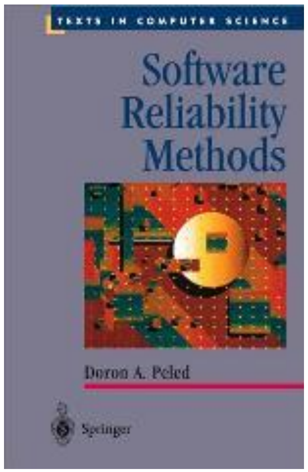
VERIFICA DEI PROGRAMMI CONCORRENTI
VPC 16-17

Analysis

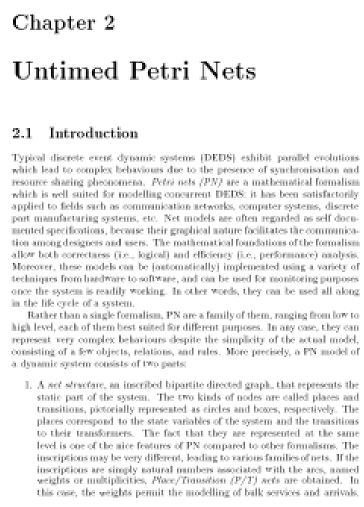
Prof.ssa Susanna Donatelli
Universita' di Torino
www.di.unito.it
susi@di.unito.it

Reference material books:



Prof. Doron A. Peled

(University of Warwick, UK)



Chapter 2

Untimed Petri Nets

2.1 Introduction

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

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

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

Notes of the EU-sponsored Jaca MATCH school

Acknowledgements

Transparencies adapted from the course notes and trasparencies of

- Prof. Doron A. Peled, University of Warwick (UK) and Bar Ilan University (Israel) http://www.dcs.warwick.ac.uk/~doron/srm.html
- Prof. Manuel Silva, Unievrsity of Zaragoza (Spain)

Second topic: analysis

- Check the kind of system to analyze.
- Choose formalisms, methods and tools.
- Express system properties.
- Model the system.
- Apply methods.
- Obtain verification results.
- Analyze results.
- Identify errors.
- Suggest correction.

Analysis

We shall review different analysis methods that apply (partially) to Petri Nets, Process Algebra, Finite State Automata

- Enumerative: analysis of derivation/reachability graph for a set of significative properties (deadlock/liveness)
- Transformation: analysis by reduction (not for FSA)
  - kit of reduction rules for PN
  - equational laws for PA (e.g. A+A = A)
- Structural analysis on incidence matrix (only for PN, extended partially to PA)
- Equivalences (defined for PA, extended to PN)
- Enumerative - again: Model checking on the state space (RG for PN, DG for PA, FSA) of temporal logic properties

Analysis

Different methods have different costs/applicability.

A good analysis of the system requires the use of different analysis methods on the same system model

(……… as done in testing)

Flowchart of analysis material

1. Basic properties
2. RG analysis
3. Structural analysis (on PN)
4. Reduction rules (PN)
5. Equivalences (PA)
6. Model checking
   - definition of linear logic LTL and its model checking algorithm
   - definition of branching logic CTL and its model checking algorithm

Basic properties - boundedness

Given a PN system S=(N,m0) with N=(P,T,F,W)

Def. *bound of place p in S*:

$$b(p) = \max \{m[p] \mid m \in RS(S)\}$$

Def. a *place* p is *bounded* in S if

$$b(p) < \infty$$

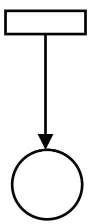
Def: a *system* S is *bounded* if

$$\forall p \in P,\ b(p) \text{ is bounded}$$

Propery: S is bounded iff its RS is finite

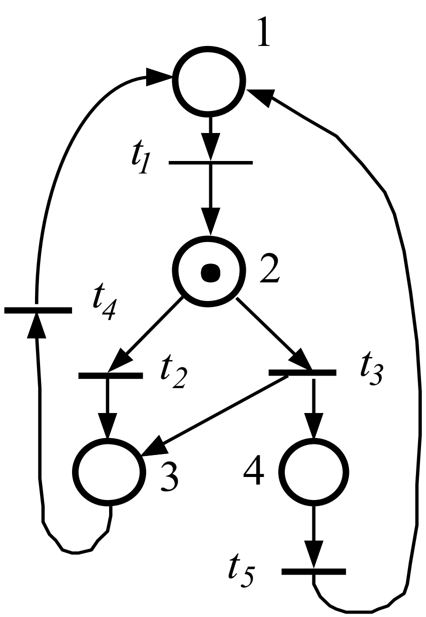
Basic properties - boundedness

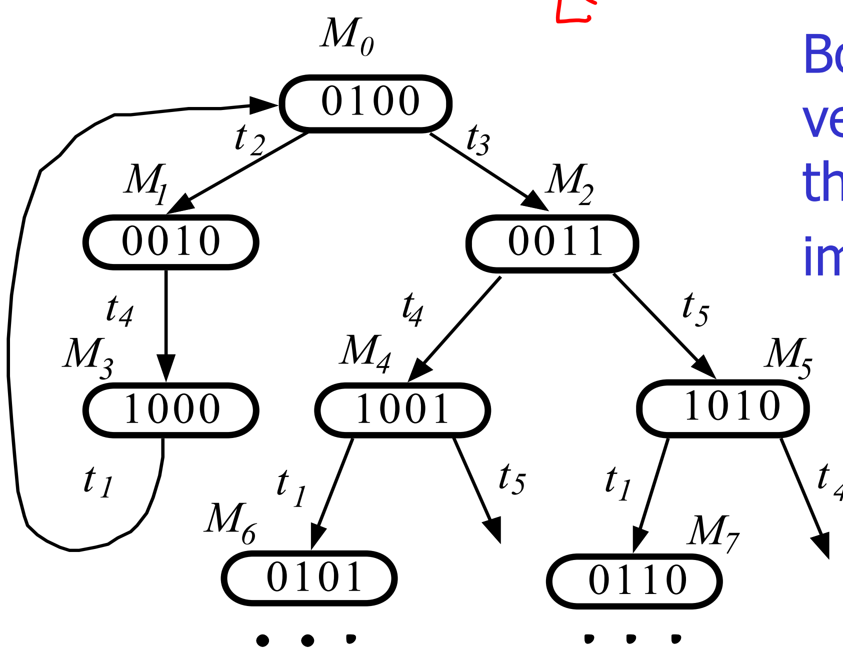
Examples of unbounded systems



A = a.nil + A||A

nil ← a   = a.nil   +[(a.nil + A||A)|| A]





Boundedness it is very important for the system to be implementable

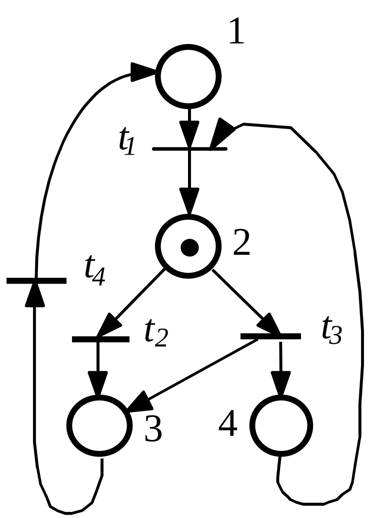
Basic properties - deadlock

Absence of deadlock iff it does not exist a reachable state that does not enable at least a transition

Def.: S = (N,m0) *is deadlock free* if

$$\forall m \in RS(m0),\ \exists\ t \in T:\ m[t>$$

The PN system below has a deadlock



Basic properties - liveness

A transition t is live if it can fire “infinitely often”

Def. $t \in T$ is live in $<N,m0>$ if

$$\forall m \in RS(m0),\ \exists\ \sigma: m[\sigma>m' \textit{ and } m'[t>$$

Def: A PN system $<N,m0>$ is live if

$$\forall t \in T,\ t \text{ is live in } <N,m0>$$

Note: A net with a finite strongly connected RG in which each transition label appears at least once on an arc is live

Def. t ∈T is live in <N,m0> if ∀m∈RS(m0), ∃ s: m[s>m' *and* m'[t>

Def: A PN system <N,m0> is live if ∀t∈T, t is live in <N,m0>

$Sys = (P \| C) \|_{\{get, put\}} (E2)$

get è vivo in Sys ?

get è vivo in Sys se

$\forall A \in DG(Sys) \ \exists\, a_1 \ldots a_n \in Act^* :$

$A \xrightarrow{a_1} A_1 \xrightarrow{a_2} \ldots \xrightarrow{a_n} A_{n+1}$

$\wedge \ A_{n+1} \xrightarrow{a} A'$

$$\begin{tikzcd}
(P\|C) \| E2 \arrow[d, shift left, "put"] \\
P\|C\|H2 \arrow[u, shift left, "get"] \arrow[d, shift left, "put"] \\
P\|C\|F2 \arrow[u, shift left, "get"]
\end{tikzcd}$$

stato 1

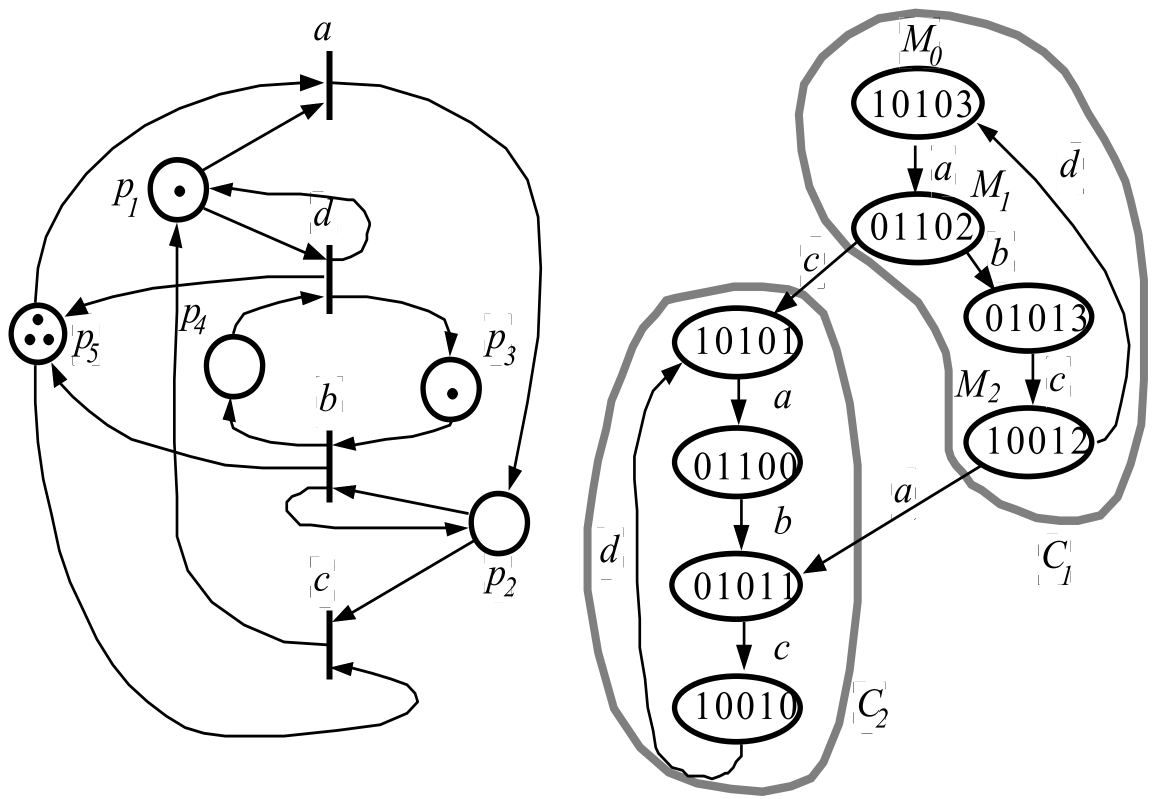
→ stato 2

→ :

stato n

Basic properties - liveness

The PN system below is live, although the RG is not strongly connected, because in each BSCC of the RG it is possible to fire all transitions



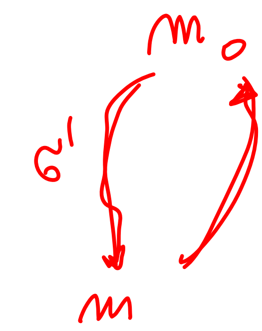
Basic properties - reversibility

Def. a marking $m \in RS(m0)$ is a *home state* if

$$\forall m' \in RS(m0),\ \exists\ \sigma: m'[\sigma>m$$

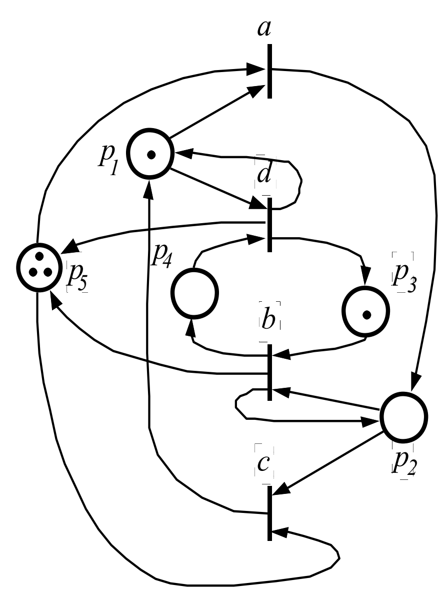
Def: a system $<N,m0>$ is *reversible* if

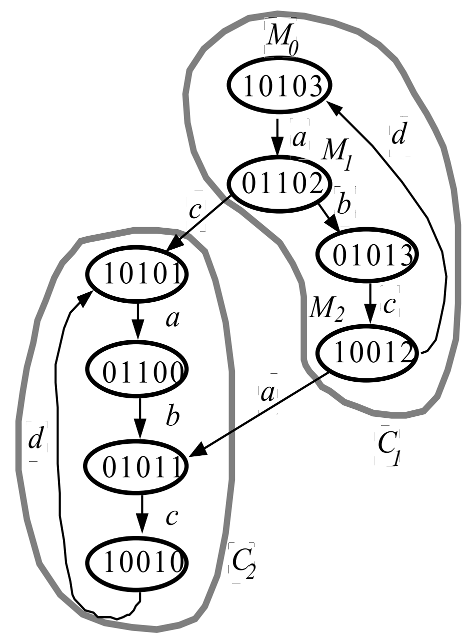
$$\forall m \in RS(m0),\ \exists\ \sigma: m[\sigma>m0$$



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

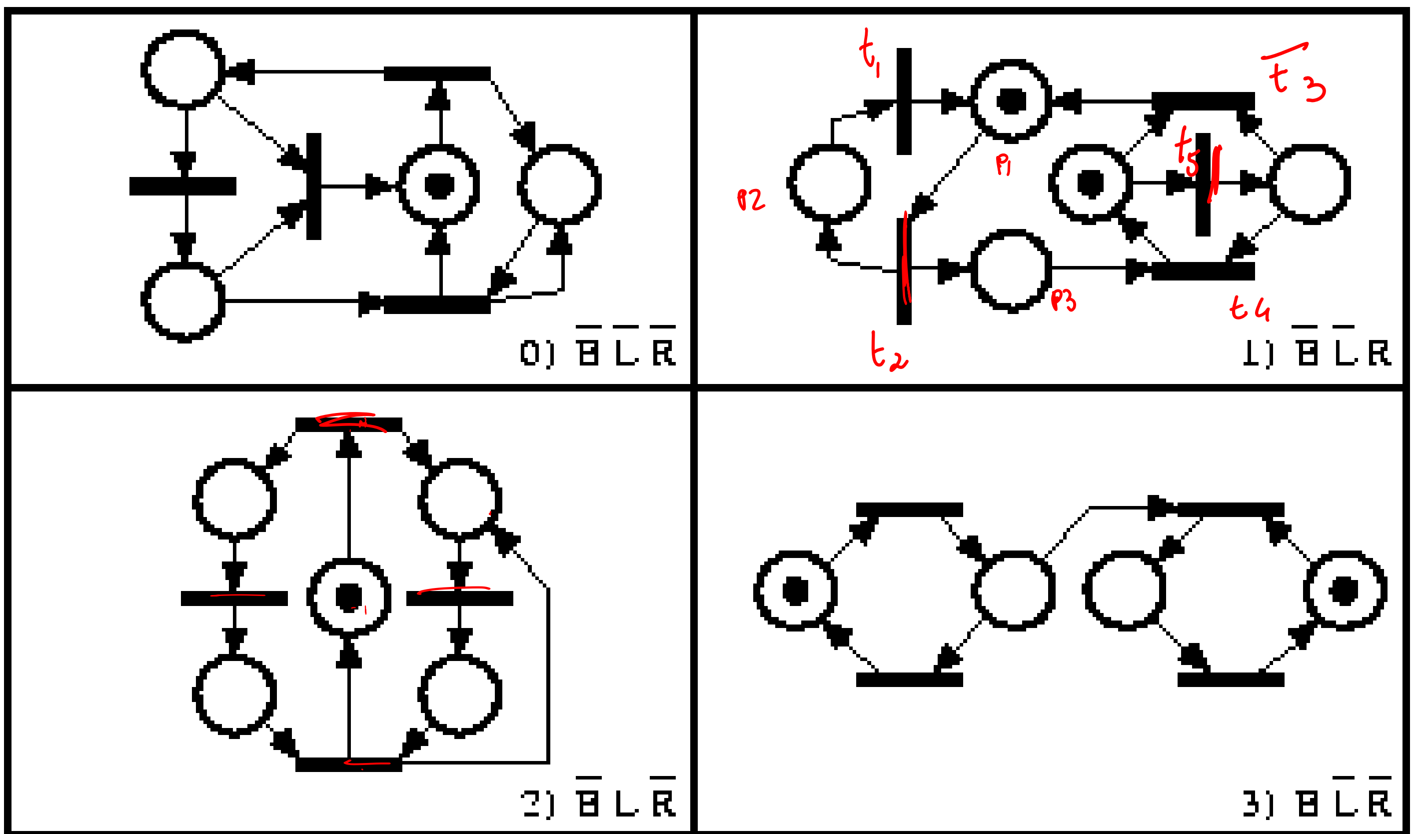
The PN system below is not reversible (there are two SCC)



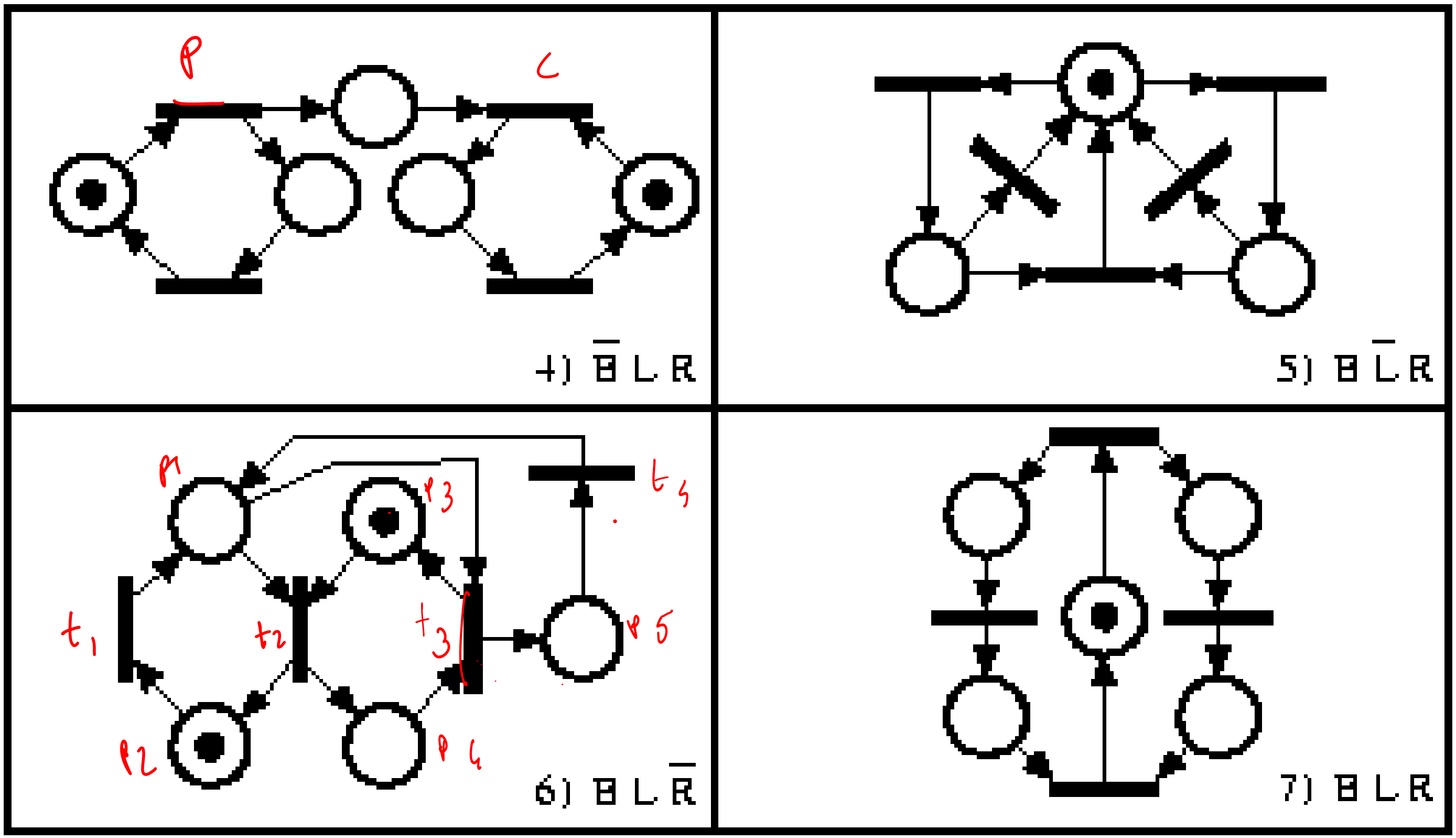


boundedness, liveness, reversability

Boundedness, liveness and reversability are disjoint properties (B,L,R true or false --> 2^3 counter-examples)



boundedness, liveness, reversability

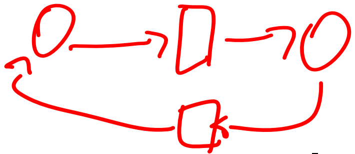


Exercise: determine what makes, in each net, a property true or false

Structural properties

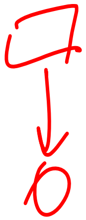
Idea: to define properties independently of m0

Def: N is structurally bounded if, ∀ finite m0, <N,m0> is bounded



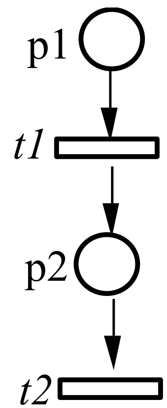
Def: N is structurally live if ∃ finite m0: <N,M0> is live

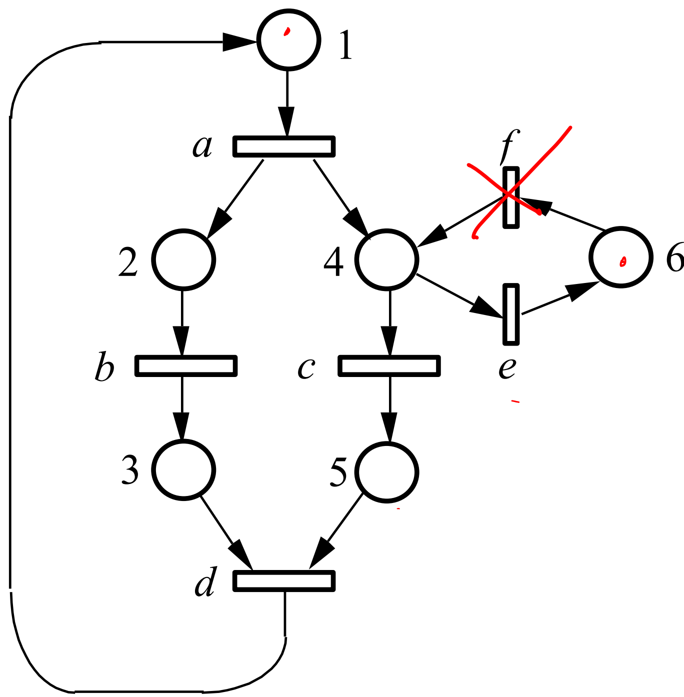
∀ finite m_0, <N,m_0> is live



A simple PN in matrix form

Structurally bounded, not structurally live





Structurally bounded, structurally live.

