Distributed Model:

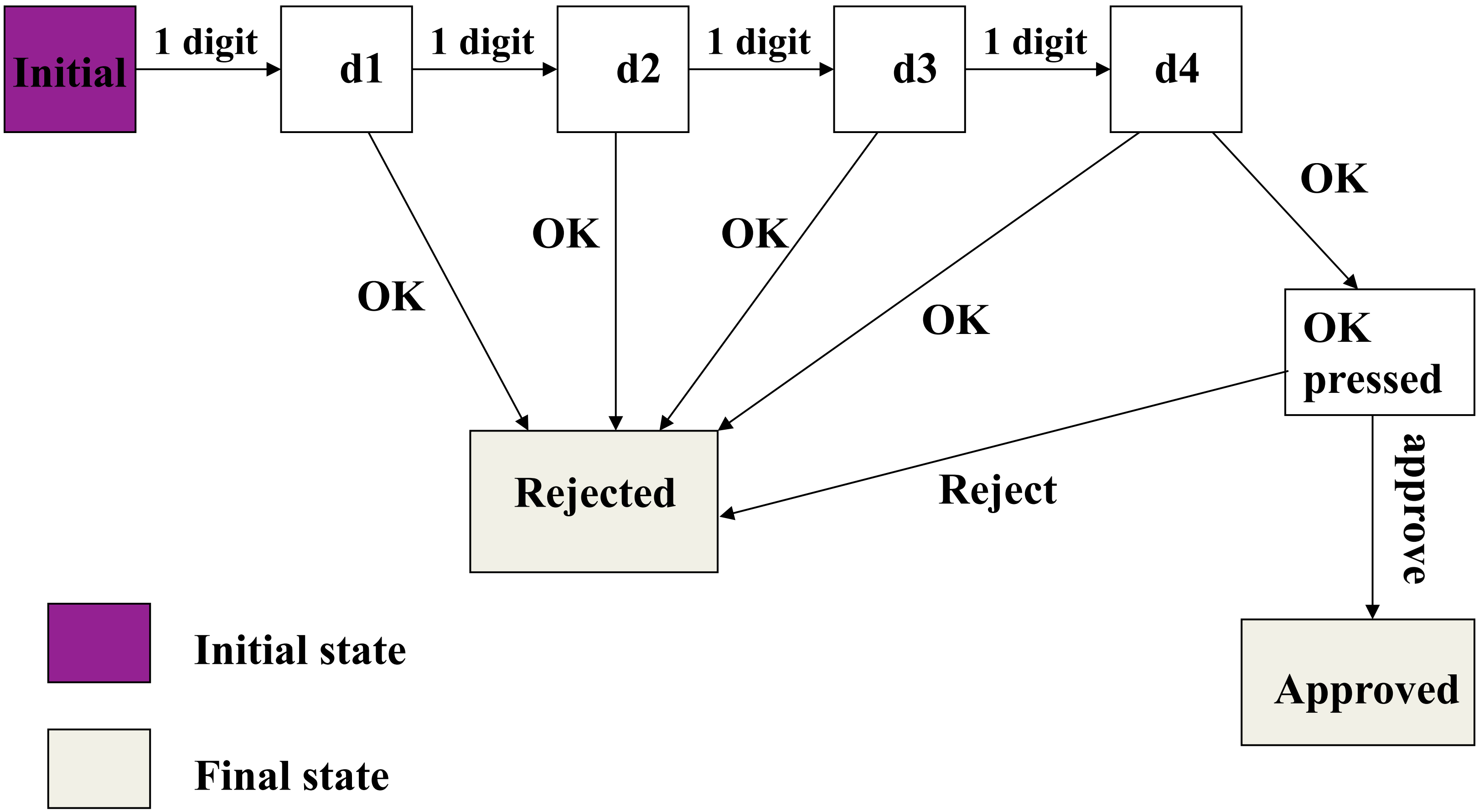
Petri Nets

Introduction

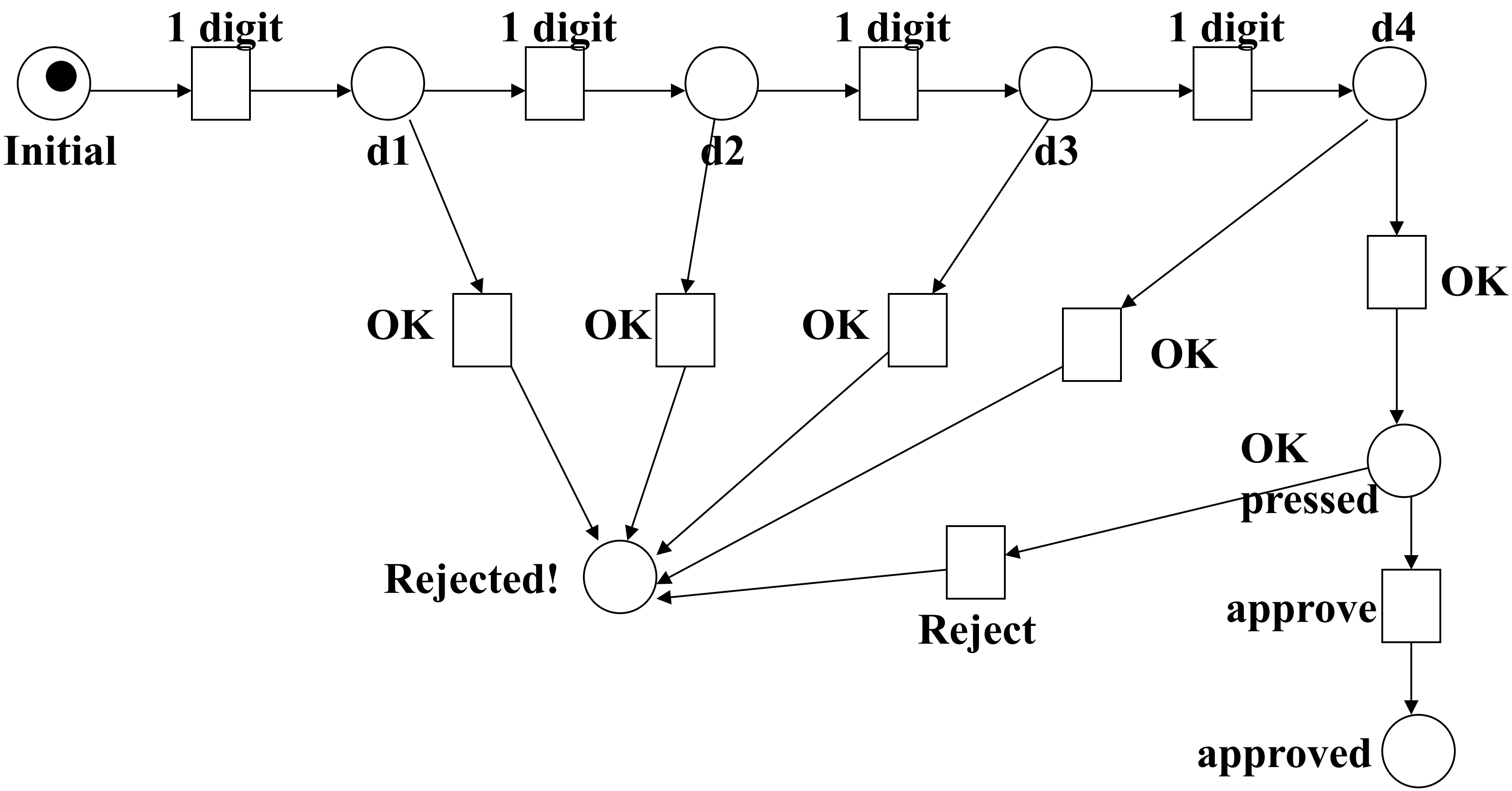
- Introduced by Carl Adam Petri in 1962.
- A diagrammatic tool to model concurrency and synchronization in distributed systems.

Example: EFTPOS FSA

(Electronic Fund Transfer Point of Sale)



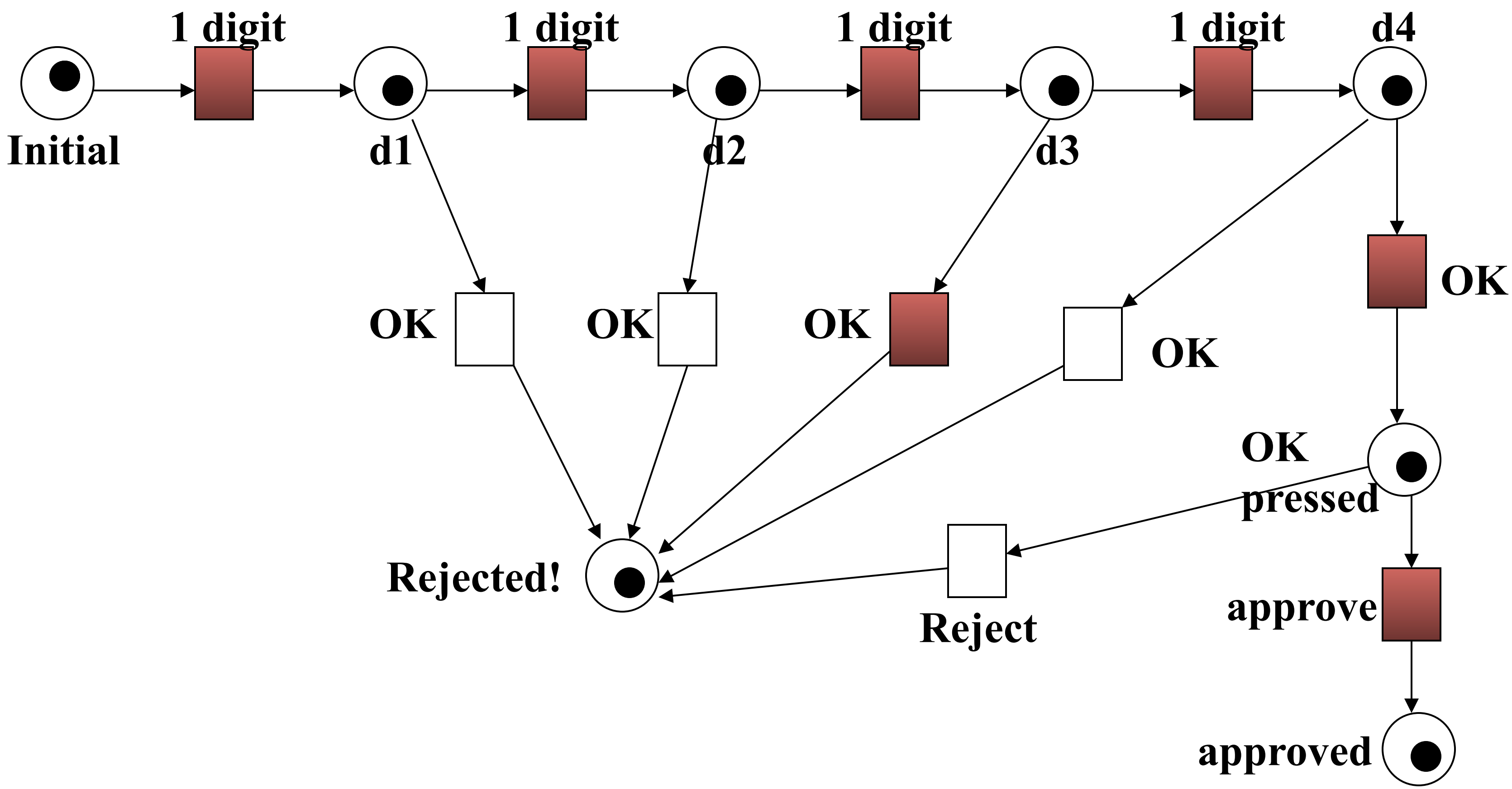
Example: EFTPOS Petri net



EFTPOS System

- Scenario 1: Normal
  - Enters all 4 digits and press OK.
- Scenario 2: Exceptional
  - Enters only 3 digits and press OK.

Example: EFTPOS System (Token Games)



A Petri Net Specification ...

- consists of: *places* (circles), *transitions* (rectangles) and *arcs* (arrows):
  - *Places* represent possible states of the system.
  - *Transitions* are events or actions which cause the change of state.
  - Every *arc* simply connects a place with a transition or a transition with a place.

A Change of State …

- is denoted by a movement of *token(s)* (black dots) from place(s) to place(s); and is caused by the *firing* of a transition.
- The firing represents an occurrence of the event or an action taken.
- The firing is subject to the input conditions, denoted by token availability.

