3. Formal Features of Petri Nets

Prof. Dr. U. Aßmann

Technische Universität Dresden

Institut für Software- und Multimediatechnik

Softwaretechnologie

http://st.inf.tu-dresden.de

14-0.1, 10/29/14

Lecturer:

Dr. Sebastian Götz

1) Reachability Graph
2) Boundedness
3) Liveness

Content

- Behavioral properties of petri nets
  - 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**. Lecture Slides http://www.daimi.aau.de/~kjensen Many other links and informations, too
  - www.daimi.aau.dk/CPnets the home page of CPN. Contains lots of example specifications. Very recommended
- W. Tichy. **Lectures on Software Engineering.** Karlsruhe University

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

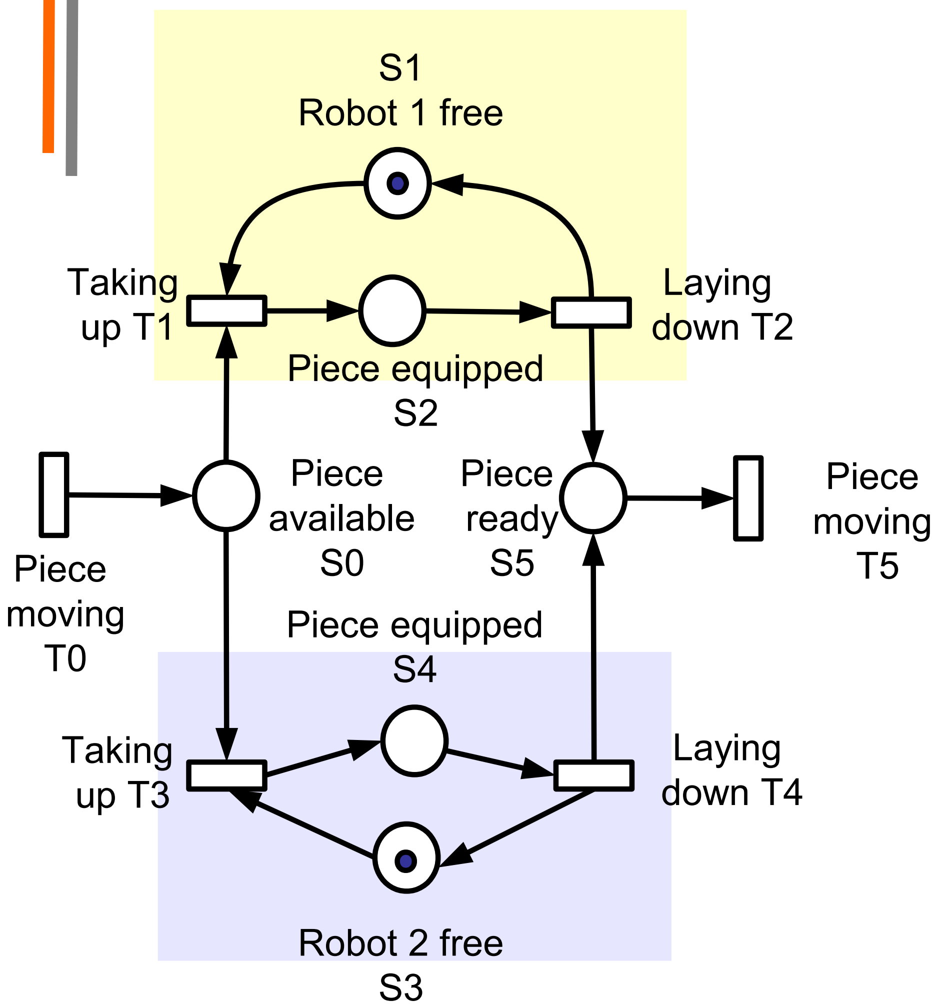
- Understand the **isomorphism** between finite automata (statecharts) and bounded Petri nets
- Understand why Petri nets are useful

