Computational Models for Embedded Systems

Vescan Andreea, PHD, Assoc. Prof.

Faculty of Mathematics and Computer Science
Babeș-Bolyai University
Cluj-Napoca

2025-2026

Lecture 10b: Petri Nets (2)

Software Systems Verification and Validation

"Tell me and I forget, teach me and I may remember, involve me and I learn."

(Benjamin Franklin)

Outline

- PetriNets
- Analysis Methods for Petri Nets
  - Boundedness
  - Conservation
  - Liveness
  - Reachability and Coverability
  - Persistence
- The Coverability Tree

- Next lecture: Having fun learning about Petri nets
- 50XP bonus points
- If you like to participate in the bonus activity, please find your results for the VARK questionnaire before Lecture 10:
- https://vark-learn.com/the-vark-questionnaire/
- Questions

Outline

- PetriNets

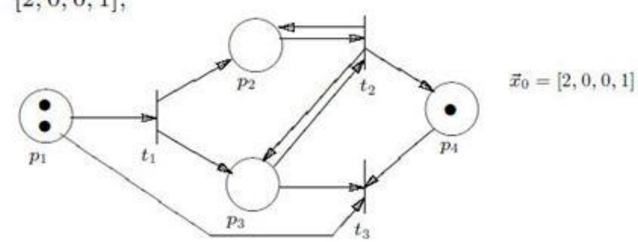
Petri net Definition

- A Petri net is a six-tuple N=(P,T,A, w, $\vec{x}_0$), where
  - P is a finite set of places
  - T is finite set of transitions
  - A is a set of arcs, $A \subseteq (P \times T) \cup (T \times P)$
  - W is a weight function, $w : A \to N$
  - $\vec{x}_0$ is an initial marking vector, $\vec{x}_0 \in N^{|P|}$
- The set $I(t) = \{p \in P \mid (p,t) \in A\}$ is the set of input places of transition t.
- The set $O(t) = \{p \in P \mid (t,p) \in A\}$ is the set of output places of transition t.
- A transition t is enabled in state $\vec{x}$ if

$$x(p) \geq w(p,t), \forall p \in I(t)$$

Petri net $N = (P, T, A, w, \vec{x}_0)$ with

$$
\begin{aligned}
P &= \{p_1, p_2, p_3, p_4\} \\
T &= \{t_1, t_2, t_3\} \\
A &= \{(p_1,t_1), (p_1,t_3), (p_2,t_2), (p_3,t_2), (p_3,t_3), (p_4,t_3), \\
&\quad (t_1,p_2), (t_1,p_3), (t_2,p_2), (t_2,p_3), (t_2,p_4)\} \\
w(a) &= 1 \ \forall a \in A \\
\vec{x}_0 &= [2, 0, 0, 1],
\end{aligned}
$$



PetriNets

- Analysis Methods for Petri Nets
  - Boundedness
  - Conservation
  - Liveness
  - Reachability and Coverability
  - Persistence

  - A technique for addressing these questions on PN:

    The Coverability Tree

Boundedness

- Boundedness: the number of tokens in any place cannot grow indefinitely
  - (1-bounded also called *safe*)
- Application: places represent buffers and registers (check there is no overflow), or a memory or a queue

A place $p \in P$ in a Petri net $N = (P, T, A, w, \vec{x_0})$ is *k-bounded* if:

$$\forall y \in R(\vec{x_0}) : y(p) \leq k.$$

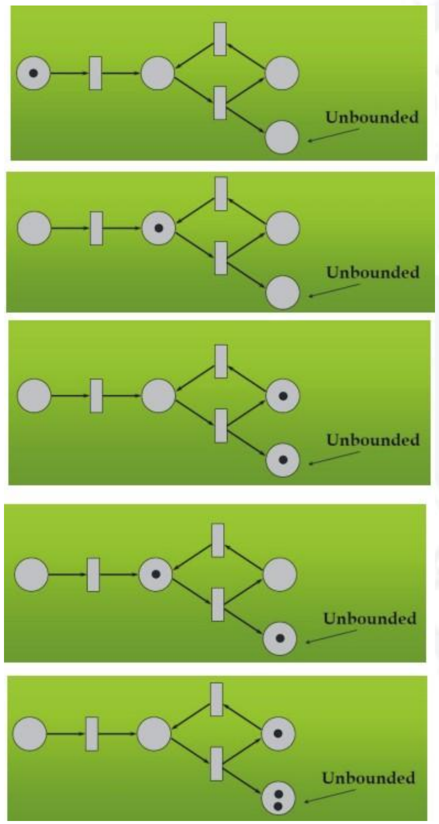
The Petri net is called *k-bounded* if all places $p \in P$ are *k-bounded*.

That definition is only useful for nets in isolation with no inputs because an input place may always be unbounded, depending of course on the environment's behavior.

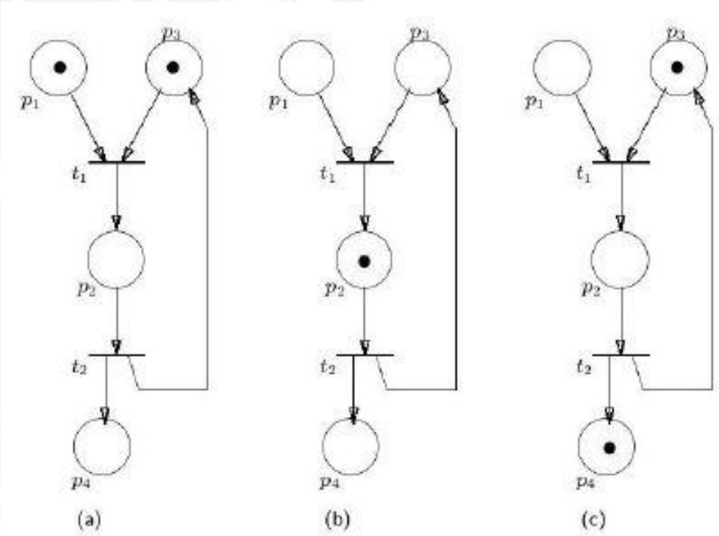
Boundedness

- A not bounded PN

BOUNDEDNESS: THE NUMBER OF TOKENS IN ANY PLACE CANNOT GROW INDEFINITELY



- A bounded PN



Conservation

- Token can also represent other things (data, requests, customers, services, resources, etc.)
- It can be interesting to know if the number of tokens for all reachable states is constant

A Petri net $N = (P, T, A, w, \vec{x_0})$ with n places is conservative with respect to a weighting vector $\vec{\gamma} = [\gamma_1, \gamma_2, \ldots, \gamma_n], \gamma_i \in N$, if

$$\sum_{i=1}^{n} \gamma_i \vec{x}(p) = \text{constant } \forall p \in P \text{ and } \vec{x} \in R(\vec{x_0})$$

The Petri net is conservative if it is conservative with respect to a weighting vector which has a positive non zero weight for all places.

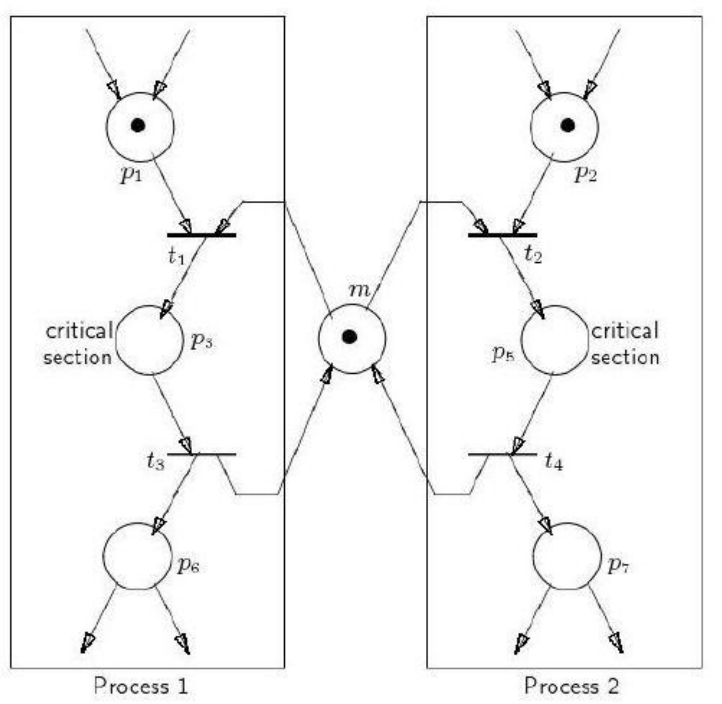
N is conservative with respect to some weighting vector if the number of tokens within the specified places is constant !

