VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

## MATHEMATICAL MODELLING (CO2011)

---

**Assignment**

# PETRI NETWORKS

---

| | |
|---|---|
| Advisor: | Nguyễn Tiến Thịnh |
| Students: | Trần Huy Đức - 1911071 |
| | Lê Trần Trung Hiếu - 1810708 |
| | Huỳnh Nguyễn Hiếu Nghĩa - 1712314 |
| | Trương Nguyễn Khôi Nguyên - 2010468 |
| | Bùi Đoàn Gia Phong - 2010509 |

HO CHI MINH CITY, NOVEMBER 2021

# Contents

# Danh sách hình vẽ

## 1 Member list & Workload

| No. | Fullname | Student ID | Tasks | Percentage of work |
|---|---|---|---|---|
| 1 | Trần Huy Đức | 1911071 | - Part 2.2.<br>- Question 5, 6, 7 Part 3. | 100% |
| 2 | Lê Trần Trung Hiếu | 1810708 | - Part 2.1.<br>- Write report. | 100% |
| 3 | Huỳnh Nguyễn Hiếu Nghĩa | 1712314 | - All question of Part 3. | 100% |
| 4 | Trương Nguyễn Khôi Nguyên | 2010468 | - Part 2.3.<br>- Question 1, 2 Part 3. | 100% |
| 5 | Bùi Đoàn Gia Phong | 2010509 | - Part 2.1.<br>- Question 3, 4 Part 3. | 100% |

## 2 Theoretical Basis

### 2.1 Petri Networks - Background

#### 2.1.1 Concept: What is Petri net (PetN)?

A Petri net is a graphical tool [a bipartite graph consisting of *places* and *transitions*] for the description and analysis of concurrent processes which arise in systems with many components (distributed systems). The graphics, together with the rules for their coarsening and refinement, were invented in August 1939 by Carl Adam Petri.

#### 2.1.2 Formal Definition

⋆ **Definition 1** (Transition system or State transition system):
A transition system is a triplet $TS = (S, A, T)$ where $S$ is the set of *states*, $A \subseteq \mathcal{A}$ is the set of *activities* (often referred to as *actions*), and $T \subseteq S \times A \times S$ is the set of *transitions*.
The following subsets are defined implicitly,

- $S^{start} \subseteq S$ is the set of *initial states* (sometime referred to as 'start' states), and $S^{end} \subseteq S$ is the set of *final states* (sometimes referred to as 'accept' states).
- WHY transition systems? The goal of (using transition systems in) a **process model** is to decide *which activities* need to be executed and in *what order*. Activities can be executed

sequentially, activities can be optional or concurrent, and the repeated execution of the same activity may be possible.

- BEHAVIOR of a transition system: can be studied and expressed via its net structure and dynamic. The transition starts in one of the initial states. Any path in the graph starting in such a state corresponds to a possible *execution sequence*.
  * A path *terminates successfully* if it ends in one of the final states.
  * A path *deadlocks* if it reaches a non-final state without any outgoing transitions.

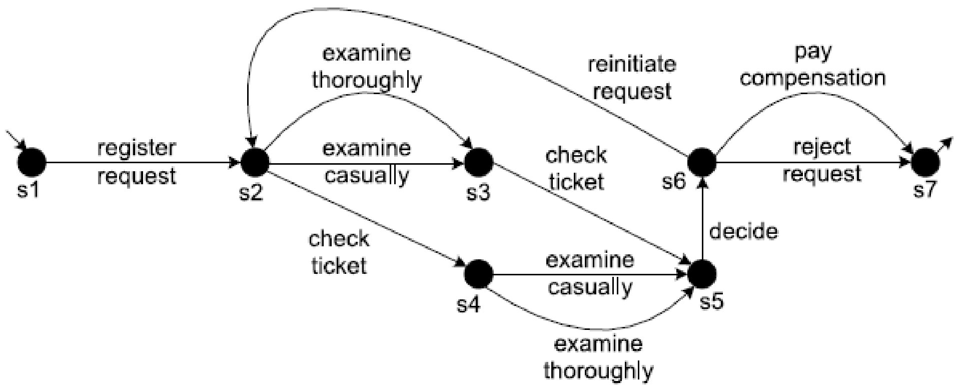


Figure 1: A small size transition system

NOTES on using transition system:

1. **Any process model with executable semantics** can be mapped onto a transition system. Therefore, many notions defined for transition systems can easily be translated to higher-level languages such as Petri nets.

2. Transition systems, however, are simple but have problems expressing concurrency succinctly, as 'state explosion'. But a **Petri Net** can be used much more compactly and efficiently.

3. Indeed, suppose that there are $n$ parallel activities, i.e., all $n$ activities need to be executed but any order is allowed. There are $n!$ possible execution sequences. The transition system requires $2^n$ states and $n \times 2^{n-1}$ transitions.

On the set of all multisets $M$ over a domain $D$:

Given a finite domain $D = \{x_1, x_2, ..., x_k\}$, a map $X\colon D \to \mathbb{N}$ defines a multi-set on $D$ as follows:

for each $x \in D$, $X(x) = m$ denotes the number of times $x$ is included in the multi-set, i.e.,

$$M = \{\underbrace{x_1, ..., x_1}_{m_1 \text{times}}, ..., \underbrace{x_k, ..., x_k}_{m_k \text{times}}\} \tag{1}$$

Evidently, *support* $(M) \subseteq D$ and we could use multiplicative format for

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

and so $M$ is identified with the list $[m_1, m_2, ..., m_k]$. Here frequencies $m_i \geq 0$, $m_i = 0$ means that $x_i$ does not appear in $M$, and support $(M)$ consists of different elements in the multi-set $M$.

◇ **EXAMPLE 1**:

A multi-set (also referred to as *bag*) is like a set in which each element may occur multiple times, and the order is **not** matter.

Given domain $D = \{a, b, c, d, e\}$, $k = |D| = 5$, we observe a multi-set with $n = 9$ elements: one $a$, two $b$'s, three $c$'s, two $d$'s, and one $e$: $M = [a, b, b, c, c, c, d, d, e] = \{a, b^2, c^3, d^2, e\} = \{e, d^2, c^3, b^2, a\}$ $\equiv [1, 2, 3, 2, 1]$.

◇ **Definition 2** (**Petri Net** is a bipartite directed graph $N$ of *places* and *transitions*):

A Petri net is a triplet $N = (P, T, F)$ where $P$ is a finite set of *places*, $T$ is a finite set of *transitions* such that $P \cap T = \emptyset$, and $F \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs, called the *flow relation*.

1. A *token* is a special *transition node*, being graphically rendered as a black dot, •. The symbolic tokens generally denote elements of the real world. Places cancontain tokens, and transitions **cannot**.

2. A transition is enabled if each of its input places contains a **token**. An **enabled transition** can *fire*, thereby consuming (*energy* of ) one token from each *input place* and producing at least one token for each *output place* next.

3. A *marking* is a distribution of tokens across places. A *marking* of net $N$ is a function $m$: $P \to \mathbb{N}$, assigning to each place $p \in P$ the number $m(p)$ of tokens at this place. Denote $M = m(P)$, the range of map $m$, viewed as a multiset.

4. A *marked* Petri net is a pair $(N, M)$, where $N = (P, T, F)$ is a Petri net and where $M$ is a *multi-set* (or *bag*, defined generally in Equation 1) over $P$ denoting the *marking* of the net.
   - We write the set of all multisets over $P$ as $\mathcal{M}(P)$ or $\mathcal{M}$ for short.
   - The set of all marked Petri nets is denoted $\mathcal{N}$.

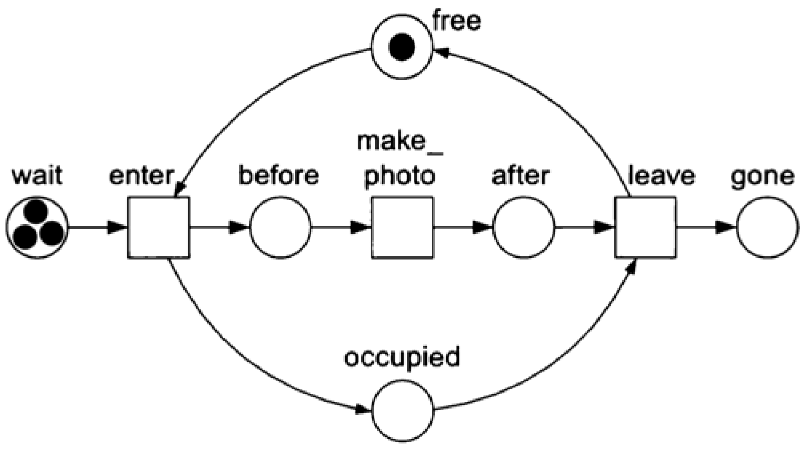


Figure 2: A Petri net for the process of an X-ray machine

◇ **EXAMPLE 2**:

The Petri net in Figure $\boxed{2}$ has three transitions: $T = \{enter,\ make\text{-}photo,\ leave\}$.
Transition *enter* is **enabled** if there is at least one token in place *wait* and at least one token in place *free*. In the marking of this net, these conditions are fulfilled. Transition *make-photo* is **enabled** if place *before* holds at least one token. This condition is **not** fulfilled.

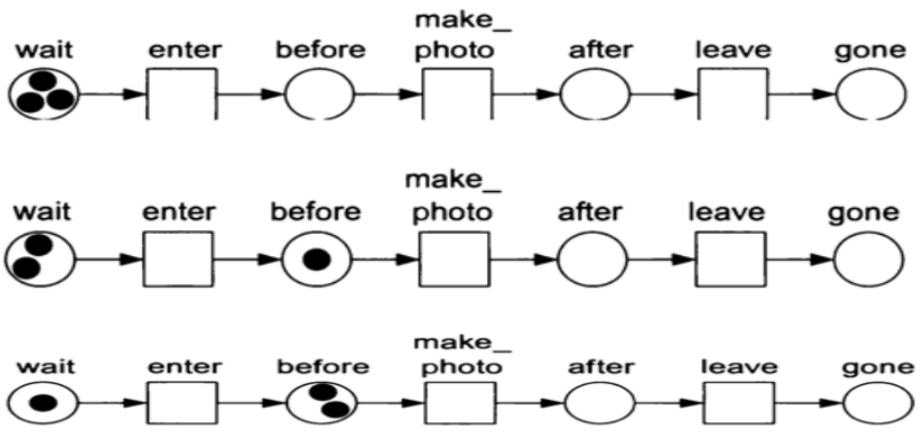
**ELUCIDATION**:

1. PLACES: In a Petri net, graphically, a place $p \in P$ is represented by a circle or ellipse. A place $p$ always models a *passive* component:
   $p$ can store, accumulate or show things. A place has discrete states.

2. TRANSITIONS: In a Petri net, graphically, a transition $t \in T$ is represented by a square or rectangle. A transition $t$ always models an *active* component:
   $t$ can produce things/tokens, consume, transport or change them.
   After each firing of a transition [consuming energy of token] the tokens are reallocated on places, henceforth building up the dynamic of **Petri net**. A redistribution or reallocation of tokens across places is a **marking**.

3. ARCS: Places and transitions are connected to each other by directed arcs, graphically, represented by an arrow. An arc **never** models a system component, but an abstract, sometimes only notional relation between components such as *logical connections*, or *access rights*.

4. OPERATIONS: The sum of two multi-sets $(A \uplus B)$, the difference $(A \setminus B)$, the presence of an element in a multi-set $(x \in M)$, and the notion of subset $(X \leq Y)$ are defined in the classic way of set theory.



