Real-Time Programming

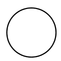
Farhang Nemati

Spring 2016

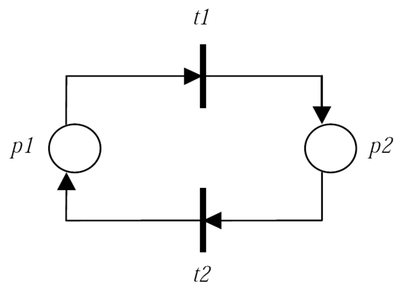
Petri Net

- A graphical and formal (mathematical) method for modeling and analyzing systems. They are specially useful for modeling systems that contain concurrent and asynchronous processing
- Model and analysis a system in design phase
- It's graphical, thus a system can be visualized
- It's mathematical, thus analysis and simulation can be done using tools
- Describes different states of a system and the conditions and events that transit the system from a state to another one

Places, Transitions, Arcs

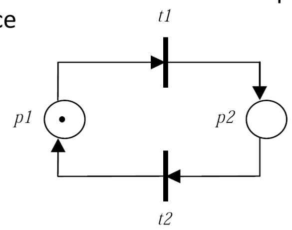
- A Petri net contains arcs and two types of nodes; places and transitions and
- Place: Shown by a circle 
  - Represents a condition, e.g., a resource is available, some data is available, a signal is arrived, a buffer is empty/full, etc.
- Transition: Shown by a solid bar or a rectangle
  - Represents an event, task, computation, processing, etc.
- Arc: Shown by a directed arc
  - Connects a place to a transition or a transition to a place. It does NOT connect two places or two transitions!

Example



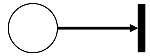
Token

- Token: Shown by a solid dot: •
  - To describe the behavior of a Petri net. Represents the fulfillment of a condition, e.g., a resource is available, data is ready, a signal is available, etc.
  - A place can contain any number of tokens. If the number of tokens in one place is too high a number is written in the place showing the number of tokens in that place

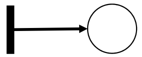


Input-Places, Output-Places, Generator, Stop

- A place that is connected by an arc **to** a transition is input-place for the transition



- A place that is connected by an arc **from** a transition is output-place for the transition

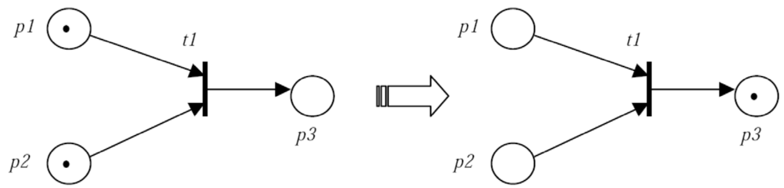


- A transition can have multiple input-places and/or output-places
- A transition without any input-place is called Generator (Source), and a transition without any output-place is called Stop (Sink)

Enabled Transition, Firing a Transition

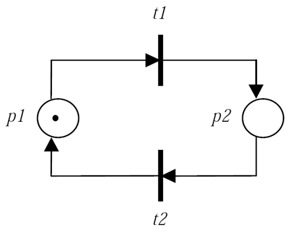
- A transition is enabled if all its input-places contain token
- Firing a transition: if a transition is enabled it can be fired
- When a transition is fired the tokens are removed from all of its input-places and tokens are added to all its output-places
  - The number of tokens taken from input-places might be different from the number of tokens added to output-places

Firing Example



Firing Sequence

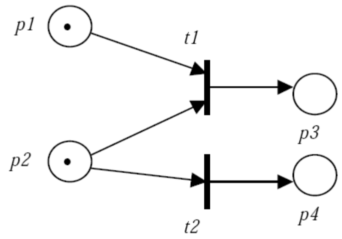
- Firing Sequence: A sequence of firing transitions



- Example: (t1, t2, t1, t2)

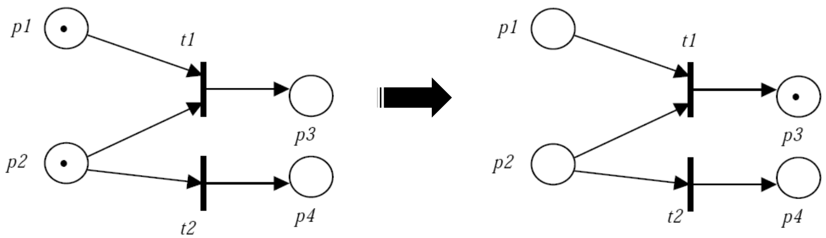
Firing Sequence

- If multiple transitions are enabled at the same time, firing sequence is not deterministic