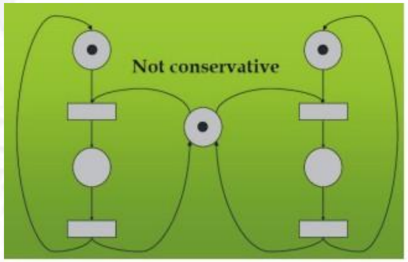
Conservation

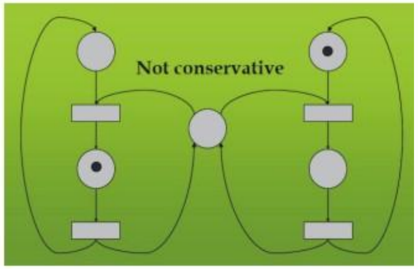
- constant number of printers?

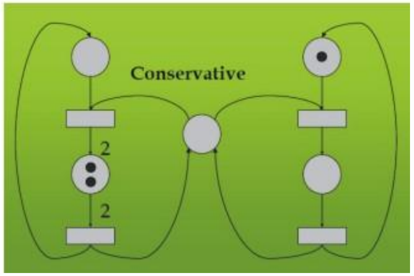
PN is conservative with respect to y = [0, 0, 1, 1, 1, 0, 0], p3,p4,p5 are relevant places

- Conservation: the total number of tokens in the net is constant









Liveness
What's a deadlock?

- Concurrent processes that interact with each other can come into a situation of deadlock:

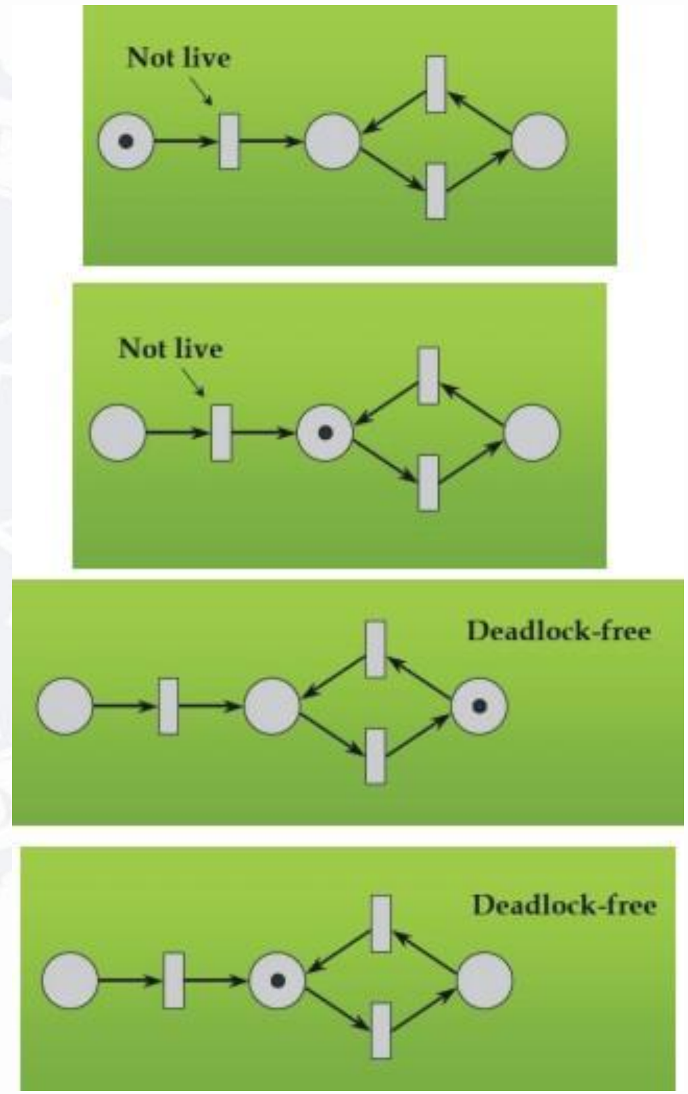
  All processes waiting for some action to be taken by another process.

