Analysis and Control of Petri Nets

Alessandro Giua

DIEE, Univ. of Cagliari

28 May 2018 - DES School, Sorrento Coast, Italy

Outline

Outline

Labeled net

A labeled Petri net (or PN generator) is a 4-tuple $G = (N, \ell, M_0, F)$:

- $N = (P, T, Pre, Post)$ is a Petri net structure;
- $\ell : T \to \Sigma$ is a labeling function that assigns a label from alphabet Σ to transitions;
- M_0 is an initial marking;
- F is a finite set of final markings.

Language generated (closed language, P-language):

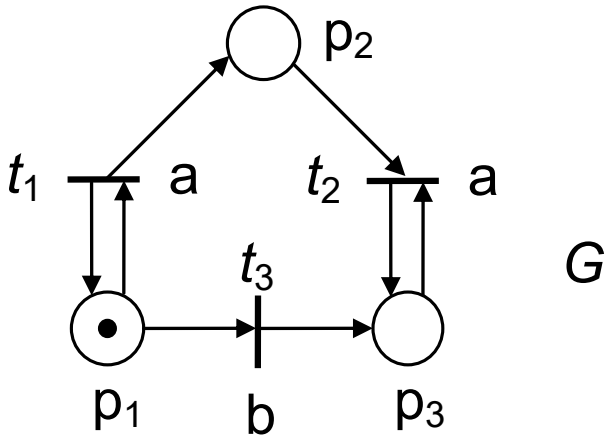
$$L(G) = \{w \in \Sigma^* \mid (\exists \sigma \in T^*)\ M_0[\sigma > M,\ w = \ell(\sigma)\}$$

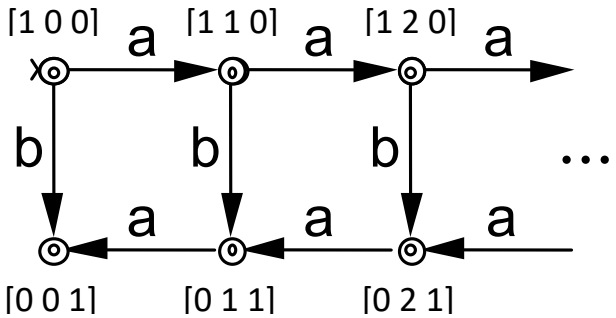
Language accepted (marked language, L-language):

$$L_m(G) = \{w \in \Sigma^* \mid (\exists \sigma \in T^*)\ M_0[\sigma > M,\ M \in F,\ w = \ell(\sigma)\}$$

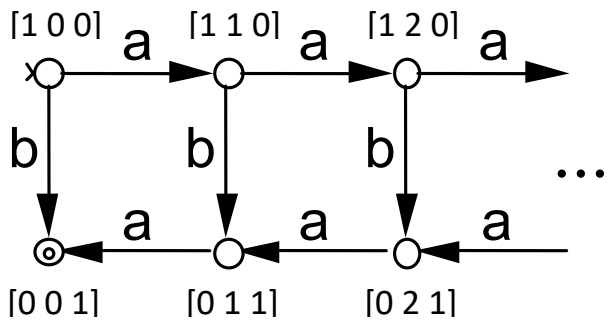
Example

Assume the set of final markings is: $F = \{\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T\}$.





$L(G)$



$L_m(G)$

Language power

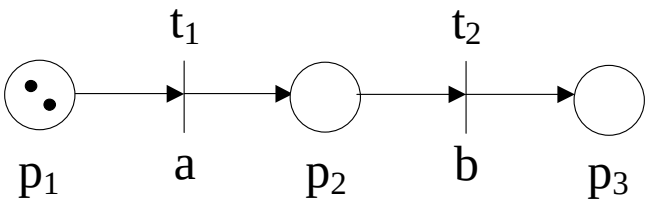
It is important to characterize the language power of a formal model:

- To understand the expressive power of the model
- To understand the tractability of the given model
- To compare different formalisms and convert between them
- To built a model independent theory
- To apply all analysis tools that pertain to a language formalism (monoids, partial order, etc.)

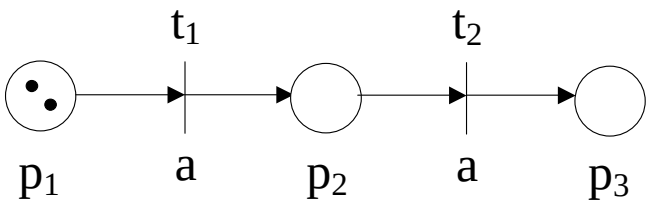
Labeling function

Three types of labelings are possible:

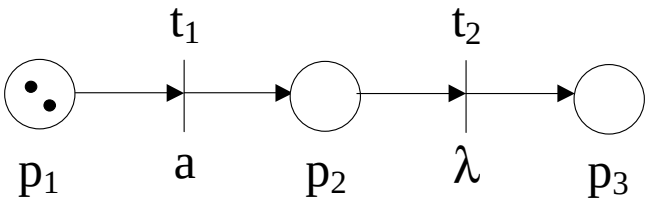
- free labeling: ℓ is a 1 to 1 mapping;
- λ-free labeling: two transitions may share the same label;
- arbitrary labeling: $\ell : T \rightarrow \Sigma \cup \{\lambda\}$, where λ is the empty string.



free



λ-free

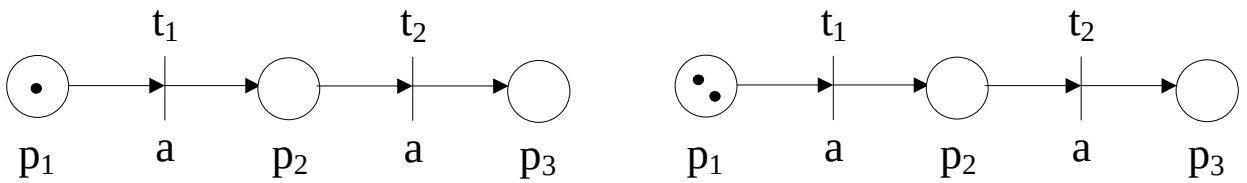


arbitrary

Deterministic labeling

A fourth type of labeling is also possible:

- deterministic labeling: the labeling is λ-free, but if t and t' (with $t \neq t'$) have the same label then for any reachable marking M: $M[t> \implies \neg M[t'>$



deterministic

λ-free (nondeterministic)

Note: the deterministic labeling is not structural but behavioral.
The property of being deterministic can be verified constructing the coverability graph: the graph must not contain a node with two (or more) output arcs sharing the same label.

Classes of PN languages

We can finally define the following classes of languages:

	free labeling	deterministic labeling	λ-free labeling	arbitrary labeling
$L(G)$	$\mathcal{P}_f$	$\mathcal{P}_d$	$\mathcal{P}$	$\mathcal{P}_\lambda$
$L_m(G)$	$\mathcal{L}_f$	$\mathcal{L}_d$	$\mathcal{L}$	$\mathcal{L}_\lambda$

It holds (here $\mathcal{X} \to \mathcal{Y}$ denotes $\mathcal{X} \subseteq \mathcal{Y}$, while $\sim$ denotes incomparability) :

$$\begin{array}{ccccccc} \mathcal{P}_f & \to & \mathcal{P}_d & \to & \mathcal{P} & \to & \mathcal{P}_\lambda \\ \wr & & \wr & & \downarrow & & \downarrow \\ \mathcal{L}_f & \to & \mathcal{L}_d & \to & \mathcal{L} & \to & \mathcal{L}_\lambda \end{array}$$

Which language should we chose?

One would like to consider the largest class (arbitrary languages). However it is important that the considered class of models allows one to answer questions such as the language containment problem:

Given two languages L and L', is $L \subseteq L'$?

Which language should we chose?

Given two languages L and L', is $L \subseteq L'$?

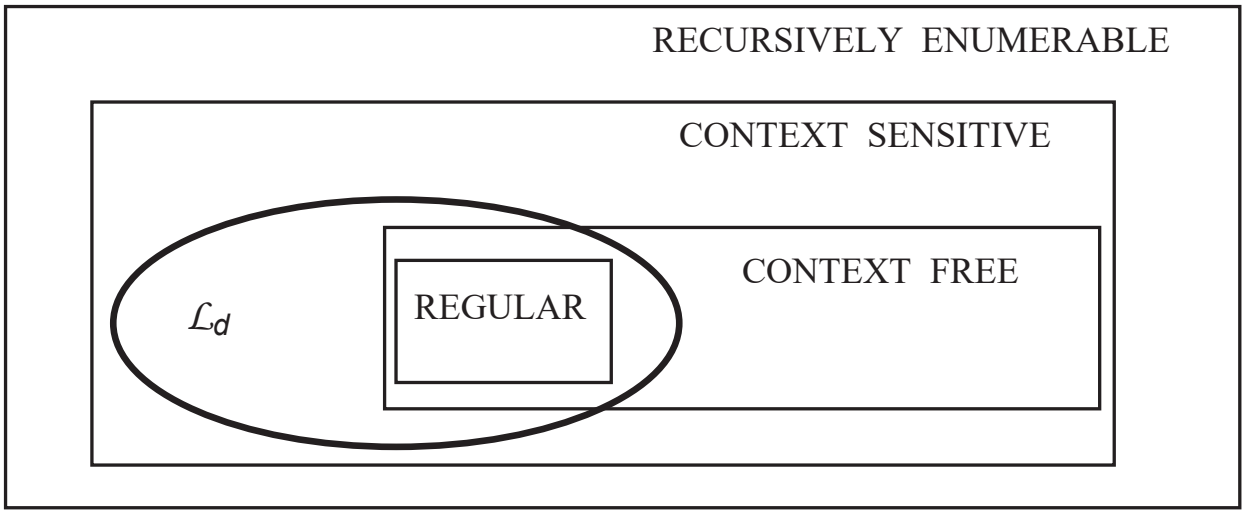
- Arbitrary languages and λ-free languages are not good because language containment is **not decidable** if $L' \in \mathcal{P}$ or $L' \in \mathcal{L}$.
- Deterministic languages are good: language containment is **decidable** if $L \in \mathcal{L}$ and $L' \in \mathcal{L}_d \cup \mathcal{P}_d$
- Free languages are not good: although easier that deterministic languages **cannot model all regular languages**.

Ex: The regular language $L = aba^*$ does not belong to $\mathcal{L}$. If fact, assume there is a single transition t with label a. Since aa does not belong to L, the firing of t is not repetitive. However, t can fire infinitely often after ab, clearly a contradiction.

Which language should we chose?

Conclusion: we are left with only two meaningful classes of PN languages: $\mathcal{P}_d$ (closed languages) and $\mathcal{L}_d$ (marked languages).

This is how $\mathcal{L}_d$ is related to other classes of formal languages.



Additional reading

- An excellent book, even if old, is:

  J.L. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice Hall, 1981.

  Covers reachability and coverability graph, and Petri net languages. It also presents a very interesting discussion of decidability properties for Petri nets even if the marking reachability problem was still open at the time it was written.

- Petri net languages are also discussed in:

  Giua A., "Supervisory control of Petri nets with language specifications," *Control of Discrete-Event Systems. Automata and Petri Net Perspectives*, C. Seatzu, M. Silva, J.H. van Schuppen (Eds), in Lecture Notes in Control and Information Science, Springer, Vol. 433, pp.235-256, 2012.

Outline

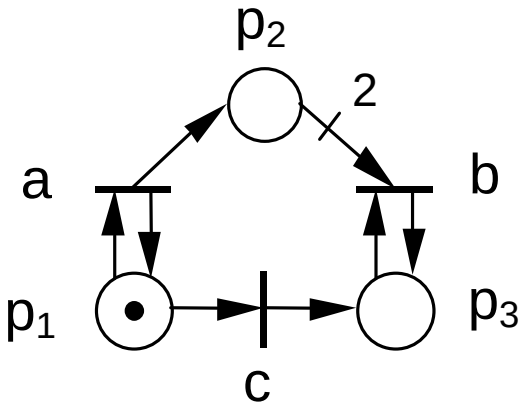
Are PN a good model for supervisory control?

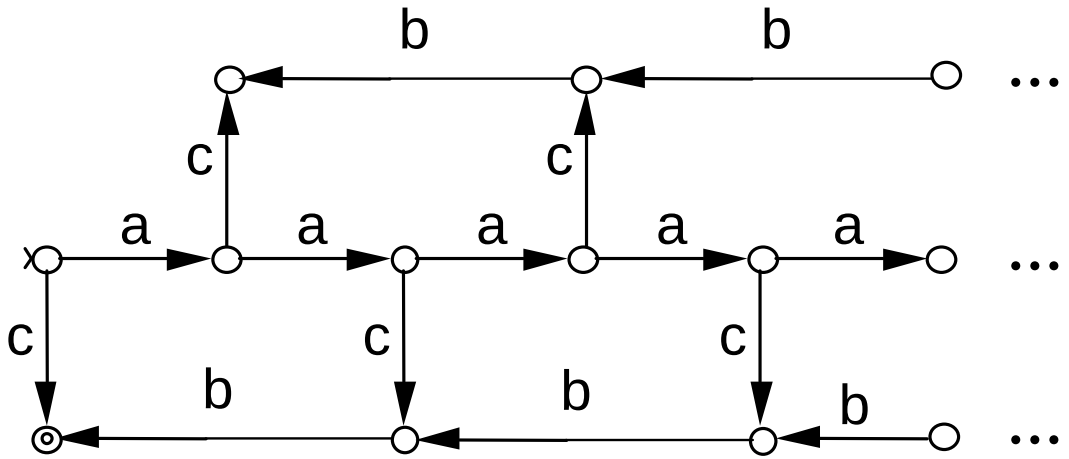
We need to answer these questions:

- Are properties of interest (controllability, closure) **decidable**? Answer: Yes if we consider deterministic languages.
- Is a net does not have a property, can it be **modified**? (trimming blocking nets, computing supremal controllable sublanguages?)
- Are there (efficient) algorithms to do this?
- Can a supervisor be always represent as a net?

