On the Computation of the Minimal Coverability Set of Petri Nets

Pierre-Alain Reynier and Frédéric Servais

LIS, Aix-Marseille Université & CNRS, France
Université Libre de Bruxelles, Belgium

RP 2019, Brussels, September 2019

Minimal Coverability Set

$\mathrm{MCS}(\mathcal{N})$ is an accurate over-approximation of the reachability set of $\mathcal{N}$.

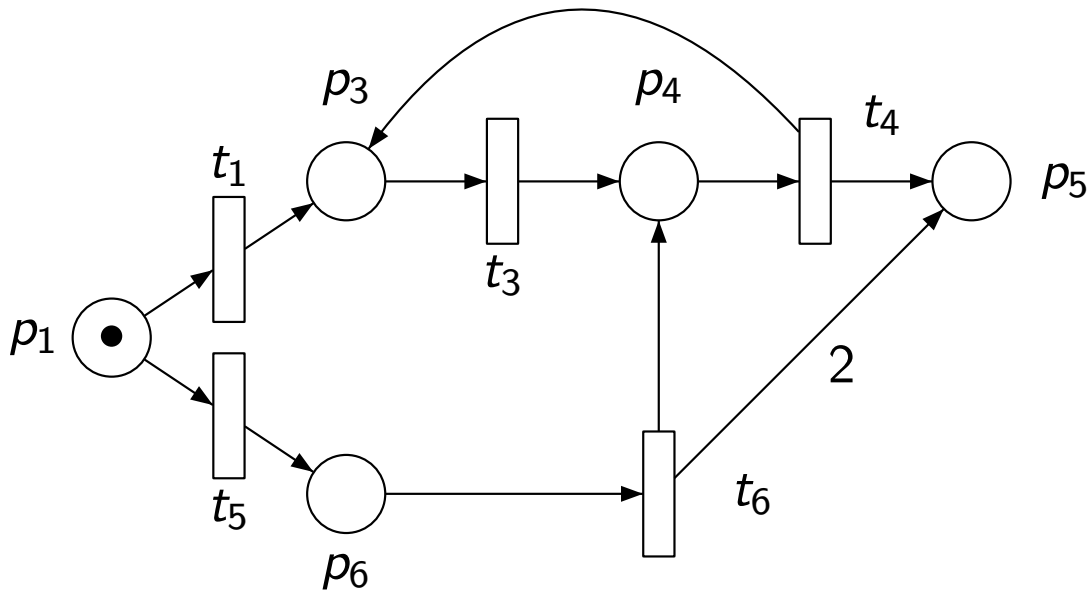
It allows to decide coverability, boundedness, place-boundedness, regularity...

Minimal Coverability Set

$\mathrm{MCS}(\mathcal{N})$ is an accurate over-approximation of the reachability set of $\mathcal{N}$.

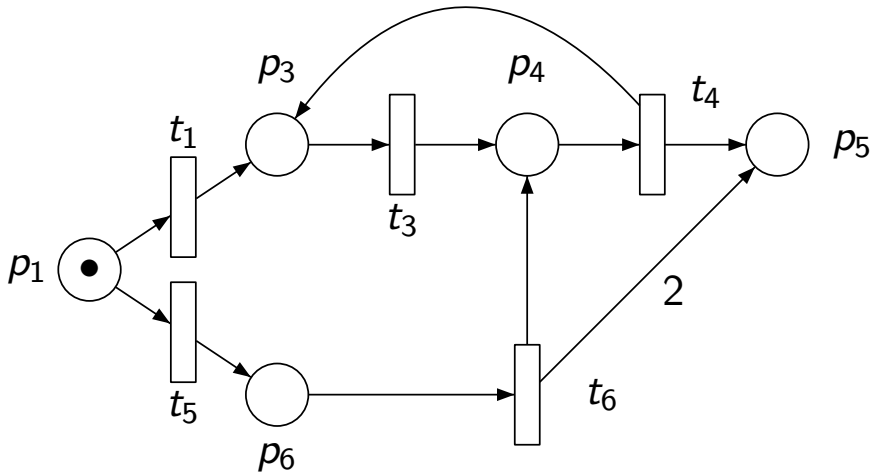
It allows to decide coverability, boundedness, place-boundedness, regularity...

Example:



