(1-\alpha)^2+\sigma^2}$$

$(0 \le \alpha \le 1)$

$$\mathrm{E}\left[X_{\mu,\sigma}(\alpha)\right] = \mu \ ; \ \mathrm{Var}\left[X_{\mu,\sigma}(\alpha)\right] = \sigma^2$$

$$\lim_{\alpha\to 1} \mathrm{E}\left[\max\left(X_{\mu,\sigma}(\alpha), X_{\mu',\sigma'}(\alpha)\right)\right] = \max\left(\mu,\mu'\right)$$

$$\mathrm{E}\left[X_{\mu,\sigma}(\alpha) + X_{\mu',\sigma'}(\alpha)\right] = \mu + \mu', \ \forall \ 0 \le \alpha < 1$$

they behave "as deterministic" for the 'max' and '+' operators in the limit ($\alpha \to 1$)

Introducing ideas: Marked Graph case

- ❑ Upper bound for the average cycle time

$$\boxed{\Gamma \le \sum_{t\in T} \bar{\mathbf{s}}[t]}$$

  - ❑ it cannot be improved for 1-live MG's using only mean values of r.v. (it is reached in a limit case for a family of random variables with arbitrary means)

Introducing ideas: Marked Graph case

$$X^i_\mu(\varepsilon) = \begin{cases} 0 & \text{with probability} \quad 1-\varepsilon^i \\ \dfrac{\mu}{\varepsilon^i} & \text{with probability} \quad \varepsilon^i \end{cases}$$

$(0 < \varepsilon < 1)$

$$\mathrm{E}\left[X^i_\mu(\varepsilon)\right] = \mu \; ; \; \mathrm{E}\left[X^i_\mu(\varepsilon)^2\right] = \frac{\mu^2}{\varepsilon^i}$$

If $X_j = X^{j-1}_{\bar{\mathbf{s}}[t_j]}(\varepsilon), \ \forall t_j \in T,$ then for varying (decreasing) values of ε:

$$\boxed{\mathrm{E}[\max(X_i, X_j)] = \bar{\mathbf{s}}[t_i] + \bar{\mathbf{s}}[t_j] + o(\varepsilon)}$$

Outline

- ❑ Preliminary comments
- ❑ Introducing the ideas: Marked Graphs case
- ❑ Generalization: use of visit ratios
- ❑ Improvements of the bounds
- ❑ A general linear programming statement
- ❑ Bibliography

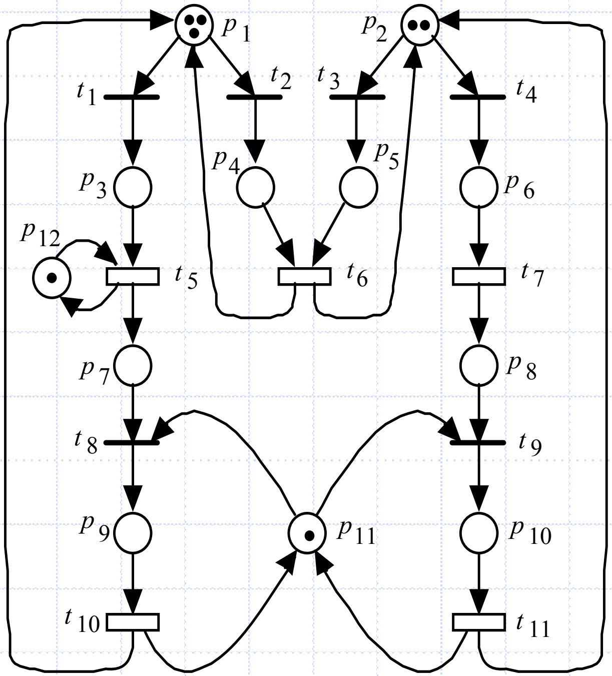
Generalization: use of visit ratios

- ❑ Visit ratios = relative throughput
  (number of visits to t_i per each visit to t_1)

$$\mathbf{v}[t] = \frac{\chi[t]}{\chi[t_1]} = \underbrace{\Gamma[t_1]}_{\text{average interfiring time of } t_1}\ \chi[t]$$

Generalization: use of visit ratios

- For some net classes **v** can be computed as:



$$\mathbf{C} \cdot \mathbf{v} = \mathbf{0};$$

$$r_1 \mathbf{v}[t_2] = r_2 \mathbf{v}[t_1];$$

$$r_3 \mathbf{v}[t_4] = r_4 \mathbf{v}[t_3];$$

$$\mathbf{v}[t_1] = 1$$

Generalization: use of visit ratios

- Little's law ($L=\lambda W$) applied to a place *p*:

$$\overline{\mu}[p] = (\mathbf{Pre}[p,T]\cdot\chi)\ \overline{\mathbf{r}}[p]$$

Assume that timed transitions are never in conflict (**conflicts are modelled with immediate transitions**), then either all output transitions of *p* are immediate or *p* has a unique output transition, say t_1, and t_1 is timed, thus:

$$\begin{aligned}\overline{\mu}[p] &= (\mathbf{Pre}[p,T]\cdot\chi)\ \overline{\mathbf{r}}[p] = \mathbf{Pre}[p,t_1]\ \ \chi[t_1]\ \overline{\mathbf{r}}[p]\\ &\geq \mathbf{Pre}[p,t_1]\ \ \chi[t_1]\ \overline{\mathbf{s}}[t_1] = \sum_{j=1}^{m} \mathbf{Pre}[p,t_j]\ \ \chi[t_j]\ \overline{\mathbf{s}}[t_j]\end{aligned}$$

Generalization: use of visit ratios

Then: $\Gamma[t_1]\ \bar{\mu}[p] \ge \sum_{j=1}^{m} \mathbf{Pre}[p,t_j]\ \Gamma[t_1]\ \chi[t_j]\ \bar{\mathbf{s}}[t_j] = \sum_{j=1}^{m} \mathbf{Pre}[p,t_j]\ \mathbf{v}[t_j]\ \bar{\mathbf{s}}[t_j]$

Hence: $\Gamma[t_1]\ \bar{\mu} \ge \mathbf{Pre}\cdot\bar{\mathbf{D}}$ where $\bar{\mathbf{D}}[t] = \bar{\mathbf{s}}[t]\mathbf{v}[t]$ is the average service demand of t

Premultiplying by a *P*-semiflow **y**

$(\mathbf{y}\cdot\mathbf{C}=\mathbf{0},\ \mathbf{y}\ge\mathbf{0},\ \text{thus}\ \mathbf{y}\cdot\bar{\mu} = \mathbf{y}\cdot\mathbf{m_0}),$

$$\begin{aligned} \Gamma[t_1] \ge \quad & \text{maximum} && \frac{\mathbf{y}\cdot\mathbf{Pre}\cdot\bar{\mathbf{D}}}{\mathbf{y}\cdot\mathbf{m_0}} \\ & \text{subject to} && \mathbf{y}\cdot\mathbf{C}=\mathbf{0} \\ &&& \mathbf{1}\cdot\mathbf{y} > 0 \\ &&& \mathbf{y}\ge\mathbf{0} \end{aligned} \quad \Longleftrightarrow \quad \begin{aligned} \Gamma[t_1] \ge \quad & \text{maximum} && \frac{\mathbf{y}\cdot\mathbf{Pre}\cdot\bar{\mathbf{D}}}{q} \\ & \text{subject to} && \mathbf{y}\cdot\mathbf{C}=\mathbf{0} \\ &&& \mathbf{1}\cdot\mathbf{y} > 0 \\ &&& q = \mathbf{y}\cdot\mathbf{m_0} \\ &&& \mathbf{y}\ge\mathbf{0} \end{aligned}$$

Generalization: use of visit ratios

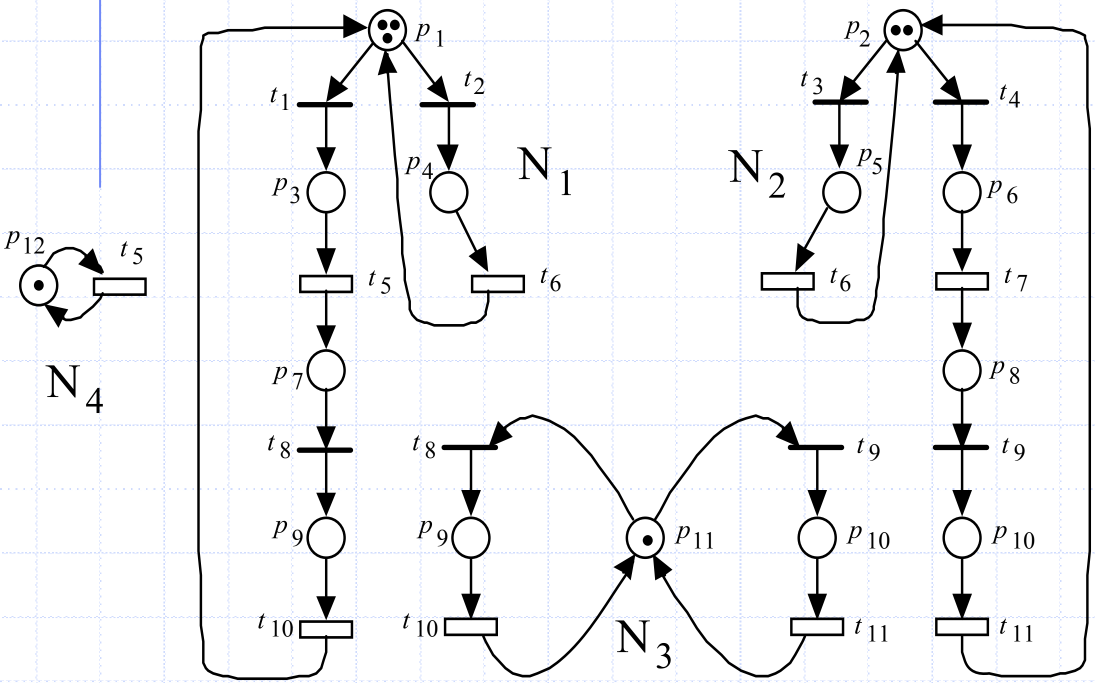
Since $\mathbf{y} \cdot \mathbf{m}_0 > 0$ (live system), we change $\mathbf{y}/q$ to $\mathbf{y}$ and we obtain ($\mathbf{1} \cdot \mathbf{y} > 0$ is removed because $\mathbf{y} \cdot \mathbf{m}_0 = 1$ implies $\mathbf{1} \cdot \mathbf{y} > 0$):

$$
\begin{array}{lll}
\Gamma[t_1] \geq & \text{maximum} & \mathbf{y} \cdot \mathbf{Pre} \cdot \overline{\mathbf{D}} \\
& \text{subject to} & \mathbf{y} \cdot \mathbf{C} = \mathbf{0} \\
& & \mathbf{y} \cdot \mathbf{m_0} = 1 \\
& & \mathbf{y} \geq \mathbf{0}
\end{array}
$$

again, a linear programming problem
(polynomial complexity on the net size)

Generalization: use of visit ratios

Interpretation: **slowest subsystem generated by *P*-semiflows, in isolation**



minimal *P*–semiflows

$\mathbf{y_1} = (1,0,1,1,0,0,1,0,1,0,0,0)$

$\mathbf{y_2} = (0,1,0,0,1,1,0,1,0.1,0,0)$

$\mathbf{y_3} = (0,0,0,0,0,0,0,0,1,1,1,0)$

$\mathbf{y_4} = (0,0,0,0,0,0,0,0,0,0,0,1)$

$$
\begin{aligned}
\Gamma[t_1] \geq \max \{ \; & (\bar{\mathbf{s}}[t_5]+\bar{\mathbf{s}}[t_6]+\bar{\mathbf{s}}[t_{10}])/3, \\
& (\bar{\mathbf{s}}[t_6]+\bar{\mathbf{s}}[t_7]+\bar{\mathbf{s}}[t_{11}])/2, \\
& \bar{\mathbf{s}}[t_{10}]+\bar{\mathbf{s}}[t_{11}], \\
& \bar{\mathbf{s}}[t_5] \; \}
\end{aligned}
$$

Generalization: use of visit ratios

- Upper bound for the average interfiring time

$$\Gamma[t_1] \le \sum_{t\in T} \mathbf{v}[t]\ \bar{\mathbf{s}}[t] = \sum_{t\in T} \overline{\mathbf{D}}[t]$$

remember the marked graphs case (**v** = **1**): $\Gamma \le \sum_{t\in T} \bar{\mathbf{s}}[t]$

Outline

- ❑ Preliminary comments
- ❑ Introducing the ideas: Marked Graphs case
- ❑ Generalization: use of visit ratios
- ❑ Improvements of the bounds
- ❑ A general linear programming statement
- ❑ Bibliography

Improvements of the bounds

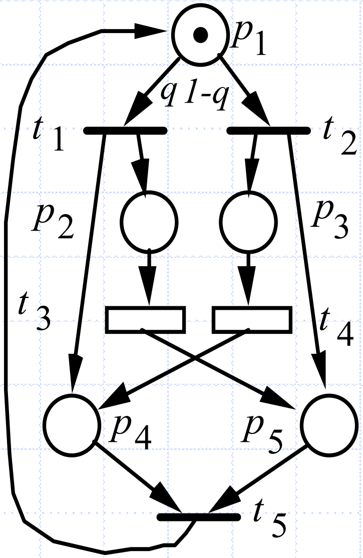
- ❑ Structural improvements

  bounds still based only on the mean values (not on higher moments of r.v., **insensitive** bounds)

  - ❑ lower bound for the average interfiring time: use of **implicit places** to increase the number of minimal *P*-semiflows
  - ❑ upper bound for the average interfiring time: use of **liveness bound of transitions** to improve the bound for some net subclasses

Improvements of the bounds

- Use of implicit places



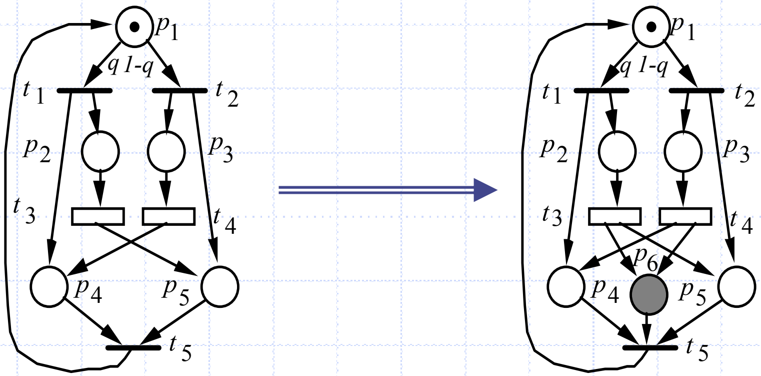
$$\Gamma[t_5] = q\bar{\mathbf{s}}[t_3] + (1-q)\bar{\mathbf{s}}[t_4]$$

$$\begin{array}{lll} \Gamma[t_1] \geq & \text{maximum} & \mathbf{y} \cdot \mathbf{Pre} \cdot \overline{\mathbf{D}} \\ & \text{subject to} & \mathbf{y} \cdot \mathbf{C} = \mathbf{0} \\ & & \mathbf{y} \cdot \mathbf{m_0} = 1 \\ & & \mathbf{y} \geq \mathbf{0} \end{array}$$

$$\Downarrow$$

$$\Gamma[t_5] \geq \max\left\{q\bar{\mathbf{s}}[t_3], (1-q)\bar{\mathbf{s}}[t_4]\right\}$$

Improvements of the bounds



$$\Gamma[t_1] \geq \begin{array}{ll} \text{maximum} & \mathbf{y}\cdot\mathbf{Pre}\cdot\overline{\mathbf{D}} \\ \text{subject to} & \mathbf{y}\cdot\mathbf{C}=\mathbf{0} \\ & \mathbf{y}\cdot\mathbf{m_0}=1 \\ & \mathbf{y}\geq\mathbf{0} \end{array}$$

$$\Gamma[t_5] = q\bar{\mathbf{s}}[t_3] + (1-q)\bar{\mathbf{s}}[t_4]$$

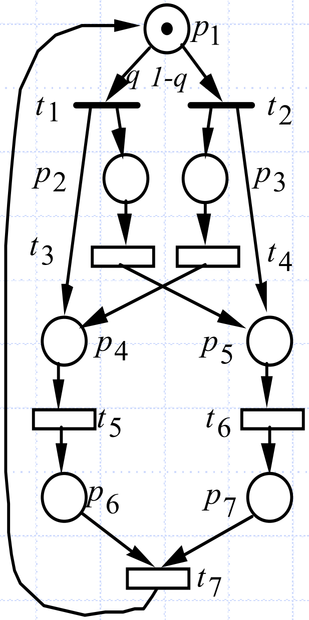
$$\Big\downarrow$$

$$\Gamma[t_5] \geq \max\left\{\, q\bar{\mathbf{s}}[t_3],\ (1-q)\bar{\mathbf{s}}[t_4],\ \underline{q\bar{\mathbf{s}}[t_3]+(1-q)\bar{\mathbf{s}}[t_4]} \,\right\}$$

in this case, we get the exact value!

Improvements of the bounds

in general…



$$\begin{aligned}\Gamma[t_1] \geq \quad & \text{maximum} && \mathbf{y}\cdot\mathbf{Pre}\cdot\overline{\mathbf{D}} \\ & \text{subject to} && \mathbf{y}\cdot\mathbf{C}=\mathbf{0} \\ & && \mathbf{y}\cdot\mathbf{m_0}=1 \\ & && \mathbf{y}\geq\mathbf{0}\end{aligned}$$

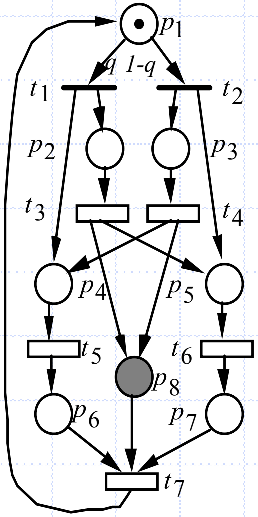
$$\Downarrow$$

$$\Gamma[t_7] \geq \max \{\ q\bar{\mathbf{s}}[t_3]+\bar{\mathbf{s}}[t_6]+\bar{\mathbf{s}}[t_7],$$

$$(1-q)\bar{\mathbf{s}}[t_4]+\bar{\mathbf{s}}[t_5]+\bar{\mathbf{s}}[t_7]\ \}$$

Improvements of the bounds

in general, the bound is non-reachable



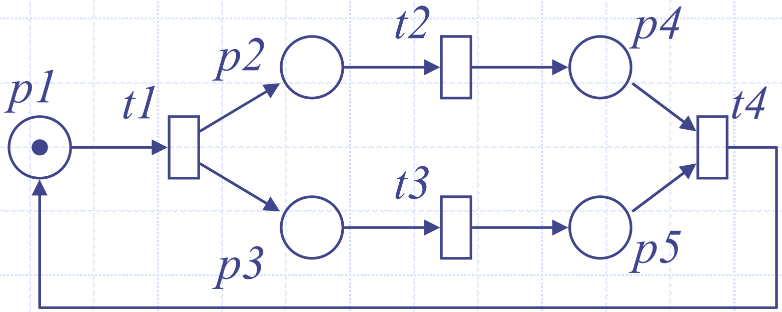
$$
\begin{aligned}
\Gamma[t_7] \geq \max \{\ & q\bar{\mathbf{s}}[t_3]+\bar{\mathbf{s}}[t_6]+\bar{\mathbf{s}}[t_7], \\
& (1-q)\bar{\mathbf{s}}[t_4]+\bar{\mathbf{s}}[t_5]+\bar{\mathbf{s}}[t_7], \\
& q\bar{\mathbf{s}}[t_3]+(1-q)\bar{\mathbf{s}}[t_4]+\bar{\mathbf{s}}[t_7]\ \}
\end{aligned}
$$

$$
\begin{aligned}
\Gamma[t_7] &= q\max\{\bar{\mathbf{s}}[t_5],\bar{\mathbf{s}}[t_3]+\bar{\mathbf{s}}[t_6]\}+(1-q)\max\{\bar{\mathbf{s}}[t_4]+\bar{\mathbf{s}}[t_5],\bar{\mathbf{s}}[t_6]\}+\bar{\mathbf{s}}[t_7] \\
&= \max\ \{\ q\bar{\mathbf{s}}[t_3]+\bar{\mathbf{s}}[t_6]+\bar{\mathbf{s}}[t_7], \\
&\qquad (1-q)\bar{\mathbf{s}}[t_4]+\bar{\mathbf{s}}[t_5]+\bar{\mathbf{s}}[t_7], \\
&\qquad q\bar{\mathbf{s}}[t_3]+(1-q)\bar{\mathbf{s}}[t_4]+(1-q)\bar{\mathbf{s}}[t_5]+q\bar{\mathbf{s}}[t_6]+\bar{\mathbf{s}}[t_7], \\
&\qquad q\bar{\mathbf{s}}[t_5]+(1-q)\bar{\mathbf{s}}[t_6]+\bar{\mathbf{s}}[t_7]\ \}
\end{aligned}
$$

(deterministic timing)

Improvements of the bounds

- Use of liveness bounds
  - upper bound for the average interfiring time:

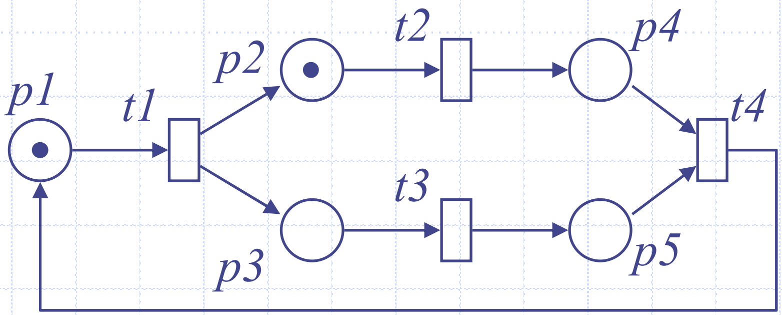


$$\Gamma \leq \sum_{t \in T} \bar{\mathbf{s}}[t]$$

reachable for 1-live marked graphs, but...

Improvements of the bounds

- it can be improved for *k*-live marked graphs

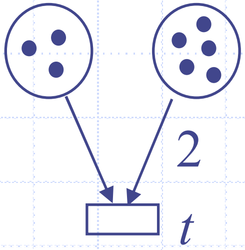


$$\Gamma \le \bar{\mathbf{s}}[t_1] + \frac{\bar{\mathbf{s}}[t_2]}{\underbrace{2}_{\text{liveness bound of } t_2}} + \bar{\mathbf{s}}[t_3] + \bar{\mathbf{s}}[t_4]$$

Improvements of the bounds

- ❑ Definitions of enabling degree, enabling bound, structural enabling bound, and liveness bound
  - ❑ instantaneous enabling degree of a transition at a given marking

$$\mathbf{e}[t](\mathbf{m}) = \sup\left\{k \in \mathbb{N} : \ \forall p \in {}^{\bullet}t, \ \mathbf{m}[p] \geq k \ \mathbf{Pre}[p,t]\right\}$$

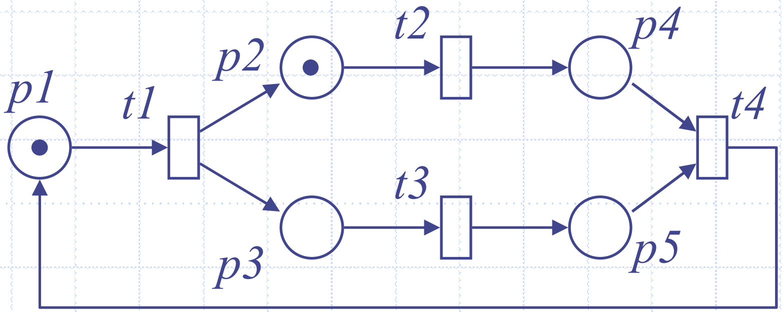


$\mathbf{e}[t](\mathbf{m}) = 2$

Improvements of the bounds

- enabling bound of a transition in a given system: maximum among the instantaneous enabling degree at all reachable markings

$$\mathbf{eb}[t] = \sup\left\{ k \in \mathrm{N} :\ \exists \mathbf{m_0} \xrightarrow{\sigma} \mathbf{m},\ \forall p \in {}^{\bullet}t,\ \mathbf{m}[p] \geq k\ \mathbf{Pre}[p,t] \right\}$$

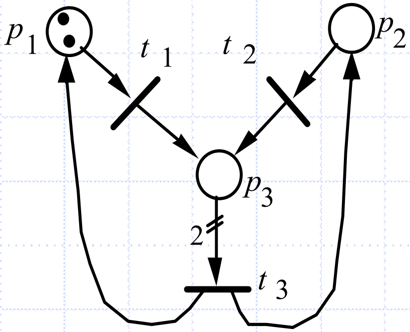


$$\mathbf{eb}[t_2] = 2$$

Improvements of the bounds

- ❑ liveness bound of a transition in a given system: number of servers available in *t* in steady state

$$\mathbf{lb}[t] = \sup\left\{ k \in \mathrm{N} : \ \forall \mathbf{m}', \mathbf{m_0} \xrightarrow{\sigma} \mathbf{m}', \exists \mathbf{m}, \mathbf{m}' \xrightarrow{\sigma'} \mathbf{m} \wedge \forall p \in {}^{\bullet}t, \mathbf{m}[p] \geq k\ \mathbf{Pre}[p,t] \right\}$$



$$\mathbf{lb}[t_1] = 1 < 2 = \mathbf{eb}[t_1]$$

Improvements of the bounds

- structural enabling bound of a transition in a given system: structural counterpart of the enabling bound (substitute reachability condition by $\mathbf{m} = \mathbf{m_0} + \boldsymbol{C} \cdot \sigma;\ \mathbf{m},\sigma \geq \mathbf{0}$)

$$\begin{aligned}\mathbf{seb}[t] = \ & \text{maximum } k \\ & \text{subject to } \ \mathbf{m_0}[p] + \mathbf{C}[p,T] \cdot \boldsymbol{\sigma} \geq k\ \mathbf{Pre}[p,t], \ \forall p \in P \\ & \qquad\qquad\quad \boldsymbol{\sigma} \geq 0\end{aligned}$$

Property: For any net system $\mathbf{seb}[t] \geq \mathbf{eb}[t] \geq \mathbf{lb}[t],\ \forall\ t$.

Property: For live and bounded free choice systems, $\mathbf{seb}[t] = \mathbf{eb}[t] = \mathbf{lb}[t],\quad \forall t$.

Improvements of the bounds

improvement of the bound for live and bounded free choice systems:

$$\Gamma[t_1] \leq \sum_{t \in T} \frac{\mathbf{v}[t]\ \bar{\mathbf{s}}[t]}{\mathbf{seb}[t]} = \sum_{t \in T} \frac{\overline{\mathbf{D}}[t]}{\mathbf{seb}[t]}$$

this bound cannot be improved for marked graphs (using only the mean values of service times)

Outline

- ❑ Preliminary comments
- ❑ Introducing the ideas: Marked Graphs case
- ❑ Generalization: use of visit ratios
- ❑ Improvements of the bounds
- ❑ A general linear programming statement
- ❑ Bibliography

A general linear programming statement

- The idea

a linear function

maximize [or minimize] $f(\overline{\mu}, \chi)$

subject to any linear constraint that we are able to state for $\overline{\mu}$, χ, and other needed additional variables

linear operational laws

A general linear programming statement

- A set of linear constraints:

$$\bar{\mu} = \mathbf{m_0} + \mathbf{C}\cdot\sigma$$ (state equation)

$$\sum_{t\in {}^{\bullet}p} \chi[t]\ \mathbf{Post}[p,t] \geq \sum_{t\in p^{\bullet}} \chi[t]\ \mathbf{Pre}[p,t], \quad \forall p \in P$$

$$\sum_{t\in {}^{\bullet}p} \chi[t]\ \mathbf{Post}[p,t] = \sum_{t\in p^{\bullet}} \chi[t]\ \mathbf{Pre}[p,t], \quad \forall p \in P \text{ bounded}$$

(flow balance equation)

$$\frac{\chi[t_i]}{r_i} = \frac{\chi[t_j]}{r_j}, \qquad \forall t_i, t_j \in T: \text{ behavioural free choice}$$

$$(\text{e.g.}\ \mathbf{Pre}[P,t_i] = \mathbf{Pre}[P,t_j])$$

$\ldots \qquad \ldots$

A general linear programming statement

$$\chi[t]\ \bar{\mathbf{s}}[t] \leq \frac{\bar{\mu}[p]}{\mathbf{Pre}[p,t]}, \qquad \forall t \in T,\ \forall p \in {}^{\bullet}t$$ (maximum throughput law)

$$\chi[t]\ \bar{\mathbf{s}}[t] \geq \frac{\bar{\mu}[p] - \mathbf{Pre}[p,t] + 1}{\mathbf{Pre}[p,t]}, \qquad \forall t \in T \text{ persistent, age memory or immediate: } {}^{\bullet}t = \{p\}$$ (minimum throughput law)

$$\ldots \qquad \ldots$$

$$\bar{\mu},\ \chi,\ \boldsymbol{\sigma} \geq \mathbf{0}$$

A general linear programming statement

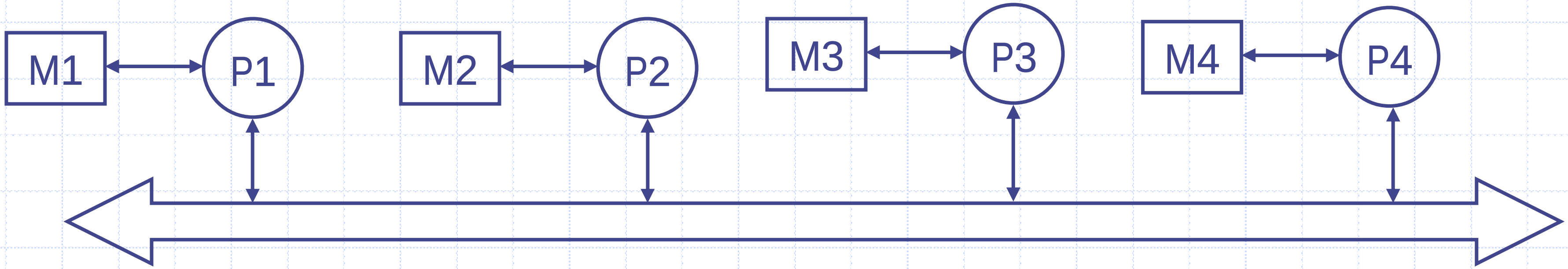
- □ It can be improved using second order moments
- □ It can be extended to well-formed coloured nets
- □ It has been recently extended to Time Petri Nets (timing based on intervals, usefull for the modelling and analysis of real-time systems)

A general linear programming statement

- It is implemented in *GreatSPN*
  - select place (transition) object ○ (▭)
  - click right mouse button and select "show"
  - click again right mouse button and select "Average M.B." ("LP Throughput Bounds")
  - click left mouse button for upper bound
  - click middle mouse button for lower bound

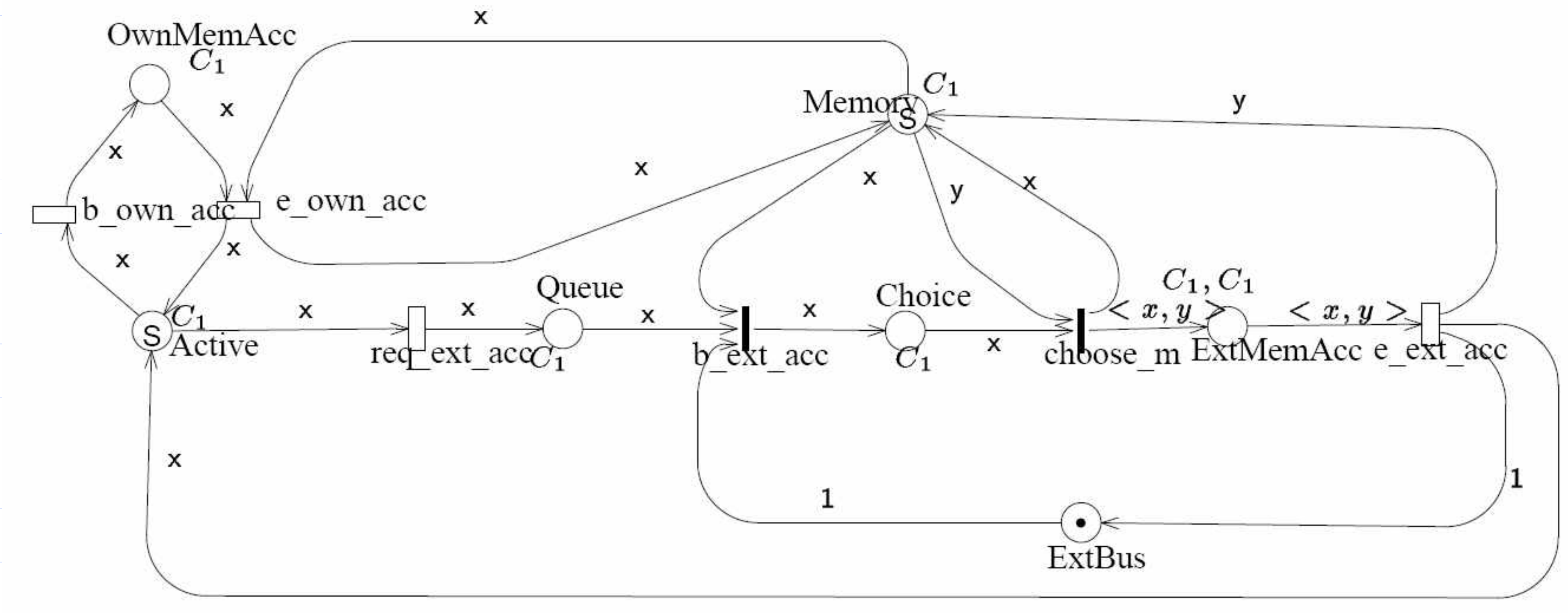
A general linear programming statement

- ❑ Example: a shared-memory multiprocessor
  - ❑ set of processing modules (with local memory) interconnected by a common bus called the "external bus"
  - ❑ a processor can access its own memory module directly from its private bus through one port, or it can access non-local shared-memory modules by means of the external bus
  - ❑ priority is given to external access through the external bus with respect to the accesses from the local processor



A general linear programming statement

- Timed Well-Formed Coloured Net (TWN) model of the shared-memory multiprocessor



Average service time of timed transitions equal to 0.5

A general linear programming statement

- ❑ The linear constraints for the LPP

$$
\begin{aligned}
&\overline{\boldsymbol{\mu}}[Active] = 4 + \boldsymbol{\sigma}[e\_e\_a] + \boldsymbol{\sigma}[e\_o\_a] - \boldsymbol{\sigma}[r\_e\_a] - \boldsymbol{\sigma}[b\_o\_a];\\
&\overline{\boldsymbol{\mu}}[Memory] = 4 + \boldsymbol{\sigma}[e\_e\_a] - \boldsymbol{\sigma}[b\_e\_a];\\
&\overline{\boldsymbol{\mu}}[OwnMemAcc] = \boldsymbol{\sigma}[b\_o\_a] - \boldsymbol{\sigma}[e\_o\_a];\\
&\overline{\boldsymbol{\mu}}[Queue] = \boldsymbol{\sigma}[r\_e\_a] - \boldsymbol{\sigma}[b\_e\_a];\\
&\overline{\boldsymbol{\mu}}[Choice] = \boldsymbol{\sigma}[b\_e\_a] - \boldsymbol{\sigma}[c\_m];\\
&\overline{\boldsymbol{\mu}}[ExtMemAcc] = \boldsymbol{\sigma}[c\_m] - \boldsymbol{\sigma}[e\_e\_a];\\
&\overline{\boldsymbol{\mu}}[ExtBus] = 1 + \boldsymbol{\sigma}[e\_e\_a] - \boldsymbol{\sigma}[b\_e\_a];\\
&\boldsymbol{\chi}[e\_e\_a] + \boldsymbol{\chi}[e\_o\_a] = \boldsymbol{\chi}[r\_e\_a] + \boldsymbol{\chi}[b\_o\_a];\\
&\boldsymbol{\chi}[b\_e\_a] = \boldsymbol{\chi}[c\_m] = \boldsymbol{\chi}[e\_e\_a] = \boldsymbol{\chi}[r\_e\_a];\\
&\boldsymbol{\chi}[b\_o\_a] = \boldsymbol{\chi}[r\_e\_a];\\
&\boldsymbol{\chi}[b\_o\_a]\ \overline{\mathrm{s}}[b\_o\_a] = \overline{\boldsymbol{\mu}}[Active]/2;\\
&\boldsymbol{\chi}[r\_e\_a]\ \overline{\mathrm{s}}[r\_e\_a] = \overline{\boldsymbol{\mu}}[Active]/2;\\
&\boldsymbol{\chi}[e\_e\_a]\ \overline{\mathrm{s}}[e\_e\_a] = \overline{\boldsymbol{\mu}}[ExtMemAcc];\\
&\boldsymbol{\chi}[e\_o\_a]\ \overline{\mathrm{s}}[e\_o\_a] \leq \overline{\boldsymbol{\mu}}[OwnMemAcc];\\
&\boldsymbol{\chi}[e\_o\_a]\ \overline{\mathrm{s}}[e\_o\_a] \leq \overline{\boldsymbol{\mu}}[Memory];\\
&\boldsymbol{\chi}[e\_o\_a]\ \overline{\mathrm{s}}[e\_o\_a] \geq \overline{\boldsymbol{\mu}}[OwnMemAcc] + \frac{\mathrm{b}[OwnMemAcc]}{\mathrm{b}[Memory]}\overline{\boldsymbol{\mu}}[Memory]\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad -\mathrm{b}[Memory];\\
&4\left(\overline{\boldsymbol{\mu}}[ExtBus] - \mathrm{b}[ExtBus]\left(1 - \frac{\overline{\boldsymbol{\mu}}[Memory]}{\mathrm{b}[Memory]}\right)\right) \leq 0;\\
&4\left(\overline{\boldsymbol{\mu}}[ExtBus] - \mathrm{b}[ExtBus]\left(1 - \frac{\overline{\boldsymbol{\mu}}[Queue]}{\mathrm{b}[Queue]}\right)\right) \leq 0;
\end{aligned}
$$

A general linear programming statement

- ❑ The “automatic” results:

$$\frac{8}{11} \leq \chi[e\_e\_a] \leq 2$$

The exact solution with exponential distribution would be

$$\chi[e\_e\_a] = 1.71999$$

Improving of lower bound with more “ad hoc” constraints:

$$\overline{\mu}[Choice] = 0;\ \mathrm{b}[Choice] = 0;\ \mathrm{b}[Queue] = 3$$

$$4\left(\overline{\boldsymbol{\mu}}[ExtBus] + \frac{\mathrm{b}[ExtBus]}{\mathrm{b}[Queue]}\overline{\boldsymbol{\mu}}[Queue] - \mathrm{b}[ExtBus]\right) \leq 0$$

The improved bound:

$$1 \leq \chi[e\_e\_a] \leq 2$$

Outline

- ❑ Preliminary comments
- ❑ Introducing the ideas: Marked Graphs case
- ❑ Generalization: use of visit ratios
- ❑ Improvements of the bounds
- ❑ A general linear programming statement
- ❑ Bibliography

Bibliography

- J. Campos, G. Chiola, J. Colom, M. Silva: **Properties and Performance Bounds for Timed Marked Graphs.** *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications,* vol. 39, no. 5, pp. 386-401, May 1992. Download here.
- J. Campos, G. Chiola, M. Silva: **Ergodicity and Throughput Bounds of Petri Nets with Unique Consistent Firing Count Vector.** *IEEE Transactions on Software Engineering,* vol. 17, no. 2, pp. 117-125, February 1991. Download here.
- J. Campos, G. Chiola, M. Silva: **Properties and Performance Bounds for Closed Free Choice Synchronized Monoclass Queueing Networks.** *IEEE Transactions on Automatic Control,* vol. 36, no. 12, pp. 1368-1382, December 1991. Download here.
- J. Campos, M. Silva: **Structural Techniques and Performance Bounds of Stochastic Petri Net Models.** *Lecture Notes in Computer Science, Advances in Petri Nets 1992,* G. Rozenberg (ed.), vol. 609, pp. 352-391, Berlin, Springer-Verlag, 1992. Download here.
- G. Chiola, C. Anglano, J. Campos, J. Colom, M. Silva: **Operational Analysis of Timed Petri Nets and Application to the Computation of Performance Bounds.** *Proceedings of the 5th International Workshop on Petri Nets and Performance Models,* pp. 128-137, Toulouse, France, IEEE-Computer Society Press, October 1993. Download here.
- J. Campos: **Performance Bounds**. In *Performance Models for Discrete Event Systems with Synchronizations: Formalisms and Analysis Techniques,* G. Balbo & M. Silva (ed.), Chapter 17, pp. 587-635, Zaragoza, Spain, Editorial KRONOS, September 1998. Download here.