Trimming a blocking Petri net

If $F = \left\{ \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \right\}$ this net is blocking. Can it be trimmed?

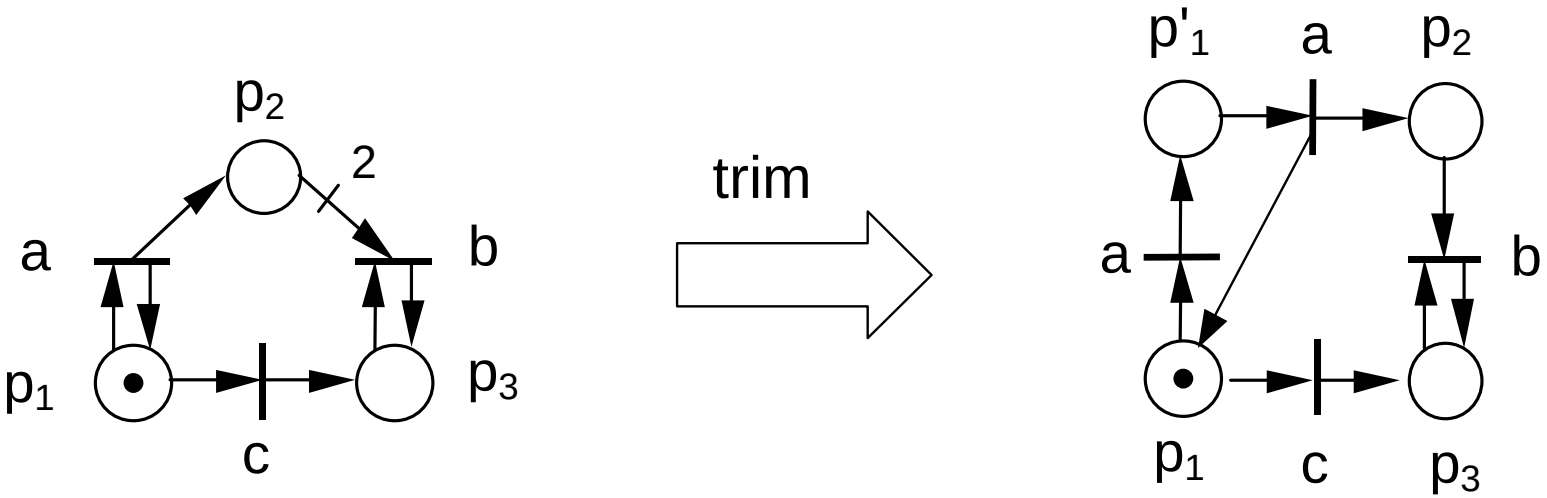




Trimming a blocking Petri net

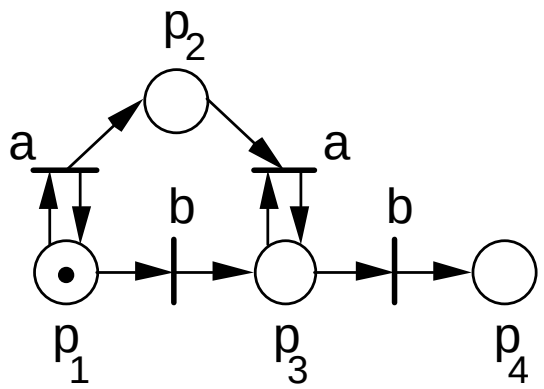
It can be trimmed but only modifying the structure!

To make sure c fires only after a has fired an even number of times use this net.



Trimming a blocking Petri net

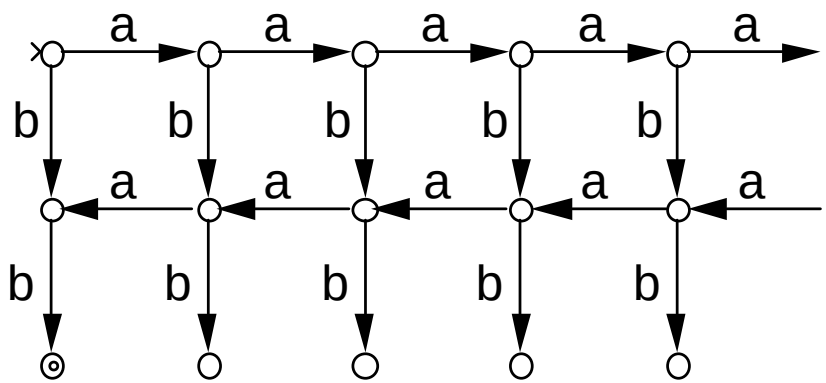
This net with $F = \left\{ \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T \right\}$ cannot be trimmed!



$$L_m(G) = \{a^m b a^m b \mid m \geq 0\}$$

$$L(G) = \{\overline{a^m b a^n b} \mid m \geq n \geq 0\}$$

However



$$\overline{L_m(G)} = \{\overline{a^m b a^m b} \mid m \geq 0\}$$

is not a PN language (can be proved using the pumping lemma for PN's).

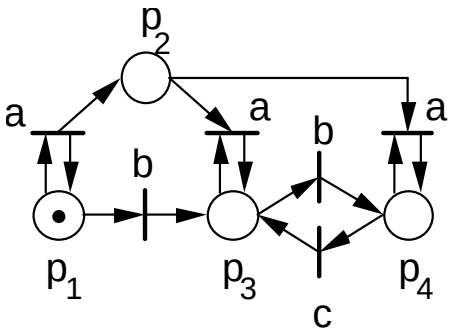
Closure under the supcon $\uparrow$ operator

The classes $\mathcal{P}_d$ and $\mathcal{L}_d$ are not closed under the $\uparrow$ operator.

Thus the supremal controllable sublanguage may not be a PN language even if the system language $L(G)$ and the specification language L are PN languages.

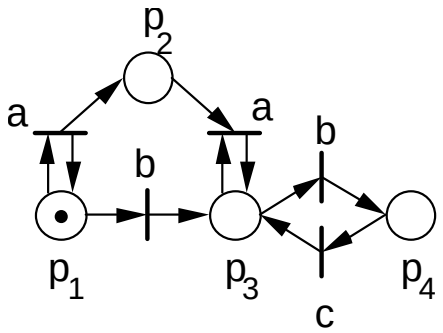
Closure under the supcon ↑ operator

Example: Let $\Sigma_u = \{a\}$. $L(E)$ and $L_m(E)$ are not controllable with respect to $L(G)$ (event a is uncontrollable).



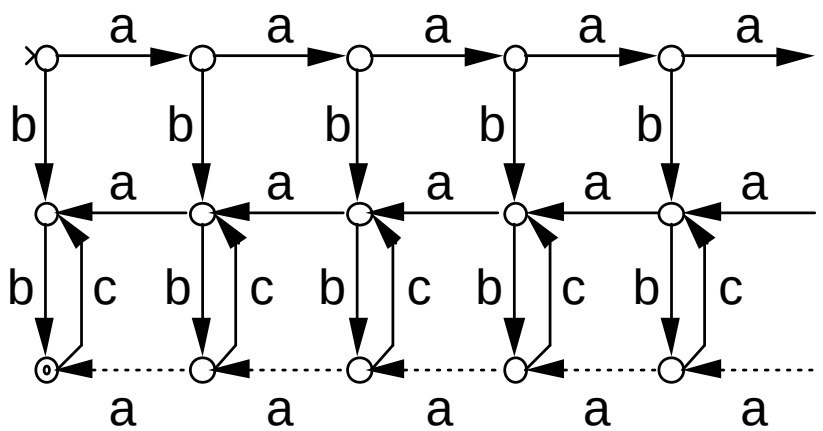
G

F = { (0 0 0 1) }^{T}



E

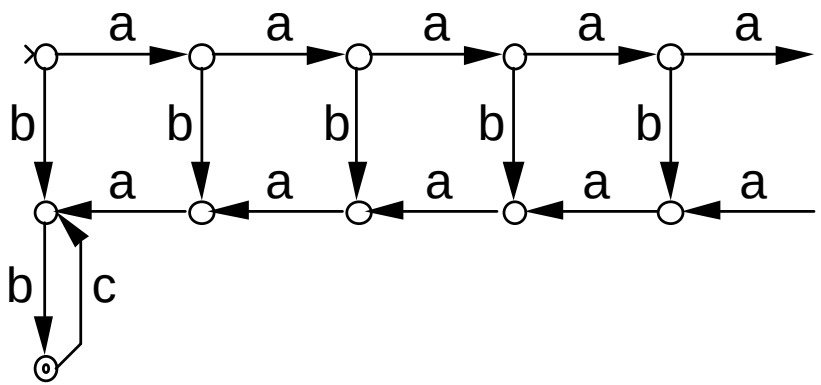
F = { (0 0 0 1) }^{T}



This automaton accepts $L_m(G)$ and (removing the dashed arcs) $L_m(E)$.

Closure under the supcon $\uparrow$ operator

Trimming the infinite state automaton we obtain the generator of the language $L_m(E)^{\uparrow}$



$L(E)^{\uparrow} \notin \mathcal{P}_d$ and $L_m(E)^{\uparrow} \notin \mathcal{L}_d$.

Additional reading

These results were originally presented in:

- A. Giua, F. DiCesare, "Blocking and Controllability of Petri Nets in Supervisory Control," *IEEE Trans. on Automatic Control*, Vol. 39, No. 4, pp. 818-823, April 1994.

- A. Giua, F. DiCesare, "Decidability and Closure Properties of Weak Petri Net Languages in Supervisory Control," *IEEE Trans. on Automatic Control*, Vol. 40, No. 5, pp. 906-910, May 1995.

Outline

Motivation

The procedure presented by Ramadge and Wonham for the synthesis of supervisors has three main features:

- the plant to be controlled is described as a language generator
- the specification to impose is a legal language (language specification)
- the design procedure is based on language operators.

As such, Supervisory Control Theory is model independent.

Wonham and Ramadge used automata, but any other language generator can be used, including Petri nets. Here we consider bounded Petri net models whose modeling power is equivalent to automata.

Supervisory control problem

Consider a plant G on alphabet Σ where events in $\Sigma_u \subseteq \Sigma$ are **not controllable**. Note that in general G is composed by concurrent composition of different modules, i.e.,

$$G = G_1 \parallel \cdots \parallel G_m$$

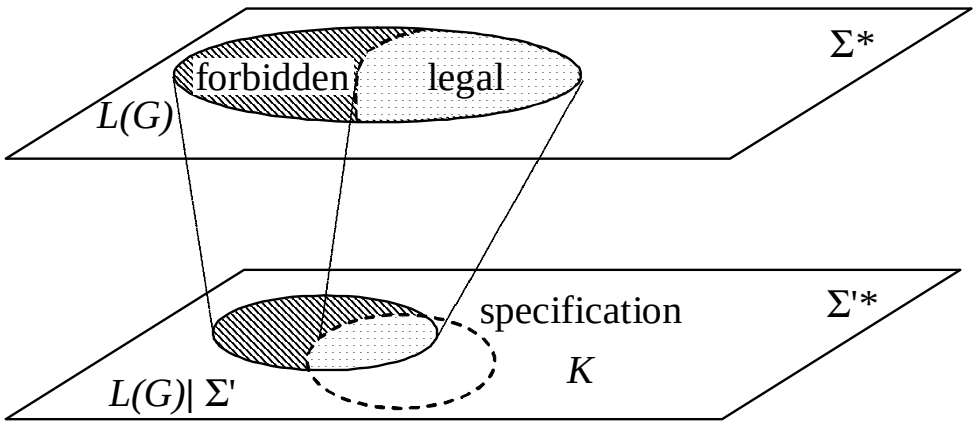
Consider a language specification H represented by an automaton H on alphabet $\Sigma' \subset \Sigma$. Note that in general H is composed by concurrent composition of different modules representing subspecifications, i.e.,

$$H = H_1 \parallel \cdots \parallel H_k$$

Supervisory control problem

The supervisor S must constrain the closed-loop language $L(S/G)$ to satisfy

$$L_{S/G}\,|_{\Sigma'} \subseteq K$$



Monolithic supervisory design

The supervisor S can be designed using the following procedure.

