U O W

Software Requirements, Specifications and Formal Methods

Dr. Shixun Huang

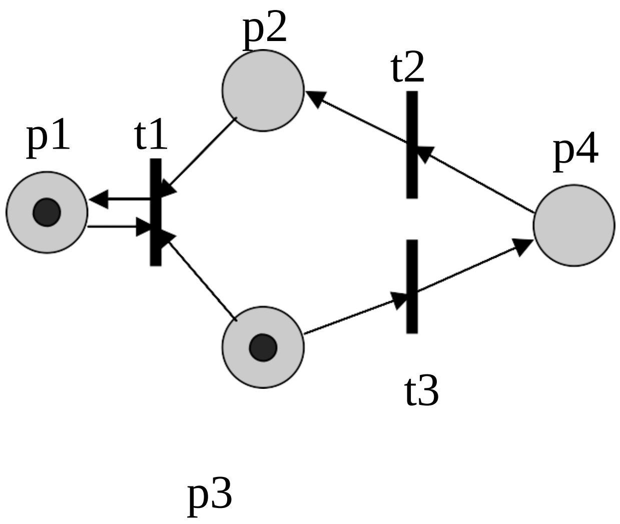
UNIVERSITY
OF WOLLONGONG
AUSTRALIA

Petri net analysis

- Petri net reachability
- Petri net liveness
- Petri net soundness
- Petri net safeness
- Petri net conservation
- Petri net conflict
- Structural Analysis: P-invariants and T-invariants
- Petri Nets Modelling

Reachability

Marking M is **reachable** from marking M_0 if there exists a sequence of firings $\sigma = t_1\ t_2\ \ldots$ (i.e., $M_0\ t_1\ M_1\ t_2\ M_2 \ldots\ M$) that transforms M_0 to M.

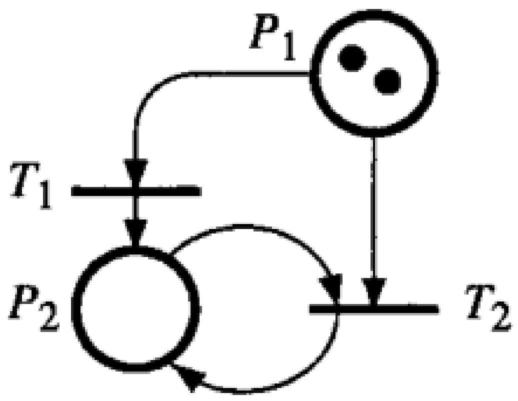


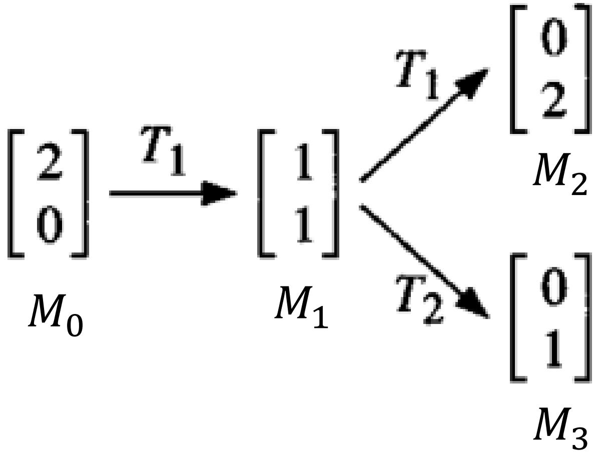
$$\begin{tikzcd}
M_0 = (1,0,1,0) \arrow[d, "t3"] \\
M_1 = (1,0,0,1) \arrow[d, "t2"] \\
M = (1,1,0,0)
\end{tikzcd}$$

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

Reachability graph

Reachability graph is made up of vertices which correspond to reachable markings and of arcs corresponding to firing of transitions resulting in the passing from one marking to another.

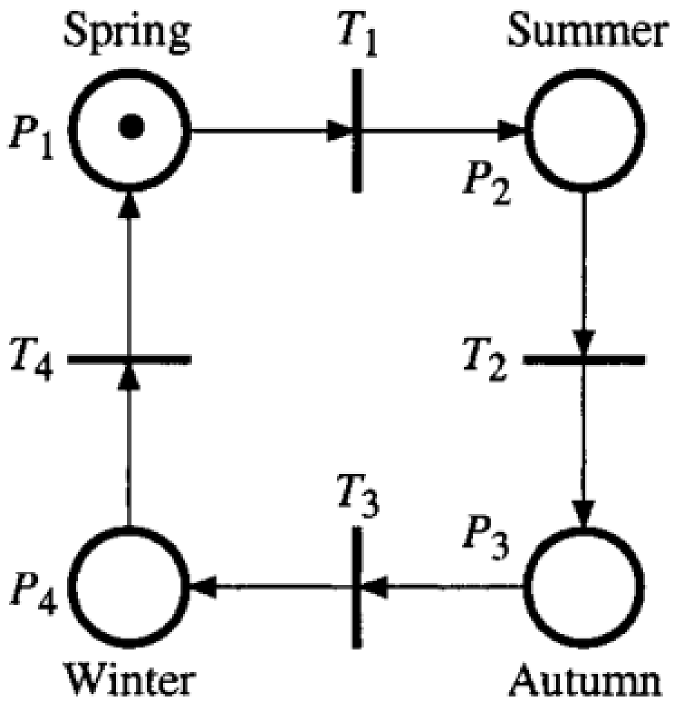



$\begin{bmatrix}2\\0\end{bmatrix}$
M_0
T_1
$\begin{bmatrix}1\\1\end{bmatrix}$
M_1
T_1
$\begin{bmatrix}0\\2\end{bmatrix}$
M_2
T_2
$\begin{bmatrix}0\\1\end{bmatrix}$
M_3

Firing sequence: T_1T_2

$M_0 \xrightarrow{T_1T_2} M_3$

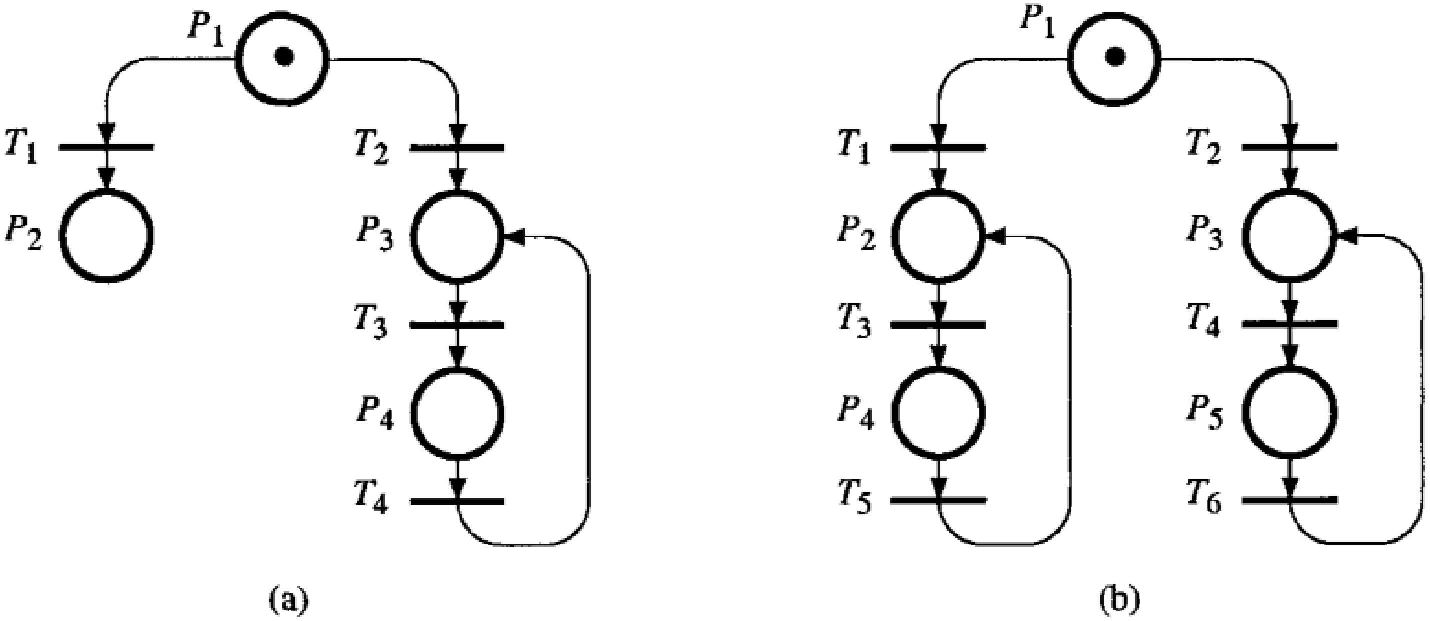
Reachability graph (cont.)



- Possible firing sequences from the initial marking M_0: T_1, T_1T_2, T_1 T_2T_3, $T_1T_2T_3T_4$.
- Firing sequence $T_1T_2T_3T_4$ makes $M_0 \xrightarrow{T_1T_2T_3T_4} M_0$
- The sequence causes a return to the initial state, and this is a **repetitive sequence**.
- A repetitive sequence which contains all the transitions (each at least once) is a **complete repetitive sequence**.

Deadlock

- A deadlock (or sink state) is a marking such that no transition is enable.
- A Petri net is deadlock-free for an initial marking M_0 if no reachable marking M_i is deadlock.



(a) With deadlock (b) Deadlock free

Do we have deadlocks in these two petri nets?

Liveness

- Liveness: whether transitions can eventually fire with different levels.
- Closely related to the complete absence of deadlocks in operating systems.
- Liveness implies deadlock freedom, not vice versa
- A **live** Petri net guarantees **deadlock-free** operation
- Different levels of liveness are defined.

Liveness (cont.)

A transition *t* in a Petri net (N, M_0) is said to be:

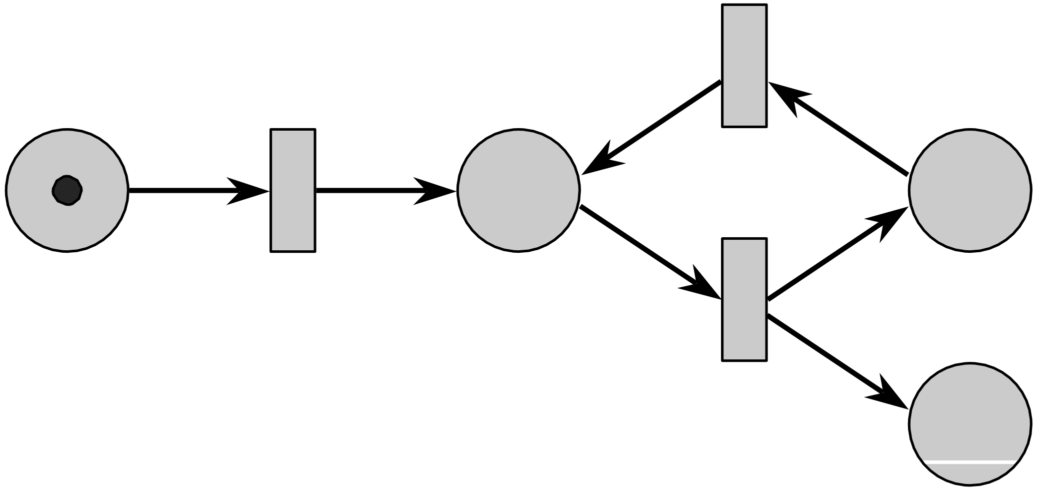
- **Dead (L0-live)** if *t* can never be fired in any firing sequence in $L(M_0)$
- **L1-live** if *t* can be fired at least once in some firing sequence $L(M_0)$
- **L2-live** if *given* any positive integer *k*, t can be fired at least *k* times in some firing sequence in $L(M_0)$
- **L3-live** if there is an infinite firing sequence in $L(M_0)$ where tj occurs infinitely often.
- **L4-live** or **live** if *t* is immediately fireable from any reachable marking from M_0(strongest condition).

Boundedness

- A **place $\boldsymbol{P_i}$ is said to be bounded** for an initial marking M_0 if there is a natural integer *k* such that, for all markings reachable from M_0, the number of tokens in P_i is not greater than k (P_i is said to be *k*-bounded).
- A **Petri net is bounded** for an initial marking M_0 if all the places are bounded for M_0 (the Petri net is *k*-bounded if all the places are *k*-bounded).

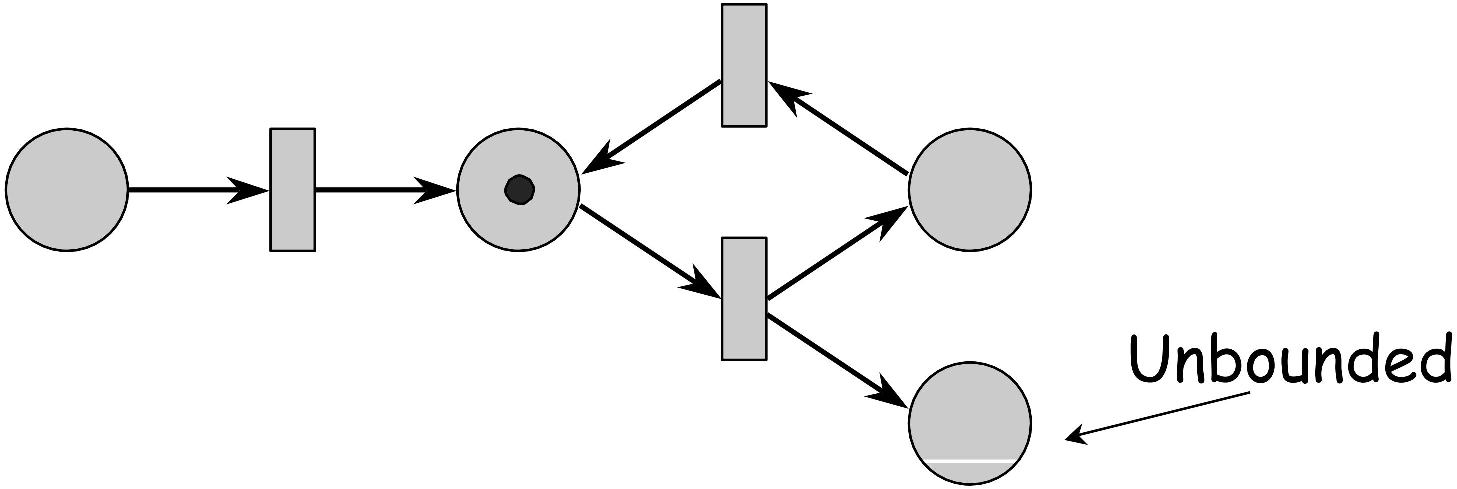
Boundedness (cont.)

Bounded or not?



Boundedness (cont.)

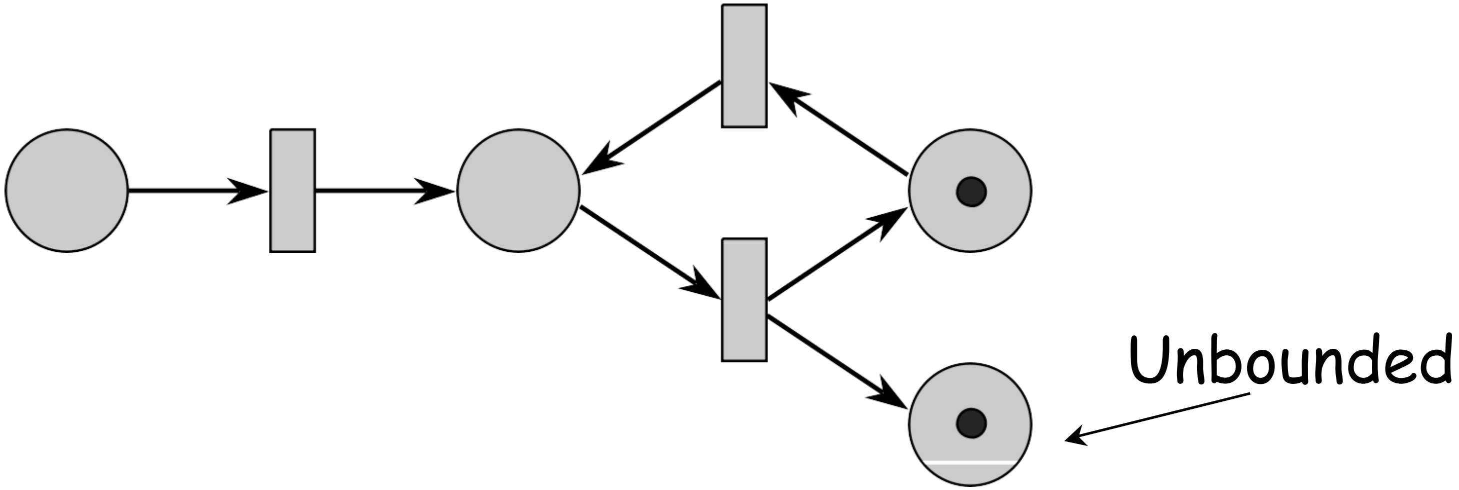
Boundedness: the number of tokens in any place cannot grow indefinitely



Unbounded

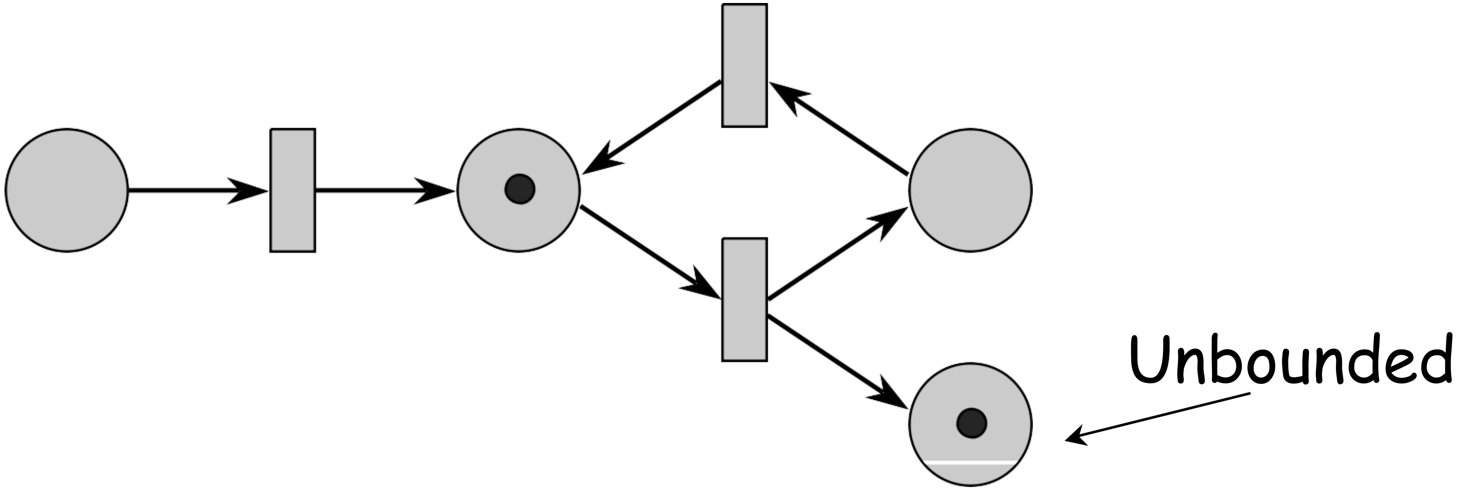
Boundedness (cont.)

Boundedness: the number of tokens in any place cannot grow indefinitely



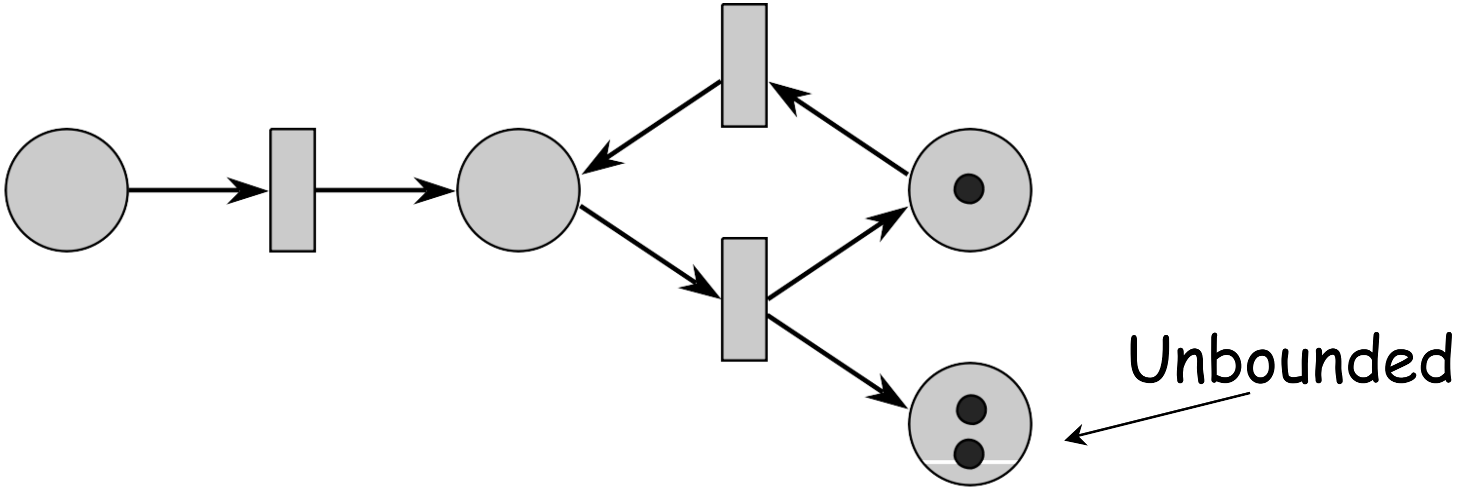
Boundedness (cont.)

Boundedness: the number of tokens in any place cannot grow indefinitely



Boundedness (cont.)

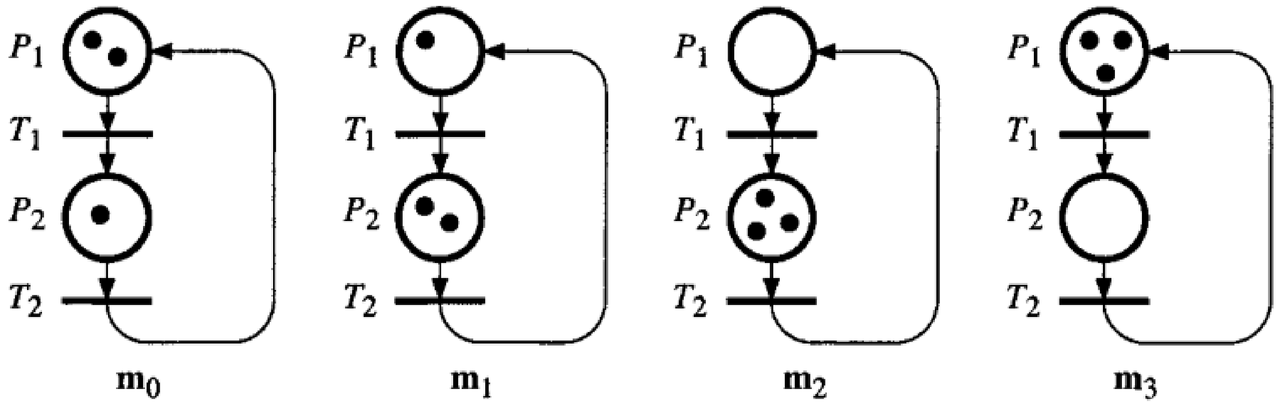
Boundedness: the number of tokens in any place cannot grow indefinitely



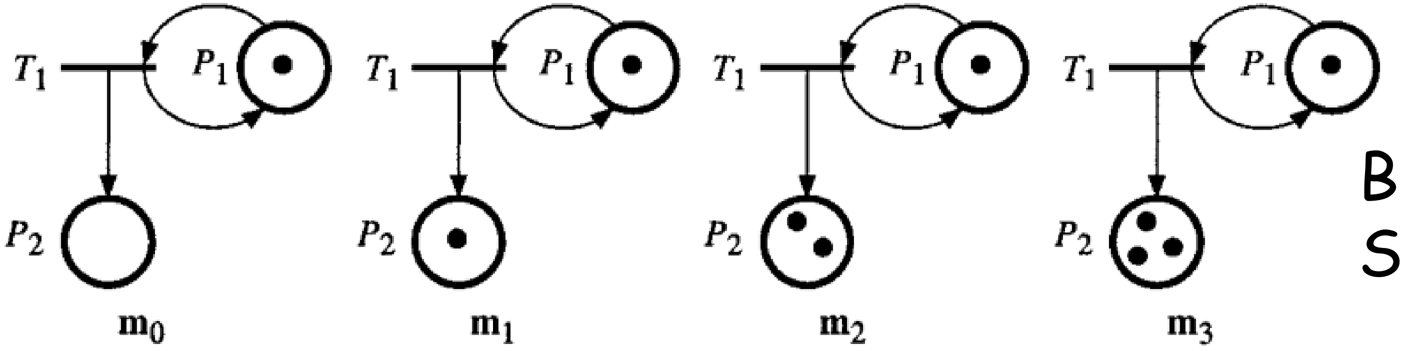
Unbounded

Safeness

- A Petri net is said to be safe for an initial marking M_0 if for all reachable markings, each place contains **zero** or **one** token.
- A **safe Petri net** is a particular case of bounded Petri net for which all the places are **1-bounded**.

