Structured Analysis – Requirements Modeling Petri Nets

Adapted from Schcha

Petri Nets

- A major difficulty with specifying real-time systems is timing
  - Synchronization problems
  - Race conditions
  - Deadlock

- Often a consequence of poor specifications

Petri Nets

- Petri nets
  - A powerful technique for specifying systems that have potential problems with interrelations

- A Petri net consists of four parts:
  - A set of places P
  - A set of transitions T
  - An input function I
  - An output function O

Petri Nets (contd)

- Set of places P is

{p_1, p_2, p_3, p_4}

- Set of transitions T is
  {t_1, t_2}

- Input functions:

  $I(t_1) = \{p_2, p_4\}$
  $I(t_2) = \{p_2\}$

- Output functions:

  $O(t_1) = \{p_1\}$
  $O(t_2) = \{p_3, p_3\}$

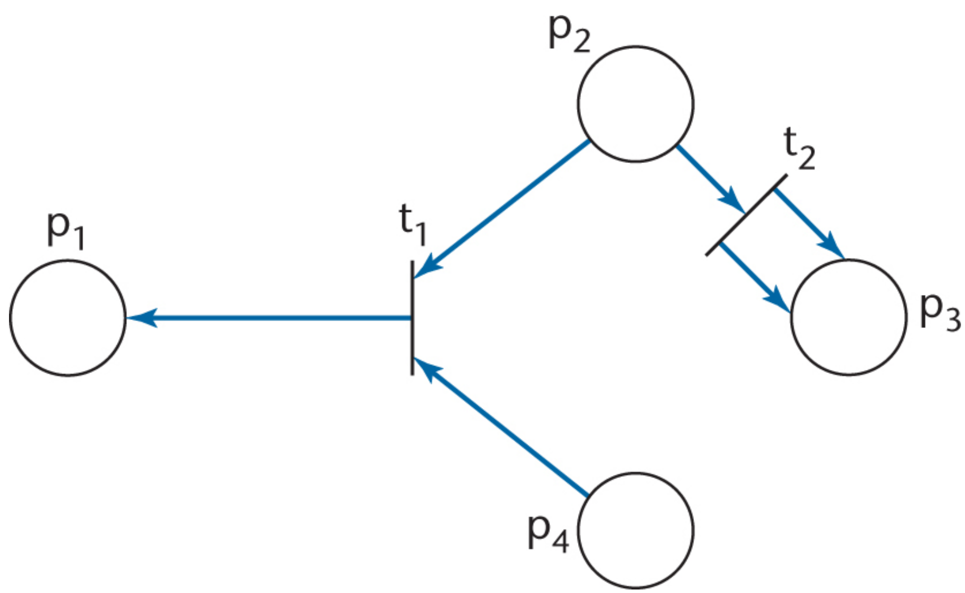


Figure 12.18

Petri Nets (contd)

- More formally, a Petri net is a 4-tuple $C = (P, T, I, O)$
  - $P = \{p_1, p_2, \ldots, p_n\}$ is a finite set of *places*, $n \geq 0$
  - $T = \{t_1, t_2, \ldots, t_m\}$ is a finite set of *transitions*, $m \geq 0$, with P and T disjoint
  - $I : T \rightarrow P^\infty$ is the *input* function, a mapping from transitions to bags of places
  - $O : T \rightarrow P^\infty$ is the *output* function, a mapping from transitions to bags of places
  - (A *bag* is a generalization of a set that allows for multiple instances of elements, as in the example on the previous slide)
  - A *marking* of a Petri net is an assignment of tokens to that Petri net

Petri Nets (contd)

