Outline

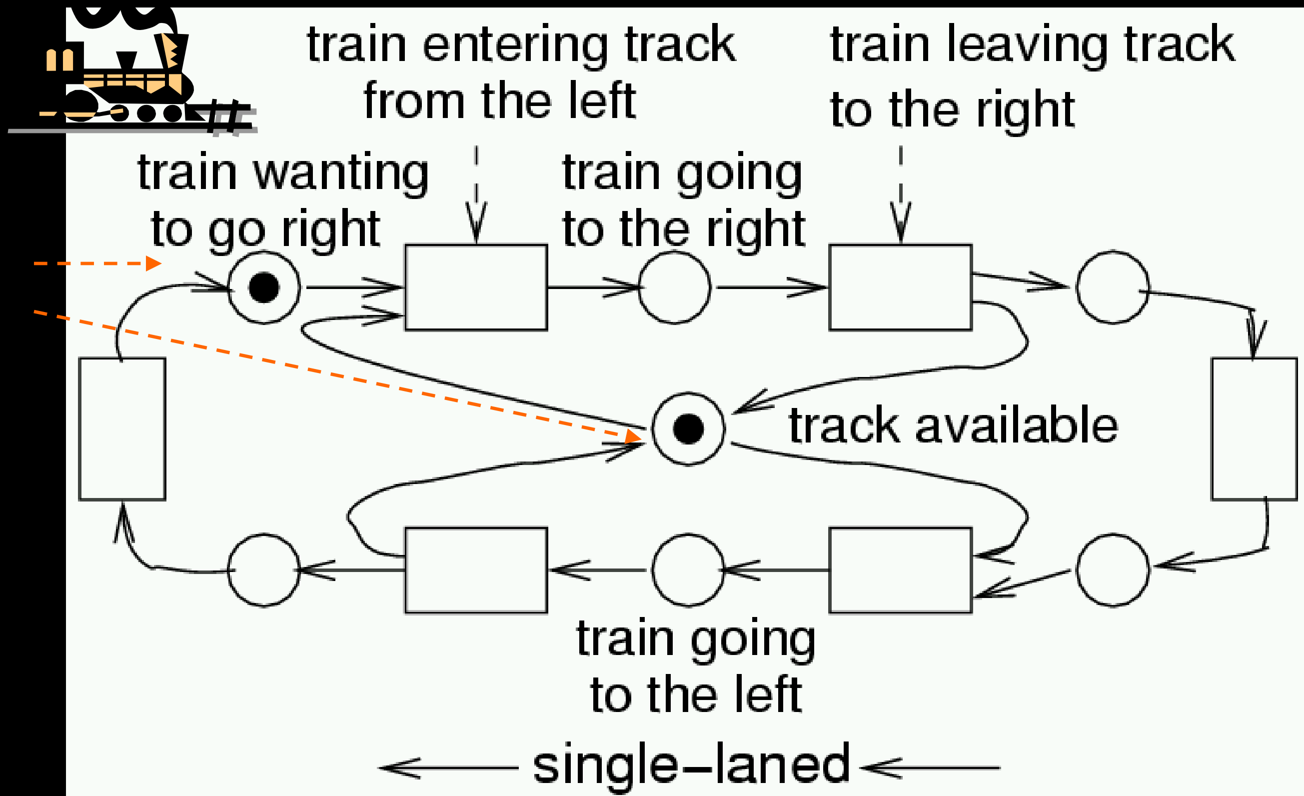
- **Petri nets**
  - **Introduction**
  - **Examples**
  - **Properties**
  - **Analysis techniques**

Petri Nets (PNs)

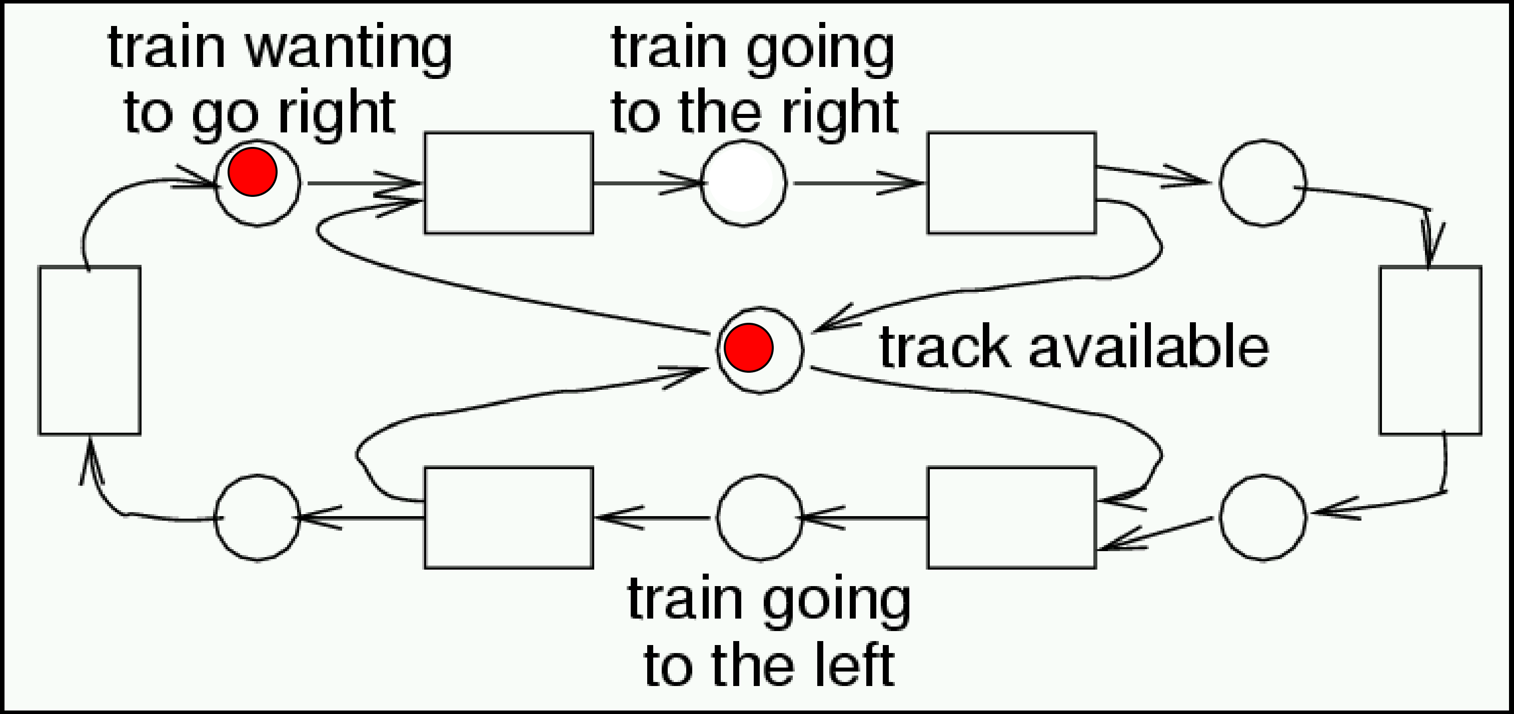
- Model introduced by C.A. Petri in 1962
  - Ph.D. Thesis: "Communication with Automata"
- Applications: distributed computing, manufacturing, control, communication networks, transportation…
- PNs describe explicitly and graphically:
  - sequencing/causality
  - conflict/non-deterministic choice
  - concurrency
- Basic PN model
  - Asynchronous model (partial ordering)
  - Main drawback: no hierarchy

Example:
Synchronization at single track rail segment

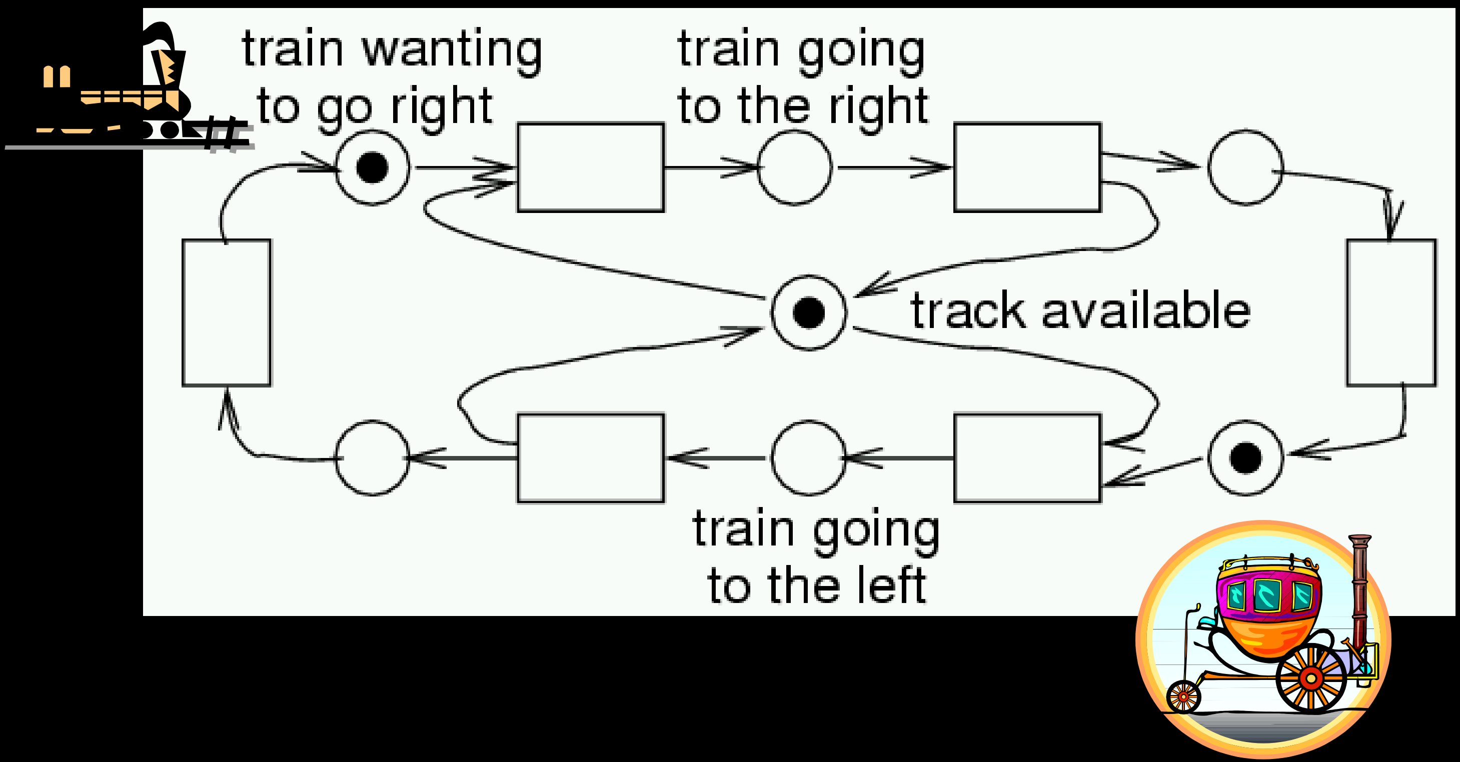
- "Preconditions"



Playing the “token game“

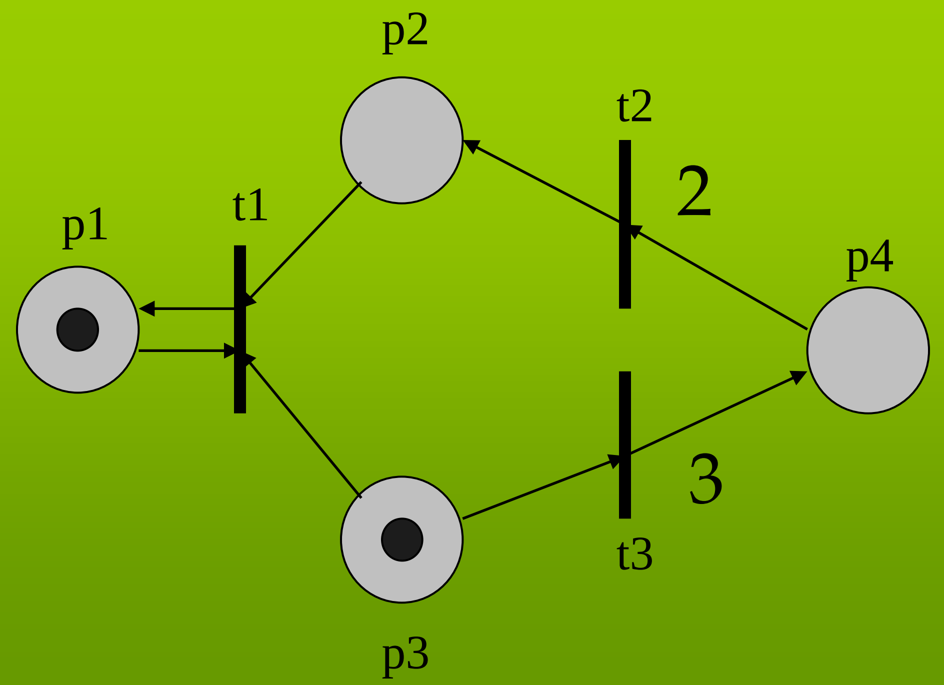


Conflict for resource “track“



Petri Net Graph

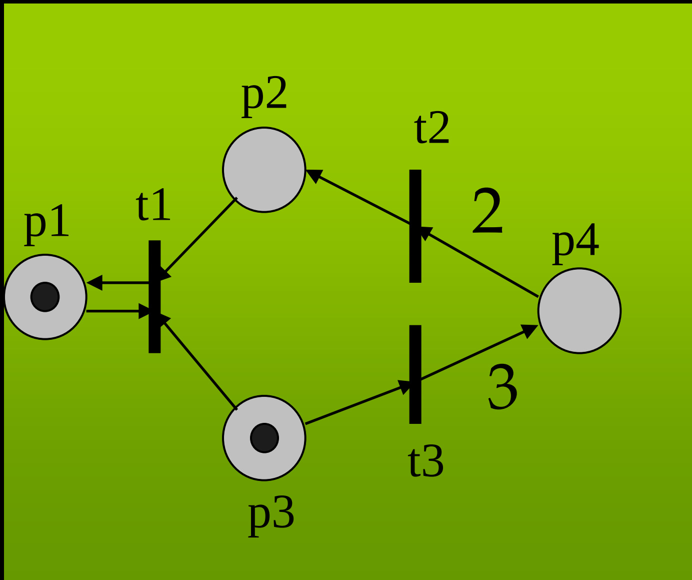
- **Bipartite weighted directed graph:**
  - **Places: circles**
  - **Transitions: bars or boxes**
  - **Arcs: arrows labeled with weights**
- **Tokens: black dots**



Petri Net

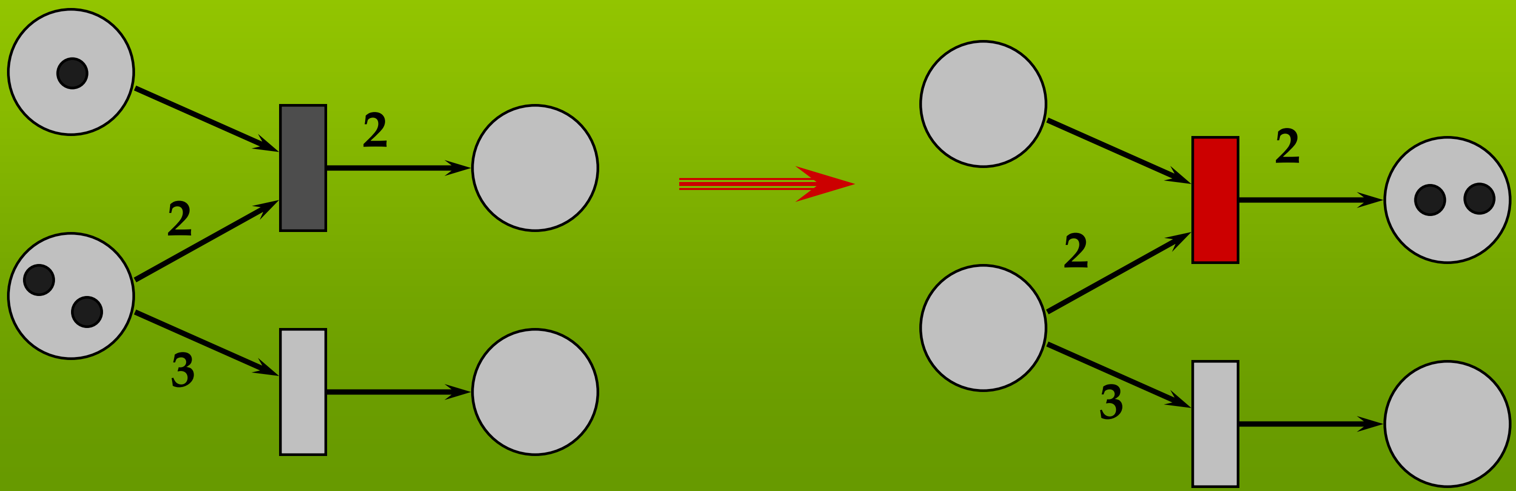
- **A PN (N,M_0) is a Petri Net Graph N**
  - **places: represent distributed state by holding tokens**
    - **marking (state) M is an n-vector (m_1,m_2,m_3…), where m_i is the non-negative number of tokens in place p_i.**
    - **initial marking (M_0) is initial state**
  - **transitions: represent actions/events**
    - **enabled transition: enough tokens in predecessors**
    - **firing transition: modifies marking**
- **…and an initial marking M_0.**

Places/Transitions: conditions/events

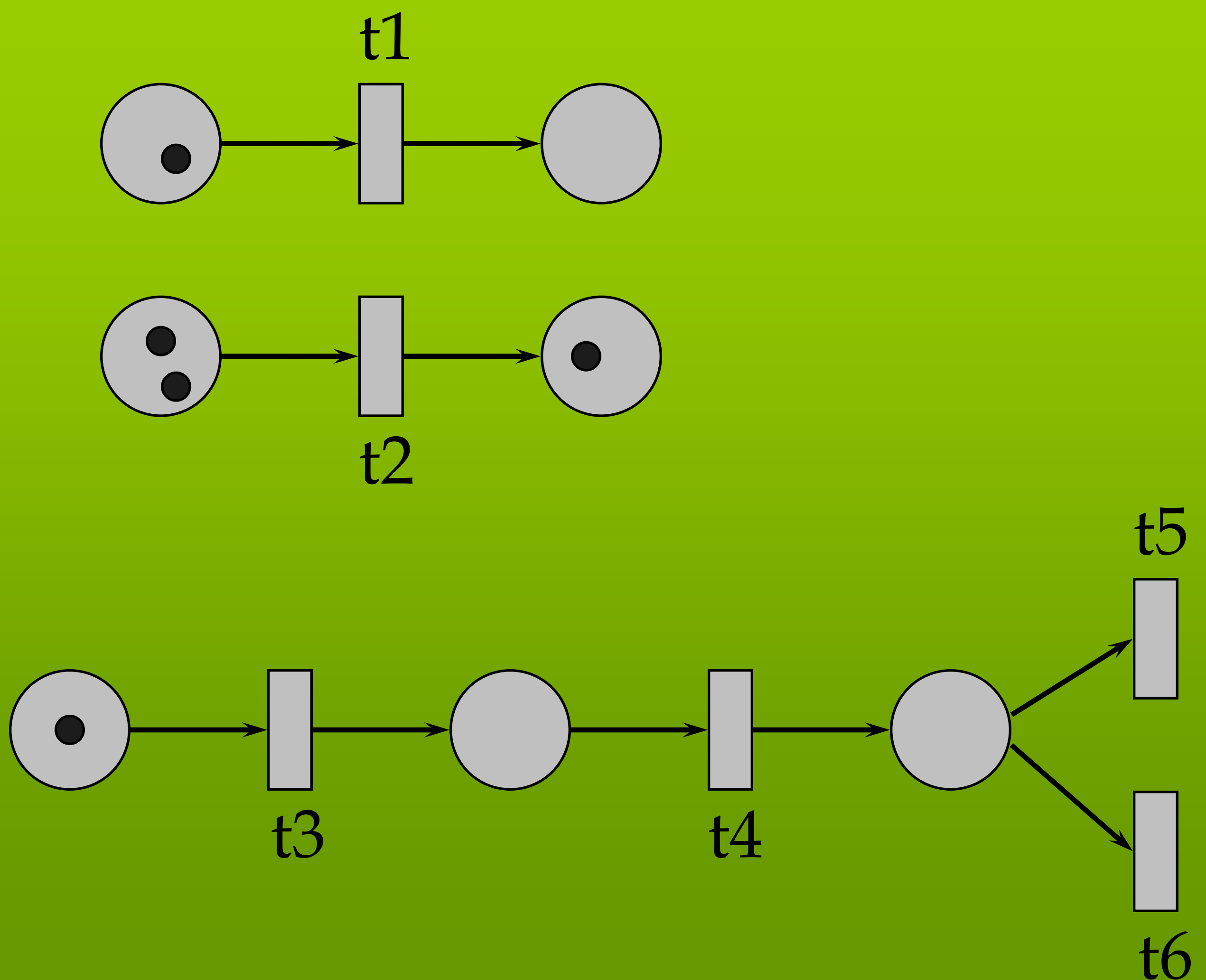


Transition firing rule

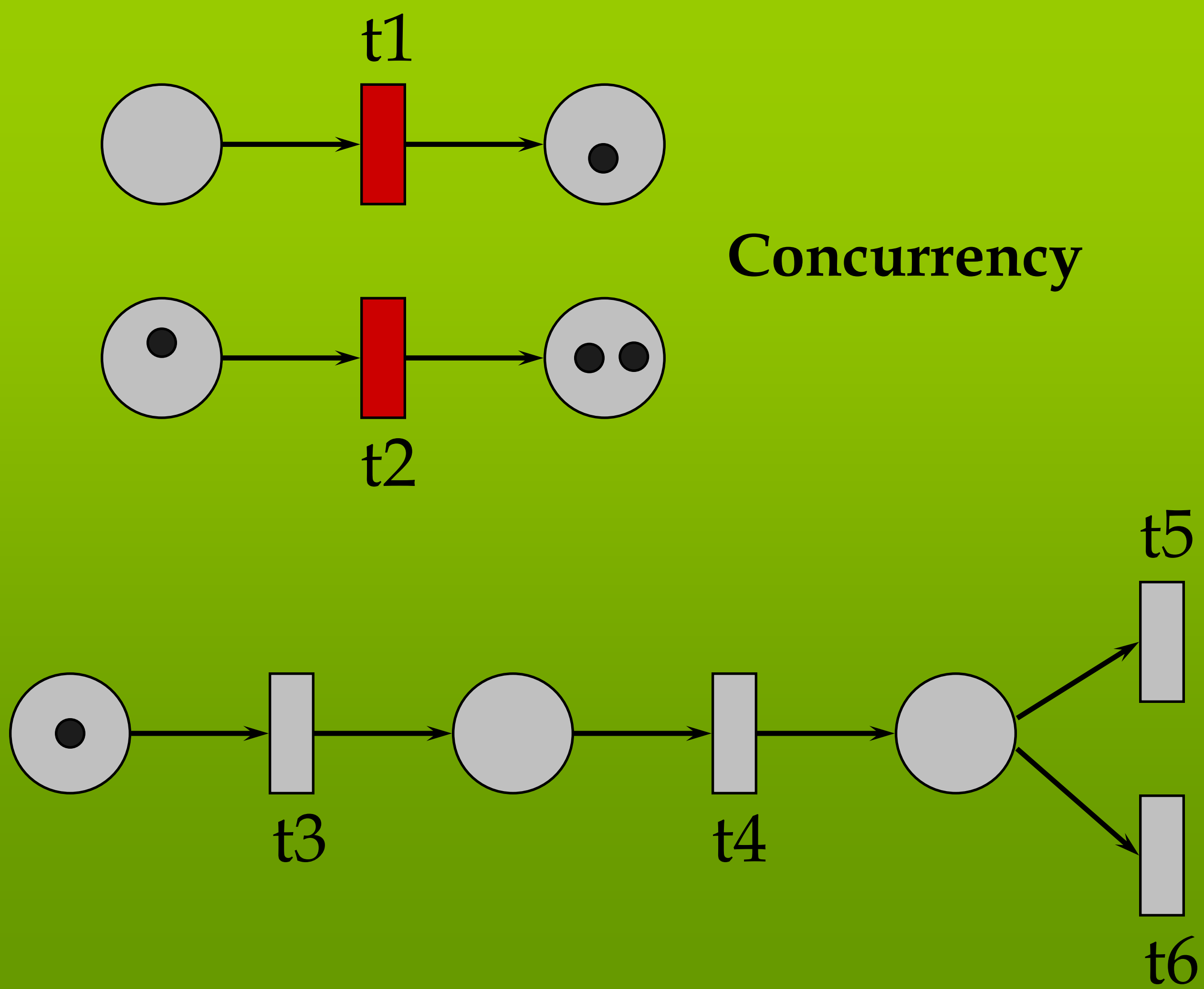
- **A marking is changed according to the following rules:**
  - **A transition is enabled if there are enough tokens in each input place**
  - **An enabled transition may or may not fire**
  - **The firing of a transition modifies marking by consuming tokens from the input places and producing tokens in the output places**