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]

Figure 3: Three different markings on a Petri net, modeling of a process of an X-ray machine

⋆ **Definition 3** (Input is place, output is transition):
Let $N = (P, T, F)$ be a Petri net. Elements of $P \cup T$ are called nodes.

- A node $x$ is an *input node* of another node $y$ if and only if there is a directed arc from $x$ to $y$ (i.e., $(x, y) \in F$). Node $x$ is an *output node* of $y$ if and only if $(y, x) \in F$.
- For any $x \in (P \cup T)$, write $\bullet x = \{y \mid (y, x) \in F\}$ - the preset of $x$, and $x\bullet = \{y \mid (x, y) \in F\}$ - the postset of $x$.
- Now for any set $X$, define $X^*$ to be the set of sequences containing elements of $X$, i.e., for any $n \in \mathbb{N}$ and $x_1, x_2, ..., x_n \in X$: $(x_1, x_2, ..., x_n) \in X^*$.

### 2.1.3 On Enabled transition and Marking changes

The marked Petri net in Figure $\boxed{4}$ has the marking with only one token, node **start**.

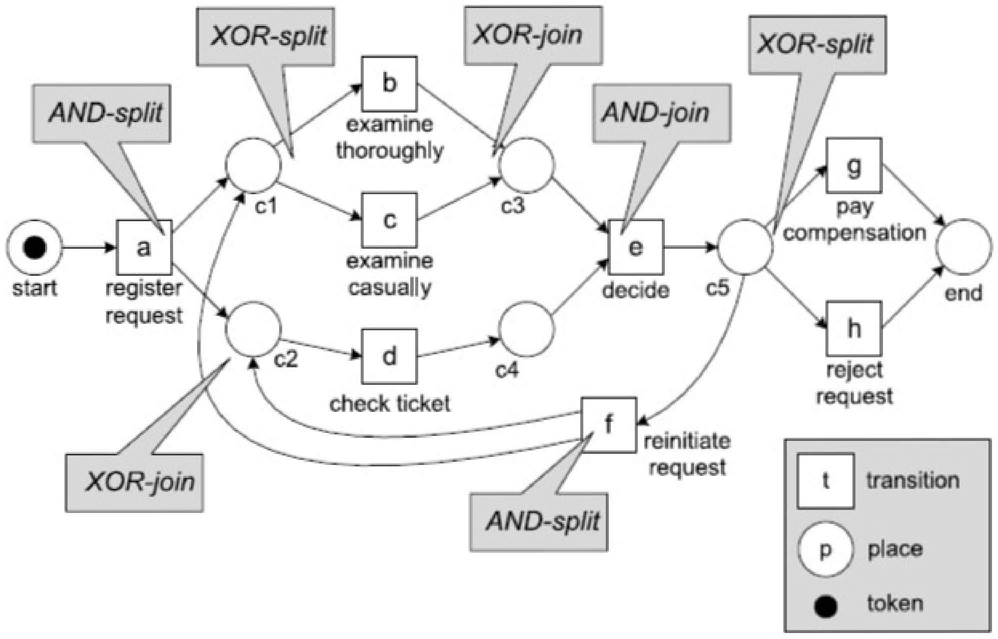


Figure 4: A marked Petri net with one initial token

- Hence, transition $a$ is enabled at marking $[start]$, now a becomes new token (with full energy).
- Firing $a$ results in the marking $[c_1, c_2]$: one token is consumed and two tokens are produced.
- At marking $[c_1, c_2]$, transition $a$ is no longer enabled (spent all energy now). However, transitions $b$, $c$, and $d$ have become enabled.
- From marking $c_1$, $c_2$, firing $b$ results in marking $[c_2, c_3]$. Here, $d$ is still enabled, but $b$ and $c$ not anymore. Because of the loop construct involving $f$ there are infinitely many firing sequences starting in $[start]$ and ending in $[end]$.
- Now with multiple token at the beginning, assume that the initial marking is: $[start^5]$. Firing $a$ now results in the marking $[start^4, c_1, c_2]$. At this marking $a$ is still enabled. Firing $a$ again results in marking $[start^3, c_1^2, c_2^2]$. Transition $a$ can fire five times in a row resulting in marking $[c_1^5, c_2^5]$.
- Note that after the first occurrence of $a$, also $b$, $c$, and $d$ are enabled and can fire concurrently.

**Task ordering principles** - The five basic task ordering principles (Figure 4) are:

1. *Sequence* pattern: putting tasks in a linear order.
2. *Or-split* pattern: selecting one branch to execute.
3. *And-split* patterns: all branches will be executed.
4. *Or-join* patterns: one of the incoming branches should be ready in order to continue.
5. *And-join* pattern: all incoming branches should be ready in order to continue.

### 2.1.4 Summary

A **Petri net** is a triplet structure $(P, T, F)$. The structure of a Petri net is determined if we know the places $P$, the transitions $T$, and the flow relation $F$ of the ways in which they are connected with each other (i.e. arcs connecting places and transitions.).

1. A **Petri net** contains zero or more places. Each place has a unique name, the place label. We can describe the places of a Petri net by the set $P$ of *place labels*.
   We can describe the transitions in a Petri net in the same way. Each transition has a unique name, the transition label. We describe the transitions of a Petri net by the set $T$ of *transition labels*.

2. **Transitions** are the *active* nodes of a Petri net, because they can change the marking through firing. We therefore name transitions with verbs to express action.
   Places are the *passive* nodes of a Petri net because they **cannot** change the marking. We name places using nouns, adjectives, or adverbs.

3. In addition to the places and transitions, we must describe the *arcs*. Like a transition in a transition system, we can represent an arc as an ordered pair $(x, y)$. The set of arcs is a *binary relation*.
   As a **Petri net** has two kinds of arcs, we obtain two binary relations:
   i) The binary relation $R_I \subseteq P \times T$ contains all arcs connecting transitions and their input places.
   ii) The binary relation $R_O \subseteq T \times P$ contains all arcs connecting transitions and their output places.

The union $R_I \cup R_O$ represents all arcs of a Petri net. This union is again a relation

$$F = R_I \cup R_O \subseteq (P \times T) \cup (T \times P) \tag{2}$$

called the **flow relation**, where $(p, t) \in F$ defines the arc from $p$ to $t$.

4. **Labeling of Arcs and Transitions**: Arcs and transitions can be labeled with *expressions* (for instance, $-$, a subtraction, and variables $x$ and $y$). These expressions have two central properties:

   i) If all variables in an expression are replaced by elements, it becomes possible to evaluate the expression in order to obtain yet another element.

   ii) The variables in these expressions are parameters describing different instances ("modes") of a transition. Such a transition can only occur if its labeling evaluates to the logical value '**true**'.

5. We can summarize the possible roles of tokens, places, and transitions with the following modeling guideline:
   **We represent events as transitions, and we represent states as places and tokens**. We represent the evolving states of a system by the distribution of tokens over the places. Each token in a place is part of the state. A token can model a physical object, information, or a combination of the two, but it can also model the state of an object or a condition.

### 2.1.5 Practical Problem

1. Consider the Petri net in Figure below:

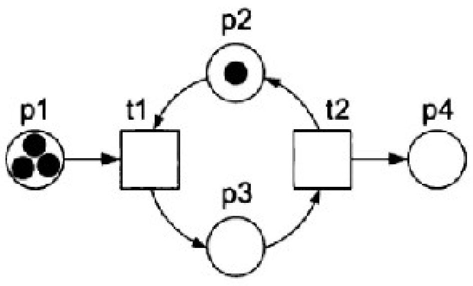


Figure 5: A simple Petri net, with only two transitions

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

- The set of places: $P = \{p_1, p_2, p_3, p_4\}$
- The set of transitions: $T = \{t_1, t_2\}$
- The two relation defined as:

$$R_I = \{(p_1, t_1), (p_2, t_1), (p_3, t_2)\}$$
$$R_O = \{(t_1, p_3), (t_2, p_2), (t_2, p_4)\}$$

- The set of flow relation:

$$F = R_I \cup R_O = \{(p_1, t_1), (p_2, t_1), (p_3, t_2), (t_1, p_3), (t_2, p_2), (t_2, p_4)\}$$

b) List presets and postsets for each transition.

$$\bullet t_1 = \{p_1, p_2\} \ ; \ t_1\bullet = \{p_3\} \ ; \ \bullet t_2 = \{p_3\} \ ; \ t_2\bullet = \{p_3, p_4\}$$

c) Determine the marking of this net.

$$M = [3.p_1, 1.p_2, 0.p_3, 0.p_4] = [3, 1, 0, 0]$$

d) Are the transitions $t_1$ and $t_2$ enabled in this net?
Transition $t_1$ is enabled, because each of the input places $p_1$ and $p_2$ holds at least one token. Transition $t_2$, in contrast, is not enabled, because input place $p_3$ is empty.
2. Given a process of a X-ray machine in which we assume the first marking in Figure $\boxed{3}$.a shows that there are three patients in the queue waiting for an X-ray. Figure $\boxed{3}$.b depicts the next marking, which occurs after the firing of transition enter.
a) Determine the two relations $R_I$ and $R_O$, and the flow relation $F = R_I \cup R_O$

- The set of places: $P = \{wait,\ before,\ after,\ gone\}$
- The set of transitions: $T = \{enter,\ make\text{–}photo,\ leave\}$
- The two relation defined as:

$$R_I = \{(wait,\ enter),\ (before,\ make\text{–}photo),\ (after,\ leave)\}$$
$$R_O = \{(enter,\ before),\ (make\text{–}photo,\ after),\ (leave,\ gone)\}$$

- The set of flow relation:

$$F = R_I \cup R_O = \{(wait,\ enter),\ (before,\ make\text{–}photo),\ (after,\ leave),\ (enter,\ before),\\ (make\text{–}photo,\ after),\ (leave,\ gone)\}$$

b) 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 theses two places to the model, to get the improved the Petri net, in Figure 6.
Now for this Petri net, can place *before* contain more than one token? Why? Rebuild the set $P$ of place labels.

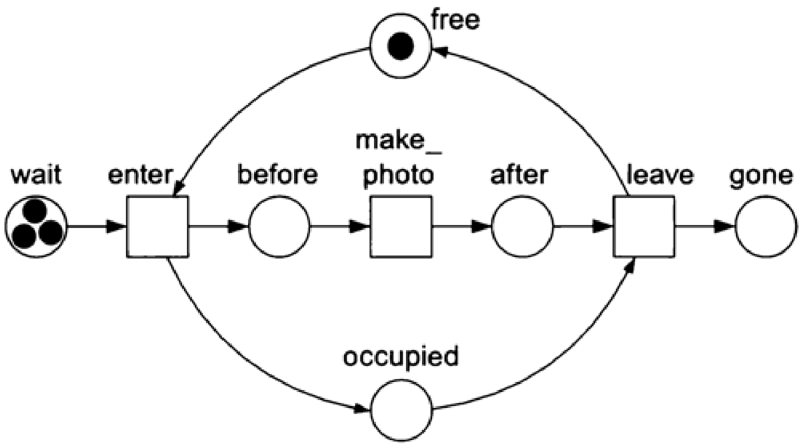


Figure 6: An improved Petri net for the business process of an X-ray machine

Place *before* cannot contain more than one token because this place receive only one token from transition *enter* and, furthermore, this transition will be enabled if and only if its input places contain at least one token. In addition, place *free* can contain only one token produced by transition *leave* and place *occupied* can contain only one token produced by transition *enter*. This means both place *before* and *after* can contain at most one token (if before get a token, after must be empty) by toggling one token between two place *free* and *occupied*.
We get the new set $P = \{wait,\ before,\ after,\ gone,\ free,\ occupied\}$.
c) As long as there is no token in place *free* (Figure 7), can transition *enter* fire again? Explain why or why not. Remake the two relations $R_I$ and $R_O$.

