

# Petri Nets

ee249 Fall 2000

*Marco Sgroi*

*Most slides borrowed from  
Luciano Lavagno's lecture ee249 (1998)*

1

## Models Of Computation for reactive systems

- **Main MOCs:**
  - **Communicating Finite State Machines**
  - **Dataflow Process Networks**
  - **Discrete Event**
  - **Codesign Finite State Machines**
  - **Petri Nets**
- **Main languages:**
  - **StateCharts**
  - **Esterel**
  - **Dataflow networks**

2

## Outline

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

3

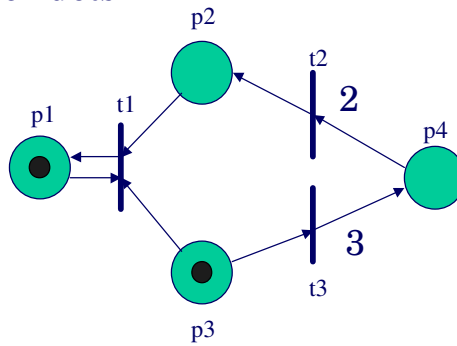
## Petri Nets (PNs)

- **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**
- **Asynchronous model (partial ordering)**
- **Main drawback: no hierarchy**

4

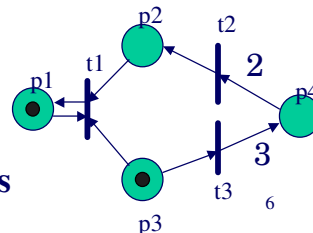
## Petri Net Graph

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



## Petri Net

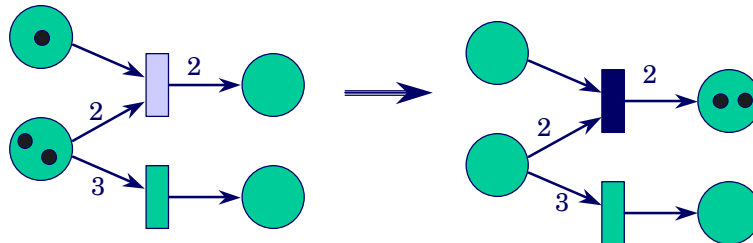
- A PN  $(N, M_0)$  is a Petri Net Graph  $N$ 
  - **places:** represent distributed state by holding tokens
    - marking (state)  $M$  is an  $n$ -vector  $(m_1, m_2, m_3, \dots)$ , where  $m_i$  is the non-negative number of tokens in place  $p_i$ .
    - initial marking  $(M_0)$  is initial state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



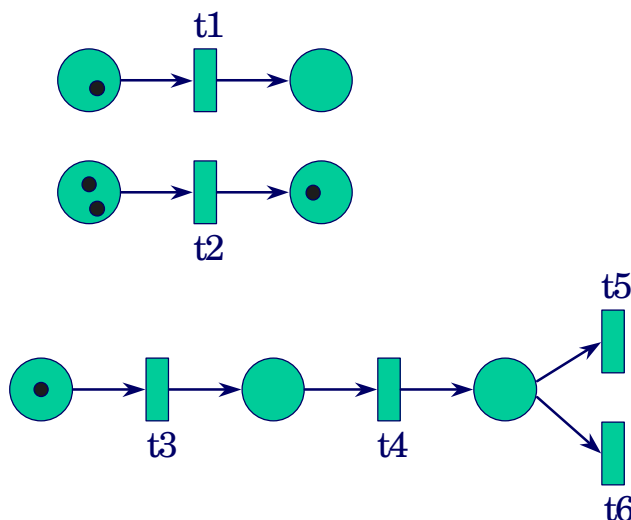
Places/Transition: conditions/events

## Transition firing rule

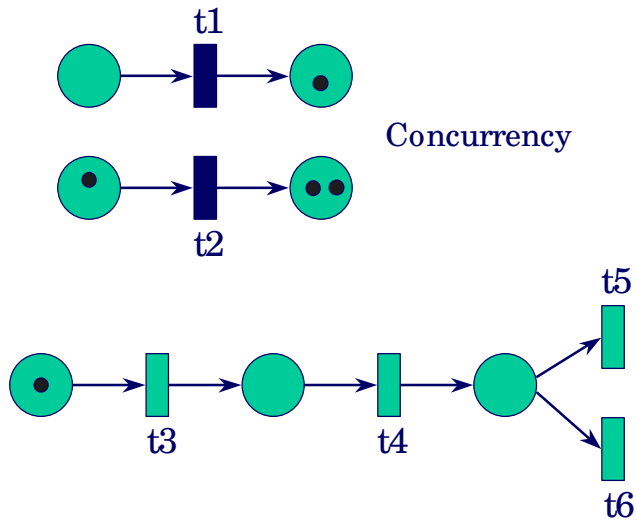
- A marking is changed according to the following rules:
  - A transition is **enabled** if there are enough tokens in each input place
  - An enabled transition **may or may not fire**
  - The firing of a transition modifies marking by **consuming** tokens from the input places and **producing** tokens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