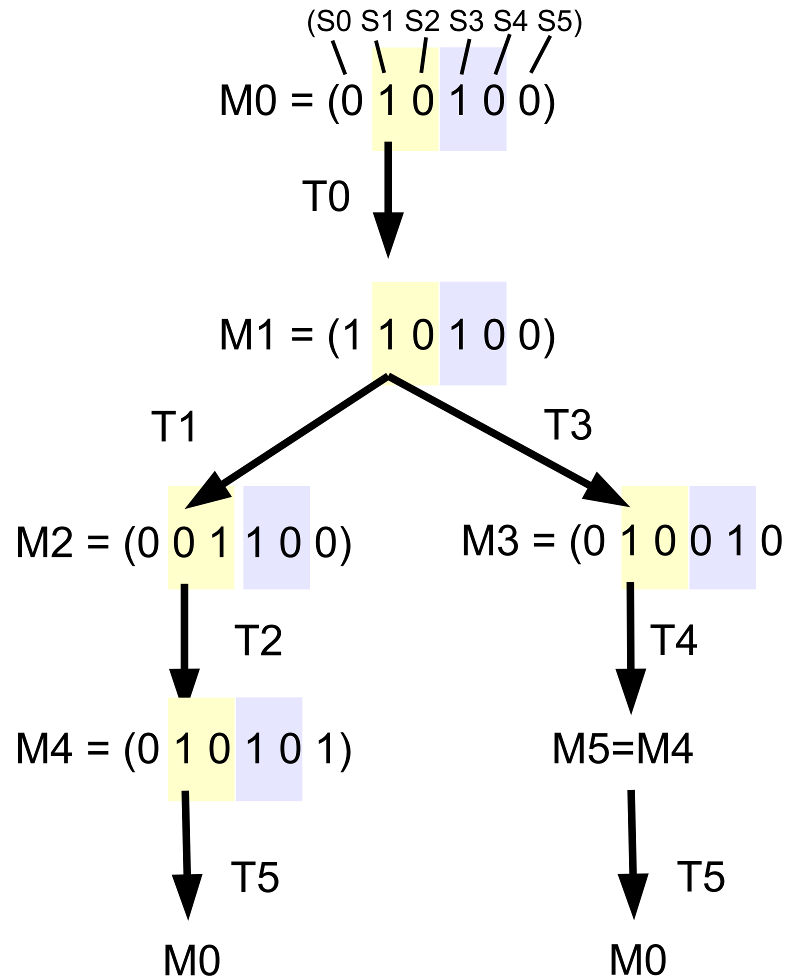
03b.1 Behavioral Properties of PN

Reachability of Markings

- If transaction t is *enabled* in the marking 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