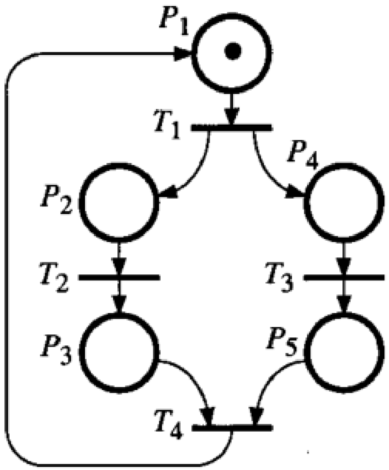


Bounded or not?
Safe or not?

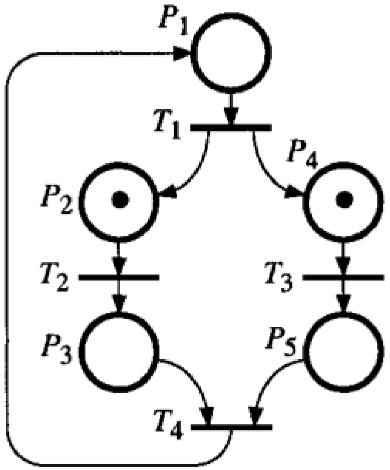


Bounded or not?
Safe or not?

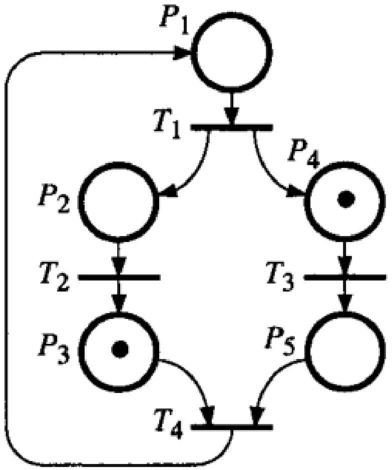
Safeness (cont.)



$\mathbf{m}_0$

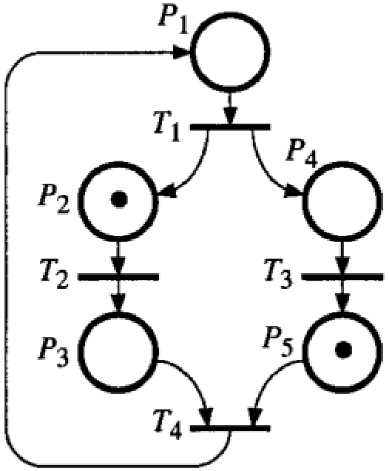


$\mathbf{m}_1$

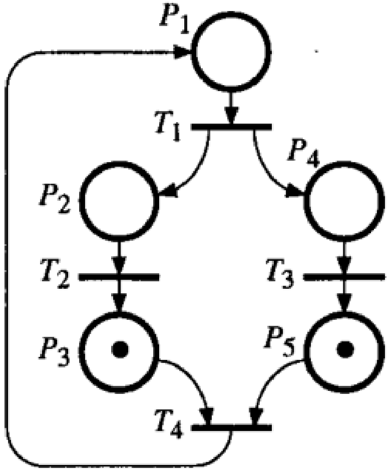


$\mathbf{m}_2$

Safe or not?



$\mathbf{m}_3$



$\mathbf{m}_4$

Petri Net Analysis

Behavioural properties

1. Reachability
2. Liveness (Deadlock-free)
3. Boundedness
4. Safeness

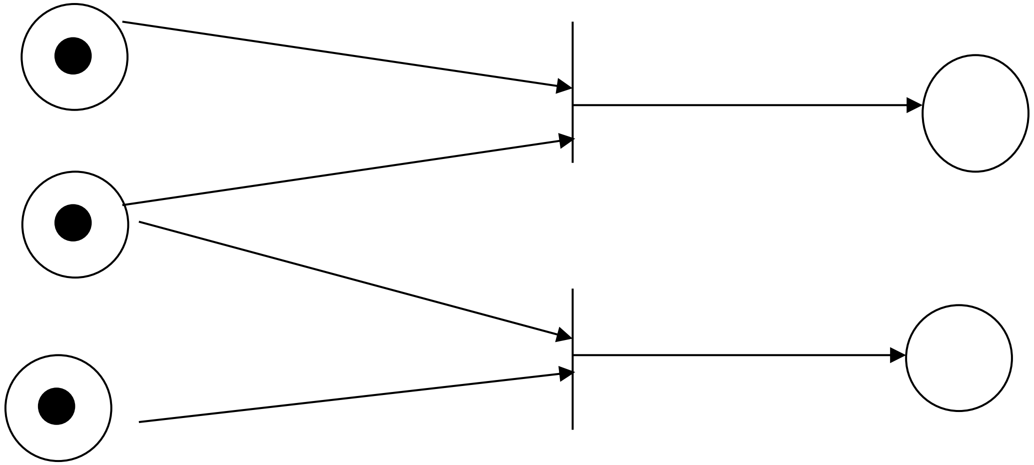
- Why do we need to analyse Petri net properties (e.g., reachability, deadlock-free, boundedness)?
- What do these properties mean to a system?

Conservation

- Petri nets can be used to model *resource allocation systems*. For example, a Petri net can model the requests, allocations, and releases of input/output devices in a computer system. In these systems some tokens may represent the resources.
- For these systems, *conservation* is an important property. We would like to show that tokens which represent resources are created nor destroyed.
- *Conservative Petri nets* are that the *total number of tokens in the net remain constant*.

Conflict

In the following Petri net, the two enabled transitions are conflict. Only one transition can fire, since, in firing, it removes the token in the sharing input and disables the other transition.



Good or bad?

Petri Net properties (further comments)

1. **Reachability**
   - Verify whether the system could reach some state from some initial state
   - Whether the system can be executed as planned
2. **Deadlock**
   - Resource-sharing environment
   - Improper resource distribution/allocation in the system
   - Resources are run out of
3. **Boundedness**
   - Checking overflow

Petri Nets: Structural Analysis

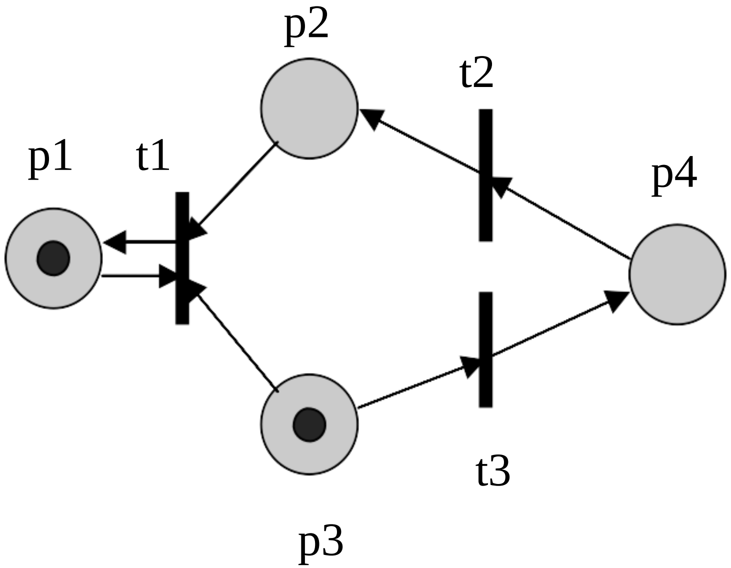
A class of techniques that can extract information about the behaviour of the system

- Describe and analyse the dynamic behaviour of concurrent systems modelled by Petri nets
- **Place invariants** and **Transition invariants**, i.e., through matrix equations

Place Invariant

Reachability Problem :

Marking M is **reachable** from marking M_0 if there exists a sequence of firings $\sigma = t_1\ t_2 \ldots t_{m-1}$ that transforms M_0 to M.



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

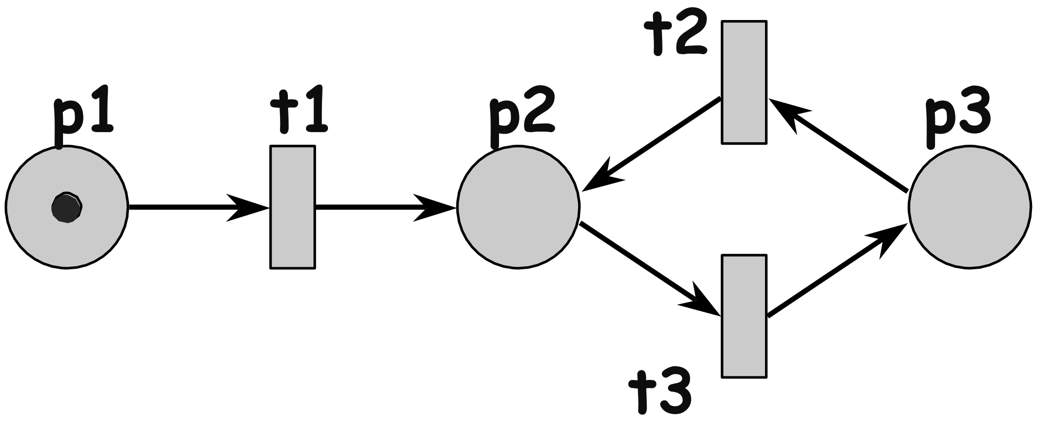
$M = (1,1,0,0)$

$$\begin{tikzcd}
M_0 = (1,0,1,0) \arrow[d, "t3"] \\
M_1 = (1,0,0,1) \arrow[d, "t2"] \\
M = (1,1,0,0)
\end{tikzcd}$$

Place Invariant: Opening question

- How to solve the following question?

Verifying Reachability of $M = (0\ \ 0\ \ 1)^T$ from $M_0 = (1\ \ 0\ \ 0)^T$



Incidence Matrix (weighted petri nets)

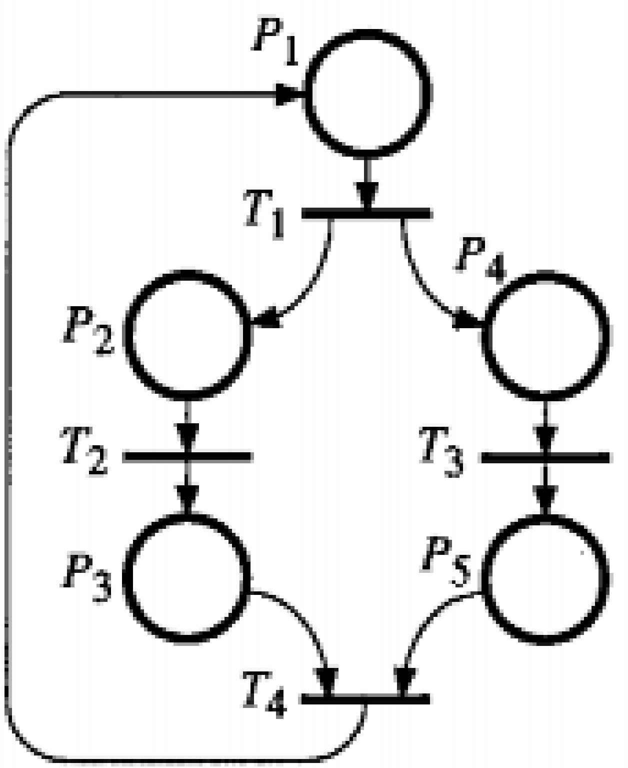
- Input incidence application: Pre: $P \times T \rightarrow \{0,1,2, \dots\}$
  - $Pre(P_i, T_j)$ is the weight of the arc $P_i \rightarrow T_j$
- Output incidence application: Post: $P \times T \rightarrow \{0,1,2, \dots\}$
  - $Post(P_i, T_j)$ is the weight of the arc $T_j \rightarrow P_i$
- Some relevant notations:
  - $°T_j = \{P_i \in P | Pre(P_i, T_j) > 0\}$: set of input places of T_j;
  - $T_j° = \{P_i \in P | Post(P_i, T_j) > 0\}$: set of output places of T_j;
  - $°P_i = \{T_j \in T \middle| Post(P_i, T_j) > 0\}$: set of input transitions of P_i;
  - $P_i° = \{T_j \in T \middle| Pre(P_i, T_j) > 0\}$: set of output transitions of P_i;

Incidence Matrix

- Transition T_j is enable for a marking M_k if $M_k(P_i) \geq Pre(P_i, T_j)$ for every $P_i \in {}^{\circ}T_j$.
- Input incidence matrix:
  - $W^- = [w_{ij}^-]$, where $w_{ij}^- = Pre(P_i, T_j)$
- Output incidence matrix:
  - $W^+ = [w_{ij}^+]$, where $w_{ij}^+ = Post(P_i, T_j)$
- Incidence matrix:
  - $W = W^+ - W^- = [w_{ij}]$

Exercise 3

Please calculate the **input incidence matrix**, **output incidence matrix** and the **incidence matrix** of the following Petri net.



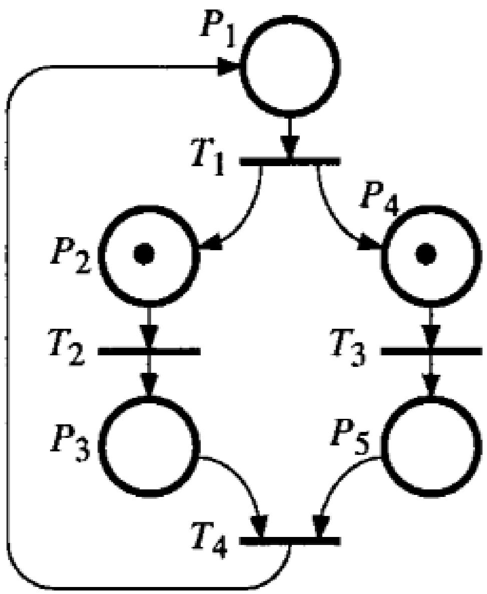
$$W = \begin{array}{c|cccc|c} & \textbf{T1} & \textbf{T2} & \textbf{T3} & \textbf{T4} & \\ \hline & -1 & 0 & 0 & 1 & \textbf{P1} \\ & 1 & -1 & 0 & 0 & \textbf{P2} \\ & 0 & 1 & 0 & -1 & \textbf{P3} \\ & 1 & 0 & -1 & 0 & \textbf{P4} \\ & 0 & 0 & 1 & -1 & \textbf{P5} \end{array}$$

Fundamental equation

- The **characteristic vector** of sequence S, written as ***s***, is the *m*-component vector whose component number *j* corresponds to the number of firings of transition T_j in sequence S, e.g., $\boldsymbol{s_1} = (0, 1, 0, 0)$.
- If the firing sequence S is such that $M_i \overset{S}{\rightarrow} M_k$, then a fundamental equation is obtained:

$$M_k = M_i + W \cdot \boldsymbol{s}$$

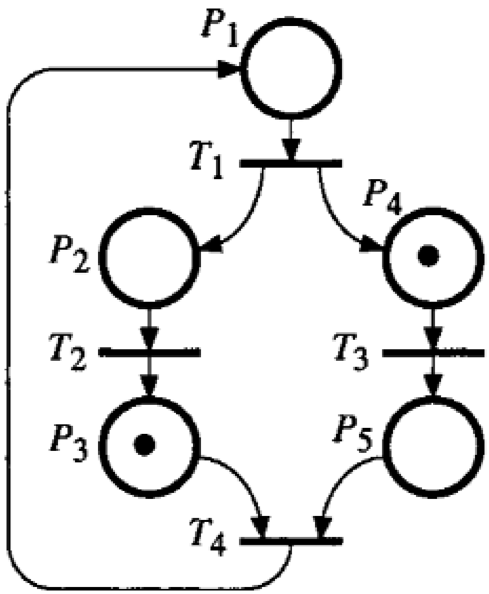
Fundamental equation (cont.)



M_1

example

$$\boldsymbol{s_1} = (0, 1, 0, 0)$$



M_2

$$\underset{M_1}{\begin{bmatrix}0\\1\\0\\1\\0\end{bmatrix}} + \underset{\mathbf{W}}{\begin{bmatrix}-1 & 0 & 0 & +1\\+1 & -1 & 0 & 0\\0 & +1 & 0 & -1\\+1 & 0 & -1 & 0\\0 & 0 & +1 & -1\end{bmatrix}} \cdot \underset{\mathbf{s}_1}{\begin{bmatrix}0\\1\\0\\0\end{bmatrix}} = \begin{bmatrix}0\\1\\0\\1\\0\end{bmatrix} + \begin{bmatrix}0\\-1\\+1\\0\\0\end{bmatrix} = \underset{M_2}{\begin{bmatrix}0\\0\\1\\1\\0\end{bmatrix}}.$$

Fundamental equation (cont.)

Another example:

- We have $M_2 \xrightarrow{T_3T_4T_1T_3}$ and for $S_2 = T_3T_4T_1T_3$, we have $\boldsymbol{s_2} = (1, 0, 2, 1)$
- The carrying out of this firing sequence gives the marking M_3 obtained by:

$$\underset{\mathbf{m}_2}{\begin{bmatrix}0\\0\\1\\1\\0\end{bmatrix}} + \underset{\mathbf{W}}{\begin{bmatrix}-1 & 0 & 0 & +1\\+1 & -1 & 0 & 0\\0 & +1 & 0 & -1\\+1 & 0 & -1 & 0\\0 & 0 & +1 & -1\end{bmatrix}} \cdot \underset{\mathbf{s}_2}{\begin{bmatrix}1\\0\\2\\1\end{bmatrix}} = \begin{bmatrix}0\\0\\1\\1\\0\end{bmatrix} + \begin{bmatrix}0\\+1\\-1\\-1\\+1\end{bmatrix} = \underset{\mathbf{m}_3}{\begin{bmatrix}0\\1\\0\\0\\1\end{bmatrix}}.$$

Fundamental equation (cont.)

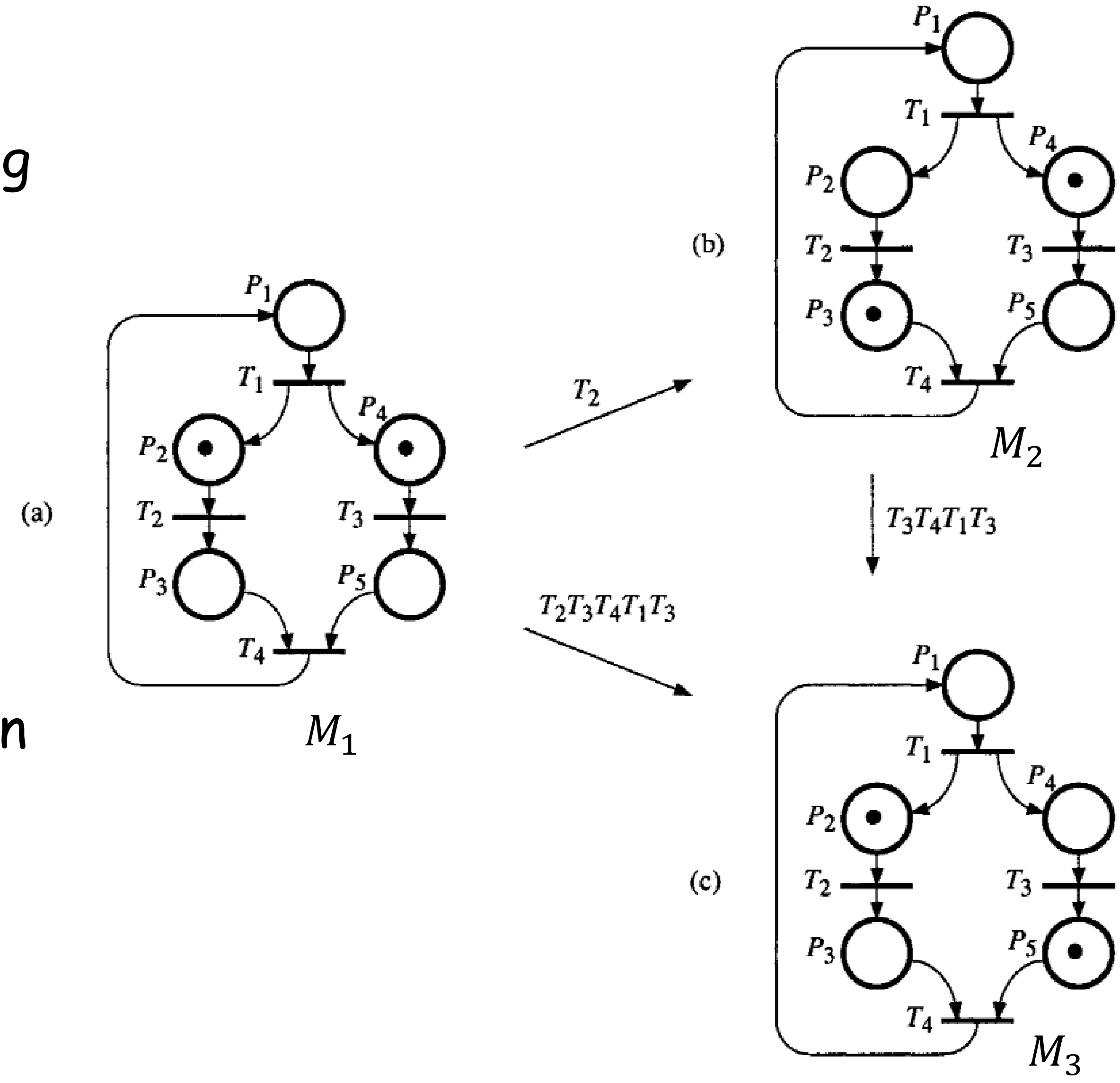
- If we consider the carrying out of the sequence $S_1 \cdot S_2$ from the marking M_1, we have

$$S_1 \cdot S_2 = T_2 T_3 T_4 T_1 T_3$$

and $\boldsymbol{s_1} + \boldsymbol{s_2} = (1, 1, 2, 1).$

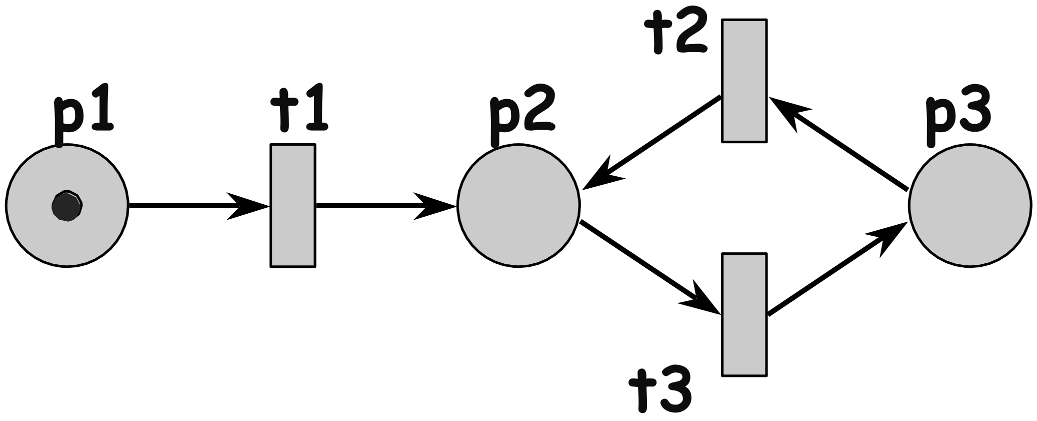
Then, we have the equation

$$M_1 + W \cdot (\boldsymbol{s_1} + \boldsymbol{s_2}) = M_3$$



Incidence Matrix Example

Opening question:
reachability of $M = (0\ \ 0\ \ 1)^T$ from $M_0 = (1\ \ 0\ \ 0)^T$

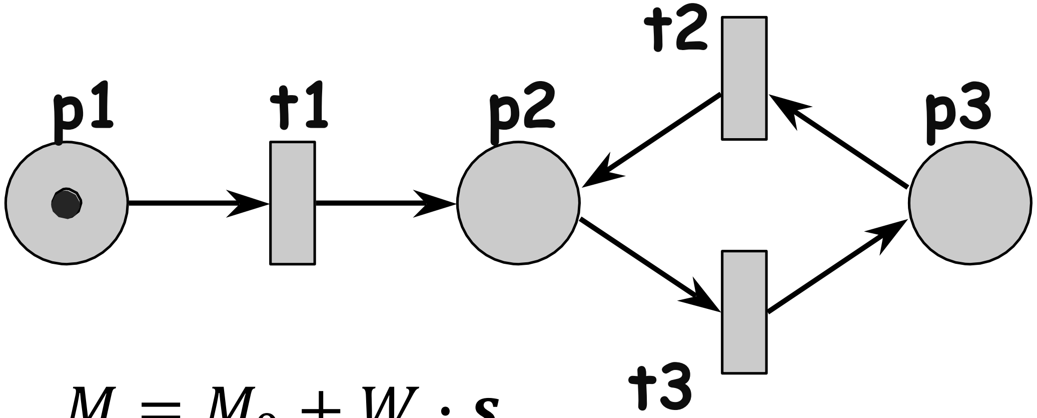


incidence matrix:

$$W = \begin{array}{cccc} t1 & t2 & t3 & \\ \left[\begin{array}{c} -1 \\ 1 \\ 0 \end{array}\right. & \begin{array}{c} 0 \\ 1 \\ -1 \end{array} & \left.\begin{array}{c} 0 \\ -1 \\ 1 \end{array}\right] & \begin{array}{c} p1 \\ p2 \\ p3 \end{array} \end{array}$$

Incidence Matrix Example (cont.)

