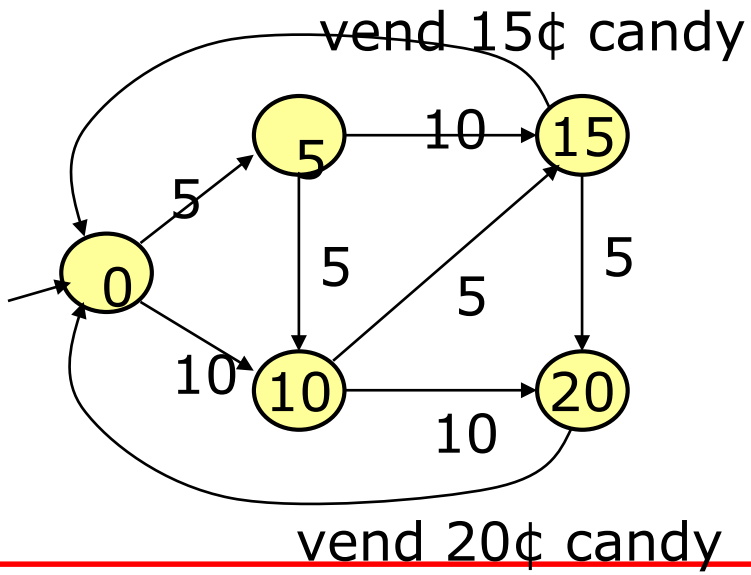
ECE 09468/09568
Discrete Event Systems

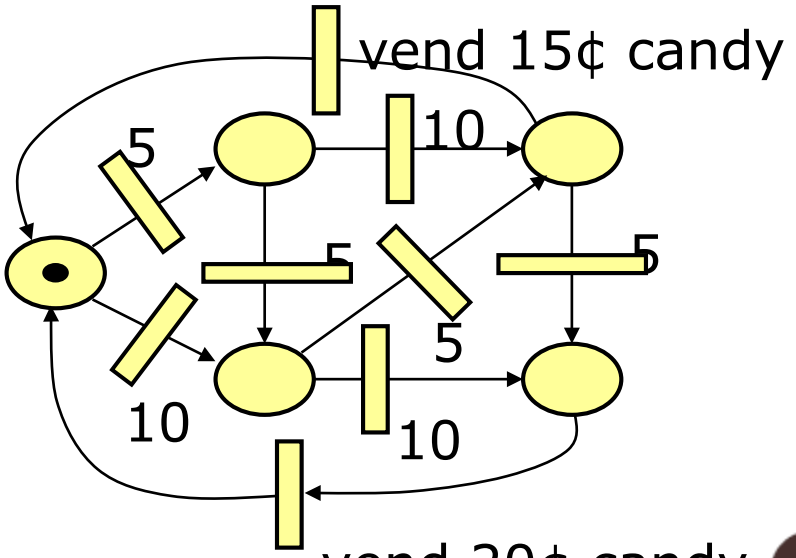
Lecture 8: **Petri Nets (PNs) – Part IV**

Dr. Ying (Gina) Tang
Department of Electrical and Computer Engineering
Rowan University

Special Petri Nets

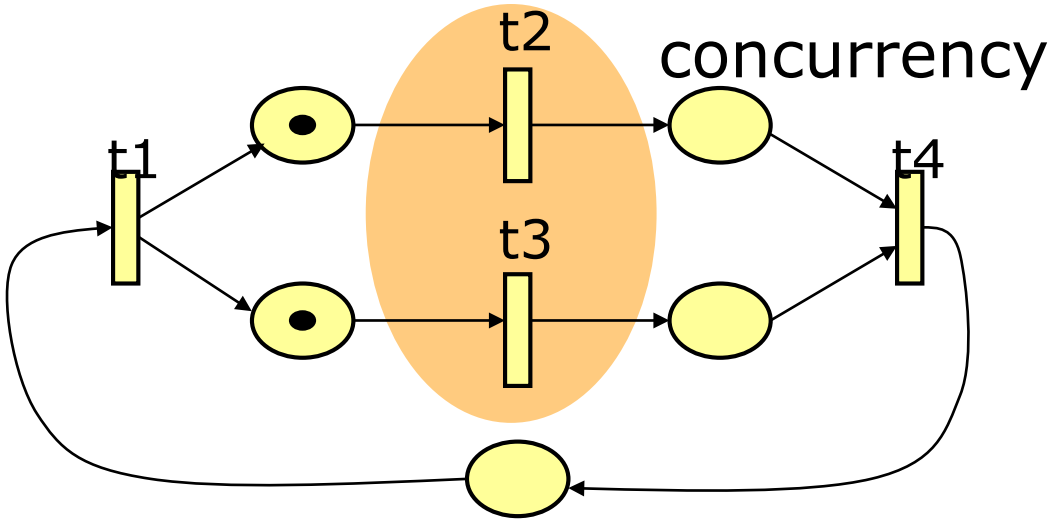
- Ordinary Petri Net: A Petri net is called binary or ordinary if all its weights are 1s. Formally, $W: F \rightarrow \{1\}$
- Petri Net State Machine: A Petri net state machine is a binary PN such that each transition has exactly one input place and exactly one output place. *Formally* $\forall t \in T: |{}^\bullet t| = |t^\bullet| = 1$