## Concurrency, causality, choice

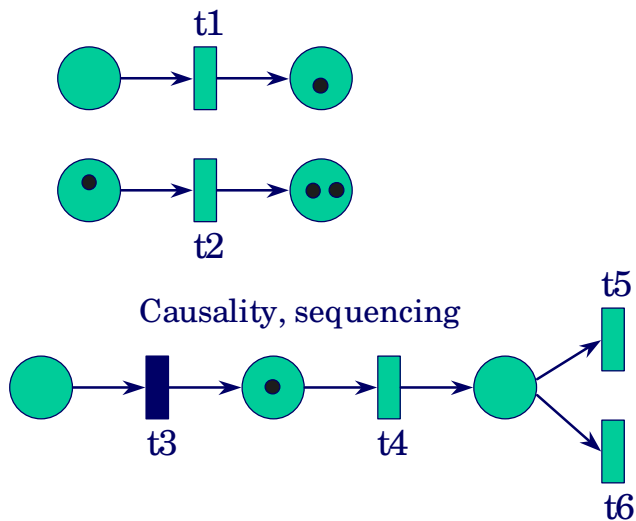


## Concurrency, causality, choice



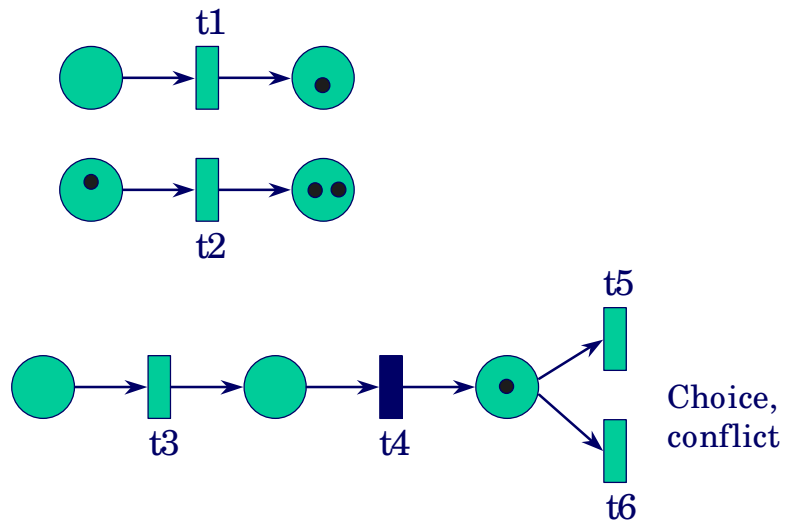
9

## Concurrency, causality, choice



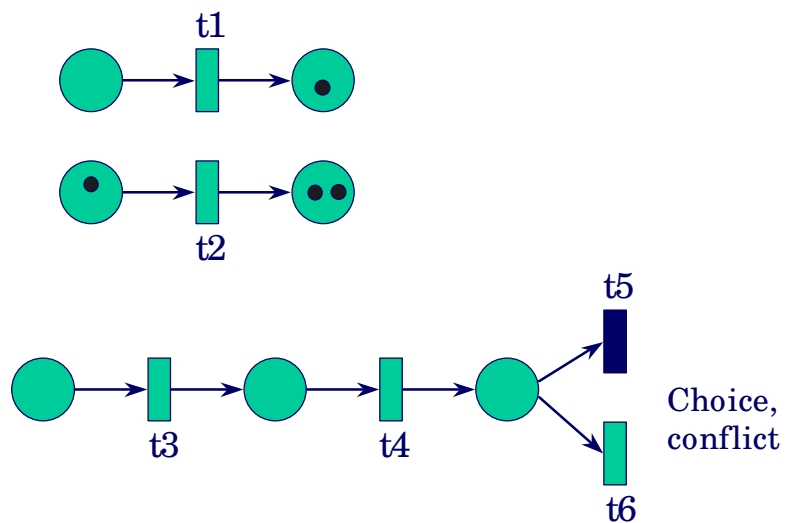
10

## Concurrency, causality, choice



11

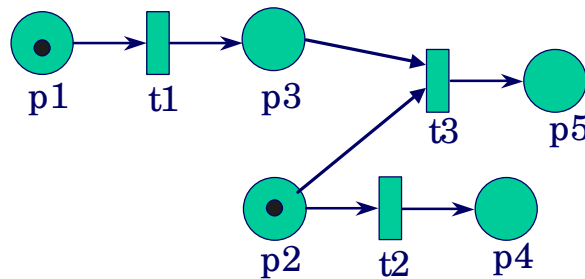
## Concurrency, causality, choice



12

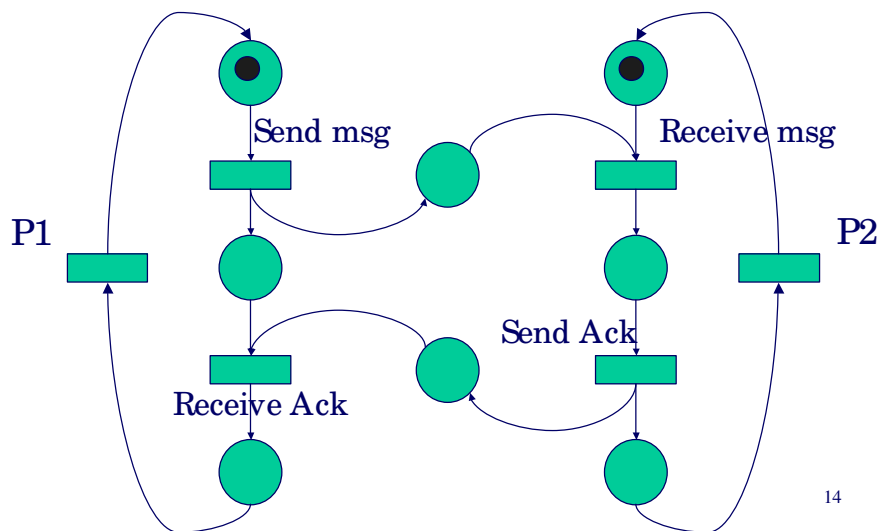
## Confusion

- **t1 and t2 are concurrent but their firing order is not irrelevant for conflict resolution (not local choice)**
- **From (1,1,0,0,0):**
  - solving a conflict (t1,t2)      (0,0,0,0,1),(0,0,1,1,0)
  - not solving a conflict (t2,t1)    (0,0,1,1,0)

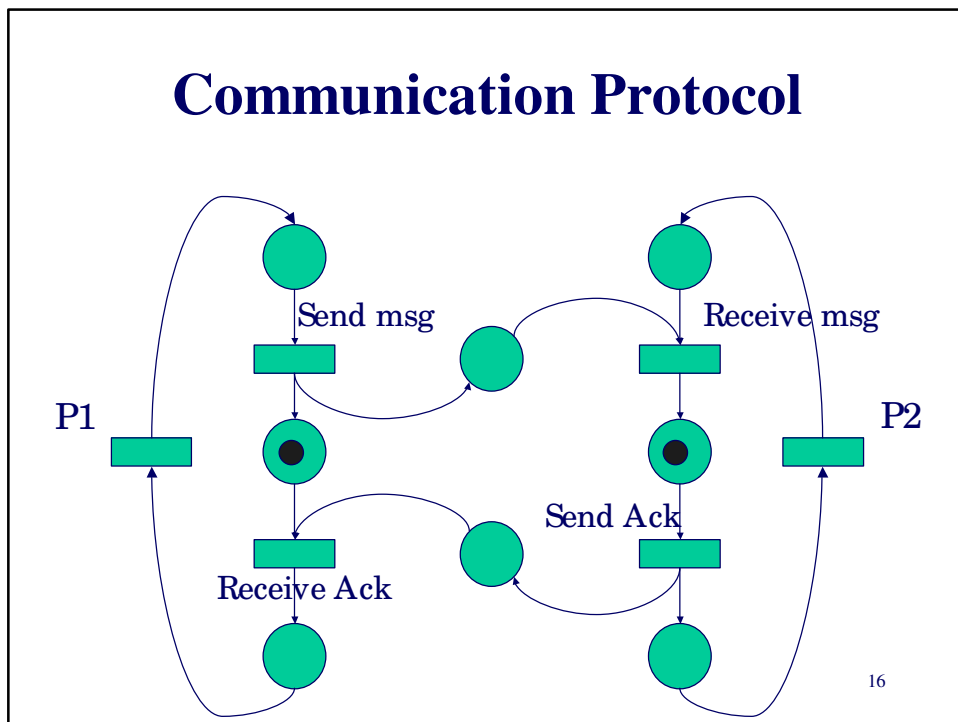
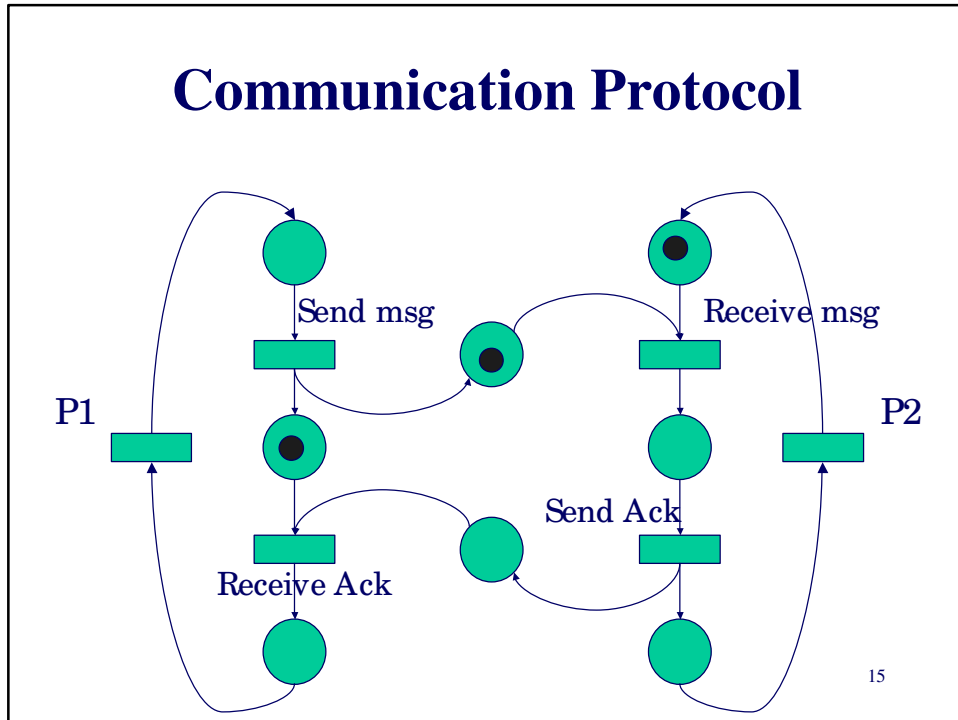