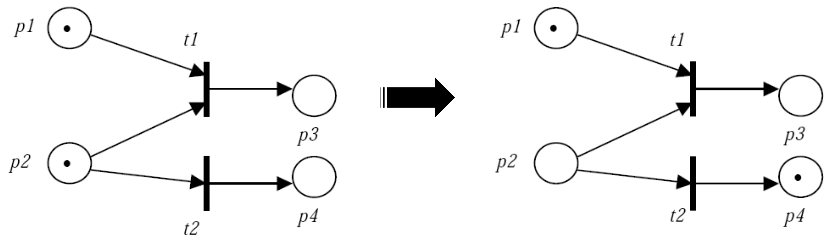
Firing Sequence

- Firing sequence is not deterministic
- 1)



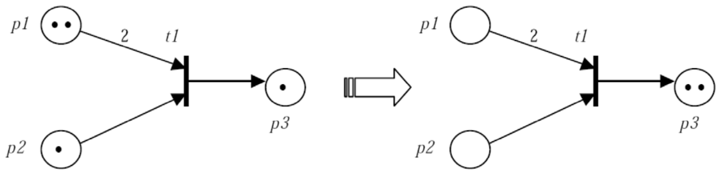
Firing Sequence

- Firing sequence is not deterministic
- 2)



Weighted Arcs

- Weighted Arcs: An arc can be weighted, i.e., by a number is written next to it. The number shows how many tokens it will take from input-place or will add to a output-place:

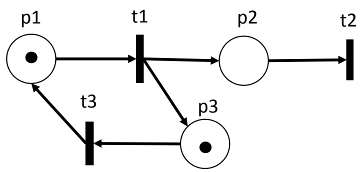


Marking

- Marking: A marking, M, of a Petri net is the distribution of tokens over the places, i.e., how many tokens are in each place. It shows the current state of the system
- A marking M can be shown by a tuple, (M(p1),M(p2), …) where M(pi) is the number of tokens in place pi: (0,1,1,0)

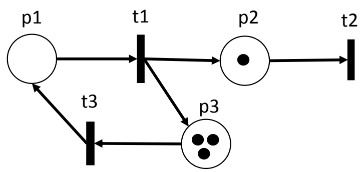
Marking Example

- Example1: (1,0,1)



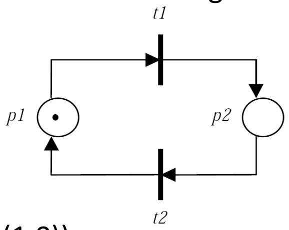
Marking Example

- Example2: (0,1,3)



Firing Sequence with Markings

- A firing sequence of the following Petri net:



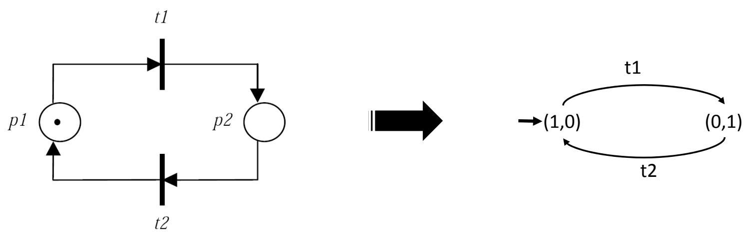
- ((1,0) (0,1) (1,0) (1,0))
- The marking at initial state of a Petri net is called Initial Marking, e.g., (1,0) of the example above.

Reachability Graph

- To be able to analyze a Petri net all possible markings have to be extracted
- Different possible markings of a Petri net drawn from an initial marking is shown by a Reachability Graph.
  - In a reachability graph nodes represent markings and the arrows connecting them represent transitions