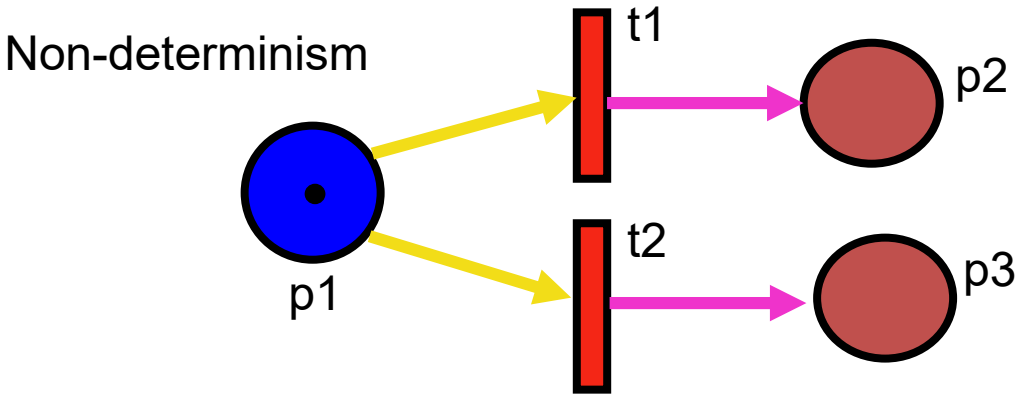
Special Petri Nets

- Marked graph: A binary Petri net is called a marked graph or event graph if each place has exactly one pre-transition and exactly one post-transition. Formally, $\forall p \in P$: $|{}^\bullet p| = |p^\bullet| = 1$
  - ❖ Marked graphs allow representation of concurrency but not decisions (conflicts)



Special Petri Nets

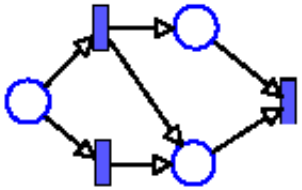
- Two events e1 and e2 are in conflict if either e1 or e2 can occur but not both, and they are concurrent if both events can occur in any order without conflicts



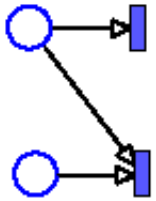
- Two transitions fight for the same token: **conflict**.
- Even if there are two tokens, there is still a conflict
- The presence of a common resource generates the conflicts
- The token in p1 enable both t1 and t2. If t1 fires, t2 will be disable. If t2 fires, t1 will be disable.

Special Petri Nets

- Free-choice net: A binary Petri net such that every arc going out of a place is either (a) unique arc incoming into a transition and no other arcs go out of the place or (b) there are many arcs, but each of them is unique arc going into the transition. *Formally* $\forall p \in P$: $|p^\bullet| \leq 1$ *or* $^\bullet(p^\bullet)=\{p\}$
  - ❖ If a place belongs to the preset of several transitions, then it should be the only input for all of these transitions. Hence either all of these conflicting transitions are simultaneously enabled, or none of them. This allows the choice (conflict resolution) as to **which transition is to fire to be made freely.**

