3. Formal Features of Petri Nets for Static Verification of Dynamic Behavior

Lecturer: Dr. Sebastian Götz

Prof. Dr. U. Aßmann
Technische Universität Dresden
Institut für Software- und Multimediatechnik
Softwaretechnologie
http://st.inf.tu-dresden.de

24.10.2018

Content

- Behavioral properties
    - Reachability
    - Liveness
    - Boundedness
- Liveness checking

Obligatory Readings

- T. Murata. Petri Nets: properties, analysis, applications. IEEE volume 77, No 4, 1989.
- Ghezzi Chapter 5
- J. B. Jörgensen. Colored Petri Nets in UML-based Software Development – Designing Middleware for Pervasive Healthcare. www.pervasive.dk/publications/files/CPN02.pdf

Literature

- K. Jensen, Colored Petri Nets. Vol. I-III. Springer, 1992-96. Landmark book series on CPN.
- W. Reisig. Elements of Distributed Algorithms – Modelling and Analysis with Petri Nets. Springer. 1998.
- W. Reisig, G. Rozenberg: Lectures on Petri Nets I+II, Lecture Notes in Computer Science, 1491+1492, Springer.
- J. Peterson. Petri Nets. ACM Computing Surveys, Vol 9, No 3, Sept 1977
- H. Balzert. Lehrbuch der Softwaretechnik. Verlag Spektrum der Wissenschaft. Heidelberg, Germany.

Goals

Understand the isomorphism between finite automata (statecharts) and bounded Petri nets

Understand why matrix algebra solves

- deadlock and liveness questions
- protocol questions

3.1 Behavioral Properties of PN

Reachability of Markings

If t is *enabled* in M, we write M[t)

A marking M_n is said to be *reachable* from a marking M_0 if there exists a firing sequence s that transforms M_0 to M_n.

- We write this M_0 [s) M_n

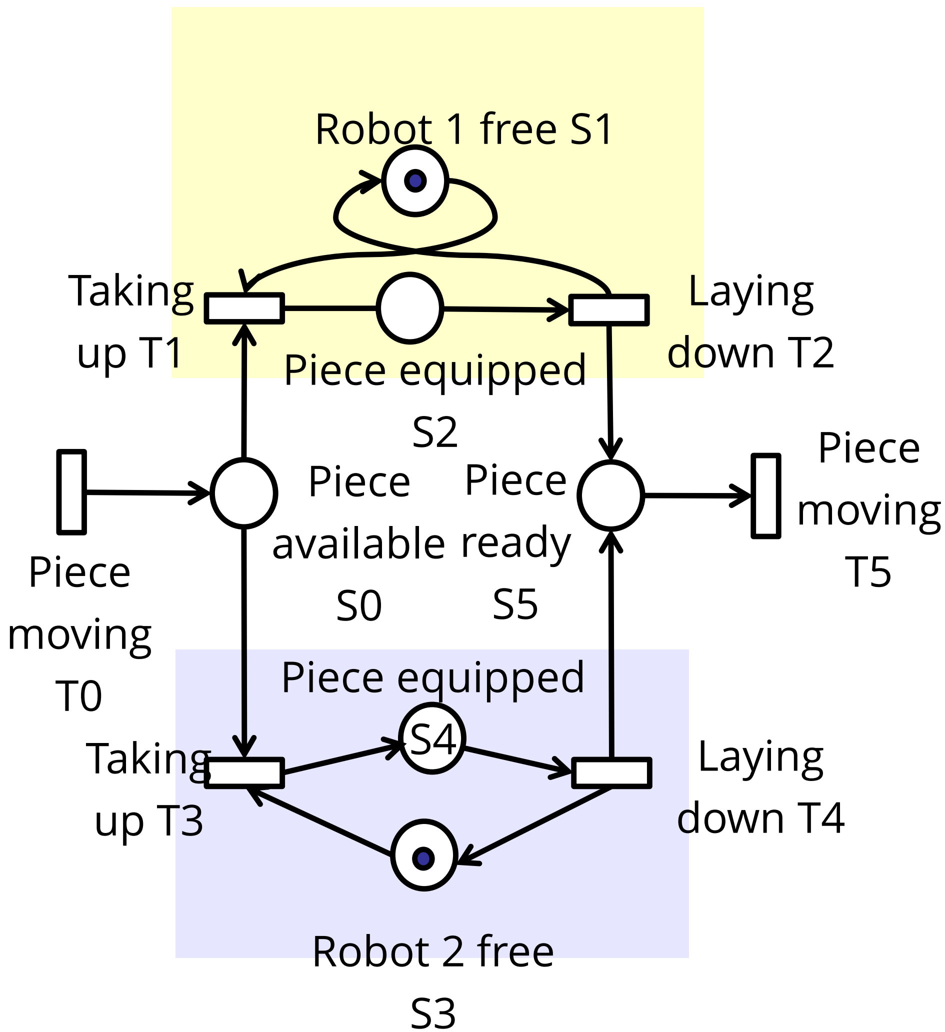
A *firing sequence* is denoted by a sequence of transitions
s = M_0 [t1) M_1 [t2) M_2 ... [tn) M_n or simply
s = t1 t2 t3 ... tn.

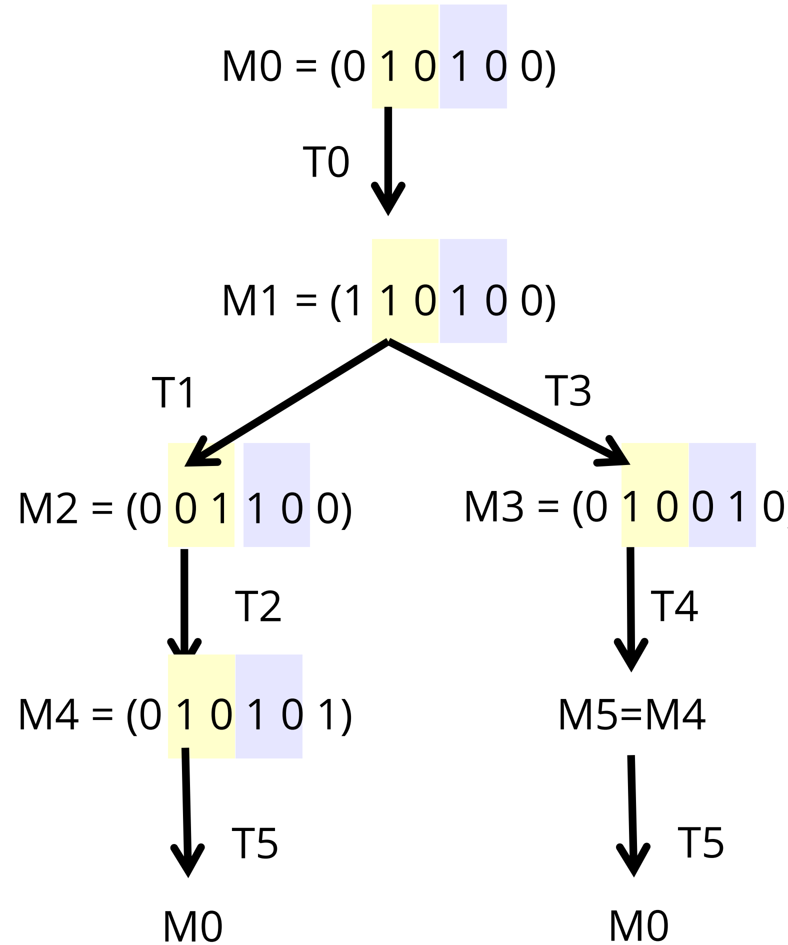
The set of all possible markings reachable from M_0 is denoted R(M_0).