$$\mathrm{MCS} = \{p_1, p_6, p_3 + \omega p_5, p_4 + \omega p_5\}$$

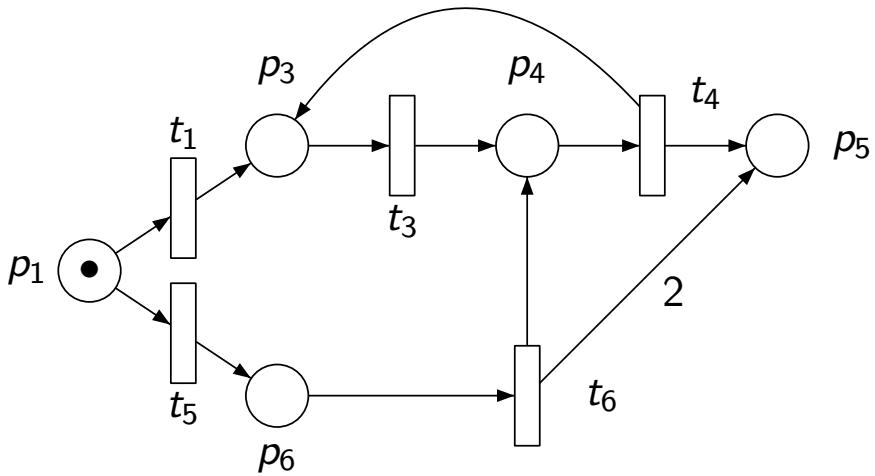
Karp & Miller Tree by example

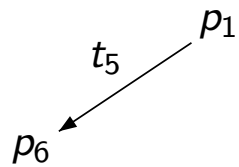


p_1

$$\mathrm{MCS} = \{p_1, p_6, p_3 + \omega p_5, p_4 + \omega p_5\}$$

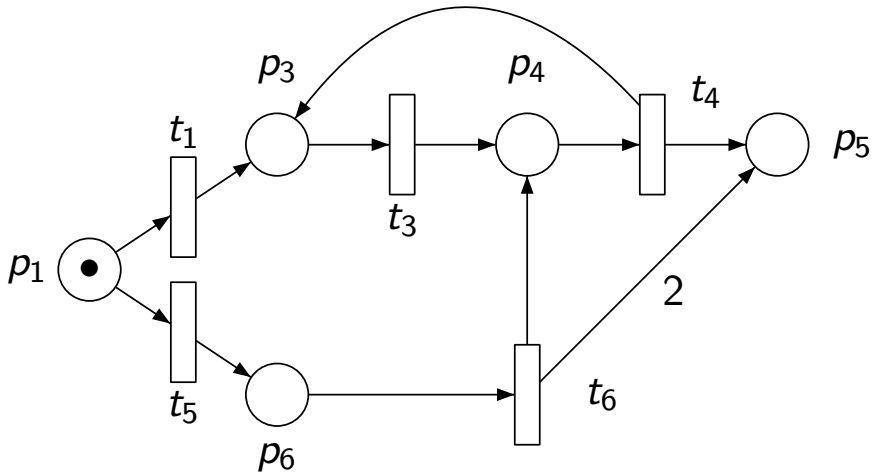
Karp & Miller Tree by example

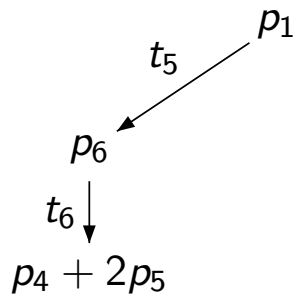




$$\mathrm{MCS} = \{p_1, p_6, p_3 + \omega p_5, p_4 + \omega p_5\}$$

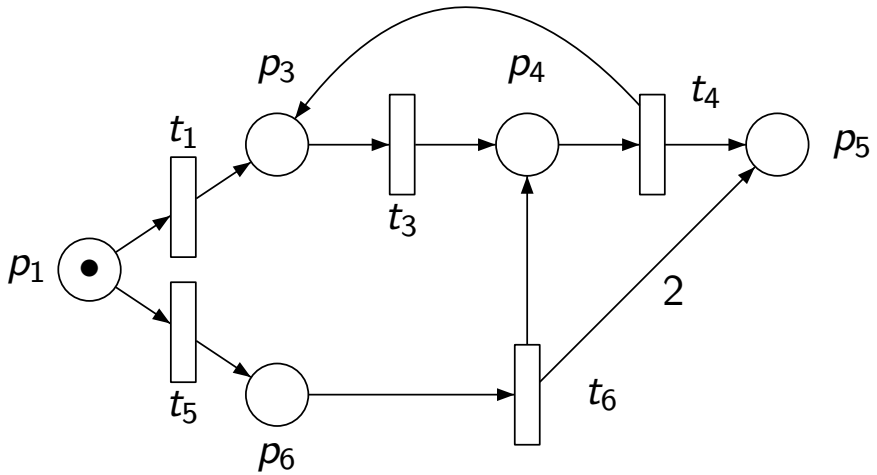
Karp & Miller Tree by example



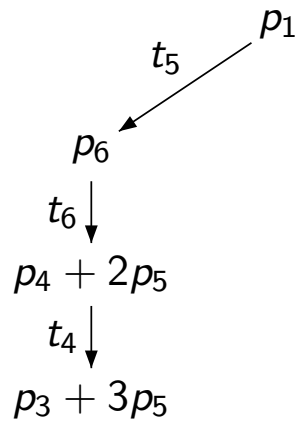


$$\mathrm{MCS} = \{p_1, p_6, p_3 + \omega p_5, p_4 + \omega p_5\}$$

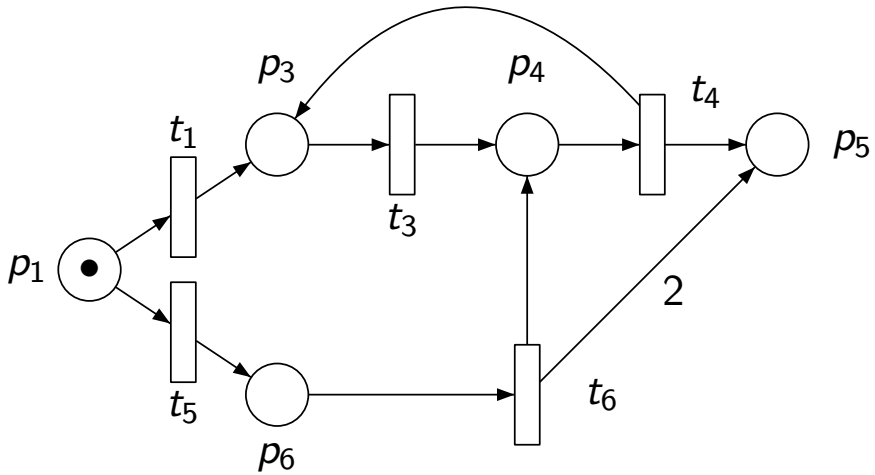
