

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH



MÔ HÌNH HOÁ TOÁN HỌC

Báo cáo Bài tập lớn

Mạng Petri Net

GVHD:	Nguyễn An Khương	
SV:	Đặng Vũ Sĩ Đan	2020017
	Khổng Thị Bích Ngọc	1727029
	Nguyễn Văn Bảo Khánh	2020039
	Nguyễn Đức Thuận	2020092
	Trần Thanh Tuấn	2020114

TP. HỒ CHÍ MINH, THÁNG 11/2021

Mục lục

1 Chuẩn bị kiến thức	4
1.1 Mạng Petri - Background	4
1.1.1 Formal definitions	4
1.1.1.a Transition system	4
1.1.1.b Multi-set	5
1.1.1.c Petri Net	5
1.1.2 On Enabled transition and Marking changes	8
1.1.3 Important usages of Petri net	9
1.1.3.a Business Process (BP)	9
1.1.3.b Information Systems and Modeling Business Processes	10
1.2 Mạng Petri - Behavior	12
1.2.1 From Firing, Reachability to Labeled Petri net	12
1.2.2 Representing Petri Nets as Special Transition Systems	15
1.3 Mạng Petri - Cấu trúc và các vấn đề cơ bản	17
1.3.1 Causality, Concurrency and Synchronization	17
1.3.1.a Causality	17
1.3.1.b Concurrency and Synchronization	17
1.3.2 Effect of Concurrency	19
1.4 SUMMARY and REVIEWED PROBLEMS	19
1.4.1 Problem 5.1	19
1.4.2 Problem 5.2	19
1.4.3 Problem 5.3	21
1.5 Mạng Petri - Bài tập lớn về mô hình hóa	22
1.5.1 Các ý niệm cơ bản để mô hình hóa với mạng Petri	22
1.5.2 Tính động của mạng Petri thông qua các tính chất đặc biệt	23
1.5.3 Các vấn đề khi mô hình hóa với mạng Petri	24
2 Trả lời câu hỏi/yêu cầu Bài tập lớn	25
2.1 Câu hỏi 1	25
2.1.1 Câu hỏi 1a	25
2.1.2 Câu hỏi 1b	26
2.1.2.a Câu hỏi 1b (i) Each place cannot contain more than one token in any marking	27
2.1.2.b Câu hỏi 1b (ii) Each place may contain any natural number of tokens in any marking	27
2.2 Câu hỏi 2	27
2.2.1 Câu hỏi 2a	28
2.2.2 Câu hỏi 2b	28
2.3 Câu hỏi 3	29
2.4 Câu hỏi 4	29
2.5 Câu hỏi 5	30
2.6 Câu hỏi 6	31
2.7 Câu hỏi 7	34
2.7.1 Petri Net của Specialist N_S - tệp specialist.py	37
2.7.2 Petri Net của Patient N_{Pa}	40



2.7.3	Petri Net $N = N_S \oplus N_{Pa}$	42
-------	---	----

Danh mục hình ảnh

1.1	Transition system với một initial state và một final state	4
1.2	Petri net cho quy trình của máy X-ray	6
1.3	Ba marking khác nhau của một Petri net, mô hình hóa quy trình của một máy X-ray	7
1.4	A marked Petri net with one initial token	8
1.5	A simple Petri net, with only two transitions	9
1.6	A Petri net model of a business process of an X-ray machine	10
1.7	An improved Petri net for the business process of an X-ray machine	11
1.8	The marking of the improved Petri net for the working process of an X-ray room after transition enter has fired.	11
1.9	A labeled marked Petri net	13
1.10	A Petri net system and its reachability graph	15
1.11	Reachability graph TS của Petri Net trong hình 1.9	16
1.12	Causality trong mạng Petri N	17
1.13	Concurrency trong mạng Petri N	18
1.14	Synchronization trong mạng Petri N	18
1.15	Reachability graph của mạng N như trong hình xx	18
1.16	A Petri net with small numbers of places and transitions	19
1.17	Mạng Petri sau khi xóa p_1	21
1.18	Mạng Petri sau khi xóa p_3	21
1.19	A Petri net allows concurrency	22
1.20	Một mạng Petri bị chặn	23
1.21	Một mạng Petri deadlock free	24
2.1	The Petri net of the specialist's state	25
2.2	State busy	26
2.3	State docu	26
2.4	A Petri net modeling states of patients	28
2.5	A Petri net modeling states of patients with a token in state insidet	28
2.6	A Petri net modeling states of patients with a token in state inside	28
2.7	A Petri net modeling states of patients with five patients in state wait	29
2.8	Mạng Petri sau khi hợp nhất	30
2.10	Reachability graph của mạng gộp N	31
2.11	Các trạng thái của bác sĩ và bệnh nhân	32
2.12	Mạng Petri cho phòng khám với 2 bác sĩ	33
2.13	Mạng Petri cho phòng khám với 2 bác sĩ	33
2.14	Mạng Petri cho phòng khám với 2 bác sĩ	34
2.15	Kết quả hiện thực Petri Net của Specialist với số lần firing là 3	39
2.16	Kết quả hiện thực Petri Net của Specialist với số lần firing là 7	40
2.17	Kết quả hiện thực Petri Net của Patient với marking ban đầu là 3,0,0	42
2.18	Kết quả hiện thực Petri Net của Patient với marking ban đầu là 0,0,2	42
2.19	Kết quả hiện thực Petri Net của Patient với số bệnh nhân ở place wait là 12	44
2.20	Kết quả hiện thực Petri Net của Patient với số bệnh nhân ở place wait là 3	45

1 Chuẩn bị kiến thức

Phần này nhóm sẽ trình bày lại các kiến thức về Petri Net, các ví dụ, và làm các bài tập (Exercises), Practice được cung cấp trong tài liệu "PETRI_NETWORKS Version 0.2" nhằm nắm vững các kiến thức để trả lời phần câu hỏi của phần Assignment.

1.1 Mạng Petri - Background

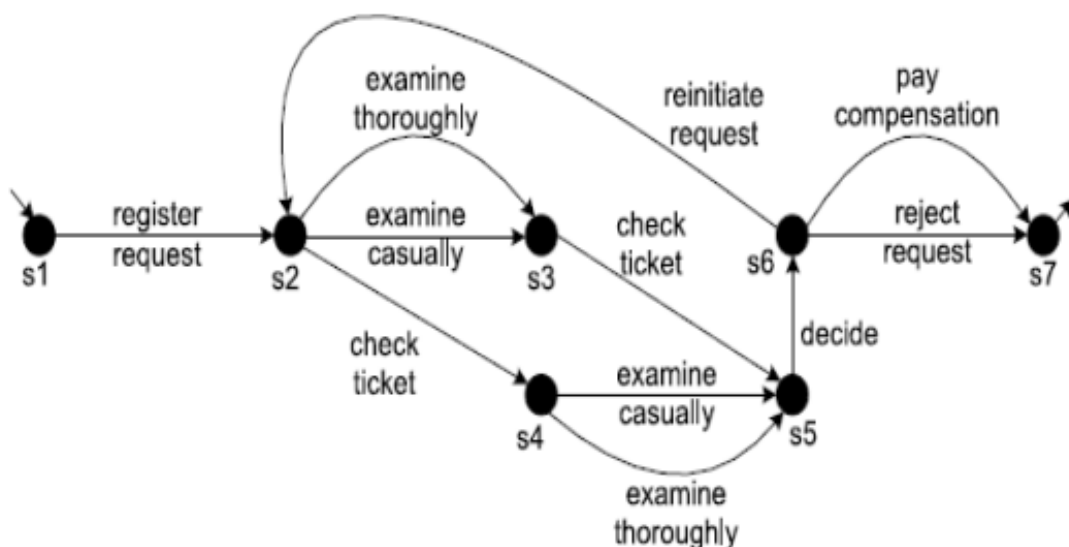
1.1.1 Formal definitions

1.1.1.a Transition system

Definition 1.1 (Transition system hay State transition system)

Một transition system là một bộ ba $TS = (S, A, T)$ với S là tập hợp các states, $A \subseteq \Lambda$ là tập hợp activities (hay actions), và $T \subseteq S \times A \times S$ là tập hợp các transitions.

- $S^{start} \subseteq S$ là tập chứa các initial states.
 $S^{end} \subseteq S$ là tập chứa các final states.
- WHY transition systems?
Mục tiêu của **process model** là để quyết định các hoạt động (activities) nào cần được thực hiện và theo thứ tự nào. Các hoạt động có thể diễn ra tuần tự hay đồng thời, và việc một hoạt động lặp đi lặp lại cũng có thể.
- BEHAVIOR của một transition system: được thể hiện qua cấu trúc mạng của nó. Quá trình chuyển đổi bắt đầu từ một trong các initial states, và mỗi đường dẫn (path) trong đồ thị bắt đầu từ các state này tương ứng với một trình tự thực thi.
* Một đường dẫn là *terminates successfully* nếu nó kết thúc ở một final state.
* Một đường dẫn là *deadlocks* nếu nó kết thúc ở một non-final state mà không có bất kì outgoing transition nào.



Hình 1.1: Transition system với một initial state và một final state

EXAMPLE 1.1.

Observe a transition system in Figure 1.1 we see the state space $S = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$. Here $S^{start} = \{s_1\}$, $S^{end} = \{s_7\}$. Could the reader fill in the set of activities $A = \{\text{register request, examine thoroughly, examine casually, ...}\}$, and determine the set $T = \{(s_1, \text{register request}, s_2), (s_2, \text{examine casually}, s_3), (s_2, \text{examine thoroughly}, s_3), (s_2, \text{checkticket}, s_4), (s_3, \text{check ticket}, s_5), (s_4, \text{examine casually}, s_5), (s_4, \text{examine thoroughly}, s_5), (s_5, \text{decide}, s_6), (s_6, \text{reinitiate request}, s_2), (s_6, \text{reject request}, s_7), (s_6, \text{pay compensation}, s_7)\}$.

Trả lời: Từ transition system trên hình 1.1, ta có thể xác định:

- $A = \{\text{register request, examine thoroughly, examine casually, check ticket, reinitiate request, decide, pay compensation, reject request}\}$
- $T = S \times A \times S = \{(s_1, \text{register request}, s_2), (s_2, \text{examine casually}, s_3), (s_2, \text{examine thoroughly}, s_3), (s_2, \text{checkticket}, s_4), (s_3, \text{check ticket}, s_5), (s_4, \text{examine casually}, s_5), (s_4, \text{examine thoroughly}, s_5), (s_5, \text{decide}, s_6), (s_6, \text{reinitiate request}, s_2), (s_6, \text{reject request}, s_7), (s_6, \text{pay compensation}, s_7)\}$

1.1.1.b Multi-set

On the set of all multisets M over a domain D

Với một miền hữu hạn $D = x_1, x_2, \dots, x_k$, ánh xạ $X : D \rightarrow \mathbb{N}$ xác định một multi-set trên D : với mỗi $x \in D$, $X(x) = m$ biểu thị số lần x xuất hiện trong multi-set.

Chúng ta có thể dùng multiplicative format cho multi-set M như sau:

$$M = \{x_1^{m_1}, x_2^{m_2}, \dots, x_k^{m_k}\}$$

và do đó M có thể xác định bằng danh sách $[m_1, m_2, \dots, m_k]$. Với tần số $m_i \geq 0$ có nghĩa là x_i xuất hiện m_i lần trong M , còn $m_i = 0$ có nghĩa x_i không xuất hiện trong M , và $\text{support}(M)$ gồm các phần tử khác nhau trong M .

EXAMPLE 1.2.

Một multi-set giống như một tập hợp trong đó mỗi phần tử có thể xuất hiện nhiều lần, và thứ tự xuất hiện không thành vấn đề.

Domain $D = \{a, b, c, d, e\}$, $k = |D| = 5$.

Multi-set với $n = 9$: $M = [a, b, b, c, c, c, d, d, e] = \{a, b^2, c^3, d^2, e\} \equiv [1, 2, 3, 2, 1]$.

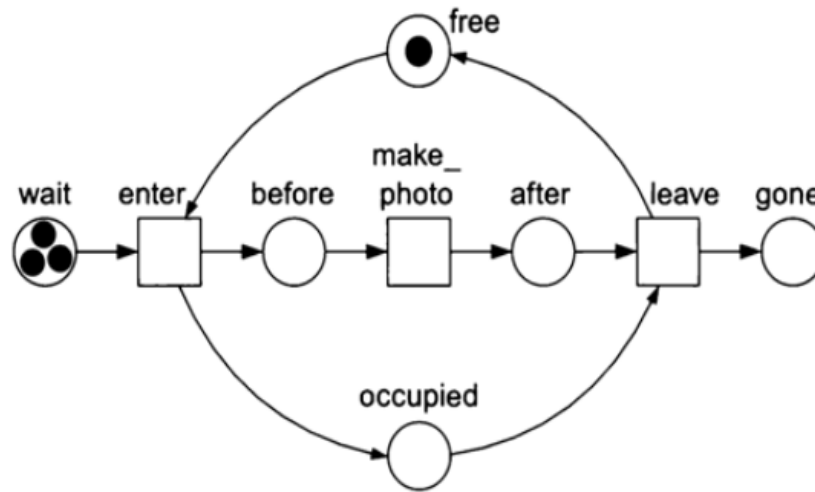
1.1.1.c Petri Net

Definition 1.2 (Petri Net N là một đồ thị có hướng phân đôi của places và transitions)

Một Petri net là bộ ba $N = (P, T, F)$ với P là tập hữu hạn các places, T là tập hữu hạn các transitions sao cho $P \cap T = \emptyset$ và $S \subseteq (P \times T) \cup (T \times P)$ là tập hợp các cung có hướng gọi là flow relation.

1. Một token là một transition node đặc biệt, hiển thị bằng chấm đen. Các token thường biểu thị yếu tố của thể giới thực. Places có thể chứa token, còn transition thì không.
2. Một transition là enabled nếu mỗi input places của nó chứa ít nhất một token. Một enabled transition có thể fire, bằng cách tiêu thụ một token từ mỗi input place của nó và tạo ra ít nhất một token cho mỗi output place.
3. Một marking là sự phân phối token giữa các places. Một marking của net N là một hàm $m : P \rightarrow \mathbb{N}$, gán cho mỗi place $p \in P$ số lượng token $m(p)$ tại place này. Ta kí hiệu $M = m(p)$, và M là một multi-set.
4. Một marked Petri net là một cặp (N, M) , trong đó $N = (P, T, F)$ là một Petri net với M là multi-set trên P biểu thị cho marking của mạng.

EXAMPLE 1.3.



Hình 1.2: Petri net cho quy trình của máy X-ray

The Petri net in figure 1.2 has three transitions (drawn as \square): $T = \{enter, make - photo, leave\}$.

- Với marking của mạng như trong hình, ta thấy transition *enter* là **enabled** do hai input place của nó là place *wait* và place *free* đều có token. Còn transition *make-photo* là không **enabled** do input place của nó là *before* không chứa token nào.
- List the places P : $P = \{wait, free, before, after, occupied, gone\}$
- Give the marking of the Petri Net in figure 1.2: Với Petri Net trong hình, ta thấy chỉ có place *wait* và place *free* chứa token. Và $m(wait) = 3$, $m(free) = 1$, do đó marking $M = [wait^3, free]$.
- Determine fully the flow relation $F \subseteq (P \times T) \cup (T \times P)$: $F = \{(wait, enter), (free, enter), (before, make - photo), (after, leave), (occupied, leave), (enter, before), (enter, occupied), (make - photo, after), (leave, free), (leave, gone)\}$

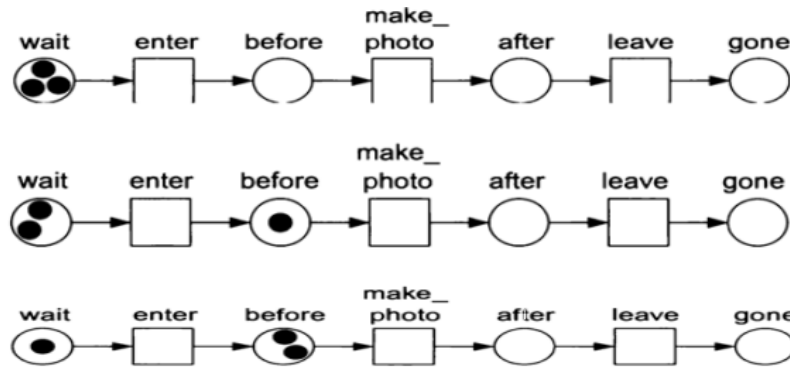
ELUCIDATION

1. PLACES: khi biểu diễn một Petri Net bằng đồ thị, mỗi place được thể hiện bằng hình tròn hoặc ellipse. Place luôn mô hình hóa một thành phần thụ động: place có thể chứa, tích trữ hoặc hiện thị điều gì đó. Một place có các trạng thái (states) rời rạc.
2. TRANSITIONS: bằng đồ thị, mỗi transition được thể hiện bằng hình chữ nhật hoặc hình vuông. Place luôn mô hình hóa một thành phần chủ động: transition có thể tạo ra things/token, tiêu thụ, vận chuyển hoặc thay đổi chúng. Mỗi lần firing của transition, các token được phân bổ lại trên các place, từ đó xây dựng tính dynamic của Petri Net.
3. ARCS: place và transition được nối với nhau bằng các directed arc, trên đồ thị, biểu diễn bằng mũi tên. Arc thể hiện mối quan hệ danh nghĩa giữa các thành phần như *logical connections*, hay *access rights*.