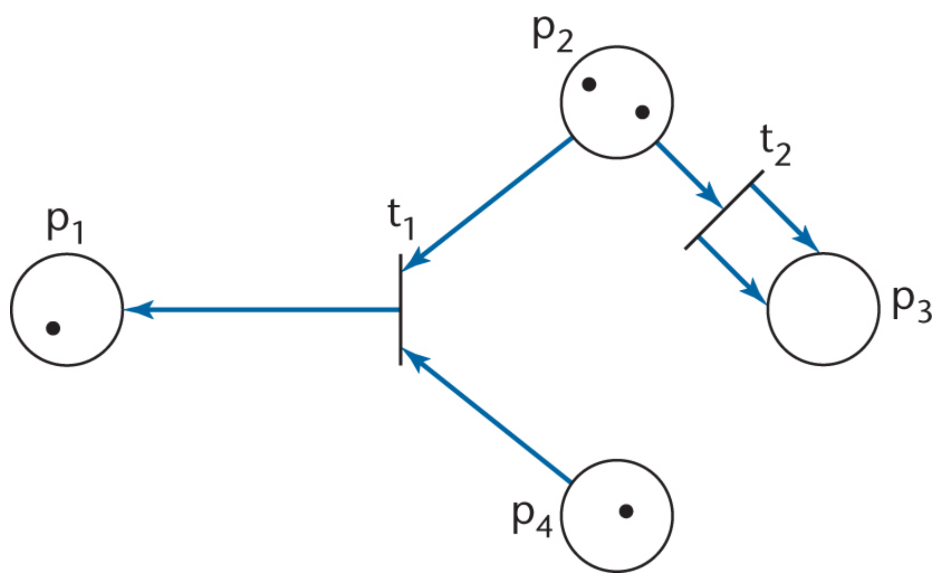


Figure 12.19

- Four tokens: one in p_1, two in p_2, none in p_3, and one in p_4
  - Represented by the vector (1, 2, 0, 1)

Petri Nets (contd)

- A transition is enabled if each of its input places has as many tokens in it as there are arcs from the place to that transition

Petri Nets (contd)

- Transition t_1 is enabled (ready to fire)
  - If t_1 fires, one token is removed from p_2 and one from p_4, and one new token is placed in p_1

- Transition t_2 is also enabled

- Important:
  - The number of tokens is not conserved

Petri Nets (contd)