Karp & Miller Tree by example

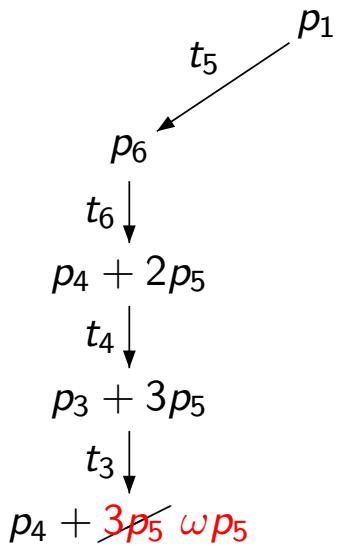


$\mathrm{MCS} = \{p_1, p_6, p_3 + \omega p_5, p_4 + \omega p_5\}$



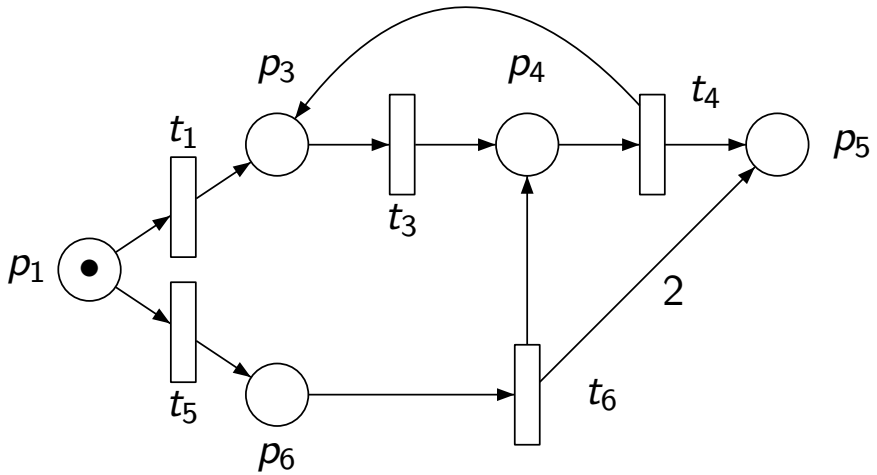
Karp & Miller Tree by example



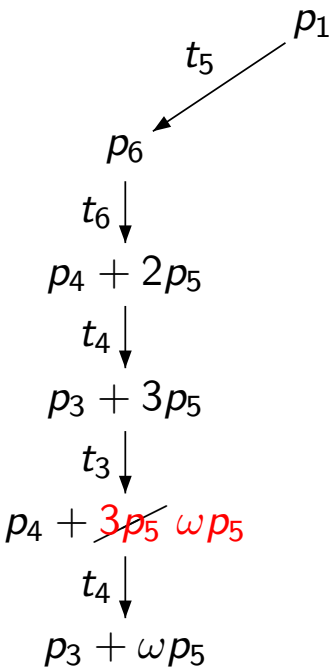


$$\mathrm{MCS} = \{p_1, p_6, p_3 + \omega p_5, p_4 + \omega p_5\}$$

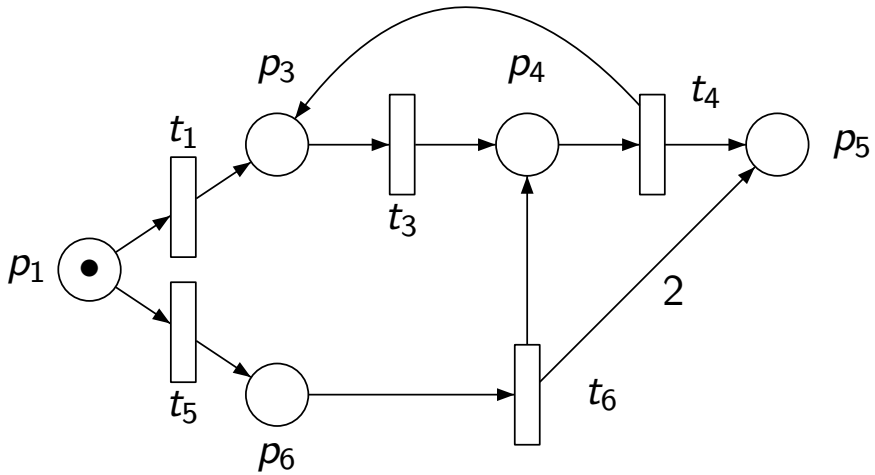
Karp & Miller Tree by example



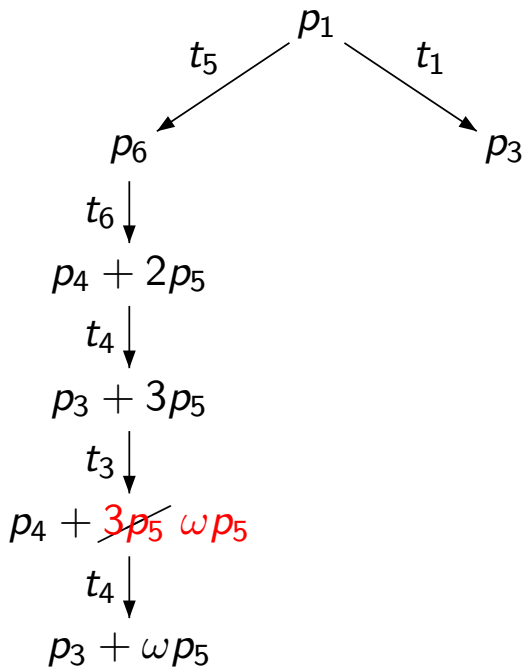
$$\mathrm{MCS} = \{p_1, p_6, p_3 + \omega p_5, p_4 + \omega p_5\}$$



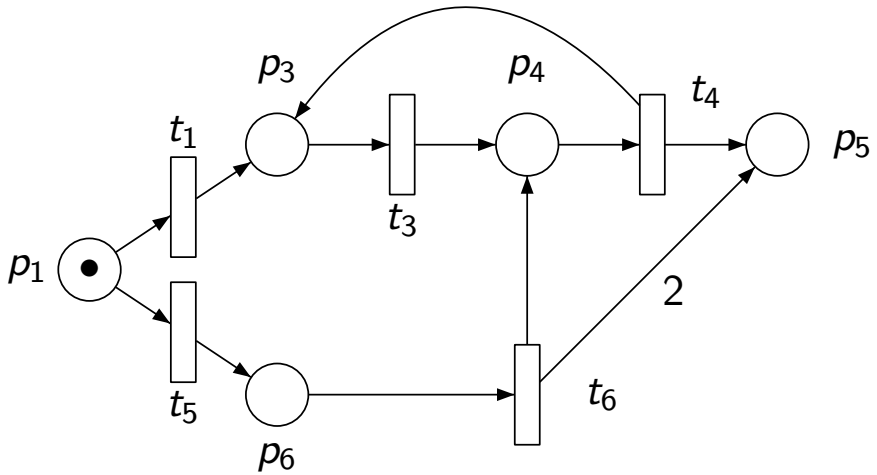
Karp & Miller Tree by example



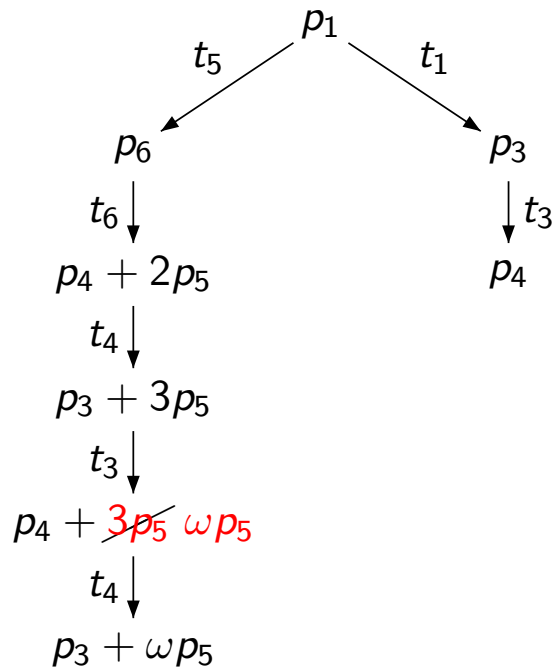
$$\mathrm{MCS} = \{p_1, p_6, p_3 + \omega p_5, p_4 + \omega p_5\}$$