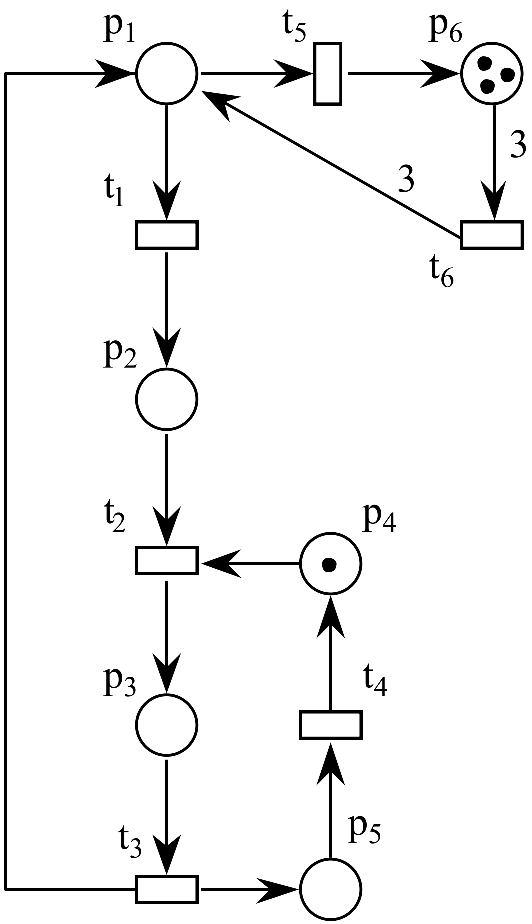
Make it NOT (structurally bounded)

Make it NOT (structurally live)

Def: N is structurally bounded if, ∀ finite m0, <N,m0> is bounded

Def: N is structurally live if ∃ finite m0: <N,M0> is live

Another example



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

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

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

Summary of properties

(1)	Bound of place p in $\langle \mathcal{N}, \mathbf{m_0} \rangle$ $\mathbf{b}(p) = \sup\{\mathbf{m}[p] \mid \mathbf{m} \in \mathrm{RS}(\mathcal{N}, \mathbf{m_0})\}$
(2)	p is bounded in $\langle \mathcal{N}, \mathbf{m_0} \rangle$ iff $\mathbf{b}(p) < \infty$
(3)	$\langle \mathcal{N}, \mathbf{m_0} \rangle$ is bounded if all places are bounded
(4)	$\langle \mathcal{N}, \mathbf{m_0} \rangle$ is deadlock-free iff $\forall \mathbf{m} \in \mathrm{RS}(\mathcal{N}, \mathbf{m_0})$ $\exists t \in T$ such that t is fireable at $\mathbf{m}$
(5)	t is live in $\langle \mathcal{N}, \mathbf{m_0} \rangle$ iff $\forall \mathbf{m} \in \mathrm{RS}(\mathcal{N}, \mathbf{m_0})$ $\exists \sigma$ such that $\mathbf{m} \xrightarrow{\sigma t} \mathbf{m}'$
(6)	$\langle \mathcal{N}, \mathbf{m_0} \rangle$ is live if all transitions are live
(7)	$\mathbf{m} \in \mathrm{RS}(\mathcal{N}, \mathbf{m_0})$ is a home state iff $\forall \mathbf{m}' \in \mathrm{RS}(\mathcal{N}, \mathbf{m_0})$ $\exists \sigma$ such that $\mathbf{m}' \xrightarrow{\sigma} \mathbf{m}$
(8)	$\langle \mathcal{N}, \mathbf{m_0} \rangle$ is reversible iff $\forall \mathbf{m} \in \mathrm{RS}(\mathcal{N}, \mathbf{m_0})$ $\exists \sigma$ such that $\mathbf{m} \xrightarrow{\sigma} \mathbf{m_0}$
(9)	Mutual exclusion in $\langle \mathcal{N}, \mathbf{m_0} \rangle$: p_i and p_j are in marking mutual exclusion iff $\nexists\ \mathbf{m} \in \mathrm{RS}(\mathcal{N}, \mathbf{m_0})$ such that $(\mathbf{m}[p_i] > 0)$ and $(\mathbf{m}[p_j] > 0)$ t_i and t_j are in firing mutual exclusion iff $\nexists\ \mathbf{m} \in \mathrm{RS}(\mathcal{N}, \mathbf{m_0})$ such that $\mathbf{m} \geq \mathbf{Pre}[P, t_i] + \mathbf{Pre}[P, t_j]$
(10)	Structural properties (abstractions of behavioural properties): $\mathcal{N}$ is structurally bounded iff $\forall \mathbf{m_0}$ (finite) $\langle \mathcal{N}, \mathbf{m_0} \rangle$ is bounded $\mathcal{N}$ is structurally live iff $\exists \mathbf{m_0}$(finite) making $\langle \mathcal{N}, \mathbf{m_0} \rangle$ a live system

Enumeration techniques

Tecniche usate da metà degli anni settanta per la verifica di protocolli:

- CCITT x.21, X.25, IBM/SNA (System Network Architecture)- data flow control layer, IBM token ring
- Alternating bit, sliding window ISO-OSI architecture – transport e session layer
- Normalmente il linguaggio di specifica è Estelle, SDL
- Riferimento di ricerca: IFIP WG6.1, con conferenze quali FORTE (Formal description Techniques for distributed systems and communication protocols) dal 1988 e PSTV (Protocol Specification, Testing, and Verification) dal 1981

Enumeration techniques

Prove property by state enumeration (only finite system/discrete/continous?)

Main problem: state space explosion and decidability

Classify properties as:

- Marking invariance
- Liveness invariance

Build RG and define two distinct algorithms for marking and liveness invariance

Enumeration techniques

Un problema indecidbile:

- Dati due sistemi decidere se i loro grafi di raggiungibilità sono uguali (o inclusi uno nell’altro) – Hack 1975

Complessità:

Sia data un sistema a reti di Petri limitato (bounded). Il problema della costruzione del grafo (dell’insieme) di raggiungibilità) non è ricorsivo primitivo.

Enumeration techniques

Boundedness è decidibile per reti P/T, ma certo non posso pensare di controllare questa proprietà sul Reachability Graph (ovviamente la costruzione non termina se la rete è unbounded)

La chiave per la decidibilità è la possibilità di costruire un grafo finito – Coverability Graph -- anche per reti unbounded, sul quale sia possibile riportare la decisione di proprietà

Basato sulla nozione di "copertura" fra marking: diciamo che $m \leq m'$ (m' copre m) se $\forall p \in P$, $m'(p) \geq m(p)$

Enumeration techniques

Algorithm 6.1 (Computation of the Reachability Graph)

Input - The net system $\mathcal{S} = \langle \mathcal{N}, \mathbf{m}_0 \rangle$
Output - The directed graph $\mathrm{RG}(\mathcal{S}) = (V, E)$ for bounded net systems

1. Initialize $\mathrm{RG}(\mathcal{S}) = (\{\mathbf{m}_0\}, \emptyset)$; $\mathbf{m}_0$ is untagged;
2. **while** there are untagged nodes in V **do**
   2.1 Select an untagged node $\mathbf{m} \in V$ and tag it
   2.2 **for** each enabled transition, t, at $\mathbf{m}$ **do**
      2.2.1 Compute $\mathbf{m}'$ such that $\mathbf{m} \xrightarrow{t} \mathbf{m}'$;
      2.2.2 **if** there exists $\mathbf{m}'' \in V$ such that $\mathbf{m}'' \xrightarrow{\sigma} \mathbf{m}'$ and $\mathbf{m}'' \lneq \mathbf{m}'$
         then the algorithm fails and exits;
            (the unboundedness condition of $\mathcal{S}$ has been detected)
      2.2.3 **if** there is no $\mathbf{m}'' \in V$ such that $\mathbf{m}'' = \mathbf{m}'$
         then $V := V \cup \{\mathbf{m}'\}$; ($\mathbf{m}'$ is an untagged node)
      2.2.4 $E := E \cup \{\langle \mathbf{m}, t, \mathbf{m}' \rangle\}$
3. The algorithm succeds and $\mathrm{RG}(\mathcal{S})$ is the reachability graph

Less than or equal, but with at least one place for which the ≤ is strict

Enumeration techniques

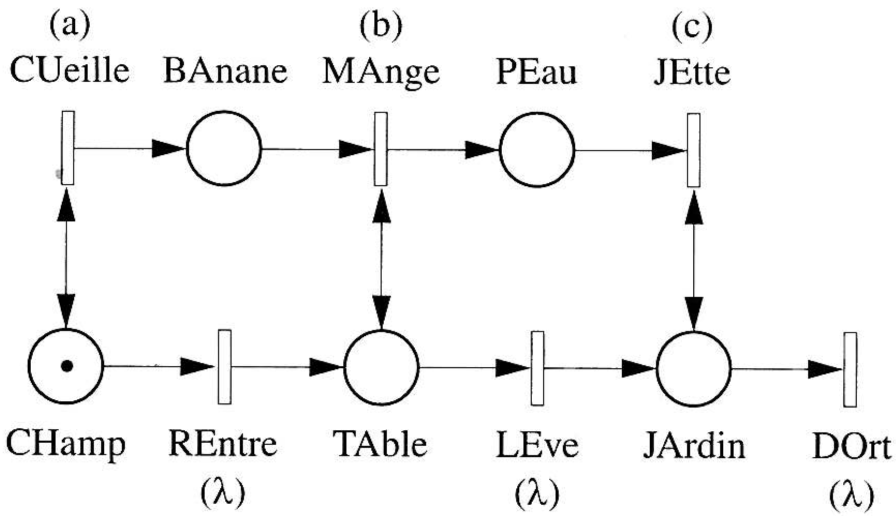
Algorithm 6.1 (Computation of the Coverability tree)

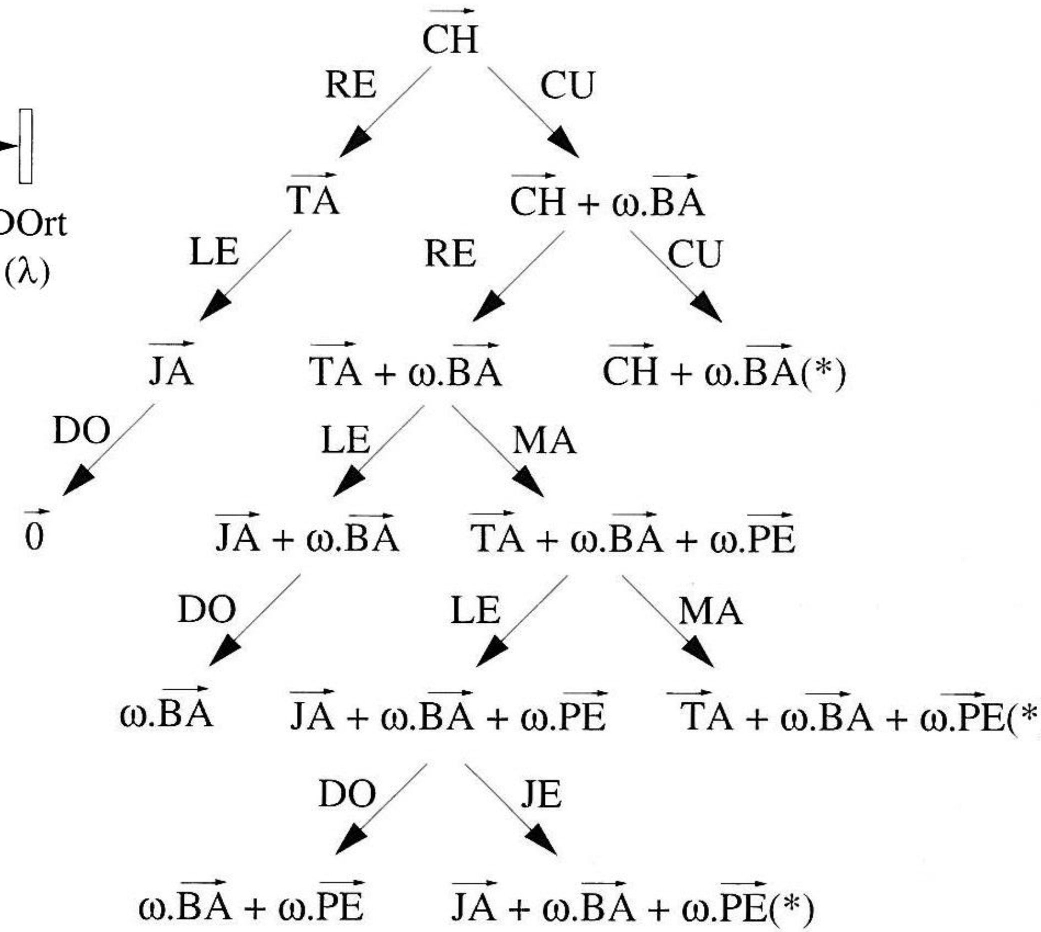
Input - The net system $\mathcal{S} = \langle \mathcal{N}, \mathbf{m}_0 \rangle$
Output - The directed graph $\mathrm{RG}(\mathcal{S}) = (V, E)$ for bounded net systems

1. Initialize $\mathrm{RG}(\mathcal{S}) = (\{\mathbf{m}_0\}, \emptyset)$; $\mathbf{m}_0$ is untagged;
2. **while** there are untagged nodes in V **do**
   2.1 Select an untagged node $\mathbf{m} \in V$ and tag it
   2.2 **for** each enabled transition, t, at $\mathbf{m}$ **do**
      2.2.1 Compute $\mathbf{m}'$ such that $\mathbf{m} \xrightarrow{t} \mathbf{m}'$;
      2.2.2 **if** there exists $\mathbf{m}'' \in V$ such that $\mathbf{m}'' \xrightarrow{\sigma} \mathbf{m}'$ and $\mathbf{m}'' \lneq \mathbf{m}'$

Etichettare m(p) con ω

      2.2.3
      2.2.4 $E := E \cup \{\langle \mathbf{m}, t, \mathbf{m}' \rangle\}$
3. The algorithm succeds and $\mathrm{RG}(\mathcal{S})$ is the r Coverability tree





If n is a natural number, then
n < ω, n+ ω = ω +n = ω, ω -n = ω

Enumeration techniques

Marking invariance (a property of a single marking that has to be verified for all markings)

ϕ(m) is a marking invariant property if:

$$\forall m \in RS(m0),\ \ \phi(m) \text{ is true}$$

Examples:

1) *k-boundedness of place p*: $\forall \mathbf{m} \in \mathrm{RS}(\mathcal{S}),$

2) *Marking mutual exclusion between p and p'*: $\forall \mathbf{m} \in \mathrm{RS}(\mathcal{S}),$

).

3) *Deadlock-freeness*: $\forall \mathbf{m} \in \mathrm{RS}(\mathcal{S}),$

4) $\sum_{p \in A} k_p \mathbf{m}[p] \ \leq \ k$

Enumeration techniques

Problemi decidibili:

- Copertura di un marking m (esiste un marking raggiungibile m': $m \leq m'$)
- Insieme di posti "simultaneously unbounded"
- Reachable transition (transizione scattabile almeno una volta in almeno una sequenza che parte dalla marcatura iniziale)
- Liveness
- Reachability – non si può risolvere per ispezione del coverability graph, problema aperto dal 69 e chiuso da Kosaraju nel 1982 e Mayr nel 1984. Il problema è EXP-space hard

Enumeration techniques

Back to

- Marking invariance
- Liveness invariance

Build RG and define two distinct algorithms for marking and liveness invariance

Enumeration techniques– marking invariance

Algorithm 6.2 (Decision procedure for marking invariance

Input - The reachability set $\mathrm{RS}(\mathcal{S})$. The property Π.
Output - TRUE if the property is verified.

1. Initialise all elements of $\mathrm{RS}(\mathcal{S})$ as untagged.
2. **while** there is an untagged node $\mathbf{m} \in \mathrm{RS}(\mathcal{S})$ **do**
   2.1 Select an untagged node $\mathbf{m} \in \mathrm{RS}(\mathcal{S})$ and tag it
   2.2 **if** $\mathbf{m}$ does not satisfy Π
   then return FALSE (the property is not verified).
3. Return TRUE

Enumeration techniques

Liveness invariance (for each reachable marking there is at least a marking reachable from it that satisfies the property)

$$\forall \mathbf{m} \in \mathrm{RS}(\mathcal{S}),\ \exists \mathbf{m}' \in \mathrm{RS}(\mathcal{N}, \mathbf{m}),\ \mathbf{m}' \text{ satisfies } \Pi$$

Examples:

1) *Liveness of t*: $\forall \mathbf{m} \in \mathrm{RS}(\mathcal{S}),\ \exists \mathbf{m}' \in \mathrm{RS}(\mathcal{N}, \mathbf{m})$ such that $\mathbf{m}'$

2) $\mathbf{m}_H$ *is home state*: $\forall \mathbf{m} \in \mathrm{RS}(\mathcal{S}),\ \exists \mathbf{m}' \in \mathrm{RS}(\mathcal{N}, \mathbf{m})$ such that $\mathbf{r}$

3) *Reversibility*: $\forall \mathbf{m} \in \mathrm{RS}(\mathcal{S}),\ \exists \mathbf{m}' \in \mathrm{RS}(\mathcal{N}, \mathbf{m})$ such tha .

Enumeration techniques – liveness invariance

Approach: reduce the problem to the Bottom strongly connected components

$$\forall \mathbf{m} \in \mathrm{RS}(\mathcal{S}),\ \exists \mathbf{m}' \in \mathrm{RS}(\mathcal{N}, \mathbf{m}),\ \mathbf{m}' \text{ satisfies } \Pi$$

Enumeration techniques – liveness invariance

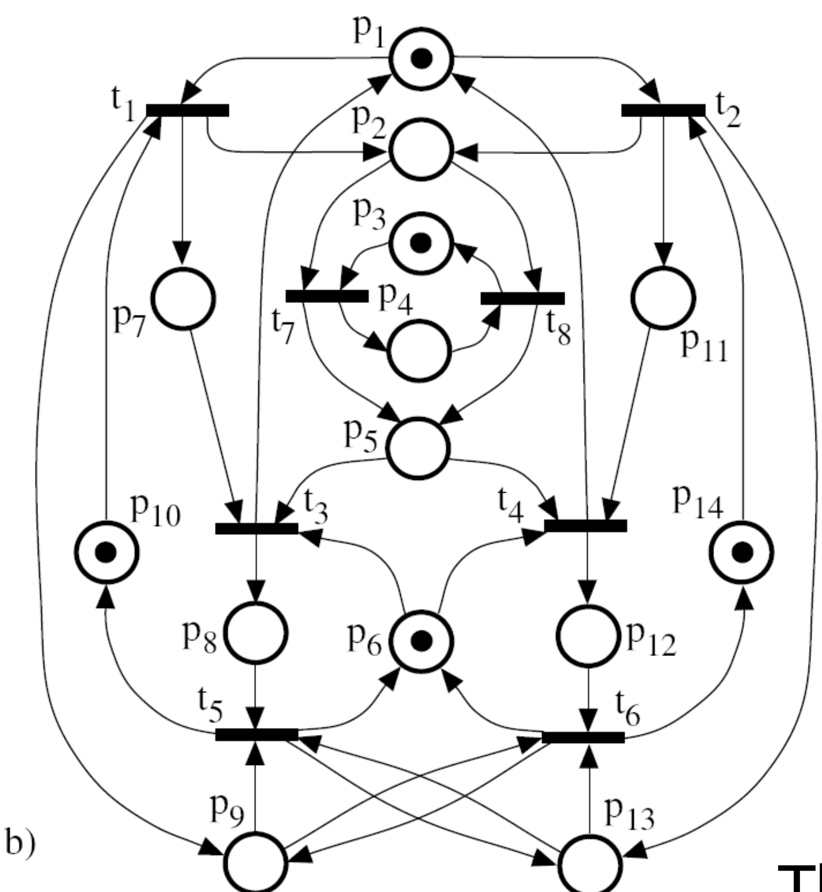
Algorithm 6.3 (Decision procedure for liveness invariances)

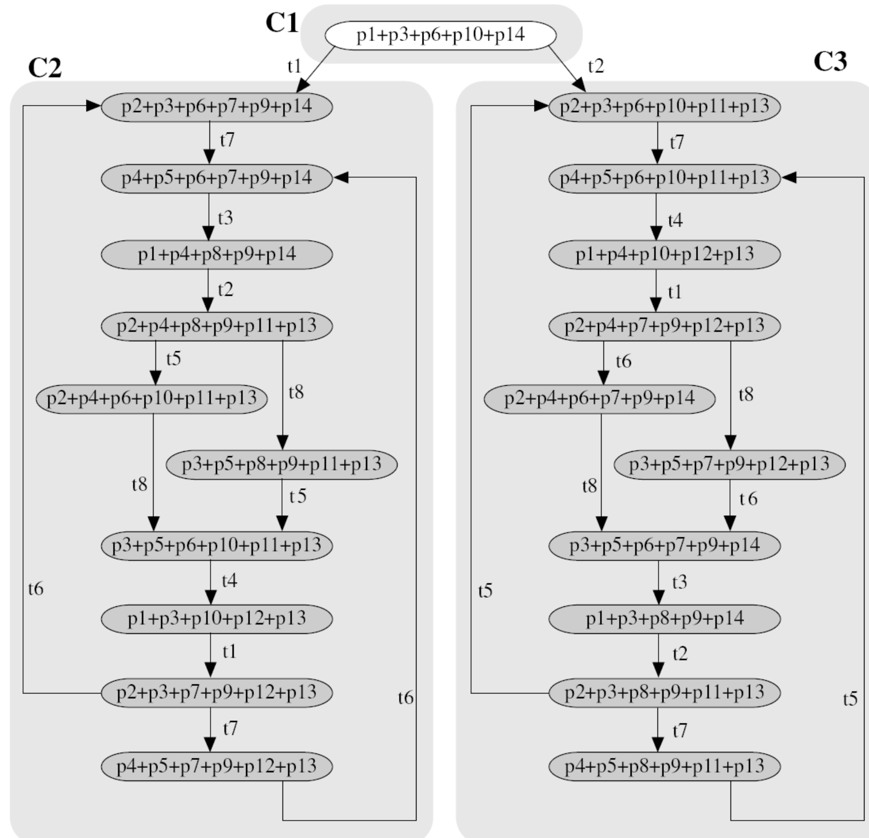
Input - The reachability graph $RG(\mathcal{N}, \mathbf{m}_0)$. The property Π
Output - TRUE if the property is verified.

1. Decompose $RG(\mathcal{N}, \mathbf{m}_0)$ into its strongly connected components $C_1, \ldots, C_r$
2. Obtain the graph $RG^c(\mathcal{S}) = (V_c, E_c)$ by shrinking $C_1, \ldots, C_r$ to a single node, i.e. $V_c = \{C_1, \ldots, C_r\}$. $\langle C_i, t, C_j \rangle \in E_c$ iff there exists $\langle \mathbf{m}, t, \mathbf{m}' \rangle \in E$, such that $\mathbf{m}$ is in the SCC C_i, $\mathbf{m}'$ is in the SCC C_j, and $i \neq j$.
3. Compute the set F of terminal strongly connected components from $RG^c(\mathcal{S})$
4. **while** there is a $C_i \in F$ **do**
   3.1 **if** C_i it does not contain a $\mathbf{m}'$ satisfying Π
   then return FALSE
   3.2 Remove C_i from F
5. Return TRUE

$$\boxed{\forall \mathbf{m} \in RS(\mathcal{S}),\ \exists \mathbf{m}' \in RS(\mathcal{N}, \mathbf{m}),\ \mathbf{m}' \text{ satisfies } \Pi}$$

Enumeration techniques – liveness invariance





The net has 2 BSCC - the net is live, but m0 is not a home state

State of the art

RG is exponential in the size of P and T

Marking invariance is linear in |RS|

Liveness invariance requires the construction of SCC (|V|+|E|) plus the check of the property on each BSCC.

Explicit techniques (like the one shown) allows the check of RG with some millions/tenths of millions state - now more with symbolic techniques

It also depends on the size of the state

State of the art - implicit/symbolic techniques

RG is exponential in the size of P and T an m0

Time and memory complexity of RG generation may be less than exponential (less than |RG| and even |RS|)

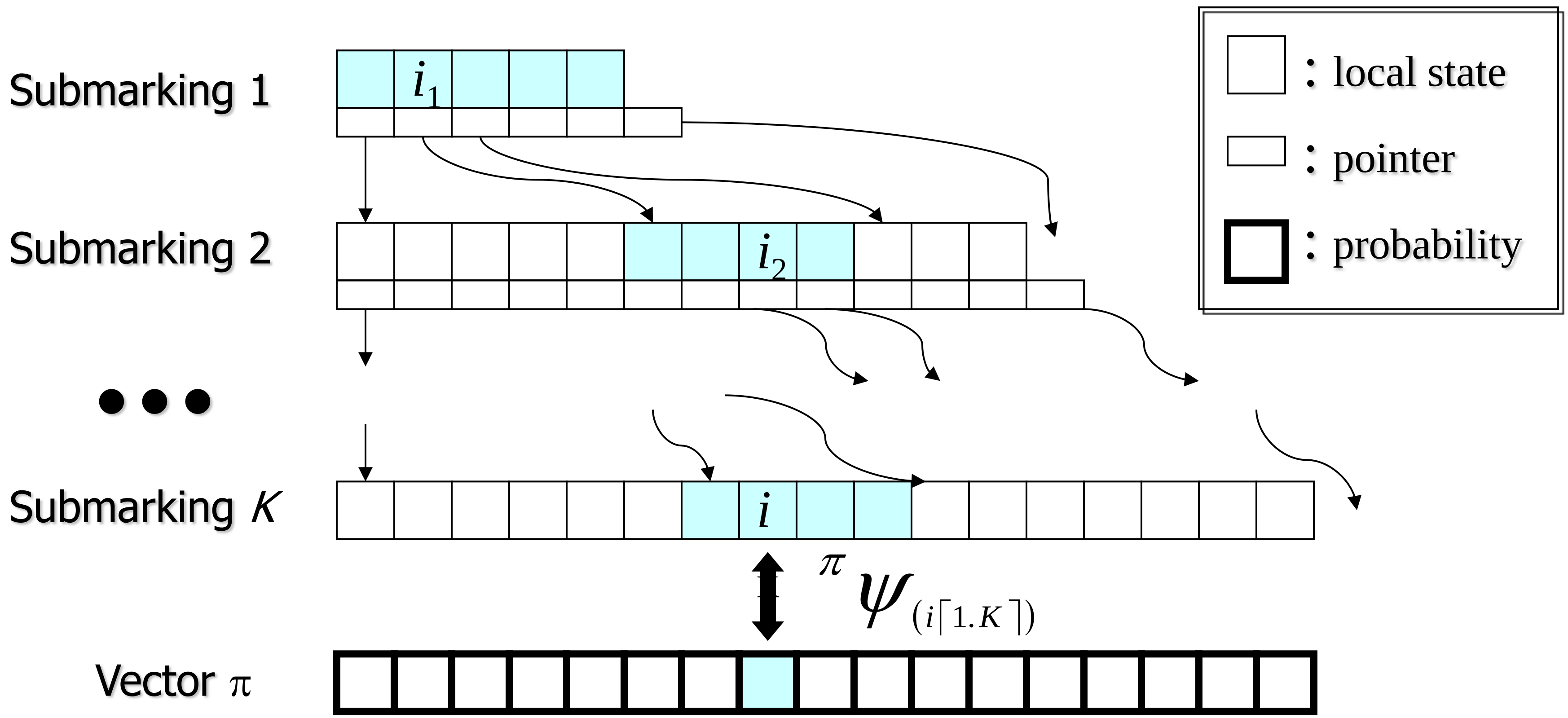
Basic ideas:

- Reuse substates for different states
- Fire more than one transition at a time
- Need to be sure that property can be checked without making the RG explicit

Multilevel data structures for RS/RG

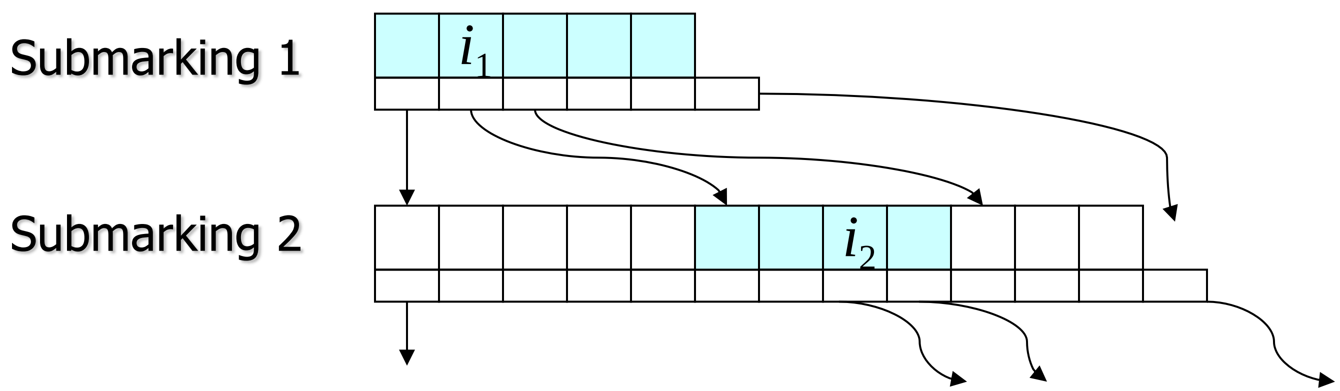
Partition P into K subsets (possibly K=|P|) and choose an order.