- R(M_0) is spanning up a state automaton, the *state space*, *reachability graph*, or *occurrence graph*
- Every marking of the PN is a state in the reachability graph

The set *of all possible firing sequences* in a net (N,M_0) is denoted L(M_0). This is the language of the automaton R(M_0).

Reachability Tree of the 2 Robots

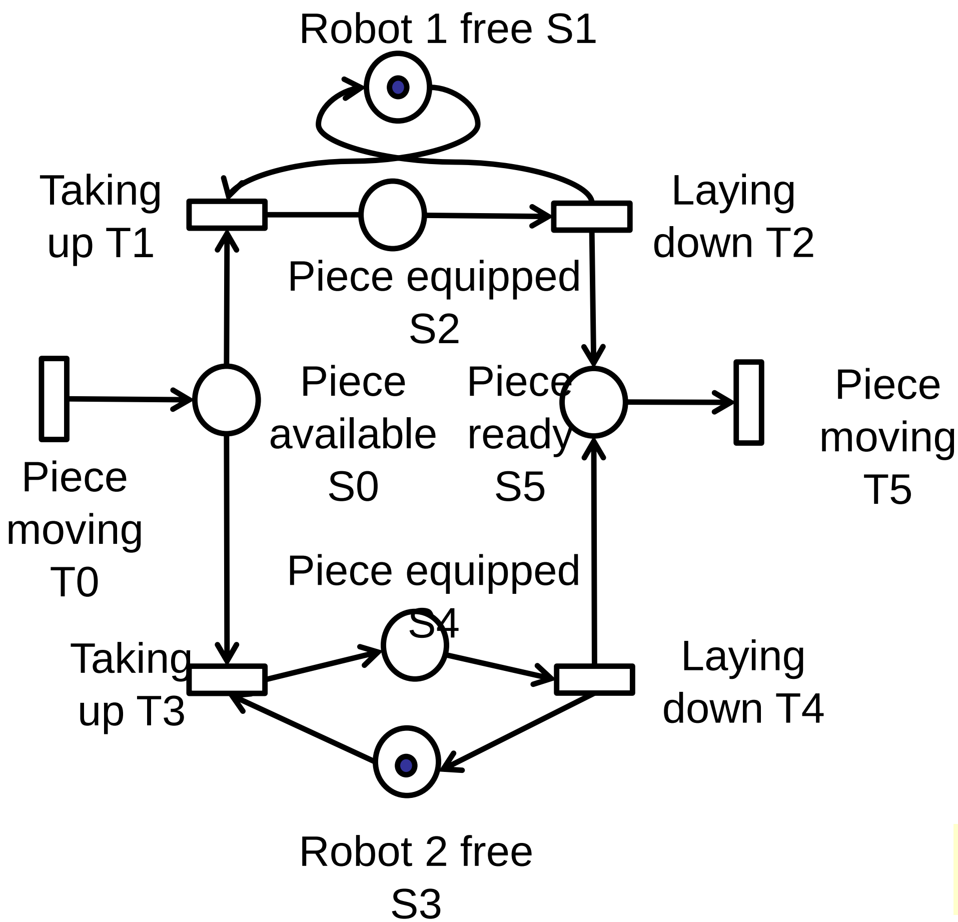


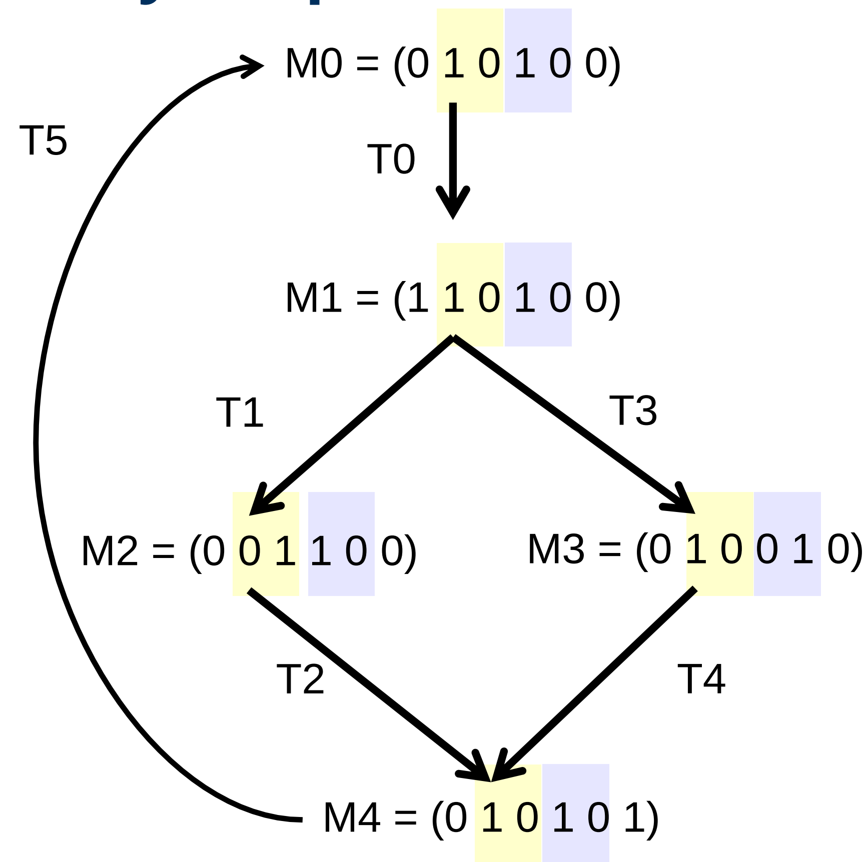


Upper part of net (S1, S2)