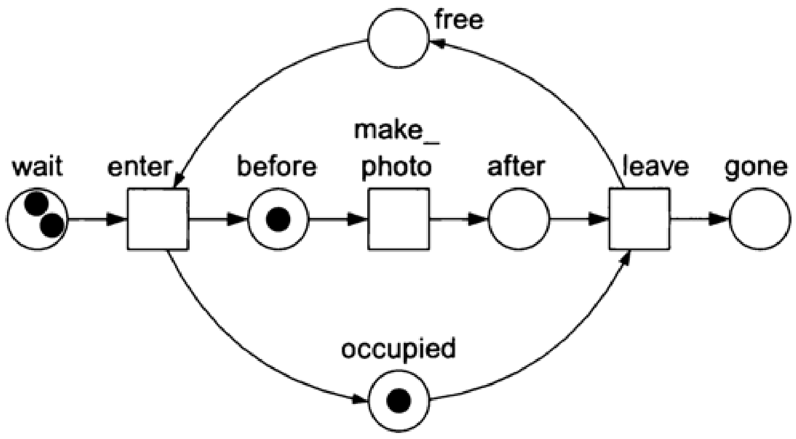


Figure 7: The marking of the improved Petri net for the working process of an X-ray room after transition *enter* has fired

As discussed above, transition *enter* cannot fire in this case because place *free* is empty.

$R_I = \{(wait,\ enter),\ (free,\ enter),\ (before,\ make\text{–}photo),\ (after,\ leave),\ (occupied,\ leave)\}$

$R_O = \{(enter,\ before),\ (enter,\ occupied),\ (make\text{–}photo,\ after),\ (leave,\ gone),\ (leave,\ free)\}$

## 2.2 Petri Networks - Behavior

### 2.2.1 From Firing, Reachability to Labeled Petri net

Process models, such as Petri nets, describe the behavior of a system, which is determined by all possible state transitions. To formally describe the behavior of a Petri net, we first have to define the term marking (this has been defines in **Definition 2)**.

The marking of a Petri net is determined by the distribution of tokens over the places of the net. A token is graphically rendered as a black dot, $\bullet$. Places can contain tokens, transitions cannot. To describe the marking of the Petri net, we must specify the number of tokens in each place. Places in a Petri net hold tokens, whereas transitions can change the marking through firing.

⋆ **Definition 4** :

To fire a transition, it must be enabled. A transition is enabled if there is at least one token in each of its input places.

Let $(N, M) \in \mathcal{N}$ be a marked Petri net with $N = (P, T, F)$ and $M \in \mathcal{M}$.

- Transition $t \in T$ is enabled at marking $M$, denoted $(N, M)[t\rangle$, if and only if $\bullet t \leq M$.

- The firing rule $\alpha[t\rangle\beta \subseteq \mathcal{N} \times T \times \mathcal{N}$ is the smallest relation satisfying

$$(N, M)[t\rangle \Longrightarrow \underbrace{(N, M)}_{\alpha}[t\rangle \underbrace{(N, (M \setminus \bullet t) \uplus t\bullet)}_{\beta} \tag{3}$$

  for any $(N, M) \in \mathcal{N}$ and any $t \in T$.

♣ **OBSERVATION 1**:

1. Places can contain tokens, transitions cannot. But transitions can change the marking through firing. That is places are passive, and transitions are active. To fire a transition, it must be enabled.

2. **Enabledness**: A marking $M = [m_1, m_2, ..., m_k] \equiv m : P \to \mathbb{N}$, by $\boxed{(1)}$ has the form

$$M = \{\underbrace{x_1, ..., x_1}_{m_1\text{times}}, ..., \underbrace{x_k, ..., x_k}_{m_k\text{times}}\},$$

   so we can equivalently say $t \in T$ is *enabled* at $M$ if and only if $\forall p \in \bullet t,\ m(p) > 0$.

3. An enabled transition $t$ can fire, thereby changing the marking $M$ to a marking $M_1$. If an enabled transition fires, it consumes one token from each of its input places and produces one token in each of its output places. We can calculate the successor marking $M_1$ from marking $M$ as follows: the number $M(p)$ of tokens in a place $p$ is equal to the number $M(p)$ of tokens in $p$ minus the consumed tokens plus the produced tokens. The number of consumed tokens is equal to one if $p$ is an input place of $t$ and zero else. Likewise, the number of produced tokens is equal to one if $p$ is an output place of $t$ and zero else.
   We can formalize the effect of the firing of a transition by a function $W$, assigning to each arc in flow relation $F$ the value $1$ and to each arc not contained in flow relation $F$ the value $0$. Formally, this function is defined by $W : (P \times T) \cup (T \times P) \to \{0, 1\}$ with $W((x, y)) = 1$ if $(x, y) \in F$ and $W((x, y)) = 0$ if $(x, y) \notin F$. Function $W$ is a weight function.

4. $(N, M)[t\rangle$ means that transition $t$ is enabled at marking $M$.

5. $(N, M)[t\rangle(N, M_1)$ denotes that firing this enabled transition $t$ results in marking $M_1$.

⋆ **Definition 5** (Transition firing):
For a Petri net $N = (P, T, F)$, let $W$ be the weight function and $m : P \to \mathbb{N}$ be the current marking. A transition $t \in T$ can fire if and only if it is enabled at $m$. The firing of $t$ yields a new

marking $m^{'} : P \to \mathbb{N}$ where for all places $p \in P$, $m \setminus p = m(p) - W((p, 0)) + W((f, p))$.

◇ **EXAMPLE 4**:

Consider the Petri net in Figure $\boxed{5}$. Transition $t_1$ is enabled at marking $m$ depicted in Figure $\boxed{5}$. The firing of $t_1$ yields the marking $m_1$, defined as:

$$\begin{aligned}
m_1(p_1) &= m(p_1) - W((p_1, t_1)) + W((t_1, p_1)) = 3 - 1 + 0 = 2 \\
m_1(p_2) &= m(p_2) - W((p_2, t_1)) + W((t_1, p_2)) = 1 - 1 + 0 = 0 \\
m_1(p_3) &= m(p_3) - W((p_3, t_1)) + W((t_1, p_3)) = 0 - 0 + 1 = 1 \\
m_1(p_4) &= m(p_4) - W((p_4, t_1)) + W((t_1, p_4)) = 0 - 0 + 0 = 0
\end{aligned}$$

⋆ **Definition 6** (Firing sequence):

Let $(N, M_0) \in \mathcal{N}$ be a marked Petri net with $N = (P, T, F)$.

1. A sequence $\sigma \in T^*$ is called a *firing sequence* of $(N, M_0)$ **if and only if**, for some natural number $n \in \mathbb{N}$, there exist markings $M_1, M_2, ..., M_n$ and transitions $T_1, T_2, ..., T_n$ such that
   - $\sigma = (t_1, t_2, ..., t_n) \in T^*$
   - and $\forall i$ with $0 \leq i < n$, then $(N, M_i)[t_{i+1}\rangle$ and $(N, M_i)[t_{i+1}\rangle(N, M_{i+1})$
2. A marking $M$ is reachable from the initial marking $M_0$ **if and only if** there exists a sequence of enabled transitions whose firing leads from $M_0$ to $M$.
   The set of reachable markings of $(N, M_0)$ is denoted $[N, M_0\rangle$.
3. (Petri net system) - A Petri net system $(P, T, F, M_0)$ consists of a Petri net $(P, T, F)$ and a distinguished marking $M_0$, the *initial marking*.

⋆ **Definition 7** (Labeled Petri net):

Often transitions are identified by a single letter, but also have a longer label describing the corresponding activity. A **labeled Petri net** with $N = (P, T, F, A, l)$ where $(P, T, F)$ is a Petri net as defined in **Definition 2**.

- $A \subseteq \mathcal{A}$ is a set of *activity labels*, and the map $l \in \{L : T \to A\}$ is a *labeling function.* [One can think of the transition label as the *observable action.* Sometimes one wants to express that particular transitions are **not** observable, or invisible.].
- Use the label $\tau$ for a special activity label, called 'invisible'. A transition $t \in T$ with $l(t) = \tau$ is said to be unobservable, silent or invisible.

♣ **OBSERVATION 2**:

1. The transitions keep firing until the net reaches a marking that **does not** enable any transition. Like the terminal state in transition systems (**Definition 1**), this marking is a *terminal marking.*

2. We may convert any Petri net into a labeled Petri: just take $A = T$ and $l(t) = t$ for any $t \in T$. The reverse is **not always** possible (many transitions have the same label).

As we know the initial marking $M_0$ for the given system $(N, M_0)$, we answer such questions by calculating the set of markings reachable from $M_0$. We represent this set as a graph - the **reachability graph** of the net. Its nodes correspond to the reachable markings and its edges to the transitions moving the net from one marking to another.
The key structure is **reachability graph** via transition systems.
◇ **EXAMPLE 5** (Reachability graph):
Consider the Petri net system in Figure 8 modeling the four seasons:

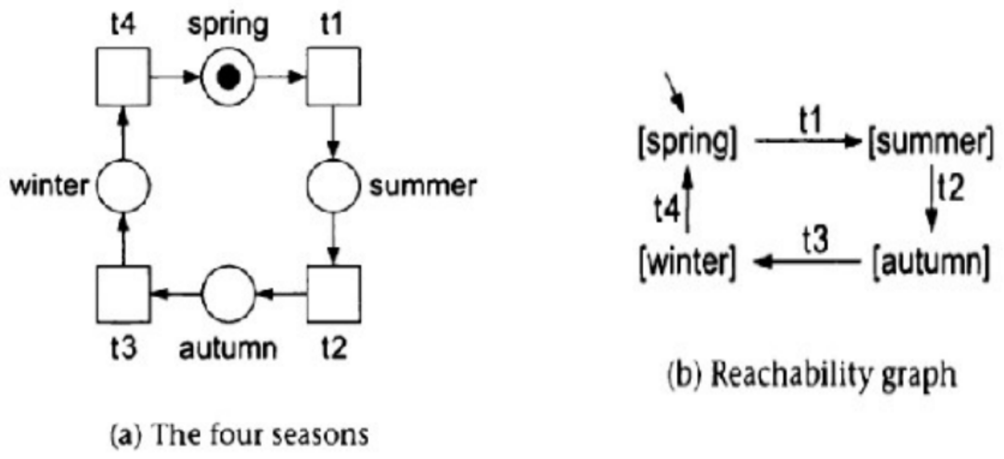


Figure 8: A Petri net system and its reachability graph

Recall that, each of the reachable markings is represented as a *multiset* (where the same element may appear multiple times). Multiset [*spring*] thus represents the marking in Figure 8.a

- The incoming edge **without source** pointing to this node denotes that this marking is the initial marking. We labeled each edge of the reachability graph on the right with the transition that fired in the corresponding marking.

- Figure 8.b depicts the accompanying reachability graph that represents the set of markings that are reachable from the initial marking shown in Figure $\boxed{8}$.a. We can conclude that the net in Figure $\boxed{8}$.a has four reachable markings.