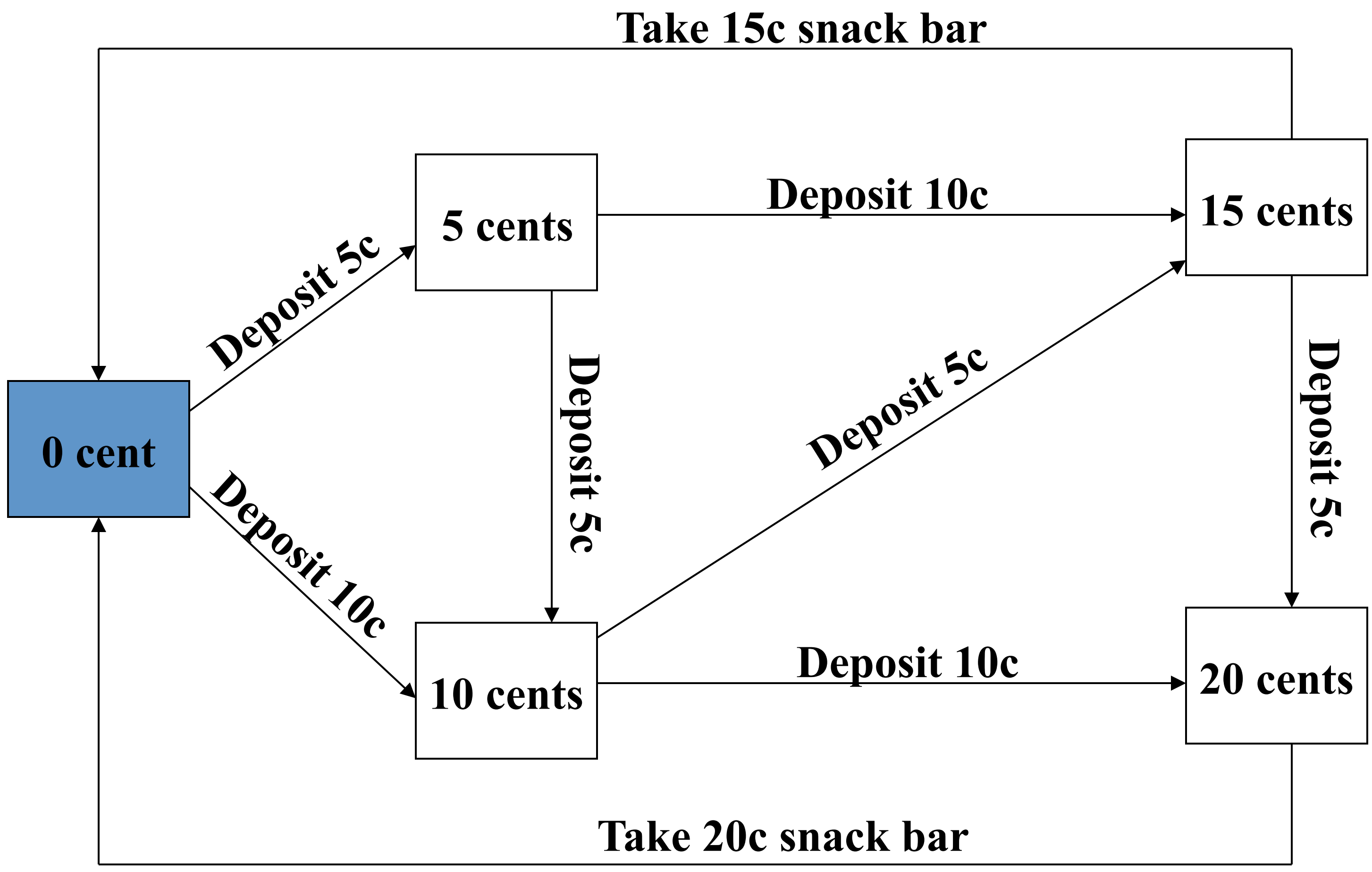
A Change of State

- A transition is *firable* or *enabled* when there are sufficient tokens in its input places.
- After firing, tokens will be transferred from the input places (old state) to the output places, denoting the new state.
- Note that the EFTPOS example is a Petri net representation of a finite state machine (FSM).

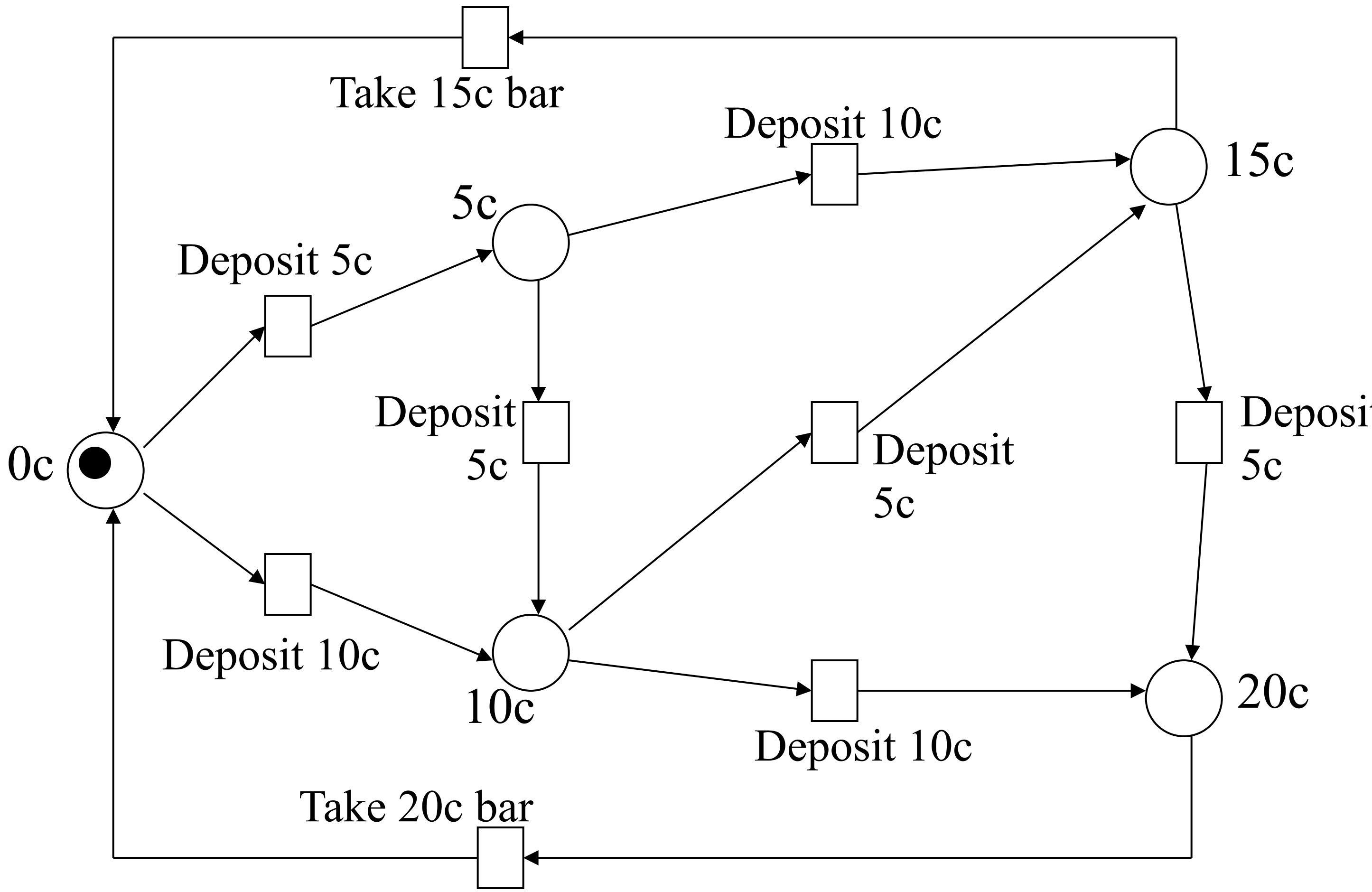
Example: Vending Machine

- The machine dispenses two kinds of snack bars – 20c and 15c.
- Only two types of coins can be used – 10c coins and 5c coins.
- The machine does not return any change.

Example: Vending Machine (STD of an FSM)



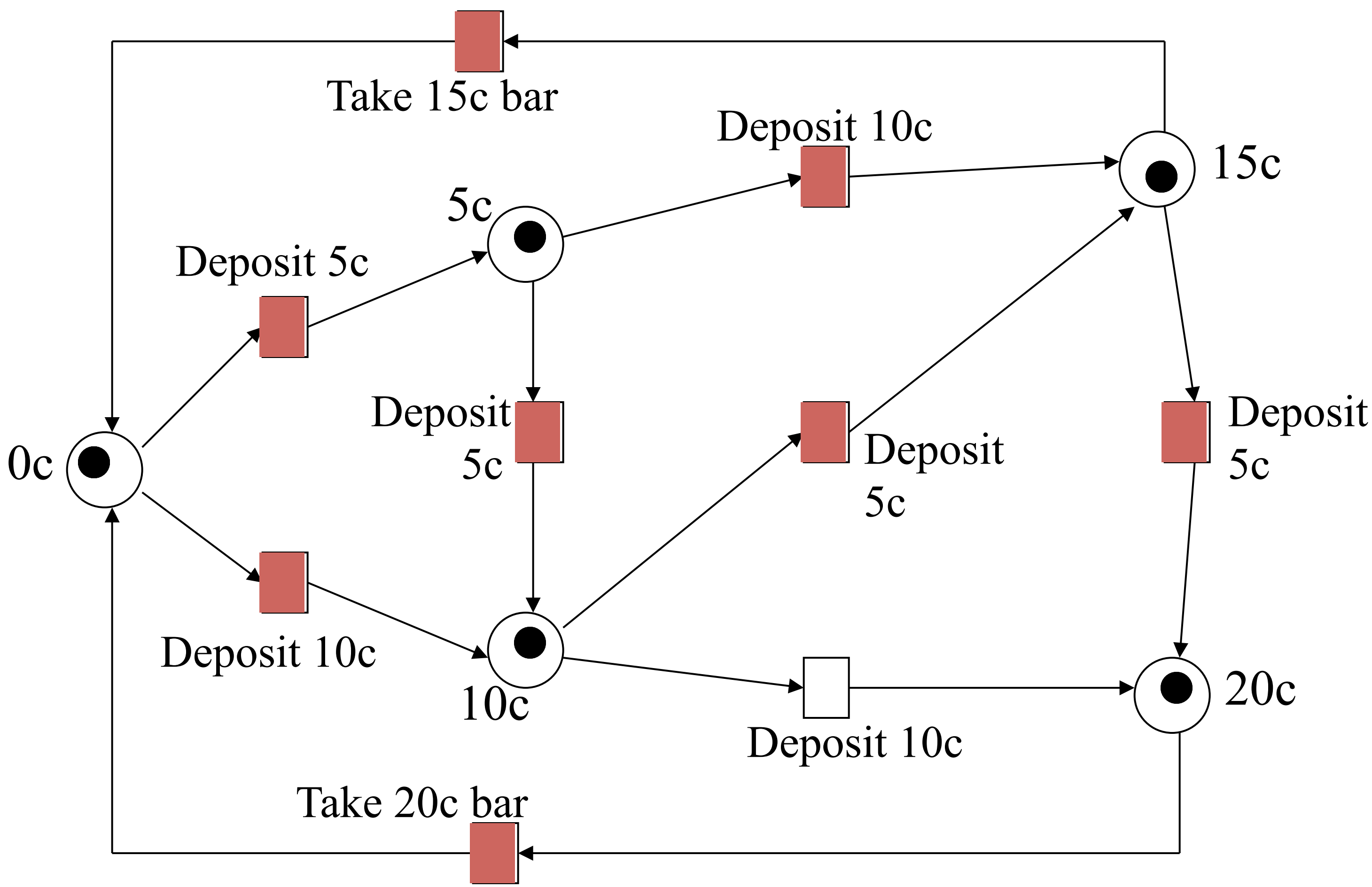
Example: Vending Machine (A Petri net)



Example: Vending Machine (3 Scenarios)

- Scenario 1:
  - Deposit 5c, deposit 5c, deposit 5c, deposit 5c, take 20c snack bar.
- Scenario 2:
  - Deposit 10c, deposit 5c, take 15c snack bar.
- Scenario 3:
  - Deposit 5c, deposit 10c, deposit 5c, take 20c snack bar.

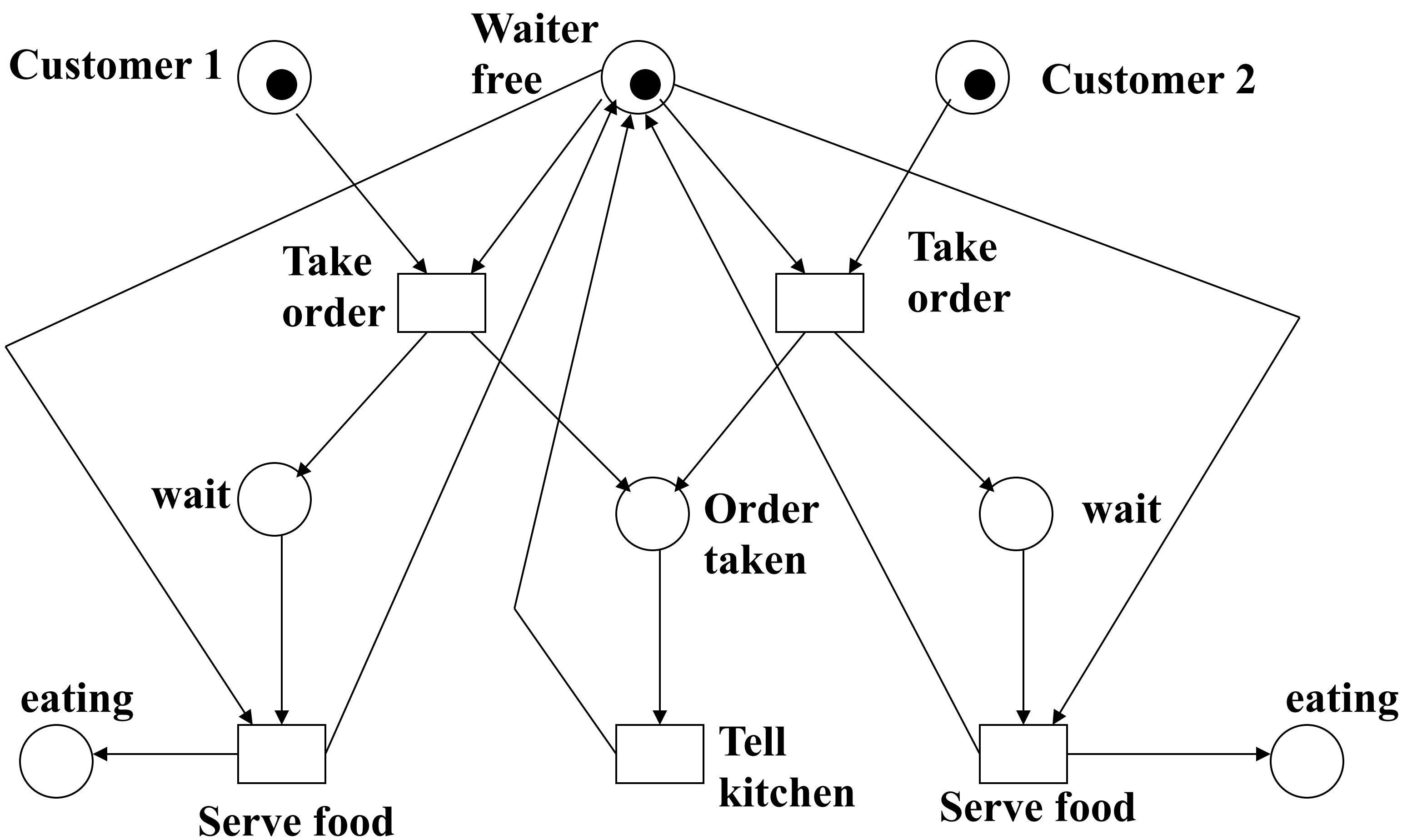
Example: Vending Machine (Token Games)



Multiple Local States

- In the real world, events happen at the same time.
- A system may have many local states to form a global state.
- There is a need to model concurrency and synchronization.