EXAMPLE 1.4.

Figure 1.3 shows a Petri net with three different markings.

- Find the places P : ta thấy trên hình có 4 place là wait, before, after, gone (được biểu diễn bằng hình tròn). Do đó $P = \{wait, before, after, gone\}$.
- Give the transitions T : ta thấy trên hình có 3 transition là enter, make-photo, leave (được biểu diễn bằng hình vuông). Do đó $T = \{enter, make-photo, leave\}$.
- Write down completely three different markings in format of lists or tables:
 - Ta thấy ở hình a, place wait chứa 3 token, các place còn lại không có token nào. Do đó $M_0 = [wait^3]$
 - Tại hình b, transition enter fired, tiêu thụ 1 token ở place wait và sinh ra một token ở place before, do đó $M_1 = [wait^2, before]$
 - Tại hình c, transition enter tiếp tục fired, tiêu thụ 1 token ở place wait và sinh ra thêm một token ở place before, do đó $M_2 = [wait, before^2]$



The first three markings in a process of the X-ray machine

- a) [top, transition enter not fired]; b) [middle, transition enter fired]; and
c) [down, transition enter has fired again]

Hình 1.3: Ba marking khác nhau của một Petri net, mô hình hóa quy trình của một máy X-ray

Definition 1.3 (Input is place, output is transition)

Cho $N = (P, T, F)$ là một Petri net. Mỗi phần tử $P \cup T$ gọi là nodes.

- Một node x là *input node* của node y nếu và chỉ nếu có một cung có hướng từ x đến y tức là $(x, y) \in F$. Node x là *output node* của y nếu và chỉ nếu $(y, x) \in F$.
- Với bất kỳ $x \in P \cup T$, ký hiệu $\bullet x = \{y | (y, x) \in F\}$ gọi là preset của x và $x^\bullet = \{y | (x, y) \in F\}$ gọi là postset x .
- Với bất kỳ tập hợp X , định nghĩa X^* là tập hợp các chuỗi chứa phần tử của X , tức là với bất kỳ $n \in \mathbb{N}$ và $x_1, x_2, \dots, x_n \in X$: $(x_1, x_2, \dots, x_n) \in X^*$.

Câu hỏi: cho biết $\bullet c1, c5^\bullet$ của hình 1.4 bên dưới.

Trả lời:

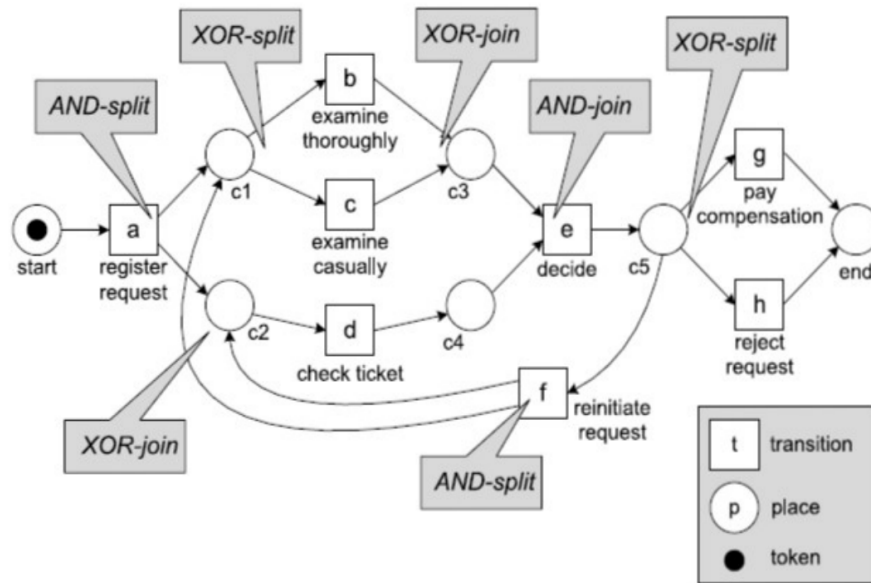
- $\bullet c1$ = register requests, reinitiate request
- $c5^\bullet$ = reinitiate request, pay compensation, reject request

1.1.2 On Enabled transition and Marking changes

EXAMPLE 1.5.

Mạng Petri được đánh dấu trong Hình 1.4 có marking với chỉ một token tại node **start**.

- Do đó, transition **a** enabled tại marking [start].
 - Firing **a** thu được kết quả marking [c1, c2]: một token được tiêu thụ và hai token được sinh ra.
- Tại marking [c1, c2] transition **a** không còn enabled. Tuy nhiên, transition **b**, **c** và **d** trở thành enabled.



Hình 1.4: A marked Petri net with one initial token

- Từ marking [c1, c2], firing **b** thu được kết quả là marking [c2, c3].
Tại đây, **d** vẫn enabled, nhưng **b** và **c** thì không còn nữa.
Do cấu trúc vòng lặp liên quan đến transition **f** nên có vô số trình tự firing bắt đầu bởi [start] và kết thúc bởi [end].
- Bây giờ với nhiều token tại thời điểm bắt đầu, giả sử initial marking là: [start⁵].
Firing **a** ta được kết quả là [start⁴, c1, c2]. Tại marking này **a** vẫn enabled.
Firing **a** lần nữa ta thu được marking [start³, c1², c2²].
Transition **a** có thể fire 5 lần liên tiếp dẫn đến marking [c1⁵, c2⁵]
- Lưu ý rằng sau lần fire **a** đầu tiên thì **a**, **b**, **c** và **d** đều enabled và có thể fire song song.

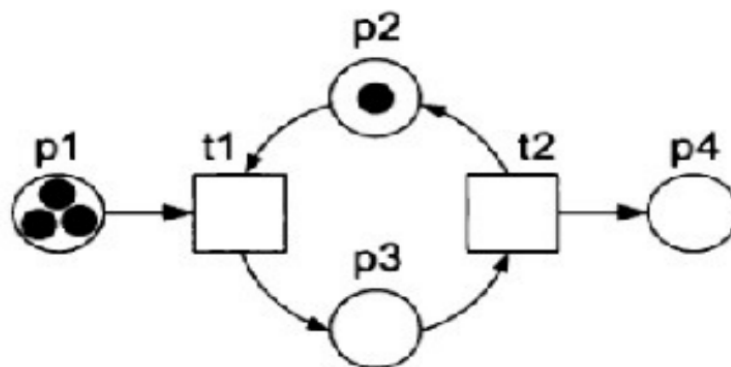
PRACTICE 1.1 Consider the Petri net in figure 1.5 below.

1. Define the net formally as a triple (P, T, F) .

- $P = \{p1, p2, p3, p4\}$
- $T = \{t1, t2\}$
- $F = \{(p1, t1), (p2, t1), (p3, t2), (t1, p3), (t2, p2), (t2, p4)\}$

2. List presets and postsets for each transition.

- Preset và postset của t1: $\bullet t1 = \{p1, p2\}$, $t1 \bullet = \{p3\}$.



Hình 1.5: A simple Petri net, with only two transitions

- Preset và postset của $t2$: ${}^{\bullet}t2 = \{p3\}$, $t2^{\bullet} = \{p2, p4\}$.
3. Determine the marking of this net.
 $M = [p1^3, p2]$ do place $p1$ có 3 token, place $p2$ có 1 token, các place còn lại không có token nào.
 4. Are the transitions $t1$ and $t2$ enabled in this net.
Với mạng này, transition $t1$ là enabled do cả hai input place của nó là place $p1$ và place $p2$ đều có chứa token. Đối với transition $t2$ thì input place là $p3$ không chứa token nào nên không enabled.

1.1.3 Important usages of Petri net

Một số những ứng dụng quan trọng của mạng Petri: Information systems (IS) và business process modeling (BPM).

1.1.3.a Business Process (BP)

Business Process (BP) là tập hợp các nhiệm vụ với sự phụ thuộc nhân quả giữa các nhiệm vụ. Nguyên tắc sắp xếp nhiệm vụ cơ bản là:

1. **Mô hình trình tự**: đặt các nhiệm vụ theo một thứ tự tuyến tính.
2. **Mô hình Or-split**: chọn một nhánh để thực thi.
3. **Mô hình And-split**: tất cả các nhánh sẽ được thực thi.
4. **Mô hình Or-join**: một trong các nhánh đến sẽ sẵn sàng để tiếp tục.
5. **Mô hình And-join**: tất cả các nhánh đến sẵn sàng để tiếp tục.

Thực thi các tác vụ: Để thực thi các tác vụ cần có tài nguyên.

- Tài nguyên có thể lâu bền (*durable*) hoặc tiêu hao được (*consumable*).
Tài nguyên lâu bền: vẫn tồn tại sau khi thực hiện nhiệm vụ. Ví dụ: con người, máy móc...
Tài nguyên tiêu hao: biến mất trong quá trình thực thi nhiệm vụ. Ví dụ: năng lượng, nguyên liệu...
- Kết quả hoặc đầu ra của một nhiệm vụ có thể được coi là nguồn lực cho các nhiệm vụ tiếp theo hoặc sản phẩm, dịch vụ cuối cùng.
Hai loại tài nguyên lâu bền đặc biệt có tầm quan trọng là: nguồn nhân lực (*human resources*) và tài nguyên dữ liệu (*data resources*).

- Vì các hoạt động của con người đôi khi được thay thế bằng hệ thống máy tính, nên thuật ngữ *agent* được sử dụng chung cho nguồn nhân lực và hệ thống.

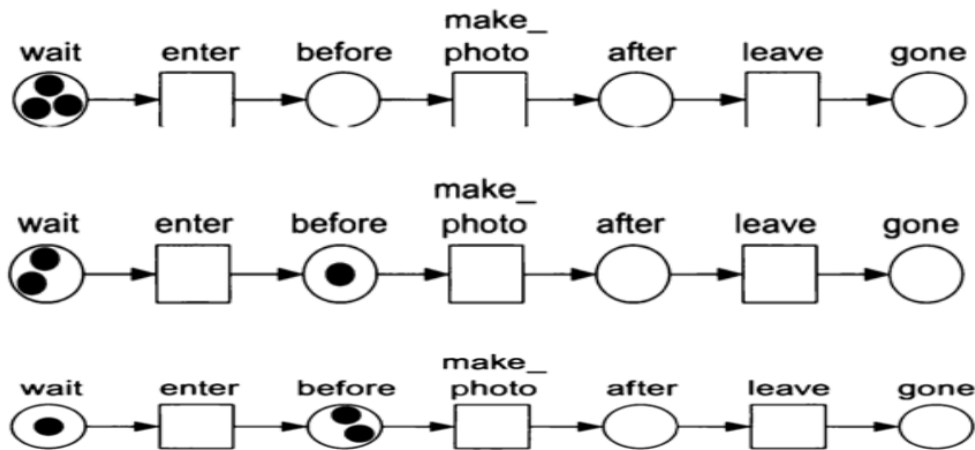
1.1.3.b Information Systems and Modeling Business Processes

Definition 1.4. (To study Petri net we first formalize the above concepts.)

1. **Business Process** bao gồm *set of activities* được thực hiện trong một tổ chức và môi trường kỹ thuật. Các hoạt động này được phối hợp để cùng thực hiện một mục tiêu kinh doanh. Mỗi quy trình kinh doanh được thực hiện bởi một tổ chức duy nhất, nhưng có thể tương tác với các quy trình kinh doanh do các tổ chức khác thực hiện.
2. **Information system** là hệ thống phần mềm để thu thập, truyền tải, lưu trữ và truy xuất thông tin do đó nó hỗ trợ con người, tổ chức hoặc hệ thống phần mềm khác.

Nhận thức về tầm quan trọng của các quy trình kinh doanh đã dẫn đến việc giới thiệu về khái niệm *process-aware information systems*. Triển khai quan trọng nhất là **workflow management systems**. Workflow management systems được cấu hình với một **process model**, trực quan đồ họa của nó là **workflow net**.

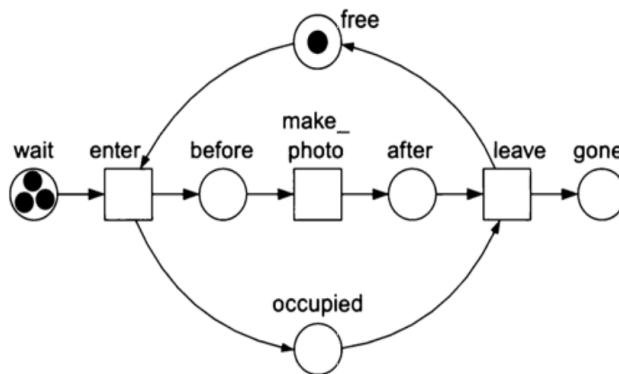
PRACTICAL PROBLEM 1.



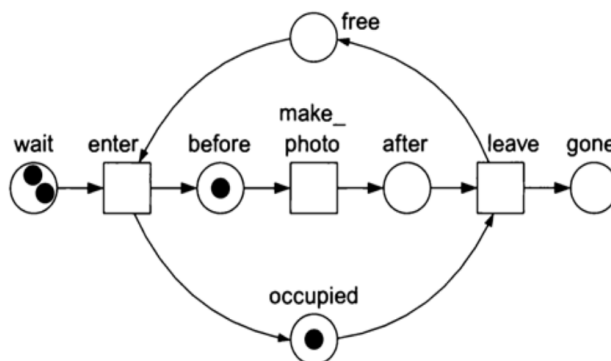
Hình 1.6: A Petri net model of a business process of an X-ray machine

1. Determine the two relations R_I and R_O and the flow relation $F = R_I \cup R_O$
Ta có $P = \{wait, before, after, gone\}$ và $T = \{enter, make_photo, leave\}$ (từ Example 1.4).
 - $R_I = P \times T = \{(wait, enter), (before, make_photo), (after, leave)\}$
 - $R_O = T \times P = \{(enter, before), (make_photo, after), (leave, gone)\}$
 - $F = R_I \cup R_O = \{(wait, enter), (before, make_photo), (after, leave), (enter, before), (make_photo, after), (leave, gone)\}$
2. A patient may enter the X-ray room only after the previous patient has left the room. We must make sure that places before and after together do not contain more than one token.
There are two possible states: the room can be free or occupied. We model this by adding these two places to the model, to get the improved the Petri net.
Now for this Petri net, can place before contain more than one token? Why? Rebuild the set P of place labels.

- Place **before** không thể chứa được hơn một token.
- Lý do: Với marking $[wait^3, free]$ như hình 1.7, transition **enter** là enabled, nếu firing transition **enter** thì sẽ được marking $[wait^2, before, occupied]$ như trong hình 1.8, lúc này place **before** có 1 token và place **free** không còn token nào nên transition **enter** không fire được nữa. Sau đó nếu ta firing lần lượt transition **make-photo** và transition **after** thì sẽ được marking $[wait^2, after, occupied]$ và marking $[wait^2, free, gone]$. Tại marking $[wait^2, free, gone]$, transition **enter** lại enabled nhưng lúc này place **before** không chứa token. Nếu ta tiếp tục firing transition **enter** thì quy trình sẽ giống như trên và place **before** chỉ luôn chứa tối đa là 1 token.
- $P = \{wait, free, before, occupied, after, gone\}$



Hình 1.7: An improved Petri net for the business process of an X-ray machine



Hình 1.8: The marking of the improved Petri net for the working process of an X-ray room after transition enter has fired.

- As long as there is no token in place **free** [Figure 1.8], can transition **enter** fire again? Explain why or why not. Remake the two relations R_I and R_O .
 - Tại marking $[wait^2, before, occupied]$ như hình 1.8 thì transition **enter** không thể fire do place **free** không chứa token nào. Nhưng nếu tiếp tục firing lần lượt transition **make-photo** và transition **after** thì sẽ được marking $[wait^2, after, occupied]$ và marking $[wait^2, free, gone]$. Tại marking $[wait^2, free, gone]$, transition **enter** lại enabled do đó tại marking này transition **enter** có thể fire lần nữa.
 - $R_I = \{(wait, enter), (free, enter), (before, make_photo), (after, leave), (occupied, leave)\}$
 - $R_O = \{(enter, before), (enter, occupied), (make_photo, after), (leave, free), (leave, gone)\}$

1.2 Mạng Petri - Behavior

1.2.1 From Firing, Reachability to Labeled Petri net

Token được biểu diễn dưới dạng một chấm đen trong đồ thị của Petri net.