A reachable marking is a visit of the data structure form an entry in level 1



Multilevel data structures for RS/RG

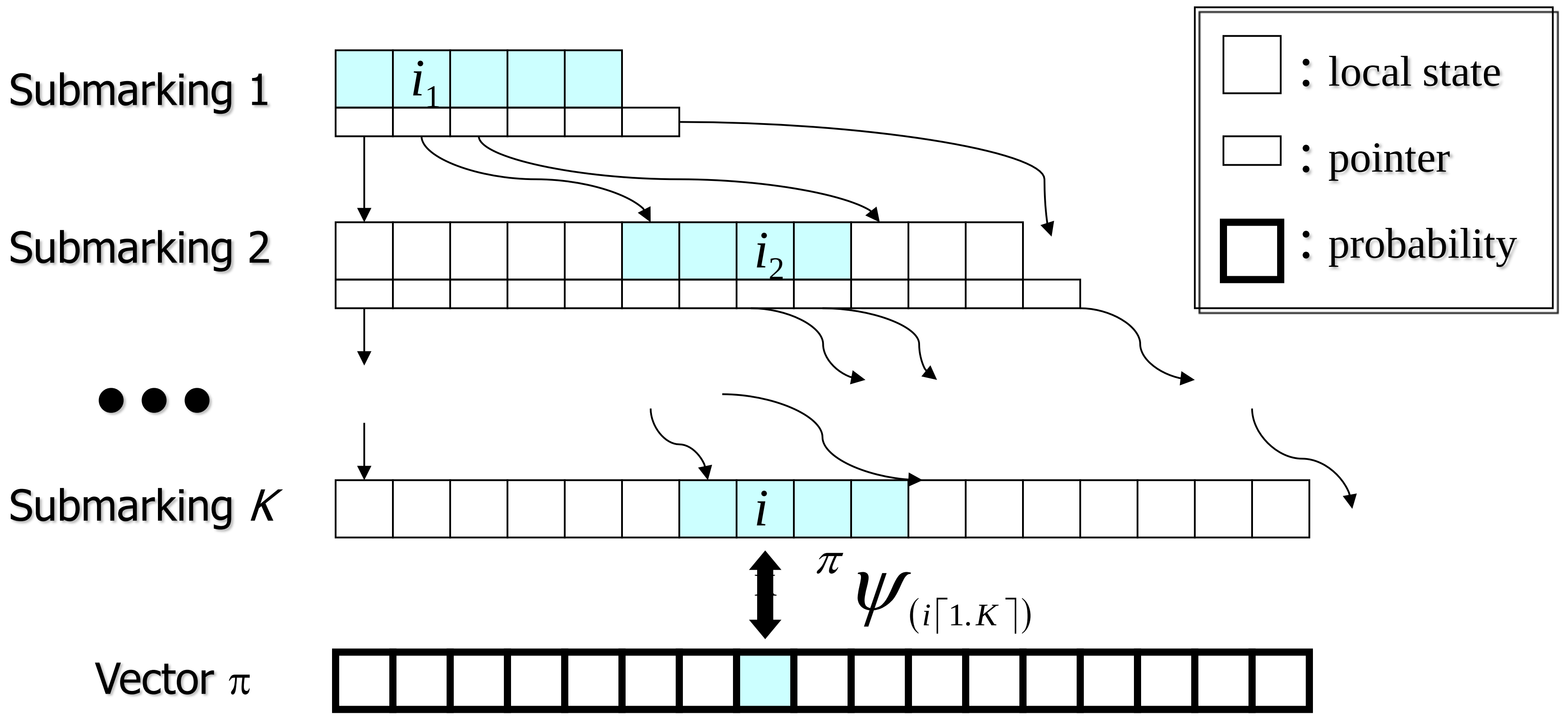
If the states of level 1 and level 2 are independent, how does the data structure looks like?



If the states of level 1 and level 2 are independent, how does the data structure looks like?

Multilevel data structures for RS/RG

Can we do better? May be certain subvector (and subtrees) are actually the same!



State of the art - implicit/symbolic techniques

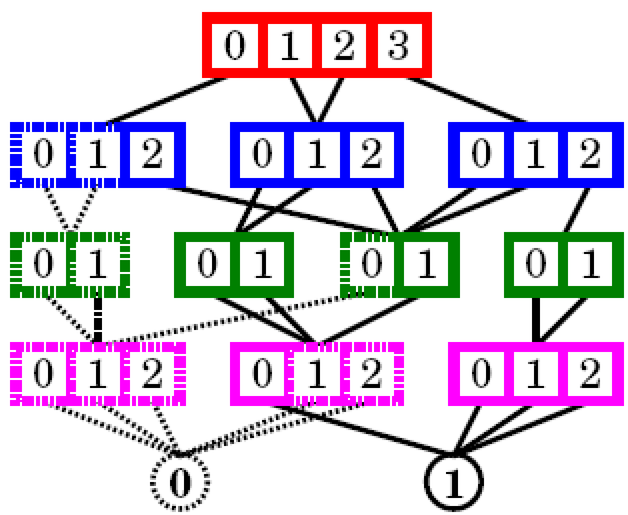
A reachable marking is a boolean function from $S_4 \times S_3 \times S_2 \times S_1$ --> {0,1}
A marking m is reachable if the visit of the data structure from top to bottom according to m goes to 1

$$\mathcal{S}_4 = \{0,1,2,3\}$$

$$\mathcal{S}_3 = \{0,1,2\}$$

$$\mathcal{S}_2 = \{0,1\}$$

$$\mathcal{S}_1 = \{0,1,2\}$$



$$\mathcal{S} = \left\{ \begin{array}{ccccccccccccccccccc} 0 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 2 & 0 & 0 & 1 & 1 & 2 & 0 & 0 & 1 & 1 & 2 & 0 & 1 & 2 & 2 & 2 & 2 & 2 & 2 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 0 & 1 & 2 \end{array} \right\}$$

Reduction techniques

List of rules with

- Structural and behavioural pre-condition
- Net reduction: $< N_i, \mathbf{m0}_i > \;\rightarrow\; < N_{i+1}, \mathbf{m0}_{i+1} >$
- the reduction is "property preserving"
- The net $< N_{i+1}, \mathbf{m0}_{i+1} >$ is "easier" to analyze than $< N_i, \mathbf{m0}_i >$ (e.g.: a smaller RG, or N_{i+1} is of a subclass for which there are structural results available)

Rewriting system (with the usual problems of completeness and confluence)

Can be used both ways (step wise refinement for "well-behaved" construction or reduction for analysis)

Reduction techniques

RA1 is a “macroplace” reduction

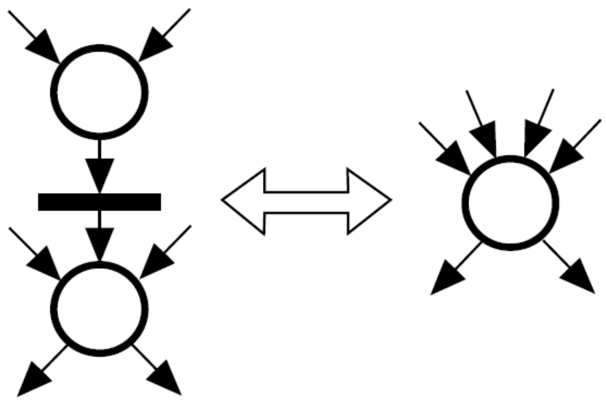
RA1 is a transition fusion

RB1 and RC1 are cases of implicit place rules

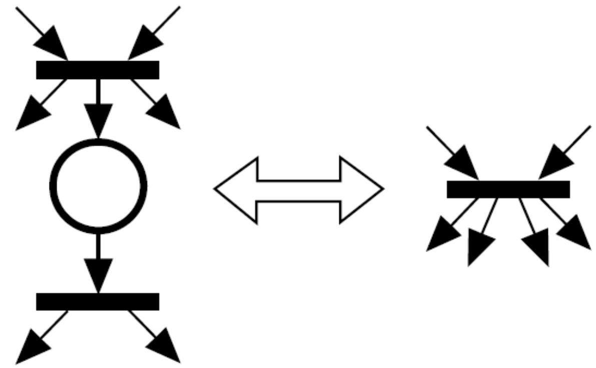
RB2 and RC2 are cases of identical and identity transitions rules

Can be obtained by duality

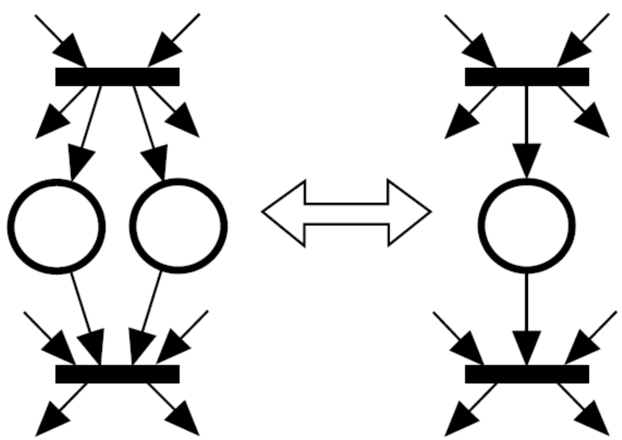
Preserve liveness, boundedness, existence of home states (but not reversibility, due to RA1)



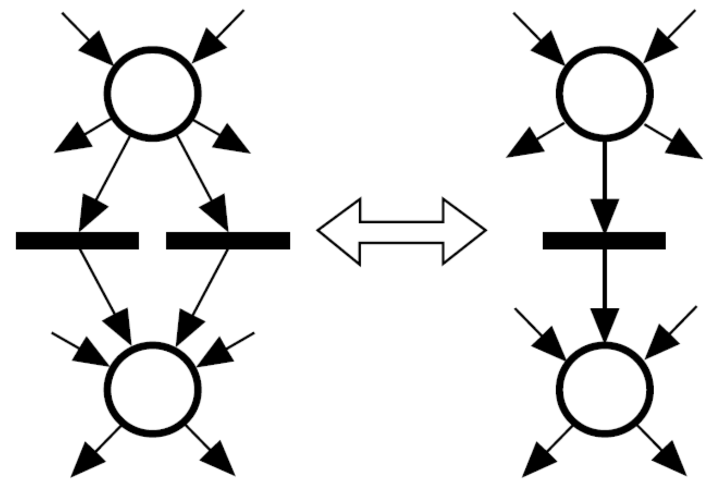
RA1. Fusion of series places



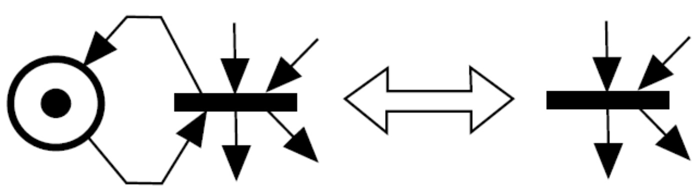
RA2. Fusion of series transitions



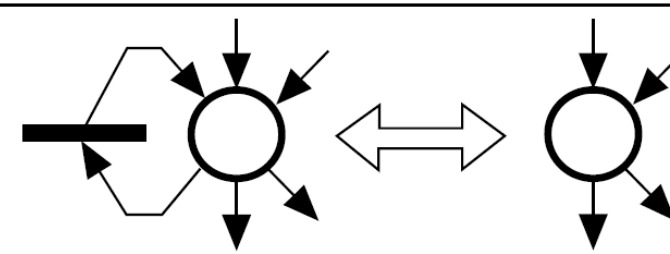
RB1. Elimination of identical place



RB2. Elimination of identical transition

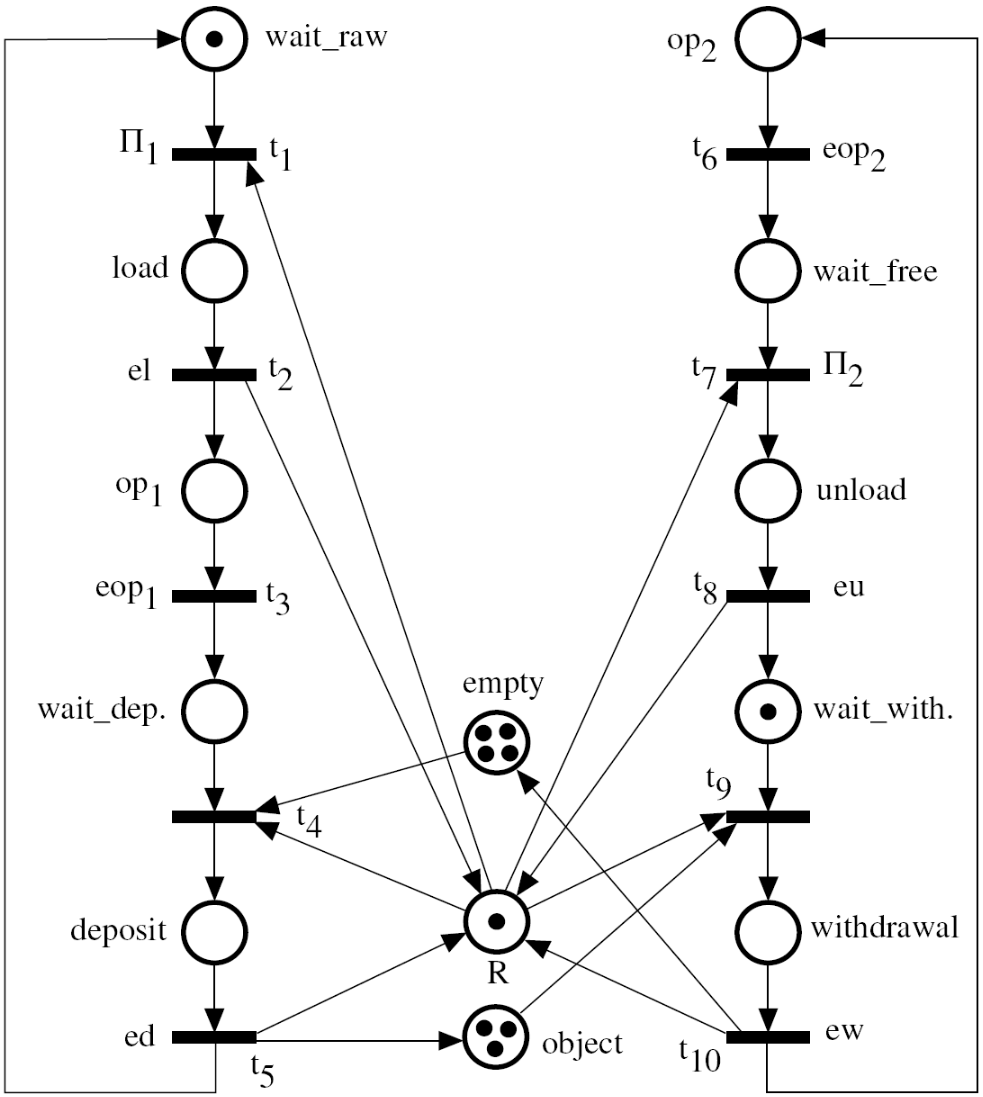


RC1. Elimination of self-loop place

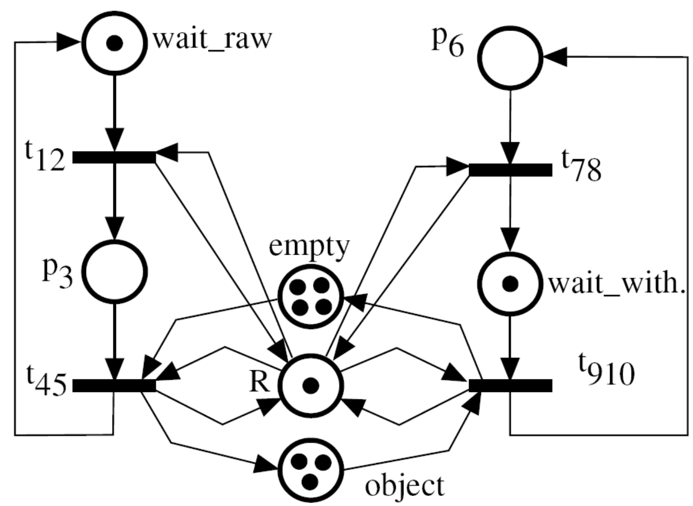


RC2. Elimination of self-loop transition

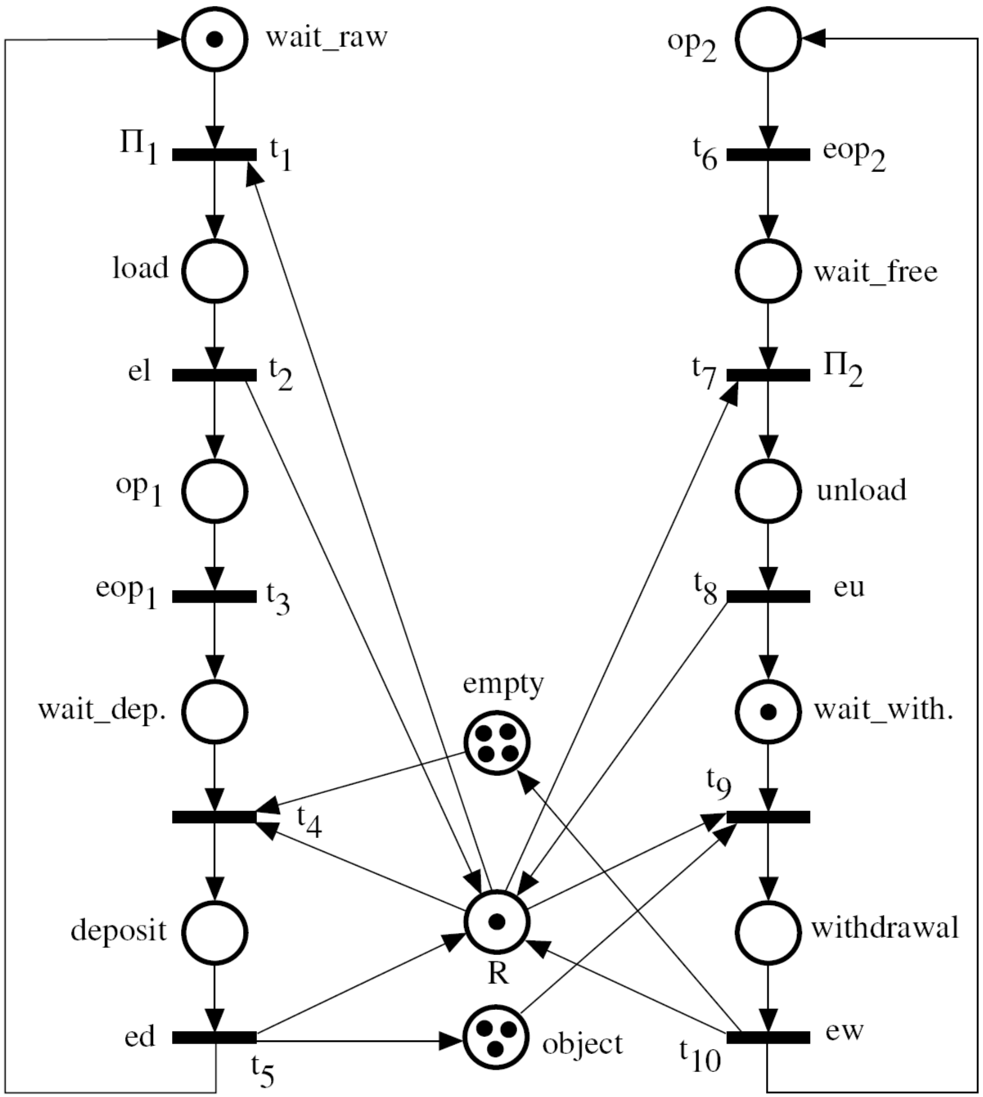
Example



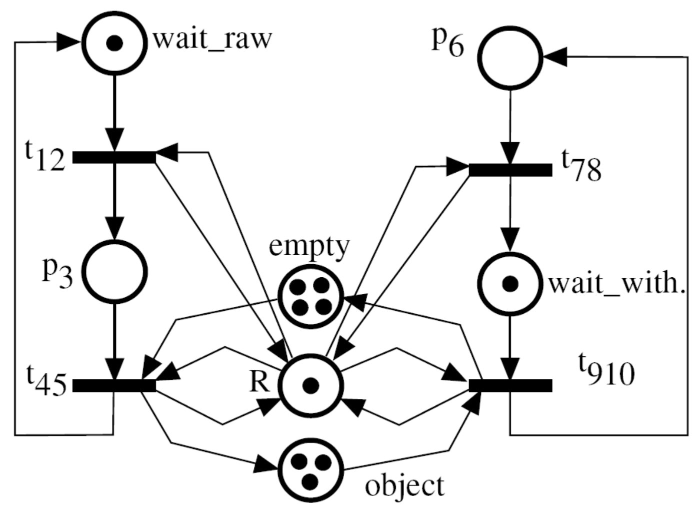
Rule applied?



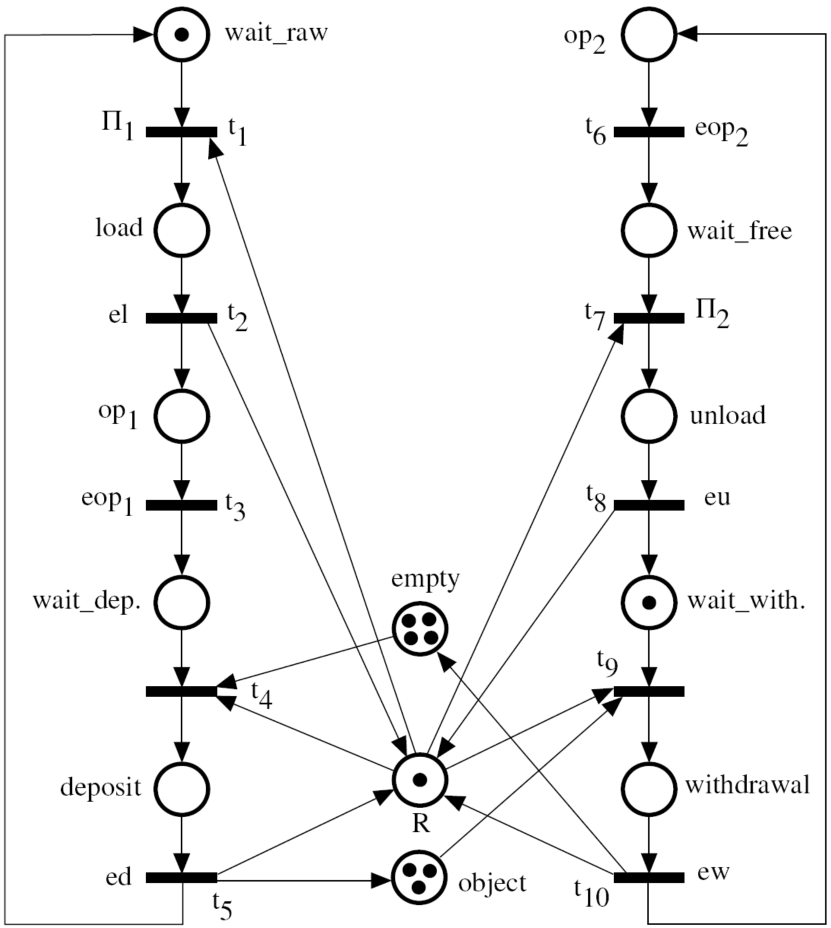
Example

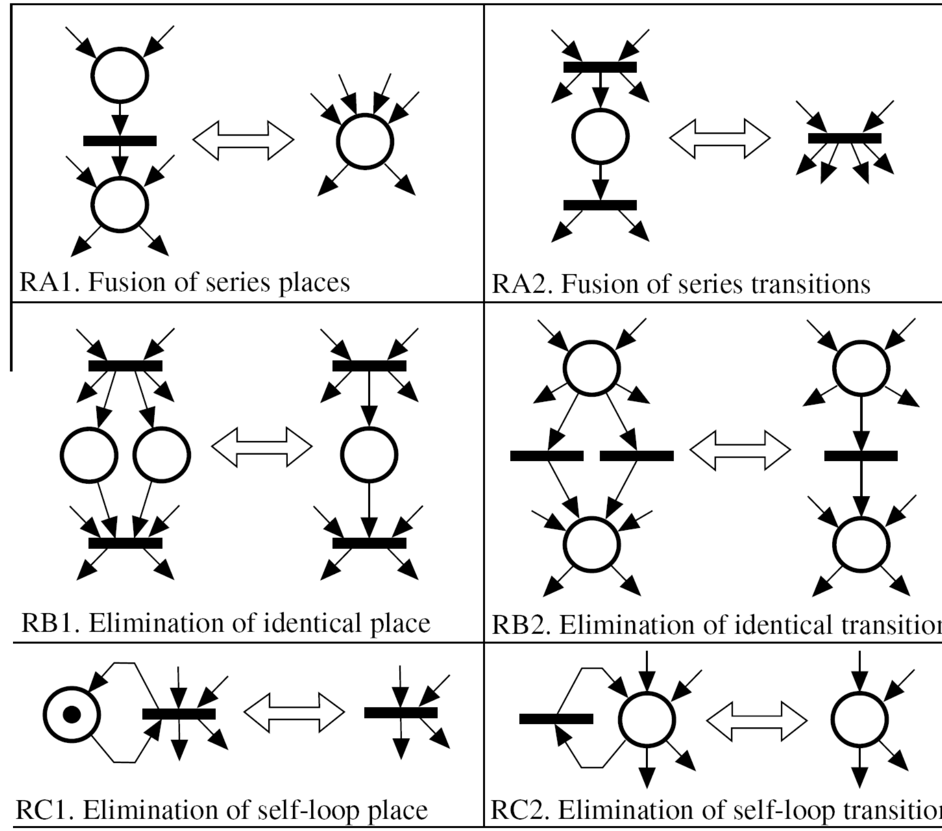


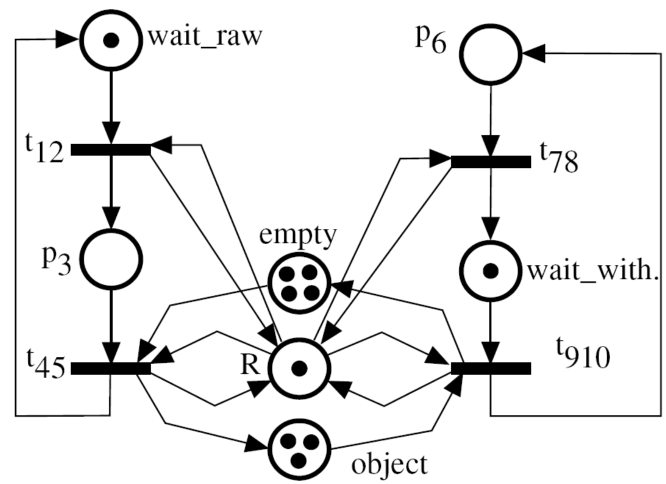
Rule applied?