- Construct by concurrent composition an overall model of the plant plus specification

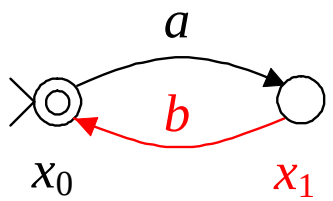
$$E = G \parallel H$$

  The language $L(E)$ contains all words in $L(G)$ that satisfy the specifications.
- Refine the generator E to eliminate blocking and uncontrollable states
- The refined generator E' represents at the same time the maximally permissive supervisor and the closed-loop system

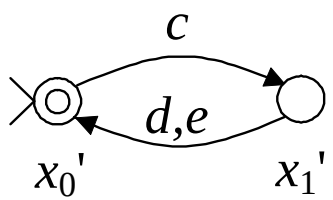
Example using automata

Given:

- G_1 with alphabet $\Sigma_1 = \{a, b\}$
- G_2 with alphabet $\Sigma_2 = \{c, d, e\}$



G_1

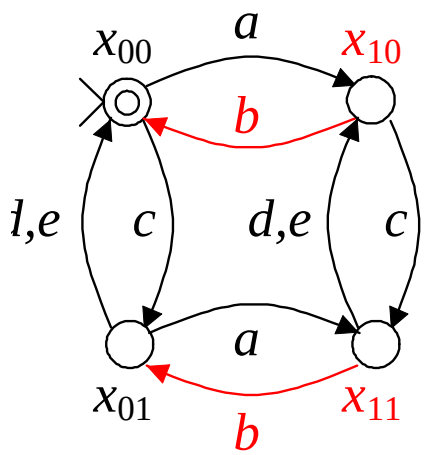


G_2

Plant on

$$\Sigma = \Sigma_1 \cup \Sigma_2$$

$$\Sigma_u = \{b\}$$

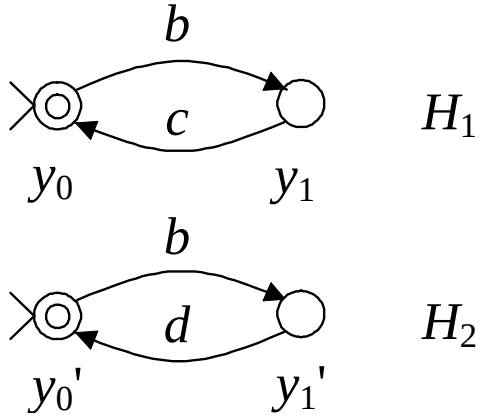


$G = G_1 || G_2$

Example using automata

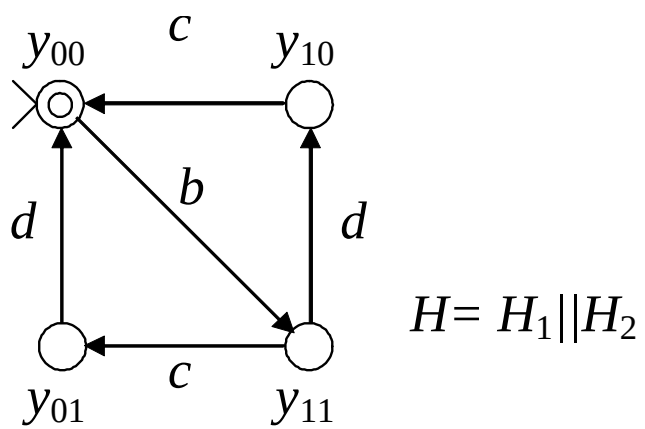
Specifications:

- H_1 with alphabet $\Sigma'_1 = \{b, c\}$
- H_2 with alphabet $\Sigma'_2 = \{b, d\}$



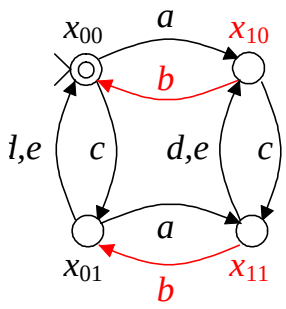
Overall specification on

$$\Sigma' = \Sigma'_1 \cup \Sigma'_2 = \{b, c, d\}$$

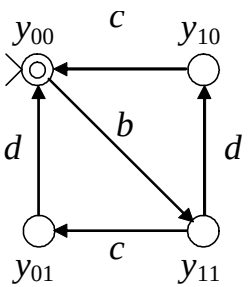


Example using automata

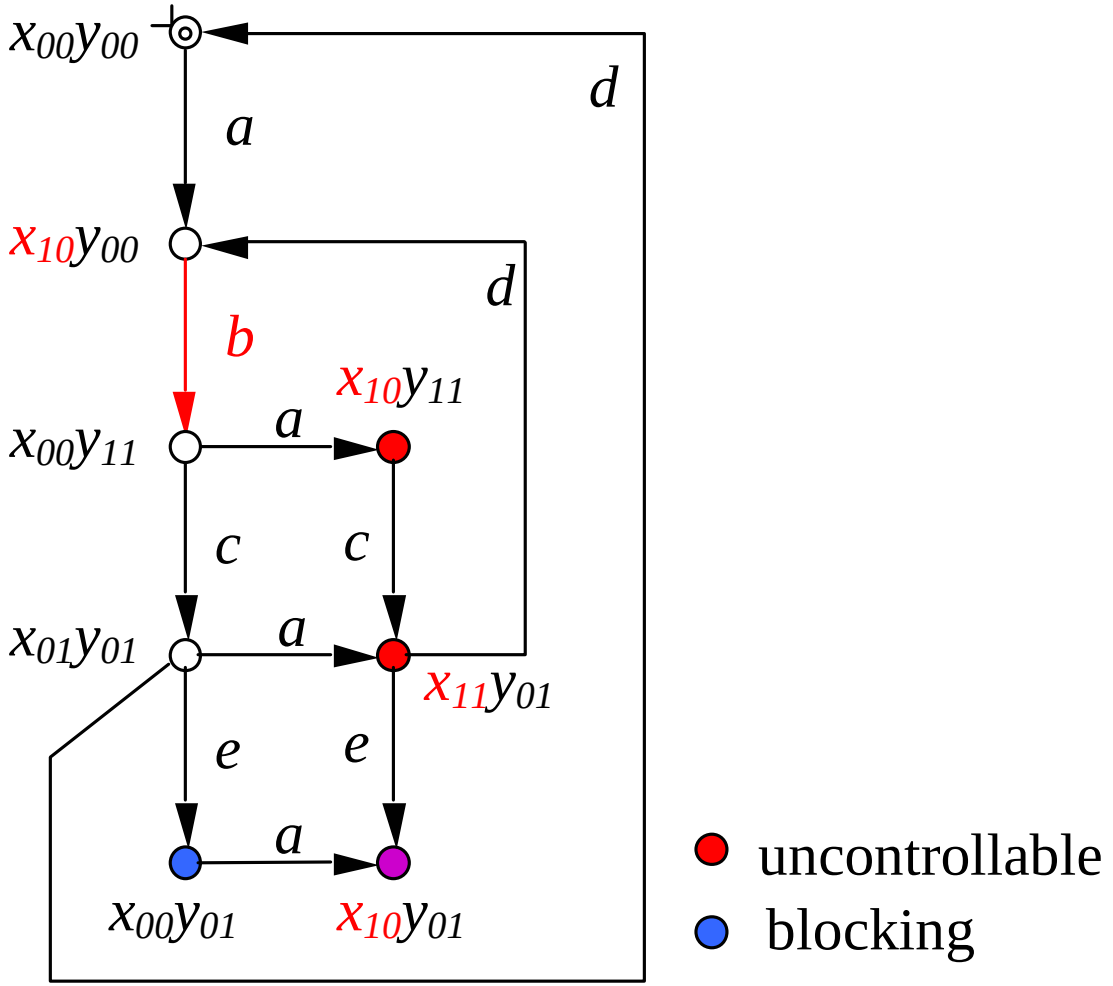
$E = G \parallel H$



$G = G_1 || G_2$



$H = H_1 || H_2$

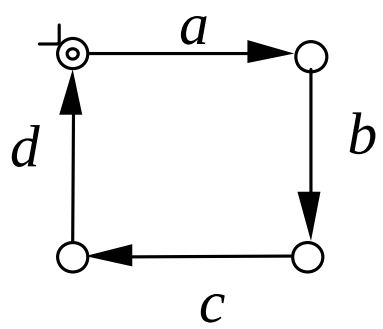


uncontrollable

blocking

Example using automata

Finally trimming the net E we obtain the maximally permissive supervisor



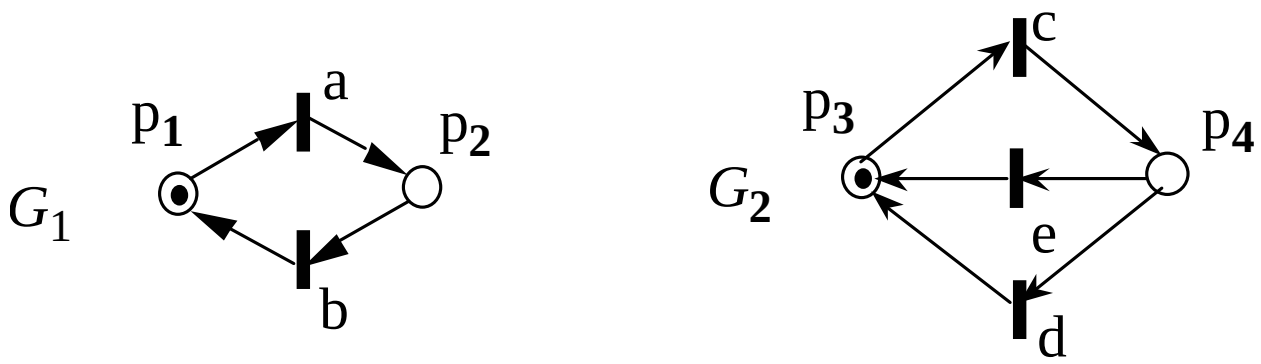
S

This also represents the behavior of the closed-loop plant.

Example using Petri nets

Given:

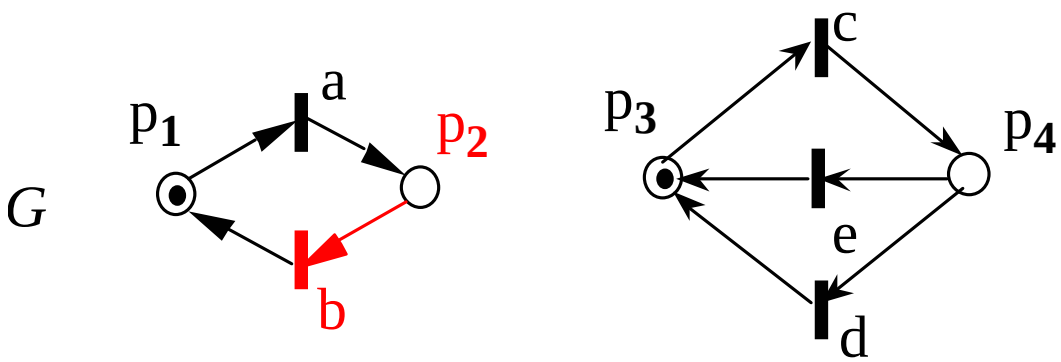
- G_1 with alphabet $\Sigma_1 = \{a, b\}$ and $F_1 = \{[1\ 0]^T\}$
- G_2 with alphabet $\Sigma_2 = \{c, d, e\}$ and $F_2 = \{[1\ 0]^T\}$



Note: the uncontrollable transition b is enabled when the marking of first subsystem is $[0\ 1]^T$.

Example using Petri nets

The concurrent composition requires "fusing" common transitions, i.e., transitions sharing the same label. Since there are not common transitions the two nets taken together also represent the composed system $G = G_1 \parallel G_2$ on alphabet $\Sigma = \Sigma_1 \cup \Sigma_2$ with $F = \{[1\ 0\ 1\ 0]^T\}$.

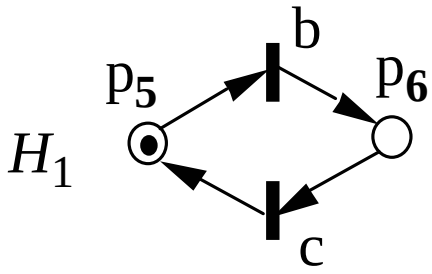


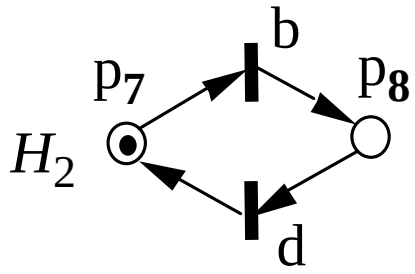
Note: the uncontrollable transition b is enabled when the marking of plant is $[0\ 1\ 1\ 0]^T$ or $[0\ 1\ 0\ 1]^T$.

Example using Petri nets

Specifications:

- H_1 with alphabet $\Sigma'_1 = \{b, c\}$ and $F'_1 = \{[1\ 0]^T\}$
- H_2 with alphabet $\Sigma'_2 = \{b, d\}$ and $F'_1 = \{[1\ 0]^T\}$