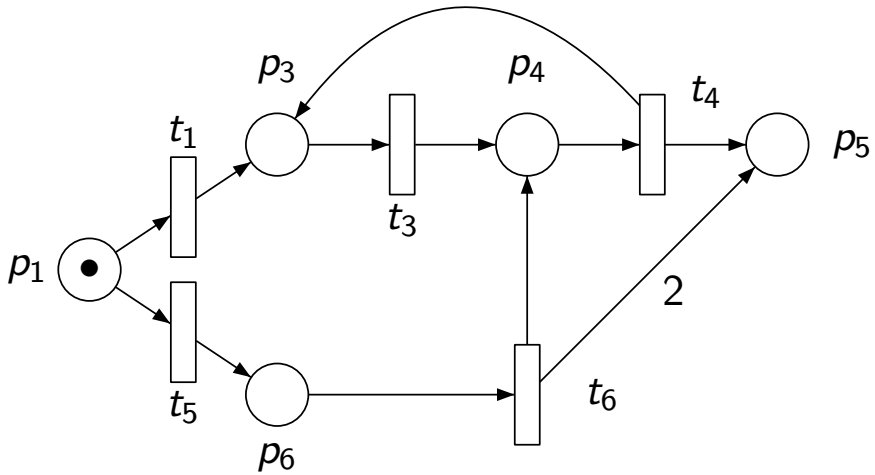
Karp & Miller Tree by example



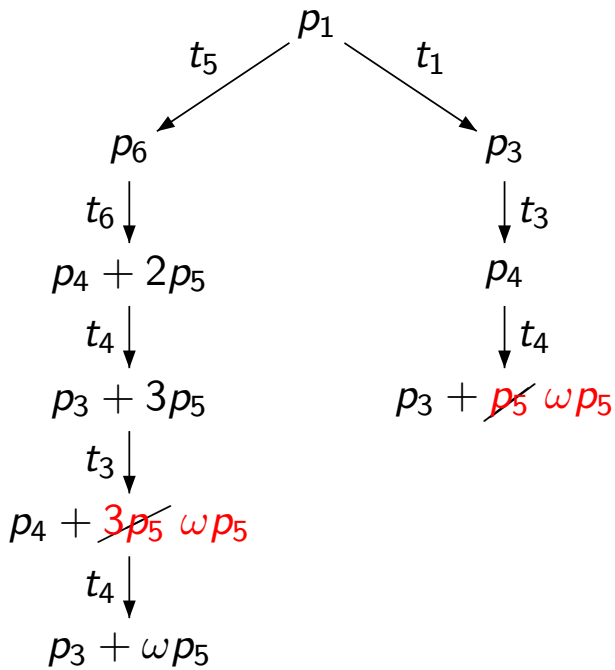
$$\mathrm{MCS} = \{p_1, p_6, p_3 + \omega p_5, p_4 + \omega p_5\}$$



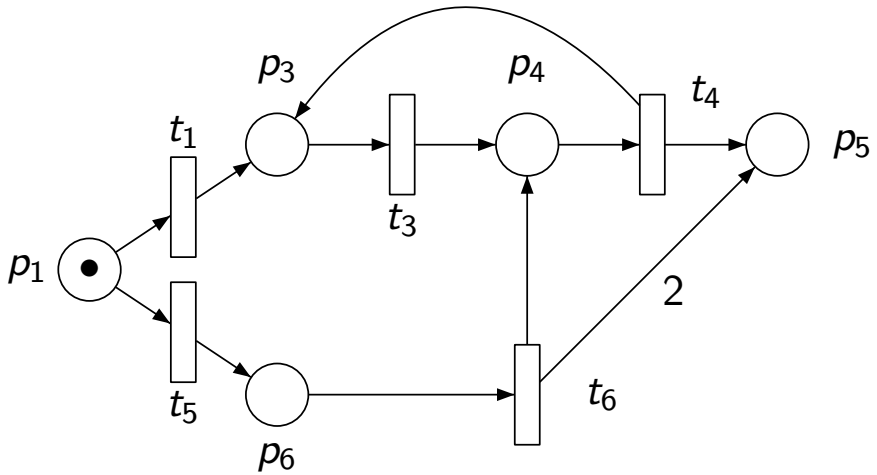
Karp & Miller Tree by example



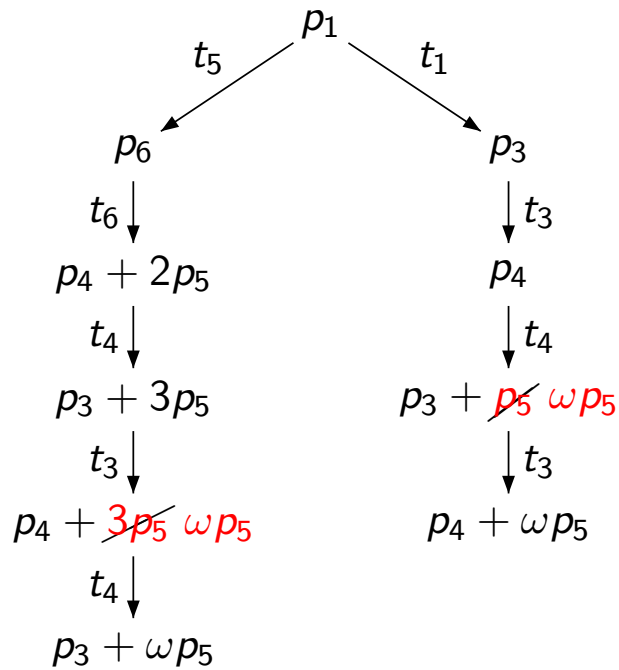
$$\mathrm{MCS} = \{p_1, p_6, p_3 + \omega p_5, p_4 + \omega p_5\}$$