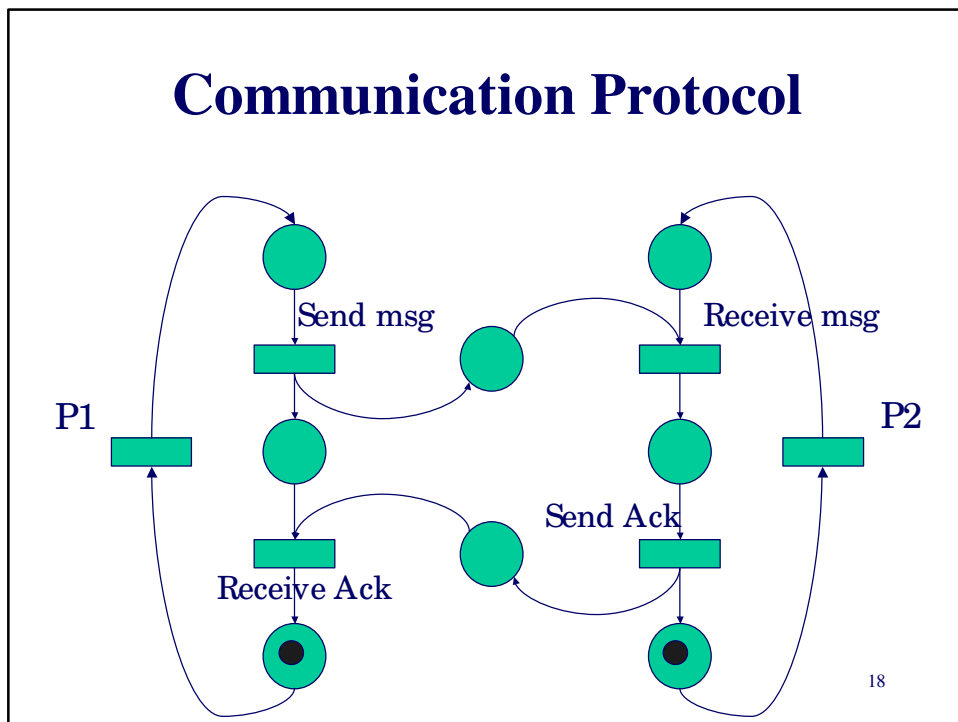
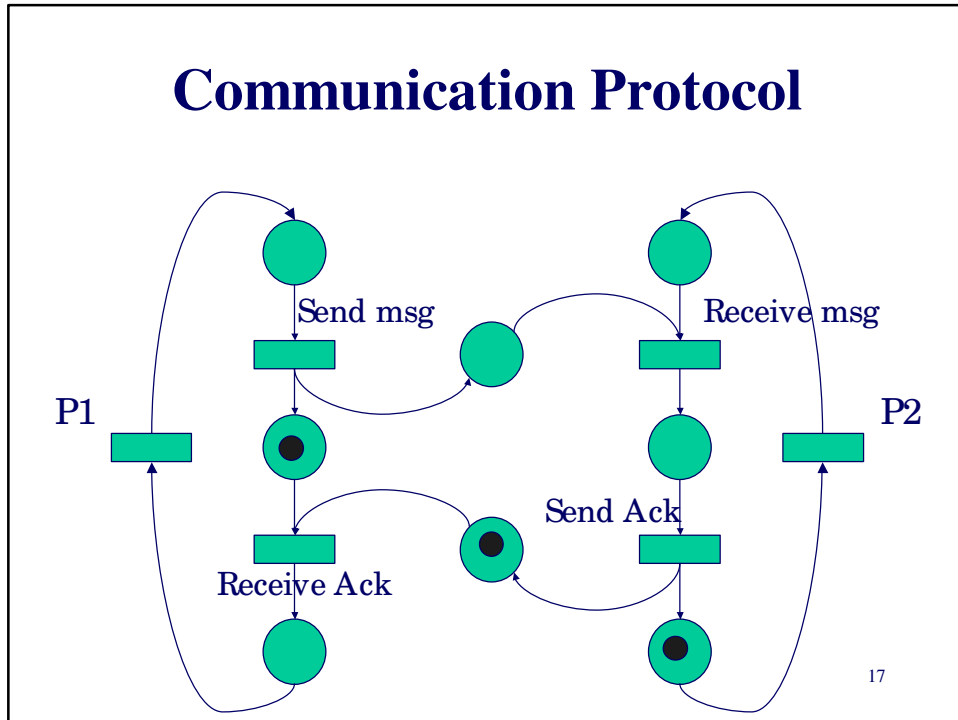
13

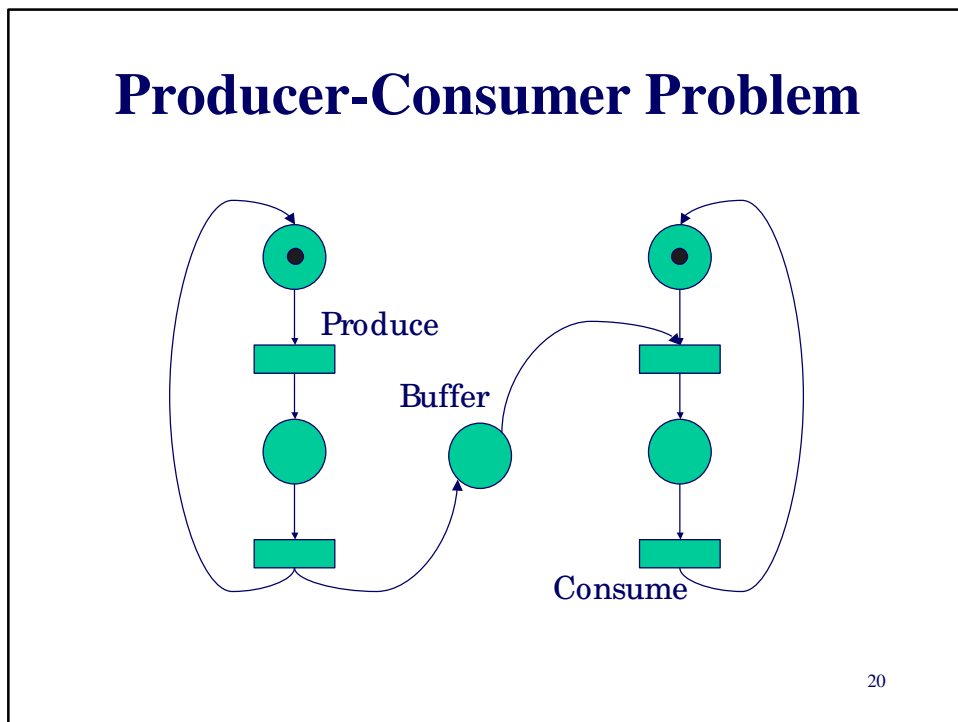
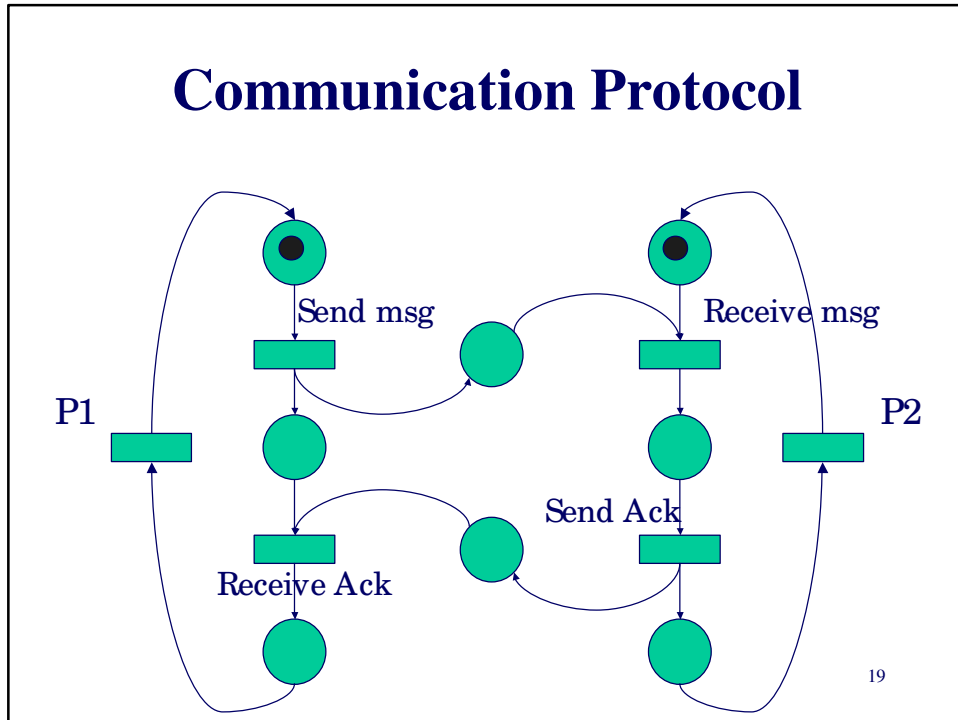
## Communication Protocol