Modelling and analysis of concurrent systems with Petri nets. Performance evaluation

6.2. Structure based performance analysis techniques: Approximations

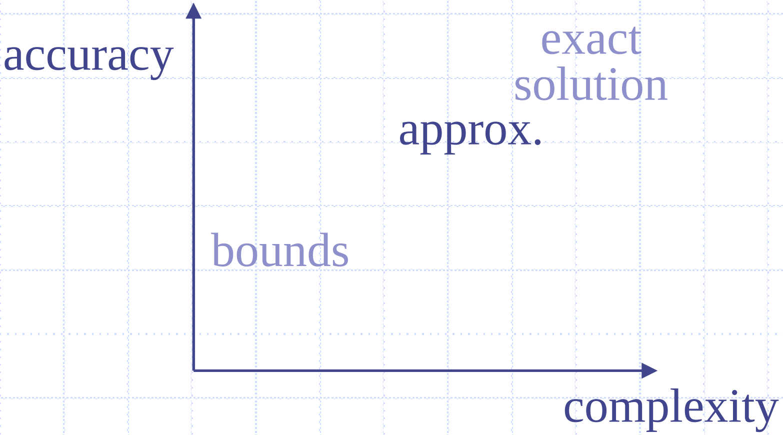
Javier Campos
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, Spain
jcampos@unizar.es

Outline

- ❑ Decomposition of models
- ❑ Flow equivalent aggregation
- ❑ Iterative algorithm: marked graphs case
- ❑ Iterative algorithm: general case
- ❑ Bibliography

Decomposition of models

- ❑ Interest of approximation techniques



Decomposition of models

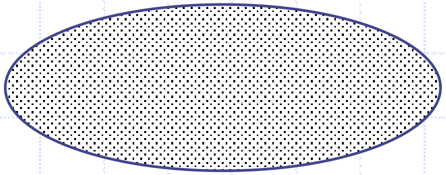
- **Basic idea:**

  reduce the complexity of the analysis of a complex system

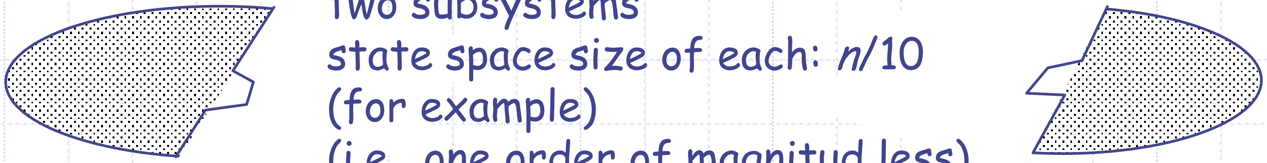
- **when**
  - the system is too complex/big to be solved by any exact analytical technique
  - a simulation is too long (essentially if many different configurations must be tested or it must be included in some optimization procedure)
  - some insights about the internal behaviour of subsystems are wanted (writing equations might help)

Decomposition of models

- ❑ Principle:
  - ❑ decompose the system into some subsystems



original system
state space size: n



two subsystems
state space size of each: $n/10$
(for example)
(i.e., one order of magnitud less)

  - ❑ reduce the analysis of the whole system by those of the subsystems in isolation

    if the solution technique was, e.g., $\mathcal{O}(n^3)$ on the state space size n, the cost of solving the isolated subsystems would be $\mathcal{O}(n^3/1000)$, i.e. three orders of magnitud less…

Decomposition of models

- ❑ Advantages:
  - ❑ drastical reduction of complexity and computational requirements
  - ❑ enables to extend the class of system that can be solved by analytical techniques
- ❑ Problems and limitations
  - ❑ Decomposition is not easy!
    - ❑ "net-driven" means to use structural information of the net model to assure that "good" qualitative properties are preserved in the isolated subsystems (e.g., liveness, boundedness…)
  - ❑ Approximation is not exact!
    - ❑ problem of error estimation or at least bounding the error
  - ❑ Accurate techniques are usually very especific to particular problems ➔ need of expertise to select the adequate technique…

Decomposition of models

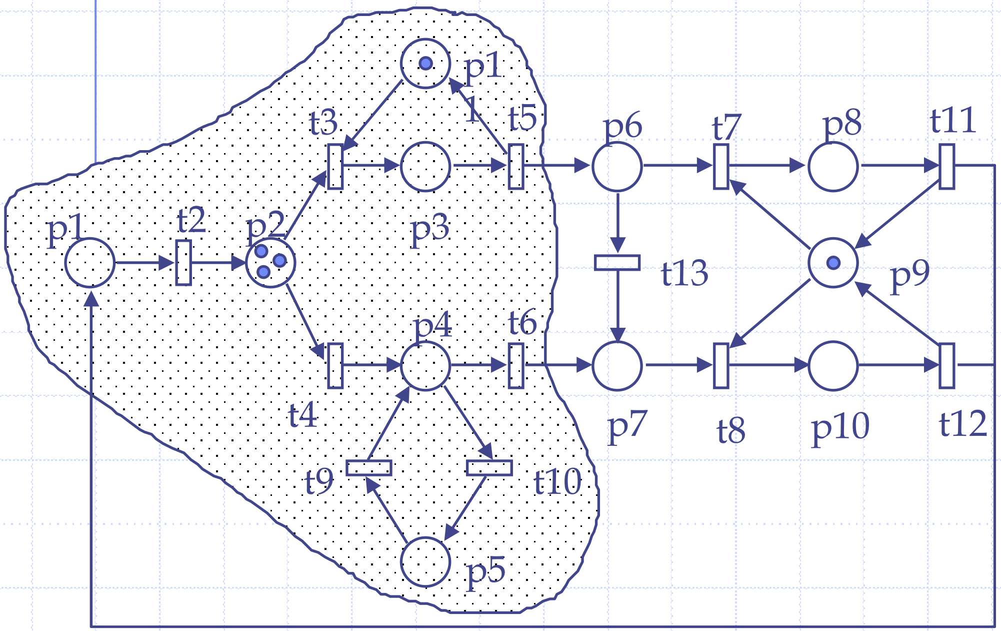
- **Steps** in an approximation technique based on decomposition:
  - Partition of the system into subsystems:
    - definition of rules for decomposition
    - consideration of functional properties that must/can be preserved
  - Characterization of subsystems in isolation:
    - definition of unknowns and variables
    - decisions related with consideration of mean variables or higher order moments of involved random variables
    - consideration or not of the "outside world"
    - need of a skeleton (high level view of the model) and characteristics considered in it
  - Estimation of the unknown parameters:
    - writing equations among unknowns
    - direct or iterative technique (in this case, definition of fixed point equations)
    - considerations on existence and uniqueness of solution
    - computational algorithm for solving the fixed point equation (implementation aspects, convergence aspects)

Outline

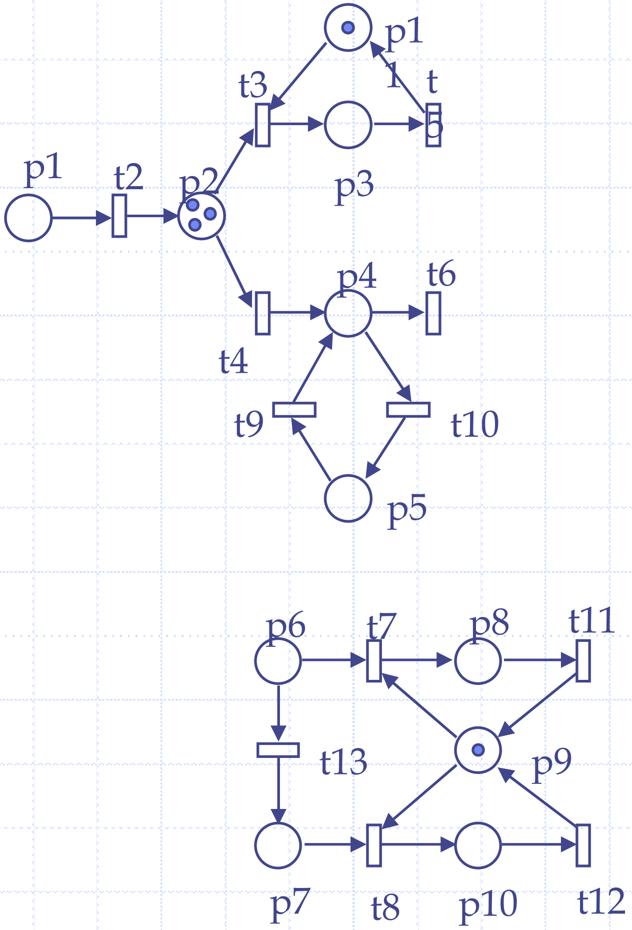
- □ Decomposition of models
- □ Flow equivalent aggregation
- □ Iterative algorithm: marked graphs case
- □ Iterative algorithm: general case
- □ Bibliography

Flow equivalent aggregation

- The system:



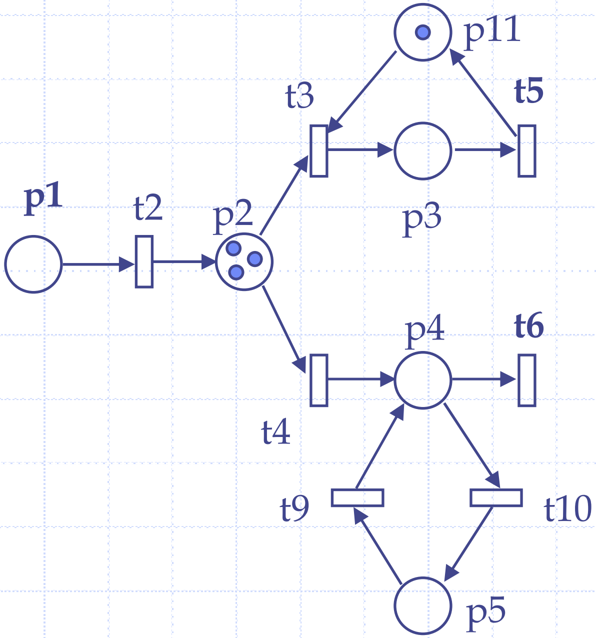
- Partition:



Flow equivalent aggregation

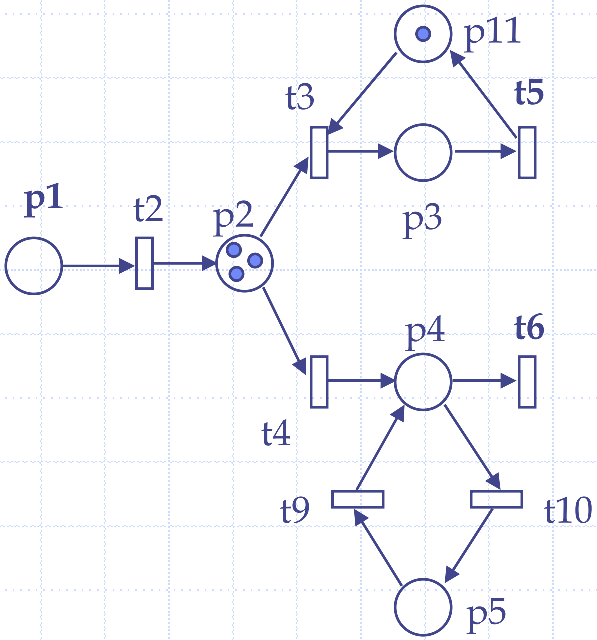
- ❑ Characterization of subsystems. Behaviour is characterized by:
  - ❑ path a token takes in the PN (what percetage leave through t5 and t6)
  - ❑ time it takes a token to be discharged

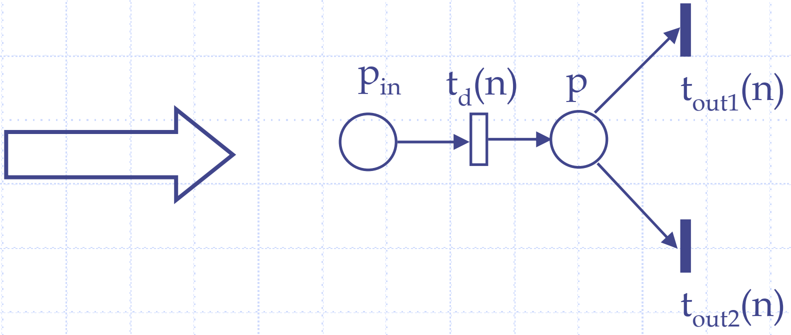
•way-in places:	p1
•sink transitions:	t5, t6



Flow equivalent aggregation

- Reduction of the subsystem:



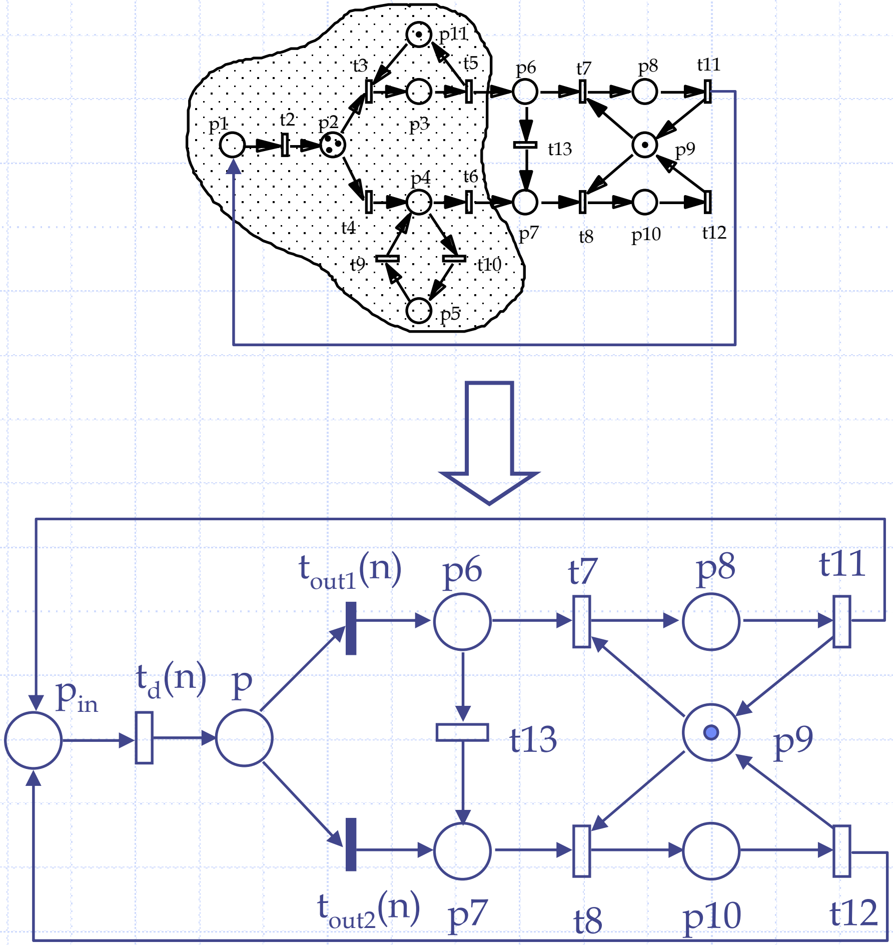


- routing rates of $t_{out1}(n)$ and $t_{out2}(n)$?
- service rate of $t_d(n)$?

(marking dependent: $n=M(p_{in})$)

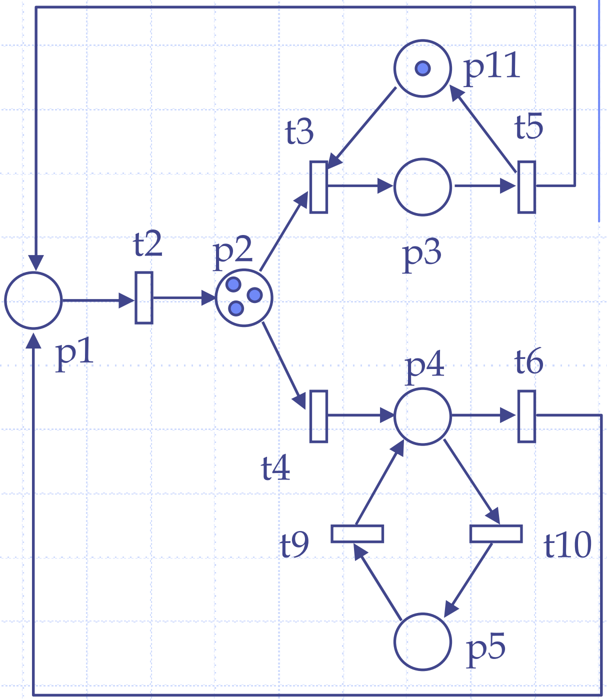
Flow equivalent aggregation

- ❑ Aggregated system:



Flow equivalent aggregation

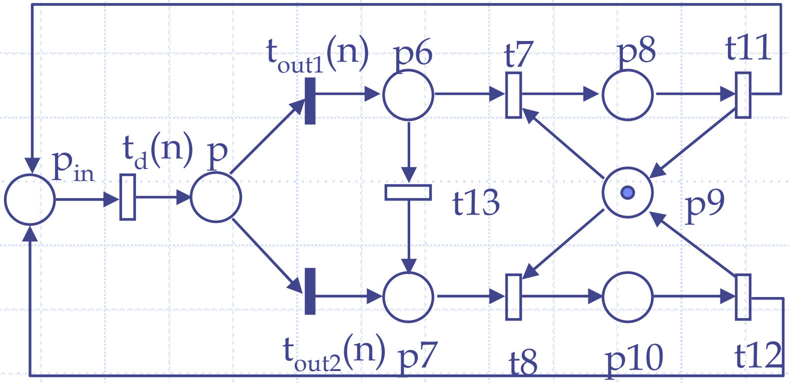
- ❑ Estimation of the unknown parameters:
  - ❑ Analyze the subnet in isolation with constant number of tokens
    - ❑ delay and routing are dependent on the number of tokens in the system
    - ❑ compute delay and routing for all possible populations



Parameters of the subsystem in isolation			
# tokens	v5	v6	thrput
1	0.500	0.500	0.400
2	0.431	0.569	0.640
3	0.403	0.597	0.780
4	0.389	0.611	0.863
5	0.382	0.618	0.914

Flow equivalent aggregation

- When the subnet is substituted back, routing and delay are going to be state dependent ($n=M(p_{in})$)

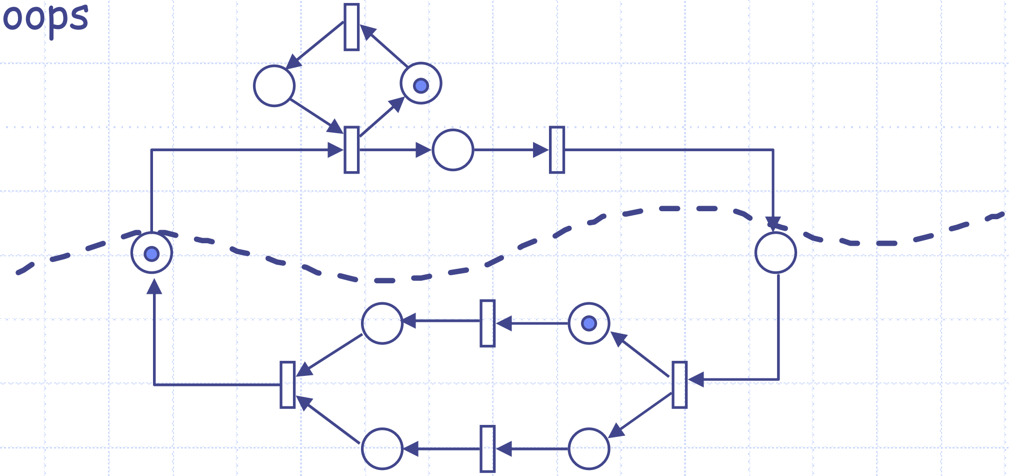


Comparison of State Spaces & throughput

#tokens	# states		throughput		%error
	aggregat	original	aggregat	original	
1	5	9	0.232	0.232	0.00
2	12	41	0.381	0.384	0.78
3	22	131	0.470	0.474	0.84
4	35	336	0.521	0.523	0.38
5	51	742	0.548	0.547	<0.10

Flow equivalent aggregation

- Limitations:
  - **Assumption**: the service time depends only on the number of customers which are currently present in the subsystem.
    - The behaviour of the subsystem is assumed independent of the arrival process
  - It is exact for product-form queueing networks.
  - The error is small if in the original model:
    - the arrivals to the subsystem are "close" to Poisson arrivals and
    - the processing times are approximately exponential
  - On the other hand, the error can be very large if
    - there exist internal loops in a subnet, or
    - there exist trapped tokens in a fork-join, or…



Outline

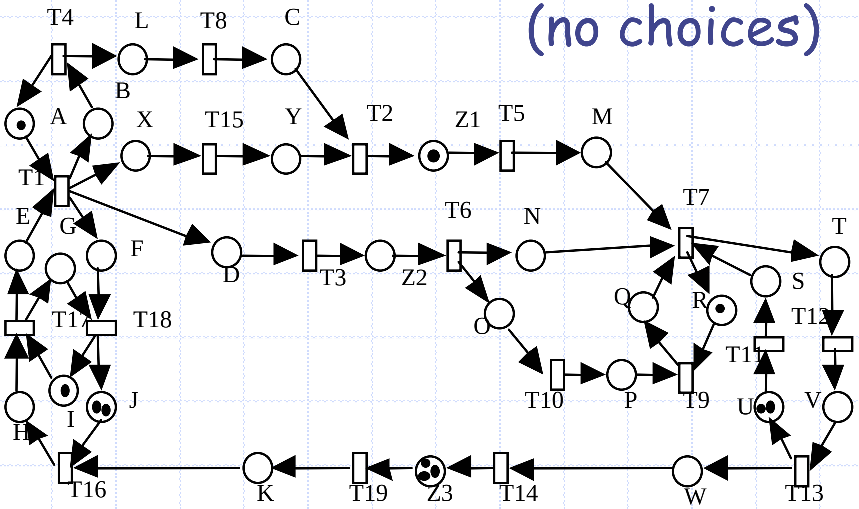
- ❑ Decomposition of models
- ❑ Flow equivalent aggregation
- ❑ Iterative algorithm: marked graphs case
- ❑ Iterative algorithm: general case
- ❑ Bibliography

Iterative algorithm: marked graphs case

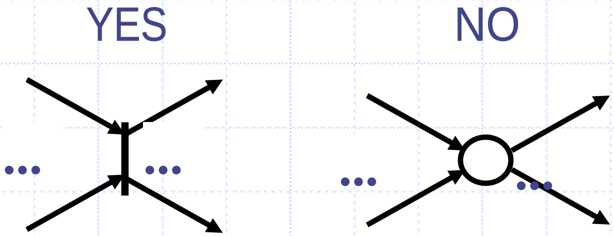
- ❑ Net-driven solution techniques
  - ❑ stressing the intimate relationship between qualitative and quantitative aspects of PN's
  - ❑ structure theory of net models

    → efficient computation techniques

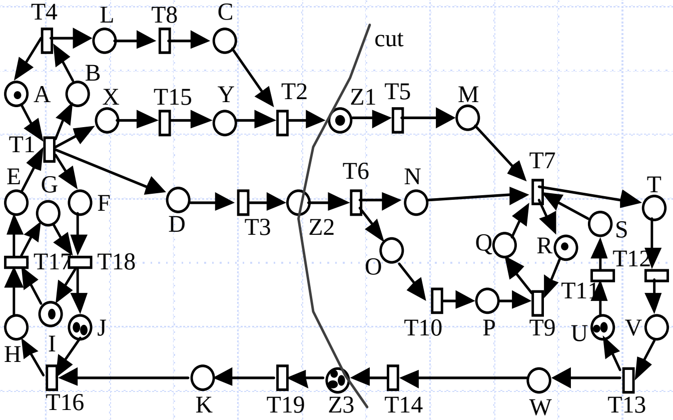
- ❑ Marked graphs: subclass of *ordinary* nets



(no choices) (no weights)

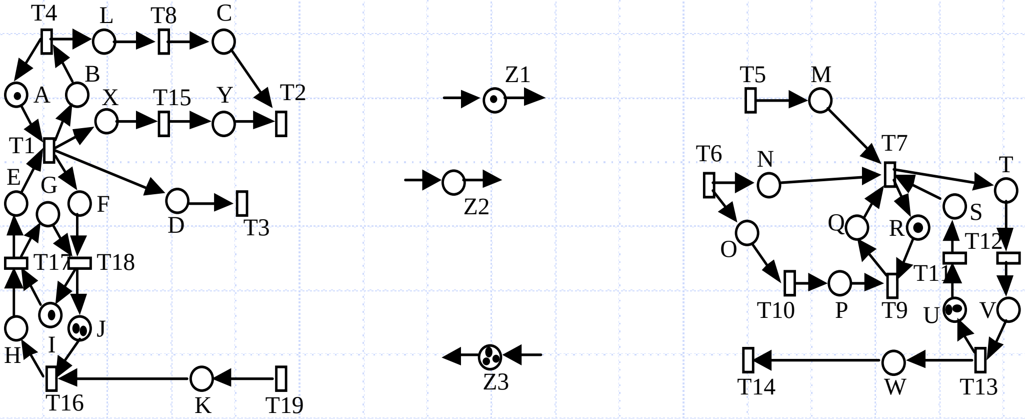


Iterative algorithm: marked graphs case

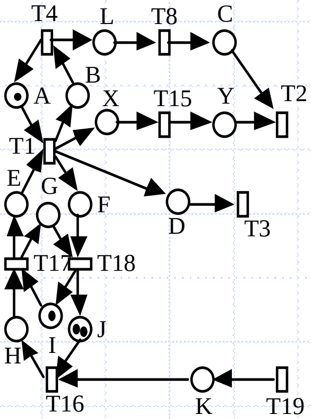


original model + definition of cut

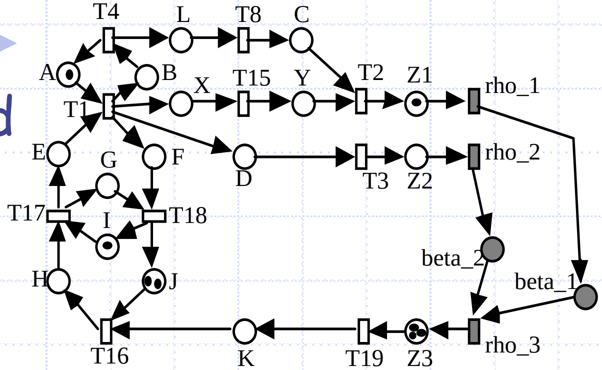
partition of the model into **modules** (subnets) connected through **buffers** (places)



Iterative algorithm: marked graphs case

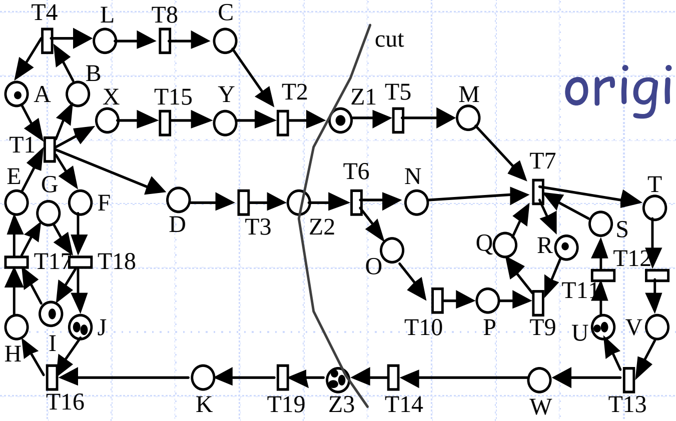


the solution of isolated modules
is difficult and useless:
(in this case) they are unbounded!

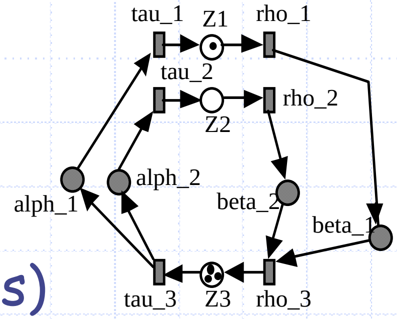


the modules must be complemented
with an abstract view of the rest;
components are obtained

Iterative algorithm: marked graphs case

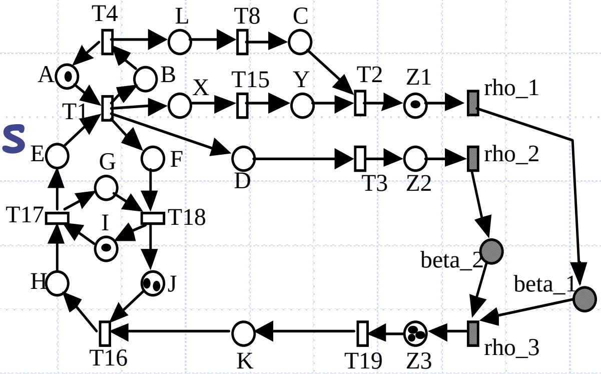


original model (89358 states)

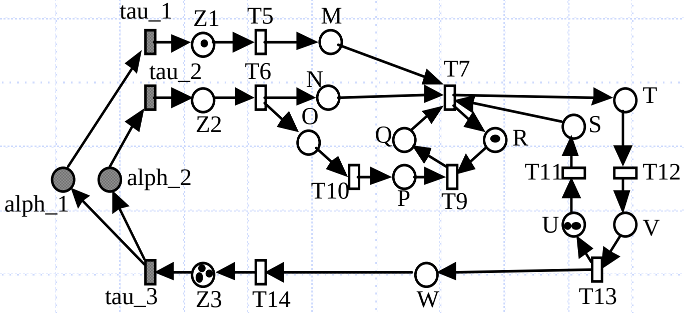


BS (231 states)

AS_1 (8288 states)



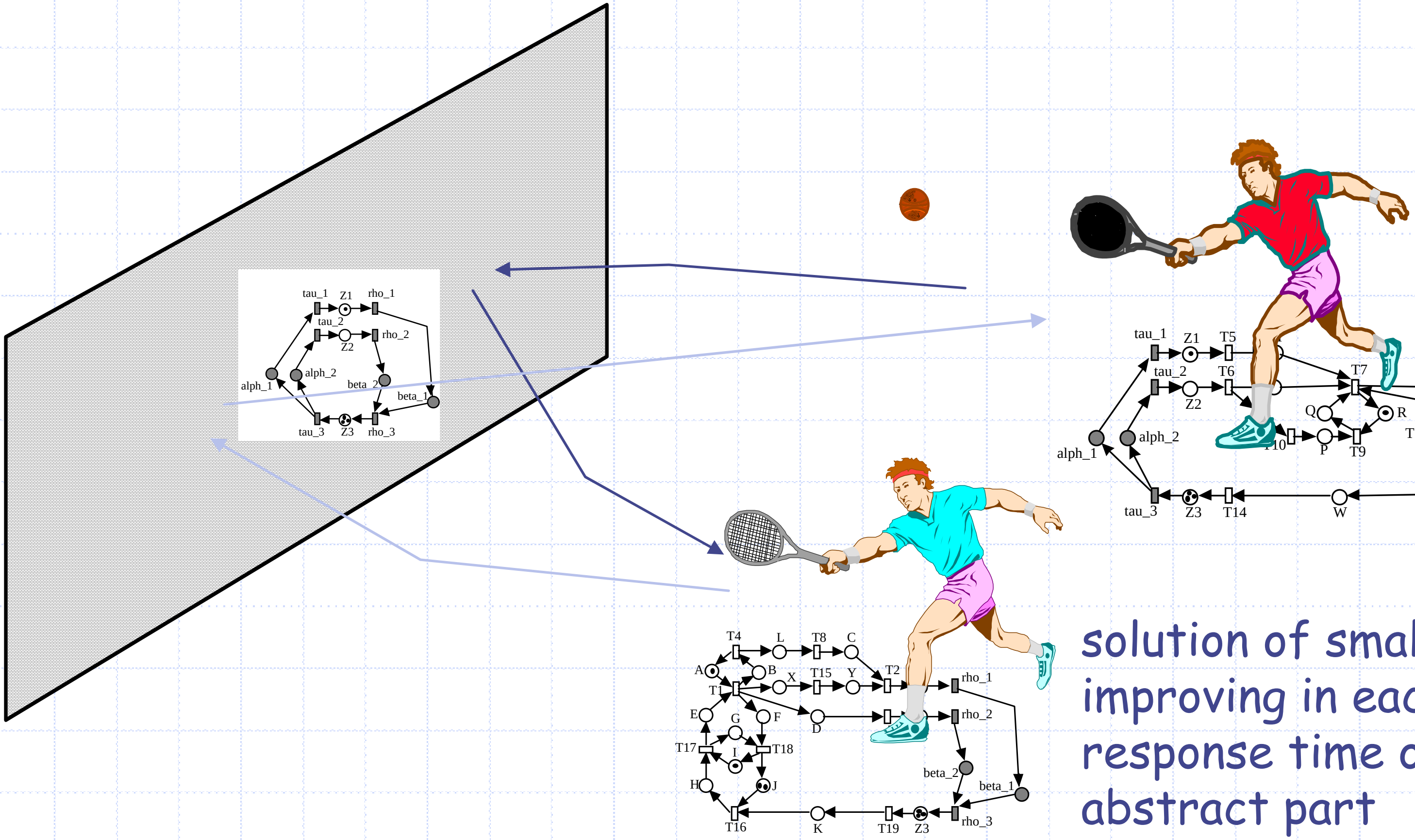
AS_2 (3440 states)



three components:
aggregated systems
(low level views)
and **basic skeleton**
(high level view)

Iterative algorithm: marked graphs case

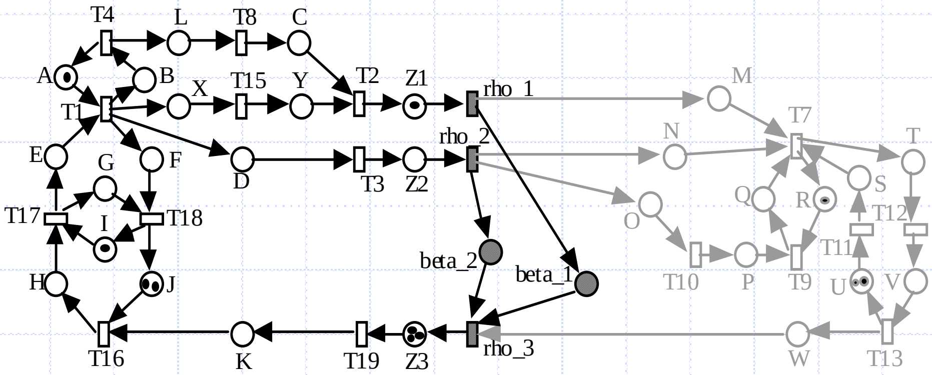
iterative solution: *pelota* algorithm (response time approximation technique)



solution of smaller CTMC's, improving in each step the response time of the abstract part

Iterative algorithm: marked graphs case

- Substitute a subnet by a set of places



  - interface transitions (input/ouput of buffers) are preserved
  - add one place from each input to each output transition
  - the set of new places can be superposed in the original model preserving the behaviour: **implicit places**

Iterative algorithm: marked graphs case

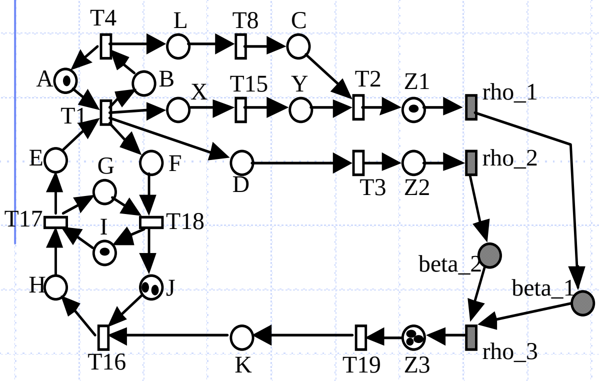
- Compute the initial marking of new places
  - minimum initial marking to make them implicit
  - computed using Floyd's *all-pairs shortest paths algorithm*:
    - the MG is considered as a weighted graph (transitions are vertices and the initial marking of places are the weigths of the arcs)

Iterative algorithm: marked graphs case

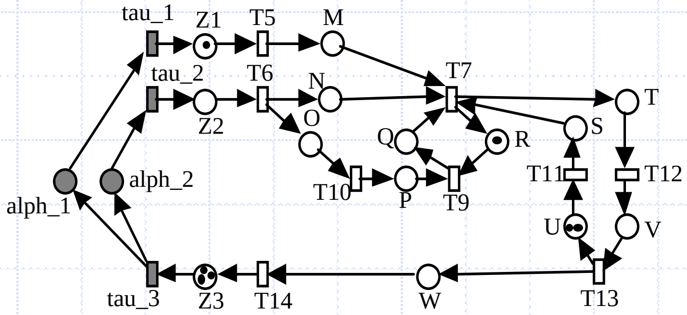
- The abstract view has "very good quality":
  - the language of firing sequences of the aggregated system is equal to that of the original system projected on the preserved transitions
  - the reachability graph of the aggregated system is isomorphous to that of the original system projected on the preserved places

Iterative algorithm: marked graphs case

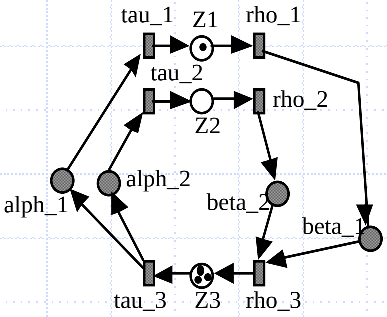
❑ Definition of unknowns:



service time of *rho_i*



service time of *tau_j*



service time of *rho_i* and *tau_j*

+ throughput of each system
+ response time of interface transitions at each system

Iterative algorithm: marked graphs case

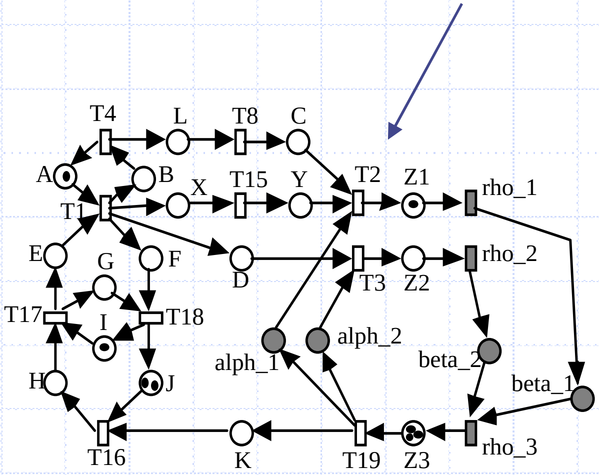
response time approximation of the **left hand subnet** for a token that

exits through T2: $R_2 = \overline{\mu}[alph\_1] \big/ \chi[t_2]$
(Little's law)

exits through T3: $R_3 = \overline{\mu}[alph\_2] \big/ \chi[t_3]$

$(\chi[t_2] = \chi[t_3] = \chi)$

first aggregated system



thus, solve the CTMC and compute: R_2, R_3 and also χ

Iterative algorithm: marked graphs case

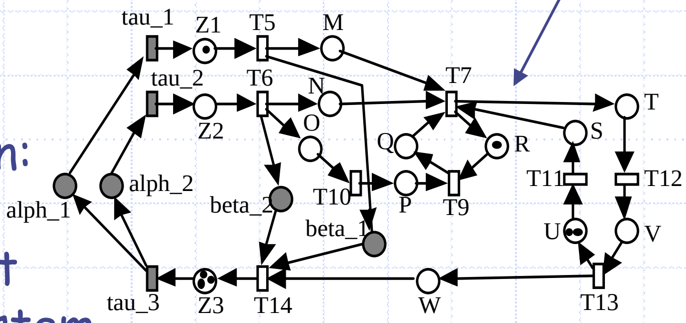
select *tau_1* and *tau_2* as:

$$tau\_1 = f.R_2$$

$$tau\_2 = f.R_3$$

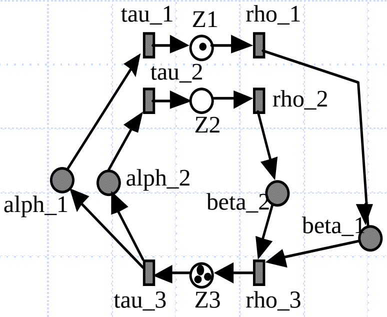
where *f* is computed using the skeleton: linear search until the throughput of the skeleton is equal to the throughput computed for the first aggregated system

second aggregated system



$$tau\_1 = f.R_2$$

$$tau\_2 = f.R_3$$



skeleton

Iterative algorithm: marked graphs case

The algorithm:

```
select a cut Q;
derive aggregated systems $AS_1$,$AS_2$ and skeleton BS;
give initial value $\mu_t^{(0)}$ for each $t \in T_{I2}$;
k:=0;  {counter for iteration steps}
repeat
  k:=k+1;
  solve aggregated system $AS_1$ with
    input:  $\mu_t^{(k-1)}$ for each $t \in T_{I2}$,
    output: ratios among $\mu_t^{(k)}$ of $t \in T_{I1}$, and  $X_1^{(k)}$;
  solve basic skeleton BS with
    input:  $\mu_t^{(k-1)}$ for each $t \in T_{I2}$,
            ratios among $\mu_t^{(k)}$ of $t \in T_{I1}$, and $X_1^{(k)}$,
    output: scale factor of $\mu_t^{(k)}$ of $t \in T_{I1}$;
  solve aggregated system $AS_2$ with
    input:  $\mu_t^{(k-1)}$ for each $t \in T_{I1}$,
    output: ratios among $\mu_t^{(k)}$ of $t \in T_{I2}$, and $X_2^{(k)}$;
  solve basic skeleton BS with
    input:  $\mu_t^{(k)}$ for each $t \in T_{I1}$,
            ratios among $\mu_t^{(k)}$ of $t \in T_{I2}$, and $X_2^{(k)}$,
    output: scale factor of $\mu_t^{(k)}$ of $t \in T_{I2}$;
until convergence of $X_1^{(k)}$ and $X_2^{(k)}$;
```

Iterative algorithm: marked graphs case

- On the (theoretical) convergence of the algorithm:
  - Theorem [D.R. Smart, *Fixed Point Theorems*, Cambridge Univ. Press, 1974]:
    $f: D \subset R^n \rightarrow R^n$ continuous in a compact, convex, non-empty D, $f(D) \subseteq D$ (i.e. contractive) $\Rightarrow \exists\ x \in D$ such that $f(x) = x$.
  - The previous algorithm can be written:

```
input: $\mu^{(0)}$ -- initial rates of interface transitions TI.
    $n := 0$   -- loop counter
    repeat
            $n := n+1$
            $\mu^{(n)} := G(\mu^{(n-1)})$
    until convergence of $\mu^{(n)}$
output: $X(\mu^{(n)})$  -- vector of approximated throughput
```

  - Theorem: for a live strongly connected MG, function G in the algorithm is continuous and there exists a compact, convex, non-empty set S such that $G(S) \subseteq S$.
  - Corollary: there exists x such that $G(x) = x$.