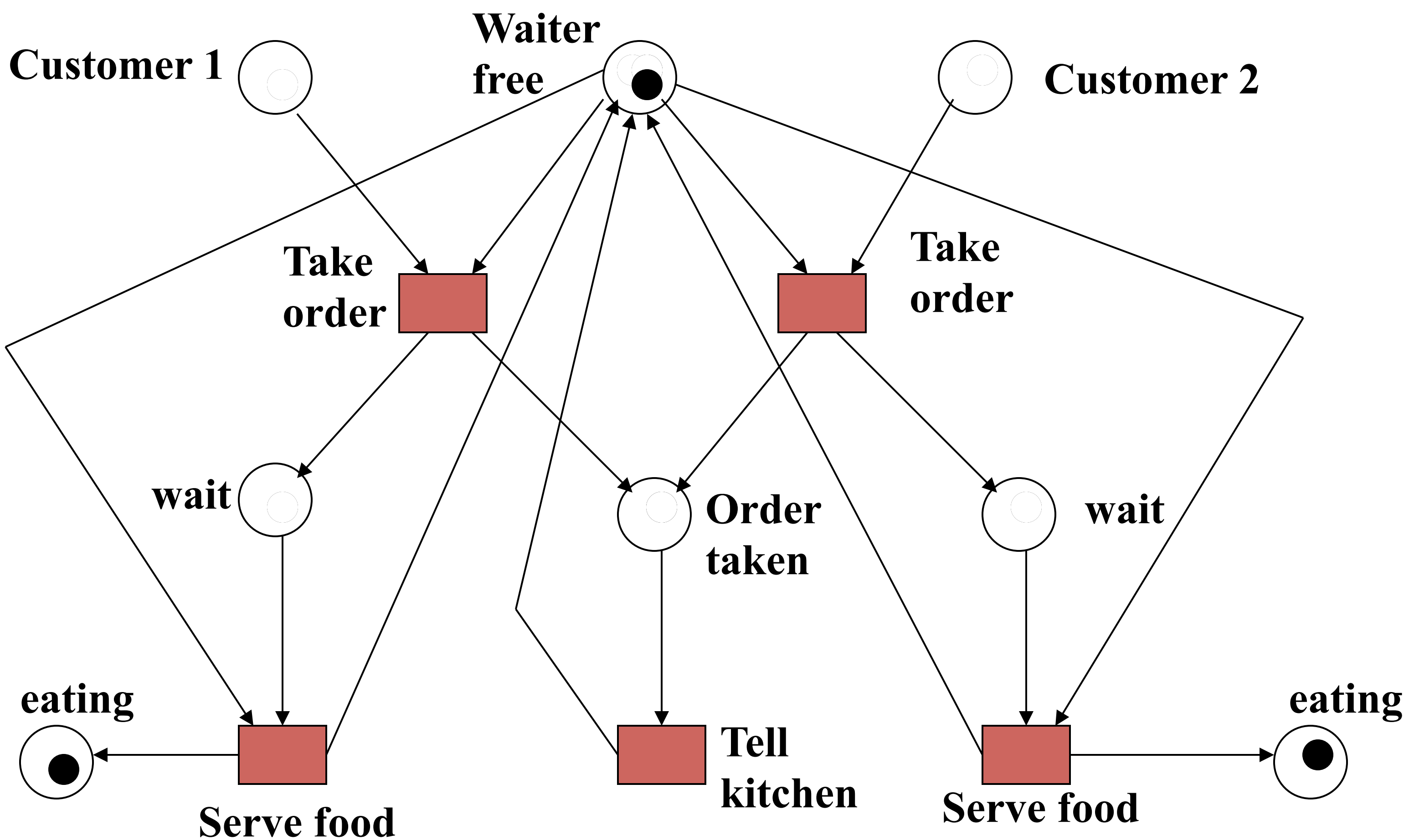
Example: In a Restaurant (A Petri Net)



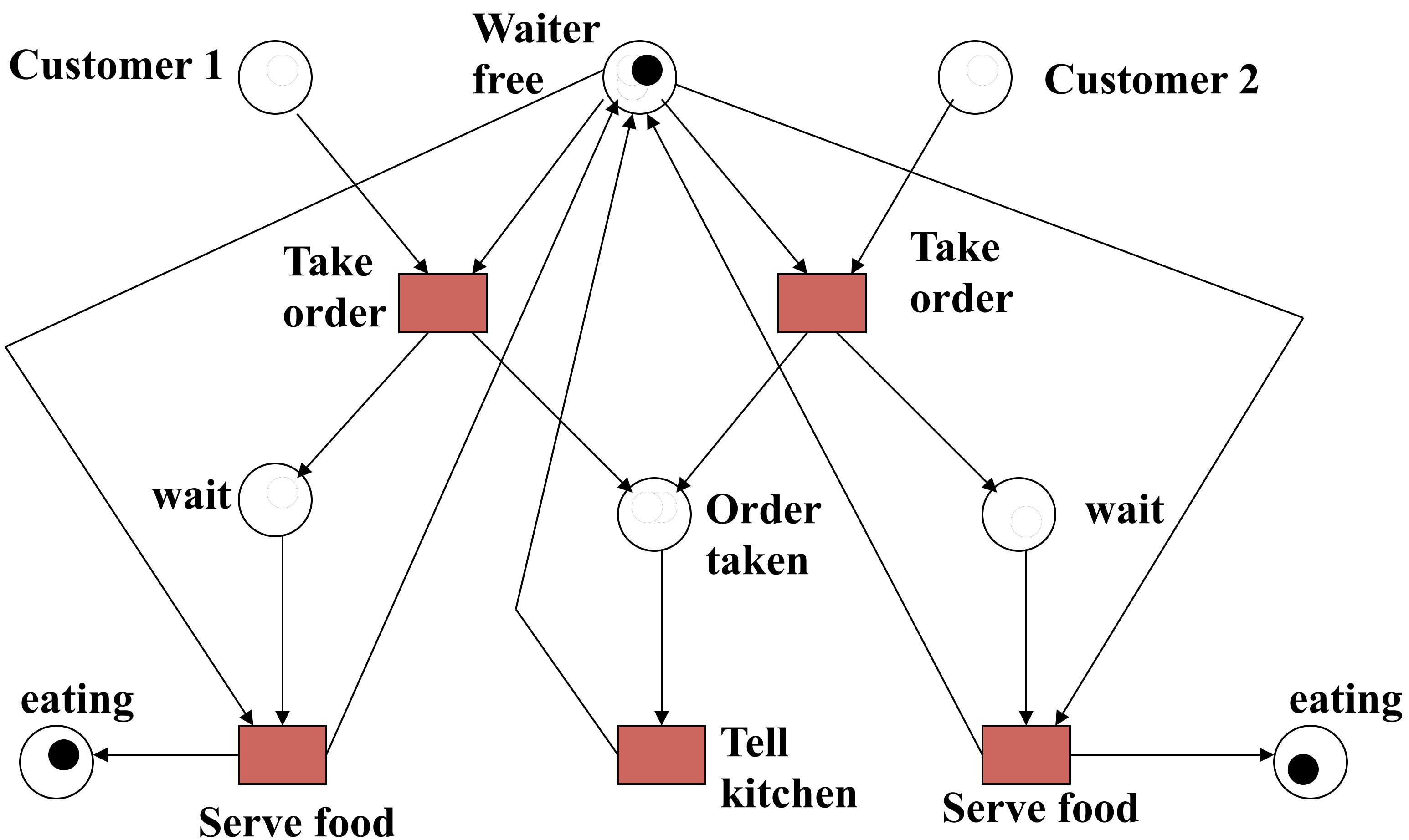
Example: In a Restaurant (Two Scenarios)

- Scenario 1:
  - Waiter takes order from customer 1; serves customer 1; takes order from customer 2; serves customer 2.
- Scenario 2:
  - Waiter takes order from customer 1; takes order from customer 2; serves customer 2; serves customer 1.

Example: In a Restaurant (Scenario 1)



Example: In a Restaurant (Scenario 2)

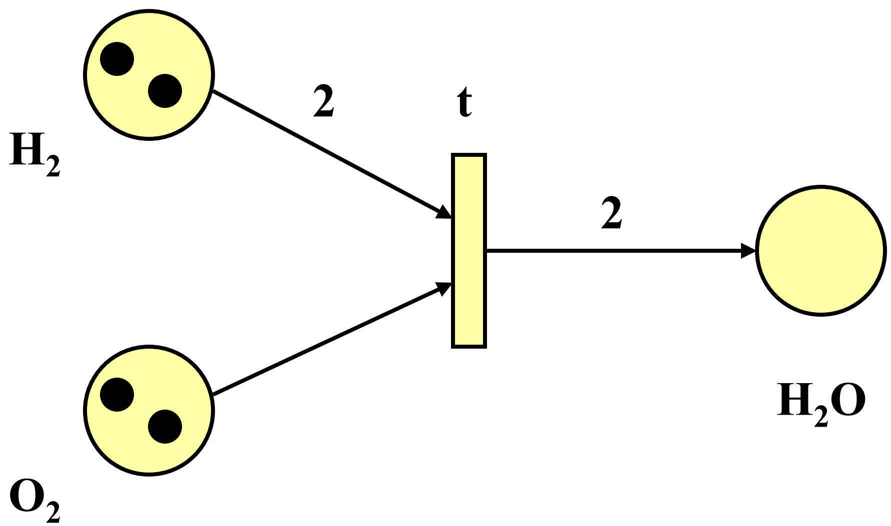


Transition (firing) rule

- A transition t is enabled if each input place p has at least w(p,t) tokens
- An enabled transition may or may not fire
- A firing on an enabled transition t removes w(p,t) from each input place p, and adds w(t,p') to each output place p'