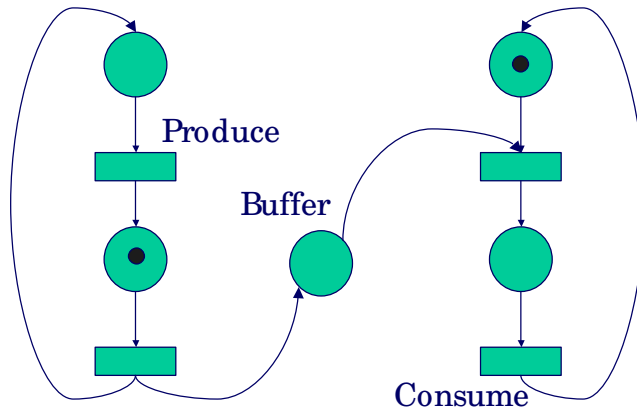
14





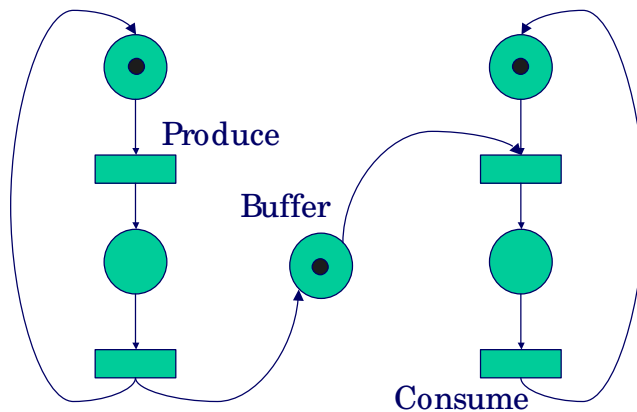


## Producer-Consumer Problem



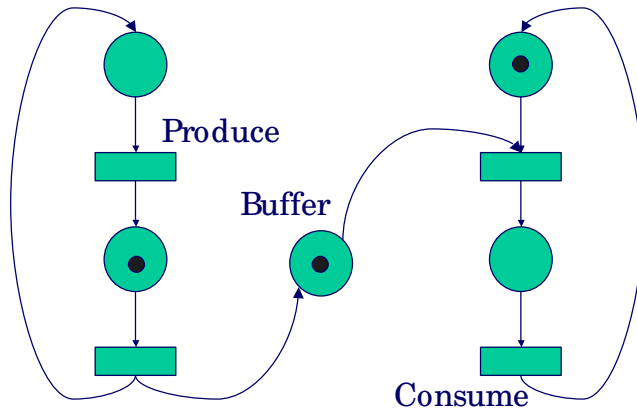
21

## Producer-Consumer Problem



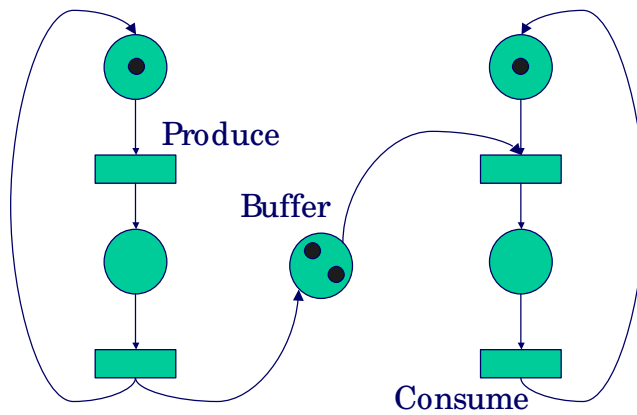
22

## Producer-Consumer Problem



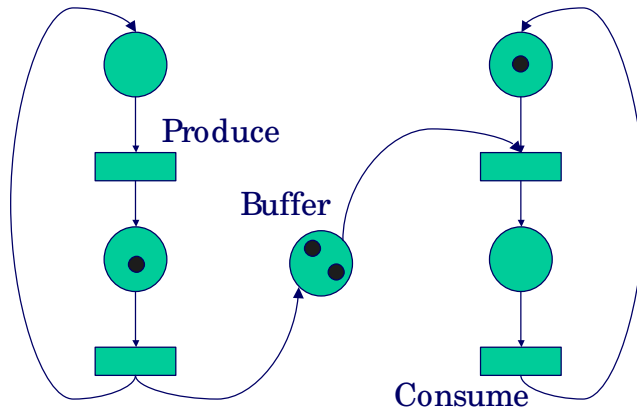
23

## Producer-Consumer Problem



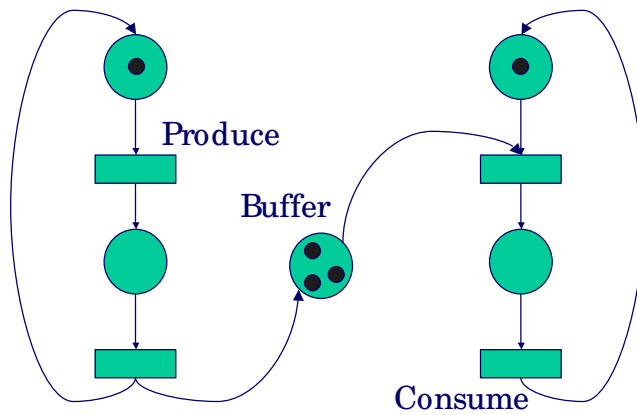
24

## Producer-Consumer Problem



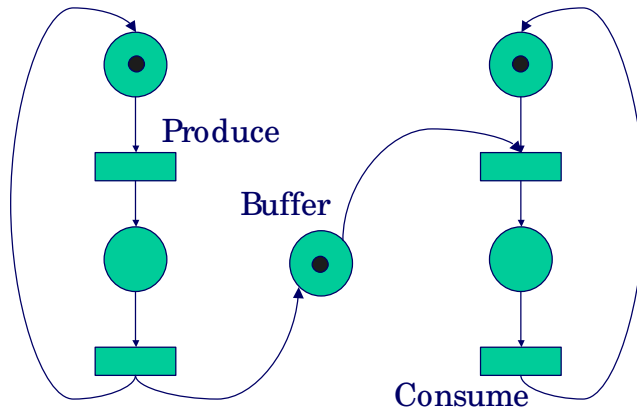
25

## Producer-Consumer Problem



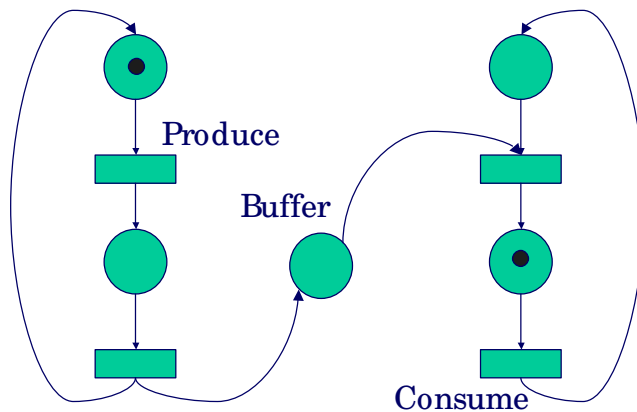
26

## Producer-Consumer Problem



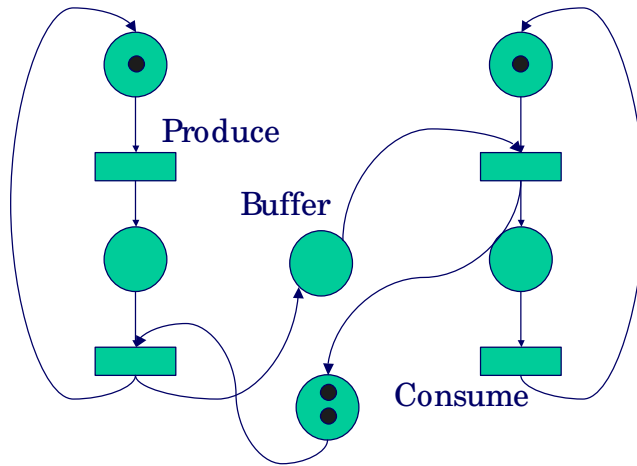
27

## Producer-Consumer Problem



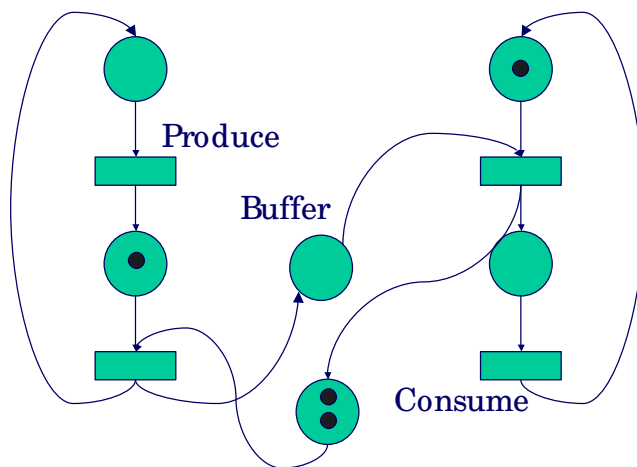
28

## Producer-Consumer Problem



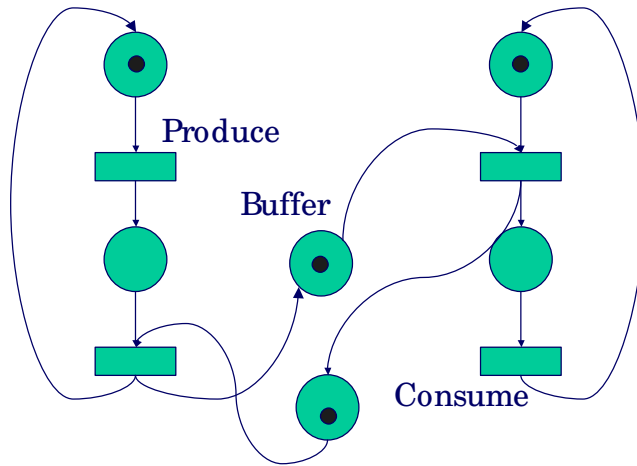
29

## Producer-Consumer Problem



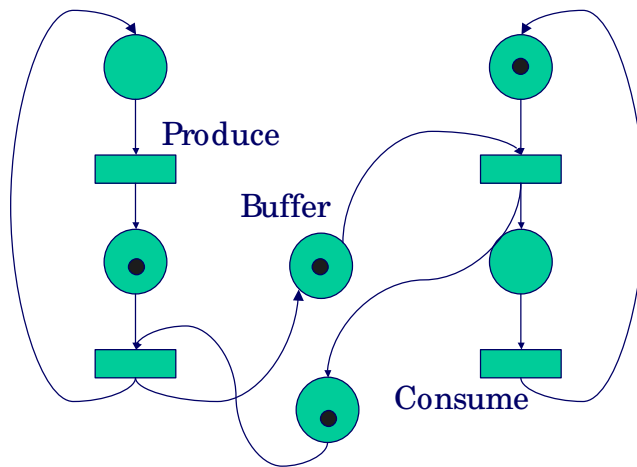
30

## Producer-Consumer Problem



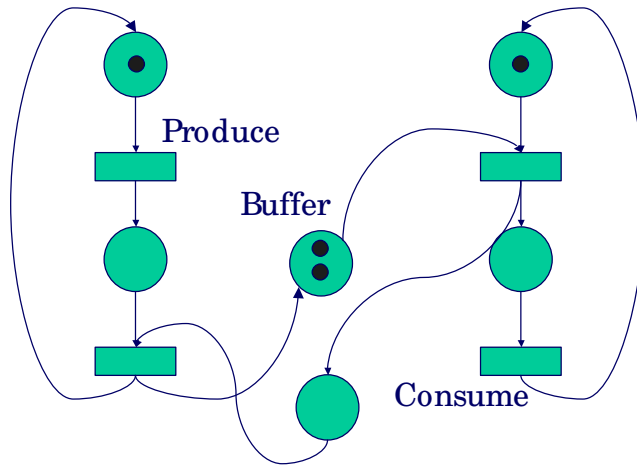
31

## Producer-Consumer Problem



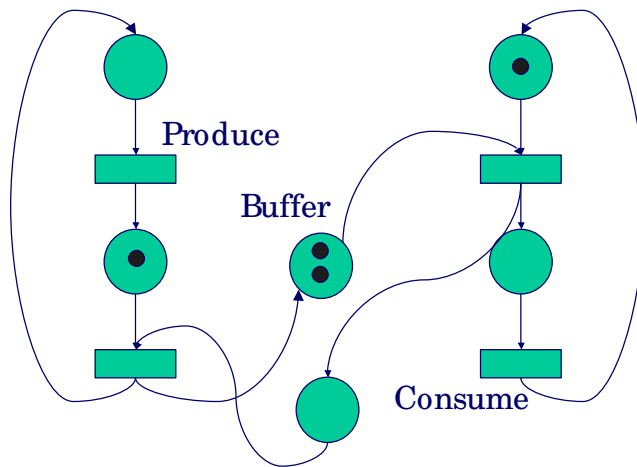
32

## Producer-Consumer Problem



33

## Producer-Consumer Problem

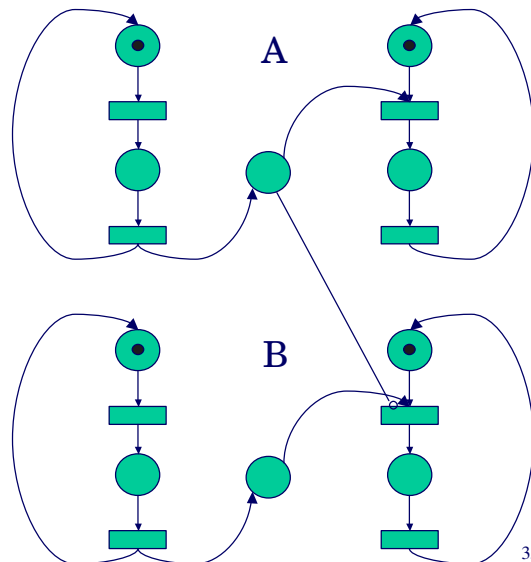