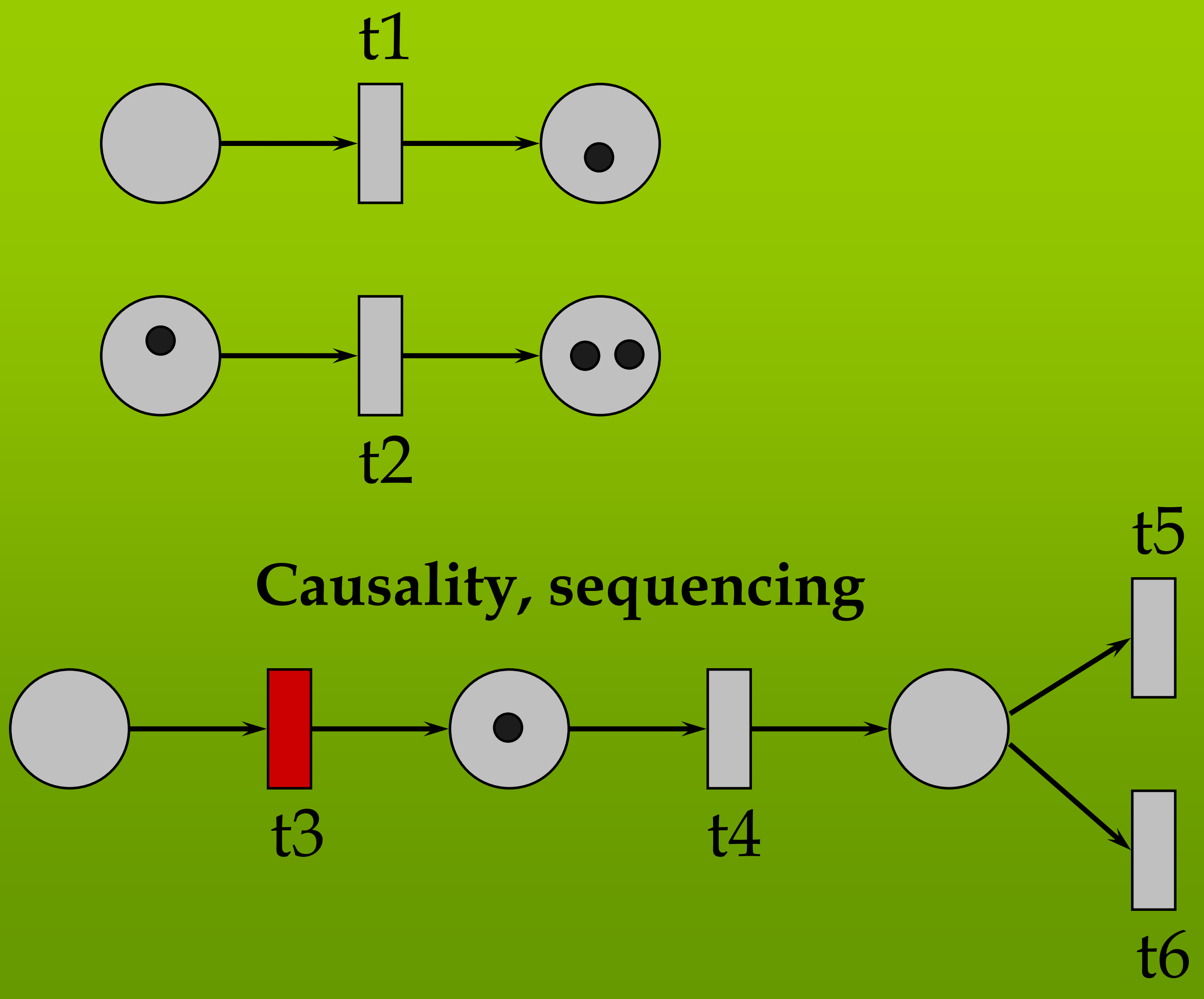
Concurrency, causality, choice



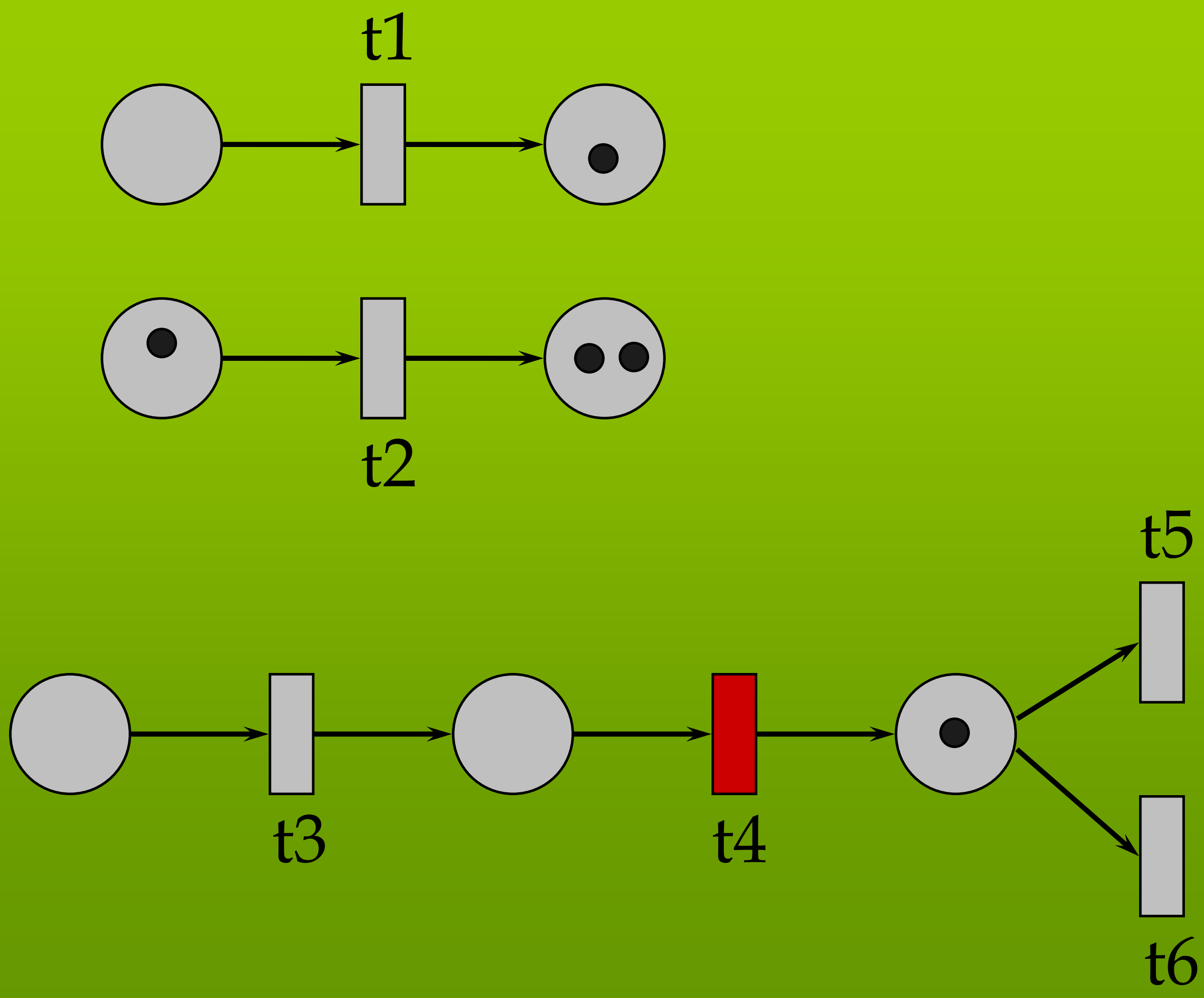
Concurrency, causality, choice



Concurrency, causality, choice

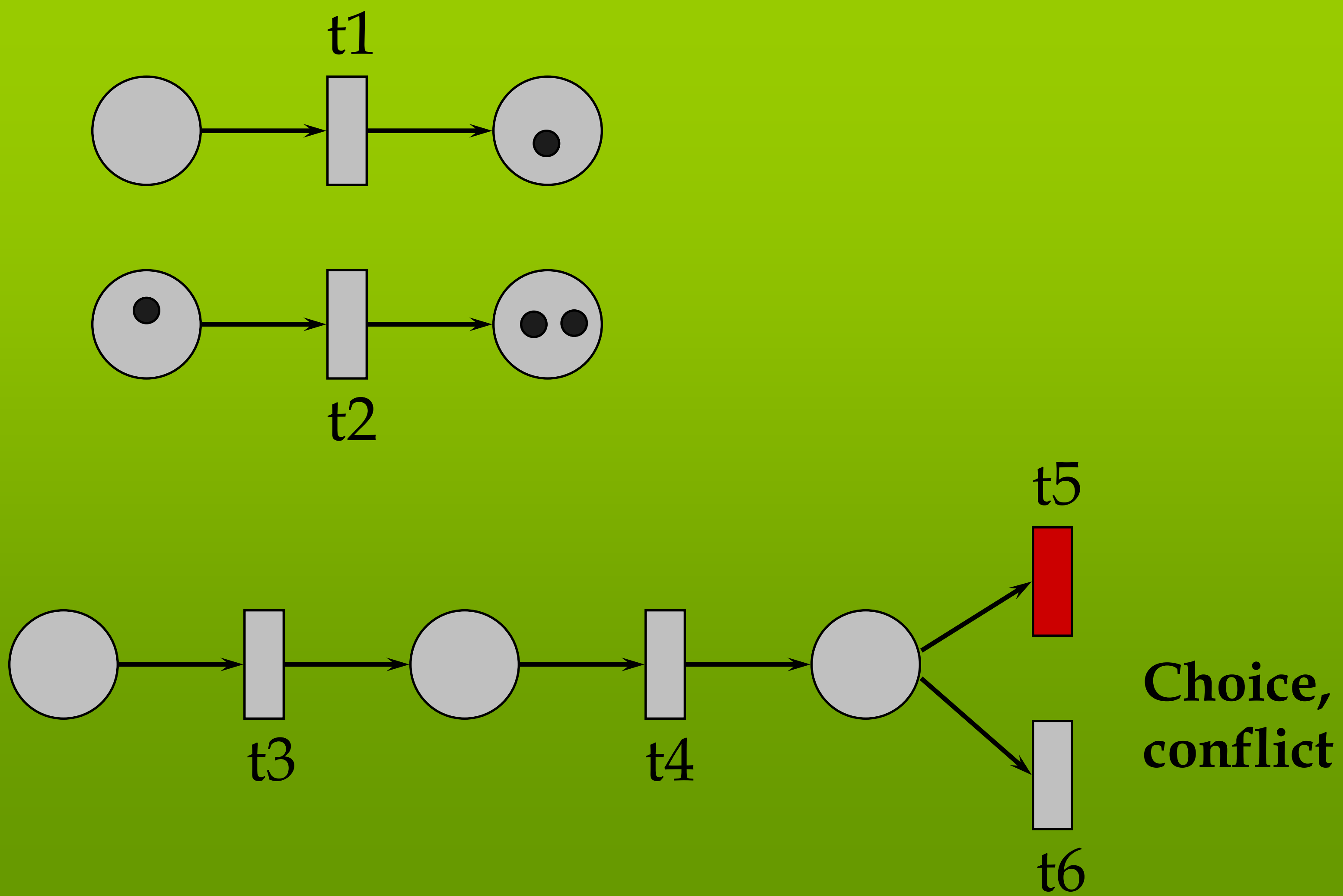


Concurrency, causality, choice

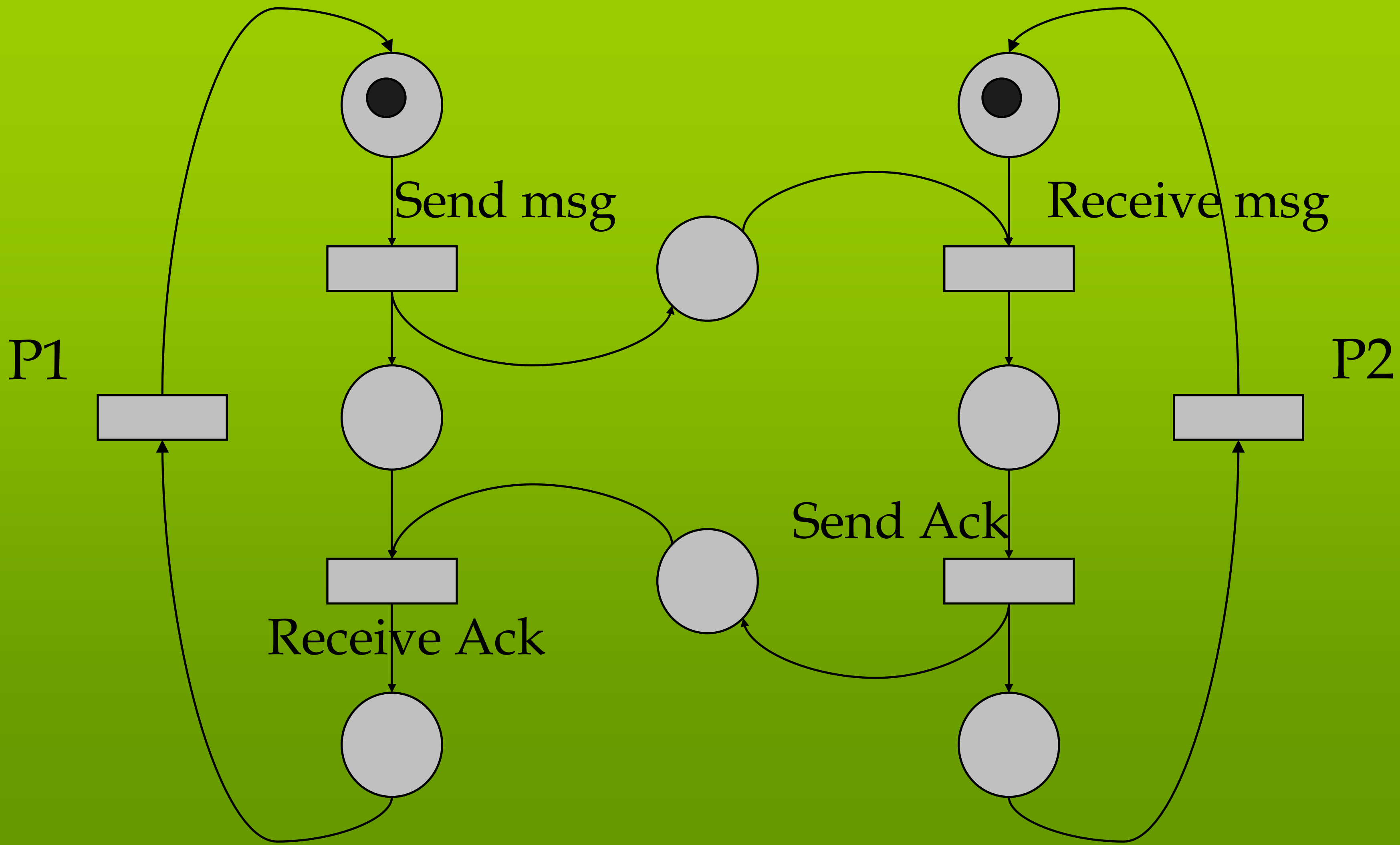


Choice, conflict

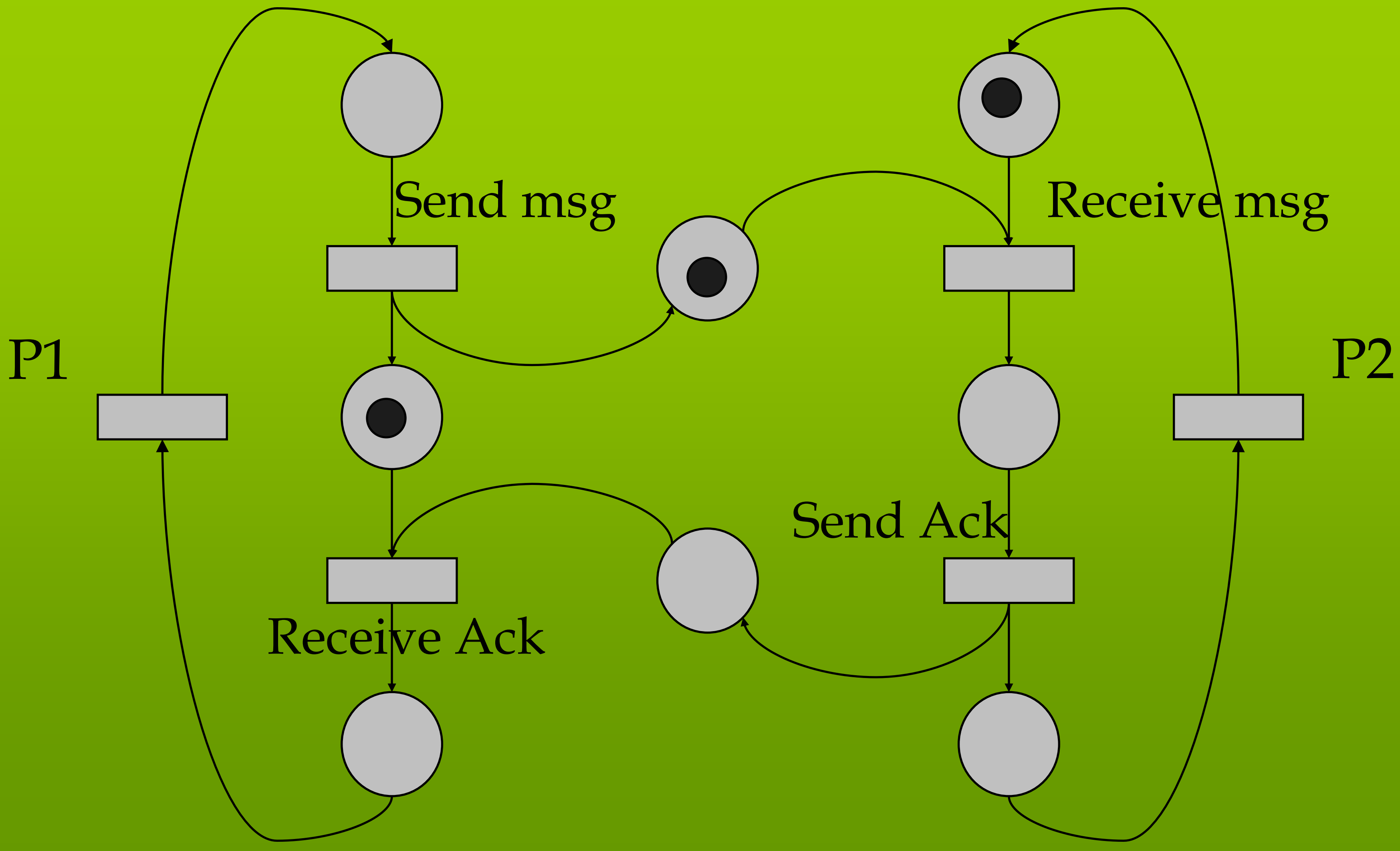
Concurrency, causality, choice



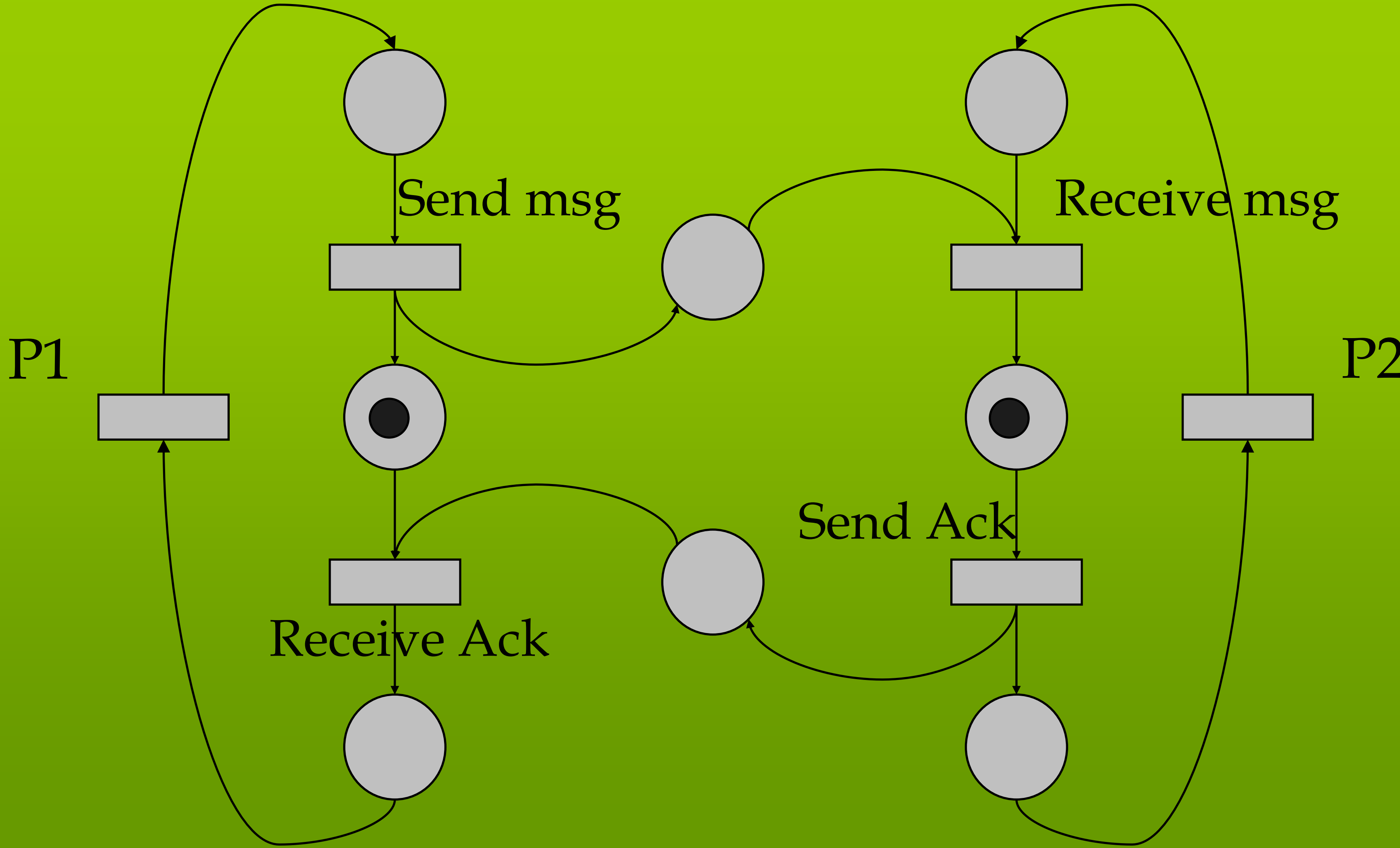
Communication Protocol



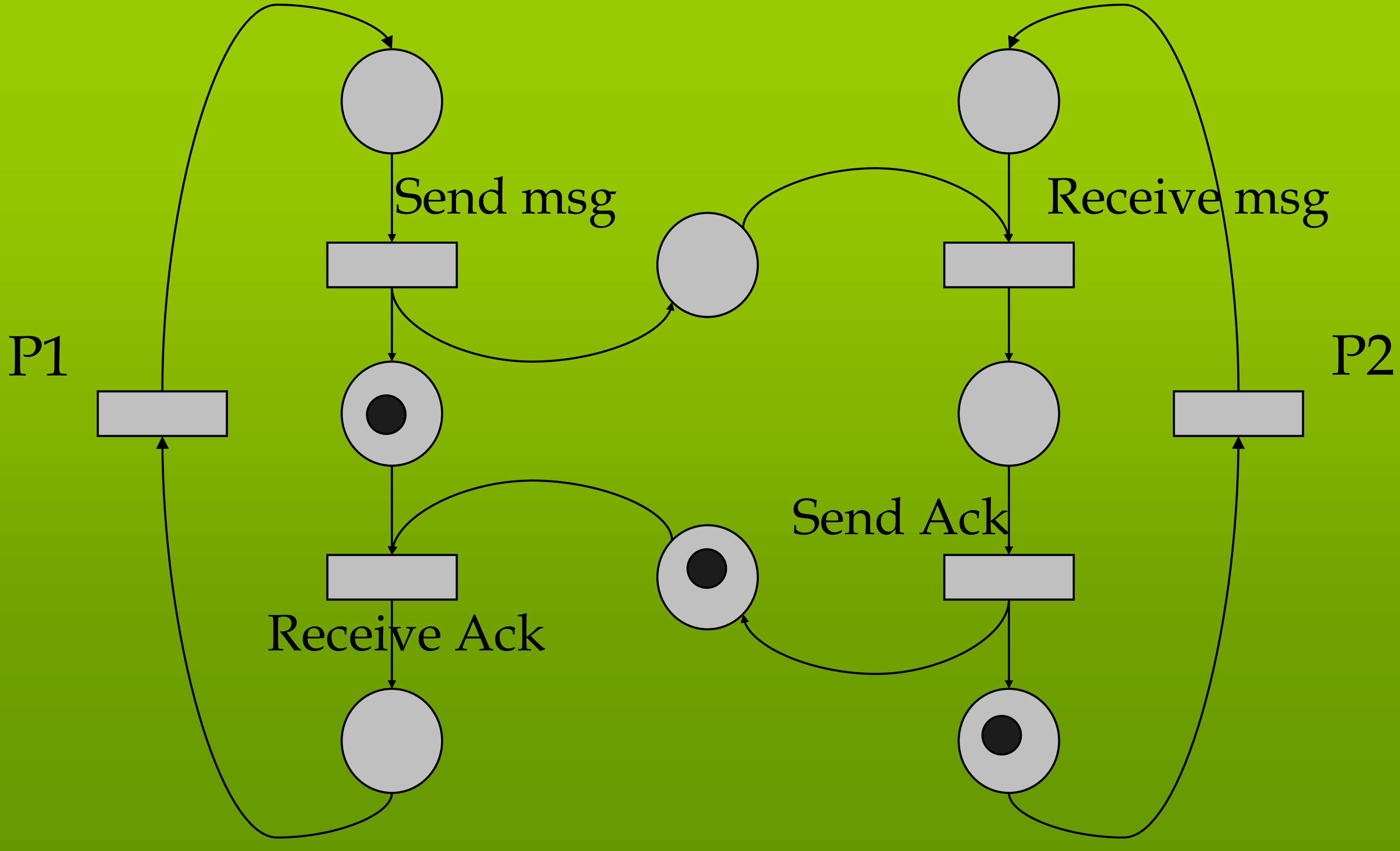
Communication Protocol



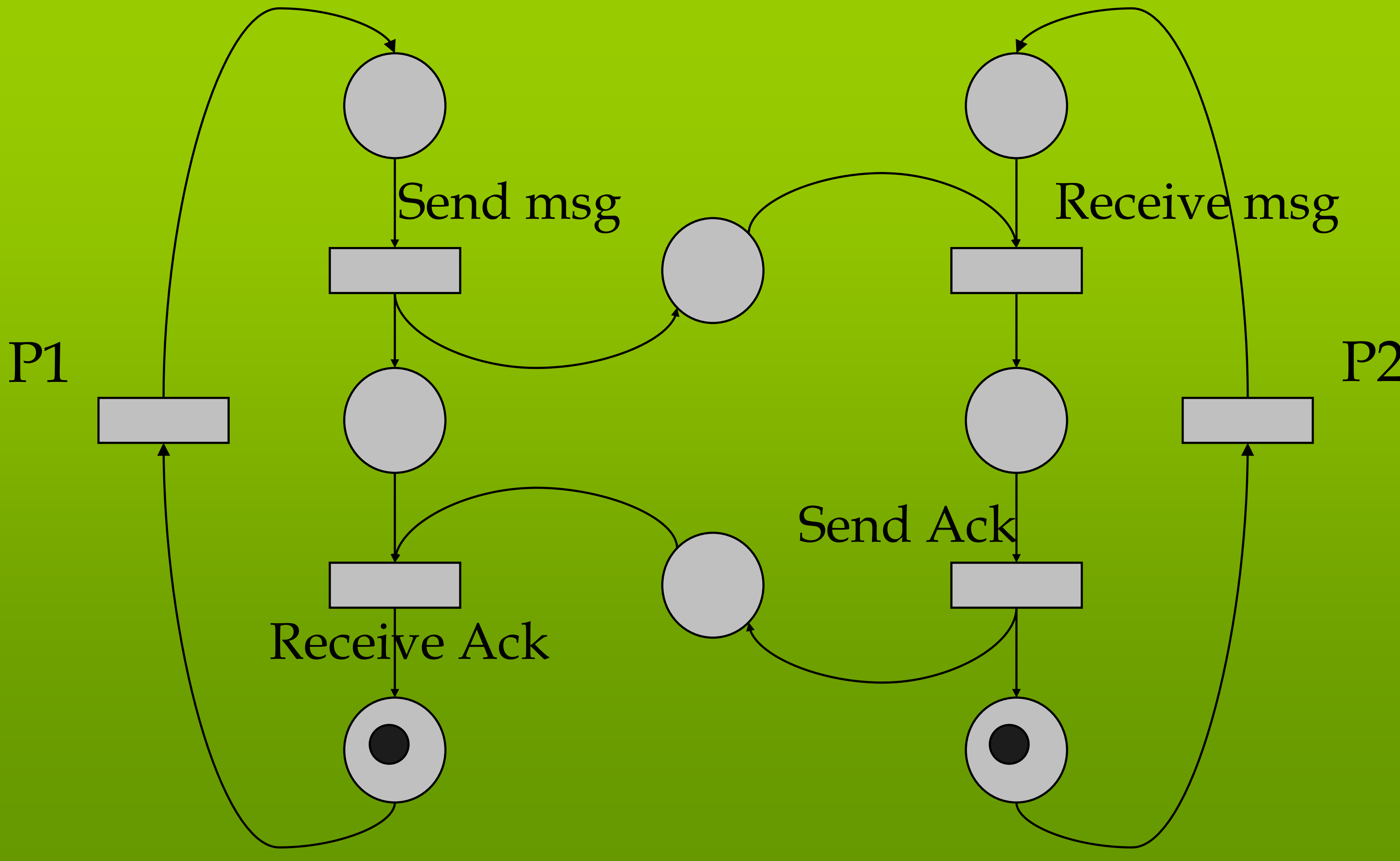
Communication Protocol



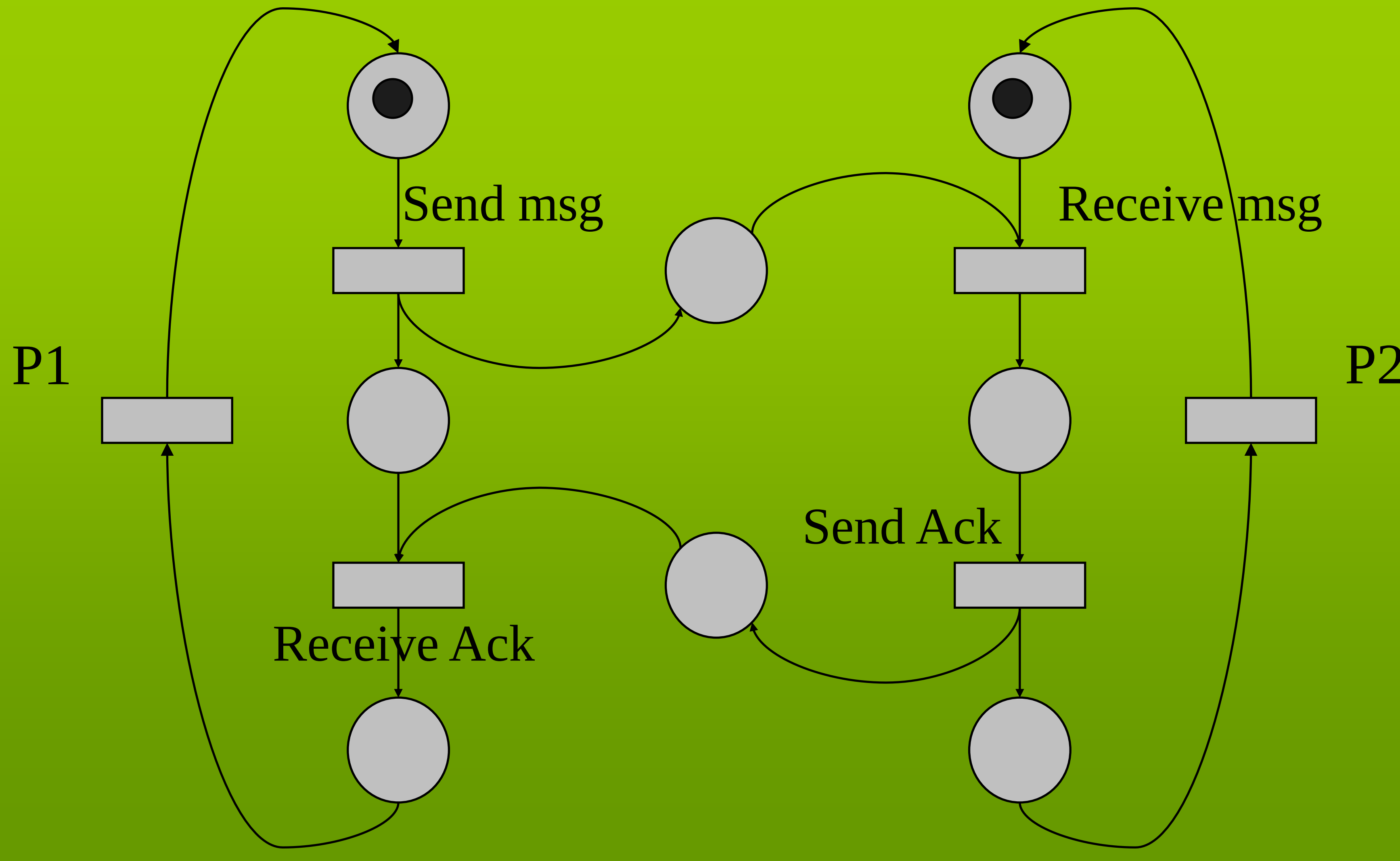
Communication Protocol



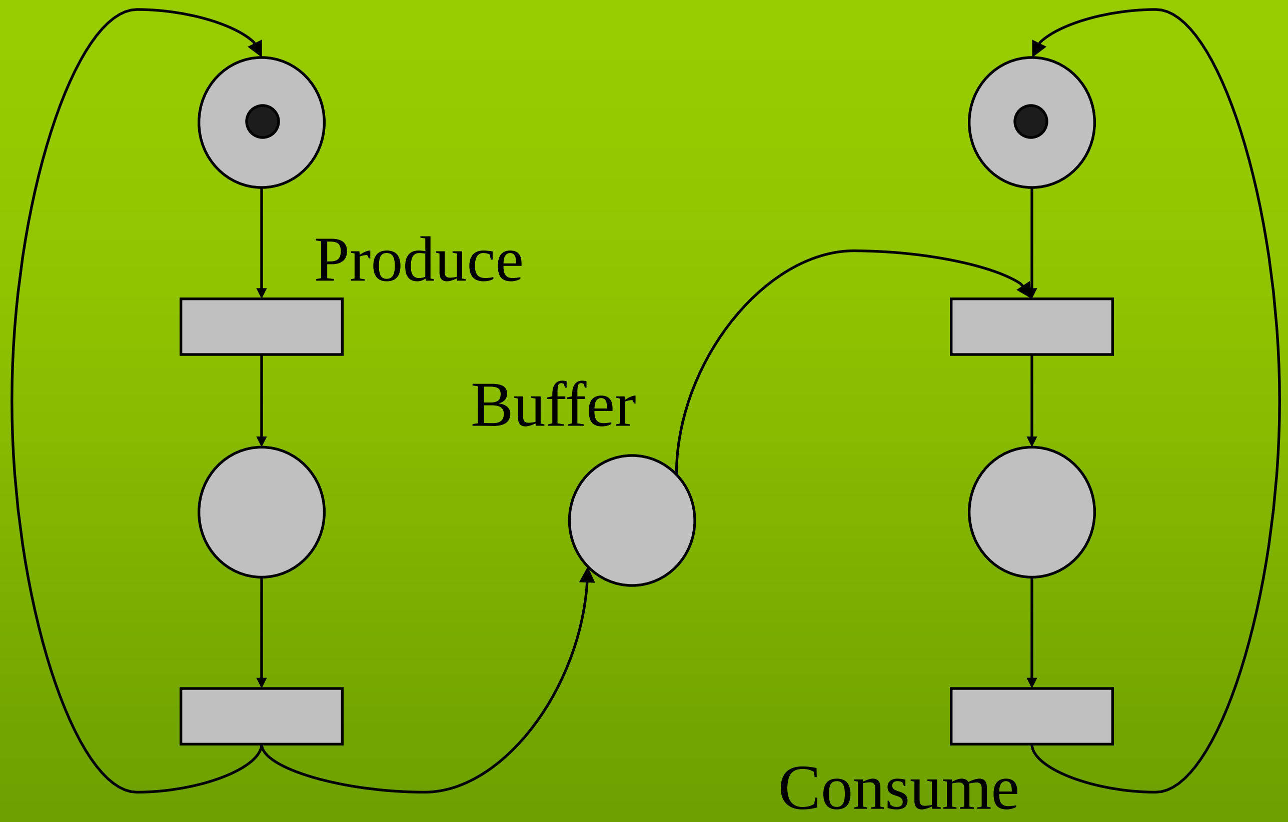
Communication Protocol



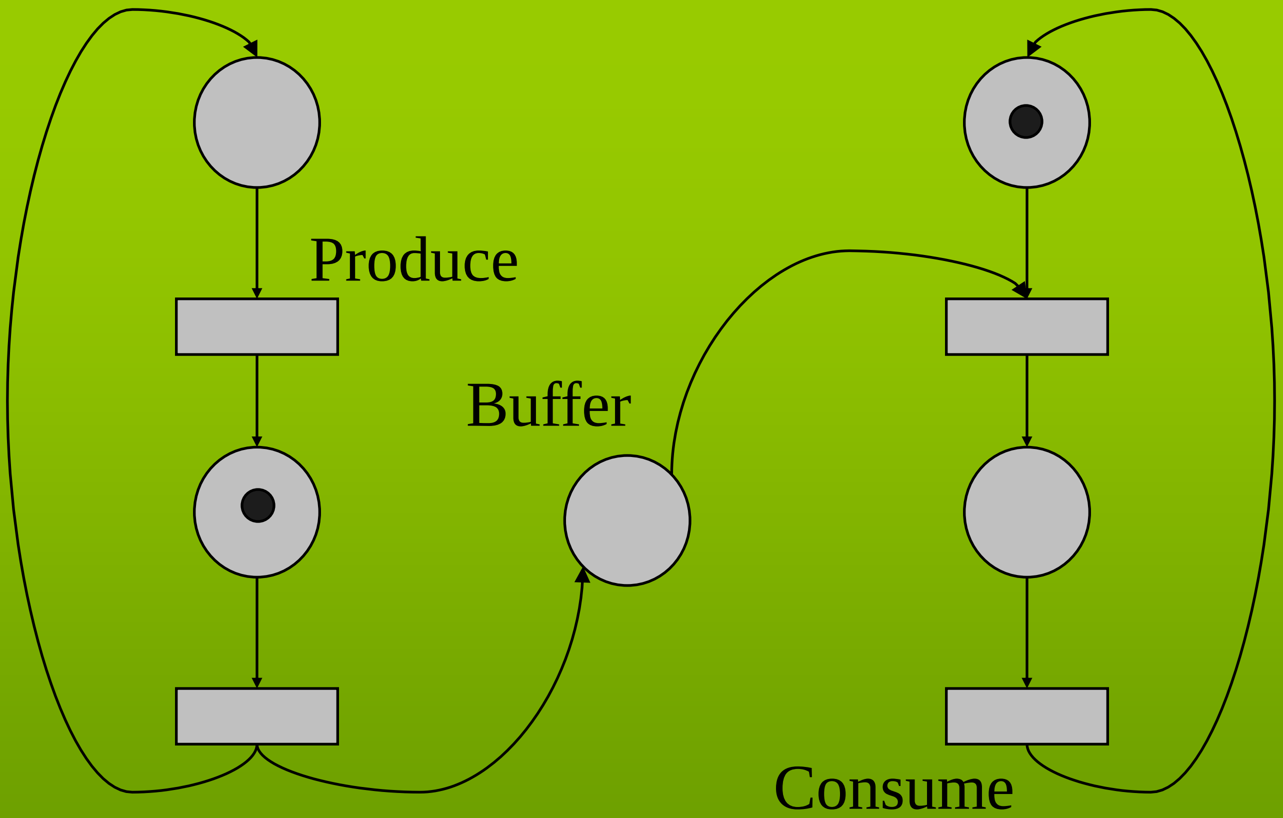
Communication Protocol



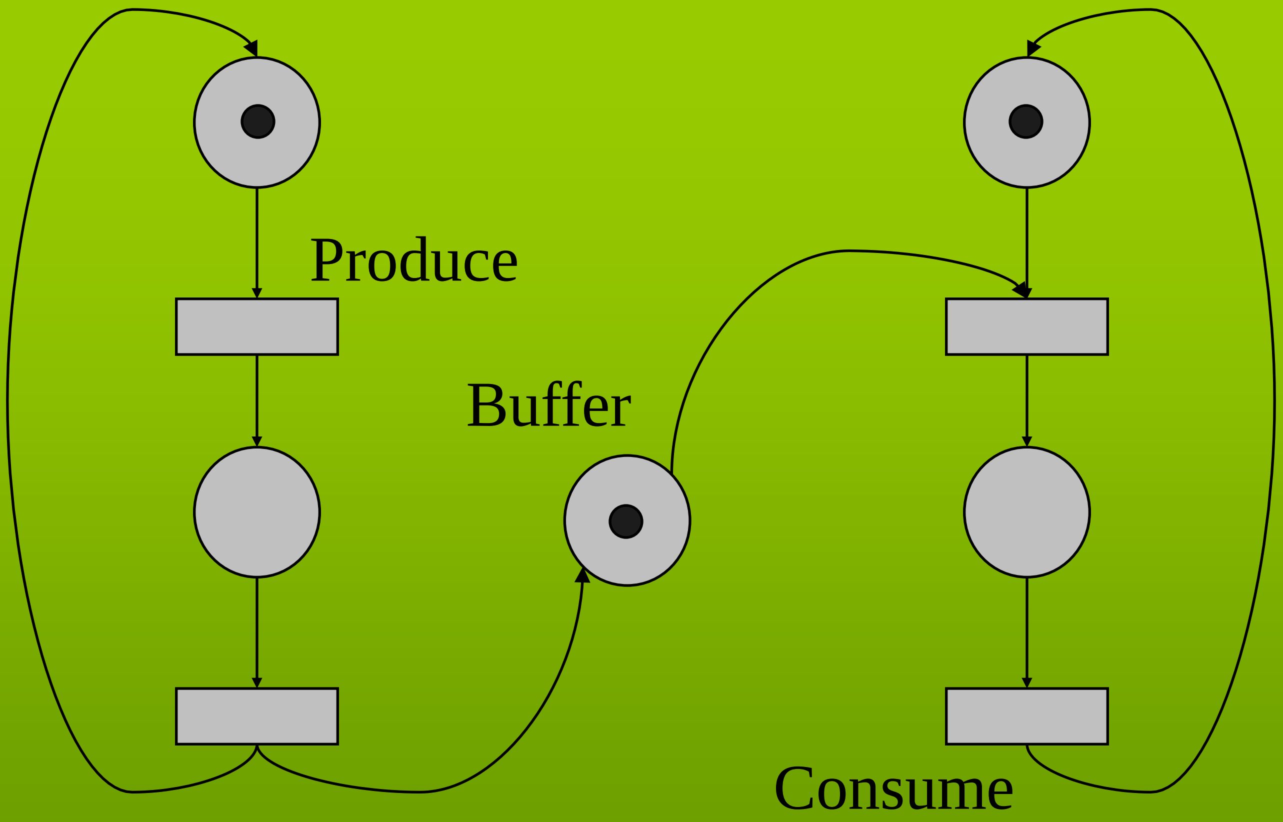
Producer-Consumer Problem



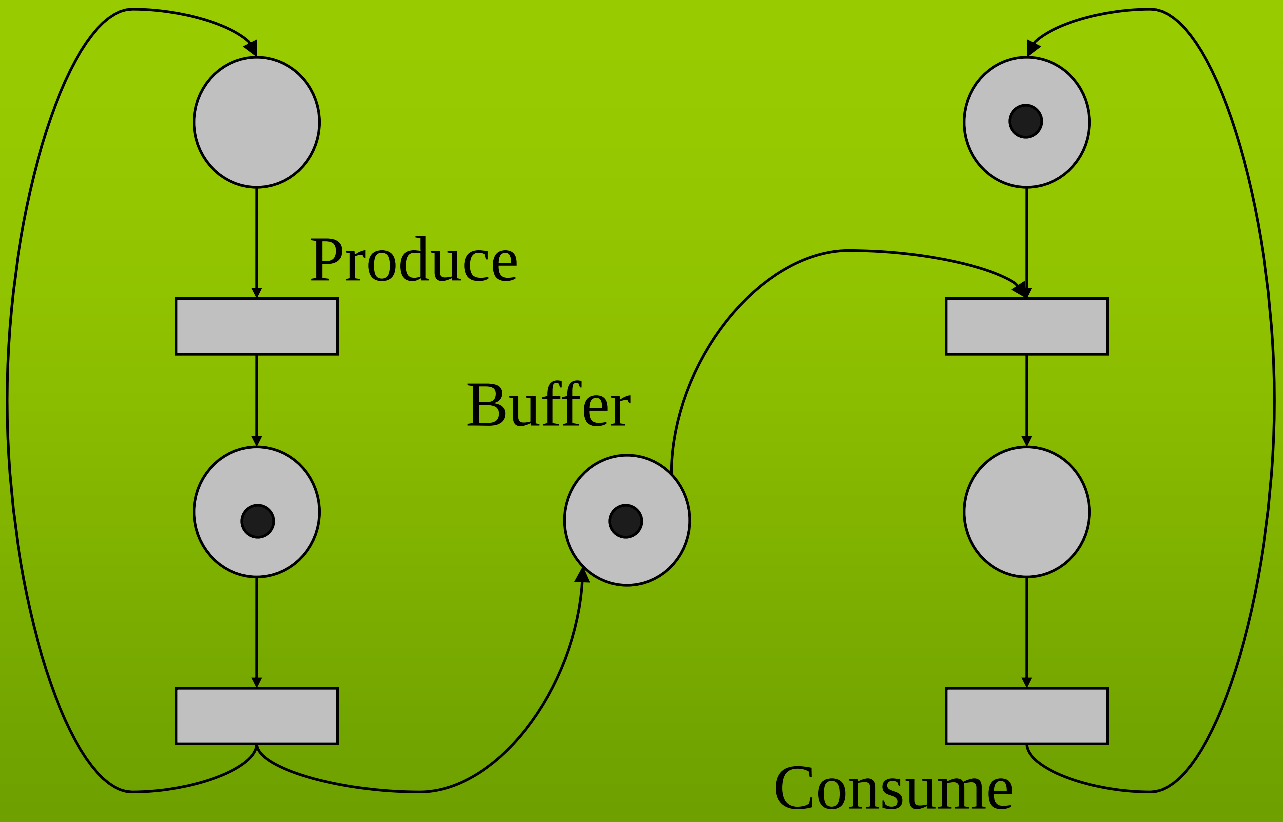
Producer-Consumer Problem



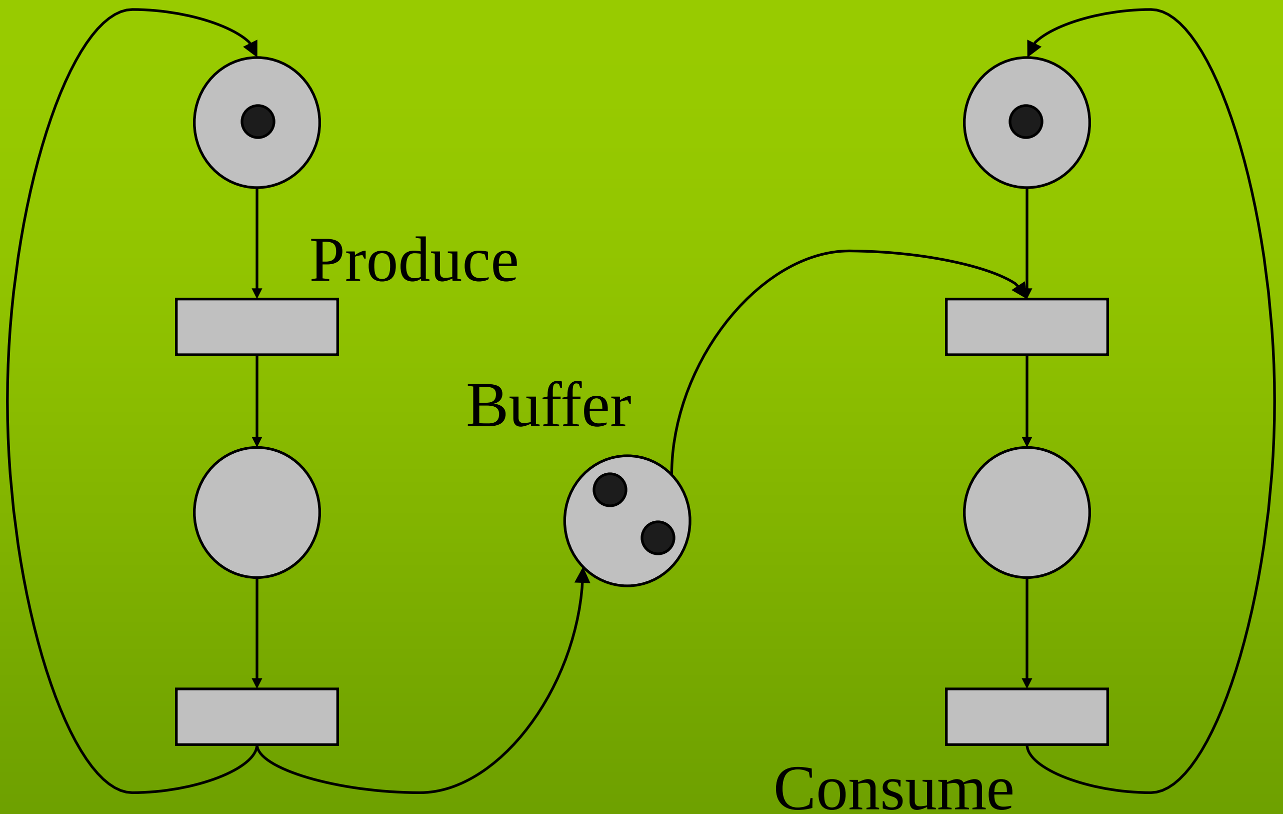
Producer-Consumer Problem



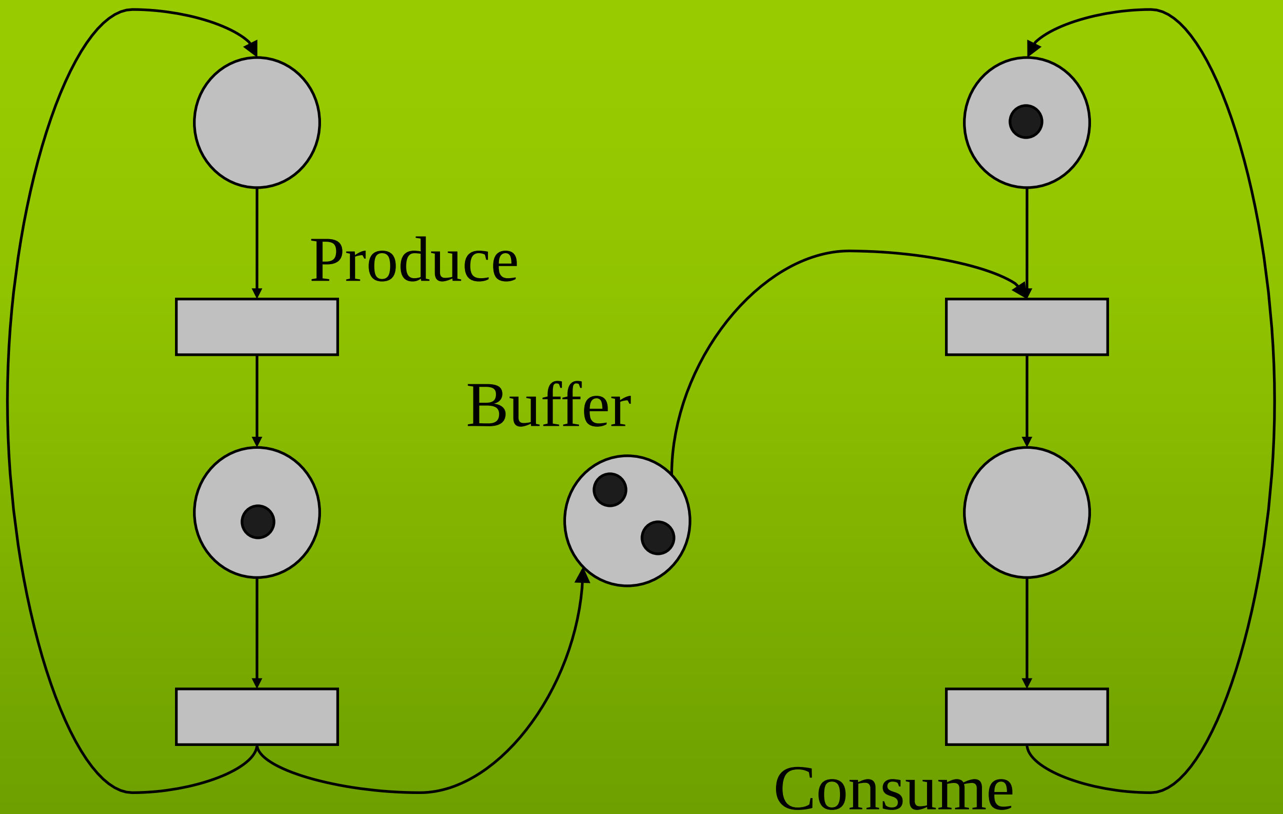
Producer-Consumer Problem