Lower part of net (S3, S4)

Folding the Tree to the Reachability Graph (Common Subtree Elimination)

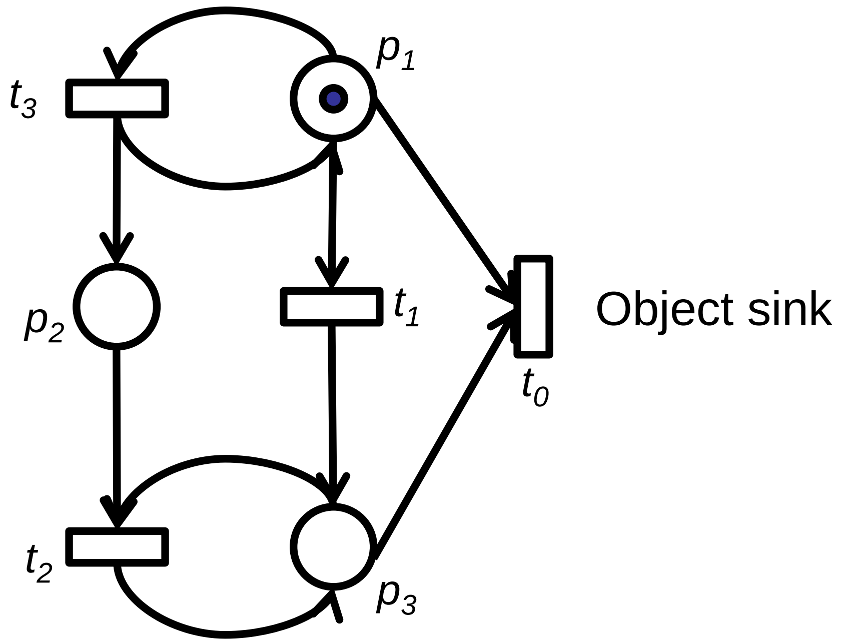


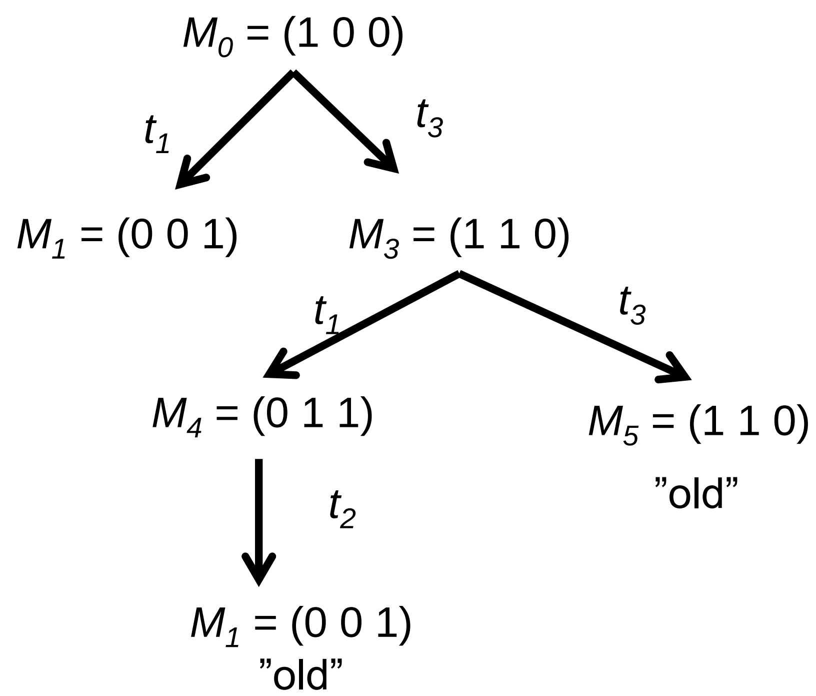


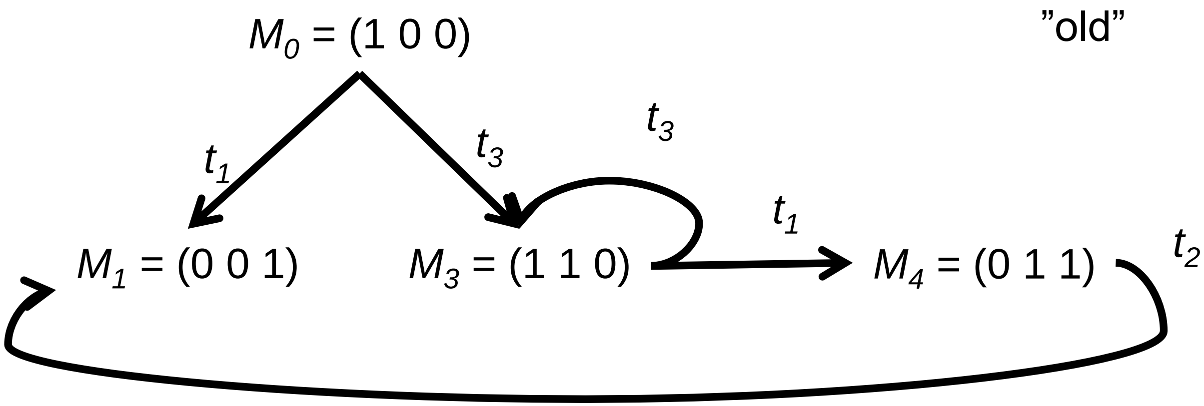
Upper part of net (S1, S2)

Lower part of net (S3, S4)

Example: The Reachability Tree and Graph







3.2 Boundedness

Boundedness and Safety

A PN (N,M_0) is *k-bounded* or simply *bounded* if every place is size-restricted by k

- $M(p) \leq k$ for every place p and every marking M in $R(M_0)$.

A PN is *safe* if it is 1-bounded.