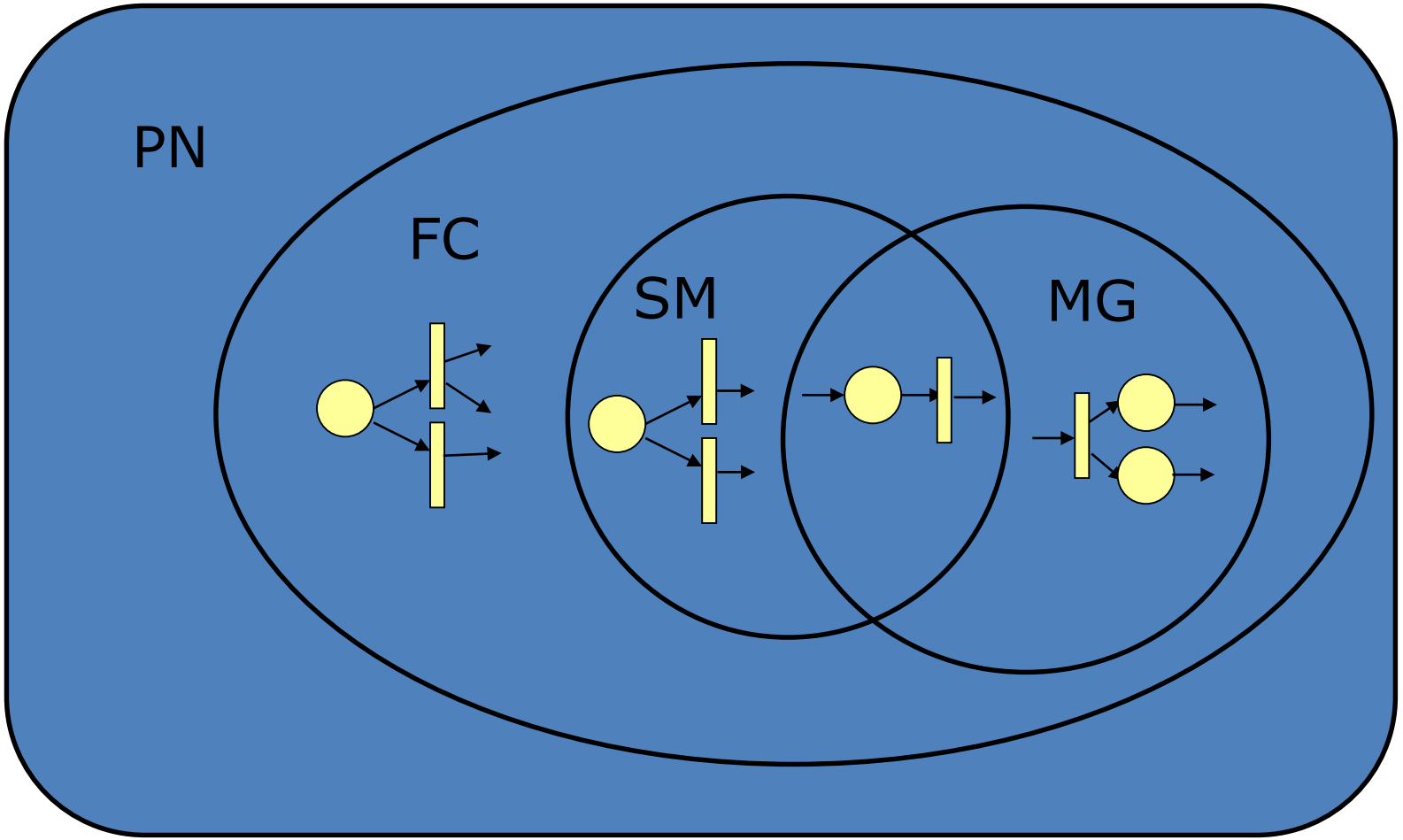


Free Choice



Not Free Choice

Subclasses of Petri Nets

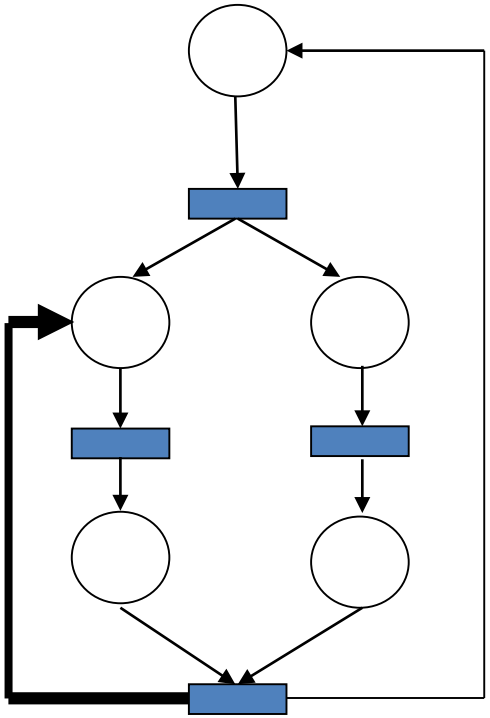


Other Properties of Petri Nets

- **Persistence**
  - a PN is persistent if, for **any two enabled transitions, the firing of one transition will not disable the other.**
  - A transition in persistent net, once it is enabled, will stay enabled until it fires.
  - Persistency is closely related to **conflict-free** nets.
  - All marked graphs are persistent but not all persistent nets are marked graphs.

Other Properties of Petri Nets

- Persistent PN but not a Marked Graph.



Adding the thick arc makes it a non-marked graph.

It is still persistent.

The places in Persistent PN may have more than one input arc.

Infinite vs Finite Capacity PN

- **Infinite capacity Petri Net:** Places can accommodate unlimited number of tokens
- **Finite capacity Petri Net:** having upper limit to the number of tokens that each place can hold
- **Strict transition rule:** where capacity constraint is applied (after firing, each output place can't have more than K(p) tokens)
- **(Weak) transition rule:** rule without capacity constraint

Firing in Finite Capacity PN

- The firing of a $t \in T$ is enabled if there is enough token on the input places:

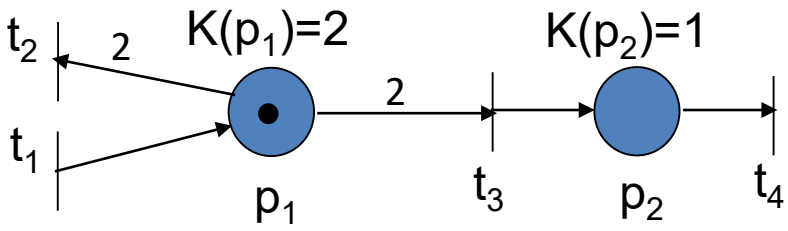
$$\forall\ p \in {}^\bullet t,\ m(p) \geq I(p, t)$$

- Capacity limit (after the m[t > m' firing):