Reachability of M = (0 0 1)^T from M$_0$ = (1 0 0)^T



$$\begin{array}{cc} & \begin{array}{ccc} t1 & t2 & t3 \end{array} \\ W = & \begin{bmatrix} -1 & 0 & 0 \\ 1 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{array}{c} p1 \\ p2 \\ p3 \end{array} \end{array}$$

$$M = M_0 + W \cdot \boldsymbol{s}$$

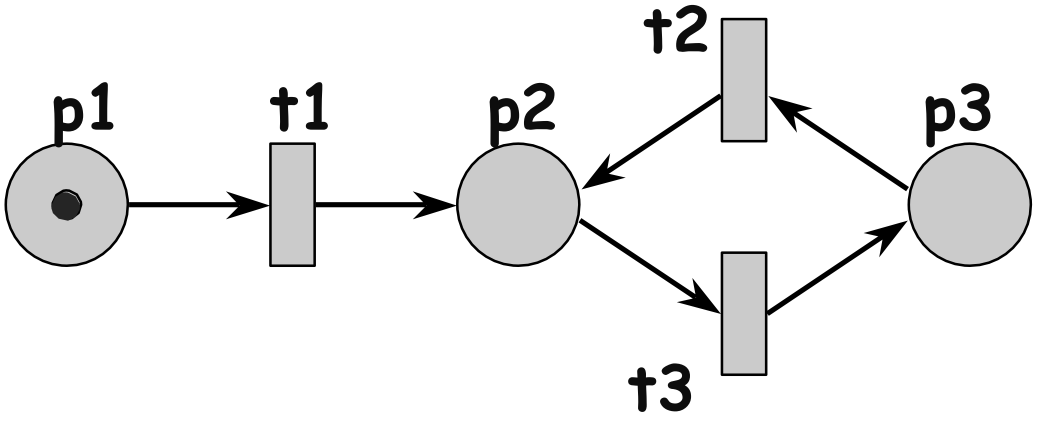
$$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 & 0 & 0 \\ 1 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix} \cdot \boldsymbol{s}$$

$$\boldsymbol{s_k} = [1 \quad k-1 \quad k]^T$$

$$\begin{array}{lccc} & t1 & t2 & t3 \\ \boldsymbol{s_1} = & [1 & 0 & 1]^T \\ \boldsymbol{s_2} = & [1 & 1 & 2]^T \\ & & \vdots & \\ \boldsymbol{s_k} = & [1 & k-1 & k]^T \end{array}$$

Incidence Matrix Example (cont.)

Reachability of $M = (0\ \ 0\ \ 1)^T$ from $M_0 = (1\ \ 0\ \ 0)^T$



$$W = \begin{array}{c} \\ \\ \\ \\ \end{array}\begin{array}{ccc|c} t1 & t2 & t3 & \\ -1 & 0 & 0 & p1 \\ 1 & 1 & -1 & p2 \\ 0 & -1 & 1 & p3 \end{array}$$

$$M = M_0 + W \cdot \boldsymbol{s}$$

$$\boldsymbol{s_k} = [1 \quad k-1 \quad k]^T$$

P-invariants

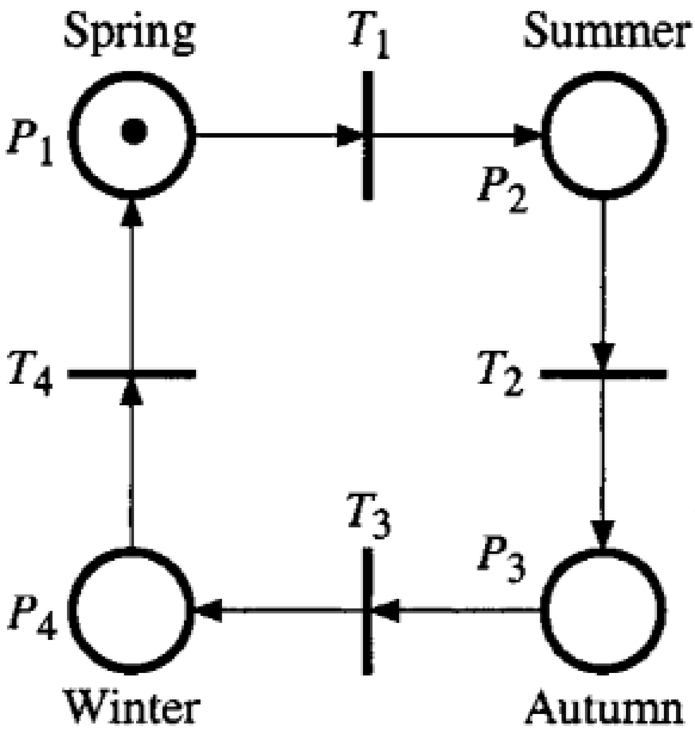
- Let *N* be a net and W be its incidence matrix.
  - A natural solution of the equation $\boldsymbol{x}^T \cdot W = 0$ such that $\boldsymbol{x} \geq \mathbf{0}$ is called a **place invariant** (or **P-invariant**) of N.
  - A P-invariant indicates that the number of tokens in all reachable markings ($\boldsymbol{x}^T$) satisfies some linear invariant.

- Let M be marking reachable with a transition sequence whose firing count is expressed by $\boldsymbol{s}$, i.e., $M = M_0 + W \cdot \boldsymbol{s}$.
  - Let $\boldsymbol{x}$ be a P-invariant. Then, the following holds:

$$\boldsymbol{x}^T \cdot M = \boldsymbol{x}^T \cdot (M_0 + W \cdot \boldsymbol{s}) = \boldsymbol{x}^T \cdot M_0 + \boldsymbol{x}^T \cdot W \cdot \boldsymbol{s} = \boldsymbol{x}^T \cdot M_0$$

P-invariants (cont.)

Example



- At all times, there will always be one and only one token for all places.

$$m_1 + m_2 + m_3 + m_4 = 1$$

⬇

At all times, we are at one and only one season.

How to prove?

P-invariants (cont.)

- Let $\boldsymbol{x}^T = (x_1, x_2, x_3, x_4)$

$$\boldsymbol{x}^T \cdot W = (x_1, x_2, x_3, x_4)\begin{pmatrix} -1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix} = 0 \Rightarrow \boldsymbol{x}^T = (1, 1, 1, 1)$$

- With the initial marking $m_0 = (1, 0, 0, 0)$,
- The constant of marking invariant is

$$\boldsymbol{x}^T \cdot M_0 = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = 1$$

The marking invariant is:

$$m_1 + m_2 + m_3 + m_4 = 1$$

$$\boldsymbol{x}^T \cdot M_k = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \end{bmatrix} = m_1 + m_2 + m_3 + m_4 = \boldsymbol{x}^T \cdot M_0 + \boldsymbol{x}^T \cdot W \cdot \boldsymbol{s} = \boldsymbol{x}^T \cdot M_0 = 1$$

T-invariants

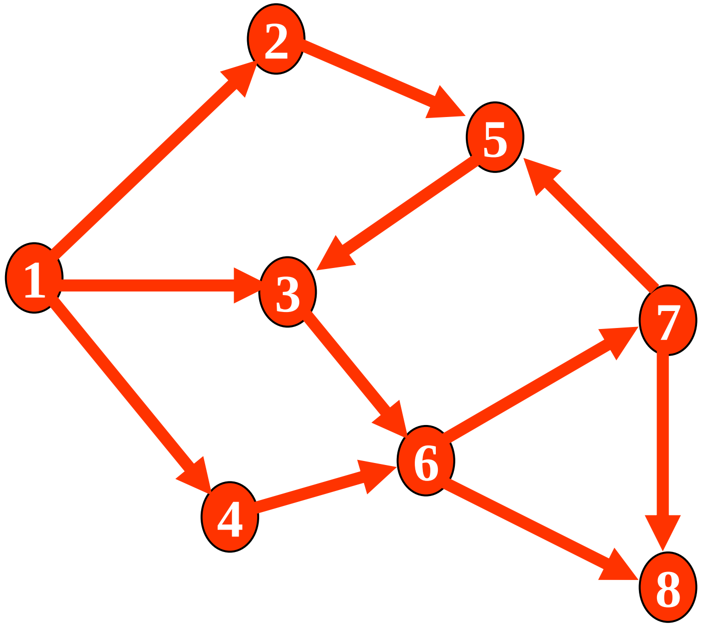
- Let N be a net and W be its incidence matrix.
  - A natural solution of the equation $\boldsymbol{W} \cdot \boldsymbol{y} = \boldsymbol{0}\ (\boldsymbol{y} \neq \boldsymbol{0})$ is known as a transition invariant (or **T-invariant**) of N.

State space method

- Verification of PN models and system properties is supported by the state space method.

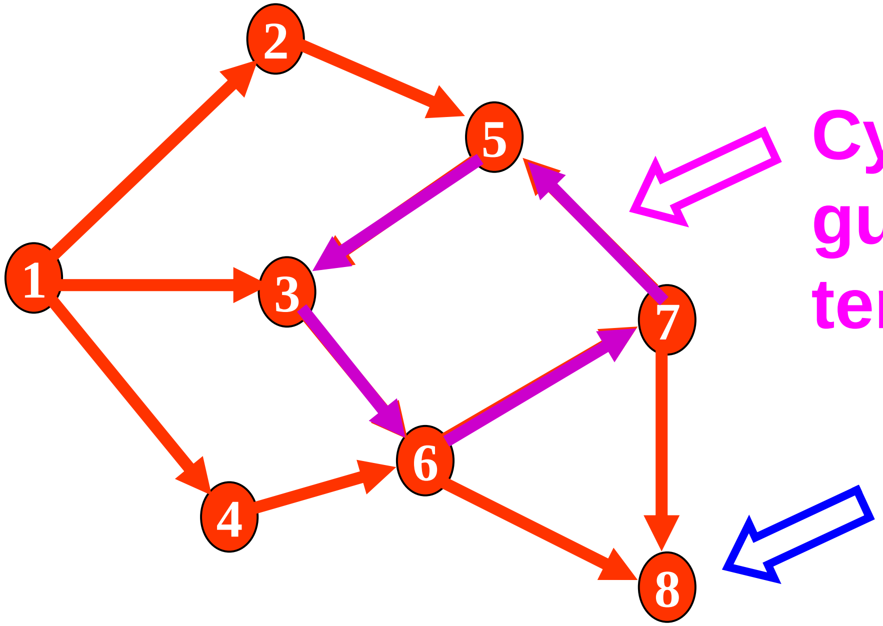
- The basic idea of state spaces is to compute all reachable states and state changes of the PN model and represent these as a directed graph, where:
  - nodes represent states,
  - arcs represent occurring events.

- State spaces can be constructed fully automatically.



Behavioural questions

- From a state space it is possible to answer a large set of questions concerning the behaviour of the system such as:
  - Are there any deadlocks?
  - Is it always possible to reach a specified state?
  - Is the system guaranteed to provide a given service?



Cycle (no guarantee for termination)

Deadlock

State spaces – pros

- State spaces are relatively easy to use, and they have a high degree of automation.
  - It is possible to hide a large portion of the underlying mathematics from the user.
  - Often the user only needs to formulate the property which is to be verified and then apply a computer tool.
- State spaces can provide counterexamples (error-traces) giving detailed debugging information specifying why an expected property does not hold.

State spaces – cons

- The main disadvantage of state spaces is the state explosion problem.
- Even relatively small systems may have an astronomical or even infinite number of reachable states.

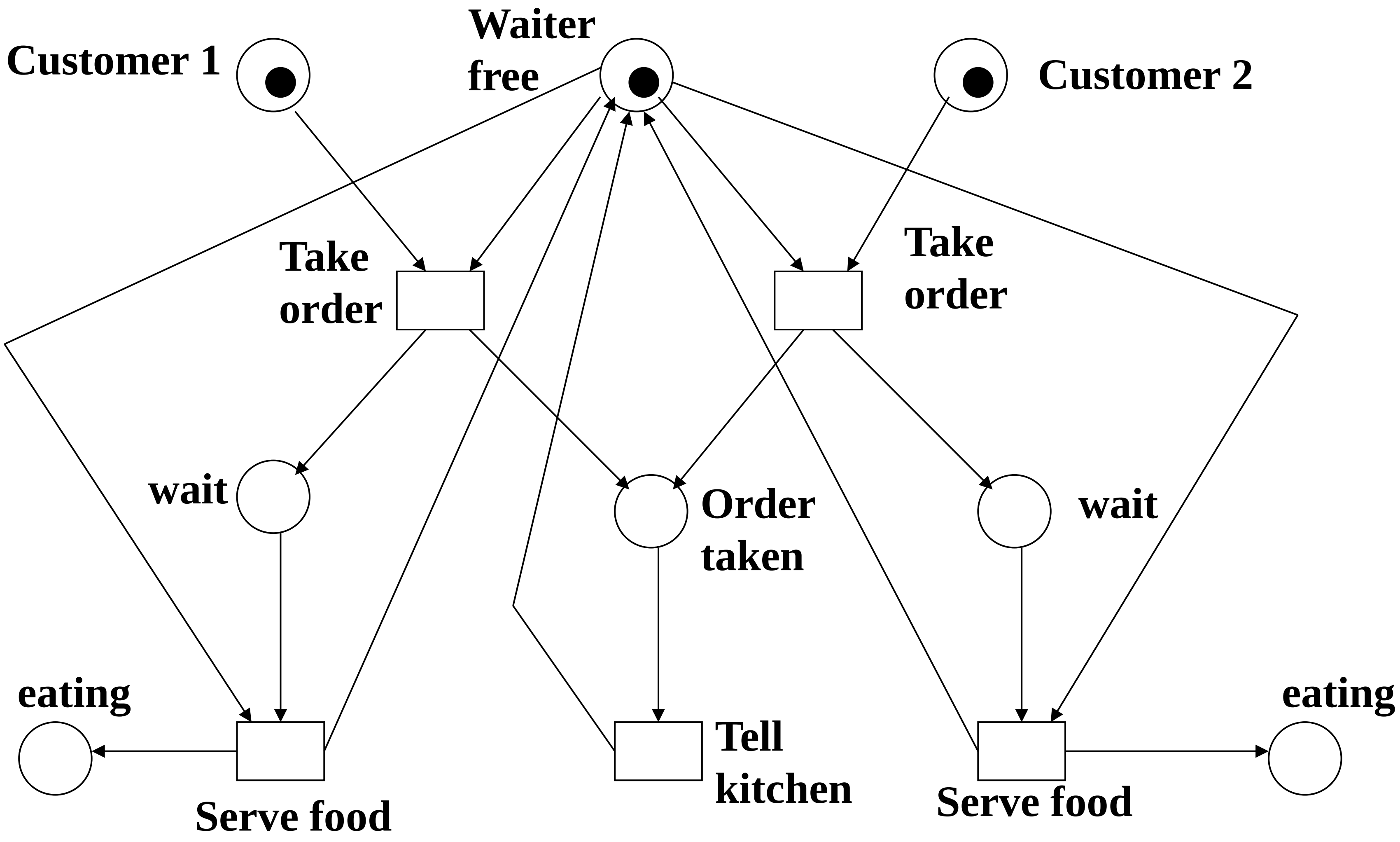
- A wide range of state space reduction methods have been developed to alleviate the state explosion problem.

Modelling with Petri Nets

Petri nets were designed for and are used mainly for *modelling*. Many systems, especially those with independent components, can be modelled by a Petri net.

The simple Petri net view of a system concentrates on two primitive concepts: *events (transitions)* and *conditions (places)*. Events are actions which take place in the system. The occurrence of these events is controlled by the state of the system. The state of the system can be described as a set of conditions. *A* condition is a predicate or logical description of the state of the system.

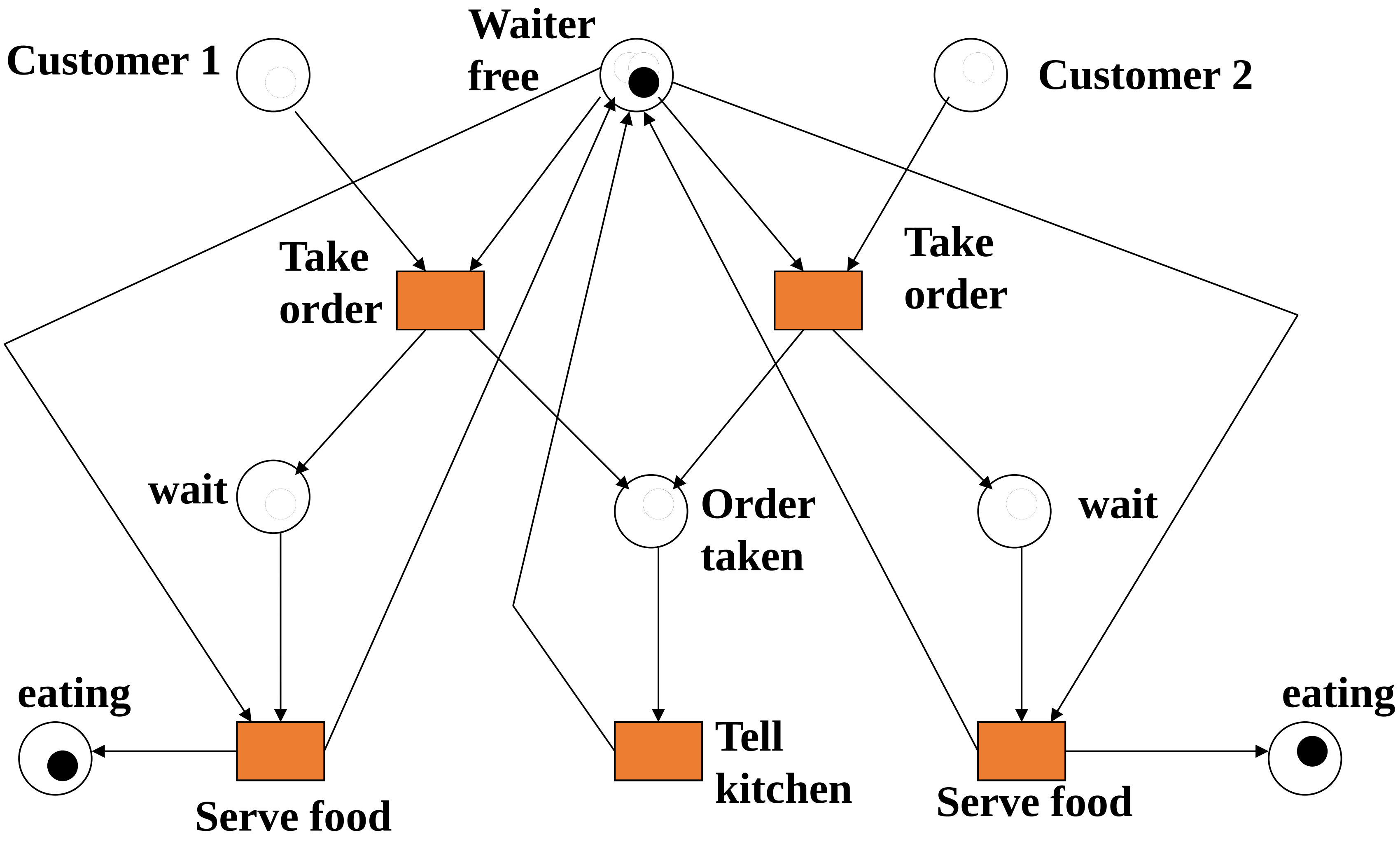
Example: In a Restaurant (A Petri Net)



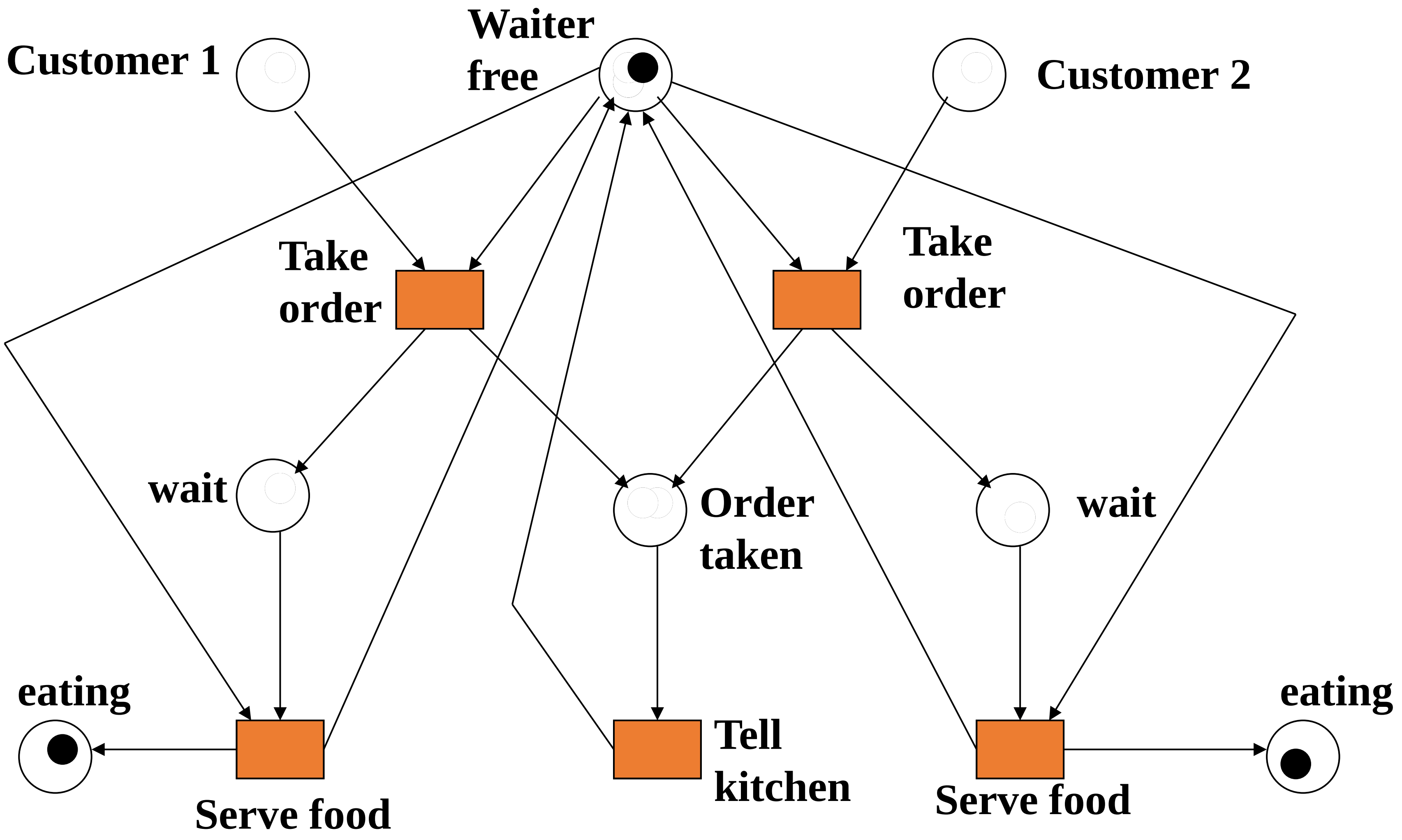
Example: In a Restaurant
(Two Scenarios)

- 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.