Bounded nets can have only finitely many states, since the number of tokens and token combinations is limited

- The reachability graph of bounded nets is finite, it corresponds to a finite automaton (which is much larger)
- The PN is much more compact, it *abbreviates* the automaton

Applications of Boundedness

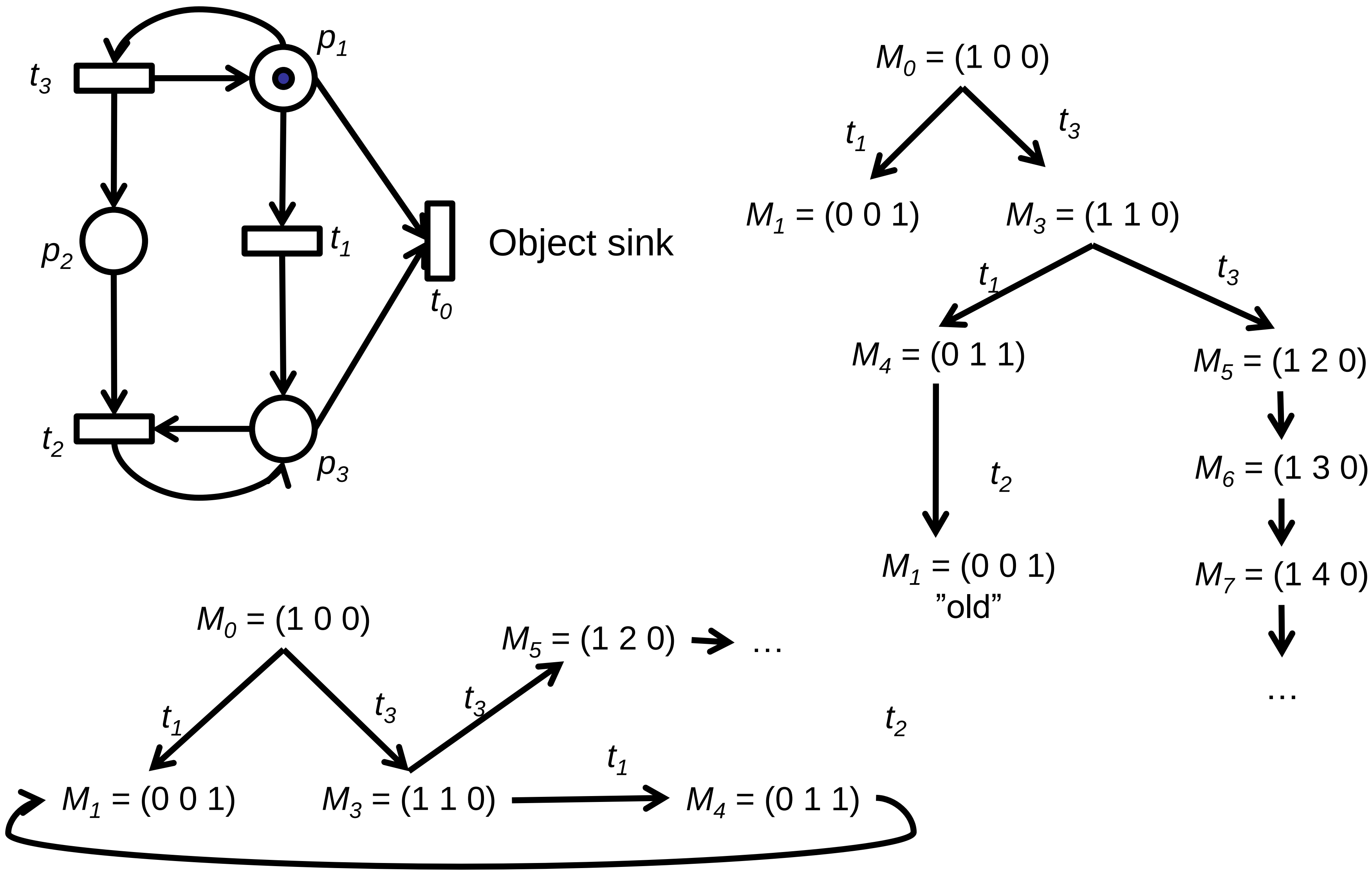
The markings of a state can express the number of available resources

- Operating Systems: number of memory blocks, number of open devices, number of open files, number of processes
- Workflows: number of actors, number of workpieces that flow

Boundedness can be used to prove that a system only consumes k resources

- Important for systems with resource constraints

Example: Unbounded net



3.3 Liveness

3.3 Liveness of Nets

Liveness is closely related to the complete absence of deadlocks in operating systems.

A PN (N,M_0) is **live** if, no matter what marking has been reached from M_0,

- all transitions are live
- i.e., it is possible to fire any transition of the net by progressing through some further firing sequence.

Liveness of Transitions

Liveness expresses whether a transition stays active or not

A transition t is called:

Dead (L0-live) if t can never be fired in any firing sequence in *$R(M_0)$*. (not fireable)

L1-live (potentially fireable) if t can be at least fired once in some firing sequence in *$R(M_0)$*. (firing at least once from the start configuration)

L2-live (k-fireable) if t can be fired at least *k* times in some firing sequence in *$R(M_0)$*, given a positive integer *k*. (firing k times from the start configuration)

L3-live (inf-fireable) if t appears infinitely often in *some* firing sequence in *$R(M_0)$*. (firing infinitely often from the start configuration)

L4-live if t is L1-live for every (even unreachable) marking *M* in *$R(M_0)$*.

Liveness of Markings and Nets

A marking is *dead* if non of its transitions are enabled.

A marking is *live* if no reachable marking is dead (equivalent: all transitions are live)

A net is *live* if M_0 is live (every t is always fire-able again from every reachable marking of M_0)