- This phenomenon is:
  - widespread
  - potentially very harmful
  - difficult to predict and to avoid (it requires the analysis of all involved processes)

Liveness
What's a deadlock?

- Liveness:
- \- from any marking any transition can become fireable

- **Liveness implies deadlock freedom, not viceversa**



Reachability and Coverability

- Many other problems (including deadlock avoidance) can be viewed as a special case of the state reachability problem.
- This asks if a particular state can be reached from a given state.

  i.e., given a state $\vec{x}$, is a state $\vec{y} \in R(\vec{x})$
- If one can answer this question for any $\vec{x}$ and $\vec{y}$,then one can also answer for all questions about boundedness, conservation, deadlock and liveness.
- Unfortunattely, we do not have an efficient method to answer this question for PN with infinite state space.
- Fortunatelly, we can still solve some of these problems by analyzing a weaker property: Coverability!

Coverability

$N = (P, T, A, w, \vec{x_0})$ is a Petri net; $\vec{x}$ and $\vec{y}$ are arbitrary states;
State $\vec{x}$ covers state $\vec{y}$ if all transitions enabled in $\vec{y}$ are also enabled in $\vec{x}$:

$$x(p) \geq y(p) \forall p \in P.$$

State $\vec{x}$ strictly covers state $\vec{y}$ if $\vec{x}$ covers $\vec{y}$ and, in addition,

$$\exists p \in P : x(p) > y(p).$$

Let $\vec{x} \in R(\vec{x_0})$. A state $\vec{y}$ is coverable by $\vec{x}$ iff there exists a state $\vec{x}' \in R(\vec{x})$ such that $x'(p) \geq y(p)$ for all $p \in P$.

From a given state $\vec{x_0}$, can we reach a state that "covers" a state $\vec{y}$?

Coverability vs. Reachability

- Coverability is weaker than reachability:
  - Coverability = can we reach one state of an infinite set that share a common property
  - Reachability = can we reach one specific state

But still,
coverability can solve a great deal of the most common problems !