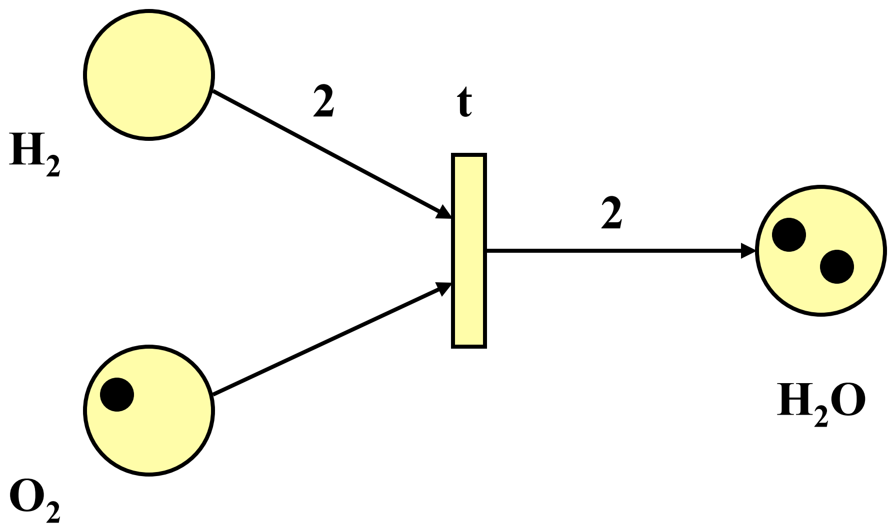
Firing example

$2H_2 + O_2 \rightarrow 2H_2O$



Firing example

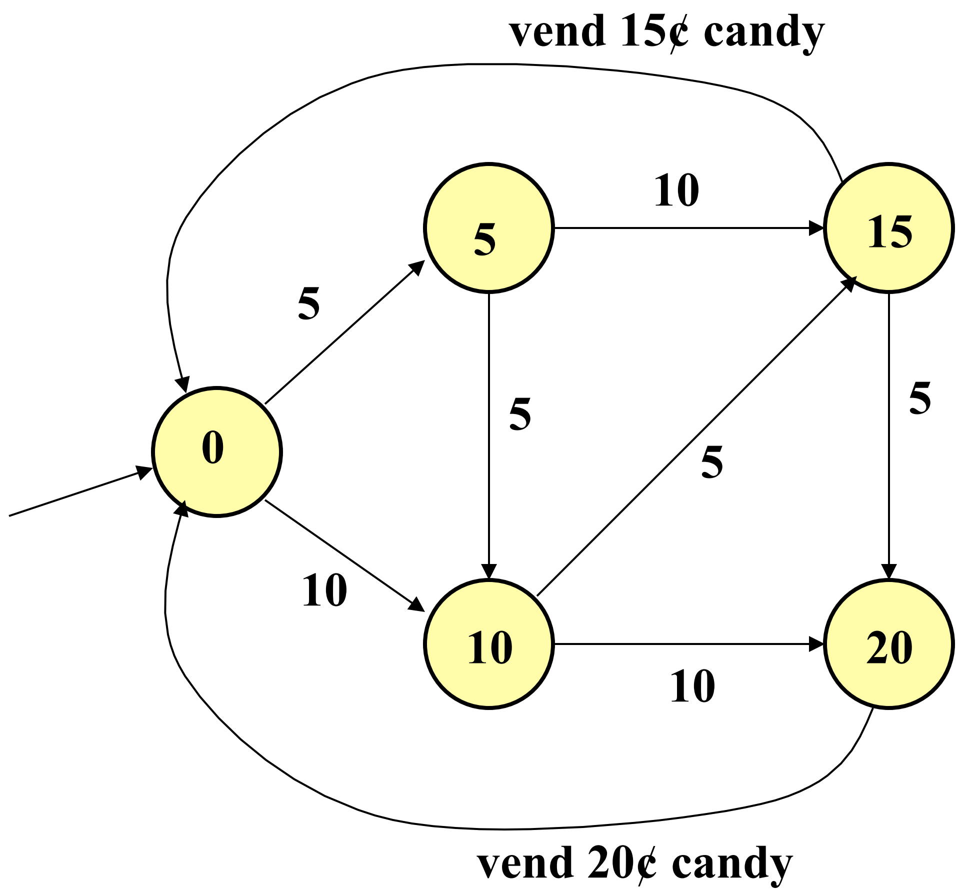
$2H_2 + O_2 \rightarrow 2H_2O$



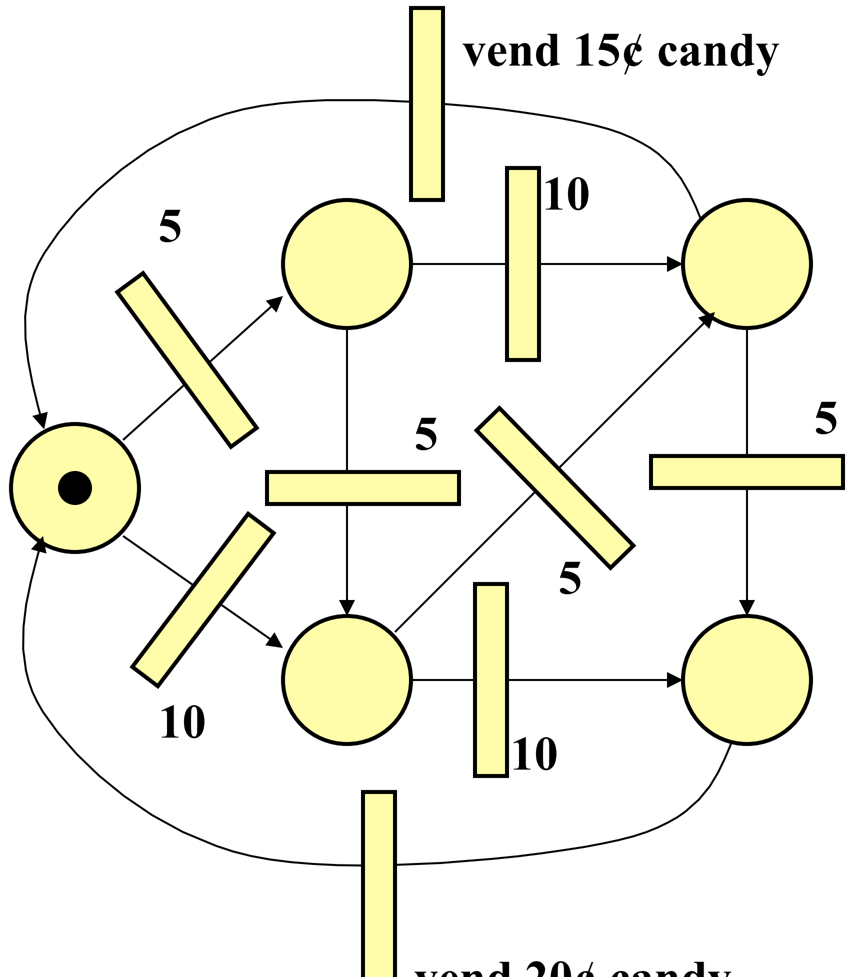
Some definitions

- **source transition**: no inputs
- **sink transition**: no outputs
- **self-loop**: a pair (p,t) s.t. p is both an input and an output of t
- **pure PN**: no self-loops
- **ordinary PN**: all arc weights are 1's
- **infinite capacity net**: places can accommodate an unlimited number of tokens
- **finite capacity net**: each place p has a maximum capacity K(p)
- **strict transition rule**: after firing, each output place can't have more than K(p) tokens
- **Theorem**: every pure finite-capacity net can be transformed into an equivalent infinite-capacity net

Modeling FSMs

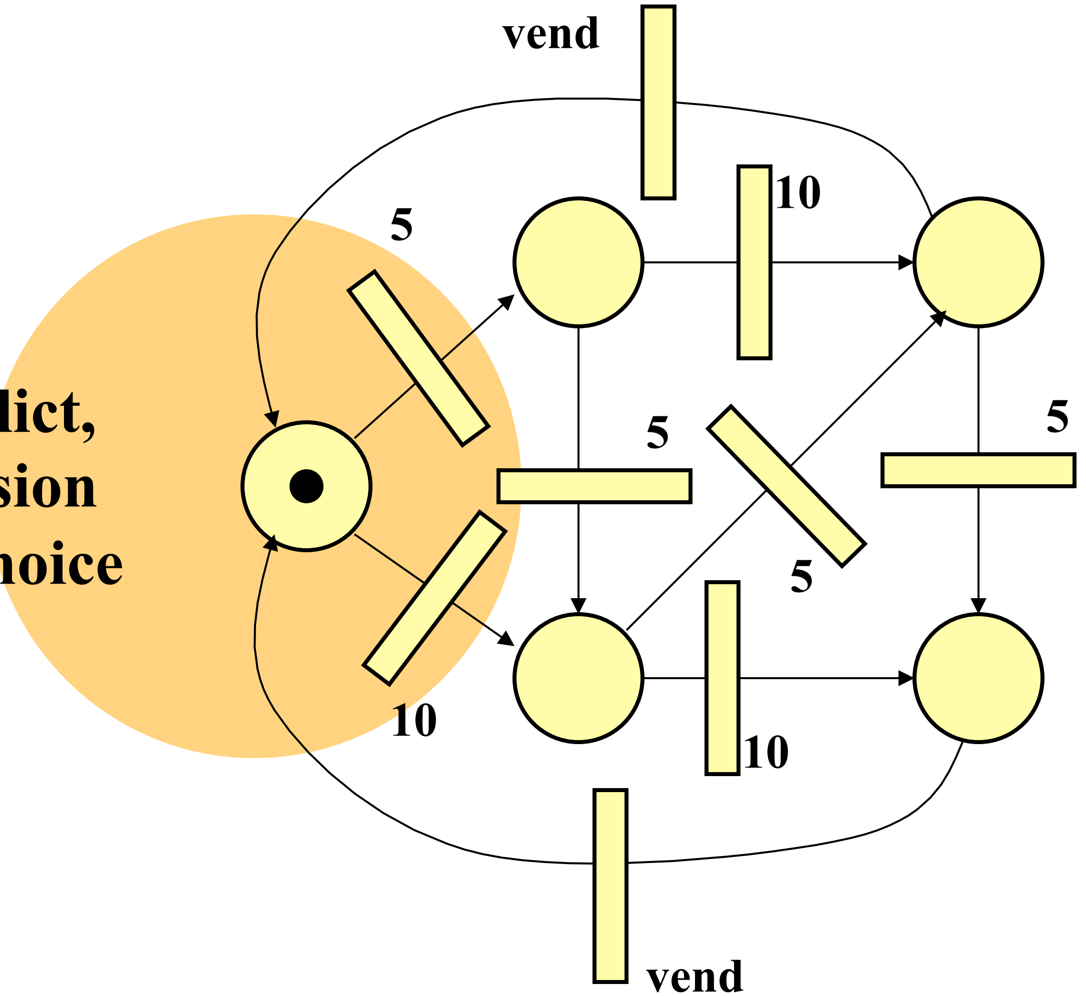


Modeling FSMs



state machines: each transition has exactly one input and one output

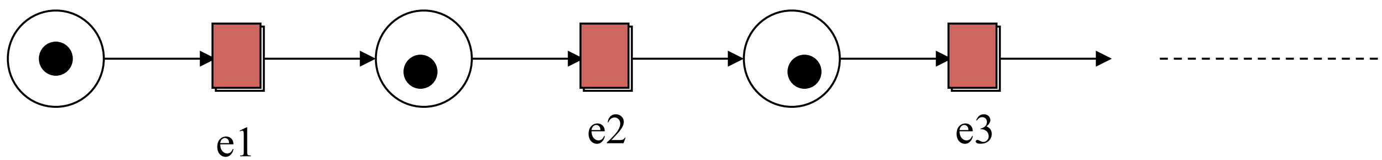
Modeling FSMs



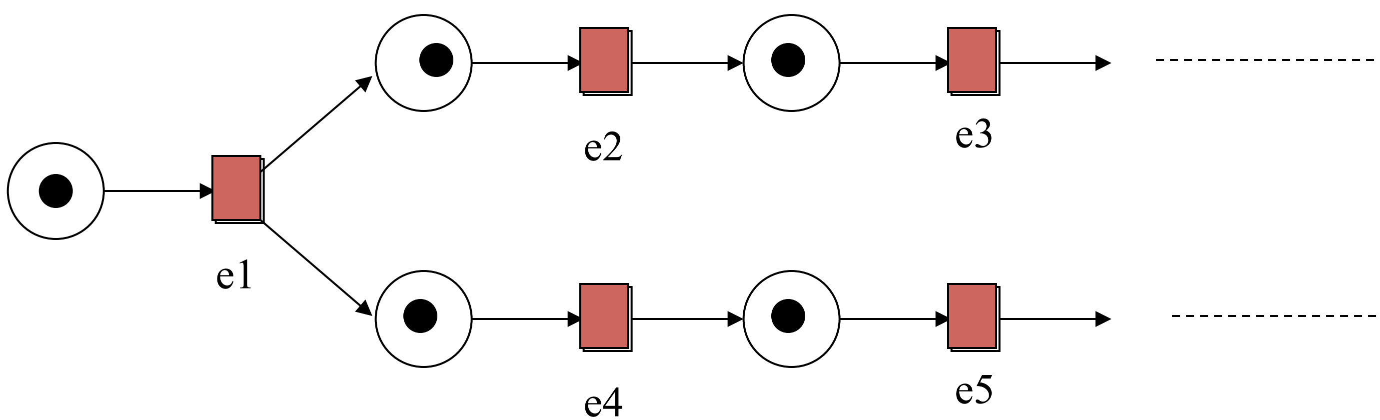
conflict,
decision
or choice

Net Structures

- A sequence of events/actions:

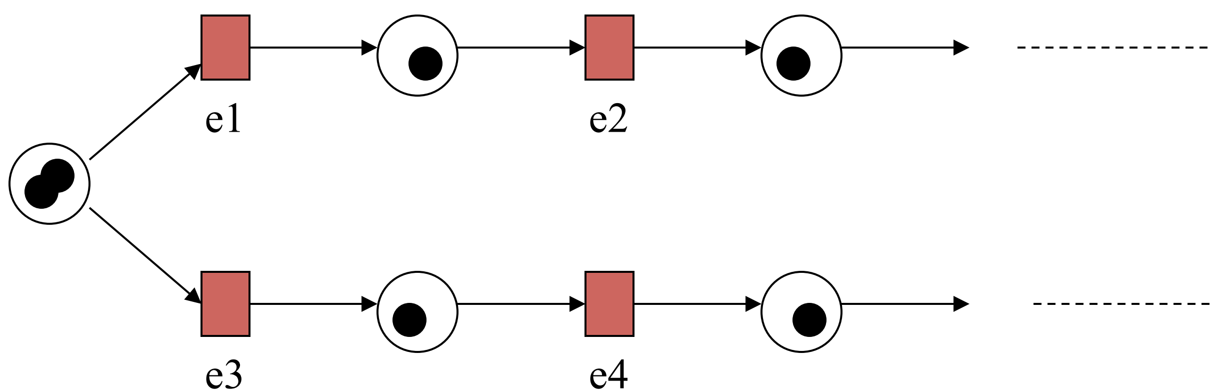


- Concurrent executions:



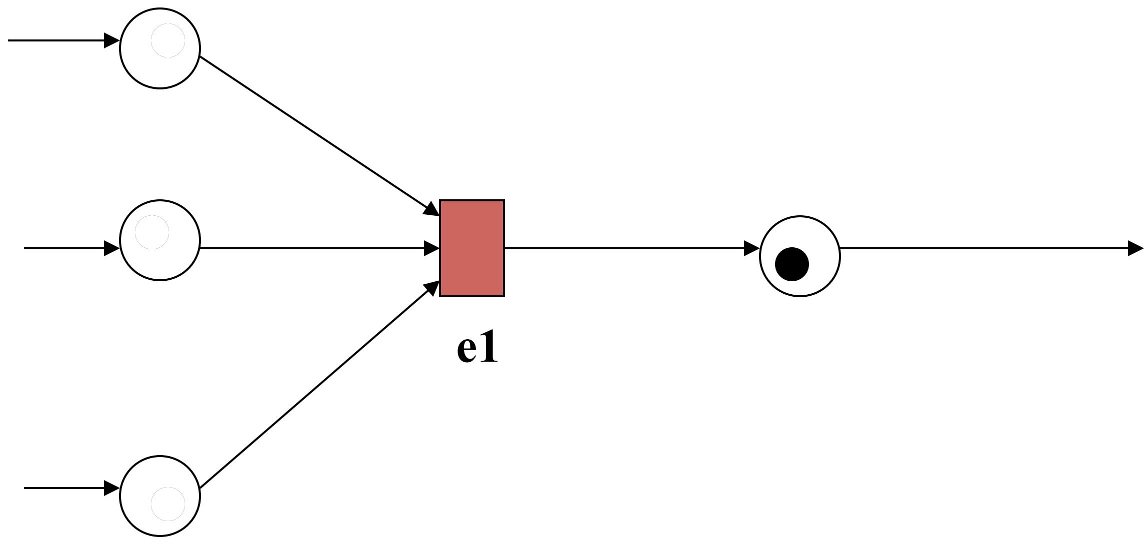
Net Structures

- Non-deterministic events - conflict, choice or decision: A choice of either e1, e2 … or e3, e4 ...