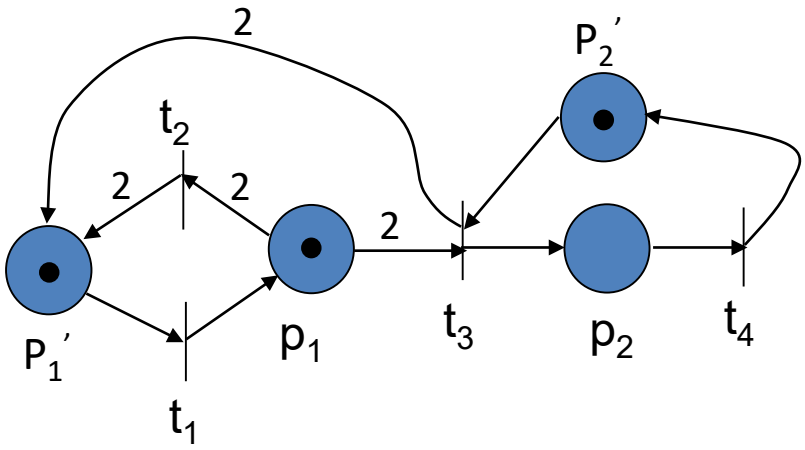
$$\forall\ p \in t^\bullet,\ m'(p) = m(p) + O(p, t) \leq K(p)$$

Finite Capacity PN

A finite capacity PN (N, M_0) with strict transition rule can be transformed to a PN (N', M_0') with week transition rule.



PN with strict transition rule



PN with (weak) transition rule

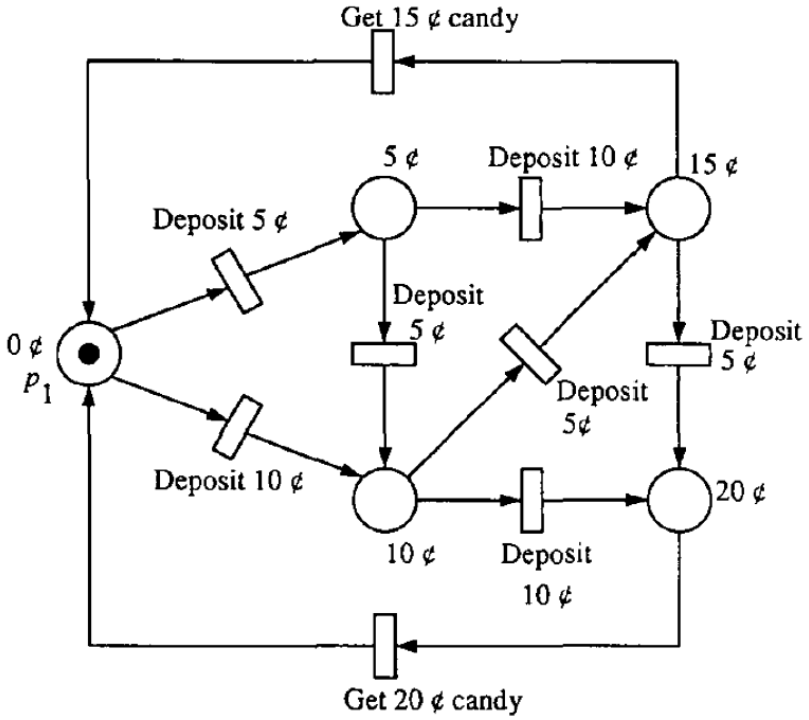
The steps to make the transform:

1) Add a complementary place p' for each place p with a finite capacity, where the initial marking of p' is given by M0'(p') = K(p)-M0(p)

2) Between each transition t and complementary places p', draw new arcs(t,p') or (p',t) where w(t,p')=w(p,t) and w(p',t)=w(t,p), so that the sum of the tokens in place p and its complementary places p' equals its capacity K(p) for each place p, before and after firing the transition t.

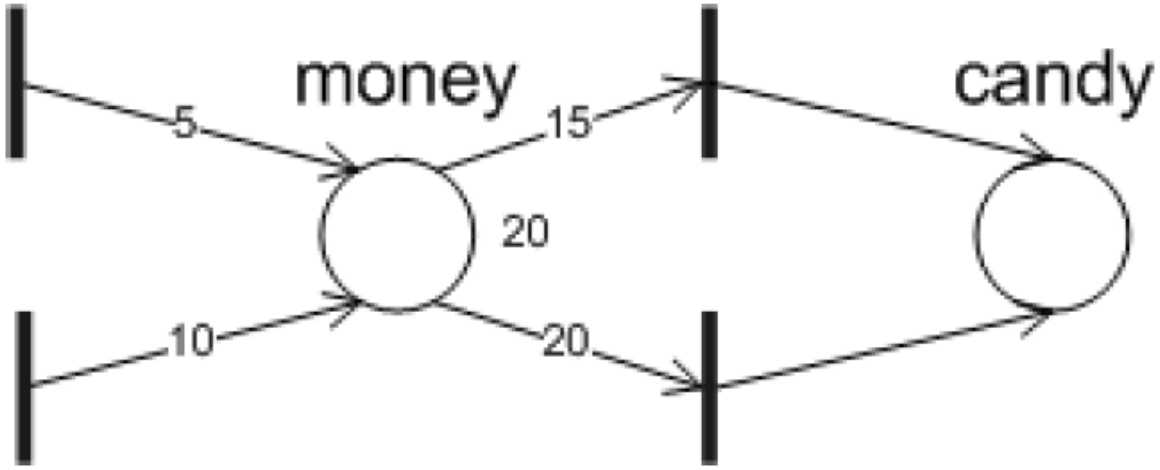
Finite Capacity PN

- Example problem: Construct a PN for a vending machine
  - ❑ Accepts 5 cents or 10 cents
  - ❑ Sells 15 cents or 20 cents candy
  - ❑ Can hold up to 20 cents
- One possible solution:
  - ❑ Model it as a PN state machine