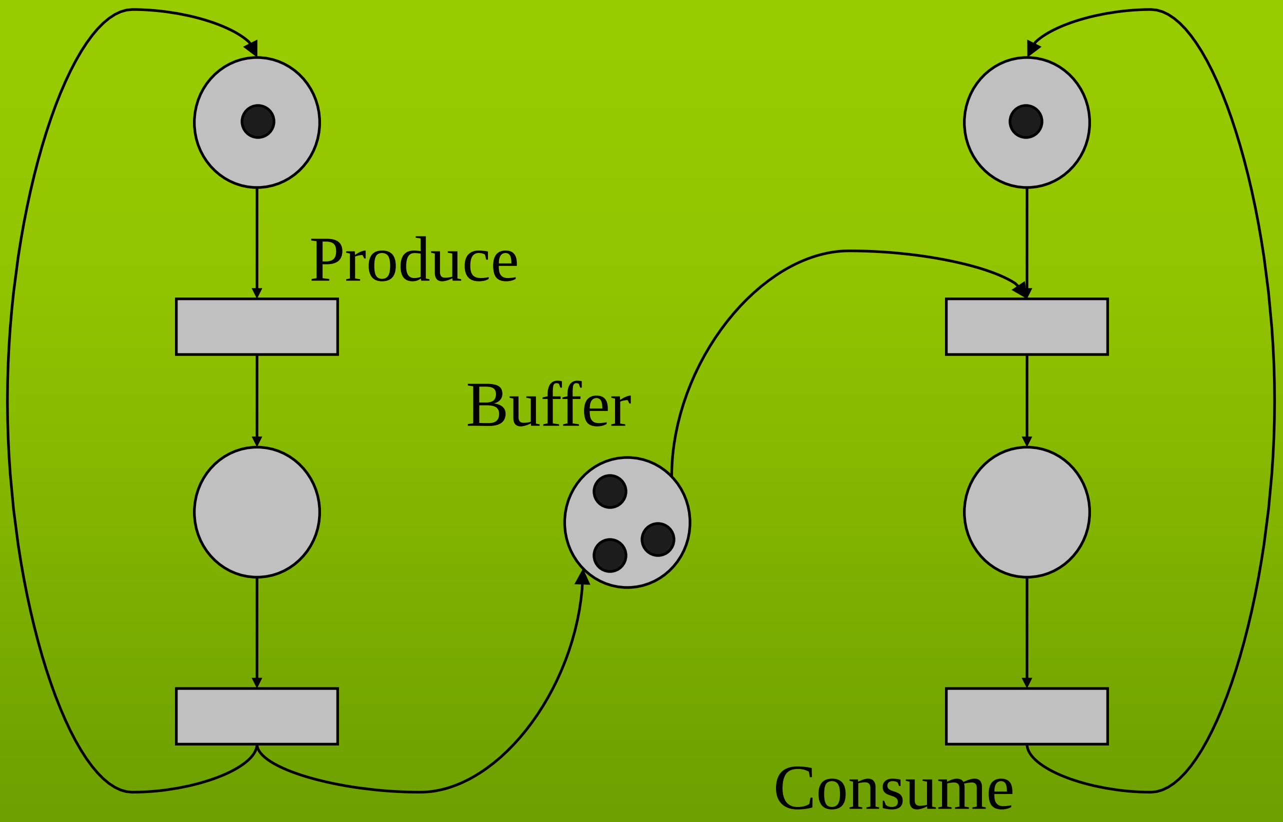
Producer-Consumer Problem



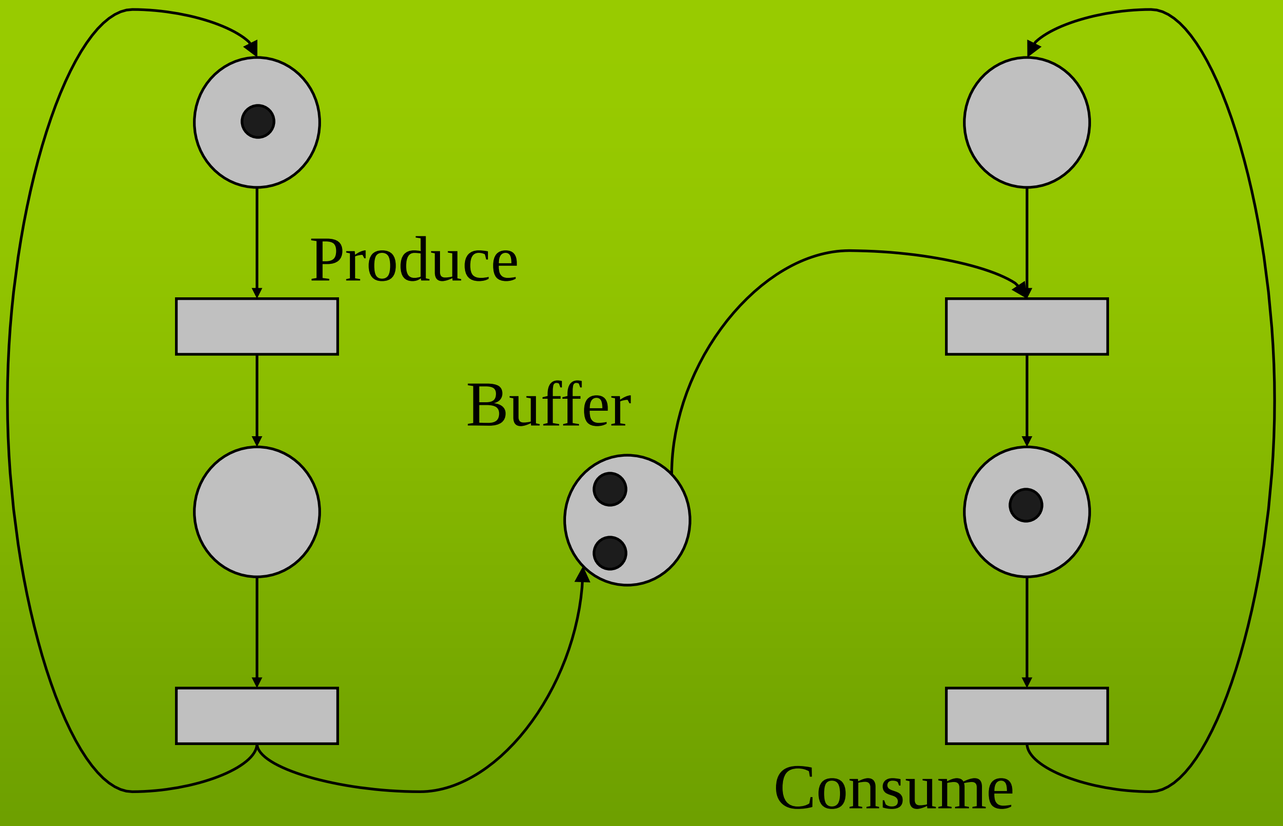
Producer-Consumer Problem



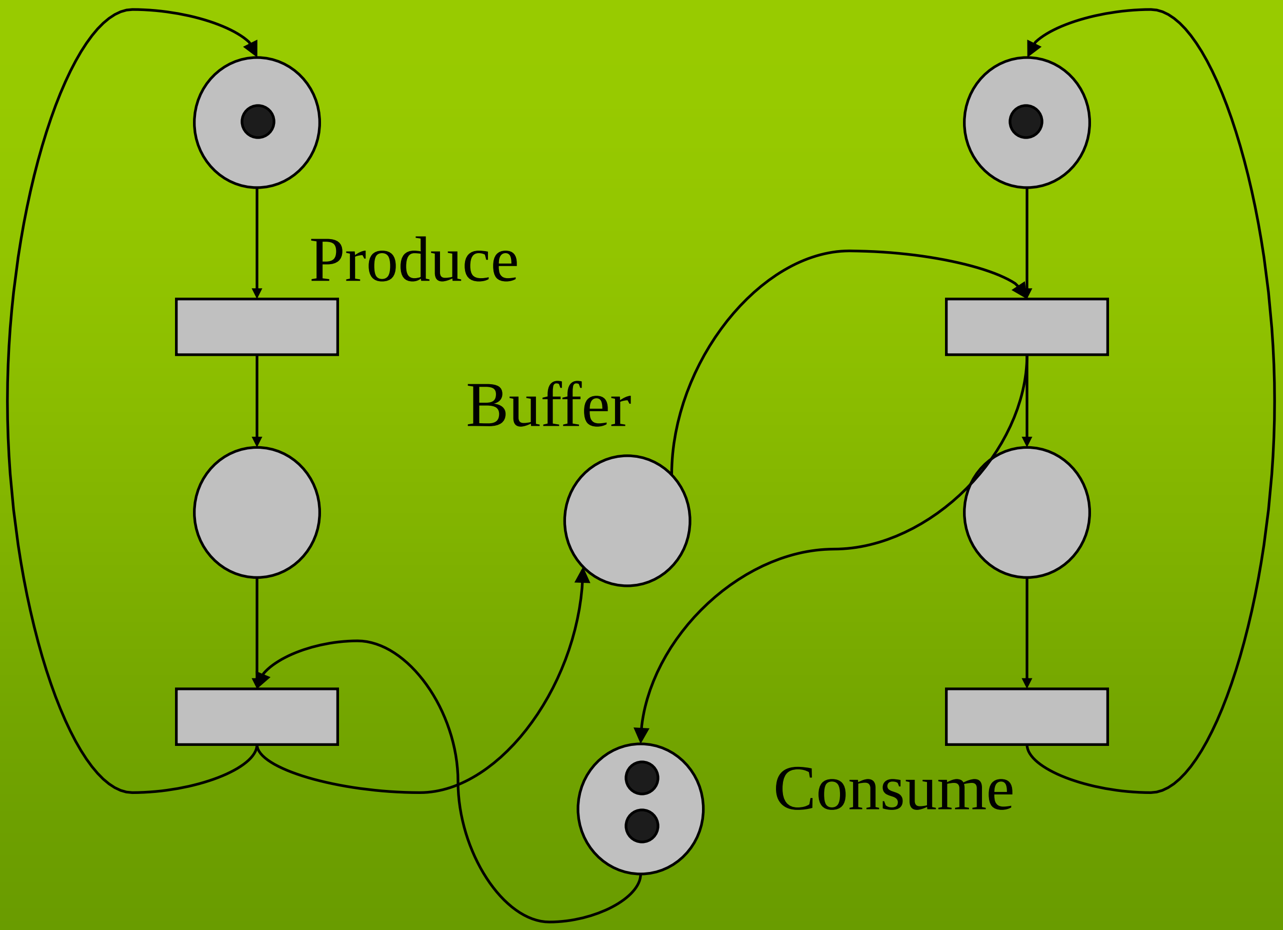
Producer-Consumer Problem



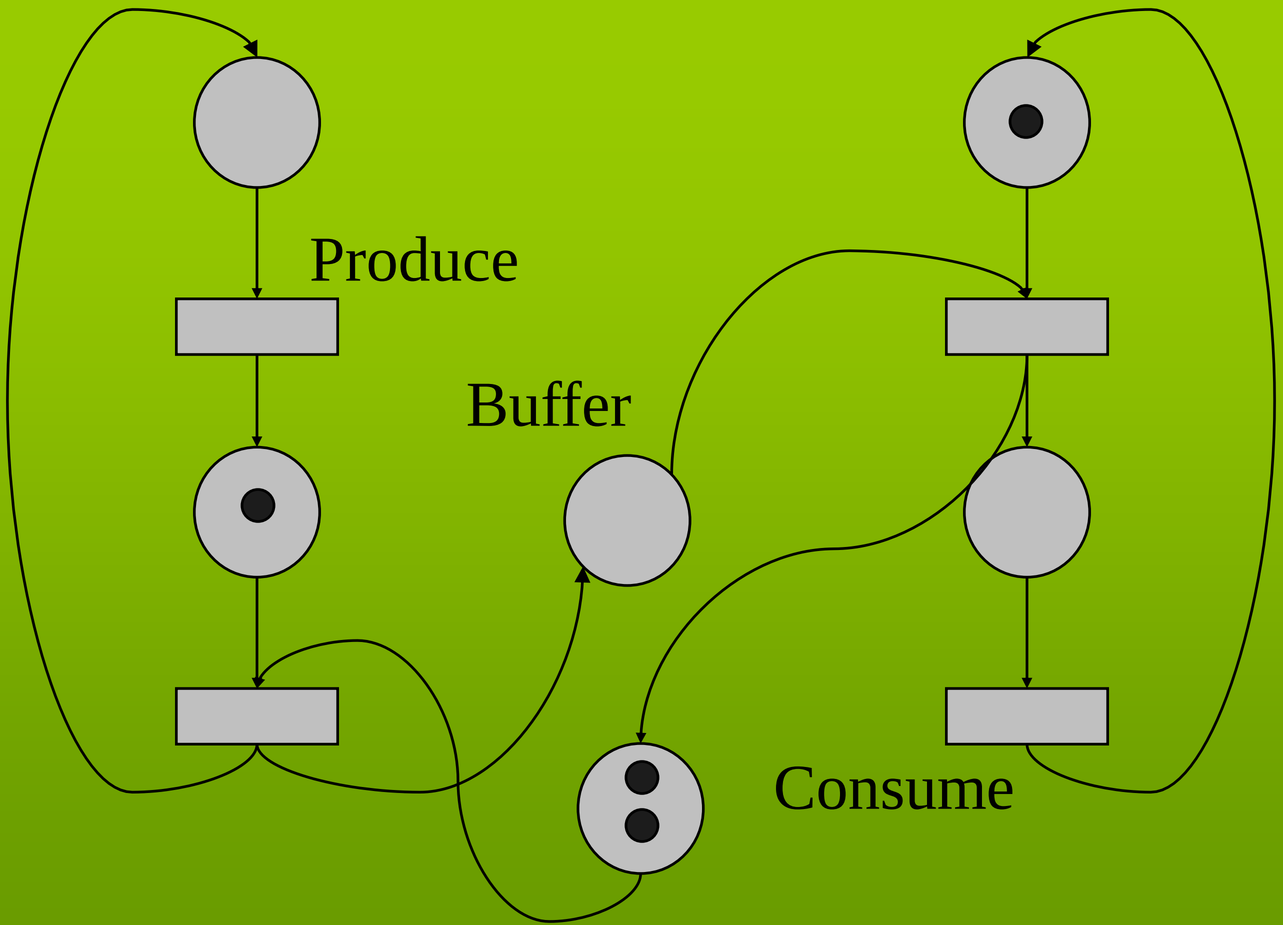
Producer-Consumer Problem



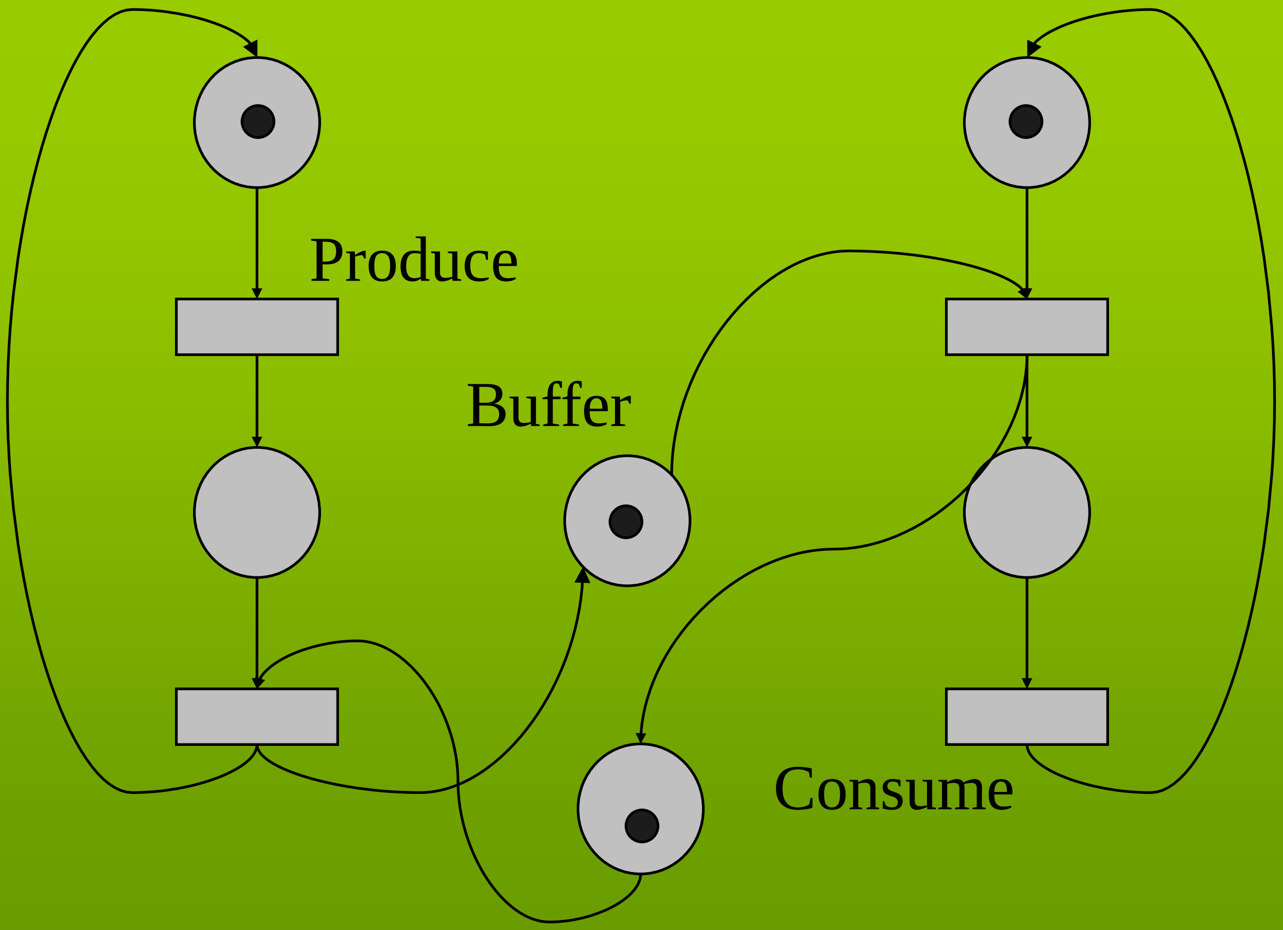
Producer-Consumer Problem



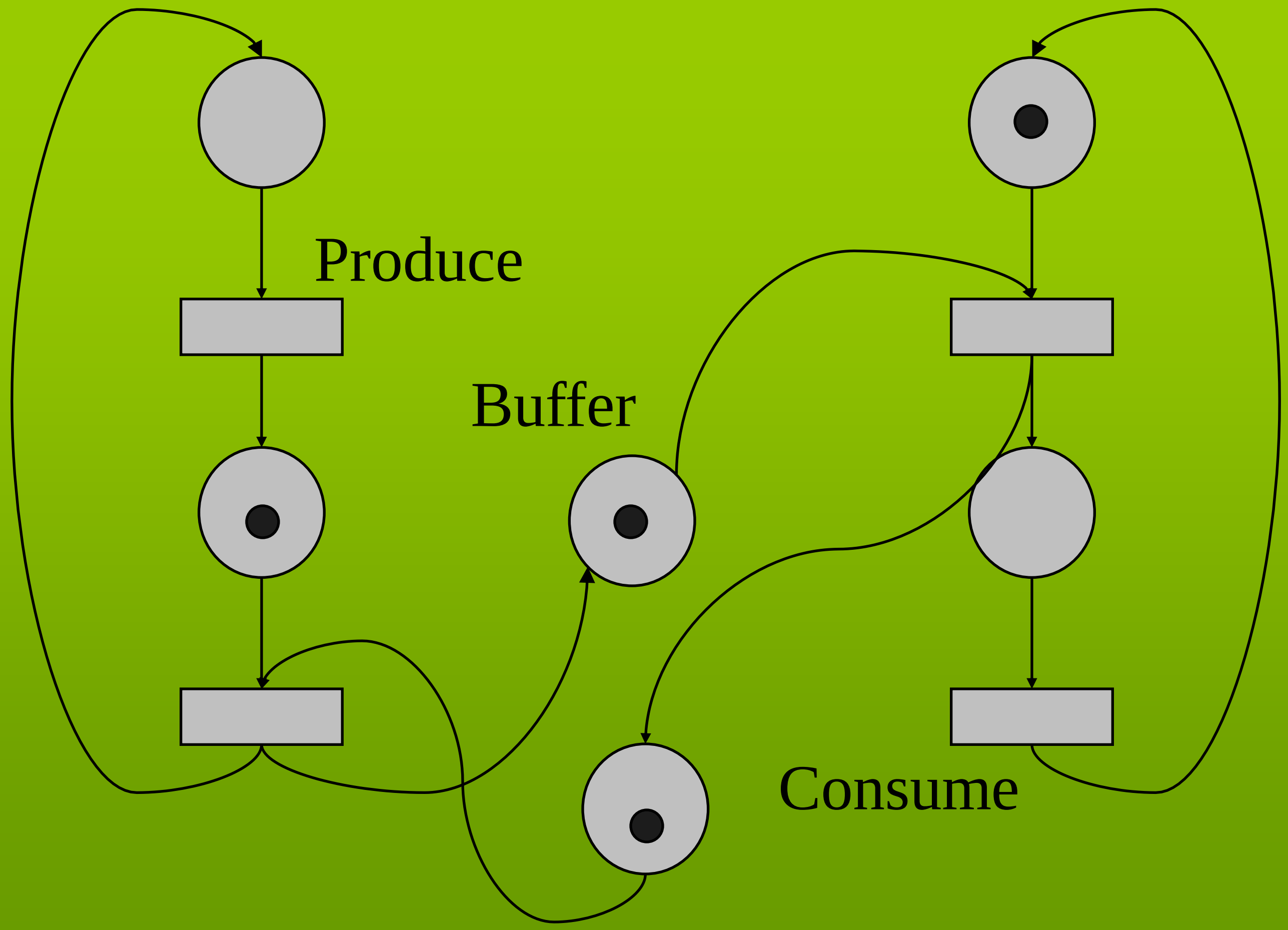
Producer-Consumer Problem



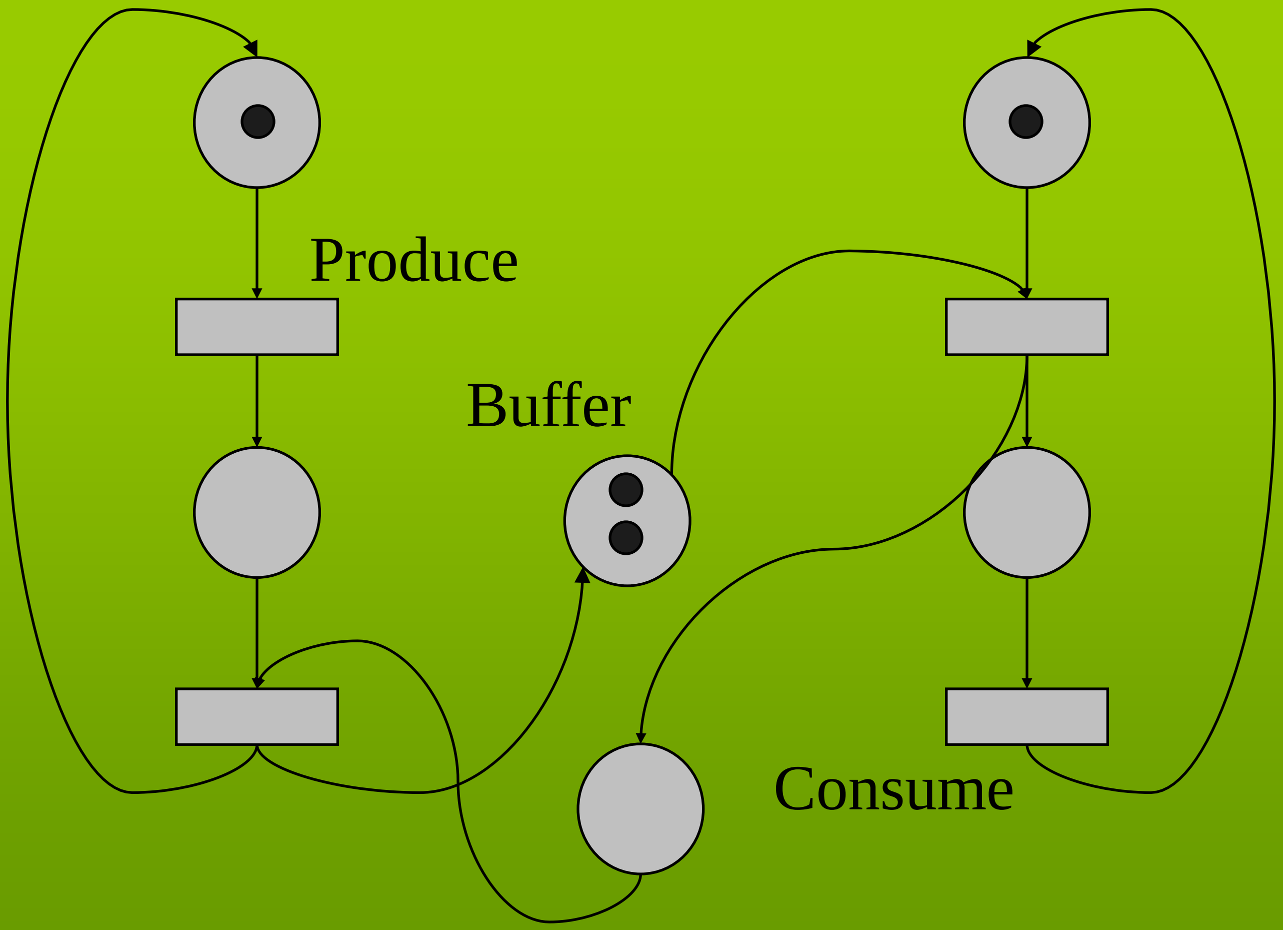
Producer-Consumer Problem



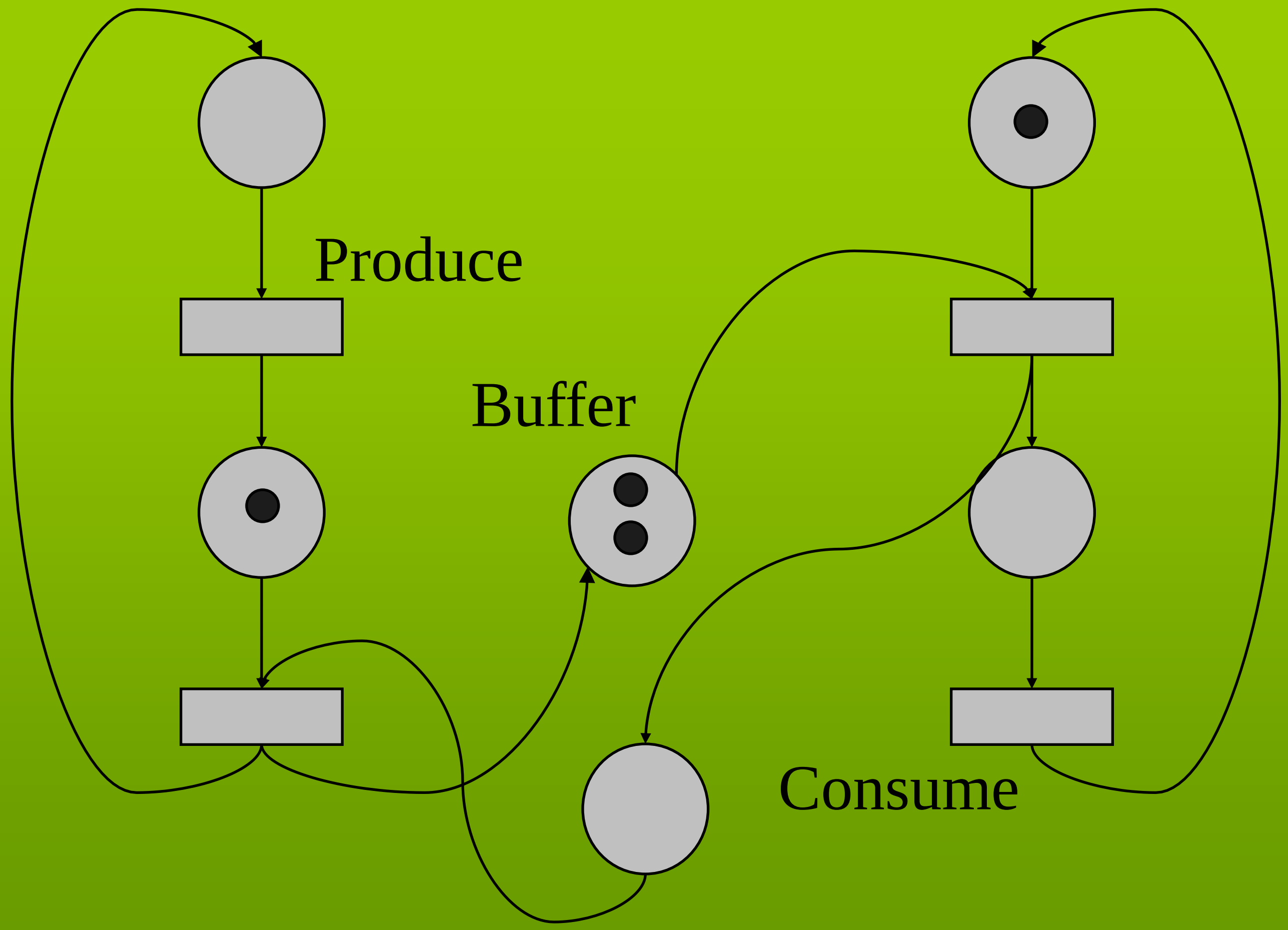
Producer-Consumer Problem



Producer-Consumer Problem



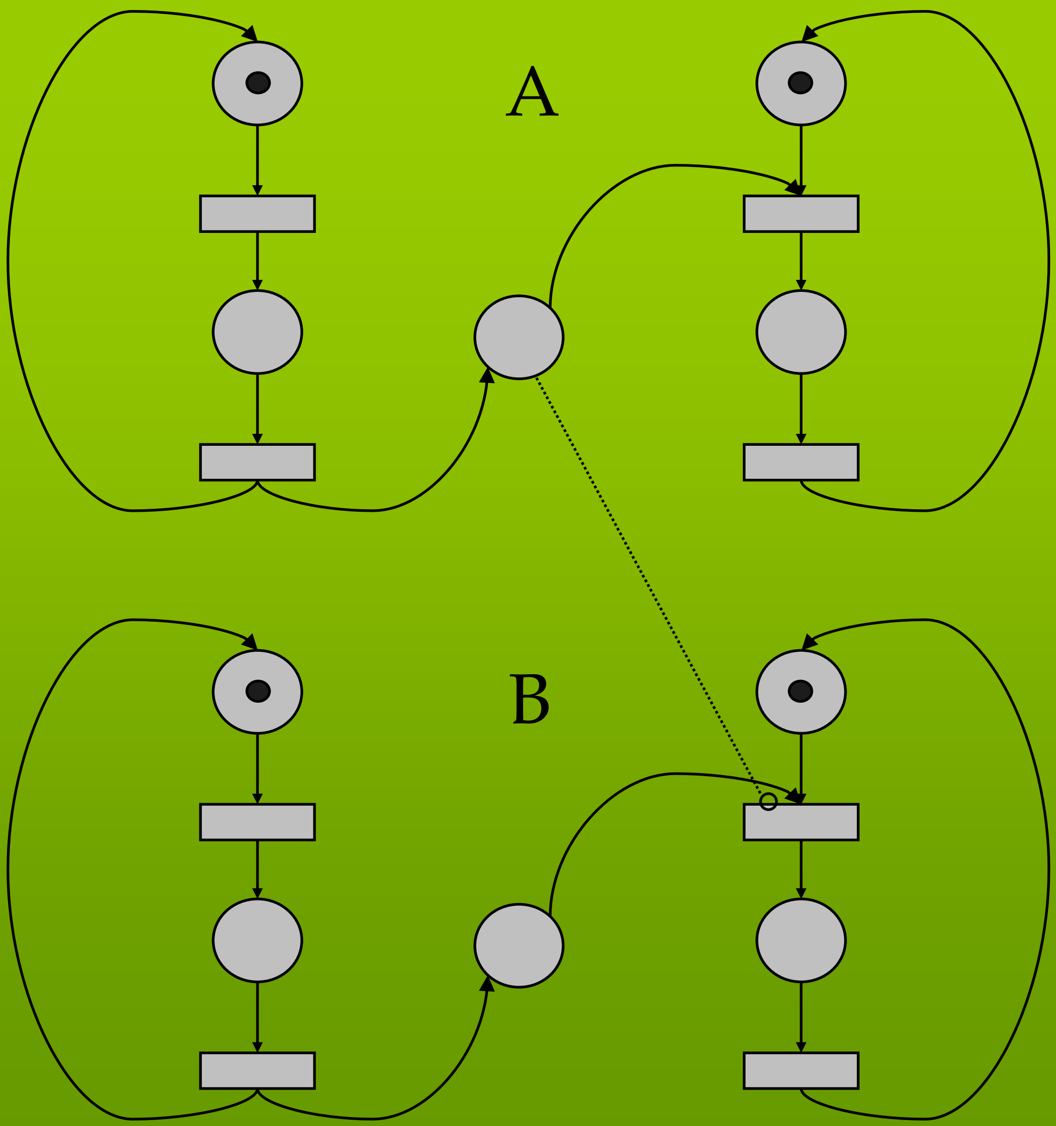
Producer-Consumer Problem



Producer-Consumer with priority

Consumer B can consume only if buffer A is empty

Inhibitor arcs

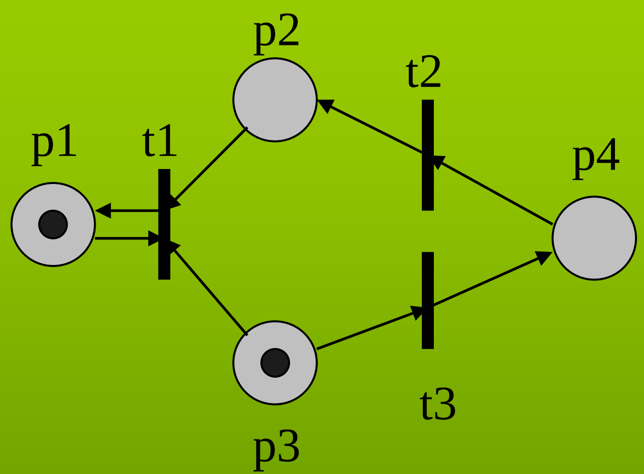


PN properties

- **Behavioral: depend on the initial marking (most interesting)**
  - **Reachability**
  - **Boundedness**
  - **Schedulability**
  - **Liveness**
  - **Conservation**
- **Structural: do not depend on the initial marking (often too restrictive)**
  - **Consistency**
  - **Structural boundedness**