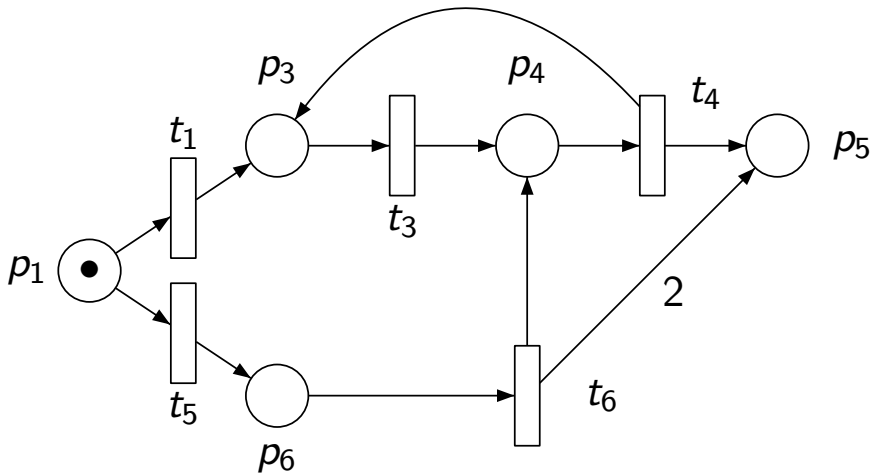
Karp & Miller Tree by example



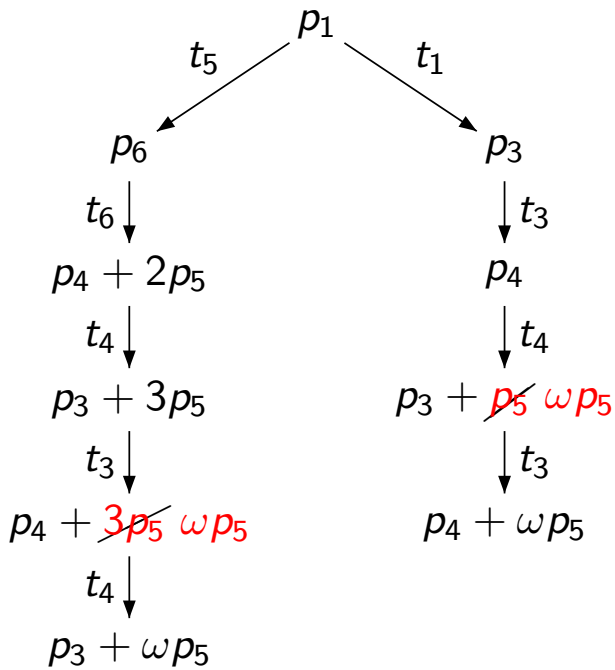
$$\mathrm{MCS} = \{p_1, p_6, p_3 + \omega p_5, p_4 + \omega p_5\}$$



Karp & Miller Tree by example



$\mathrm{MCS} = \{p_1, p_6, p_3 + \omega p_5, p_4 + \omega p_5\}$



K&M Algorithm performs redundant computations: all branches are developped independently.

Computation of the MCS

- Karp & Miller Tree (1969) : impractical
- Minimal Coverability Tree (MCT) [Finkel, 1991] : incomplete
  improved $\mathrm{K\&M}$ Algorithm with pruning
- MCT Patch [Luttge, 1995]: does not terminate
- CoverProc procedure [Geeraerts et al, 2005] : not $\mathrm{K\&M}$-based
- Our algorithm: $\mathrm{K\&M}$ Algorithm with pruning
- Piipponen and Valmari, 2016: new algorithm, very efficient implementation
- Others...

K&M Algorithm

Waiting List composed of pairs (n', t).

1. Pop an element (n', t) of the Waiting List.
2. Build a new node n such that $n' \xrightarrow{t} n$
3. If n is already covered go to 1.
4. Perform acceleration if possible.
5. Add n to the tree and (n, t) to Waiting List for all possible t.

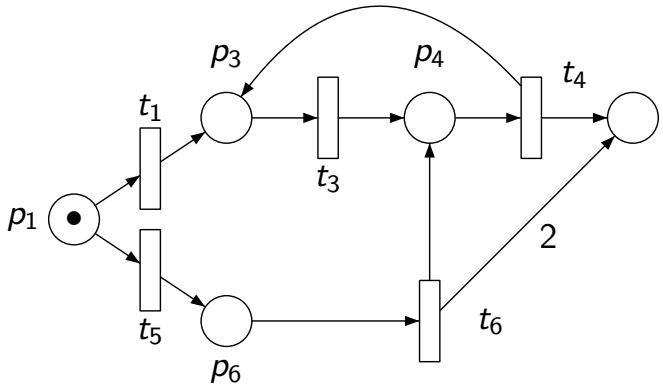
Our Algorithm

Waiting List composed of pairs (n', t).

1. Pop an element (n', t) of the Waiting List.
2. Build a new node n such that $n' \xrightarrow{t} n$
3. If n is already covered go to 1.
4. Perform acceleration if possible.
5. Pruning:

   for each node x covered by n such that $x \in \text{Act} \vee x \notin \text{Ancestor}(n)$, deactivate the subtree rooted in x
6. Add n to the tree and (n, t) to Waiting List for all possible t.

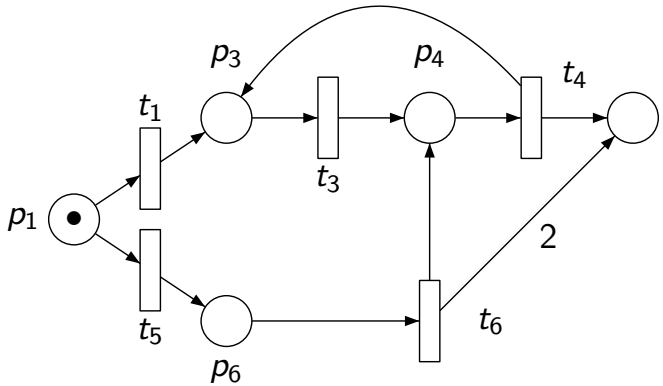
Monotone Pruning Algorithm by example



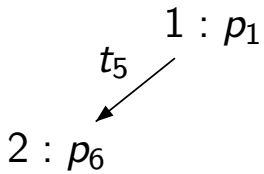
$\mathrm{MCS} = \{p_1, p_6, p_3 + \omega p_5, p_4 + \omega p_5\}$

$1 : p_1$

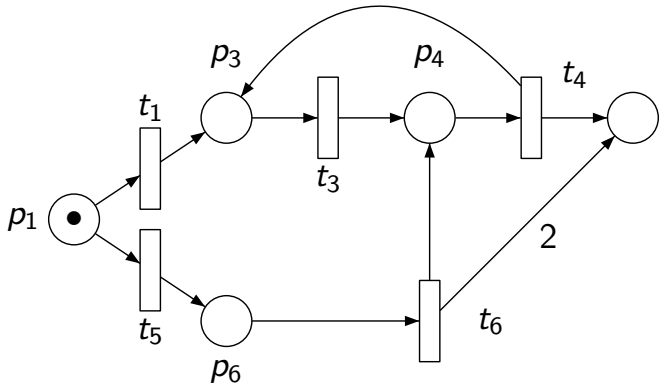
Monotone Pruning Algorithm by example



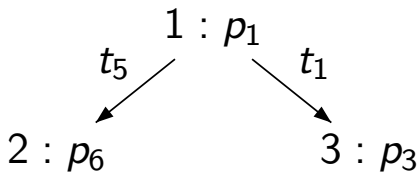
$$\mathrm{MCS} = \{p_1, p_6, p_3 + \omega p_5, p_4 + \omega p_5\}$$