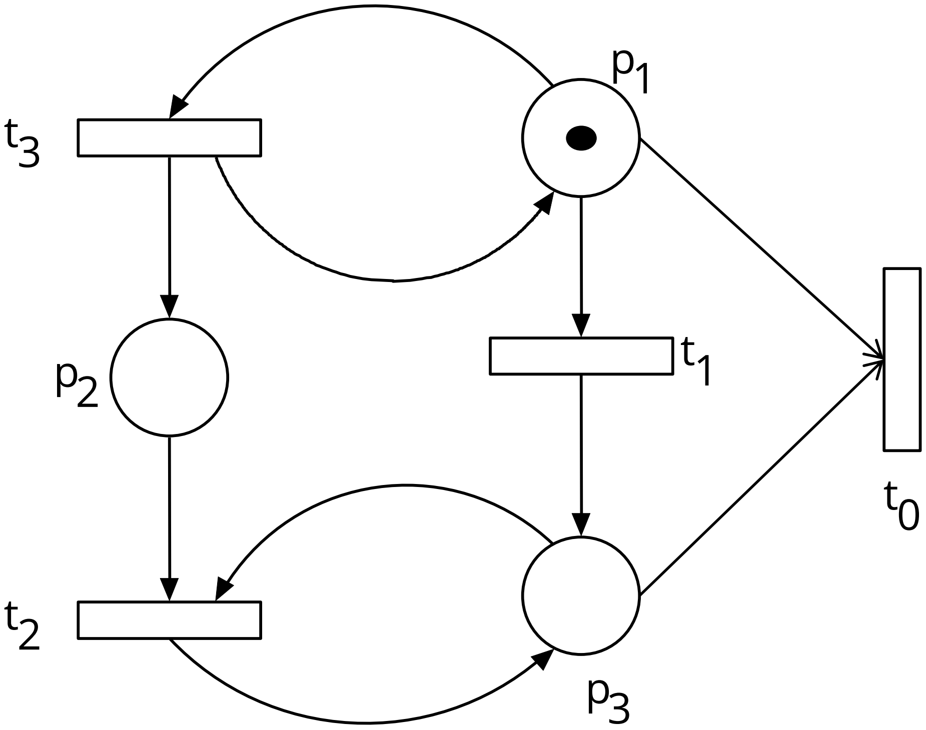
Example: Liveness

t_1 L1-live (fireable only once, bridge)

Hence, t_3 is L3-live (on a cycle), but not L4-live, since it cannot be activated anymore once t_1 is crossed

t_0 is L0-live (dead, since t_1 is bridge and either p_1 or p_3 is filled)

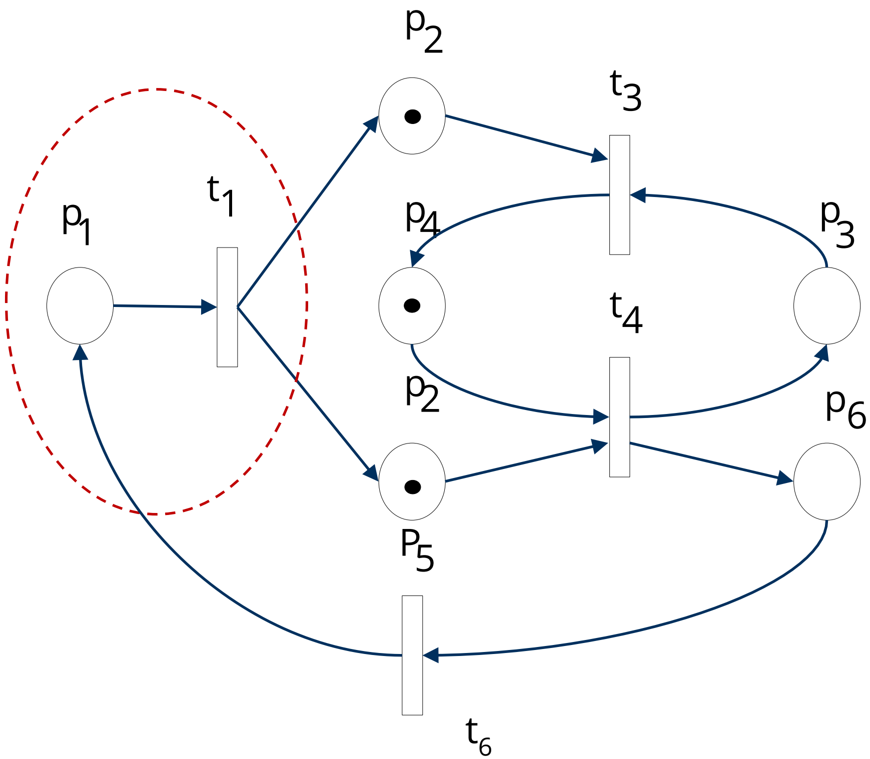
t_2 L2-live (fireable when t_1 is crossed)



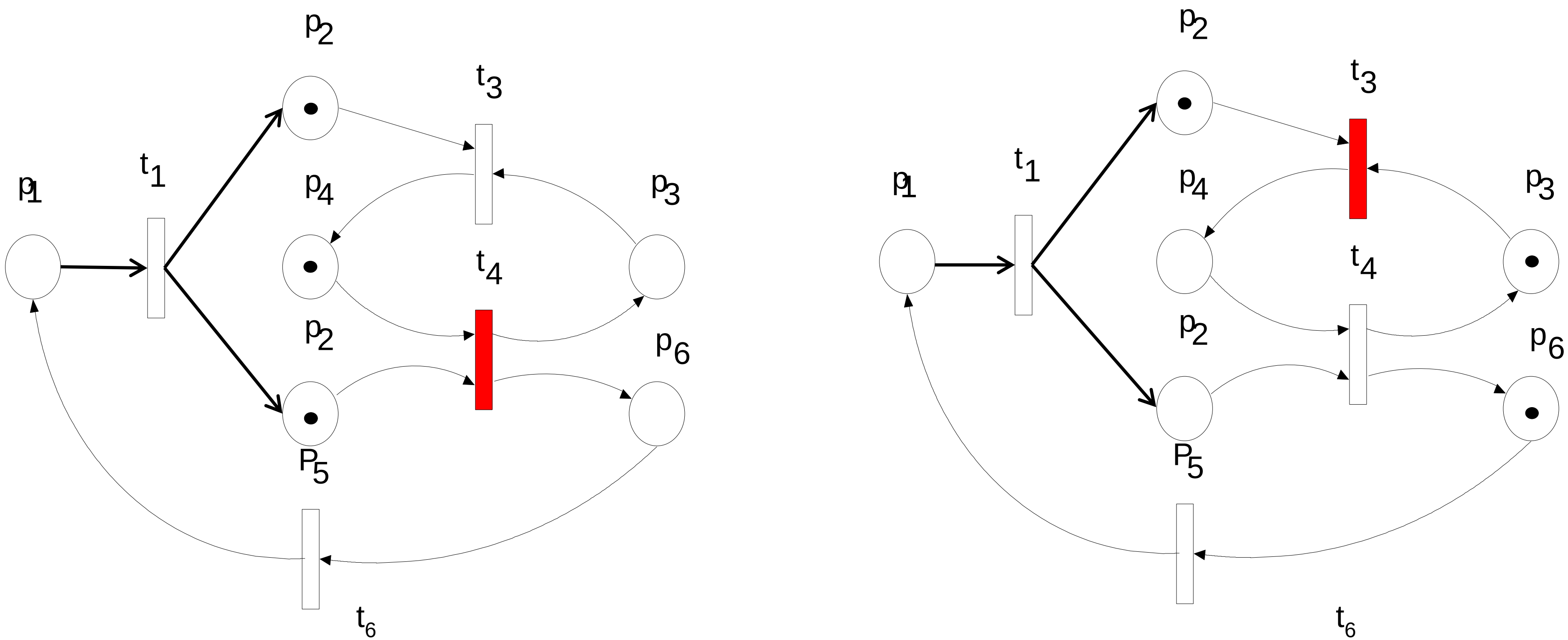
Example: Liveness

A safe, live PN. M0 can be reproduced again, e.g.,
t_1 t_4 t_3 t_6 reproduces a filled p_1 and p_2