Reachability

- **Marking M is reachable from marking M_0 if there exists a sequence of firings $\sigma = M_0\ t_1\ M_1\ t_2\ M_2 \ldots M$ that transforms M_0 to M.**
- **The reachability problem is decidable.**



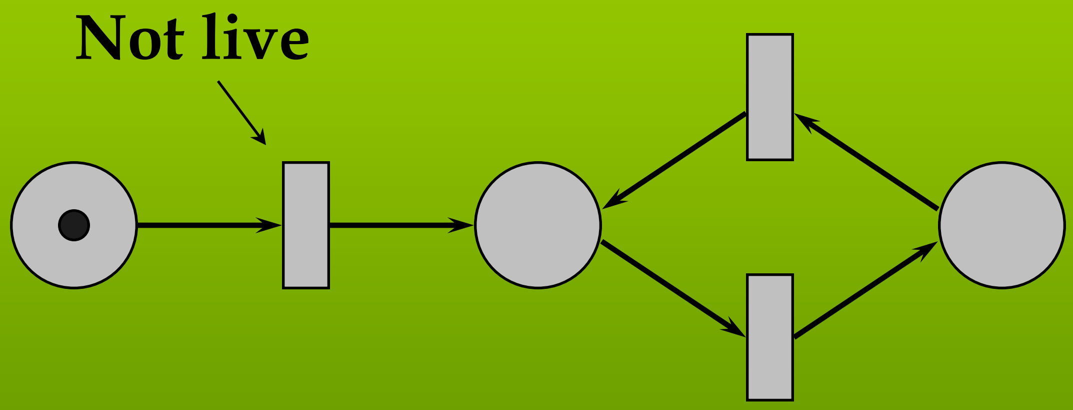
$M_0 = (1,0,1,0)$

$M = (1,1,0,0)$

$$M_0 = (1,0,1,0) \xrightarrow{t3} M_1 = (1,0,0,1) \xrightarrow{t2} M = (1,1,0,0)$$

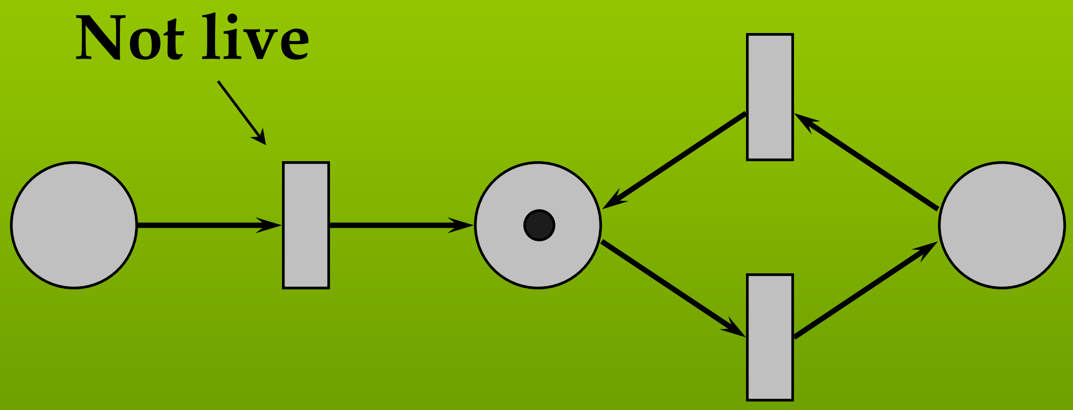
Liveness

- **Liveness: from any marking any transition can become fireable**
  - **Liveness implies deadlock freedom, not viceversa**



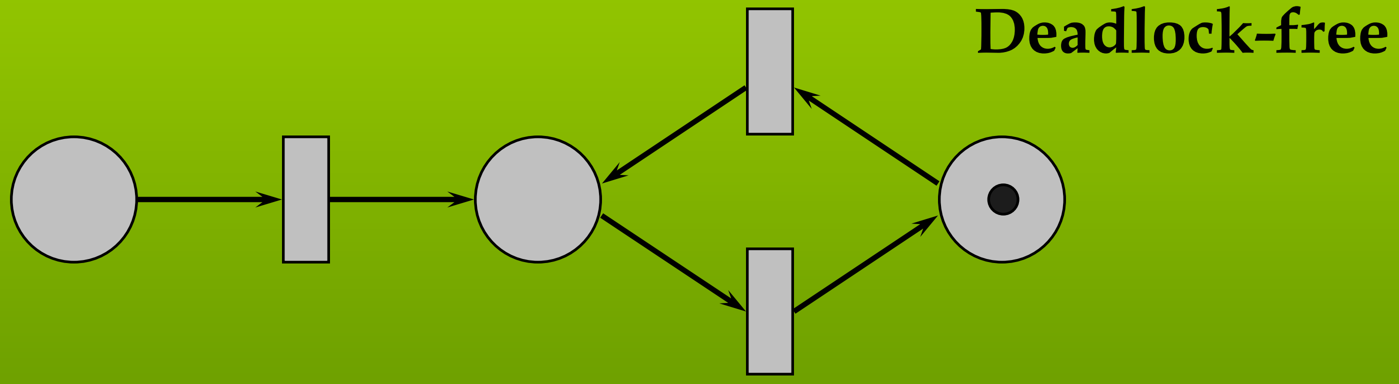
Liveness

- **Liveness: from any marking any transition can become fireable**
  - **Liveness implies deadlock freedom, not viceversa**



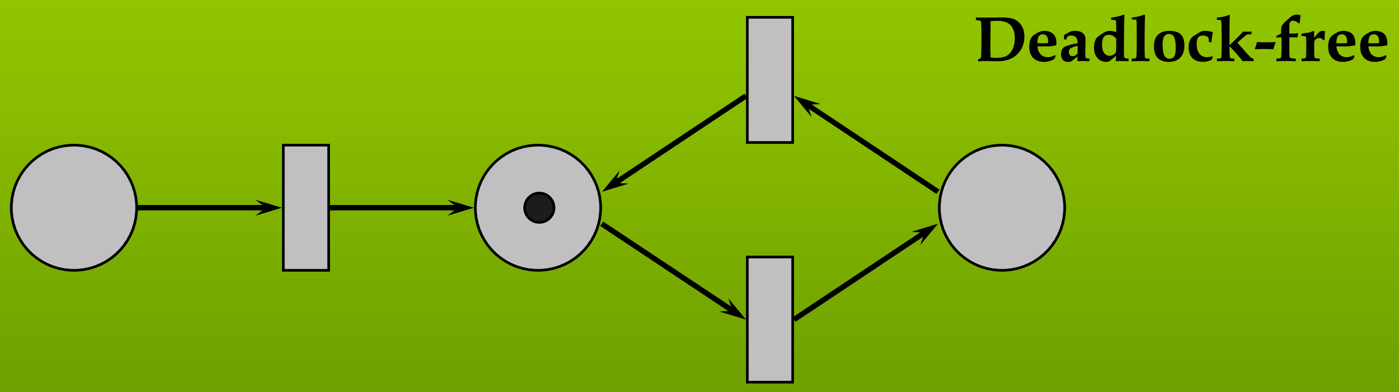
Liveness

- **Liveness: from any marking any transition can become fireable**
  - **Liveness implies deadlock freedom, not viceversa**



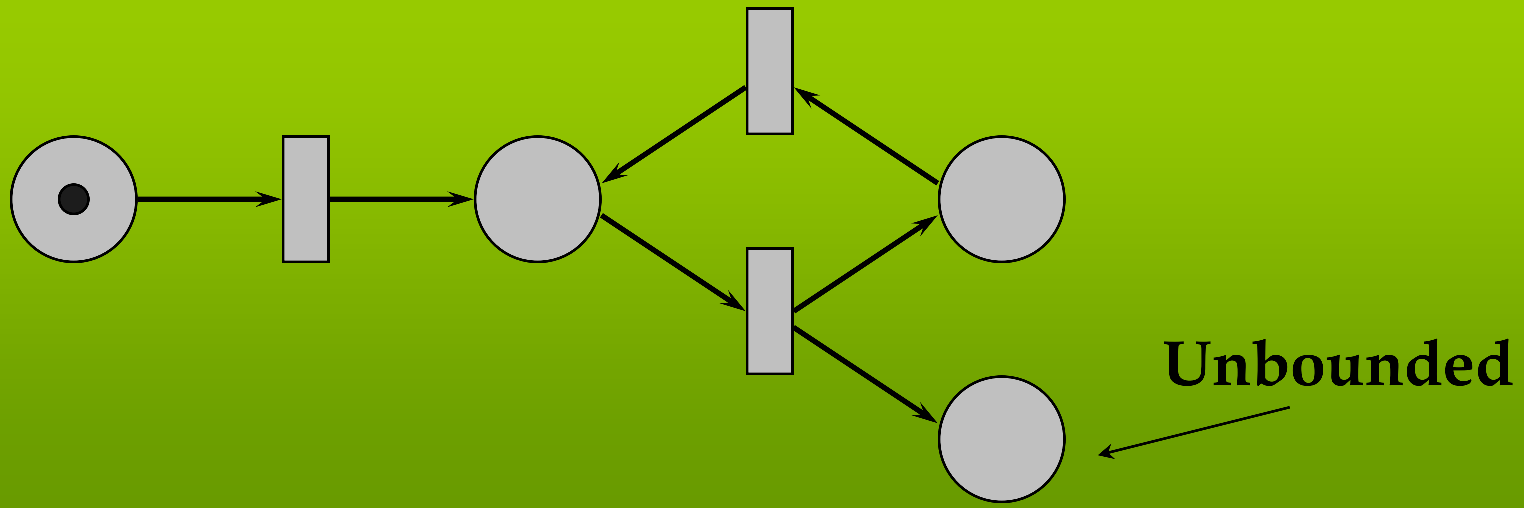
Liveness

- **Liveness: from any marking any transition can become fireable**
  - **Liveness implies deadlock freedom, not viceversa**