Iterative algorithm: marked graphs case

On the practical convergence:

Service rates (arbitrary):
T2=0.2; T4=0.7; T6=0.3; T8=0.8; T9=0.6; T10=0.5;
Ti=1.0,   i=1,3,5,7,11,12,13,14,15,16,17,18,19

Throughput of the original system: 0.138341
State space of the original system: 89358

Results using the approximation technique:
State space AS1: 8288; State space AS2: 3440;
State space BS: 231

AS1				AS2			
X1	tau_1	tau_2	tau_3	X2	rho_1	rho_2	rho_3
0.17352	0.05170	0.16810	0.88873	0.12714	0.89026	0.21861	0.14354
0.14093	0.06265	0.19707	0.91895	0.13795	0.88267	0.21363	0.13509
0.13856	0.06325	0.19821	0.92054	0.13841	0.88239	0.21343	0.13467
0.13844	0.06328	0.19827	0.92062	0.13843	0.88237	0.21342	0.13465
0.13843	0.06328	0.19827	0.92064	0.13843	0.88238	0.21342	0.13465

Outline

- ❑ Decomposition of models
- ❑ Flow equivalent aggregation
- ❑ Iterative algorithm: marked graphs case
- ❑ Iterative algorithm: general case
- ❑ Bibliography

Iterative algorithm: general case

- The story was:
  - Marked graphs case
  - Weighted *T*-systems
    - Non-trivial extension!
      - Definition of new structure concepts (gain, weighted marking, resistance)
      - More complex aggregated subsystems
      - Similar iterative algorithm
  - DSSP: deterministic systems of sequential processes
    - Decomposition problems, partial results…
  - General case: new decomposition approach

Iterative algorithm: general case

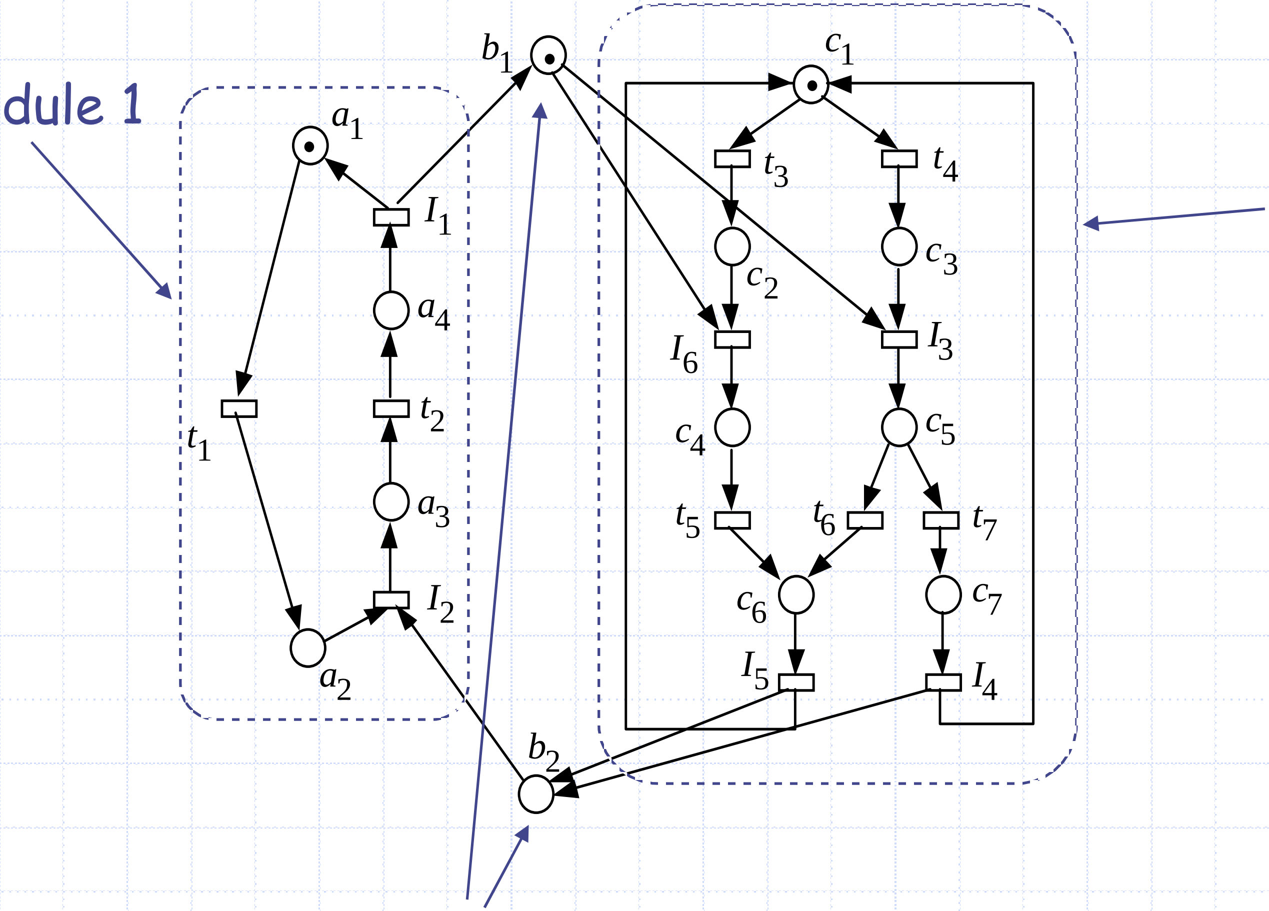
- Arbitrary *P/T* system + **structured view**

partition into **modules** (functional units) connected through places (**buffers**)

Iterative algorithm: general case

- All *P/T* systems have serveral structured views, varying between:
  - a single module (empty set of buffers)
  - as many modules as transitions (all places are considered as buffers)

Iterative algorithm: general case



module 1

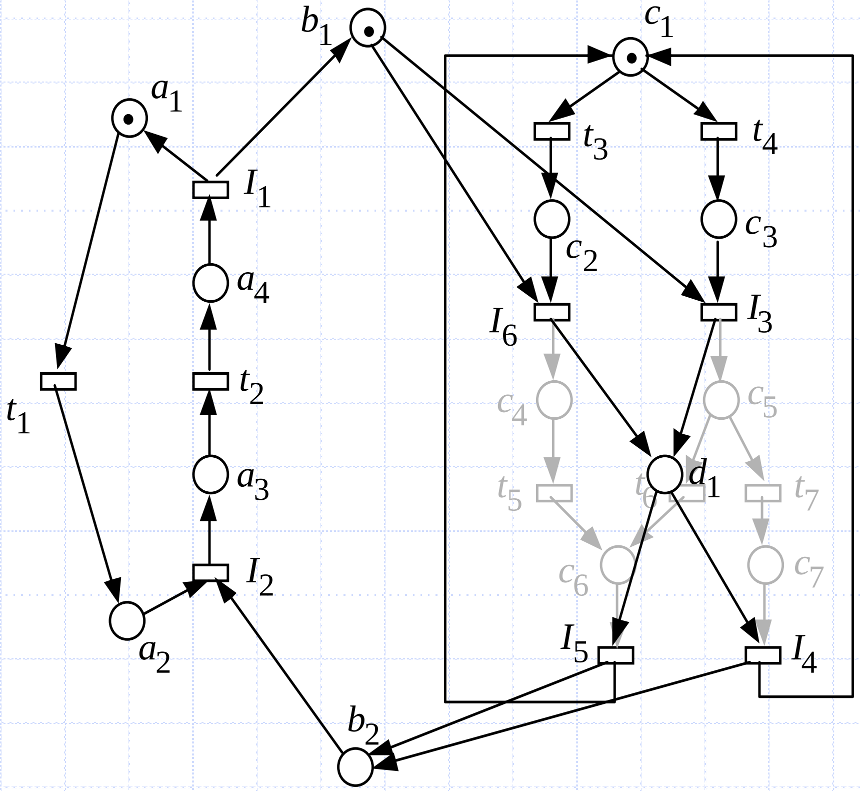
module 2

buffers

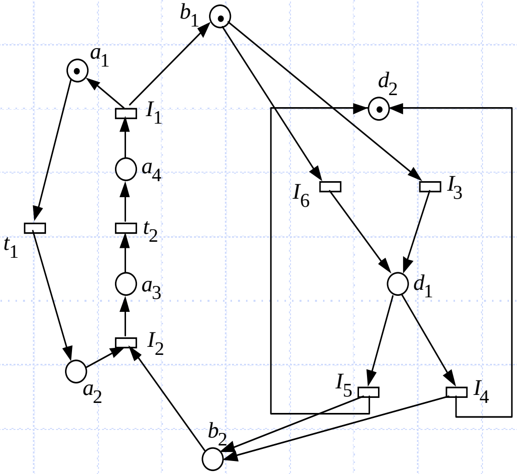
Iterative algorithm: general case

- Substitute a subnet by a set of implicit places

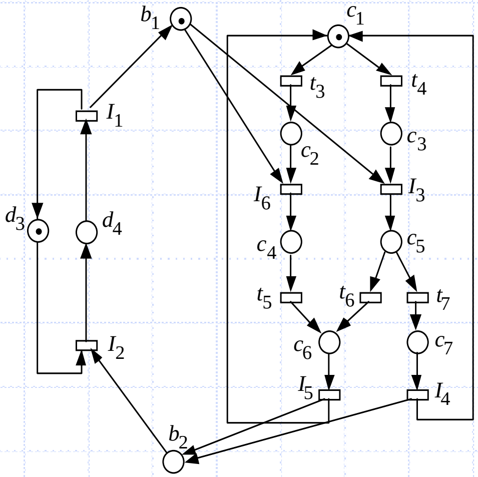
  derived from minimal *P*-semiflows of the subnet (sum of the incidence rows of places)



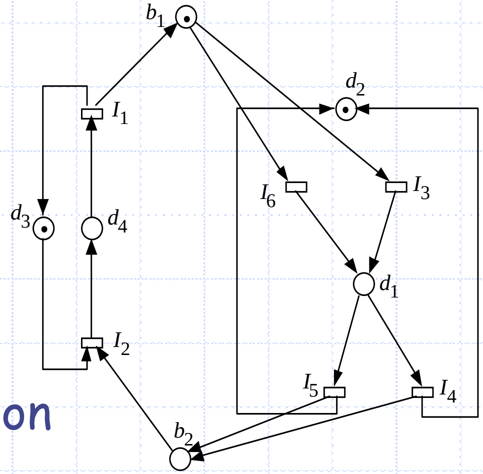
Iterative algorithm: general case



first aggregated system



second aggregated system



skeleton

Iterative algorithm: general case

- The quality of the abstract view is "not as good as" in the MG's case
  - the language of firing sequences of the aggregated system **includes** that of the original system projected on the preserved transitions
  - the reachability graph of the aggregated system **includes** that of the original system projected on the preserved nodes

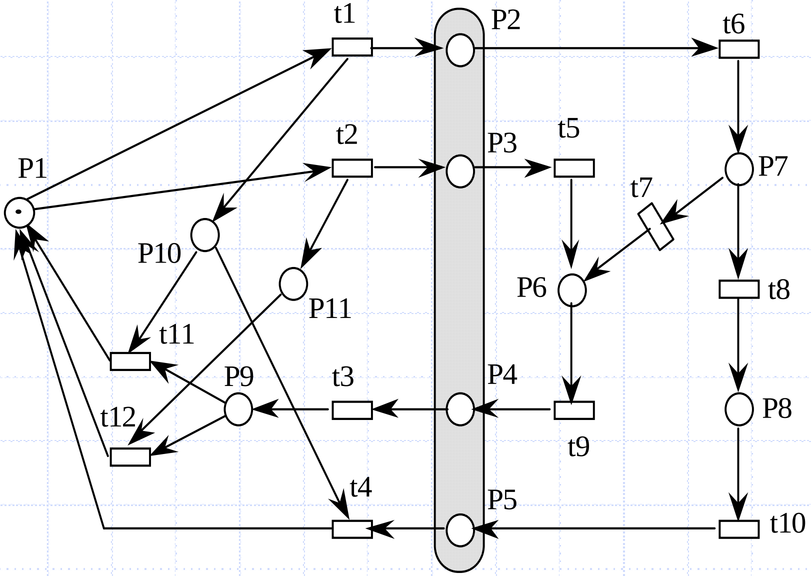
Iterative algorithm: general case

- Problems in the composition:

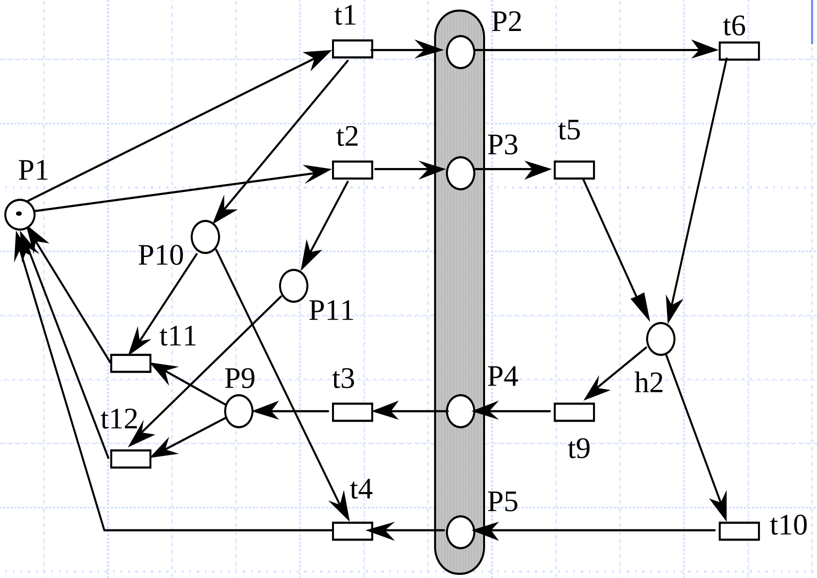
The RG of an aggregated system may include *spurious* markings and firing sequences that do not correspond to actual markings and firing sequences of the original system

we can obtain even **non-ergodic** systems (CTMC cannot be solved)

Iterative algorithm: general case



original system:
limited and reversible, thus ergodic

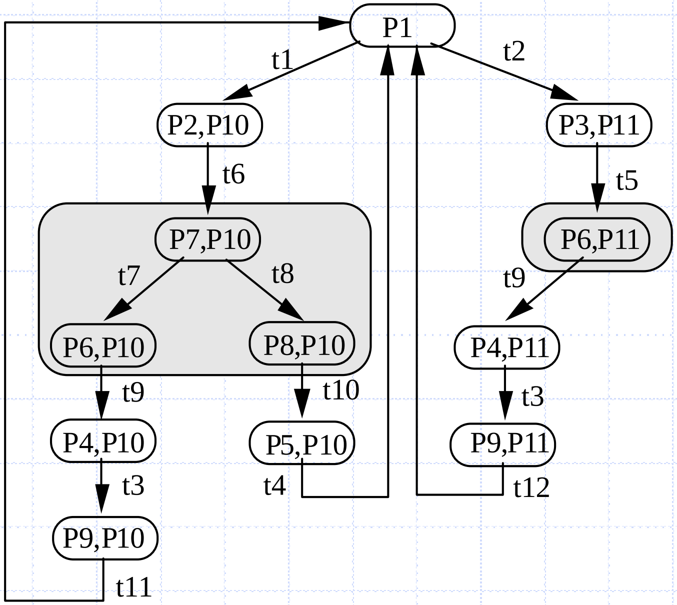


aggregated system:
it has a total deadlock

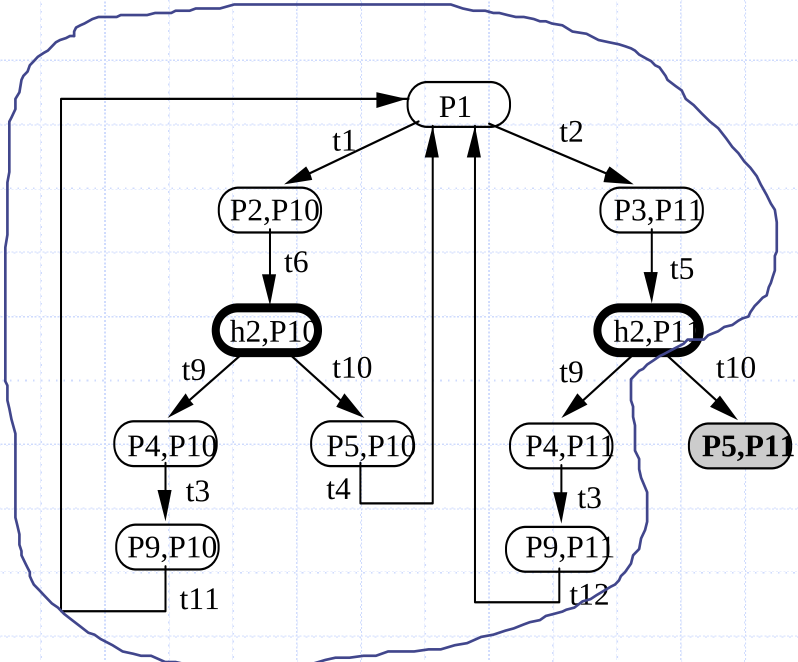
Iterative algorithm: general case

- Solution for the problem:

  select only the strongly connected component of the RG that includes the projection of the initial marking



RG of the original system



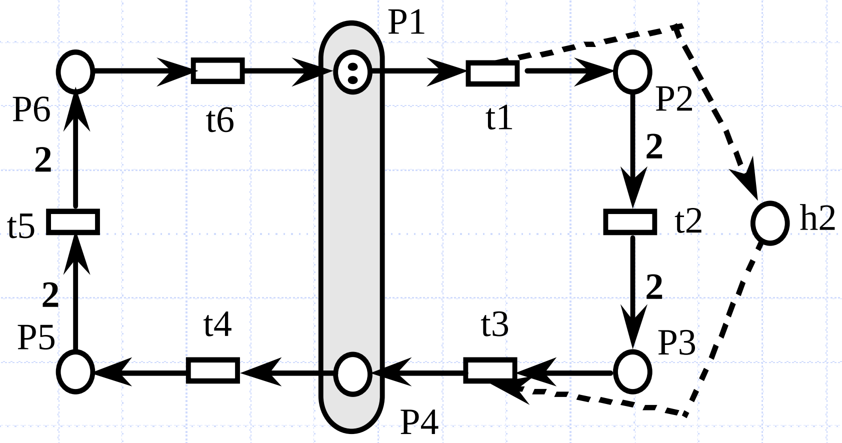
RG of the aggregated system

Iterative algorithm: general case

- ❑ More problems:

  Spurious markings (and/or firing seq.) may still be present,

  but the solution is possible!



Iterative algorithm: general case

- It is possible to eliminate all the spurious markings with additional computational effort
  - use a *Kronecker* expression of the infinitesimal generator of the original system
    - implement a depth-first search to build the RS
    - reduce the infinitesimal generators of the aggregated systems, using the information about reachability in the original system

- The whole reachability set must be derived but the CTMC is not solved (throughput is approximated from the solution of CTMC of subsystems)

Outline

- □ Decomposition of models
- □ Flow equivalent aggregation
- □ Iterative algorithm: marked graphs case
- □ Iterative algorithm: general case
- □ Bibliography

Bibliography

- J. Campos, J. Colom, H. Jungnitz, M. Silva: **Approximate Throughput Computation of Stochastic Marked Graphs**. *IEEE Transactions on Software Engineering*, vol. 20, no. 7, pp. 526-535, July 1994.
  Download here.
- C. Pérez-Jiménez, J. Campos, M. Silva: **Approximate Throughput Computation of Stochastic Weighted T-Systems**. *IEEE Transactions on Systems, Man, and Cybernetics. Part A: Systems and Humans*, vol. 37, no. 3, pp. 431-444, May 2007.
  Download here.
- C. Pérez-Jiménez, J. Campos: **On State Space Decomposition for the Numerical Analysis of Stochastic Petri Nets**. *Proceedings of the 8th International Workshop on Petri Nets and Performance Models*, pp. 32-41, Zaragoza, Spain, IEEE Computer Society Press, September 1999.
  Download here.
- C. Pérez-Jiménez: **Técnicas de aproximación de *throughput* en redes de Petri estocásticas**. PhD Thesis. Dpto. Informática e Ingeniería de Sistemas, Universidad de Zaragoza. April 2002.
  Download here.

Modelling and analysis of concurrent systems with Petri nets. Performance evaluation

6.3. Structure based performance analysis techniques: Kronecker algebra-based exact solution

Javier Campos
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, Spain
jcampos@unizar.es

Outline

- Kronecker product and DTMC
- Kronecker sum and CTMC
- Structured view of stochastic Petri nets
- Reachability set construction
- CTMC generation and solution
- Bibliography

Kronecker product and DTMC

- Kronecker product

$$\text{Given } \mathbf{A} = \begin{bmatrix} \mathbf{a}_{0,0} & \mathbf{a}_{0,1} \\ \mathbf{a}_{1,0} & \mathbf{a}_{1,1} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{b}_{0,0} & \mathbf{b}_{0,1} & \mathbf{b}_{0,2} \\ \mathbf{b}_{1,0} & \mathbf{b}_{1,1} & \mathbf{b}_{1,2} \end{bmatrix},$$

$$\mathbf{C} = \mathbf{A} \otimes \mathbf{B} = \left[\begin{array}{c|c} \mathbf{a}_{0,0}\mathbf{B} & \mathbf{a}_{0,1}\mathbf{B} \\ \hline \mathbf{a}_{1,0}\mathbf{B} & \mathbf{a}_{1,1}\mathbf{B} \end{array}\right] =$$

$$\left[\begin{array}{ccc|ccc} \mathbf{a}_{0,0}\mathbf{b}_{0,0} & \mathbf{a}_{0,0}\mathbf{b}_{0,1} & \mathbf{a}_{0,0}\mathbf{b}_{0,2} & \mathbf{a}_{0,1}\mathbf{b}_{0,0} & \mathbf{a}_{0,1}\mathbf{b}_{0,1} & \mathbf{a}_{0,1}\mathbf{b}_{0,2} \\ \mathbf{a}_{0,0}\mathbf{b}_{1,0} & \mathbf{a}_{0,0}\mathbf{b}_{1,1} & \mathbf{a}_{0,0}\mathbf{b}_{1,2} & \mathbf{a}_{0,1}\mathbf{b}_{1,0} & \mathbf{a}_{0,1}\mathbf{b}_{1,1} & \mathbf{a}_{0,1}\mathbf{b}_{1,2} \\ \hline \mathbf{a}_{1,0}\mathbf{b}_{0,0} & \mathbf{a}_{1,0}\mathbf{b}_{0,1} & \mathbf{a}_{1,0}\mathbf{b}_{0,2} & \mathbf{a}_{1,1}\mathbf{b}_{0,0} & \mathbf{a}_{1,1}\mathbf{b}_{0,1} & \mathbf{a}_{1,1}\mathbf{b}_{0,2} \\ \mathbf{a}_{1,0}\mathbf{b}_{1,0} & \mathbf{a}_{1,0}\mathbf{b}_{1,1} & \mathbf{a}_{1,0}\mathbf{b}_{1,2} & \mathbf{a}_{1,1}\mathbf{b}_{1,0} & \mathbf{a}_{1,1}\mathbf{b}_{1,1} & \mathbf{a}_{1,1}\mathbf{b}_{1,2} \end{array}\right]$$

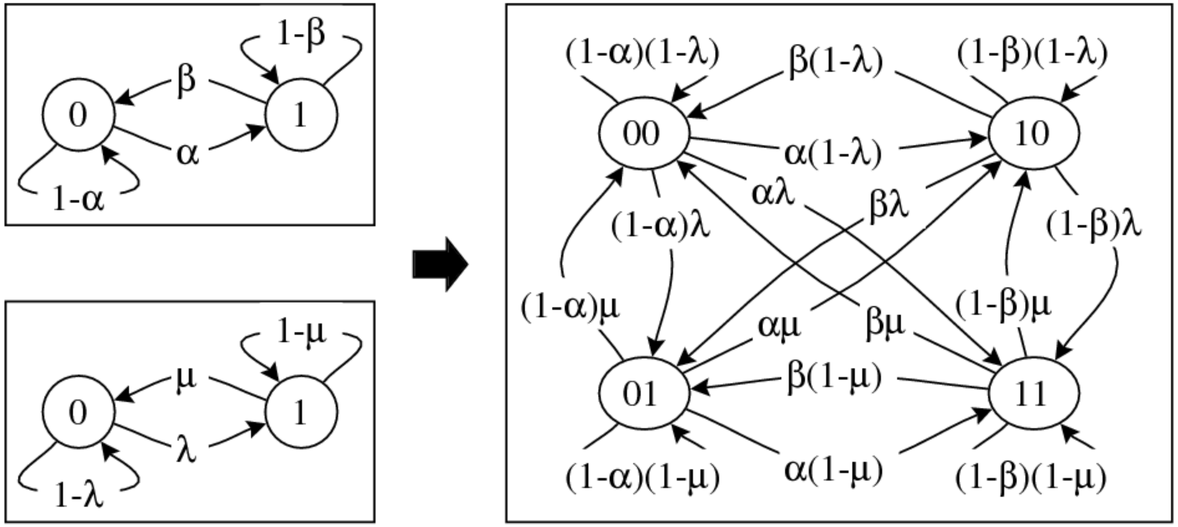
Kronecker product and DTMC

- If we merge two independent Discrete Time Markov Chains (DTMC) with state spaces $\mathcal{S}_1$ and $\mathcal{S}_2$ and transition probabilities $\mathbf{P}_1$ and $\mathbf{P}_2$, the resulting state space and transition probability matrix are:

$$\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \qquad \text{and} \qquad \mathbf{P} = \mathbf{P}_1 \otimes \mathbf{P}_2$$

Kronecker product and DTMC

- Example



$$\mathcal{S}^1 = \{\underline{0}, \underline{1}\} \qquad \mathcal{S}^2 = \{\underline{0}, \underline{1}\} \qquad \mathcal{S} = \{0 \equiv \underline{00},\ 1 \equiv \underline{01},\ 2 \equiv \underline{10},\ 3 \equiv \underline{11}\}$$

$$\mathbf{P}^1 = \begin{bmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{bmatrix} \qquad \mathbf{P} = \left[\begin{array}{cc|cc} (1-\alpha)(1-\lambda) & (1-\alpha)\lambda & \alpha(1-\lambda) & \alpha\lambda \\ (1-\alpha)\mu & (1-\alpha)(1-\mu) & \alpha\mu & \alpha(1-\mu) \\ \hline \beta(1-\lambda) & \beta\lambda & (1-\beta)(1-\lambda) & (1-\beta)\lambda \\ \beta\mu & \beta(1-\mu) & (1-\beta)\mu & (1-\beta)(1-\mu) \end{array}\right]$$

$$\mathbf{P}^2 = \begin{bmatrix} 1-\lambda & \lambda \\ \mu & 1-\mu \end{bmatrix}$$

Outline

- ❑ Kronecker product and DTMC
- ❑ Kronecker sum and CTMC
- ❑ Structured view of stochastic Petri nets
- ❑ Reachability set construction
- ❑ CTMC generation and solution
- ❑ Bibliography

Kronecker sum and CTMC

- ❑ Kronecker sum

$$\text{Given } \mathbf{A} = \begin{bmatrix} \mathbf{a}_{0,0} & \mathbf{a}_{0,1} \\ \mathbf{a}_{1,0} & \mathbf{a}_{1,1} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{b}_{0,0} & \mathbf{b}_{0,1} & \mathbf{b}_{0,2} \\ \mathbf{b}_{1,0} & \mathbf{b}_{1,1} & \mathbf{b}_{1,2} \\ \mathbf{b}_{2,0} & \mathbf{b}_{2,1} & \mathbf{b}_{2,2} \end{bmatrix},$$

$$\mathbf{C} = \mathbf{A} \oplus \mathbf{B} = \mathbf{A} \otimes \mathbf{I}_3 + \mathbf{I}_2 \otimes \mathbf{B} =$$

$$\left[\begin{array}{ccc|ccc} \mathbf{a}_{0,0} & & & \mathbf{a}_{0,1} & & \\ & \mathbf{a}_{0,0} & & & \mathbf{a}_{0,1} & \\ & & \mathbf{a}_{0,0} & & & \mathbf{a}_{0,1} \\ \hline \mathbf{a}_{1,0} & & & \mathbf{a}_{1,1} & & \\ & \mathbf{a}_{1,0} & & & \mathbf{a}_{1,1} & \\ & & \mathbf{a}_{1,0} & & & \mathbf{a}_{1,1} \end{array}\right] + \left[\begin{array}{ccc|ccc} \mathbf{b}_{0,0} & \mathbf{b}_{0,1} & \mathbf{b}_{0,2} & & & \\ \mathbf{b}_{1,0} & \mathbf{b}_{1,1} & \mathbf{b}_{1,2} & & & \\ \mathbf{b}_{2,0} & \mathbf{b}_{2,1} & \mathbf{b}_{2,2} & & & \\ \hline & & & \mathbf{b}_{0,0} & \mathbf{b}_{0,1} & \mathbf{b}_{0,2} \\ & & & \mathbf{b}_{1,0} & \mathbf{b}_{1,1} & \mathbf{b}_{1,2} \\ & & & \mathbf{b}_{2,0} & \mathbf{b}_{2,1} & \mathbf{b}_{2,2} \end{array}\right] =$$

$$\left[\begin{array}{ccc|ccc} \mathbf{a}_{0,0}+\mathbf{b}_{0,0} & \mathbf{b}_{0,1} & \mathbf{b}_{0,2} & \mathbf{a}_{0,1} & & \\ \mathbf{b}_{1,0} & \mathbf{a}_{0,0}+\mathbf{b}_{1,1} & \mathbf{b}_{1,2} & & \mathbf{a}_{0,1} & \\ \mathbf{b}_{2,0} & \mathbf{b}_{2,1} & \mathbf{a}_{0,0}+\mathbf{b}_{2,2} & & & \mathbf{a}_{0,1} \\ \hline \mathbf{a}_{1,0} & & & \mathbf{a}_{1,1}+\mathbf{b}_{0,0} & \mathbf{b}_{0,1} & \mathbf{b}_{0,2} \\ & \mathbf{a}_{1,0} & & \mathbf{b}_{1,0} & \mathbf{a}_{1,1}+\mathbf{b}_{1,1} & \mathbf{b}_{1,2} \\ & & \mathbf{a}_{1,0} & \mathbf{b}_{2,0} & \mathbf{b}_{2,1} & \mathbf{a}_{1,1}+\mathbf{b}_{2,2} \end{array}\right]$$

Kronecker sum and CTMC

- ❑ If we merge two independent Continuous Time Markov Chains (CTMC) with state spaces S_1 and S_2 and infinitesimal generators $\mathbf{Q}_1$ and $\mathbf{Q}_2$, the resulting state space and infinitesimal generator are:

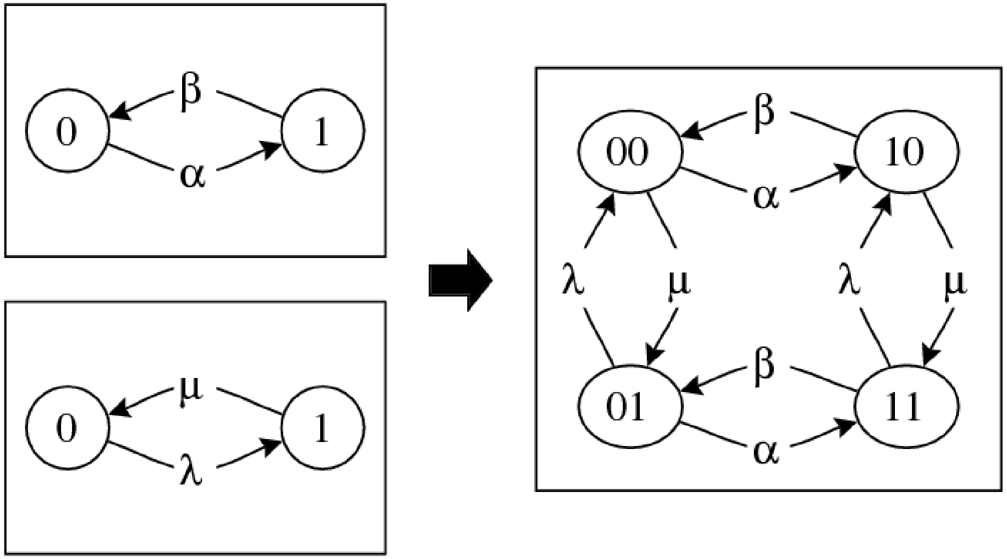
$$S = S_1 \times S_2 \qquad \text{and} \qquad \mathbf{R} = \mathbf{R}_1 \oplus \mathbf{R}_2$$

(and $\mathbf{Q} = \mathbf{Q}_1 \oplus \mathbf{Q}_2$)

$$\mathbf{Q} = \mathbf{R} - \mathrm{rowsum}(\mathbf{R})$$

Kronecker sum and CTMC

- Example



$$\mathcal{S}^1 = \{\underline{0}, \underline{1}\} \qquad \mathcal{S}^2 = \{\underline{0}, \underline{1}\} \qquad \mathcal{S} = \{0 \equiv \underline{00},\ 1 \equiv \underline{01},\ 2 \equiv \underline{10},\ 3 \equiv \underline{11}\}$$

$$\mathbf{R}^1 = \begin{bmatrix} & \alpha \\ \beta & \end{bmatrix} \qquad \mathbf{R}^2 = \begin{bmatrix} & \lambda \\ \mu & \end{bmatrix} \qquad \mathbf{R} = \left[\begin{array}{cc|cc} & \lambda & \alpha & \\ \mu & & & \alpha \\ \hline \beta & & & \lambda \\ & \beta & \mu & \end{array}\right]$$

Outline

- □ Kronecker product and DTMC
- □ Kronecker sum and CTMC
- □ Structured view of stochastic Petri nets
- □ Reachability set construction
- □ CTMC generation and solution
- □ Bibliography

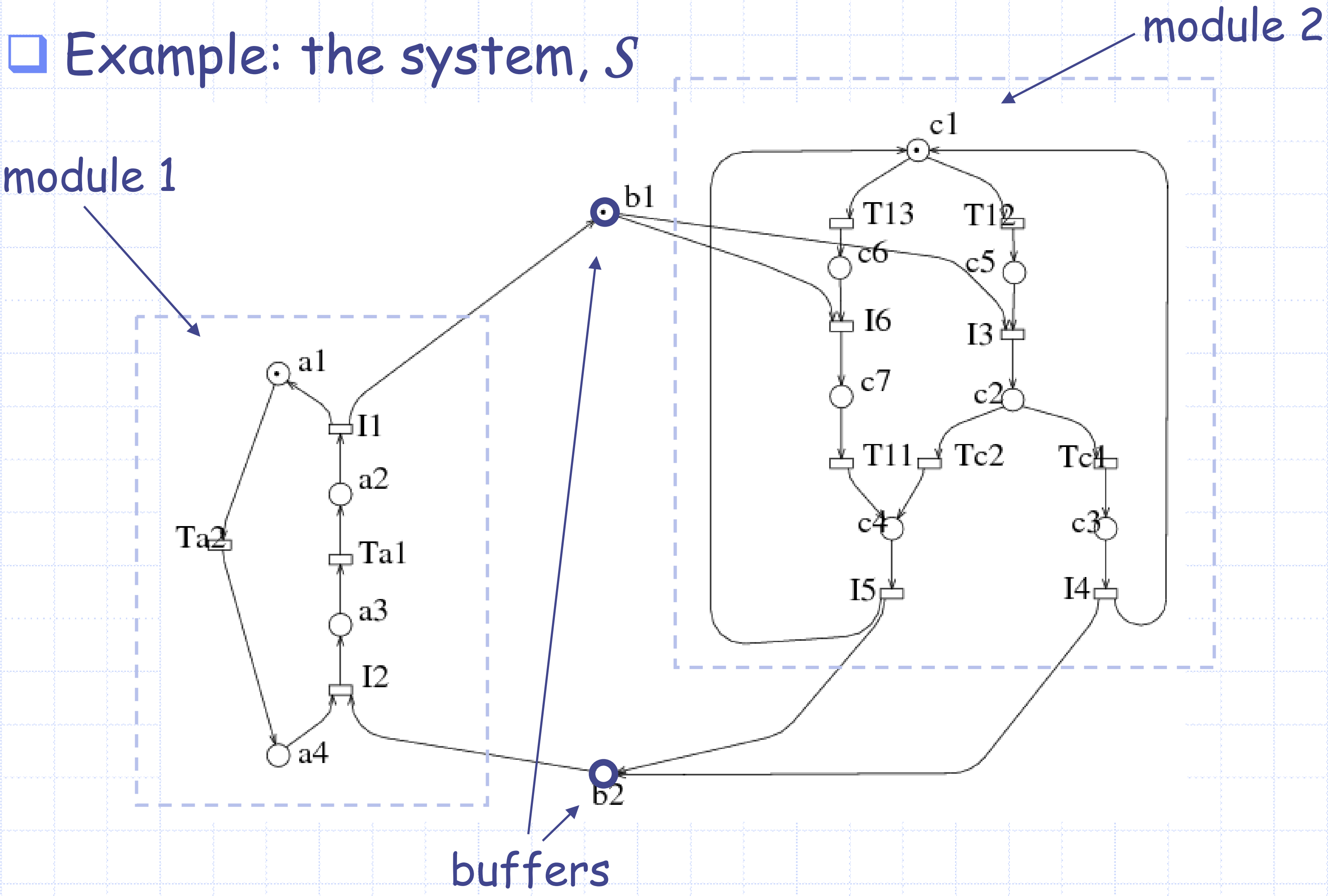
Structured view of stochastic Petri nets