Definition 1.5 (Firing rule)

Cho $(N, M) \in \mathbf{N}$ là một Petri net với $N = (P, T, F)$ và $M \in \mathbf{M}$

- *Transition* là **enabled** nếu có ít nhất một "token" ở mỗi vị trí đầu vào của nó.
- *Transition* $t \in T$ là **enabled** tại marking M kí hiệu $(N, M)[t]$ khi và chỉ khi $\bullet t \leq M$.
- **Firing rule** $\alpha[t] \beta \subseteq \mathbf{N} \times T \times \mathbf{N}$ là một quan hệ thỏa mãn công thức:

$$(N, M)[t] \Rightarrow (N, M)[t] (N, (M \setminus \bullet t) \uplus t \bullet)$$

với mọi $(N, M) \in \mathbf{N}$ và $t \in T$

OBSERVATION 1

1. Place có thể chứa token, transition thì không thể. Nhưng transition có thể thay đổi marking thông qua quá trình firing. Place là bị động, và transition là chủ động. Để một transition có thể fire, nó phải enabled.
2. Enabledness: Marking $M = [m_1, m_2, \dots, m_k] \equiv m : P \rightarrow \mathbf{N}$ có dạng là :
 $M = [x_1, \dots, x_1, \dots, x_k, \dots, x_k]$, chúng ta có thể gọi transtion $t \in T$ là enabled tại M khi và chỉ khi với mọi $p \in \bullet t, m(p) > 0$.
3. Một transition t có thể fire, do đó thay đổi marking M thành marking M_1 . Nếu một transition fire nó sẽ tiêu thụ một token từ những input place và sinh ra một token ở những output place của nó.
4. $(N, M)[t]$: transition t là enabled tại marking M
5. $(N, M)[t](N, M)$: quá trình firing transition t sẽ sinh ra marking M_1

Definition 1.6 (Firing sequence)

Cho $(N, M_0) \in \mathbf{N}$ là 1 marked Petri net với $N = (P, T, F)$.

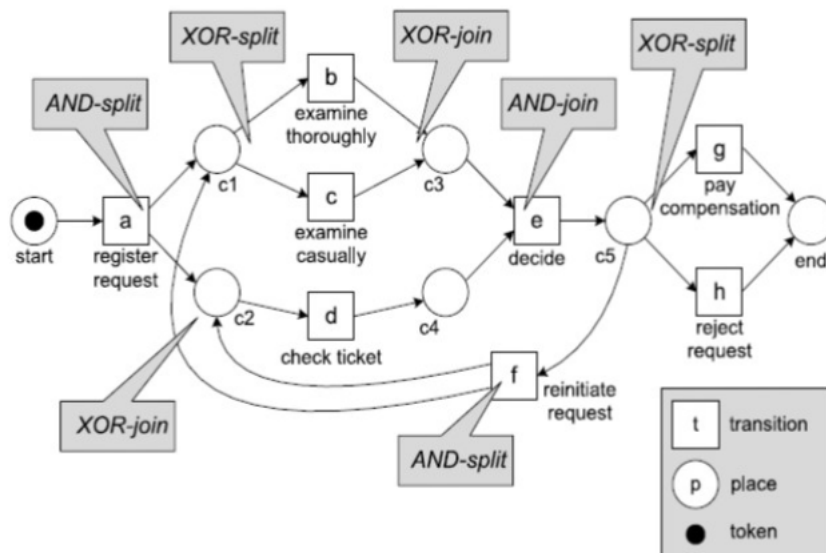
1. Một chuỗi $\sigma \in T^*$ được gọi là một chuỗi firing của (N, M_0) khi và chỉ khi đối với 1 vài số tự nhiên $n \in \mathbf{N}$ tồn tại các marking M_1, M_2, \dots, M_n và các transition T_1, T_2, \dots, T_n mà:
 - $\sigma = (t_1 t_2, \dots, t_n) \in T^*$
 - với mọi i mà $0 \leq i \leq n$ thì $(N, M_i)[t_i]$ và $(N, M_i)[t_i + 1](N, M_{i+1})$
2. Marking M là *reachable* từ marking M_0 ban đầu khi và chỉ khi tồn tại một chuỗi enabled transition mà khi firing có thể dẫn từ M_0 đến M . Tập các *reachable marking* của (N, M_0) được kí hiệu là $[N, M_0]$
3. Hệ thống Petri Net (P, T, F, M_0) bao gồm một Petri Net (P, T, F) và một marking M_0 - marking ban đầu.

EXAMPLE 1.6.

In Fig. 1.9, write marking $M_0 = [start] = [1, 0, 0, 0, 0, 0, 0]$, we get the marked Petri net (N, M_0) and see that

- Một chuỗi rỗng $\sigma_0 = \langle \rangle$ là enabled trong (N, M_0) , do đó là một firing sequence của (N, M_0) .
- Một chuỗi rỗng $\sigma_1 = \langle a, b \rangle$ là enabled trong (N, M_0) , và firing σ_1 dẫn đến marking $[c2, c3]$. Có thể viết $(N, [start])[a, b](N, [c2, c3])$ hay $(N, M_0)[\sigma_1](N, [c2, c3])$.
- The sequence $\sigma_2 = \langle a, b, d, e \rangle$ other possible firing sequence, and should we get $(N, M_0)[\sigma_2](N, [c5])$?
 - Ta có $M_0 = [start] = [1, 0, 0, 0, 0, 0, 0]$. Sau khi firing a ta được marking $M_1 = [c1, c2] \Rightarrow (N, M_0)[a](N, [c1, c2])$.
 - Tại $M_1 = [c1, c2]$, firing b ta được marking $M_2 = [c2, c3] \Rightarrow (N, [c1, c2])[b](N, [c2, c3])$.
 - Tại $M_2 = [c2, c3]$, firing d ta được marking $M_3 = [c3, c4] \Rightarrow (N, [c2, c3])[d](N, [c3, c4])$.
 - Tại $M_3 = [c3, c4]$, firing e ta được marking $M_4 = [c5] \Rightarrow (N, [c3, c4])[e](N, [c5])$.

Vậy σ_2 là một firing sequence trong (N, M_0) và $(N, M_0)[\sigma_2](N, [c5])$.



Hình 1.9: A labeled marked Petri net

- Is $\sigma = \langle a, c, d, e, f, b, d, e, g \rangle$ a firing? What is the reachable marking M in the output $(N, M_0)[\sigma](N, M)$
 - Ta có $M_0 = [start] = [1, 0, 0, 0, 0, 0, 0]$. Sau khi firing a ta được marking $[c1, c2] \Rightarrow (N, M_0)[a](N, [c1, c2])$.
 - Tại $[c1, c2]$, firing c ta được marking $[c2, c3] \Rightarrow (N, [c1, c2])[c](N, [c2, c3])$.
 - Tại $[c2, c3]$, firing d ta được marking $[c3, c4] \Rightarrow (N, [c2, c3])[d](N, [c3, c4])$.
 - Tại $[c3, c4]$, firing e ta được marking $[c5] \Rightarrow (N, [c3, c4])[e](N, [c5])$.
 - Tại $[c5]$, firing f ta được marking $[c1, c2] \Rightarrow (N, [c5])[f](N, [c1, c2])$.
 - Tại $[c1, c2]$, firing b ta được marking $[c2, c3] \Rightarrow (N, [c1, c2])[b](N, [c2, c3])$.
 - Tại $[c2, c3]$, firing d ta được marking $[c3, c4] \Rightarrow (N, [c2, c3])[d](N, [c3, c4])$.

- Tại $[c3, c4]$, firing e ta được marking $[c5] \Rightarrow (N, [c3, c4])[e](N, [c5])$.
- Tại $[c5]$, firing g ta được marking $[end] \Rightarrow (N, [c5])[f](N, [end])$.

Vậy σ là một firing sequence trong (N, M_0) và $(N, M_0)[\sigma_2](N, [c5])$, do đó reachable marking M in the output $(N, M_0)[\sigma](N, M)$ là $[c5]$.

- Check that the set $[N, M_0]$ (of reachable markings of (N, M_0)) has seven reachable.
 Tại các câu trên, sau khi đến marking $[c5]$, có 3 enabled transition là **f, g, h**. Tại đây, nếu firing **g** hoặc **h** sẽ dẫn đến marking $[end]$ và không còn enabled transition nào. Còn nếu firing **f** thì sẽ quay lại marking $[c1, c2]$ và thực hiện lại quá trình như trên đến khi nào đạt đến marking $[end]$. Vậy toàn bộ các reachable marking của (N, M_0) là [start](sequence rỗng), $[c1, c2]$, $[c2, c3]$, $[c1, c4]$, $[c3, c4]$, $[c5]$, $[end]$. Vậy (N, M_0) có tổng cộng 7 reachable markings.

Definition 1.7 (Labeled Petri net)

Thường thì transition được định danh bằng 1 ký tự đơn, nhưng cũng có một nhãn dài hơn để mô tả một hoạt động tương ứng. Một **labeled Petri Net** $N = (P, T, F, A, l)$ với (P, T, F) là một Petri Net như đã định nghĩa tại Definition 1.2.

- $A \subseteq A$ là một tập nhãn của hoạt động *activity labels*, và ánh xạ $l \in \{L : T \rightarrow A\}$ là hàm dán nhãn (*labeling function*).
- Sử dụng nhãn τ cho những nhãn hoạt động đặc biệt, gọi là "invisible". Một transition $t \in T$ với $l(t) = \tau$ được gọi là *unobservable*, *silent* hay *invisible*.

OBSERVATION 2

1. Ví dụ về X-ray machine tại Practical Problem 1 diễn tả việc có thể đi qua nhiều marking trong Petri Net bằng một chuỗi firing. Các transition liên tục firing cho đến khi mạng chạm tới một marking mà không transition nào là enable. Giống như terminal state trong transition system, marking này được gọi là *terminal marking*.
2. Chúng ta có thể chuyển bất kì Petri net nào thành "labeled" Petri net chỉ cần lấy $A = T$ và $l(t) = t$ cho bất kì $t \in T$. Quá trình ngược lại không phải lúc nào cũng đúng, do nhiều transition có thể có cùng nhãn.

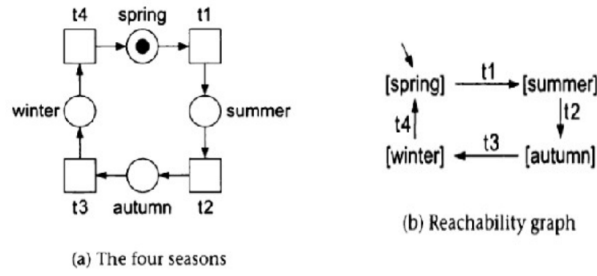
QUESTION 1.1 On markings in nets, when we have modeled a system as a Petri net system (N, M_0) then some matter occur, including

1. How many markings are reachable?
2. Which markings are reachable?
3. Are there any reachable terminal markings?

Khi chúng ta biết marking M_0 ban đầu của Petri Net (N, M_0) , chúng ta trả lời các câu hỏi trên bằng việc tính toán tập các reachable marking từ M_0 . Chúng ta biểu diễn tập này bằng một đồ thị -đây là **reachability graph** của mạng. Những node của đồ thị này tương ứng với những reachable markin và những cạnh của nó tương ứng với những transition thay đổi mạng từ 1 marking đến marking khác.

EXAMPLE 1.7. (Reachability graph)

Xem xét hệ thống Petri net bên dưới mô hình 4 mùa:



Hình 1.10: A Petri net system and its reachability graph

Nhớ lại rằng mỗi một reachable marking thì được biểu diễn như 1 tập *multiset*. Multiset [spring] thể hiện 1 marking như hình bên trên.

- Những cạnh vào mà không có nguồn chỉ vào node kí hiệu marking này là marking ban đầu. Chúng ta đã dán nhãn mỗi cạnh của đồ thị bên phải với transition được "fire" với marking tương ứng.
- Hình bên phải phía trên là tập các reachable marking từ marking ban đầu như ở hình bên trái.
- Nếu một marking M có thể đạt được từ marking ban đầu M_0 thì đồ thị có một đường đi từ node bắt đầu đến node biểu diễn M . Đường đi này biểu diễn chuỗi transition mà được "fire" để đạt được marking M từ M_0 .

Chúng ta qui transition này như một *run*. *Run* hữu hạn nếu đường đi và chuỗi transition là hữu hạn. Nếu không nó là vô hạn.

Câu hỏi: The path from marking [spring] to marking [winter] is a finite run (t1, t2, t3) of the net in figure 1.10(a). Does it have infinite run?

Trả lời: Petri Net trong hình 1.10(a) có một infinite run là (t1,t2,t3,t4,t1,...).

1.2.2 Representing Petri Nets as Special Transition Systems

Đối với một Petri net đơn giản, dễ để xây dựng một reachability graph tương ứng, nhưng đối với những mạng phức tạp hơn, reachability graph có thể trở nên to lớn dẫn đến việc quên đi những marking.

GENERIC AIM: Chúng ta mô tả behavior của một hệ thống Petri net $(N, M_0) \equiv (P, T, F, M_0)$ như một state transition system (S, TR, S_0) bằng cách thể hiện làm thế nào để xác định không gian trạng thái S , transition relation TR và trạng thái ban đầu S_0 của hệ thống (P, T, F, M_0) . Transition system biểu diễn không gian trạng thái của hệ thống được mô hình hóa, do đó biểu diễn tất cả các marking của mạng.

Definition 1.8 (Reachability graph)

Định nghĩa: Cho (N, M_0) với $N = (P, T, F, A, l)$ là marked labeled Petri net.

(N, M_0) định nghĩa một transition system $TS = (S, A_1, TR)$ với:

- $S = [N, M], S^{start} = M_0, A_1 = A$, và
- $TR = \{(M, M_1 \in SxS) | \exists t \in T(N, M)[t](N, M_1)\}$ hoặc với nhãn $l(t)$:
 $TR = \{(M, l(t), M_1 \in SxAS) | \exists t \in T(N, M)[t](N, M_1)\}$

TS thường là "reachability graph" của (N, M_0)

ELUCIDATION

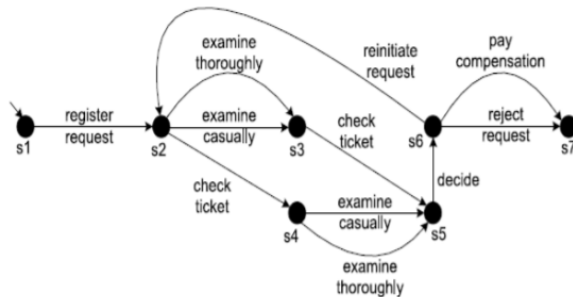
- Những phần tử của A trong mạng N là những nhãn dán nhưng khi được chuyển đến A_1 chúng được gọi là *actions* trong đầu ra của hệ thống transition TS
- Trạng thái ban đầu S_0 được định nghĩa bằng marking ban đầu: $S_0 = S^{start} = M_0$. Marking M trong N trở thành những trạng thái trong TS
- Sự mô tả của TR cho hệ thống Petri net thì phức tạp hơn. Xem xét 2 trạng thái bất kì trong không gian trạng thái S , 2 marking M, M_1 .
- Transition (M, M_1) là một thành phần của TR nếu có một transition $t \in T$ enabled tại marking M , và sự firing của t trong marking M sinh ra marking (M_1)
Chúng ta hình thức hóa điều này bằng việc định nghĩa như là 1 tập $(M, M_1) \in S \times S$ thỏa mãn:
 $\exists t \in T : (N, M)[t](N, M_1)$
Mặt khác, transition (M, M_1) không có thể và (M, M_1) không phải là thành phần của TR
- Tập hợp tất cả marking chứa những marking có thể đạt được từ marking M_0 ban đầu, nhưng cũng có những marking không phải. Kết quả là với một Petri net cho trước (N, M_0) , reachability graph TS là 1 đồ thị con của transition system đầy đủ.

PRACTICE 1.2. Build up the transition system TS generated from the labeled marked Petri net shown in Fig 1.9.

Đầu tiên, ta có $P = \{\text{start}, c1, c2, c3, c4, c5, \text{end}\}$, $T = \{a, b, c, d, e, f, g, h\}$, $A = \{\text{register request, examine thoroughly, examine casually, check ticket, decide, reinstate request, pay compensation, reject request}\}$ và $M_0 = [\text{start}] = [1, 0, 0, 0, 0, 0]$.

Theo Definition 1.8 (Reachability graph), từ mạng N ta định nghĩa một transition system (S, A_1, TR) , trong đó:

- $S = [N, M_0] = \{[\text{start}], [c1, c2], [c2, c3], [c1, c4], [c3, c4], [c5], [\text{end}]\}$ (các reachable marking của (N, M_0) đã tìm được tại EXAMPLE 1.6) và $S^{start} = \{M_0\} = [\text{start}]$
- $TR = \{(M, l(t), M_1) \in S \times A \times S | \exists t \in T(N, M)[t](N, M_1)\} = \{([\text{start}], \text{register request}, [c1, c2]), ([c1, c2], \text{examine thoroughly}, [c2, c3]), ([c1, c2], \text{examine casually}, [c2, c3]), ([c1, c2], \text{check ticket}, [c1, c4]), ([c2, c3], \text{check ticket}, [c3, c4]), ([c1, c4], \text{examine thoroughly}, [c3, c4]), ([c1, c4], \text{examine casually}, [c3, c4]), ([c3, c4], \text{decide}, [c5]), ([c5], \text{reinstate request}, [c1, c2]), ([c5], \text{pay compensation}, [\text{end}]), ([c5], \text{reject request}, [\text{end}])\}$



Hình 1.11: Reachability graph TS của Petri Net trong hình 1.9

Reachability graph TS của Petri Net trong hình 1.9 được thể hiện trong hình 1.11 ở trên, trong đó s1 tương ứng [start], s2 tương ứng [c1, c2], s3 tương ứng [c2, c3], s4 tương ứng [c1, c4], s5 tương ứng [c3, c4], s6 tương ứng [c5] và s7 tương ứng [end].