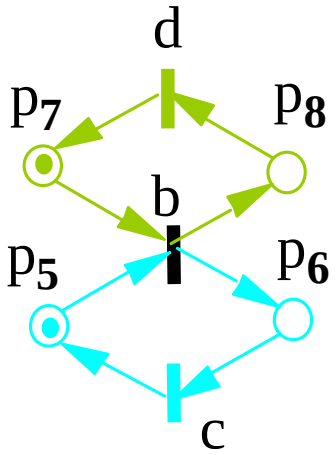




Overall specification on

$$\Sigma' = \Sigma'_1 \cup \Sigma'_2 = \{b, c, d\}$$

with $F' = \{[1\ 0\ 1\ 0]^T\}$

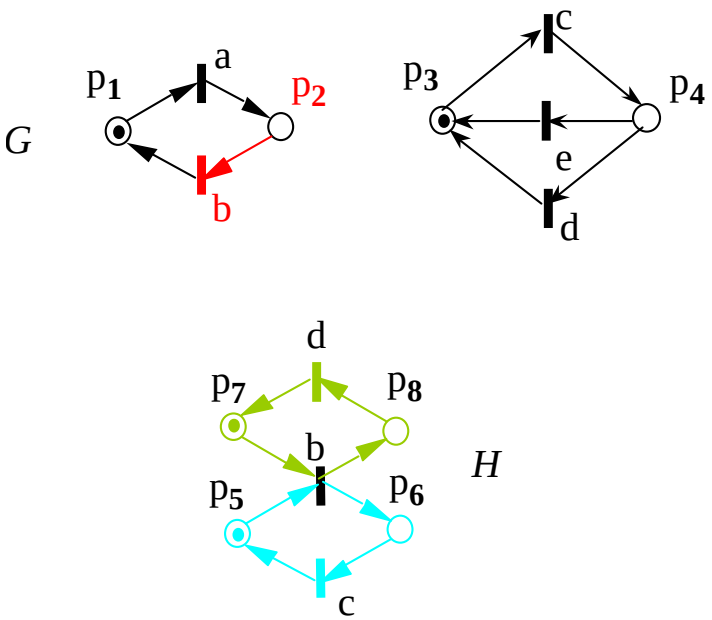


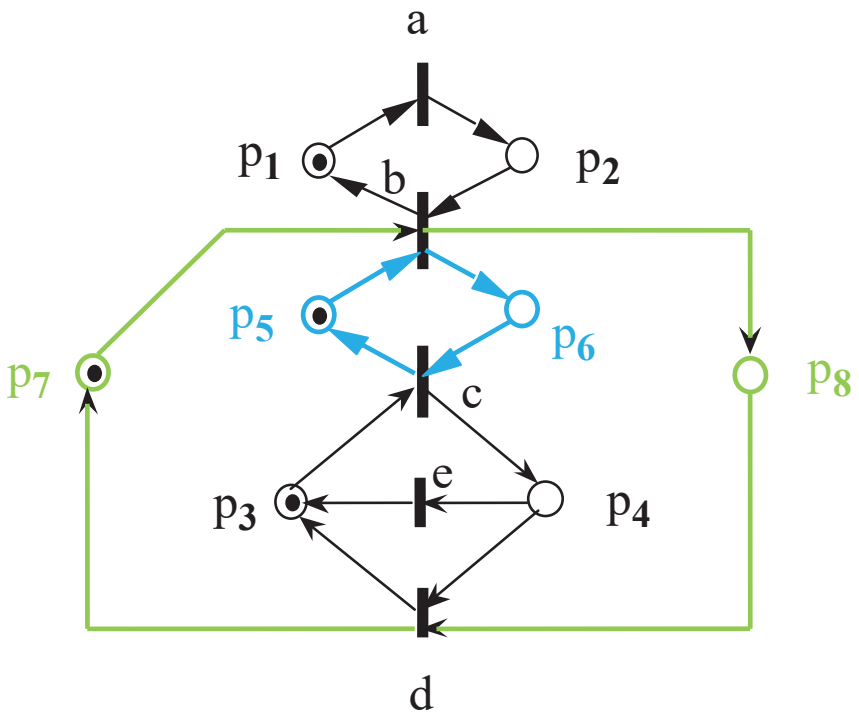
H

Example using Petri nets

$$E = G \parallel H$$

$$F = \{[1\ 0\ 1\ 0\ 1\ 0\ 1\ 0]^T\}$$



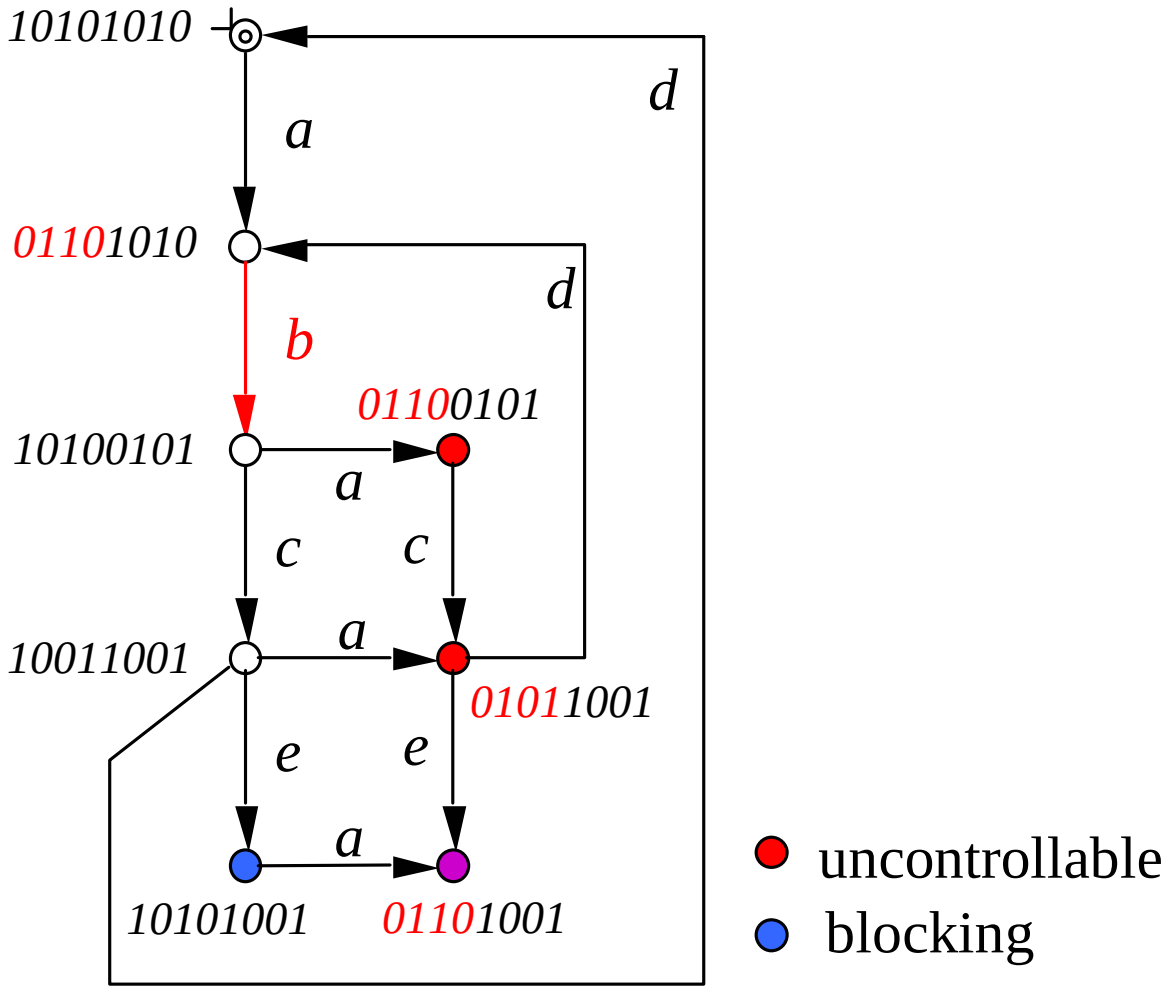


How can we check if this generator is controllable and non-blocking?

Example using Petri nets

Several techniques can be used (reachability set, structural analysis, unfolding).

Here we consider the exhaustive analysis of the state space based on the construction of the reachability set of the net.



Example using Petri nets

To avoid reaching a bad marking:

- We may remove transition e
- We must control transition a: it should fire from

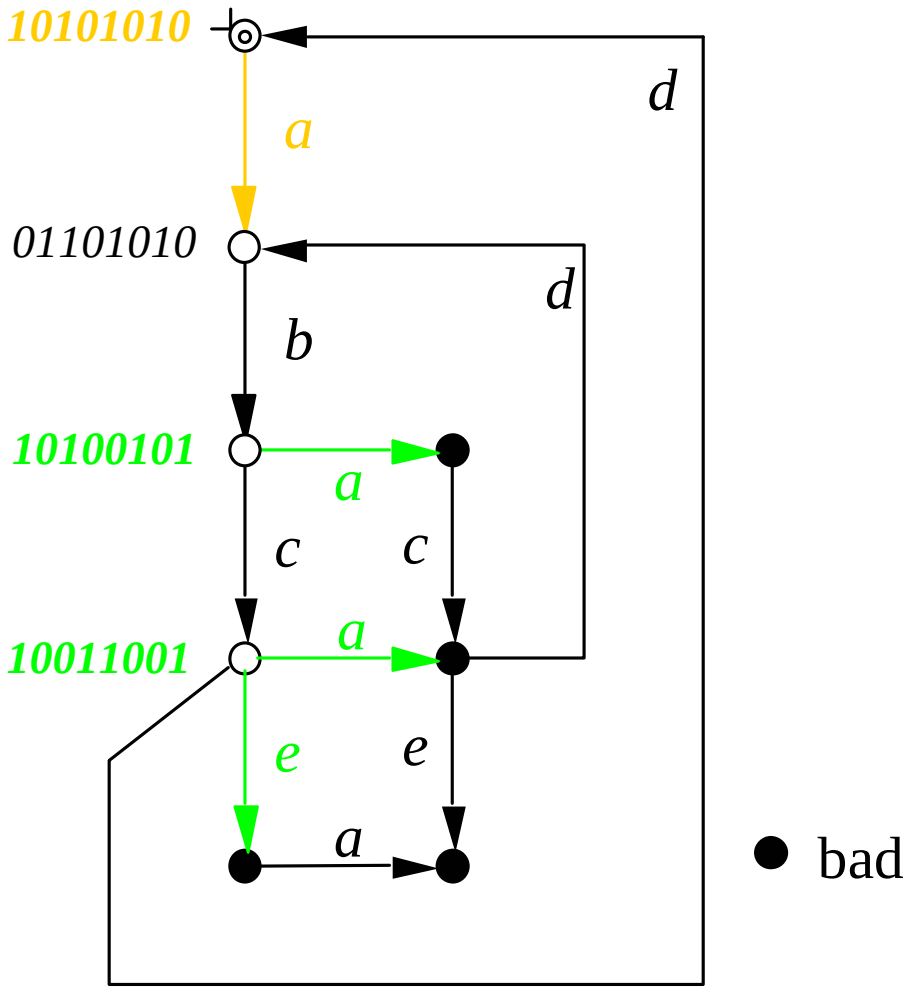
$$[1\ 0\ 1\ 0\ 1\ 0\ \mathbf{1}\ 0]^T$$

but not from

$$[1\ 0\ 1\ 0\ 0\ 1\ \mathbf{0}\ 1]^T$$

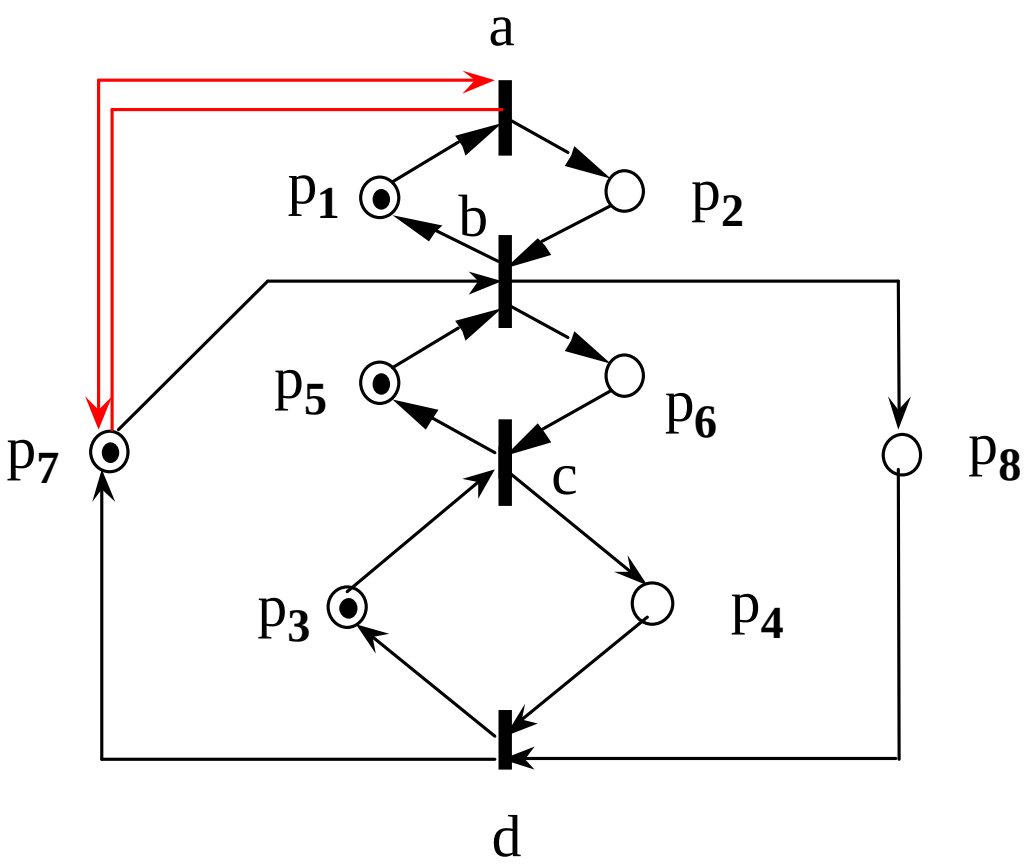
or

$$[1\ 0\ 0\ 1\ 1\ 0\ \mathbf{0}\ 1]^T$$



Example using Petri nets

Idea: block transition a if p_7 is not marked. This can be done adding a self-loop arc from p_7 to transition a.



Algorithm to refine a Petri nets

In general the following procedure can be used to trim a net.

Algorithm Let t be a transition to be controlled and α be its label.

1. Determine the set of admissible reachable markings that enable t, and partition this set in the disjoint subsets M_e (the markings from which t should be allowed to fire), and M_d (the markings from which t should not be allowed to fire, to avoid reaching a forbidden marking).

   If $M_e = \emptyset$ remove t and stop, else continue.

Algorithm to refine a Petri nets

2 Determine a construct in the form:

$$\begin{aligned}\mathcal{P}(\boldsymbol{\mu}) = \quad & [(\boldsymbol{\mu}(p_{1,1}) \geq n_{1,1}) \wedge \ldots \wedge (\boldsymbol{\mu}(p_{1,k_1}) \geq n_{1,k_1})] \vee \\ & \ldots \\ & \vee [(\boldsymbol{\mu}(p_{r,1}) \geq n_{r,1}) \wedge \ldots \wedge (\boldsymbol{\mu}(p'_{r,k_r}) \geq n_{r,k_r})]\end{aligned}$$

such that $\mathcal{P}(\boldsymbol{\mu}) =$ TRUE if $\boldsymbol{\mu} \in M_e$, and $\mathcal{P}(\boldsymbol{\mu}) =$ FALSE if $\boldsymbol{\mu} \in M_d$. Here $p_{i,j} \in P$, and $\vee$ and $\wedge$ are the logical OR and AND.