p1, t1 form a fork

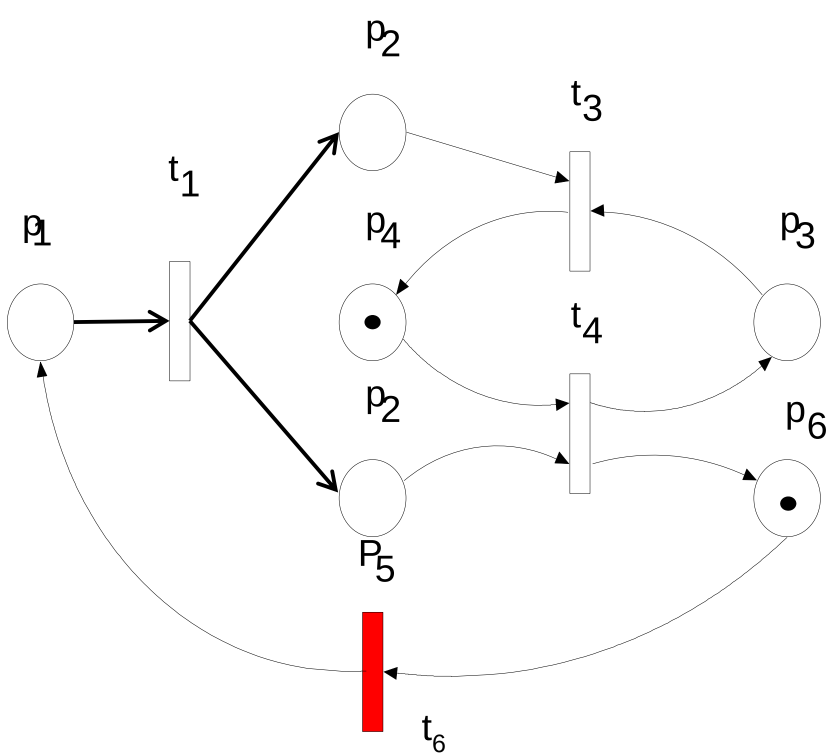


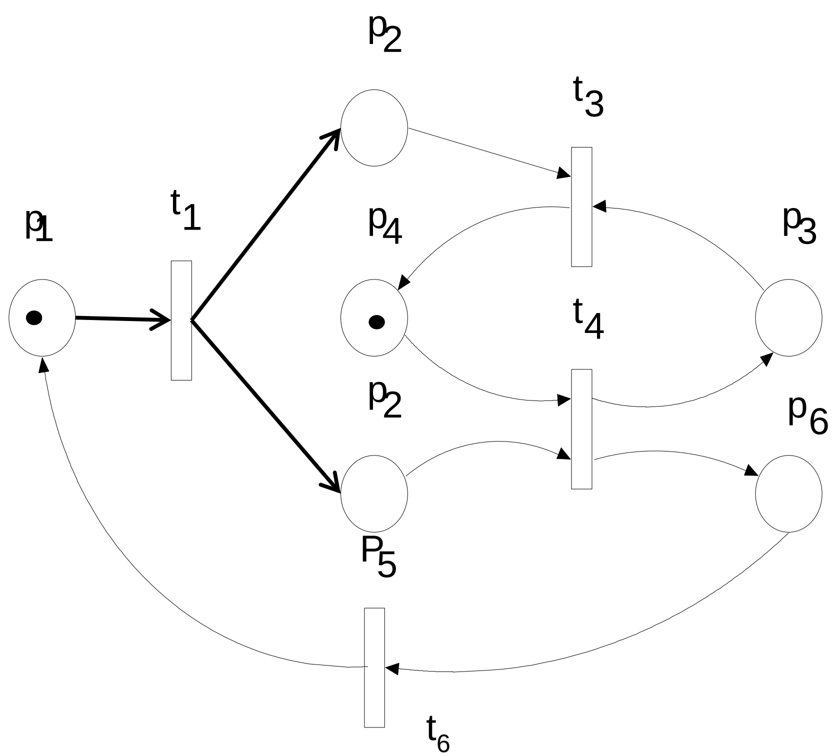
Example: Liveness



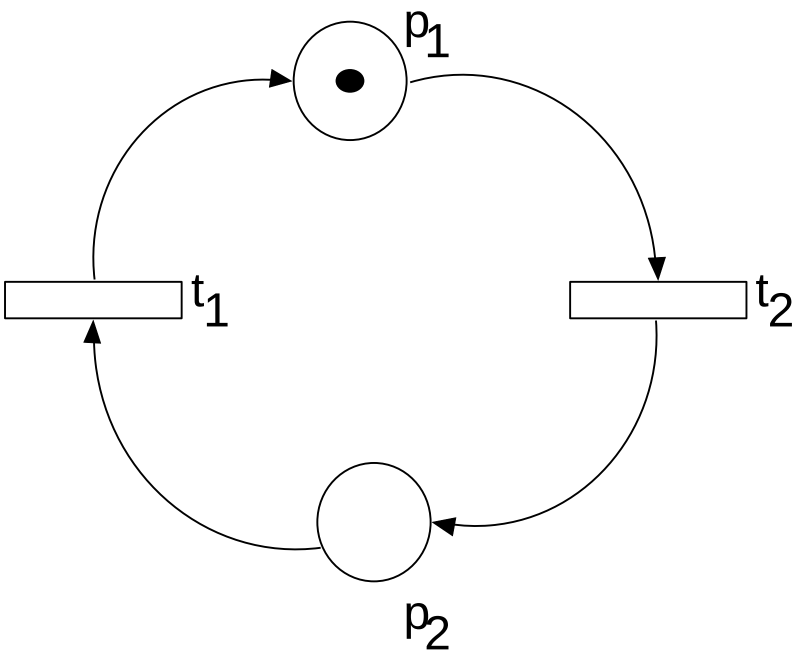
p2 is a synchronization dependency; process p5 can run earlier, p2 has to wait.
Note: the content of p2 must be reproduced again
Net is unbounded, due to the reproduction facilities of t6