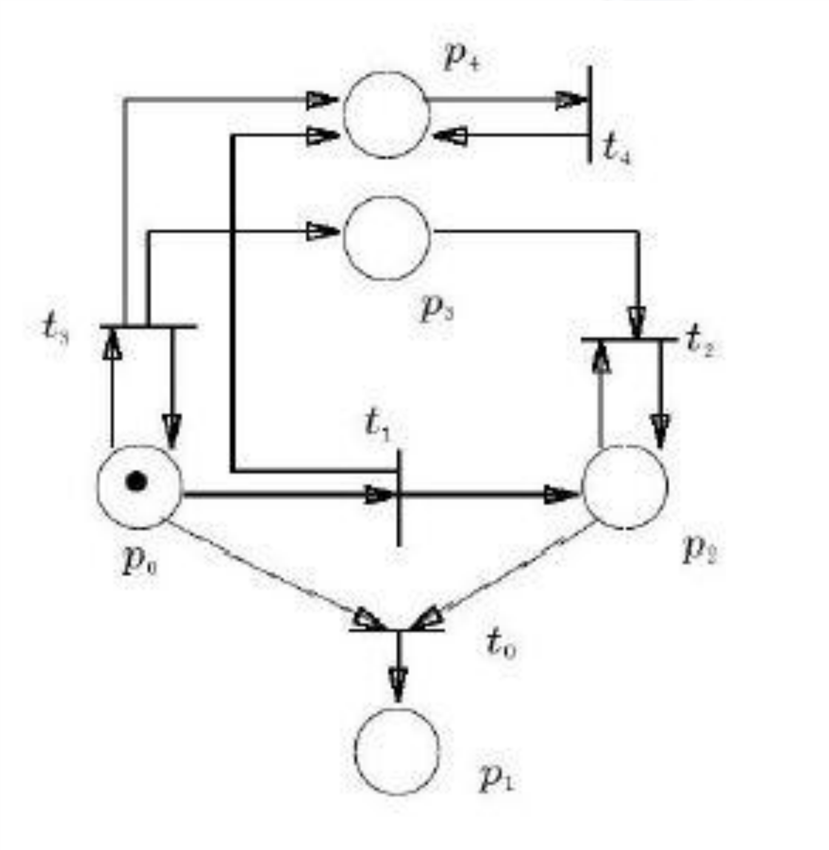
Persistence

- Understanding of the impact of one activity on another activity is of particular interest
- Eg: can the occurrence of an interrupt cause some tasks to miss their deadline?

→ Persistence captures some aspects of the relationship between transitions.

- Two transitions are persistent with respect to each other if, when both are enabled the firing of one does not disable the other.
- A Petri net is persistent if any two transitions are persistent with respect to each other.

Persistence



- Example

- This is not persistent because, if both t1 and t3 are enabled, the firing of t3 will disable t1. But t2 and t3 are persistent with each other.

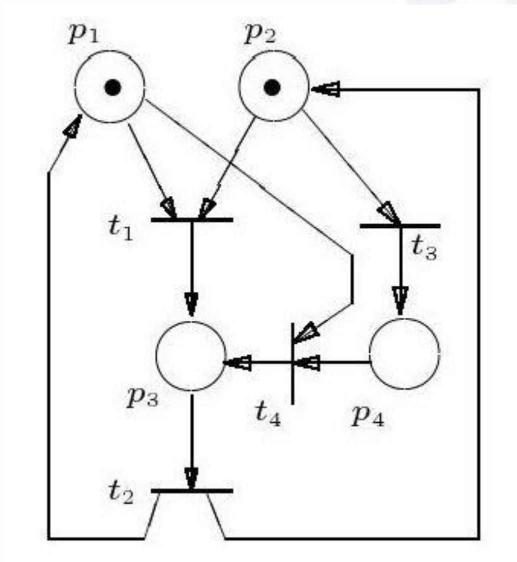
Coverability tree

- Efficient technique for addressing some of the problems discussed previously.

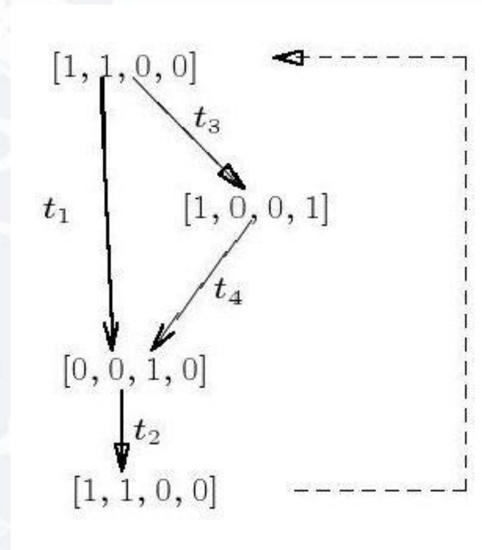
- Intuition: tree with the arcs representing transitions and nodes denoting sets of states that can be covered by a sequence of transitions.