- We come back to structured view of PN’s

partition of PN into **modules** (functional units) connected through places (**buffers**)

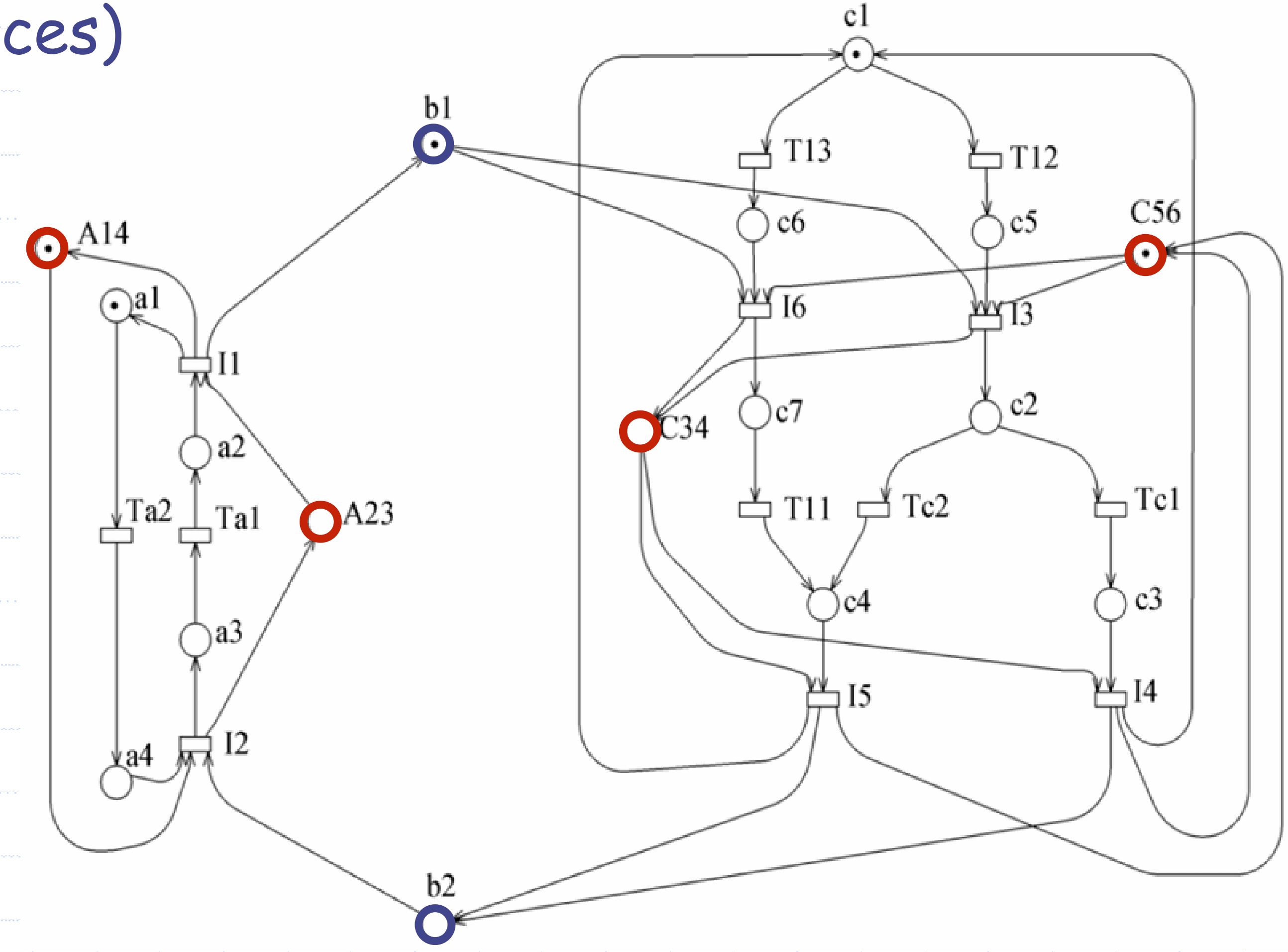
Structured view of stochastic Petri nets

- Example: the system, $\mathcal{S}$



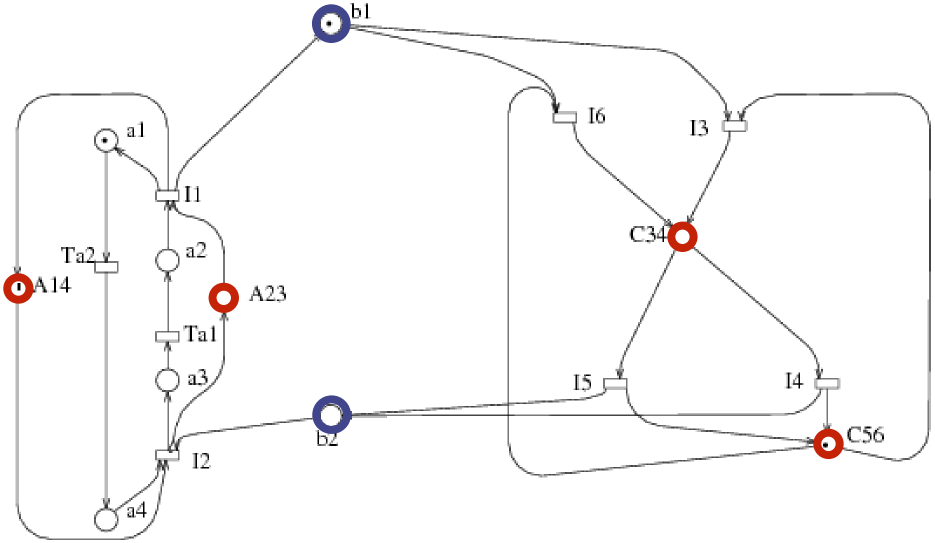
Structured view of stochastic Petri nets

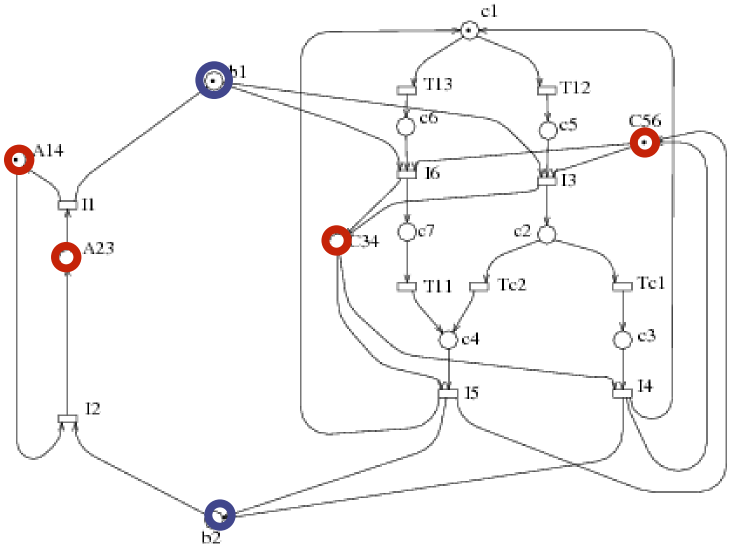
- ❑ Extended system, $\mathcal{ES}$ (addition of a set of implicit places)



Structured view of stochastic Petri nets

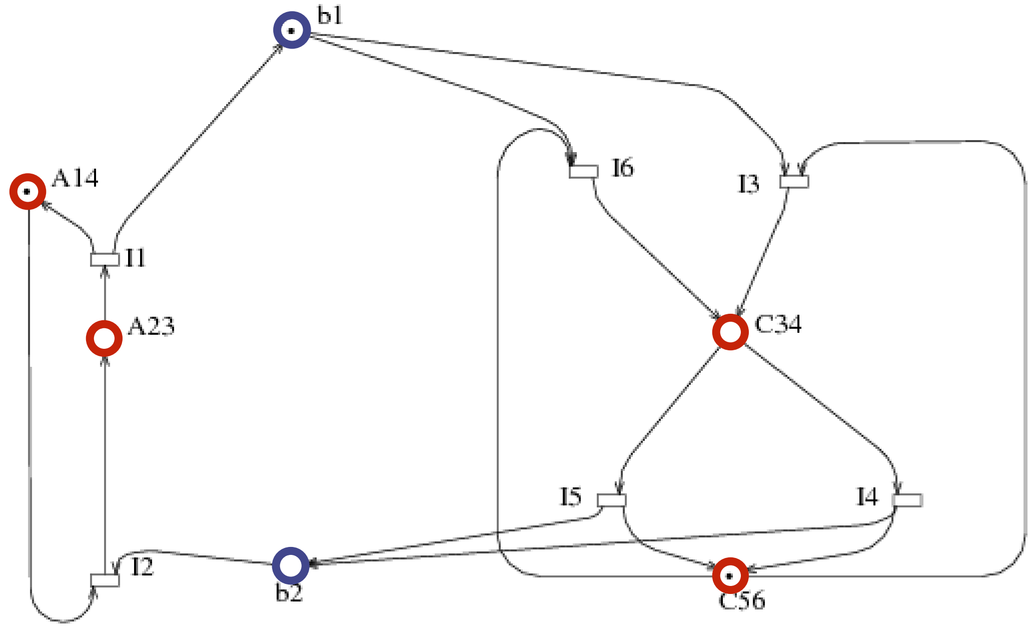
- Low level (sub)systems, $\mathcal{LS}$





Structured view of stochastic Petri nets

- ❑ Basic skeleton, $\mathcal{BS}$: high level view



Outline

- ❑ Kronecker product and DTMC
- ❑ Kronecker sum and CTMC
- ❑ Structured view of stochastic Petri nets
- ❑ Reachability set construction
- ❑ CTMC generation and solution
- ❑ Bibliography

Reachability set construction

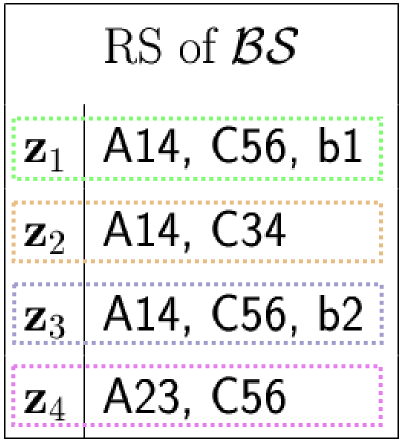
- We define the following subsets of reachability sets, for each $\mathbf{z} \in \mathrm{RS}(\mathcal{BS})$ (i.e. $\mathbf{z}$ is a high level state)

$$\mathrm{RS}_{\mathbf{z}}(\mathcal{ES}) \;=\; \{\mathbf{m} \in \mathrm{RS}(\mathcal{ES}) \;:\; \mathbf{m}|_{\underline{H_1\cup\ldots\cup H_K}\cup\underline{B}} = \mathbf{z}\}$$

$$\begin{aligned}\mathrm{RS}_{\mathbf{z}}(\mathcal{S}) \;=\; \{\mathbf{m} \in \mathrm{RS}(\mathcal{S}) \;\; &\text{such that} \\ &\exists \mathbf{m}' \in \mathrm{RS}_{\mathbf{z}}(\mathcal{ES}) : \mathbf{m}'|_{P_1\cup\ldots\cup P_K\cup B} = \mathbf{m}\}\end{aligned}$$

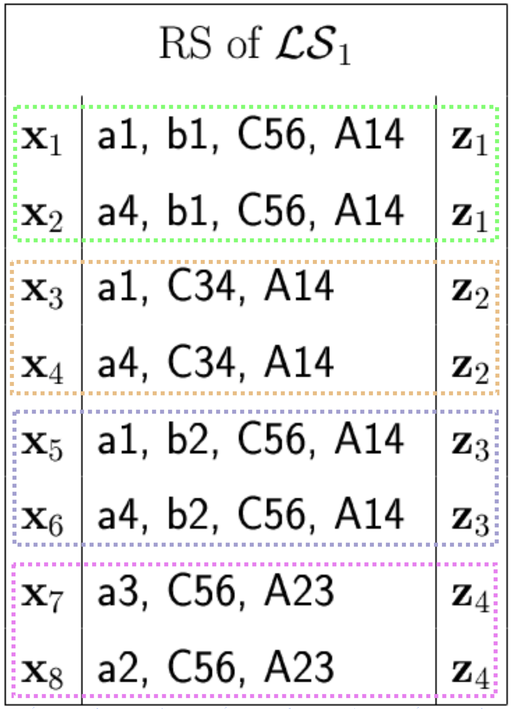
$$\mathrm{RS}_{\mathbf{z}}(\mathcal{LS}_i) \;=\; \{\mathbf{m}_i \in \mathrm{RS}(\mathcal{LS}_i) \;:\; \mathbf{m}_i|_{\underline{H_1\cup\ldots\cup H_K}\cup\underline{B}} = \mathbf{z}\}$$

Reachability set construction



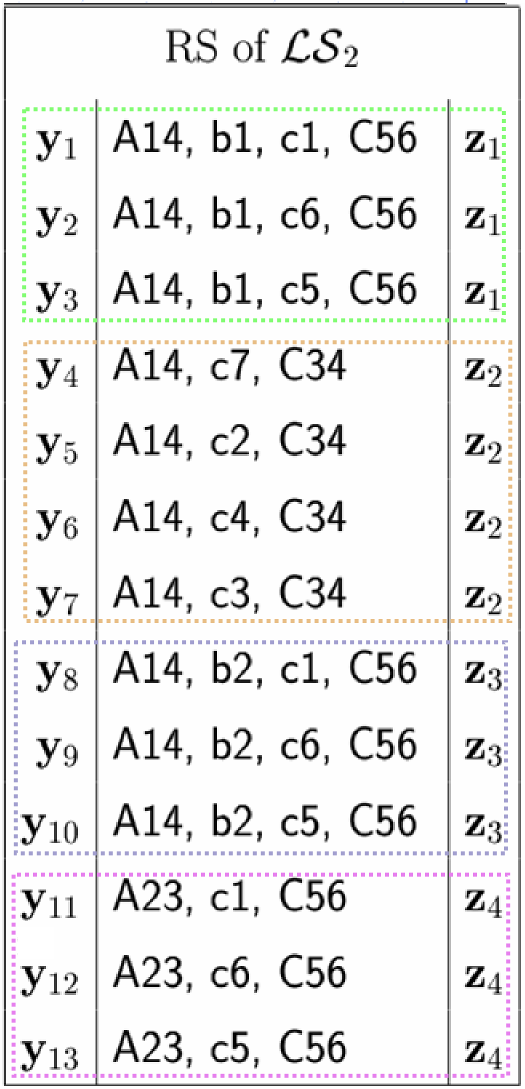
RS of $\mathcal{BS}$

$\mathbf{z}_1$	A14, C56, b1
$\mathbf{z}_2$	A14, C34
$\mathbf{z}_3$	A14, C56, b2
$\mathbf{z}_4$	A23, C56



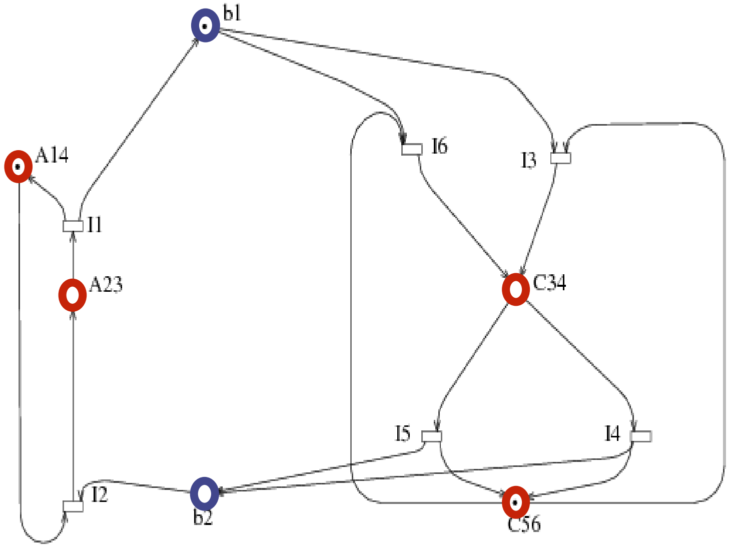
RS of $\mathcal{LS}_1$

$\mathbf{x}_1$	a1, b1, C56, A14	$\mathbf{z}_1$
$\mathbf{x}_2$	a4, b1, C56, A14	$\mathbf{z}_1$
$\mathbf{x}_3$	a1, C34, A14	$\mathbf{z}_2$
$\mathbf{x}_4$	a4, C34, A14	$\mathbf{z}_2$
$\mathbf{x}_5$	a1, b2, C56, A14	$\mathbf{z}_3$
$\mathbf{x}_6$	a4, b2, C56, A14	$\mathbf{z}_3$
$\mathbf{x}_7$	a3, C56, A23	$\mathbf{z}_4$
$\mathbf{x}_8$	a2, C56, A23	$\mathbf{z}_4$



RS of $\mathcal{LS}_2$

$\mathbf{y}_1$	A14, b1, c1, C56	$\mathbf{z}_1$
$\mathbf{y}_2$	A14, b1, c6, C56	$\mathbf{z}_1$
$\mathbf{y}_3$	A14, b1, c5, C56	$\mathbf{z}_1$
$\mathbf{y}_4$	A14, c7, C34	$\mathbf{z}_2$
$\mathbf{y}_5$	A14, c2, C34	$\mathbf{z}_2$
$\mathbf{y}_6$	A14, c4, C34	$\mathbf{z}_2$
$\mathbf{y}_7$	A14, c3, C34	$\mathbf{z}_2$
$\mathbf{y}_8$	A14, b2, c1, C56	$\mathbf{z}_3$
$\mathbf{y}_9$	A14, b2, c6, C56	$\mathbf{z}_3$
$\mathbf{y}_{10}$	A14, b2, c5, C56	$\mathbf{z}_3$
$\mathbf{y}_{11}$	A23, c1, C56	$\mathbf{z}_4$
$\mathbf{y}_{12}$	A23, c6, C56	$\mathbf{z}_4$
$\mathbf{y}_{13}$	A23, c5, C56	$\mathbf{z}_4$



Reachability set construction

$$\mathrm{PS}_{\mathbf{z}}(\mathcal{S}) = \{\mathbf{z}|_B\} \times \mathrm{RS}_{\mathbf{z}}(\mathcal{LS}_1)|_{P_1} \times \cdots \times \mathrm{RS}_{\mathbf{z}}(\mathcal{LS}_K)|_{P_K}$$

$$\mathrm{PS}(\mathcal{S}) = \biguplus_{\mathbf{z} \in \mathrm{RS}(\mathcal{BS})} \mathrm{PS}_{\mathbf{z}}(\mathcal{S})$$

$$\mathrm{RS}(\mathcal{S}) \subseteq \mathrm{PS}(\mathcal{S}) = \biguplus_{\mathbf{z} \in \mathrm{RS}(\mathcal{BS})} \mathrm{PS}_{\mathbf{z}}(\mathcal{S})$$

$$\mathrm{RS}_{\mathbf{z}}(\mathcal{S}) \subseteq \mathrm{PS}_{\mathbf{z}}(\mathcal{S})$$

RS of $\mathcal{S}$	
$\mathbf{v}_1$	a1, b1, c1
$\mathbf{v}_2$	a1, b1, c6
$\mathbf{v}_3$	a1, b1, c5
$\mathbf{v}_4$	a4, b1, c1
$\mathbf{v}_5$	a1, c7
$\mathbf{v}_6$	a4, b1, c6
$\mathbf{v}_7$	a4, b1, c5
$\mathbf{v}_8$	a1, c2
$\mathbf{v}_9$	a1, c4
$\mathbf{v}_{10}$	a4, c7
$\mathbf{v}_{11}$	a4, c2
$\mathbf{v}_{12}$	a1, c3
$\mathbf{v}_{13}$	a4, c4

RS of $\mathcal{S}$	
$\mathbf{v}_{14}$	a1, c1, b2
$\mathbf{v}_{15}$	a4, c3
$\mathbf{v}_{16}$	a4, c1, b2
$\mathbf{v}_{17}$	a1, b2, c6
$\mathbf{v}_{18}$	a1, b2, c5
$\mathbf{v}_{19}$	a4, b2, c6
$\mathbf{v}_{20}$	a4, b2, c5
$\mathbf{v}_{21}$	a3, c1
$\mathbf{v}_{22}$	a3, c6
$\mathbf{v}_{23}$	a3, c5
$\mathbf{v}_{24}$	a2, c1
$\mathbf{v}_{25}$	a2, c6
$\mathbf{v}_{26}$	a2, c5

Outline

- ❑ Kronecker product and DTMC
- ❑ Kronecker sum and CTMC
- ❑ Structured view of stochastic Petri nets
- ❑ Reachability set construction
- ❑ CTMC generation and solution
- ❑ Bibliography

CTMC generation and solution

- Basic idea: split the behaviour in two:
  - transitions that **change** the high level view
  - transitions that **do not change** the high level view

CTMC generation and solution

For the system $\mathcal{S}$: $\mathbf{Q} = \mathbf{R} - \mathrm{rowsum}(\mathbf{R})$

For the components $\mathcal{LS}_i$: $\mathbf{Q}_i = \mathbf{R}_i - \mathrm{rowsum}(\mathbf{R}_i)$

Technique:

1. Consider **Q** and **R** in blocks $(\mathbf{z},\mathbf{z}')$, of size $|RS_{\mathbf{z}}(\mathcal{S})| \cdot |RS_{\mathbf{z}'}(\mathcal{S})|$

2. Consider $\mathbf{Q}_i$ and $\mathbf{R}_i$ in blocks $(\mathbf{z},\mathbf{z}')$, of size $|RS_{\mathbf{z}}(\mathcal{LS}_i)| \cdot |RS_{\mathbf{z}'}(\mathcal{LS}_i)|$

3. Describe each block of **Q** and **R** as tensor expression of the blocks of $\mathbf{Q}_i$ and $\mathbf{R}_i$

CTMC generation and solution

- □ Blocks **R(z,z)** have non-null entries that are due only to non interface transitions

$$\mathbf{G}(\mathbf{z},\mathbf{z}) = \bigoplus_{i=1}^{K} \mathbf{R}_i(\mathbf{z},\mathbf{z})$$

- □ Blocks **R(z,z')** with **z** ≠ **z'** have non-null entries that are due only to the firing of interface transitions (TI)

$$\mathbf{K}_i(t)(\mathbf{z},\mathbf{z}')[\mathbf{m},\mathbf{m}'] = \begin{cases} 1 & \text{if } \mathbf{m}\xrightarrow{t}\mathbf{m}' \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{G}(\mathbf{z},\mathbf{z}') = \sum_{t\in \mathrm{TI}_{\mathbf{z},\mathbf{z}'}} w(t) \bigotimes_{i=1}^{K} \mathbf{K}_i(t)(\mathbf{z},\mathbf{z}')$$

CTMC generation and solution

- The result:
  - Transition rates among **reachable states** are correctly computed

    for all $\mathbf{z},\mathbf{z}' \in RS(\mathcal{BS})$:

    $\mathbf{R}(\mathbf{z},\mathbf{z}')$ is a submatrix of $\mathbf{G}(\mathbf{z},\mathbf{z}')$

  - Unreachable states are never assigned a non-null probability

    for all $\mathbf{m} \in RS(\mathcal{S})$ and for all $\mathbf{m}' \in PS(\mathcal{S}) \setminus RS(\mathcal{S})$:

    $\mathbf{G}[\mathbf{m},\mathbf{m}'] = 0$

CTMC generation and solution

- Computational costs
  - To solve an SPN with classic method
    - Build and store the RG
    - Compute the associated CTMC
    - Solve the characteristic equation $\pi \cdot \mathbf{Q} = 0$
  - To solve an SPN with Kronecker approach
    - Build and store the K+1 auxiliary models
    - Compute the RG_i of each auxiliary model
    - Compute matrices $\mathbf{R}_i(\mathbf{z},\mathbf{z}')$ and $\mathbf{K}_i(t)(\mathbf{z},\mathbf{z}')$
    - Solve the characteristic equation $\pi \cdot \mathbf{Q} = 0$

    ➔ Whole system RG and matrix is never stored

Outline

□ Kronecker product and DTMC

□ Kronecker sum and CTMC

□ Structured view of stochastic Petri nets

□ Reachability set construction

□ CTMC generation and solution

□ Bibliography

Bibliography

- J. Campos, S. Donatelli, M. Silva: **Structured Solution of Asynchronously Communicating Stochastic Modules.** *IEEE Transactions on Software Engineering*, vol. 25, no. 2, pp. 147-165, March 1999. Download here.

Modelling and analysis of concurrent systems with Petri nets. Performance evaluation

7. Software performance engineering with UML and PNs

Javier Campos
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, Spain
jcampos@unizar.es

Outline

- □ Software Performance Engineering: basics
- □ A Software Performance Process
- □ Annotated UML Diagrams
- □ Integrating with Petri nets: case study
- □ Performance analysis
- □ Automation of the approach
- □ Real example
- □ Conclusions
- □ Bibliography

Software Performance Engineering: basics

- Traditional software development
  - Main focus on software correctness
    - Functional requirements, capabilities
    - What the software will do?
  - Non-functional requirements
    - quality requirements like accuracy, performance, security, modifiability, easiness of use...
    - introduced later in the development process:

"Fix-it-later" approach

Software Performance Engineering: basics

- Typical example of fix-it-later approach:
  - Denver airport story (1994)
    - Integrated automated baggage handling system
      - Planned development budget increased by 2 billion US$
      - Opening of the airport was delayed 16 months
      - To make it work it was necessary to reduce its complexity and loads, the concept of "fully automated" was gone

      - Conceptually: line balancing problem

Software Performance Engineering: basics

- Software Performance Engineering
  - A systematic, quantitative approach to construct software systems that meet performance objectives
  - Two important dimensions
    - Responsiveness: ability to meet its objectives for response time or throughput
    - Scalability: ability to continue to meet responsiveness as the demand for the software functions increases

Software Performance Engineering: basics

- The objective of the approach
  - Predicting performance goals at early phases of the life cycle
  - Evaluating performance goals at final phases
- The way
  - Use of performance modelling
    - Formal models coupled with software requirements, architectures, specifications and design documents
    - Automation of the approach (CASE tool development)

Software Performance Engineering: basics

- Research community
  - Term “SPE” coined in 1981 by Connie U. Smith
  - The International Workshop on Software and Performance (WOSP)
    - Santa Fe, US, 1998; Ottawa, CA, 2000; Rome, IT, 2002;
      Redwood City, US, 2004; Palma de Mallorca, ES, 2005;
      Buenos Aires, AR, 2007
    - An international workshop sponsored by ACM SIGMETRICS, ACM SIGSOFT, IFIP WG 6.3 and 7.3
  - About 5000 entries in scholar.google.com

Outline

- ❑ Software Performance Engineering: basics
- ❑ A Software Performance Process
- ❑ Annotated UML Diagrams
- ❑ Integrating with Petri nets: case study
- ❑ Performance analysis
- ❑ Automation of the approach
- ❑ Real example
- ❑ Conclusions
- ❑ Bibliography

A Software Performance Process

- What, when and how conduct SPE activities during software development

- Integrated method for SPE:
  - Integration of software models and performance models
  - Integration of performance analysis in the software life cycle
  - Methodology suitable for automation (tool)

A Software Performance Process

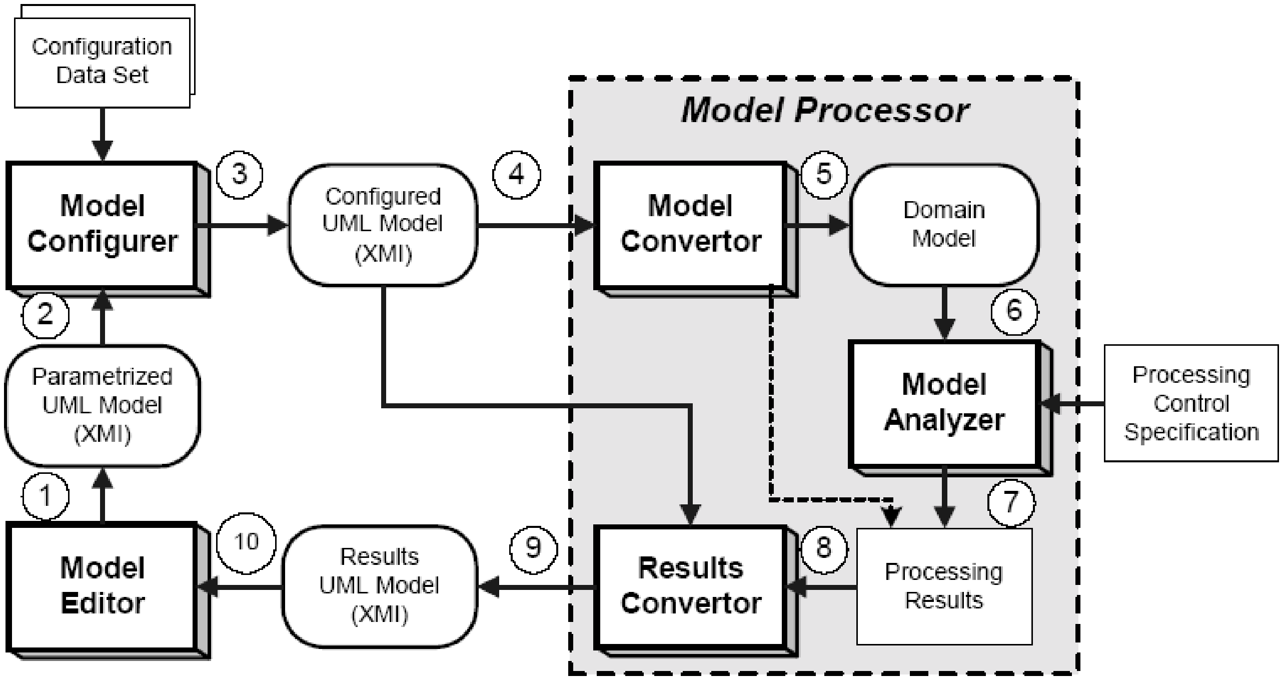
- Integration of software models and performance models
  - System design: The behaviour and architecture of the system is described by a set of UML diagrams
  - Annotated design: the UML design is annotated according to a standard OMG profile
  - Performance model: the annotated design is translated to a performance modelling formalism (SPN)

A Software Performance Process

- Integration of performance analysis in the software life cycle
  - The method applies at software specification time
  - The precision of performance predictions matches the software knowledge available at each stage
  - Feedback information is possible
    - when a direct correspondence exists between software specification abstraction level and performance model evaluation results
    - understanding the quantitative impact of design alternatives (effect of system changes on performance)

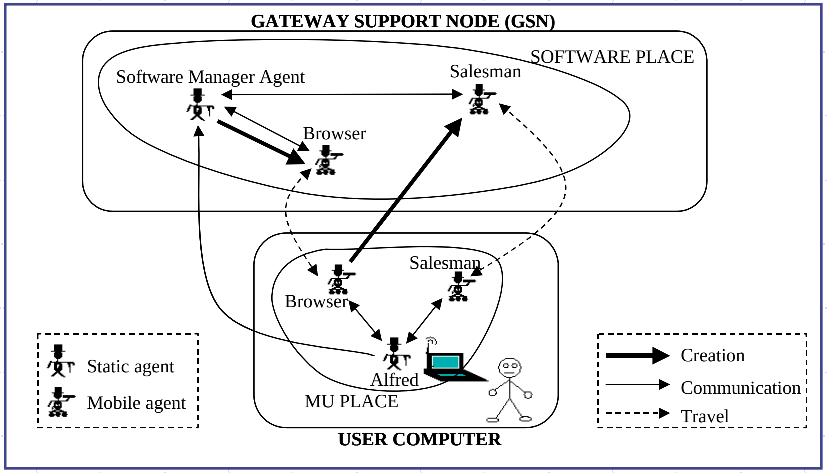
A Software Performance Process

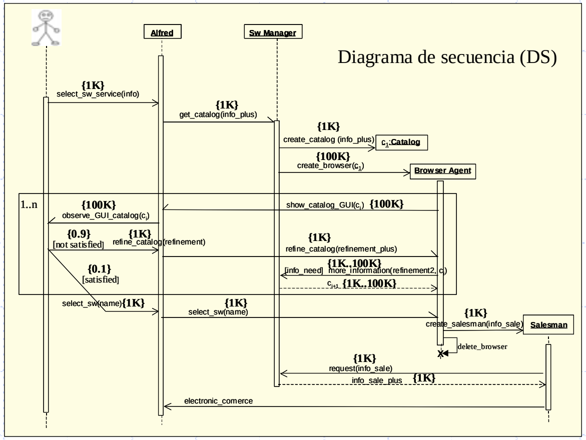
- ❑ Methodology suitable for automation (tool)
  - ❑ Following the OMG architectural framework for SPE tools

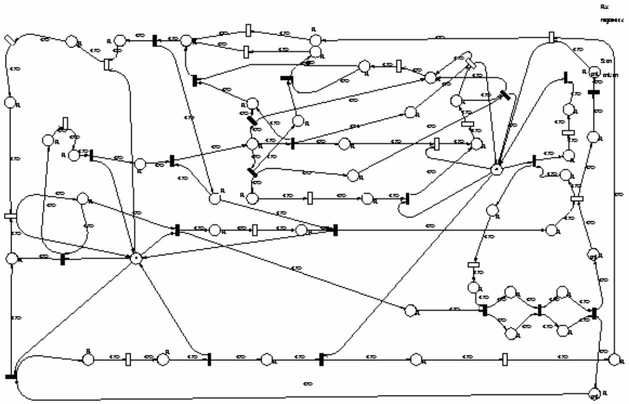


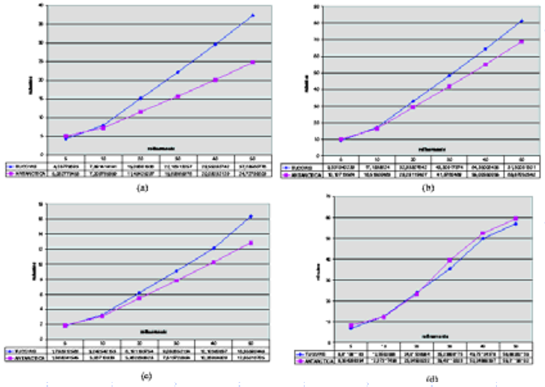
A Software Performance Process

❑ The overall picture









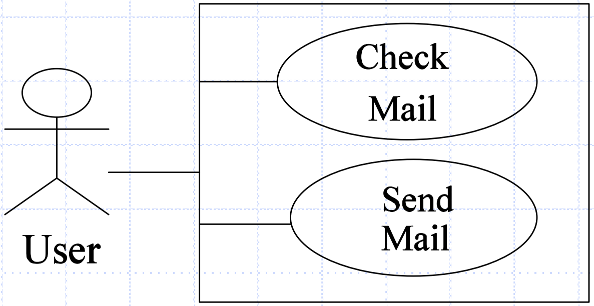
Outline

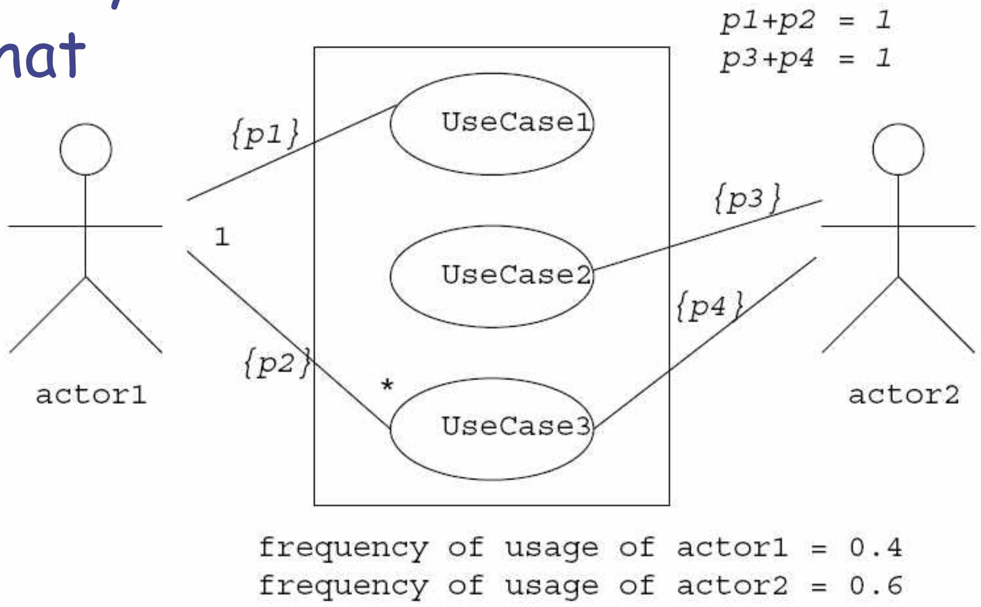
- ❑ Software Performance Engineering: basics
- ❑ A Software Performance Process
- ❑ Annotated UML Diagrams
- ❑ Integrating with Petri nets: case study
- ❑ Performance analysis
- ❑ Automation of the approach
- ❑ Real example
- ❑ Conclusions
- ❑ Bibliography

Annotated UML Diagrams

- ❑ Use Cases and actors:
  - ❑ Starting point to describe system behaviour
  - ❑ Specify the requirements of a system, subsystem or class and their functionality
  - ❑ Tag: probability that an actor executes a use case
  - ❑ Detailed later with sequence diagrams

“Mail client” model





frequency of usage of actor1 = 0.4
frequency of usage of actor2 = 0.6

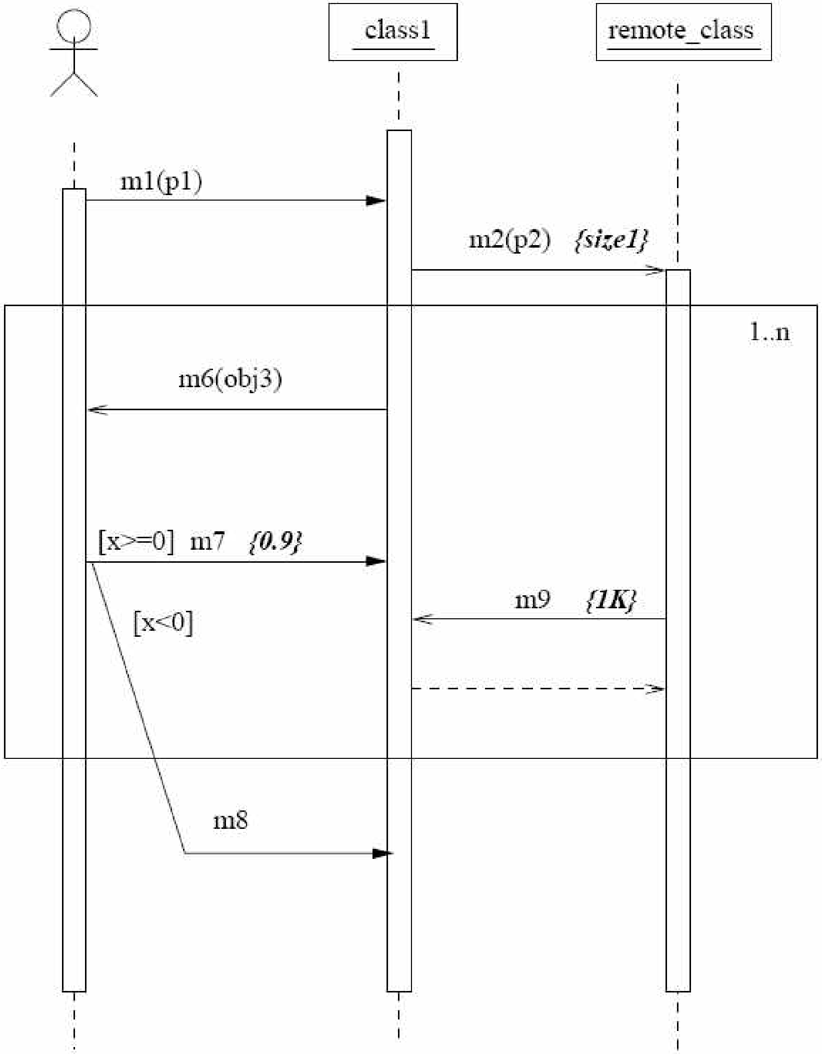
Annotated UML Diagrams

- Sequence Diagrams:
  - Used to detail Use Cases
  - Specify a set of partially ordered messages
  - Each message defines a communication mechanism and the roles to be played by sender/receiver