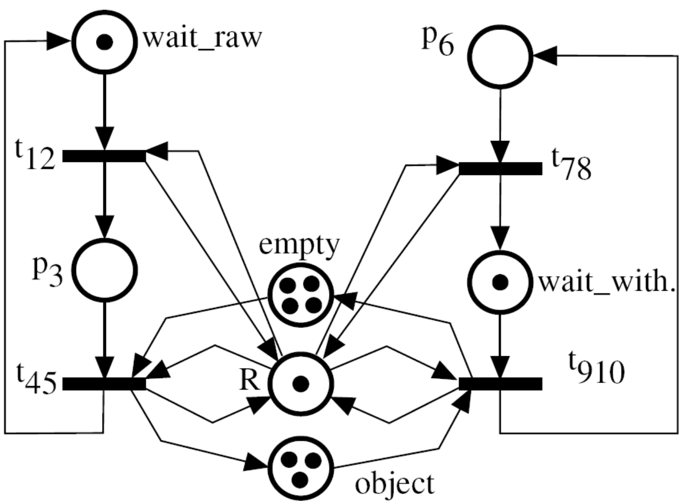
Reduction techniques

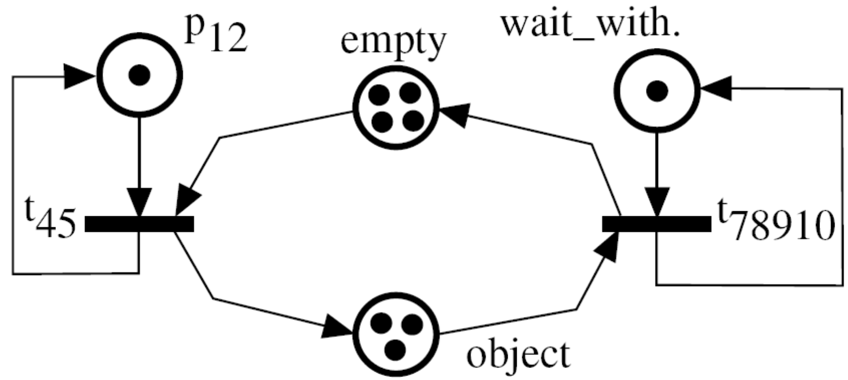


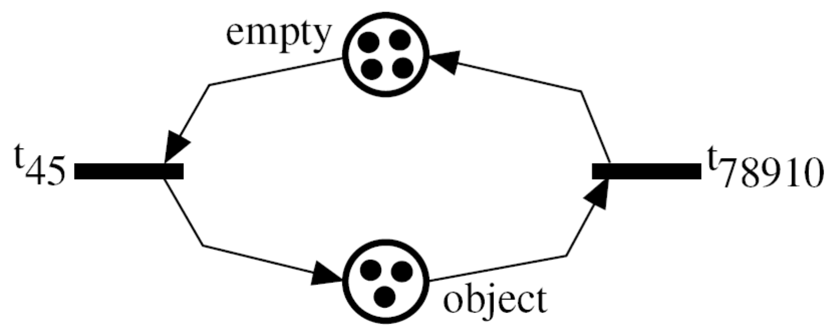




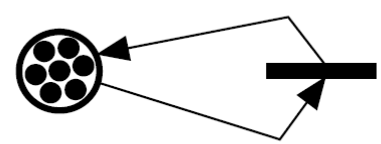
Example

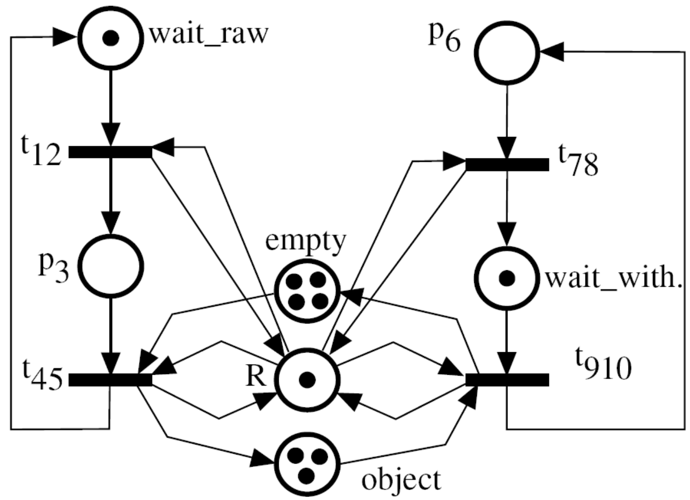






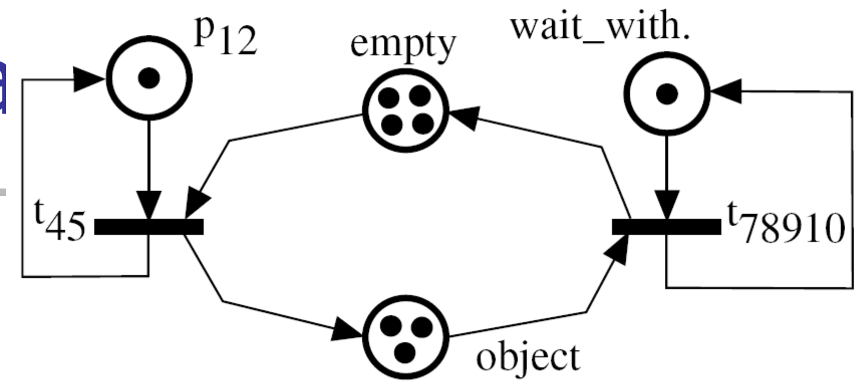
Net is live/has home states and is 7-bounded



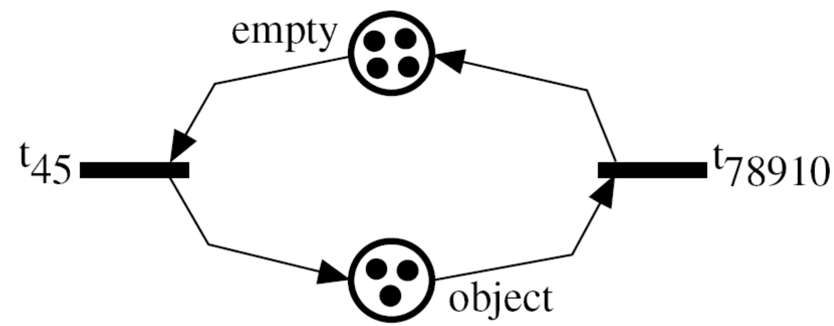


wait_raw
p_6
t_{12}
t_{78}
p_3
empty
wait_with.
t_{45}
R
t_{910}
object

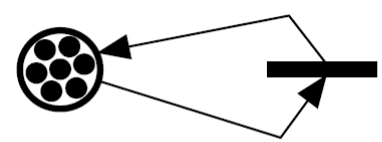
Exa

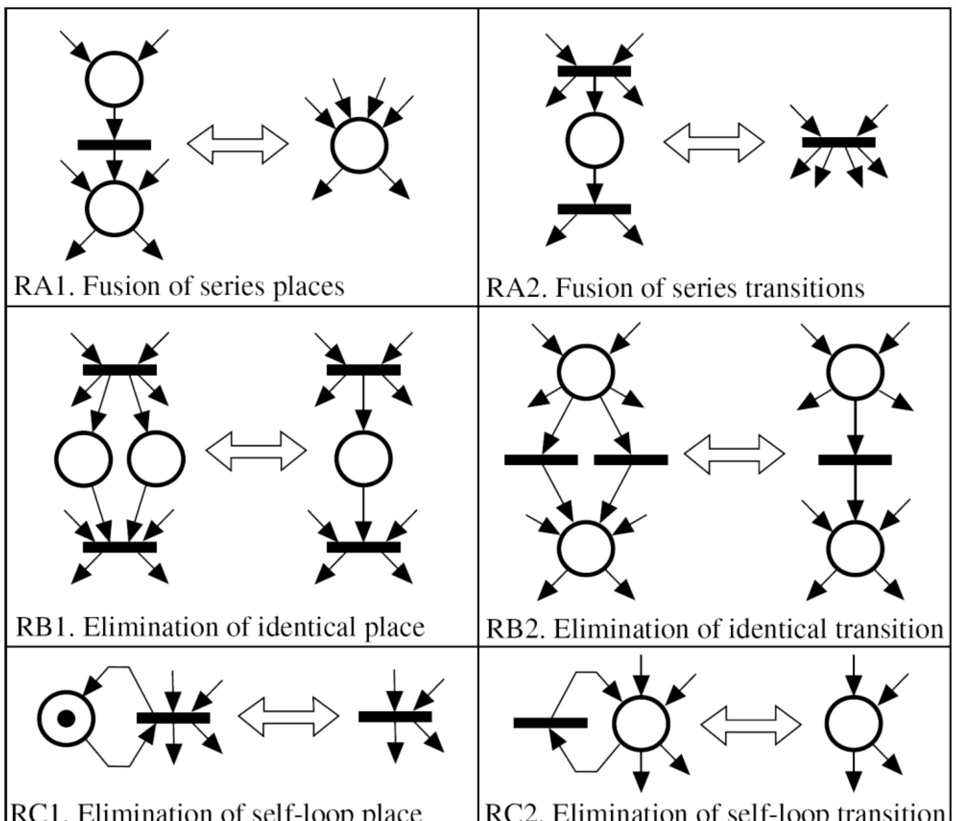


p_{12}
empty
wait_with.
t_{45}
t_{78910}
object



empty
t_{45}
t_{78910}
object





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

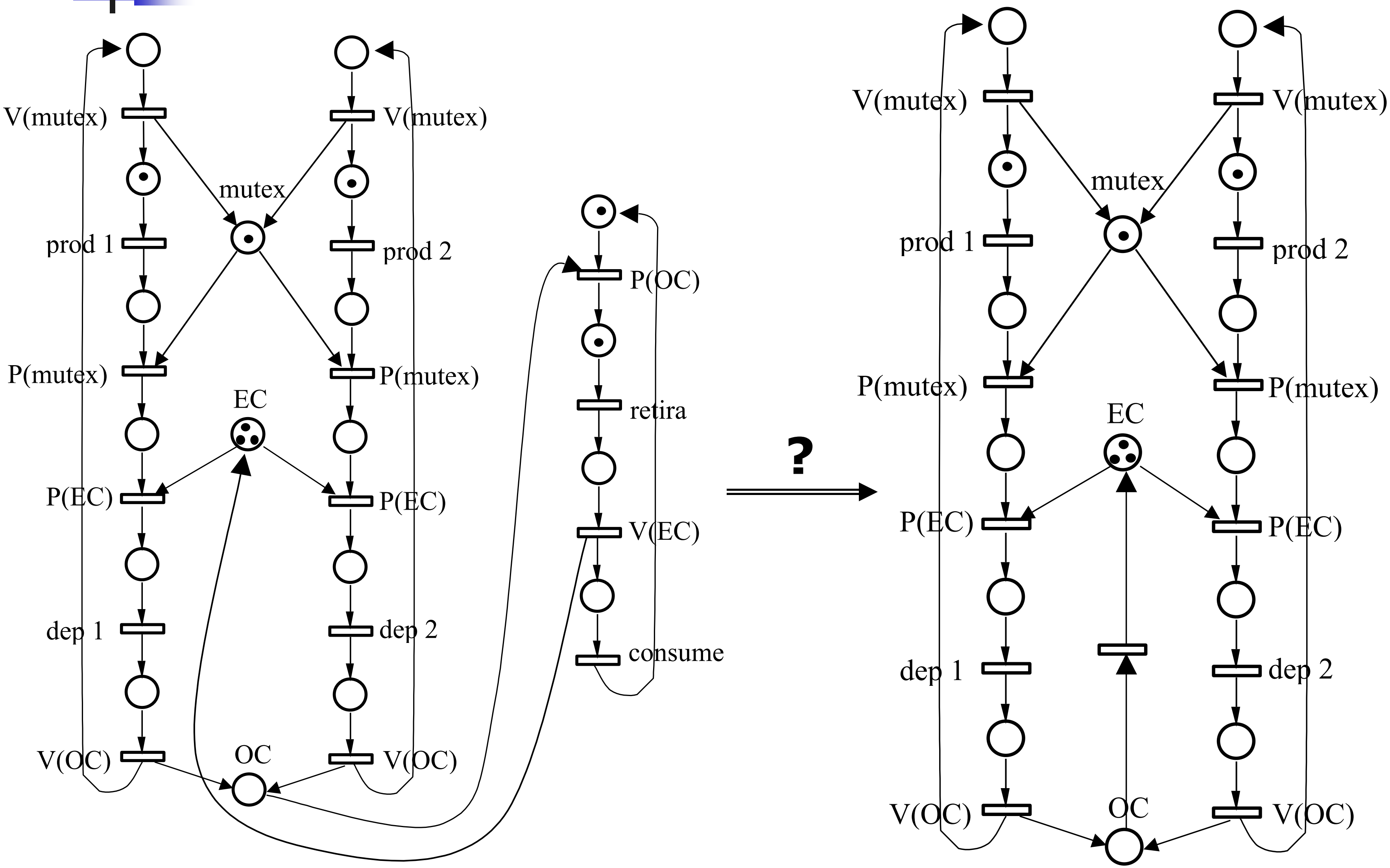
ıd

Example

Example: a system with two producers and one consumer

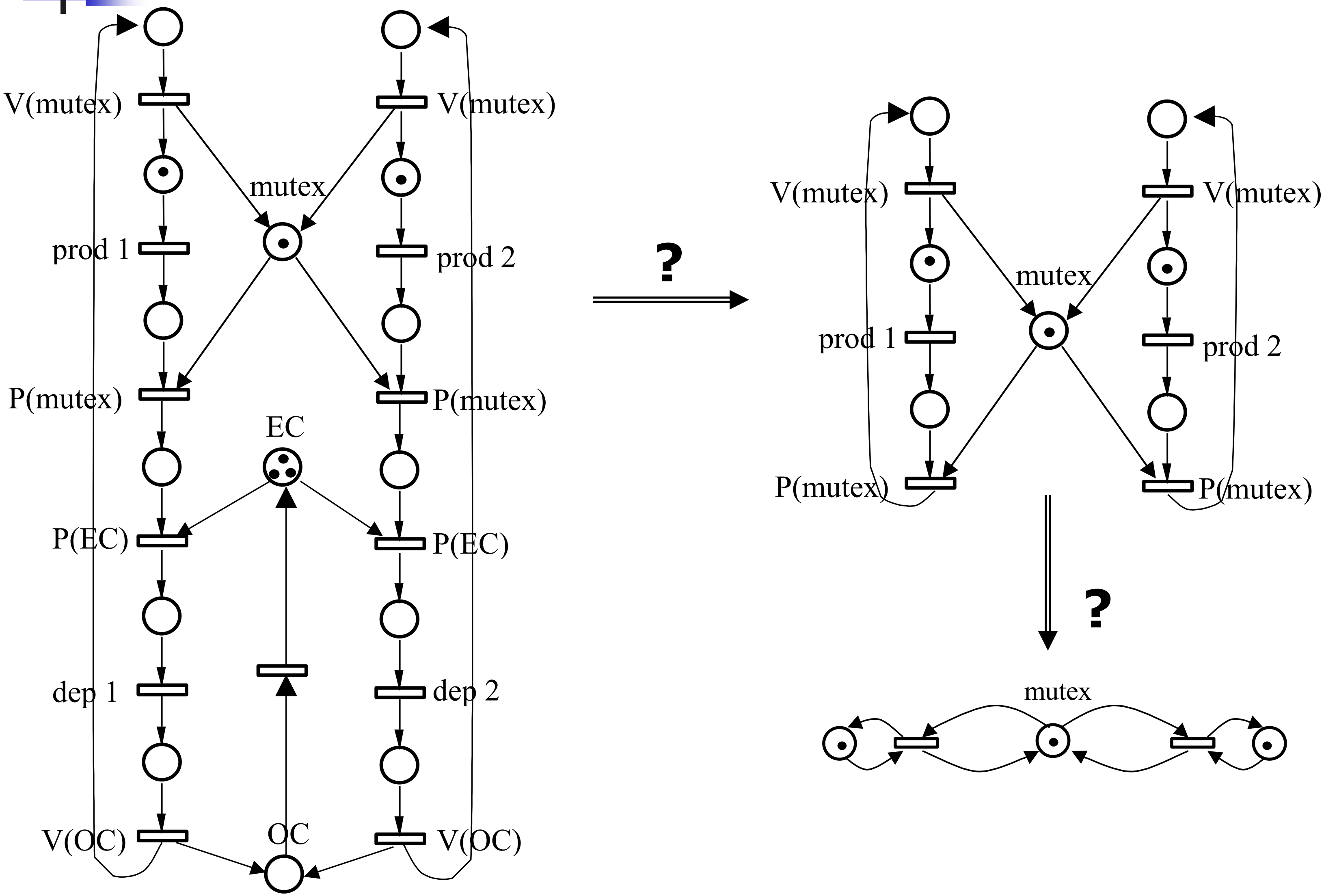
Producer 1	Producer 2	Consumer
loop produce 1 P(mutex) P(EC) deposit 1 V(OC) V(mutex) endloop	loop produce 2 P(mutex) P(EC) deposit 2 V(OC) V(mutex) endloop	loop P(OC) get V(EC) consume endloop

Example



V(mutex)
V(mutex)
mutex
prod 1
prod 2
P(mutex)
P(mutex)
EC
P(EC)
P(EC)
dep 1
dep 2
V(OC)
OC
V(OC)
P(OC)
retira
V(EC)
consume
?
V(mutex)
V(mutex)
mutex
prod 1
prod 2
P(mutex)
P(mutex)
EC
P(EC)
P(EC)
dep 1
dep 2
V(OC)
OC
V(OC)

Example


V(mutex)
V(mutex)
mutex
prod 1
prod 2
P(mutex)
P(mutex)
EC
P(EC)
P(EC)
dep 1
dep 2
V(OC)
OC
V(OC)
?
V(mutex)
V(mutex)
mutex
prod 1
prod 2
P(mutex)
P(mutex)
?
mutex

Structural techniques

Analysis technique based only on the structure N of the system are called “structural”

M0 can play a role, but the complexity of the analysis depends only on the incidence matrix C, and not on the initial marking M0.

Two classes of techniques:

- based on linear programming (convex geometry)
- based on topological properties of the graph

Structural techniques – convex geometry

The idea behind this type of analysis is to use the State Equation (SE) to verify the property.

Example: $\forall \mathbf{m} \in \mathrm{RS}(\mathcal{S}) : \mathbf{m}[p] = 0 \vee \mathbf{m}[p'] = 0;$

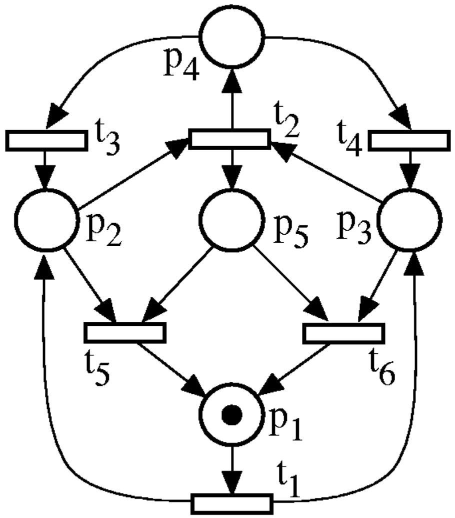
can be reduced to the ABSENCE of solution for

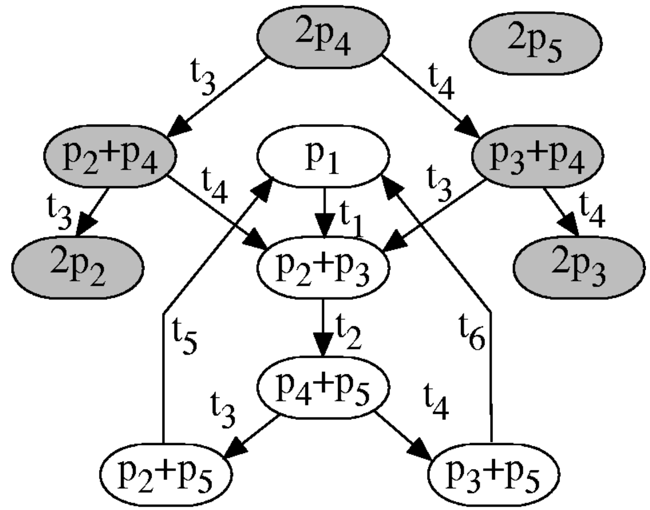
$$\{\mathbf{m} = \mathbf{m}_0 + \mathbf{C} \cdot \boldsymbol{\sigma} \wedge \mathbf{m}[p] > 0 \wedge \mathbf{m}[p'] > 0\}$$

This approach leads to semi-decidable procedure, since

$$\mathrm{RS}(\mathcal{S}) \subset \mathrm{LRS}^{SE}(\mathcal{S})$$

Structural techniques – spurious solution





Example: we cannot conclude that p2 and p4 are in p-mutex

Structural techniques

Def: a p-flow is a vector y:P→Q s.t., *y*.C = 0

Def: a t-flow is a vector x:T→Q *s.t.,* C.*x* = 0

Flows form a *vector space*, and can be generated from a *basis*

Def: a non-negative p-flow is a p-semiflow

Def: a non-negative t-flow is a t-semiflow

Def: the support ||y|| of a p-semiflow y is

$$||\mathbf{y}|| = \{p \in P | \mathbf{y}[p] > 0\}$$

Change p- in t- to get the dual definitions

Structural techniques

Def: a net is *conservative* if there exists at least one p-semiflow y such that $||y||=P$

Def: a net is *consistent* if there exists at least one t-semiflow x such that $||x||=T$

The set of canonical semiflow can be infinite

Def: a p-semiflow is *canonical* if the g.c.d. of its non null elements is 1

Def: a *generator* set of p-semiflows $\Psi = \{y_1,\ldots, y_n\}$ is the set of the least number of p-semiflows such that, for any p-semiflow y

$$y = \sum_{\mathbf{y}_j \in \Psi} k_j \cdot \mathbf{y}_j,\ k_j \in \mathbb{Q} \text{ and } \mathbf{y}_j \in \Psi.$$

and the p-semiflows of Ψ =are said to be "minimal"

Structural techniques

Proposition: a semiflow is minimal iff it is canonical and its support does not contain stricty the support of any other p-semiflow. Moreover the set of minimal semiflow of a net is finite and unique

but……minimal p-semiflow can be exponential in C, therefore their computation cannot be polynomial (although very often this number is “small”)

Note: GreatSPN computes minimal p- and t-semiflows

Other option: to compute a set of canonicals whose support “covers” P (or T), as done in the tool INA

Structural techniques

From semiflow we can generate linear invariants.