Coverability tree
- Finite state space

- PN



- Coverability tree



- The tree simply contains all the reachable states.
- Works fine for finite state spaces, not for infinite state spaces (would lead to infinite coverability trees)

Coverability tree - definition

Let $N = (P, T, A, w, \vec{x_0})$ be a Petri Net.
A coverability tree is a tree where the arcs denote transitions $t \in T$ and the nodes represent w-enhanced states of the Petri net.
The root node of the tree is $\vec{x_0}$.
A terminal node is an w-enhanced state in which no transition is enabled.
A duplicate node is an w-enhanced state which already exists somewhere else in the coverability tree.
An arc t connects two nodes $\vec{x}$ and $\vec{y}$ in the tree, iff firing of t in state $\vec{x}$ leads to state $\vec{y}$.

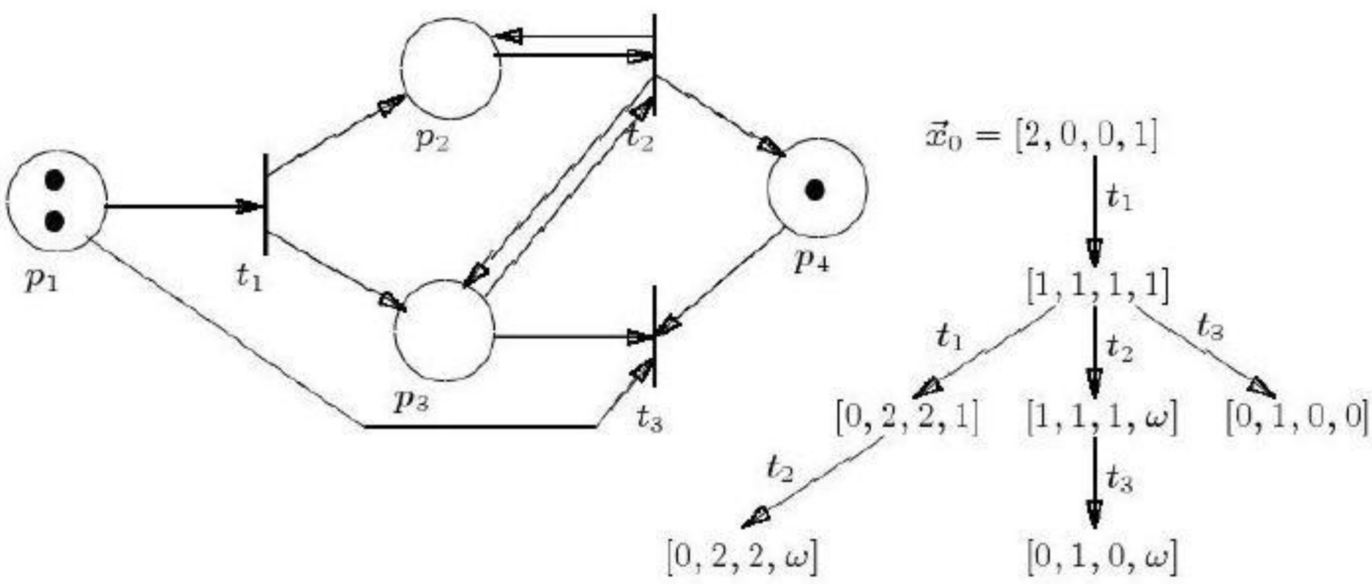
Coverability tree
- algorithm

Given is the Petri net $N = (P, T, A, w, \vec{x_0})$.

Algorithm:

Step 1. Set L, the list of open nodes, to $L := \vec{x_0}$.
Step 2. Take one node from L, named x, and remove it from L;
Step 2.1. if $G(\vec{x}, t) = \vec{x} \forall t \in T$
then x is a terminal node goto Step 3;
Step 2.2. for all $\vec{x}' \in G(\vec{x}, t), t \in T, \vec{x}\neg = \vec{x}'$
Step 2.2.1. do if $x(p) = \omega$ then set $x'(p) := x$;
Step 2.2.2. if there is a node $\vec{y}$ already in the tree,
such that $\vec{x}'$ covers $\vec{y}$
and there is a path from $\vec{y}$ to $\vec{x}'$,
then set $x(p) := \omega$ for all p for which $x'(p) > y(p)$;
Step 2.2.3. if $\vec{x}'$ is not a duplicate node then $L := L \cup \{\vec{x}'\}$;
Step 3. if L is not empty then goto Step 2.