34

## Producer-Consumer with priority

Consumer B can  
consume only if  
buffer A is empty

Inhibitor arcs



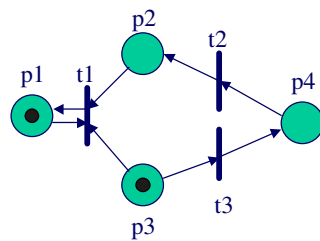
## PN properties

- **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

36

## Reachability

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



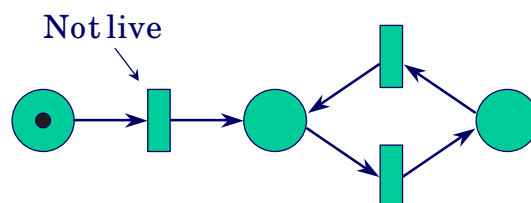
$M_0 = (1,0,1,0)$   
 $M = (1,1,0,0)$

$M_0 = (1,0,1,0)$   
 $\downarrow t_3$   
 $M_1 = (1,0,0,1)$   
 $\downarrow t_2$   
 $M = (1,1,0,0)$

37

## Liveness

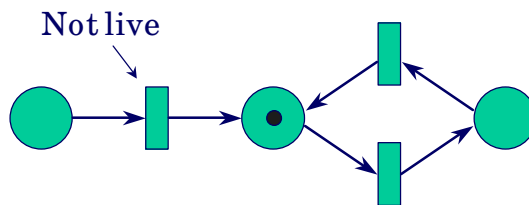
- **Liveness**: from any marking any transition can become fireable
  - Liveness implies deadlock freedom, not viceversa



38

## Liveness

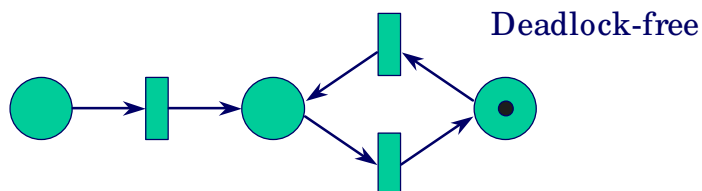
- **Liveness**: from any marking any transition can become fireable
  - Liveness implies deadlock freedom, not viceversa



39

## Liveness

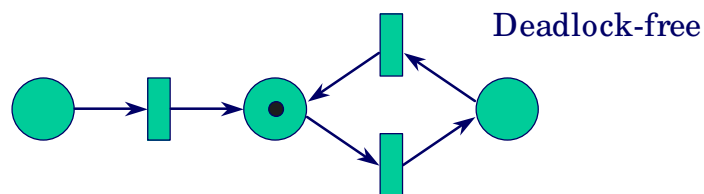
- **Liveness**: from any marking any transition can become fireable
  - Liveness implies deadlock freedom, not viceversa



40

## Liveness

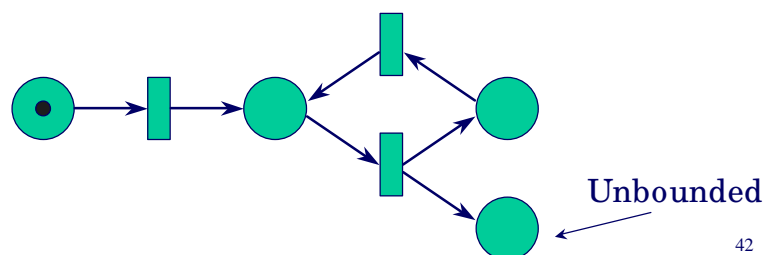