1.3 Mạng Petri - Cấu trúc và các vấn đề cơ bản

Đầu tiên, chúng ta nhắc lại các thuật ngữ chính, quy tắc và các ý tưởng có liên quan, trong mạng Petri $N = (P, T, F)$:

1. Khi firing một transition t , số lượng token ở tại place p bất kỳ ($m'(p)$) sẽ bằng số lượng token ban đầu ($m(p)$), trừ đi số lượng token tiêu thụ $w((p,t))$, và cộng vào số lượng token được sinh ra ($w((t,p))$). Hay $\forall p \in P : m'(p) = m(p) - w((p,t)) + w((t,p))$.
2. Khi firing một transition, có thể làm thay đổi (tăng hoặc giảm) tổng số lượng token trong mạng. Do số lượng input place và output place của transition t có thể không bằng nhau, do đó số lượng token tiêu thụ sẽ khác số token được sinh ra khi kích hoạt transition t , dẫn đến số lượng token trong mạng sẽ thay đổi.
3. Khi vài transition cùng enabled tại cùng một thời điểm, không thể xác định được transition nào sẽ được kích hoạt, mà chỉ biết một trong số chúng sẽ được kích hoạt, tình huống này được gọi là **nondeterministic choice**.

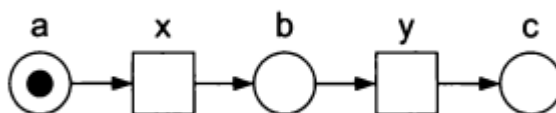
1.3.1 Causality, Concurrency and Synchronization

Nói chung, các transition đại diện cho các sự kiện. Giả sử rằng có ba sự kiện: x , y và z . Các cấu trúc mạng điển hình như sau:

1. Sự kiện y diễn ra sau sự kiện x .
2. Sự kiện x và sự kiện y diễn ra đồng thời (đồng thời hoặc theo thứ tự bất kỳ).
3. Sự kiện z xảy ra sau cả hai sự kiện x và y .
4. Sự kiện x xảy ra trước khi hai sự kiện x và y đồng thời diễn ra.
5. Sự kiện y hoặc sự kiện z xảy ra sau sự kiện x .

1.3.1.a Causality

Causality (nhân quả) chỉ một quan hệ giữa hai sự kiện trong cùng hệ thống mà chúng phải diễn ra theo trình tự xác định. Trong mạng Petri net, mỗi quan hệ này được biểu diễn bằng hai transition được nối với nhau qua một place trung gian, như trong hình 1.12 dưới đây.



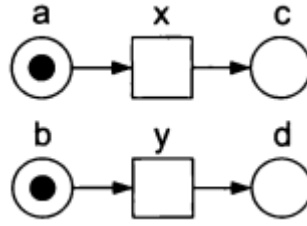
Hình 1.12: Causality trong mạng Petri N

Ta thấy trong hình thể hiện, sự kiện y chỉ có thể kích hoạt khi sự kiện x đã kích hoạt.

1.3.1.b Concurrency and Synchronization

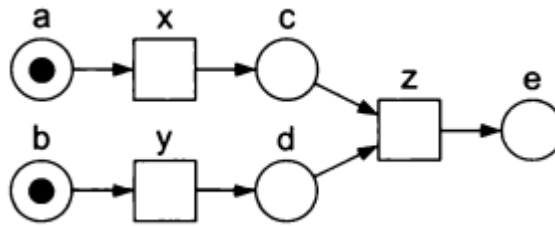
Concurrency (đồng thời) là khi vài sự kiện trong hệ thống có thể diễn ra đồng thời. Nếu hai transition trong mạng Petri không kết nối trực tiếp với nhau, như trong hình 1.13, concurrency có thể xảy ra.

Ta thấy hai sự kiện x và y trong hình x có thể diễn ra đồng thời, do không có mối quan hệ nhân quả giữa chúng.



Hình 1.13: Concurrency trong mạng Petri N

Synchronization (đồng bộ), thể hiện trong mạng Petri bằng một transition với ít nhất hai input place, như trong hình 1.14, sự kiện z chỉ có thể kích hoạt sau khi cả sự kiện x và sự kiện y đã được kích hoạt. Trong các mô hình đồng thời, thường cần đến đồng bộ.



Hình 1.14: Synchronization trong mạng Petri N

Câu hỏi: Chúng ta có thể tìm reachability graph của mạng N trong hình 1.13 không?

Trả lời: Đầu tiên, ta có $P = \{a, b, c, d\}$, $T = \{x, y\}$ và $M_0 = [1, 1, 0, 0]$. Ta xem N là một labeled Petri net $N = (P, T, F, A, l)$, với $A = T$ và $l(t) = t$ với $t \in T$.

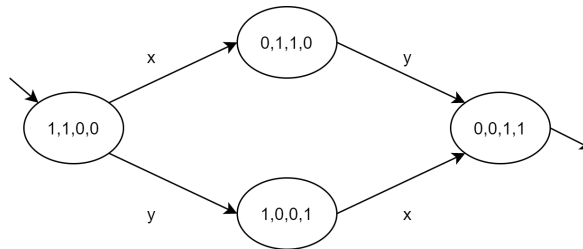
Theo Definition 1.8 (Reachability graph), từ mạng N ta định nghĩa một transition system (S, A_1, TR) , trong đó:

$S = [N, M_0] = \{(1, 1, 0, 0), (0, 1, 1, 0), (1, 0, 0, 1), (0, 0, 1, 1)\}$, $S^{start} = \{M_0\}$, $A_1 = A$.

$TR = \{(M, l(t), M_1) \in S \times A \times S \mid \exists t \in T (N, M)[t](N, M_1)\} = \{((1, 1, 0, 0), x, (0, 1, 1, 0)), ((1, 1, 0, 0), y, (1, 0, 0, 1)), ((0, 1, 1, 0), y, (0, 0, 1, 1)), ((1, 0, 0, 1), x, (0, 0, 1, 1))\}$, với:

- $((1, 1, 0, 0), x, (0, 1, 1, 0))$ là khi ta từ marking $[1, 1, 0, 0]$, firing x và đạt được marking $[0, 1, 1, 0]$.
- $((0, 1, 1, 0), y, (0, 0, 1, 1))$ là khi ta từ marking $[0, 1, 1, 0]$, firing y và đạt được marking $[0, 0, 1, 1]$.
- $((1, 1, 0, 0), y, (1, 0, 0, 1))$ là khi ta từ marking $[1, 1, 0, 0]$, firing y và đạt được marking $[1, 0, 0, 1]$.
- $((1, 0, 0, 1), x, (0, 0, 1, 1))$ là khi ta từ marking $[1, 0, 0, 1]$, firing x và đạt được marking $[0, 0, 1, 1]$.

Ta nhận thấy có bốn marking và hai transition sequence có thể có (transition x trước và transition y sau, và ngược lại), khi đó ta có thể thể hiện reachability graph của mạng N trong hình 1.15 như sau:



Hình 1.15: Reachability graph mạng N như trong hình xx

1.3.2 Effect of Concurrency

Chúng ta có thể định lượng vấn đề đồng thời của một process hoặc một Petri net bằng cách sử dụng transition system.

Fact: Nếu mô hình chứa nhiều mối quan hệ đồng thời của các transition hoặc nhiều token ở cùng một place, khi đó transition system TS sẽ lớn hơn rất nhiều so với mạng Petri N . Như chúng ta đã biết, nếu có n hoạt động đồng thời, trong khi mạng Petri chỉ cần n transition và n place để thể hiện, thì khi đó transition system cần đến 2^n state và $n \times 2^{n-1}$ transition. Nói chung, một marked Petri net (N, M_0) có thể có vô số reachable states.

1.4 SUMMARY and REVIEWED PROBLEMS

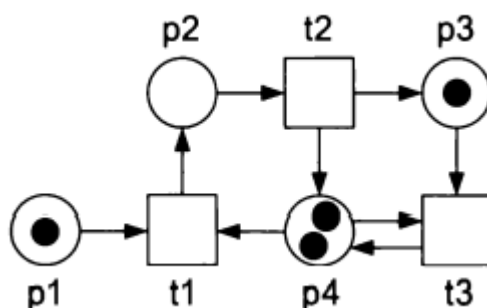
1.4.1 Problem 5.1

Giải thích các thuật ngữ của Petri net, và cung cấp ví dụ cụ thể cho mỗi thuật ngữ:

1. "enabled transition": một enabled transition (transition có thể kích hoạt) là transition mà mỗi input place của nó đều chứa một token.
2. "firing of a transition": Khi một enabled transition kích hoạt (fired), chúng tiêu thụ token từ mỗi input place của nó và tạo ra ít nhất một token cho mỗi output place của nó. Ví dụ:
3. "reachable marking": Marking M là reachable marking của M_0 nếu tồn tại một chuỗi enabled transition mà khi firing có thể dẫn từ M_0 đến M .
4. "terminal marking": terminal marking là marking mà tại đó không có bất cứ transition nào là enabled.
5. "nondeterministic choice": vài transition cùng enabled tại cùng một thời điểm, không thể xác định được transition nào sẽ được kích hoạt, mà chỉ biết một trong số chúng sẽ được kích hoạt, tình huống này được gọi là nondeterministic choice.

1.4.2 Problem 5.2

Cho mạng Petri như hình 1.16.



Hình 1.16: A Petri net with small numbers of places and transitions

1. Formalize this net as a quadruplet (P, T, F, M_0) .

Dựa vào mạng petri được cho, ta xác định được (P, T, F, M_0) như sau:

- $P = \{p1, p2, p3, p4\}$

- $T = \{t_1, t_2, t_3\}$
- $F = \{(p_1, t_1), (p_2, t_2), (p_3, t_3), (p_4, t_1), (p_4, t_3), (t_1, p_2), (t_2, p_3), (t_2, p_4), (t_3, p_4)\}$
- $M_0 = [p_1, p_3, p_4^2]$

2. Give the preset and the postset of each transition.

- Preset và postset của t_1 : $\bullet t_1 = \{p_1, p_4\}$, $t_1 \bullet = \{p_2\}$.
- Preset và postset của t_2 : $\bullet t_2 = \{p_2\}$, $t_2 \bullet = \{p_3, p_4\}$.
- Preset và postset của t_3 : $\bullet t_3 = \{p_3, p_4\}$, $t_3 \bullet = \{p_4\}$.

3. Which transitions are enabled at M_0 ?

Tại $M_0 = [p_1, p_3, p_4^2]$, ta thấy t_1 và t_3 là enabled, vì:

- Hai input place của t_1 đều có chứa token: p_1 chứa một token và p_4 chứa hai token.
- Hai input place của t_3 đều có chứa token: p_3 chứa một token và p_4 chứa hai token.

Transition t_2 không enabled tại M_0 , do input place của nó là p_2 không chứa token nào.

4. Give all reachable markings. What are the reachable terminal markings?

Tại $M_0 = [p_1, p_3, p_4^2]$, ta thấy có hai enabled transitions là t_1 và t_3 :

- Nếu firing t_1 tại M_0 , ta được marking $[p_2, p_3, p_4]$. Tại $[p_2, p_3, p_4]$, enabled transitions là t_2 và t_3 :
 - Nếu firing t_3 tại $[p_2, p_3, p_4]$, ta được marking $[p_2, p_4]$ và enabled transitions là t_2 . Firing t_2 tại marking $[p_2, p_4]$, ta được marking $[p_3, p_4^2]$ và enabled transitions là t_3 . Firing t_3 tại marking $[p_3, p_4^2]$ ta được marking $[p_4^2]$ và không còn enabled transitions nào.
 - Nếu firing t_2 tại $[p_2, p_3, p_4]$, ta được marking $[p_3^2, p_4^2]$ và enabled transitions là t_3 . Firing t_3 là được marking $[p_3, p_4^2]$, tiếp tục firing t_3 lần nữa ta được marking $[p_4^2]$, và không còn enabled transitions nào.
- Nếu firing t_3 tại M_0 , ta được marking $[p_1, p_4^2]$ và enabled transitions là t_1 . Firing t_1 tại marking $[p_1, p_4^2]$, đạt được marking $[p_2, p_4]$ và enabled transitions là t_2 . Firing t_2 tại marking $[p_2, p_4]$, ta được marking $[p_3, p_4^2]$ và enabled transitions là t_3 . Firing t_3 tại marking $[p_3, p_4^2]$ ta được marking $[p_4^2]$ và không còn enabled transitions nào.

Vậy toàn bộ các reachable marking là $[p_1, p_3, p_4^2]$, $[p_2, p_3, p_4]$, $[p_2, p_4]$, $[p_3^2, p_4^2]$, $[p_3, p_4^2]$, $[p_1, p_4^2]$, $[p_4^2]$. Và reachable terminal marking duy nhất là $[p_4^2]$ do tại marking này không transitions nào enabled.

5. Is there a reachable marking in which we have a nondeterministic choice?

Từ toàn bộ reachable marking đã tìm thấy, ta thấy tại marking $[p_1, p_3, p_4^2]$ (M_0) có hai transition t_1 và t_3 là enabled, và tại marking $[p_2, p_3, p_4]$ có hai transition t_2 và t_3 là enabled.

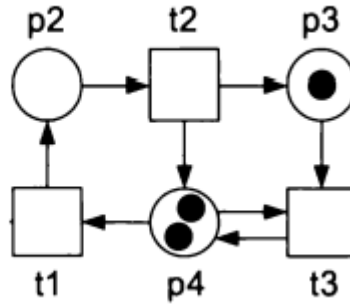
Vậy tại hai marking $[p_1, p_3, p_4^2]$ và $[p_2, p_3, p_4]$ xảy ra nondeterministic choice.

6. Does the number of reachable markings increase or decrease if we remove

(1) place p_1 and its adjacent arcs and

Sau khi xóa p_1 và các cung nối, mạng Petri được cho trở thành như hình 1.17 bên dưới:

Trước khi xóa p_1 , ta chỉ có thể firing t_1 một lần do tại p_1 chỉ có 1 token, sau khi xóa p_1 , ta thấy để t_1 là enabled chỉ cần input place t_4 có token là đủ. Sau khi xóa, mỗi khi firing t_1 sẽ tạo ra một token tại p_2 , lúc này t_2 là enabled, và khi firing t_2 lại tạo ra 1 token tại p_2 , và quá trình này có thể lặp lại tạo thành một chuỗi firing t_1, t_2, t_1, \dots vô tận. Lúc này mạng sẽ không có reachable terminal



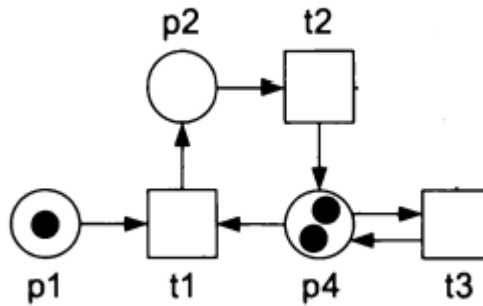
Hình 1.17: Mạng Petri sau khi xóa p_1

marking, và mỗi lần firing t_2 sẽ tạo ra một token tại p_3 , do đó số lượng reachable marking cũng có thể là vô tận.

Vậy khi xóa p_1 , số lượng reachable marking tăng lên.

(2) place p_3 and its adjacent arcs?

Sau khi xóa p_3 và các cung nối, mạng Petri được cho trở thành như hình 1.18 dưới đây. Lúc này



Hình 1.18: Mạng Petri sau khi xóa p_3

$M_0 = [p_1, p_4^2]$, và có hai enabled transitions là t_1 và t_3 , nhưng việc firing t_3 không làm thay đổi marking. Nếu firing t_1 tại $M_0 = [p_1, p_4^2]$, sẽ đạt được marking là $[p_2, p_4]$ và enabled transitions là t_2 . Firing t_2 sẽ được marking $[p_4^2]$ và không còn enabled transition nào nữa.

Vậy ta thấy rõ khi xóa p_3 , số lượng reachable marking giảm.

1.4.3 Problem 5.3

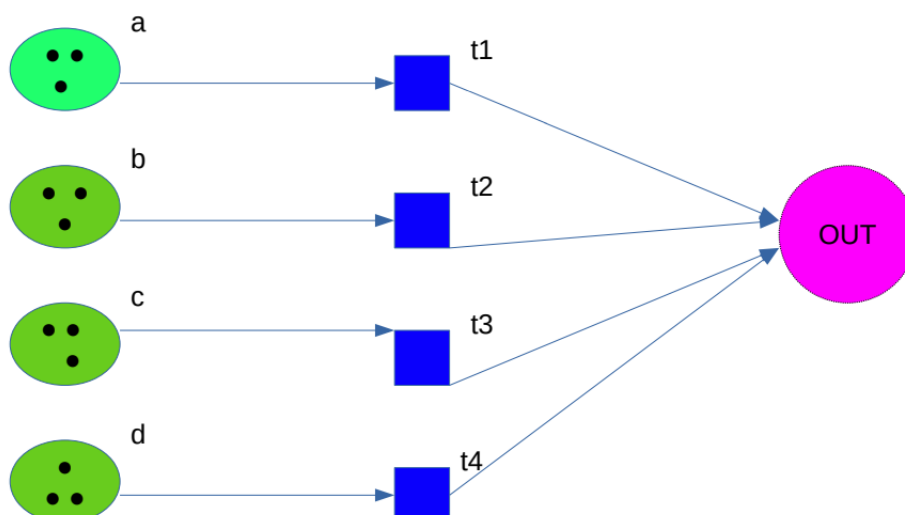
Cho mạng Petri (N, M_0) như hình 1.19.