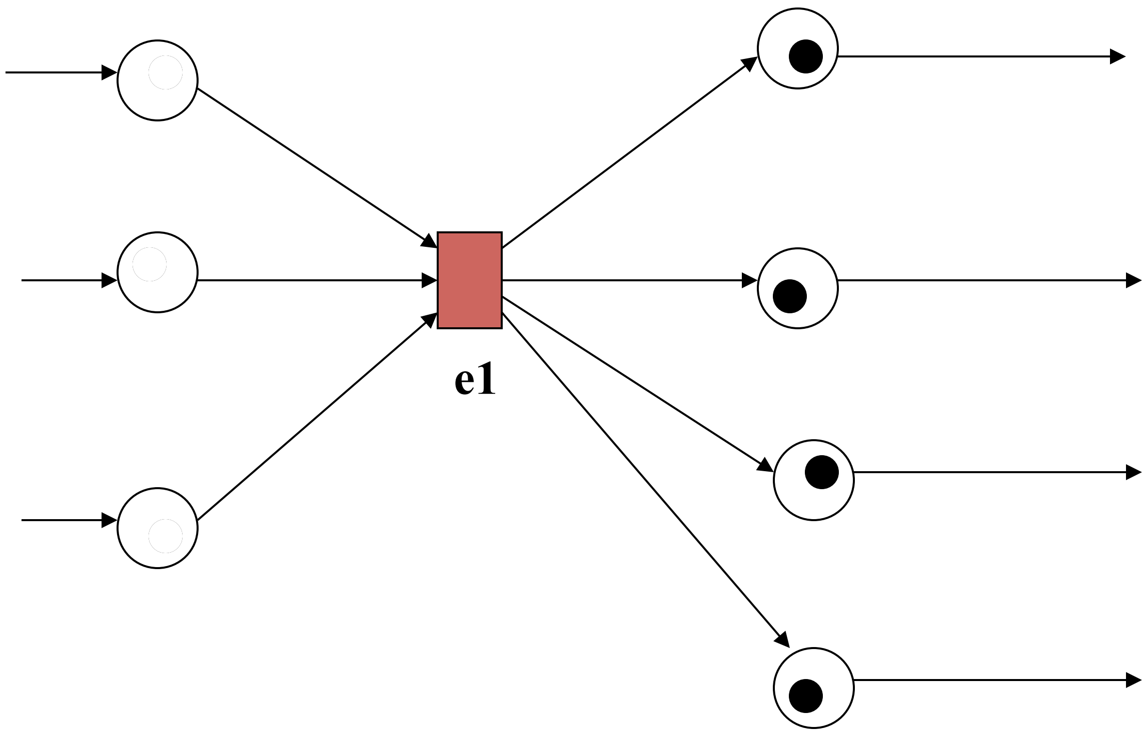
Net Structures

- Synchronization

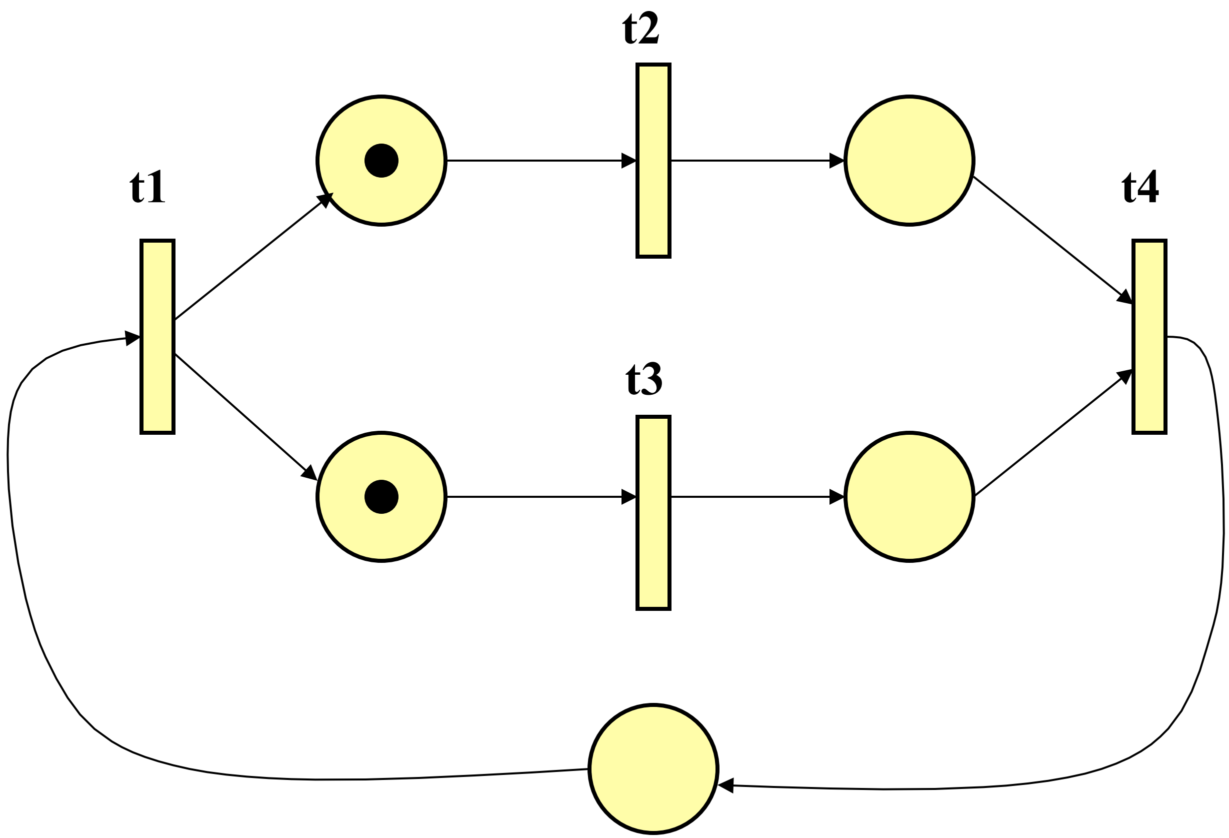


Net Structures

- Synchronization and Concurrency

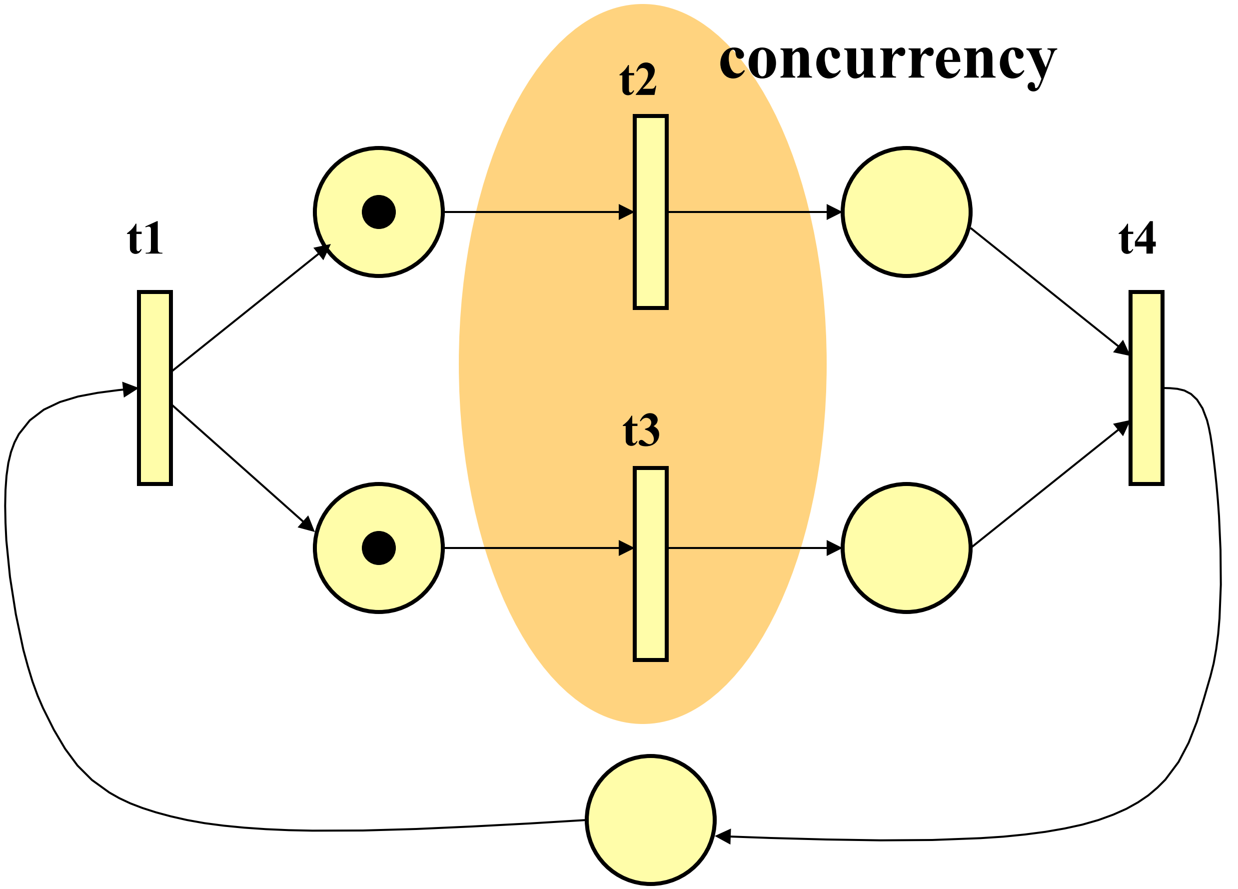


Modeling concurrency



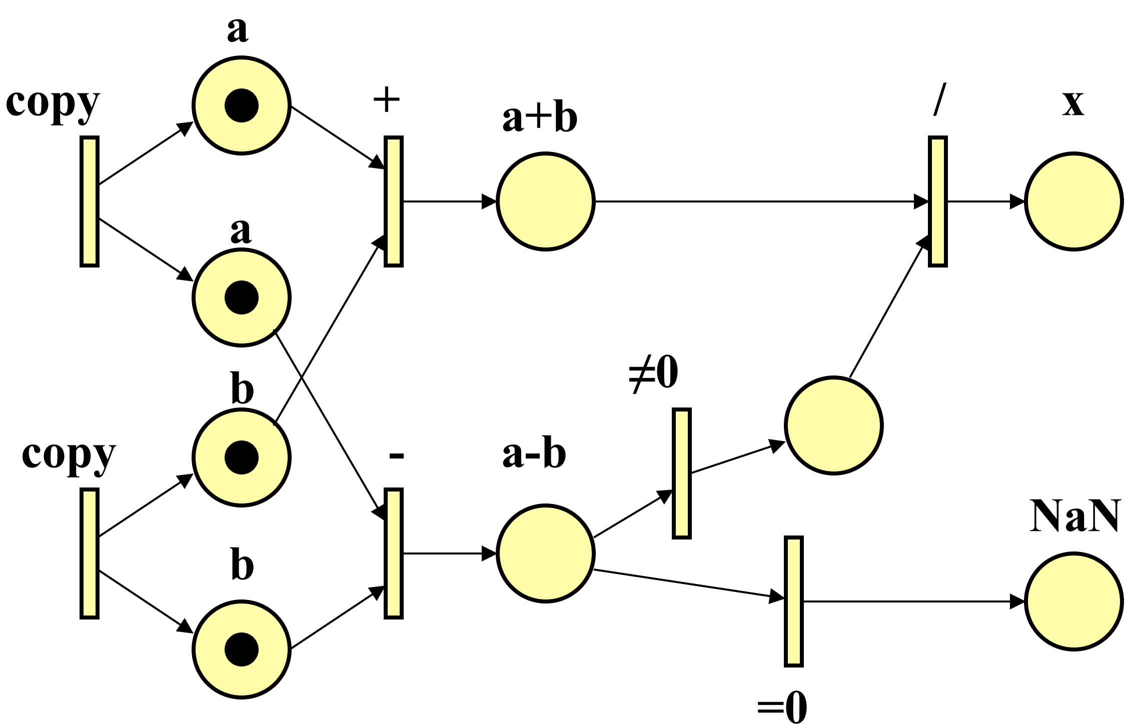
marked graph: each place has exactly one incoming arc and one outgoing arc.