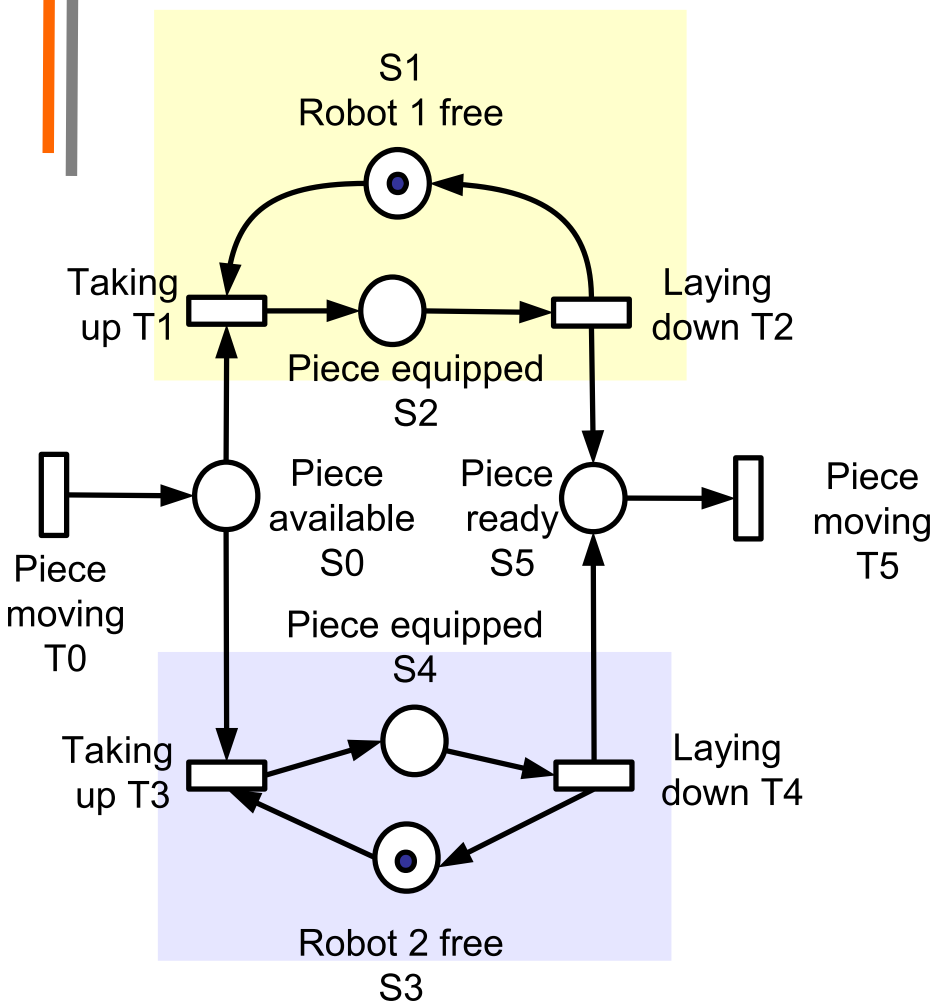


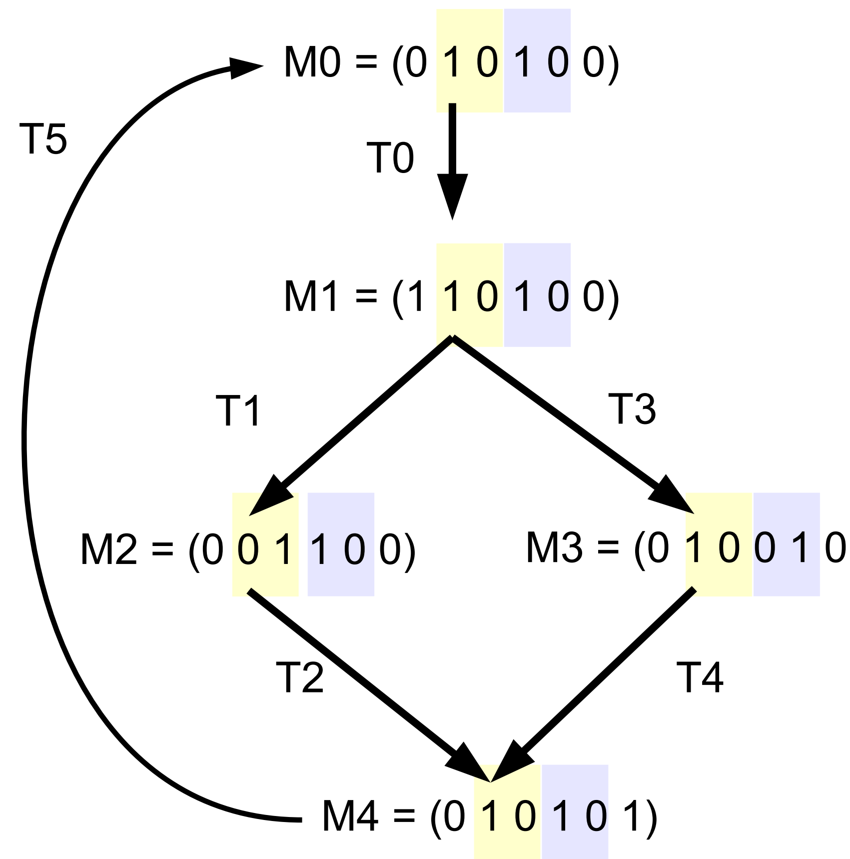


Upper part of net (S1, S2)

Lower part of net (S3, S4)

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



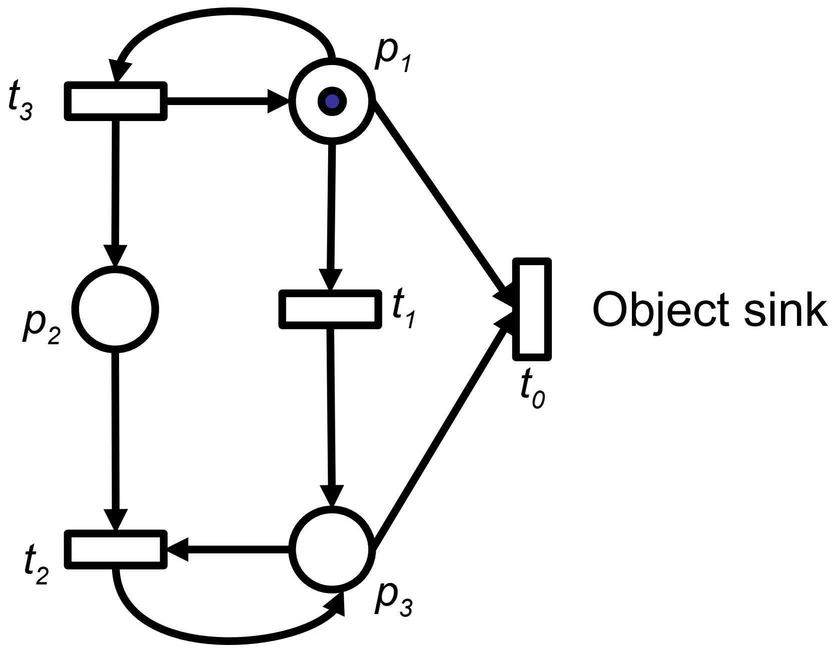


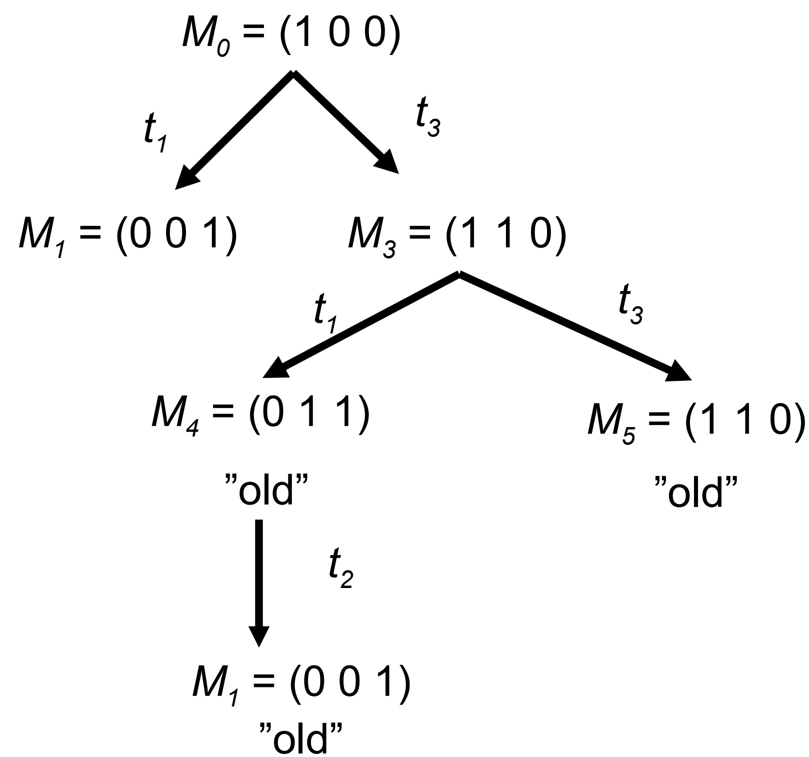
Upper part of net (S1, S2)