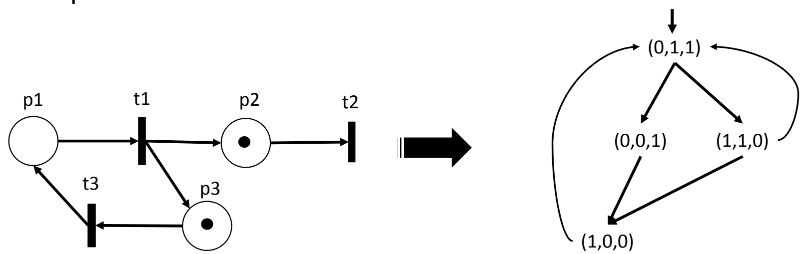
Reachability Graph Example

- Example1



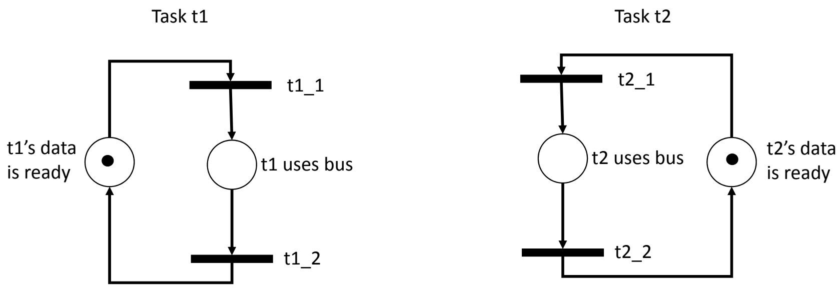
Reachability Graph Example

- Example2

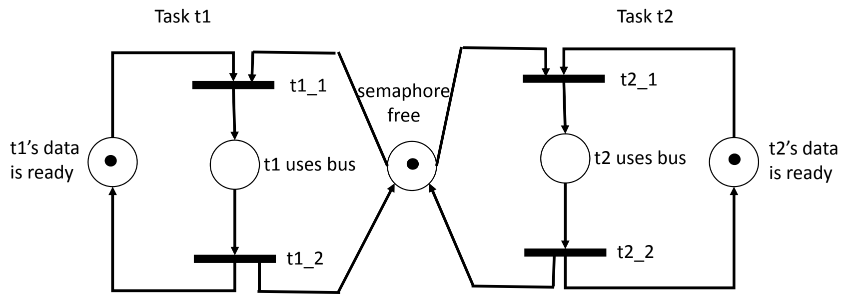


Mutual Exclusion

- Assume two tasks that use a bus to transfer some data. Only one task at a time is allowed to use the bus

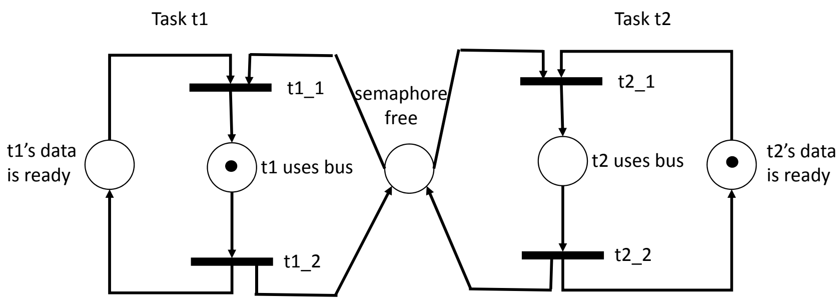


Mutual Exclusion



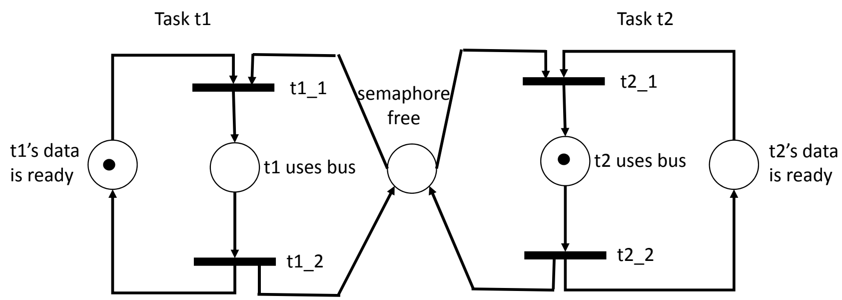
Mutual Exclusion

- A possible marking



Mutual Exclusion

- Another possible marking



Properties of Petri Net

The benefit of modeling a system using a formal modeling method like Petri net is to be able to analyze some properties of the system. The following properties of a system can be analyzed using a Petri net model:

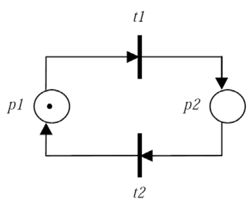
- Reachability
- Liveness
- Boundedness
- Fairness