- Petri nets are indeterminate
  - Suppose t_1 fires

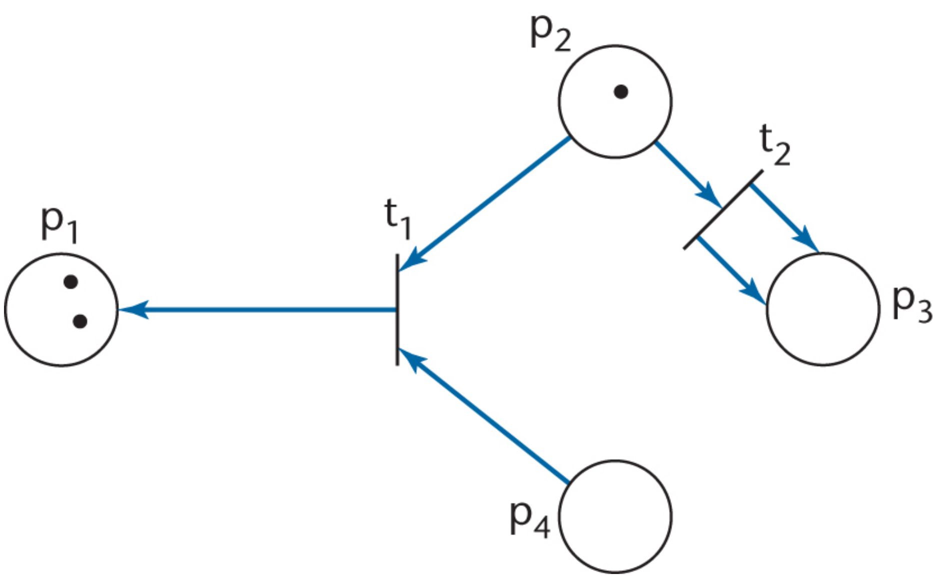
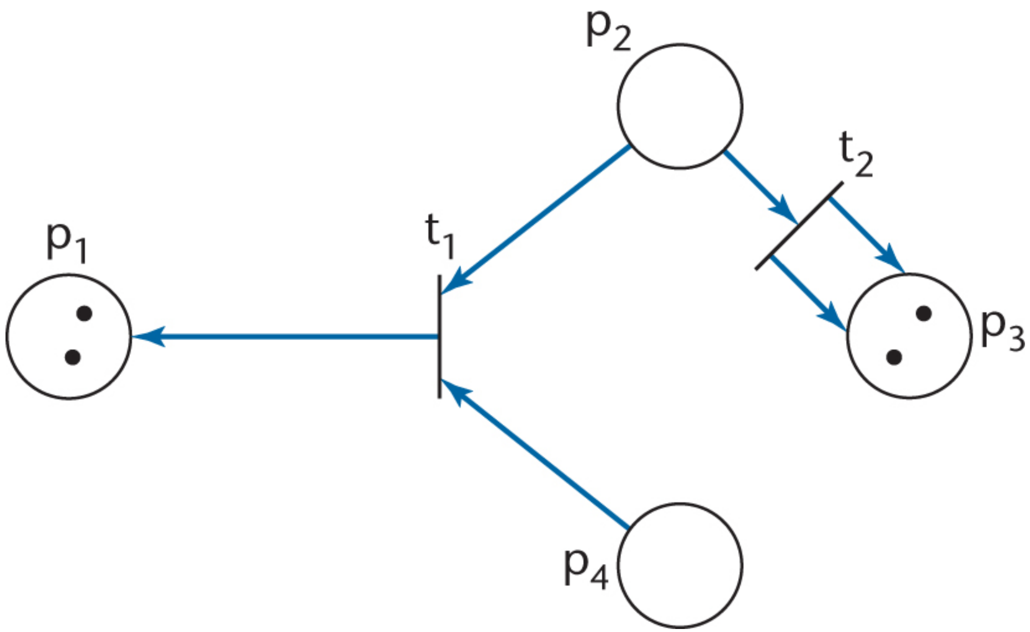


Figure 12.20

- The resulting marking is (2, 1, 0 ,0)

Petri Nets (contd)

- Now t_2 is enabled
  - It fires



- The marking is now (2, 0, 2, 0)

Figure 12.21

Petri Nets (contd)

- More formally, a marking M of a Petri net

  $C = (P, T, I, O)$

  is a function from the set of places P to the non-negative integers

  $M : P \rightarrow \{0, 1, 2, \ldots\}$

- A marked Petri net is then a 5-tuple (P, T, I, O, M)

Petri Nets (contd)

- Inhibitor arcs
  - An inhibitor arc is marked by a small circle, not an arrowhead

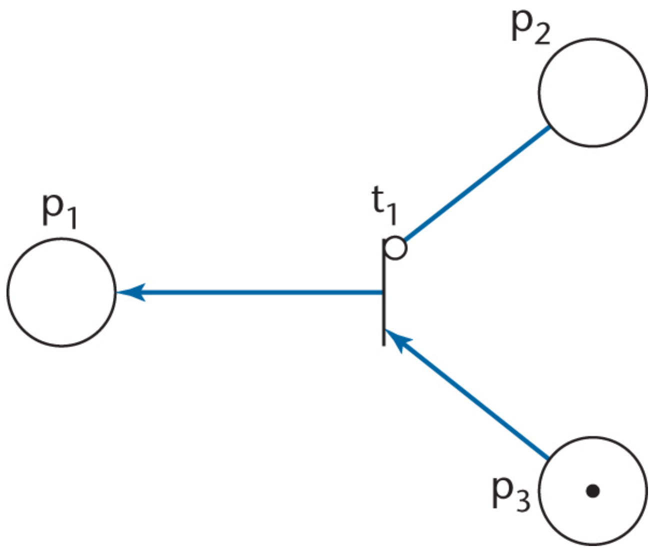


Figure 12.22

- Transition t_1 is enabled

Petri Nets (contd)