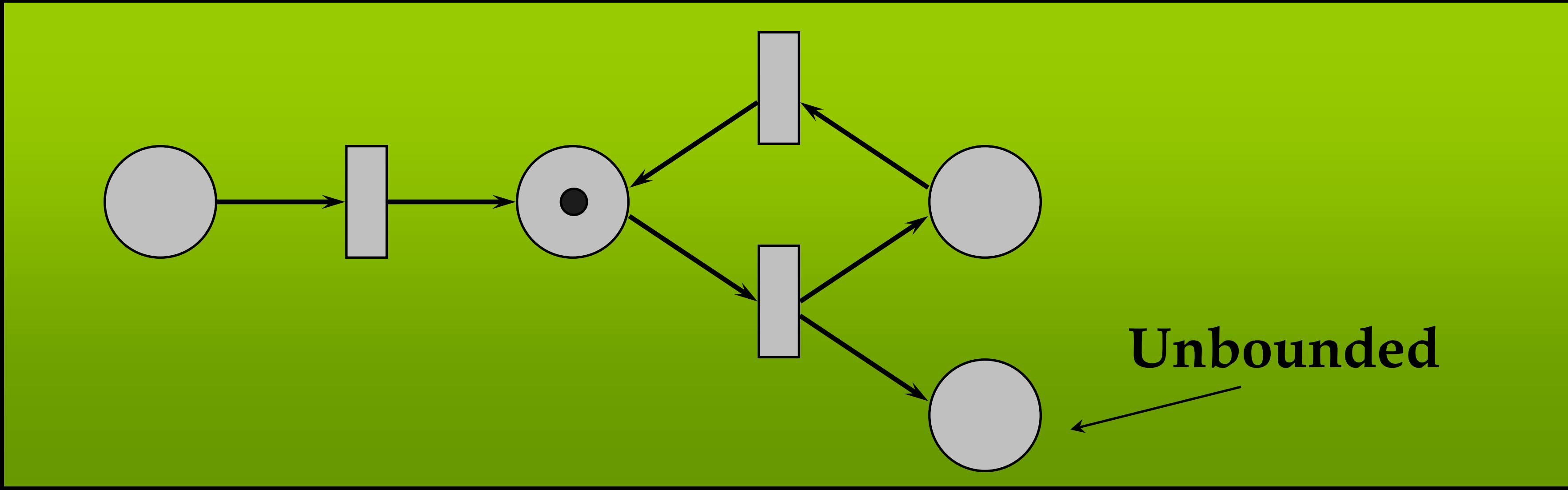
Boundedness

- **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)**



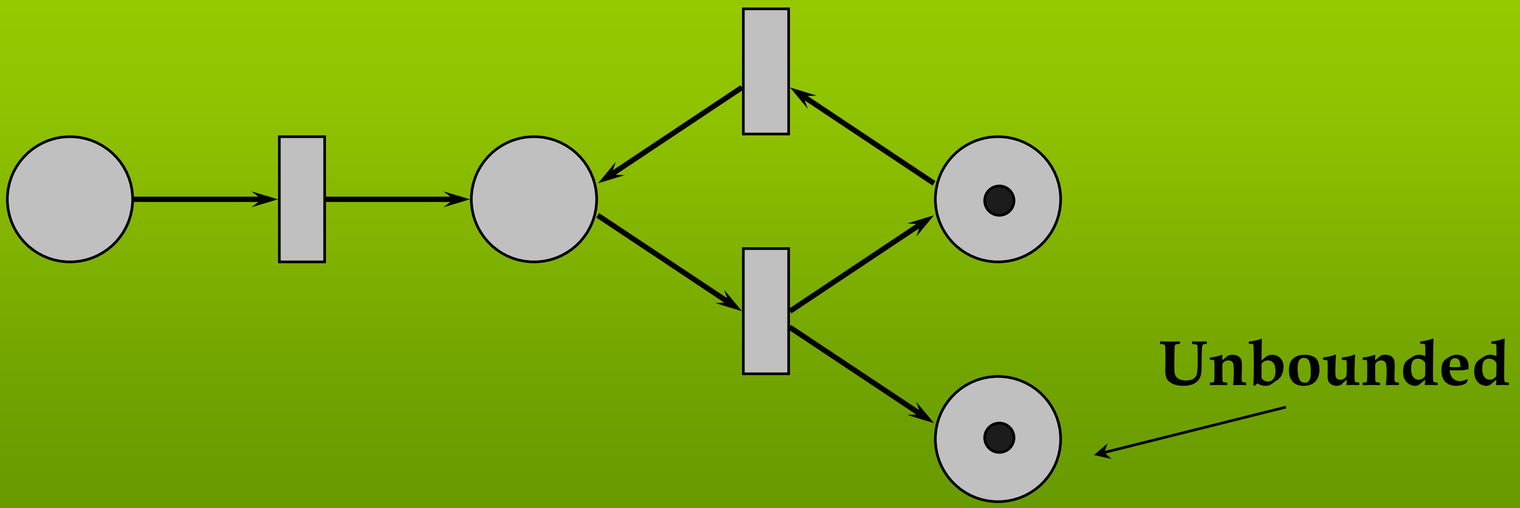
Boundedness

- **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)**



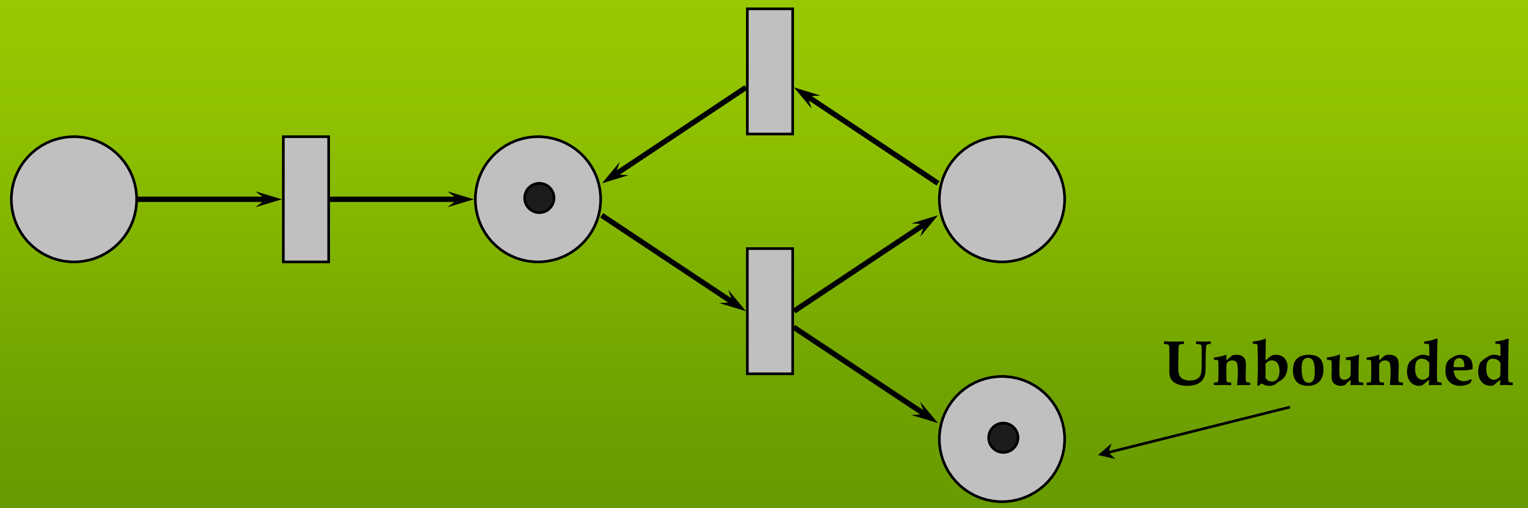
Boundedness

- **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)**



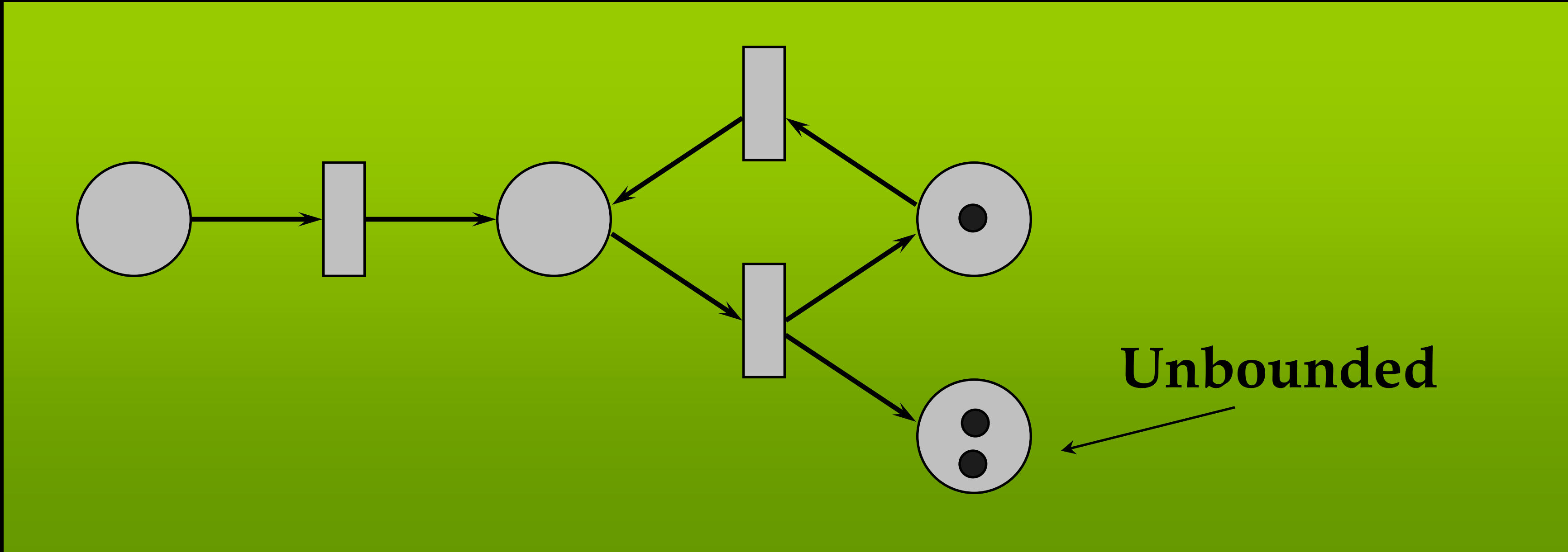
Boundedness

- **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)**



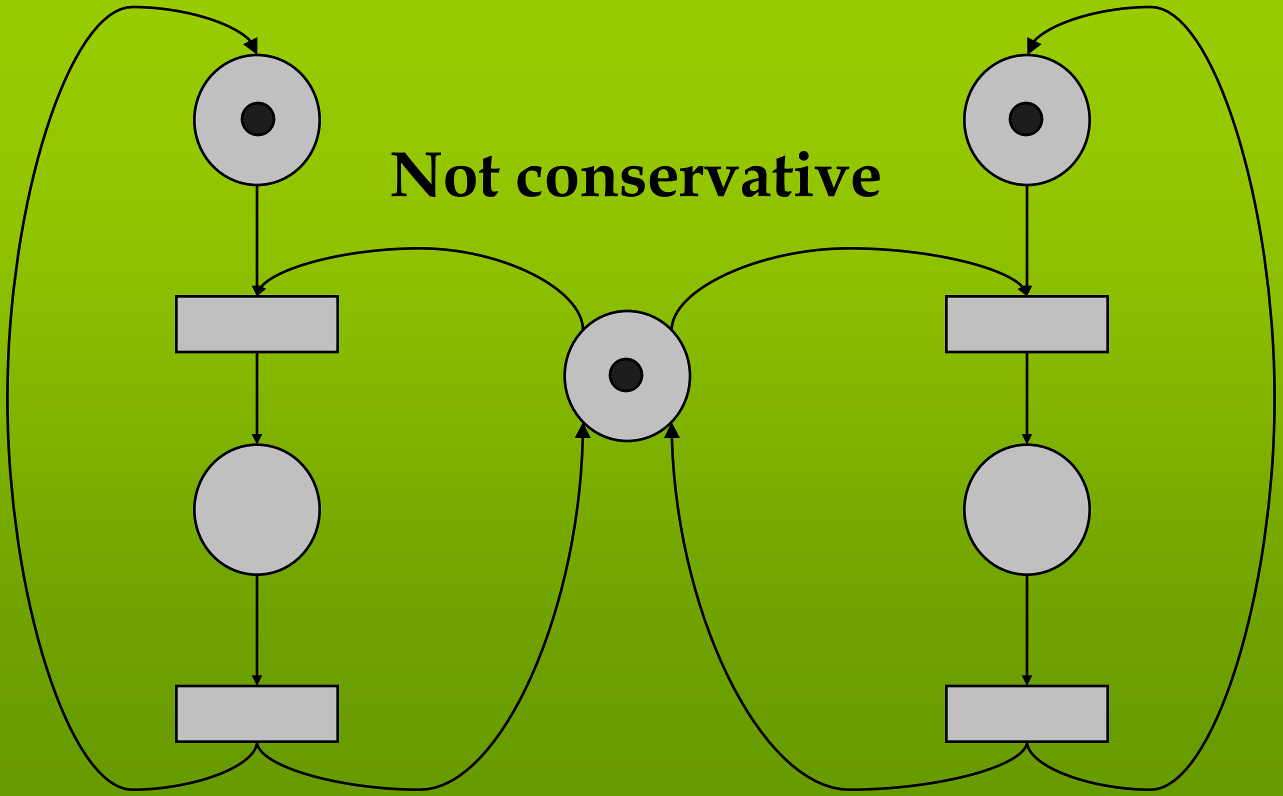
Boundedness

- **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)**



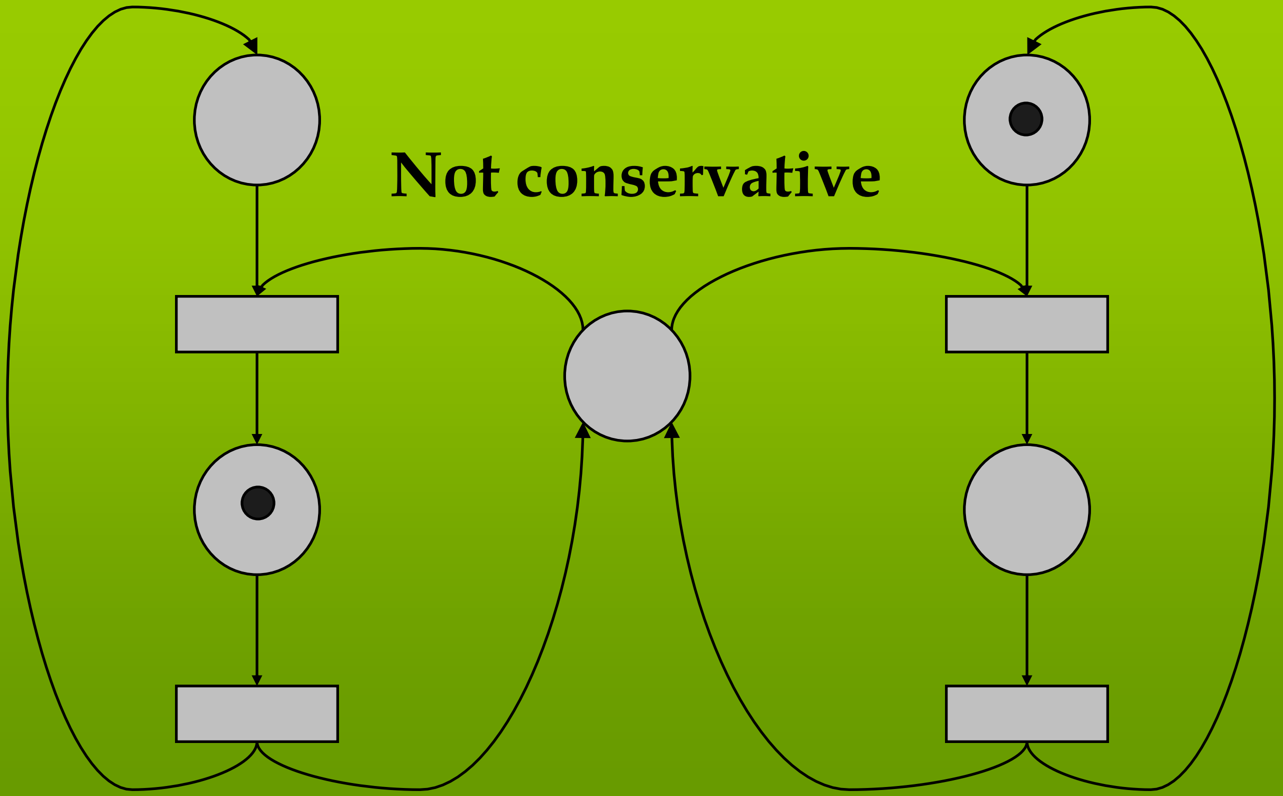
Conservation

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



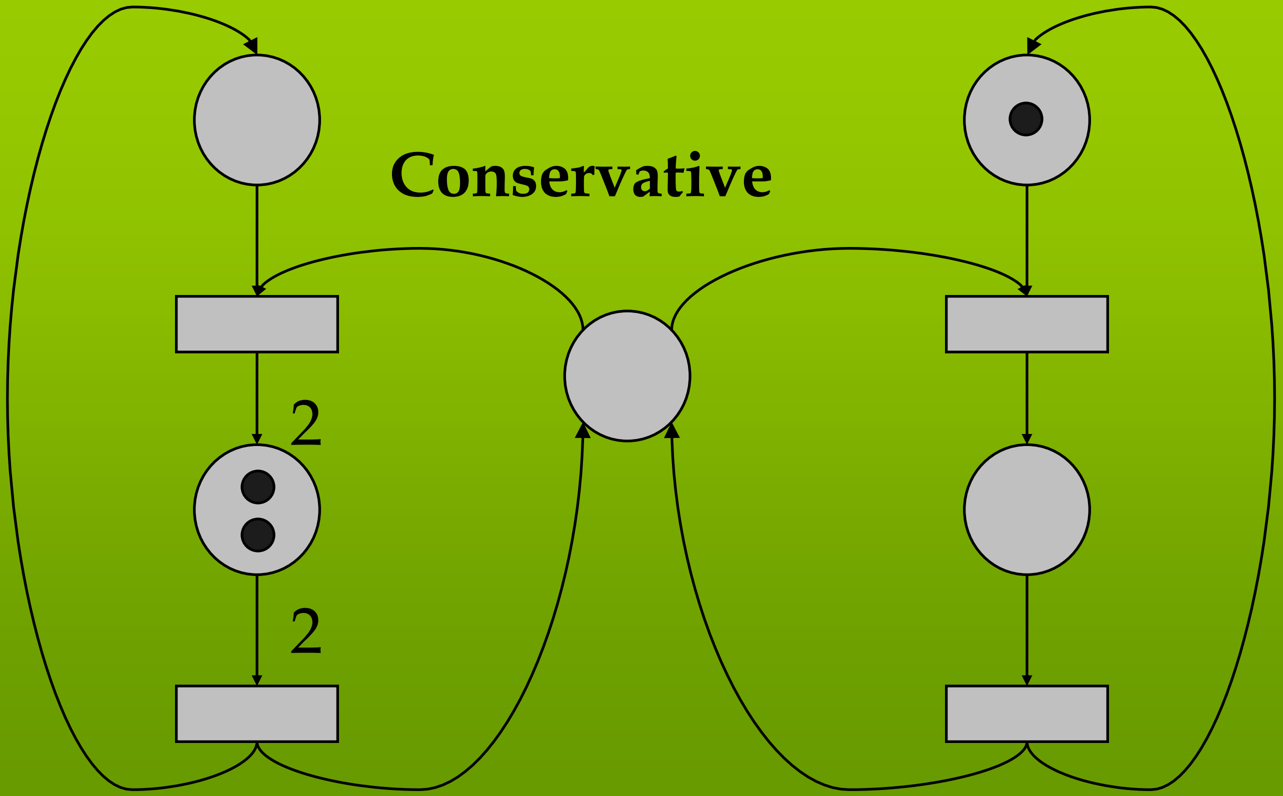
Conservation

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



Conservation

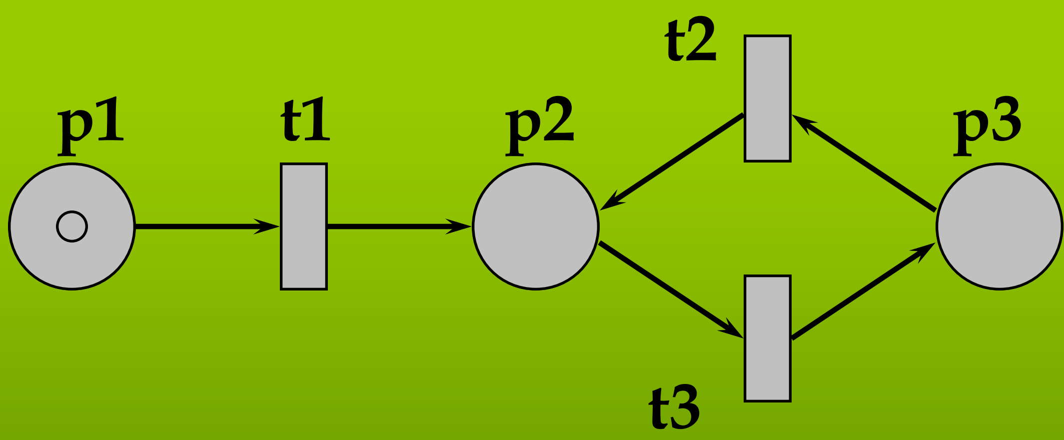
- **Conservation: the total number of tokens in the net is constant**



Analysis techniques

- **Structural analysis techniques**
  - **Incidence matrix**
  - **T- and S- Invariants**
- **State Space Analysis techniques**
  - **Coverability Tree**
  - **Reachability Graph**

Incidence Matrix



$$A = \begin{array}{c} \\ \end{array}\begin{array}{ccc} t1 & t2 & t3 \end{array}$$

$$A = \left|\begin{array}{ccc} -1 & 0 & 0 \\ 1 & 1 & -1 \\ 0 & -1 & 1 \end{array}\right| \begin{array}{c} p1 \\ p2 \\ p3 \end{array}$$

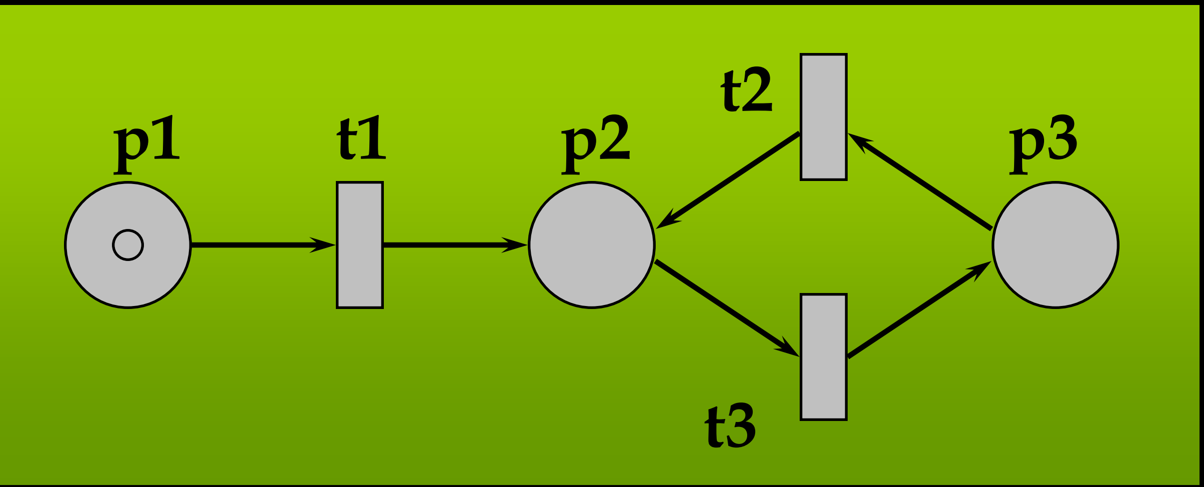
- **Necessary condition for marking M to be reachable from initial marking M_0:**

there exists firing vector v s.t.:

$$M = M_0 + A\,v$$

State equations

- **E.g. reachability of M = |0 0 1|^T from M_0 = |1 0 0|^T**

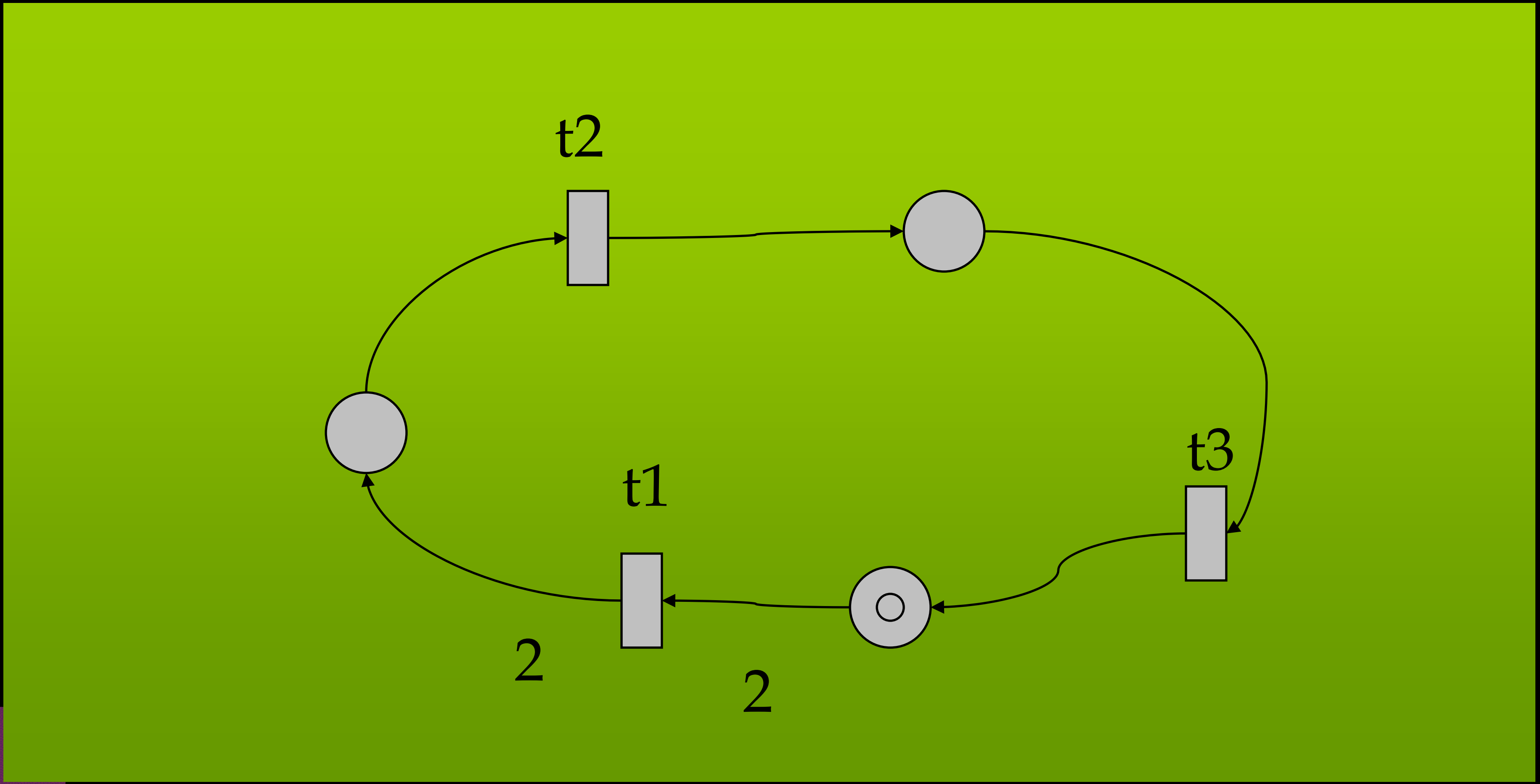


$$A = \begin{vmatrix} -1 & 0 & 0 \\ 1 & 1 & -1 \\ 0 & -1 & 1 \end{vmatrix}$$

$$v_1 = \begin{vmatrix} 1 \\ 0 \\ 1 \end{vmatrix} \qquad \begin{vmatrix} 0 \\ 0 \\ 1 \end{vmatrix} = \begin{vmatrix} 1 \\ 0 \\ 0 \end{vmatrix} + \begin{vmatrix} -1 & 0 & 0 \\ 1 & 1 & -1 \\ 0 & -1 & 1 \end{vmatrix} \begin{vmatrix} 1 \\ 0 \\ 1 \end{vmatrix}$$

but also $v_2 = |\ 1\ 1\ 2\ |^T$ or any $v_k = |\ 1\ (k)\ (k+1)\ |^T$

Necessary Condition only



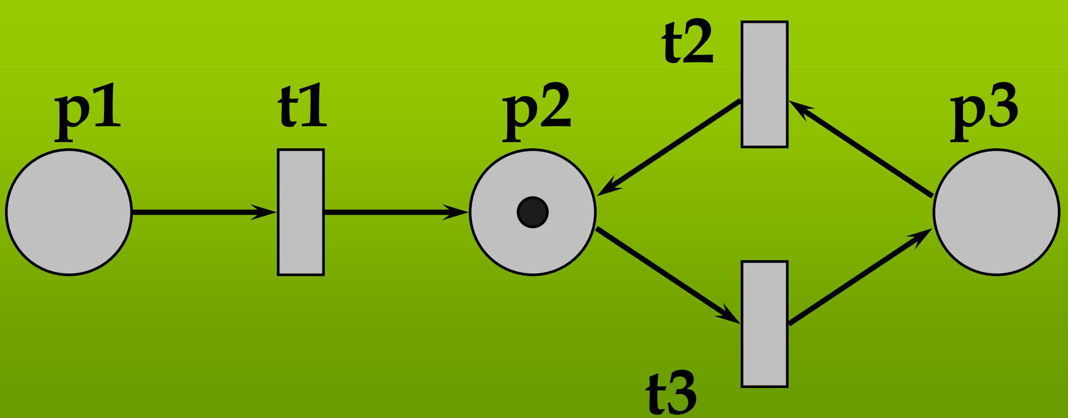
Deadlock!!

State equations and invariants

- **Solutions of $Ax = 0$ (in $M = M_0 + Ax$, $M = M_0$)**

  T-invariants

  - **sequences of transitions that (if fireable) bring back to original marking**
  - **periodic schedule in SDF**
  - **e.g. $x = | 0\ 1\ 1 |^T$**



$$A = \begin{vmatrix} -1 & 0 & 0 \\ 1 & 1 & -1 \\ 0 & -1 & 1 \end{vmatrix}$$

Application of T-invariants

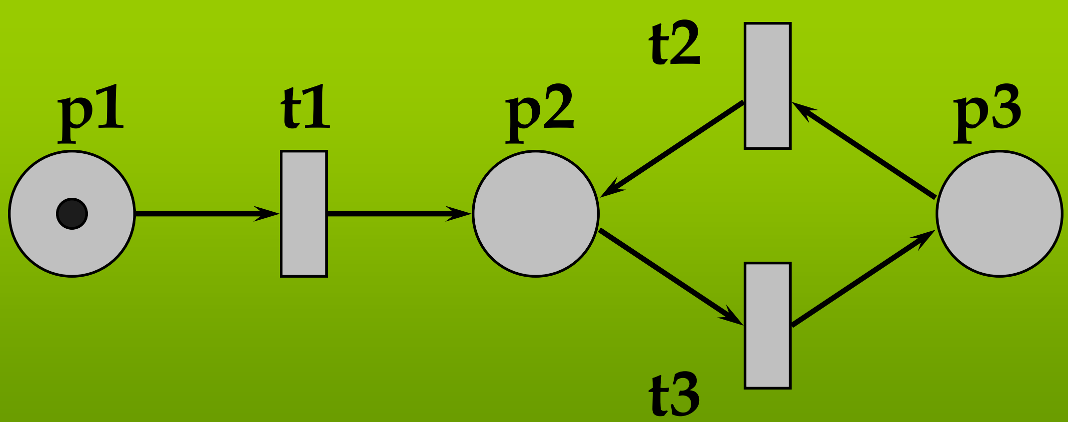
- **Scheduling**
  - **Cyclic schedules: need to return to the initial state**

State equations and invariants

- **Solutions of yA = 0**

  S-invariants

  - **sets of places whose weighted total token count does not change after the firing of any transition (y M = y M')**
  - **e.g. y =| 1 1 1 |^T**



$$A^T = \begin{vmatrix} -1 & 1 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{vmatrix}$$

Application of S-invariants

- **Structural Boundedness: bounded for any finite initial marking M_0**
- **Existence of a positive S-invariant is sufficient condition for structural boundedness**
  - **initial marking is finite**
  - **weighted token count does not change**

Summary of algebraic methods

- **Extremely efficient**

  (polynomial in the size of the net)
- **Generally provide only necessary or sufficient information**
- **Excellent for ruling out some deadlocks or otherwise dangerous conditions**
- **Can be used to infer structural boundedness**

Coverability Tree

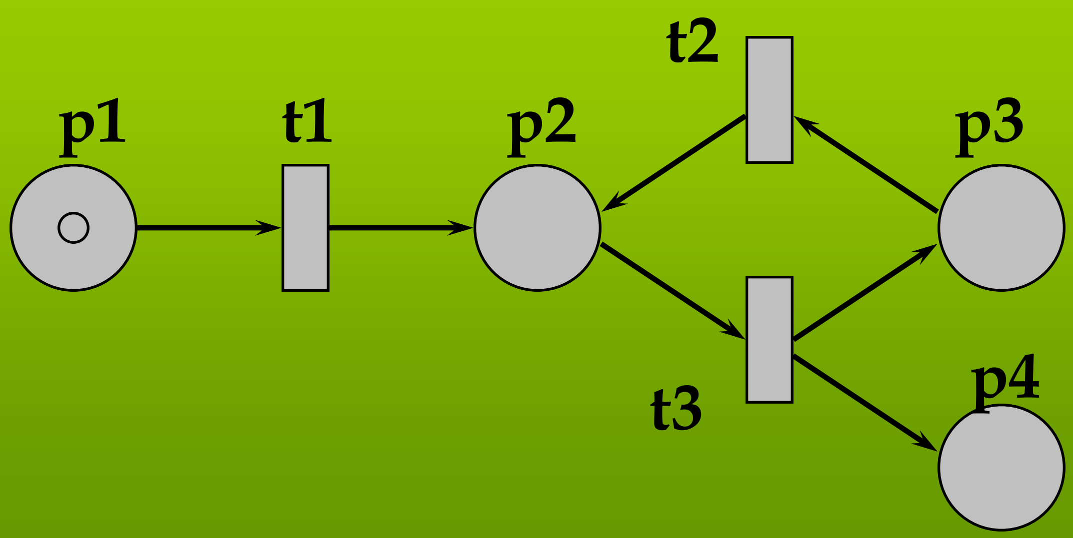
- **Build a (finite) tree representation of the markings**

Karp-Miller algorithm

- **Label initial marking M0 as the root of the tree and tag it as *new***
- **While new markings exist do:**
  - **select a new marking M**
  - **if M is identical to a marking on the path from the root to M, then tag M as *old* and go to another new marking**
  - **if no transitions are enabled at M, tag M *dead-end***
  - **while there exist enabled transitions at M do:**
    - **obtain the marking M' that results from firing t at M**
    - **on the path from the root to M if there exists a marking M'' such that M'(p)>=M''(p) for each place p and M' is different from M'', then replace M'(p) by ω for each p such that M'(p) >M''(p)**
    - **introduce M' as a node, draw an arc with label t from M to M' and tag M' as *new*.**

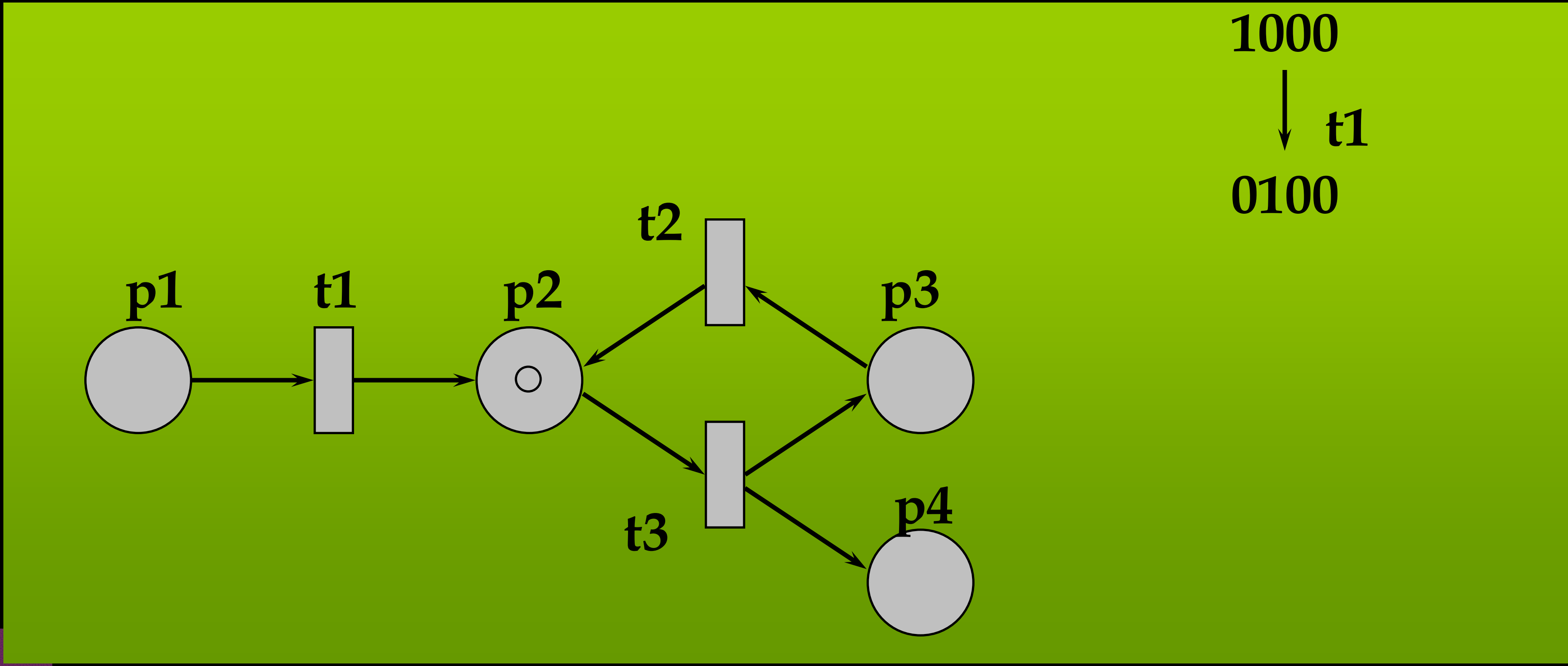
Coverability Tree

- **Boundedness is decidable with *coverability tree***



Coverability Tree

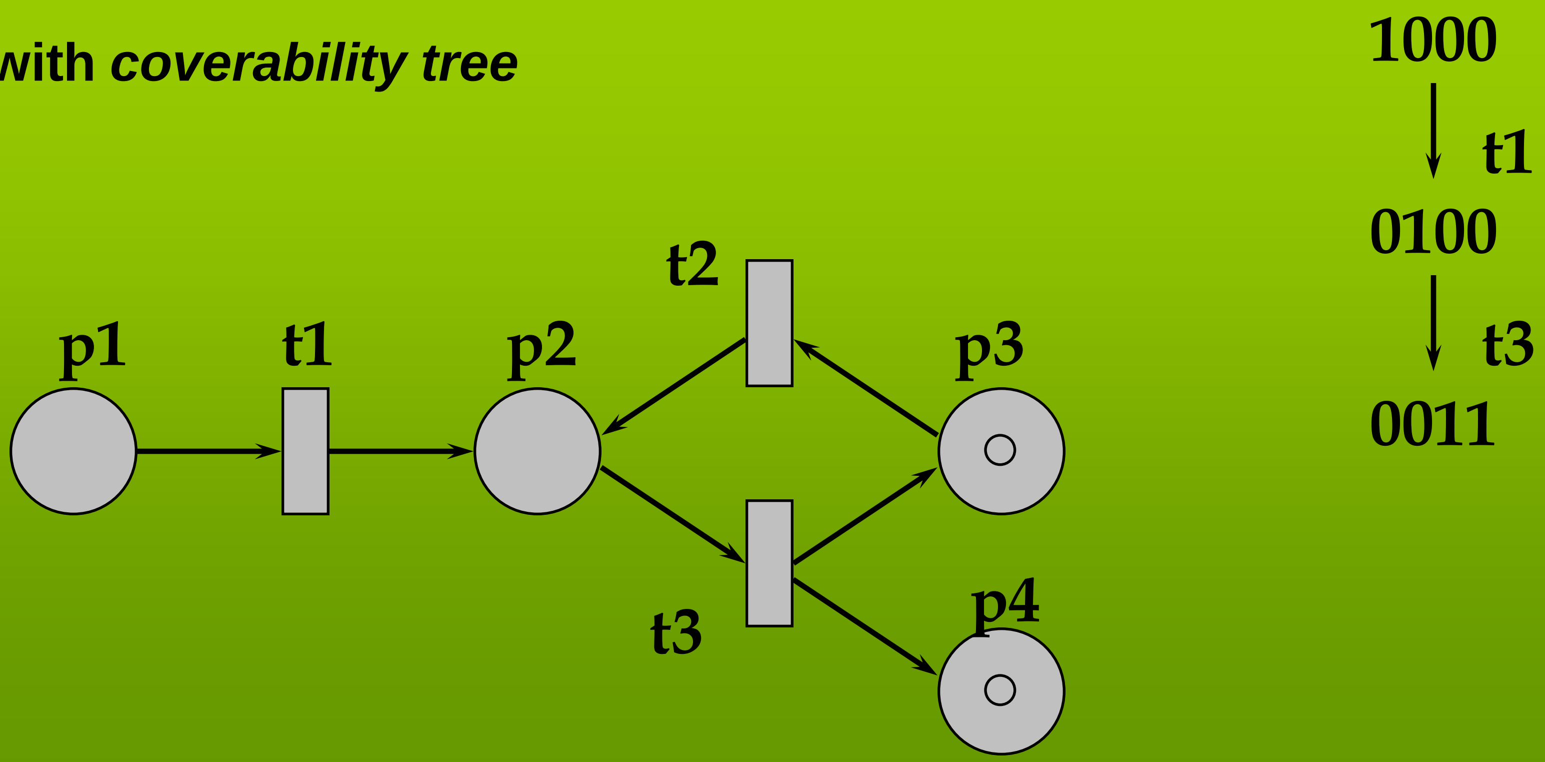
- **Boundedness is decidable with *coverability tree***