Finite Capacity PN

- Another solution:
  - ❑ Model it as a finite capacity PN

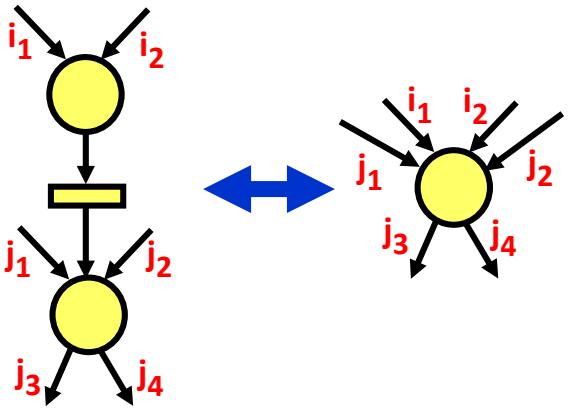


- Can you transform the above PN to an equivalent one without capacity limits?

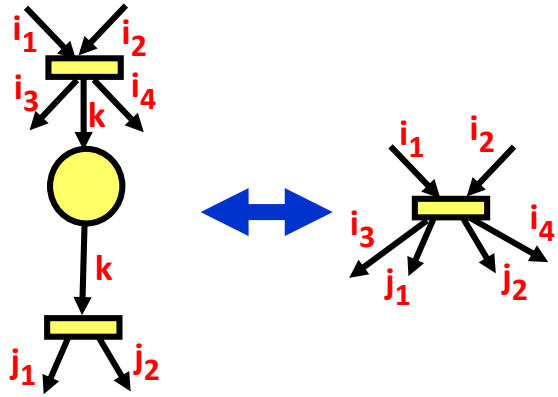
Reduction Techniques

- Help to cope with the complexity problem
- Reduction rules that preserve liveness, safeness and boundedness
  - Fusion of Series Places
  - Fusion of Series Transitions
  - Fusion of Parallel Places
  - Fusion of Parallel Transitions
  - Elimination of Self-loop Places
  - Elimination of Self-loop Transitions

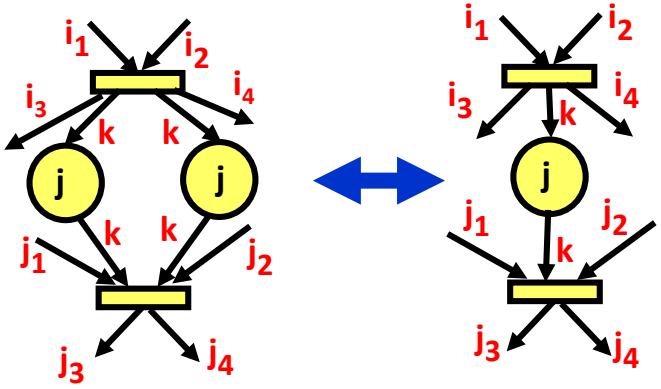
Reduction Rules



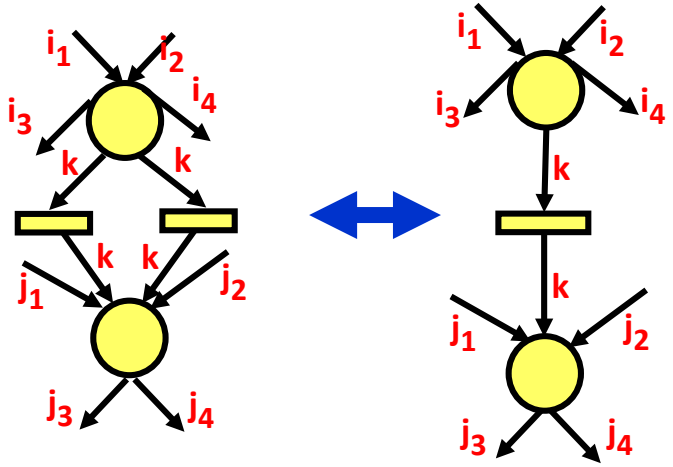
Rule a: Fusion of series places



Rule b: Fusion of series transitions



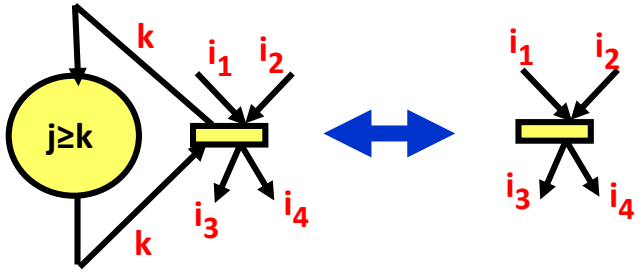
Rule c: Fusion of parallel places
(j is non-negative integer)