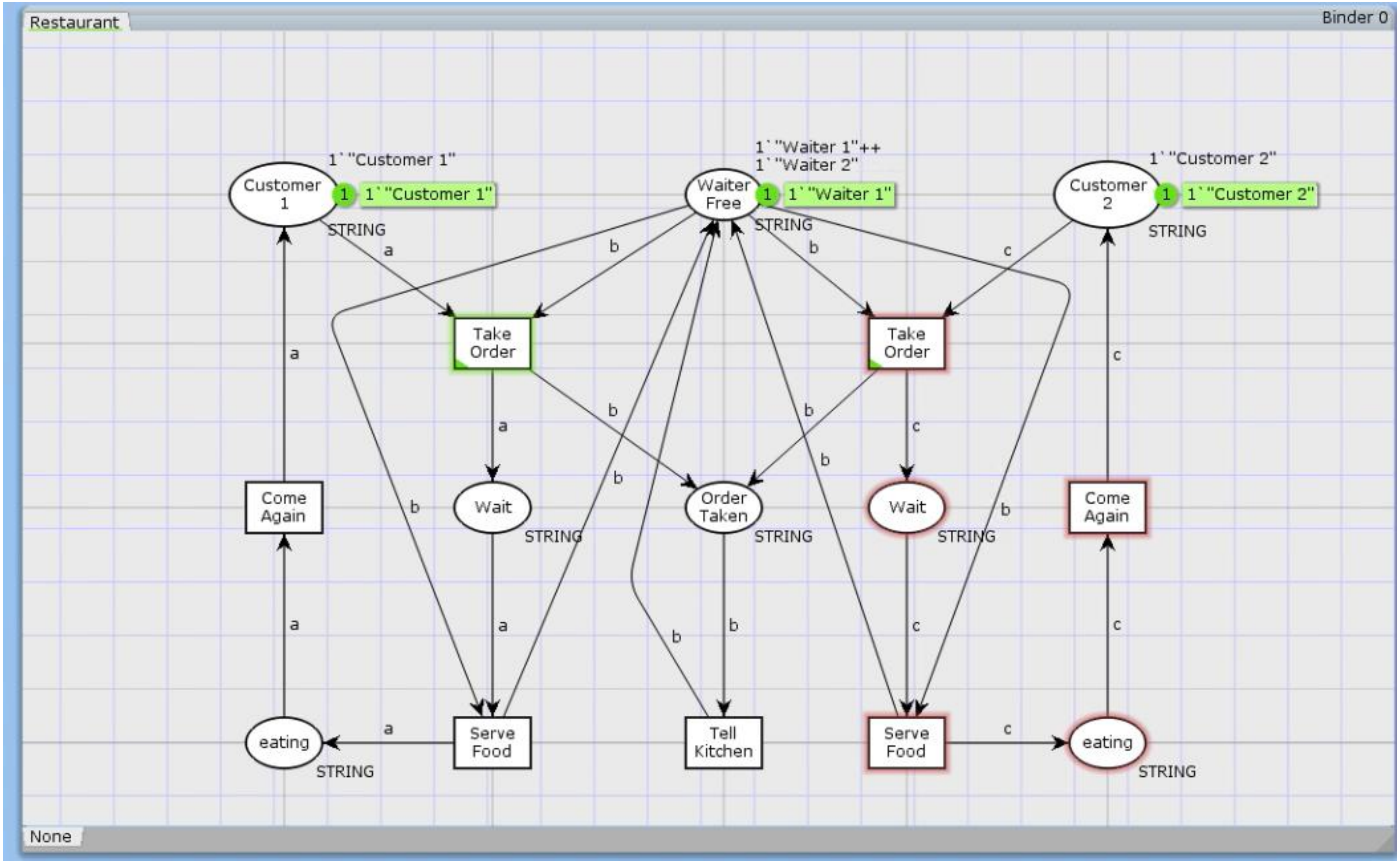
Example: In a Restaurant (Scenario 1)



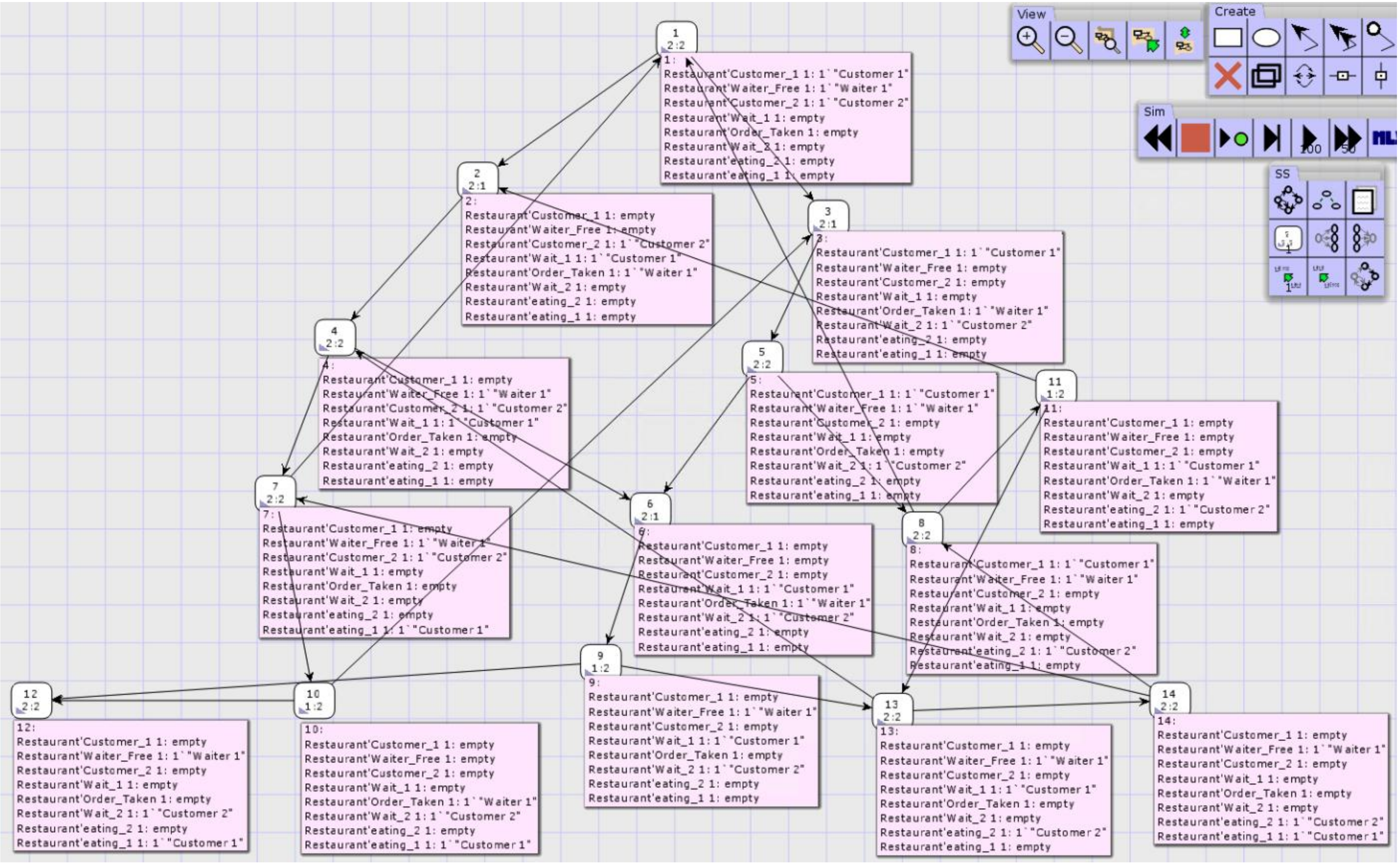
Example: In a Restaurant (Scenario 2)



A Restaurant: CPN Tool Simulation

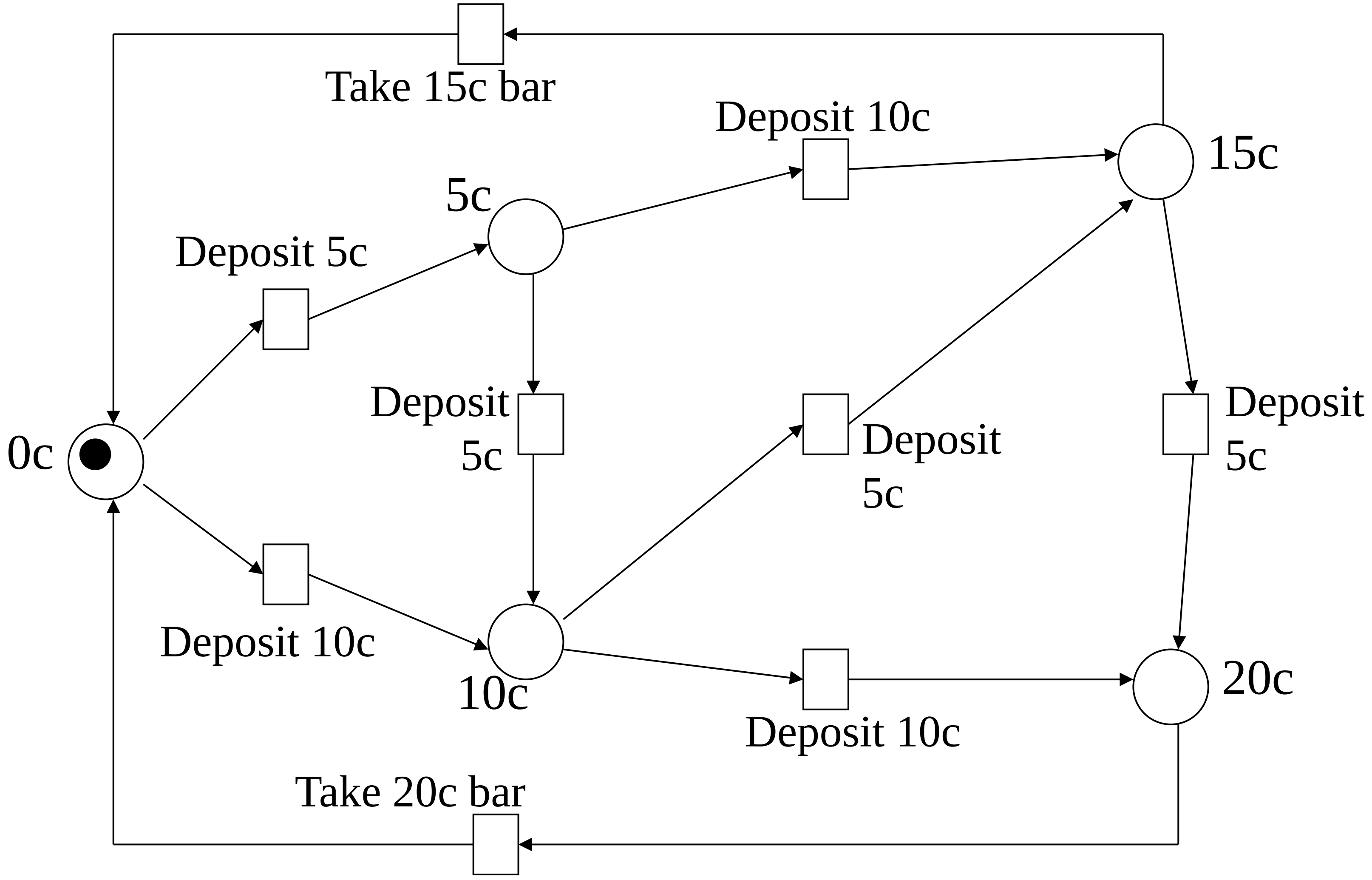


A Restaurant: State Space Analysis



```
1
2:2
1:
Restaurant'Customer_1 1: 1`"Customer 1"
Restaurant'Waiter_Free 1: 1`"Waiter 1"
Restaurant'Customer_2 1: 1`"Customer 2"
Restaurant'Wait_1 1: empty
Restaurant'Order_Taken 1: empty
Restaurant'Wait_2 1: empty
Restaurant'eating_2 1: empty
Restaurant'eating_1 1: empty

2
2:1
2:
Restaurant'Customer_1 1: empty
Restaurant'Waiter_Free 1: empty
Restaurant'Customer_2 1: 1`"Customer 2"
Restaurant'Wait_1 1: 1`"Customer 1"
Restaurant'Order_Taken 1: 1`"Waiter 1"
Restaurant'Wait_2 1: empty
Restaurant'eating_2 1: empty
Restaurant'eating_1 1: empty

3
2:1
3:
Restaurant'Customer_1 1: 1`"Customer 1"
Restaurant'Waiter_Free 1: empty
Restaurant'Customer_2 1: empty
Restaurant'Wait_1 1: empty
Restaurant'Order_Taken 1: 1`"Waiter 1"
Restaurant'Wait_2 1: 1`"Customer 2"
Restaurant'eating_2 1: empty
Restaurant'eating_1 1: empty

4
2:2
4:
Restaurant'Customer_1 1: empty
Restaurant'Waiter_Free 1: 1`"Waiter 1"
Restaurant'Customer_2 1: 1`"Customer 2"
Restaurant'Wait_1 1: 1`"Customer 1"
Restaurant'Order_Taken 1: empty
Restaurant'Wait_2 1: empty
Restaurant'eating_2 1: empty
Restaurant'eating_1 1: empty

5
2:2
5:
Restaurant'Customer_1 1: 1`"Customer 1"
Restaurant'Waiter_Free 1: 1`"Waiter 1"
Restaurant'Customer_2 1: empty
Restaurant'Wait_1 1: empty
Restaurant'Order_Taken 1: empty
Restaurant'Wait_2 1: 1`"Customer 2"
Restaurant'eating_2 1: empty
Restaurant'eating_1 1: empty

6
2:1
6:
Restaurant'Customer_1 1: empty
Restaurant'Waiter_Free 1: empty
Restaurant'Customer_2 1: empty
Restaurant'Wait_1 1: 1`"Customer 1"
Restaurant'Order_Taken 1: 1`"Waiter 1"
Restaurant'Wait_2 1: 1`"Customer 2"
Restaurant'eating_2 1: empty
Restaurant'eating_1 1: empty

7
2:2
7:
Restaurant'Customer_1 1: empty
Restaurant'Waiter_Free 1: 1`"Waiter 1"
Restaurant'Customer_2 1: 1`"Customer 2"
Restaurant'Wait_1 1: empty
Restaurant'Order_Taken 1: empty
Restaurant'Wait_2 1: empty
Restaurant'eating_2 1: empty
Restaurant'eating_1 1: 1`"Customer 1"

8
2:2
8:
Restaurant'Customer_1 1: 1`"Customer 1"
Restaurant'Waiter_Free 1: 1`"Waiter 1"
Restaurant'Customer_2 1: empty
Restaurant'Wait_1 1: empty
Restaurant'Order_Taken 1: empty
Restaurant'Wait_2 1: empty
Restaurant'eating_2 1: 1`"Customer 2"
Restaurant'eating_1 1: empty

9
1:2
9:
Restaurant'Customer_1 1: empty
Restaurant'Waiter_Free 1: 1`"Waiter 1"
Restaurant'Customer_2 1: empty
Restaurant'Wait_1 1: 1`"Customer 1"
Restaurant'Order_Taken 1: empty
Restaurant'Wait_2 1: 1`"Customer 2"
Restaurant'eating_2 1: empty
Restaurant'eating_1 1: empty

10
1:2
10:
Restaurant'Customer_1 1: empty
Restaurant'Waiter_Free 1: empty
Restaurant'Customer_2 1: empty
Restaurant'Wait_1 1: empty
Restaurant'Order_Taken 1: 1`"Waiter 1"
Restaurant'Wait_2 1: 1`"Customer 2"
Restaurant'eating_2 1: empty
Restaurant'eating_1 1: 1`"Customer 1"

11
1:2
11:
Restaurant'Customer_1 1: empty
Restaurant'Waiter_Free 1: empty
Restaurant'Customer_2 1: empty
Restaurant'Wait_1 1: 1`"Customer 1"
Restaurant'Order_Taken 1: 1`"Waiter 1"
Restaurant'Wait_2 1: empty
Restaurant'eating_2 1: 1`"Customer 2"
Restaurant'eating_1 1: empty

12
2:2
12:
Restaurant'Customer_1 1: empty
Restaurant'Waiter_Free 1: 1`"Waiter 1"
Restaurant'Customer_2 1: empty
Restaurant'Wait_1 1: empty
Restaurant'Order_Taken 1: empty
Restaurant'Wait_2 1: 1`"Customer 2"
Restaurant'eating_2 1: empty
Restaurant'eating_1 1: 1`"Customer 1"

13
2:2
13:
Restaurant'Customer_1 1: empty
Restaurant'Waiter_Free 1: 1`"Waiter 1"
Restaurant'Customer_2 1: empty
Restaurant'Wait_1 1: 1`"Customer 1"
Restaurant'Order_Taken 1: empty
Restaurant'Wait_2 1: empty
Restaurant'eating_2 1: 1`"Customer 2"
Restaurant'eating_1 1: empty

14
2:2
14:
Restaurant'Customer_1 1: empty
Restaurant'Waiter_Free 1: 1`"Waiter 1"
Restaurant'Customer_2 1: empty
Restaurant'Wait_1 1: empty
Restaurant'Order_Taken 1: empty
Restaurant'Wait_2 1: empty
Restaurant'eating_2 1: 1`"Customer 2"
Restaurant'eating_1 1: 1`"Customer 1"
```

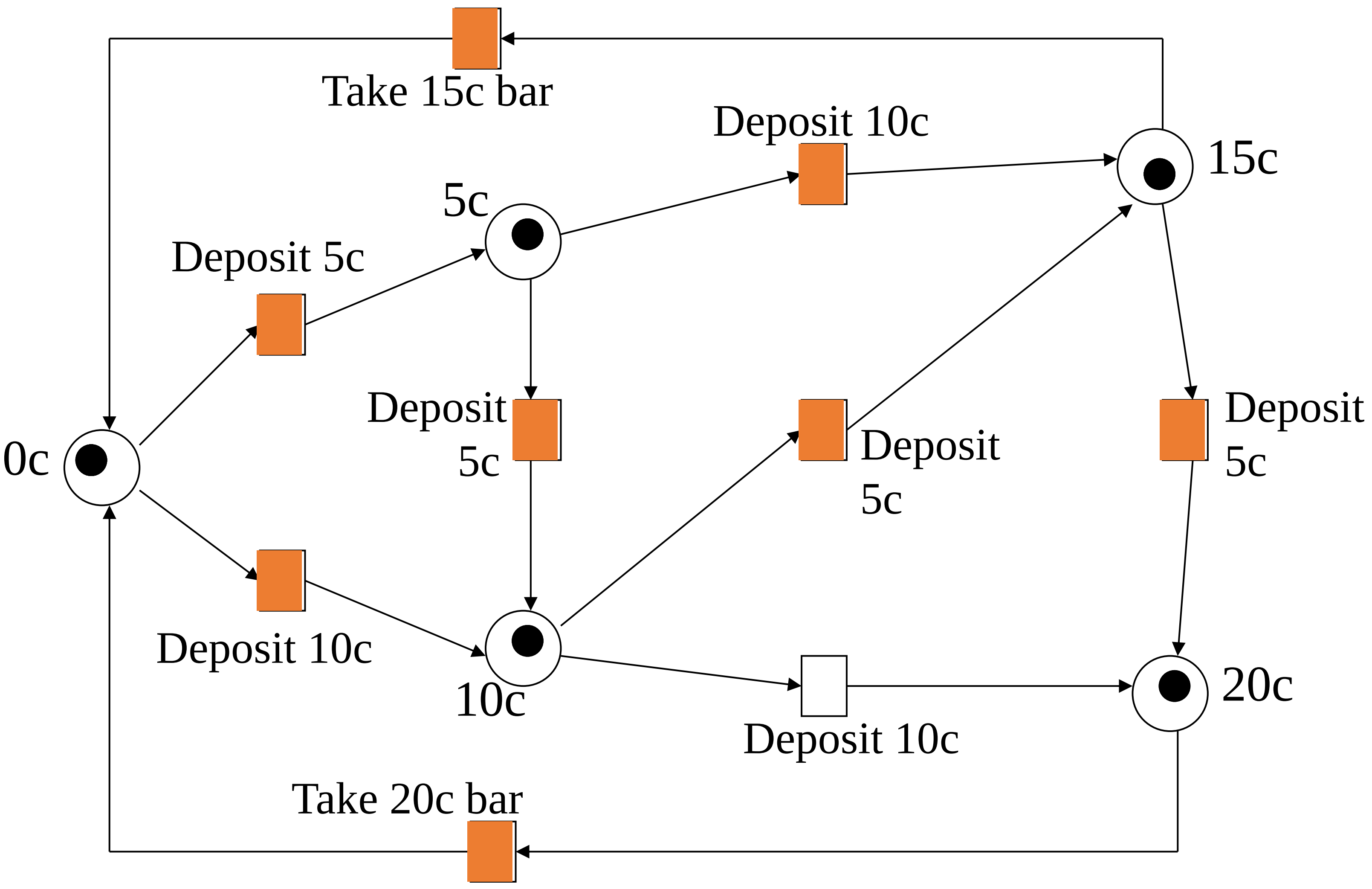
Example: Vending Machine (A Petri net)



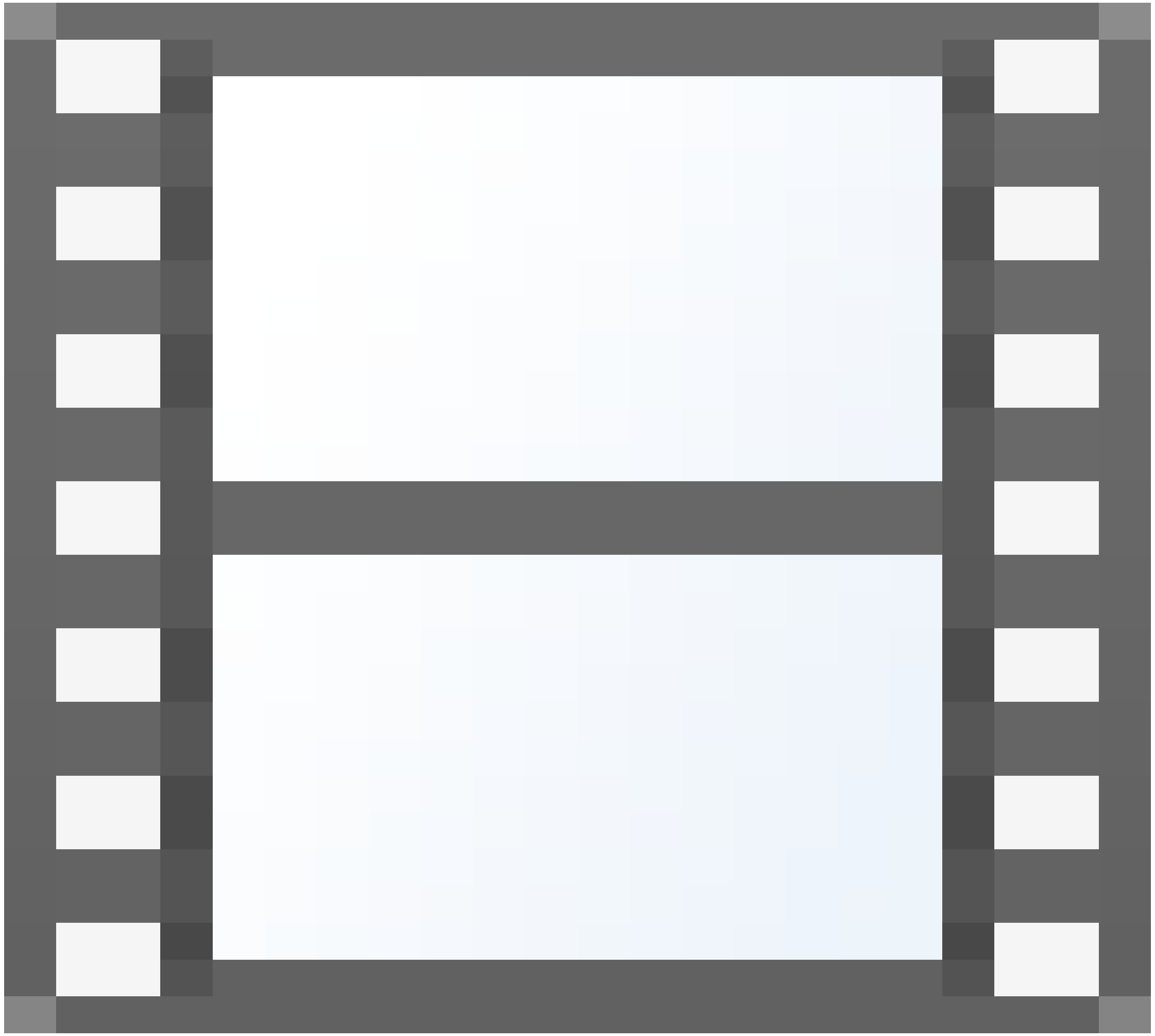
Example: Vending Machine (3 Scenarios)

- 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.