Modeling concurrency

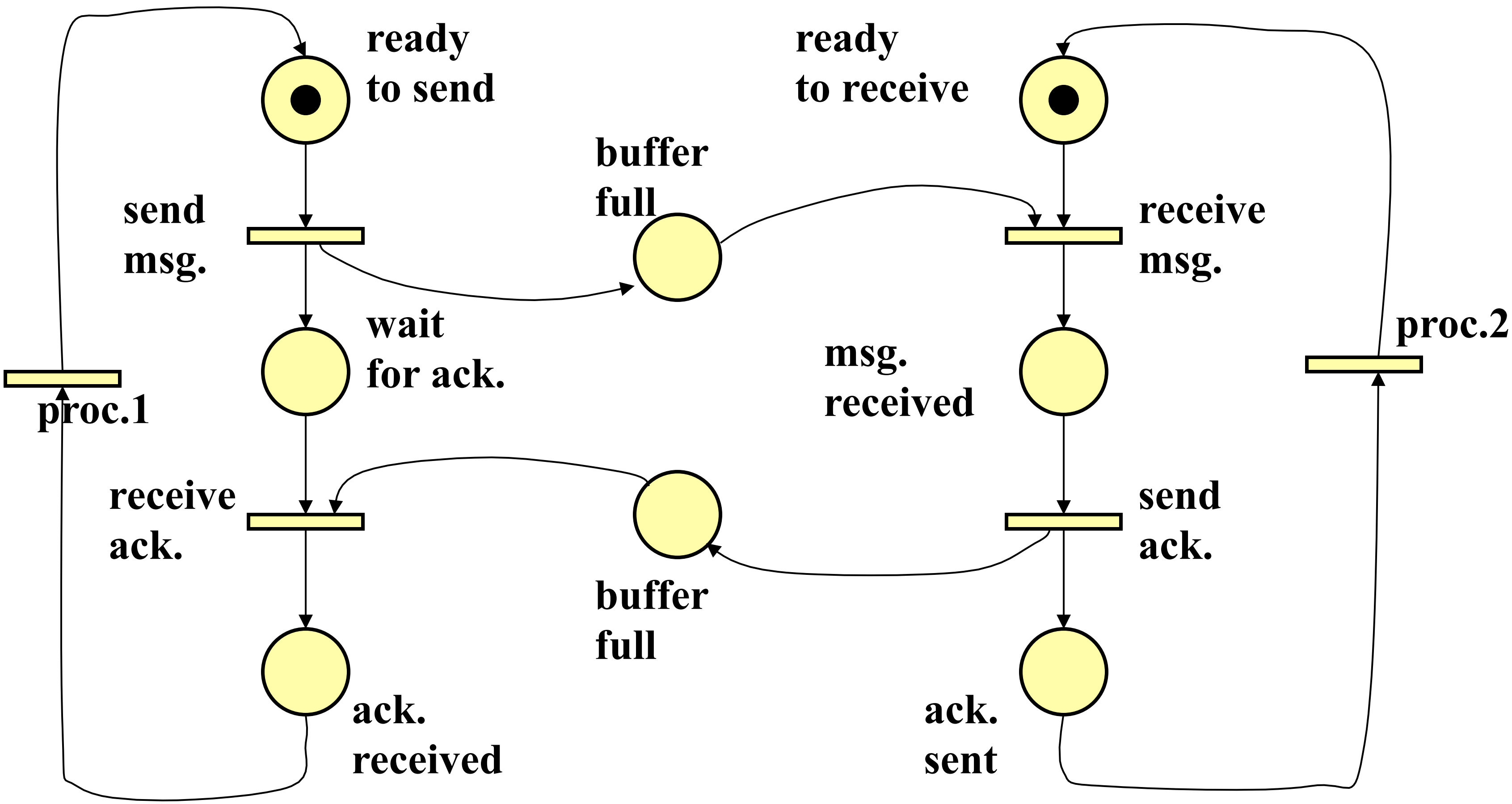


Modeling dataflow computation

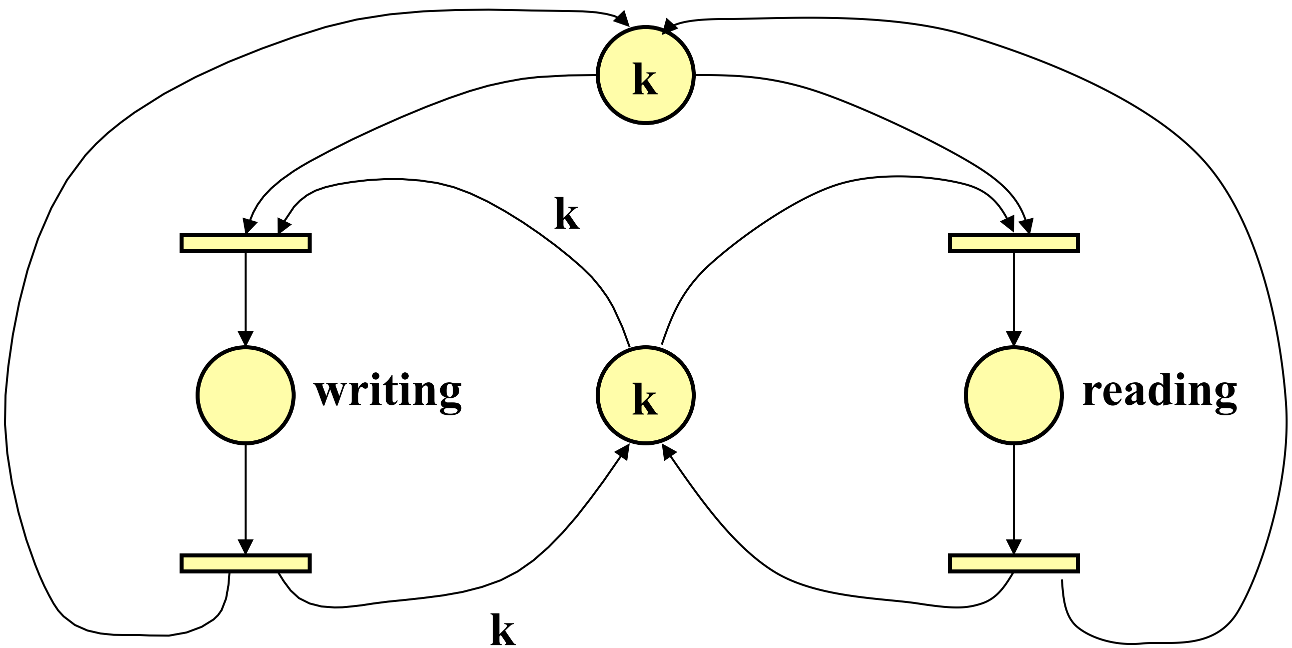
$x = (a+b)/(a-b)$



Modeling communication protocols



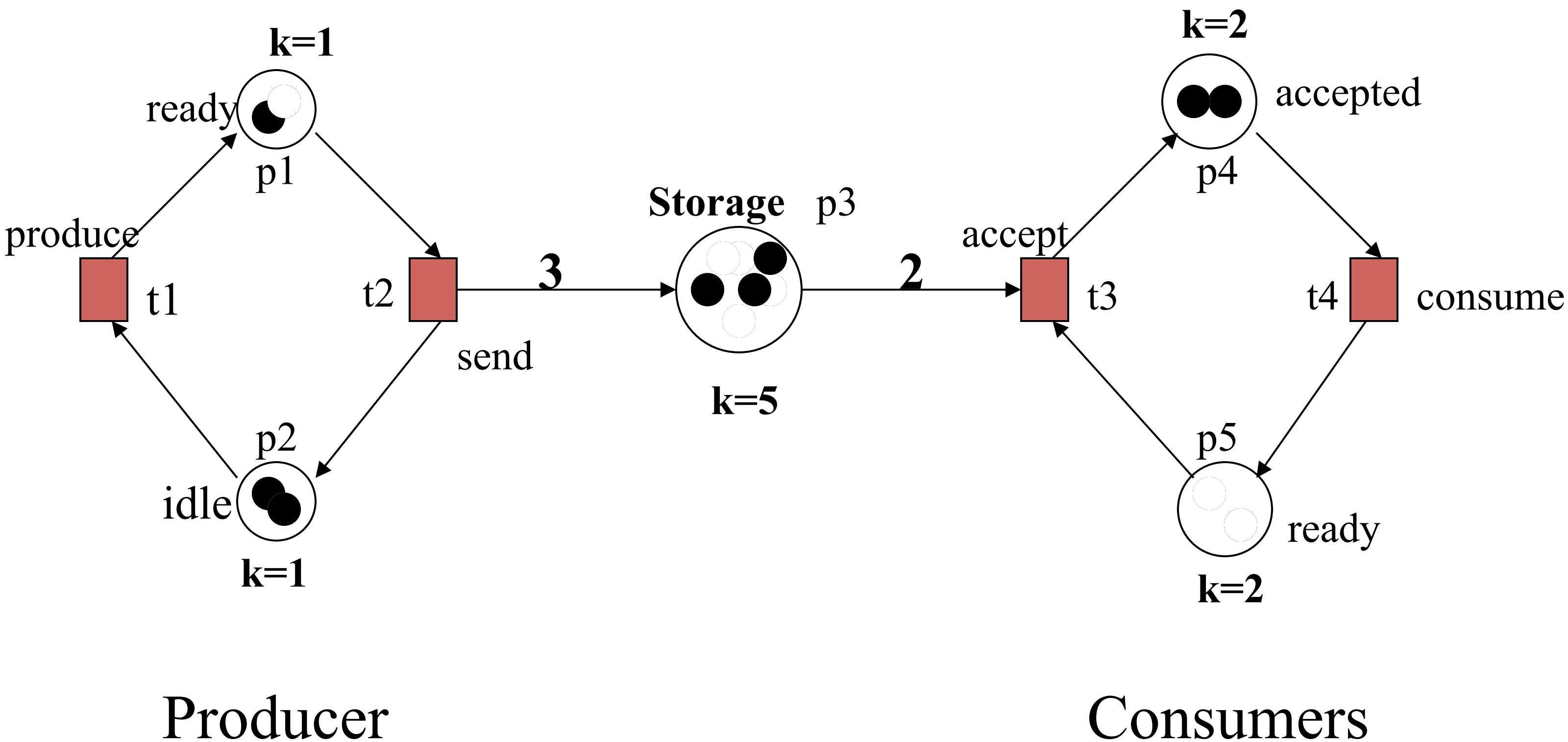
Modeling synchronization control



Another Example

- A producer-consumer system, consist of one producer, two consumers and one storage buffer with the following conditions:
  - The storage buffer may contain at most 5 items.
  - The producer sends 3 items in each production.
  - At most one consumer is able to access the storage buffer at one time.
  - Each consumer removes two items when accessing the storage buffer.

A Producer-Consumer System



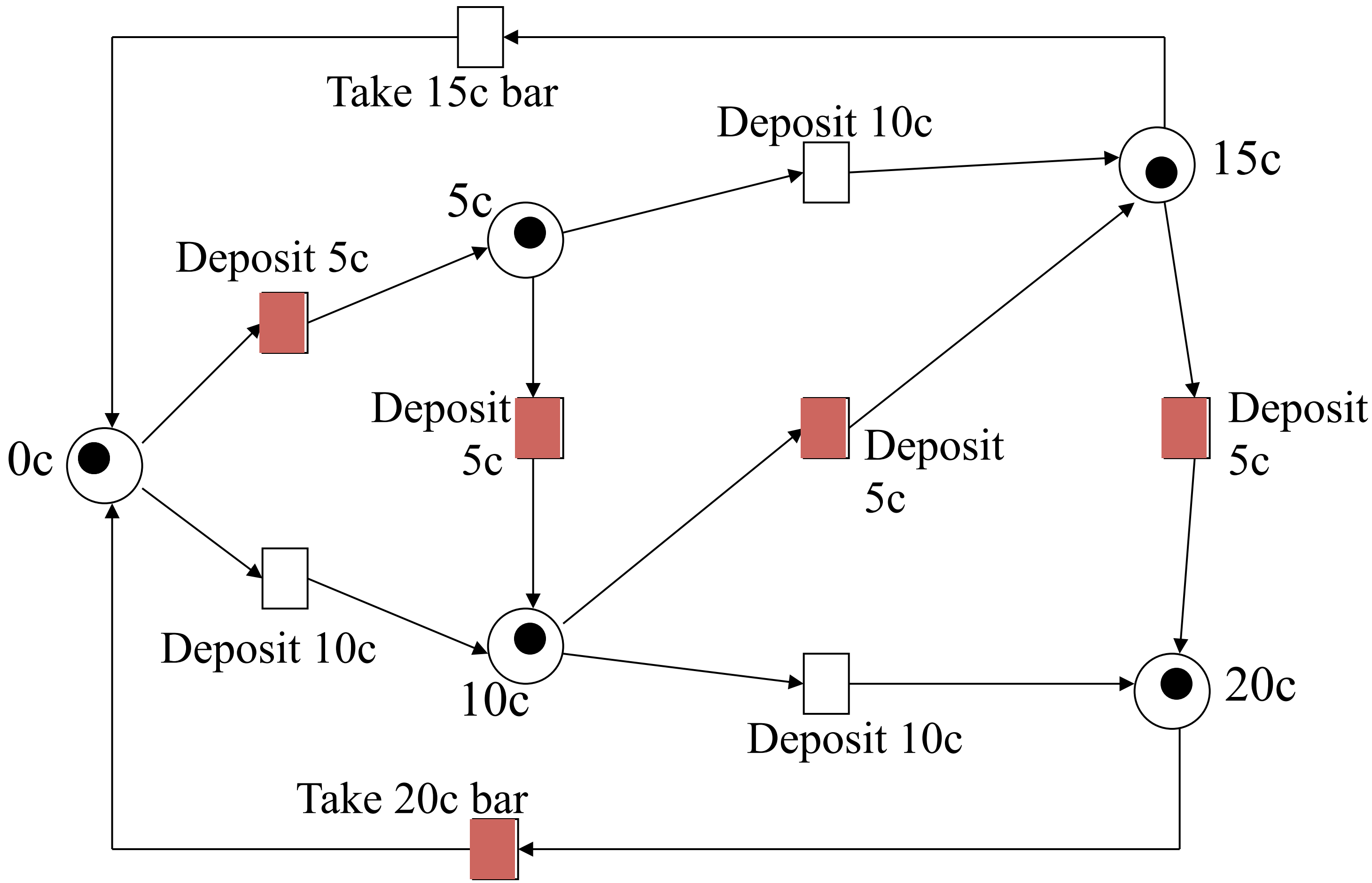
A Producer-Consumer Example

- In this Petri net, every place has a *capacity* and every arc has a *weight*.
- This allows multiple tokens to reside in a place to model more complex behaviour.

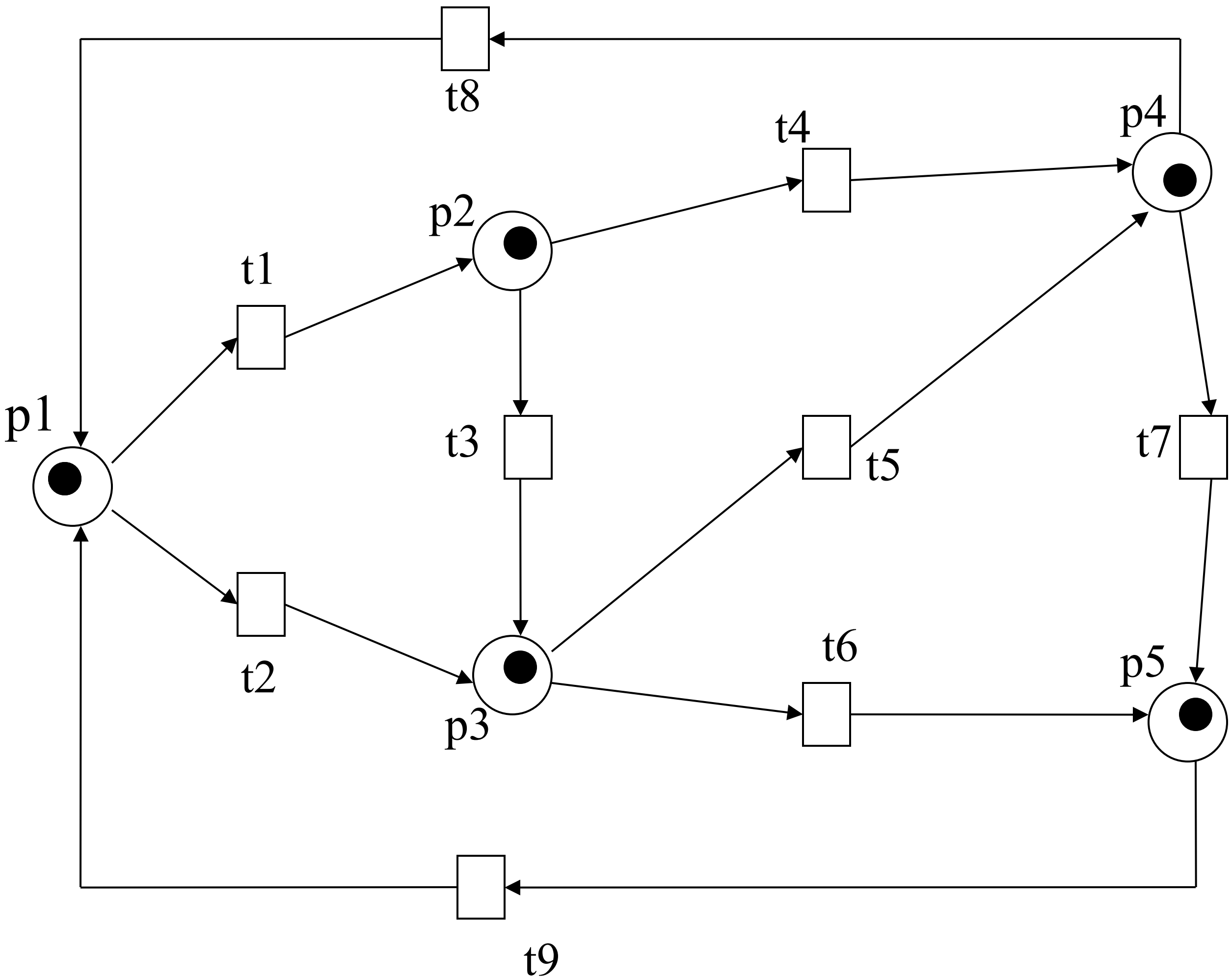
Behavioural Properties

- Reachability
  - "Can we reach one particular state from another?"
- Boundedness
  - "Will a storage place overflow?"
- Liveness
  - "Will the system die in a particular state?"