- In general, a transition is enabled if there is at least one token on each (normal) input arc, and no tokens on any inhibitor input arcs

Petri Nets: The Elevator Problem Case Study

- A product is to be installed to control n elevators in a building with m floors

- Each floor is represented by a place F_f, $1 \leq f \leq m$

- An elevator is represented by a token

- A token in F_f denotes that an elevator is at floor F_f

Petri Nets: The Elevator Problem Case Study (contd)

- First constraint:

  1. Each elevator has a set of m buttons, one for each floor. These illuminate when pressed and cause the elevator to visit the corresponding floor. The illumination is canceled when the corresponding floor is visited by an elevator

- The elevator button for floor f is represented by place EB_f, $1 \leq f \leq m$

- A token in EB_f denotes that the elevator button for floor f is illuminated

Petri Nets: The Elevator Problem Case Study (contd)

- A button must be illuminated the first time the button is pressed and subsequent button presses must be ignored

Petri Nets: The Elevator Problem Case Study (contd)

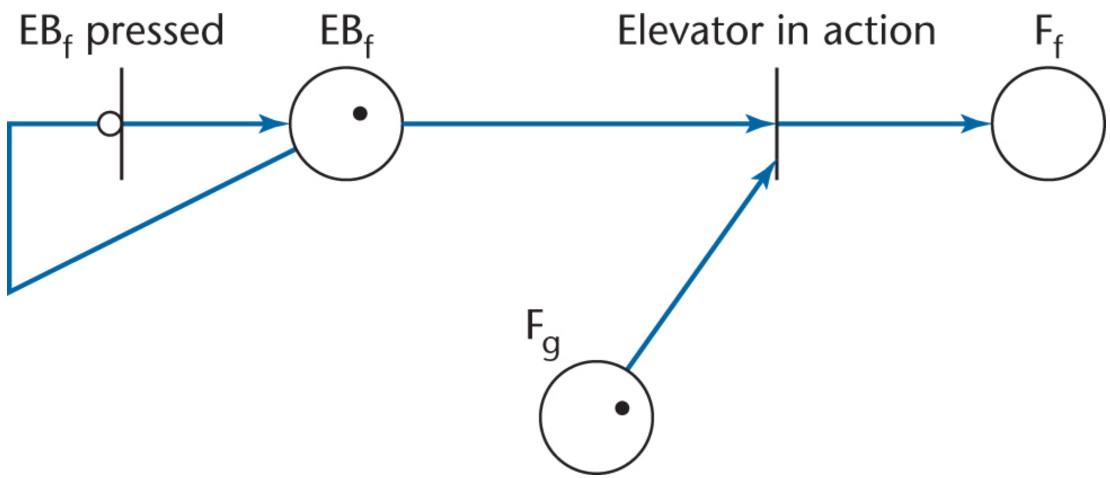


Figure 12.23

- If button EB_f is not illuminated, no token is in place and transition EB_f pressed is enabled
  - The transition fires, a new token is placed in EB_f
- Now, no matter how many times the button is pressed, transition EB_f pressed cannot be enabled

Petri Nets: The Elevator Problem Case Study (contd)

- When the elevator reaches floor g
  - A token is in place F_g
  - Transition Elevator in action is enabled, and then fires
- The tokens in EB_f and F_g are removed
  - This turns off the light in button EB_f
- A new token appears in F_f
  - This brings the elevator from floor g to floor f

Petri Nets: The Elevator Problem Case Study (contd)

- Motion from floor $_g$ to floor $_f$ cannot take place instantaneously
  - We need timed Petri nets

Petri Nets: The Elevator Problem Case Study (contd)

- Second constraint:

    2. Each floor, except the first and the top floor, has two buttons, one to request an up-elevator, one to request a down-elevator. These buttons illuminate when pressed. The illumination is canceled when the elevator visits the floor, and then moves in desired direction