1. Write down P, T, M_0 of N .

Dựa vào mạng petri được cho, ta xác định được P, T, M_0 như sau:

- $P = \{a, b, c, d\}$
- $T = \{t_1, t_2, t_3, t_4\}$
- $M_0 = [a^3, b^3, c^3, d^3]$

2. If not allow concurrency in this process (marked Petri net) then how many states of the transition system TS can be created? ? How many transitions are there?



Hình 1.19: A Petri net allows concurrency

1.5 Mạng Petri - Bài tập lớn về mô hình hóa

1.5.1 Các ý niệm cơ bản để mô hình hóa với mạng Petri

Một **token** có thể đại diện cho nhiều loại đối tượng, nó có thể giữ các vai trò sau trong một mô hình:

- Một đối tượng vật lý (a physical object), ví dụ như: một sản phẩm, một linh kiện, một loại thuốc hoặc một người...
- Một đối tượng thông tin (an information object), ví dụ như: một tin nhắn, một tín hiệu, một báo cáo...
- Một tập hợp các đối tượng (a collection of objects), ví dụ như: một xe tải chở hàng hóa, một nhà kho chứa đầy linh kiện, một file chứa thông tin.
- Một tín hiệu trạng thái (an indicator of a state), ví dụ như: một trạng thái của một quá trình (hoàn thành hay chưa), đèn báo hiệu giao thông...
- Một tín hiệu điều kiện (an indicator of a condition), ví dụ như: sự hiện diện của **token** báo hiệu một điều kiện được thỏa mãn (hoặc không).

Place có thể chứa nhiều **tokens**, do đó, vai trò của một **place** gắn liền với các **tokens** chứa trong nó. Một **place** có thể giữ các vai trò sau trong một mô hình:

- Một bộ đệm (a buffer), ví dụ như: một kho hàng, một hàng chờ, hoặc một thùng bưu phẩm;
- Phương trạm liên lạc (a communication medium), ví dụ như: đường dây điện thoại, người trung gian hoặc mạng lưới liên lạc;
- Một địa điểm địa lý (a geographic location), ví dụ như: một địa điểm trong nhà kho, văn phòng hoặc bệnh viện...
- Một trạng thái hoặc một điều kiện (a possible state or state condition), ví dụ như: một trạng thái sẵn sàng hoặc bận của một bác sĩ...

Place là các yếu tố thụ động trong mạng Petri. Các **token** trong một **place** thể hiện một phần trạng thái của mạng Petri. Tuy nhiên, **place** không thể thay đổi một trạng thái (đó là lý do vì sao chúng ta gọi **place** là một yếu tố bị động). Ngược lại, các **transition** lại có thể thay đổi trạng thái của mạng Petri. Khi một **transition** được kích hoạt (firing), nó sẽ thay đổi trạng thái của mạng Petri. Vai trò của **transition** trong mạng Petri là:

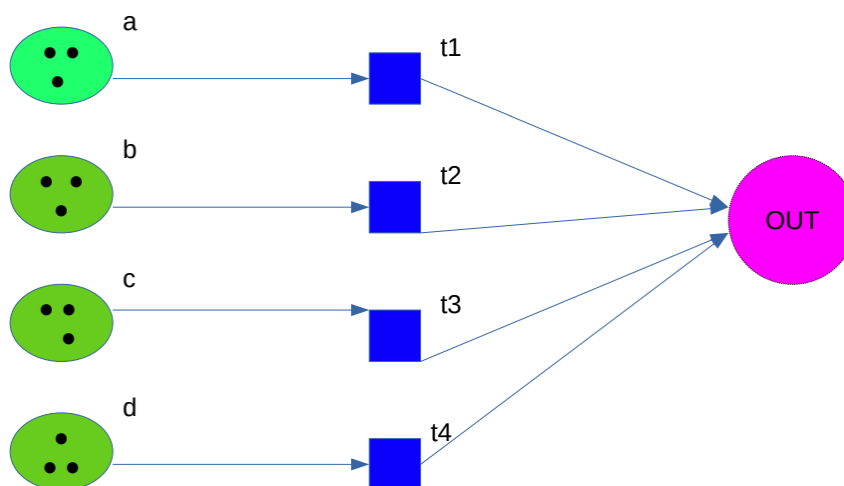
- Một sự kiện (an event), ví dụ như: một ca phẫu thuật, một lần chuyển mùa, một lần chuyển từ đèn đỏ sang đèn xanh.
- Một biến đổi (transformation) của một đối tượng, ví dụ như: sửa chữa một thiết bị, cập nhật cơ sở dữ liệu, đóng dấu lên một văn bản...
- Di chuyển một đối tượng, ví dụ như: vận chuyển hàng hóa, gửi file...

Kết luận: từ việc phân tích vai trò của **tokens**, **places** và **transitions** trong mạng Petri, ta có thể rút ra kết luận sau: các **transitions** được sử dụng để mô hình hóa các sự kiện, **place** và **token** được sử dụng để mô hình hóa trạng thái của mạng Petri.

1.5.2 Tính động của mạng Petri thông qua các tính chất đặc biệt

1. Một mạng Petri (N, M_0) được gọi là chặn k (k -bounded) nếu không có place nào chứa nhiều hơn k tokens. Hay có thể biểu diễn bởi công thức $\forall p \in P, \forall M \in [N, M_0] : M(p) \leq k$.
2. Một mạng Petri là an toàn (safe) khi và chỉ khi là chặn 1 (1-bounded).
3. Một mạng Petri bị chặn (bounded) khi và chỉ khi tồn tại một số $k \in \mathbb{N}$ sao cho mạng này bị chặn k .

Xét ví dụ sau:



Hình 1.20: Một mạng Petri bị chặn

Để thấy mạng Petri trên là 12-bounded tương ứng với trạng thái tất cả các token dồn về place "OUT". Mạng Petri này không an toàn (safe) và bị chặn với $k = 12$.

Để thể hiện được tính động của mạng Petri, chúng ta thảo luận một số tính chất sau:

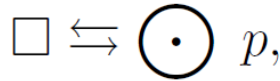
- Một mạng Petri (N, M_0) được gọi là deadlock free nếu tại mọi marking có ít nhất một transition được cho phép. Ta có thể biểu diễn bởi công thức $\forall M \in [N, M_0], \exists t \in T | (N, M)[t]$.

- Một transition $i \in T$ trong một mạng Petri (N, M_0) được gọi là live nếu từ bất kỳ marking nào chúng ta đều có thể cho phép t . Ta có thể biểu diễn tính chất này bởi công thức: $\forall M \in [N, M_0], \exists M_1 \in [N, M](N, M_1)[t]$.
- Một mạng Petri được gọi là live nếu mọi transition của nó đều live.

Chú ý: một mạng Petri là deadlock free không đồng nghĩa với việc nó là live.

Ta cùng nhìn lại mạng Petri trong hình 14. Có thể kết luận mạng Petri đó không phải là deadlock free vì tại marking $[OUT^{12}]$, không có transition nào được cho phép.

Xét ví dụ sau:



Hình 1.21: Một mạng Petri deadlock free

Ta thấy mạng Petri này chỉ có một marking là $[p^1]$, và tại marking này transition duy nhất của mạng này luôn được cho phép. Do đó ta kết luận mạng Petri này là deadlock free.

1.5.3 Các vấn đề khi mô hình hóa với mạng Petri

Mỗi quá trình có các nhân tố và hoạt động riêng biệt, do đó mạng Petri để mô hình hóa chúng cũng có các place, token và transition riêng biệt. Tuy nhiên, một hệ thống có thể bao gồm nhiều quá trình, và các quá trình này có thể có chung các token tại một số place, hay thực hiện các transition giống nhau.

Do đó, mạng Petri lớn để mô hình hóa toàn bộ một hệ thống không thể là sự kết hợp rời rạc của các mạng con cấu thành. Nó phải là sự liên kết chồng chất giữa các mạng con với nhau.

QUESTION 5.3: How could we build the grand Petri net of a large system without losing essential and useful information/knowledge of constituents' nets, as well as showing the true dynamic of the whole process/system?

Trả lời: Ta có thể xây dựng mạng Petri lớn để mô hình toàn bộ hệ thống thông qua 2 bước như sau:

1. Chia nhỏ hệ thống thành các quá trình nhỏ cấu thành. Trên mỗi quá trình, ta liệt kê các thành phần liên quan đến quá trình đó. Từ đó ta xây dựng các mạng Petri nhỏ để mô hình từng quá trình.
2. Kết hợp các mạng Petri nhỏ của từng quá trình với nhau để xây dựng mạng Petri cho toàn bộ hệ thống.

Definition 1.9 (The superimposition (or merging) operator)

Xem xét một hệ thống bao gồm hai thành phần, với N_1, N_2 là các mạng Petri mô hình chúng. Giả sử $N_1 = (P_1, T_1, F_1, M_0)$ và $N_2 = (P_2, T_2, F_2, M_0)$, các P_i có thể rời rạc, nhưng phải cùng M_0 :

1. Toán tử superimposition, $\oplus : T_1 \times T_2 \longrightarrow T$, trong đó $T = T_1 \cap T_2$, được định nghĩa như sau:

- Nếu $\bullet t_1 = \bullet t_2$ thì $(t_1, t_2) \mapsto \oplus(t_1, t_2) = t \in T$ với $\bullet t := \bullet t_1$ (các t_i giống nhau, ta chỉ giữ lại một trong mạng Petri lớn).
- Nếu $\bullet t_1 \neq \bullet t_2$ thì $(t_1, t_2) \mapsto \oplus(t_1, t_2) = t_1, t_2 \subseteq T$ (các t_i khác nhau, ta đưa cả 2 vào trong mạng Petri lớn).

2. Mạng Petri lớn (merged Petri net) được định nghĩa như sau:

$$N = N_1 \oplus N_2 = (P_1 \cup P_2, T, F_1 \cup F_2, M_0).$$

2 Trả lời câu hỏi/yêu cầu Bài tập lớn

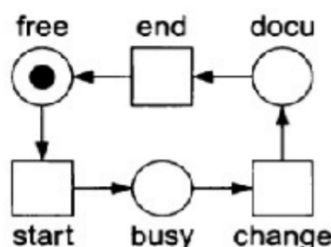
Trong phần này nhóm sẽ thực hiện các yêu cầu/câu hỏi của Bài tập lớn trong phần *Practical assignment: Consulting Medical Specialists*, cụ thể là áp dụng *Petri Net* để mô hình hóa business process của một bác sĩ chuyên khoa (specialist) trong phòng khám ngoại trú của bệnh viện X. Mỗi yêu cầu/câu hỏi sẽ được trình bày/tra lời chi tiết dựa trên những kiến thức đã chuẩn bị.

2.1 Câu hỏi 1

Given the Petri net N_S modeling the state of the specialist, as in Fig. 2.1. In the displayed marking, the specialist is in state free.

2.1.1 Câu hỏi 1a

Câu hỏi: 1a) Write down states and transitions of the Petri net N_S .



A Petri net modeling the state of the specialist.

Hình 2.1: The Petri net of the specialist's state

Đầu tiên, từ Petri net N_S mô hình hóa trạng thái của bác sĩ đề cho như trong hình 2.1, ta có Petri Net $N_S = (P_S, T_S, F_S)$ với:

- Các place $P_S = \{free, busy, docu\}$. Bác sĩ có ba state (free, busy, docu), mỗi state của bác sĩ được mô hình như một place của Petri Net N_S .
- Các transition $T_S = \{start, change, end\}$, các transition tương ứng với các event đã đề cập:
 - Transition start: ứng với event start: bác sĩ bắt đầu khám cho bệnh nhân.
 - Transition change: ứng với event change: bác sĩ chuyển từ khám cho bệnh nhân sang đưa ra kết quả.
 - Transition end: ứng với event end: bác sĩ chuyển từ trạng thái kết luận kết quả sang rảnh rỗi.

Mỗi event của bác sĩ được mô hình như một transition của Petri Net N_S .

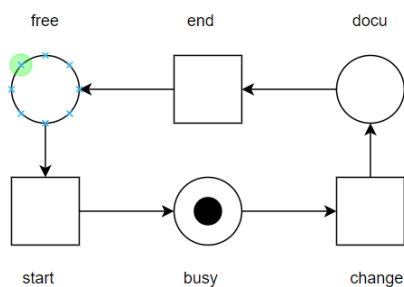
- Tập flow relation $F_S = \{(free, start), (start, busy), (busy, change), (change, docu), (docu, end), (end, free)\}$

Với M_0 là marking tương ứng với sự phân bố của các token tại các place như net N_S được cho tại hình 2.1, thì $M_0 = [free]$.

Tiếp theo, state của Petri Net là sự phân bố của các token vào các place, hay còn được xem như là marking của Petri Net. Khi một transition fire sẽ làm thay đổi trạng thái/state của Petri net. Vậy, từ đó ta có các state của Petri Net N_S như sau:

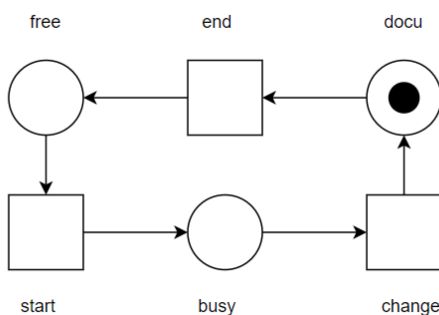
- State đầu tiên có một token ở place free, tương ứng với marking $[free]$ như trong hình 2.1 đề cho, thể hiện bác sĩ đang ở state free.

- State tiếp theo có một token ở place busy như trong hình 2.2, tương ứng với marking [busy], đạt được khi fire transition start từ marking [free]. State này thể hiện bác sĩ đang ở state busy.



Hình 2.2: State busy

- State còn lại có một token ở place docu như trong hình 2.3, tương ứng với marking [docu], đạt được khi fire transition change từ marking [busy]. State này thể hiện bác sĩ đang ở state docu.



Hình 2.3: State docu

Đó là ba state có thể có của Petri N_S được cho tại hình 2.1. Tại marking [docu], nếu firing transition end thì sẽ quay lại marking [free], và có thể tiếp tục firing như đã trình bày lặp lại nhiều lần.

2.1.2 Câu hỏi 1b

Câu hỏi: 1b) Could you represent it as a transition system assuming that

- Each place cannot contain more than one token in any marking; and
- Each place may contain any natural number of tokens in any marking.

Ta biểu diễn Petri Net N_S dưới dạng transition system (S, TR, s_0) , với (S, TR, s_0) được định nghĩa như sau:

- Tập state S là tập M gồm toàn bộ marking có thể có của Petri Net system N_S trong hình 2.1, với mỗi câu (i) và (ii), ta sẽ dùng điều kiện cụ thể của mỗi câu để tìm ra tập S . Và marking của mạng được biểu diễn dưới dạng bộ ba (x, y, z) , trong đó x thể hiện số token ở place *free*, y thể hiện số token ở place *busy*, z thể hiện số token ở place *docu*.
- Initial state s_0 tương ứng với marking ban đầu
- Tập transition relation TR chứa tất cả transition có thể xảy ra từ các state nào của S . Gồm các transition (m, m') , với m và m' là các state/markings thuộc S , và tại m thì transition t (Petri Net transition) là enabled, và firing t sẽ dẫn đến m' .

2.1.2.a Câu hỏi 1b (i) Each place cannot contain more than one token in any marking

Với điều kiện của câu 1b (i) là mỗi place không thể chứa nhiều hơn 1 token, nghĩa là mỗi x, y hoặc z trong marking (x,y,z) chỉ có thể là 0 hoặc 1. Từ đó ta có transition system (S, TR, s_0) như sau:

- Tập state $S = \{(0,0,0), (1,0,0), (0,1,0), (0,0,1), (1,1,0), (1,0,1), (0,1,1), (1,1,1)\}$ được hình thành từ điều kiện mỗi place không thể chứa nhiều hơn 1 token, ta thấy mỗi state/marking trong tập này thì mỗi phần tử x,y,z đều tối đa là 1.
- Initial state $s_0 = M_0 = [\text{free}] = (1,0,0)$ (marking trong hình 2.1).
- Tập transition relation $TR = \{((1,0,0), (0,1,0)), ((0,1,0), (0,0,1)), ((0,0,1), (1,0,0)), ((1,1,0), (1,0,1)), ((1,0,1), (0,1,1)), ((0,1,1), (1,1,0))\}$ là các transition có thể có từ các state/marking của S đã trình bày. Ví dụ transition $((1,0,0), (0,1,0))$ là từ marking $(1,0,0)$ firing transition *start* và đạt được marking $(0,0,1)$. Và transition $((0,1,1), (1,1,0))$ là từ marking $(0,1,1)$ firing transition *end* (có hai enabled transition là *change* và *end*) và đạt được marking $(1,1,0)$.