Reachability

- Given an initial Marking M_0, a marking M is said to be reachable if there exists a firing sequence that transforms M_0 to M
- To check reachability of marking from an initial marking the reachability graph has to be drawn from the initial marking
- This property is useful to check
  - If the system can arrive in a forbidden (erroneous) state, e.g., deadlock
  - If the system can arrive in a desired state

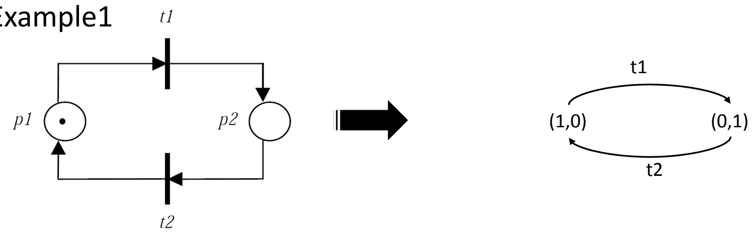
Reachability

- Example1: is (0,1) reachable from the following Petri net? How about (1,1)?



Reachability Example

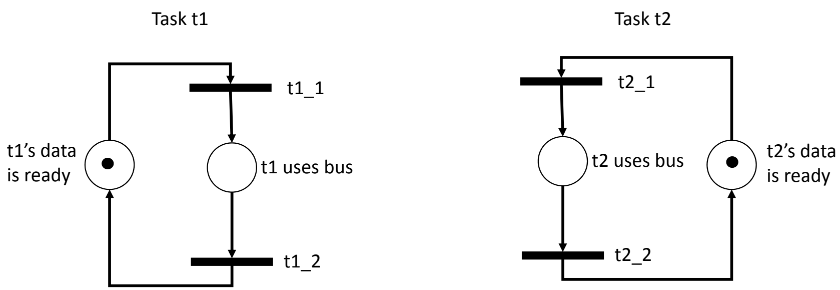
- Example1



- (0,1) is reachable but (1,1) is not reachable

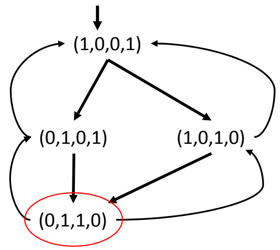
Reachability Example

- Example2: is (0,1,1,0) which is a forbidden state reachable?



Reachability Example

- Example2: Let extract the reachability graph

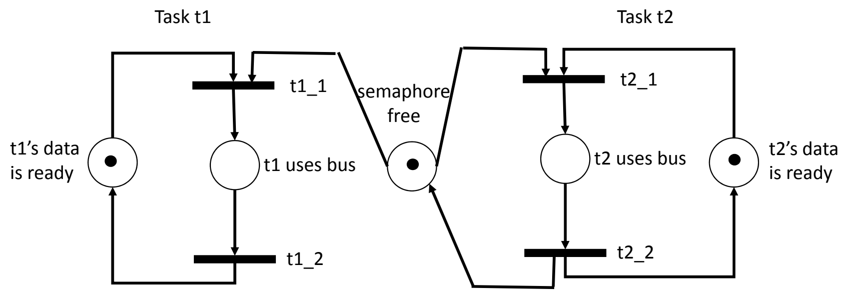


Liveness

- A Petri net is live if it can never stuck in a deadlock state, i.e., there is no marking where no transition can be fired
  - There is no marking in which a transition is disabled permanently, i.e., a transition can be fired infinite times
- This property is used to check whether the system will arrive in a deadlock state

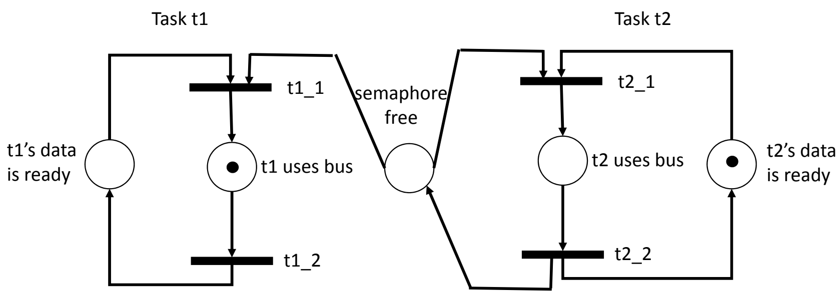
Liveness Example

- Example (t1 does not release the semaphore). Is the following Petri net live?