From p-semiflow we obtain the "*token conservation law*":

$$\mathbf{y} \in \mathbb{N}^n,\ \mathbf{y} \cdot \mathbf{C} = 0 \Longrightarrow$$

$$\forall \mathbf{m_0},\ \forall \mathbf{m} \in \mathrm{RS}(\mathcal{N}, \mathbf{m_0}), \mathbf{y} \cdot \mathbf{m} = \mathbf{y} \cdot \mathbf{m_0}$$

The weighted sum of tokens is constant for all reachable markings

From t-semiflows we obtain the "*cyclic behaviour law*":

$$\mathbf{x} \in \mathbb{N}^m,\ \mathbf{C} \cdot \mathbf{x} = 0 \Longrightarrow$$

$$\exists \mathbf{m_0},\ \exists \sigma \in \mathrm{L}(\mathcal{N}, \mathbf{m_0}) \text{ s.t } \mathbf{m_0} \xrightarrow{\sigma} \mathbf{m_0} \text{ and } \boldsymbol{\sigma} = \mathbf{x}$$

Structural techniques

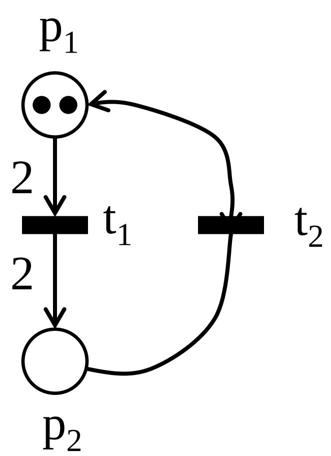
Note. The token conservation law can be proved observing that if $\mathbf{m}$ is a reachable marking, then $\mathbf{m} = \mathbf{m0} + \mathbf{C} \,.\boldsymbol{\sigma}$, and premultiplying by $\mathbf{y}$, we get

$$\mathbf{y}.\ \mathbf{m} = \mathbf{y}.\ \mathbf{m0} + \mathbf{y}.\mathbf{C}.\ \boldsymbol{\sigma} = \mathbf{y}.\ \mathbf{m0})$$

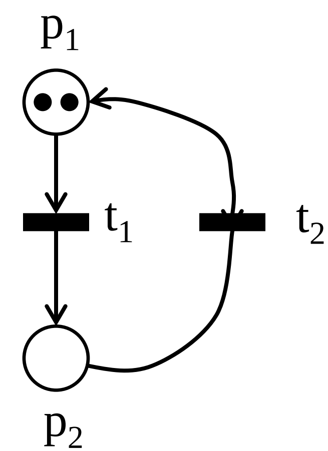
Similarly, we can observe that, if $\mathbf{m}$ is a reachable marking, and σ is a firing sequence firable in $\mathbf{m}$, of firing vector $\boldsymbol{x}$, which is a T-semiflow, then $\mathbf{m}\ [\sigma > \mathbf{m}$

(indeed if $\boldsymbol{x}$ is a *T-semiflow*, $\mathbf{m'} = \mathbf{m} + \mathbf{C}\ .\ \mathbf{x} = \mathbf{m}$)

Example

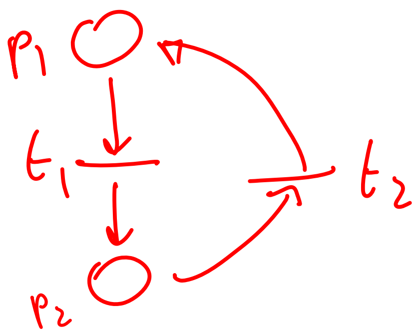


$\neq$



T-inv:	$t_1 + 2t_2$	$t_1 + t_2$
P-inv:	$p_1 + p_2 = 2$	$p_1 + p_2 = 2$

Calcolo di P-semiflow



$$C = \begin{array}{c|cc} & t_1 & t_2 \\ \hline p_1 & -1 & +1 \\ p_2 & +1 & -1 \end{array}$$

$$y \cdot C = 0$$

$$y = \begin{array}{cc} p_1 & p_2 \\ \left[y_1 \right. & \left. y_2 \right] \end{array}$$

$$-y_1 + y_2 = 0$$

$$y_1 - y_2 = 0$$

$$y_1 = 1$$

$$y_2 = 1$$

$$m : [m_1, m_2]$$

$$m_0 = [0, 2]$$

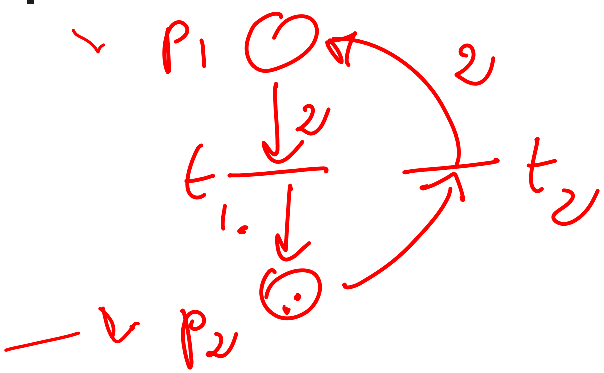
$$y \cdot m = y \cdot m_0$$

$$y_1 \cdot m_1 + y_2 \cdot m_2 = y_1 \cdot 0 + y_2 \cdot 2$$

$$\Downarrow \quad y_1 = y_2 = 1$$

$$m_1 + m_2 = 2$$

Calcolo di P-semiflow



C	t_1	t_2
p_1	-2	$+2$
p_2	$+1$	-1

$$-2y_1 + y_2 = 0$$

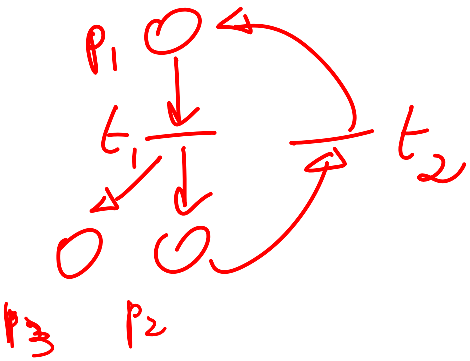
$$2y_1 - y_2 = 0$$

$$y_1 = 1 \qquad y_2 = 2$$

$$y_1 \cdot m_1 + y_2 \cdot m_2 = \underbrace{y_1 \cdot 0 + y_2 \cdot 2}$$

$$\boxed{m_1 + 2\, m_2 = 4} \Longrightarrow \begin{array}{l} \# p_1 \leq 4 \\ \# p_2 \leq 2 \end{array}$$

Esercizi



$$
\begin{array}{c|cc}
 & t_1 & t_2 \\
\hline
p_1 & -1 & +1 \\
p_2 & +1 & -1 \\
p_3 & +1 & 0
\end{array}
$$

$$
\begin{array}{ccc}
p_1 & p_2 & p_3
\end{array}
$$

$$y = [y_1 \; y_2 \; y_3]$$

$$y \cdot C = 0$$

Calcolare i P-semiflussi di questa rete

Calcolare la caratterizzazione lineare dello spazio degli stati delle due reti

$$m = m^0 + C \cdot \sigma$$

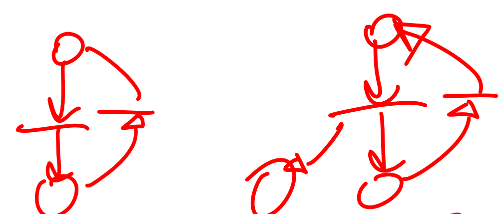
σ e m incognite

calcolare LRS^{BE}

$$m_1 = m_1^0 + \ldots$$

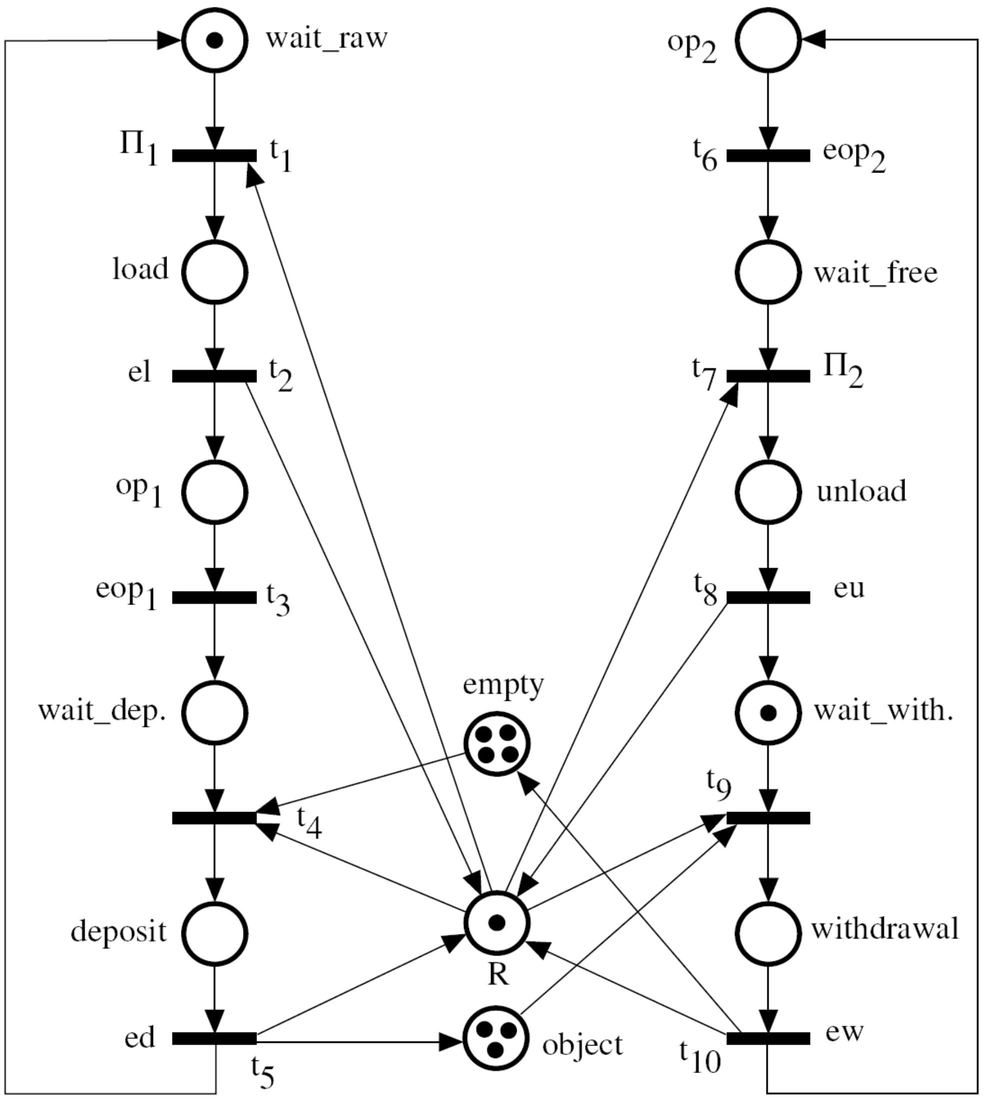
$$m_2 = m_2^0 + \ldots$$

per due reti



$m_0 = [0, 1] \circ [0, 2]$

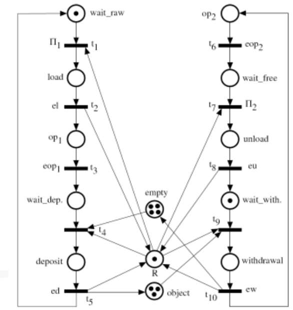
Structural techniques: examples



- **m**wait_raw + **m**load + **m**op1 + **m**wait_dep + **m**deposit = 1
- **m**deposit + **m**object+ **m**withdrawal + **m**empty = 7
- **m**op2 +**m**wait_free +**m**unload +**m**wait_with +**m**withdrawal = 1
- **m**R +**m**load + **m**deposit + **m**unload + **m**withdrawal = 1

NOTE: all weights of the semiflows are 1

Structural techniques: examples



$$\mathbf{m}[\text{wait\_raw}]+\mathbf{m}[\text{load}]+\mathbf{m}[\text{op}_1]+\mathbf{m}[\text{wait\_dep.}]+\mathbf{m}[\text{deposit}]=1$$

$$\mathbf{m}[\text{op}_2]+\mathbf{m}[\text{wait\_free}]+\mathbf{m}[\text{unload}]+\mathbf{m}[\text{wait\_with.}]+\mathbf{m}[\text{withdrawal}]=1$$

$$\mathbf{m}[\text{empty}]+\mathbf{m}[\text{deposit}]+\mathbf{m}[\text{object}]+\mathbf{m}[\text{withdrawal}]=7$$

$$\mathbf{m}[\text{R}]+\mathbf{m}[\text{load}]+\mathbf{m}[\text{unload}]+\mathbf{m}[\text{deposit}]+\mathbf{m}[\text{withdrawal}]=1$$

Consequences:

1. Bounds: $\forall p_i \in P \setminus \{\text{empty}, \text{object}\}$, $\mathbf{m}[p_i] \leq 1$; $\mathbf{m}[\text{empty}] \leq 7$; and $\mathbf{m}[\text{object}] \leq 7$.

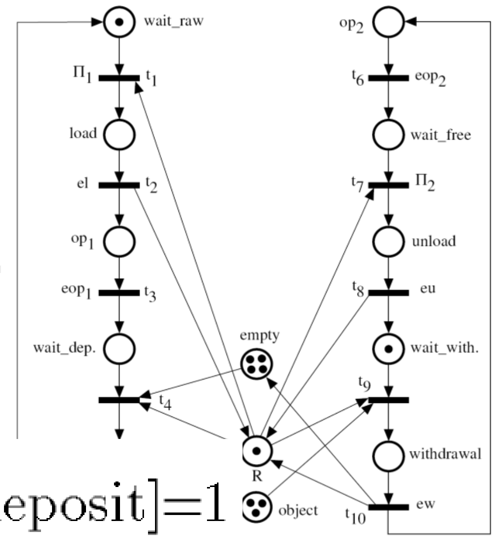
2. The places in each one of the following sets are in marking mutual exclusion:

   a) $\{\text{wait\_raw}, \text{load}, \text{op}_1, \text{wait\_dep.}, \text{deposit}\}$

   b) $\{\text{op}_2, \text{wait\_free}, \text{unload}, \text{wait\_with.}, \text{withdrawal}\}$

   c) $\{\text{R}, \text{load}, \text{unload}, \text{deposit}, \text{withdrawal}\}$

Structural techniques: examples



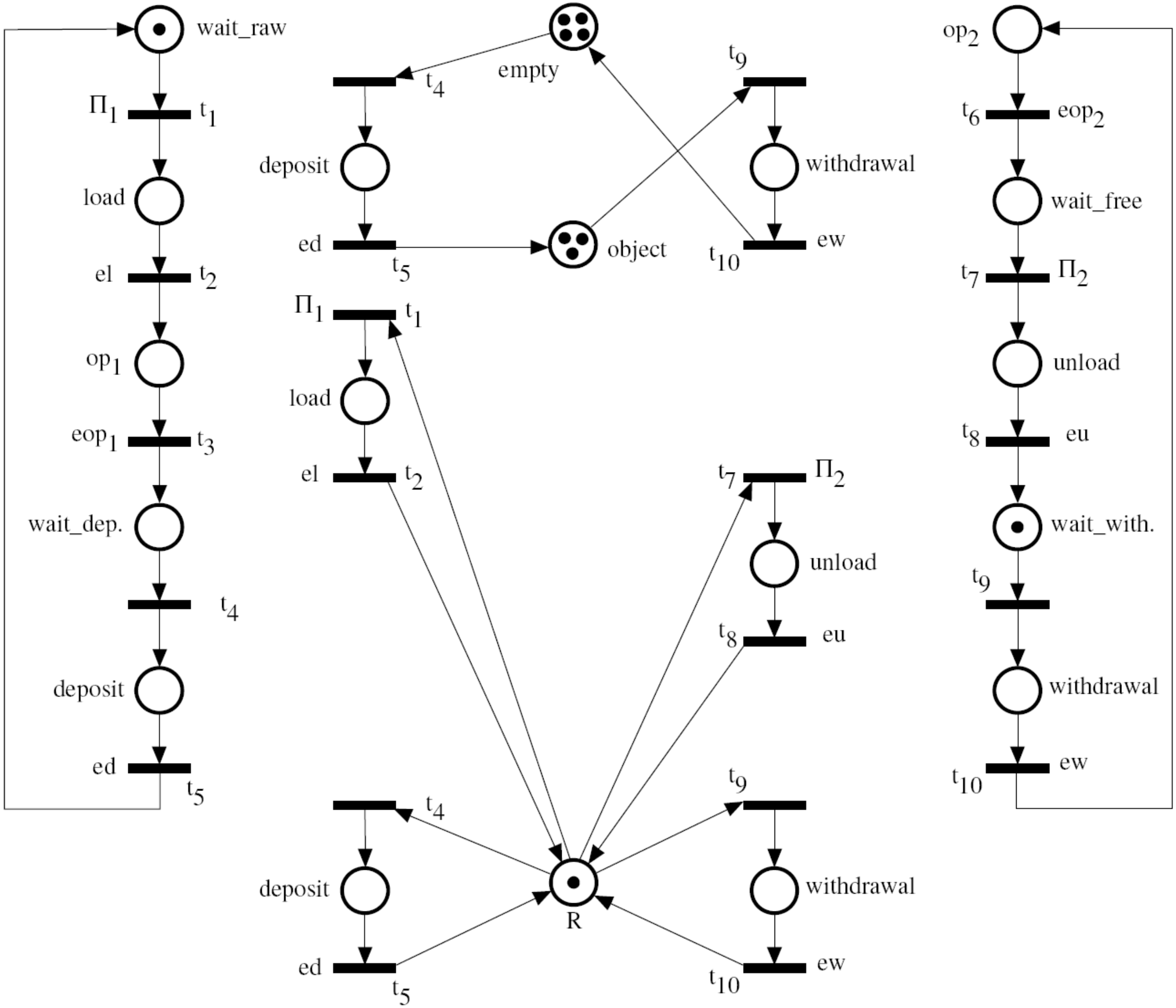
$$\mathbf{m}[\text{wait\_raw}]+\mathbf{m}[\text{load}]+\mathbf{m}[\text{op}_1]+\mathbf{m}[\text{wait\_dep.}]+\mathbf{m}[\text{deposit}]=1$$

$$\mathbf{m}[\text{op}_2]+\mathbf{m}[\text{wait\_free}]+\mathbf{m}[\text{unload}]+\mathbf{m}[\text{wait\_with.}]+\mathbf{m}[\text{withdrawal}]=1$$

$$\mathbf{m}[\text{empty}]+\mathbf{m}[\text{deposit}]+\mathbf{m}[\text{object}]+\mathbf{m}[\text{withdrawal}]=7$$

$$\mathbf{m}[\text{R}]+\mathbf{m}[\text{load}]+\mathbf{m}[\text{unload}]+\mathbf{m}[\text{deposit}]+\mathbf{m}[\text{withdrawal}]=1$$

Since P-semiflows are non negative, we can use them to get a decomposed view of the system



wait_raw
Π_1 t_1
load
el t_2
op_1
eop_1 t_3
wait_dep.
t_4
deposit
ed
t_5
empty
t_4
deposit
ed
t_5
object
t_9
withdrawal
ew
t_{10}
op_2
t_6 eop_2
wait_free
t_7 Π_2
unload
t_8 eu
wait_with.
t_9
withdrawal
ew
t_{10}
Π_1 t_1
load
el t_2
t_7 Π_2
unload
t_8 eu
t_4
deposit
ed
t_5
R
t_9
withdrawal
ew
t_{10}

Invariants

Absence of deadlock:

for a state to be a deadlock, no token should be present in the places that are the unique enabling condition of a transition, therefore

$m_{load} = m_{op1} = m_{deposit} = m_{op2} = m_{unload} = m_{withdrawal} = 0$

and the invariants reduce to:

$m_{wait\_raw} + m_{wait\_dep} = \mathbf{1}$

$m_{wait\_free} + m_{wait\_with.} = \mathbf{1}$

$m_{empty} + m_{object} = \mathbf{7}$

$m_{R} = \mathbf{1}$

since R is marked, for no t to be enabled we must have that $m_{wait\_raw} = m_{wait\_free} = 0$, and therefore

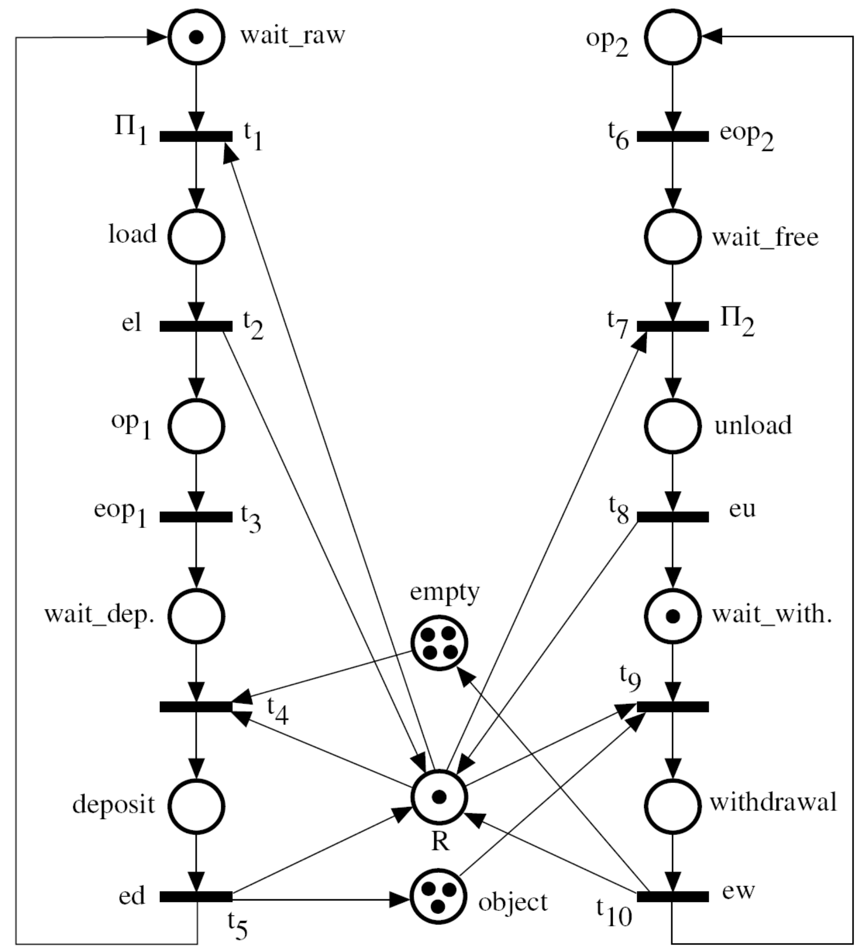
$m_{wait\_dep} = \mathbf{1}$

$m_{wait\_with.} = \mathbf{1}$

$m_{empty} + m_{object} = \mathbf{7}$

$m_{R} = \mathbf{1}$

And to avoid the firing of t4 and t9, it is necessary that $m_{empty} + m_{object} = 0$, which violates the condition $m_{empty} + m_{object} = \mathbf{7}$ **above**



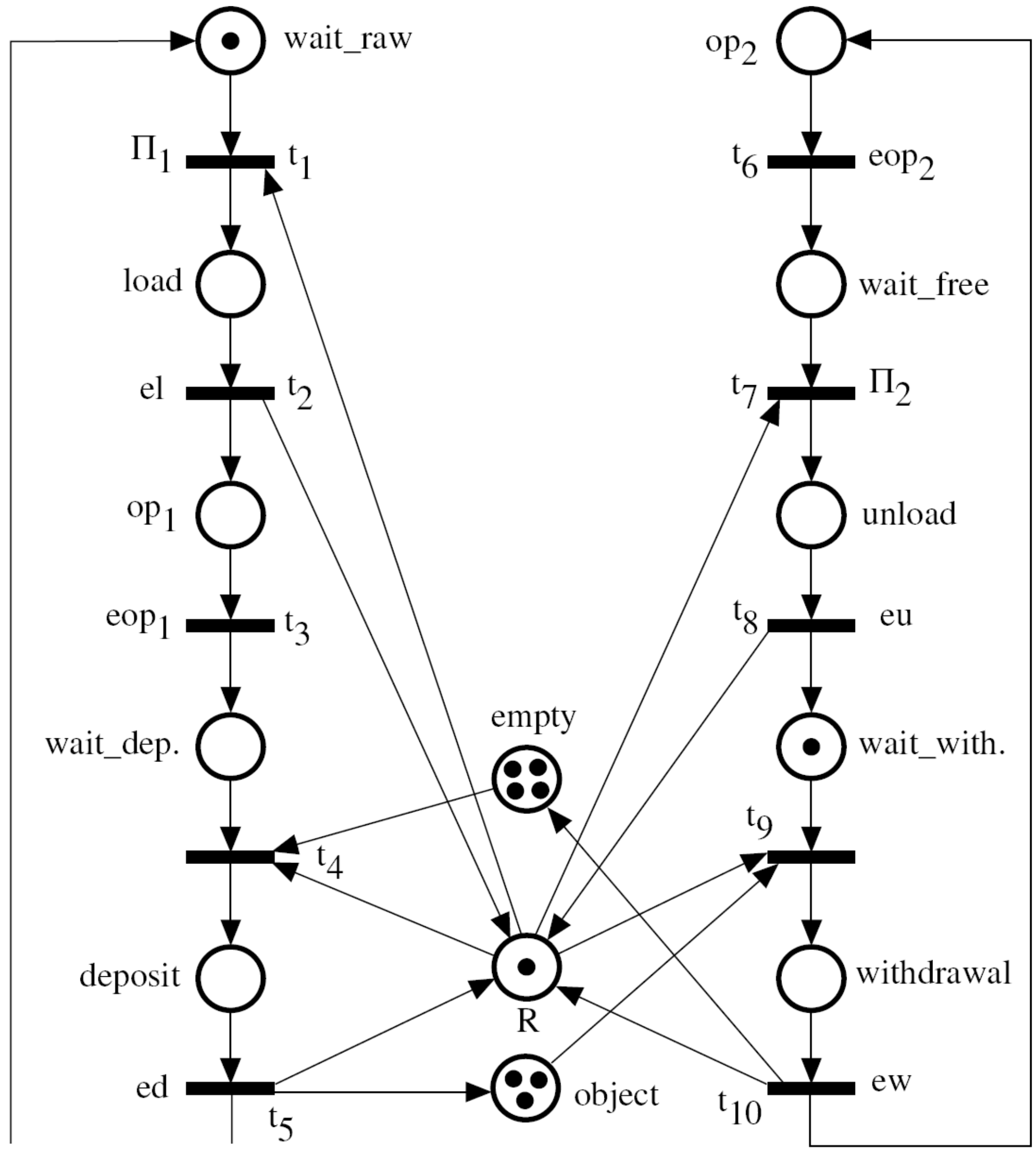
$$\mathbf{m}[\text{wait\_raw}]+\mathbf{m}[\text{load}]+\mathbf{m}[\text{op}_1]+\mathbf{m}[\text{wait\_dep.}]+\mathbf{m}[\text{deposit}]=1$$

$$\mathbf{m}[\text{op}_2]+\mathbf{m}[\text{wait\_free}]+\mathbf{m}[\text{unload}]+\mathbf{m}[\text{wait\_with.}]+\mathbf{m}[\text{withdrawal}]=1$$

$$\mathbf{m}[\text{empty}]+\mathbf{m}[\text{deposit}]+\mathbf{m}[\text{object}]+\mathbf{m}[\text{withdrawal}]=7$$

$$\mathbf{m}[\text{R}]+\mathbf{m}[\text{load}]+\mathbf{m}[\text{unload}]+\mathbf{m}[\text{deposit}]+\mathbf{m}[\text{withdrawal}]=1$$