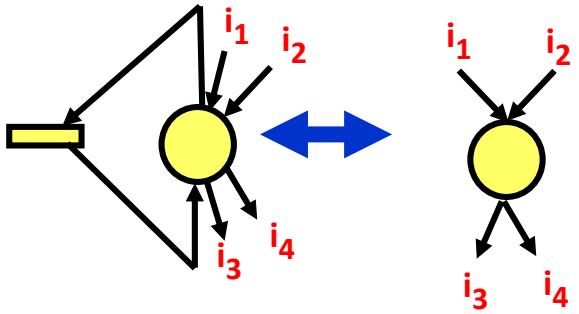
Rule d: Fusion of parallel transitions

Reduction Rules

j is # of token in the place

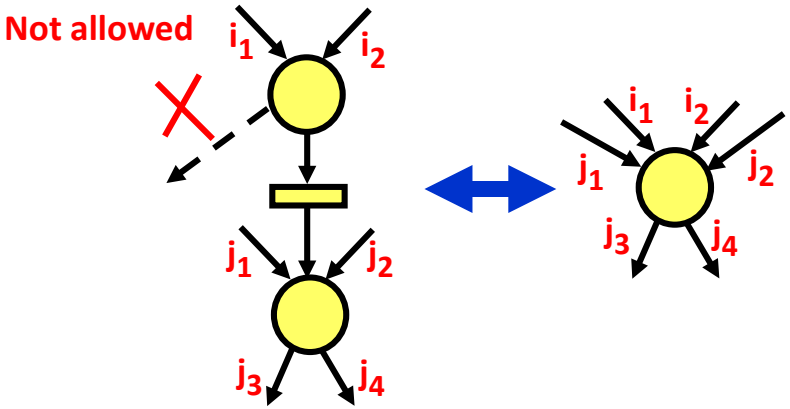


Rule e: Elimination of self-loop places

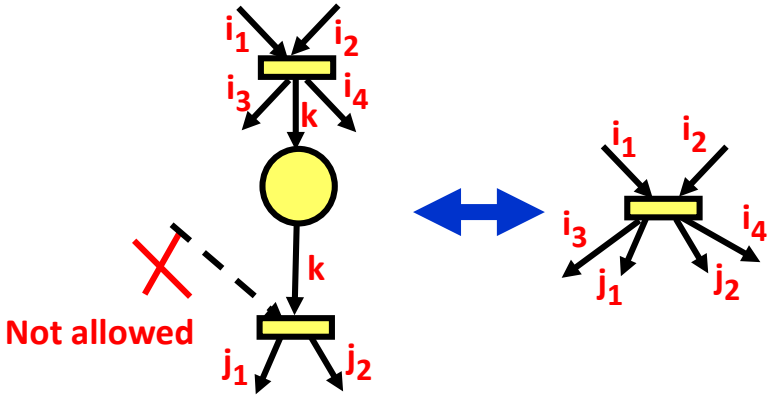


Rule f: Elimination of self-loop transitions

Reduction Rules



Rule a: Fusion of series places

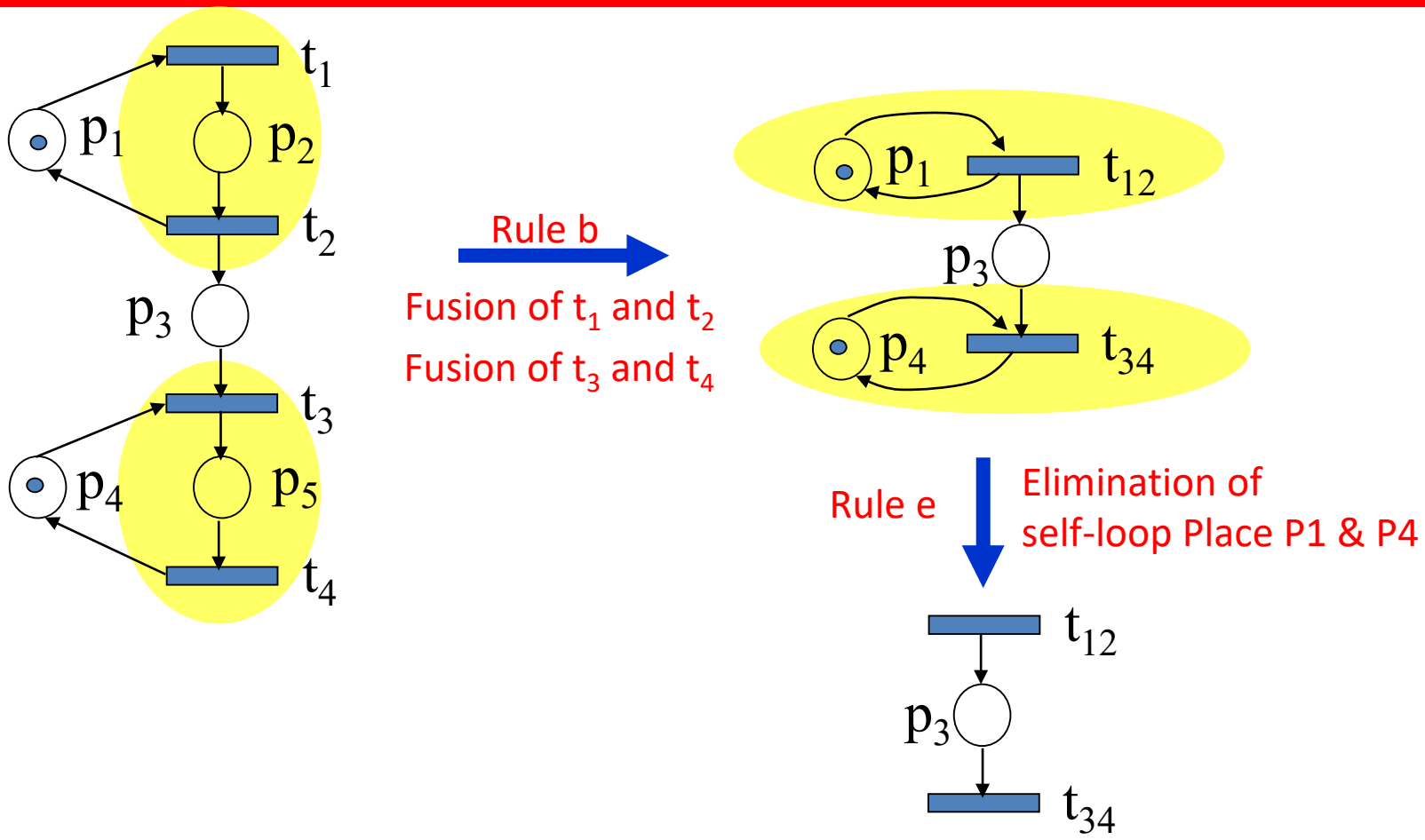


Rule b: Fusion of series transitions

Important Considerations

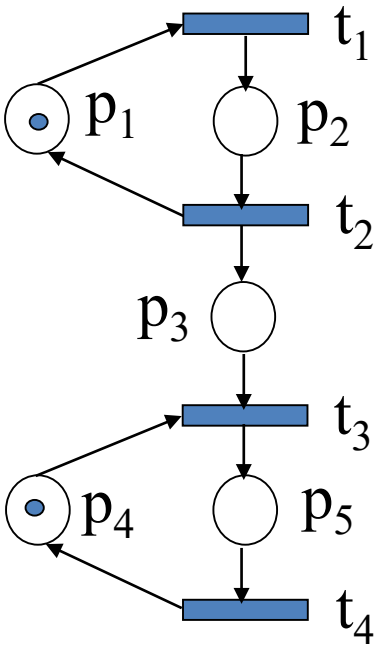
- The six rules can preserve the **boundedness, liveness and reversibility**
- Safeness can be guaranteed by Rules a, c, d, and f.
- Rules b and e can only guarantee safeness when k = j = 1
- Fusion of two places (Rule a) cannot preserve the properties if the multiplicity of the concerned arcs is not single.
- Wrong use of rule a or b as indicated by crossed arcs guarantee no properties preserved.
- Rules can be applied sequentially or concurrently subject to the network configuration. The approach should be carefully used.