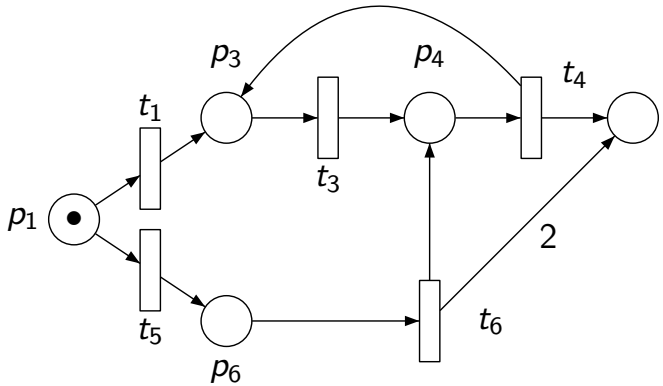
Monotone Pruning Algorithm by example



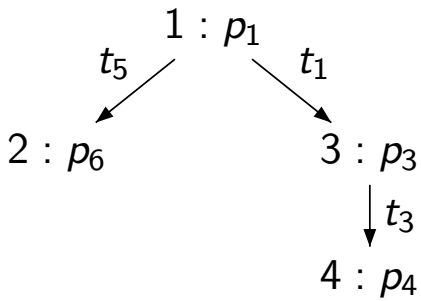
$$\mathrm{MCS} = \{p_1, p_6, p_3 + \omega p_5, p_4 + \omega p_5\}$$



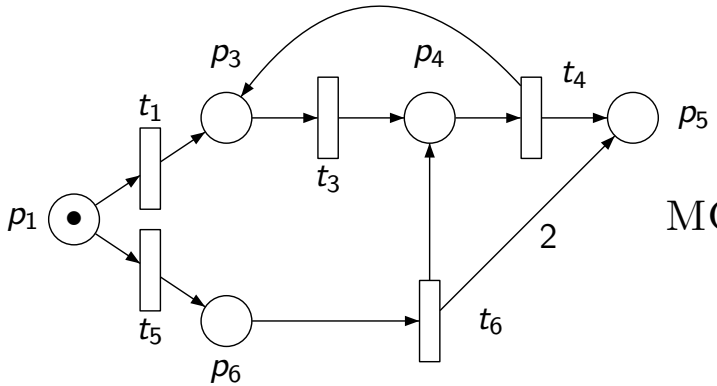
Monotone Pruning Algorithm by example



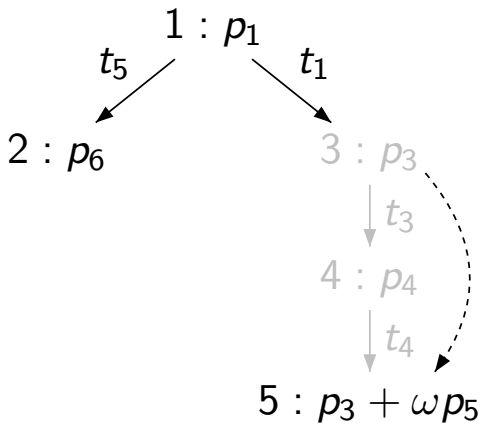
$$\mathrm{MCS} = \{p_1, p_6, p_3 + \omega p_5, p_4 + \omega p_5\}$$



Monotone Pruning Algorithm by example

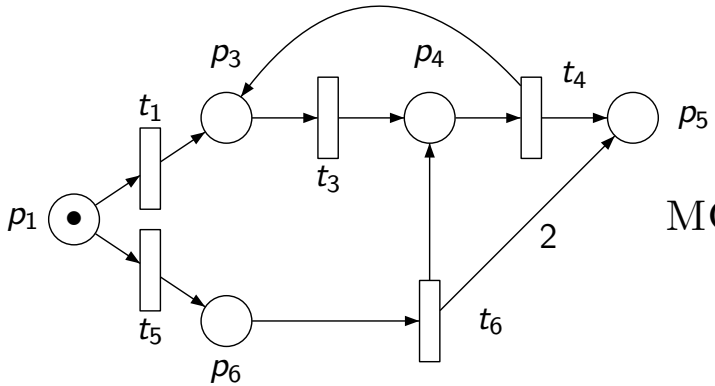


$$\mathrm{MCS} = \{p_1, p_6, p_3 + \omega p_5, p_4 + \omega p_5\}$$

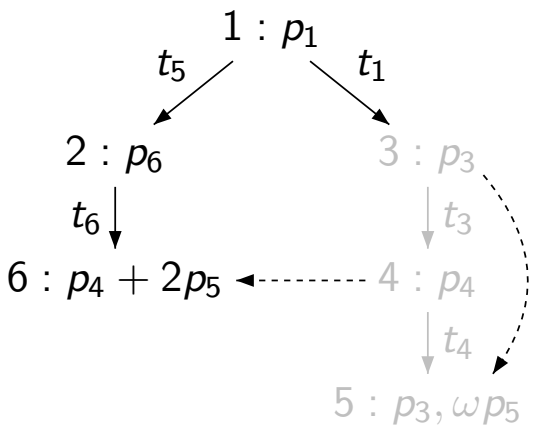


Step 5: Acceleration of node 5 w.r.t. node 3 deactivates nodes 3, 4

Monotone Pruning Algorithm by example



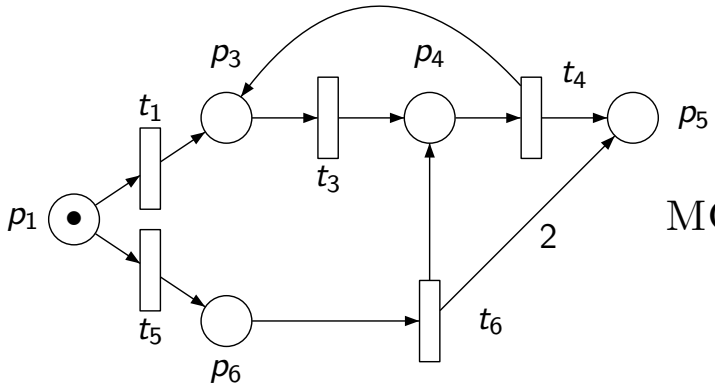
$$\mathrm{MCS} = \{p_1, p_6, p_3 + \omega p_5, p_4 + \omega p_5\}$$



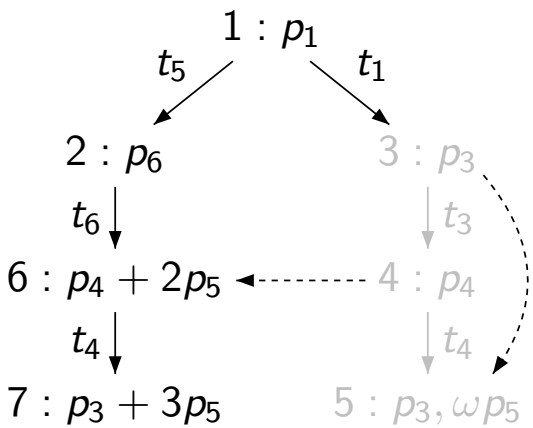
Step 5: Acceleration of node 5 w.r.t. node 3 deactivates nodes 3, 4