Recalling the Vending Machine (Token Game)



A *marking* is a state ...



M0 = (1,0,0,0,0)

M1 = (0,1,0,0,0)

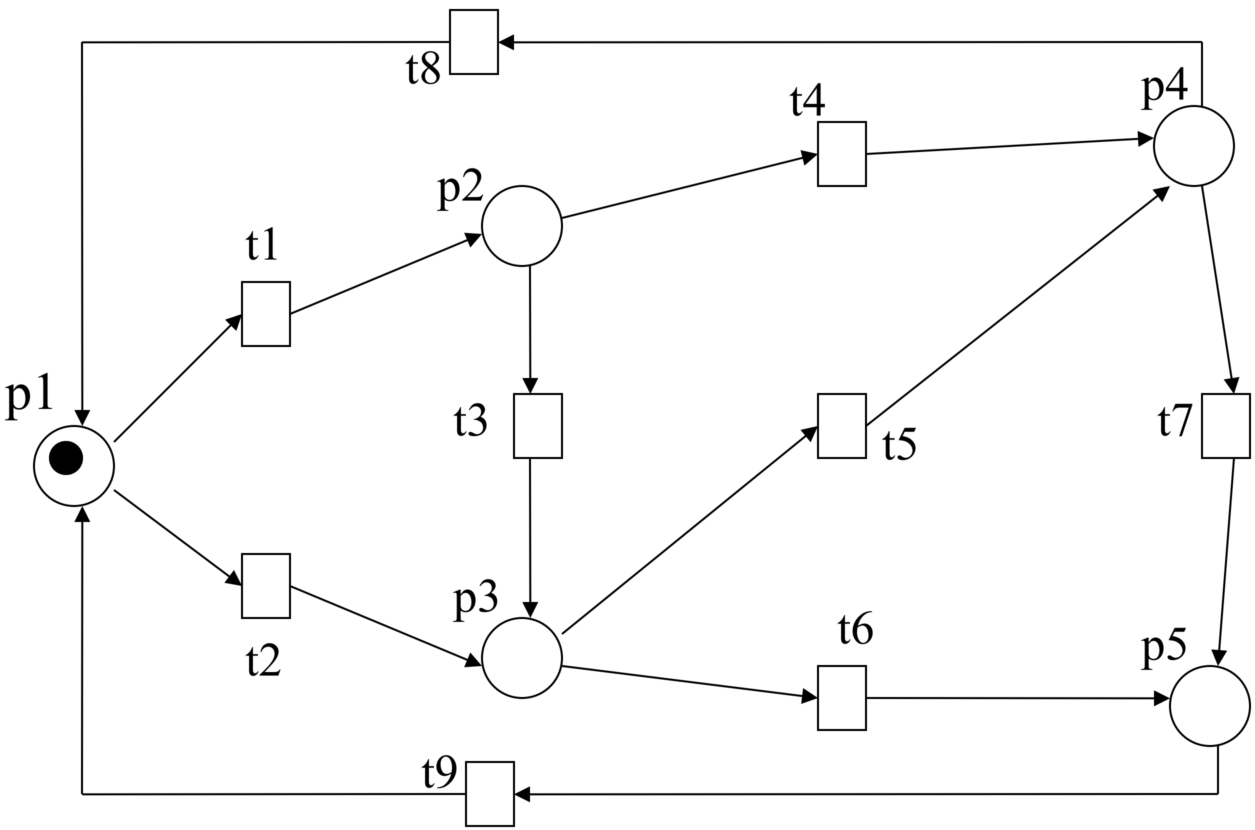
M2 = (0,0,1,0,0)

M3 = (0,0,0,1,0)

M4 = (0,0,0,0,1)

Initial marking:M0

Reachability



$M0 = (1,0,0,0,0)$

$M1 = (0,1,0,0,0)$

$M2 = (0,0,1,0,0)$

$M3 = (0,0,0,1,0)$

$M4 = (0,0,0,0,1)$

Initial marking:M0

$$M0 \xrightarrow{t1} M1 \xrightarrow{t3} M2 \xrightarrow{t5} M3 \xrightarrow{t8} M0 \xrightarrow{t2} M2 \xrightarrow{t6} M4$$

Reachability

A firing or occurrence sequence:

$$M0 \xrightarrow{t1} M1 \xrightarrow{t3} M2 \xrightarrow{t5} M3 \xrightarrow{t8} M0 \xrightarrow{t2} M2 \xrightarrow{t6} M4$$

- "M2 is *reachable* from M1 and M4 is *reachable* from M0."
- In fact, in the vending machine example, all markings are reachable from every marking.

Boundedness

- A Petri net is said to be *k-bounded* or simply *bounded* if the number of tokens in each place does not exceed a finite number *k* for any marking reachable from M0.
- The Petri net for vending machine is 1-bounded.
- A 1-bounded Petri net is also *safe*.

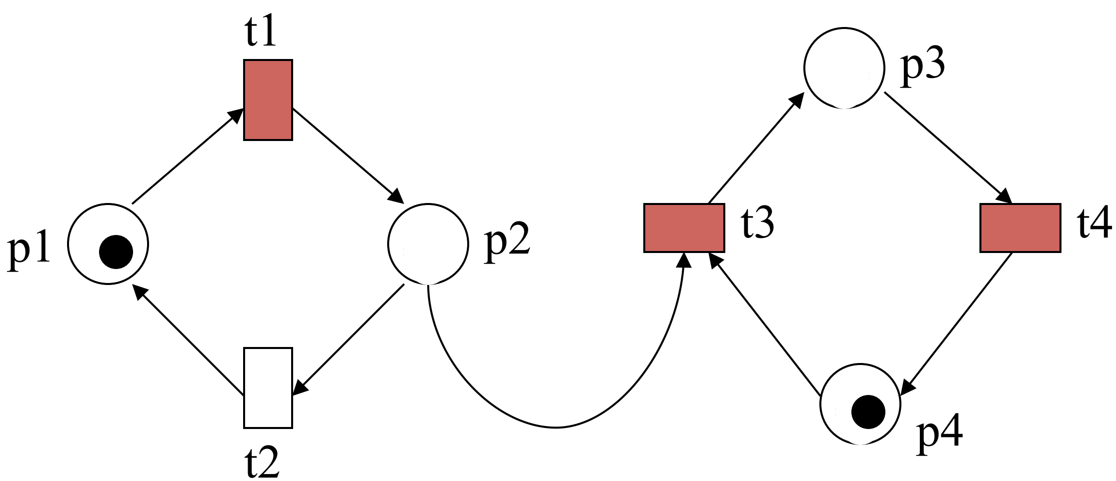
Liveness

- A Petri net with initial marking M0 is *live* if, no matter what marking has been reached from M0, it is possible to ultimately fire *any* transition by progressing through some further firing sequence.
- A live Petri net guarantees *deadlock-free* operation, no matter what firing sequence is chosen.

Liveness

- The vending machine is live and the producer-consumer system is also live.
- A transition is *dead* if it can never be fired in any firing sequence.

An Example



M0 = (1,0,0,1)

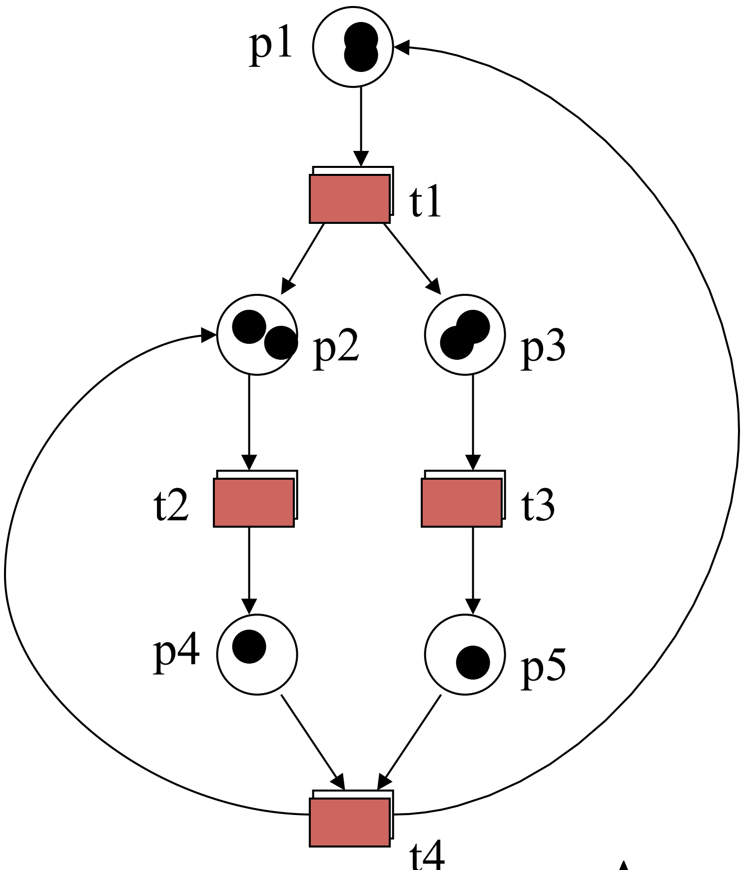
M1 = (0,1,0,1)

M2 = (0,0,1,0)

M3 = (0,0,0,1)

A bounded but non-live Petri net

Another Example



M0 = (1, 0, 0, 0, 0)

M1 = (0, 1, 1, 0, 0)

M2 = (0, 0, 0, 1, 1)

M3 = (1, 1, 0, 0, 0)

M4 = (0, 2, 1, 0, 0)

An unbounded but live Petri net

Analysis Methods

- Reachability Analysis:
  - Reachability or coverability tree.
  - State explosion problem.
- Incidence Matrix and State Equations.
- Structural Analysis
  - Based on net structures.

Behavioral properties (1)

- Properties that depend on the initial marking
- Reachability
  - Mn is reachable from M0 if exists a sequence of firings that transform M0 into Mn
  - reachability is decidable, but exponential
- Boundedness
  - a PN is bounded if the number of tokens in each place doesn't exceed a finite number k for any marking reachable from M0
  - a PN is safe if it is 1-bounded

Behavioral properties (2)

- Liveness
  - a PN is live if, no matter what marking has been reached, it is possible to fire any transition with an appropriate firing sequence
  - equivalent to deadlock-free
  - strong property, different levels of liveness are defined (L0=dead, L1, L2, L3 and L4=live)
- Reversibility
  - a PN is reversible if, for each marking M reachable from M0, M0 is reachable from M
  - relaxed condition: a marking M' is a home state if, for each marking M reachable from M0, M' is reachable from M

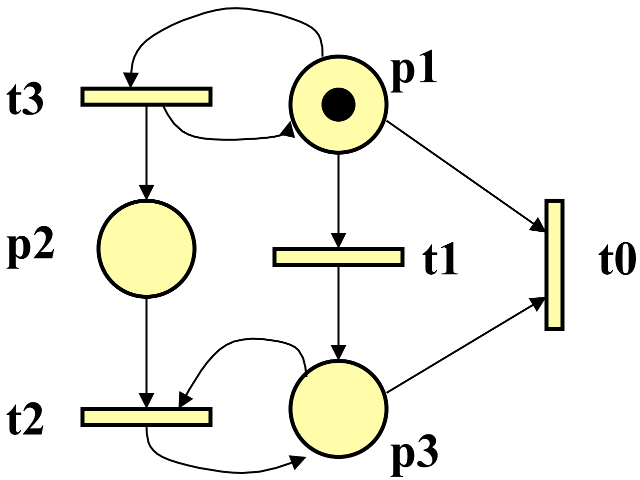
Behavioral properties (3)

- Coverability
  - a marking is coverable if exists M' reachable from M0 s.t. M'(p)>=M(p) for all places p
- Persistence
  - a PN is persistent if, for any two enabled transitions, the firing of one of them will not disable the other
  - then, once a transition is enabled, it remains enabled until it's fired
  - all marked graphs are persistent
  - a safe persistent PN can be transformed into a marked graph

Analysis methods (1)