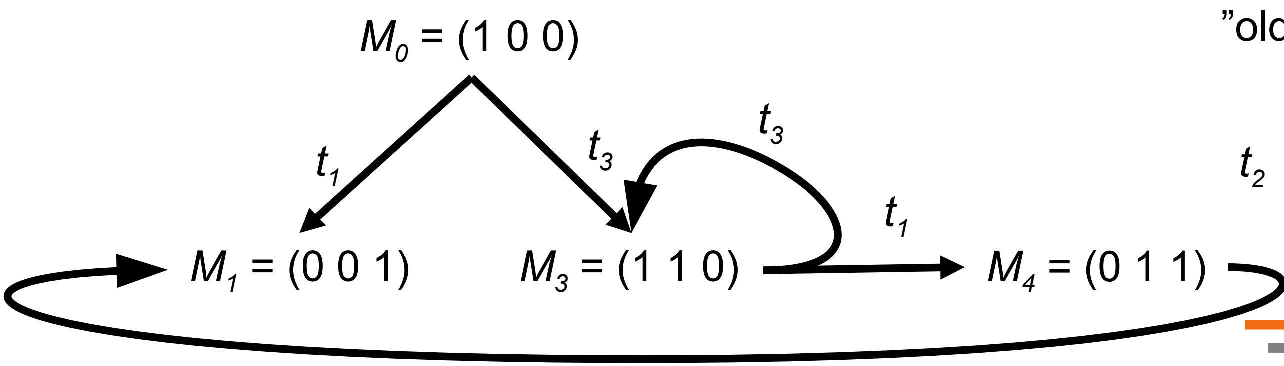
Lower part of net (S3, S4)

Example: The Reachability Tree and Graph

Only one token per place at a time.