Step 6: Node 6 covers node 4 and thus deactivates node 5

Monotone Pruning Algorithm by example



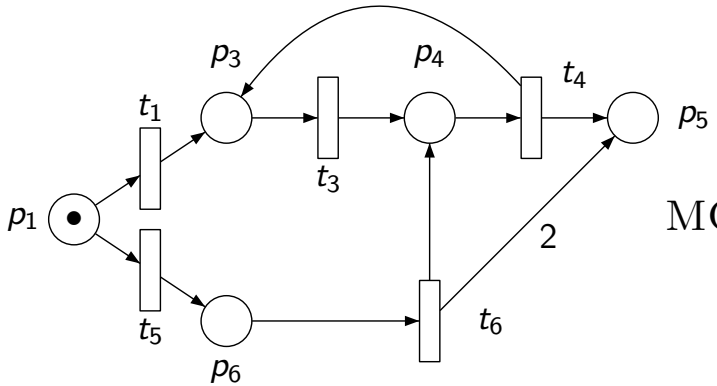
$$\mathrm{MCS} = \{p_1, p_6, p_3 + \omega p_5, p_4 + \omega p_5\}$$



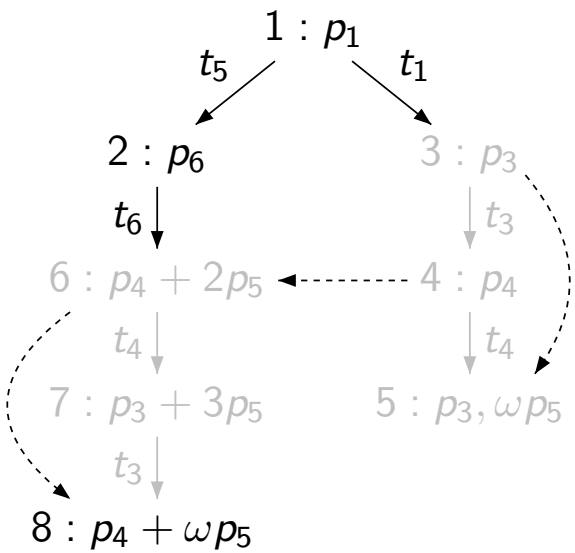
Step 5: Acceleration of node 5 w.r.t. node 3 deactivates nodes 3, 4

Step 6: Node 6 covers node 4 and thus deactivates node 5

Monotone Pruning Algorithm by example



$$\mathrm{MCS} = \{p_1, p_6, p_3 + \omega p_5, p_4 + \omega p_5\}$$

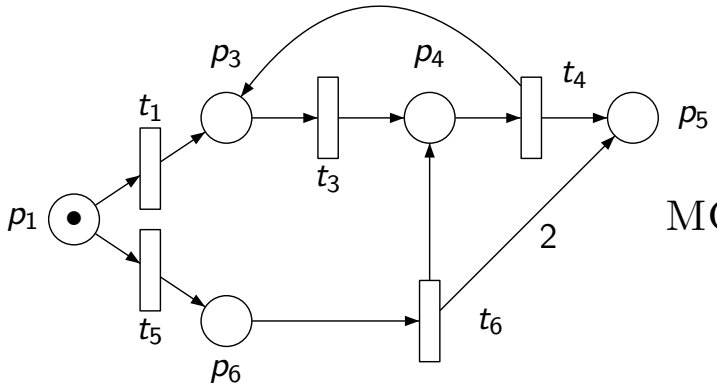


Step 5: Acceleration of node 5 w.r.t. node 3 deactivates nodes 3, 4

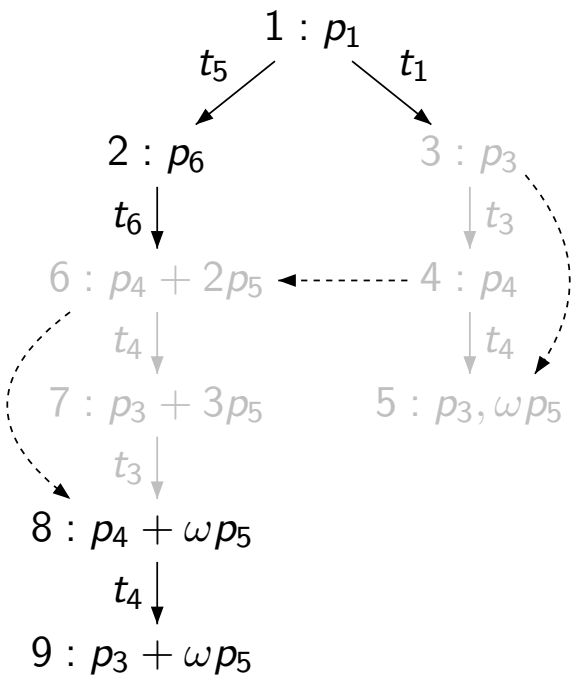
Step 6: Node 6 covers node 4 and thus deactivates node 5

Step 8: Acceleration of node 8 w.r.t. node 6 deactivates nodes 6, 8

Monotone Pruning Algorithm by example



$$\mathrm{MCS} = \{p_1, p_6, p_3 + \omega p_5, p_4 + \omega p_5\}$$

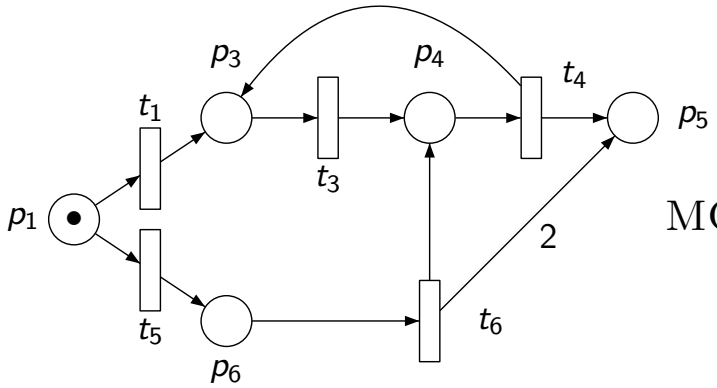


Step 5: Acceleration of node 5 w.r.t. node 3 deactivates nodes 3, 4

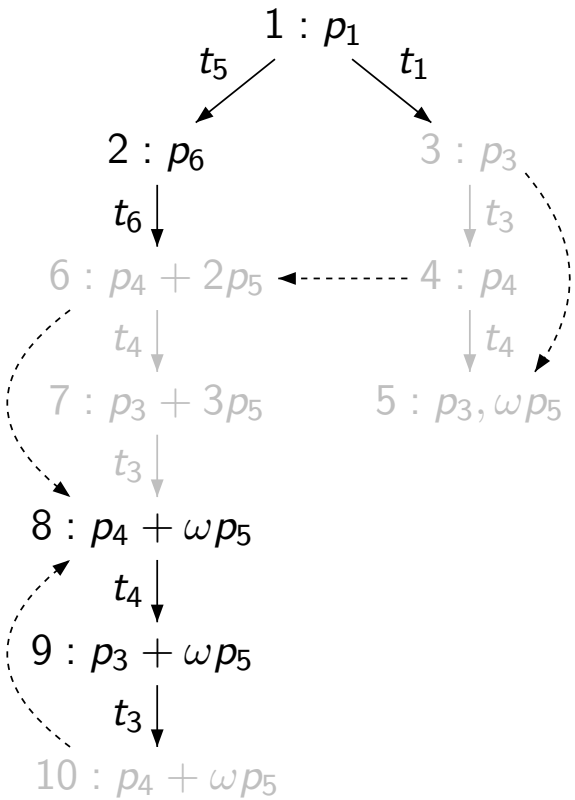
Step 6: Node 6 covers node 4 and thus deactivates node 5