Algorithm to refine a Petri nets

3 Replace transition t with r transitions $t^1, \ldots, t^r$ labeled α. The input (output) arcs of transition t^i, $i = 1, \ldots, r$ will be those of transition t plus $n_{i,j}$ arcs inputting from (outputting to) place $p_{i,j}$, $j = 1, \ldots, k_i$.

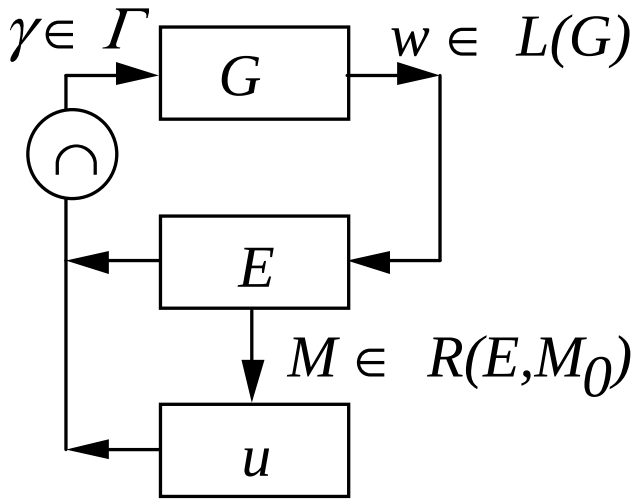
This construct can always be determined if the net is conservative.

It does not require adding places to the net but only splitting transitions.

Final comment

Note that the control law needs not be implemented in a net structure.

The use of the generator E allows one to use state feedback to solve a language specification problem (that should require event feedback).



Additional reading

- A first survey on Petri nets and supervisory control is:

  L.E. Holloway, B.H. Krogh, A. Giua, "A Survey of Petri Net Methods for Controlled Discrete Event Systems," *Discrete Event Dynamic Systems*, Vol. 7, pp. 151-190, 1997.

- Petri net supervisory control with language specifications is also discussed in:

  Giua A., "Supervisory control of Petri nets with language specifications," *Control of Discrete-Event Systems. Automata and Petri Net Perspectives*, C. Seatzu, M. Silva, J.H. van Schuppen (Eds), in Lecture Notes in Control and Information Science, Springer, Vol. 433, pp.235-256, 2012.

Outline

Motivation

Petri net supervisory design with state specifications has received a lot of attention.

We consider a special class of problems:

- The state specifications are given as a set of linear conditions called generalized mutual exclusion constraints (GMECs).
- When all transition are controllable an optimal solution takes the form of a monitor place (Giua, DiCesare and Silva, 1992).
- When some transitions are not controllable it may still be possible to find monitor solutions, but they may be suboptimal (more restrictive that necessary). This approach is due to Moody and Antsaklis (1996).

Other approaches for state based feedback control of Petri nets: incidence matrix control (Li and Wonham, 1993), controlled Petri nets (Holloway and Krogh, 1990).

Mutual exclusion

Mutual exclusion: Two logical variables A_1 and A_2 are exclusive if:

$$(A_1 \Longrightarrow \neg A_2) \wedge (A_2 \Longrightarrow \neg A_1)$$

Ex: Machine M_1 and machine M_2 should not be working at the same time

Assume variable A_i is true when place p_i $(i = 1, 2)$ in a net is marked:

$$M(p_1) \cdot M(p_2) = 0.$$

If the places are 1-bounded can be rewritten in linear form:

$$M(p_1) + M(p_2) \leq 1.$$

For a net with m place we generalize this condition:

$$\mathbf{w}^T M = w_1 M(p_1) + w_2 M(p_2) + \cdots + w_m M(p_m) \leq k,$$

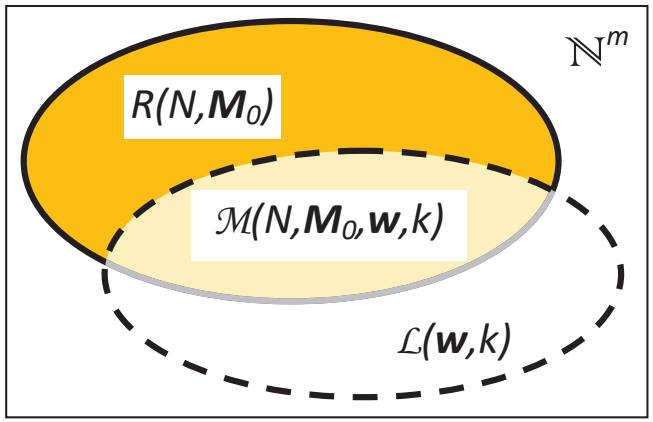
Generalized mutual exclusion constraints

Definition

A generalized mutual exclusion constraint (GMEC) is a pair $(\mathbf{w}, k)$ where:

$$\mathbf{w} = \begin{bmatrix} w_1 & w_2 & \cdots & w_m \end{bmatrix}^T \in \mathbb{Z}^m \quad \text{and} \quad k \in \mathbb{Z}.$$

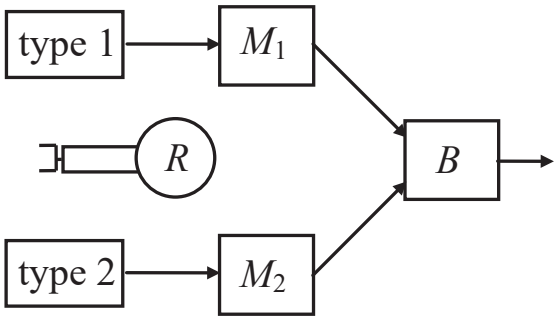
A GMEC $(\mathbf{w}, k)$ is a state specification that defines a set of legal markings on the reachability set $R(N, M_0)$ of a net.

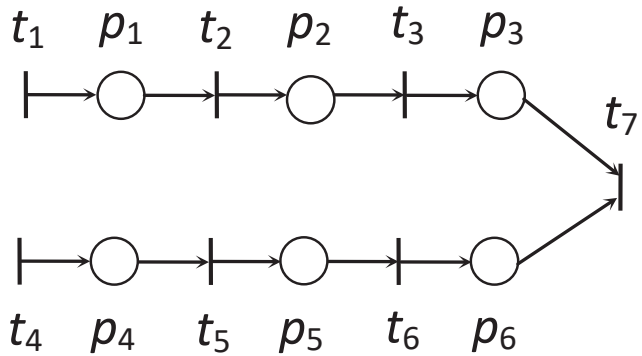


Example:

- Set of legal markings:
  $\mathcal{L}(\mathbf{w}, k) = \{M \in \mathbb{N}^m \mid \mathbf{w}^T M \leq k\}$.
- Set of legal reachable marking:
  $\mathcal{M}(N, M_0, \mathbf{w}, k) = R(N, M_0) \cap \mathcal{L}(\mathbf{w}, k)$.

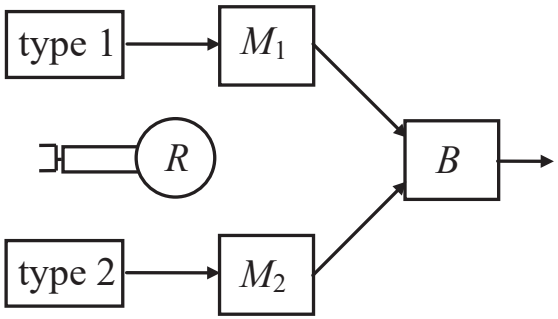
Example

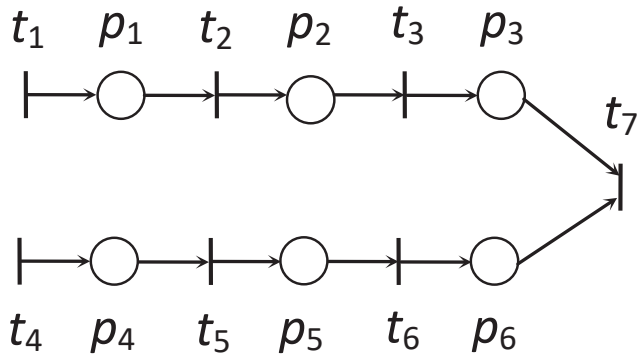




Place	**Description**
p_1	loading M_1
p_2	M_1 working
p_3	parts type 1 in buffer
p_4	loading M_2
p_5	M_2 working
p_6	parts type 2 in buffer

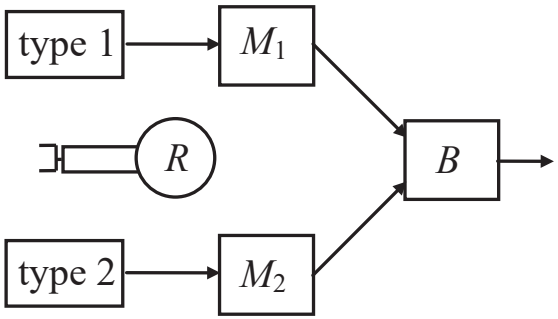
Example

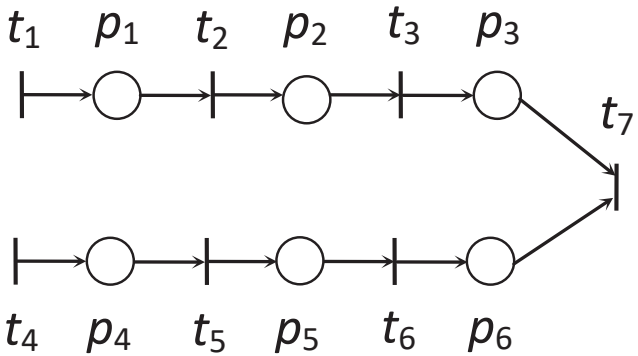




1. only one robot available
2. single server machines
3. buffer has k slots; type 1 parts take 1 slot, type 2 parts take 2 slots
4. difference in buffer among parts type 1 and 2 should be less or equal to k'

Example





1. $M(p_1) + M(p_4) \leq 1$
2. $M(p_2) \leq 1$
   $M(p_5) \leq 1$
3. $M(p_3) + 2M(p_6) \leq k$
4. $M(p_3) - M(p_6) \leq k'$
   $-M(p_3) + M(p_6) \leq k'$

Multiple GMECs and descriptive power

Assume a set of GMECs is given $(\mathbf{w}_i, k_i)$ for $i = 1, \ldots, q$. We can define a multiple GMEC $(\mathbf{W}, \mathbf{k})$ where

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \cdots & \mathbf{w}_q \end{bmatrix} \quad \text{and} \quad \mathbf{k} = \begin{bmatrix} k_1 & k_2 & \cdots & k_q \end{bmatrix}^T$$

The set of legal markings is now

$$\mathcal{L}(\mathbf{W}, \mathbf{k}) = \{M \in \mathbb{N}^m \mid \mathbf{W}^T M \leq \mathbf{k}\} = \bigcap_{i=1,\ldots,q} \mathcal{L}(\mathbf{w}_i, k_i).$$

This is an integer convex set $\Longrightarrow$ not all possible sets of legal marking $\mathcal{L} \subseteq \mathbb{N}^m$ can be represented by GMECs.

Example: assume in $\mathbb{N}^2$ a legal set is $\mathcal{L} = \{M_1,\ M_2\} = \{[2\ 0]^T,\ [0\ 2]^T\}$. There exists no multiple GMEC $(\mathbf{W}, \mathbf{k})$ such that $\mathcal{L} = \mathcal{L}(\mathbf{W}, \mathbf{k})$ since $M = (M_1 + M_2)/2 = [1\ 1]^T \notin \mathcal{L}$.

Special case: safe nets

A net is safe is all places are 1-bounded.

Proposition (Giua, DiCesare, and Silva, 1992)

If a net $\langle N, M_0 \rangle$ *is safe, given an arbitrary marking specification* $\mathcal{L}$ *there exists a multiple GMEC* $(\mathbf{W}, \mathbf{k})$ *such that*

$$R(N, M_0) \cap \mathcal{L} = R(N, M_0) \cap \mathcal{L}(\mathbf{W}, \mathbf{k}).$$

The number of single GMECs that compose the multiple GMEC may be very large.