2.1.2.b Câu hỏi 1b (ii) Each place may contain any natural number of tokens in any marking

Với điều kiện của câu 1b (ii) là mỗi place có thể chứa số lượng token bất kỳ trong bất kỳ marking nào. Từ đó ta có transition system (S, TR, s_0) như sau:

- Tập state $S = (x, y, z) | (x, y, z) \in \mathbb{N}$ thể hiện việc trong một marking (x,y,z) mỗi place đều có thể chứa số token bất kỳ thuộc tập số tự nhiên, do mỗi phần tử x,y,z đều thuộc \mathbb{N} .
- Initial state $s_0 = M_0 = [\text{free}] = (1,0,0)$.
- Tập transition relation TR được biểu diễn bằng hội của 3 tập sau:

$$TR = \{((x+1, y, z), (x, y+1, z)) | (x, y, z) \in \mathbb{N}\}$$

$$\cup \{((x, y+1, z), (x, y, z+1)) | (x, y, z) \in \mathbb{N}\}$$

$$\cup \{((x, y, z+1), (x+1, y, z)) | (x, y, z) \in \mathbb{N}\}$$

Trong đó, $\{((x+1, y, z), (x, y+1, z)) | (x, y, z) \in \mathbb{N}\}$ là tập các transition $((x+1,y,z),(x,y+1,z))$ với x,y,z là các số tự nhiên bất kỳ thuộc \mathbb{N} , do đó với marking $(x+1,y,z)$ thì chắc chắn $x+1$ tối thiểu là 1, và khi đó transition *start* là enabled và firing transition *start* đạt được marking $(x,y+1,z)$.
 Tập $\{((x, y+1, z), (x, y, z+1)) | (x, y, z) \in \mathbb{N}\}$ là tập các transition $((x,y+1,z),(x,y,z+1))$ với x,y,z là các số tự nhiên bất kỳ thuộc \mathbb{N} , do đó với marking $(x,y+1,z)$ thì chắc chắn $y+1$ tối thiểu là 1, và khi đó transition *change* là enabled và firing transition *change* đạt được marking $(x,y,z+1)$.
 Tập $\{((x, y, z+1), (x+1, y, z)) | (x, y, z) \in \mathbb{N}\}$ là tập các transition $((x,y,z+1),(x+1,y,z))$ với x,y,z là các số tự nhiên bất kỳ thuộc \mathbb{N} , do đó với marking $(x,y,z+1)$ thì chắc chắn $z+1$ tối thiểu là 1, và khi đó transition *end* là enabled và firing transition *end* đạt được marking $(x+1,y,z)$.

2.2 Câu hỏi 2

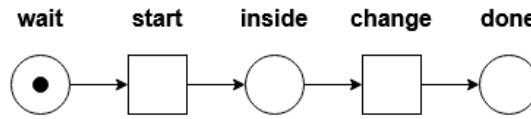
Câu hỏi: Define N_{Pa} the Petri net modeling the state of patients.

Trả lời: Theo đề bài, ta nhận thấy bệnh nhân có ba state (wait, inside, done) và hai transition (start, change). Ta mô hình hóa mỗi state của bệnh nhân như một place và mỗi transition là một Petri net transition. Từ đó, ta định nghĩa $N_{Pa} = (P_{Pa}, T_{Pa}, F_{Pa})$ như sau:

- $P_{Pa} = \{\text{wait}, \text{inside}, \text{done}\}$
- $T_{Pa} = \{\text{start}, \text{change}\}$

- $F_{Pa} = \{(wait, start), (inside, change), (start, inside), (change, done)\}$

Hình 2.4 thể hiện Petri Net N_{Pa} với một token ở place *wait* thể hiện có một bệnh nhân đang ở state *wait*:



Hình 2.4: A Petri net modeling states of patients

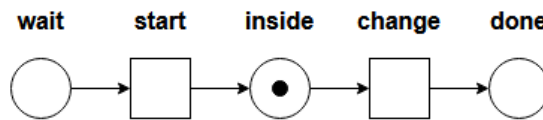
2.2.1 Câu hỏi 2a

Câu hỏi: Explain the possible meaning of a token in state *inside* of the Petri net N_{Pa} .

Trả lời: Một token trong một place thể hiện việc một bệnh nhân đang ở state tương ứng. Sau đây là một số marking của Petri net N_{Pa} với một token bên trong state **inside** và ý nghĩa của chúng:

- Tại marking [inside] như hình 2.5, có một token bên trong place **inside** và không có token nào trong place **wait** và **done**.

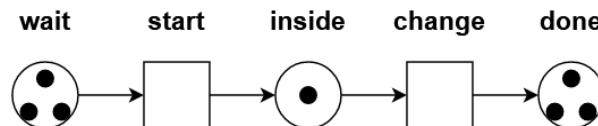
Điều này thể hiện rằng đang có một bệnh nhân ở bên trong phòng khám để được chuyên gia khám bệnh và lúc này không có ai trong phòng để chờ khám cũng như chưa có ai khám xong.



Hình 2.5: A Petri net modeling states of patients with a token in state **inside**

- Tại marking [wait³, inside, done³] như hình 2.6, có một token bên trong place **inside** cùng với đó là ba token trong mỗi place **done** và **wait**.

Việc này đồng nghĩa rằng đang có một bệnh nhân ở bên trong phòng khám để được chuyên gia khám bệnh. Lúc này vẫn còn ba bệnh nhân trong phòng chờ để đợi tới lượt khám. Và cũng có ba bệnh nhân đã hoàn thành việc khám bệnh của mình.



Hình 2.6: A Petri net modeling states of patients with a token in state **inside**

2.2.2 Câu hỏi 2b

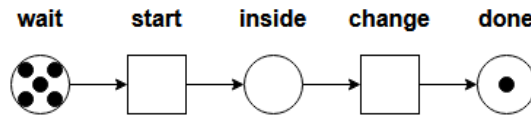
Câu hỏi: Construct the Petri net N_{Pa} , assuming that there are five patients in state *wait*, no patient in state *inside*, and one patient is in state *done*.

Trả lời:

Như đã trình bày, một token trong một place thể hiện việc một bệnh nhân đang ở state tương ứng. Do đó:

- Có năm bệnh nhân tại state wait tương ứng việc có năm token tại place **wait**, nghĩa là có năm bệnh nhân chờ đợi để vào phòng khám với bác sĩ.
- Không có bệnh nhân tại state inside tương ứng không có token nào tại place **inside**, nghĩa là không có bệnh nhân nào đang được bác sĩ điều trị.
- Có một bệnh nhân tại state done tương ứng việc có một token tại place **done**, nghĩa là có một bệnh nhân đã được khám xong.

Từ đó, ta tìm được marking tương ứng của mạng là $[wait^5, done]$. Và Petri Net N_{Pa} với marking $[wait^5, done]$ được thể hiện như trong hình 2.7 dưới đây.



Hình 2.7: A Petri net modeling states of patients with five patients in state **wait**, no patient in state **inside**, and one patient is in state **done**

2.3 Câu hỏi 3

Câu hỏi: Determine the superimposed (merged) Petri net model $N = N_S \oplus N_{Pa}$ allowing a specialist treating patients, assuming there are four patients are waiting to see the specialist/ doctor, one patient is in state done, and the doctor is in state free. (The model then describes the whole course of business around the specialist).

Trả lời: Từ N_S và N_{Pa} có được từ các câu trên, ta hợp nhất chúng lại để có được mạng Petri lớn như sau:

- Gọi $N_S = (P_S, T_S, F_S, M_0)$ và $N_{Pa} = (P_{Pa}, T_{Pa}, F_{Pa}, M_0)$.
- $P = P_S \cup P_{Pa} = [free, busy, docu, wait, inside, done]$
- $T = T_S \cup T_{Pa} = [start, end, change]$. Trong đó:
 - $start_S \neq start_{Pa}$
 - $change_S \neq change_{Pa}$
- $F = F_S \cup F_{Pa}$

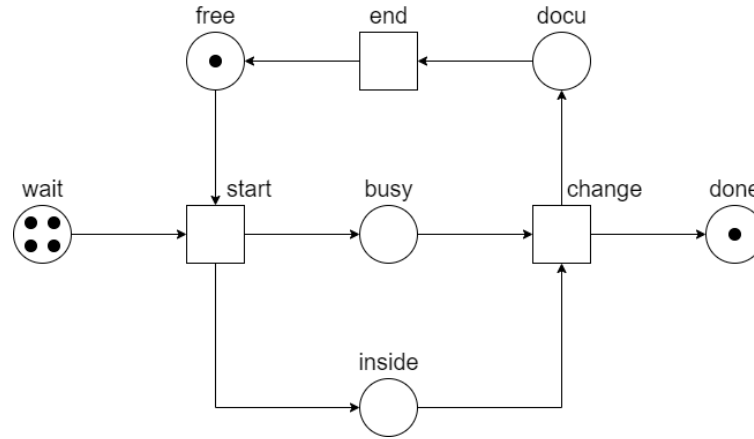
Ta được mạng Petri sau khi hợp nhất 2 mạng con $N = (P, T, F, M_0)$ như sau:

2.4 Câu hỏi 4

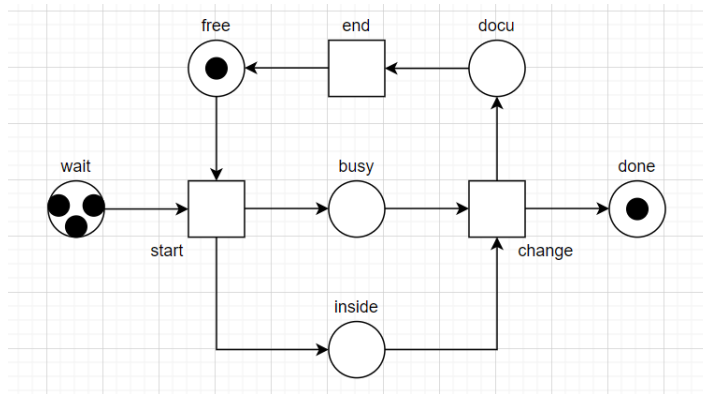
Câu hỏi: Consider an initial marking $M_0 = [3.wait, done, free]$ in the grand net $N = N_S \oplus N_{Pa}$. Which markings are reachable from M_0 by firing one transition once? Why?

Trả lời:

Đầu tiên, net $N = N_S \oplus N_{Pa}$ với $M_0 = [3.wait, done, free]$ được thể hiện trong hình xx dưới đây:



Hình 2.8: Mạng Petri sau khi hợp nhất



Hình 2.9: Net $N = N_S \oplus N_{Pa}$ với $M_0 = [3.wait, done, free]$

- Với M_0 , tại place wait có 3 token thể hiện có 3 bệnh nhân đang chờ gặp bác sĩ, place done có 1 token thể hiện có 1 bệnh nhân đã được bác sĩ điều trị, place free có 1 token thể hiện bác sĩ đang rảnh và sẵn sàng điều trị cho bệnh nhân.
- Tại $M_0 = [3.wait, done, free]$, ta thấy chỉ có transition **start** là enabled do hai input place của nó là wait và free đều chứa token.
- Firing transition **start** tại M_0 , dẫn đến marking $[2.wait, done, busy, inside]$.
- Tại marking $[2.wait, done, busy, inside]$, place wait có 2 token thể hiện còn 2 bệnh nhân đang chờ gặp bác sĩ, place done có 1 token thể hiện có 1 bệnh nhân đã được bác sĩ điều trị, place busy có 1 token thể hiện bác sĩ đang điều trị cho bệnh nhân, và place inside có 1 token thể hiện một bệnh nhân đang được điều trị bởi bác sĩ.

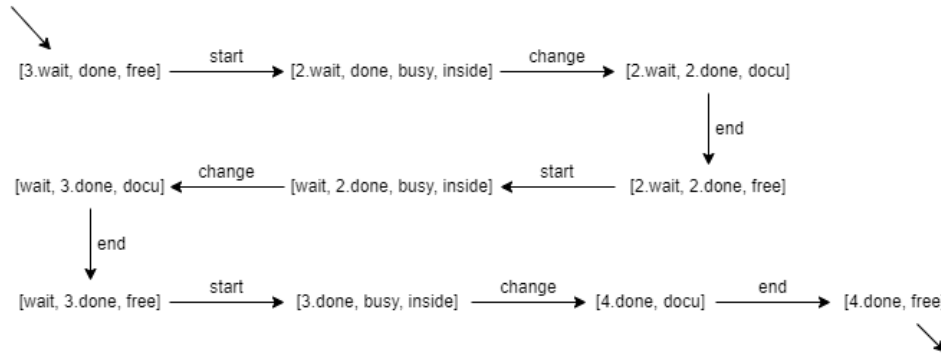
Vậy từ $M_0 = [3.wait, done, free]$, firing một transition (transition **start**) ta được marking $[2.wait, done, busy, inside]$.

2.5 Câu hỏi 5

Câu hỏi: Is the superimposed Petri net N deadlock free? Explain properly.

Trả lời: Theo định nghĩa, một marked Petri net (N, M_0) được coi là deadlock free nếu tại mọi reachable marking luôn có ít nhất 1 transition được enabled.

Với initial marking $M_0 = [3.wait, done, free]$ và Petri Net $N = N_S \oplus N_{Pa}$, ta thể hiện reachability graph của market Petri net (N, M_0) bao gồm tất cả các reachable marking từ M_0 , được thể hiện trong hình 2.10 bên dưới.



Hình 2.10: Reachability graph của mạng gộp N với $M_0 = [3.wait, done, free]$

Trong đó tập các reachable marking từ M_0 là $[N, M_0] = \{[3.wait, done, free], [2.wait, done, busy, inside], [2.wait, 2.done, docu], [2.wait, 2.done, free], [wait, 2.done, busy, inside], [wait, 3.done, docu], [wait, 3.done, free], [3.done, busy, inside], [4.done, docu], [4.done, free]\}$. Mỗi node trong đồ thị tương ứng với một reachable marking, mũi tên nối 2 node thể hiện việc thay đổi mạng từ reachable marking này (node nguồn) đến reachable marking kia (node đích) và nhãn dán trên mỗi mũi tên là transition được fire để tạo ra sự thay đổi đó.

Ta thấy tại reachable marking $[4.done, free]$ không có transition nào được kích hoạt (enabled), tức là không thỏa mãn định nghĩa deadlock free đã nói trên.

Kết luận: Petri net $N = N_S \oplus N_{Pa}$ không thỏa mãn deadlock free.

2.6 Câu hỏi 6

Bối cảnh: Vì nhu cầu khám chữa bệnh của bệnh nhân nhiều và đa dạng hơn so với dự đoán, phòng khám ngoại trú đã mời thêm một bác sĩ thứ hai có chuyên khoa khác với chuyên khoa của bác sĩ thứ nhất. Phòng khám ngoại trú này cũng bố trí một phòng khám bệnh cho bác sĩ thứ hai ngay bên cạnh bác sĩ thứ nhất. Giờ đây phòng khám ngoại trú có 2 phòng khám bệnh khác nhau là phòng khám nội khoa và phòng khám ngoại khoa, 2 bác sĩ khác nhau là bác sĩ nội khoa và bác sĩ ngoại khoa lần lượt làm việc tại các phòng khám trên. Các bệnh nhân khi đến khám tại phòng khám ngoại trú này sẽ được bốc số, và lần lượt đi vào phòng khám với chuyên khoa phù hợp (phòng khám nội khoa hoặc phòng khám ngoại khoa).

Giả thiết:

- **Bác sĩ:** Mỗi bác sĩ làm việc trong một phòng khám riêng. Mỗi bác sĩ có 2 thuộc tính (id, chuyên khoa), trong đó thuộc tính chuyên khoa được thể hiện bởi màu của token. Tại mỗi thời điểm, bác sĩ sẽ ở 1 trong 3 trạng thái sau:
 1. Free: Bác sĩ sẵn sàng để khám cho bệnh nhân tiếp theo.
 2. Busy: Bác sĩ đang thực hiện khám bệnh cho bệnh nhân.
 3. Docu: Bác sĩ đang ghi lại chẩn đoán vào hồ sơ bệnh án và viết đơn thuốc cho bệnh nhân.
- **Bệnh nhân:** Mỗi bệnh nhân sẽ chờ đến lượt để vào phòng khám mong muốn. Mỗi bệnh nhân cũng có 2 thuộc tính (id, type), trong đó id là số thứ tự khi bệnh nhân đến bốc số và type là chuyên khoa mà bệnh nhân đăng ký khám. Thuộc tính type cũng được thể hiện bởi màu của token. Tại mỗi thời điểm, bệnh nhân sẽ ở 1 trong 3 trạng thái sau:

1. Wait: Bệnh nhân chờ đến lượt để vào phòng khám.
2. Inside: Bệnh nhân đang ở trong phòng khám với bác sĩ.
3. Done: Bệnh nhân đã được khám xong bởi bác sĩ.

• **Sự kiện:** Có 3 sự kiện trong quá trình phòng khám hoạt động như sau:

1. Start: Bác sĩ và bệnh nhân bắt đầu quá trình khám bệnh. Bác sĩ và bệnh nhân phải tương thích kiểu, hay nói cách khác bệnh nhân phải đến đúng phòng khám có bác sĩ chuyên khoa muốn khám.
2. Change: Kết thúc quá trình khám bệnh, bác sĩ sẽ bắt đầu ghi bệnh án và kê đơn thuốc cho bệnh nhân. Bệnh nhân kết thúc quá trình khám bệnh.
3. End: Bác sĩ hoàn thành hồ sơ bệnh án và kê đơn, sẵn sàng khám cho bệnh nhân tiếp theo.