Step 8: Acceleration of node 8 w.r.t. node 6 deactivates nodes 6, 8

Monotone Pruning Algorithm by example



$$\mathrm{MCS} = \{p_1, p_6, p_3 + \omega p_5, p_4 + \omega p_5\}$$



Step 5: Acceleration of node 5 w.r.t. node 3 deactivates nodes 3, 4

Step 6: Node 6 covers node 4 and thus deactivates node 5

Step 8: Acceleration of node 8 w.r.t. node 6 deactivates nodes 6, 8

Step 10: Node 10 is immediately declared as inactive as it is covered by node 8

In this paper

While our algorithm is a simple, elegant optimisation of the $\mathrm{K\&M}$ algorithm, the 2011 proof is horrible, 10 pages long.
In this paper:

- shorter elegant proof (less than 2 pages)
- the acceleration function is a parameter: you can plug any reasonable function, the proof still holds.
- better suited for generalisation to other WSTS.

Proof of Monotone Pruning Algorithm

Theorem

MP Algorithm terminates and is correct.

Proof of Monotone Pruning Algorithm

Theorem

MP Algorithm terminates and is correct.

Termination: as for $\mathrm{K\&M}$ Algorithm, relies on the fact that $\leq$ is a well quasi order on $(\mathbb{N} \cup \{\omega\})^P$

Proof of Monotone Pruning Algorithm

Theorem

MP Algorithm terminates and is correct.

Termination: as for $\mathrm{K\&M}$ Algorithm, relies on the fact that $\leq$ is a well quasi order on $(\mathbb{N} \cup \{\omega\})^P$

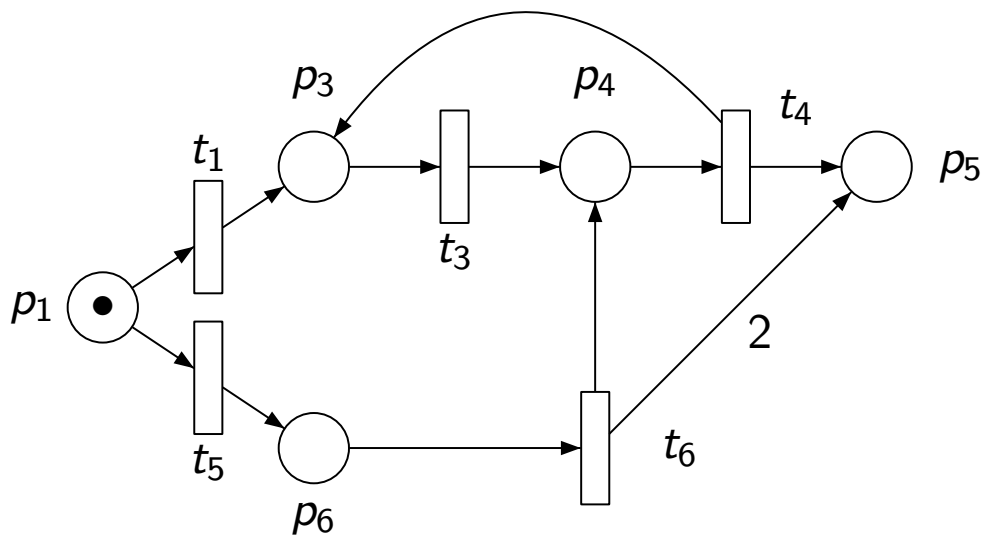
Correction:

- Soundness: each marking built by MP algorithm can be part of a coverability set
  ➜ easily follows similarly as $\mathrm{K\&M}$ Algorithm
- Completeness: prove that every reachable marking is covered by an active node
  ➜ this is the main challenge

Proof of completeness

We want to prove, for each sequence $m_0 \xrightarrow{\varrho} m$, that there exists an active node built by MP that covers m.

Consider for example the sequence $\rho = t_1\, t_3\, t_4\, t_3$:
$p_1 \xrightarrow{t_1} p_3 \xrightarrow{t_3} p_4 \xrightarrow{t_4} p_3 + p_5 \xrightarrow{t_3} p_4 + p_5$



Proof of completeness

At each step, the algorithm satisfies the following invariant.

For any reachable marking m, either m is covered by an active node or there exists an active node x and a sequence of transition ρ such that:

- $x \xrightarrow{\rho} \cdot$ covers the marking
- (x, t), where t is the first transition of ρ, is in the Waiting List
- every intermediate marking when firing ρ from x is not covered by any active node

In other words: if it is not covered now, it will be covered later.

Experiments

Test				K&M		MP		CoverProc
name	P	T	MCS	nodes	time (s)	nodes	time (s)	time (s)
BasicME	5	4	3	5	< 0.01	5	< 0.01	0.12
Kanban	16	16	1	72226	9.1	114	< 0.01	0.19
Lamport	11	9	14	83	0.02	24	< 0.01	0.17
Manufacturing	13	6	1	81	0.01	30	< 0.01	0.14
Peterson	14	12	20	609	0.2	35	0.02	0.25
Read-write	13	9	41	11139	6.33	76	.06	1.75
Mesh2x2	32	32	256	x	x	6241	18.1	330
Multipool	18	21	220	x	x	2004	4.9	365
pncsacover	31	36	80	x	x	1604	1.6	113
csm	14	13	16	x	x	102	.03	0.34
fms	22	20	24	x	x	809	0.28	2.1

- Drastic reduction of the number of states over K&M Algorithm
- A lot faster than both K&M and CoverProc Algorithms
- After running K&M Algorithm, one has to extract the MCS from the coverability set computed by K&M.

Contributions

Simpler and more general proof for the MP algorithm:

- Acceleration function as a parameter
- May allow extension to other WSTS

The MP algorithm is nice :

- Simple modification of the $\mathrm{K\&M}$ algorithm
- Any strategy of exploration is correct
- Any reasonable acceleration function can be plugged in
- Drastic improvement over $\mathrm{K\&M}$