- If a marking $M$ is reachable from the initial marking $M_0$, then the reachability graph has a path from the start node to the node representing marking $M$. This path represents a sequence of transitions that **have to be fired** to reach marking $M$ from $M_0$.

We refer to this transition sequence as a *run* (as an execution in finite automaton). A run is *finite* if the path and hence the transition sequence is finite. Otherwise, the run is *infinite*.

### 2.2.2 Representing Petri Nets as Special Transition Systems

Our discussion shows that we can verify certain properties of a Petri net system by inspecting its reachability graph. For a simple Petri net system, it is easy to construct the accompanying reachability graph, but for more complex nets, reachability graphs can become huge, and it is possible to **forget markings**.

**GENERIC AIM**:

We describe the behavior of a Petri net system $(N, M_0) \equiv (P, T, F, M_0)$ as a state transition system $(S, TR, S_0)$ by showing how to determine the state space $S$, the transition relation $TR$, and the initial state $S_0$ for the system $(P, T, F, M_0)$.

**Why are state transition systems suitable for representing Petri Nets**?

The transition system represents the state space of the modeled system, thus representing all possible markings $M$ of the net.

⋆ **Definition 8** (Reachability graph):

Let $(N, M_0)$ with $N = (P, T, F, A, l)$ be a marked labeled Petri net.

$(N, M_0)$ defines a transition system $TS = (S, A_1, TR)$ with $S = [N, M_0\rangle$, $S^{start} = \{M_0\}, A_1 = A$, and $TR = \{(M, M_1) \in S \times S \mid \exists t \in T(N, M)[t\rangle(N, M_1)\}$, or with label $l(t)$:

$$TR = \{(M, l(t), M_1) \in S \times A \times S \mid \exists t \in T(N, M)[t\rangle(N, M_1)\} \tag{4}$$

$TS$ is often referred to as the **reachability graph** of $(N, M_0)$.

**ELUCIDATION**:

- Elements of $A$ in net $N$ are *labels*, but when transformed to $A_1$ they are called *actions* in the output transition system $TS$.

- The initial state $S_0$ is defined by the initial marking: $S_0 = S^{start} = \{M_0\}$. The markings $M$ (multisets) in $N$ becomes the states in $TS$.
- The description of the transition relation $TR$ for the Petri net system is more delicate. Let us consider two arbitrary states in state space $S$ - that is, two markings $M$ and $M_1$.
- Transition $(M, M_1)$ is an element of the transition relation $TR$ if there is a transition $t \in T$ enabled at marking $M$, and the firing of $t$ in marking $M$ yields marking $M_1$.
  We formalize this by defining transition relation $TR$ as the set of all pairs $(M, M_1) \in S \times S$ satisfying $\exists t \in T$: $(N, M)[t\rangle(N, M_1)$.
  Otherwise, transition $(M, M_1)$ is not possible, and $(M, M_1)$ is not an element of $TR$.
- The set $\mathcal{M}$ (all markings) contains markings that are reachable from the given initial marking $M_0$, but also markings that are not. As a result, for a given marked Petri net $(N, M_0)$, its reachability graph $TS$ is a subgraph of the full transition system.

### 2.2.3 Practical Problem

3. Build up the transition system $TS$ generated from the labeled marked Petri net shown in Figure 4.

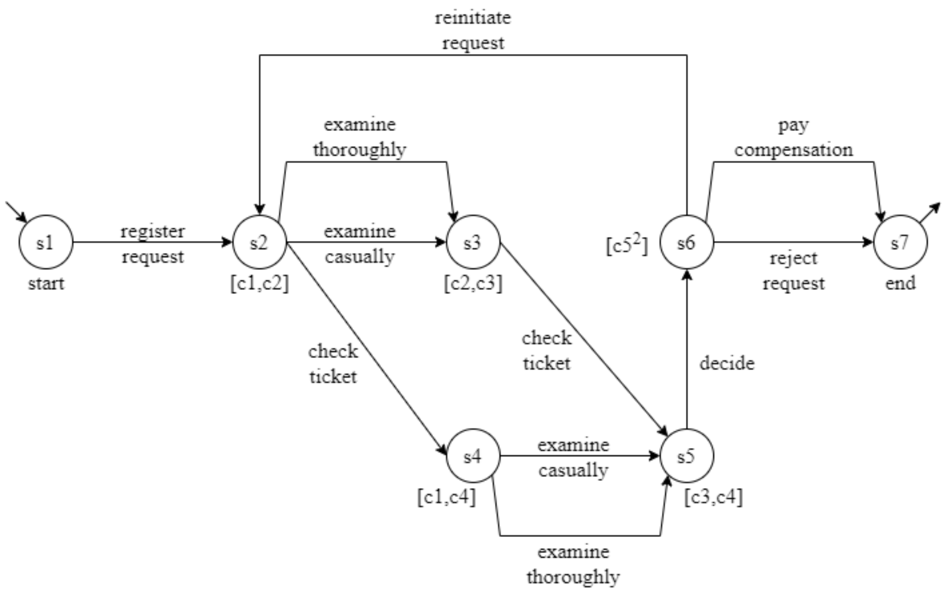


Figure 9: The reachability graph $TS$ of the labeled marked Petri net in Figure 4

## 2.3 Petri Networks - Structures and Basic Problems

Places and transitions in a Petri net are connected by *arcs*. Connecting of nodes determines the behavior of the network. Formally, the way in which transitions are connected determines the order in which they can fire. First we recall key terms, rules and relevant ideas, in a **Petri net** $N = (P, T, F)$.

1. When a transition $t$ fires, the resulting number of tokens in any place $p$ is equal to the initial number of tokens **minus** the number of consumed tokens plus the number of produced tokens.

2. The total number of tokens in the net changes if the number of input places of transition $t$ is **not** the same as the number of output places of transition $t$. Accordingly, the *firing of a transition* may increase or decrease the overall number of tokens.

3. When several transitions are *enabled* at the same moment, it is **not** determined which of them will fire. This situation is a nondeterministic choice. Even though we **do not** know in this case which transition will fire, we know that one of them will be fired.

### 2.3.1 Causality, Concurrency and Synchronization

In general, transitions represent *events*. Let us assume that there are three events: $x$, $y$ and $z$. We first discuss typical network structures to state that:

1. Event $y$ happens after event $x$.

2. Event $x$ and event $y$ take place concurrently (at the same time or in any order).

3. Event $z$ happens after both event $x$ and $y$.

⋆ **Definition 9** :

1. **Causality** is formally understood as a relationship between two events in a system that must take place in a certain order. In a **Petri net** $N$, we may represent this relationship by two transitions connected through an intermediate place. In Figure $\boxed{10}$, transition $y$ can fire only after transition $x$ has fired.

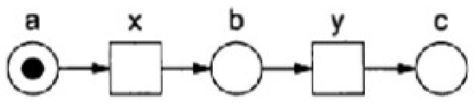


Figure 10: Causality in a Petri net

2. **Concurrency** (i.e., parallelism) is an important feature of (information) systems. In a concurrent system, several events **can occur simultaneously**. For example, several users may access an information system like a database at the same time.

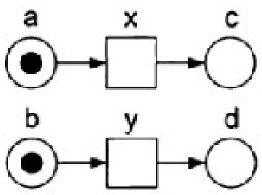


Figure 11: Concurrency in a Petri net

In Figure 11, transitions $x$ and $y$ can fire independently of each other; that is, there is **no causal relationship** between the firing of $x$ and of $y$. With network structures, as those shown in Figure 11, we can model concurrency. In concurrent models, there is often a need for **synchronization**.

3. We can model synchronization in a **Petri net** as a transition with at least two input places. In Figure 12, transition $z$ has two input places and **can only fire** after transitions $x$ and $y$ have fired.

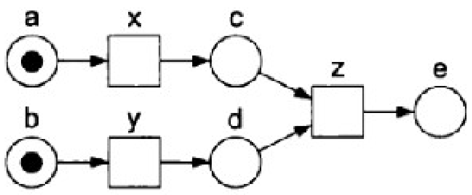


Figure 12: Synchronization in a Petri net

In industry or any process, assume that transitions $x$ and $y$ represent two concurrent production steps. Transition $z$ can then represent an assembly step that can take place only after the results of the two previous production steps are available, see Figure 12.

### 2.3.2 Practical Problem

4. Consider the Petri net system in Figure 13:

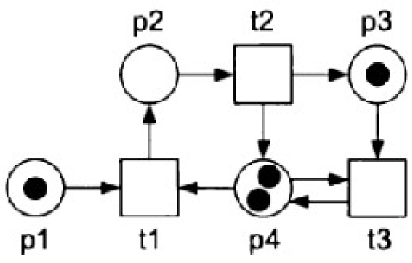


Figure 13: A Petri net with small numbers of places and transitions

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

b) Give the preset and the postset of each transition.

$$p_1\bullet = \{t_1 \mid (p_1, t_1) \in F\}$$
$$p_2\bullet = \{t_2 \mid (p_2, t_2) \in F\}$$
$$p_3\bullet = \{t_3 \mid (p_3, t_3) \in F\}$$
$$\bullet p_3 = \{t_3 \mid (t_3, p_3) \in F\}$$

c) Which transitions are enabled at $M_0$? $t_1$

d) Give all reachable markings. What are the reachable terminal markings? $p_2$, $p_3$

e) Is there a reachable marking in which we have a nondeterministic choice? $t_3$

f) Does the number of reachable markings increase or decrease if we remove

(i) place $p_1$ and its adjacent arcs? No

(ii) place $p_3$ and its adjacent arcs? Yes

5. Consider a marked Petri net $N = (P, T, F)$ with $|P| = 4 = n$, with the $4$ start places each initially get $3$ tokens, and the exit node is specially designated as place OUT. Besides, assume $|T| = 4$, and the initial marking is $M_0$, as seen in Figure 14. Denote $TS$ for the whole transition system made by the Petri net $N = (P, T, F)$.

a) Write down $M_0, P, T$ of $N$.

- The initial marking of the Petri net: $M_0 = \{3.a, 3.b, 3.c, 3.d\}$.