Invariants



Absence of deadlock – more compat terms:

if $m_{load} + m_{op1} + m_{deposit} + m_{op2} + m_{unload} + m_{withdrawal} \geq 1$

then one of *(t2 or t3 or t5 or t6 or t8 or t10) is firable*

if instead $m_{wait\_raw} + m_{wait\_free} \geq 1$

then *(t1 or t7) is firable*

else *(t4 or t9) is firable*

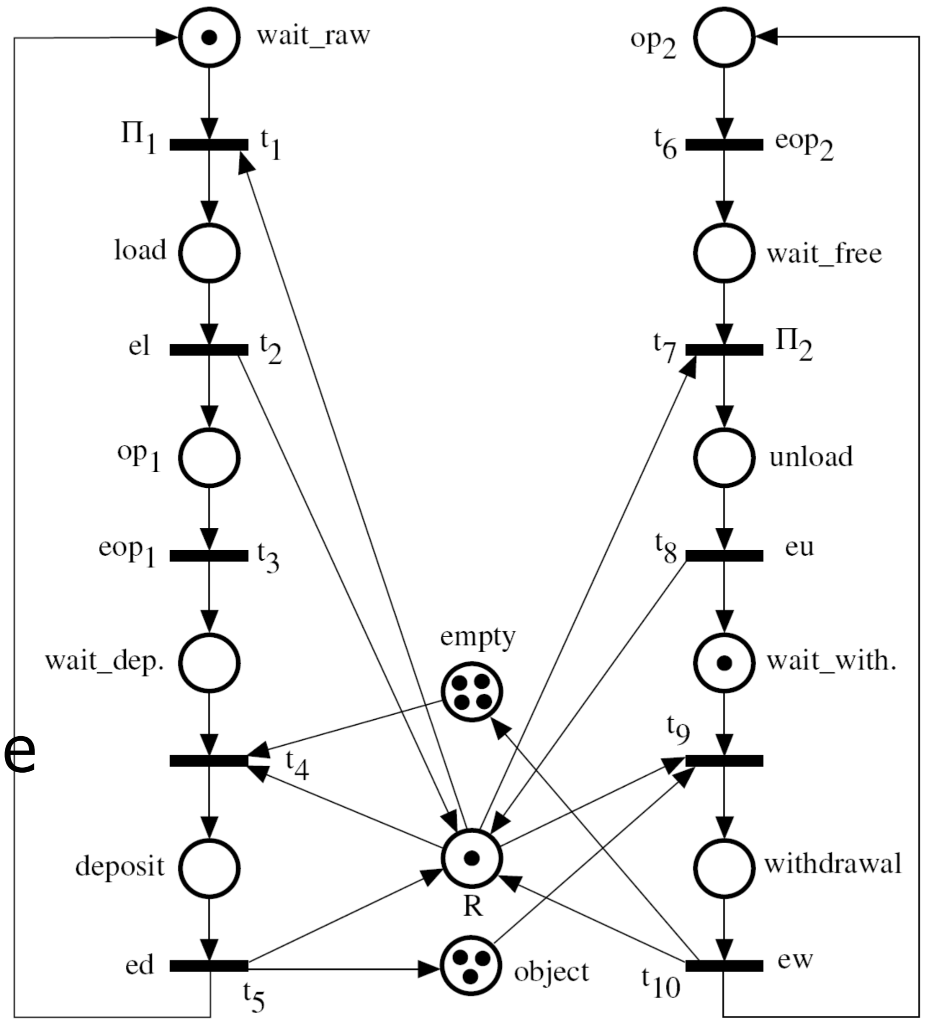
$$\mathbf{m}[\text{wait\_raw}]+\mathbf{m}[\text{load}]+\mathbf{m}[\text{op}_1]+\mathbf{m}[\text{wait\_dep.}]+\mathbf{m}[\text{deposit}]=1$$

$$\mathbf{m}[\text{op}_2]+\mathbf{m}[\text{wait\_free}]+\mathbf{m}[\text{unload}]+\mathbf{m}[\text{wait\_with.}]+\mathbf{m}[\text{withdrawal}]=1$$

$$\mathbf{m}[\text{empty}]+\mathbf{m}[\text{deposit}]+\mathbf{m}[\text{object}]+\mathbf{m}[\text{withdrawal}]=7$$

$$\mathbf{m}[\text{R}]+\mathbf{m}[\text{load}]+\mathbf{m}[\text{unload}]+\mathbf{m}[\text{deposit}]+\mathbf{m}[\text{withdrawal}]=1$$

Invariants



Liveness: since

$\sigma =$ t1 t2 t3 t4 t5 t9 t10 t6 t7 t8

is firable, then the net is reversible

Fairness: since the net has a single right annuller: $x = (1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1)$

then, for all possible scheduling Π1 and Π2, all components work

Structural techniques:

If a P/T system is bounded and live, then

$$\text{rank}(C) < |\Phi|$$

where Φ is the set of equivalence classes built on the equal conflict relation, relation defined as:

$$t \text{ eq } t' \text{ iff } Pre(t) = Pre(t')$$

Linear Programming and Petri Nets

Boundedness can be characterized also as a LP problem, defining the structural bound sb(p) as:

$$\mathbf{sb}(p) = \sup\{\mathbf{m}(p) | \mathbf{m} = \mathbf{m_0} + \mathbf{C} \cdot \boldsymbol{\sigma} \geq 0, \boldsymbol{\sigma} \geq 0\}$$

that leads to

$$\begin{array}{rll} \mathbf{sb}(p) = & \max. & \mathbf{e_p} \cdot \mathbf{m} \\ & \text{s.t.} & \mathbf{m} = \mathbf{m_0} + \mathbf{C} \cdot \boldsymbol{\sigma} \geq 0 \\ & & \boldsymbol{\sigma} \geq 0 \end{array}$$

Because $\mathrm{RS}(\mathcal{S}) \subset \mathrm{LRS}^{SE}(\mathcal{S})$, in general, we have that $\mathbf{sb}(p) \geq \mathbf{b}(p)$

Linear Programming and Petri Nets

the token variation due to transitions is not positive

Various characterization of boundedness

Property 6.5 *The following three statements are equivalent:*

1. *p is structurally bounded, i.e. p is bounded for any $\mathbf{m_0}$.*
2. *There exists $\mathbf{y} \geq \mathbf{e_p}$ such that $\mathbf{y} \cdot \mathbf{C} \leq 0$. (place-based characterization)*
3. *For all $\mathbf{x} \geq 0$ such that $\mathbf{C} \cdot \mathbf{x} \geq 0$, $\mathbf{C}[p, T] \cdot \mathbf{x} = 0$. (transition-based characterization)*

variation on p due to the firing of x is null

Structural techniques:

Property 6.5 *The following three statements are equivalent:*

1. *p is structurally bounded, i.e. p is bounded for any $\mathbf{m_0}$.*
2. *There exists $\mathbf{y} \geq \mathbf{e_p}$ such that $\mathbf{y} \cdot \mathbf{C} \leq 0$. (place-based characterization)*
3. *For all $\mathbf{x} \geq 0$ such that $\mathbf{C} \cdot \mathbf{x} \geq 0$, $\mathbf{C}[p, T] \cdot \mathbf{x} = 0$. (transition-based characterization)*

Property 6.6 *The following three statements are equivalent:*

1. *$\mathcal{N}$ is structurally bounded, i.e. $\mathcal{N}$ is bounded for any $\mathbf{m_0}$.*
2. *There exists $\mathbf{y} \geq \mathbf{1}$ such that $\mathbf{y} \cdot \mathbf{C} \leq 0$. (place-based characterization)*
3. *For all $\mathbf{x} \geq 0$ such that $\mathbf{C} \cdot \mathbf{x} \geq 0$, $\mathbf{C} \cdot \mathbf{x} = 0$; i.e. $\nexists\, \mathbf{x} \geq 0$ s.t. $\mathbf{C} \cdot \mathbf{x} \not\geq 0$. (transition-based characterization)*

Analysis for super/sub classes of P/T nets

Superclasses: let N be a net with inhibitor arcs and/or priorities and N’ the corresponding net w/o inhibitors or priority, then

$$RG(N,m) \subseteq RG(N',m)$$

indeed the elimination of inh. and priorities allows for more behaviour

- safety properties true on N’ are true on N as well (e.g. place boundedness)
- liveness properties true on N’ might not be true on N (for ex., by removing inh. and priorities certain transitions that were dead may become enabled)

Analysis for super/sub classes of P/T nets

Analysis for super/sub classes of P/T nets

Subclasses: by limiting the structure of the net (state machine and marked graph) we have more powerful analysis algorithm. We consider the case of ordinary state machines and marked graphs.

State Machines

Boundedness: a connected state machines is covered by a single P-semiflow in which all places have weight 1. The net is conservative and therefore structurally bounded.

Liveness: A state machine N is live if and only if N is strongly connected and at least one place is marked in the initial marking

Analysis for super/sub classes of P/T nets

Marked graphs (we use the definition that says that an ordinary net is a marked graph if each place as 1! input transition and 1! output transition)

All elementary circuits of the net are P-semiflows (indeed no token can enter or leave a circuit).

A marked graph N is live if and only if all elementary circuits contain at least one place marked in the initial marking.

A marked graph N is structurally bounded if and only if N is strongly connected

Analysis for super/sub classes of P/T nets

Free choice.

Commoner's Theorem - 1972

A free choice system (N,m0) is live if and only if all *syphons* contains a *trap* marked in the initial marking

Syphon: $P' \subseteq P$: $\bullet P' \subseteq P'\bullet$ (all transitions putting tokens into P' also remove tokens) → it tends to empty

Trap: $P' \subseteq P$: $P'\bullet \subseteq \bullet P'$ (all transitions removing tokens into P' also put tokens) → it tends to trap tokens in it

Analysis for super/sub classes of P/T nets

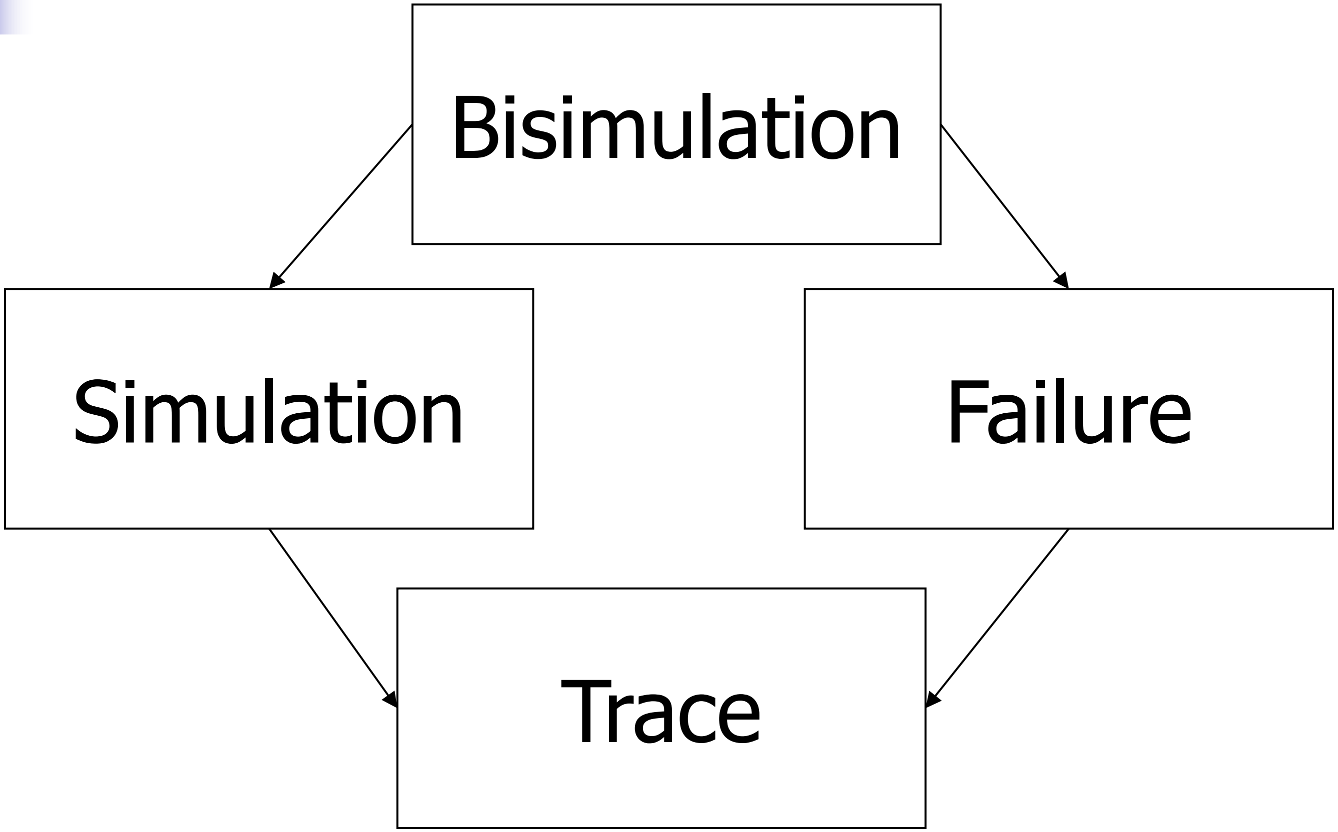
Free choice.

Rank theorem – Esparza et al 1992.

Let N be a free choice (equal conflict or DSSP) net. (N,m0) is live and bounded if and only if

- N is strongly connected and
- N is covered by state machines
- Rank(C)=$|\Phi|$ -1
- All syphons of N are initially marked

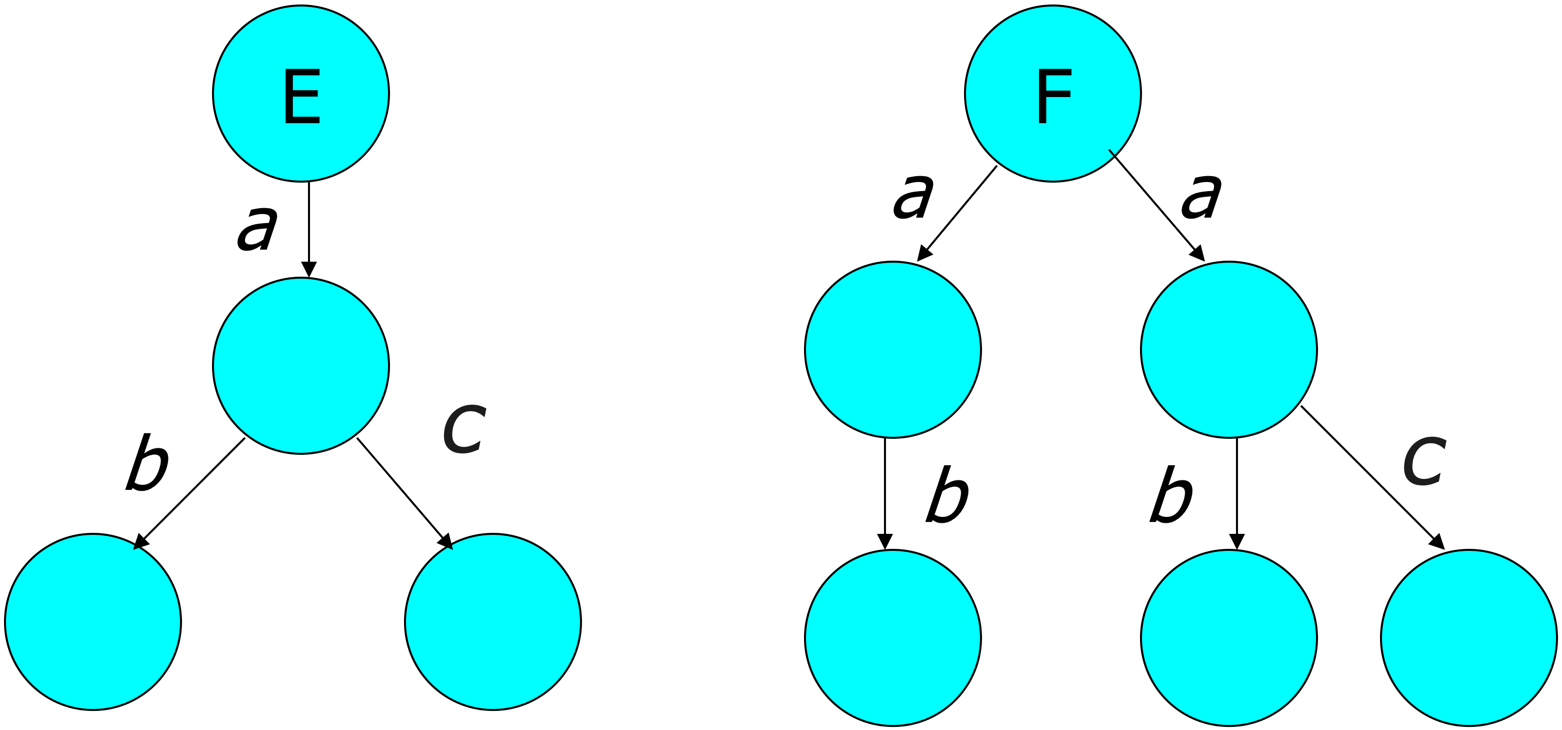
Hierarchy of equivalences



where and arrow from $\approx_1$ to $\approx_2$ ($\approx_1$ *more refined than* $\approx_2$) means:

$$P \approx_1 Q ==> P \approx_2 Q$$

Trace equivalence: Systems have same finite sequences.



E=*a*.(*b*+*c*)

F=(*a*.*b*)+*a*.(*b*+*c*)

Same traces

Trace equivalence: agents have same finite sequences.

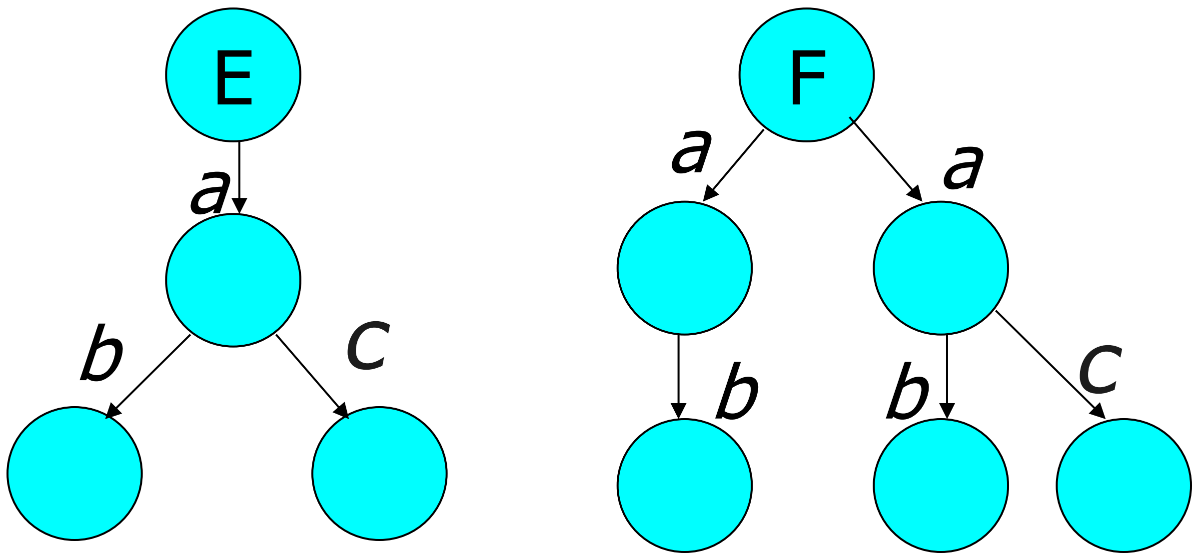
Def: the set T(E) of *traces* of the agent E is the set of all finite sequences that can be produced by the evolution of E

Def: if E and F are agents, we say that *E is equivalent to F according to trace equivalence,*

$$E \approx_{tr} F \qquad \text{iff} \qquad T(E) = T(F)$$

Note: For agents with a finite number of states equivalence over finite traces implies equivalence over infinite ones

Failures: comparing also what we cannot do after a finite sequence.



Failure of agent E: (σ, X), where after executing σ from E, none of the events in X is enabled. Agent F has failure (*a*, {*c*}), which is not a failure of E.

Failures: comparing also what we cannot do after a finite sequence.

A failure for an agent E is a pair (σ, X), with

- $\sigma \in T(E)$
- $X \subseteq Act$
- E --σ--> F and NONE of the actions in X is possible in F

Note: it may $\exists$ F': E --σ--> F' and X is possible in F'

Note: if (σ, X) is a failure for E, then (σ, Y), $Y \subseteq X$, is a failure for E

Failures: comparing also what we cannot do after a finite sequence.

Let Fail(E) be the set of all failures of E, then

$$E \approx_{fl} F \quad \text{iff} \quad \text{Fail}(E) = \text{Fail}(F)$$

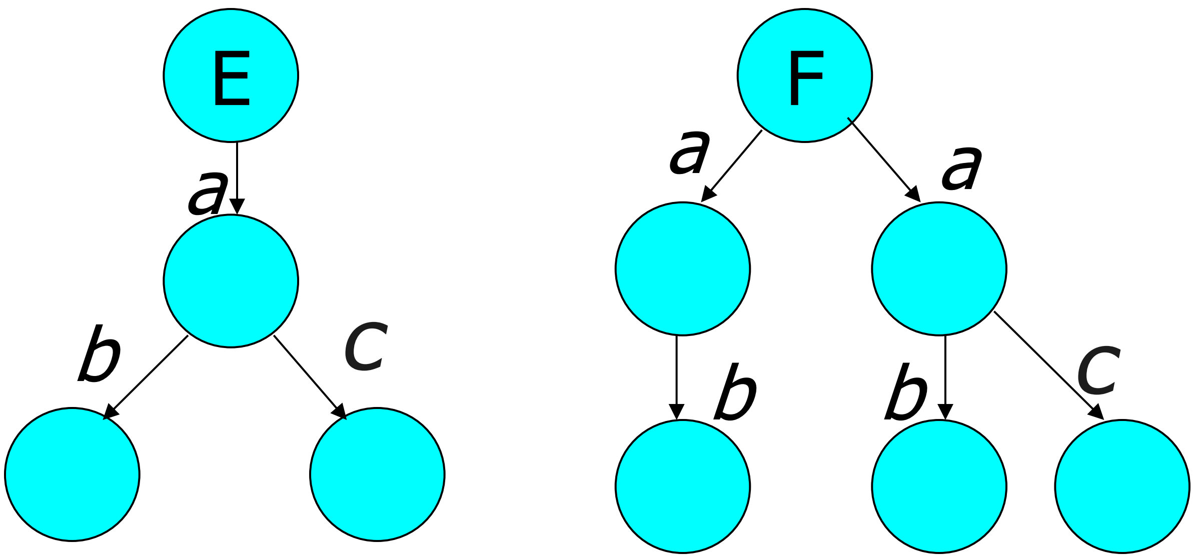
Property: $\approx_{fl}$ is more refined than $\approx_{tr}$, that is to say

$$E \approx_{fl} F \implies E \approx_{tr} F$$

proof: Fail(E) includes also the set (σ, 0) of the finite traces of E

$\approx_{fl}$ is strictly more refined than $\approx_{tr}$

To prove this it is enough to consider the following counter-example



T(E) = T(F), but FAIL(E) $\neq$ FAIL(F)

Simulation equivalence

Basic idea:

- Define a simulation relation over agents E *R* F

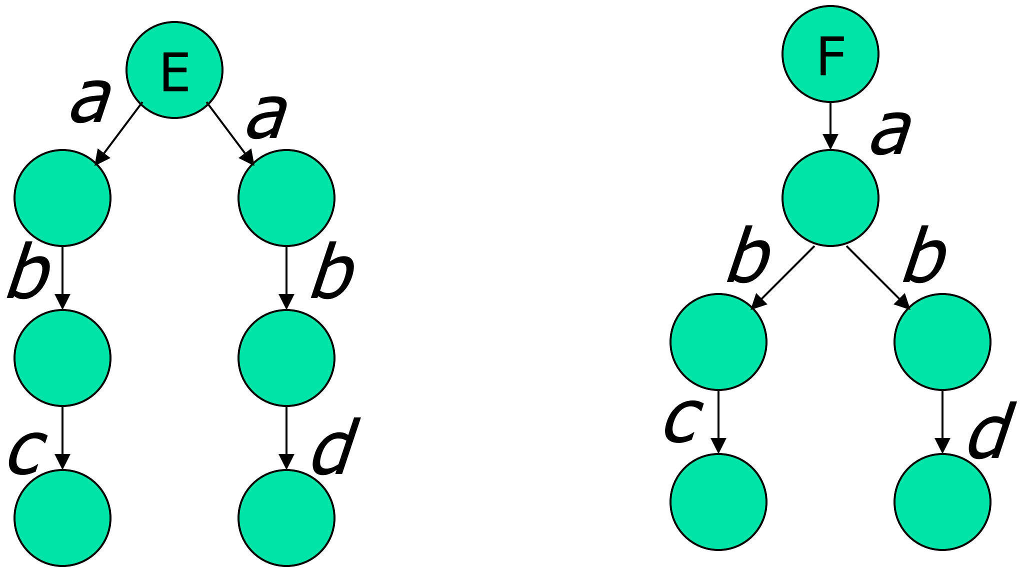
- Then E $\approx_{sim}$ F if there exists two simulation relations *R* and *Q* such that

  E *R* F and F *Q* E

Definition: R$\subseteq$S$\times$S is a simulation relation if

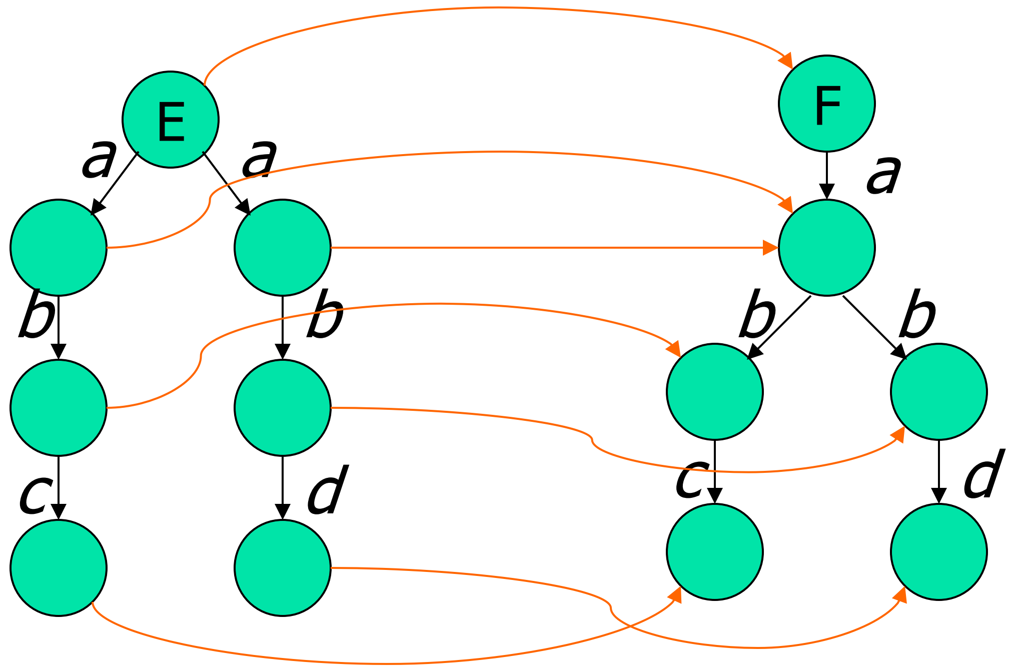
- E R F
- If E′ R F′ and E′—*a*→E″, then there exists F″, F′—*a*→F″, and E″ R F″
- and we say F simulates E.