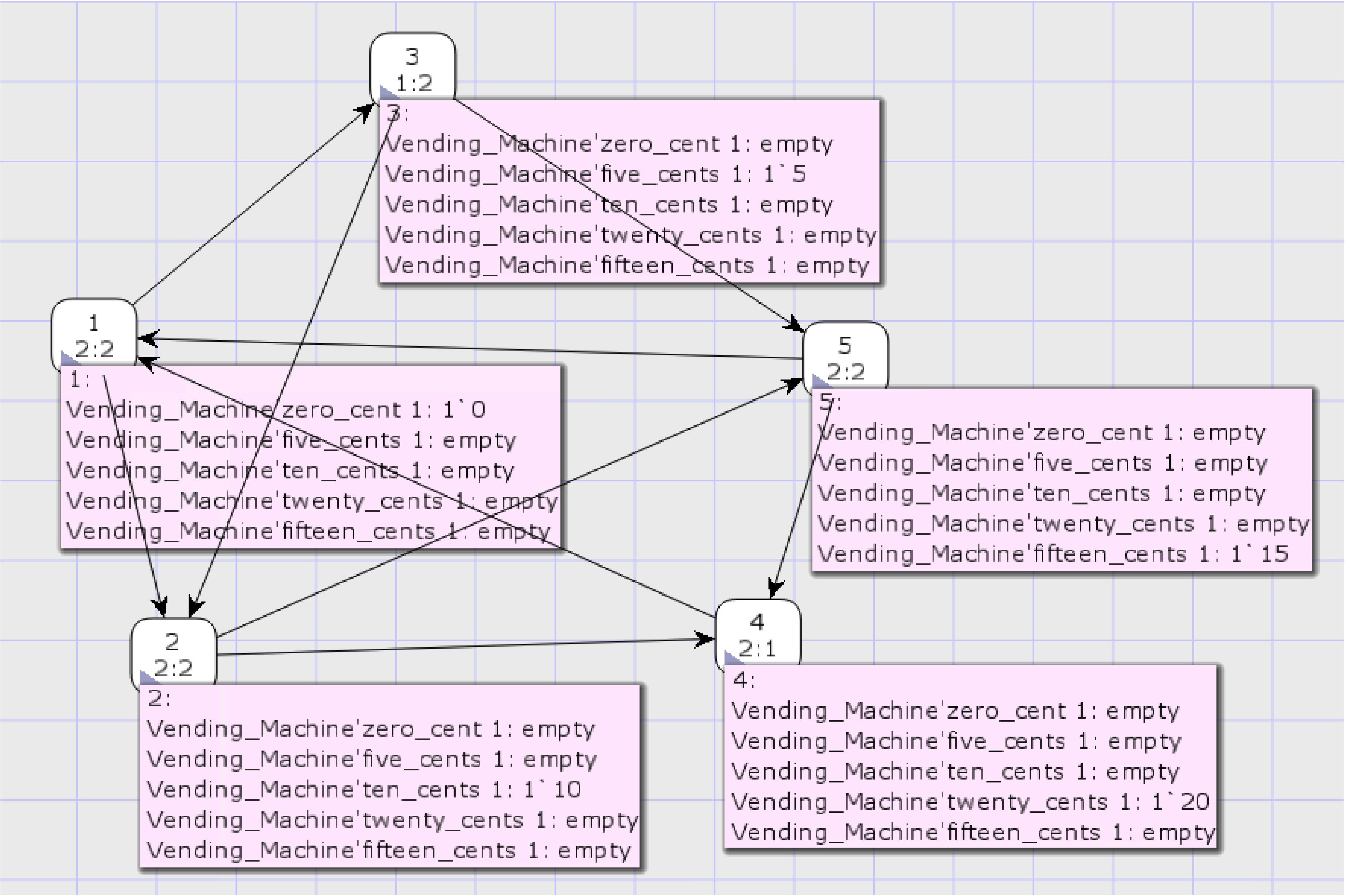
Example: Vending Machine (Token Games)



CPN Tool Simulation



Vending Machine: State Space Analysis



```
3:
Vending_Machine'zero_cent 1: empty
Vending_Machine'five_cents 1: 1`5
Vending_Machine'ten_cents 1: empty
Vending_Machine'twenty_cents 1: empty
Vending_Machine'fifteen_cents 1: empty
```

```
1:
Vending_Machine'zero_cent 1: 1`0
Vending_Machine'five_cents 1: empty
Vending_Machine'ten_cents 1: empty
Vending_Machine'twenty_cents 1: empty
Vending_Machine'fifteen_cents 1: empty
```

```
5:
Vending_Machine'zero_cent 1: empty
Vending_Machine'five_cents 1: empty
Vending_Machine'ten_cents 1: empty
Vending_Machine'twenty_cents 1: empty
Vending_Machine'fifteen_cents 1: 1`15
```

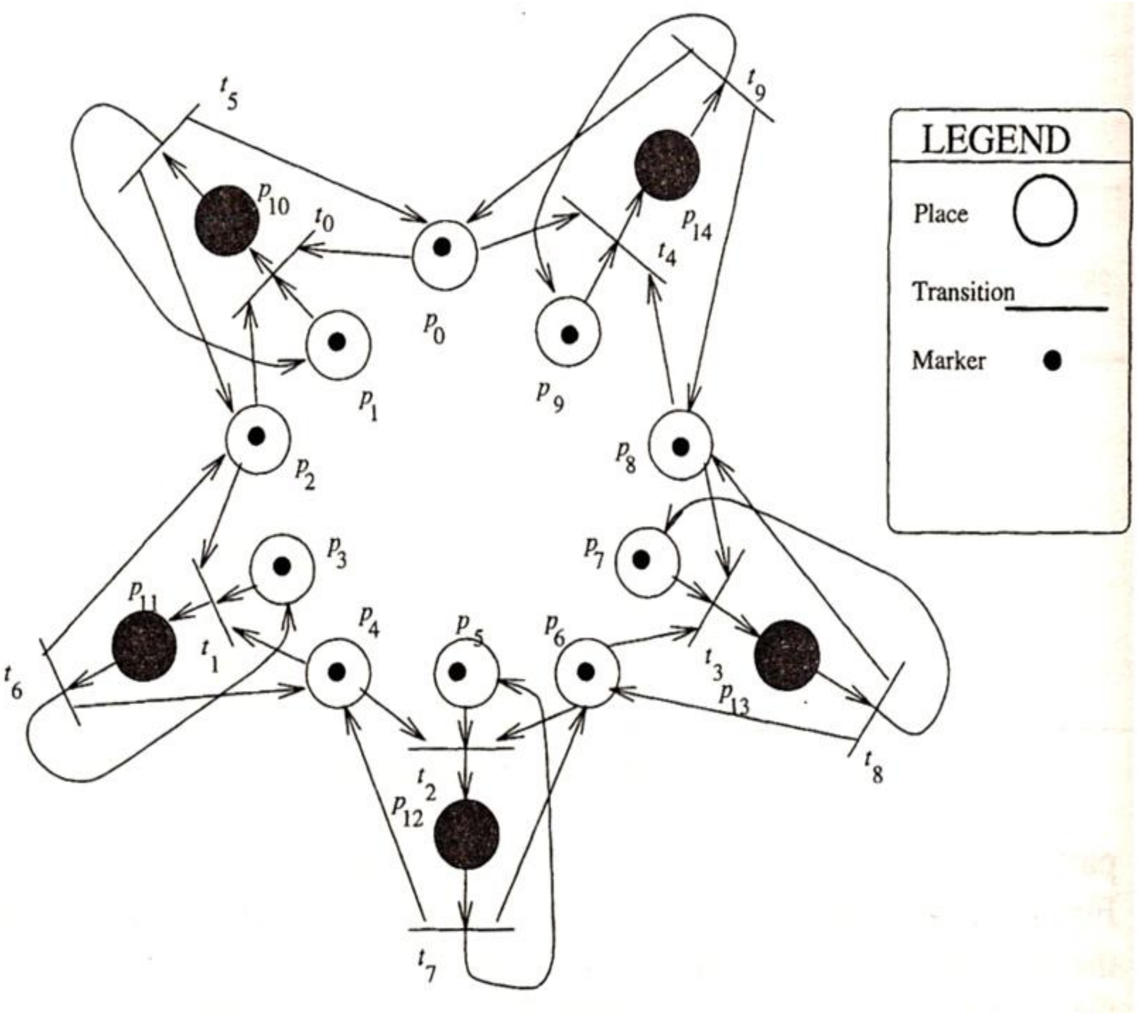
```
2:
Vending_Machine'zero_cent 1: empty
Vending_Machine'five_cents 1: empty
Vending_Machine'ten_cents 1: 1`10
Vending_Machine'twenty_cents 1: empty
Vending_Machine'fifteen_cents 1: empty
```

```
4:
Vending_Machine'zero_cent 1: empty
Vending_Machine'five_cents 1: empty
Vending_Machine'ten_cents 1: empty
Vending_Machine'twenty_cents 1: 1`20
Vending_Machine'fifteen_cents 1: empty
```

Concurrent modelling example
Dining Philosophers Problem

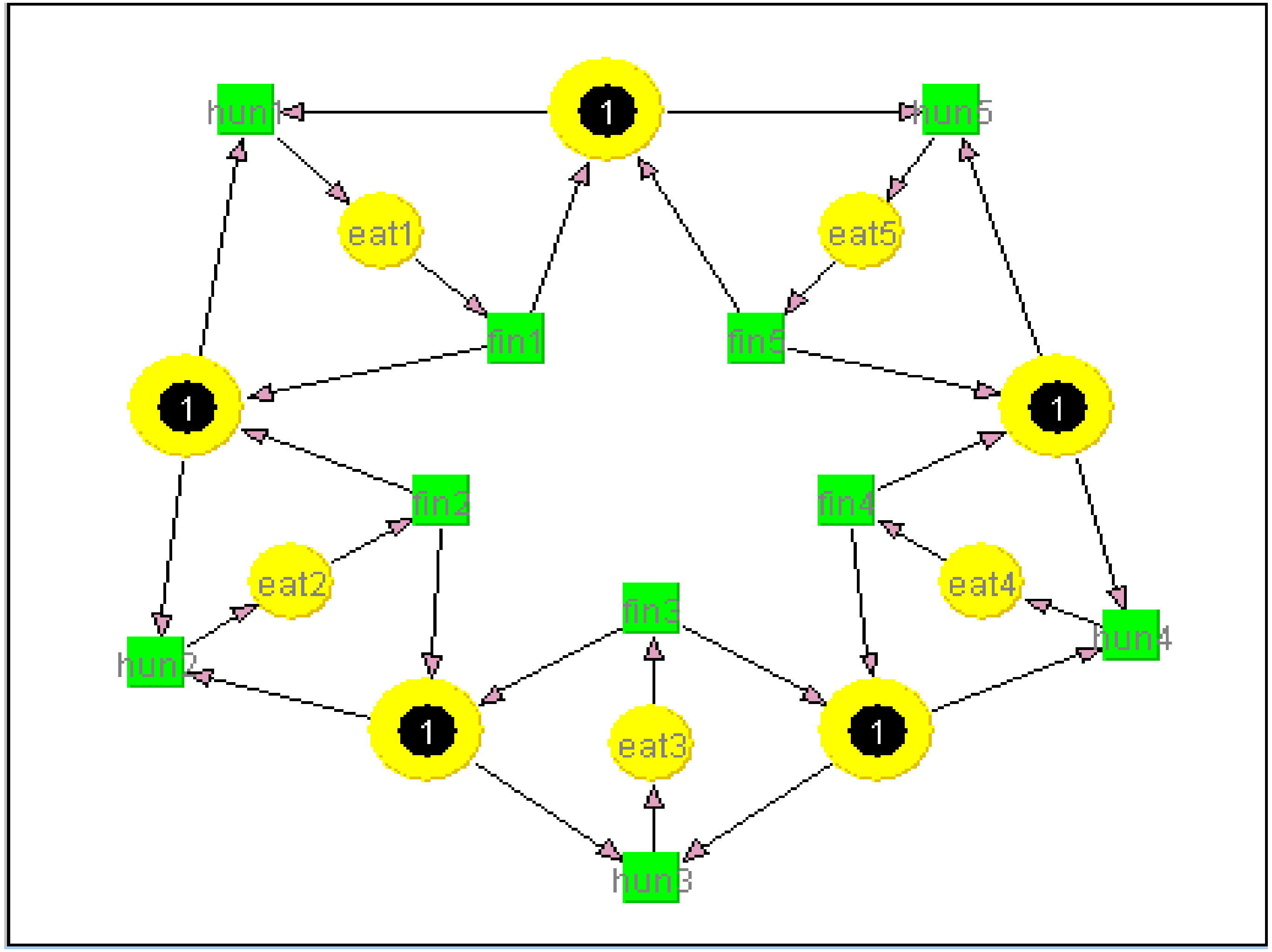
- The dining philosophers problem is summarized as five philosophers sitting at a table doing one of two things: *eating or thinking*. *While eating, they are not thinking, and while thinking, they are not eating*. The five philosophers sit at a circular table with a large bowl of noodle in the centre. A fork is placed in between each pair of adjacent philosophers, and as such, each philosopher has one fork to his left and one fork to his right. As noodle is difficult to serve and eat with a single fork, it is assumed that *a philosopher must eat with two forks*. Each philosopher can only use the forks on his immediate left and immediate right.

Petri Net examples (Dining Philosophers)

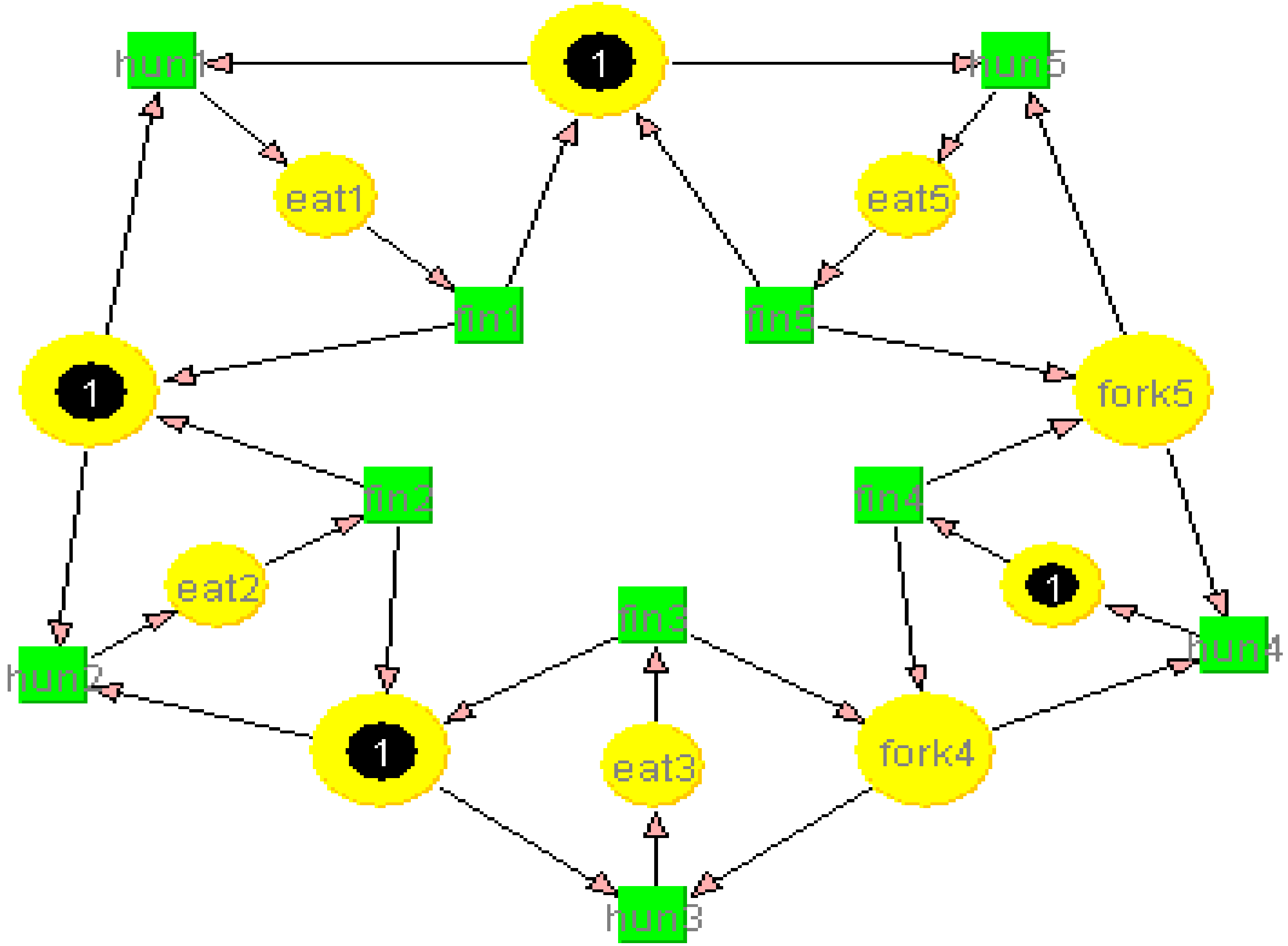


- Five philosophers alternatively think and eating
- forks: p_0, p_2, p_4, p_6, p_8
- Philosophers eating: p_{10}, p_{11}, p_{12}, p_{13}, p_{14}
- Philosophers thinking/meditating: p_1, p_3, p_5, p_7, p_9

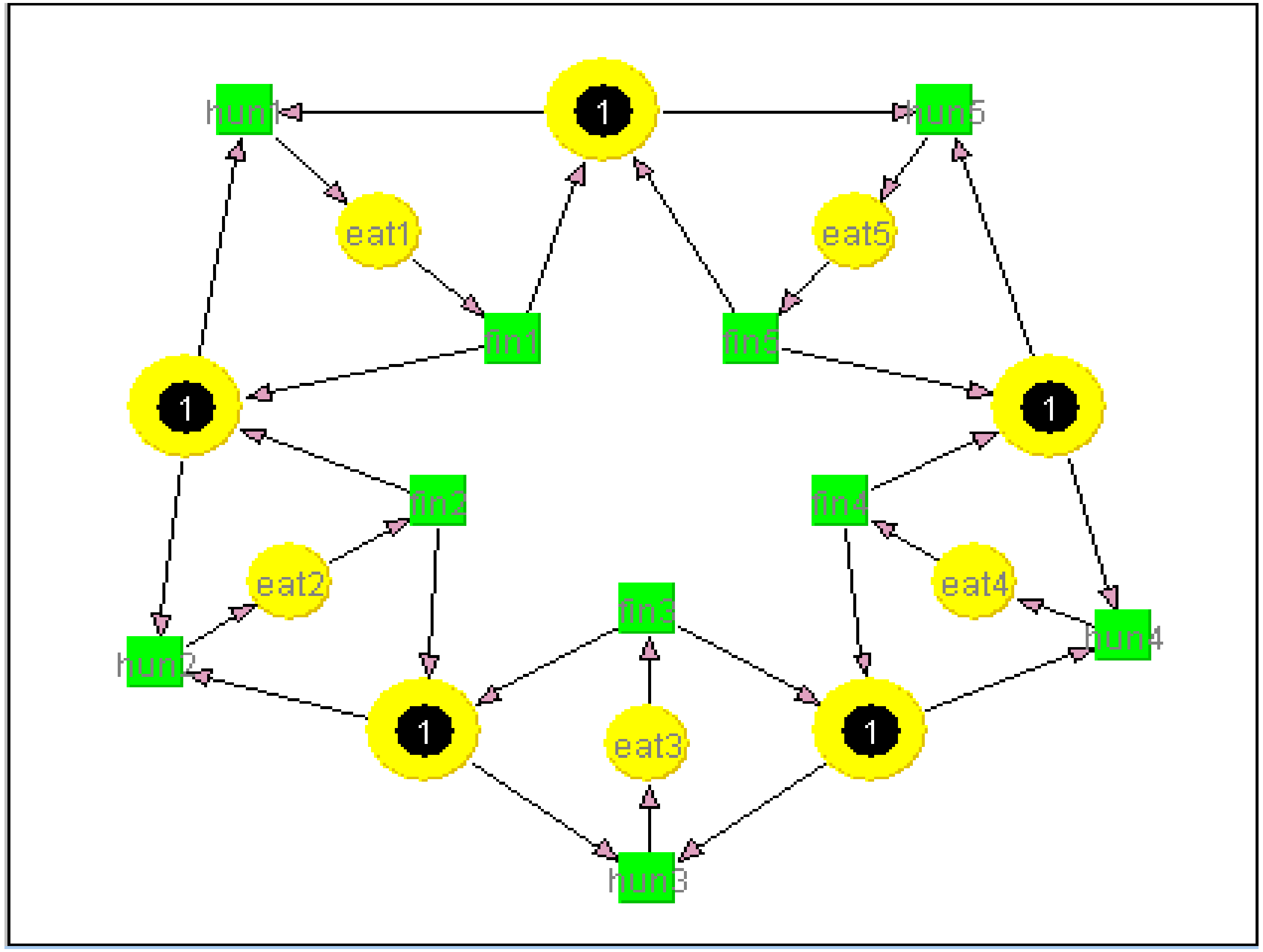
The Petri net structure



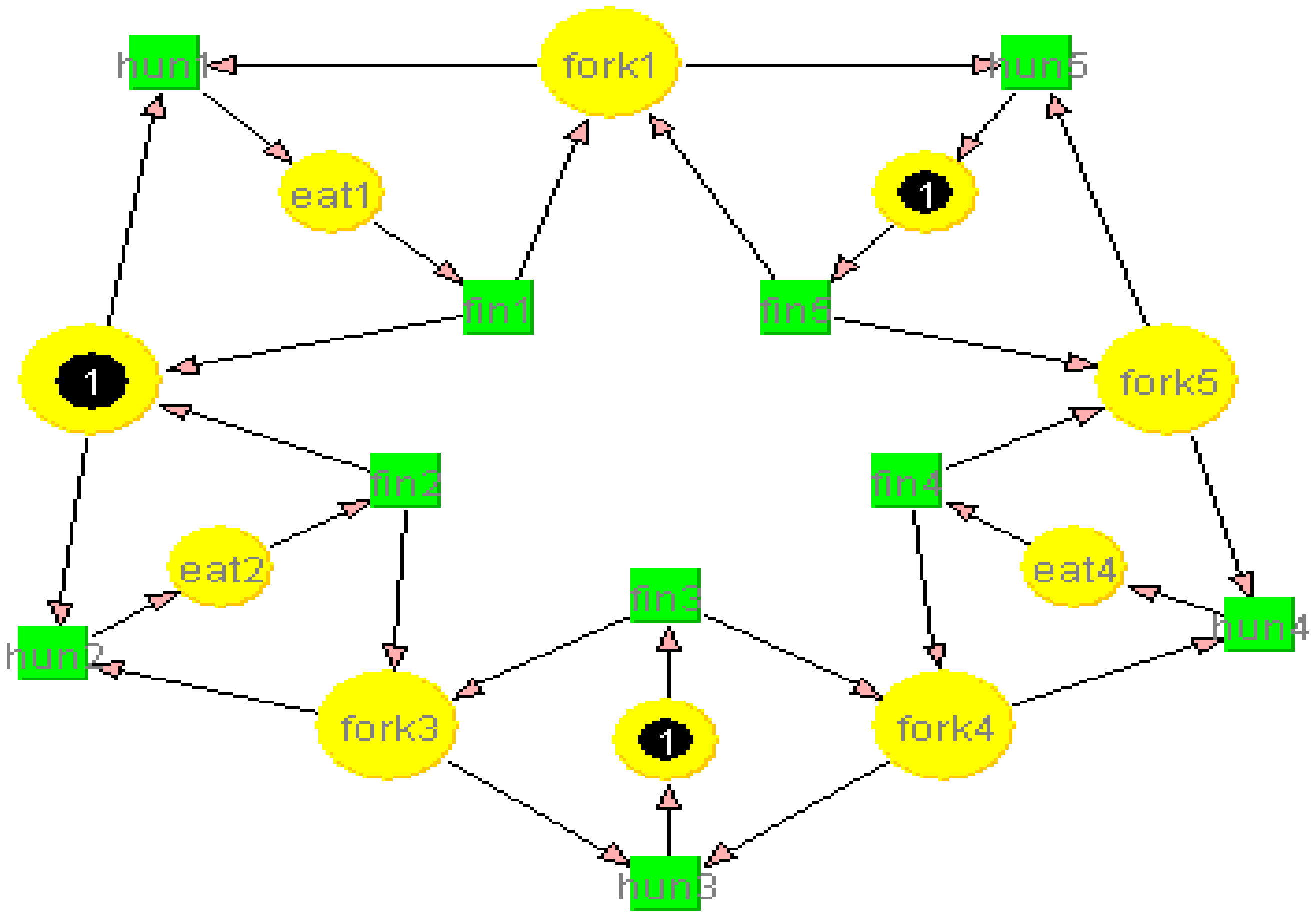
Suppose eater 4 picks up forks first so the transition hun4 fires. The state of Petri net as follows:



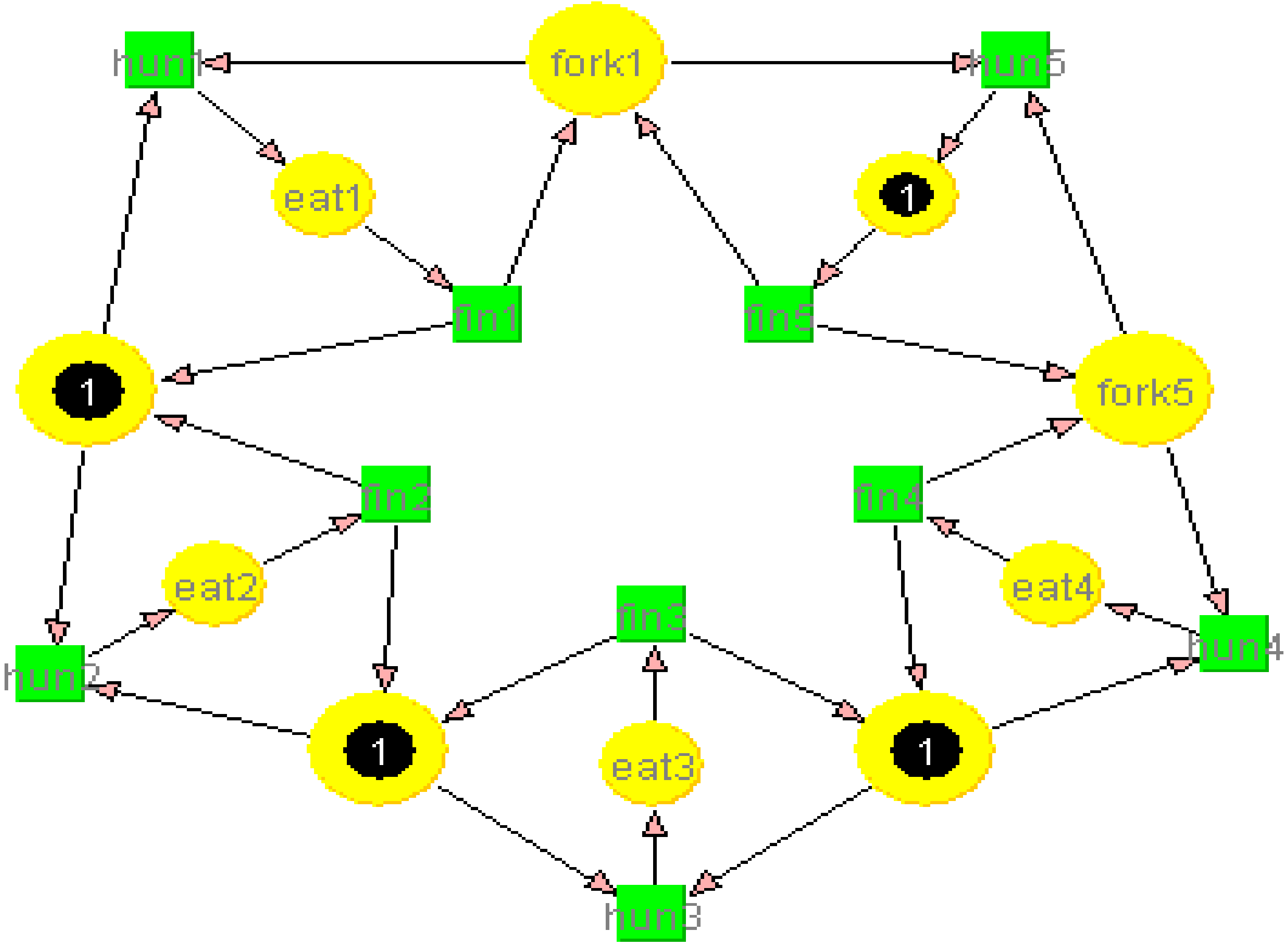
After finishing eating, transition fin4 fires. The state of Petri net is:



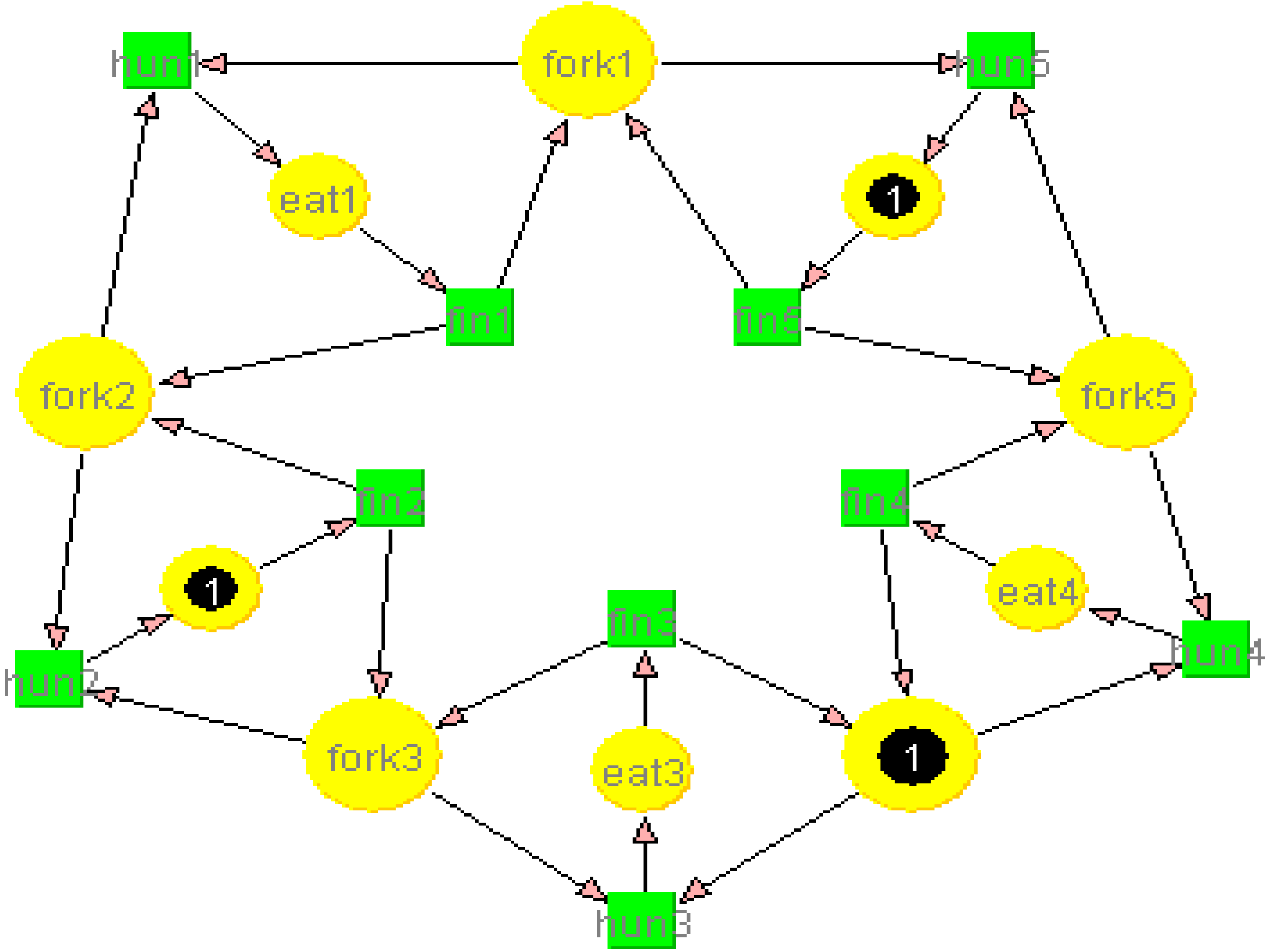
Suppose that eater 3 and eater 5 pick up the forks in the same time so the transition hun3 and hun5 fire in the same time. The state of Petri net became as:



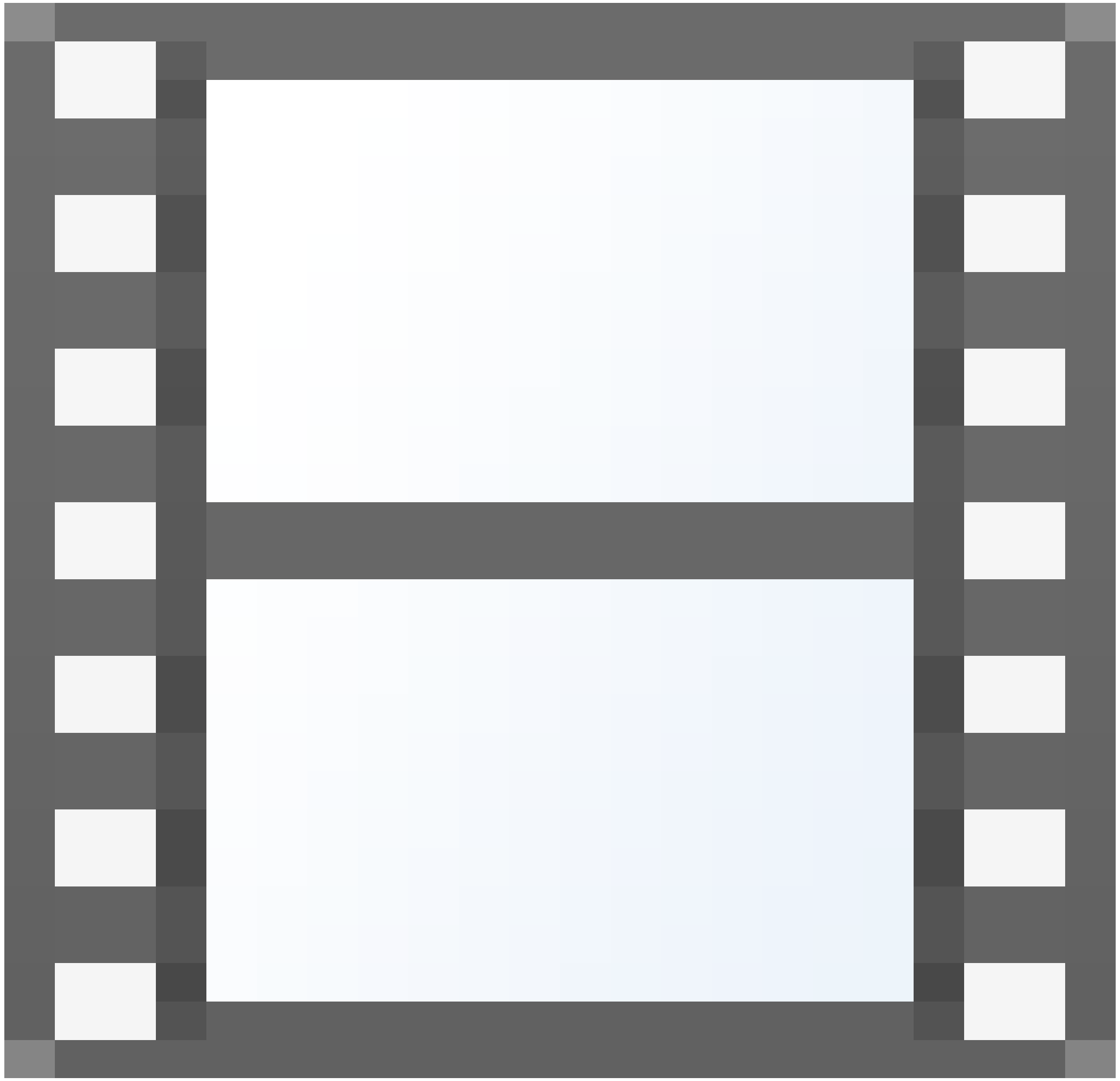
Suppose that eater 3 finished his eating first, so fin3 fires. The state of the Petri net becomes:



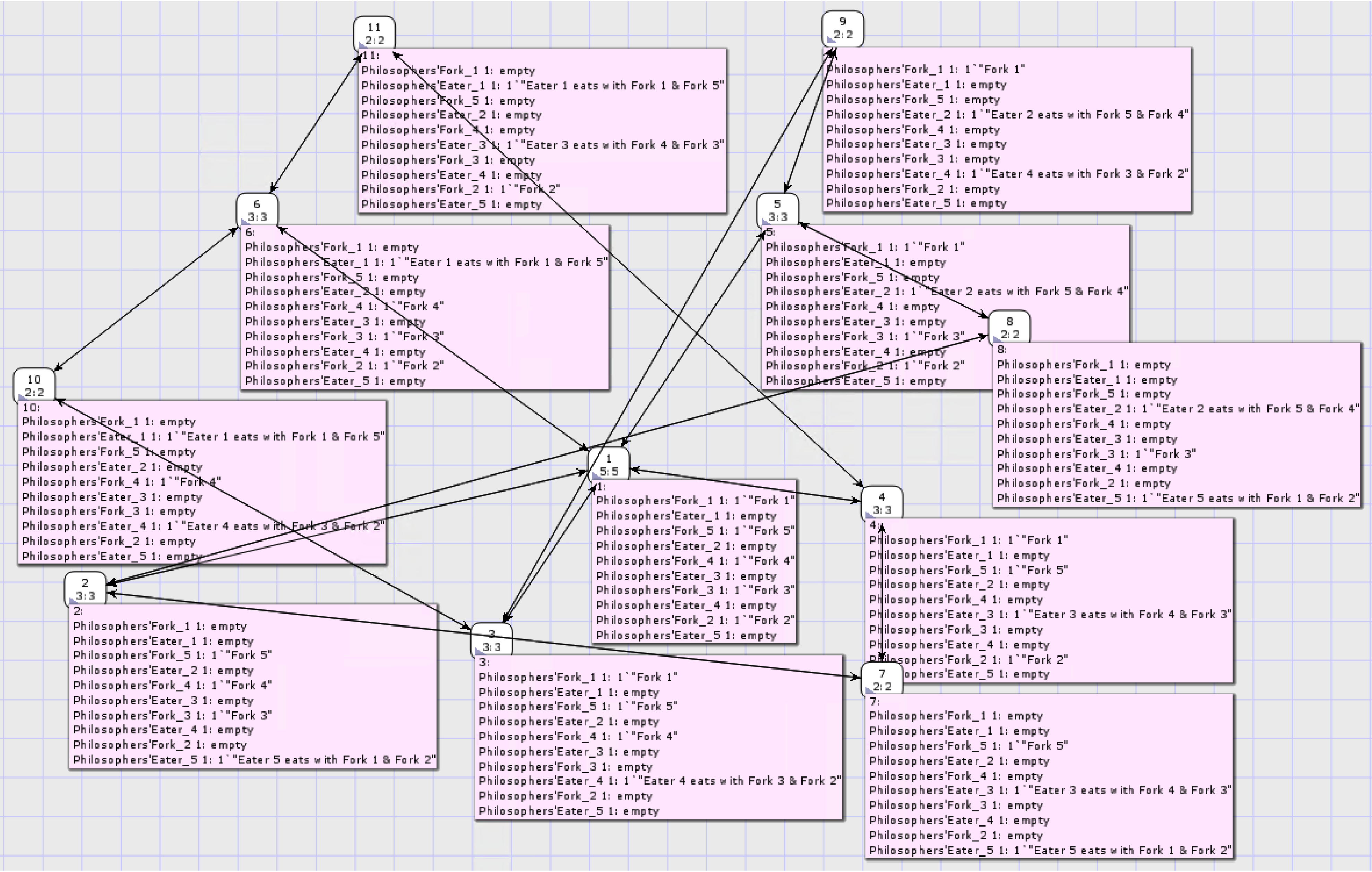
Eater 2 has two forks available. Transition hun2 fires. The state of the Petri net changed to the follows:



CPN Tools Simulation



Philosophers: State Space Analysis



```
11:
Philosophers'Fork_1 1: empty
Philosophers'Eater_1 1: 1`"Eater 1 eats with Fork 1 & Fork 5"
Philosophers'Fork_5 1: empty
Philosophers'Eater_2 1: empty
Philosophers'Fork_4 1: empty
Philosophers'Eater_3 1: 1`"Eater 3 eats with Fork 4 & Fork 3"
Philosophers'Fork_3 1: empty
Philosophers'Eater_4 1: empty
Philosophers'Fork_2 1: 1`"Fork 2"
Philosophers'Eater_5 1: empty
```

```
Philosophers'Fork_1 1: 1`"Fork 1"
Philosophers'Eater_1 1: empty
Philosophers'Fork_5 1: empty
Philosophers'Eater_2 1: 1`"Eater 2 eats with Fork 5 & Fork 4"
Philosophers'Fork_4 1: empty
Philosophers'Eater_3 1: empty
Philosophers'Fork_3 1: empty
Philosophers'Eater_4 1: 1`"Eater 4 eats with Fork 3 & Fork 2"
Philosophers'Fork_2 1: empty
Philosophers'Eater_5 1: empty
```

```
6:
Philosophers'Fork_1 1: empty
Philosophers'Eater_1 1: 1`"Eater 1 eats with Fork 1 & Fork 5"
Philosophers'Fork_5 1: empty
Philosophers'Eater_2 1: empty
Philosophers'Fork_4 1: 1`"Fork 4"
Philosophers'Eater_3 1: empty
Philosophers'Fork_3 1: 1`"Fork 3"
Philosophers'Eater_4 1: empty
Philosophers'Fork_2 1: 1`"Fork 2"
Philosophers'Eater_5 1: empty
```

```
5:
Philosophers'Fork_1 1: 1`"Fork 1"
Philosophers'Eater_1 1: empty
Philosophers'Fork_5 1: empty
Philosophers'Eater_2 1: 1`"Eater 2 eats with Fork 5 & Fork 4"
Philosophers'Fork_4 1: empty
Philosophers'Eater_3 1: empty
Philosophers'Fork_3 1: 1`"Fork 3"
Philosophers'Eater_4 1: empty
Philosophers'Fork_2 1: 1`"Fork 2"
Philosophers'Eater_5 1: empty
```

```
8:
Philosophers'Fork_1 1: empty
Philosophers'Eater_1 1: empty
Philosophers'Fork_5 1: empty
Philosophers'Eater_2 1: 1`"Eater 2 eats with Fork 5 & Fork 4"
Philosophers'Fork_4 1: empty
Philosophers'Eater_3 1: empty
Philosophers'Fork_3 1: 1`"Fork 3"
Philosophers'Eater_4 1: empty
Philosophers'Fork_2 1: empty
Philosophers'Eater_5 1: 1`"Eater 5 eats with Fork 1 & Fork 2"
```

```
10:
Philosophers'Fork_1 1: empty
Philosophers'Eater_1 1: 1`"Eater 1 eats with Fork 1 & Fork 5"
Philosophers'Fork_5 1: empty
Philosophers'Eater_2 1: empty
Philosophers'Fork_4 1: 1`"Fork 4"
Philosophers'Eater_3 1: empty
Philosophers'Fork_3 1: empty
Philosophers'Eater_4 1: 1`"Eater 4 eats with Fork 3 & Fork 2"
Philosophers'Fork_2 1: empty
Philosophers'Eater_5 1: empty
```

```
1:
Philosophers'Fork_1 1: 1`"Fork 1"
Philosophers'Eater_1 1: empty
Philosophers'Fork_5 1: 1`"Fork 5"
Philosophers'Eater_2 1: empty
Philosophers'Fork_4 1: 1`"Fork 4"
Philosophers'Eater_3 1: empty
Philosophers'Fork_3 1: 1`"Fork 3"
Philosophers'Eater_4 1: empty
Philosophers'Fork_2 1: 1`"Fork 2"
Philosophers'Eater_5 1: empty
```

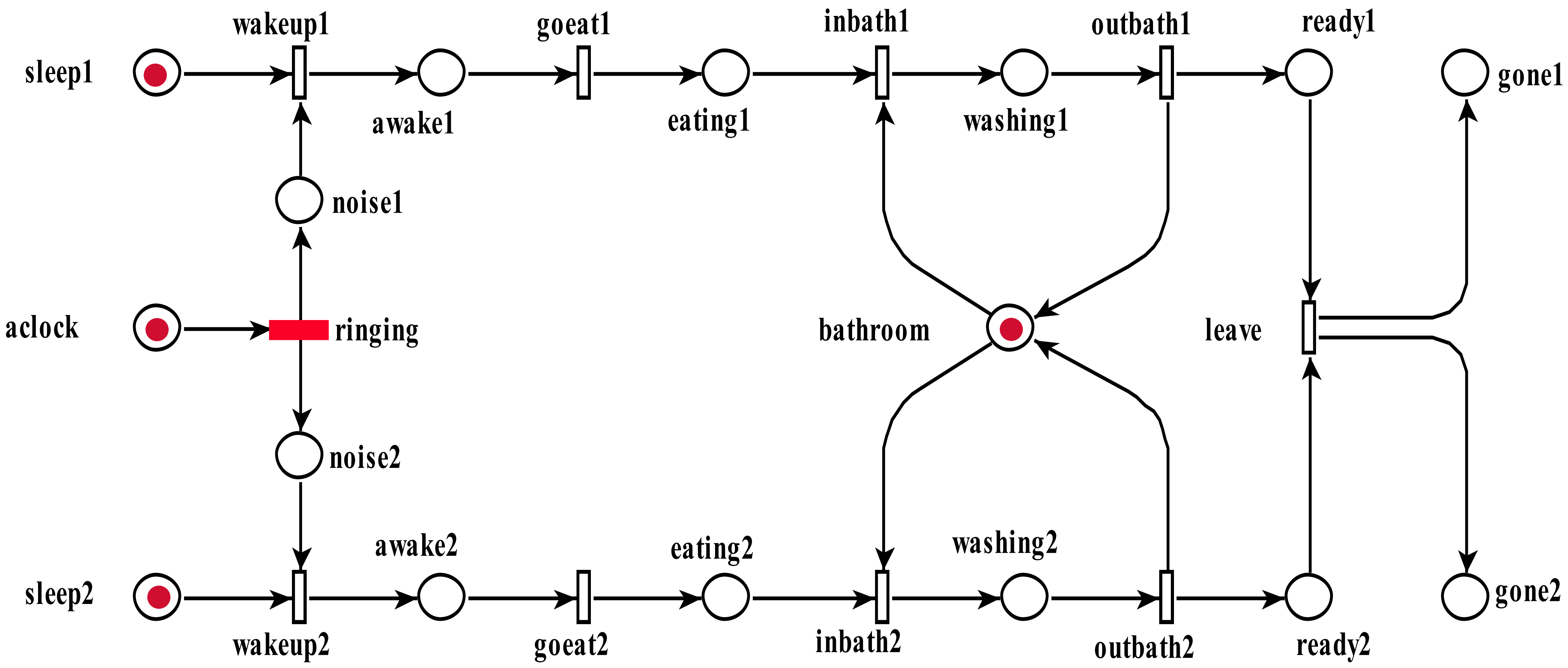
```
4:
Philosophers'Fork_1 1: 1`"Fork 1"
Philosophers'Eater_1 1: empty
Philosophers'Fork_5 1: 1`"Fork 5"
Philosophers'Eater_2 1: empty
Philosophers'Fork_4 1: empty
Philosophers'Eater_3 1: 1`"Eater 3 eats with Fork 4 & Fork 3"
Philosophers'Fork_3 1: empty
Philosophers'Eater_4 1: empty
Philosophers'Fork_2 1: 1`"Fork 2"
Philosophers'Eater_5 1: empty
```

```
2:
Philosophers'Fork_1 1: empty
Philosophers'Eater_1 1: empty
Philosophers'Fork_5 1: 1`"Fork 5"
Philosophers'Eater_2 1: empty
Philosophers'Fork_4 1: 1`"Fork 4"
Philosophers'Eater_3 1: empty
Philosophers'Fork_3 1: 1`"Fork 3"
Philosophers'Eater_4 1: empty
Philosophers'Fork_2 1: empty
Philosophers'Eater_5 1: 1`"Eater 5 eats with Fork 1 & Fork 2"
```