Coverability tree - example



CT

Expressing properties

Boundedness

- A Petri net can be k-bounded if the symbol ω Never appears in its coverability tree.
- If the coverability tree contains an ω , a transition cycle to exceed a given k-bound can be identified.
- The coverability tree does not inform about the number of cycles required.

Expressing properties

Coverability Tree: Coverability and Reachability

- The coverability problem can be solved by inspection of the coverability tree.
- The shortest transition sequence leading to a covering state can be found efficiently.
- The reachability problem cannot be solved in general.

CT

Expressing properties

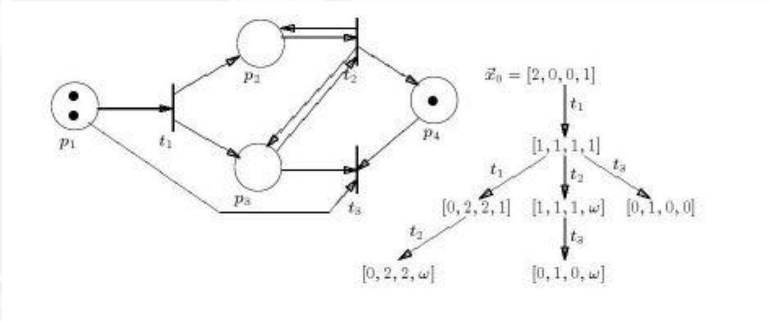
- Conservation

Recal: $\sum_{i=1}^{n} \gamma_i x(p) =$ constant for all $p \in P$ and $\vec{x} \in R(\vec{x_0})$.

If there is an ω the corresponding i must be 0.

Alg:

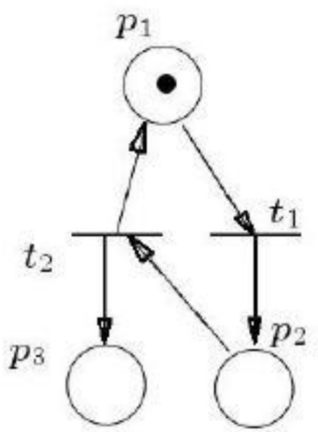
- Check if previous condition is fulfilled
- Check conservation (evaluated the weighted sum for each node in the CT
- If result = 0 then PN is conservative w.r.t. weighted vector



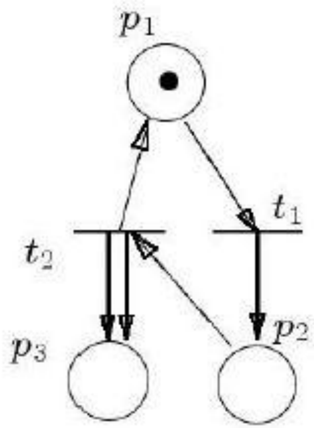
Limitations

Distinct Petri Nets with Identical Coverability Tree

- PN can have any number of token in p3



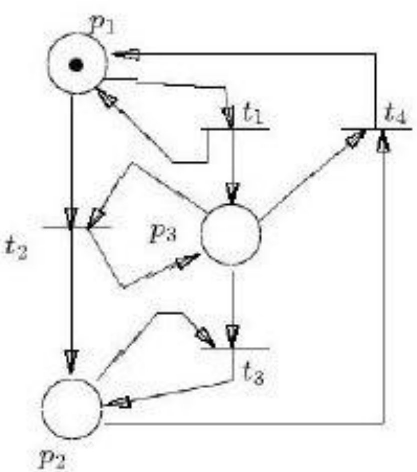
- PN can only have an even number of tokens in p3