Outline

- Generalized mutual exclusion constraints
- **Monitor places**
- Monitors design for nets with uncontrollable transitions

Monitor

Given a net with incidence matrix $\mathbf{C}$, the monitor corresponding to GMEC $(\mathbf{w}, k)$ is a new place p_m whose incidence matrix is the vector $1 \times n$:

$$\mathbf{C}_m = \left[\begin{array}{ccc} C_m(t_1) & \cdots & C_m(t_n) \end{array} \right] = -\mathbf{w}^T \mathbf{C}$$

and whose initial marking is

$$M_0(p_m) = k - \mathbf{w}^T M_0.$$

The monitor is a Petri net controller that added to a net ensures that the constraint is never violated. The closed loop system has the following incidence matrix and initial marking

$$\mathbf{C}_{cc} = \left[\begin{array}{c} \mathbf{C} \\ \mathbf{C}_m \end{array} \right] \qquad M_{cc,0} = \left[\begin{array}{c} M_0 \\ M_0(p_m) \end{array} \right]$$

Example (cont'd)

The net of the example has incidence matrix and initial marking

$$\mathbf{C} = \begin{array}{c} \begin{matrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 \end{matrix} \\ \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \end{array} \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{matrix} \qquad M_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{matrix}$$

The monitor p_{m1} corresponding to GMEC $(\mathbf{w}_1, k_1)$ with

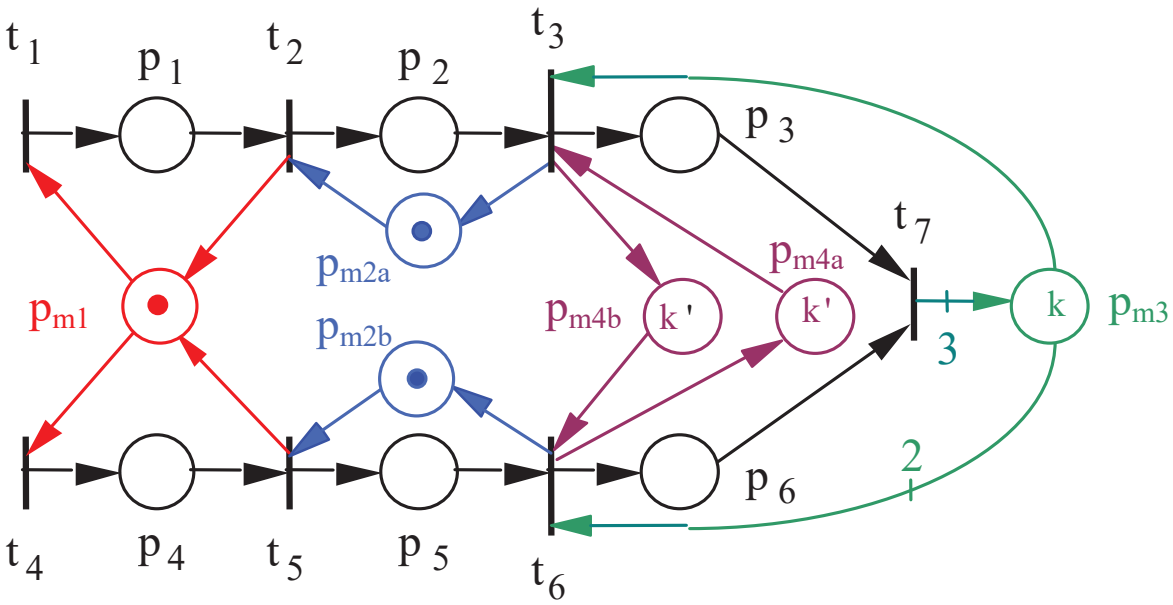
$$\mathbf{w}_1 = [\ 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0\ ]^T \qquad k_1 = 1$$

has incidence matrix:

$$\mathbf{C}_{m1} = -\mathbf{w}_1^T \mathbf{C} = \begin{array}{c} \begin{matrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 \end{matrix} \\ \begin{bmatrix} -1 & 1 & 0 & -1 & 1 & 0 & 0 \end{bmatrix} \end{array},$$

and initial marking $M_0(p_{m1}) = k_1 - \mathbf{w}_1^T M_0 = 1 - 0 = 1$.

Example (cont'd)



1. $M(p_1) + M(p_4) \leq 1$
2. $M(p_2) \leq 1$
   $M(p_5) \leq 1$
3. $M(p_3) + 2M(p_6) \leq k$
4. $M(p_3) - M(p_6) \leq k'$
   $-M(p_3) + M(p_6) \leq k'$

Monitors and invariants

Consider the GMEC:

$$w_1 M(p_1) + w_2 M(p_2) + \cdots + w_m M(p_m) \leq k.$$

Introducing a slack variable $M(p_m)$ we obtain this invariant law:

$$w_1 M(p_1) + w_2 M(p_2) + \cdots + w_m M(p_m) + M(p_m) = k.$$

This means that the vector $\mathbf{x} = [w_1 \ w_2 \ \ldots \ w_m \ 1]^T$ is an invariant for the closed loop systems, i.e.,

$$\mathbf{x}^T \ \mathbf{C}_{cc} = \mathbf{0}.$$

Properties of a monitor

1. A monitor for $(\mathbf{w}, k)$ is a maximally permissive controller, i.e., it only blocks the firing of transitions that lead to reachable markings that are not legal, i.e., markings not in $\mathcal{M}(N, M_0, \mathbf{w}, k)$.

2. A monitor for $(\mathbf{w}, k)$ is realizable if and only if the initial marking of the net is legal.
   - The incidence matrix $\mathbf{C}_m$ can always be computed.
   - The initial marking $M_0(p_m)$ of the monitor must be positive:

$$M_0(p_m) = k - \mathbf{w}^T M_0 \geq 0 \iff \mathbf{w}^T M_0 \leq k \iff M_0 \in \mathcal{L}(\mathbf{w}, k).$$

3. A monitor is self-loop free, i.e., cannot have pre and post arcs to and from the same transition. This means that the incidence matrix $\mathbf{C}_m$ uniquely defines the arcs incident on monitor place p_m.

Outline

- Generalized mutual exclusion constraints
- Monitor places
- **Monitors design for nets with uncontrollable transitions**

Uncontrollable transitions

Assume set of transitions is partitioned $T = T_{uc} \cup T_c$ with

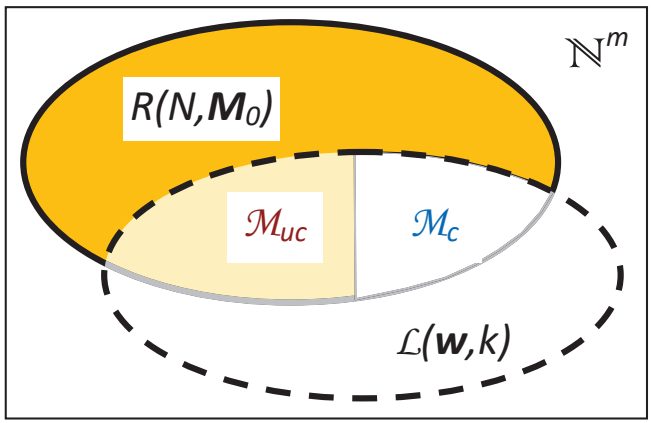
- T_{uc}: set of uncontrollable transitions: cannot be control disabled.
- T_c: set of controllable transitions: can be control disabled.

Set of legal reachable marking $\mathcal{M}(N, M_0, \mathbf{w}, k) = \mathcal{M}_{uc} \cup \mathcal{M}_c$ with

- $\mathcal{M}_{uc}$: uncontrollable set. For any marking $M \in \mathcal{M}_{uc}$ it holds:

$$\exists \sigma \in T_{uc}^* \ : \ M[\sigma\rangle M_f, \ M_f \notin \mathcal{L}(\mathbf{w}, k).$$

- $\mathcal{M}_c = \mathcal{M}(N, M_0, \mathbf{w}, k) \setminus \mathcal{M}_{uc}$: controllable set.



We need to refine our control structure to avoid reaching all markings in $\mathcal{M}_{uc}$ thus restricting the plant to the controllable set $\mathcal{M}_c$.

Controllable monitors

Consider a net with $T = T_{uc} \cup T_c$ and monitor p_m for GMEC $(\mathbf{w}, k)$.

Definition

Monitor p_m and GMEC $(\mathbf{w}, k)$ are admissible if for any marking of the closed loop systems $M_{cc} = \left[\begin{array}{c|c} M^T & M(p_m) \end{array} \right]^T$ and for all $t \in T_{uc}$ it holds: $M[t\rangle \implies M(p_m) \geq Pre(p_m, t)$.

No plant-enabled uncontrollable transition is disabled by the monitor.

Definition

Monitor p_m and GMEC $(\mathbf{w}, k)$ are structurally controllable (or controllable for short) if for all $t \in T_{uc}$ it holds $Pre(p_m, t) = 0$.

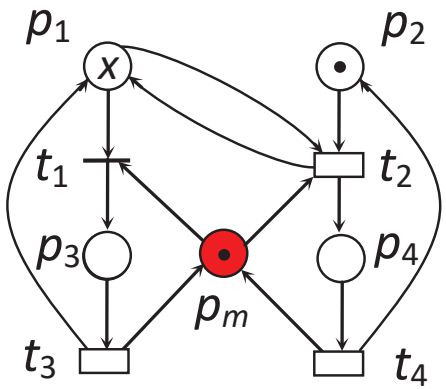
The monitor has no pre arcs going to uncontrollable transitions.

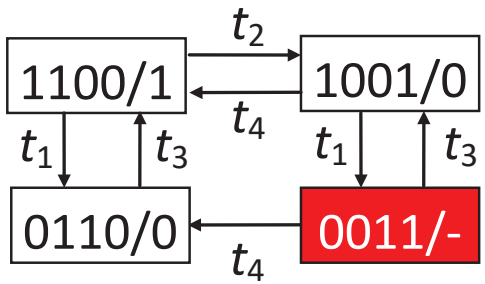
Example

Note: p_m is controllable $\Longrightarrow$ p_m is admissible.

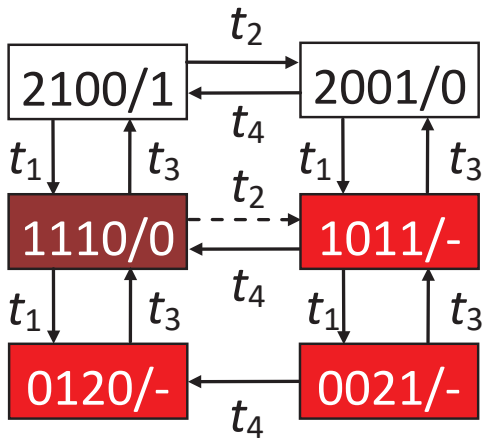
Consider the net with GMEC $M(p_3) + M(p_4) \leq 1$ where uncontrollable transitions are shown as white boxes and $M_0(p_1) = x$.

- The GMEC and its monitor are not controllable.
- The GMEC and its monitor are admissible for $x = 1$.





x=1

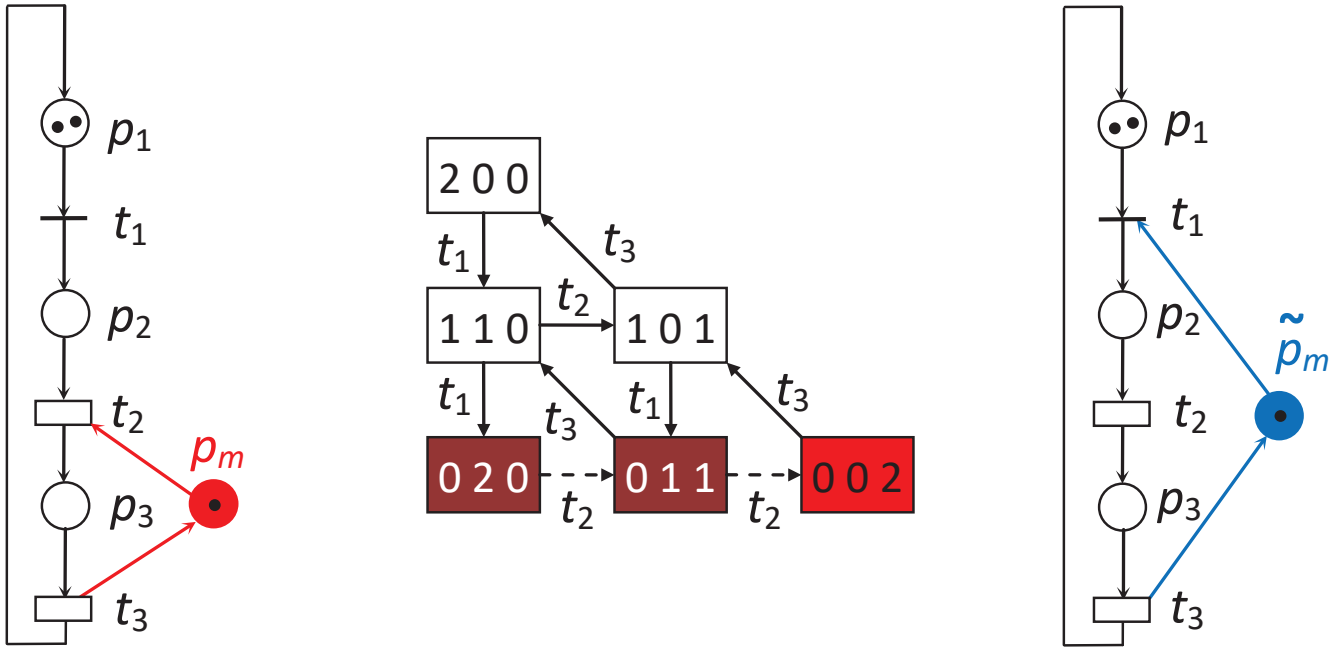


x=2

Controllability is stronger than admissibility but easier to check.

Example

If a GMEC $(\mathbf{w}, k)$ is not controllable, can we find a controllable GMEC $(\tilde{\mathbf{w}}, \tilde{k})$ that prevents reaching marking in $\mathcal{M}_{uc}$?