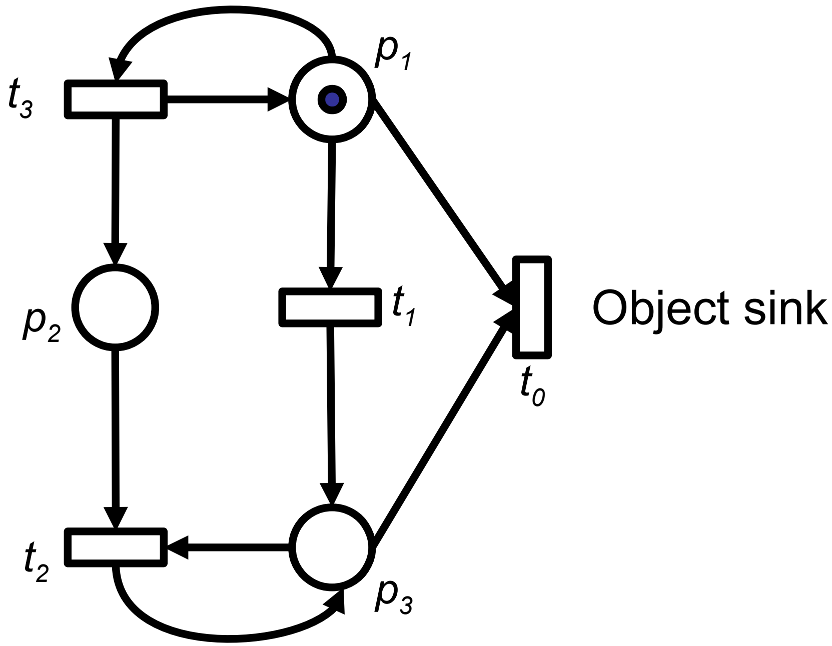


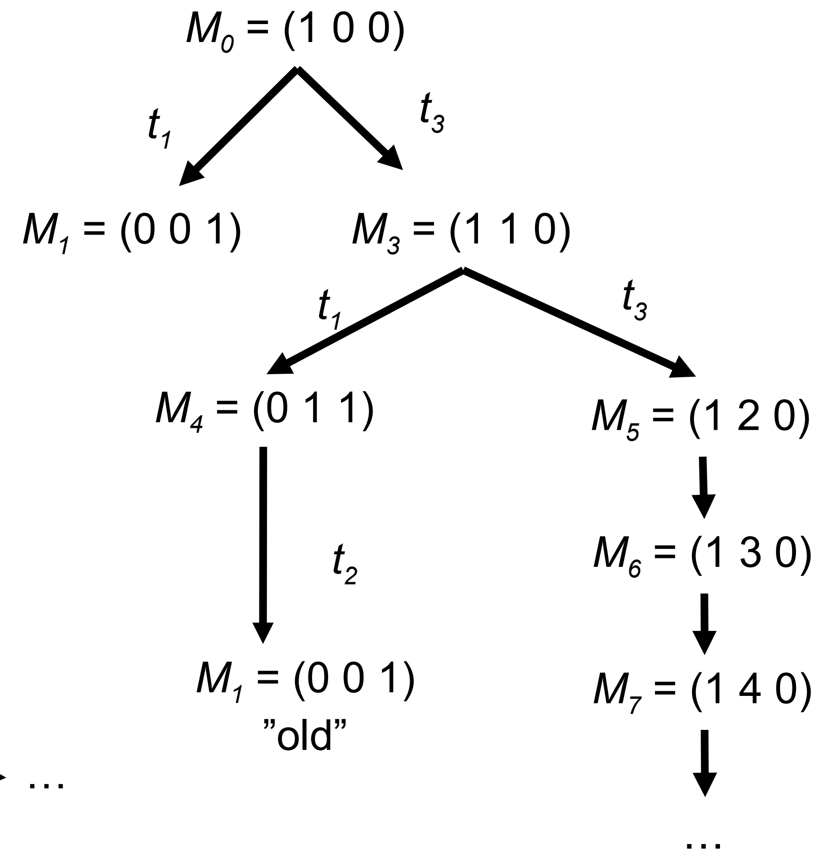
Boundedness and Safety

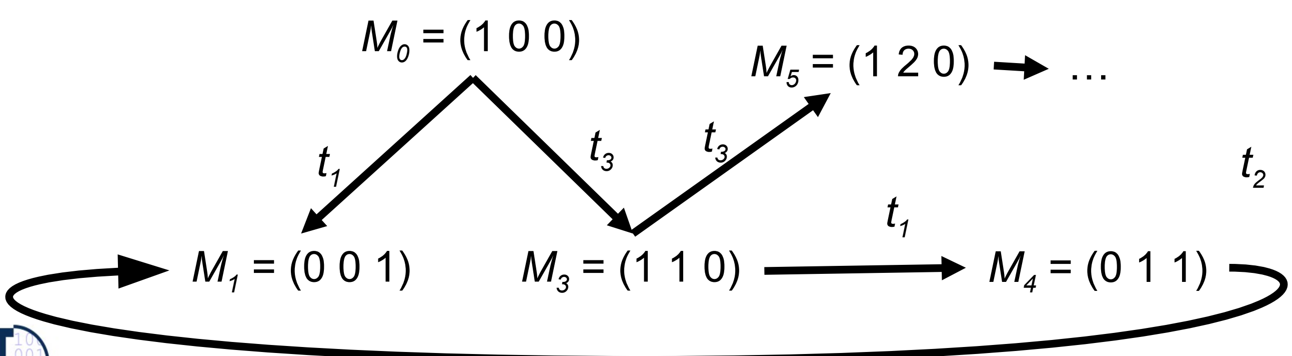
- A PN (N,M_0) is *k-bounded* or simply *bounded* if every place is size-restricted by k
  - $M(p) \leq k$ for every place p and every marking M in $R(M_0)$.
- A PN is *safe* if it is 1-bounded.

- 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

Example: Unbounded net







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 consumes k resources at most
  - Important for systems with resource constraints

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:

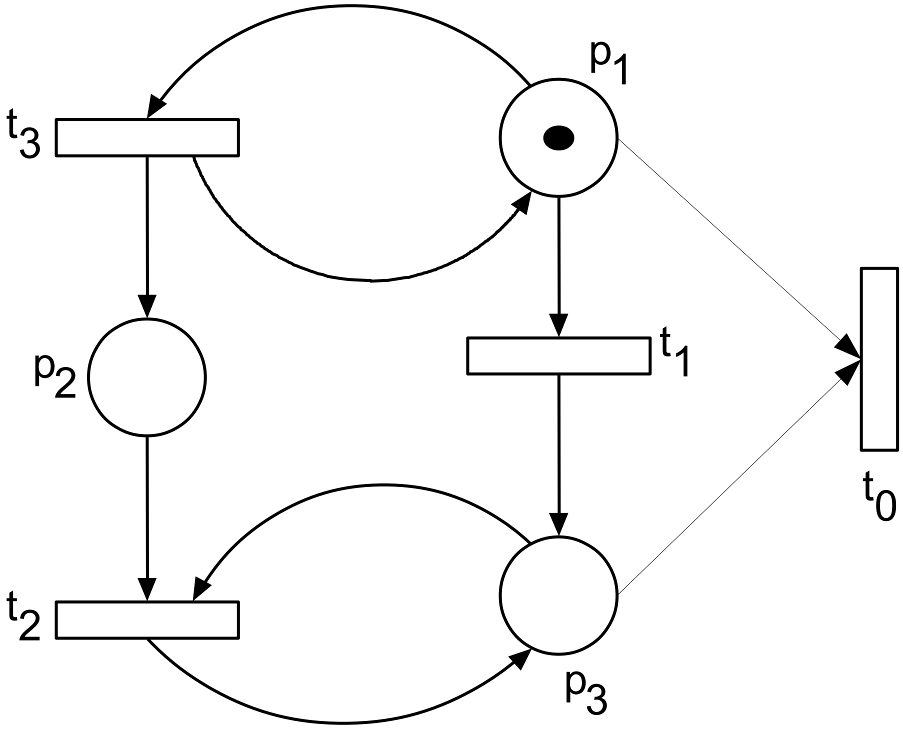
- *Dead (L0-live)* if t can never be fired in any firing sequence in $R(M_0)$. (not fireable)
- *L1-live (potentially fireable)* if t can be at least fired once in some firing sequence in $R(M_0)$. (firing at least once from the start configuration)
- *L2-live (k-fireable)* if t can be fired at least *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)
- *live (L4-live)* if t is L1-live for every marking *M* in $R(M_0)$. (This is more: t is always fireable again in a reachable marking)

Liveness of Markings and Nets

- A marking is *dead* if no transition is 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)