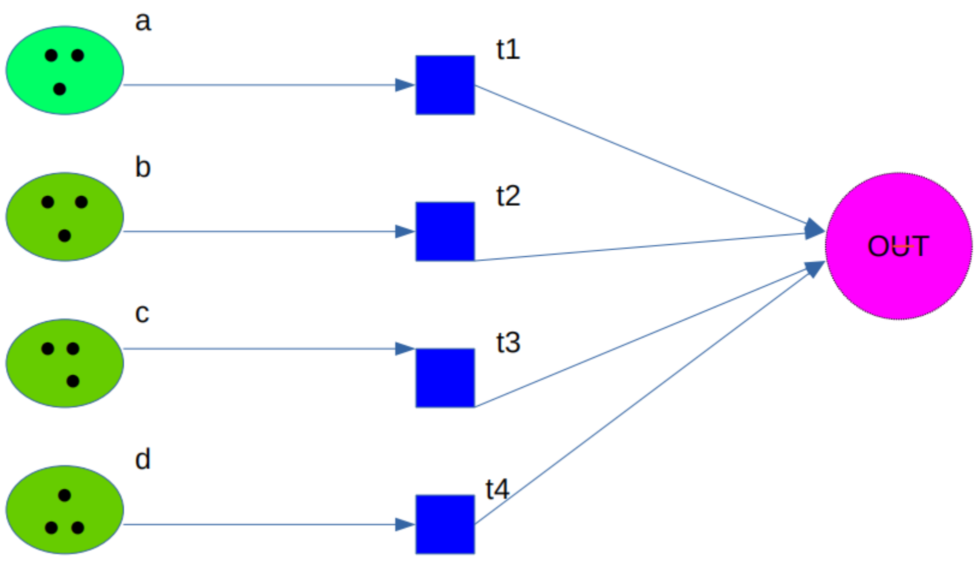


Figure 14: A Petri net allows concurrency

- The set of places: $P = \{a, b, c, d, \text{ OUT}\}$.
- The set of transitions: $T = \{t_1, t_2, t_3, t_4\}$.

b) 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?
For every change in places other than OUT, we will all have different marking states, so to simplify the problem we only consider the set of markings in places except OUT.
As we can see for each place $p_i$ with $i = 1, 2, 3, 4$, there will be $4$ marking states corresponding to each place, belonging to the set $A = \{3, 2, 1, 0\}$ which means the number of tokens in each place can be integers $3, 2, 1$ or $0$. Considering the set of markings of $p_1$ and $p_2$, we have the combinations $(M(p_1), M(p_2)) = \{(3, 3), (3, 2), (3, 1), (3, 0), (2, 3), (2, 2), (2, 1), (2, 0), (1, 3), (1, 2), (1, 1), (1, 0), (0, 3), (0, 2), (0, 1), (0, 0)\}$ are $4 \times 4 = 4^2$ states, so similarly for $(M(p_1), M(p_2), M(p_3))$ is $4^3$ and $(M(p_1), M(p_2), M(p_3), M(p_4))$ is $4^4$.
Let the set of marking values of places except place OUT be $W$, assuming $W_1 = (a, b, c, d)$ where $a, b, c, d$ are integers and $a, b, c, d < 3$ then we can always have paths from state $W_1$ to state $W_0 = (3, 3, 3, 3)$ which is the initial state of $p_1, p_2, p_3, p_4$ by adding $1$ to the numbers one by one, $a$ or $b$ or $c$ or $d$ until $a = b = c = d = 3$, each addition is a place value in the set $W$. So prove that for every value in $W$ there are always paths from $W_0$.

So it proves that in the Petri net of the problem, there will be $4^4$ corresponding reachability marking states. So when moving from this Petri net to Transition System, that TS will have $4^4 = 256$ states.

## 3 Petri Networks - Assignment on Modelling: Consulting Medical Specialists

**SCENARIO**: Under a SARS pandemic where a huge lack of ICU beds occurs in city H, patients should consult specialists in the outpatient clinic of a hospital, we describe the course of business around a specialist in this outpatient clinic of hospital X as a process model, formally, we use **Petri net**.

**ASSUMPTION and DATA**:

- **Specialist**: Each patient has an appointment with a certain specialist. The specialist receives patients. At each moment, the specialist is in one of the following three states:

  (1): the specialist is free and waits for the next patient (state *free*),

  (2): the specialist is busy treating a patient (state *busy*), or

  (3): the specialist is documenting the result of the treatment (state *docu*).

- **Every patient** who visits a specialist is in one of the following three states:

  (1): the patient is waiting (state *wait*, gets value $n$ if there are $n$ patients waiting),

  (2): the patient is treated by the specialist (state *inside*), or

  (3): the patient has been treated by the specialist (state *done*).

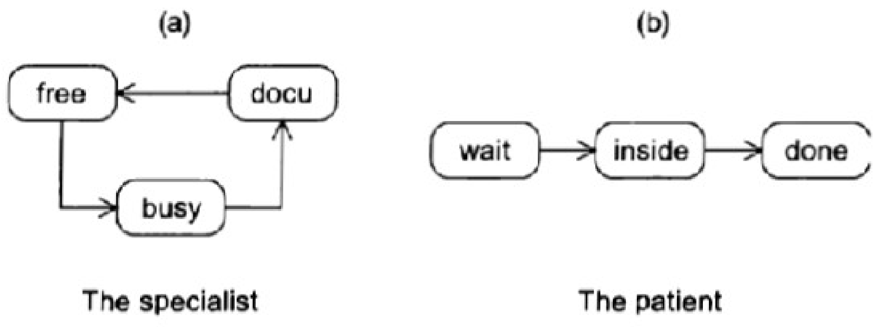


Figure 15: The transition systems of a specialist and a patient

Figure 15 shows the states of the specialist and the states of a patient. The specialist goes through the three states in an iterative way. A patient goes through these states only once (per visit).

- **Event data**: three events are important for this business process.
  First, the specialist starts with the treatment of a patient (event "start").
  Second, the specialist finishes the treatment of a patient and starts documenting the results of the treatment (event "change").
  Third, the specialist stops documenting (event "end").

**OBJECTIVE**: We model this business process (step by step) as a Petri net.

**1. Given the Petri net $N_S$ modeling the state of the specialist, as in Figure 16.**

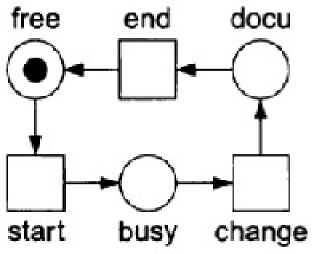


Figure 16: The Petri net of the specialist's state

**In the displayed marking, the specialist is in state *free*.**

**a) Write down states and transitions of the Petri net $N_S$.**

<u>**Solution**</u>:

$N_S$ is a marked Petri net, where:

$P = \{free,\ busy,\ docu\}$ is the set of places.

$T = \{start,\ change,\ end\}$ is the set of transitions.

$F = \{(free,\ start),\ (start,\ busy),\ (busy,\ change),\ (change,\ docu),\ (docu,\ end),\ (end,\ start)\}$ is the set of flow relations.

$M_0 = [free] = [1, 0, 0]$ is the initial marking.

According to Figure 16, we get 2 reachable markings $M_1 = [busy] = [0, 1, 0]$ and $M_2 = [docu]$

$= [0, 0, 1]$. In place *docu*, the token is going to return to the place *free.* when the transition *end* fires. Therefore, there are 3 states in total corresponding to 3 reachable markings discussed above.

**b) Could you represent it as a transition system assuming that**

**(i) Each place cannot contain more than one token in any marking.**

**Solution**:

We get marking distribution that each place cannot contain more than one token as following:

$$M = \{[0, 0, 0], [1, 0, 0], [0, 1, 0], [0, 0, 1], [1, 1, 0], [1, 0, 1], [0, 1, 1], [1, 1, 1]\}$$

As we known, the set of states $S = [N_S, M_0\rangle$, so

$$S = \{(0, 0, 0), (1, 0, 0), (0, 1, 0), (0, 0, 1), (1, 1, 0), (1, 0, 1), (0, 1, 1), (1, 1, 1)\}$$

The initial marking $M_0 = [1, 0, 0]$, so $S^{start} = \{M_0\} = (1, 0, 0)$.

Therefore, we obtain the transition relation

$$\begin{aligned} TR = \{&((1, 0, 0), (0, 1, 0)); ((0, 1, 0), (0, 0, 1)); ((0, 0, 1), (1, 0, 0)); ((1, 1, 0), (1, 0, 1)); ((1, 0, 1), (0, 1, 1));\\ &((0, 1, 1), (1, 1, 0))\} \end{aligned}$$

Because of the number of tokens constraint, each place can contain at most one token, so $TR$ cannot include transitions with state $(1, 1, 1)$ like $((1, 1, 1), (0, 2, 1))$ or $((1, 1, 1), (1, 0, 2))$.

**(ii) Each place may contain any natural number of tokens in any marking.**

**Solution**:

Because there is no restriction of the number of tokens, we can express set $S$ as following:

$$S = \{(x, y, z) \mid x, y, z \in \mathbb{N}\}$$

With the initial state $S_0 = M_0 = (1, 0, 0)$, the transition relation $TR$ can be represented by the union of three subsets:

$$\begin{aligned} TR = &\{((x + 1, y, z), (x, y + 1, z))|x, y, z \in \mathbb{N}\}\\ &\cup \{((x, y + 1, z), (x, y, z + 1))|x, y, z \in \mathbb{N}\}\\ &\cup \{((x, y, z + 1), (x + 1, y, z))|x, y, z \in \mathbb{N}\} \end{aligned}$$

When a transition (*start*, *change* or *end*) fires, it will consume a token from preset (input of transition) to produce a token to postset (output of transition).

**2. Figure $\boxed{16}$ of net $N_S$ is made by information of Specialist and Event data above. Define $N_{Pa}$ as the Petri net modeling the state of patients. By the similar ideas,**

**a) explain the possible meaning of a token in state *inside* of the net $N_{Pa}$**

**Solution**:

According to Event data above, a token in state inside of the net $N_{Pa}$ means the patient is treated by the specialist. In addition, the token representing the specialist must be in place *busy* to satisfy this case.

**b) 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*.**

**Solution**:

We construct $N_{Pa}$ by defining triplet $(P, T, F)$ and initial marking: $P = \{wait,\ inside,\ done\}$

$T = \{start,\ change\}$

$F = \{(wait,\ start),\ (start,\ inside),\ (inside,\ change),\ (change,\ done)\}$

$M_0 = [5.wait,\ done] = [5, 0, 1]$

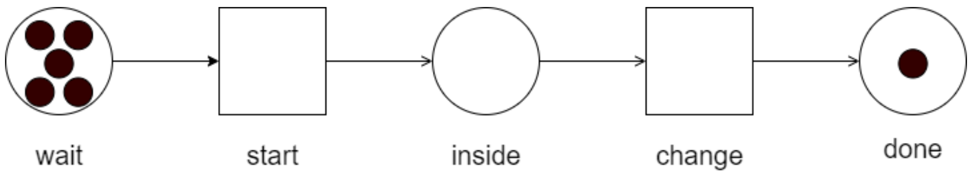


Figure 17: The Petri net of the patient's state

**3. 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).**

**Solution**:

The superimposed Petri net $N = N_S \oplus N_{Pa}$ defined by following sets:

$P = P_S \cup P_{Pa} = \{free,\ busy,\ docu,\ wait,\ inside,\ done\}$

$T = T_S \cup T_{Pa} = \{start,\ change,\ end\}$

$F = \{(free,\ start),\ (wait,\ start),\ (start,\ busy),\ (start,\ inside),\ (busy,\ change),\ (inside,\ change),\ (change,\ done),\ (change,\ docu),\ (docu,\ end),\ (end,\ free)\}$

$M_0 = [4.wait,\ 0.inside,\ 1.done,\ 1.free,\ 0.busy,\ 0.docu] = [4, 0, 1, 1, 0, 0]$