  Represent patterns of interaction between objects

Annotated UML Diagrams

- ❑ Sequence Diagrams (cont):
  - ❑ Tags: message sizes, messages routing rates
  - ❑ Will be used to derive a SPN performance model of a particular scenario (together with a set of state charts)

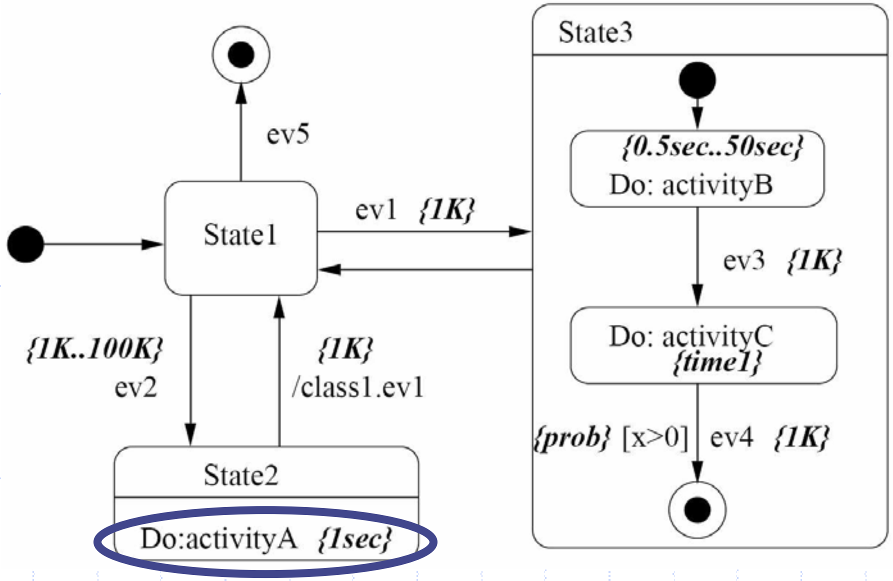


Annotated UML Diagrams

- Statecharts:
  - Used to describe the behaviour of a model element, such as an object
  - Describe possible state sequences and actions during the life of the object
  - Complete view of system behaviour: life of all the objects involved → used to derive a SPN performance model
  - Particular scenario: Statecharts together with a Sequence Diagram → used to derive a SPN performance model of concrete executions

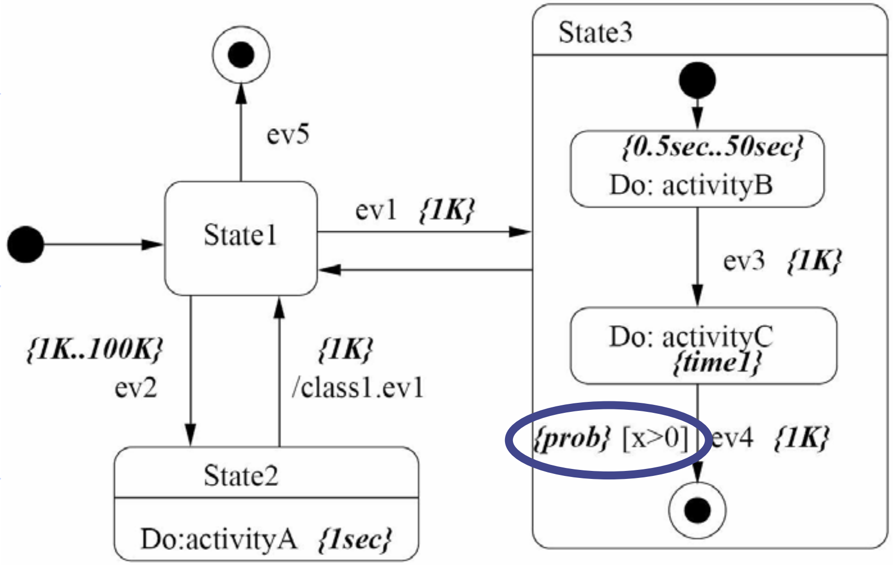
Annotated UML Diagrams

- Statecharts (cont):
  - Elements for integration of performance information: activities, guards and events
    - Activities: tasks performed in a given state → → annotated computation time



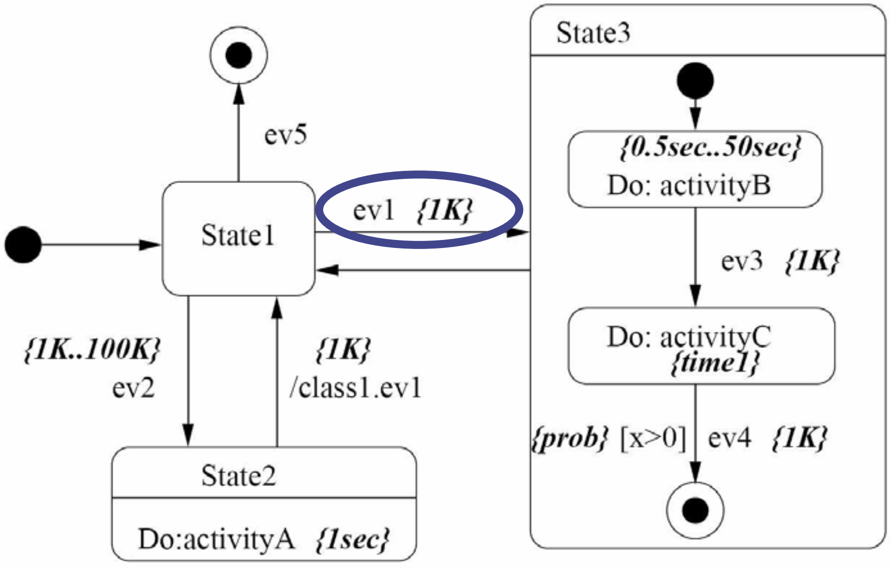
Annotated UML Diagrams

- Statecharts (cont):
  - Elements for integration of performance information: activities, guards and events
    - Guards: conditions in a transition that must hold to fire the event → annotated routing rates



Annotated UML Diagrams

- Statecharts (cont):
  - Elements for integration of performance information: activities, guards and events
    - Events: messages in the sequence diagram between server and receiver objects → annotated message size



Annotated UML Diagrams

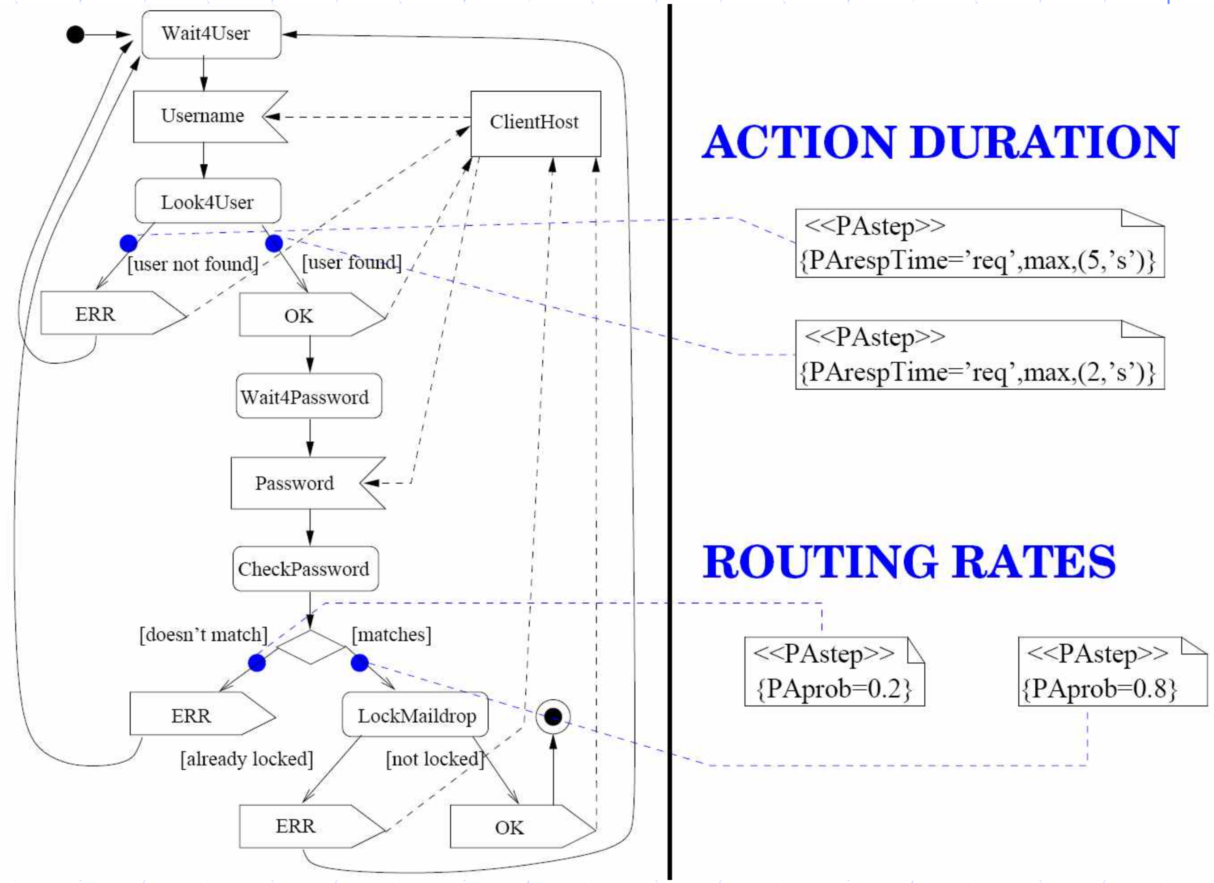
- Activity Diagrams:
  - Refine doActivities in a Statechart
  - We use them for detailing internal control flow of a process
  - In contrast to Statecharts, driven by external events

    → more detailed modelling of Statecharts

  - Used to derive a SPN performance model

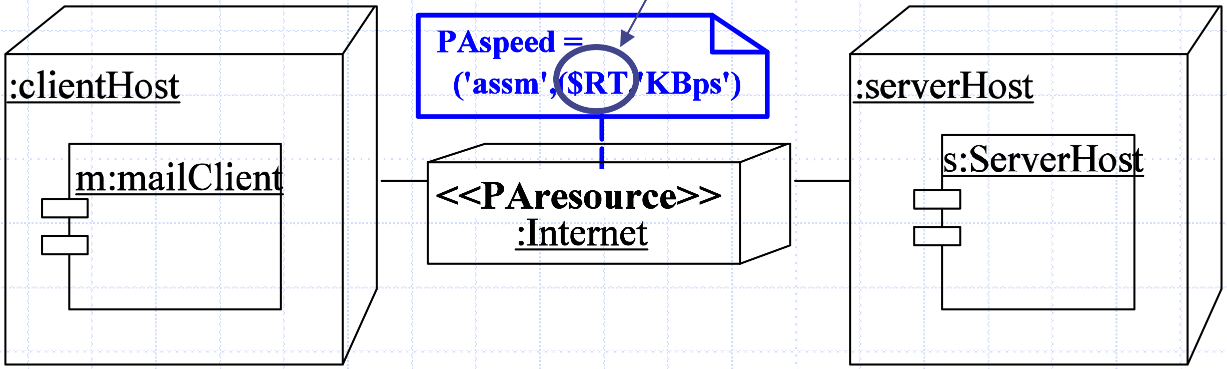
Annotated UML Diagrams

- ❑ Activity Diagrams (cont):
  - ❑ Performance annotations:
    - ❑ Routing rates
    - ❑ Activity durations



Annotated UML Diagrams

- ❑ Deployment diagram:
  - ❑ Models the distribution of software components in the hardware platform/network and O.S. resources
  - ❑ Annotated with transfer bit rate of the communication network

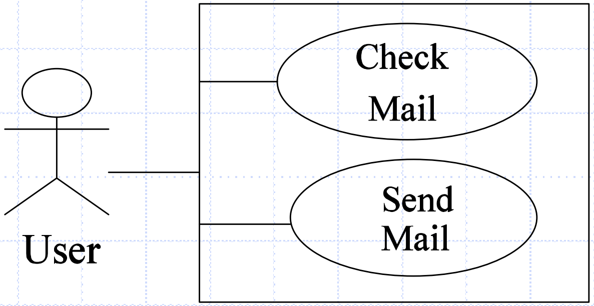


Outline

- ❑ Software Performance Engineering: basics
- ❑ A Software Performance Process
- ❑ Annotated UML Diagrams
- ❑ Integrating with Petri nets: case study
- ❑ Performance analysis
- ❑ Automation of the approach
- ❑ Real example
- ❑ Conclusions
- ❑ Bibliography

Integrating with Petri nets: case study

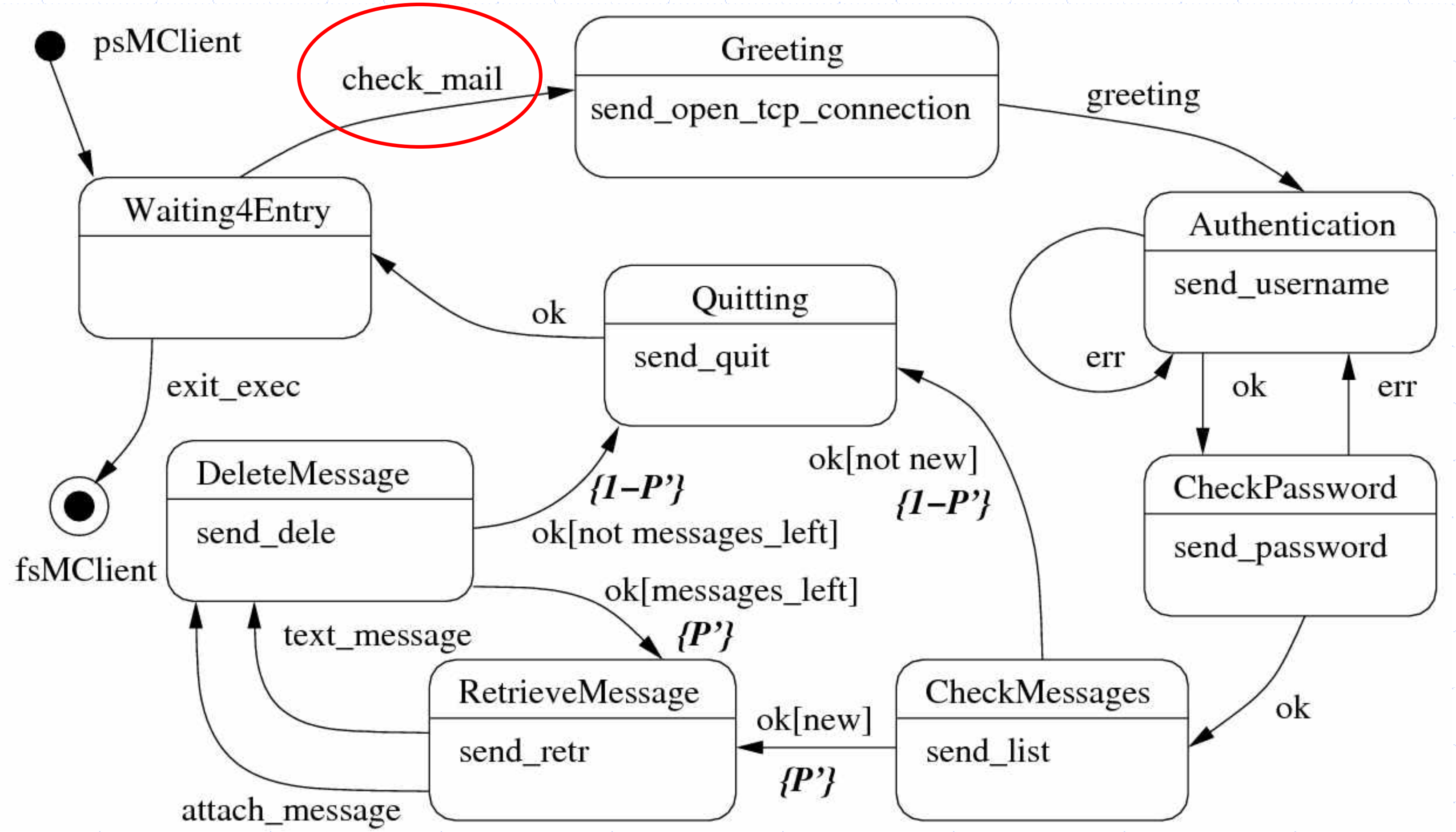
- A basic mail client



- We focus in the first use case:
  - check mail from a server using the POP3 protocol

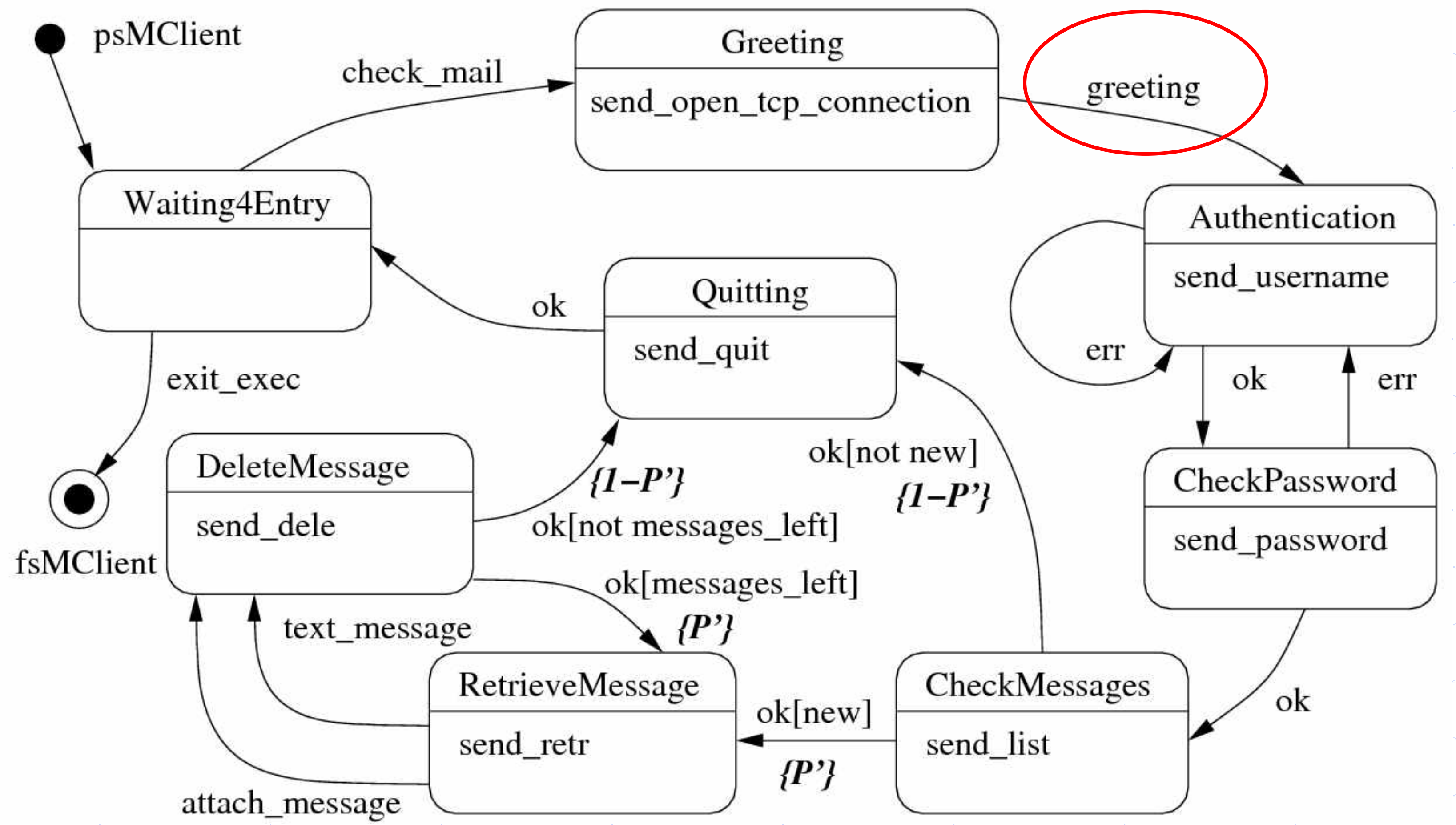
Integrating with Petri nets: case study

- The client tries to establish a TCP connection with the server via port 110 (Statechart for the class ClientHost: client behaviour)



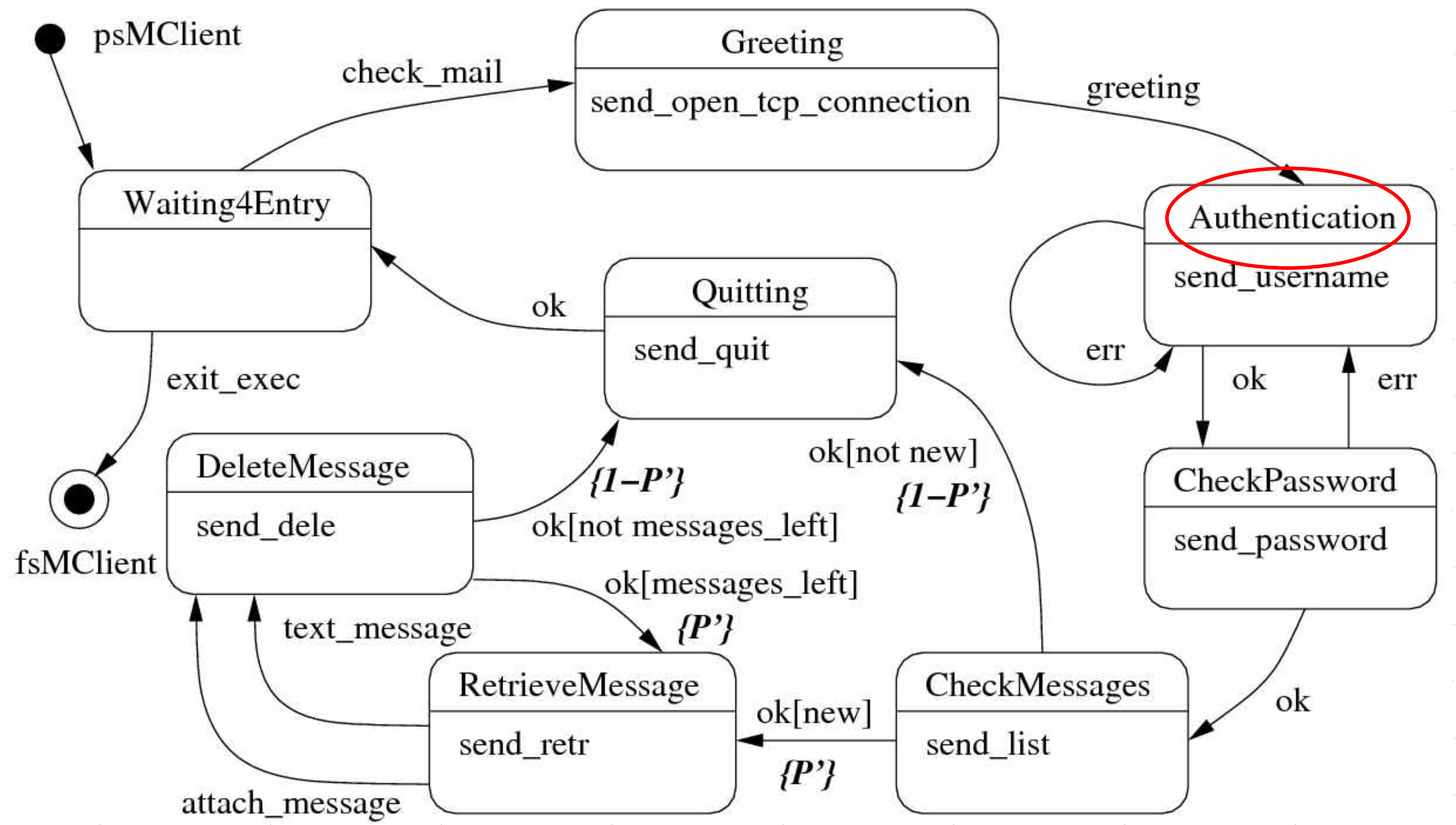
Integrating with Petri nets: case study

- If it succeeds → reception of greeting message



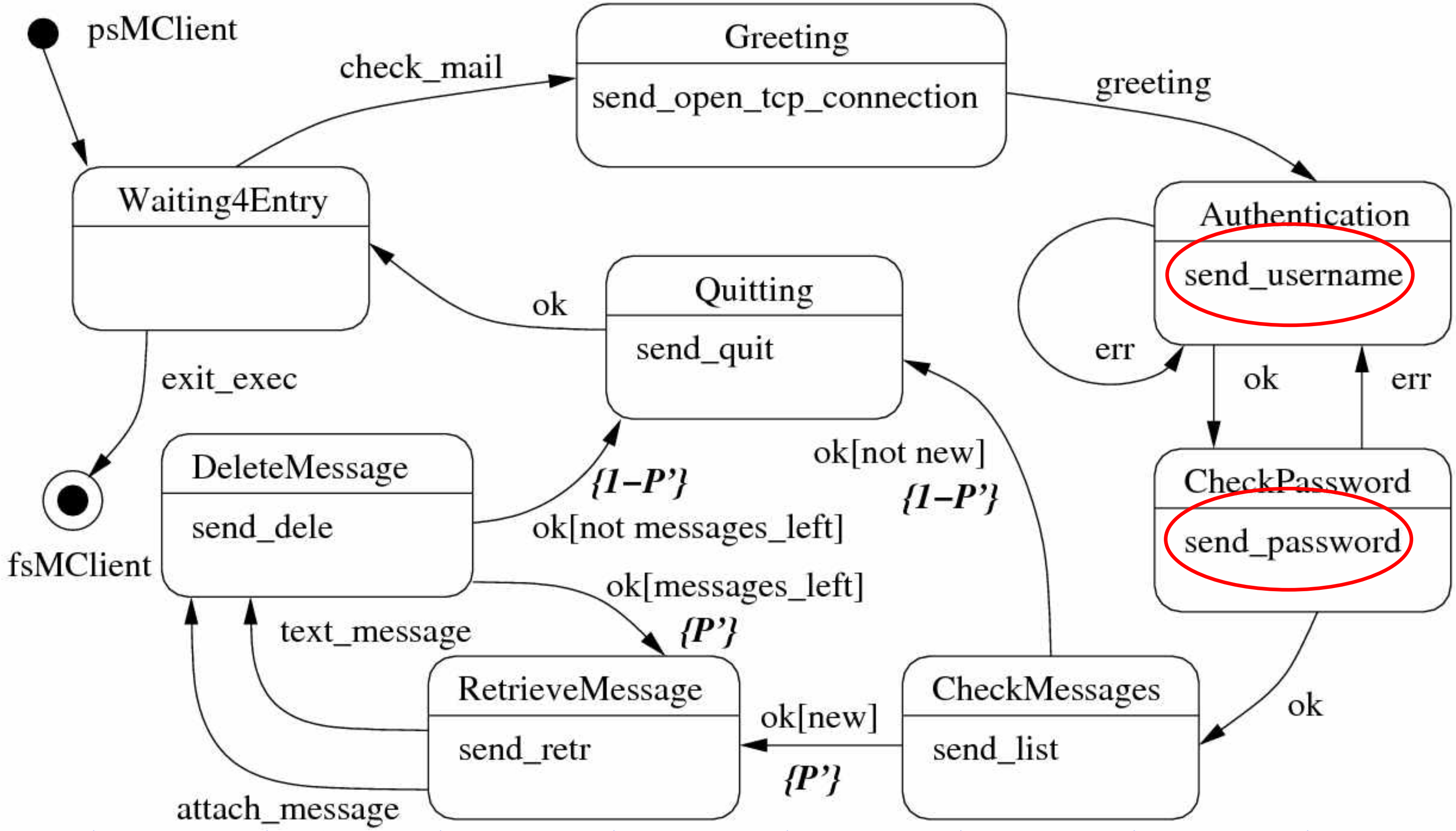
Integrating with Petri nets: case study

- Both client and server begin authentication (authorization) phase



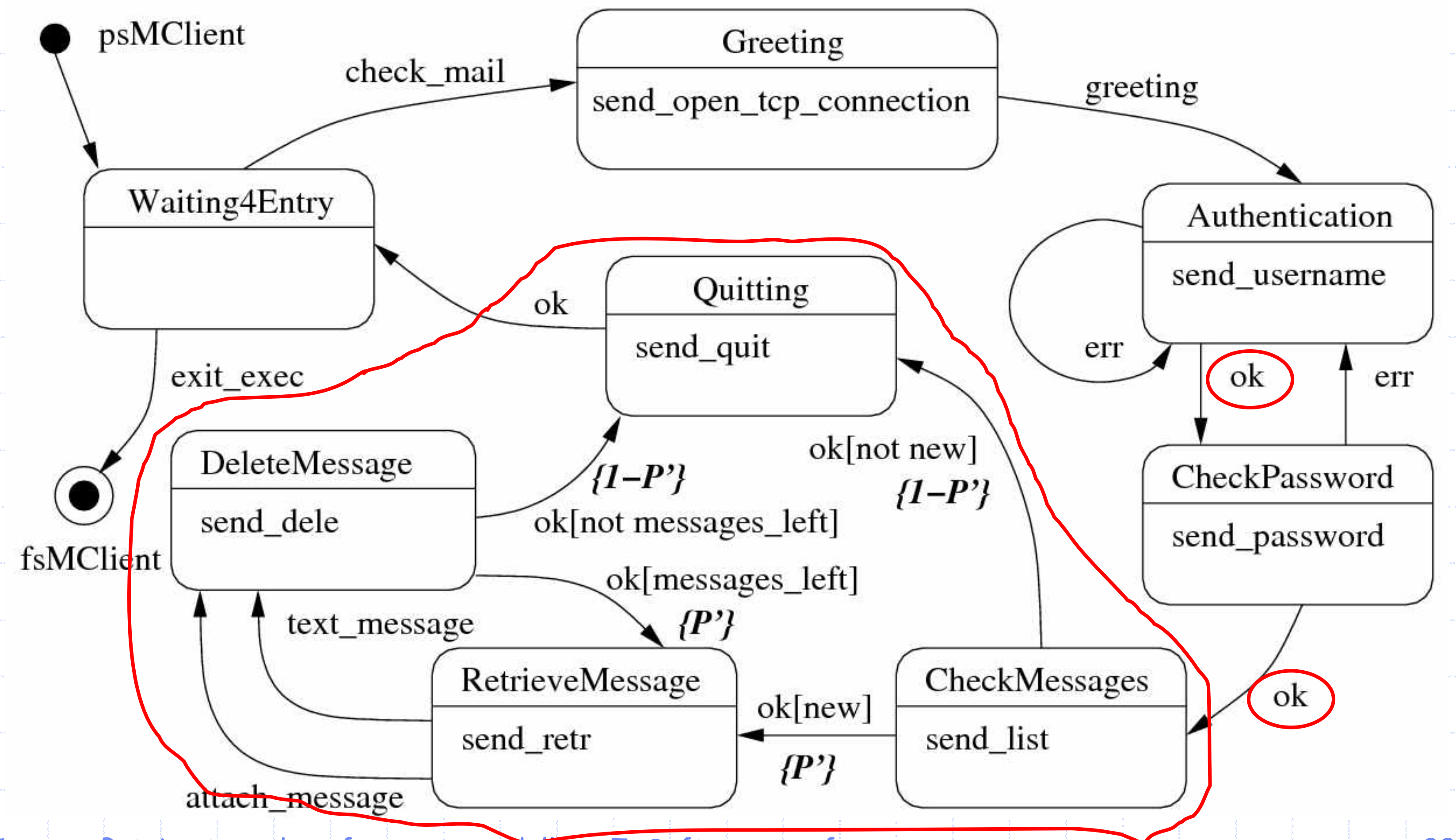
Integrating with Petri nets: case study

- The client sends username/password through USER and PASS command combination



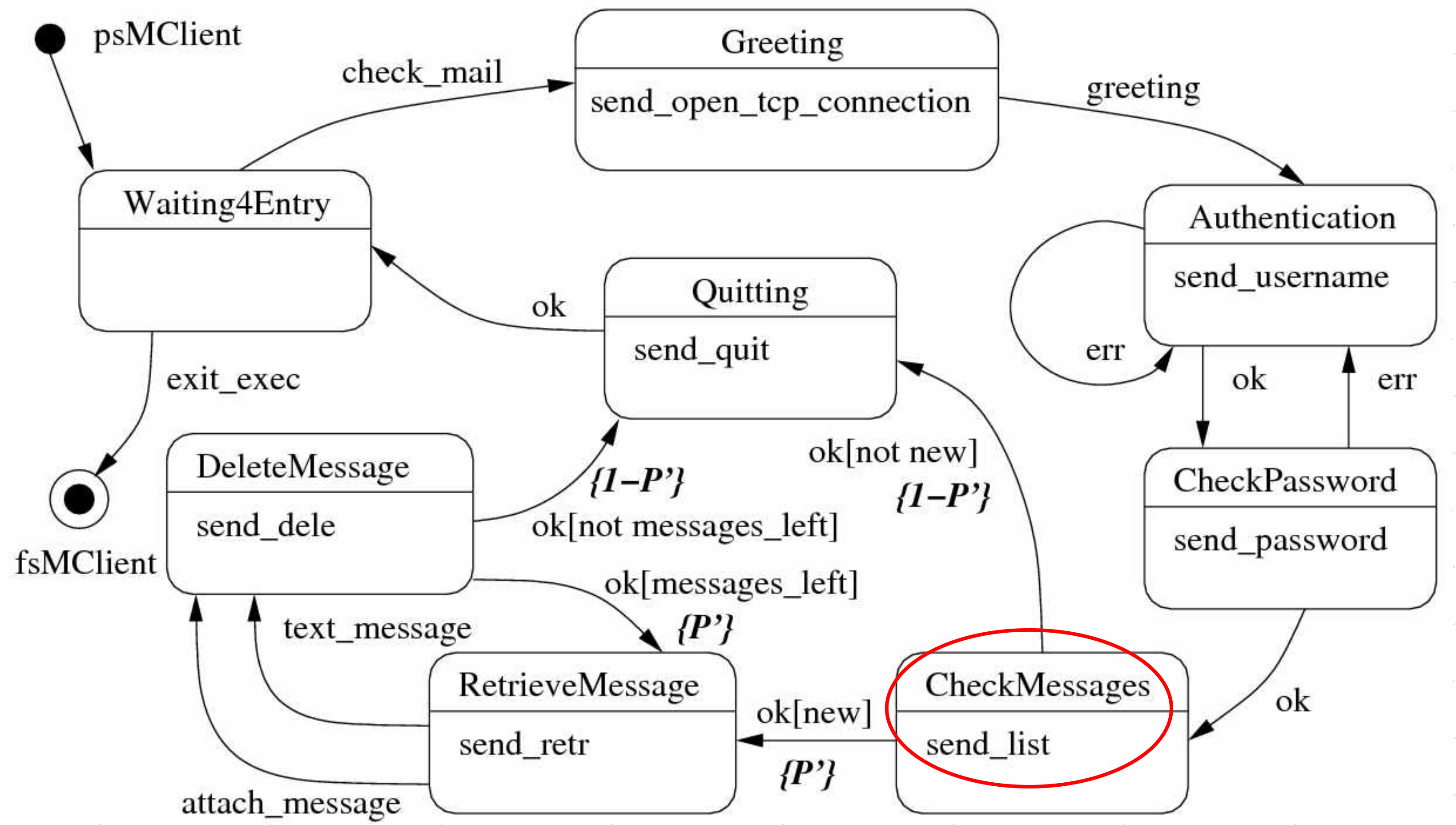
Integrating with Petri nets: case study

- If server answers "ok" to both messages, the POP3 session enters the transaction phase, otherwise... "err"...