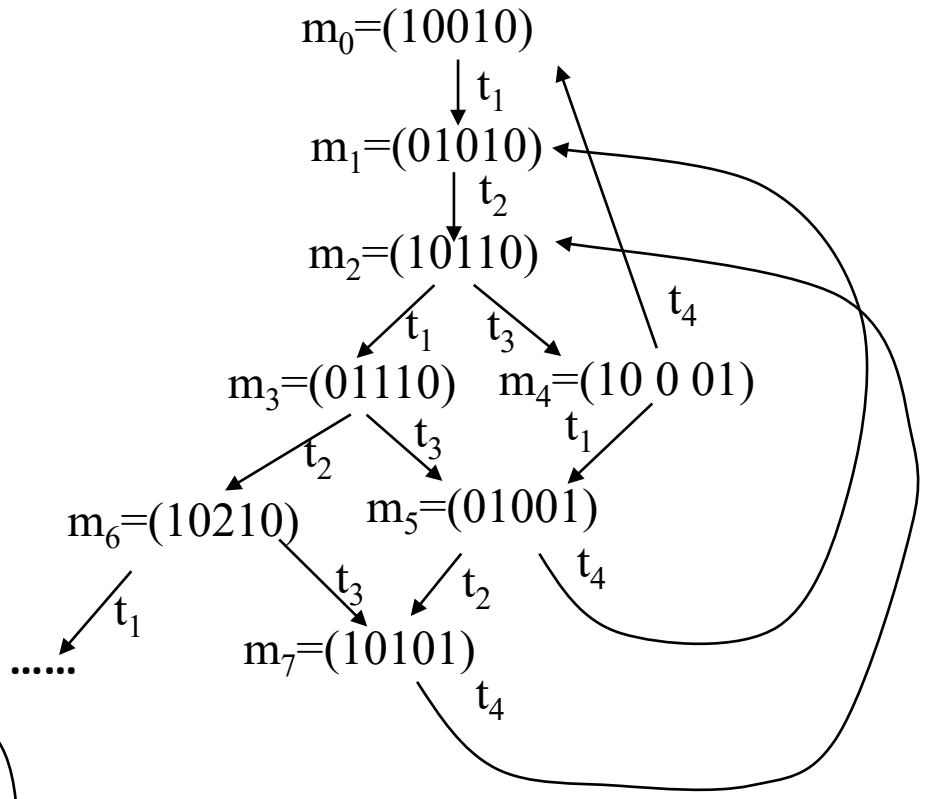
Example: Producer/Consumer

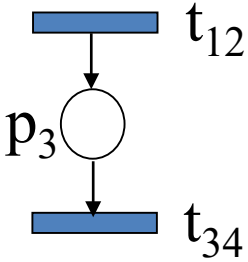


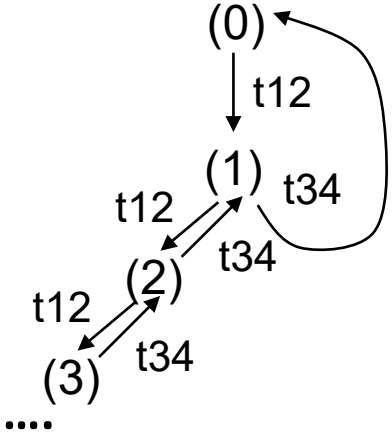
It is live, reversible, not bounded.

Example: Producer/Consumer

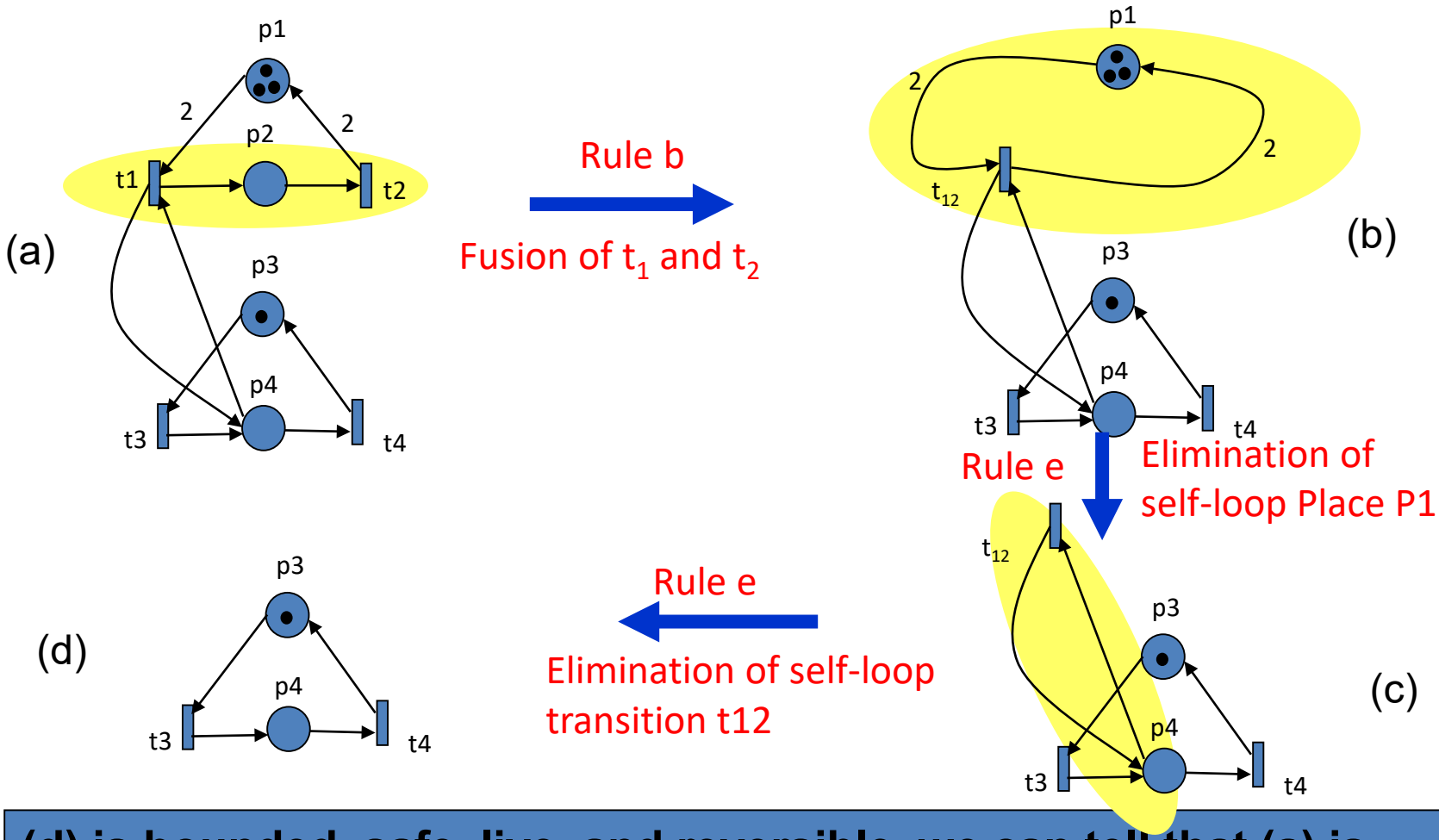








Example of Reduction Rules



(d) is bounded, safe, live, and reversible. we can tell that (a) is bounded, live, and reversible but it may not be safe. When rule (e) is used, to preserve safeness, # of the tokens in self-loop place has to be 1 and weights of the arcs have to be 1.

- Your turn:
  - ❑ Apply the reduction rules to simplify the following PN