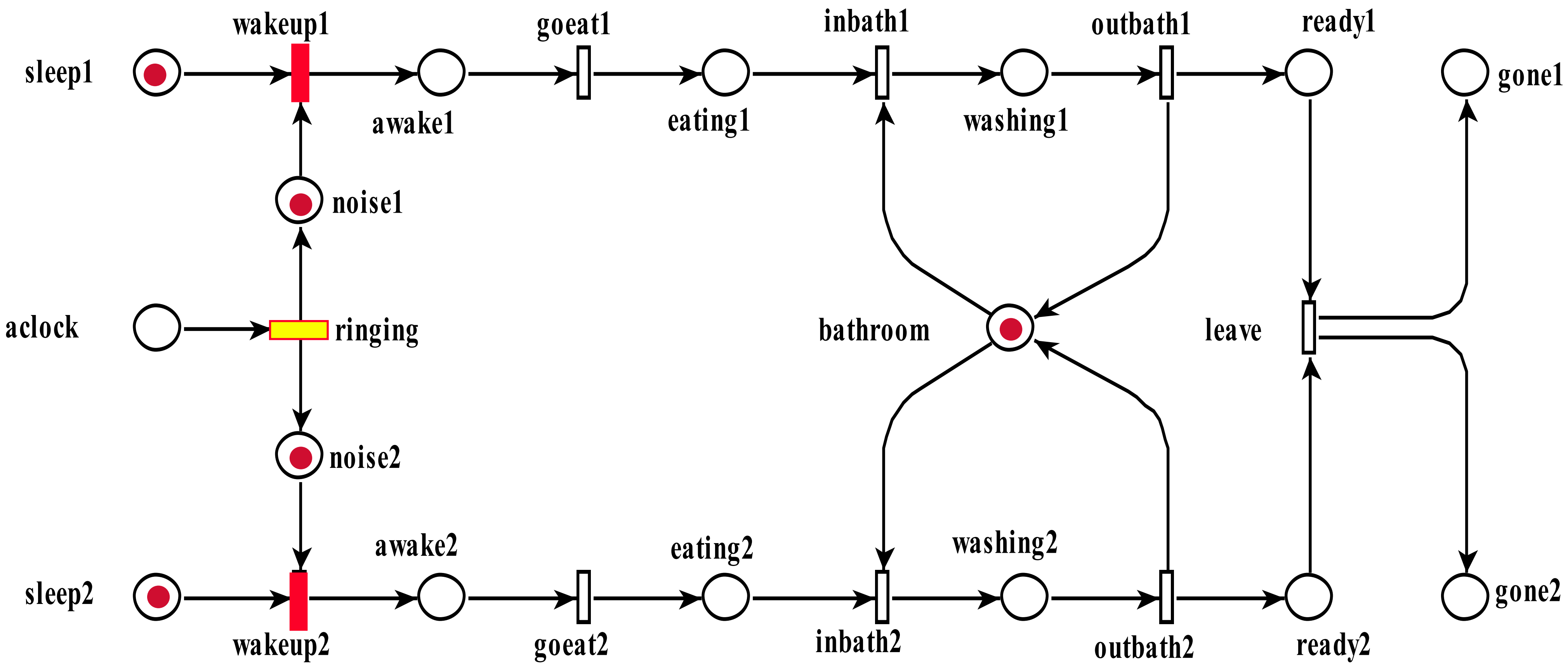
```
3:
Philosophers'Fork_1 1: 1`"Fork 1"
Philosophers'Eater_1 1: empty
Philosophers'Fork_5 1: 1`"Fork 5"
Philosophers'Eater_2 1: empty
Philosophers'Fork_4 1: 1`"Fork 4"
Philosophers'Eater_3 1: empty
Philosophers'Fork_3 1: empty
Philosophers'Eater_4 1: 1`"Eater 4 eats with Fork 3 & Fork 2"
Philosophers'Fork_2 1: empty
Philosophers'Eater_5 1: empty
```

```
7:
Philosophers'Fork_1 1: empty
Philosophers'Eater_1 1: empty
Philosophers'Fork_5 1: 1`"Fork 5"
Philosophers'Eater_2 1: empty
Philosophers'Fork_4 1: empty
Philosophers'Eater_3 1: 1`"Eater 3 eats with Fork 4 & Fork 3"
Philosophers'Fork_3 1: empty
Philosophers'Eater_4 1: empty
Philosophers'Fork_2 1: empty
Philosophers'Eater_5 1: 1`"Eater 5 eats with Fork 1 & Fork 2"
```