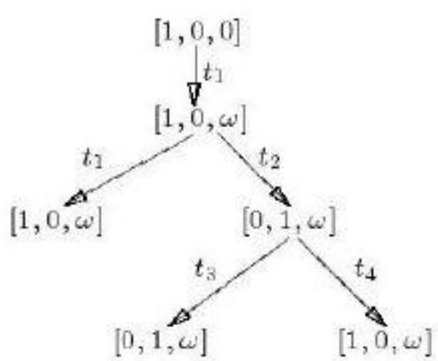


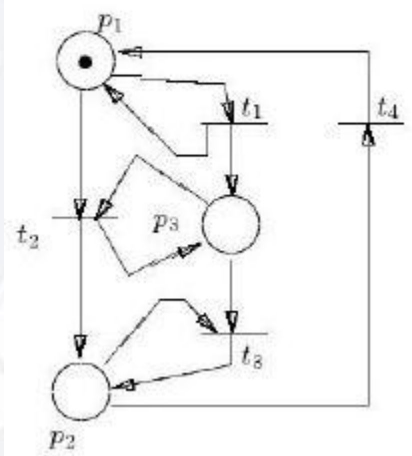
Limitations

Distinct Petri Nets with Identical Coverability Tree

- Can deadlock after t1,t2,t3
- Cannot deadlock







Limitations?

- Sometimes, it can however help solve deadlock, liveness or reachability problems:
  - If a state without an ω symbol is in the coverability tree it is also reachable.
  - Conversely, if a state cannot be covered, then it cannot be reached !

References

Sources

- **Models of Computations and Concurrency** or **Models of computation and their applications to Embedded systems modeling**, Lionel Morel, TUCS,
  http://users.abo.fi/lmorel/MoCs/
  - http://lionel.morel.ouvaton.org/wp/?page_id=13

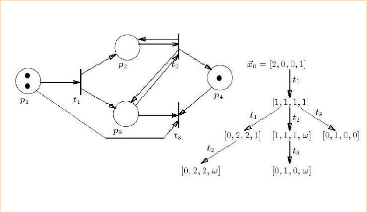
- **Design of Embedded Systems: Models, Validation and Synthesis**, Alberto Sangiovanni-Vincentelli
  https://inst.eecs.berkeley.edu/~ee249/fa07/

CMES – Today Bring it All Together

Petri nets

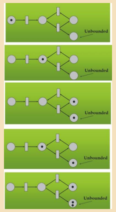
- Analysis Methods for Petri Nets
  - Boundedness
  - Conservation
  - Liveness
  - Reachability and Coverability
  - Persistence

- The Coverability Tree

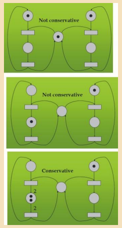


Petri nets

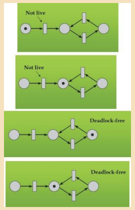
Boundedness



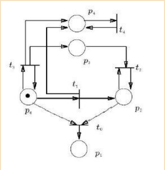
Conservation



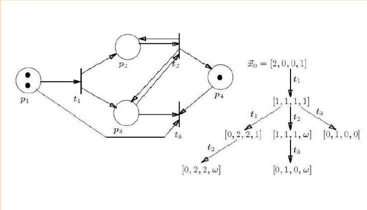
Liveness



Persistence



- The Coverability Tree



Thank You For Your Attention!

- ExitTicket
- Mentimeter
  - menti.com

Next Lecture

- 12 December 2025, 18-20, room C335 + 20-21 (extra hour)
- Having fun learning about Petri nets
- 50XP bonus points
- If you like to participate in the bonus activity,
- please find your results for the VARK questionnaire before Lecture 11:
- https://vark-learn.com/the-vark-questionnaire/

Software Systems Verification and Validation

“Tell me and I forget, teach me and I may remember, involve me and I learn.”

(Benjamin Franklin)