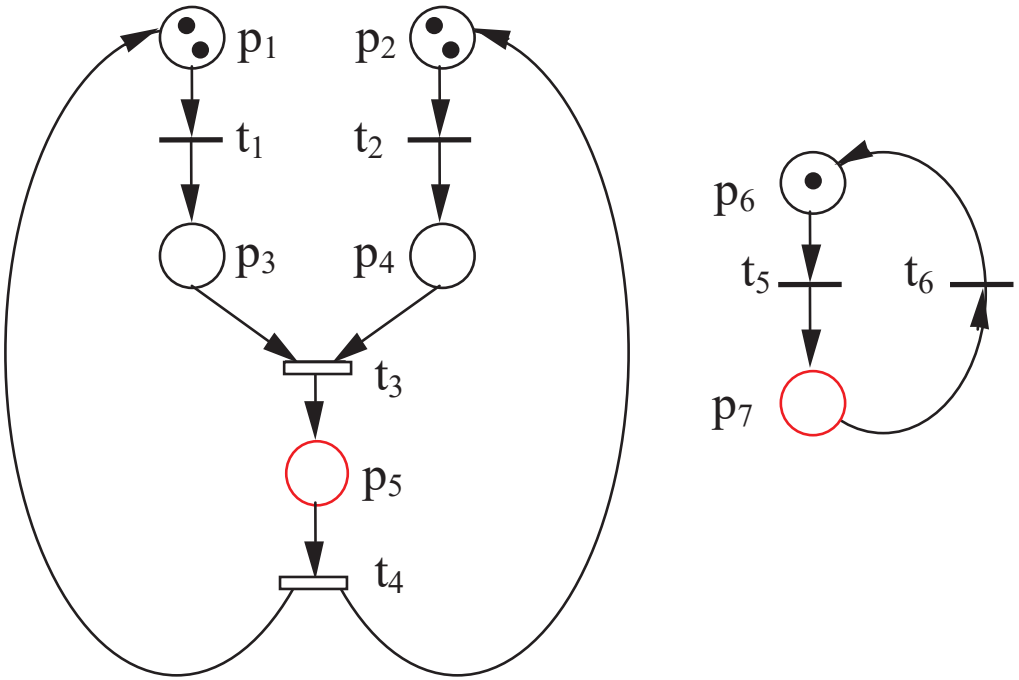
GMEC $M(p_3) \leq 1$ and its monitor p_m are not controllable.
The more restrictive GMEC $M(p_2) + M(p_3) \leq 1$ and its monitor $\tilde{p}_m$ are controllable and prevent reaching uncontrollable markings.

The bad news (Giua, DiCesare and Silva, 1992)

A maximally permissive controllable GMEC may not exist on nets with uncontrollable transitions.

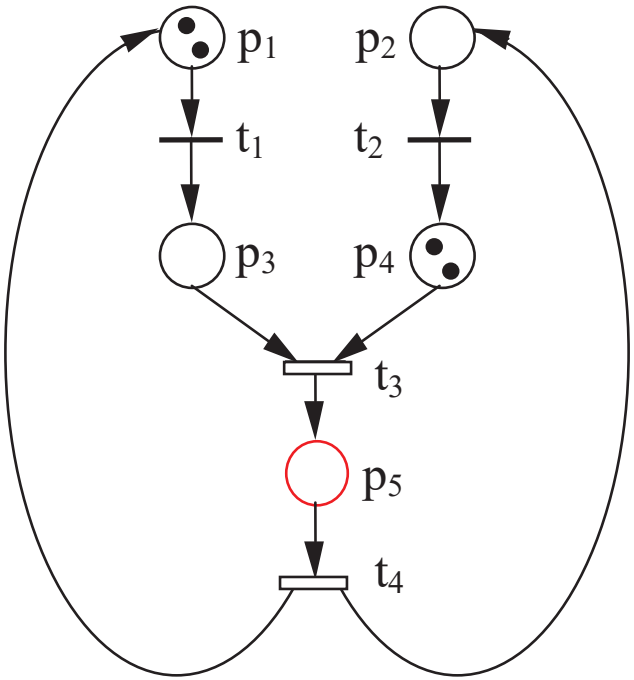
Want to enforce

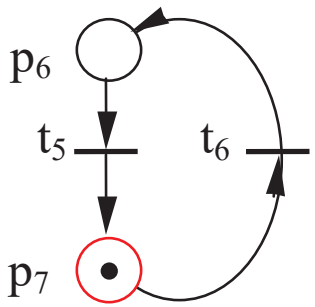
$$M(p_5) + M(p_7) \leq 1.$$



The bad news (Giua, DiCesare and Silva, 1992)

A maximally permissive controllable GMEC may not exist on nets with uncontrollable transitions.





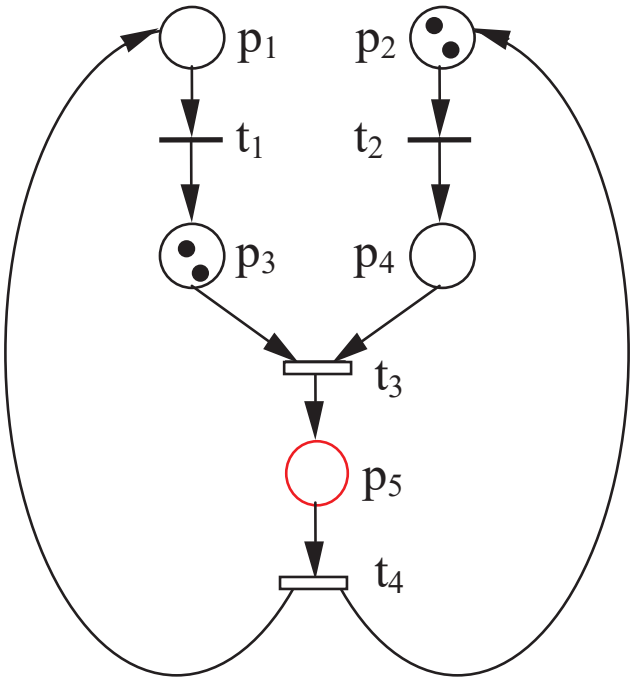
Want to enforce

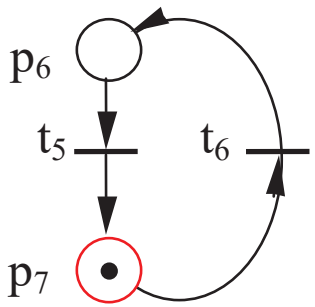
$$M(p_5) + M(p_7) \leq 1.$$

$M_1 = [2\ 0\ 0\ 2\ 0\ 1]^T$ is ok

The bad news (Giua, DiCesare and Silva, 1992)

A maximally permissive controllable GMEC may not exist on nets with uncontrollable transitions.





Want to enforce

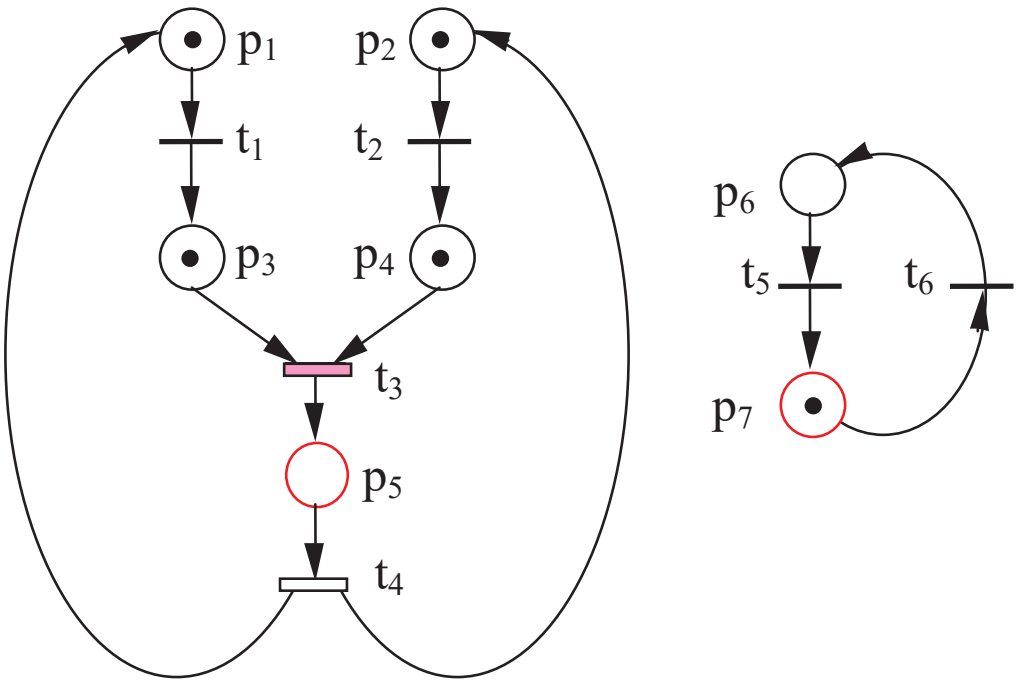
$$M(p_5) + M(p_7) \leq 1.$$

$M_1 = [2\ 0\ 0\ 2\ 0\ 1]^T$ is ok

$M_2 = [0\ 2\ 2\ 0\ 0\ 1]^T$ is ok

The bad news (Giua, DiCesare and Silva, 1992)

A maximally permissive controllable GMEC may not exist on nets with uncontrollable transitions.



Want to enforce

$$M(p_5) + M(p_7) \leq 1.$$

$M_1 = [2\ 0\ 0\ 2\ 0\ 1]^T$ is ok

$M_2 = [0\ 2\ 2\ 0\ 0\ 1]^T$ is ok

$M = [1\ 1\ 1\ 1\ 0\ 1]^T$ is NOT

A (set of) GMEC cannot allow M_1 and M_2 while preventing $M = (M_1 + M_2)/2$ (set of legal markings must be convex).

The good news (Moody and Antsaklis, 1996)

If we are not interested in finding a maximally permissive solution a suboptimal monitor solution often exists (may not be unique).

A suboptimal solution may be found looking for a more restrictive but controllable GMEC.

Proposition (Moody and Antsaklis' parametrization)

Given a GMEC $(\mathbf{w}, k)$ *with* $\mathbf{w} \in \mathbb{Z}^m$, *the GMEC* $(\tilde{\mathbf{w}}, \tilde{k})$ *where:*

$$\tilde{\mathbf{w}} = (r\mathbf{w} + \mathbf{z}) \quad \textit{and} \quad \tilde{k} = r(k+1) - 1$$

with $r \in \mathbb{N}$ *and* $\mathbf{z} \in \mathbb{N}^m$ *is more restrictive than* $(\mathbf{w}, k)$, *i.e.,*

$$\mathcal{L}(\tilde{\mathbf{w}}, \tilde{k}) \subseteq \mathcal{L}(\mathbf{w}, k).$$

Characterizing controllable GMECs

Given a net with incidence matrix $\mathbf{C}$ let $\mathbf{C}_{uc}$ be the uncontrollable incidence matrix obtained from $\mathbf{C}$ removing all columns associated to controllable transition.

If GMEC $(\mathbf{w}, k)$ is controllable, its monitor places p_m has no pre arc going to uncontrollable transitions, i.e.,

$$\mathbf{C}_{m,uc} = -\mathbf{w}^T \mathbf{C}_{uc} \geq \mathbf{0}^T \qquad \Longleftrightarrow \qquad \mathbf{w}^T \mathbf{C}_{uc} \leq \mathbf{0}^T.$$

Finding controllable GMECs

Algorithm (Moody and Antsaklis, 1996)

A GMEC $(\mathbf{w}, k)$ *is given.*

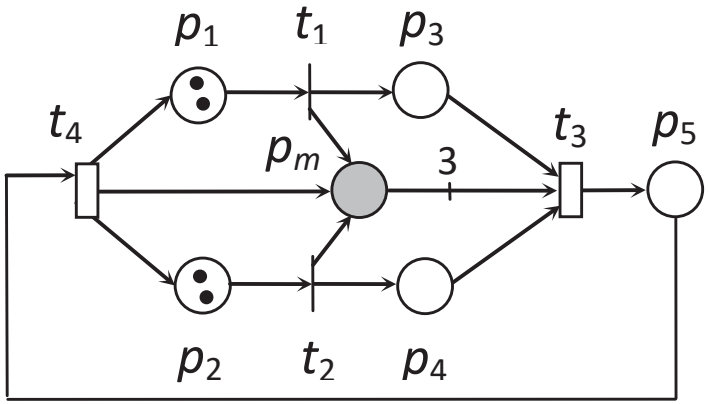
1. *Construct matrix*

$$\left|\begin{array}{c|c|c} \mathbf{C}_{uc} & \mathbf{I}_{m\times m} & \mathbf{0} \\ \hline \mathbf{v}^T & \mathbf{z}^T & 0 \end{array}\right| = \left|\begin{array}{c|c|c} \mathbf{C}_{uc} & \mathbf{I}_{m\times m} & \mathbf{0} \\ \hline \mathbf{w}^T\mathbf{C}_{uc} & \mathbf{0}^T & 1 \end{array}\right|$$

2. **If** $\mathbf{v}^T \leq \mathbf{0}$ **then** *the GMEC is controllable* **else** *keep adding to the last row (eventually multiplied by a positive integer) previous rows to change to* 0 *positive elements in* $\mathbf{v}^T$.
3. *If a vector* $\mathbf{v}^T \leq \mathbf{0}$ *is obtained, we have determined a controllable GMEC* $(\tilde{\mathbf{w}}, \tilde{k})$ *with* $\tilde{\mathbf{w}} = (r\mathbf{w} + \mathbf{z})$ *and* $\tilde{k} = r(k+1) - 1$ *in the form of the M&A parameterization.*

Example

We want enforce GMEC $(\mathbf{w}, k)$ such that $M(p_1) + M(p_2) + 3M(p_5) \leq 4$.