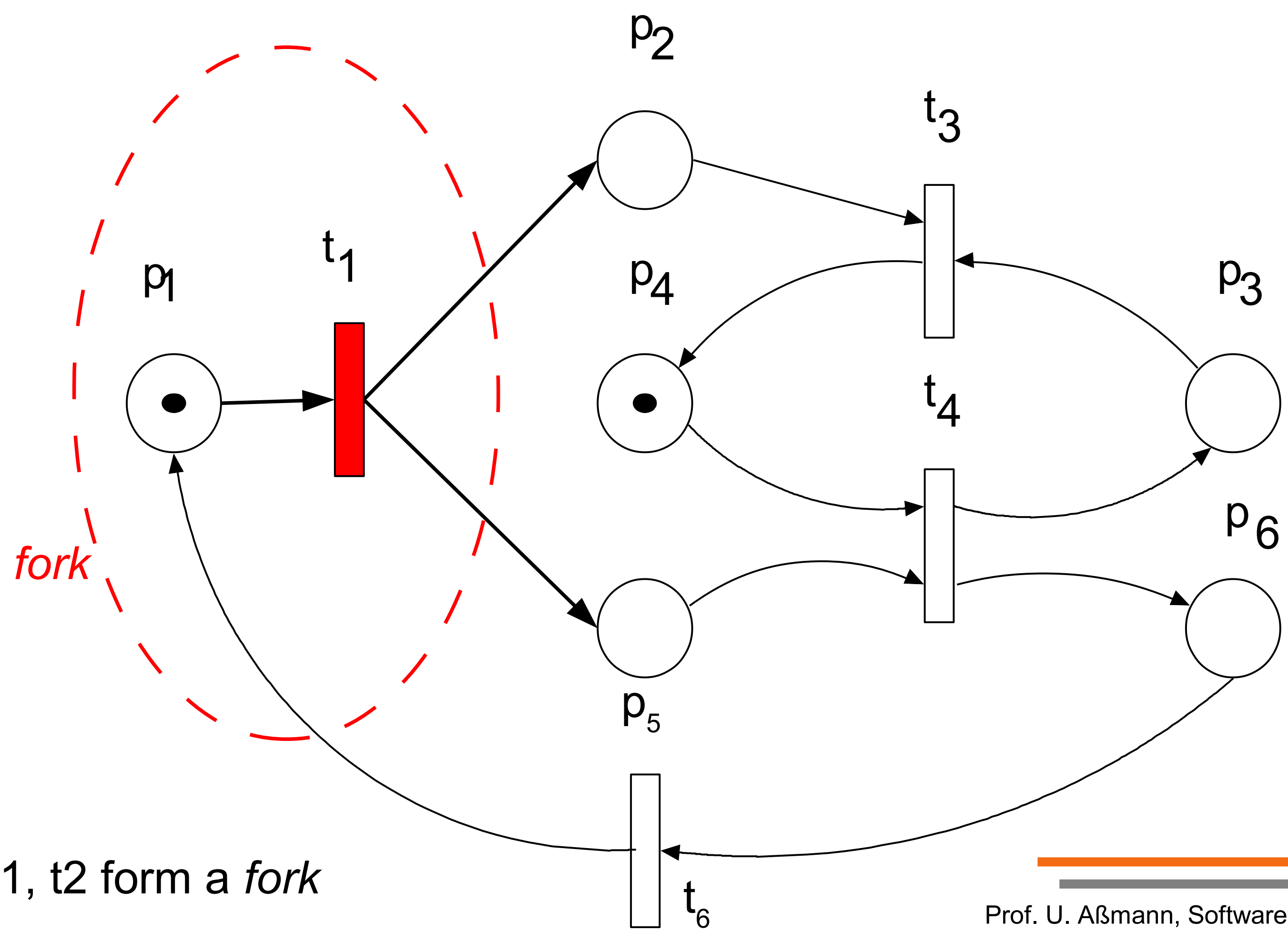
Example: Liveness

- Assumption: net is boolean
- t_1 L1-live (fireable only once, brigde)
- 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)