Example: Liveness





Well, everything: Safe, Live



3.4 Liveness of PN with the Incidence Matrix and T-Invariants

The Relation to Matrices

Bounded Petri nets have a direct mapping to matrices

Via matrix algebra, an algorithm can be derived which tests the liveness of a PN

That is the basis of the check tools

Incidence Matrix

The *incidence matrix (transition matrix, switching matrix)* represents a PN in matrix form

- Markings are represented as vectors
- Firing sequences are represented as vectors (firing vectors)
- Matrix-vector multiplication shows the influence of the PN on a marking
- Multiplying incidence matrix with a firing vector gives new marking

A PN with n transitions and m places, the incidence matrix
$A = [a_{ij}]$ is a n x m matrix of integers

- rows: transitions
- columns: places

Weights and the Incidence Matrix

Weights on edges become entries in the matrix as follows

An entry of the incidence matrix shows the effect a_{ij} on a place I by adding the incoming tokens and subtracting the outgoing tokens from transition j:

$a_{ij} = w(t_j, s_i) - w(s_i, t_j)$:

- $w(t_j, s_i)$, the weight of the arc from transition j to output place i (*incoming weight to place*)
- $w(s_i, t_j)$, the weight of the arc from input place i to (outgoing) transition j. (*outgoing weight from place*)

Transition i is enabled at a marking M iff
$a_{ij} \leq M(j)$, j = 1, 2, ... , m.

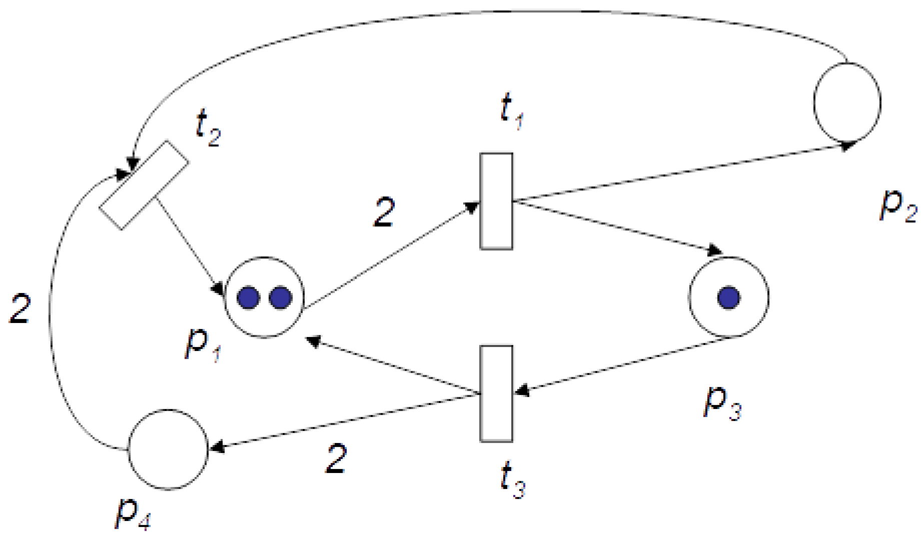
After firing, a marking
$m' = m+(w(t_j, s_i) - w(s_i, t_j))$ results.