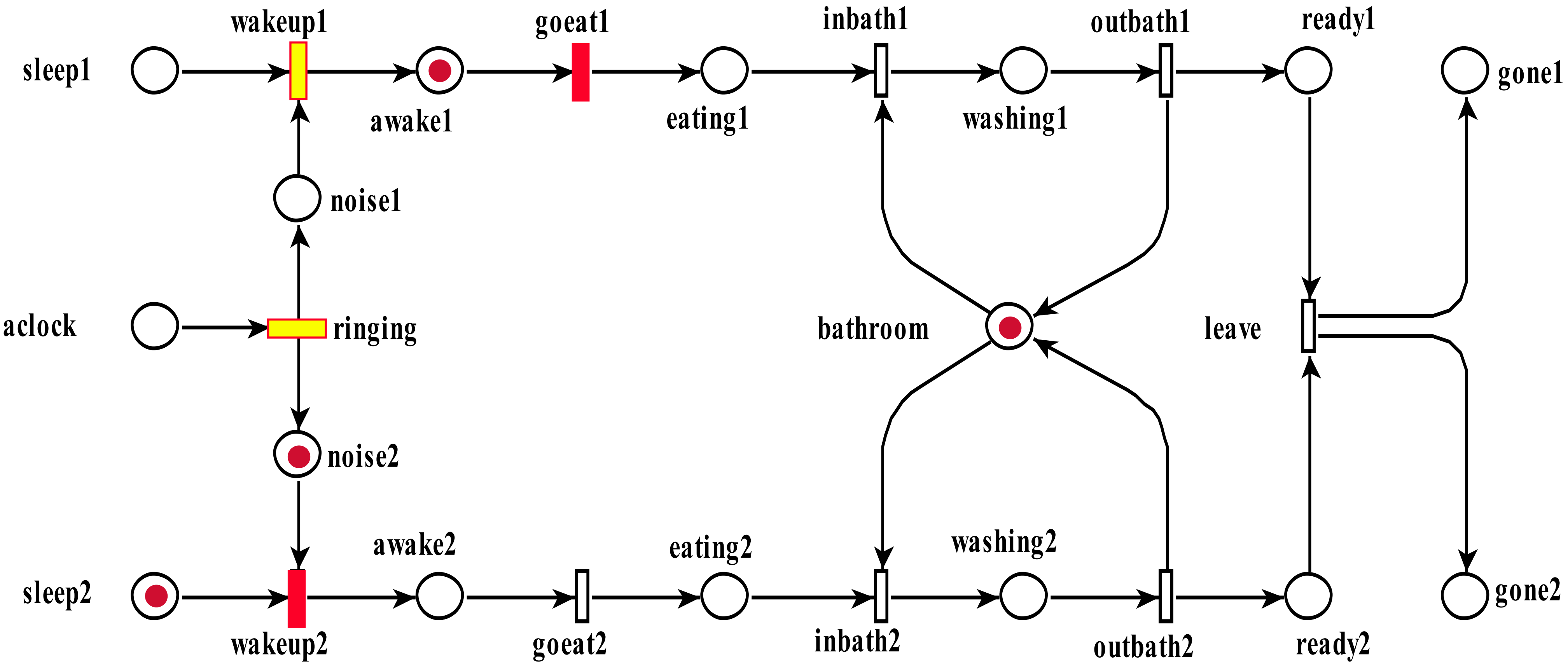
Two people waking up



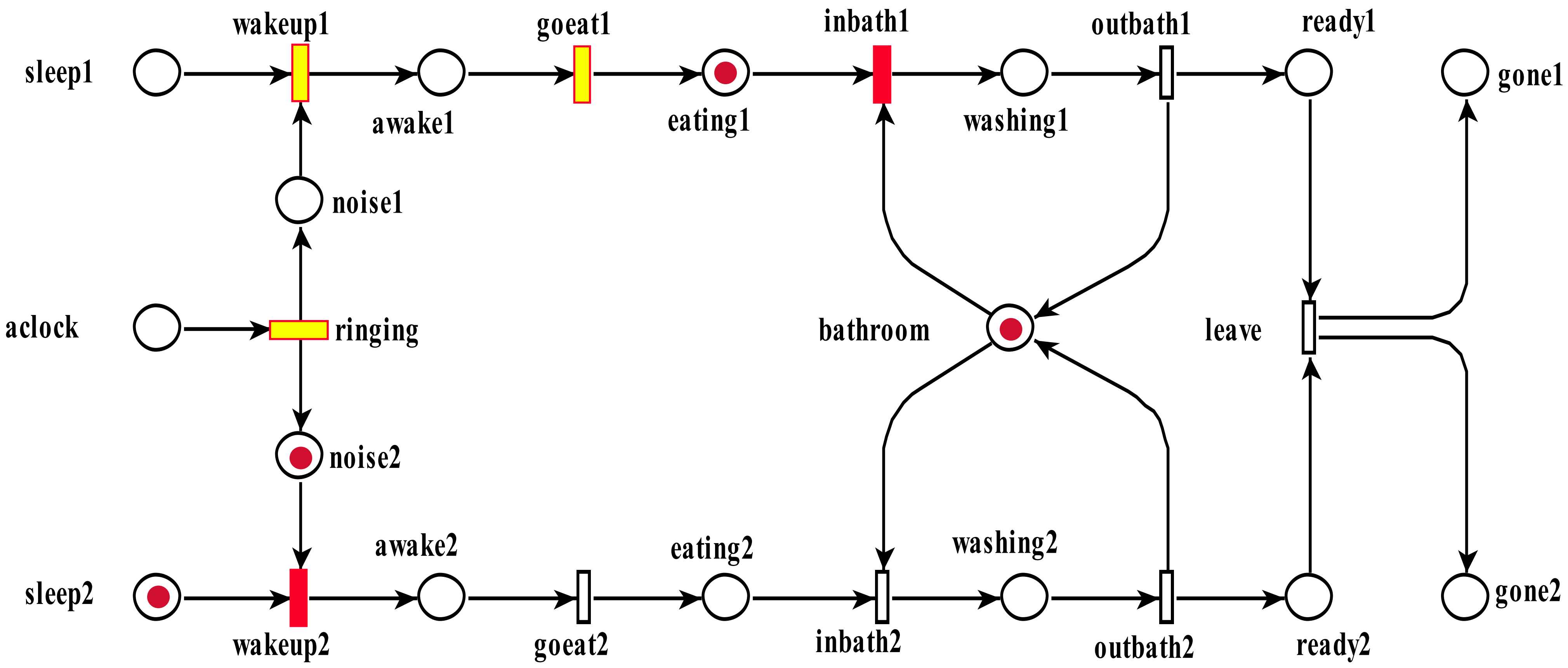
Two people waking up



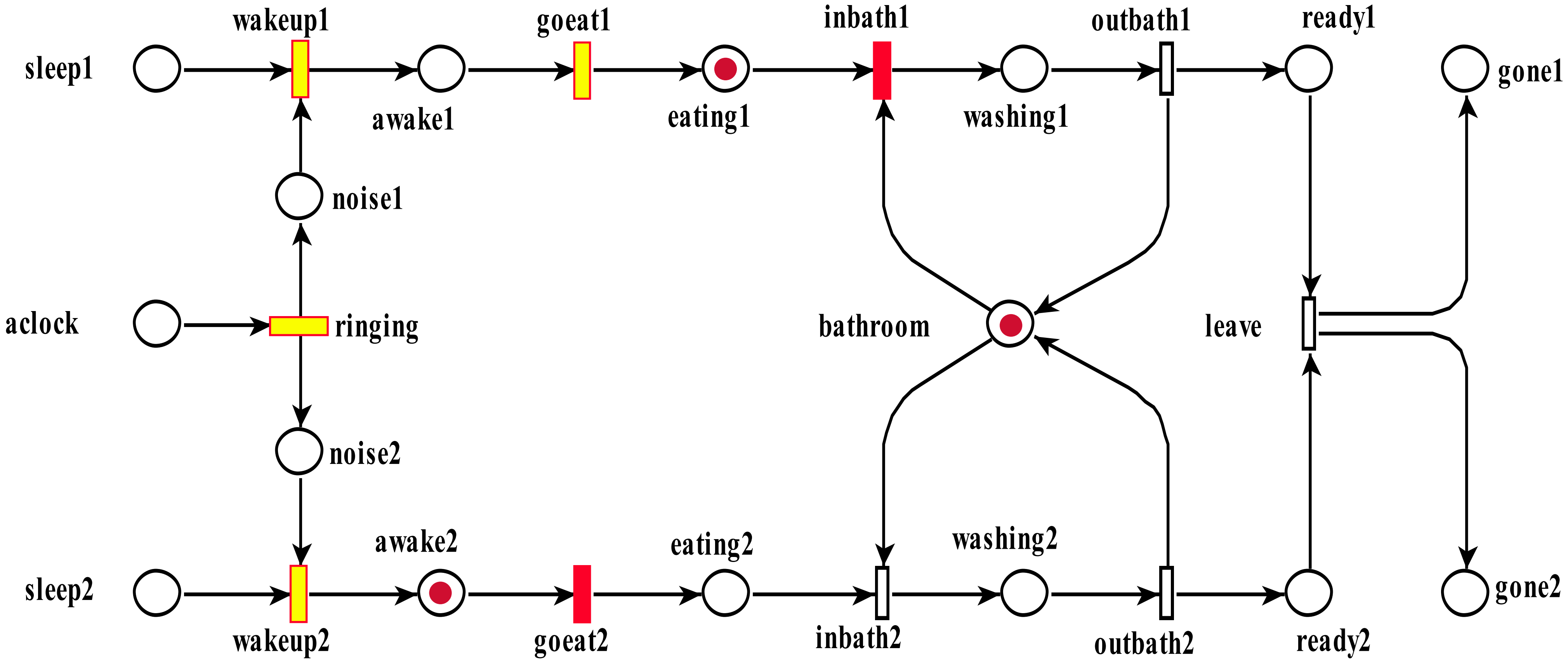
Two people waking up



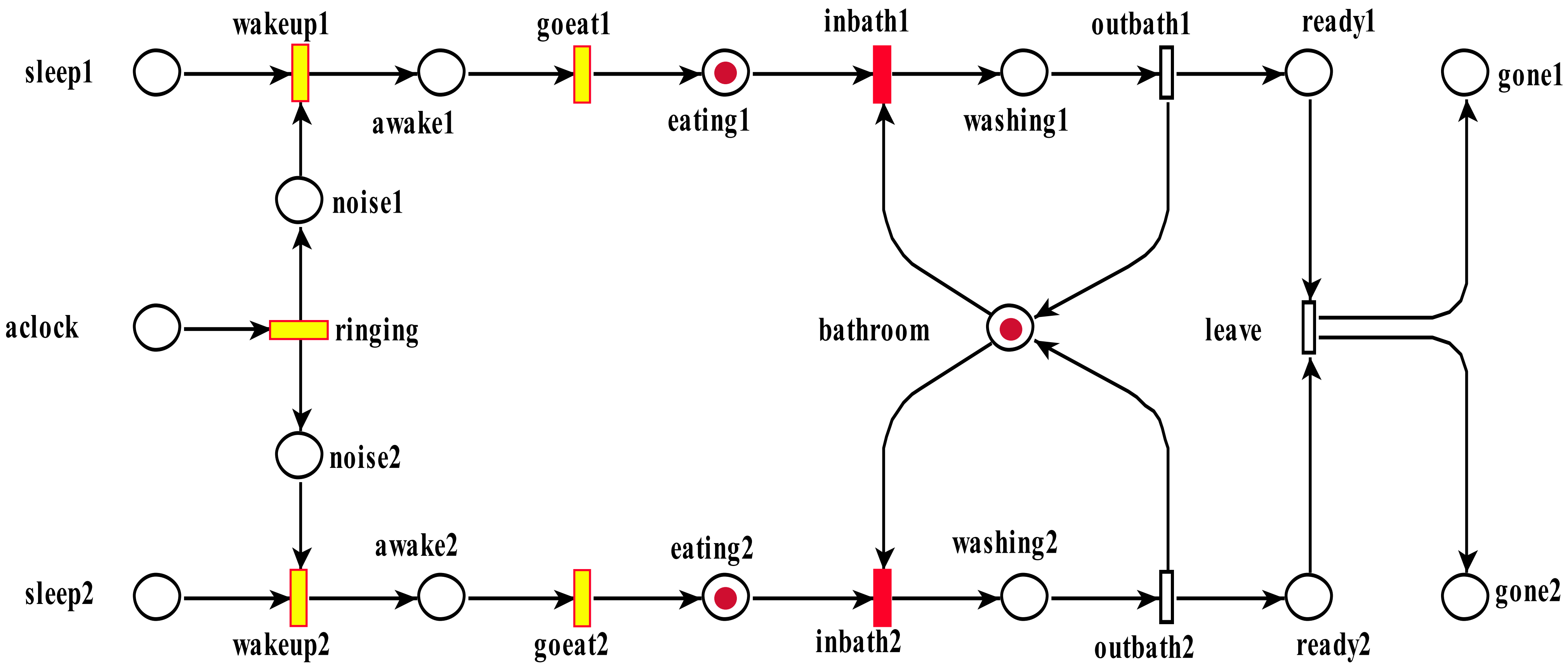
Two people waking up



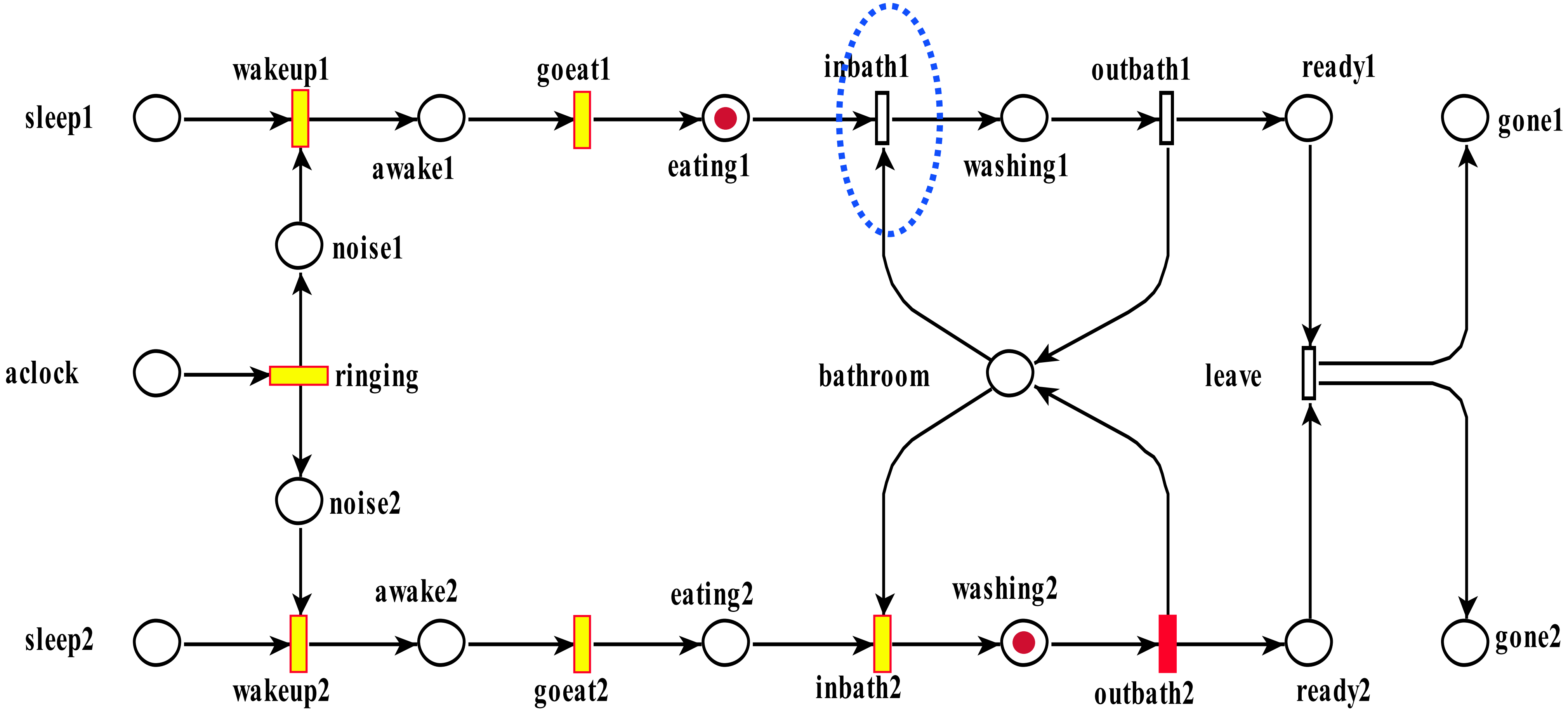
Two people waking up



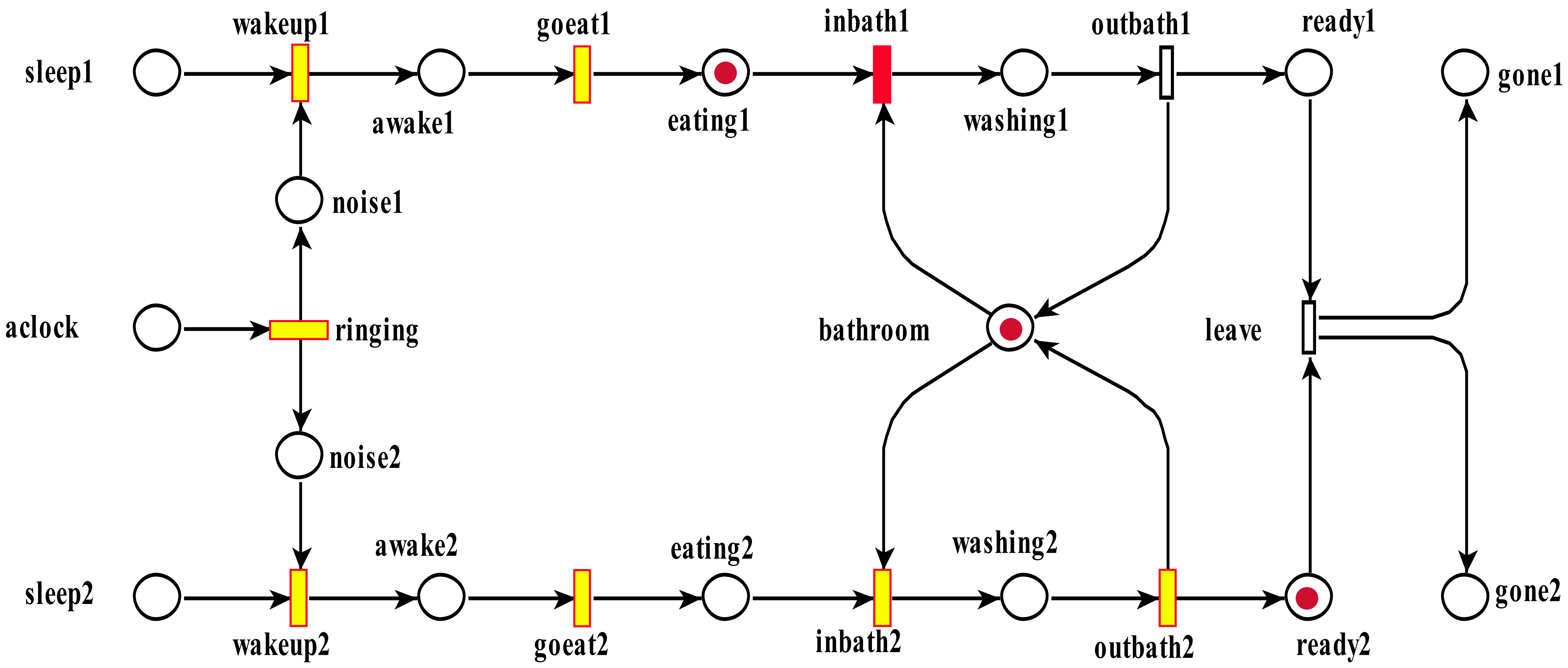
Two people waking up



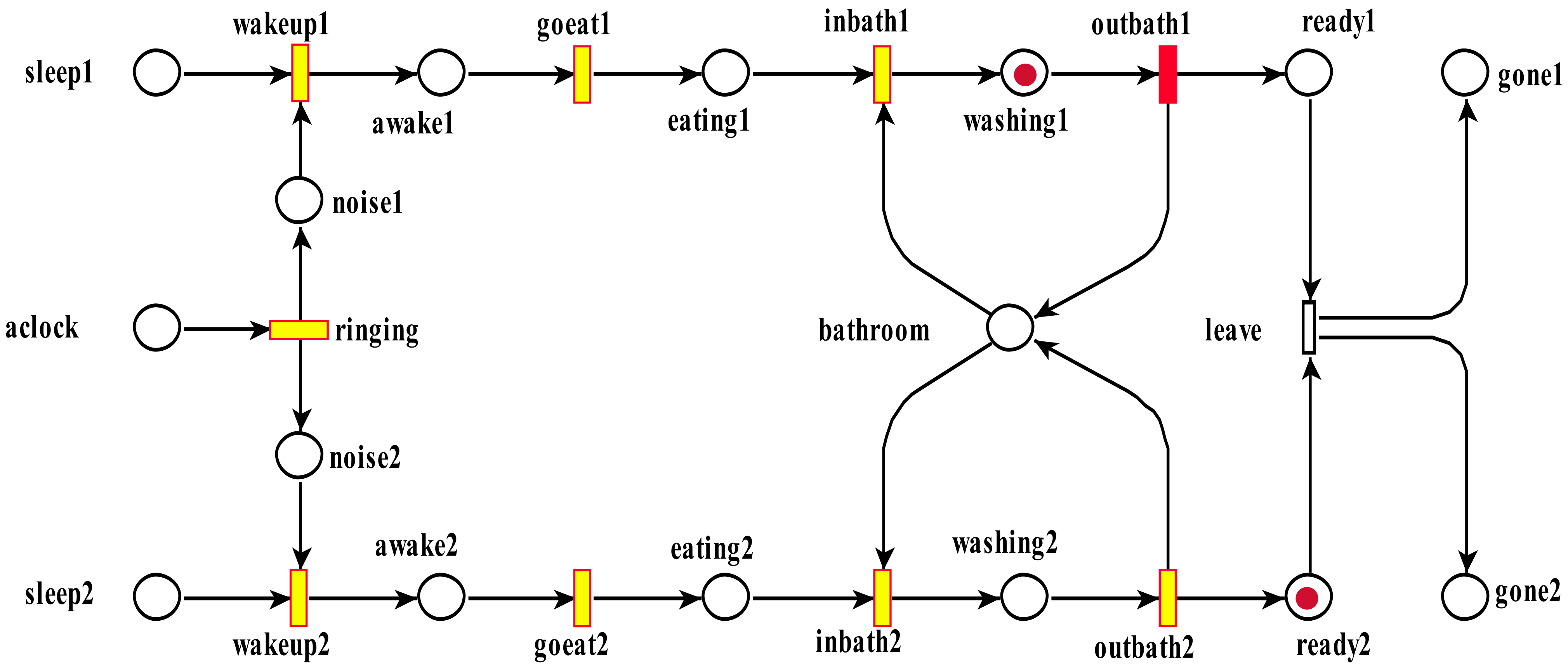
Two people waking up



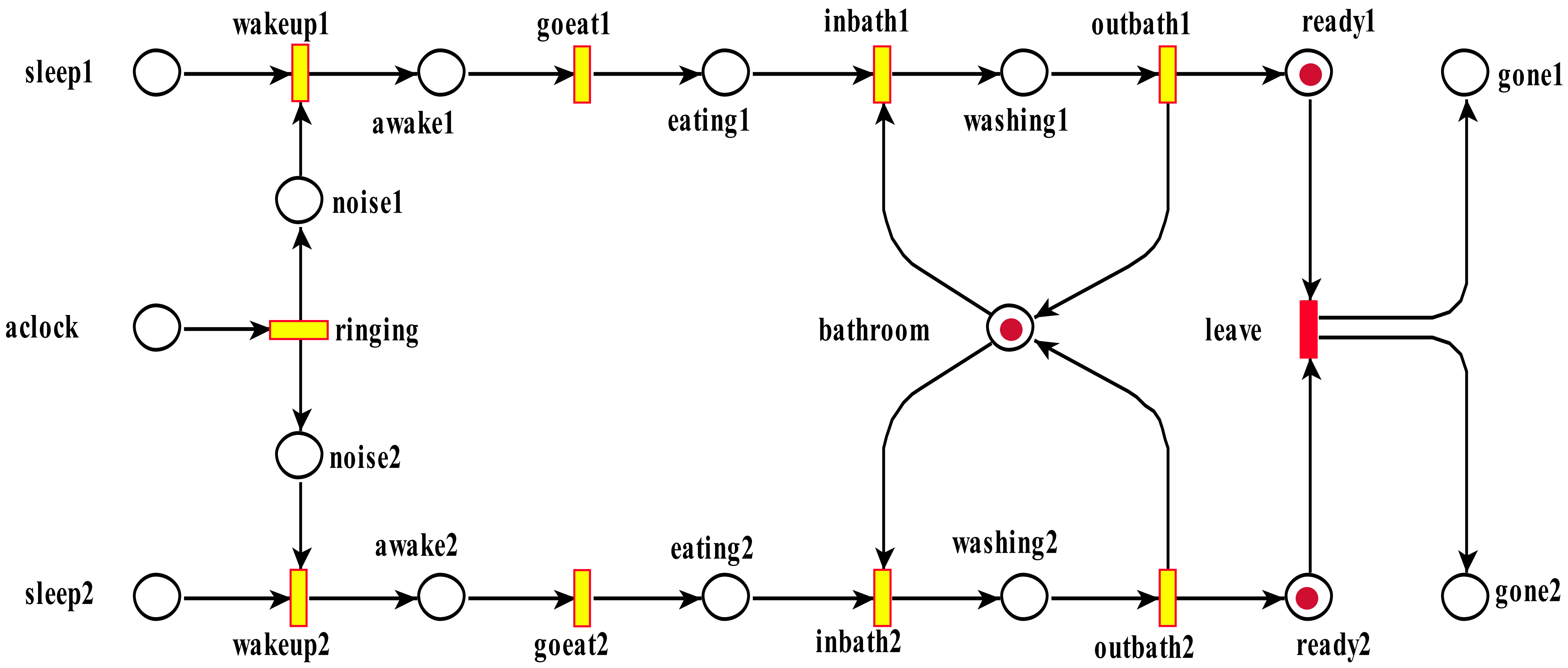
Two people waking up



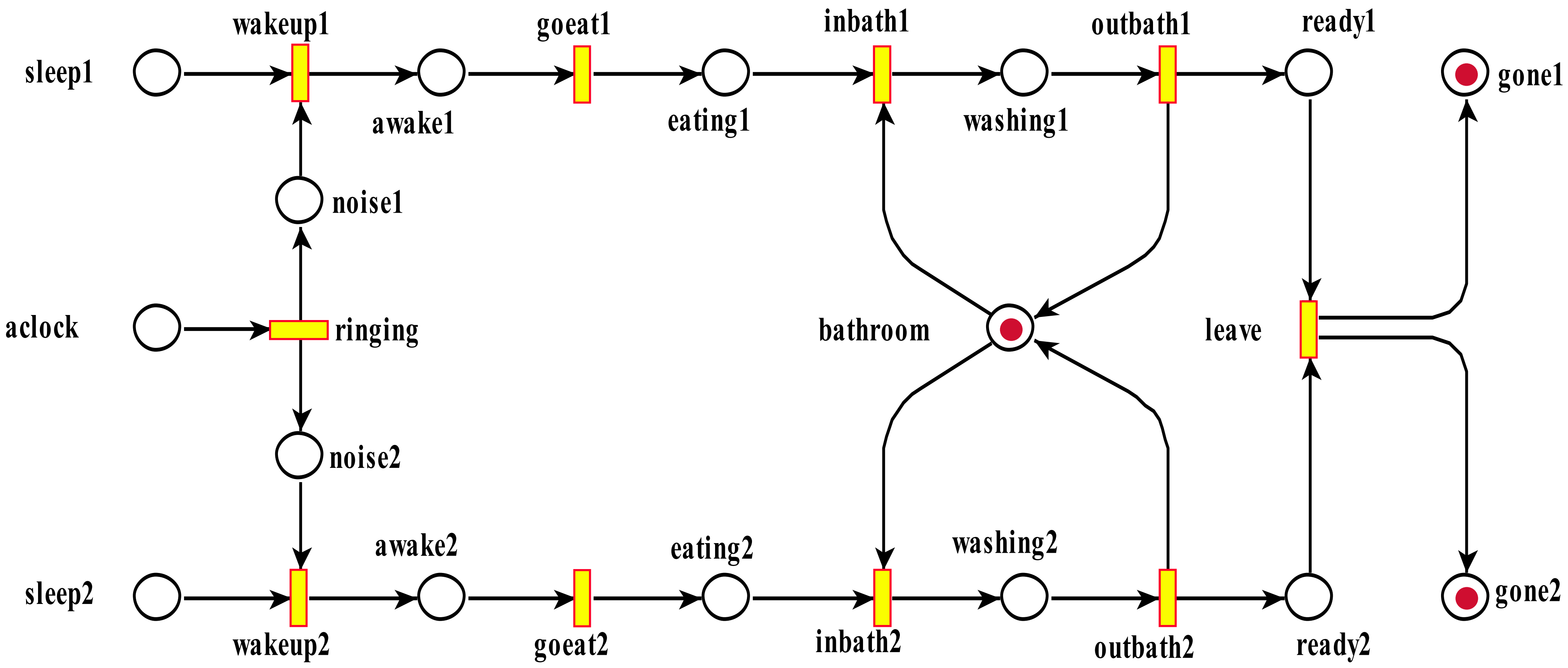
Two people waking up



Two people waking up

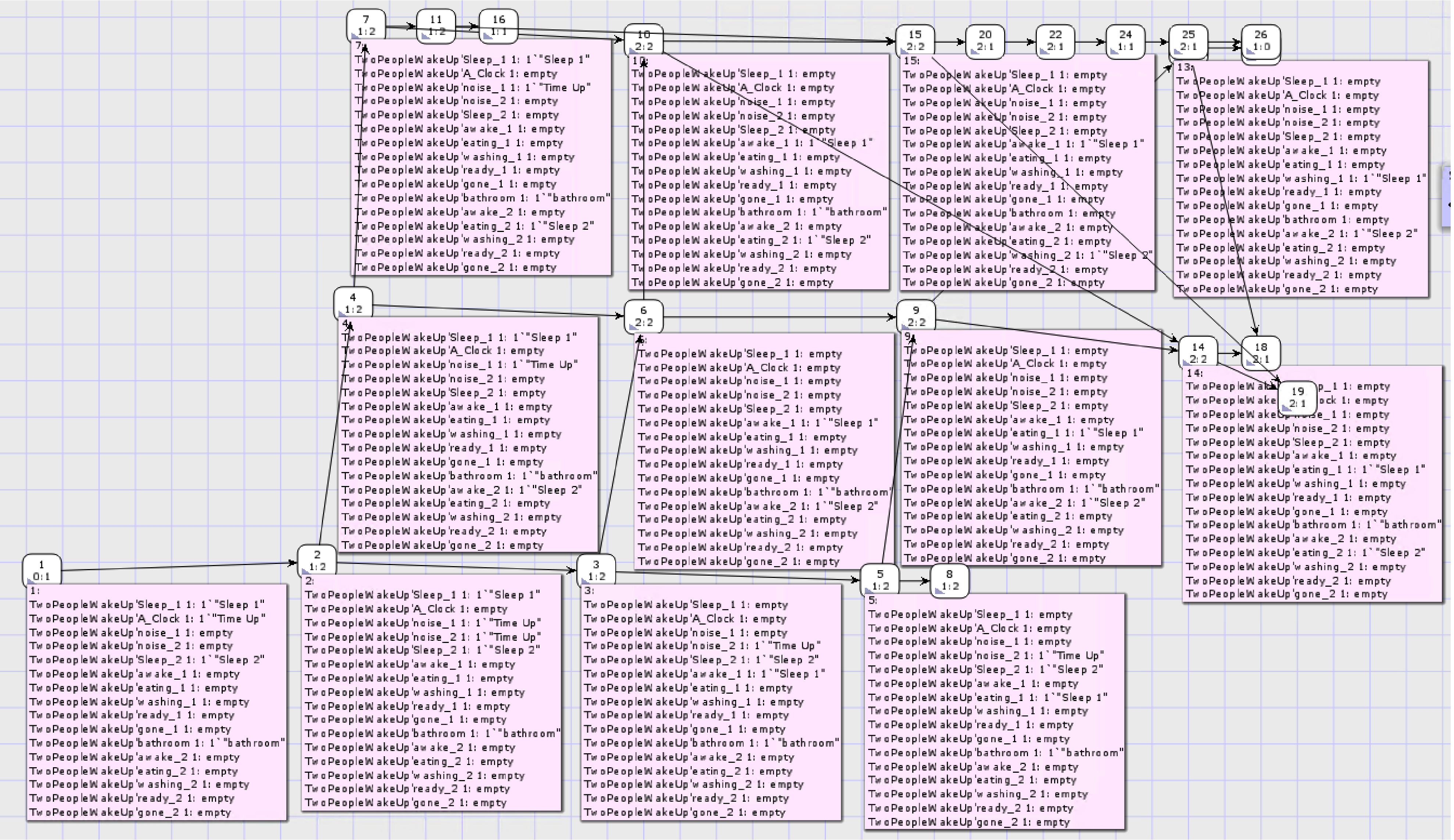


Two people waking up



Two People Wait Up CPN Simulation

State Space Analysis



```
1:
TwoPeopleWakeUp'Sleep_1 1: 1`"Sleep 1"
TwoPeopleWakeUp'A_Clock 1: 1`"Time Up"
TwoPeopleWakeUp'noise_1 1: empty
TwoPeopleWakeUp'noise_2 1: empty
TwoPeopleWakeUp'Sleep_2 1: 1`"Sleep 2"
TwoPeopleWakeUp'awake_1 1: empty
TwoPeopleWakeUp'eating_1 1: empty
TwoPeopleWakeUp'washing_1 1: empty
TwoPeopleWakeUp'ready_1 1: empty
TwoPeopleWakeUp'gone_1 1: empty
TwoPeopleWakeUp'bathroom 1: 1`"bathroom"
TwoPeopleWakeUp'awake_2 1: empty
TwoPeopleWakeUp'eating_2 1: empty
TwoPeopleWakeUp'washing_2 1: empty
TwoPeopleWakeUp'ready_2 1: empty
TwoPeopleWakeUp'gone_2 1: empty
```

```
2:
TwoPeopleWakeUp'Sleep_1 1: 1`"Sleep 1"
TwoPeopleWakeUp'A_Clock 1: empty
TwoPeopleWakeUp'noise_1 1: 1`"Time Up"
TwoPeopleWakeUp'noise_2 1: 1`"Time Up"
TwoPeopleWakeUp'Sleep_2 1: 1`"Sleep 2"
TwoPeopleWakeUp'awake_1 1: empty
TwoPeopleWakeUp'eating_1 1: empty
TwoPeopleWakeUp'washing_1 1: empty
TwoPeopleWakeUp'ready_1 1: empty
TwoPeopleWakeUp'gone_1 1: empty
TwoPeopleWakeUp'bathroom 1: 1`"bathroom"
TwoPeopleWakeUp'awake_2 1: empty
TwoPeopleWakeUp'eating_2 1: empty
TwoPeopleWakeUp'washing_2 1: empty
TwoPeopleWakeUp'ready_2 1: empty
TwoPeopleWakeUp'gone_2 1: empty
```

```
3:
TwoPeopleWakeUp'Sleep_1 1: empty
TwoPeopleWakeUp'A_Clock 1: empty
TwoPeopleWakeUp'noise_1 1: empty
TwoPeopleWakeUp'noise_2 1: 1`"Time Up"
TwoPeopleWakeUp'Sleep_2 1: 1`"Sleep 2"
TwoPeopleWakeUp'awake_1 1: 1`"Sleep 1"
TwoPeopleWakeUp'eating_1 1: empty
TwoPeopleWakeUp'washing_1 1: empty
TwoPeopleWakeUp'ready_1 1: empty
TwoPeopleWakeUp'gone_1 1: empty
TwoPeopleWakeUp'bathroom 1: 1`"bathroom"
TwoPeopleWakeUp'awake_2 1: empty
TwoPeopleWakeUp'eating_2 1: empty
TwoPeopleWakeUp'washing_2 1: empty
TwoPeopleWakeUp'ready_2 1: empty
TwoPeopleWakeUp'gone_2 1: empty
```

```
5:
TwoPeopleWakeUp'Sleep_1 1: empty
TwoPeopleWakeUp'A_Clock 1: empty
TwoPeopleWakeUp'noise_1 1: empty
TwoPeopleWakeUp'noise_2 1: 1`"Time Up"
TwoPeopleWakeUp'Sleep_2 1: 1`"Sleep 2"
TwoPeopleWakeUp'awake_1 1: empty
TwoPeopleWakeUp'eating_1 1: 1`"Sleep 1"
TwoPeopleWakeUp'washing_1 1: empty
TwoPeopleWakeUp'ready_1 1: empty
TwoPeopleWakeUp'gone_1 1: empty
TwoPeopleWakeUp'bathroom 1: 1`"bathroom"
TwoPeopleWakeUp'awake_2 1: empty
TwoPeopleWakeUp'eating_2 1: empty
TwoPeopleWakeUp'washing_2 1: empty
TwoPeopleWakeUp'ready_2 1: empty
TwoPeopleWakeUp'gone_2 1: empty
```

```
4:
TwoPeopleWakeUp'Sleep_1 1: 1`"Sleep 1"
TwoPeopleWakeUp'A_Clock 1: empty
TwoPeopleWakeUp'noise_1 1: 1`"Time Up"
TwoPeopleWakeUp'noise_2 1: empty
TwoPeopleWakeUp'Sleep_2 1: empty
TwoPeopleWakeUp'awake_1 1: empty
TwoPeopleWakeUp'eating_1 1: empty
TwoPeopleWakeUp'washing_1 1: empty
TwoPeopleWakeUp'ready_1 1: empty
TwoPeopleWakeUp'gone_1 1: empty
TwoPeopleWakeUp'bathroom 1: 1`"bathroom"
TwoPeopleWakeUp'awake_2 1: 1`"Sleep 2"
TwoPeopleWakeUp'eating_2 1: empty
TwoPeopleWakeUp'washing_2 1: empty
TwoPeopleWakeUp'ready_2 1: empty
TwoPeopleWakeUp'gone_2 1: empty
```

```
6:
TwoPeopleWakeUp'Sleep_1 1: empty
TwoPeopleWakeUp'A_Clock 1: empty
TwoPeopleWakeUp'noise_1 1: empty
TwoPeopleWakeUp'noise_2 1: empty
TwoPeopleWakeUp'Sleep_2 1: empty
TwoPeopleWakeUp'awake_1 1: 1`"Sleep 1"
TwoPeopleWakeUp'eating_1 1: empty
TwoPeopleWakeUp'washing_1 1: empty
TwoPeopleWakeUp'ready_1 1: empty
TwoPeopleWakeUp'gone_1 1: empty
TwoPeopleWakeUp'bathroom 1: 1`"bathroom"
TwoPeopleWakeUp'awake_2 1: 1`"Sleep 2"
TwoPeopleWakeUp'eating_2 1: empty
TwoPeopleWakeUp'washing_2 1: empty
TwoPeopleWakeUp'ready_2 1: empty
TwoPeopleWakeUp'gone_2 1: empty
```

```
9:
TwoPeopleWakeUp'Sleep_1 1: empty
TwoPeopleWakeUp'A_Clock 1: empty
TwoPeopleWakeUp'noise_1 1: empty
TwoPeopleWakeUp'noise_2 1: empty
TwoPeopleWakeUp'Sleep_2 1: empty
TwoPeopleWakeUp'awake_1 1: empty
TwoPeopleWakeUp'eating_1 1: 1`"Sleep 1"
TwoPeopleWakeUp'washing_1 1: empty
TwoPeopleWakeUp'ready_1 1: empty
TwoPeopleWakeUp'gone_1 1: empty
TwoPeopleWakeUp'bathroom 1: 1`"bathroom"
TwoPeopleWakeUp'awake_2 1: 1`"Sleep 2"
TwoPeopleWakeUp'eating_2 1: empty
TwoPeopleWakeUp'washing_2 1: empty
TwoPeopleWakeUp'ready_2 1: empty
TwoPeopleWakeUp'gone_2 1: empty
```

```
14:
TwoPeopleWak    p_1 1: empty
TwoPeopleWake    ock 1: empty
TwoPeopleWakeU    se_1 1: empty
TwoPeopleWakeUp'noise_2 1: empty
TwoPeopleWakeUp'Sleep_2 1: empty
TwoPeopleWakeUp'awake_1 1: empty
TwoPeopleWakeUp'eating_1 1: 1`"Sleep 1"
TwoPeopleWakeUp'washing_1 1: empty
TwoPeopleWakeUp'ready_1 1: empty
TwoPeopleWakeUp'gone_1 1: empty
TwoPeopleWakeUp'bathroom 1: 1`"bathroom"
TwoPeopleWakeUp'awake_2 1: empty
TwoPeopleWakeUp'eating_2 1: 1`"Sleep 2"
TwoPeopleWakeUp'washing_2 1: empty
TwoPeopleWakeUp'ready_2 1: empty
TwoPeopleWakeUp'gone_2 1: empty
```

```
7:
TwoPeopleWakeUp'Sleep_1 1: 1`"Sleep 1"
TwoPeopleWakeUp'A_Clock 1: empty
TwoPeopleWakeUp'noise_1 1: 1`"Time Up"
TwoPeopleWakeUp'noise_2 1: empty
TwoPeopleWakeUp'Sleep_2 1: empty
TwoPeopleWakeUp'awake_1 1: empty
TwoPeopleWakeUp'eating_1 1: empty
TwoPeopleWakeUp'washing_1 1: empty
TwoPeopleWakeUp'ready_1 1: empty
TwoPeopleWakeUp'gone_1 1: empty
TwoPeopleWakeUp'bathroom 1: 1`"bathroom"
TwoPeopleWakeUp'awake_2 1: empty
TwoPeopleWakeUp'eating_2 1: 1`"Sleep 2"
TwoPeopleWakeUp'washing_2 1: empty
TwoPeopleWakeUp'ready_2 1: empty
TwoPeopleWakeUp'gone_2 1: empty
```

```
10:
TwoPeopleWakeUp'Sleep_1 1: empty
TwoPeopleWakeUp'A_Clock 1: empty
TwoPeopleWakeUp'noise_1 1: empty
TwoPeopleWakeUp'noise_2 1: empty
TwoPeopleWakeUp'Sleep_2 1: empty
TwoPeopleWakeUp'awake_1 1: 1`"Sleep 1"
TwoPeopleWakeUp'eating_1 1: empty
TwoPeopleWakeUp'washing_1 1: empty
TwoPeopleWakeUp'ready_1 1: empty
TwoPeopleWakeUp'gone_1 1: empty
TwoPeopleWakeUp'bathroom 1: 1`"bathroom"
TwoPeopleWakeUp'awake_2 1: empty
TwoPeopleWakeUp'eating_2 1: 1`"Sleep 2"
TwoPeopleWakeUp'washing_2 1: empty
TwoPeopleWakeUp'ready_2 1: empty
TwoPeopleWakeUp'gone_2 1: empty
```

```
15:
TwoPeopleWakeUp'Sleep_1 1: empty
TwoPeopleWakeUp'A_Clock 1: empty
TwoPeopleWakeUp'noise_1 1: empty
TwoPeopleWakeUp'noise_2 1: empty
TwoPeopleWakeUp'Sleep_2 1: empty
TwoPeopleWakeUp'awake_1 1: 1`"Sleep 1"
TwoPeopleWakeUp'eating_1 1: empty
TwoPeopleWakeUp'washing_1 1: empty
TwoPeopleWakeUp'ready_1 1: empty
TwoPeopleWakeUp'gone_1 1: empty
TwoPeopleWakeUp'bathroom 1: empty
TwoPeopleWakeUp'awake_2 1: empty
TwoPeopleWakeUp'eating_2 1: empty
TwoPeopleWakeUp'washing_2 1: 1`"Sleep 2"
TwoPeopleWakeUp'ready_2 1: empty
TwoPeopleWakeUp'gone_2 1: empty
```

```
13:
TwoPeopleWakeUp'Sleep_1 1: empty
TwoPeopleWakeUp'A_Clock 1: empty
TwoPeopleWakeUp'noise_1 1: empty
TwoPeopleWakeUp'noise_2 1: empty
TwoPeopleWakeUp'Sleep_2 1: empty
TwoPeopleWakeUp'awake_1 1: empty
TwoPeopleWakeUp'eating_1 1: empty
TwoPeopleWakeUp'washing_1 1: 1`"Sleep 1"
TwoPeopleWakeUp'ready_1 1: empty
TwoPeopleWakeUp'gone_1 1: empty
TwoPeopleWakeUp'bathroom 1: empty
TwoPeopleWakeUp'awake_2 1: 1`"Sleep 2"
TwoPeopleWakeUp'eating_2 1: empty
TwoPeopleWakeUp'washing_2 1: empty
TwoPeopleWakeUp'ready_2 1: empty
TwoPeopleWakeUp'gone_2 1: empty
```

Modelling examples

You can find more examples about Petri Net applications by searching Internet

http://www.informatik.uni-hamburg.de/TGI/PetriNets