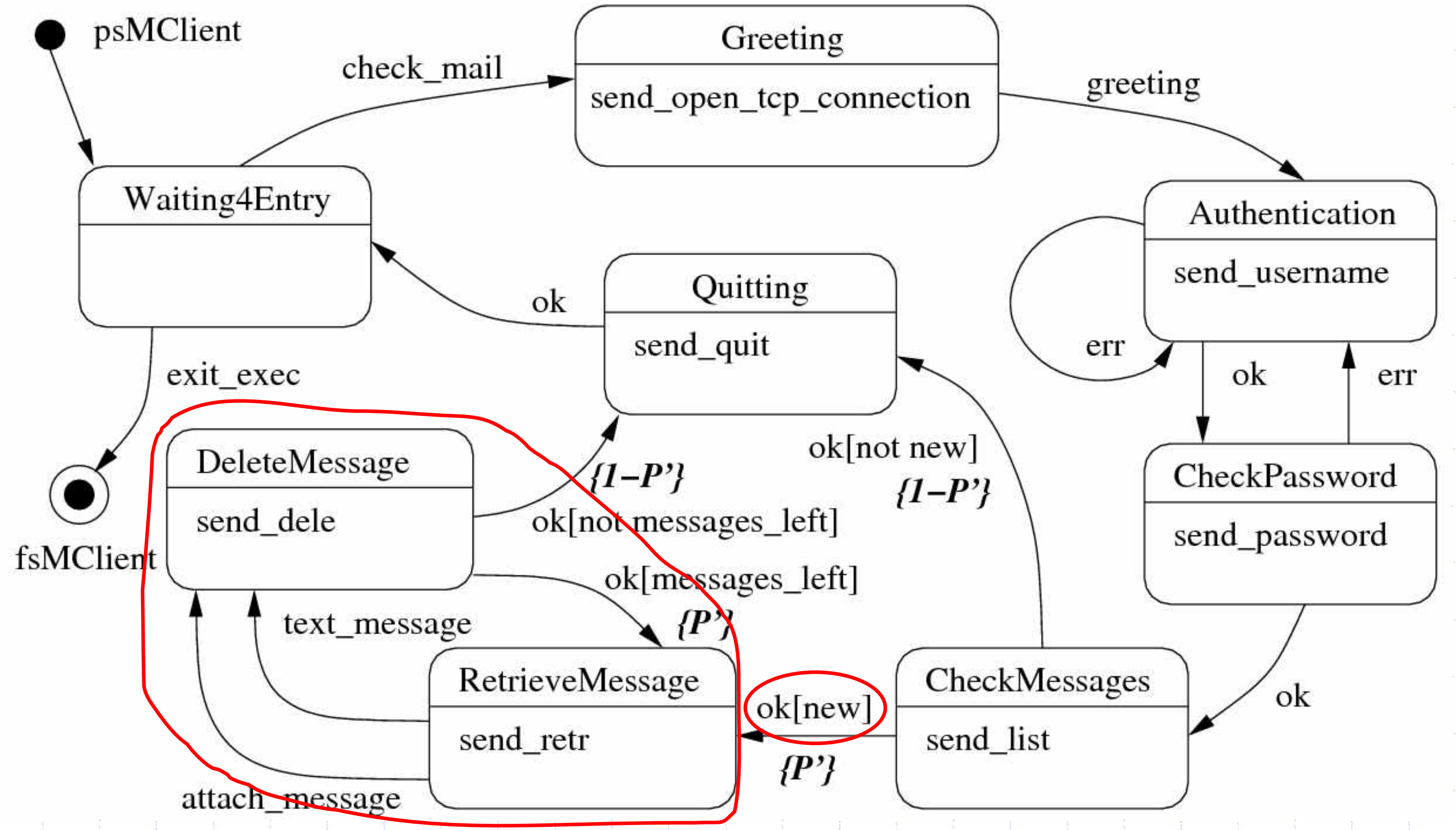
Integrating with Petri nets: case study

- The client checks for new mail using LIST command



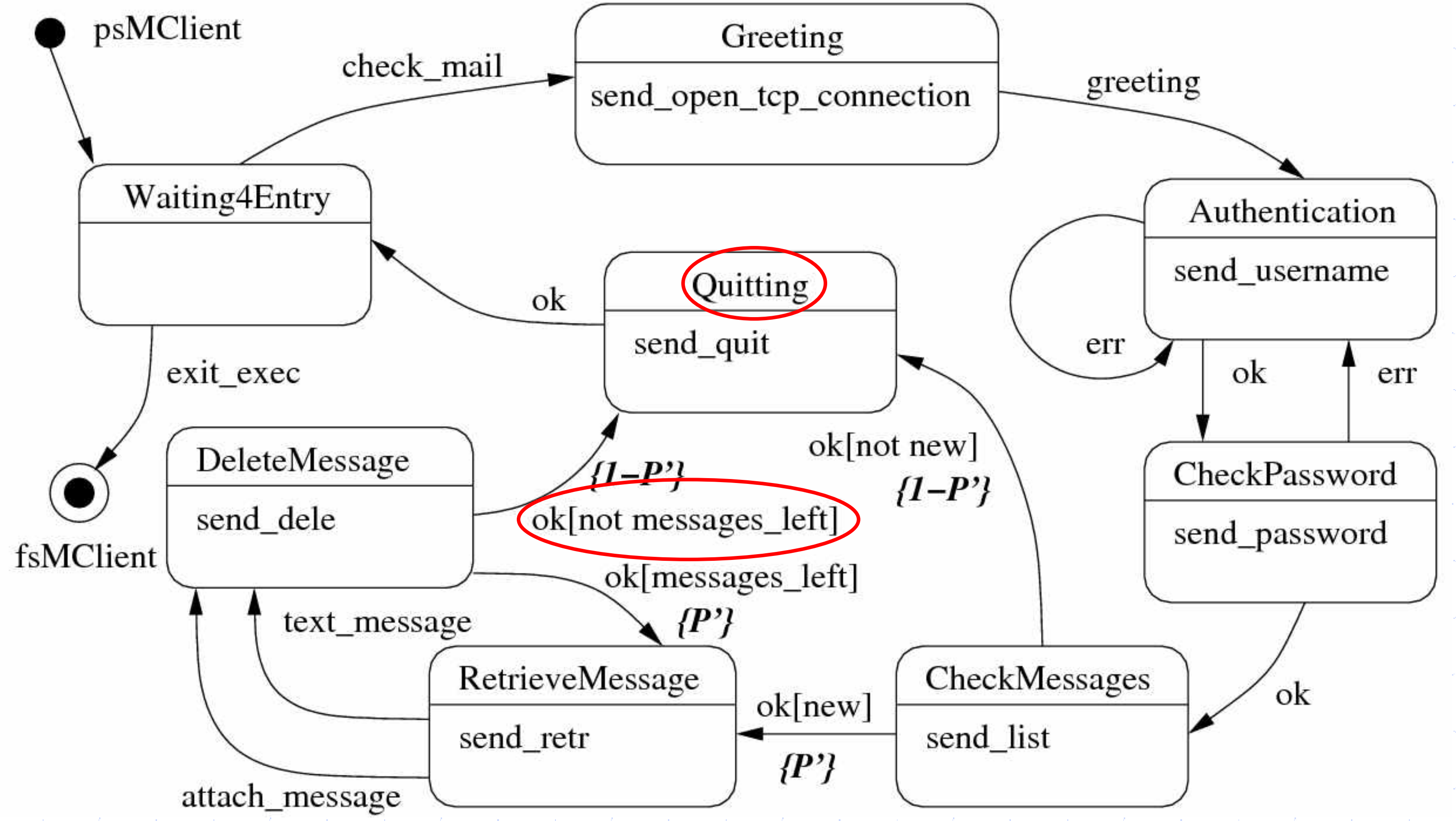
Integrating with Petri nets: case study

- If there is any new mail, the client obtains every mail by means of RETR and DELE commands



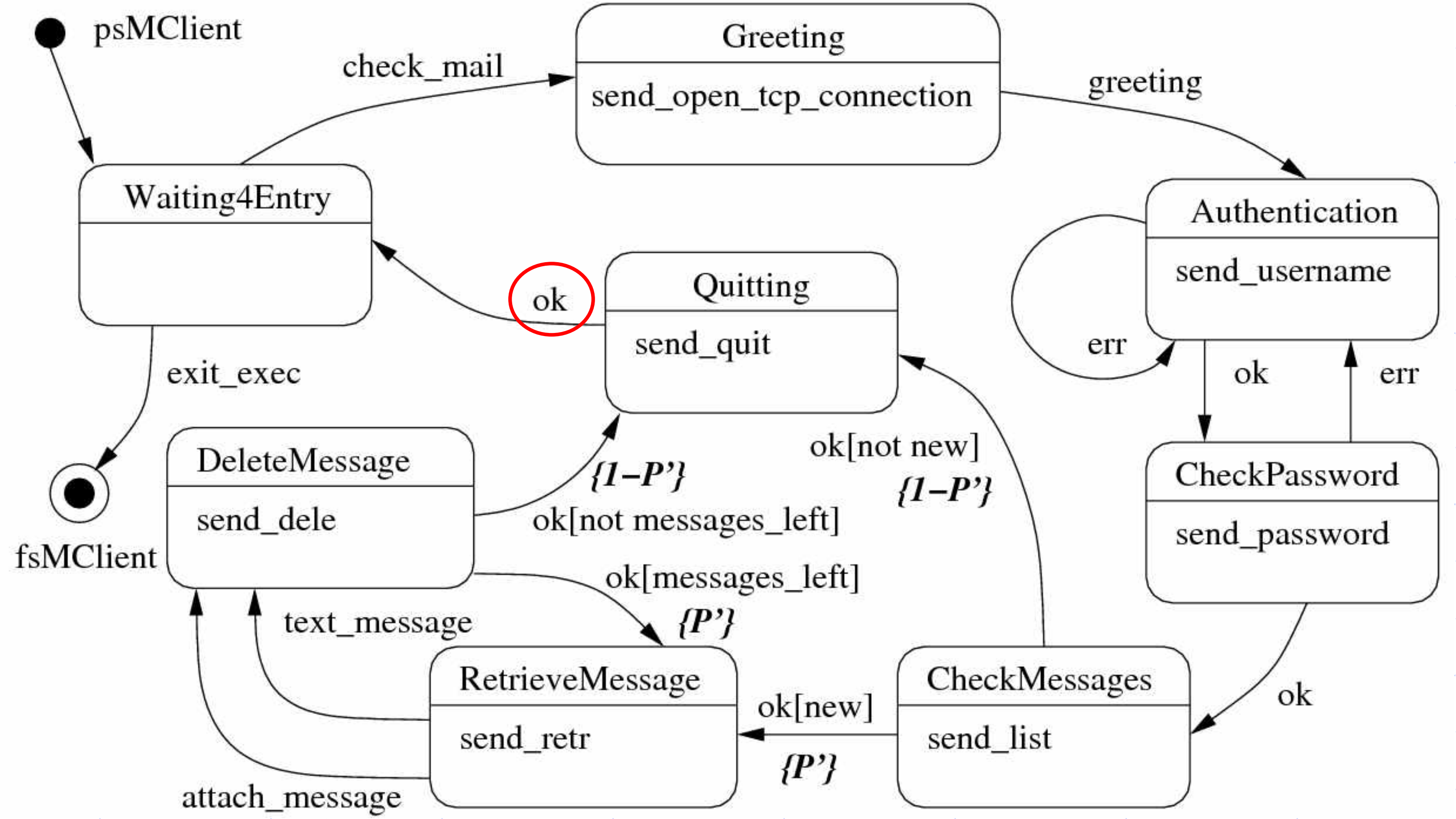
Integrating with Petri nets: case study

- Once all mails have been downloaded, interaction ends with QUIT command



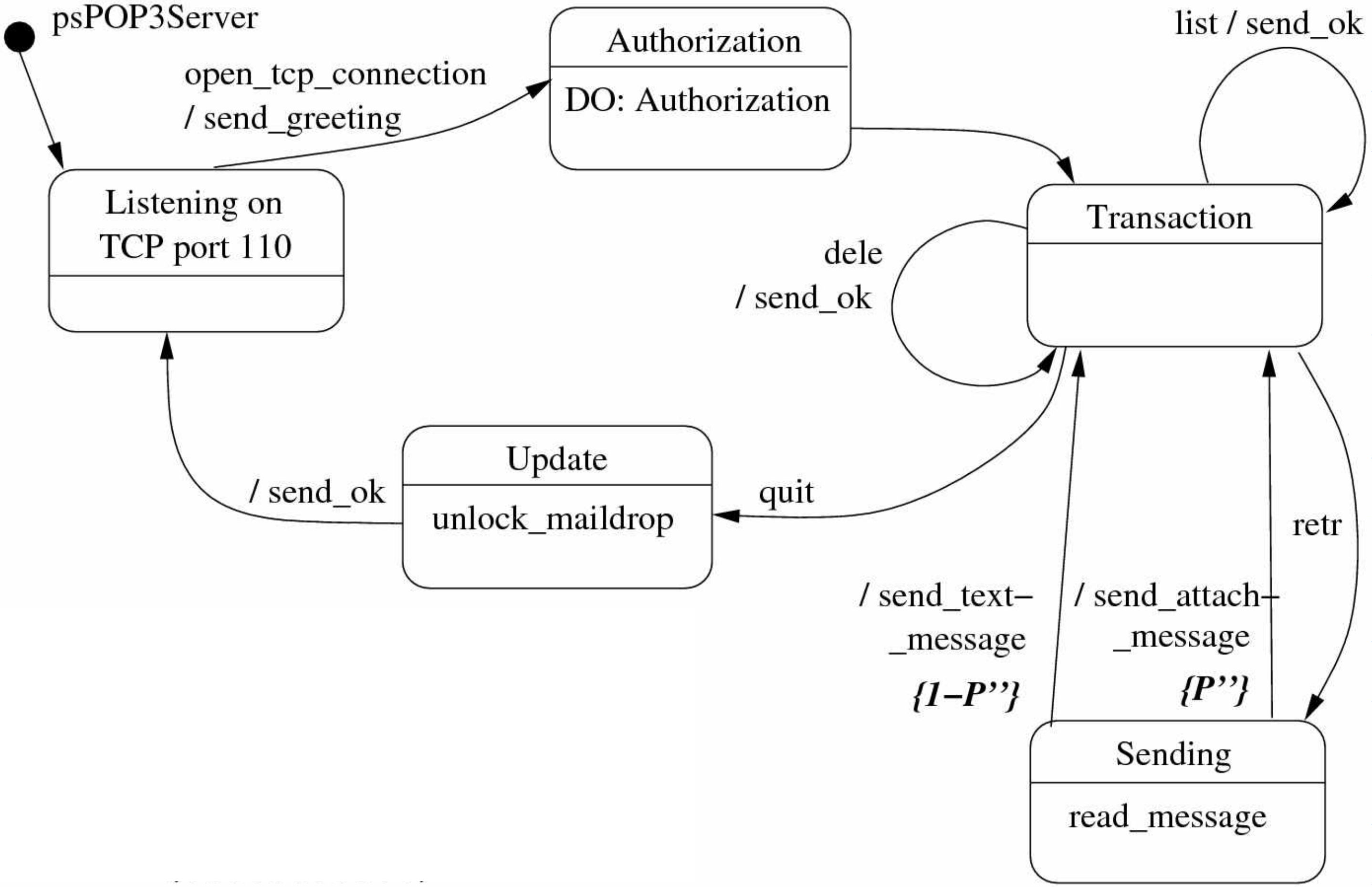
Integrating with Petri nets: case study

- The POP3 server enters the update state and releases acquired resources during transaction phase



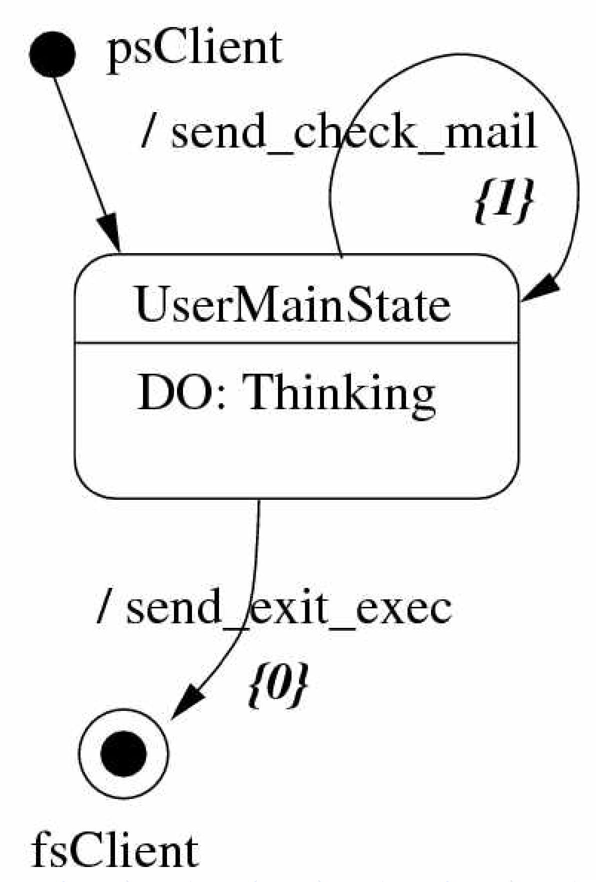
Integrating with Petri nets: case study

- Statechart for the class ServerHost: server behaviour



Integrating with Petri nets: case study

- Statechart for the actor User: user's behaviour

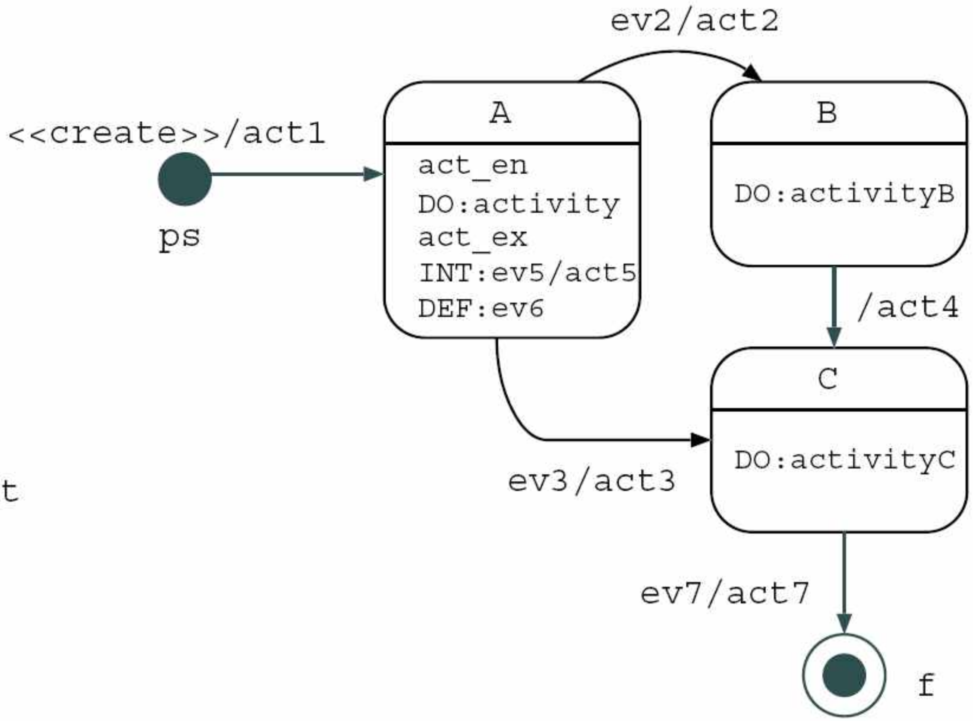


Integrating with Petri nets: case study

- Translation of statecharts to Labelled GSPN's
  - Compositional approach
  - From basic modelling elements of statecharts to LGSPN's
    - Initial and final states
    - Simple states (activities, entry and exit)
    - Transitions (internal and outgoing)
  - Translation:
    - input model (statechart element) → output model (LGSPN)
  - Composition of LGSPN's
    - Using a composition operator that fuses nodes with equal labels

Integrating with Petri nets: case study

- "Flat" UML statechart



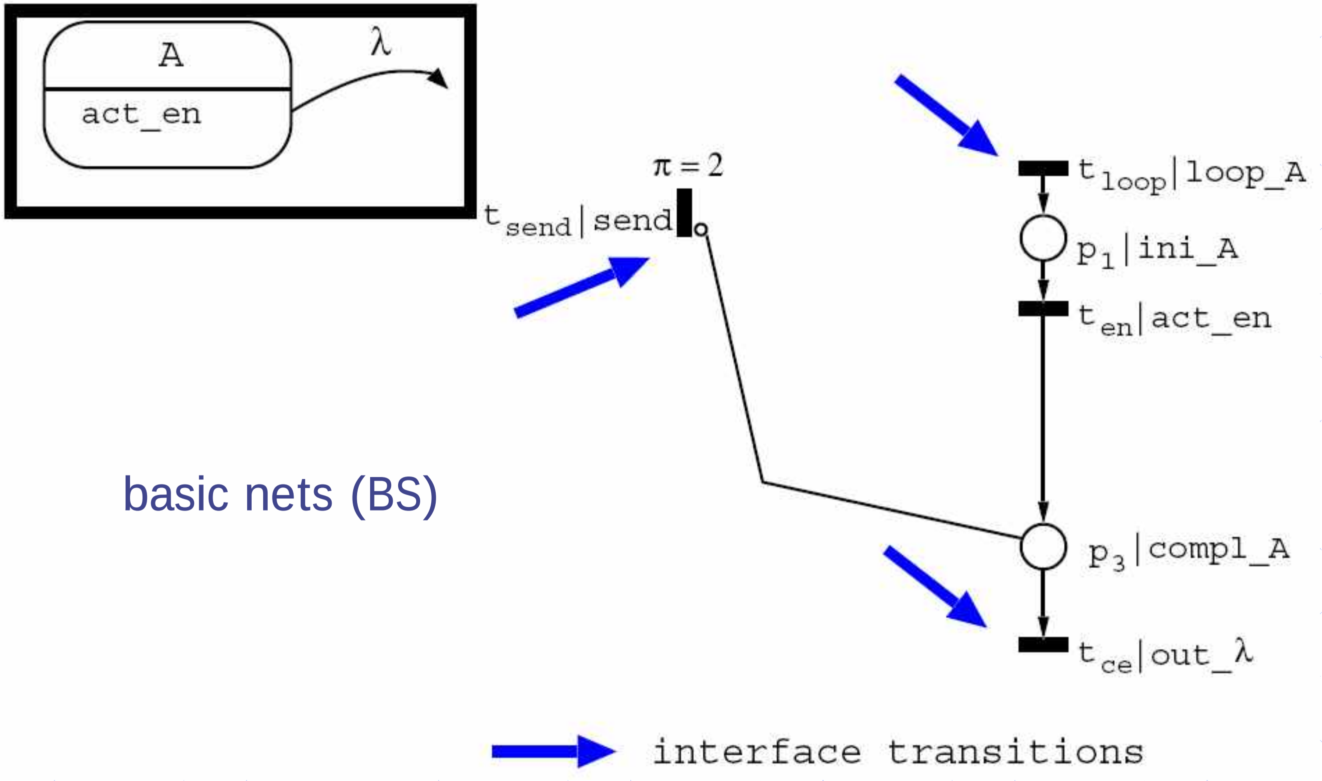
```
act_en = A.entry
activity = A.doActivity
act_ex = A.exit
{tr5} = A.internal
tr5.trigger = ev5
tr5.effect = act5
{ev6} = A.deferrableEvent

{tr1} = ps.outgoing
{tr4} = B.outgoing
tr4.trigger = 0
tr4.effect = act4
```

Integrating with Petri nets: case study

- Each simple state is modelled by a LGSPN representing the basic elements of states and transitions...

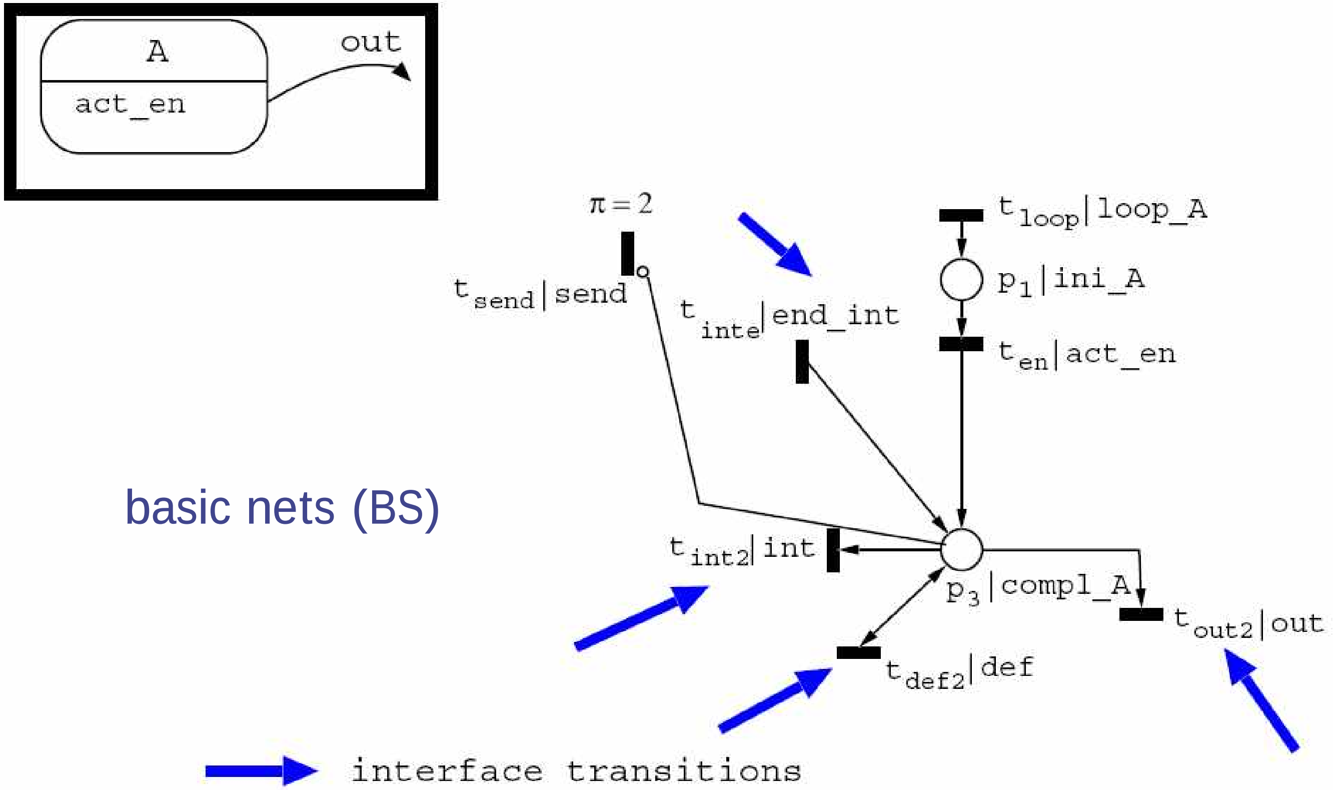
Simple state with no activity and immediate outgoing transition



Integrating with Petri nets: case study

- Each simple state is modelled by a LGSPN representing the basic elements of states and transitions...

Simple state with no activity and no immediate outgoing transition

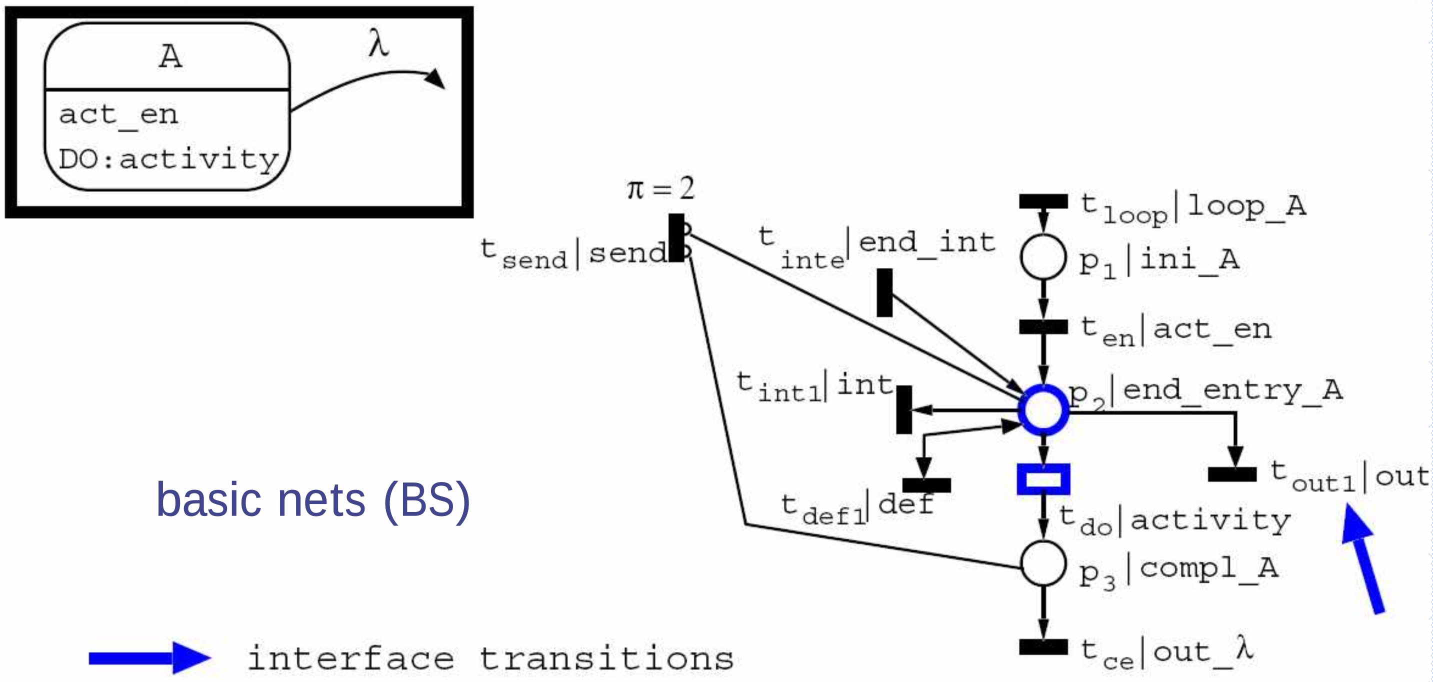


basic nets (BS)

Integrating with Petri nets: case study

- Each simple state is modelled by a LGSPN representing the basic elements of states and transitions...

Simple state with activity and immediate outgoing transition



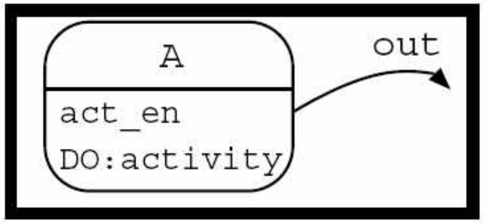
basic nets (BS)

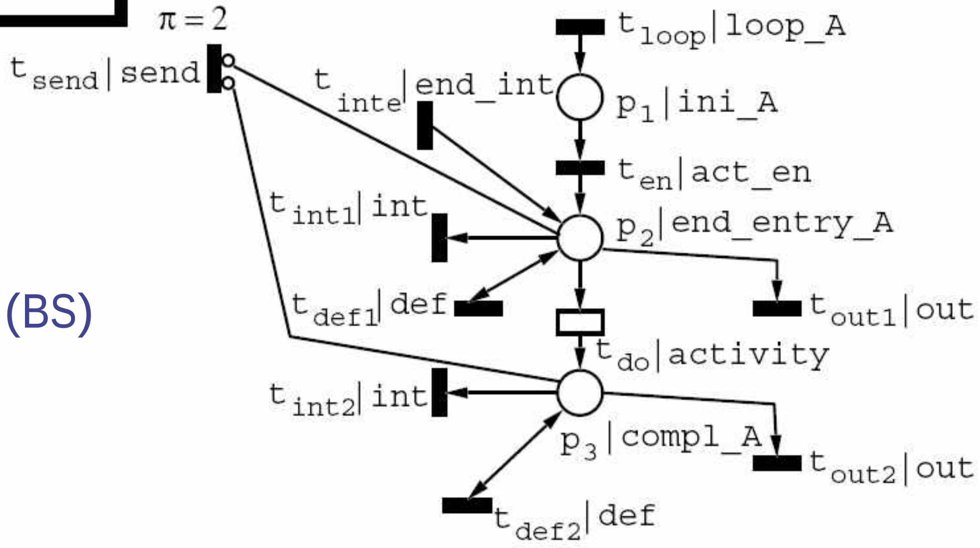
interface transitions

Integrating with Petri nets: case study

- ❑ Each simple state is modelled by a LGSPN representing the basic elements of states and transitions...

Simple state with activity and no immediate outgoing transition

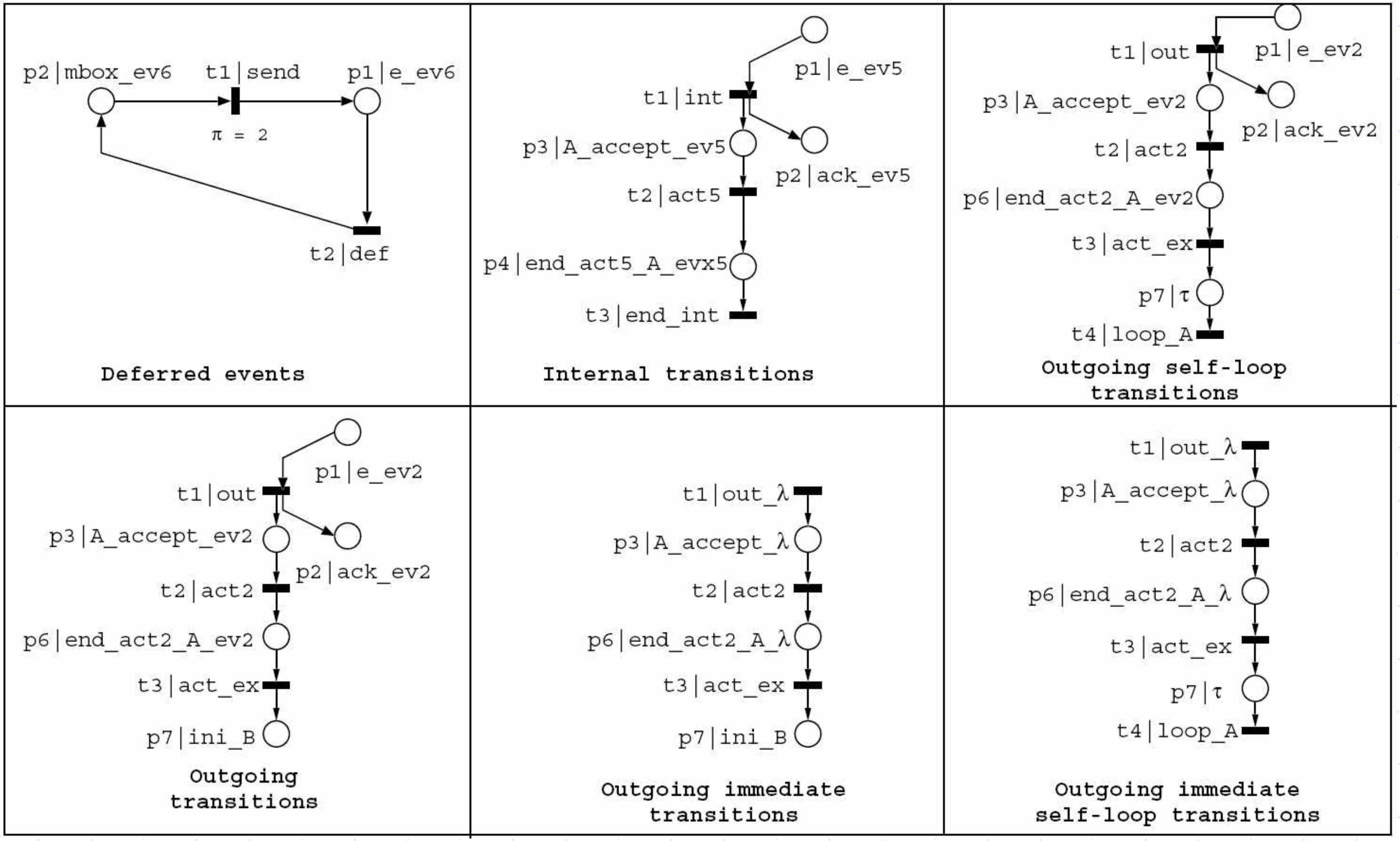




basic nets (BS)

Integrating with Petri nets: case study

Translation of other elements: Deferred events, internal transitions, outgoing transitions



Deferred events

Internal transitions

Outgoing self-loop transitions

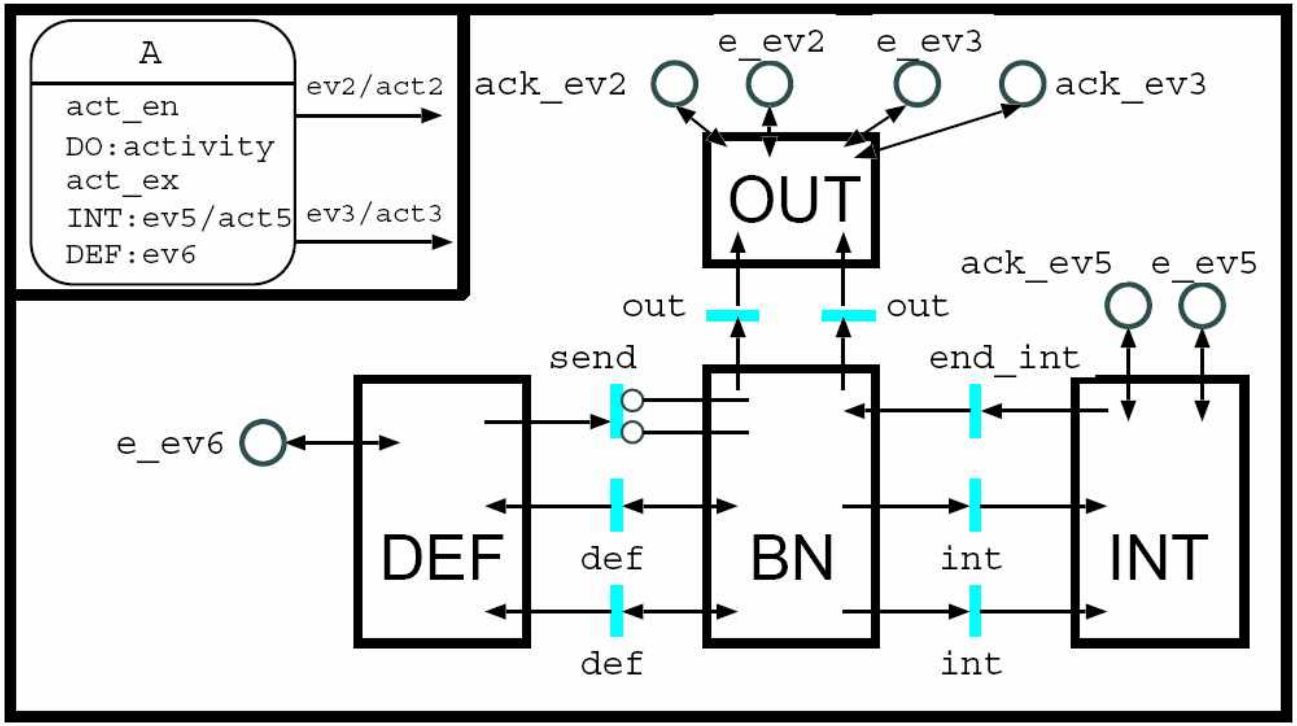
Outgoing transitions

Outgoing immediate transitions

Outgoing immediate self-loop transitions

Integrating with Petri nets: case study

- ❑ Composing the simple state...



$$\mathcal{LS}_s = (((INT \underset{Lev^P}{||} DEF) \underset{Lev^P}{||} OUT\_S) \underset{Lev^P}{||} OUT) \underset{Lev^P,Ltr^T}{||} BN$$

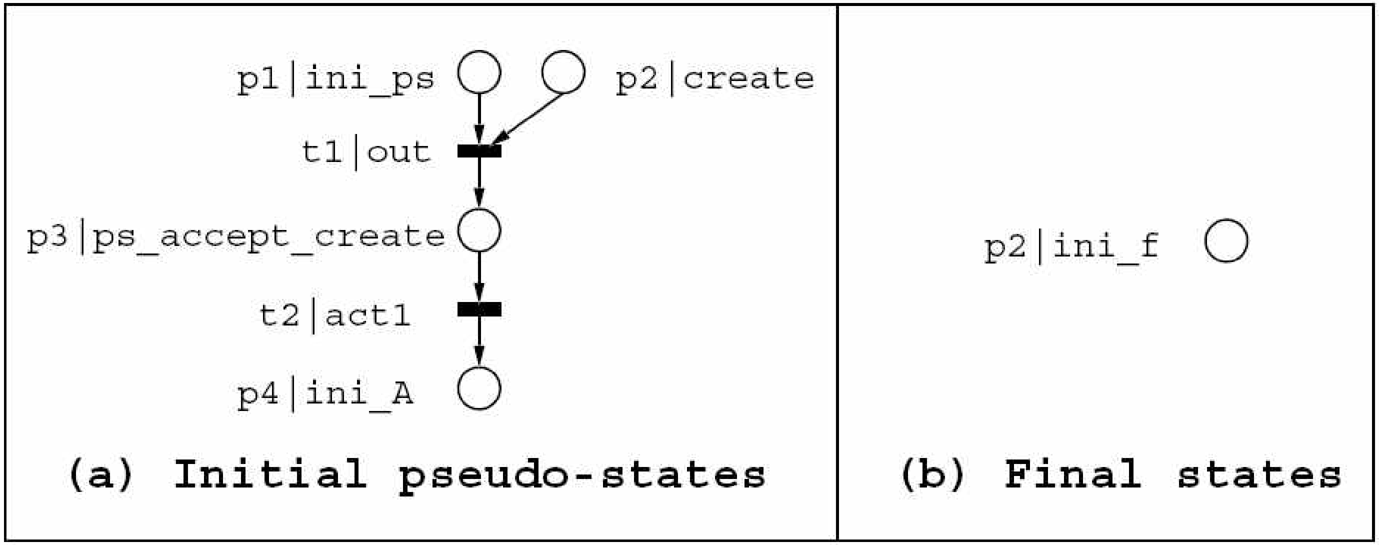
Integrating with Petri nets: case study

- Translation of other elements ("non-flat" SC)
  - Composite states,
  - concurrent states,
  - submachine states,
  - fork and join,
  - junction and choice,
  - synchronous states...