Simulation equivalence



What distinguish E and F?

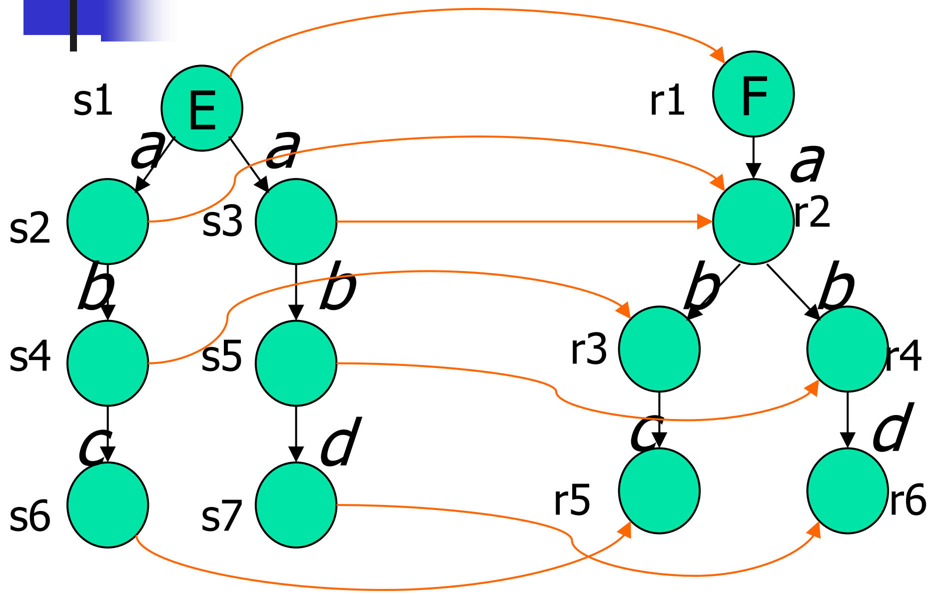
Simulation equivalence



F simulates E if F “can reply” to the moves of E

E and F are not $\approx_{sim}$, since F simulates E, (E *R* F), but it does not exists a Q: E simulates F

Simulation equivalence



E R F

If E′ R F′ and E′—*a*→E″,
then ∃ F″,
F′—*a*→F″, and E″ R F″.

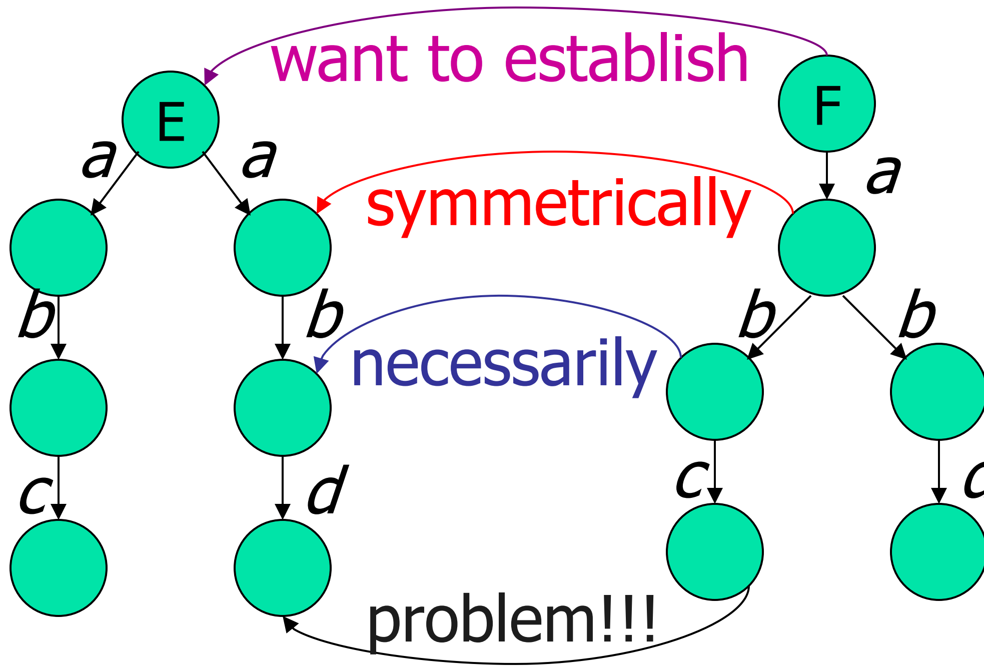
F Q E (E simulates F)

If F′ R E′ and F′—*a*→F″,
then ∃ E″,
E′—*a*→E″, and F″ R E″

R = {(s1,r1), (s2,r2), (s4,r3), (s6,r5), (s3,r2), (s5,r4), (s7,r6)}

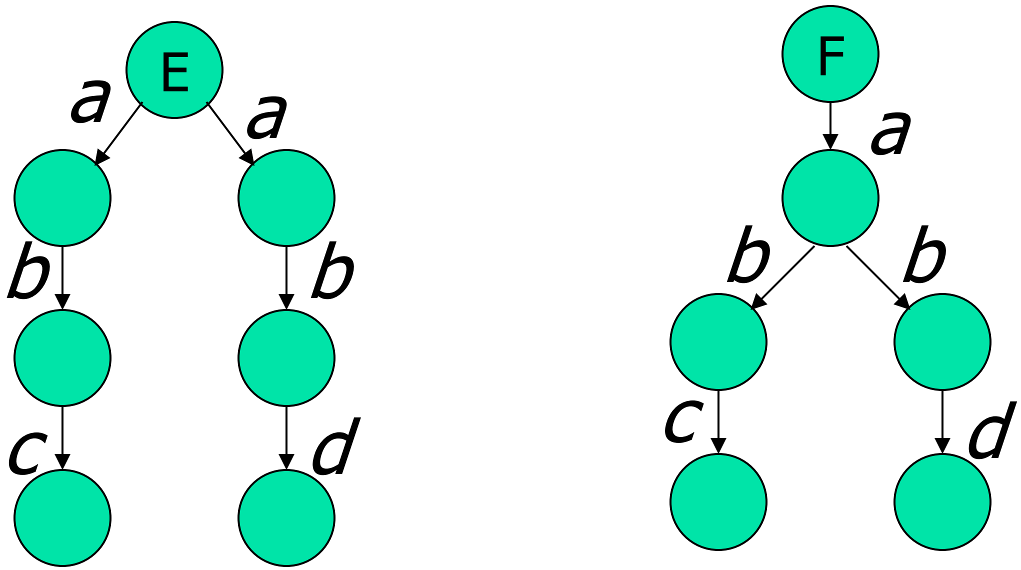
Q = {(r1,s1), (r2,s3………

Here, simulation works only in one direction. No equivalence!



- Relation over set of agents S. $R \subseteq S \times S$.
- E R F
- If E′ R F′ and $E' \xrightarrow{a} E''$, then there exists F″, $F' \xrightarrow{a} F''$, and E″ R F″.

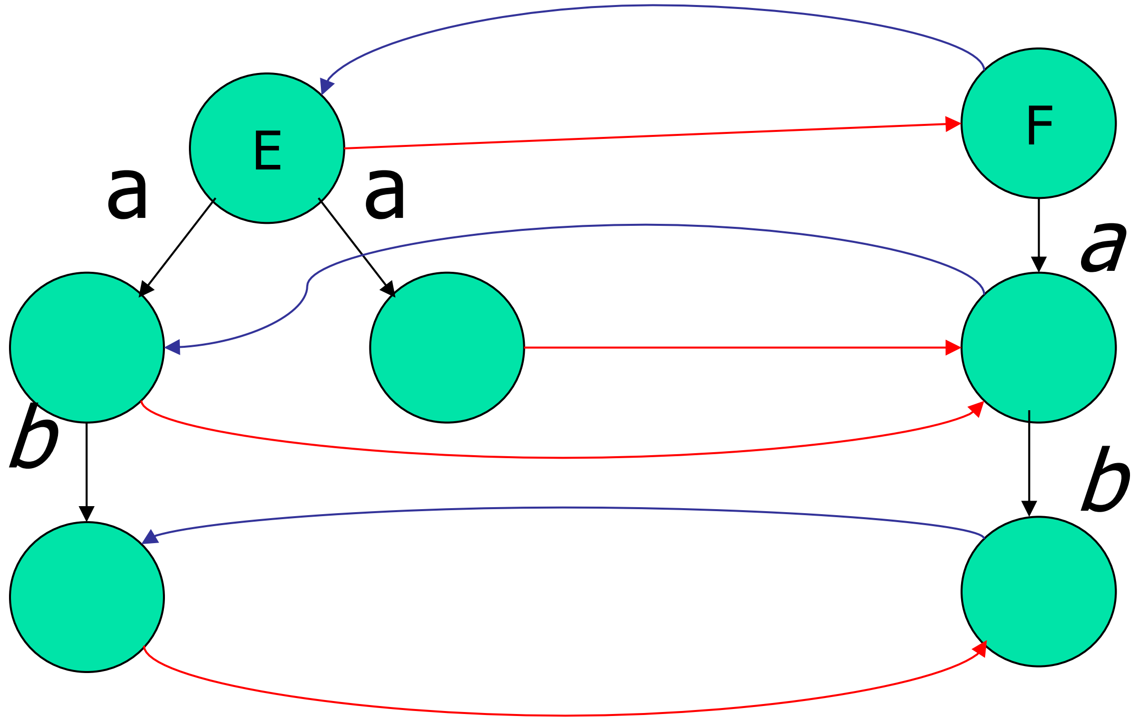
Simulation equivalent implies trace equivalent



$\approx_{sim}$ is strictly more refined than $\approx_{tr}$

(indeed E $\approx_{tr}$ F, but not E $\approx_{sim}$ F)

Simulation and failure are not comparable



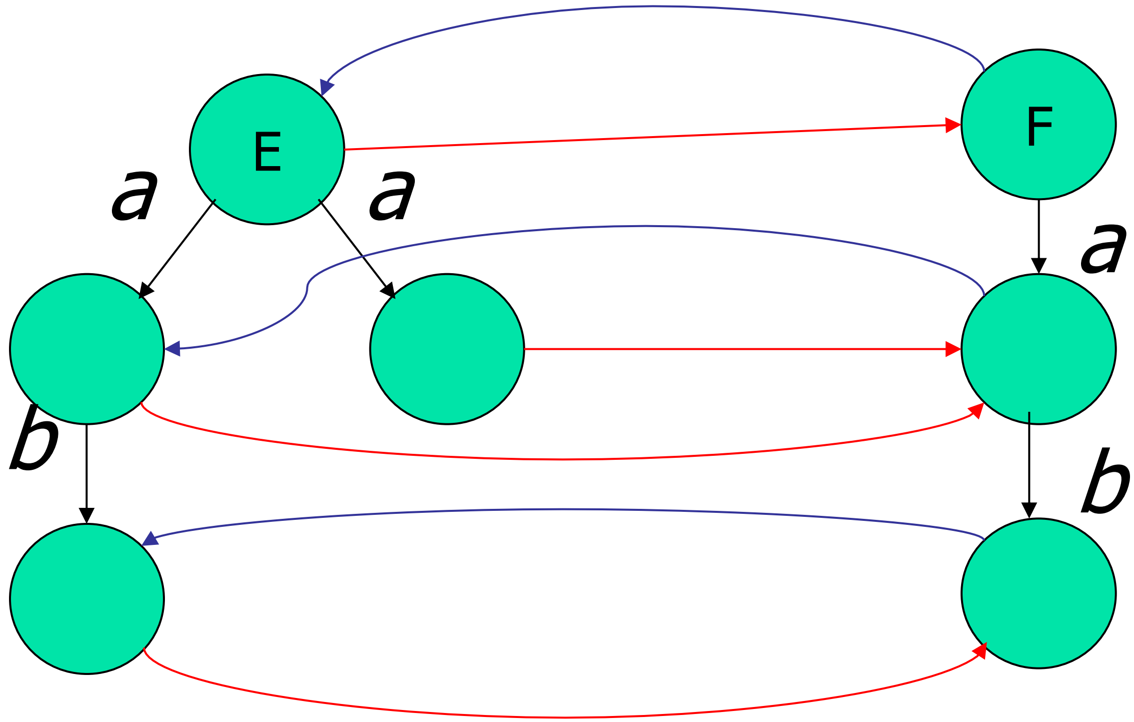
E $\approx_{sim}$ F, but E = *a.b+a* has a failure (*a*,{*b*}), while F has not

E and F of the previous slide are instead E $\approx_{fl}$ F, but not $\approx_{sim}$

Bisimulation between G_1 and G_2

- Let $N = N_1 \cup N_2$
- A relation R : N x N is a bisimulation if
  If (m,n) in R then
  1. If $m \xrightarrow{a} m'$ then $\exists n' : n \xrightarrow{a} n'$
     and (m',n') in R
  2. If $n \xrightarrow{a} n'$ then $\exists m' : m \xrightarrow{a} m'$
     and (m',n') in R.
- Other simulation relations are possible, I.e.,
  $m \overset{a}{\Longrightarrow} m'$ when $m \xrightarrow{\tau} \ldots \xrightarrow{a} \ldots \xrightarrow{\tau} m'$.

Bisimulation: same relation simulates in both directions



Not in this case: different simulation relations and there is no other simulation of E and F and of F and E.

Algorithm for bisimulation:

Input: the set of agents S, the set of actions Act

Create the initial partition P = {S}

Repeat until there is no change in P:

find if there are two (not necessarily different) elements T1 and T2 in P, and an action a$\in$Act such that the following holds: T1 can be split into two non empty and disjoint subsets S_1 and S_2, such that:

- $\forall$ agent E $\in S_1$, $\exists$ E' $\in$T2: E—a->E'
- Not($\exists$) E $\in S_2$, such that, for some agent E' $\in$T2 it holds that E—a->E'

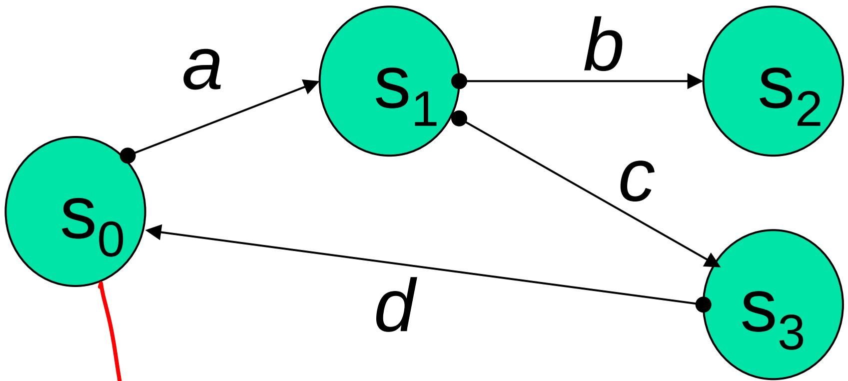
If there are such sets, replace T1 in P with S_1 and S_2

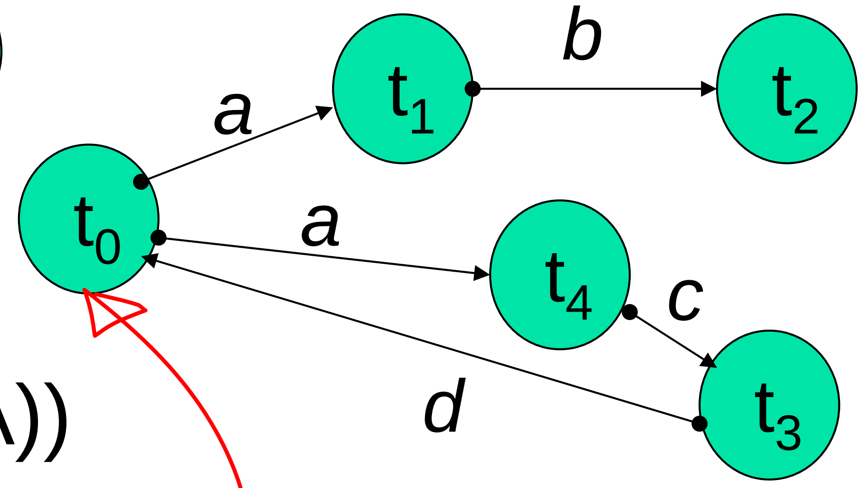
Output: a partition {T1, T2, … Tn } of the set of agents S: for any two agents E and E' in Ti, E $\approx_{bis}$ E'

Correctness of algorithm

- Invariant: if (m,n) in relation R (bisimulation relaton in our case) then m and n remain in the same partition element throughout the algorithm.
- Termination: can split only a finite number of times.

Example:

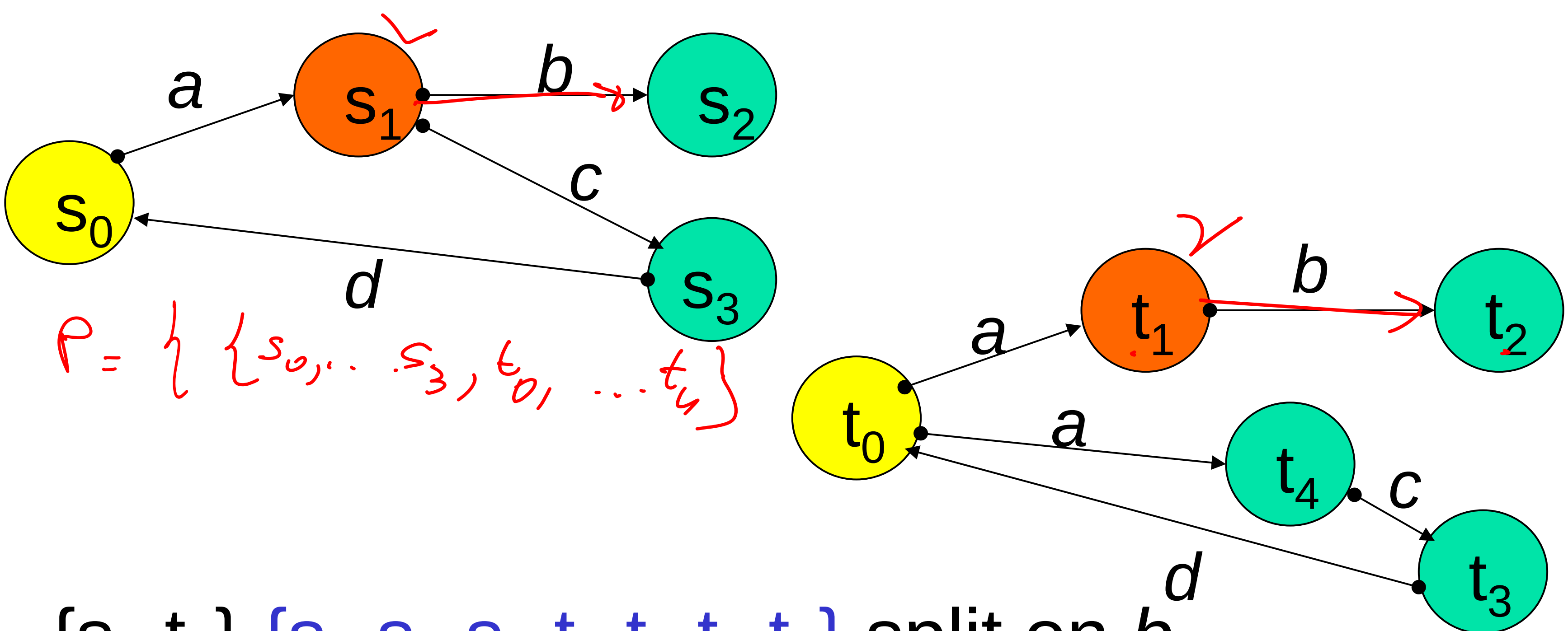




A=a.((b.nil)+(c.d.A))

B=(a.(b.nil))+(a.c.d.B)

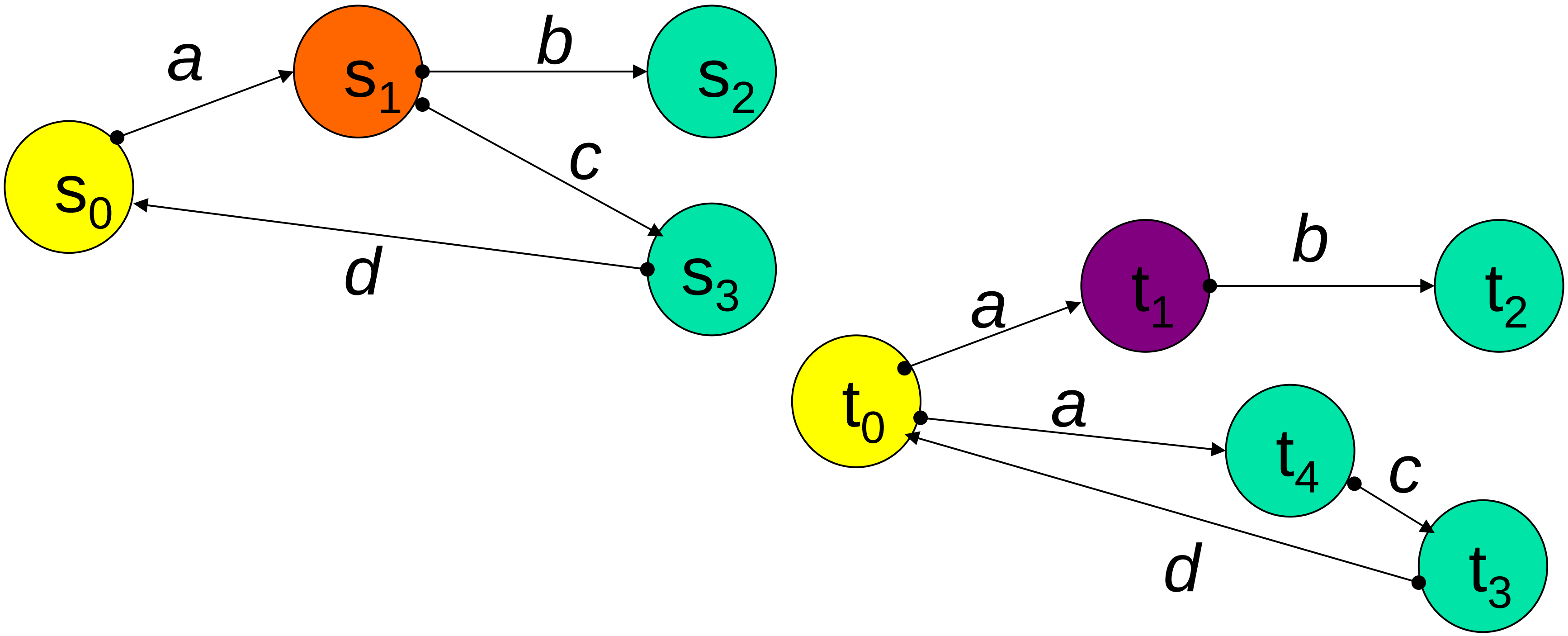
Example:



$\{s_0,t_0\},\{s_1,s_2,s_3,t_1,t_2,t_3,t_4\}$ split on *b*

$\{s_0,t_0\},\{s_1,t_1\},\{s_2,s_3,t_2,t_3,t_4\}$

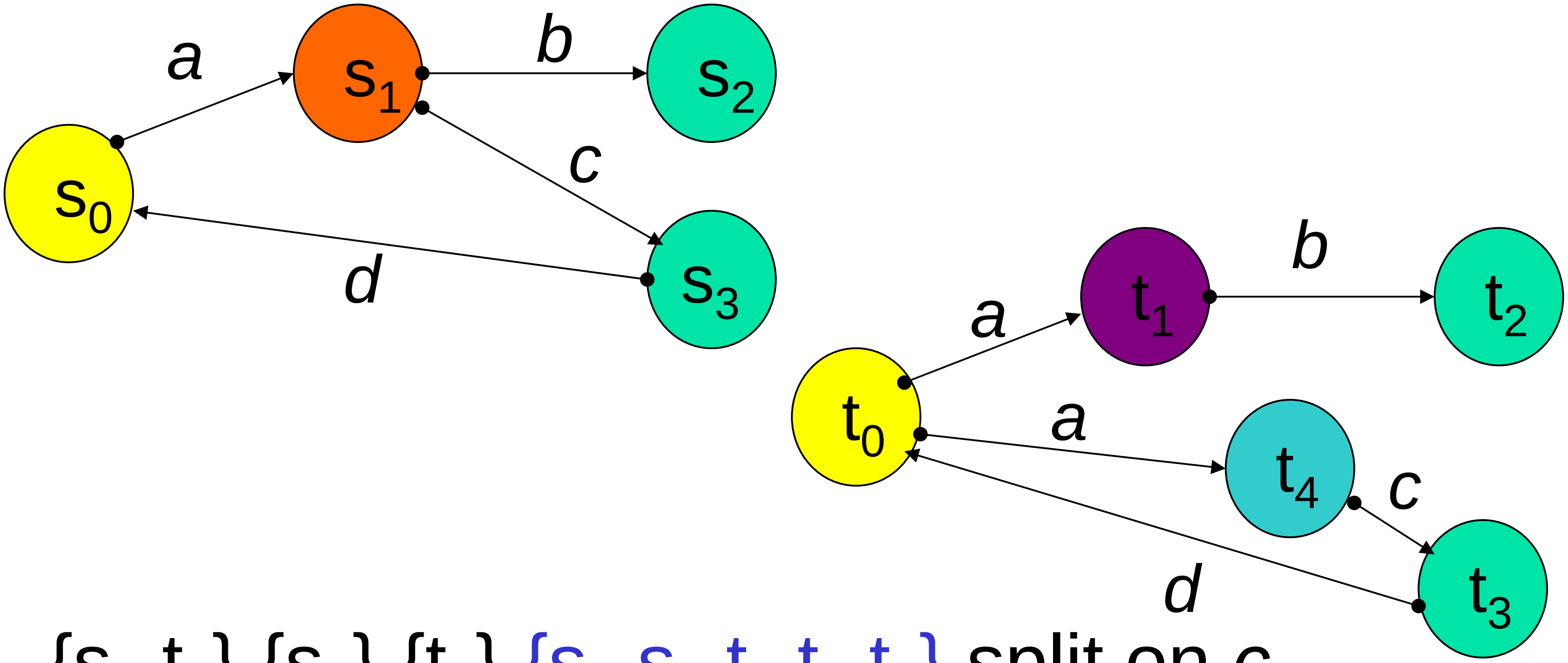
Example:



$\{s_0,t_0\},\{s_1,t_1\},\{s_2,s_3,t_2,t_3,t_4\}$ split on *c*

$\{s_0,t_0\},\{s_1\},\{t_1\},\{s_2,s_3,t_2,t_3,t_4\}$

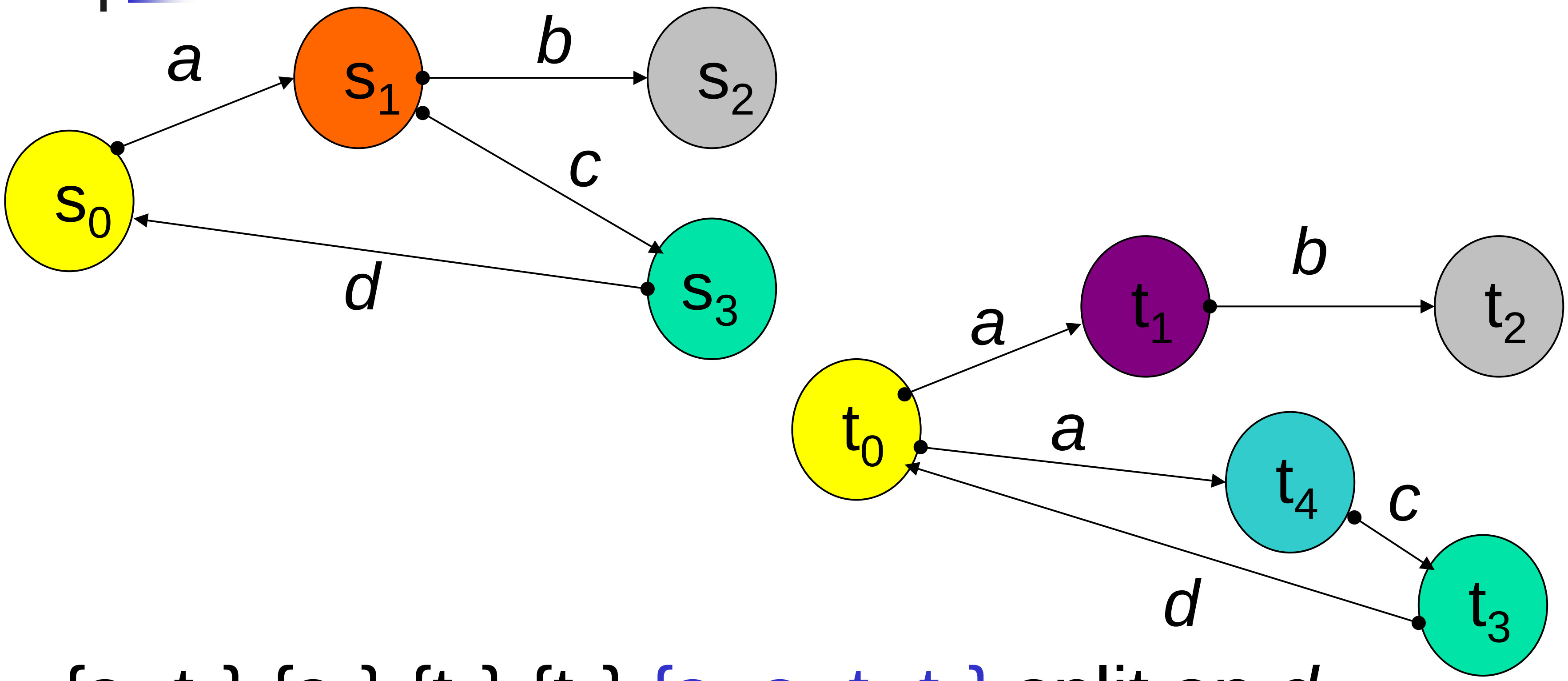
Example:



$\{s_0,t_0\},\{s_1\},\{t_1\},\{s_2,s_3,t_2,t_3,t_4\}$ split on *c*

$\{s_0,t_0\},\{s_1\},\{t_1\},\{t_4\},\{s_2,s_3,t_2,t_3\}$

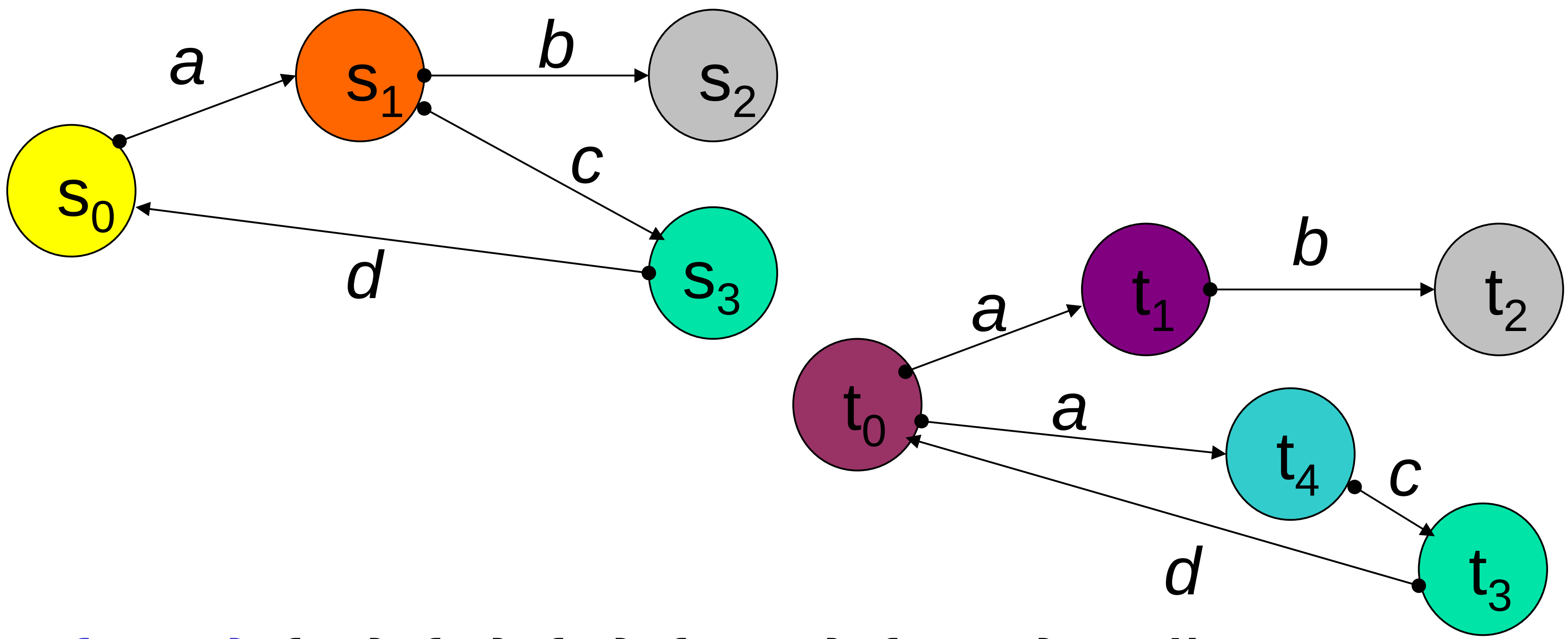
Example:



$\{s_0,t_0\},\{s_1\},\{t_1\},\{t_4\},\{s_2,s_3,t_2,t_3\}$ split on *d*

$\{s_0,t_0\},\{s_1\},\{t_1\},\{t_4\},\{s_3, t_3\},\{s_2,t_2\}$

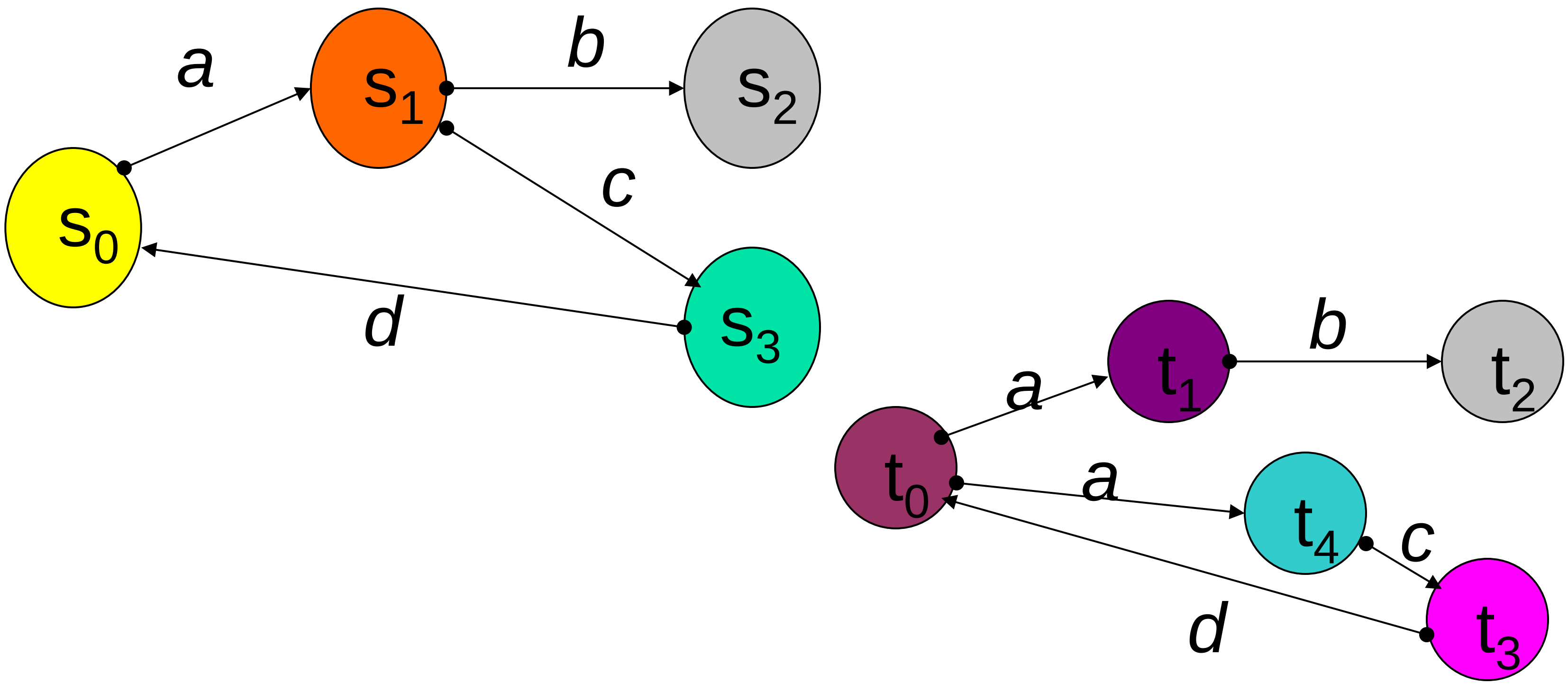
Example:



$\{s_0,t_0\},\{s_1\},\{t_1\},\{t_4\},\{s_2,t_2\},\{s_3,t_3\}$ split on *a*

$\{s_0\},\{t_0\},\{s_1\},\{t_1\},\{t_4\},\{s_3, t_3\},\{s_2,t_2\}$

Example:



$\{s_0\},\{t_0\},\{s_1\},\{t_1\},\{t_4\},\{s_2,s_3,t_2,t_3\}$ split on *d*

$\{s_0\},\{t_0\},\{s_1\},\{t_1\},\{t_4\},\{s_3\},\{t_3\},\{s_2,t_2\}$

State based bisimulation

When states are labelled with atomic propositions and actions are not distinguishable, we can define a state based bisimulation. How can we modify the algorithm?

Input: the set of agents S, the set of actions Act

Create the initial partition P = {S} $\longrightarrow$ $P = \{C_i\}_{i=1}^{|AP|}$

$\forall s, s' \in S$

$s, s' \in C_i \Longleftrightarrow \ell(s) = \ell(s')$

Repeat until there is no change in P:

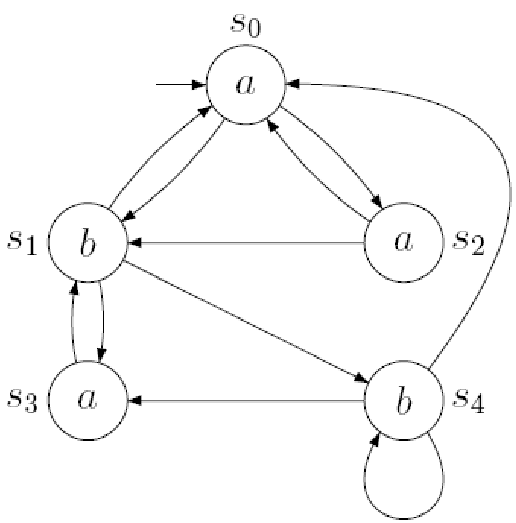
find if there are two (not necessarily different) elements T1 and T2 in P, and an action a $\in$Act such that the following holds: T1 can be split into two non empty and disjoint subsets S_1 and S_2, such that:

- $\forall$ agent E $\in S_1$, $\exists$ E' $\in$T2: E—~~a~~->E'
- Not($\exists$) E $\in S_2$, such that, for some agent E' $\in$T2 it holds that E—a->E'

If there are such sets, replace T1 in P with S_1 and S_2

Output: a partition {T1, T2, … Tn } of the set of agents S: for any two agents E and E' in Ti, E $\approx_{bis}$ E'

Esempio bisimulazione, by Katoen



(1) Initial partition: $\Pi = \{\underbrace{\{s_0, s_2, s_3\}}_{B_1}, \underbrace{\{s_1, s_4\}}_{B_2}\}$

(2) Successor blocks:

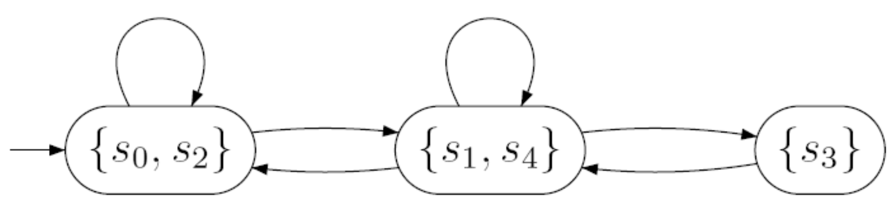
s	s_0	s_2	s_3	s_1	s_4
$Succ(s)$	$\{B_1, B_2\}$	$\{B_1, B_2\}$	$\{B_2\}$	$\{B_1, B_2\}$	$\{B_1, B_2\}$

(3) Partition refinement: $\Pi = \{\underbrace{\{s_0, s_2\}}_{B_3}, \underbrace{\{s_3\}}_{B_4}, \underbrace{\{s_1, s_4\}}_{B_2}\}$

(4) Successor blocks:

s	s_0	s_2	s_3	s_1	s_4
$Succ(s)$	$\{B_2, B_3\}$	$\{B_2, B_3\}$	$\{B_2\}$	$\{B_2, B_3, B_4\}$	$\{B_2, B_3, B_4\}$

(5) Partition stable $\Longrightarrow$ $s_0 \sim s_2$ and $s_1 \sim s_4$



Complexity

The best known complexity of an algorithm for partition refinement is $O(m \, log n)$, where $m=|E|$ is the number of transitions and $n=|V|$ is the number of states (in practical cases, m is significantly bigger than n (Paige and Tarjan algorithm).

(per gli studenti di simulazione: note that this is the same complexity as computing lumpability in Markov chain)

Equations under $\approx_{bis}$ and congruence

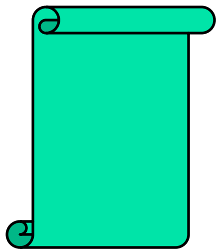
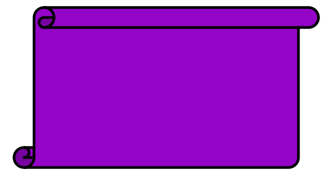
- commutative : $A+B \approx_{bis} B+A$ and $A||B \approx_{bis} B||A$
- associative: $A+(B+C) \approx_{bis} (A+B)+C$ and $A||(B||C) \approx_{bis} (A||B)||C$
- idempotence of non deterministic choice: $A+A \approx_{bis} A$

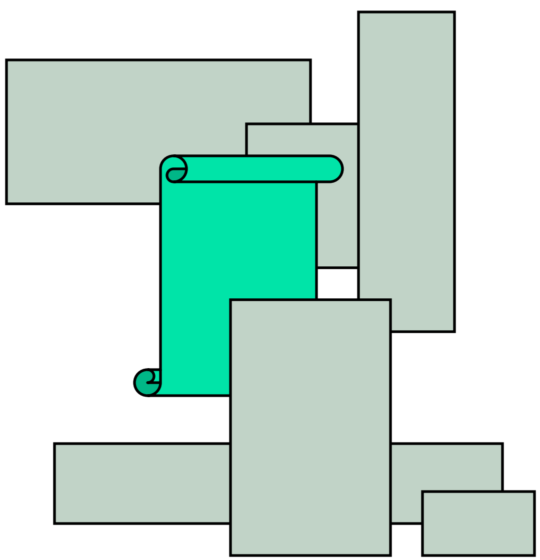
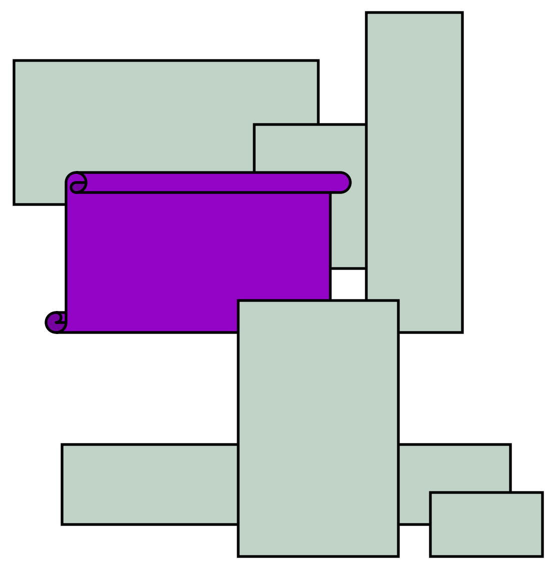
Def.: a congruence is an equivalence relation that also satisfy replacement under any context, that is to say: if $\approx_{cong}$ is a congruence, and $B \approx_{cong} C$, then

$$A \approx_{cong} A(B/C)$$

Note: it has been proved (Milner's book) that $\approx_{bis}$ is a congruence

Equivalence relations and congruence

Se  $\approx_c$ 

Allora  $\approx_c$ 

Consequences of $\approx_{bis}$ being a congruence

It is possible to compute the derivation graph in an incremental manner, for example if A = B||B, and B $\approx_{bis}$ C, and C is easier to analyze than B, we can substitute C for B in A, and still have a process algebra term that is bisimilar to the original one.

Another way to take advantage of the partitioning algorithm for the computation of bisimulation is to observe that if we substitute each element of the partition with a single node, and we obtain a derivation graph in which each node represents a set of states, the states with the same "future evolution".

We will see more of this on the symbolic reachability graph construction of Well-formed nets

$\approx_{bis}$ and action refinement

Let

$$E = a||b.c \quad \text{and} \quad E' = a.b.c + b.a.c + b.c.a$$

Note that E $\approx_{bis}$ E′ (E′ are all possible interleaving of E).

We can use process algebra to show that interleaving semantics is not closed under action refinement, indeed: take F = a||d and F′= a.d + d.a (F $\approx_{bis}$ E′) and refine d into b.c, then

$$\begin{tikzcd}
F = a||d \arrow[r, "\text{refinement}"'] \arrow[dd, "\text{interleaving}"] & a||b.c \arrow[d, "\text{interleaving}"] \\
 & a.b.c + b.a.c+b.c.a \arrow[d, dotted, dash, "\text{Not bisimilar (not even trace!)}"] \\
F' = a.d + d.a \arrow[r, "\text{refinement}"'] & a.b.c + b.c.a
\end{tikzcd}$$

Action refinement does not maintain any of the four defined equivalences!

Example of $\approx_{wbis}$

Take the two place buffer term and the two buffer term obtained from the parallel composition of two single buffer terms with relabelling and restriction:

Are they equivalent? Trace? Failure? Sim? Bis? wbis?

A relation $R : N_1 \times N_2$ is a weak bisimulation ($\approx_{wbis}$) , if, given (m,n) in R, then

- If m=*a*=>m' then ∃n':n=*a*=>n'
  and (m',n') in R
- If n=*a*=>n' then ∃m':m=*a*=>m'
  and (m',n') in R.

where m=*a*=>m' when m—τ→…—*a*→… —τ→m' (it is read "m goes in m' with the extended action a)

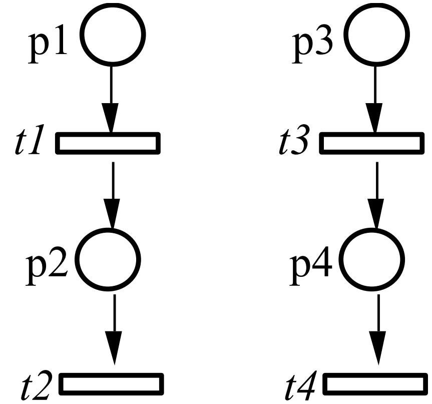
Exercise

Compare the reachability graphs of the net below with the following initial marking:

Net A: M0(p1)= 1 and M0(p3)= 1

Net B: M0(p1)= 2

When t1 and t3 have the same label "a" and t2 and t4 have the same label "b"

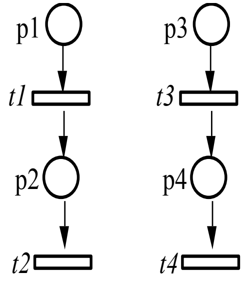


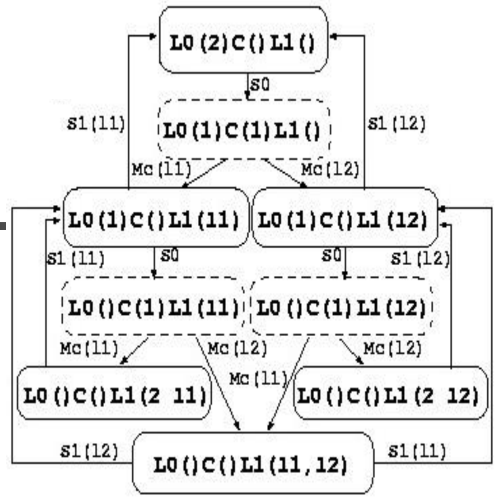
Exercise

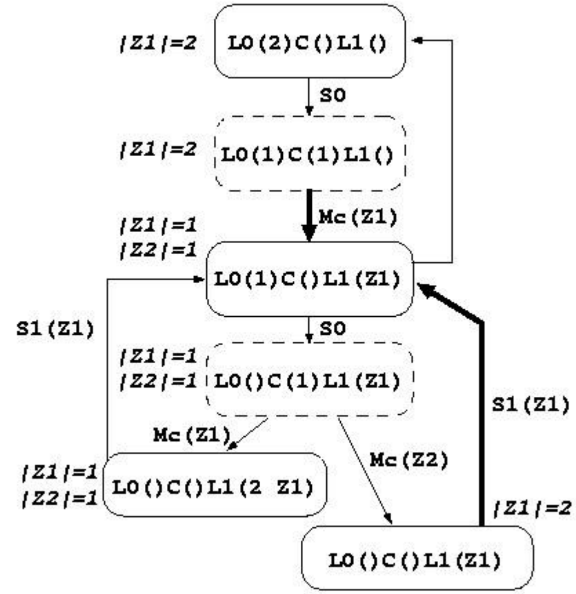
Compute the bisimulation relation over RG(A), and build the $RG(A)_{|bsim}$ in which each element of the partition is considered as a single state.

Compute the bisimulation relation over RG(A) ∪ RG(B), and check if the two initial markings belong to the same equivalence class.

Compare $RG_{|bsim}$ and RG(B) in terms of number of states, arcs and structure.




L0(2)C()L1()
S0
S1(l1)
S1(l2)
L0(1)C(1)L1()
Mc(l1)
Mc(l2)
L0(1)C()L1(l1)
L0(1)C()L1(l2)
S1(l1)
S0
S0
S1(l2)
L0()C(1)L1(l1)
L0()C(1)L1(l2)
Mc(l1)
Mc(l2)
Mc(l2)
Mc(l1)
L0()C()L1(2 l1)
L0()C()L1(2 l2)
S1(l2)
L0()C()L1(l1,l2)
S1(l1)


|Z1|=2
L0(2)C()L1()
S0
|Z1|=2
L0(1)C(1)L1()
Mc(Z1)
|Z1|=1
|Z2|=1
L0(1)C()L1(Z1)
S1(Z1)
S0
|Z1|=1
|Z2|=1
L0()C(1)L1(Z1)
S1(Z1)
Mc(Z1)
Mc(Z2)
|Z1|=1
|Z2|=1
L0()C()L1(2 Z1)
|Z1|=2
L0()C()L1(Z1)

Our course - recall

Concentrate on distributed systems (as inherently protocols are)

Learn several formalisms to model system and properties (automata, process algebras, Petri Nets, temporal logic, timed automata).

Learn advantages and limitations, in order to choose the right methods and tools.

Learn how to combine existing formalisms and existing "solution" methods.