Coverability Tree

- **Boundedness is decidable**

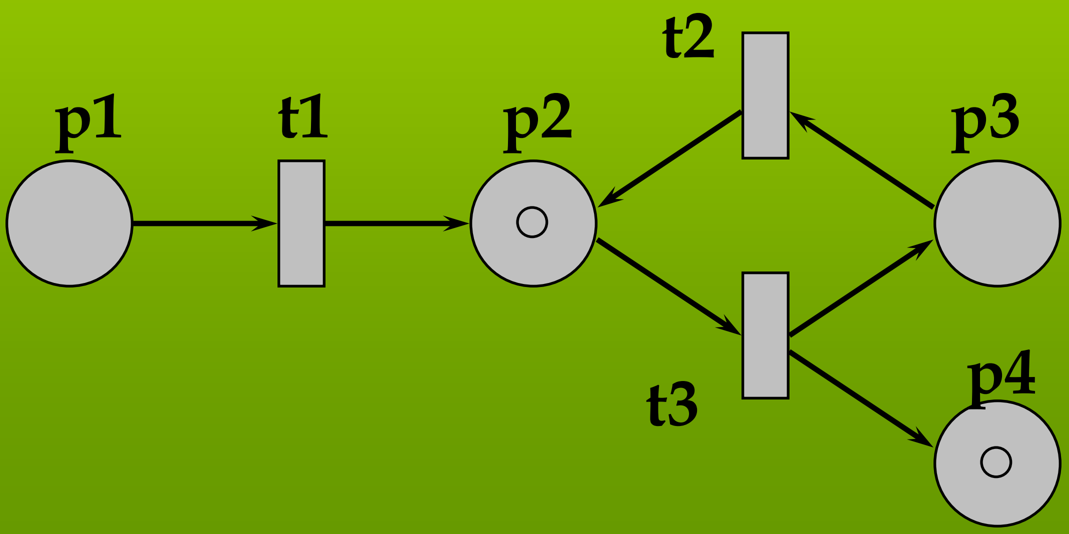
  with *coverability tree*

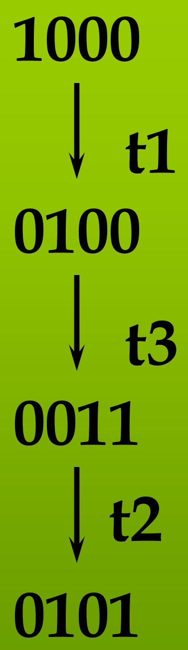


Coverability Tree

- **Boundedness is decidable**

  with *coverability tree*

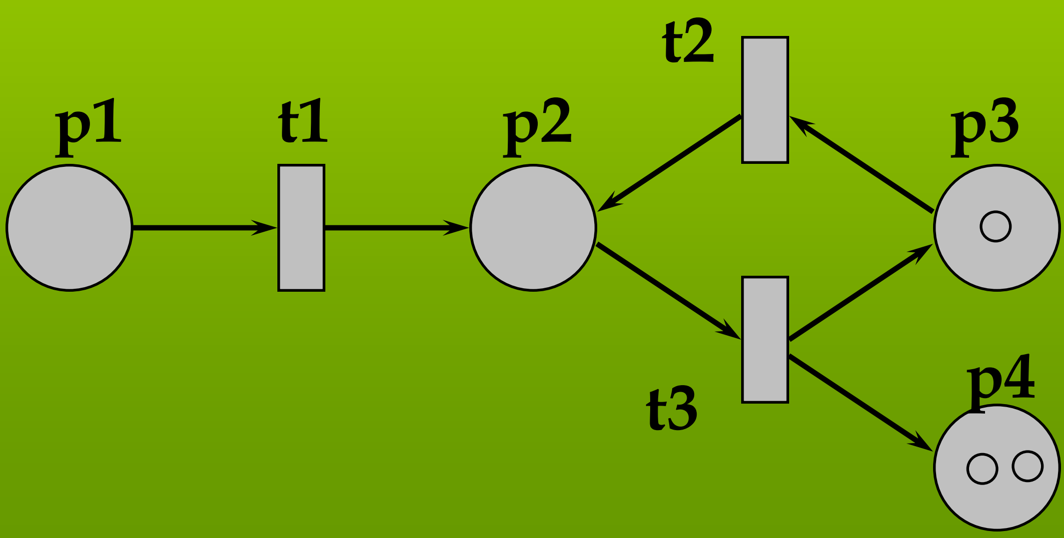


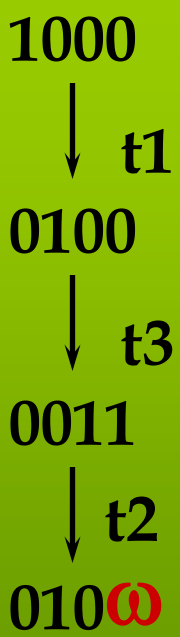


Coverability Tree

- **Boundedness is decidable**

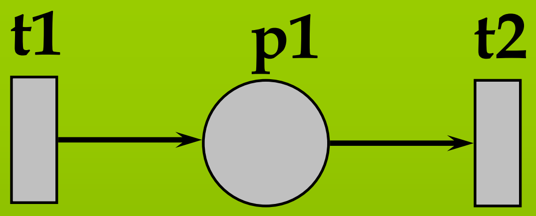
  with *coverability tree*

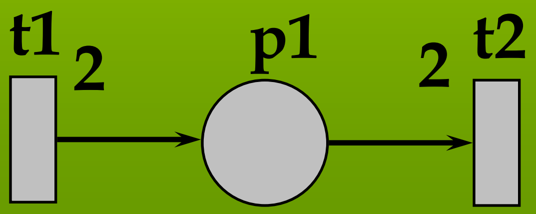




Coverability Tree

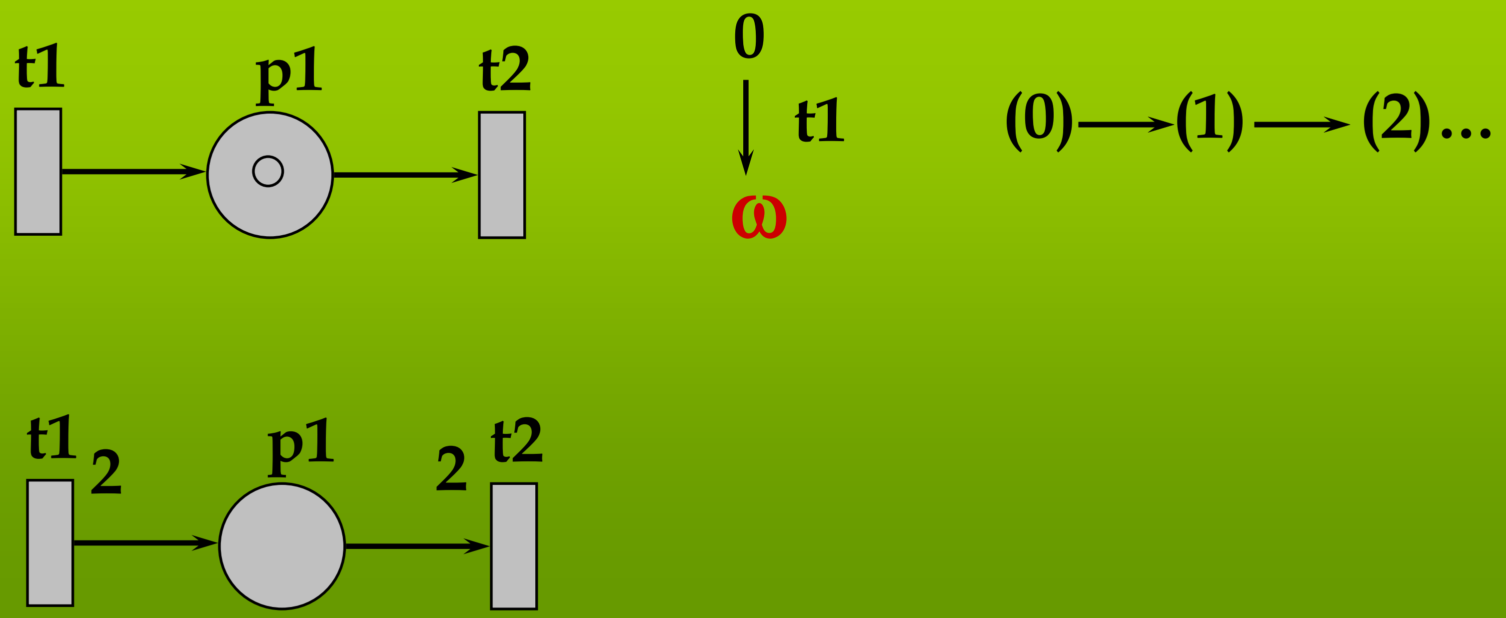
- **Is (1) reachable from (0)?**





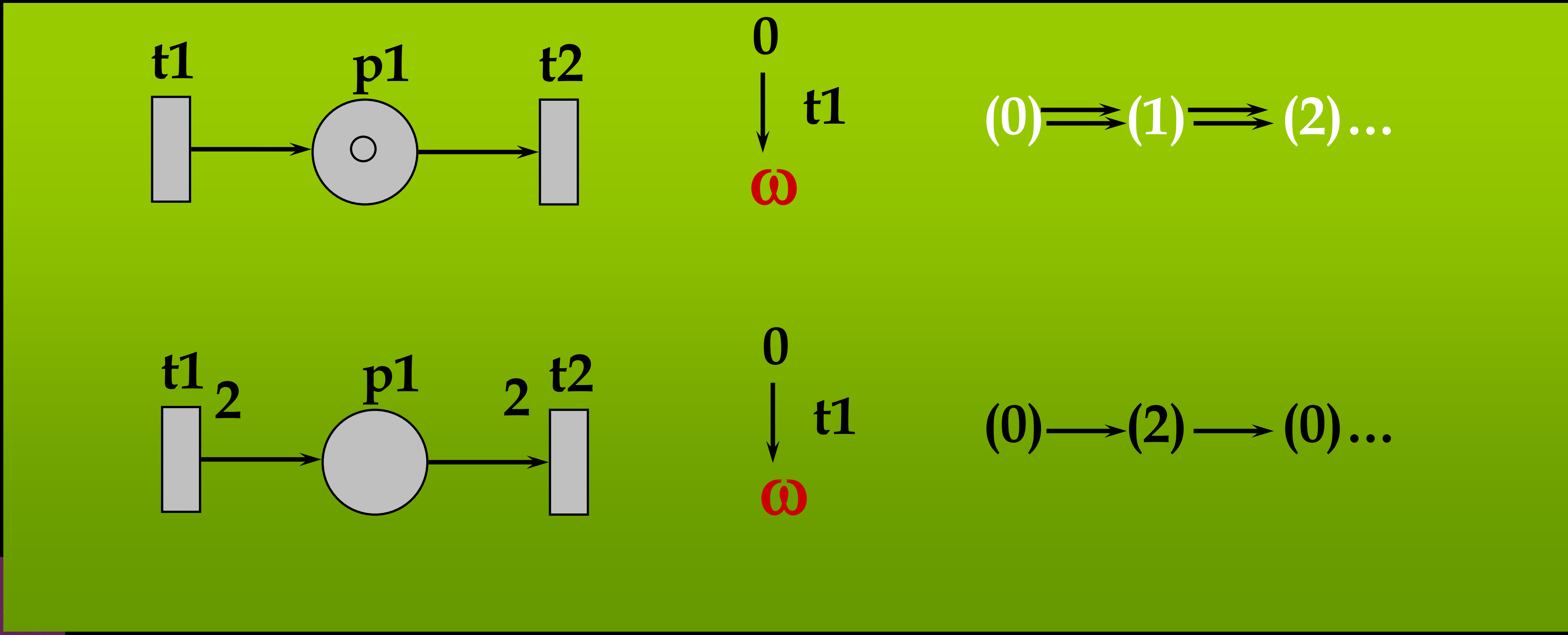
Coverability Tree

- **Is (1) reachable from (0)?**



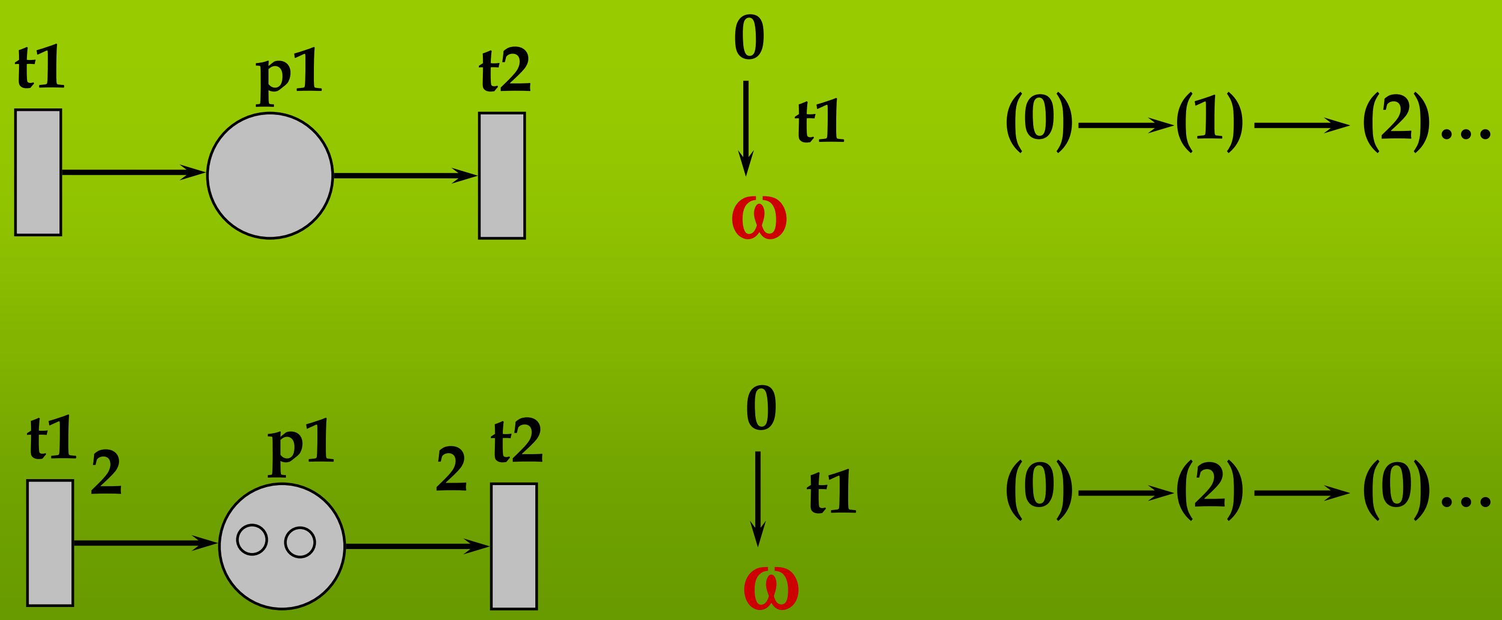
Coverability Tree

- **Is (1) reachable from (0)?**



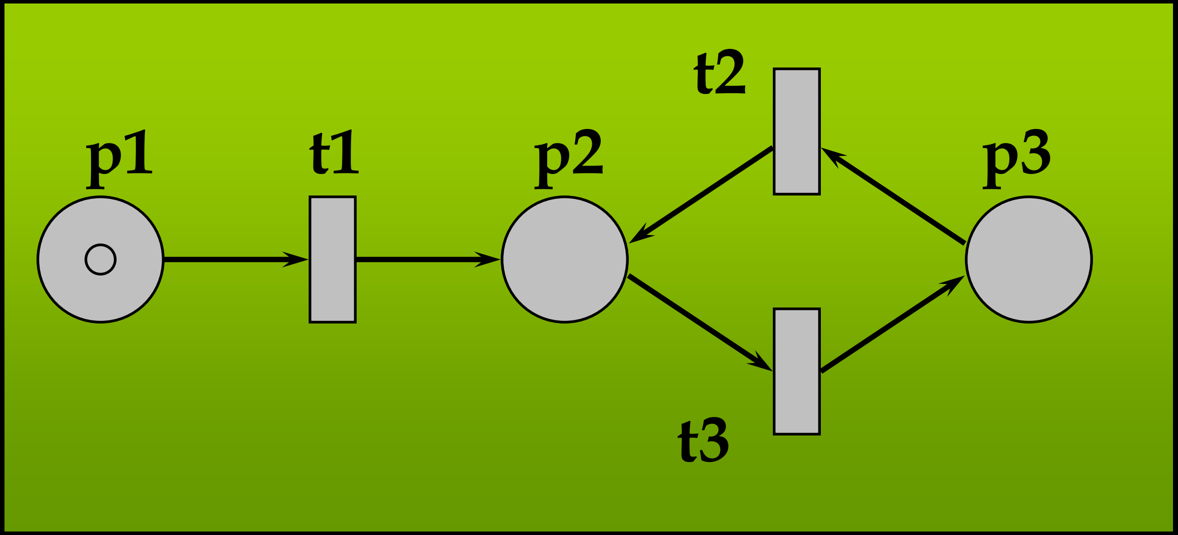
Coverability Tree

- **Is (1) reachable from (0)?**



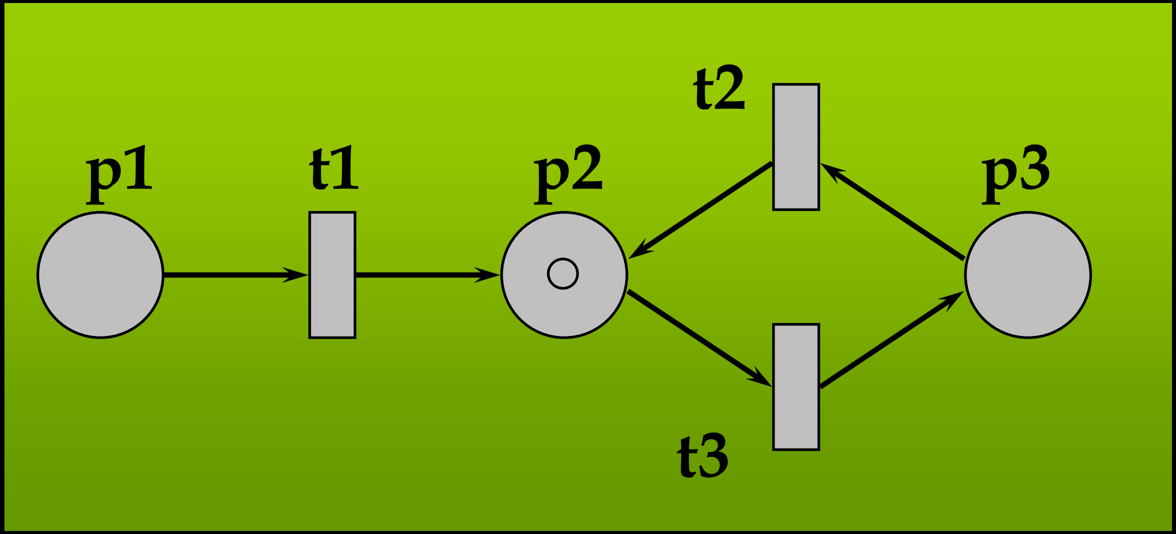
- **Cannot solve the reachability problem**

Reachability graph



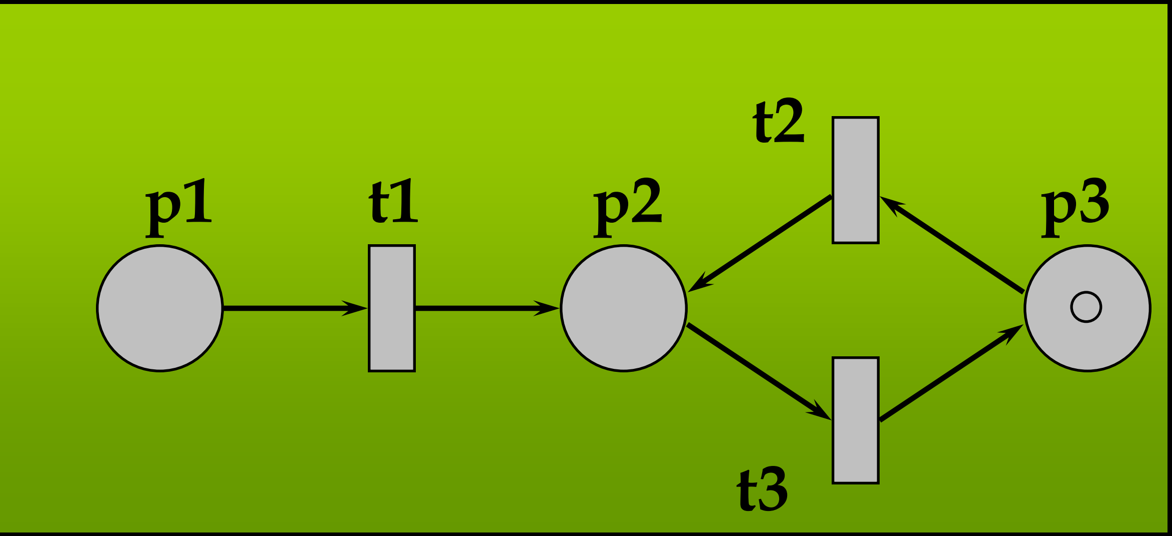
- **For bounded nets the Coverability Tree is called Reachability Tree since it contains all possible reachable markings**

Reachability graph



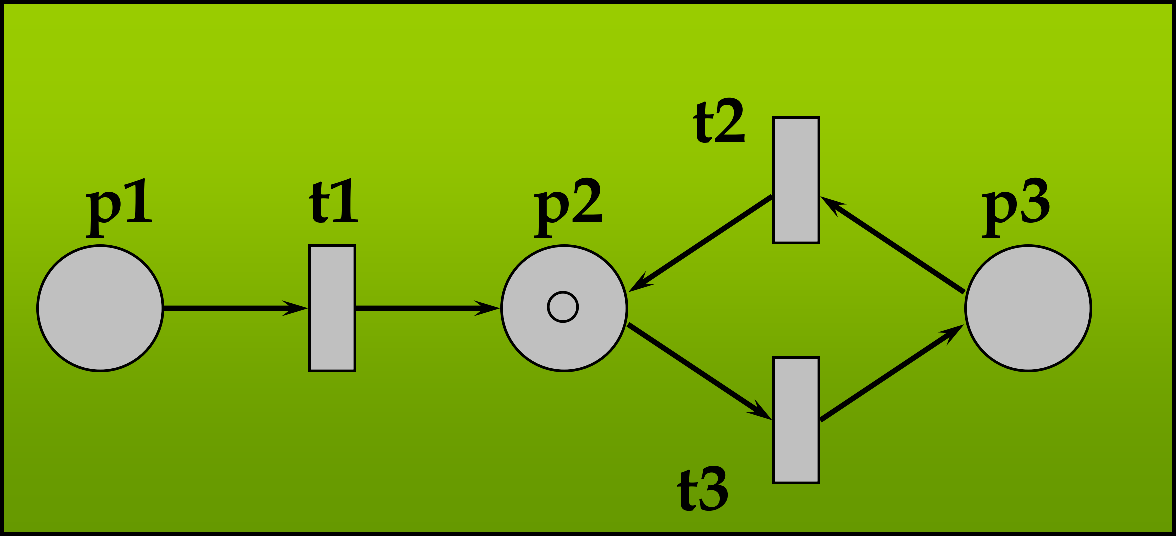
- **For bounded nets the Coverability Tree is called Reachability Tree since it contains all possible reachable markings**

Reachability graph



- For bounded nets the Coverability Tree is called Reachability Tree since it contains all possible reachable markings

Reachability graph

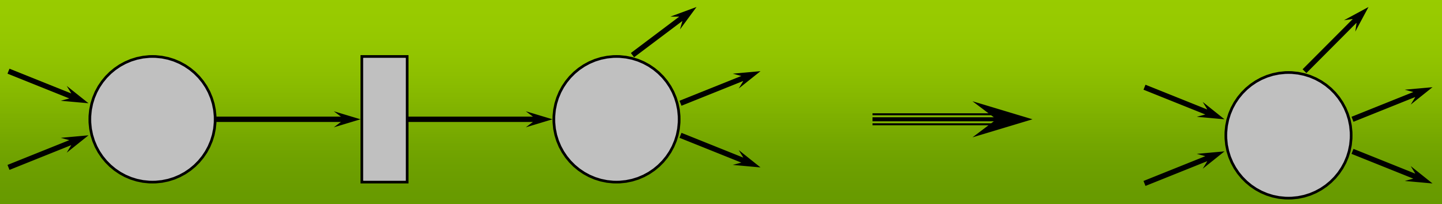


- **For bounded nets the Coverability Tree is called Reachability Tree since it contains all possible reachable markings**

Subclasses of Petri nets

- **Reachability analysis is too expensive**
- **State equations give only partial information**
- **Some properties are preserved by reduction rules**

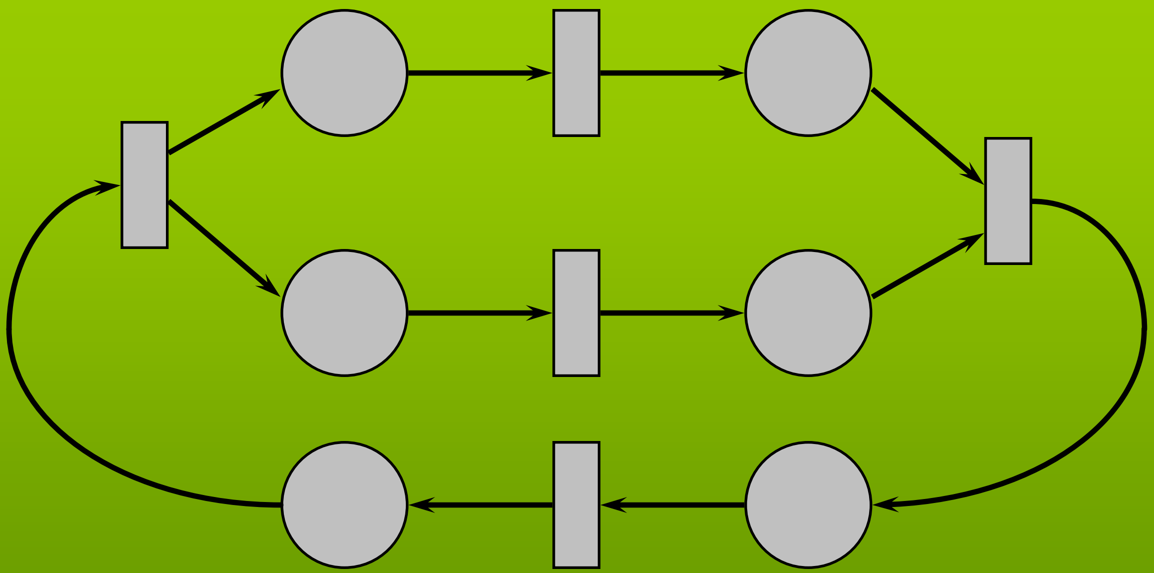
  e.g. for liveness and safeness



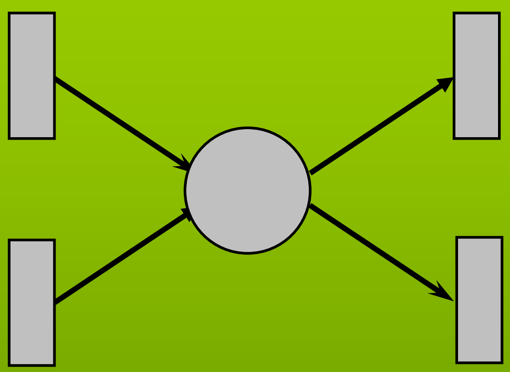
- **Even reduction rules only work in some cases**
- **Must restrict class in order to prove stronger results**

Marked Graphs

- **Every place has at most 1 predecessor and 1 successor transition**
- **Models only causality and concurrency (no conflict)**



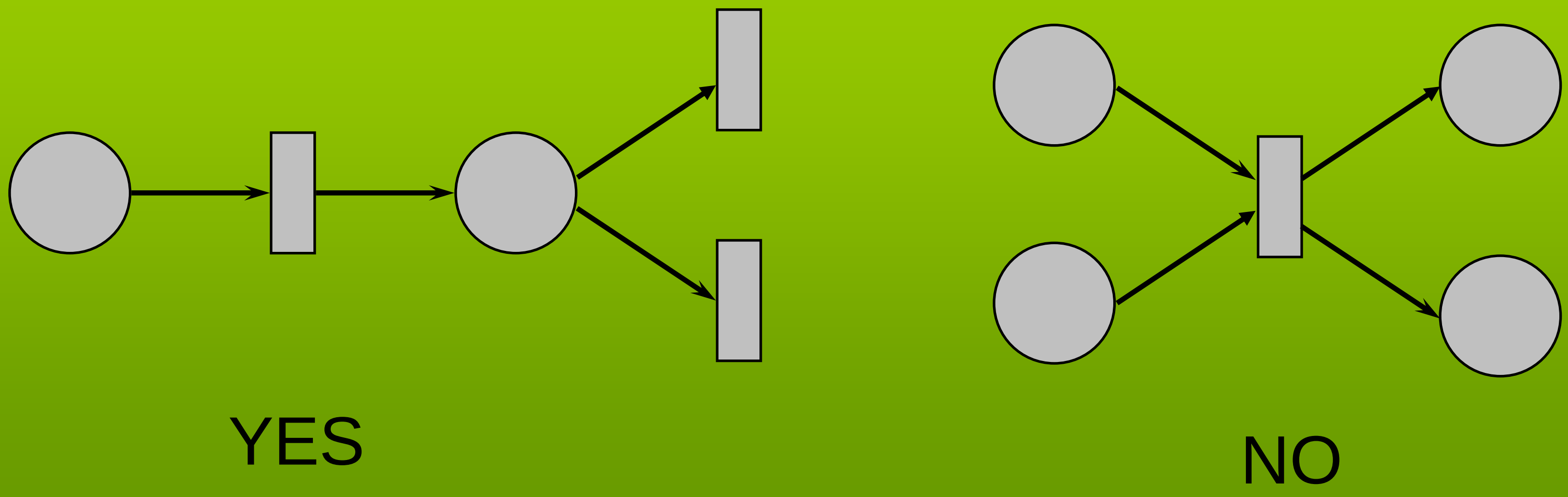
YES



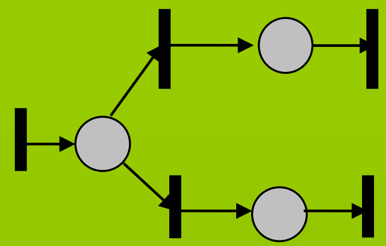
NO

State Machines

- **Every transition has at most 1 predecessor and 1 successor place**
- **Models only causality and conflict**
  - **(no concurrency, no synchronization of parallel activities)**

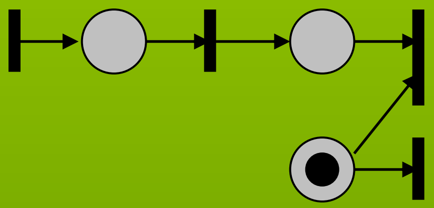


Free-Choice Petri Nets (FCPN)

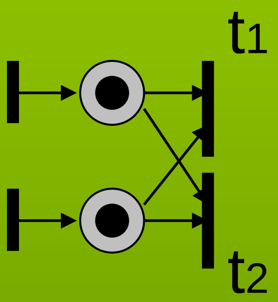


Free-Choice (FC)

every transition after choice has exactly 1 predecessor place



Confusion (not-Free-Choice)



Extended Free-Choice

Free-Choice: the outcome of a choice depends on the value of a token (abstracted non-deterministically) rather than on its arrival time.

