ASSIGNMENT

Petri-Net Modeling

Mathematical Modeling (CO2011)

Date report: November 29, 2021

Ho Chi Minh City University of Technology
Faculty of Computer Science and Engineering

Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Table of Contents

Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Summary Theory
Definition
Marking
Enabling
Transtition firing
Modeling
Code explanantion
Library
Abstract prototype
Class Place
Class Transition
Helper function
Main
Problem Solution
Question 1
Question 2
Question 3
Question 4
Question 5
Question 6

Summary Theory

Definition

A Petri net is a triple (P,T,F), where

1. P is a finite set of places
2. T is a finite set of transitions.
3. F $\subseteq (P * T) \bigcup (T * P)$ is a flow relation.

- Consequently, there is a one-to-one correspondence between the graphical representation of a Petri net and the triple (P, T, F).
- A Petri net system (P, T, F, m_0) consists of a Petri net (P, T, F) and a distinguished marking m_0, the inital marking.

Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Summary Theory

Marking

- A marking of a Petri net is a function $m : P \to \mathbb{N}$, assigning to each place $p \in P$ the number $m(p)$ of tokens at this place. The set M of all markings of this net is the set of all such functions.
- We can now define the concept of enabling. A transition t is enabled at marking m if every in-put place of t contains at least one token.

Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Summary Theory

Enabling

In a Petri net (P, T, F), a transition t $\in$ T is enabled at marking m: P $\rightarrow \mathbb{N}$ if and only if for all p $\in tokens, m(p) > 0$.

Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Transition firing

- For a Petri net (P, T, F), let w be the weight function and m: P $\rightarrow \mathbb{N}$ be the current marking. A transition t $\in$ T can fire if and only if it is enabled at m. The firing of t yields a new marking m': P $\rightarrow \mathbb{N}$ where for all place p $\in$ P, $m'(p) = m(p) - w((p,t)) + w((t,p))$.
- An enabled transition can fire by consuming a token from each input place and producing a token for each output place.

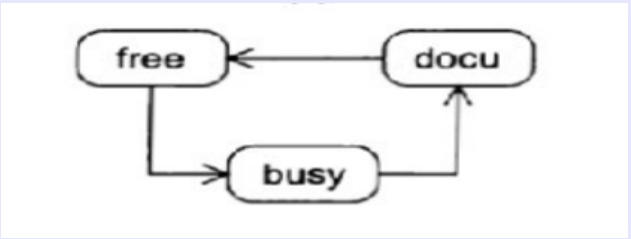
Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Modeling

In the outpatient clinic of a hospital, patients consult specialists. Each patient has an appointment with a curtain specialist. We describe the course of business around a specialist as a process model. As a formalism, we can see:

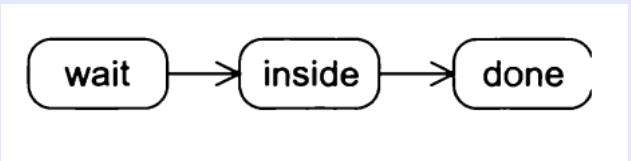
The Specialist

receives patients. At each moment, the specialist is in one of the following three states:



The Patient

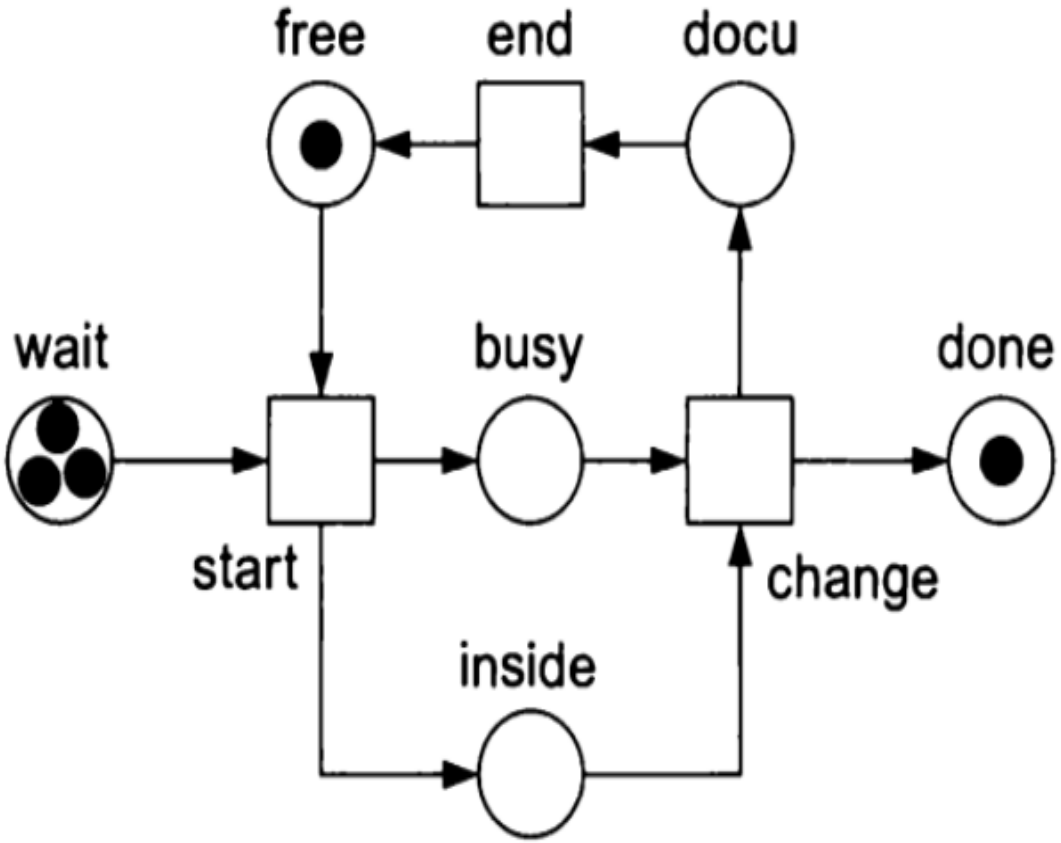
who visits a specialist is in one of the following three states. A patient goes through these states only once (per visit).



Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Modeling

By combining two states above, we have created this Petri net to describe the course of business around a specialist in this outpatient clinic of hospital X



Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Modeling

- Transition start, change, and end represents the three possible events.
- The tokens in place wait represent the waiting patients.
- A token in place inside represents a patient who is being treated or examined by the specialist.
- Tokens in place done represent patients who have been treated or examined.
- The token in place free indicates that the specialist is state-free.
- If the specialist is busy treating a patient, a token is in place busy.
- A token in place docu indicates that the specialist is documenting certain results.

Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Code explanation

Library

```
include <iostream>
include <vector>
include <windows.h>
```

- The library <**iostream**> is standard input / output streams library.
- We store the state transitions by <**vector**>.
- The library <**windows.h**> is a Windows-specific header file for the C and C++ programming languages which contains declarations for all of the functions in the Windows API.

Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Code explanation

Abstract prototype

```
//prototype
    class Transition;
    class Place;
    vector<Place*> setOfPlace;
    vector<Transition*> setOfTransition;
    void gotoXY(int column, int line);

//global variable
    bool tokenRun = true;
```

The class **Transition** and **Place** will be described after. Two vectors **setOfPlace** and **setOfTransition** are used to store the state. Boolean variable **tokenRun** is created to check if there is any change in the progress. Function **gotoXY** makes the printed model go back to **(0;0)** position and continue printing.

Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Code explanation

Class

```
class Place {
public:
    Place();
    ~Place();
    void addToken();
    void addToken(int t);
    void flowTransition();
    void setFlowTo(Transition* tr);

    unsigned int token;
    vector<Transition*> inDegree;
    vector<Transition*> outDegree;
};
```

The constructor **Place()** sets the token to zero and pushes back this **Place** to **setOfPlace**. The destructor **Place()** also sets the token to zero and clear vector **inDegree** and **outDegree**. **inDegree** is the number of transition connect to this place, and the **outDegree** is opposite. The functions **flowTransition** and **setFlowto** are described below.

Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Code explanation

Class

```
class Transition {
public:
    Transition();
    ~Transition();
    void firing();
    void setFlowTo(Place* p);
    friend void acceptFlow();

    unsigned int token;
    bool flag;
    vector<Place*> inDegree;
    vector<Place*> outDegree;
};
```

The constructor **Transition()** sets the **flag** to false, the **token** to zero and pushes back this **Trainsition** to **setOfTransition**. The destructor also reset the token to zero and clear the **inDegree** and **outDegree**. The function **firing**, **setFlowto** and **accceptFlow** are described below.

Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Code explanation

Helper function

```
void Transition::firing() {
    if (this->token == inDegree.size()) {
        this->token = 0;
        for (auto i : outDegree) i->token++;
    }
}

void Transition::setFlowTo(Place* p) {
    this->outDegree.push_back(p);
    p->inDegree.push_back(this);
}
```

We check if the token of this Place is equal to the number of its **inDegree**. Then, plus one each out place.
The function **setFlowTo** use to set up directed connection from source to destination

Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Code explanation

Helper function

```
void acceptFlow() {
  for (auto j : setOfTransition) {
      bool check = true;
      for (auto i : j->inDegree) {
          if (i->token == 0) {
              check = false;
              break;
          }
      }
      j->flag = check;
   }
}
```

//prototype

```
void PRINT(int free, int wait, int busy, int inside, int done, int
docu);
```


Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Code explanation

Main

```
Place* free = new Place();
Place* wait = new Place();
Transition* start = new Transition();
free->setFlowTo(start);
wait->setFlowTo(start);
Place* busy = new Place();
Place* inside = new Place();
start->setFlowTo(busy);
start->setFlowTo(inside);
```

Initialize place "free", "wait", transition "start", place "busy" and "inside".

Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Code explanation

Main

```
Transition* change = new Transition();
busy->setFlowTo(change);
inside->setFlowTo(change);
Place* done = new Place();
Place* docu = new Place();
change->setFlowTo(done);
change->setFlowTo(docu);
```

Initialize transition "change", place "done" and "docu".

Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Code explanation

Main

```
Transition* end = new Transition();
docu->setFlowTo(end);
end->setFlowTo(free);
free->addToken(1);
wait->addToken(3);
done->addToken(1);
```

Initialize "end" transition.
Add the token from input given.

Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Code explanation

Main

```
do {
    tokenRun = false;
    PRINT(free->token, wait->token, busy->token,
            inside->token, done->token, docu->token);
    acceptFlow();
    for (auto i : setOfPlace) i->flowTransition();
    Sleep(1000);
    for (auto i : setOfTransition) i->firing();
    Sleep(2000);
} while (tokenRun);
```

We must set the default after this process to **tokenRun**, there are nothing change. Then, we set up the flag of all transition and go through each place and flow, firing each transition. If there exist any change, this model will continue.

Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Problem Solution

Question 1. Given the Petri net N_S modeling the state of the specialist:

- **Question 1a.** In the Petri net N_S:
  1. State: free, busy, docu.
  2. Transition: start, change, end.
- **Question 1b.** A transition system could represented in the marking of the net as a triple (x, y, z) with x specifying the number of tokens in place free, y in place busy, and z in place docu:

  (1,0,0) -> (0,1,0) -> (0,0,1) -> (1,0,0) -> ...

Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Problem Solution

Question 2. Define N_{Pa} as the Petri net modeling the state of patients:

1. **Question a.** The possible meaning of a token in state $inside$ of the net: The state $inside$ contains one patient treated by the specialist.
2. **Question b.** construct the Petri net N_{Pa} asuming that there are five patients in state wait, no patient in state inside, and one patient is in state done:

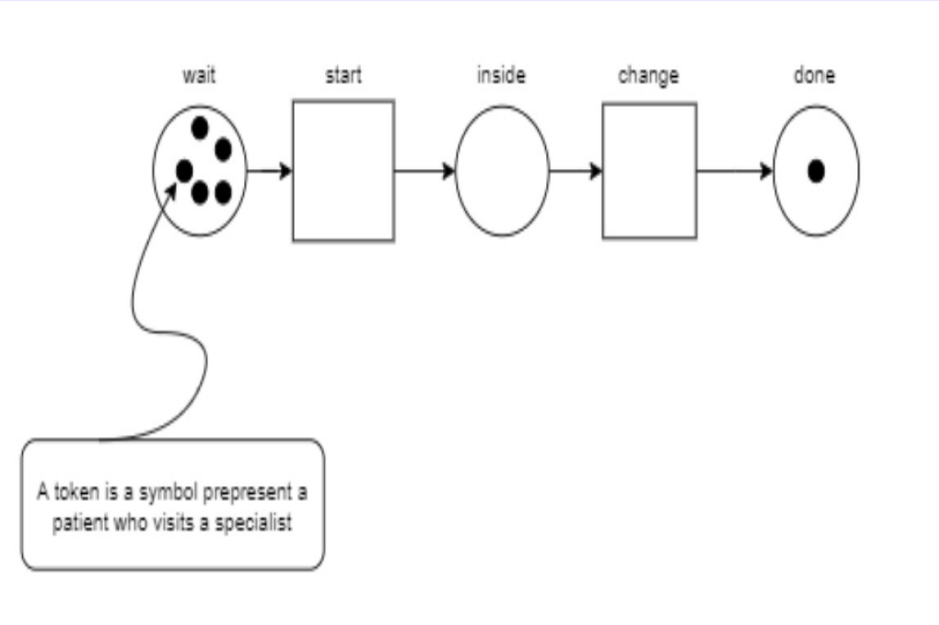


Figure: A Petri net modeling state of patients.

Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Problem Solution

Question 3. Determine the superimposed (merged) Petri net N_S and N_{Pa} allow specialist to treat patients, assume there are four patients are waiting to see specialists/doctors, one patient in state done, and one doctor in state free:

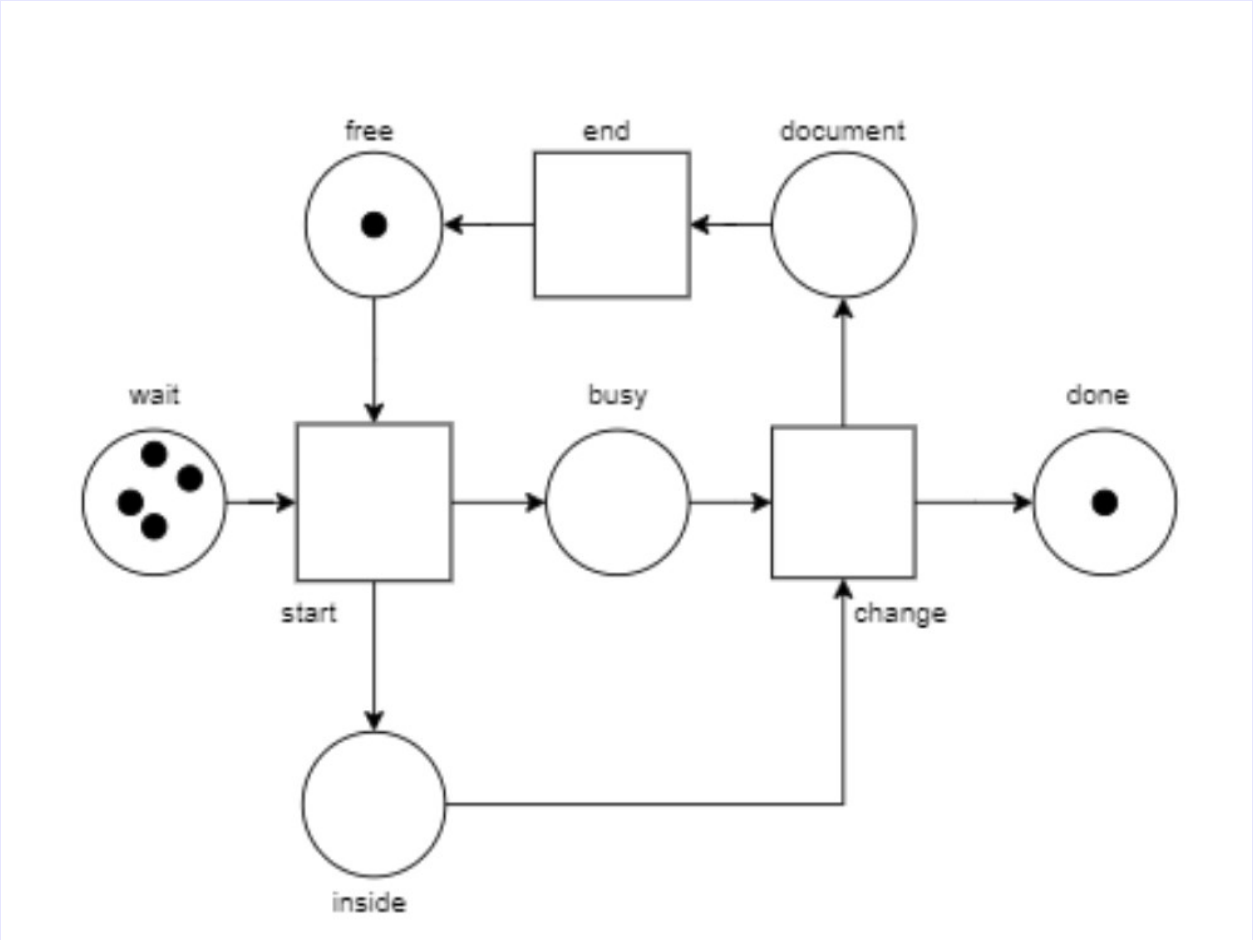


Figure: A merged Petri net modeling state of patients and specialists.

Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Problem Solution

Question 4. Consider an initial marking $M_0 = [3.wait, done, free]$ in the grand net N.

The marking to be reachable from M_0 by firing one transition once is $M = [2.wait, inside, done, busy]$. Because only transition *start* is be firing once, so one token in *wait* and *free* are changed to in *inside* and *busy*.

Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Problem Solution

Question 5.

The superimposed Petri net N is a **deadlock free** Petri net if the number of tokens in *busy* is different from another *inside*.
Explain:

- Each time a token in *free* is done a process, the number of tokens in *wait* reduce 1 and in *done* increase 1.
- The number of tokens in *wait* is finite, until in *wait* will not contain any token, so firing transition *start* is a impossible problem, causing some tokens in *free* is not be free of fire. Leading to transition as start, change, end could not be firing.

Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Problem Solution

Question 6. Propose a similar Petri net with two specialists already for treating patients, with explicitly explained construction.

In reality, with more than one specialists, the process can take place in parallel. But in this situation, we propose that there only has a treating place *busy* for specialists and *inside* for patients. So a process will take place in sequence.
According to initial marking:

- We have two specialists in *free* place. At each time, a place flow one token to every transition which connected to it, without loss of generality, assume that specialist with smallest IDs go treating first.
- Similar to patients, a patient who arrive first will get smaller IDs value than. In these arguments, we also assume that the order of service for patients with smaller IDs takes precedence.

Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Problem Solution

Question 6. Propose a similar Petri net with two specialists already for treating patients, with explicitly explained construction.

In each such area, there will be 3 cases corresponding to the paths defined above:

1. **Case 1:** Where there is no token, this area is not taking any action
2. **Case 2:** In which there is only 1 token. At this point, there will be 2 more sub-cases: that token represents the patient or that token represents the doctor. For convenience, we will sign in the area: if the token represents the patient - denoted Px or the token represents the doctor - denoted Sy.
3. **Case 3:** We do not include at the ticket counter because there can be as many patients as possible. Including 2 tokens. The case of having 2 tokens of the same doctor only occurs with the initial initialization marking in the free area because subsequent markings are sequential (since each subsequent zone can only contain a maximum of 1 doctor and 1 patient).

Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Problem Solution

Question 6. Propose a similar Petri net with two specialists already for treating patients, with explicitly explained construction.

Consider an initial marking $M_0 = [3.wait, done, 2.free]$. A process will work as follow:

Marking	Place					
	free	busy	docu	wait	inside	done
$M_0 = [3.wait, done, 2.free]$	{S0, S1}	{}	{}	{P1,P2,P3}	{}	{P0}
$M_1 = [2.wait, done, free, busy, inside]$	{S1}	{S0}	{}	{P2,P3}	{P1}	{P0}
$M_2 = [wait, 2.done, busy, inside, document]$	{}	{S1}	{S0}	{P3}	{P2}	{P0, P1}
$M_3 = [wait, 3.done, document, free]$	{S0}	{}	{S1}	{P3}	{}	{P0,P1,P2}
$M_4 = [3.done, free, busy, inside]$	{S1}	{S0}	{}	{}	{P3}	{P0,P1,P2}
$M_5 = [4.done, free, document]$	{S1}	{}	{S0}	{}	{}	{P0,P1,P2,P3}
$M_6 = [4.done, 2.free]$	{S0,S1}	{}	{}	{}	{}	{P0,P1,P2,P3}

Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Problem Solution

Question 6. Propose a similar Petri net with two specialists already for treating patients, with explicitly explained construction.

Definition of table

- Example in free*, character '*' mean this stage is a final stage.
- Each cell is a set of tokens that its place.
- The columns highlighted in blue represents states of the doctors/specialists.
- The columns highlighted in orange represents states of the patients.

Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

Problem Solution

Question 6. Propose a similar Petri net with two specialists already for treating patients, with explicitly explained construction.

Explain of table

- The element has the smallest index in a set which has priority to leave the set first.
- Transferring the state: the element of the set move next to the column which has same color in the direction from left to right in cycle rule, which mean the element in last column move to first column.

Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú

THANKS FOR READING

Lý Gia Huy
Lâm Nhật Tân
Du Thành Đạt
Đoàn Trần Cao Trí
Nguyễn Đắc Hoàng Phú