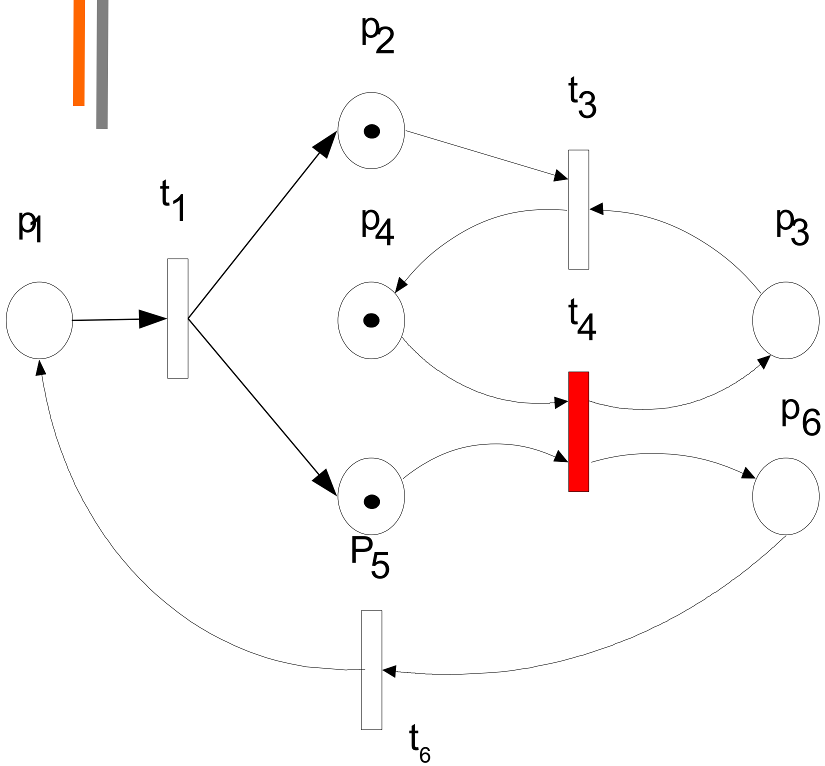
Example: Liveness

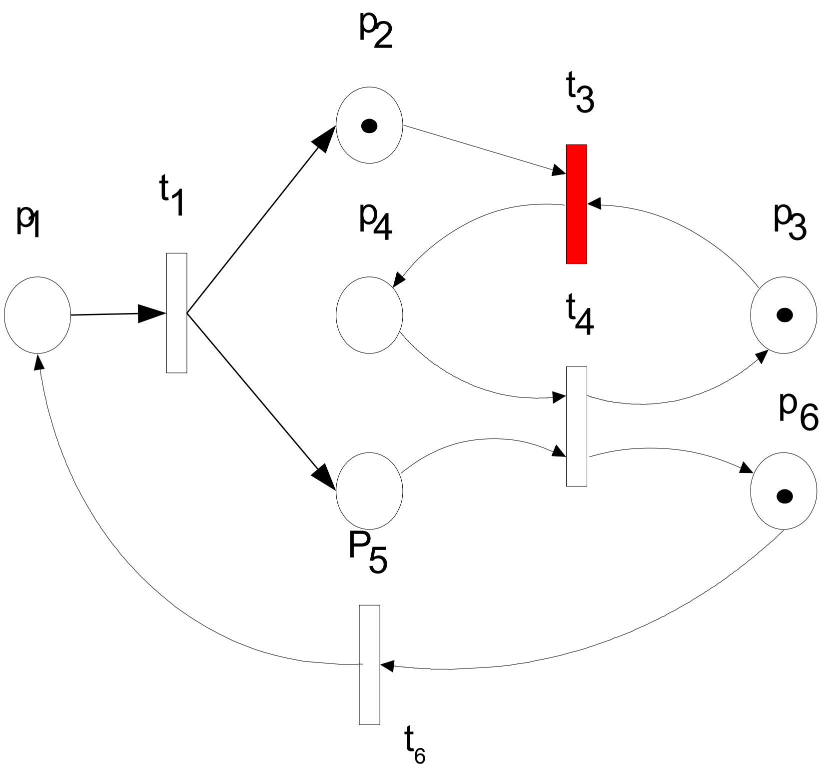
- A safe, live PN. M0 can be reproduced again, e.g., with t1 t2 t4 t3 t6 reproduces a filled p1 and p2