Free-Choice nets

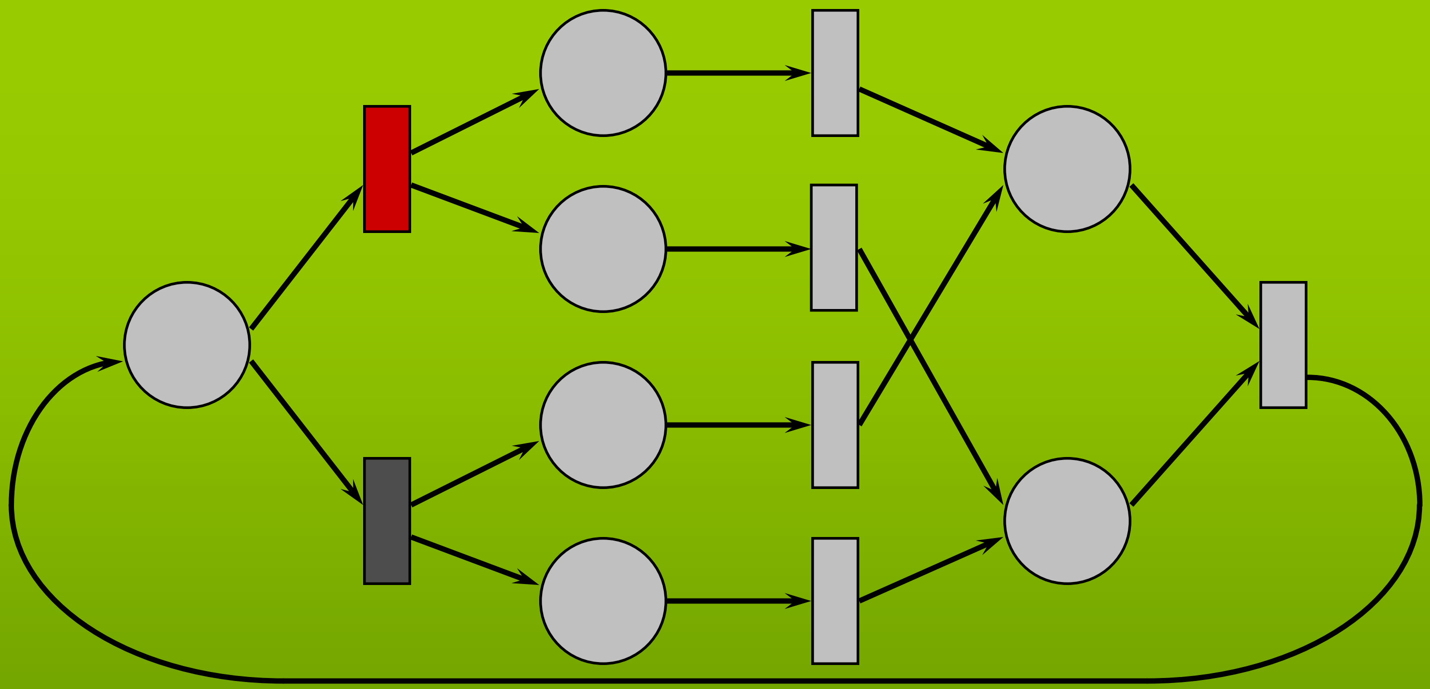
- **Introduced by Hack ('72)**
- **Extensively studied by Best ('86) and Desel and Esparza ('95)**
- **Can express concurrency, causality and choice without confusion**
- **Very strong structural theory**
  - **necessary and sufficient conditions for liveness and safeness, based on decomposition**
  - **exploits duality between MG and SM**

MG (& SM) decomposition

- **An Allocation is a control function that chooses which transition fires among several conflicting ones (A: P T).**
- **Eliminate the subnet that would be inactive if we were to use the allocation...**
- **Reduction Algorithm**
  - **Delete all unallocated transitions**
  - **Delete all places that have all input transitions already deleted**
  - **Delete all transitions that have at least one input place already deleted**
- **Obtain a Reduction (one for each allocation) that is a conflict free subnet**

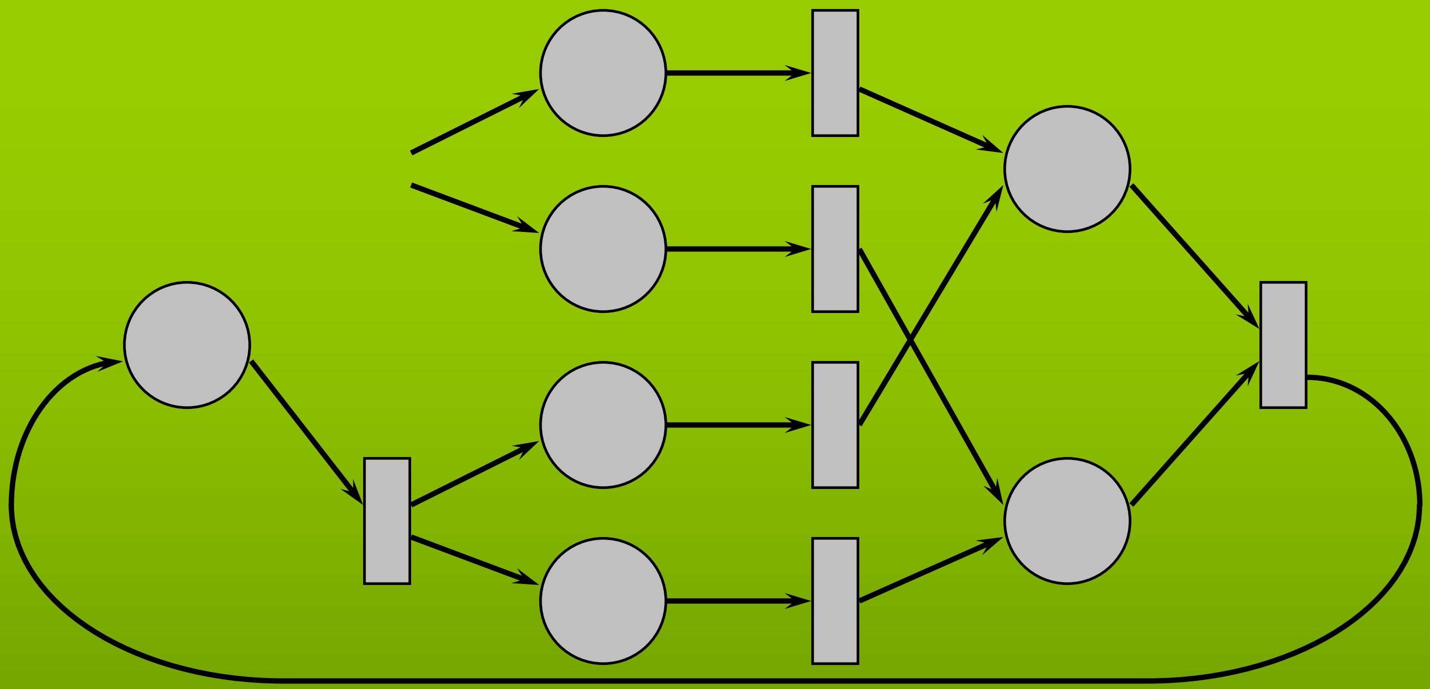
MG reduction and cover

- **Choose one successor for each conflicting place:**



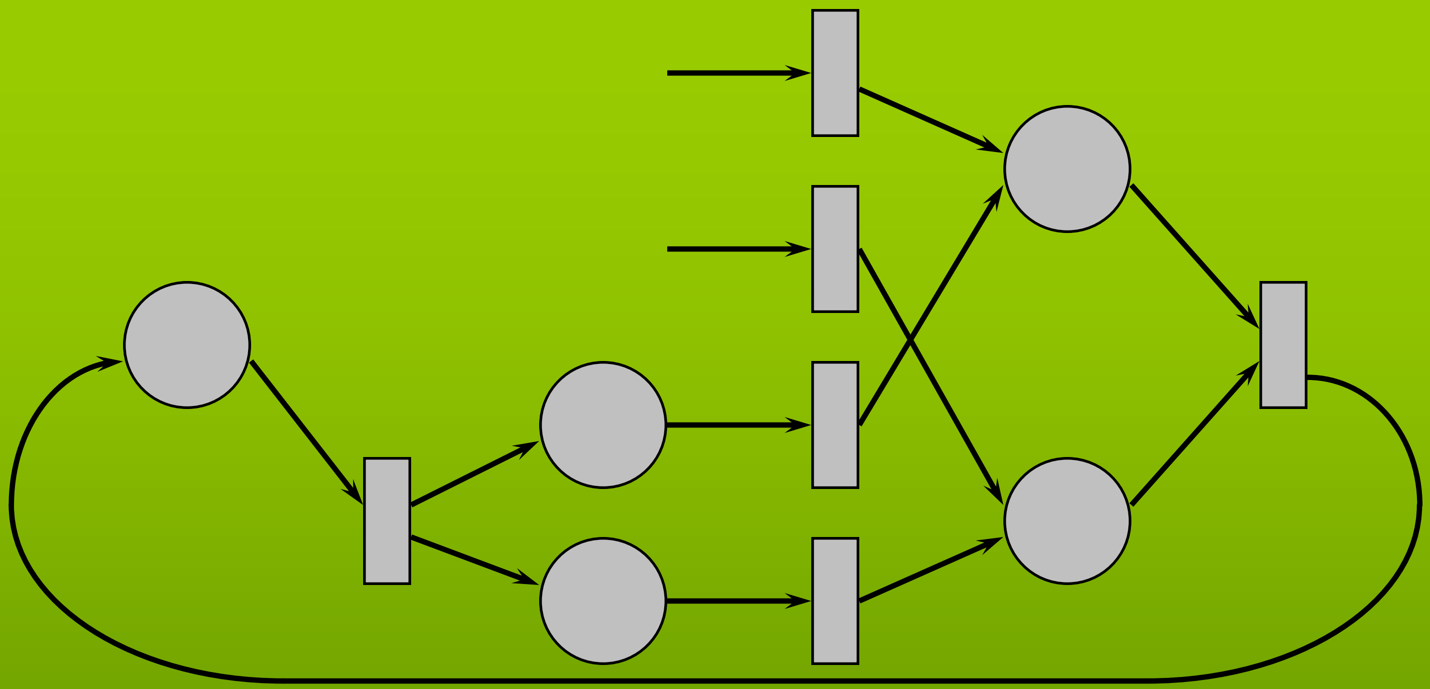
MG reduction and cover

- **Choose one successor for each conflicting place:**



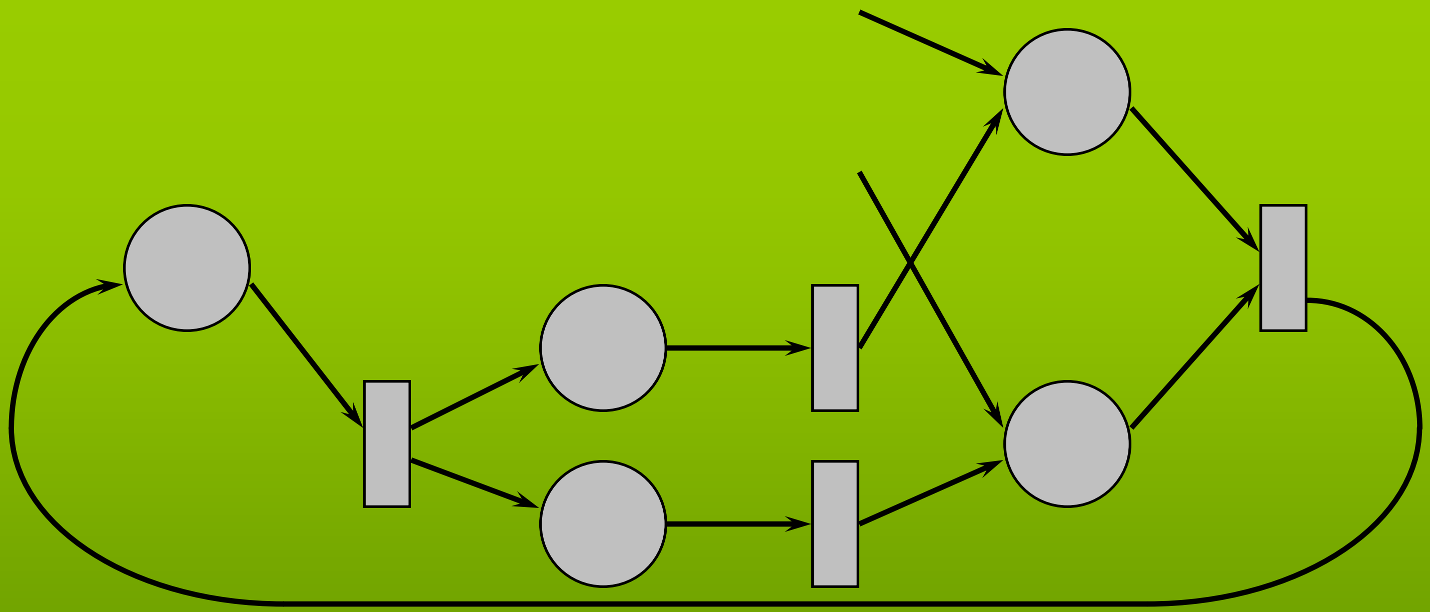
MG reduction and cover

- **Choose one successor for each conflicting place:**



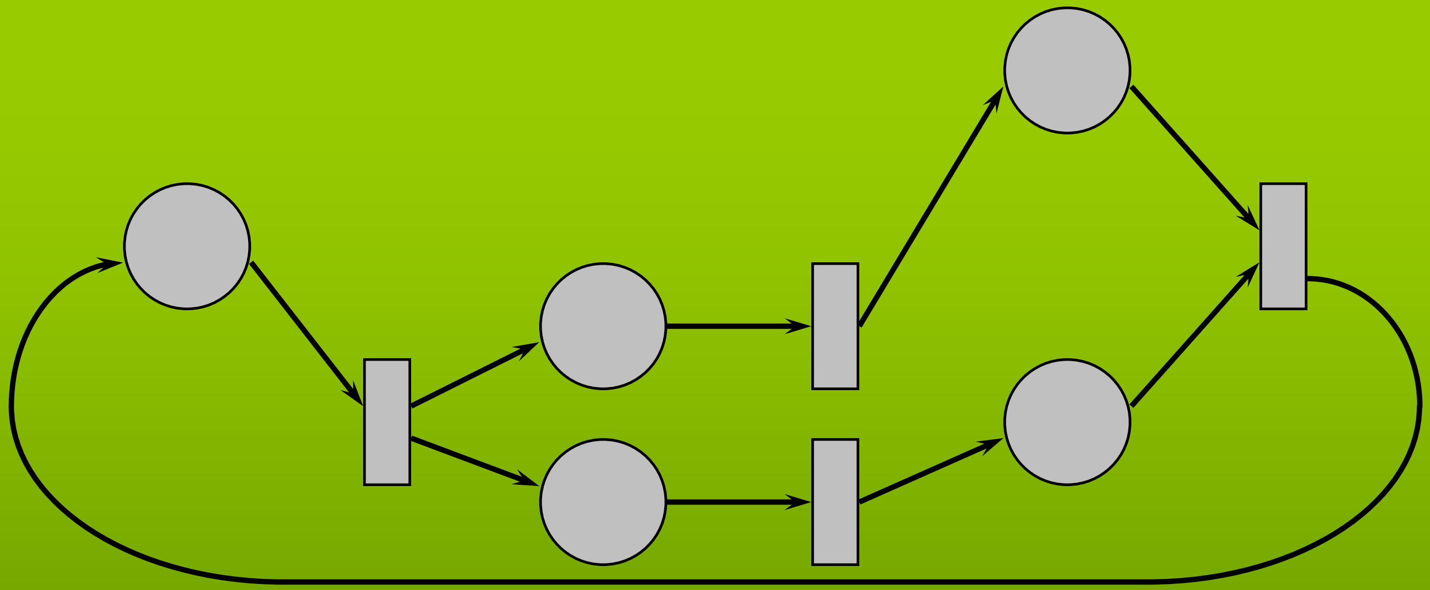
MG reduction and cover

- **Choose one successor for each conflicting place:**



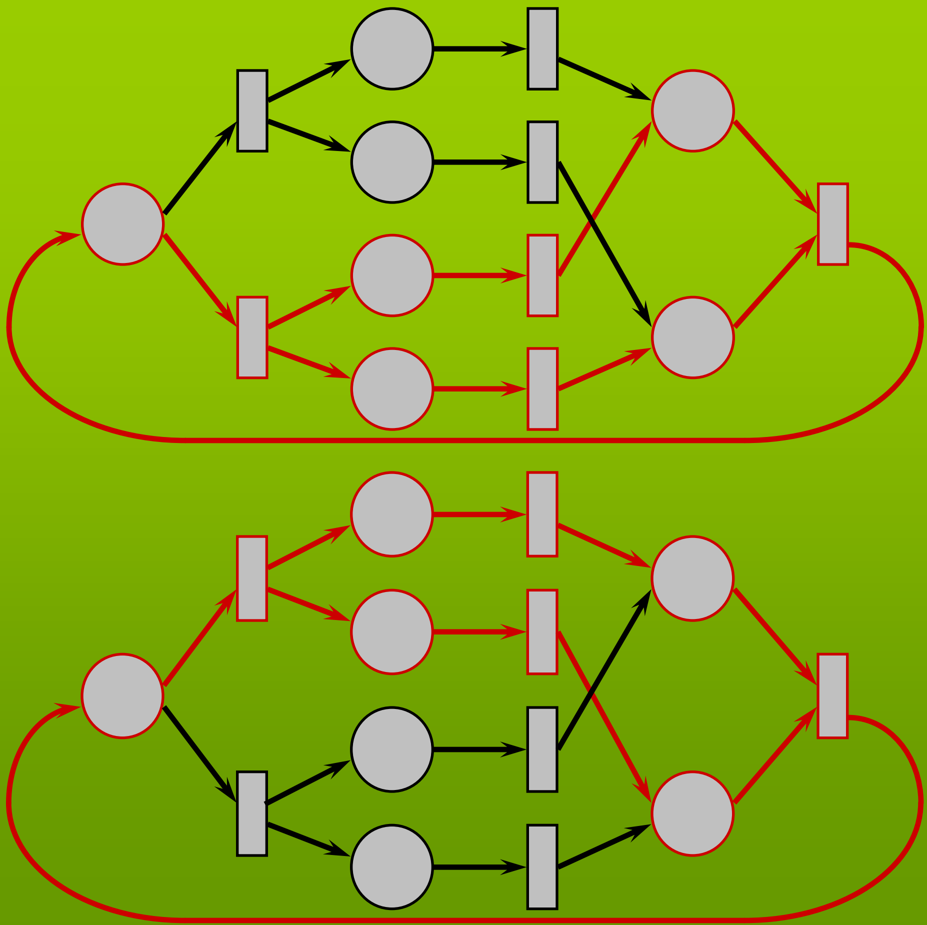
MG reduction and cover

- **Choose one successor for each conflicting place:**



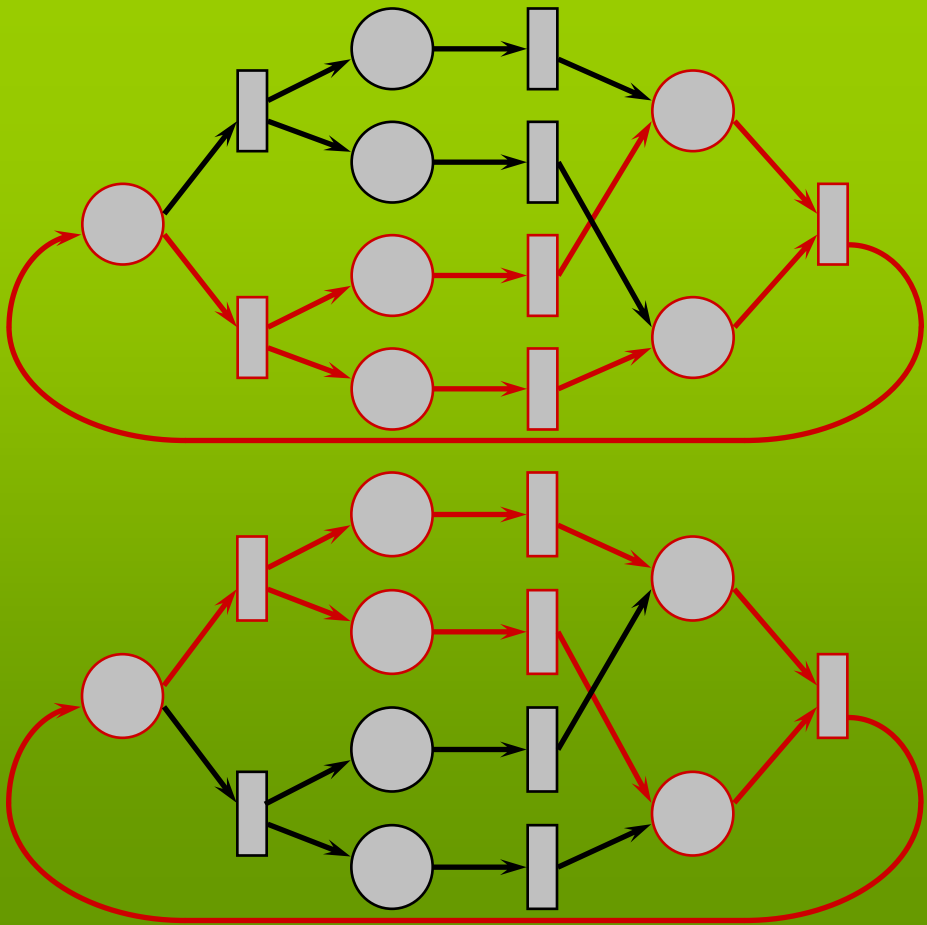
MG reductions

- **The set of all reductions yields a cover of MG components (T-invariants)**



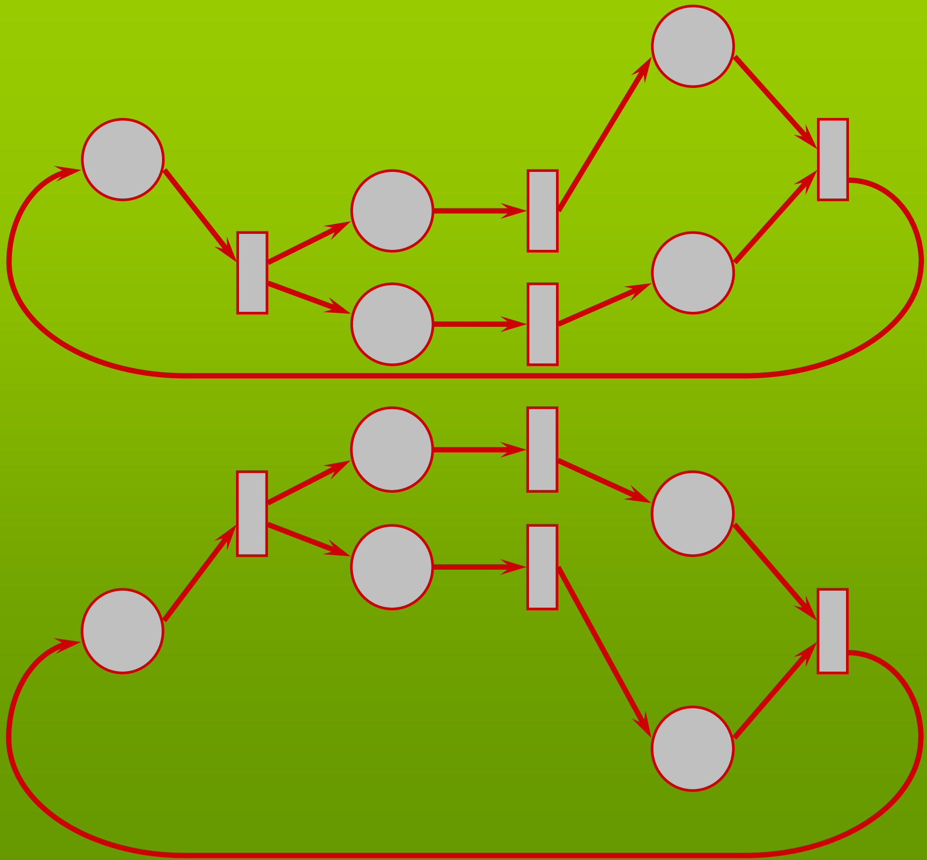
MG reductions

- **The set of all reductions yields a cover of MG components (T-invariants)**



MG reductions

- **The set of all reductions yields a cover of MG components (T-invariants)**



Hack’ s theorem (‘72)

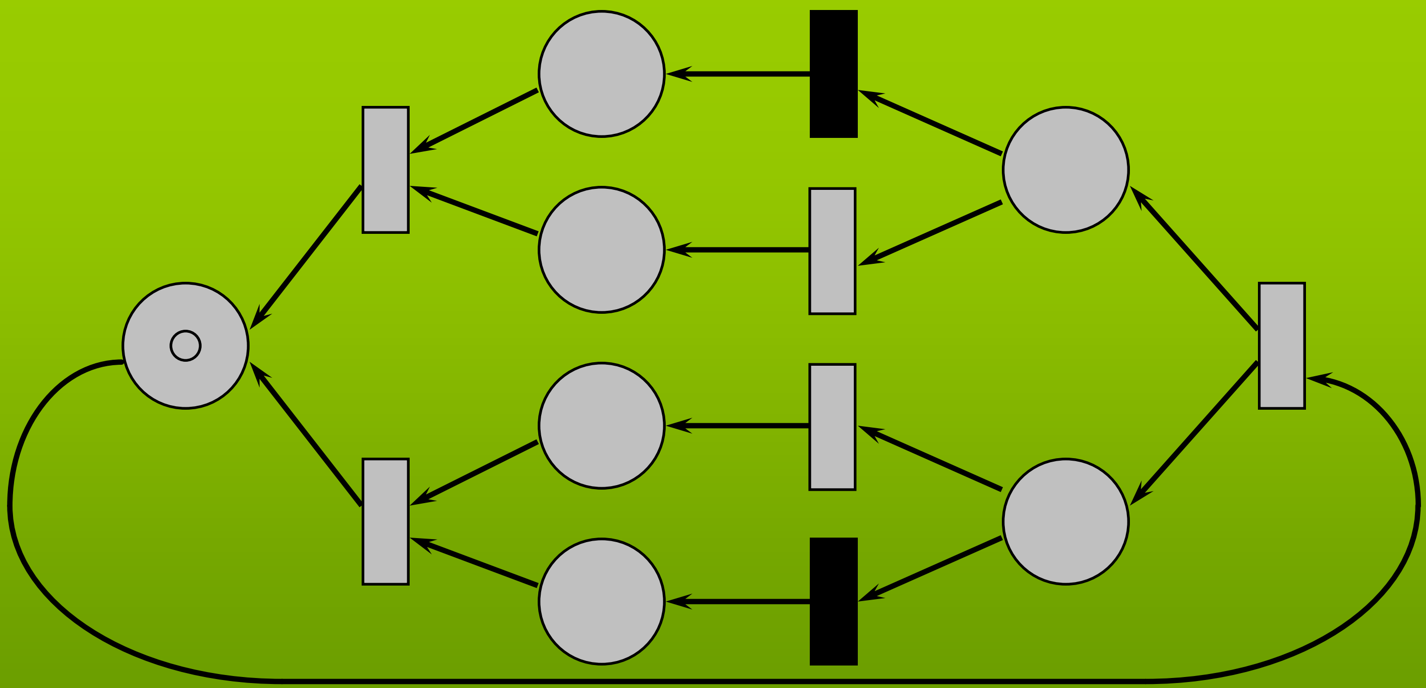
- **Let N be a Free-Choice PN:**
  - **N has a live and safe initial marking (well-formed)**

if and only if

  - **every MG reduction is strongly connected and not empty, and the set of all reductions covers the net**
  - **every SM reduction is strongly connected and not empty, and the set of all reductions covers the net**