Hình 2.11: Các trạng thái của bác sĩ và bệnh nhân

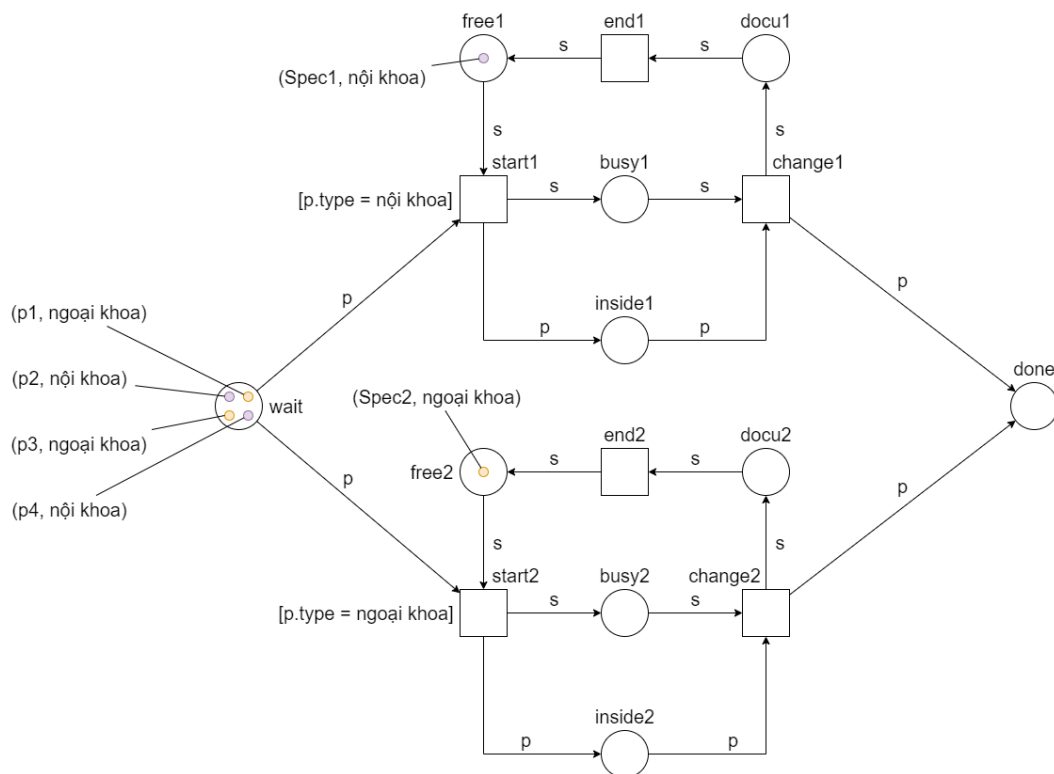
Từ các thông tin đã phân tích ở trên, ta xây dựng mạng Petri $N = (P, T, F)$ với $P = \{free1, busy1, docu1, free2, busy2, docu2, wait, inside1, inside2, done\}$, và $T = \{start1, change1, end1, start2, change2, end2\}$. Trong đó, các states $free1, busy1, docu1$ là của riêng bác sĩ nội khoa; các states $free2, busy2, docu2$ là của riêng bác sĩ ngoại khoa. State $inside1$ là của riêng bệnh nhân nội khoa và state $inside2$ là của riêng bệnh nhân ngoại khoa.

Ngoài ra, ta sử dụng 2 màu tím và vàng để lần lượt biểu diễn type nội khoa và type ngoại khoa. Các bệnh nhân ở place "wait" sẽ chỉ được đi vào nhánh có bác sĩ trùng màu với họ, hay nói cách khác bệnh nhân phải vào phòng khám có chuyên khoa phù hợp với bệnh của họ. Tại một thời điểm phòng khám ngoại khoa này chỉ phục vụ tối đa 2 bệnh nhân.

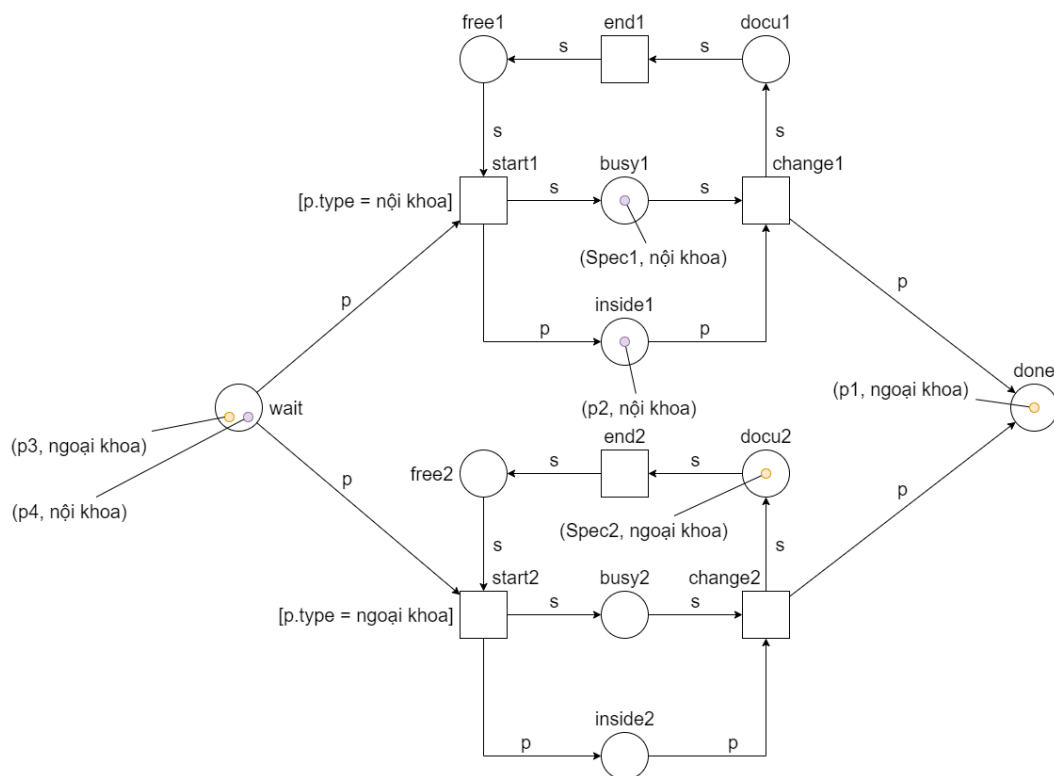
Hình 2.12 thể hiện một đánh dấu của mạng Petri mà chúng ta đang xây dựng. Đánh dấu này thể hiện thời điểm phòng khám ngoại khoa mới mở cửa và bắt đầu ngày làm việc. Cả 2 bác sĩ đều đang ở trạng thái "free" và sẵn sàng khám bệnh, có 4 bệnh nhân đang ở trạng thái "wait". Đánh dấu này được viết lại như sau: $M_0 = [4.wait, free1, free2]$

Hình 2.13 thể hiện một đánh dấu mà ở thời điểm đó bác sĩ 1 (bác sĩ nội khoa) đang ở trạng thái "busy", bác sĩ 2 (bác sĩ ngoại khoa) đang ở trạng thái "docu". Bệnh nhân có mã số p1 đang ở trạng thái "done", bệnh nhân có mã số p2 đang ở trạng thái "inside". Đánh dấu này được viết lại như sau: $M_1 = [2.wait, busy1, docu2, inside1, done]$

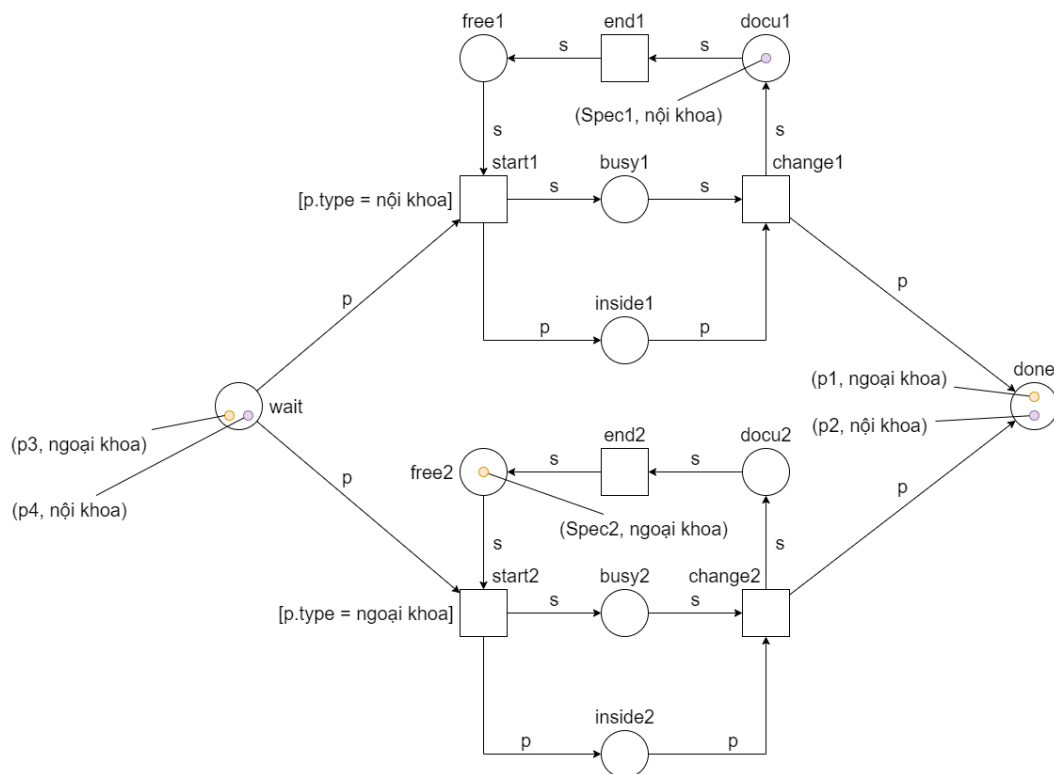
Hình 2.14 thể hiện một đánh dấu mà ở thời điểm đó bác sĩ 1 (bác sĩ nội khoa) đang ở trạng thái "docu", bác sĩ 2 (bác sĩ ngoại khoa) đang ở trạng thái "free". Bệnh nhân có mã số p1 và p2 đang ở trạng thái "done". Đánh dấu này được viết lại như sau: $M_2 = [2.wait, docu1, free2, 2.done]$



Hình 2.12: Mạng Petri cho phòng khám với 2 bác sĩ



Hình 2.13: Mạng Petri cho phòng khám với 2 bác sĩ



Hình 2.14: Mạng Petri cho phòng khám với 2 bác sĩ

2.7 Câu hỏi 7

Để hiện thực cho các Petri Net trong Bài tập lớn này, nhóm sẽ dùng ngôn ngữ Python, mỗi Petri Net sẽ là một chương trình riêng với các place, transition, và flow relation được thiết lập riêng biệt.

Các chương trình đều có các class cơ bản sau để tạo nên một Petri Net và các hàm để thực thi việc firing của mạng:

- Class Place: các place của Petri net, có các thuộc tính là tên (name) và số token đang chứa (holding).
- Class Transition: các transition của mạng, có các thuộc tính là tên (name), tập ingoing arcs (out-arcs) và tập outgoing arcs(in-arcs) của transition đó, như là tập F của Petri Net. Các phương thức là check-block() để kiểm tra transition đó có enabled tại marking hiện tại, và phương thức fire() để fire transition đó.
- Các class ArcBase, Out(ArcBase), In(ArcBase) là các class để hỗ trợ lưu trữ ingoing arcs và outgoing arcs của transition, và có phương thức trigger() để thay đổi số lượng token của mỗi place khi transition fire.
- Class Petri Net, có các thuộc tính là places và transitions là các place và transition của Petri Net. Có phương thức run() để in ra các thông tin của mạng và thực hiện quá trình firing để in ra các marking.

Mã nguồn của các Class trên như sau:

```
1 class Place:
2     def __init__(self, name, holding):
3         """
4         Place vertex in the petri net.
5         :holding: Numer of token the place is initialized with.
```

```
6         """
7         self.name = name
8         self.holding = holding
9
10
11 class ArcBase:
12     def __init__(self, place, amount=1):
13         """
14         Arc in the petri net.
15         :place: The one place acting as source/target of the arc as arc in the net
16         :amount: The amount of token removed/added from/to the place.
17         """
18         self.place = place
19         self.amount = amount
20
21
22 class Out(ArcBase):
23     def trigger(self):
24         """
25         Remove token.
26         """
27         self.place.holding -= self.amount
28
29     def non_blocking(self):
30         """
31         Validate action of outgoing arc is possible.
32         """
33         return self.place.holding >= self.amount
34
35
36 class In(ArcBase):
37     def trigger(self):
38         """
39         Add tokens.
40         """
41         self.place.holding += self.amount
42
43
44 class Transition:
45     def __init__(self, name, out_arcs, in_arcs):
46         """
47         Transition vertex in the petri net.
48         :out_arcs: Collection of ingoing arcs, to the transition vertex.
49         :in_arcs: Collection of outgoing arcs, to the transition vertex.
50         """
51         self.name = name
52         self.out_arcs = set(out_arcs)
53         self.in_arcs = set(in_arcs)
54         self.arcs = self.out_arcs.union(in_arcs)
55     def check_block(self):
56         not_blocked = all(arc.non_blocking() for arc in self.out_arcs)
```

```
57         return not_blocked
58
59     def fire(self):
60         """
61         Fire!
62         """
63         not_blocked = all(arc.non_blocking() for arc in self.out_arcs)
64         if not_blocked:
65             for arc in self.arcs:
66                 arc.trigger()
67         return not_blocked # return if fired, just for the sake of debugging
68
69
70 class PetriNet:
71     def __init__(self, places, transitions):
72         """
73         The petri net runner.
74         :transitions: The transitions encoding the net.
75         """
76         self.places = places
77         self.transitions = transitions
78
79     def run(self):
80         """
81         Print Petri Net Info: P, T, F and initial marking
82         """
83         print("Petri Net with:")
84         flag = 1;
85         print("\tP = {" , end='')
86         for key in self.places:
87             if flag != len(self.places):
88                 print(self.places[key].name, end=',')
89                 flag = flag+1
90             else:
91                 print(self.places[key].name, end='')
92         print("}")
93
94         flag = 1;
95         print("\tT = {" , end='')
96         for key in self.transitions:
97             if flag != len(self.transitions):
98                 print(self.transitions[key].name, end=',')
99                 flag = flag+1
100             else:
101                 print(self.transitions[key].name, end='')
102         print("}")
103
104
105         flag = 1;
106         print("\tF = {" , end='')
107         for key in self.transitions:
```

```
108         if flag != len(self.transitions):
109             t = self.transitions[key]
110             for arc in t.out_arcs:
111                 print("(" + arc.place.name + "," + t.name + ")", end = ",")
112             for arc in t.in_arcs:
113                 print("(" + t.name + "," + arc.place.name + ")", end = ",")
114             flag = flag+1
115         else:
116             t = self.transitions[key]
117             count = 1
118             print("(" + arc.place.name + "," + t.name + ")", end = "")
119             for arc in t.out_arcs:
120                 print("(" + arc.place.name + "," + t.name + ")", end = ",")
121             for arc in t.in_arcs:
122                 if count != len(t.in_arcs):
123                     print("(" + t.name + "," + arc.place.name + ")", end = ",")
124                     count = count + 1
125                 else:
126                     print("(" + t.name + "," + arc.place.name + ")", end = "")
127             print("}")
128
129         print("Intital Marking {}\n".format([self.places[key].holding for key in self.places]))
130
131         """
132         Firing operation...
133         """
```

Mỗi chương trình của các Petri Net sẽ có phần thiết lập place, transiton ở main và thực thi quá trình firing ở phương thức run khác nhau được trình bày riêng trong mỗi phần dưới đây.