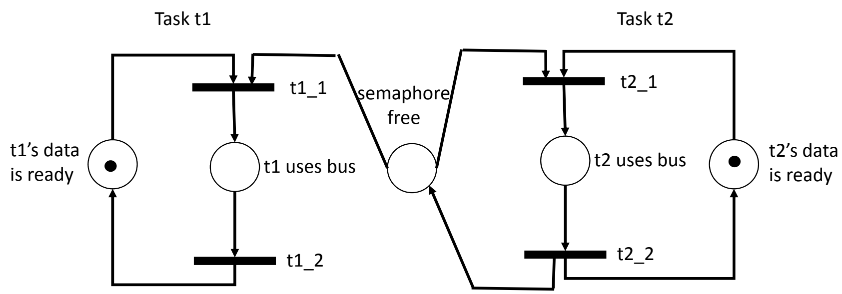
Liveness Example

- A possible marking sequence (t1_1, t1_2):

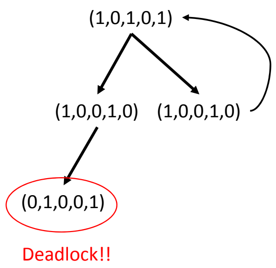


Liveness Example

- It can not proceed anymore, i.e., Deadlock!



Liveness Example

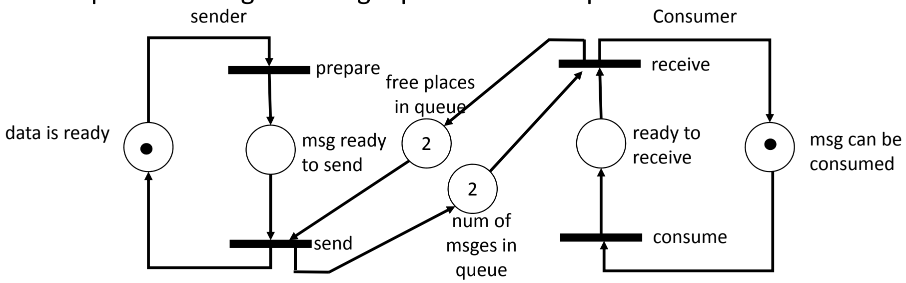


Boundedness

- A Petri net is bounded if it has no marking where the number of tokens in any place is more than k, where k is a positive integer
- If k=1 the Petri net is said to be safe
- This property is used to check if the maximum limits of resources are exceeded

Boundedness Example

- Example. Modeling a message queue used in a producer-consumer

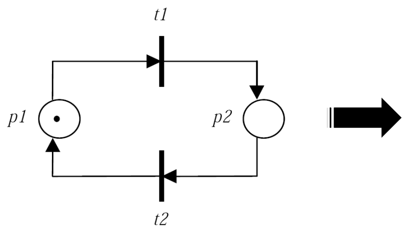


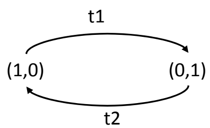
- What is the Bound (limit) of the model? Can at any time the queue have more than 4 free spaces or contain more than 4 messages?

Fairness

- Two transitions are said to be mutually limited fair if there is a limited number of firings for each one before the other one is fired, i.e., one of them can be fired up to a limit until the other one is fired
- Global Fairness:
  - An infinite firing sequence is said to be globally fair if every transition is fired infinite times
  - A Petri net is globally fair if any firing sequence from any initial marking is globally fair

Fairness Example





- (t1,t2,t1,t2,t1, …) : every transition is fired infinite times, thus the Petri net is globally fair
- t1 and t2 are mutually limited fair because none of them can be fired more than once unless the other one is fired

Coverable Marking

- A marking M is said to be coverable by a marking M_i if from an initial marking M_0 marking M_i is reachable and for every place in M_i the number of tokens are equal or greater than the same place in M, i.e. for every place p: $M[p] \leq M_i[p]$
- Example:

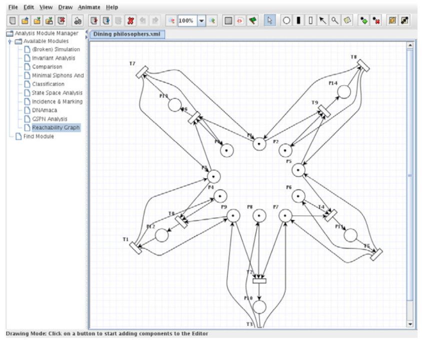
$\rightarrow$ (1,0,3,0) $\longrightarrow$ **(0,1,1,3)** $\longrightarrow$ (1,0,1,2) $\longrightarrow$ **(1,1,2,4)**