→ Details in the literature

Integrating with Petri nets: case study

- The same for the initial pseudo-states and final states

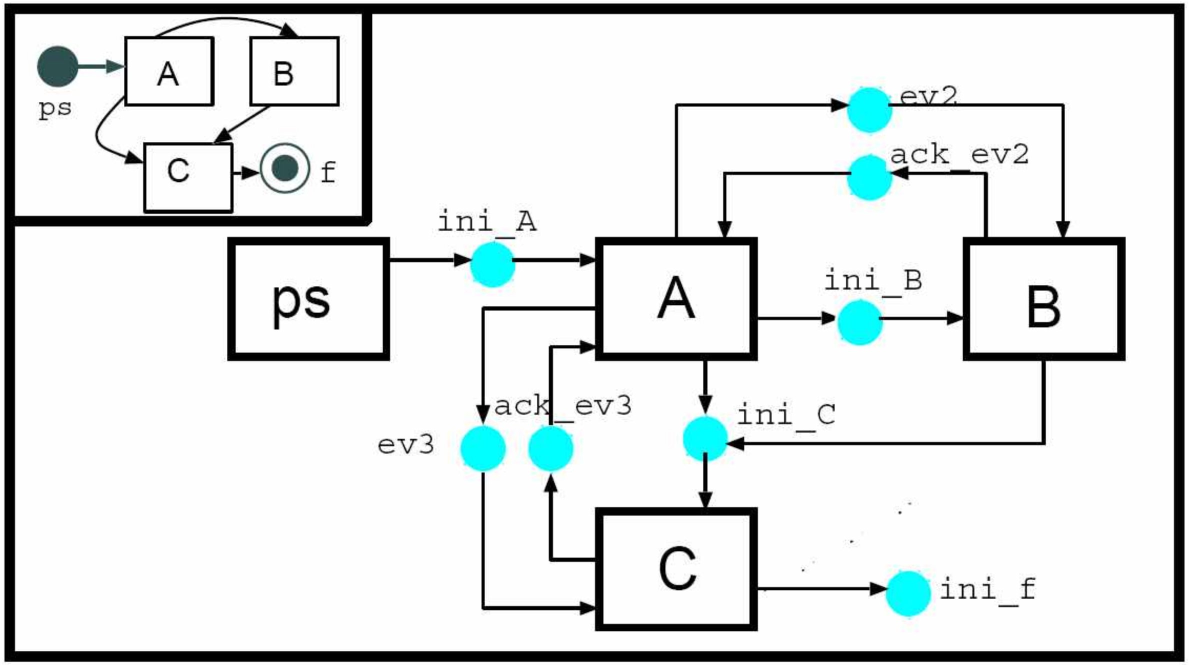


(a) Initial pseudo-states

(b) Final states

Integrating with Petri nets: case study

- The LGSPN model of the Statechart is the composition of all simple states, and initial and final states

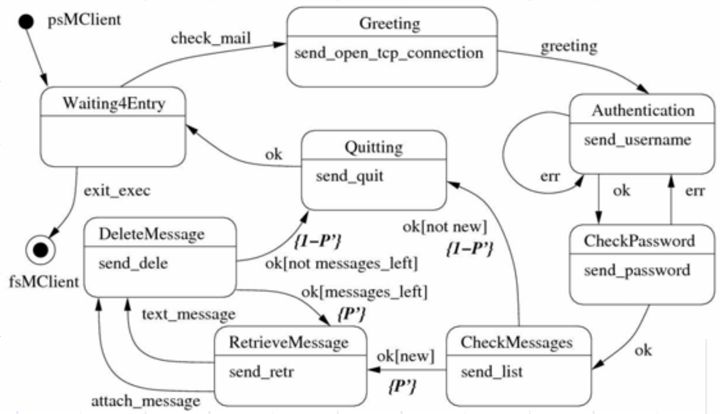


$$\mathcal{LS}_{sm} = (((\mathcal{LS}_{ps} \,||\, \mathcal{LS}_A) \,||\, \mathcal{LS}_B) \,||\, \mathcal{LS}_C) \,||\, \mathcal{LS}_f$$

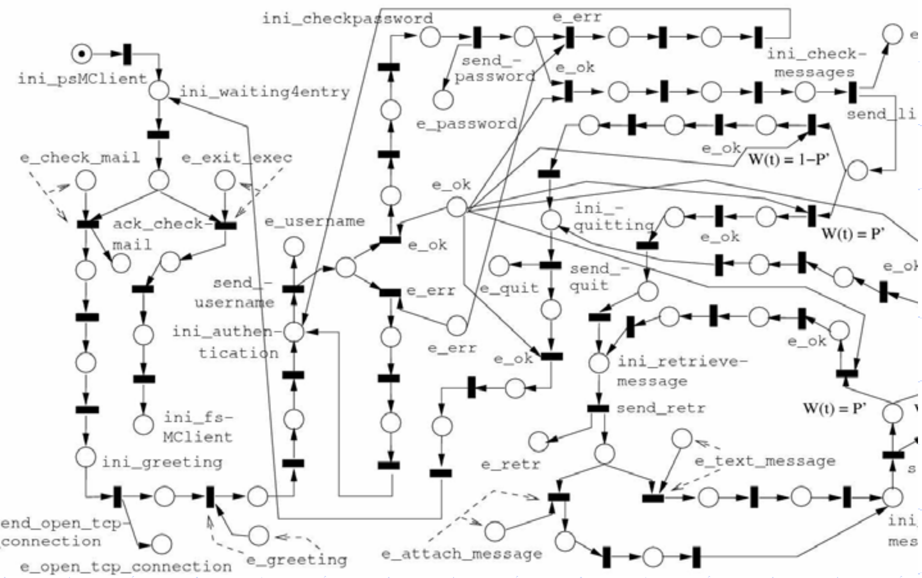
- If there are several Statecharts → composition of all of them

Integrating with Petri nets: case study

- ❑ Coming back to the mail example...

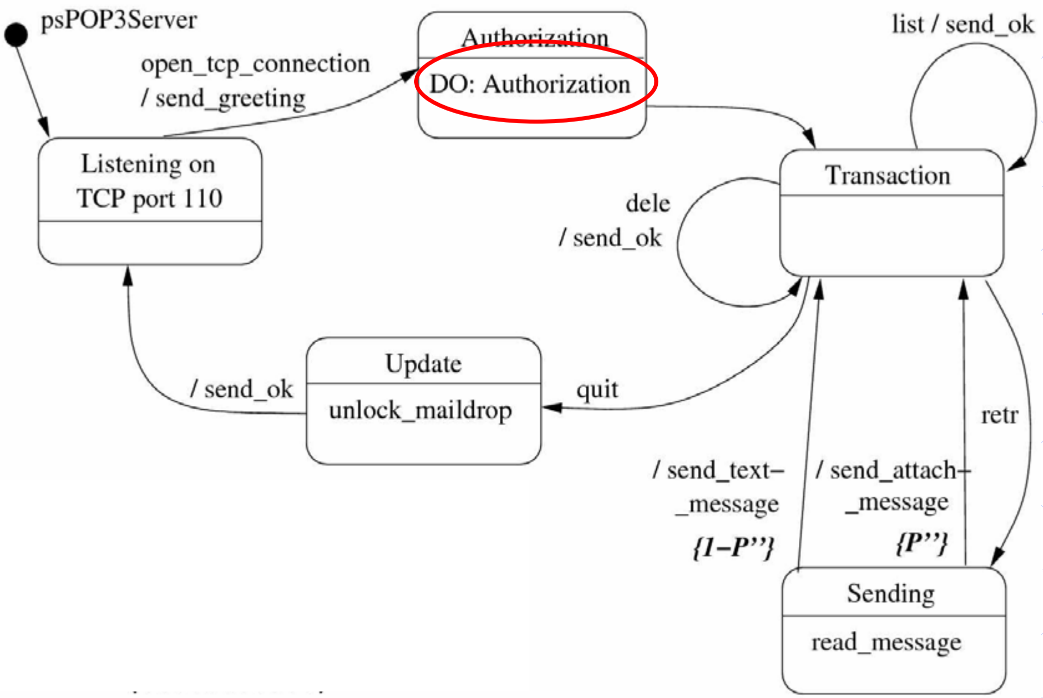


Statechart for the class ClientHost: client behaviour



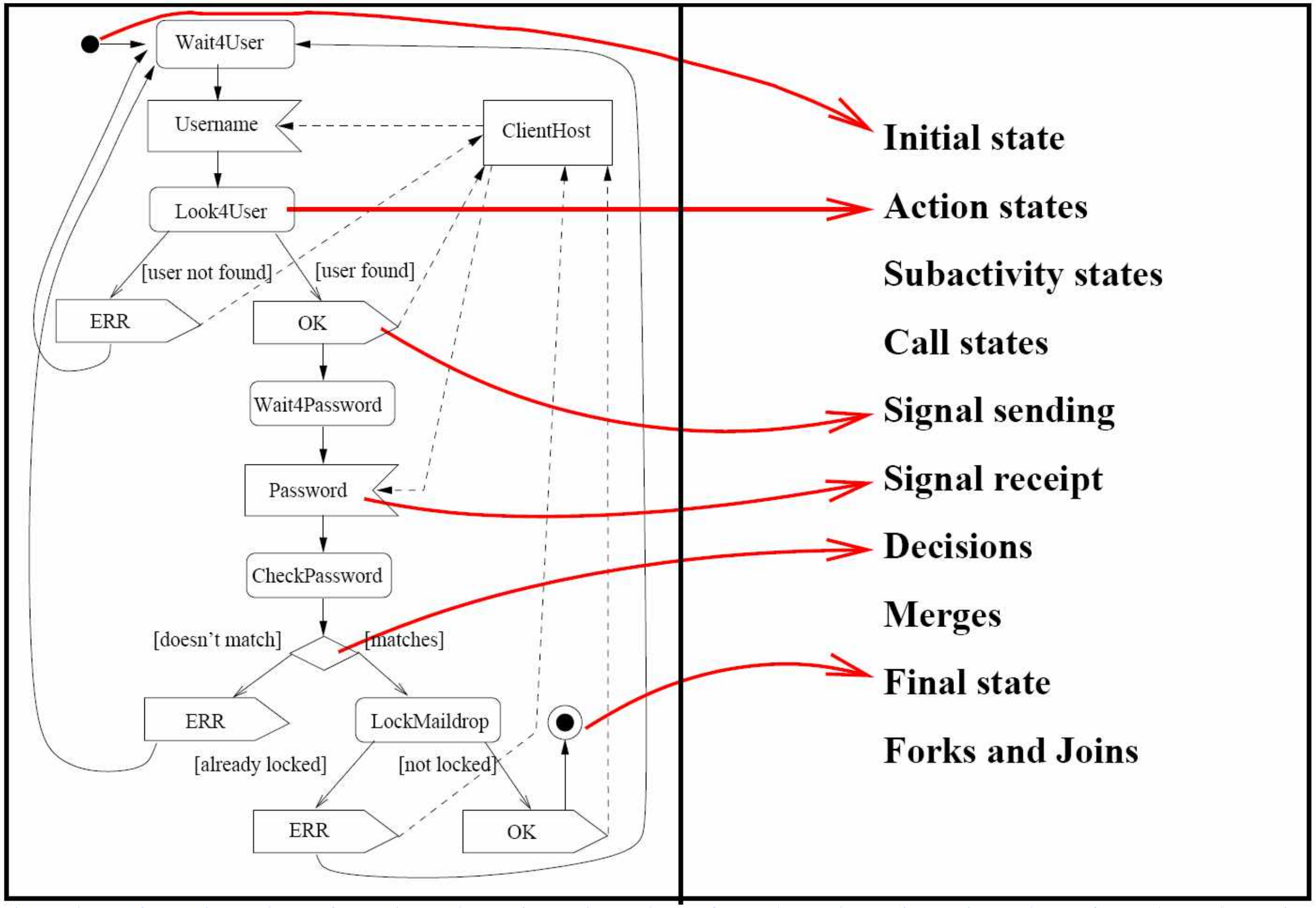
Integrating with Petri nets: case study

- ❑ In the behaviour of Serverhost
  - ❑ We decide to describe more in detail activity associated to state Authorization using an Activity Diagram



Integrating with Petri nets: case study

- ❑ Refinement of Authorization with an AD



Initial state

Action states

Subactivity states

Call states

Signal sending

Signal receipt

Decisions

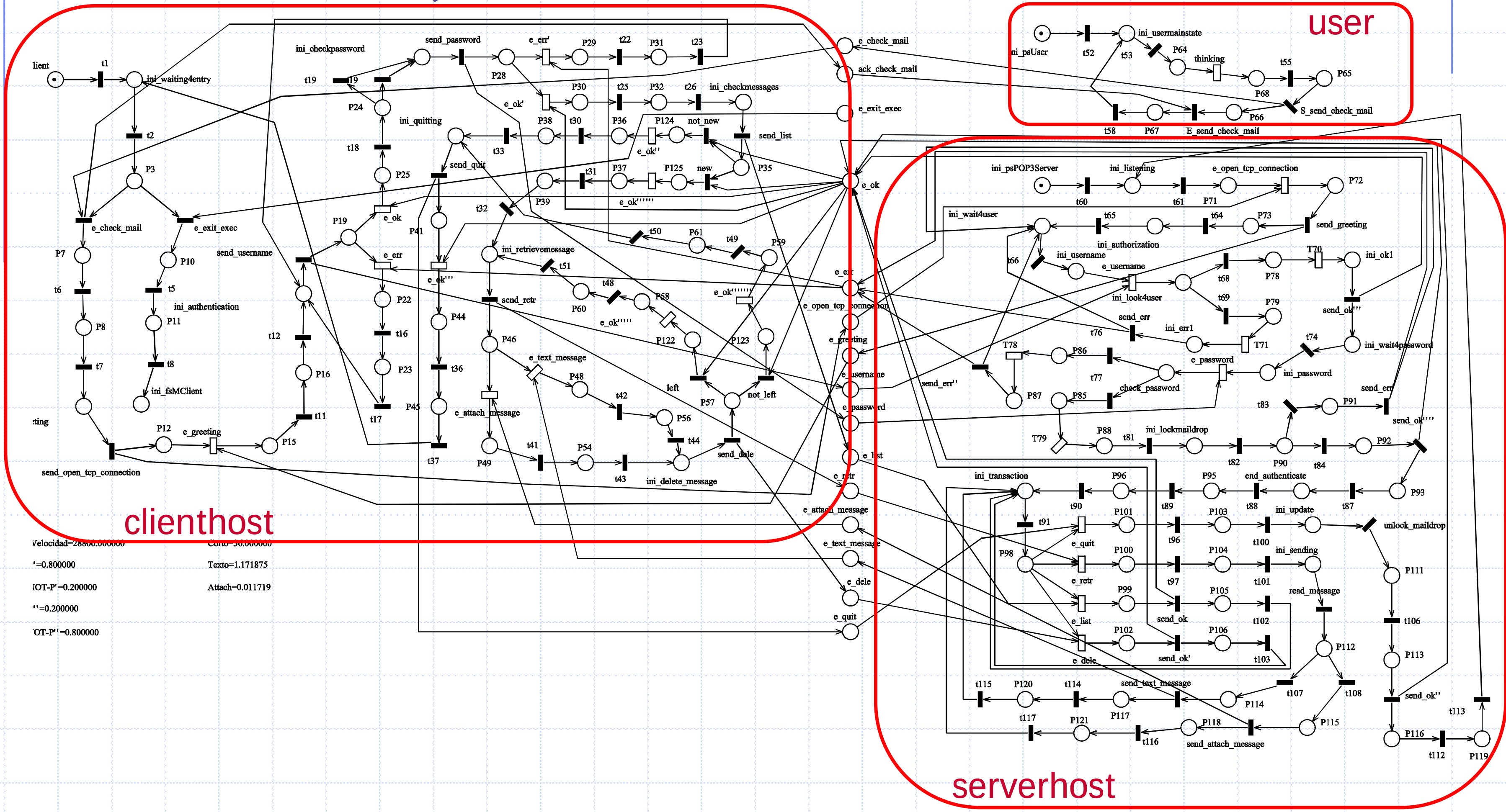
Merges

Final state

Forks and Joins

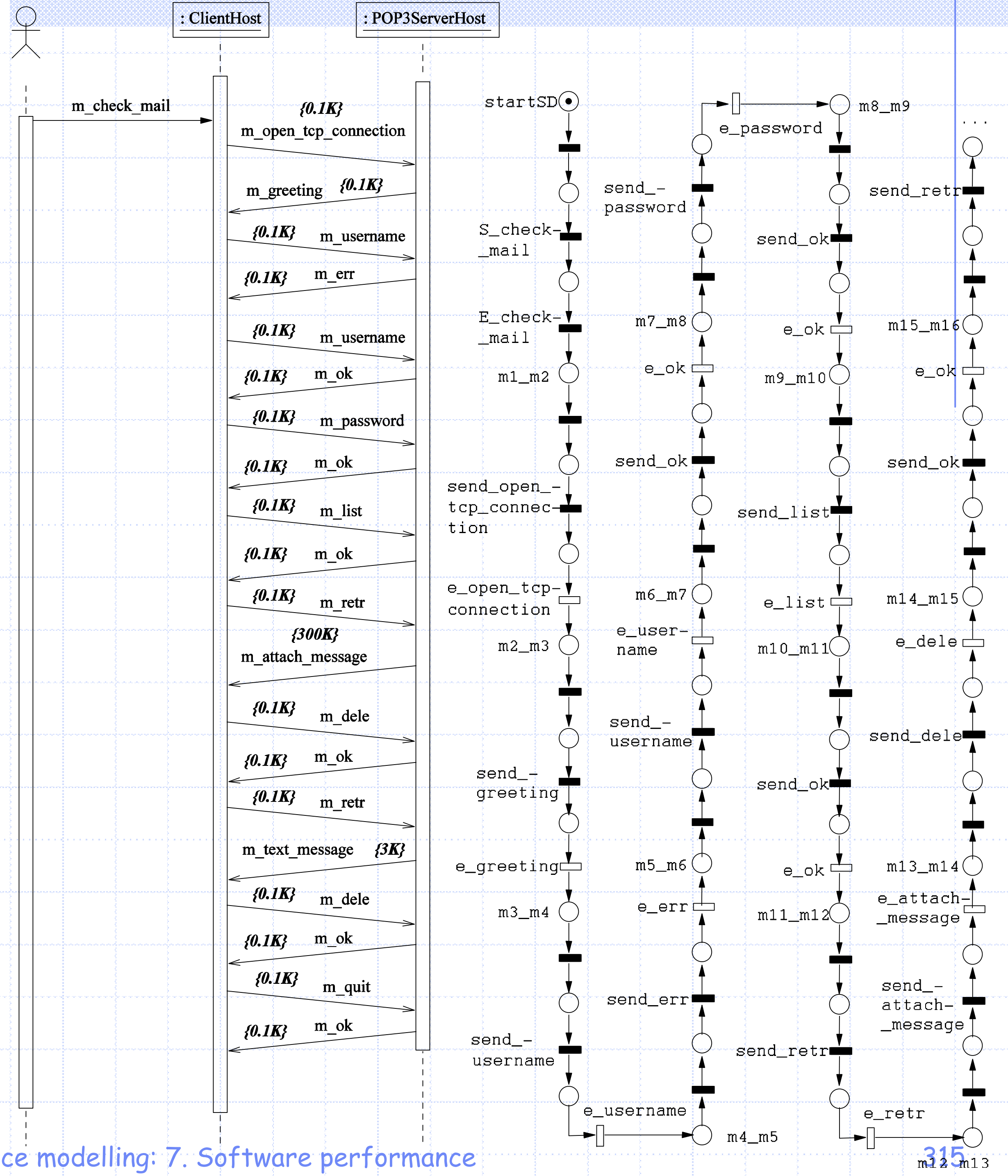
Integrating with Petri nets: case study

- Statecharts and activity diagrams all together (clienthost, serverhost, user)



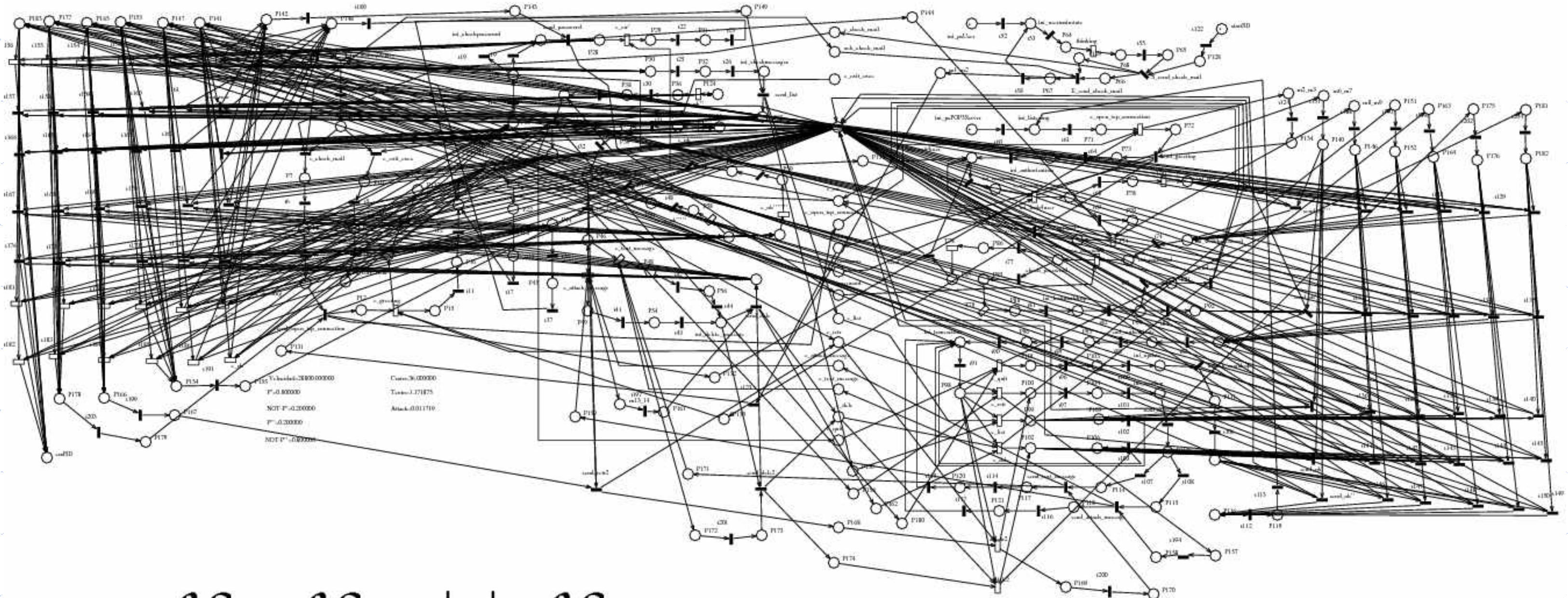
...: case study

- Finally, Sequence Diagram
  - Represents a particular scenario of execution
  - Example of interaction between clienthost and serverhost



Integrating with Petri nets: case study

- ❑ Superposition of Statecharts, Activity Diagrams and Sequence Diagram → analysable model of the concrete execution



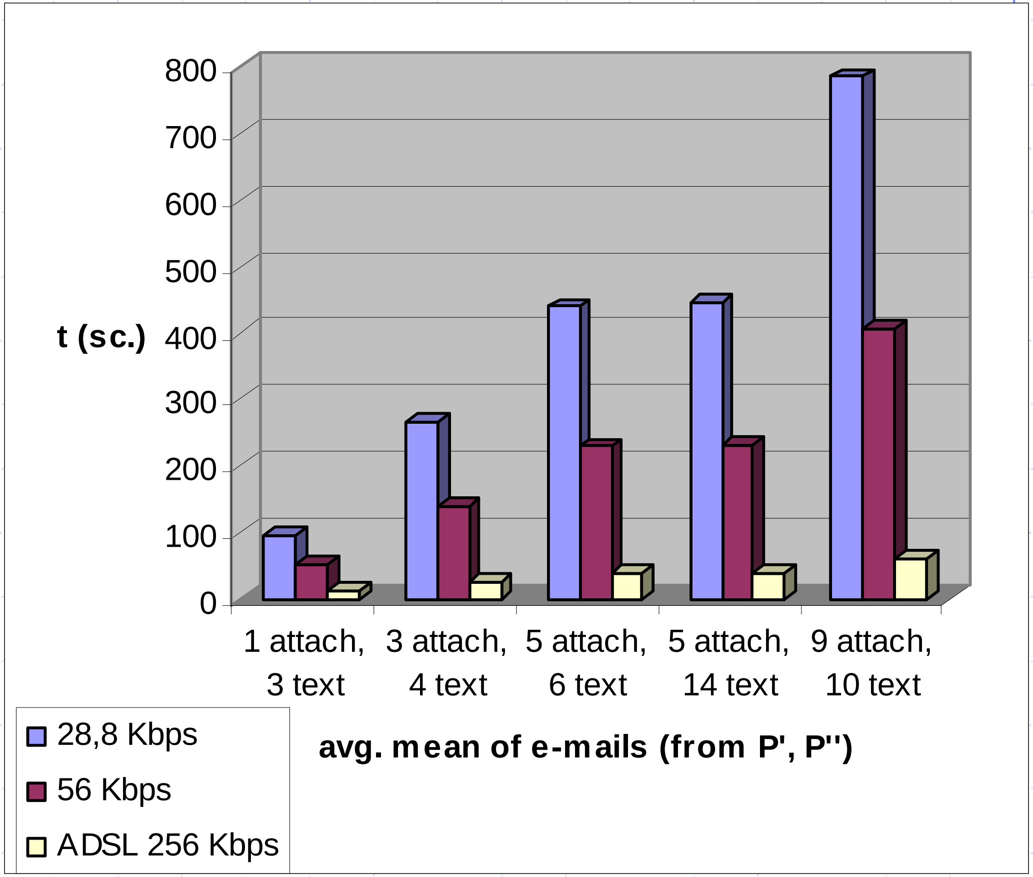
$$\mathcal{LS} = \mathcal{LS}_{sc} \underset{Lev_T,\emptyset}{||} \mathcal{LS}_{sdi}$$

Outline

- ❑ Software Performance Engineering: basics
- ❑ A Software Performance Process
- ❑ Annotated UML Diagrams
- ❑ Integrating with Petri nets: case study
- ❑ Performance analysis
- ❑ Automation of the approach
- ❑ Real example
- ❑ Conclusions
- ❑ Bibliography

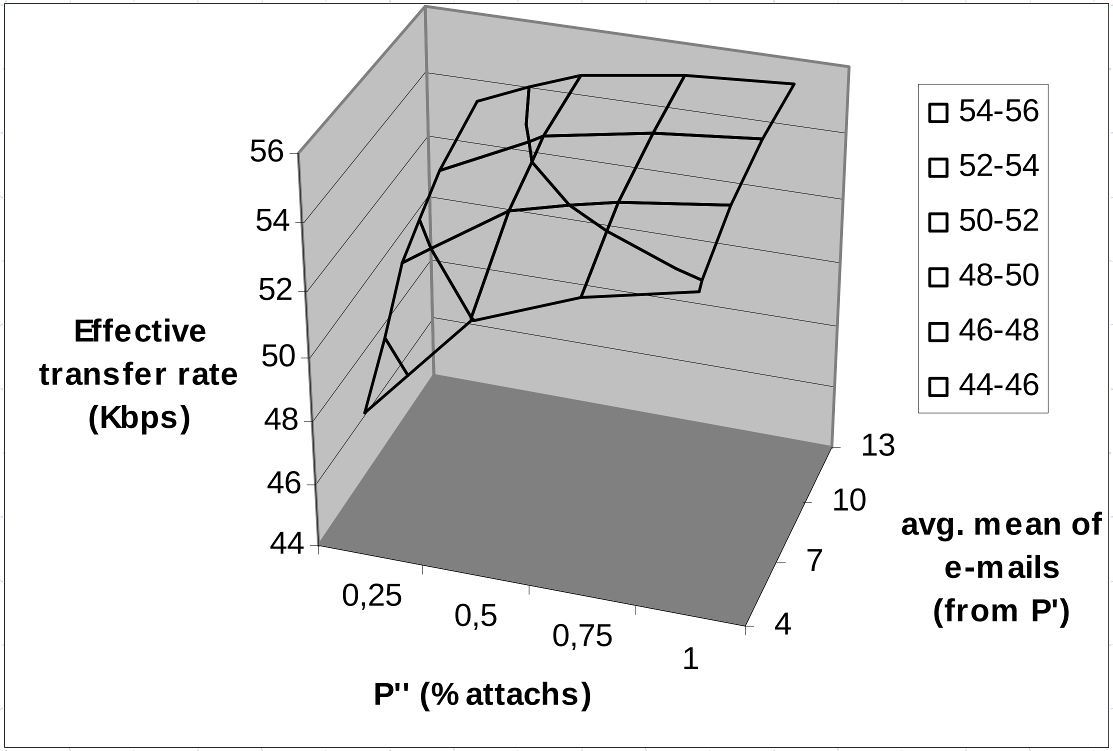
Performance analysis

- Effect on the downloading time for different connection speeds of
  - number of mails
  - proportion of them with attached files



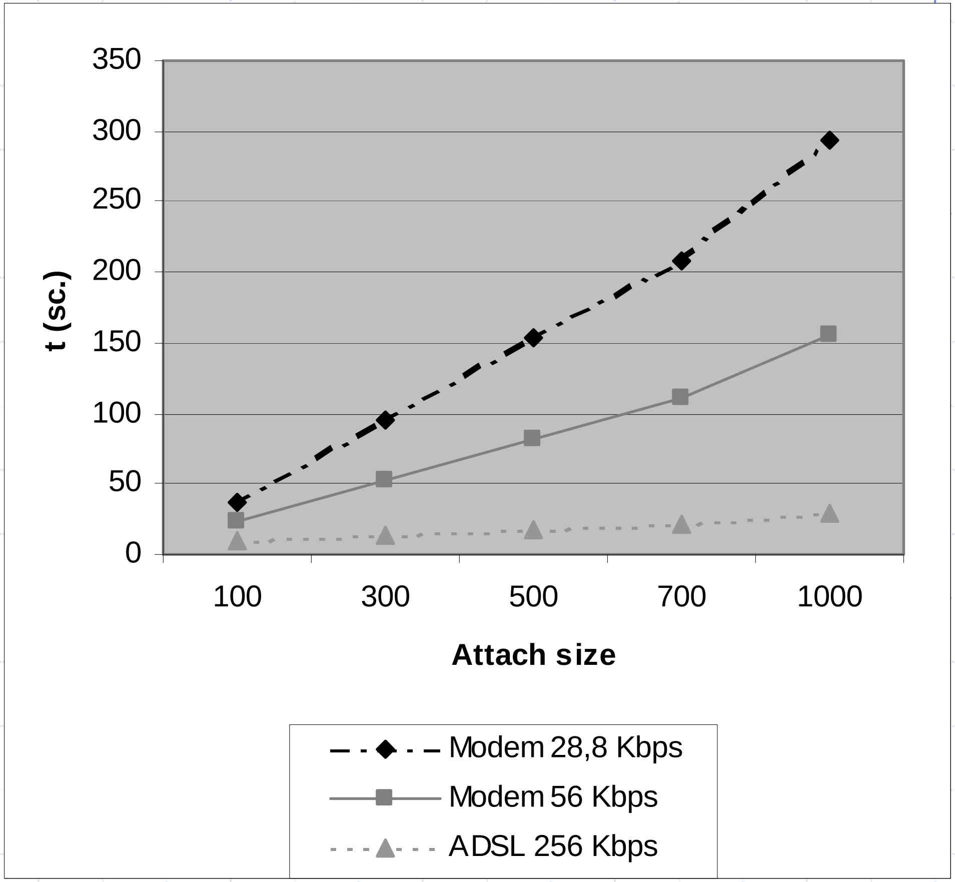
Performance analysis

- Effective transfer rate of the client (connection speed 56 Kbps)
  - Higher amount of data minimizes the relative amount of time spent by protocol messages



Performance analysis

- Execution time of the SD scenario varying
  - Attach files sizes
  - Network speed



Outline

- ❑ Software Performance Engineering: basics
- ❑ A Software Performance Process
- ❑ Annotated UML Diagrams
- ❑ Integrating with Petri nets: case study
- ❑ Performance analysis
- ❑ Automation of the approach
- ❑ Real example
- ❑ Conclusions
- ❑ Bibliography

Automation of the approach

- "A key factor in the successful application of early performance analysis is automation."
- "Characterizes the maturity of the approach and the generality of its applicability."

ArgoSPE:

A Software Performance Engineering Tool

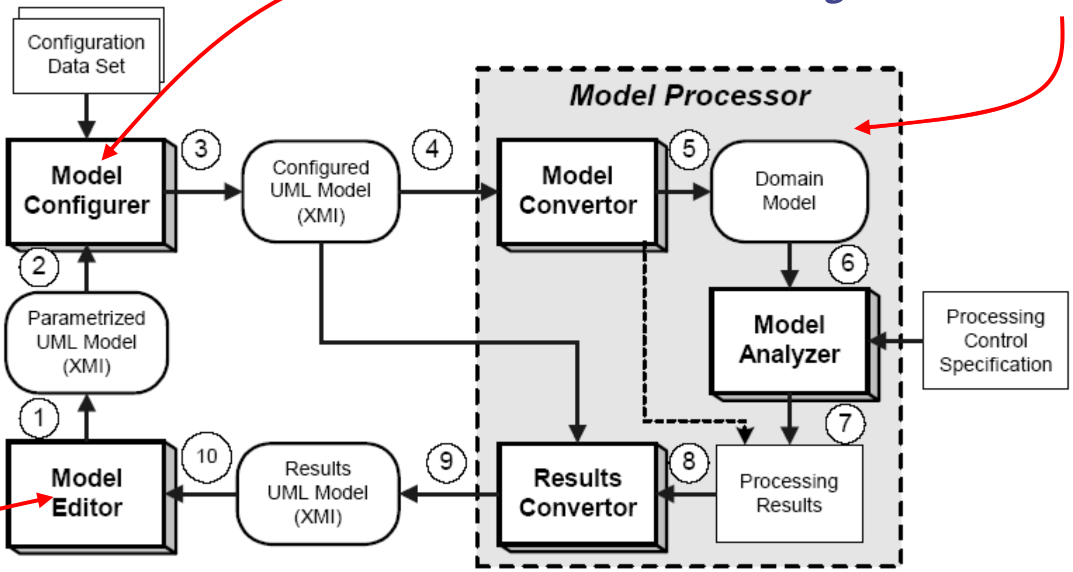
Automation of the approach

- ArgoSPE
  - Implements most of the features explained in this talk and some others
  - The system is modeled as a set of UML diagrams
  - Annotated according to the UML Profile on schedulability, performance and time specification
    - Activity durations, routing probabilities, message sizes, network speed, population, initial state, resident classes
  - Performance queries are defined on UML diagrams
    - State population, stay time, message delay, network delay, response time
  - Translated into GSPN

Automation of the approach

- ❑ Architecture of ArgoSPE:
  - ❑ Follows the architectural framework proposed in UML-SPT

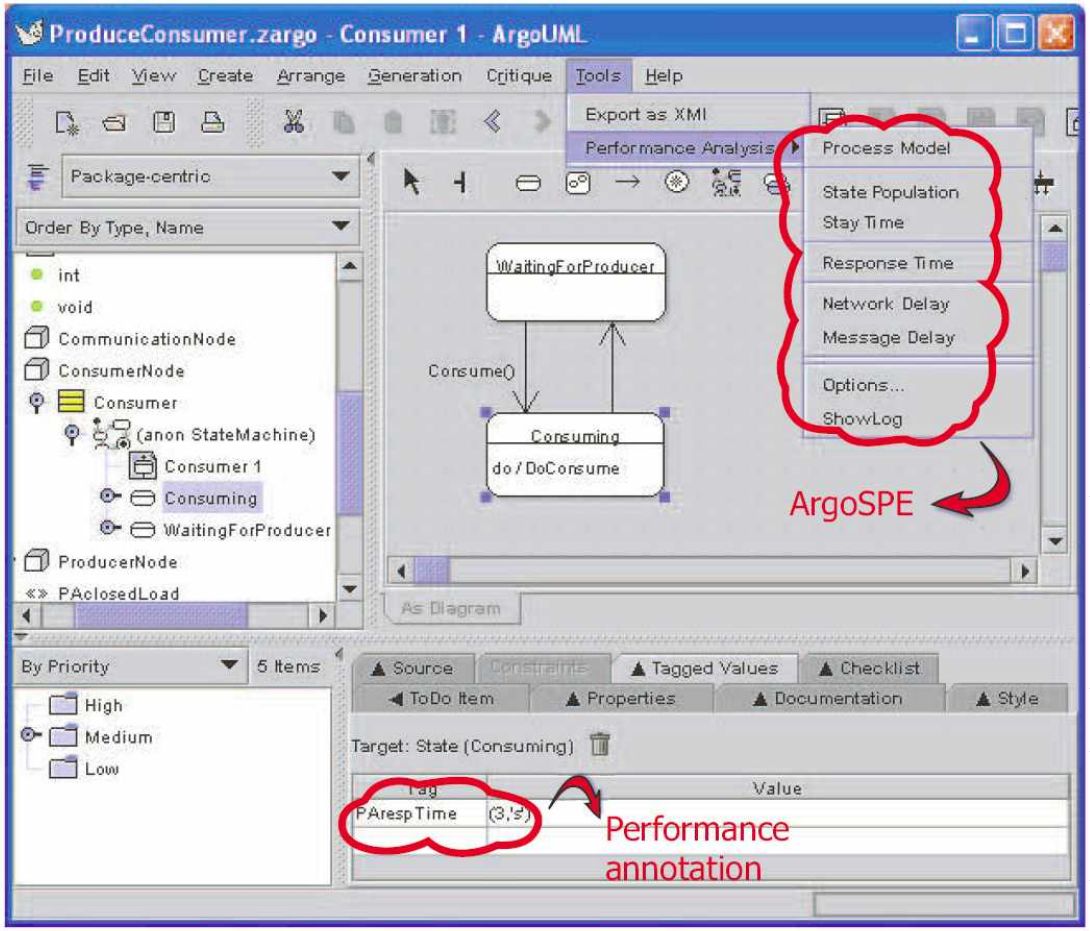
ArgoSPE modules



ArgoUML CASE tool

Automation of the approach

- ArgoSPE menu integrated in ArgoUML editor



Target: State (Consuming)

Tag	Value
PArespTime	(3,'s')

ArgoSPE

Performance annotation

Automation of the approach

- ❑ Details:
  - ❑ A tool paper presented in PN'06 Conference:

    "ArgoSPE: Model-based software performance evaluation"

    José Merseguer and Elena Gómez-Martínez

  - ❑ Tigris.org: Open Source Software Engineering Tools

    http://argospe.tigris.org

    download the tool, tool description, detailed user documentation, developer documentation, examples...

    free software available under GNU General Public License

Outline

- ❑ Software Performance Engineering: basics
- ❑ A Software Performance Process
- ❑ Annotated UML Diagrams
- ❑ Integrating with Petri nets: case study
- ❑ Performance analysis
- ❑ Automation of the approach
- ❑ Real example
- ❑ Conclusions
- ❑ Bibliography

Real example

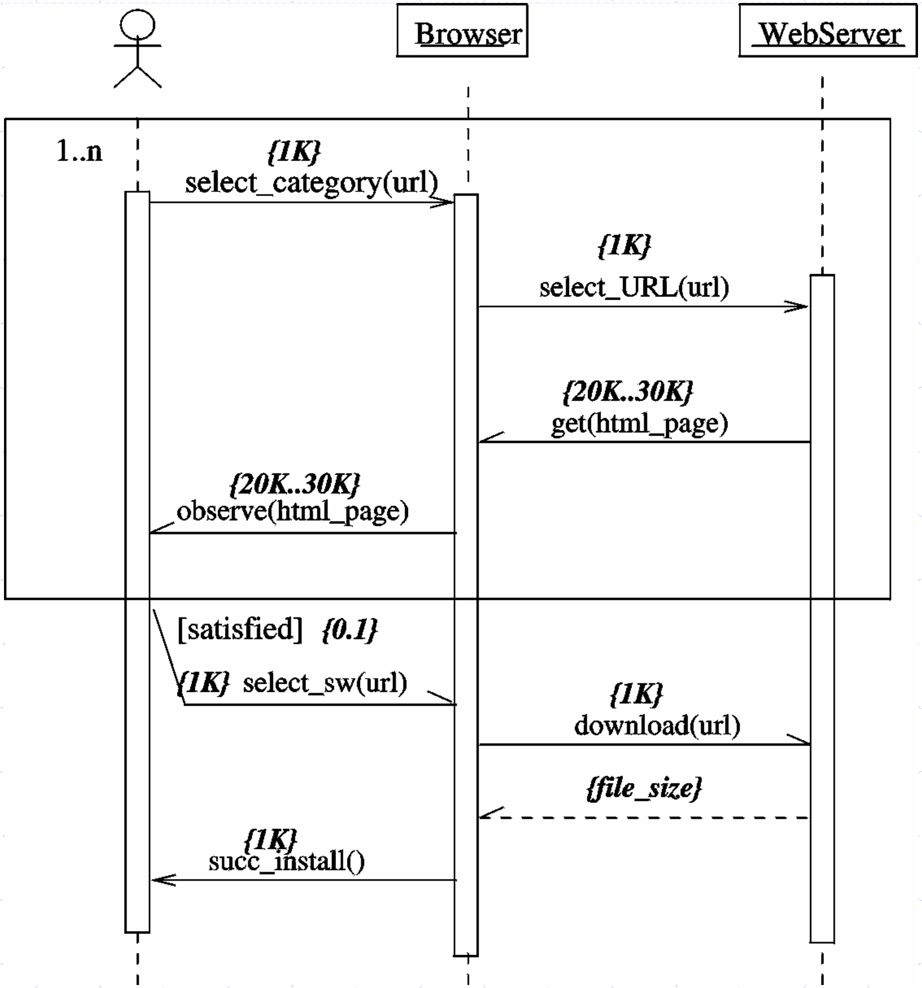
- ❑ Retrieving and installing software using internet in a mobile environment
- ❑ Usual solution: tucows-like (Tucows.com = the largest online software download site)
- ❑ SPE approach for a new mobile agent-based solution (Antarctica project of University of Vasque Country)
- ❑ Goal: compare performance indices of both solutions (before implementing Antarctica)
  - ❑ Minimize network connection time
  - ❑ Study the impact on performance of agents intelligence