$$T_{uc} = \{t_3, t_4\}.$$

$$\mathbf{w} = [1\ 1\ 0\ 0\ 3]^T, \quad k = 4.$$

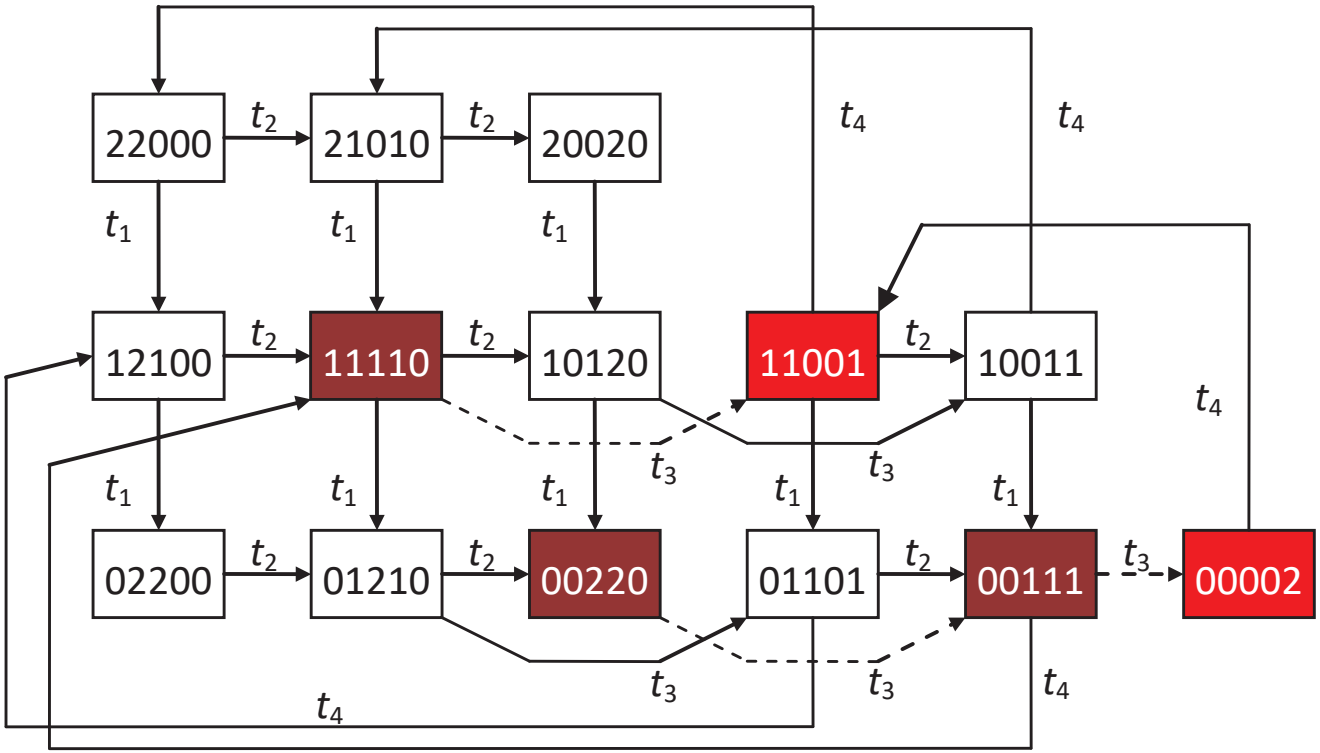
$$\mathbf{C} = [\mathbf{C}_c\ \mathbf{C}_{uc}] = \begin{bmatrix} -1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}.$$

$$\mathbf{C}_m = -\mathbf{w}^T\mathbf{C} = \begin{bmatrix} 1 & 1 & -3 & 1 \end{bmatrix}, \qquad \mathbf{w}^T C_{uc} = \begin{bmatrix} 3 & -1 \end{bmatrix} \not\leq \mathbf{0}^T.$$

The GMEC is not controllable: 3 arcs going to uncontrollable transition t_3.

Example (cont'd)

$$M(p_1) + M(p_2) + 3M(p_5) \leq 4.$$



Example (cont'd)

$$\left|\begin{array}{c|c|c} \mathbf{C}_{uc} & \mathbf{I}_{m\times m} & \mathbf{0} \\ \hline \mathbf{w}^T\mathbf{C}_{uc} & \mathbf{0}^T & 1 \end{array}\right| = \left|\begin{array}{cc|ccccc|c} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ \hline 3 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right|$$

Adding to last row three times row 3 we transform the last row:

$$\left|\begin{array}{c|c|c} \mathbf{v} & \mathbf{z}^T & r \end{array}\right| = \left|\begin{array}{cc|ccccc|c} 0 & -1 & 0 & 0 & 3 & 0 & 0 & 1 \end{array}\right|$$

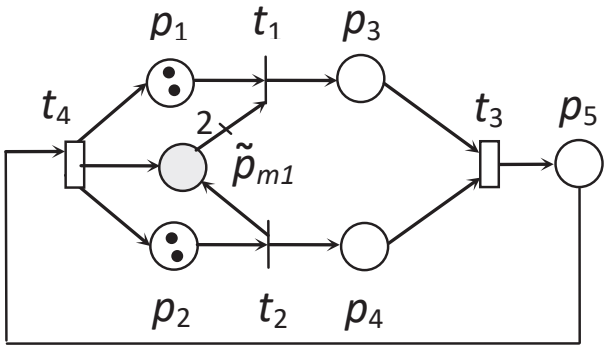
The algorithm stops determining controllable GMEC $(\tilde{\mathbf{w}}_1, \tilde{k}_1)$ with

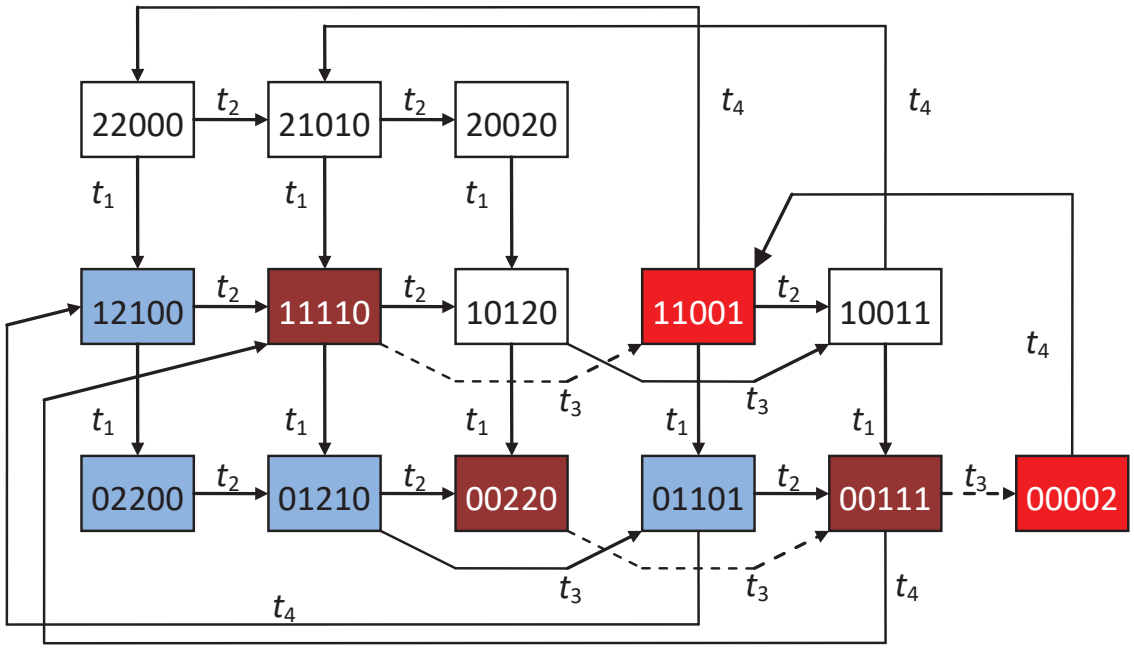
$$\tilde{\mathbf{w}}_1 = (r\mathbf{w} + \mathbf{z}) = 1\ [1\ 1\ 0\ 0\ 3]^T + [0\ 0\ 3\ 0\ 0]^T = [1\ 1\ 3\ 0\ 3]^T$$
$$\tilde{k}_1 = r(k+1) - 1 = k = 4.$$

enforcing $M(p_1) + M(p_2) + 3M(p_3) + 3M(p_5) \leq 4$.

Example (cont'd)

GMEC $(\tilde{\mathbf{w}}_1, \tilde{k}_1)$: $M(p_1) + M(p_2) + 3M(p_3) + 3M(p_5) \leq 4$.





The monitor is suboptimal: some controllable markings (blue) are not reachable under control.

Example (cont'd)

$$\left|\begin{array}{c|c|c} \mathbf{C}_{uc} & \mathbf{I}_{m\times m} & \mathbf{0} \\ \hline \mathbf{w}^T\mathbf{C}_{uc} & \mathbf{0}^T & 1 \end{array}\right| = \left|\begin{array}{cc|ccccc|c} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ \hline 3 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right|$$

Other choices are possible. Adding to last row three times row 4:

$$\left|\begin{array}{c|c|c} \mathbf{v} & \mathbf{z}^T & r \end{array}\right| = \left|\begin{array}{cc|ccccc|c} 0 & -1 & 0 & 0 & 0 & 3 & 0 & 1 \end{array}\right|$$

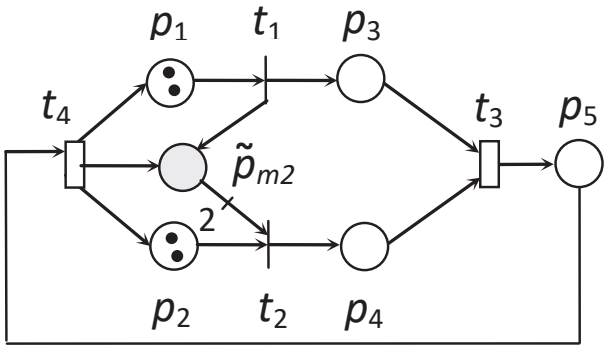
The algorithm stops determining controllable GMEC $(\tilde{\mathbf{w}}_2, \tilde{k}_2)$ with

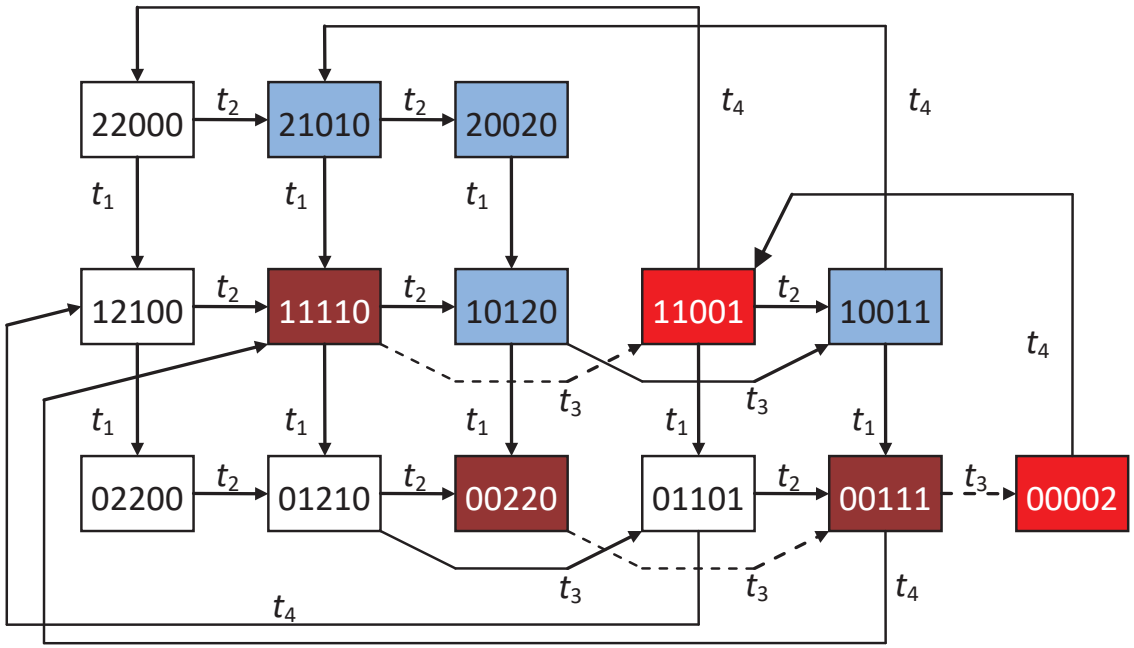
$$\tilde{\mathbf{w}}_2 = (r\mathbf{w} + \mathbf{z}) = 1\ [1\ 1\ 0\ 0\ 3]^T + [0\ 0\ 0\ 3\ 0]^T = [1\ 1\ 0\ 3\ 3]^T$$
$$\tilde{k}_2 = r(k+1) - 1 = k = 4.$$

enforcing $M(p_1) + M(p_2) + 3M(p_4) + 3M(p_5) \leq 4$.

Example (cont'd)

GMEC $(\tilde{\mathbf{w}}_2, \tilde{k}_2)$: $M(p_1) + M(p_2) + 3M(p_4) + 3M(p_5) \leq 4$.





This monitor is suboptimal as well: some controllable markings (blue) are not reachable under control.

Some remarks on this procedure

- The given algorithm may cycle indefinitely: it is possible to add a check for avoiding entering into cycles.
- A controllable GMEC may not exist: the algorithm will stop with $\mathbf{v} \not\leq \mathbf{0}$ (assuming no cycle).
- The algorithm may compute a controllable GMEC whose monitor place is realizable: this happens if $M_0 \notin M_c$.
- The procedure is easy to implement (no state space enumeration) but does not guarantees properties such as liveness, non-blockingness, reversibility. This topic is still open for on-going research.

Additional background reading

- The original paper defining monitors and GMECs:

  A. Giua, F. DiCesare, M. Silva, "Generalized Mutual Exclusion Constraints for Nets with Uncontrollable Transitions", *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics* (Chicago, USA), pp. 974-799, October 1992.

- A text-book presenting the Moody and Antsaklis approach:

  J.O. Moody, P.J. Antsaklis, *Supervisory Control of Discrete Event Systems using Petri Nets*, Kluwer Academic Publishers, 1998.