2.7.1 Petri Net của Specialist N_S - tệp specialist.py

Đối với Petri Net của Specialist/bác sĩ, phần thiết lập place và transition được thực hiện như mã nguồn dưới đây để tạo ra Petri Net N_S như trong câu 1. Marking ban đầu cũng được thiết lập là $[1,0,0] = [free]$ như trong hình 2.1 được cho ở câu 1.

```
1 if __name__ == "__main__":
2     args = make_parser().parse_args()
3
4     ps = dict(
5         free=Place("free",1),
6         busy=Place("busy",0),
7         docu=Place("docu",0),
8     )
9     ts = dict(
10         start=Transition(
11             "start",
12             [Out(ps["free"])],
13             [In(ps["busy"])]
14         ),
15         change=Transition(
```

```
16         "change",
17         [Out(ps["busy"])],
18         [In(ps["docu"])]
19     ),
20     end=Transition(
21         "end",
22         [Out(ps["docu"])],
23         [In(ps["free"])]
24     ),
25 )
26
27 petri_net = PetriNet(ps, ts)
28 petri_net.run(args.num_of_fires)
```

Do Petri Net N_S không có deadlock nên nó có thể firing vô hạn lần, do đó đối với phần hiện thực này, nhóm sẽ cho người dùng nhập vào số lần firing mong muốn, và sẽ firing đủ số lần đó. Đoạn mã nguồn dưới đây dùng để nhận vào tham số số lần firing, và hàm run sẽ nhận thêm 1 tham số là số lần firing đó.

```
1 def make_parser():
2     """
3     :return: A parser reading in some of our simulation paramaters.
4     """
5     from argparse import ArgumentParser
6     parser = ArgumentParser()
7     parser.add_argument('--num_of_fires', type=int)
8     return parser
9
10 if __name__ == "__main__":
11     args = make_parser().parse_args()
12     ...
13     petri_net.run(args.num_of_fires)
```

Để thực thi quá trình firing, tại mỗi marking, ta sẽ kiểm tra transition nào là enabled, và firing một trong các transition đó, in ra các thông tin: marking hiện tại, enabled transition, transition được firing và marking đạt được, cho đến khi đủ số lần firing thì dừng lại. Mã nguồn để thực thi quá trình firing trong phương thức run() của class PetriNet như sau:

```
1 def run(self, num_of_fires):
2     """
3     Print Petri Net Info: P, T, F and initial marking
4     """
5     ...
6
7     """
8     Firing operation...
9     """
10    print("Firing " + repr(num_of_fires) + " times from Intital Marking:")
11
12    time = 1
13    while time <= num_of_fires:
14        time = time+1
```

```

15     count = 1
16     flag = True
17     #find enabled transition
18     ET = []
19     for key in self.transitions:
20         t = self.transitions[key]
21         if t.check_block():
22             ET.append(key);
23     print("At marking {}".format([self.places[key].holding for key in self.places]),
24           end=", ")
25     #firing random enabled transition
26     if len(ET) == 0:
27         print("none enabled transition.")
28         break;
29     else:
30         print("Transition: {}".format([et for et in ET]) + " are enabled.")
31         key = random.choice(ET)
32         t = self.transitions[key]
33         if t.fire():
34             print("{} fired!".format(t.name))
35             print(" => {}".format([self.places[key].holding for key in self.places]))
36
37     print("\nAfter firing " + repr(num_of_fires) + " times: marking is
38           {}".format([self.places[key].holding for key in self.places]), end="")
39     #find enabled transition
40     ET = []
41     for key in self.transitions:
42         t = self.transitions[key]
43         if t.check_block():
44             ET.append(key);
45     #firing random enabled transition
46     if len(ET) == 0:
47         print("none enabled transition.")
48     else:
49         print(", transition: {}".format([et for et in ET]) + " are enabled.")

```

Sau đây là kết quả chạy chương trình với số lần firing là 3:

```

C:\Users\ThinkKING\Desktop\source_MHH>python specialist.py --num_of_fires 3
Petri Net with:
    P = {free,busy,docu}
    T = {start,change,end}
    F = {(free,start),(start,busy),(busy,change),(change,docu),(docu,end),(docu,end),(end,free)}
Intital Marking [1, 0, 0]

Firing 3 times from Intital Marking:
At marking [1, 0, 0], Transition: ['start'] are enabled.
start fired!
=> [0, 1, 0]
At marking [0, 1, 0], Transition: ['change'] are enabled.
change fired!
=> [0, 0, 1]
At marking [0, 0, 1], Transition: ['end'] are enabled.
end fired!
=> [1, 0, 0]

After firing 3 times: marking is [1, 0, 0], transition: ['start'] are enabled.

```

Hình 2.15: Kết quả hiện thực Petri Net của Specialist với số lần firing là 3

Với số lần firing là 7, ta được kết quả như hình dưới đây:

```
C:\Users\ThinkKING\Desktop\source_MHH>python specialist.py --num_of_fires 7
Petri Net with:
    P = {free,busy,docu}
    T = {start,change,end}
    F = {(free,start),(start,busy),(busy,change),(change,docu),(docu,end),(docu,end),(end,free)}
Intital Marking [1, 0, 0]

Firing 7 times from Intital Marking:
At marking [1, 0, 0], Transition: ['start'] are enabled.
start fired!
=> [0, 1, 0]
At marking [0, 1, 0], Transition: ['change'] are enabled.
change fired!
=> [0, 0, 1]
At marking [0, 0, 1], Transition: ['end'] are enabled.
end fired!
=> [1, 0, 0]
At marking [1, 0, 0], Transition: ['start'] are enabled.
start fired!
=> [0, 1, 0]
At marking [0, 1, 0], Transition: ['change'] are enabled.
change fired!
=> [0, 0, 1]
At marking [0, 0, 1], Transition: ['end'] are enabled.
end fired!
=> [1, 0, 0]
At marking [1, 0, 0], Transition: ['start'] are enabled.
start fired!
=> [0, 1, 0]

After firing 7 times: marking is [0, 1, 0], transition: ['change'] are enabled.
```

Hình 2.16: Kết quả hiện thực Petri Net của Specialist với số lần firing là 7

Ta thấy kết quả hiện thực in ra được các thông tin của Petri Net và thực hiện đủ số lần firing cũng như thông tin mỗi lần fire. Đồng thời, ta thấy khi firing đủ số lần thì vẫn có transition là enabled và vẫn có thể firing tiếp.

2.7.2 Petri Net của Patient N_{Pa}

Đối với Petri Net của Patient/bệnh nhân N_{Pa} nhóm sẽ hiện thực theo hướng cho người dùng nhập vào marking ban đầu của mạng, và tại mỗi marking, ta sẽ kiểm tra transition nào là enabled, và firing một trong các transition đó, đến khi nào đạt đến terminal marking thì dừng lại.

Phần nhận vào tham số marking ban đầu và thiết lập Petri Net N_{Pa} được trình bày trong mã nguồn dưới đây:

```
1 def make_parser():
2     """
3     :return: A parser reading in some of our simulation paramaters.
4     """
5     from argparse import ArgumentParser
6     parser = ArgumentParser()
7     parser.add_argument('--marking', type=int, nargs='+')
8     return parser
9
10
11 if __name__ == "__main__":
12     args = make_parser().parse_args()
13
14     ps = dict(
```

```
15     wait=Place("wait",args.marking[0]),
16     inside=Place("inside",args.marking[1]),
17     done=Place("done",args.marking[2]),
18 )
19 ts = dict(
20     start=Transition(
21         "start",
22         [Out(ps["wait"])],
23         [In(ps["inside"])]
24     ),
25     change=Transition(
26         "change",
27         [Out(ps["inside"])],
28         [In(ps["done"])]
29     ),
30 )
31
32 petri_net = PetriNet(ps, ts)
33 petri_net.run()
```

Phần mã nguồn cho quá trình firing như sau:

```
1 def run(self):
2     """
3     Print Petri Net Info: P, T, F and initial marking
4     """
5     ...
6
7     """
8     Firing operation...
9     """
10    while True :
11        count = 1
12        flag = True
13        #find enabled transition
14        ET =[]
15        for key in self.transitions:
16            t = self.transitions[key]
17            if t.check_block():
18                ET.append(key);
19        print("At marking {}".format([self.places[key].holding for key in self.places]),
20              end=" ")
21        #firing random enabled transition
22        if len(ET) == 0:
23            print("none enabled transition.")
24            break;
25        else:
26            print("Transition: {}".format([et for et in ET]) + " are enabled.")
27            key = random.choice(ET)
28            t = self.transitions[key]
29            if t.fire():
```

```
29         print("{} fired!".format(t.name))
30         print(" => {}".format([self.places[key].holding for key in self.places]))
```

Sau đây là kết quả chạy chương trình với marking ban đầu là $[3,0,0]$, ta thấy các thông tin của N_{Pa} và marking ban đầu được in ra, và quá trình firing cùng thông tin tương ứng cũng được in ra. Tại marking $[0,0,3]$ có nghĩa có 3 token ở place và không có token nào ở place wait và inside, thì không còn enabled transition nào nên quá trình firing dừng lại.

```
C:\Users\ThinkKING\Desktop\source_MHH>python patient.py --marking 3 0 0
Petri Net with:
    P = {wait,inside,done}
    T = {start,change}
    F = {(wait,start),(start,inside),(inside,change)(inside,change),(change,done)}
Intital Marking [3, 0, 0]

At marking [3, 0, 0], Transition: ['start'] are enabled.
start fired!
=> [2, 1, 0]
At marking [2, 1, 0], Transition: ['start', 'change'] are enabled.
change fired!
=> [2, 0, 1]
At marking [2, 0, 1], Transition: ['start'] are enabled.
start fired!
=> [1, 1, 1]
At marking [1, 1, 1], Transition: ['start', 'change'] are enabled.
change fired!
=> [1, 0, 2]
At marking [1, 0, 2], Transition: ['start'] are enabled.
start fired!
=> [0, 1, 2]
At marking [0, 1, 2], Transition: ['change'] are enabled.
change fired!
=> [0, 0, 3]
At marking [0, 0, 3], none enabled transition.
```

Hình 2.17: Kết quả hiện thực Petri Net của Patient với marking ban đầu là $[3,0,0]$

Nếu đầu vào là một terminal marking thì sẽ không có transition nào được fire, ta thử với marking ban đầu là $[0,0,2]$ thì nhận thấy không có transition nào được fire.

```
C:\Users\ThinkKING\Desktop\source_MHH>python patient.py --marking 0 0 2
Petri Net with:
    P = {wait,inside,done}
    T = {start,change}
    F = {(wait,start),(start,inside),(inside,change)(inside,change),(change,done)}
Intital Marking [0, 0, 2]

At marking [0, 0, 2], none enabled transition.
```

Hình 2.18: Kết quả hiện thực Petri Net của Patient với marking ban đầu là $[0,0,2]$

2.7.3 Petri Net $N = N_S \oplus N_{Pa}$

Đối với Petri Net $N = N_S \oplus N_{Pa}$, nhóm sẽ hiện thực theo hướng:

- Cho nhập vào tham số số lượng bệnh nhân ở place wait, và tối đa là 10 token trong place wait như đề bài cho.
- Thiết lập một token ở place free tương ứng việc có một specialist và specialist đang ở state free, một token đang ở place done tương ứng có một bệnh nhân đã khám xong (tương tự như câu 3,4), các place còn lại không có token nào.

- Tại mỗi marking, ta sẽ kiểm tra transition nào là enabled, và firing một trong các transition đó, đến khi nào đạt đến terminal marking thì dừng lại.

Phần nhận vào tham số số bệnh nhân tại place wait và thiết lập Petri Net được trình bày trong mã nguồn dưới đây:

```
1 def make_parser():
2     """
3     :return: A parser reading in some of our simulation paramaters.
4     """
5     from argparse import ArgumentParser
6     parser = ArgumentParser()
7     parser.add_argument('--wait_patients', type=int)
8     return parser
9
10
11 if __name__ == "__main__":
12     args = make_parser().parse_args()
13
14     ps = dict(
15         wait=Place("wait",args.wait_patients),
16         free=Place("free",1),
17         busy=Place("busy",0),
18         inside=Place("inside",0),
19         docu=Place("docu",0),
20         done=Place("done",1),
21     )
22     ts = dict(
23         start=Transition(
24             "start",
25             [Out(ps["wait"]), Out(ps["free"])],
26             [In(ps["inside"]), In(ps["busy"])]
27         ),
28         change=Transition(
29             "change",
30             [Out(ps["inside"]), Out(ps["busy"])],
31             [In(ps["done"]), In(ps["docu"])]
32         ),
33         end=Transition(
34             "end",
35             [Out(ps["docu"])],
36             [In(ps["free"])]
37         ),
38     )
39
40     petri_net = PetriNet(ps, ts)
41     petri_net.run()
```

Phần mã nguồn cho quá trình firing như sau:

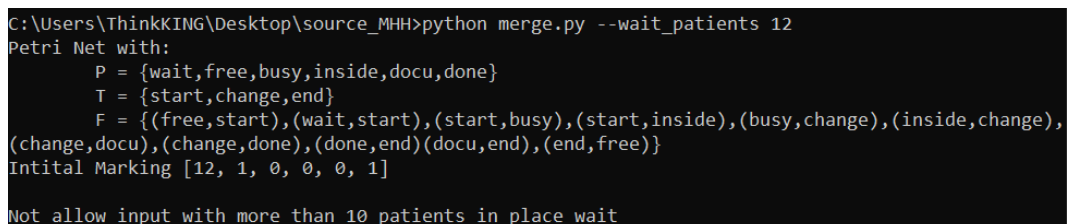
```
1 def run(self):
2     """
```

```

3      Print Petri Net Info: P, T, F and initial marking
4      """
5      ...
6
7      """
8      Check valid Intital Marking
9      """
10     if self.places["wait"].holding > 10:
11         print("Not allow input with more than 10 patients in place wait")
12         return
13
14     """
15     Firing operation...
16     """
17     while True :
18         count = 1
19         flag = True
20         #find enabled transition
21         ET =[]
22         for key in self.transitions:
23             t = self.transitions[key]
24             if t.check_block():
25                 ET.append(key);
26             print("At marking {}".format([self.places[key].holding for key in self.places]),
27                   end=", ")
28             #firing random enabled transition
29             if len(ET) == 0:
30                 print("none enabled transition.")
31                 break;
32             else:
33                 print("Transition: {}".format([et for et in ET]) + " are enabled.")
34                 key = random.choice(ET)
35                 t = self.transitions[key]
36                 if t.fire():
37                     print("{} fired!".format(t.name))
38                     print(" => {}".format([self.places[key].holding for key in self.places]))

```

Ta thử chạy với số lượng bệnh nhân ở place wait là 12, vượt quá yêu cầu tối đa đề bài là 10, thì kết quả như sau:



```

C:\Users\ThinkKING\Desktop\source_MHH>python merge.py --wait_patients 12
Petri Net with:
  P = {wait,free,busy,inside,docu,done}
  T = {start,change,end}
  F = {(free,start),(wait,start),(start,busy),(start,inside),(busy,change),(inside,change),
(change,docu),(change,done),(done,end)(docu,end),(end,free)}
Intital Marking [12, 1, 0, 0, 0, 1]

Not allow input with more than 10 patients in place wait

```

Hình 2.19: Kết quả hiện thực Petri Net của Patient với số bệnh nhân ở place wait là 12

Sau đây là kết quả chạy chương trình với số lượng bệnh nhân đầu vào ở place wait là 3, ta thấy các thông tin của Petri Net N và marking ban đầu được in ra, và quá trình firing cùng thông tin tương ứng cũng

được in ra. Tại marking $[0,1,0,0,0,4]$ không còn enabled transition nào nên quá trình firing dừng lại. (thứ tự các place trong marking là wait, free, busy, inside, docu, done)

```
C:\Users\ThinkKING\Desktop\source_MHH>python merge.py --wait_patients 3
Petri Net with:
  P = {wait,free,busy,inside,docu,done}
  T = {start,change,end}
  F = {(free,start),(wait,start),(start,busy),(start,inside),(busy,change),(inside,change),
(change,docu),(change,done),(done,end)(docu,end),(end,free)}
Initial Marking [3, 1, 0, 0, 0, 1]

At marking [3, 1, 0, 0, 0, 1], Transition: ['start'] are enabled.
start fired!
=> [2, 0, 1, 1, 0, 1]
At marking [2, 0, 1, 1, 0, 1], Transition: ['change'] are enabled.
change fired!
=> [2, 0, 0, 0, 1, 2]
At marking [2, 0, 0, 0, 1, 2], Transition: ['end'] are enabled.
end fired!
=> [2, 1, 0, 0, 0, 2]
At marking [2, 1, 0, 0, 0, 2], Transition: ['start'] are enabled.
start fired!
=> [1, 0, 1, 1, 0, 2]
At marking [1, 0, 1, 1, 0, 2], Transition: ['change'] are enabled.
change fired!
=> [1, 0, 0, 0, 1, 3]
At marking [1, 0, 0, 0, 1, 3], Transition: ['end'] are enabled.
end fired!
=> [1, 1, 0, 0, 0, 3]
At marking [1, 1, 0, 0, 0, 3], Transition: ['start'] are enabled.
start fired!
=> [0, 0, 1, 1, 0, 3]
At marking [0, 0, 1, 1, 0, 3], Transition: ['change'] are enabled.
change fired!
=> [0, 0, 0, 0, 1, 4]
At marking [0, 0, 0, 0, 1, 4], Transition: ['end'] are enabled.
end fired!
=> [0, 1, 0, 0, 0, 4]
At marking [0, 1, 0, 0, 0, 4], none enabled transition.
```

Hình 2.20: Kết quả hiện thực Petri Net của Patient với số bệnh nhân ở place wait là 3

Tài liệu

- [1] **PROCESS MINING**, 2nd edition, 2016, Springer, by W.M.P. Aalst, van der.
- [2] **Modeling Business Processes: A Petri Net-Oriented Approach**, 2011 Massachusetts Institute of Technology, by Wil van der Aalst and Christian Stahl.
- [3] Kurt Jensen, Wil M.P. van der Aalst, Gianfranco Balbo, Maciej Koutny, Karsten Wolf (Eds.).
Transactions on Petri Nets and Other Models of Concurrency VII LNCS 7480, Springer, 2013 .