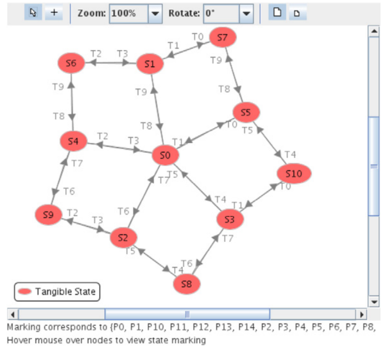
(0,1,1,3) is coverable by (1,1,2,4)

Tools for Modeling Petri Net

- There are many programs available for modeling, analyzing and simulating a system using Petri net models.
- PIPE 2: http://pipe2.sourceforge.net/
  - A free editor program for Petri net
  - Platform independent (written in Java)

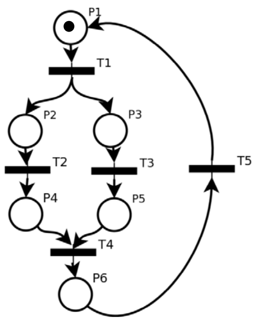
PIPE 2





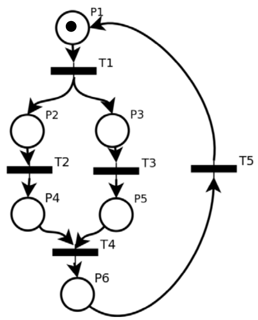
Petri Net Properties Example

- Modeling of a program where two tasks are repeatedly created, run and joined.
- Is the Petri net:
  - Bounded?
  - Live?
  - Fair?

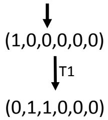


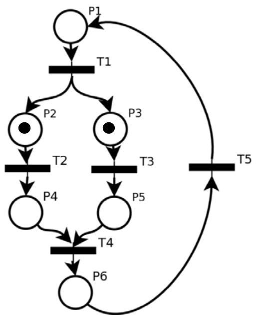
Petri Net Properties Example

(1,0,0,0,0,0)

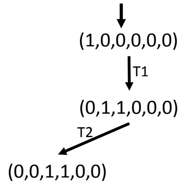


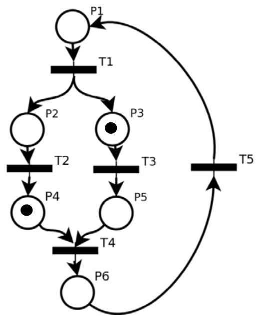
Petri Net Properties Example



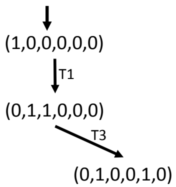


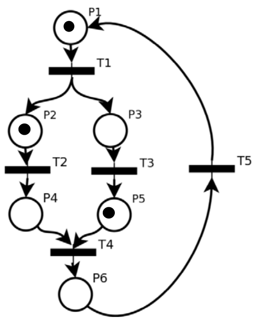
Petri Net Properties Example



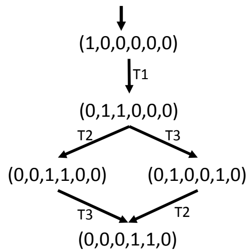


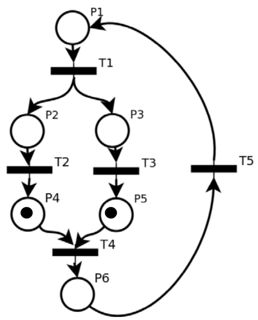
Petri Net Properties Example



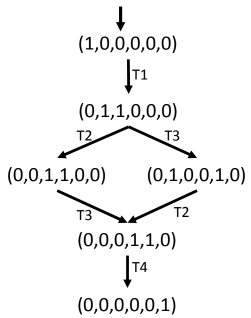


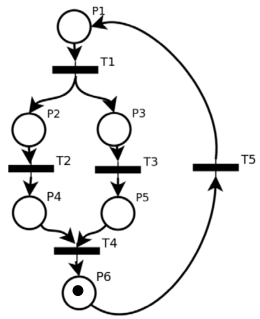
Petri Net Properties Example





Petri Net Properties Example





Petri Net Properties Example