- **Liveness**: from any marking any transition can become fireable
  - Liveness implies deadlock freedom, not viceversa



41

## Boundedness

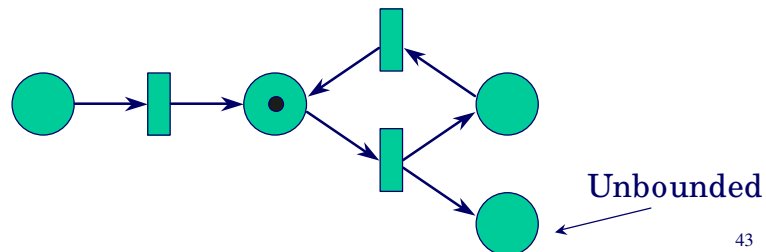
- **Boundedness**: the number of tokens in any place cannot grow indefinitely
  - (1-bounded also called *safe*)
  - Application: places represent buffers and registers (check there is no overflow)



42

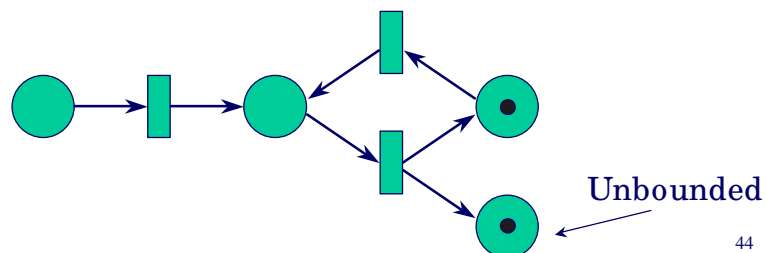
## Boundedness

- **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)



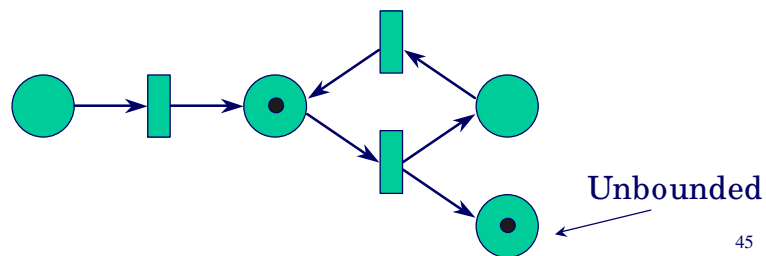
## Boundedness

- **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

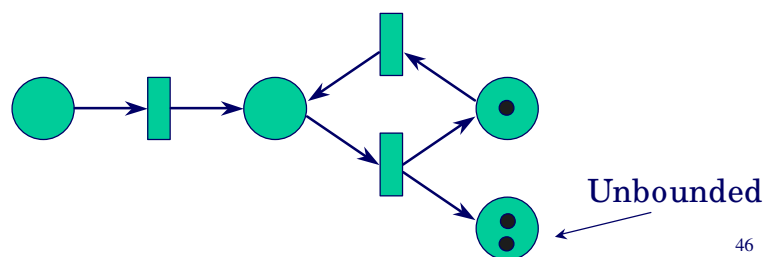
- **Boundedness:** the number of tokens in any place cannot grow indefinitely
  - (1-bounded also called *safe*)
  - Application: places represent buffers and registers (check there is no overflow)



45

## Boundedness

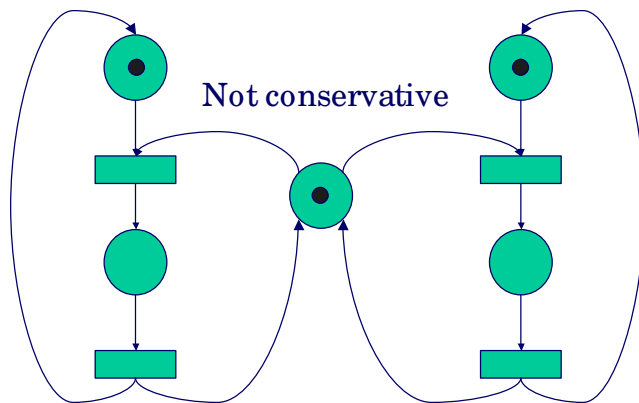
- **Boundedness:** the number of tokens in any place cannot grow indefinitely
  - (1-bounded also called *safe*)
  - Application: places represent buffers and registers (check there is no overflow)