Example: Computing the Incidence Matrix

p1 looses 2 to t1; p1 gets 1 from t2; p1 gets 1 from t3

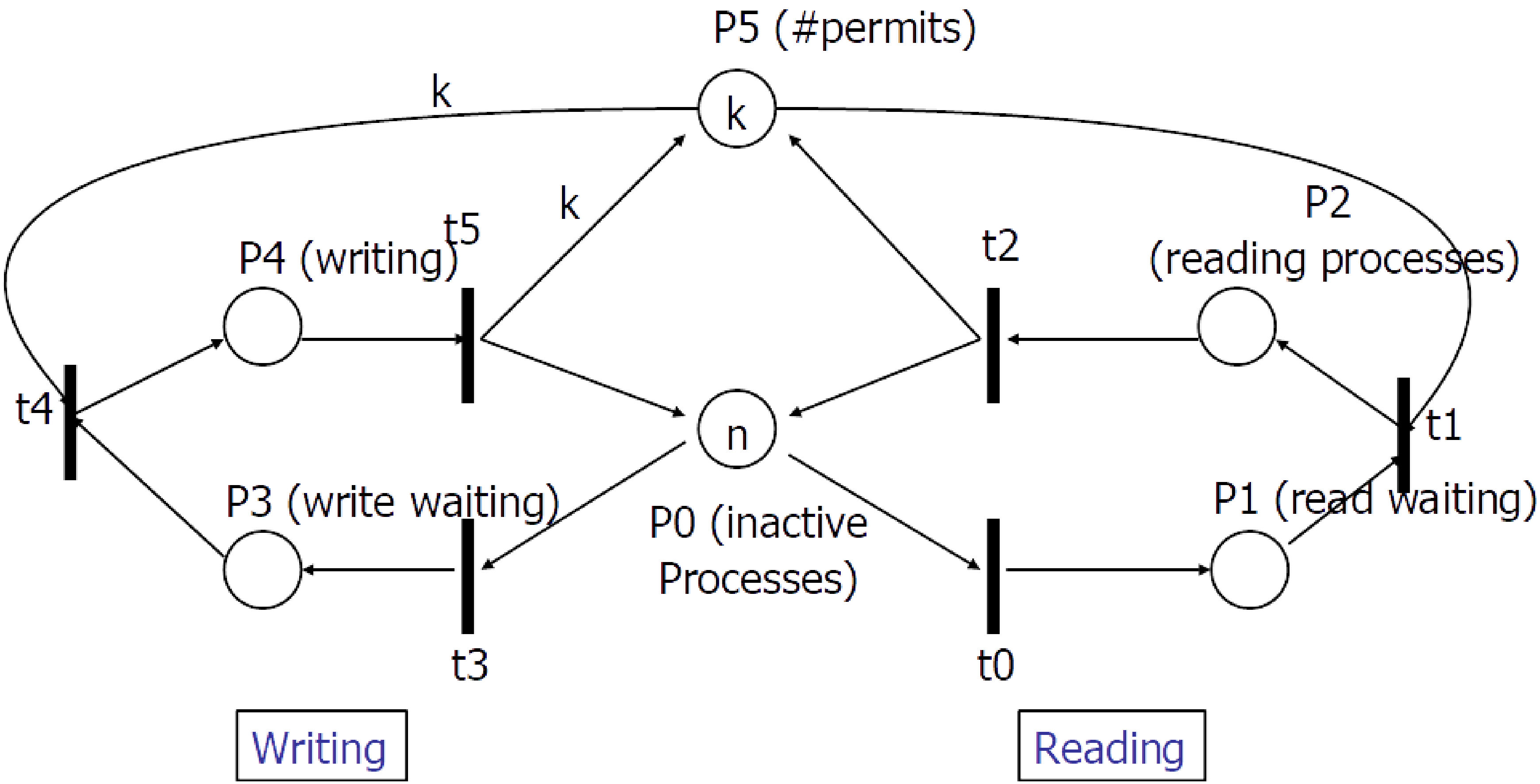
p2 looses 1 to t2; p2 gets 1 from t1

...



	p1	p2	p3	p4
t1	-2	1	1	0
t2	1	-1	0	-2
t3	1	0	-1	2

A System of n Processes Sharing k Resources



Incidence Matrix Transposed

p0 looses 1 to t0; p1 gets one from t0;
p1 looses one from t1; p2 gets 1 from t1 …
p5 looses k to t4; p5 gets k from t5

	t0	t1	t2	t3	t4	t5
p0	-1		1	-1		1
p1	1	-1				
p2		1	-1			
p3				1	-1	
p4					1	-1
p5		-1	1		-k	k

Firing Vectors

Marking M_k is written as an m x 1 column vector.

- The j-th entry of M_k denotes the number of tokens in place j after the k-th firing.

The *firing vector* u_k is a n x 1 column unit vector of n – 1 zeros and one nonzero entry,

- 1 in the i-th position indicates that transition i fires at the k-th firing
- The firing vector characterizes a firing transition

The *recurrence* for incidence matrix A is a matrix equation over firing vectors:
$M_k = M_{k-1} + A^T u_k$, k = 1, 2 ....

- This equation summarizes all reachable states

State Equation and Firing Count Vectors

Suppose M_d is reachable from M_0 through a firing sequence $\{u_1, u_2, \ldots, u_d\}$

The *state equation* to compute M_d from M_0 is: $M_d = M_0 + A^T \sum_{k=1}^{d} u_k$

Which can be rewritten as $A^T x = \Delta M$

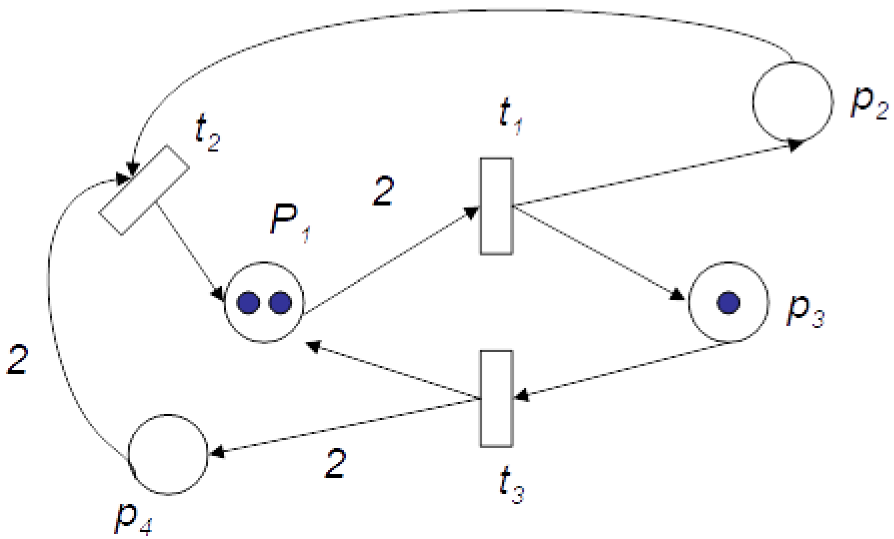
- Where $\Delta M = M_d - M_0$ and

x is an n x 1 column vector (*firing count vector*)

- The i-th entry of x denotes the number of time transition i must fire to transform M_0 to M_d.
- A firing count vector characterizes a firing sequence

$$x = \sum_{k=1}^{d} u_k$$

Example: Incidence matrix



A PN with initial marking $(2\ 0\ 1\ 0)^T$. The state equation is shown, where t_3 fires to result in $M_1 = (3\ 0\ 0\ 2)^T$.

$$M_k = M_{k-1} + A^T u_k$$

$$\begin{bmatrix} 3 \\ 0 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} -2 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & -2 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Invariants of Transition Matrices

A null evaluation of the transposed incidence matrix, $A^T x = 0$, is called a *T-invariant (transition invariant)*

- A T-invariant is a firing count vector (firing sequence) that transforms a marking into itself
- Does not specify the order, but the number of firings for each transition

A null evaluation of the incidence matrix, $Ay = 0$ is called an *S-invariant (state invariant)*

The invariants are used for studying structural properties.

Structural properties with T-Invariants

An invariant (vector) y is *minimal* if there is no other invariant y1 such that y1(p) ≤ y(p) for all p.

- An invariant can be written as linear combinations of minimal support invariants
- A *minimal T-invariant* is a minimal firing sequence that transfers a marking into itself

Theorem: An n-vector $x > 0$ is a T-invariant iff there exists a marking M_0 such that its firing sequence σ leads from M_0 back to M_0

T-Invariants and Liveness

The theorem delivers directly a check procedure to check liveness of a PN

- *Switch vector:* the sum of all minimal T-invariants.
- Calculate minimal T-invariants as elementary null evaluations
- Check that switch vector does not have null entries
- Build the reachability graph, and test whether the initial marking M_0 is reachable from all reachable configurations.

Repeat: Purpose of Liveness: Protocol Checking for Components

Describe the behavior of two components with two PN

Link their ports

Check on liveness of the unified PN

- If the unified net is not live, components will not fit to each other...

Check on boundedness:

- Estimate consumed resources

The End

Thanks to Björn Svensson who did many of the slides, summarizing [Murata]