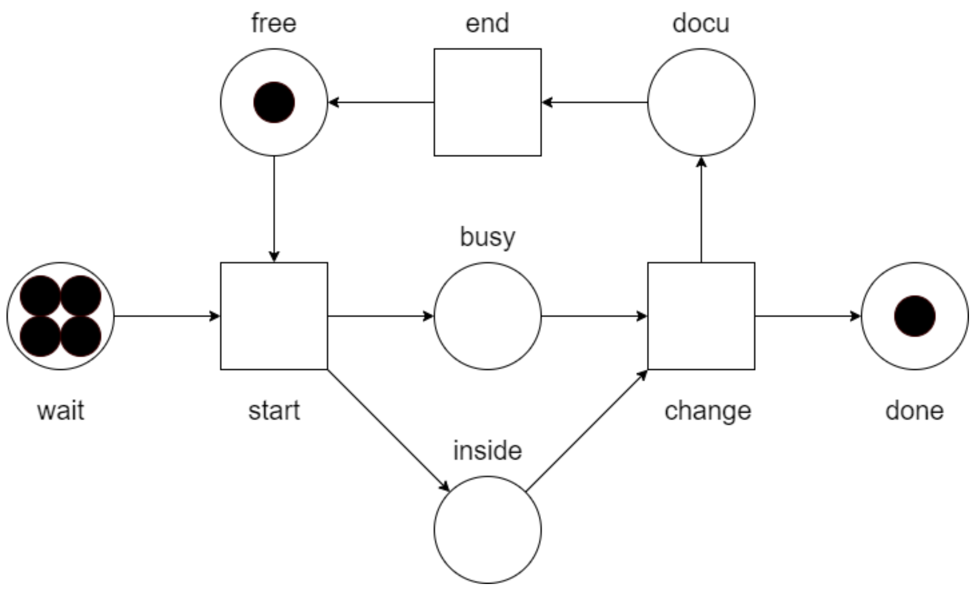


Figure 18: The superimposed Petri net

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

**Solution**:

The grand Petri net and given initial marking is shown in Figure $\boxed{19}$ below. The figure shows that only transition *start* is enabled, so the reachable marking from $M_0$ is

$$M_1 = [2.wait,\, 1.inside,\, 1.done,\, 0.free,\, 1.busy,\, 0.docu] = [2, 1, 1, 0, 1, 0]$$

We can express this firing as following:

$$(N,\ [3, 0, 1, 1, 0, 0])[start\rangle(N,\ [2, 1, 1, 0, 1, 0])$$

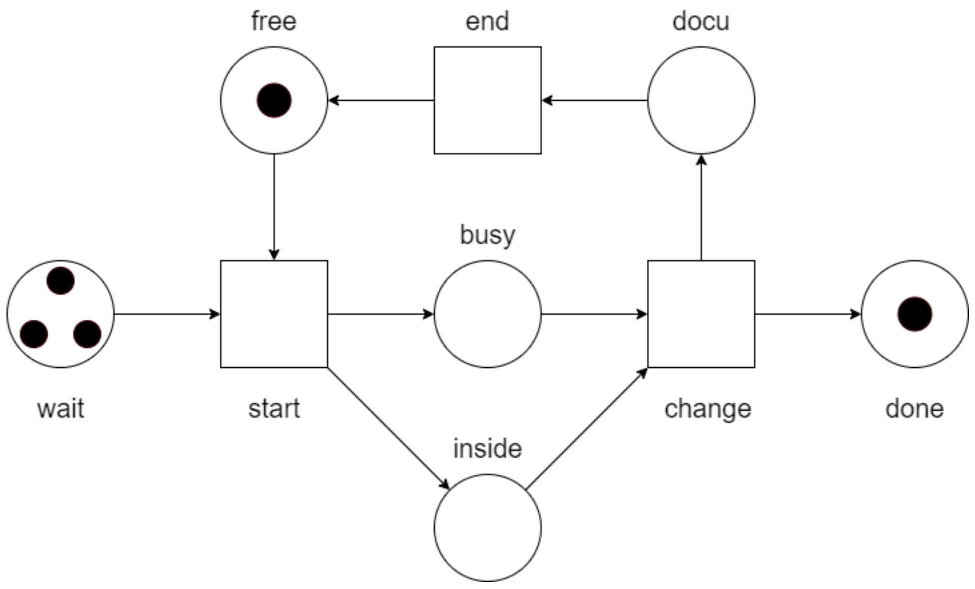


Figure 19: The grand net with given initial marking

**5. Is the superimposed Petri net $N$ deadlock free? Explain properly.**

**Solution**:

The grand net is deadlock free. We can prove this by tracing all reachable markings with firing sequence $\sigma = (start,\ change,\ end)^3$:

$$(N,\ [3,0,1,1,0,0])[start\rangle(N,\ [2,1,1,0,1,0])$$
$$(N,\ [2,1,1,0,1,0])[change\rangle(N,\ [2,0,2,0,0,1])$$
$$(N,\ [2,0,2,0,0,1])[end\rangle(N,\ [2,0,2,1,0,0])$$
$$(N,\ [2,0,2,1,0,0])[start\rangle(N,\ [1,1,2,0,1,0])$$
$$(N,\ [1,1,2,0,1,0])[change\rangle(N,\ [1,0,3,0,0,1])$$
$$(N,\ [1,0,3,0,0,1])[end\rangle(N,\ [1,0,3,1,0,0])$$
$$(N,\ [1,0,3,1,0,0])[start\rangle(N,\ [0,1,3,0,1,0])$$
$$(N,\ [0,1,3,0,1,0])[change\rangle(N,\ [0,0,4,0,0,1])$$
$$(N,\ [0,0,4,0,0,1])[end\rangle(N,\ [0,0,4,1,0,0])$$

We get 9 reachable markings in total, all tokens in place *wait* at initial marking move to place *done* and, furthermore, there always has a transition that is enabled at every reachable marking throughout the whole process until place *free* has a token and place *wait* gets empty.

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

**Solution**:

We construct a Petri net for 2 specialists by constructing 2 similar subnets to specify states of each one. As Figure $\boxed{20}$ shows below, we have 2 spaces for 2 specialists, but place *free* is constructed as a sharing space for both of them. Place *busy* and *docu* of both subnets is similar to the given original Petri net. This net has 2 new places available to indicate that each subnets is occupied or not. If a subnet is occupied (there's no token in place *available*), the transition *start* isn't enabled, therefore, each subnet can only contain at most one token which represents a specialist. When any transition start fires, it consumes a token from the matching place available and produces a token to the other one. Therefore, a transition *start* that has fired cannot fire again until the other one fires. This net avoids 2 problems. First, if a subnet is being occupied by a specialist, the other specialist cannot enter this subnet. Second, this net also avoids resource disputes, the number of patients that are treated by each specialist is approximately half of the total patients, both transition starts take turn consuming tokens that represents the patients.

The marked Petri net $(N, M_0)$ is defined as following:

$P = \{free,\ busy_1,\ busy_2,\ docu_1,\ docu_2,\ available_1,\ available_2\}$

$T = \{start_1,\ start_2,\ change_1,\ change_2,\ end_1,\ end_2\}$

$F = \{(free,\ start_1),\ (free,\ start_2),\ (available_1,\ start_1),\ (available_2,\ start_2),\ (start_2,\ available_1),\ (start_1,\ available_2),\ (start_1,\ busy_1),\ (start_2,\ busy_2),\ (busy_1,\ change_1),\ (busy_2,\ change_2),\ (change_1,\ docu_1),\ (change_2,\ docu_2),\ (docu_1,\ end_1),\ (docu_2,\ end_2),\ (end_1,\ free),\ (end_2,\ free)\}$

$M_0 = [2.free,\ 1.available_1,\ 0.available_2,\ 0.busy_1,\ 0.busy_2,\ 0.docu_1,\ 0.docu_2]$

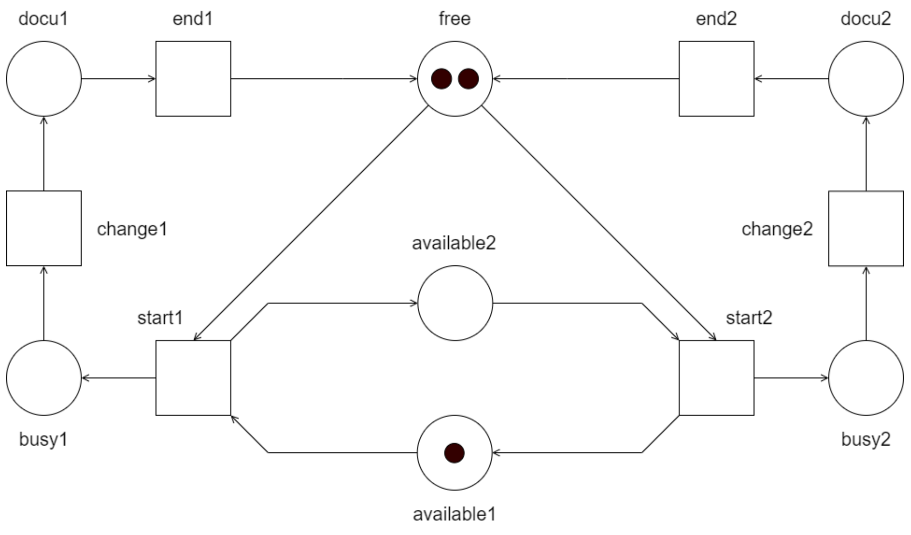


Figure 20: The Petri net for 2 specialists

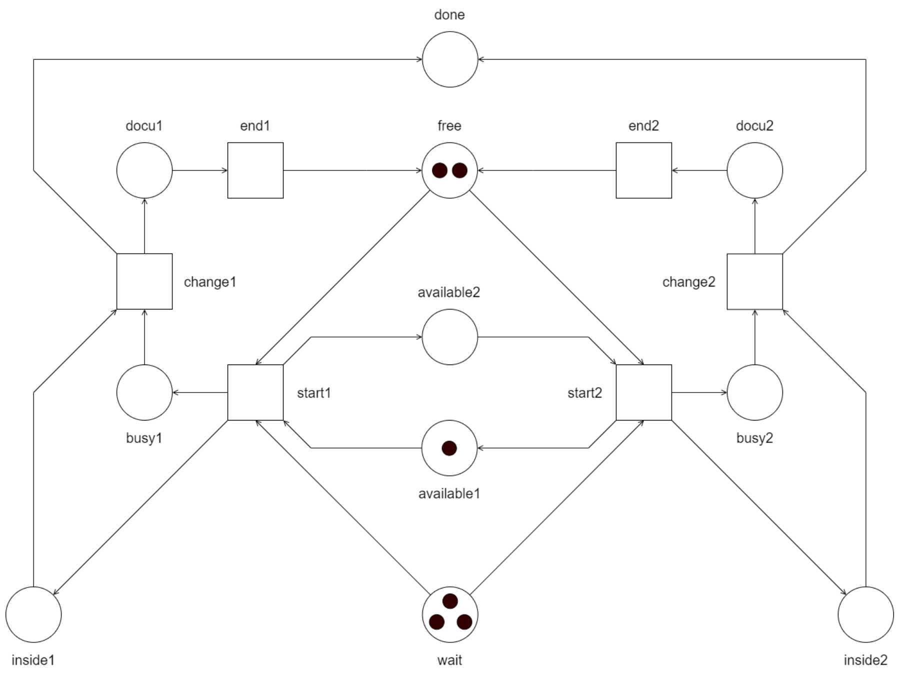


Figure 21: The grand Petri net for 2 specialist and patient

**7. Write a computational package to realize (implement) Items 1, 2, 3 and 4 above.**

**You could employ any programming language (mathematical-statistical like R, Matlab, Maple, Singular, or multi-purposes language like C or Python), that you are familiar with, to write computational programs. The programs of your package should allow input with max 10 patients in place *wait*.**

<u>**Solution**</u>:

Before writing a program simulating Petri nets, we model Petri net as following concept:

• The program is written with C++ programming language.