Real example

- Steps:
  - Model both solutions using annotated UML diagrams
  - Generate PN performance models for both solutions
  - Analyze performance indices under different scenarios
  - Recommend the best choices and in which cases the use of the new mobile agent-based approach is preferable

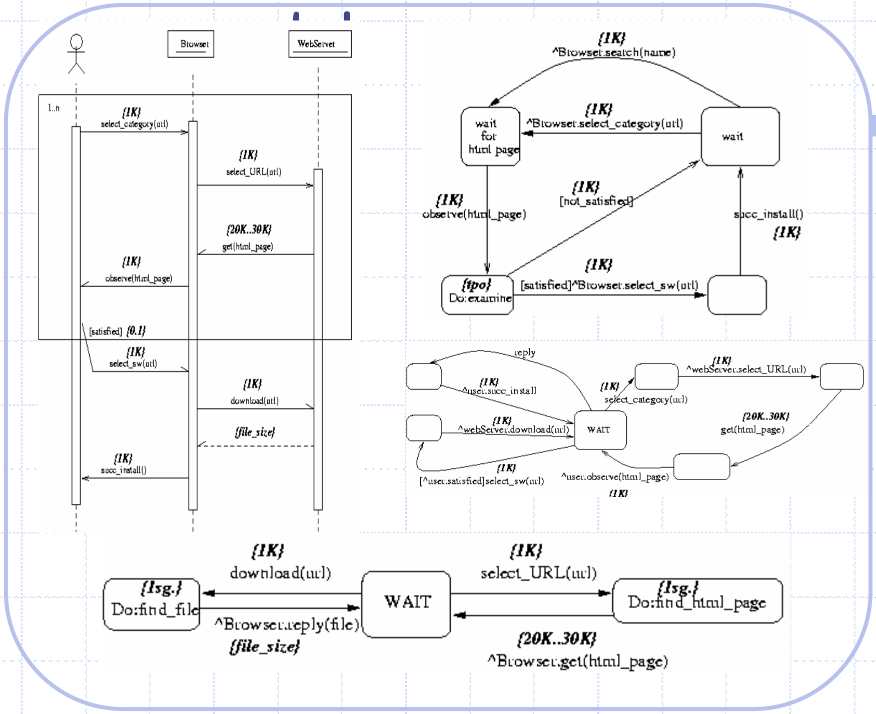
Real example

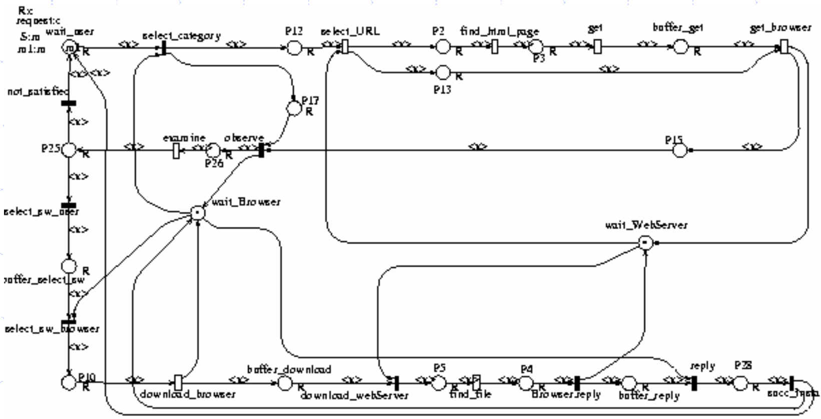
- ❑ Tucows-like approach
  - ❑ Sequence Diagram with durations and routing annotations



Real example

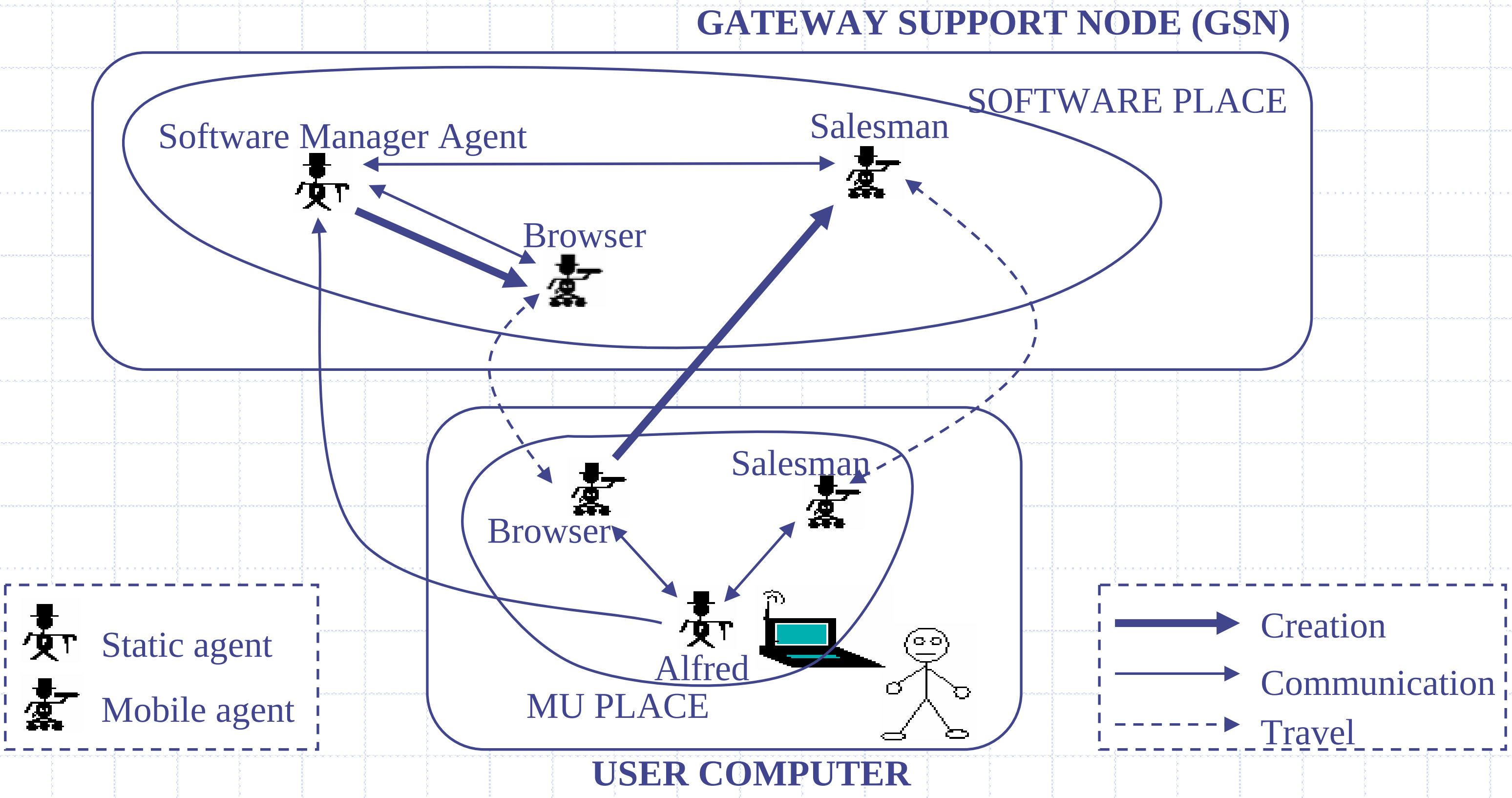
- □ Tucows-like approach (cont.)
  - □ Sequence Diagram + Statecharts →
    → performance





Real example

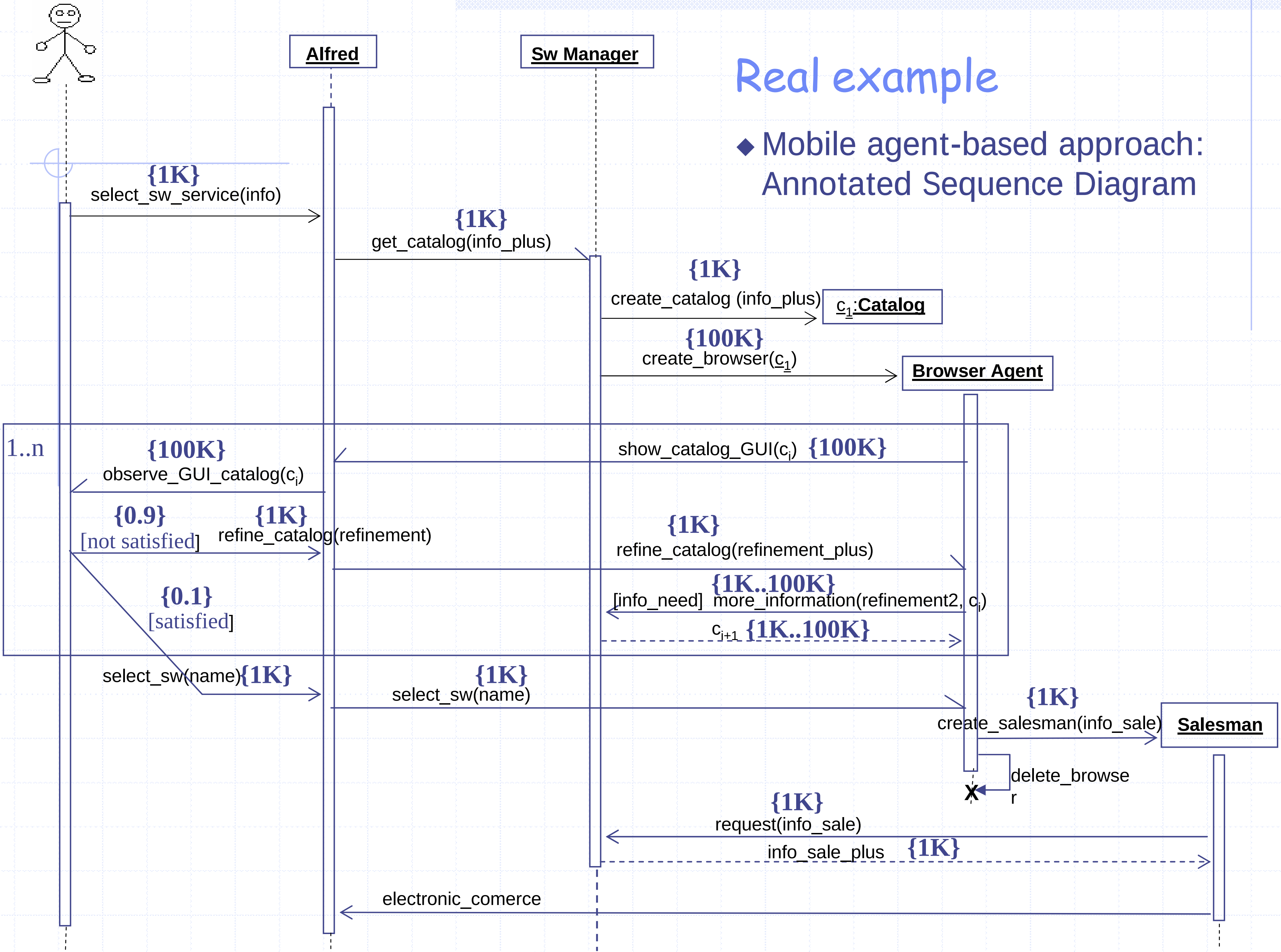
- Mobile agent-based approach: description



Alfred, the butler!

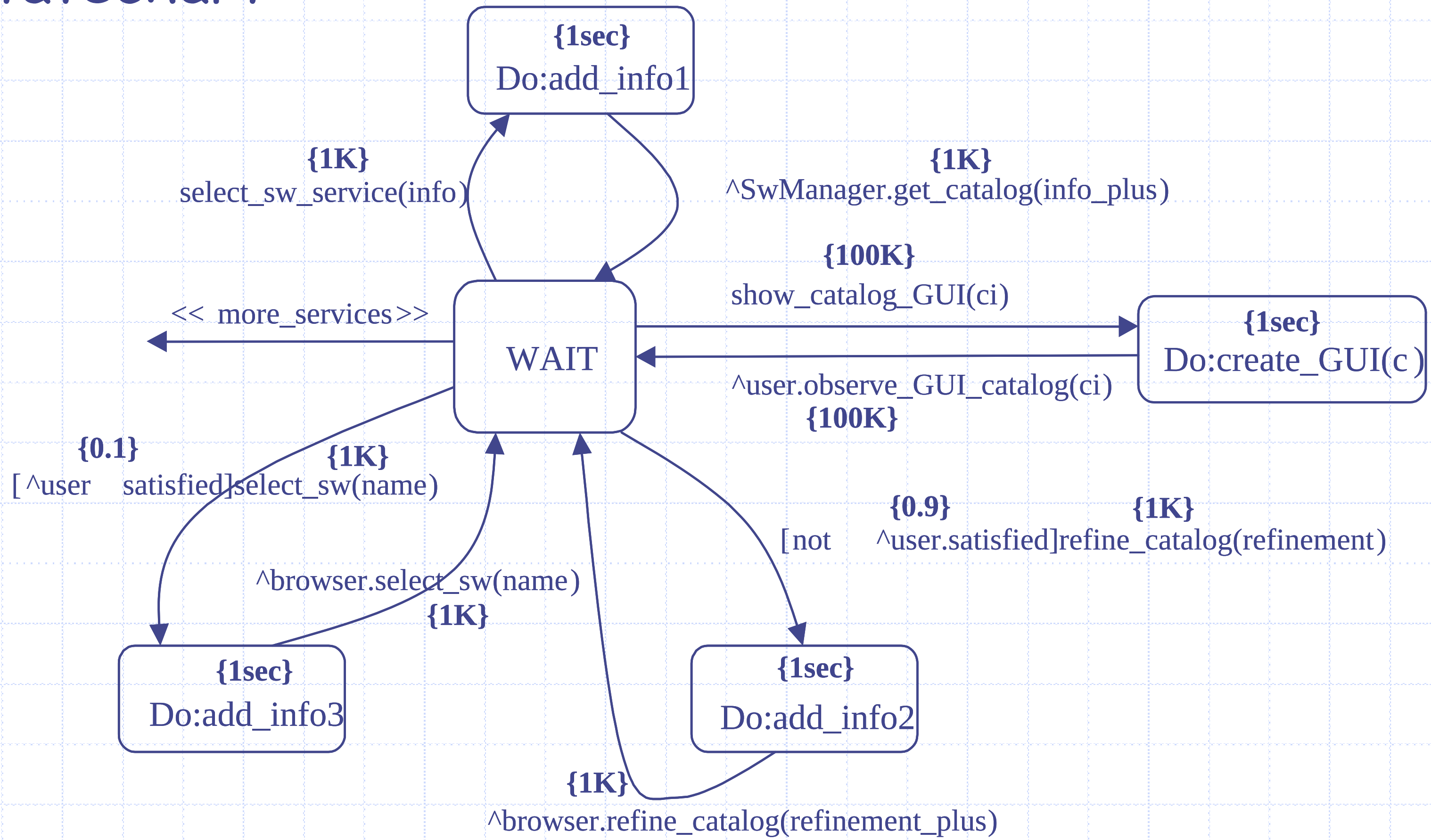
Real example

- Mobile agent-based approach: Annotated Sequence Diagram



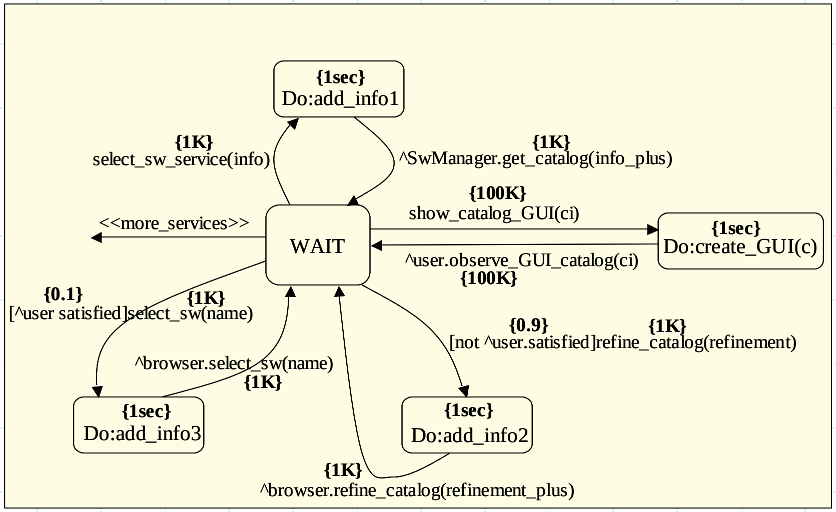
Real example

- Mobile agent-based approach: Alfred class Statechart

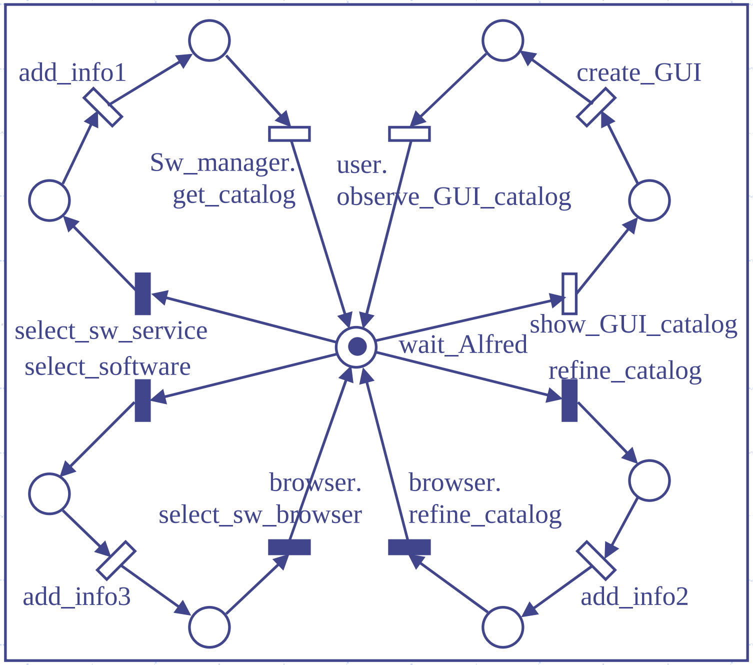


Real example

- Simplified version of the LGSPN component corresponding to Alfred

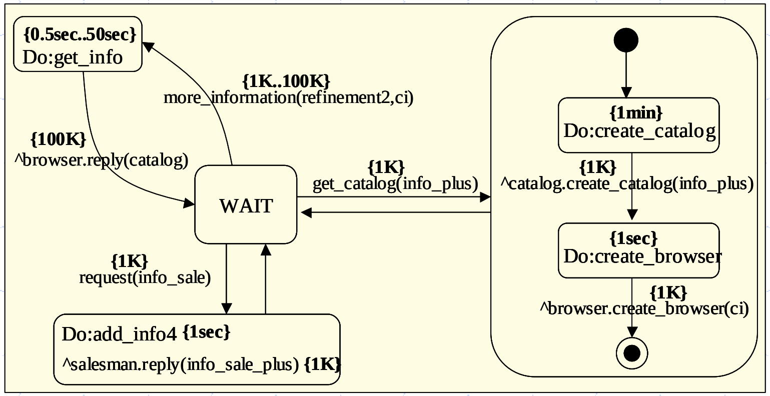


Alfred agent

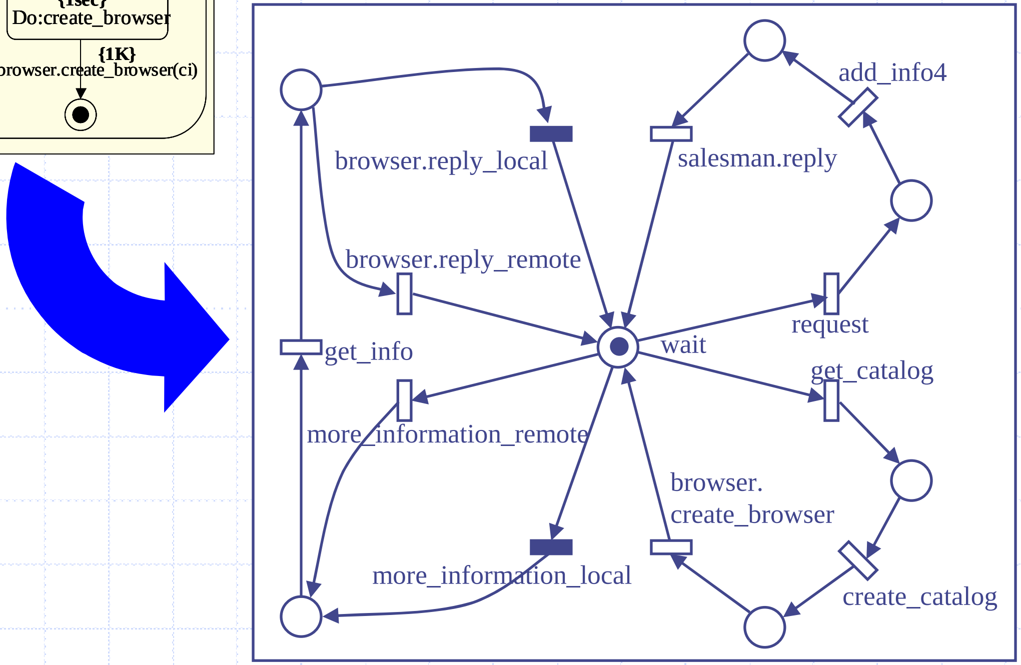


Real example

- Simplified version of the LGSPN component corresponding to the software manager agent

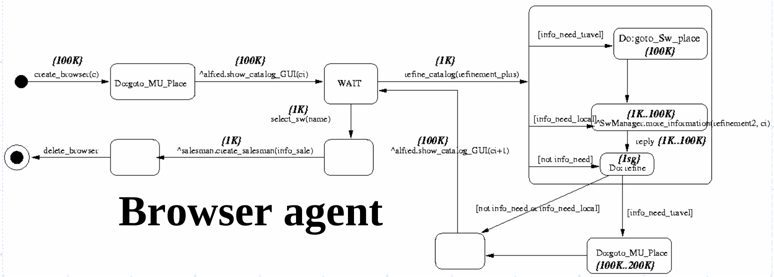


Software manager agent

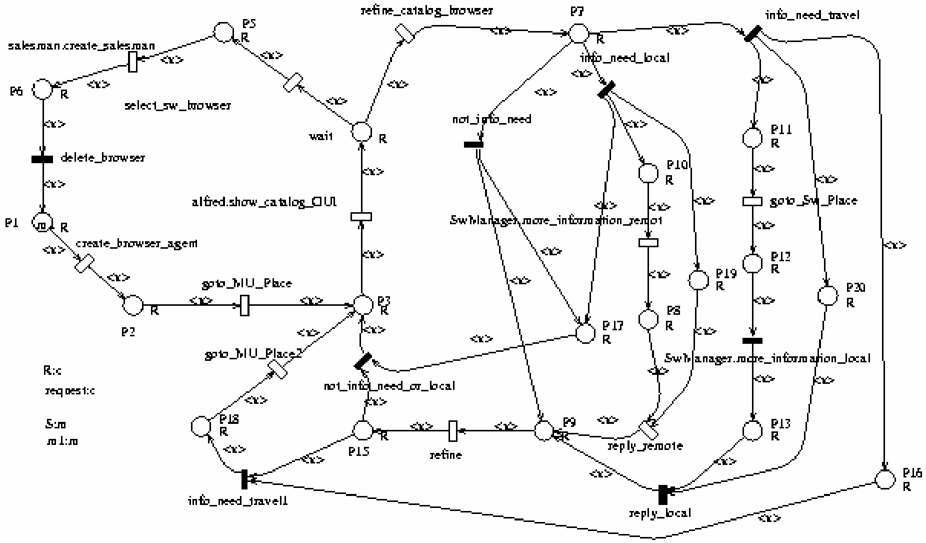


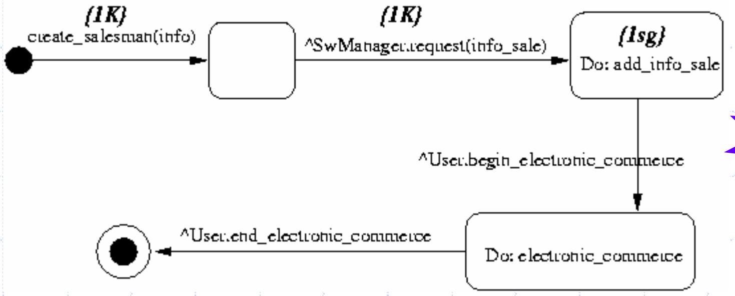
Real example

- The other agents

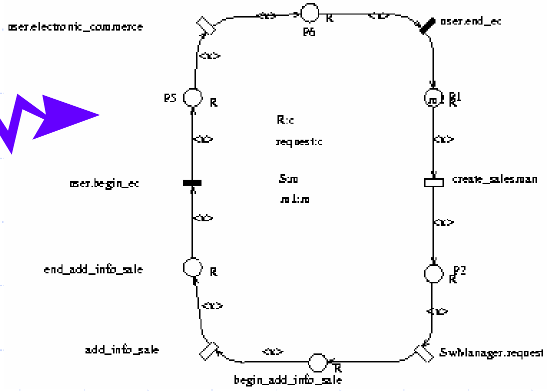


Browser agent



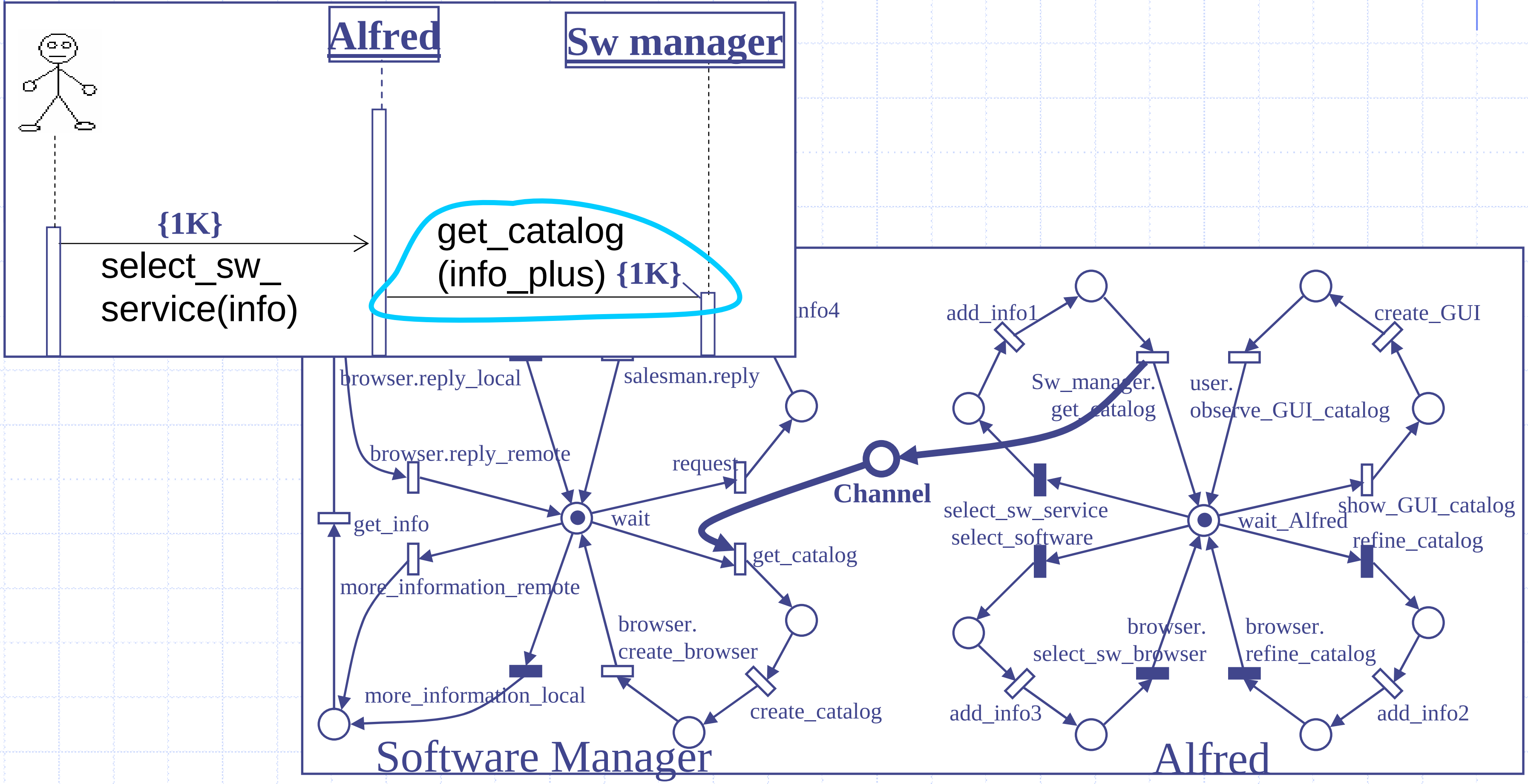

{1K}
create_salesman(info)
{1K}
^SwManager.request(info_sale)
{1sg}
Do: add_info_sale
^User.begin_electronic_commerce
^User.end_electronic_commerce
Do: electronic_commerce

Salesman



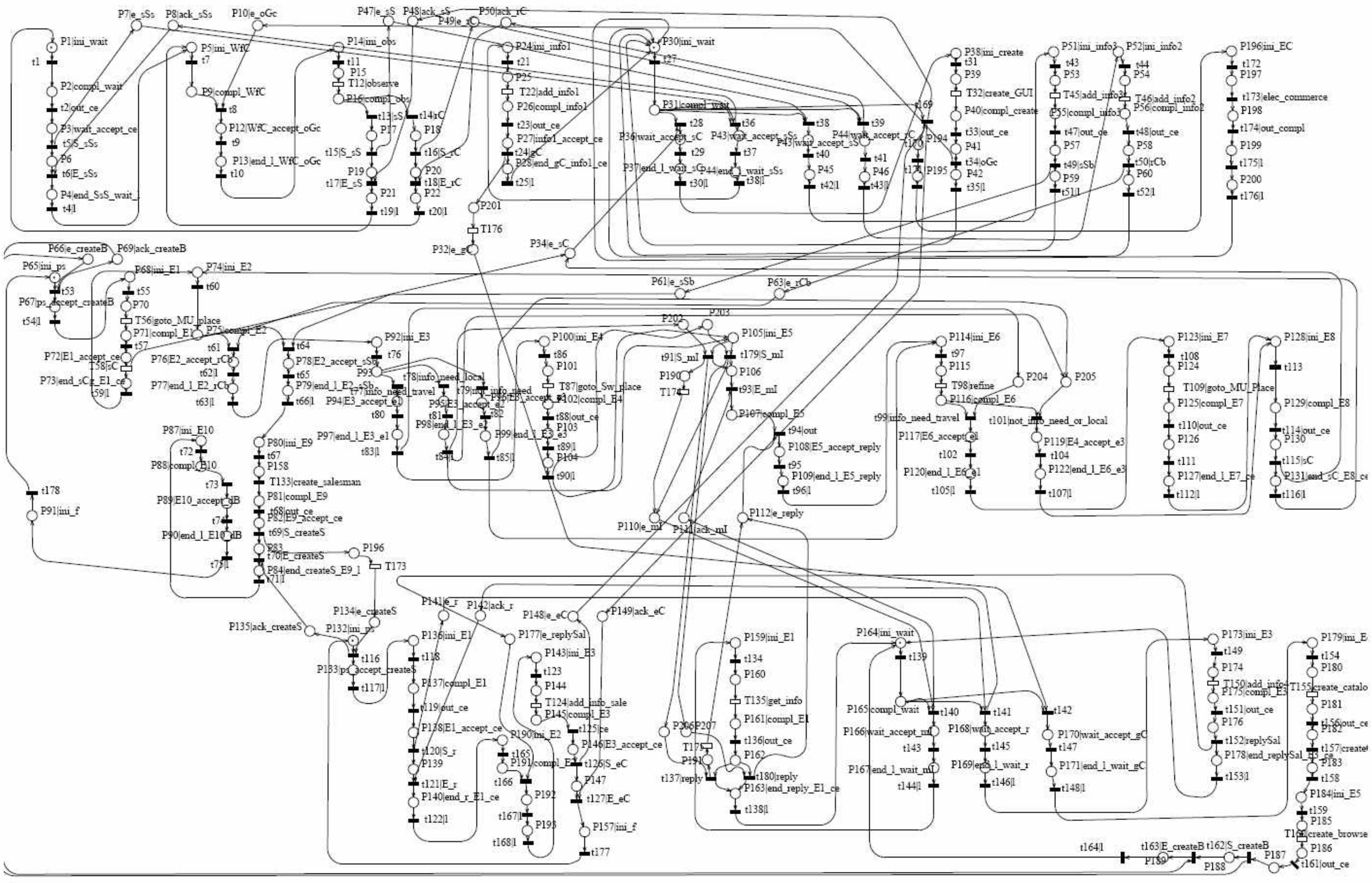
Real example

- Simplified version of composition between statechart LGSPN models and sequence diagram model



Real example

- Mobile agent-based approach: performance model



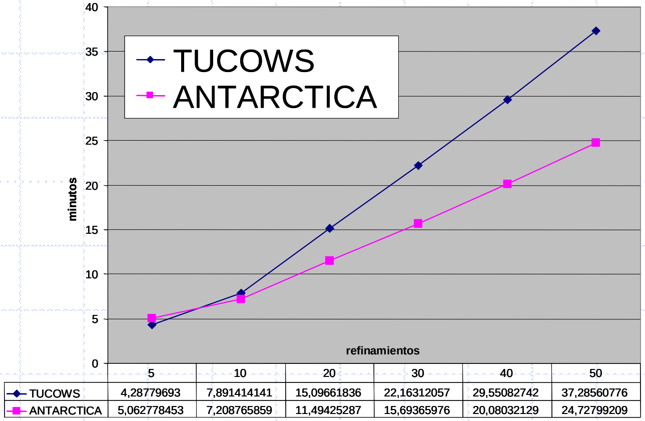
Real example

- ❑ Comparison of both approaches

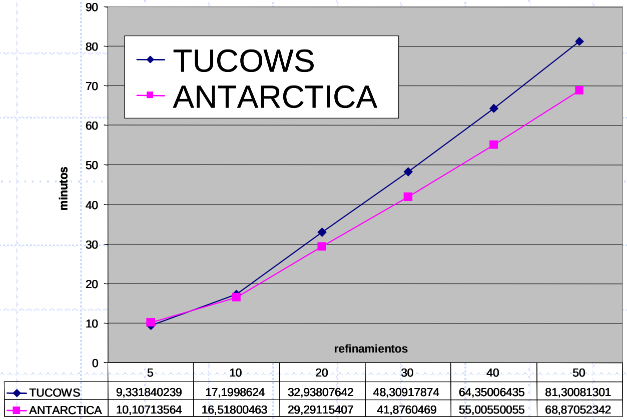
slow network

fast user

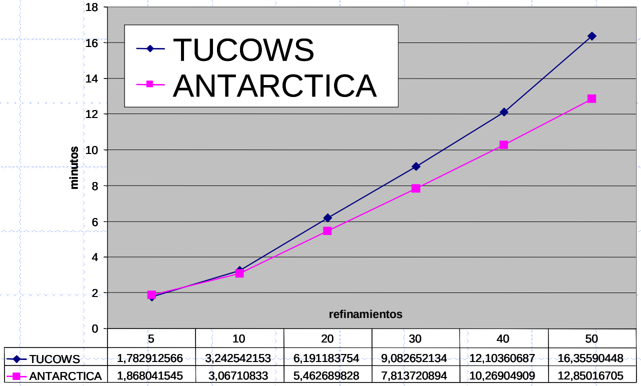
slow user



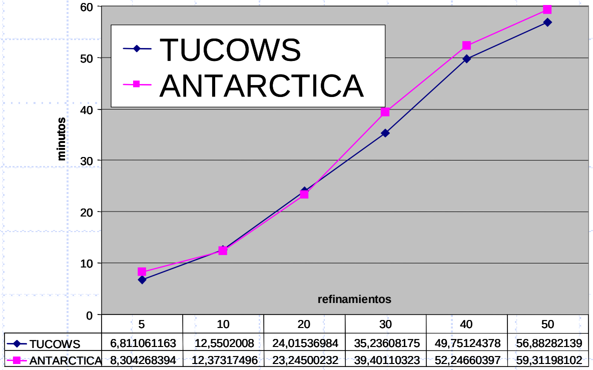
	5	10	20	30	40	50
◆ TUCOWS	4,28779693	7,891414141	15,09661836	22,16312057	29,55082742	37,28560776
■ ANTARCTICA	5,062778453	7,208765859	11,49425287	15,69365976	20,08032129	24,72799209



	5	10	20	30	40	50
◆ TUCOWS	9,331840239	17,1998624	32,93807642	48,30917874	64,35006435	81,30081301
■ ANTARCTICA	10,10713564	16,51800463	29,29115407	41,8760469	55,00550055	68,87052342



	5	10	20	30	40	50
◆ TUCOWS	1,782912566	3,242542153	6,191183754	9,082652134	12,10360687	16,35590448
■ ANTARCTICA	1,868041545	3,06710833	5,462689828	7,813720894	10,26904909	12,85016705



	5	10	20	30	40	50
◆ TUCOWS	6,811061163	12,5502008	24,01536984	35,23608175	49,75124378	56,88282139
■ ANTARCTICA	8,304268394	12,37317496	23,24500232	39,40110323	52,24660397	59,31198102

fast network

Outline

- ❑ Software Performance Engineering: basics
- ❑ A Software Performance Process
- ❑ Annotated UML Diagrams
- ❑ Integrating with Petri nets: case study
- ❑ Performance analysis
- ❑ Automation of the approach
- ❑ Real example
- ❑ Conclusions
- ❑ Bibliography

Conclusions

- Importance of integrated approach for SPE
  - Integration of
    - (pragmatic) software models and
    - (formal) performance models
  - Integration of performance analysis in the software life cycle
  - Methodology suitable for automation (tool)

Conclusions

- In usual software industry practice we are still close to the "fix-it-later" approach concerning non-functional requirements

  "make it run, make it run right, make it run fast"

- Important research effort on the SPE field
  - The role of the WOSP conference series
  - Sit together software engineers, performance modellers and analysts, and software developers

- So, we are in the good direction...

Outline

- □ Software Performance Engineering: basics
- □ A Software Performance Process
- □ Annotated UML Diagrams
- □ Integrating with Petri nets: case study
- □ Performance analysis
- □ Automation of the approach
- □ Real example
- □ Conclusions
- □ Bibliography

Bibliography

- ❑ J. Campos, J. Merseguer: **On the integration of UML and Petri nets in software development.** *Lecture Notes in Computer Science*, vol. 4024, pp. 19-36, 2006. (Invited talk in *27th International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency, ICATPN 2006.*) Download here.
- ❑ J. Merseguer: **Software Performance Engineering based on UML and Petri nets.** PhD Thesis. Dpto. Informática e Ingeniería de Sistemas, Universidad de Zaragoza. March 2003. Download here.