46

## Conservation

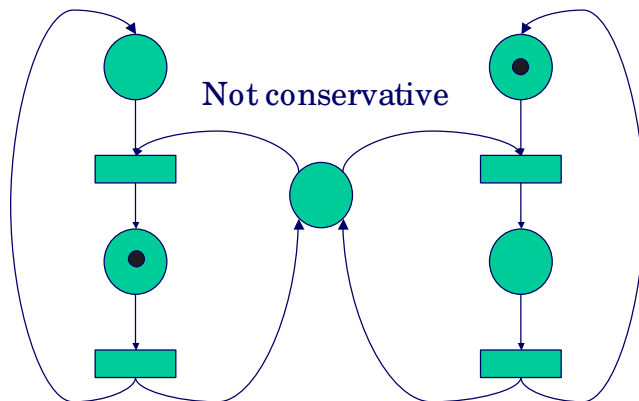
- **Conservation:** the total number of tokens in the net is constant



47

## Conservation

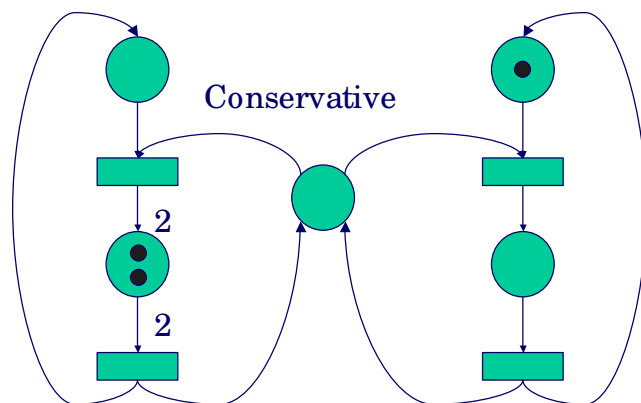
- **Conservation:** the total number of tokens in the net is constant



48

## Conservation

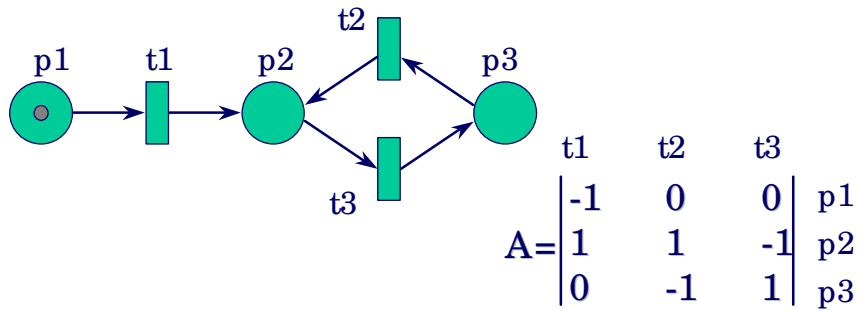
- **Conservation:** the total number of tokens in the net is constant



## Analysis techniques

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

### Incidence Matrix



- Necessary condition for marking M to be reachable from initial marking M<sub>0</sub>:

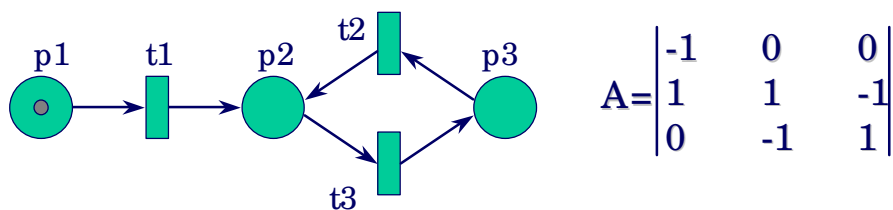
there exists firing vector v s.t.:

$$M = M_0 + A v$$

51

### State equations

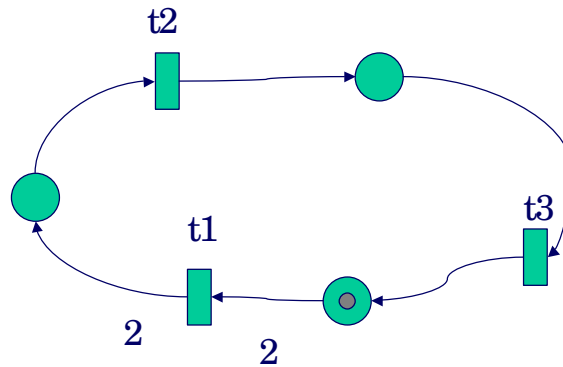
- E.g. reachability of M = |0 0 1|<sup>T</sup> from M<sub>0</sub> = |1 0 0|<sup>T</sup>



$$v_1 = \begin{array}{|c} 1 \\ 0 \\ 1 \end{array} = \begin{array}{|c} 0 \\ 0 \\ 1 \end{array} = \begin{array}{|c} 1 \\ 0 \\ 0 \end{array} + \begin{array}{|c} -1 & 0 & 0 \\ 1 & 1 & -1 \\ 0 & -1 & 1 \end{array} \begin{array}{|c} 1 \\ 0 \\ 1 \end{array}$$

but also v<sub>2</sub> = |1 1 2|<sup>T</sup> or any v<sub>k</sub> = |1 (k) (k+1)|<sup>T</sup>

## Necessary Condition only



Firing vector: (1,2,2)

Deadlock!!

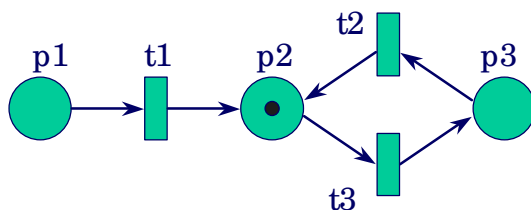
53

## 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.  $x = | 0 \ 1 \ 1 |^T$

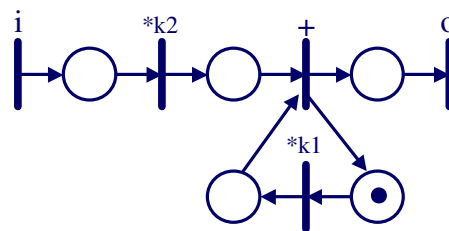


$$A = \begin{vmatrix} -1 & 0 & 0 \\ 1 & 1 & -1 \\ 0 & -1 & 1 \end{vmatrix}$$

54

## Application of T-invariants

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

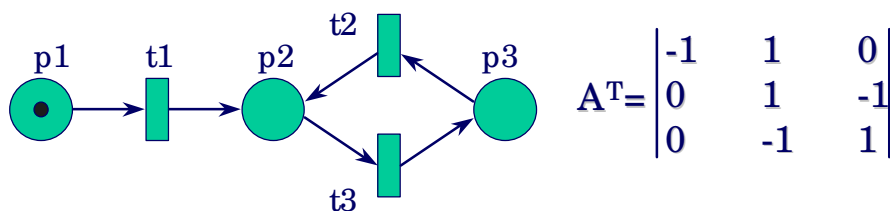


T-invariant: (1,1,1,1,1)  
 Schedule:  $i *k2 *k1 + o$

55

## 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 ( $yM = yM'$ )
    - e.g.  $y = 1\ 1\ 1\ 1^T$



56

## Application of S-invariants

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

57

## 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**

58

## Coverability Tree

- Build a (finite) tree representation of the markings

### Karp-Miller algorithm

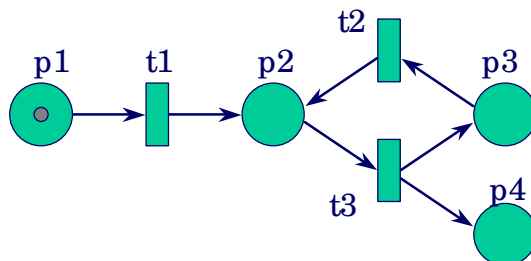
- Label initial marking  $M_0$  as the root of the tree and tag it as *new*
- While new markings exist do:
  - select a new marking  $M$
  - if  $M$  is identical to a marking on the path from the root to  $M$ , then tag  $M$  as *old* and go to another new marking
  - if no transitions are enabled at  $M$ , 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  $M'(p) \geq M''(p)$  for each place  $p$  and  $M'$  is different from  $M''$ , then replace  $M'(p)$  by  $\omega$  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*.