• To create places or transitions, we use an adjacency list constructed of vector package of C++. The list has 2 subvectors of string datatype, the first one contains names of places, the other one contains names of transitions. We name this list **component**, the first subvector is $\textbf{component}_0$, the second subvector is $\textbf{component}_1$. For example:

```
vector<vector<string>> component = {{"free", "busy", "docu"}, {"start", "change",
    "end"}}
```

• To specify maximum token number that a place can contain, we use a vector of integer as following:

```
vector<int> maxToken = {a, b, c}
```

Where, $a$, $b$ and $c$ are given natural numbers and, furthermore, $a$ is the maximum token that place *free* can contain, $b$ is the maximum token of place *busy* and the left one belongs to place *docu*. In summary, we assign value $i$-th of vector maxToken to place $i$-th of $\text{component}_0$, $i$ is index of either vector.

• To model relations of places and transitions, we construct a matrix with 5 rules:

1. Rows indicate transitions, row $i$-th represents transition $i$-th of $\text{component}_1$. Columns indicate places, column $i$-th represents place $i$-th of $\text{component}-0$.

2. If there's an arc between place $j$-th and transition $i$-th and direction is from $j$-th to $i$, the value at $(i, j)$ of the matrix is 1.

3. If there's an arc between place $j$-th and transition $i$-th and direction is from $i$-th to $j$-th, the value at $(i, j)$ of the matrix is $-1$.

4. If there's a bidirectional arc between place $j$-th and transition $i$-th, the value at $(i, j)$ of the matrix is 2.

5. If there's no arc between place $j$-th and transition $i$-th, the value at $(i, j)$ of the matrix is 0.

For example:

```
vector<vector<int>> relation = {{1, −1, 0}, {0, 1, −1}, {−1, 0, 1}}
```

This matrix stands for the Petri net of the specialist discussed above. The relation can be explicitly shown as following:

| | *free* | *busy* | *docu* |
|---|---|---|---|
| *start* | 1 | −1 | 0 |
| *change* | 0 | 1 | −1 |
| *end* | −1 | 0 | 1 |

• To create initial marking for the net, we construct a vector of integer as following:

```
vector<int> init_marking = {1, 0, 0}
```

Value $i$-th of init_marking is assigned to places $i$-th of component$_0$, $i$ is index of either vector.

• To create a Petri net, we call constructor:

```
PetriNet net = PetriNet(component, relation, maxToken, init_marking)
```

• To print out places the net contains, we call function printPlaces(). Similarly, we call printTrans() to print transitions.

• We call function run() to simulate the net. This function will execute until no transition is enabled.

• We call function run(int maxFiring) to simulate the net in case it contains a cycle. Parameter maxFiring is total firing that the net can make. This function is created to avoid an infinite loop.

• After calling function run() or run(int maxFiring), we can call function printMarkingLog() to print out markings of the running process. We also call function reset() to set the net as beginning.

• To find out reachable markings, we call function findReachable(). This function captures all state changes of the net. In case the net contains a cycle, a state change can be made by 2 different firing sequences, function findReachable() just keep the version that first found.

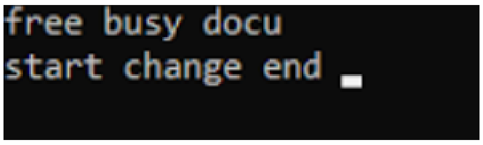
**a) Write down states and transitions of the Petri net $N_S$ and represent it as a transition system.**

**(i) Write down states and transitions of the Petri net $N_S$**

In function main():

```
int max = INT_MAX;
vector<vector<string>> component = {{"free", "busy", "docu"},
{"start", "change", "end"}};
vector<int> maxToken = {max, max, max};
vector<vector<int>> relation = {{1, −1, 0}, {0, 1, −1}, {−1, 0, 1}};
vector<int> init_marking = {1, 0, 0};
PetriNet net = PetriNet(component, relation, maxToken, init_marking);
net.printPlaces();
cout << endl;
net.printTrans();
```

Result is shown in console window:



```
free busy docu
start change end
```

Figure 22: Places and transitions of specialist Petri net

**(ii) Represent Ns as a transition system**

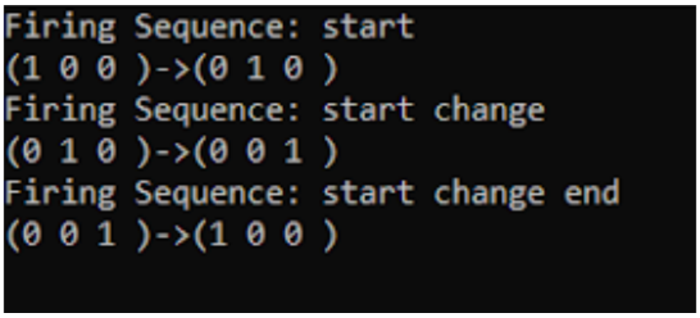
◇ **Case 1: Each place cannot contain more than one token in any marking**

• **Assume that the initial marking is** [1, 0, 0]:

In function main():

```
int max = 1;
vector<vector<string>> component = {{"free", "busy", "docu"},
{"start", "change", "end"}};
vector<int> maxToken = {max, max, max};
vector<vector<int>> relation = {{1, −1, 0}, {0, 1, −1}, {−1, 0, 1}};
vector<int> init_marking = {1, 0, 0};
PetriNet net = PetriNet(component, relation, maxToken, init_marking);
net.findReachable();
```

Result is shown in console window:



```
Firing Sequence: start
(1 0 0 )->(0 1 0 )
Firing Sequence: start change
(0 1 0 )->(0 0 1 )
Firing Sequence: start change end
(0 0 1 )->(1 0 0 )
```

Figure 23: Reachable markings from initial marking [1, 0, 0]

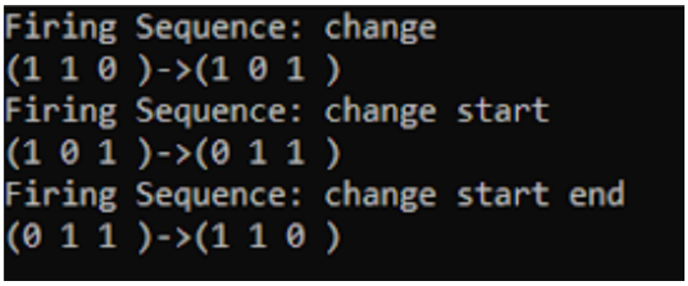
Sentences "Firing sequence: ..." represent a firing sequence from the initial marking. Each sentence below "Firing sequence: ... shows the last firing of the current firing sequence of the net.

• **Assume that the initial marking is** [1, 1, 0]:

In function main(), we modify:

```
vector<int> init_marking = {1, 1, 0};
```

Result is shown in console window:



```
Firing Sequence: change
(1 1 0 )->(1 0 1 )
Firing Sequence: change start
(1 0 1 )->(0 1 1 )
Firing Sequence: change start end
(0 1 1 )->(1 1 0 )
```

Figure 24: Reachable markings from initial marking [1, 1, 0]

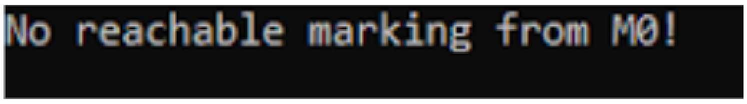
Similar to initial marking [1, 1, 0], the initial marking [1, 0, 1] or [0, 1, 1] shows the same reachable markings.

• **Assume that the initial marking is** [1, 1, 1]:

In function main(), we modify:

```
vector<int> init_marking = {1, 1, 1};
```

Result is shown in console window:



```
No reachable marking from M0!
```

Figure 25: Reachable markings from initial marking [1, 1, 1]

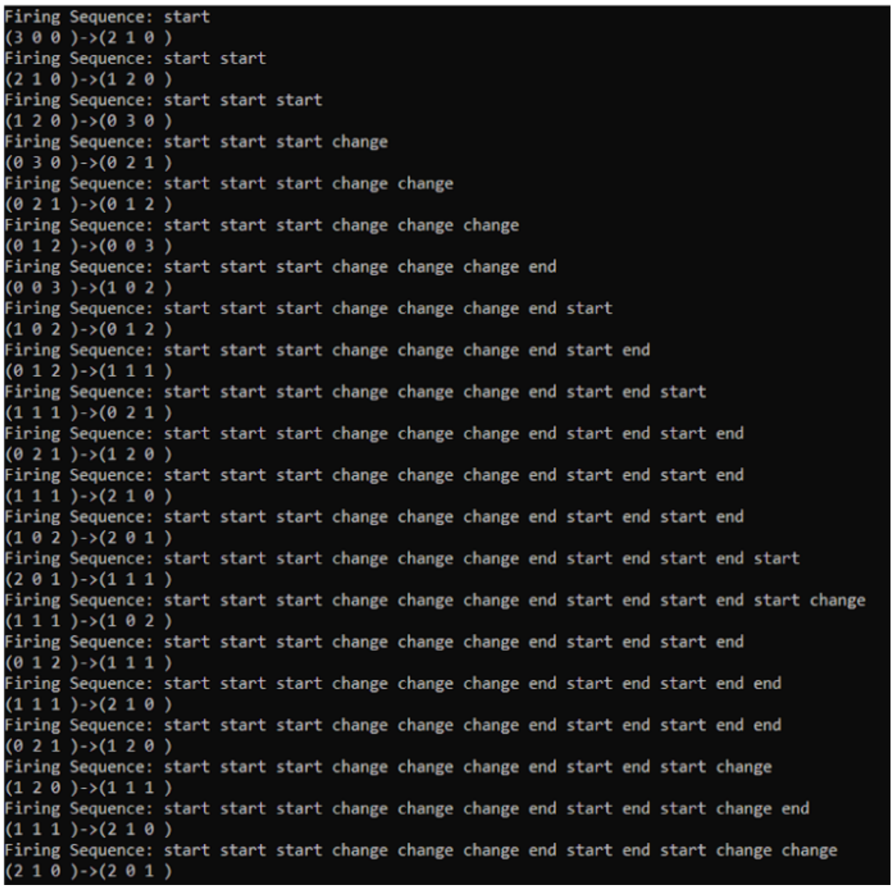
From these reachable marking, we can conclude set $TR$ of transition system is equal to $\{((1,0,0),(0,1,0));((0,1,0),(0,0,1));((0,0,1),(1,0,0));((1,1,0),(1,0,1));((1,0,1),(0,1,1));((0,1,1),(1,1,0))\}$.

◇ **Case 2: Each place may contain any natural number of tokens in any marking**.