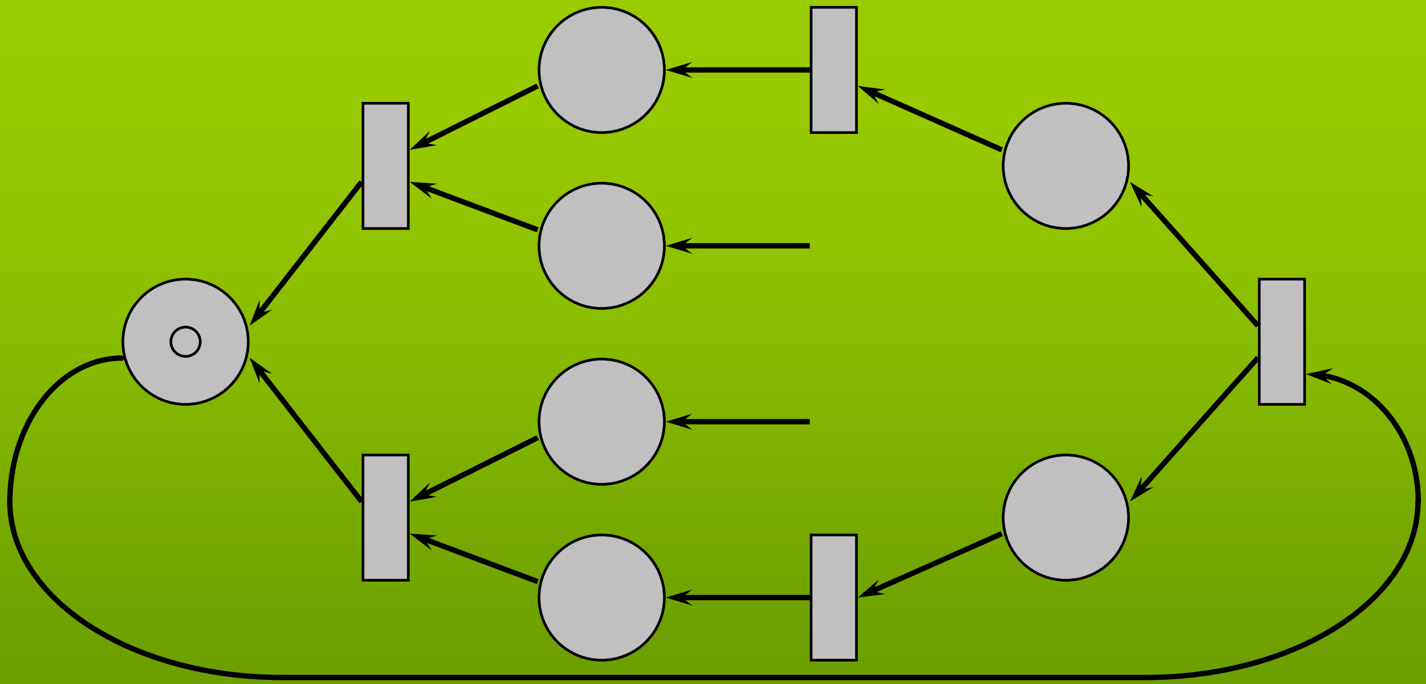
Hack' s theorem

- **Example of non-live (but safe) FCN**



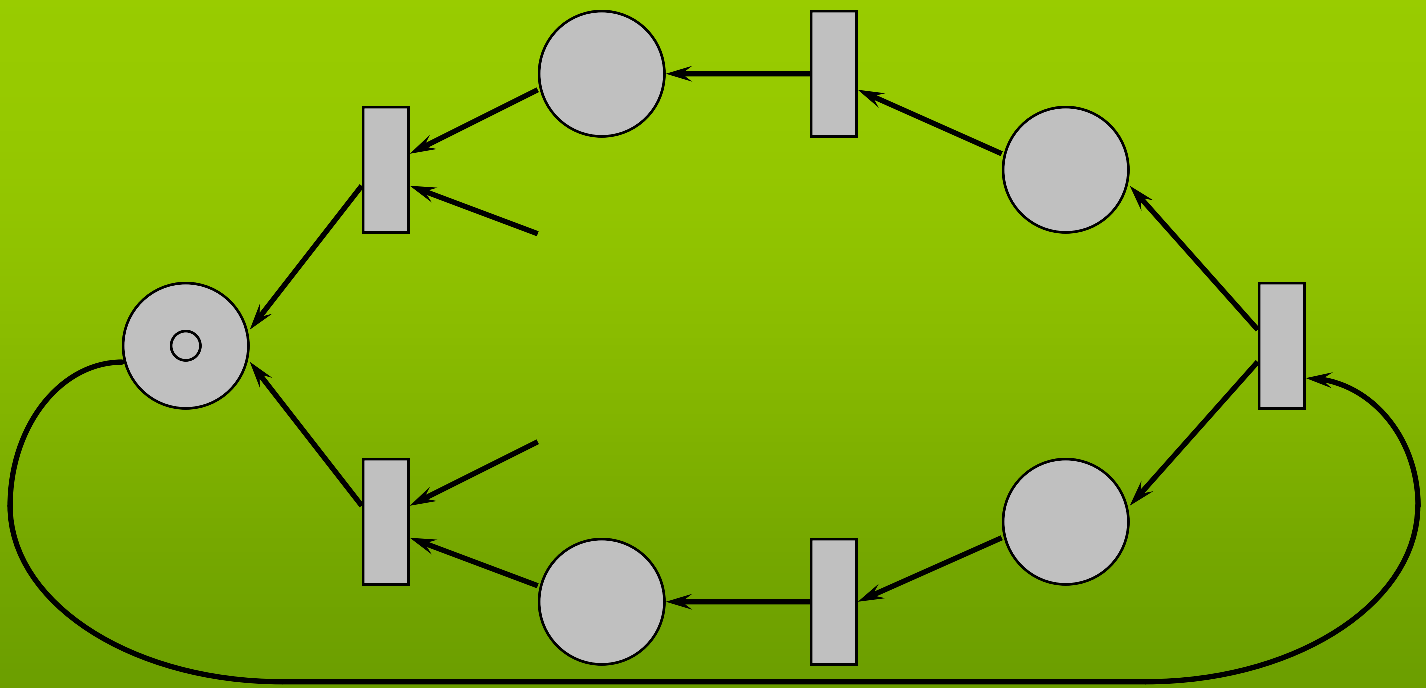
Hack’s theorem

- **Example of non-live (but safe) FCN**



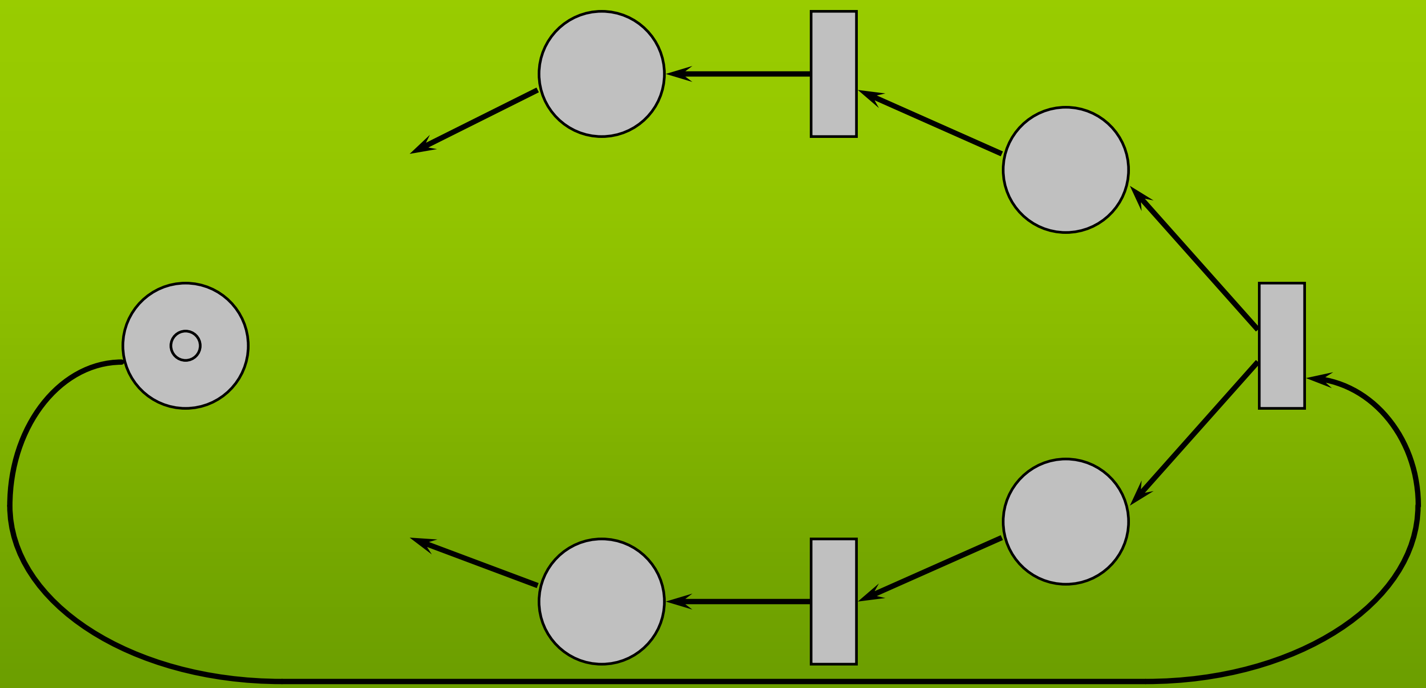
Hack’s theorem

- **Example of non-live (but safe) FCN**



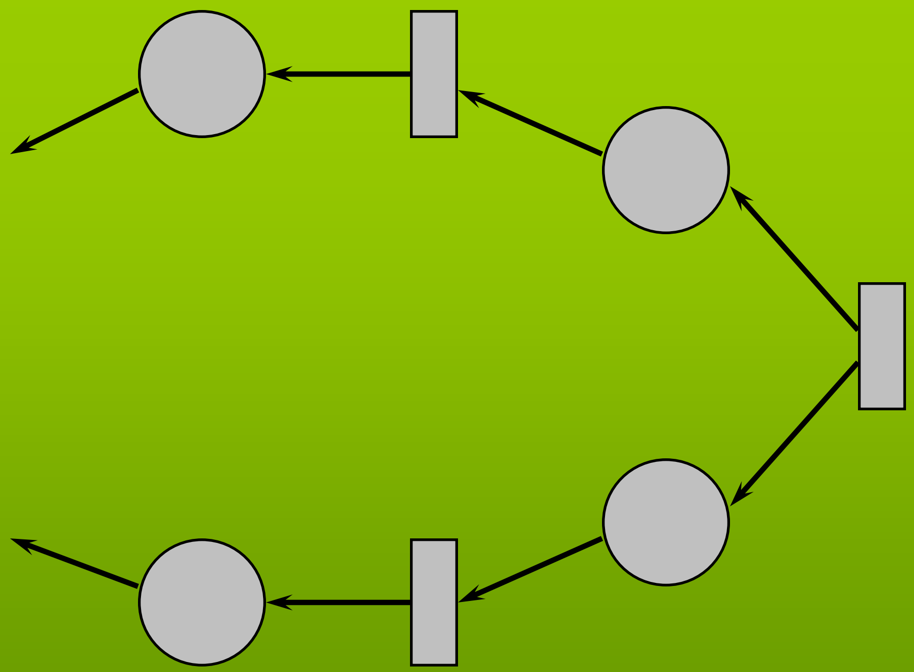
Hack’s theorem

- **Example of non-live (but safe) FCN**



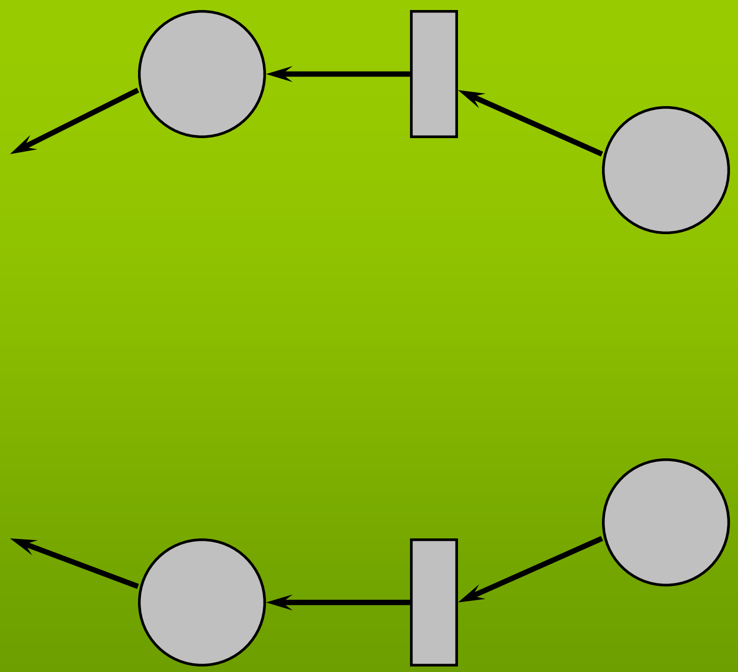
Hack’s theorem

- **Example of non-live (but safe) FCN**



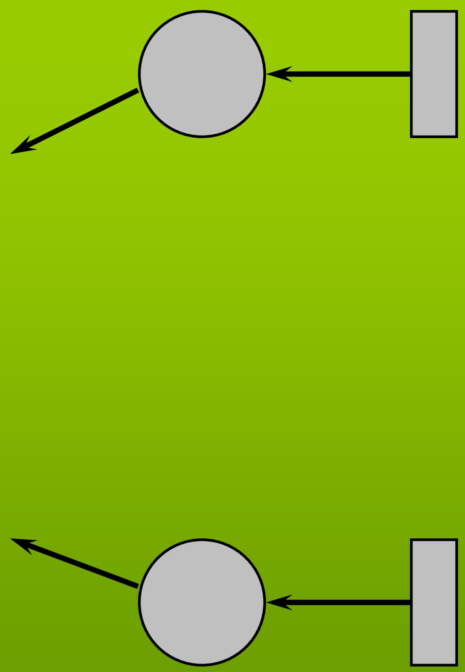
Hack’s theorem

- **Example of non-live (but safe) FCN**



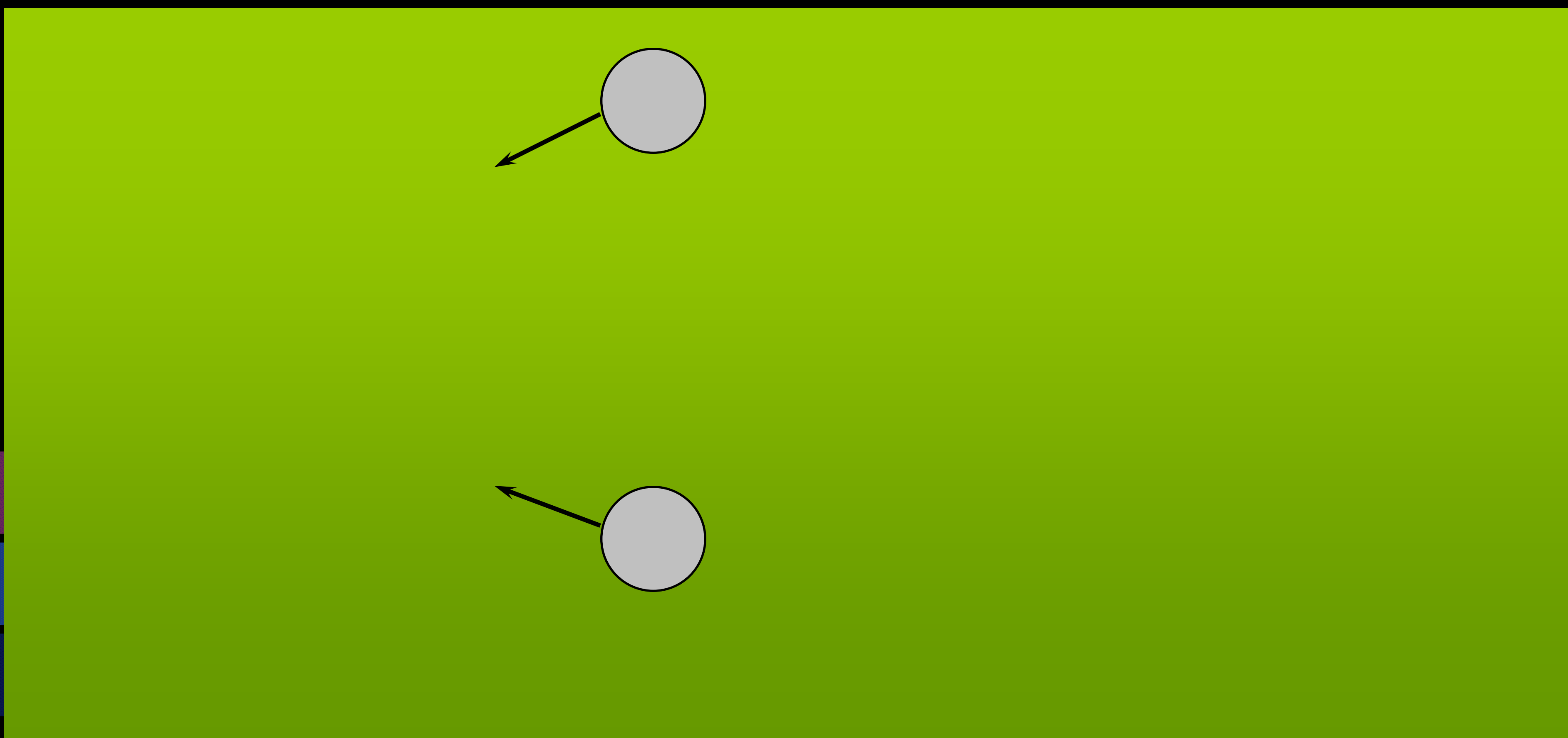
Hack' s theorem

- **Example of non-live (but safe) FCN**



Hack's theorem

- **Example of non-live (but safe) FCN**

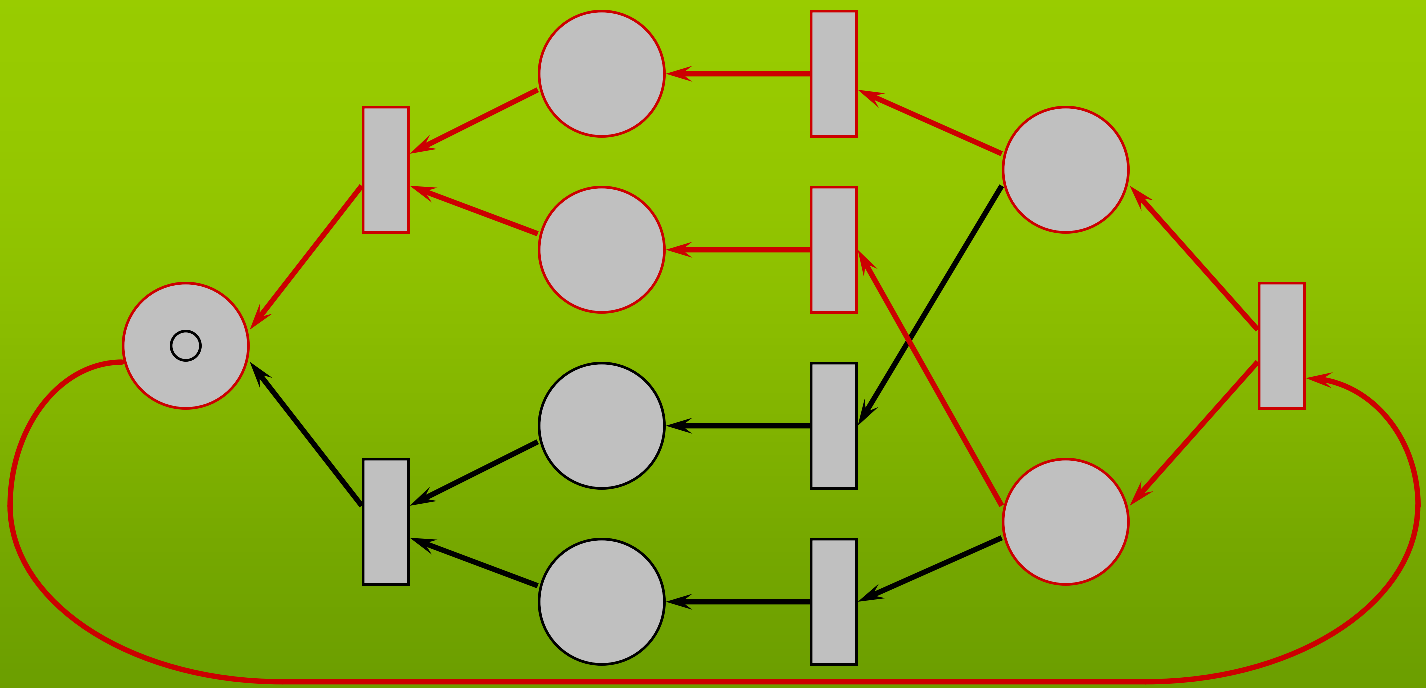


Hack's theorem

- **Example of non-live (but safe) FCN**

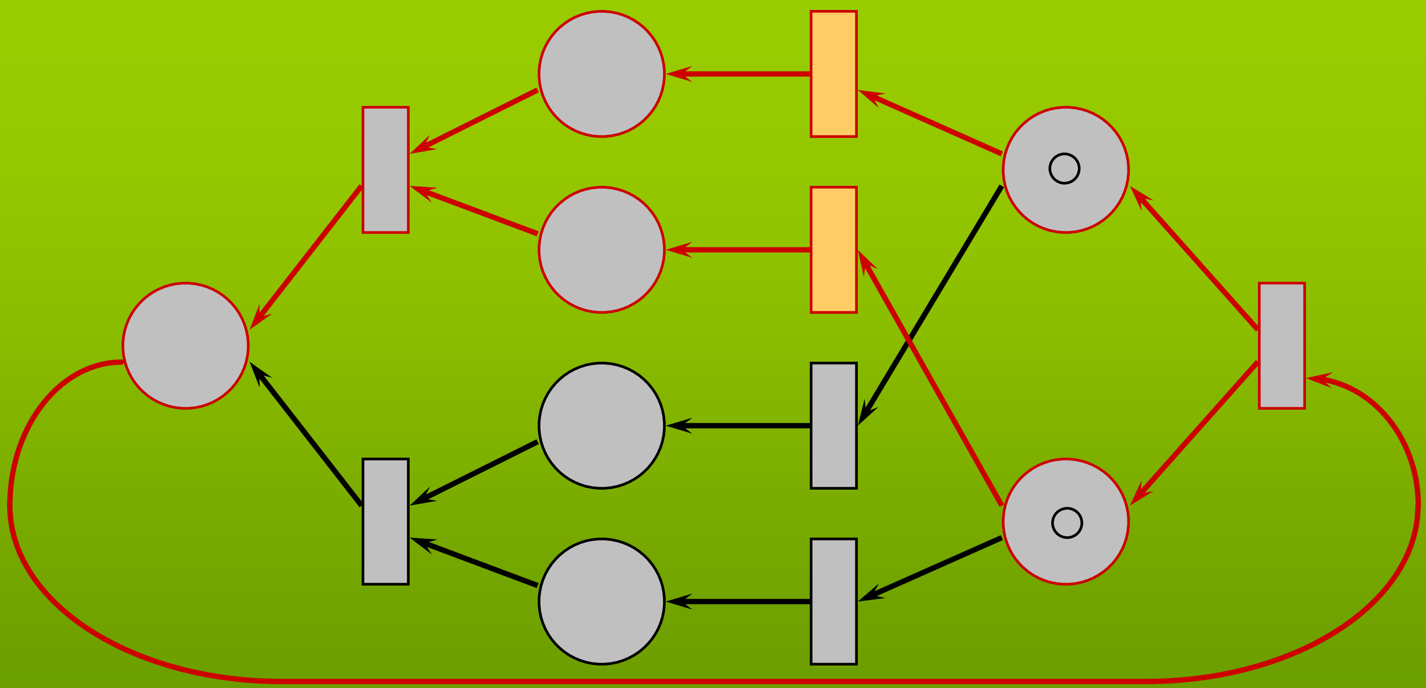
Hack's theorem

- **Example of non-live (but safe) FCN**



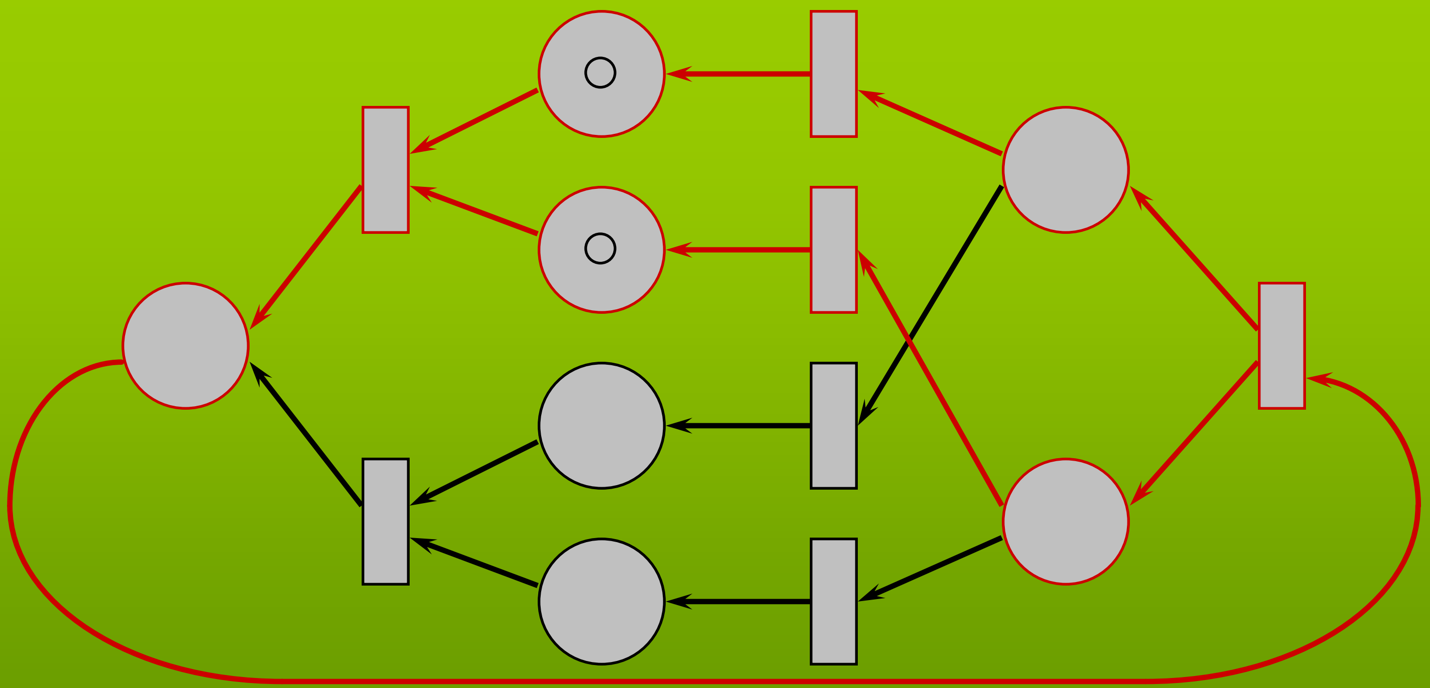
Hack's theorem

- **Example of non-live (but safe) FCN**



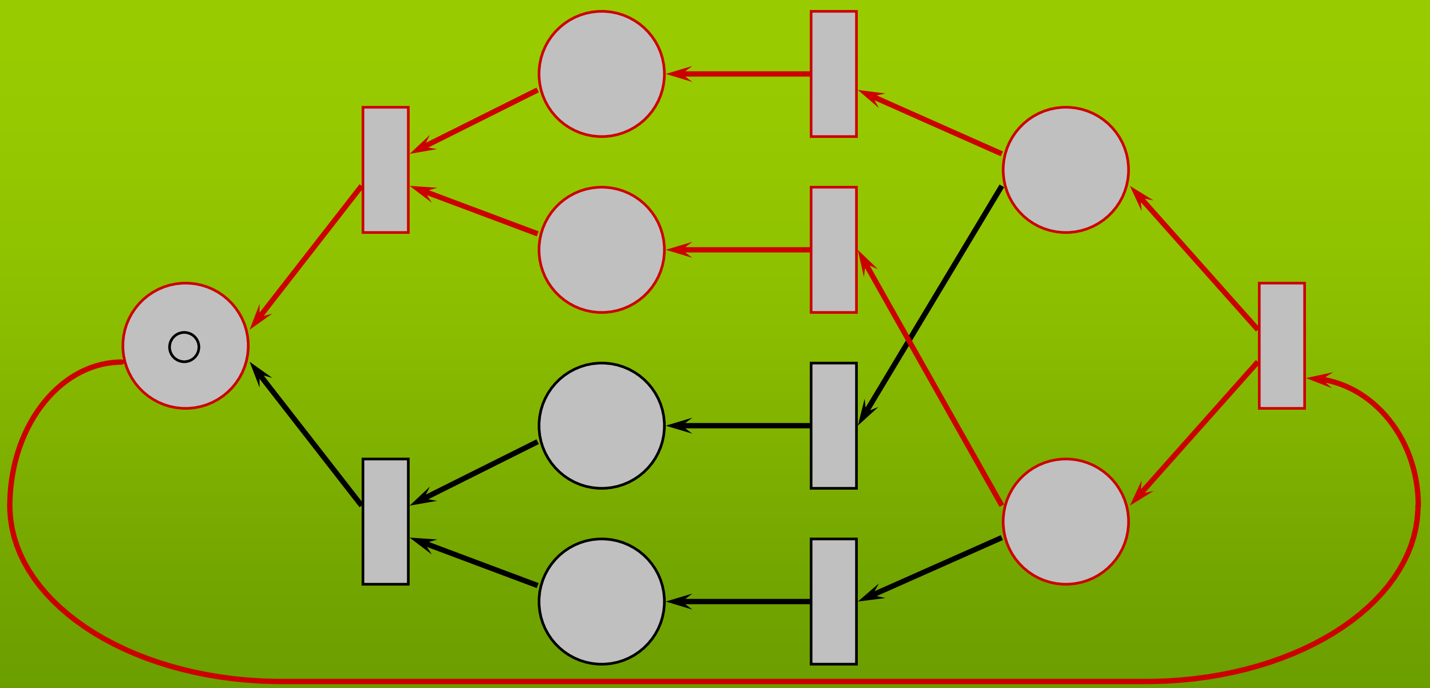
Hack's theorem

- **Example of non-live (but safe) FCN**



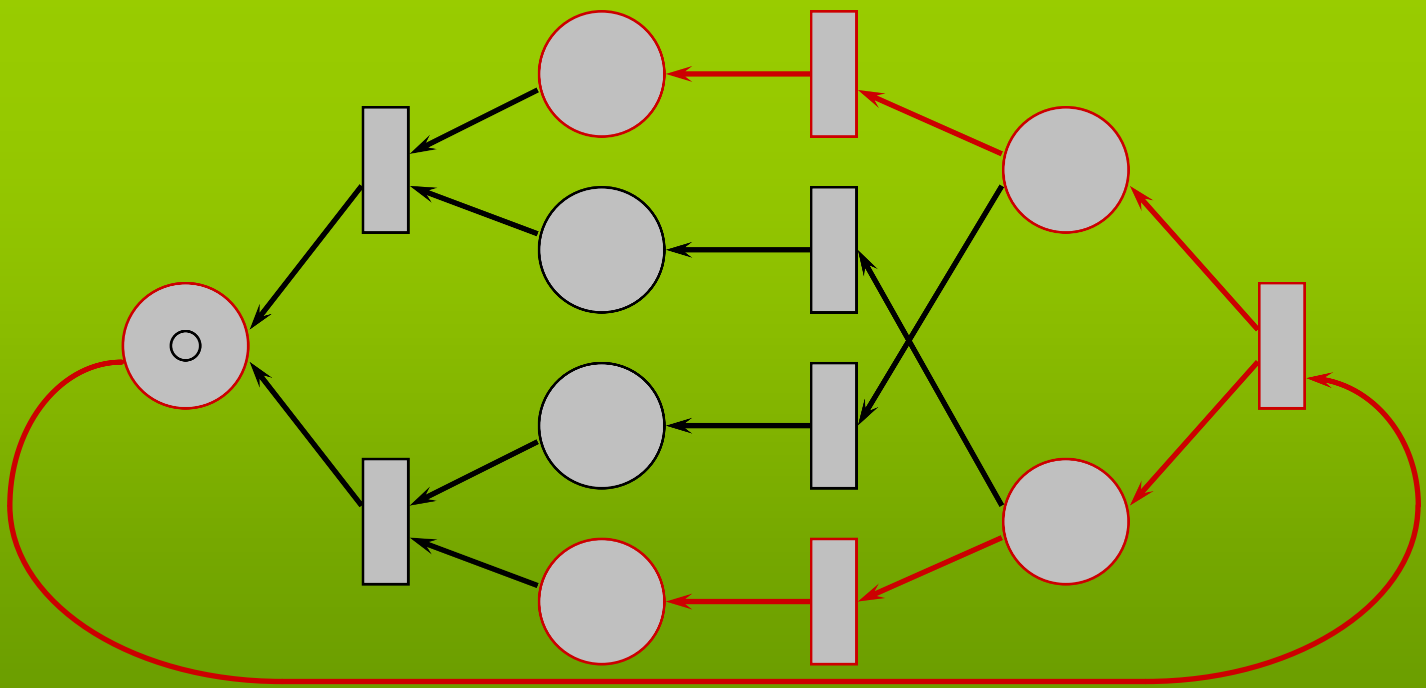
Hack's theorem

- **Example of non-live (but safe) FCN**



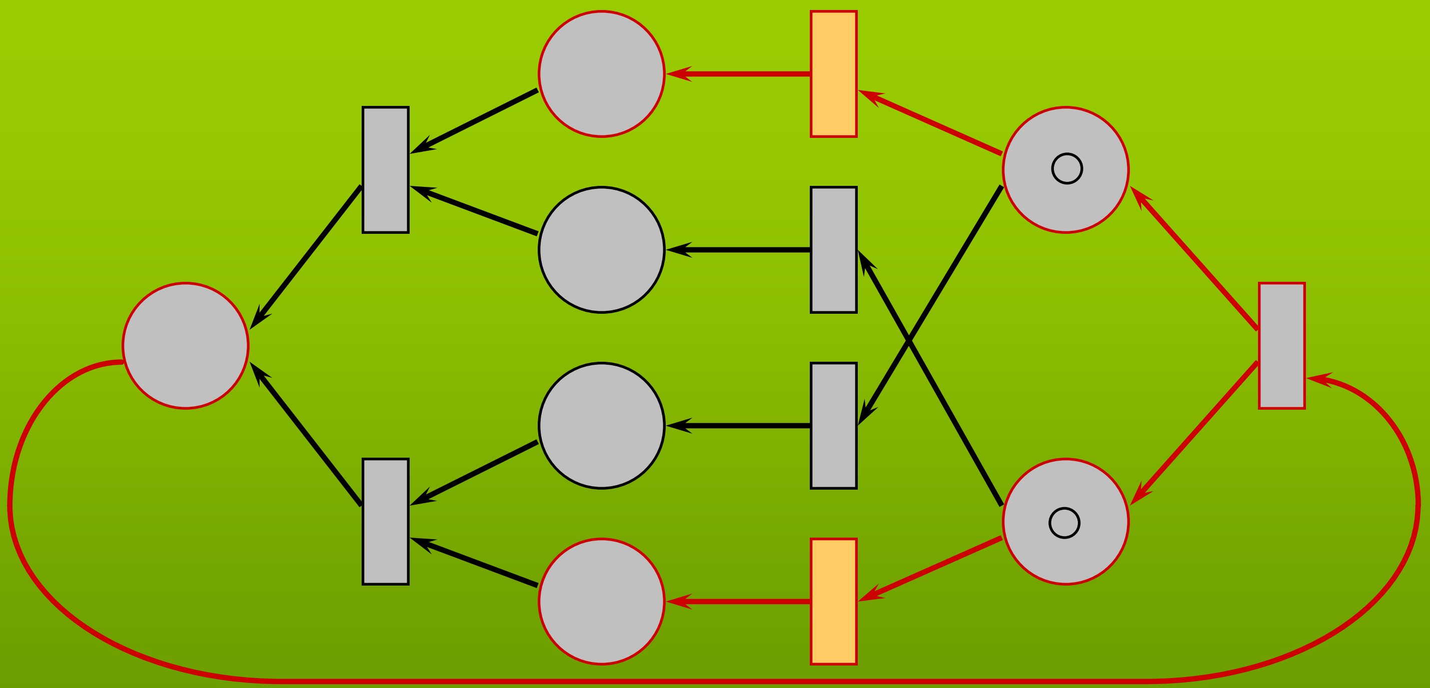
Hack's theorem

- **Example of non-live (but safe) FCN**



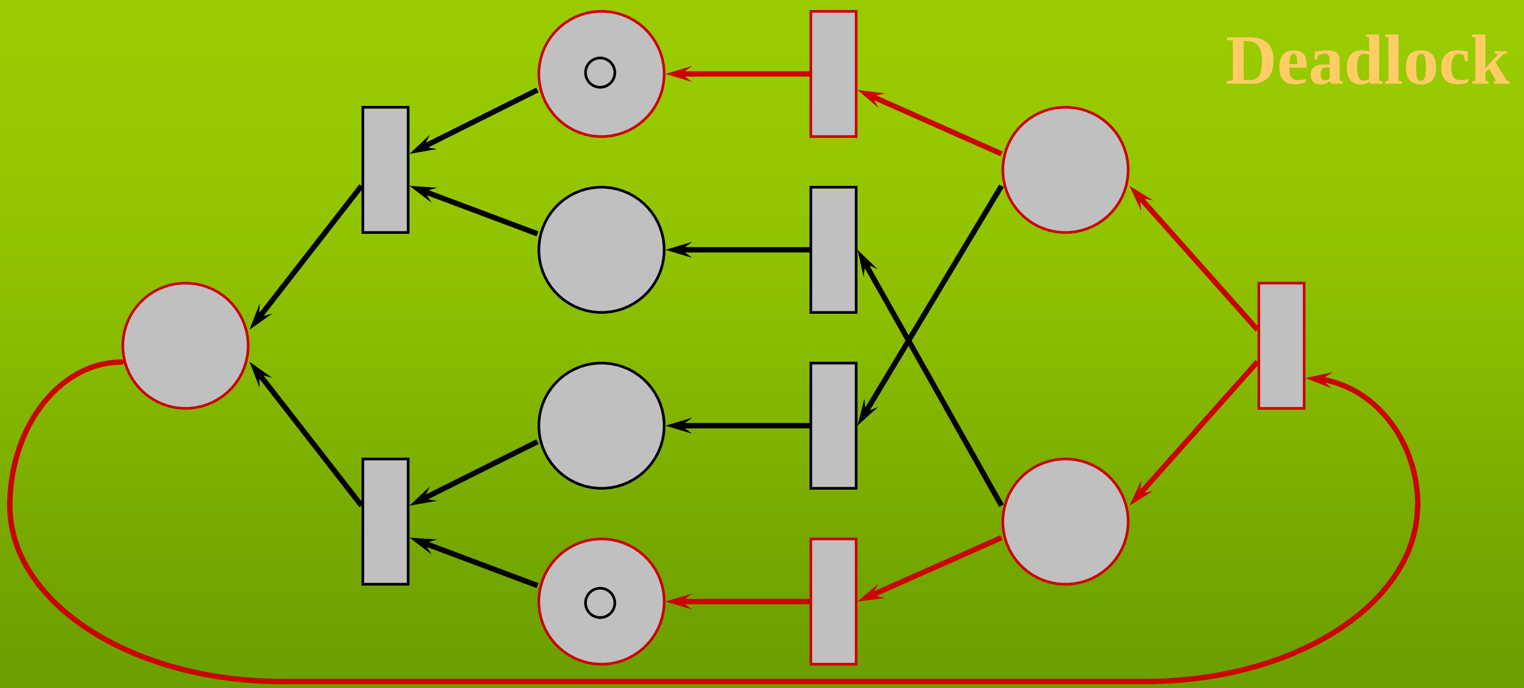
Hack's theorem

- **Example of non-live (but safe) FCN**



Hack's theorem

- **Example of non-live (but safe) FCN**



Summary of LSFC nets

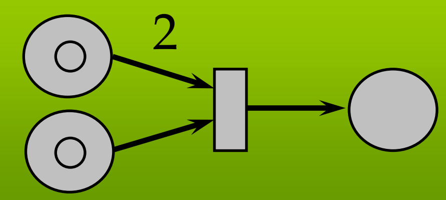
- **Largest class for which structural theory really helps**
- **Structural component analysis may be expensive**

  (exponential number of MG and SM components in the worst case)
- **But…**
  - **number of MG components is generally small**
  - **FC restriction simplifies characterization of behavior**

Petri Net extensions

- **Add interpretation to tokens and transitions**
  - **Colored nets (tokens have value)**
- **Add time**
  - **Time/timed Petri Nets (deterministic delay)**
    - **type (duration, delay)**
    - **where (place, transition)**
  - **Stochastic PNs (probabilistic delay)**
  - **Generalized Stochastic PNs (timed and immediate transitions)**
- **Add hierarchy**
  - **Place Charts Nets**

PNs Summary



- PN Graph: places (buffers), transitions (actions), tokens (data)
- Firing rule: transition enabled if there are enough tokens in each input place
- Properties
  - Structural (consistency, structural boundedness…)
  - Behavioral (reachability, boundedness, liveness…)
- Analysis techniques
  - Structural (only CN or CS): State equations, Invariants
  - Behavioral: coverability tree
- Reachability
- Subclasses: Marked Graphs, State Machines, Free-Choice PNs

References

- T. Murata **Petri Nets: Properties, Analysis and Applications**
- http://www.informatik.uni-hamburg.de/TGI/PetriNets/