p1, t1, t2 form a *fork*

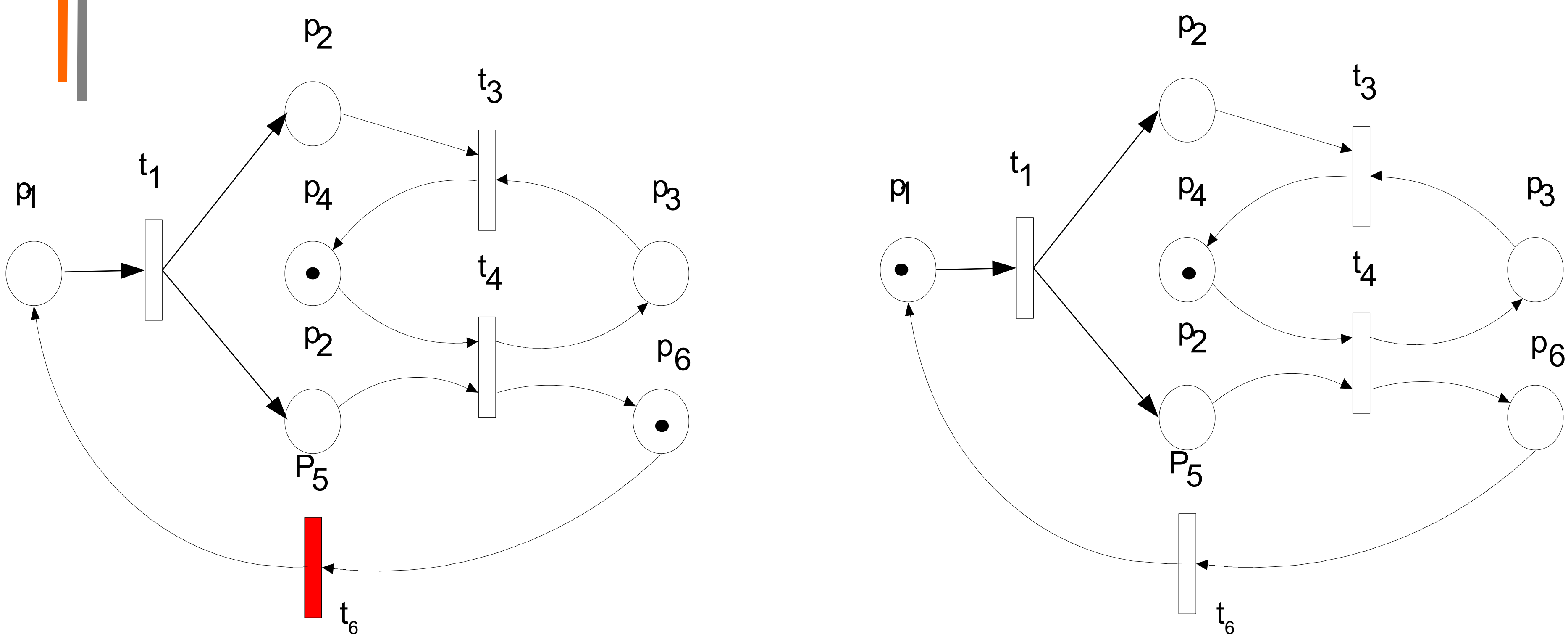
Example: Liveness





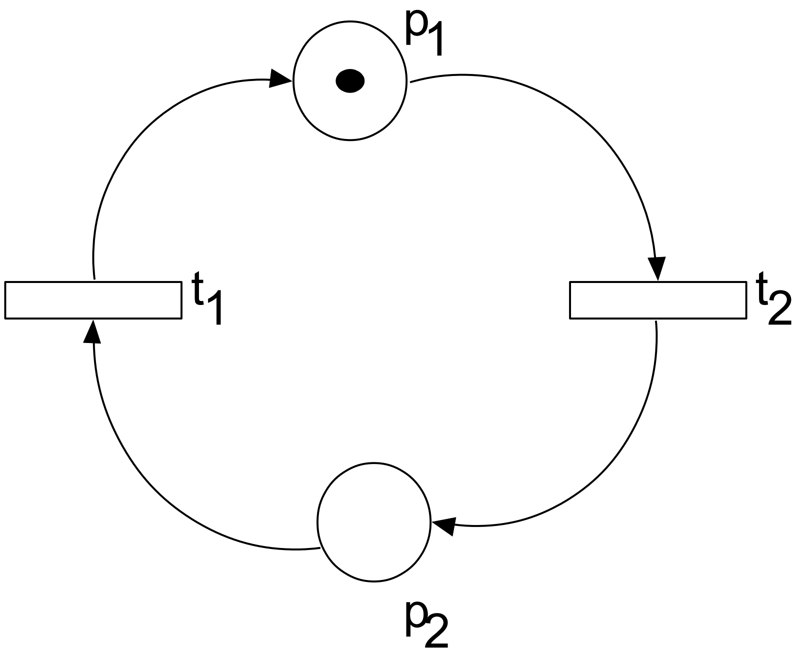
p2 is a synchronization dependency; process p5 can run earlier, p2 has to wait.
Note: the content of p2 must be reproduced again

Example: Liveness



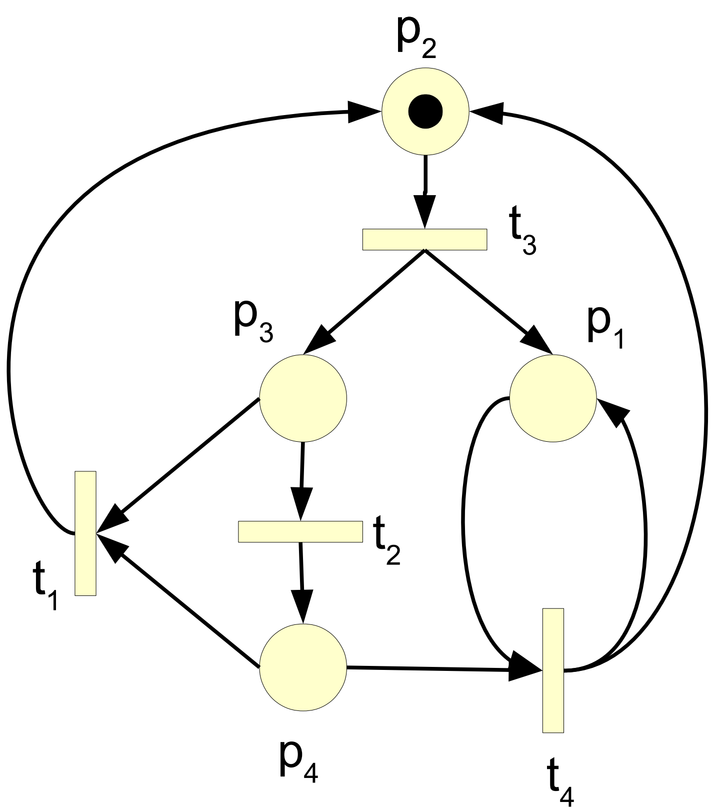
Net is unbounded, due to the reproduction facilities of t6,
i.e., running t6 before t3:
t1 t2 t4 t6 t1 t2 t4 t6 ...

Well, everything: Safe, Live



Not Bounded, Not Live

- Not live, because t1 is never enabled
- Unbounded:
  (0100) t3 (1010) t2 (1001) t4
  (2100) t3 (2010) t2 (2001) t4
  (3100) ...



What have we learned?

- Behavioral properties of petri nets
  - Reachability
  - Liveness
  - Boundedness
- Formal approaches (matrix algebra) out of this lectures scope

The End