- Floor buttons are represented by places FB^u_f and FB^d_f

Petri Nets: The Elevator Problem Case Study (contd)

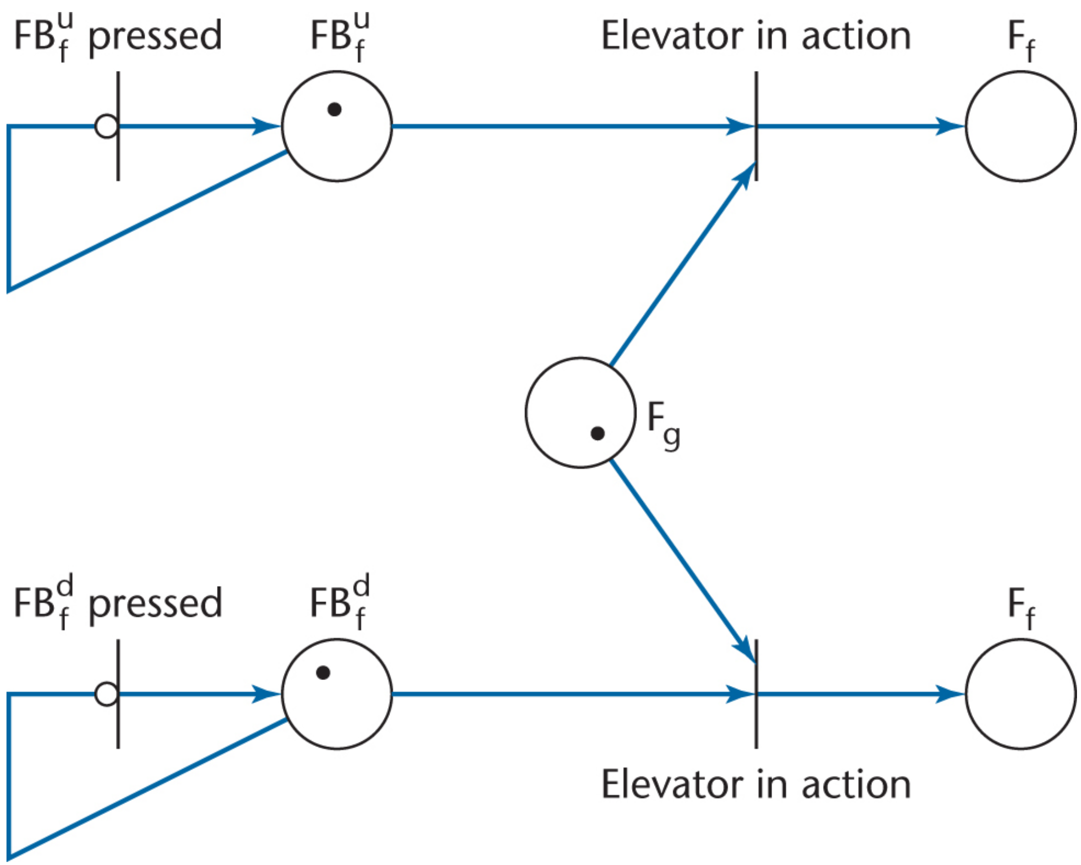


Figure 12.24

Petri Nets: The Elevator Problem Case Study (contd)

- The Petri net in the previous slide models the situation when an elevator reaches floor f from floor g with one or both buttons illuminated

- If both buttons are illuminated, only one is turned off

- A more complex model is needed to ensure that the correct light is turned off

Petri Nets: The Elevator Problem Case Study (contd)

- Third constraint:

  C_3. If an elevator has no requests, it remains at its current floor with its doors closed

- If there are no requests, no Elevator in action transition is enabled

Petri Nets: The Elevator Problem Case Study (contd)

- Petri nets can also be used for design

- Petri nets possess the expressive power necessary for specifying synchronization aspects of real-time systems