In function main(), we modify:

```
int max = INT_MAX;
vector<int> init_marking = {3, 0, 0};
```

Result is shown in console window:



```
Firing Sequence: start
(3 0 0 )->(2 1 0 )
Firing Sequence: start start
(2 1 0 )->(1 2 0 )
Firing Sequence: start start start
(1 2 0 )->(0 3 0 )
Firing Sequence: start start start change
(0 3 0 )->(0 2 1 )
Firing Sequence: start start start change change
(0 2 1 )->(0 1 2 )
Firing Sequence: start start start change change change
(0 1 2 )->(0 0 3 )
Firing Sequence: start start start change change change end
(0 0 3 )->(1 0 2 )
Firing Sequence: start start start change change change end start
(1 0 2 )->(0 1 2 )
Firing Sequence: start start start change change change end start end
(0 1 2 )->(1 1 1 )
Firing Sequence: start start start change change change end start end start
(1 1 1 )->(0 2 1 )
Firing Sequence: start start start change change change end start end start end
(0 2 1 )->(1 2 0 )
Firing Sequence: start start start change change change end start end start end
(1 1 1 )->(2 1 0 )
Firing Sequence: start start start change change change end start end start end
(1 0 2 )->(2 0 1 )
Firing Sequence: start start start change change change end start end start end start
(2 0 1 )->(1 1 1 )
Firing Sequence: start start start change change change end start end start end start change
(1 1 1 )->(1 0 2 )
Firing Sequence: start start start change change change end start end start end
(0 1 2 )->(1 1 1 )
Firing Sequence: start start start change change change end start end start end end
(1 1 1 )->(2 1 0 )
Firing Sequence: start start start change change change end start end start end end
(0 2 1 )->(1 2 0 )
Firing Sequence: start start start change change change end start end start change
(1 2 0 )->(1 1 1 )
Firing Sequence: start start start change change change end start end start change end
(1 1 1 )->(2 1 0 )
Firing Sequence: start start start change change change end start end start change change
(2 1 0 )->(2 0 1 )
```

Figure 26: Reachable markings from initial marking [3, 0, 0] with unlimited numbers of token at each place

As the Figure 26 shows above, we get all possible reachable markings of this unlimited token number Petri net. Therefore, we can express set $TR$ of transition system as discussed above:

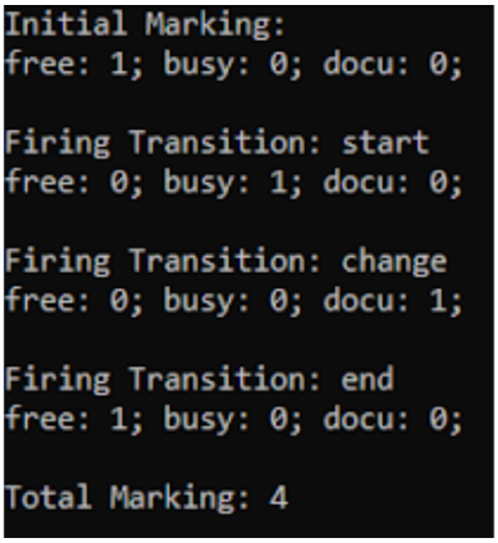
$$\begin{aligned} TR = &\{((x+1,y,z),(x,y+1,z))|x,y,z \in \mathbb{N}\} \\ &\cup \{((x,y+1,z),(x,y,z+1))|x,y,z \in \mathbb{N}\} \\ &\cup \{((x,y,z+1),(x+1,y,z))|x,y,z \in \mathbb{N}\} \end{aligned}$$

**(iii) Simulate the state of specialists with Petri net $N_S$**

In function main():

```
int max = 1;
vector<vector<string>> component = {{"free", "busy", "docu"},
{"start", "change", "end"}};
vector<int> maxToken = {max, max, max};
vector<vector<int>> relation = {{1, −1, 0}, {0, 1, −1}, {−1, 0, 1}};
vector<int> init_marking = {1, 0, 0};
PetriNet net = PetriNet(component, relation, maxToken, init_marking);
net.run(3);
net.printMarkingLog();
cout << endl << "Total Marking: " << net.MarkingLogCount() << endl;
```

Result is shown in console window:



```
Initial Marking:
free: 1; busy: 0; docu: 0;

Firing Transition: start
free: 0; busy: 1; docu: 0;

Firing Transition: change
free: 0; busy: 0; docu: 1;

Firing Transition: end
free: 1; busy: 0; docu: 0;

Total Marking: 4
```

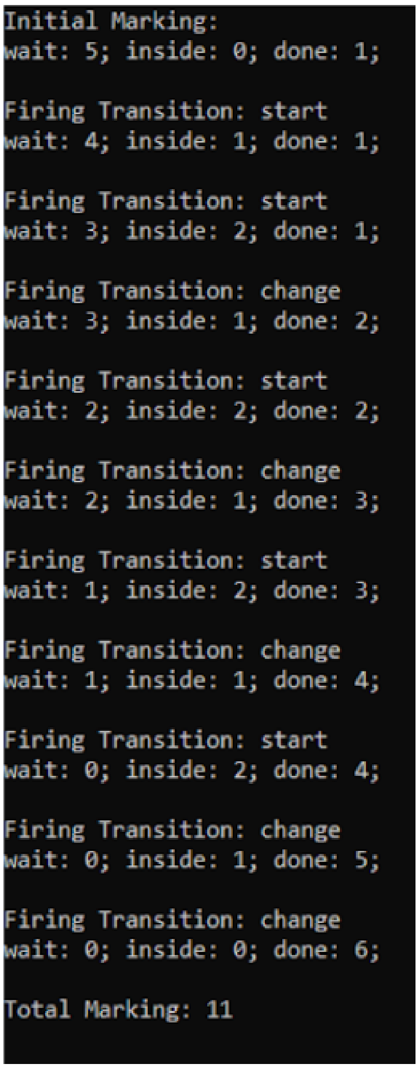
Figure 27: 4 markings, including initial marking, are created by firing 3 times

**b) Simulate the state of patients with Petri net $N_{Pa}$**

In function main():

```
int max = INT_MAX;
vector<vector<string>> component = {{"wait", "inside", "done"}, {"start", "change"}};
vector<int> maxToken = {max, max, max};
vector<vector<int>> relation = {{1, −1, 0}, {0, 1, −1}};
vector<int> init_marking = {5, 0, 1};
PetriNet net = PetriNet(component, relation, maxToken, init_marking);
net.run();
net.printMarkingLog();
cout << endl << "Total Marking: " << net.MarkingLogCount() << endl;
```

Result is shown in console window:



```
Initial Marking:
wait: 5; inside: 0; done: 1;

Firing Transition: start
wait: 4; inside: 1; done: 1;

Firing Transition: start
wait: 3; inside: 2; done: 1;

Firing Transition: change
wait: 3; inside: 1; done: 2;

Firing Transition: start
wait: 2; inside: 2; done: 2;

Firing Transition: change
wait: 2; inside: 1; done: 3;

Firing Transition: start
wait: 1; inside: 2; done: 3;

Firing Transition: change
wait: 1; inside: 1; done: 4;

Firing Transition: start
wait: 0; inside: 2; done: 4;

Firing Transition: change
wait: 0; inside: 1; done: 5;

Firing Transition: change
wait: 0; inside: 0; done: 6;

Total Marking: 11
```

Figure 28: All markings of Petri net represent patients' state

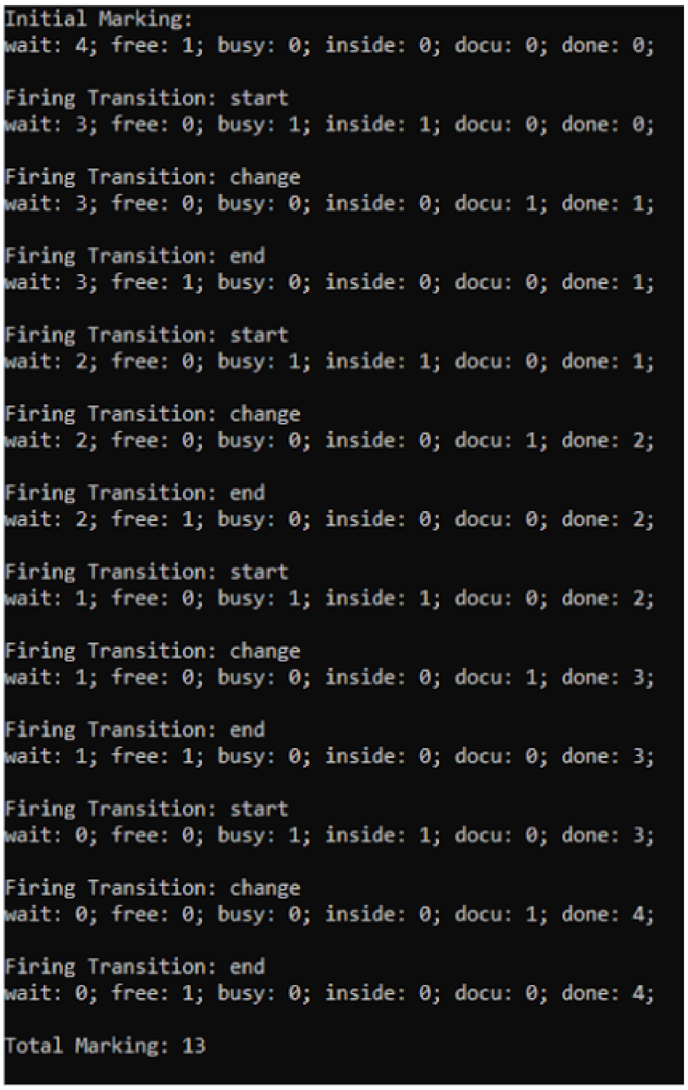
**c) Simulate the grand net** $N = N_S \oplus N_{Pa}$

In function main():

```
int max = INT_MAX;
vector<vector<string>> component = {{"wait", "free", "busy", "inside", "docu", "done"},
{"start", "change", "end"}};
vector<vector<int>> relation = {{1, 1, -1, -1, 0, 0}, {0, 0, 1, 1, -1, -1}, {0, -1, 0, 0,
    1, 0}};
```

```
vector<int> maxToken = {max, max, max, max, max, max };
vector<int> init_marking = {4, 1, 0, 0, 0,  0};
PetriNet net = PetriNet(component, relation, maxToken, init_marking);
net.run();
net.printMarkingLog();
cout << endl << "Total Marking: " << net.MarkingLogCount() << endl;
```

Result is shown in console window:



```
Initial Marking:
wait: 4; free: 1; busy: 0; inside: 0; docu: 0; done: 0;

Firing Transition: start
wait: 3; free: 0; busy: 1; inside: 1; docu: 0; done: 0;

Firing Transition: change
wait: 3; free: 0; busy: 0; inside: 0; docu: 1; done: 1;

Firing Transition: end
wait: 3; free: 1; busy: 0; inside: 0; docu: 0; done: 1;

Firing Transition: start
wait: 2; free: 0; busy: 1; inside: 1; docu: 0; done: 1;

Firing Transition: change
wait: 2; free: 0; busy: 0; inside: 0; docu: 1; done: 2;

Firing Transition: end
wait: 2; free: 1; busy: 0; inside: 0; docu: 0; done: 2;

Firing Transition: start
wait: 1; free: 0; busy: 1; inside: 1; docu: 0; done: 2;

Firing Transition: change
wait: 1; free: 0; busy: 0; inside: 0; docu: 1; done: 3;

Firing Transition: end
wait: 1; free: 1; busy: 0; inside: 0; docu: 0; done: 3;

Firing Transition: start
wait: 0; free: 0; busy: 1; inside: 1; docu: 0; done: 3;

Firing Transition: change
wait: 0; free: 0; busy: 0; inside: 0; docu: 1; done: 4;

Firing Transition: end
wait: 0; free: 1; busy: 0; inside: 0; docu: 0; done: 4;

Total Marking: 13
```

Figure 29: All possible markings of the grand net

## 4 Conclusion

In this report, the group has accomplished the following tasks:

- Learn about Petri Networks theory to solve the questions that the lecturers have produced.
- Apply programming language to demo the Petri Networks in a few simple cases.

The questions have been resolved, but there are still some mistakes. We look forward to the comments of the lecturers so that we can better understand the content of this assignment.

## References

[1] Man VM Nguyen. *Data Analytics- Foundation: Inference, Regression and Stochastic Processes*. Saarbrücken Germany: LAP LAMBERT Academic Publishing (2020) ISBN 978-620-2-79791-7.

[2] Kurt Jensen, Lars M.Kristensen. *Coloured Petri Nets - Modelling and Validation of Concurrent Systems*. Springer (2009).