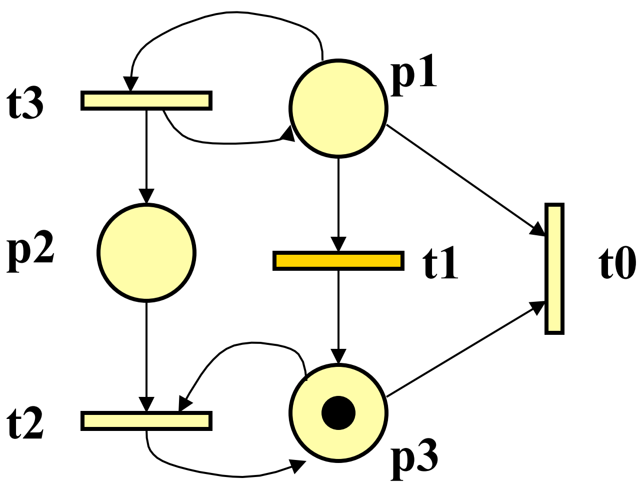
- Coverability tree
  - tree representation of all possible markings
    - root = M0
    - nodes = markings reachable from M0
    - arcs = transition firings
  - if net is unbounded, then tree is kept finite by introducing the symbol ω
  - Properties
    - a PN is bounded iff ω doesn't appear in any node
    - a PN is safe iff only 0's and 1's appear in nodes
    - a transition is dead iff it doesn't appear in any arc
    - if M is reachable form M0, then exists a node M' that covers M

Coverability tree example



M0=(100)

Coverability tree example

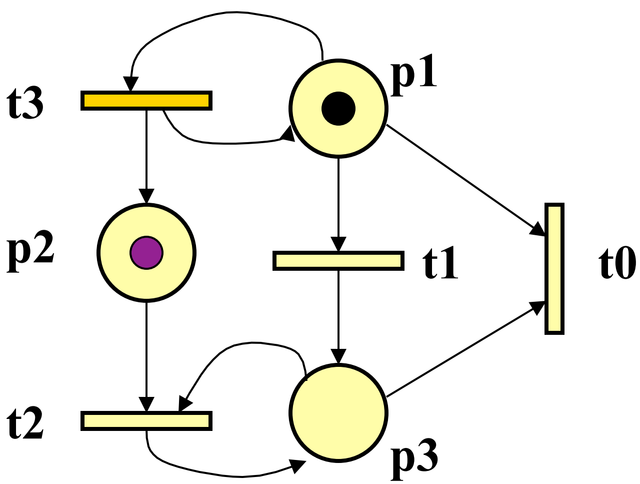


M0=(100)

t1

M1=(001)
"dead end"

Coverability tree example

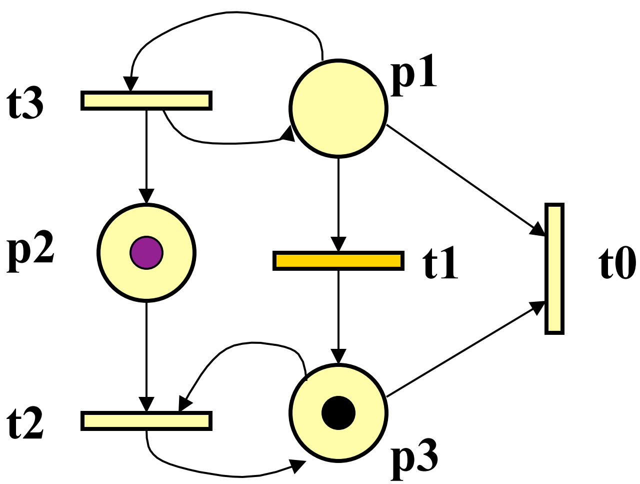


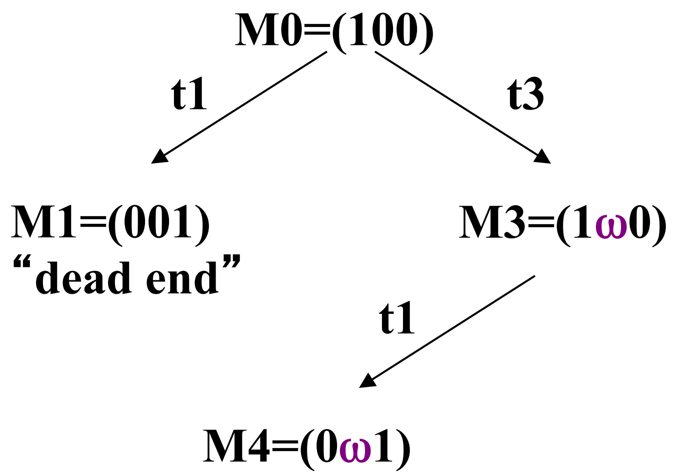
M0=(100)

t1 → M1=(001) "dead end"

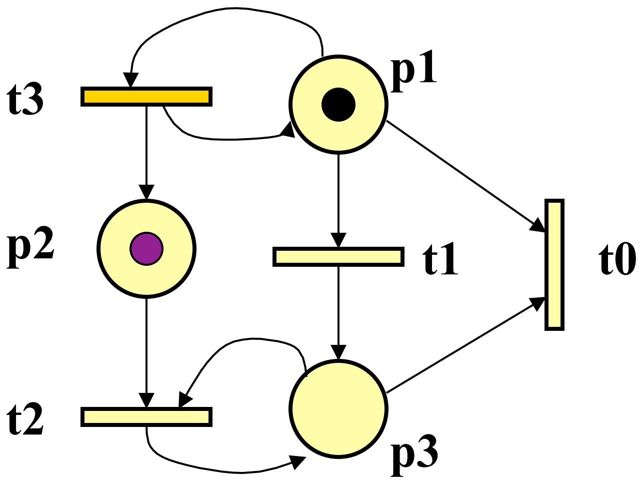
t3 → M3=(1ω0)

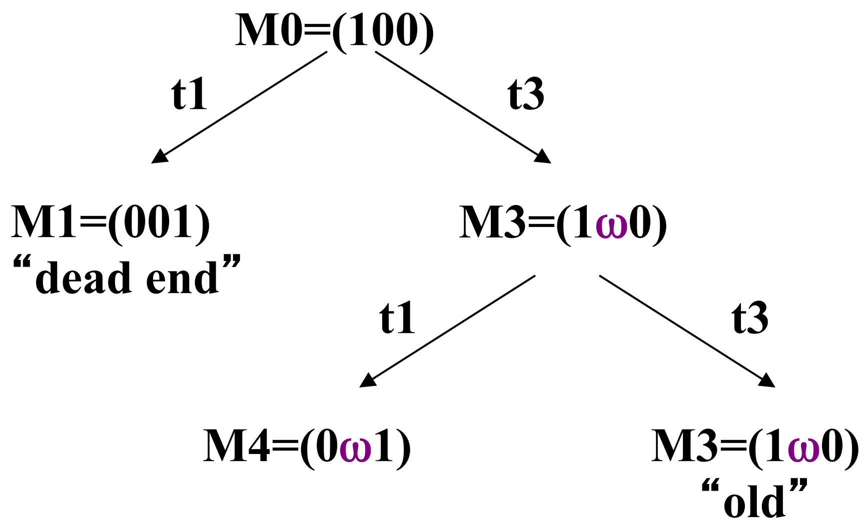
Coverability tree example



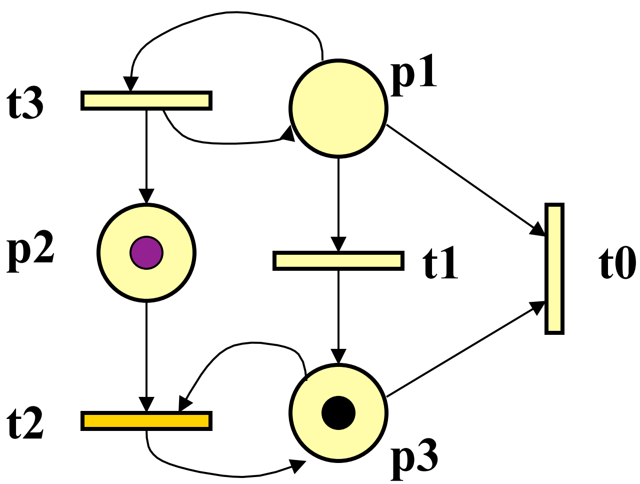


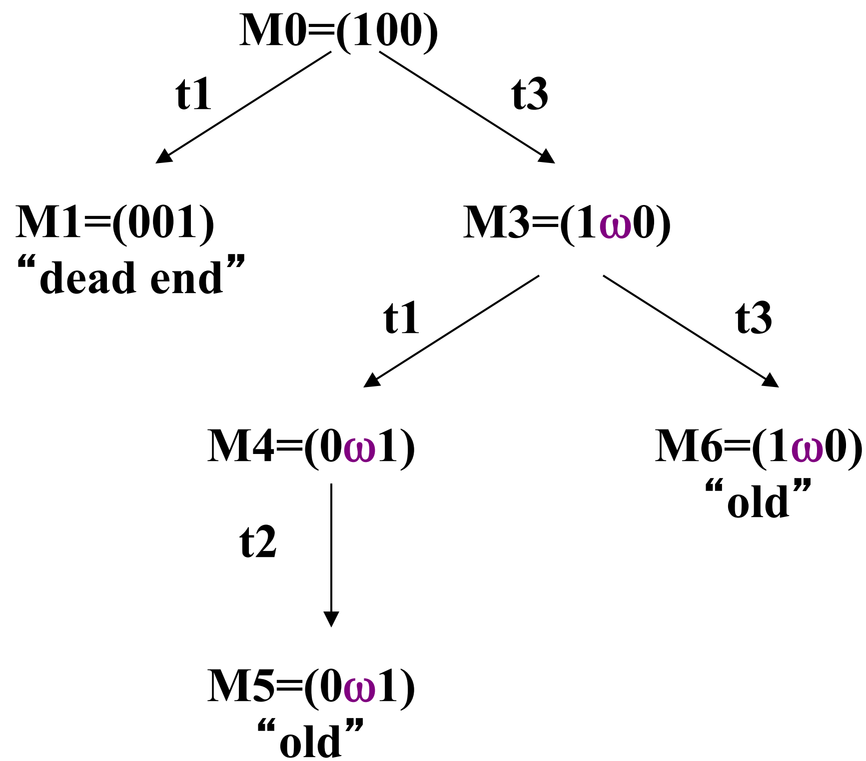
Coverability tree example



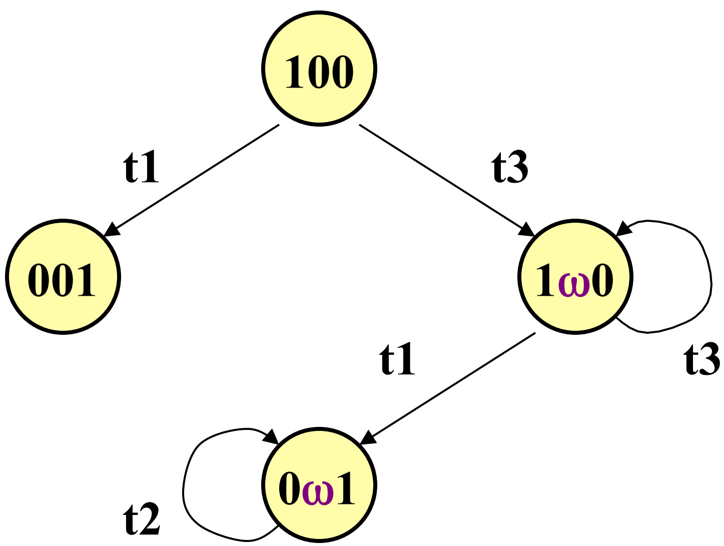


Coverability tree example

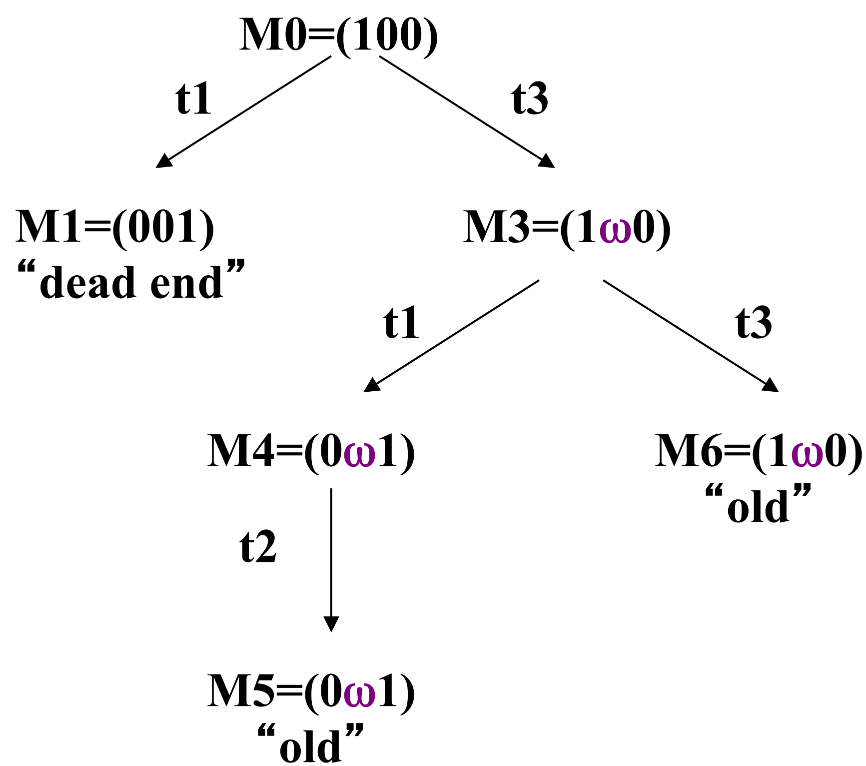




Coverability tree example



coverability graph



coverability tree

Subclasses of Petri Nets (1)

- Ordinary PNs
  - all arc weights are 1's
  - same modeling power as general PN, more convenient for analysis but less efficient
- State machine
  - each transition has exactly one input place and exactly one output place
- Marked graph
  - each place has exactly one input transition and exactly one output transition

Subclasses of Petri Nets (2)

- Free-choice
  - every outgoing arc from a place is either unique or is a unique incoming arc to a transition
- Extended free-choice
  - if two places have some common output transition, then they have all their output transitions in common
- Asymmetric choice (or simple)
  - if two places have some common output transition, then one of them has all the output transitions of the other (and possibly more)

Extensions

- High-level nets
  - Tokens have “colors”, holding (complex) information.
- Timed nets
  - Time delays associated with transitions and/or places.
  - Fixed delays or interval delays.
  - Stochastic Petri nets: exponentially distributed random variables as delays.

Thanks

- Chris Ling
- Gabriel Eirea