59

## Coverability Tree

- Boundedness is decidable  
with *coverability tree*

1000



60

### Coverability Tree

- Boundedness is decidable**  
with *coverability tree*

1000  
 ↓ t1  
 0100

61

### Coverability Tree

- Boundedness is decidable**  
with *coverability tree*

1000  
 ↓ t1  
 0100  
 ↓ t3  
 0011

62

### Coverability Tree

- Boundedness is decidable**  
with *coverability tree*

```

    graph LR
      p1((p1)) -- t1 --> p2((p2))
      p2 -- t2 --> p3((p3))
      p2 -- t3 --> p4((p4))
      p3 -- t2 --> p2
      p4 -- t3 --> p2
  
```

1000  
↓ t1  
0100  
↓ t3  
0011  
↓ t2  
0101

63

### Coverability Tree

- Boundedness is decidable**  
with *coverability tree*

```

    graph LR
      p1((p1)) -- t1 --> p2((p2))
      p2 -- t2 --> p3((p3))
      p2 -- t3 --> p4((p4))
      p3 -- t2 --> p2
      p4 -- t3 --> p2
  
```

1000  
↓ t1  
0100  
↓ t3  
0011  
↓ t2  
0100

**Cannot solve the reachability and liveness problems**

64

### Coverability Tree

- Boundedness is decidable with coverability tree**

1000  
 ↓ t1  
 0100  
 ↓ t3  
 0011  
 ↓ t2  
 0100

**Cannot solve the reachability and liveness problems**

65

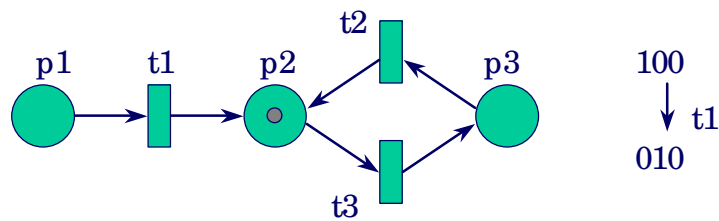
### Reachability graph

100

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

66

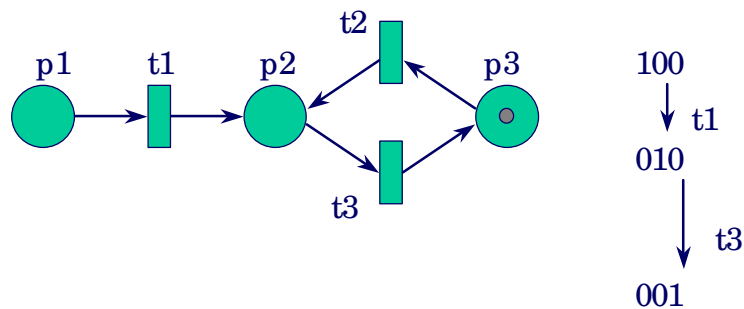
## Reachability graph



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

67

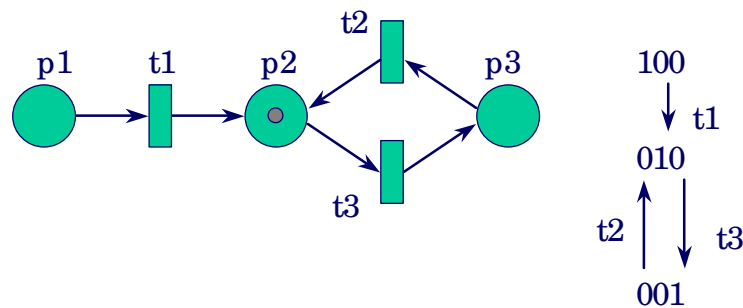
## Reachability graph



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

68

## Reachability graph

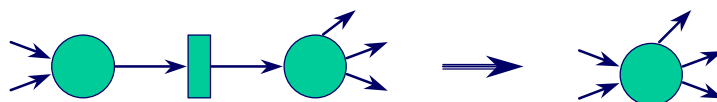


- For bounded nets the Coverability Tree is called **Reachability Tree** since it contains all possible reachable markings

69

## Subclasses of Petri nets

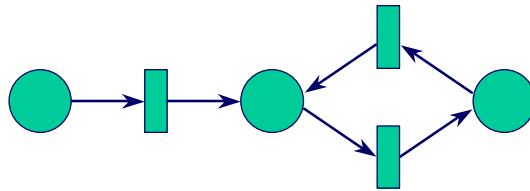
- Reachability analysis is too expensive
- State equations give only partial information
- Some properties are preserved by **reduction rules**  
e.g. for liveness and safeness



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

## Subclasses of Petri nets: SMs

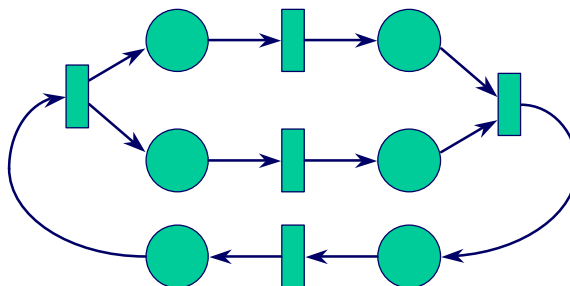
- **State machine:** every transition has at most 1 predecessor and 1 successor
- **Models only causality and conflict**
  - (no concurrency, no synchronization of parallel activities)



71

## Subclasses of Petri nets: MGs

- **Marked Graph:** every place has at most 1 predecessor and 1 successor
- **Models only causality and concurrency** (no conflict)

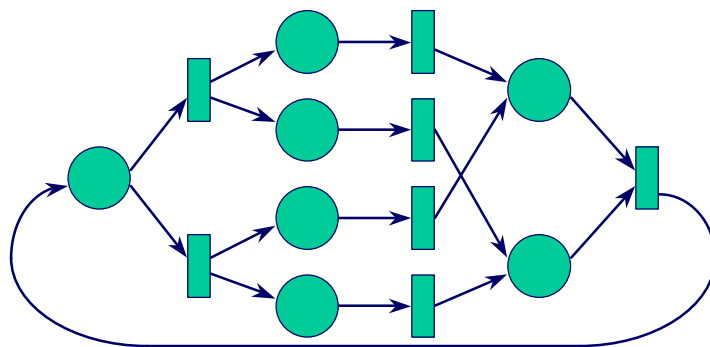


- **Same as underlying graph of SDF**

72

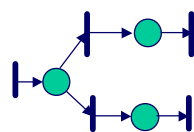
## Subclasses of Petri nets: FC nets

- **Free-Choice net: every transition after choice has **exactly 1** predecessor**

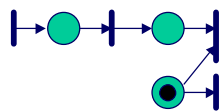


73

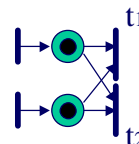
## Free-Choice Petri Nets (FCPN)



Free-Choice (FC)



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.**

Easy to analyze

74