MODELING OF CURRICULUM USING PETRI NETS

A THESIS REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTER OF SCIENCE (M.Sc.)
IN
APPLIED MATHEMATICS

Submitted by

LUBHAVIKA PARASHAR (2K21/MSCMAT/29)

KISHOR KUMAR (2K21/MSCMAT/58)

Under the supervision of

MS. PAYAL

DEPARTMENT OF APPLIED MATHEMATICS
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi 110042

MAY, 2023

DEPARTMENT OF APPLIED MATHEMATICS
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

We, LUBHAVIKA PARASHAR and KISHOR KUMAR, Roll No.'s –2K21/MSCMAT/29, 2K21/MSCMAT/58 students of M.Sc.(Applied Mathematics), hereby declare that the project dissertation titled MODELING OF CURRICULUM USING PETRI NETS, which is submitted by us to the Department of Applied Mathematics, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of degree of Master of Science in Mathematics, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Place: Delhi

Date: May 27, 2023

Lubhavika Parashar

Kishor Kumar

DEPARTMENT OF APPLIED MATHEMATICS
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the Project Dissertation titled MODELING OF CURRICULUM USING PETRI NETS which is submitted by Lubhavika Parashar and Kishor Kumar, Roll No.'s – 2K21/MSCMAT/29 and 2K21/MSCMAT/58 of Department of Applied Mathematics, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Masters of Science in Mathematics, is a record of the project work carried out by the students under my supervision. To the best of my knowledge, this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi — Ms. Payal

Date: May 27, 2023 — **SUPERVISOR**

DEPARTMENT OF APPLIED MATHEMATICS
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

We wish to express our sincerest gratitude to Ms. Payal for her continuous guidance and mentorship that she provided us during the project. She showed us the path to achieve our targets by explaining all the tasks to be done and explaining to us the importance of this project as well as its industrial relevance. She was always ready to help us and clear our doubts regarding any hurdles in this project. Without her constant support and motivation, this project would not have been successful. Also, this paper shows the core structure of the syllabus of an M.Sc. Mathematics Curriculum and the concurrent learning management flow of the program in Delhi Technological University, a big thanks to the Mathematics Department, DTU; their professors and students for providing their valuable time and help in our research paper.

Place: Delhi

Date: May 27, 2023

Lubhavika Parashar

Kishor Kumar

Abstract

The notion of Petri Net, formerly developed by Carl Adam Petri, is useful for modeling and analyzing a system's behavior. Petri Net is a graphical tool, defined as a bipartite graph consisting of two types of nodes, places (conditions) and transitions (events).

Petri net modeling can be a vital tool to help make decisions at various levels and types of organizations. In this paper, we endorse a Petri net model of curriculum management to support the decision-making process among university administrators, students, and lecturers and outline the various steps involved in verifying a curriculum, its behavioral properties, modeling, and analysis using Petri nets. Management can also use this model to verify the inner consistency of an existing and upcoming development syllabus, determine the student's career progress, and determine the probability of a student dropping out.

Contents

List of Tables

List of Figures

Chapter 1

PETRI NETS

1.1 Petri Net Structure

A Petri net is a conceptual, formal representation of the flow of information. Petri net's features, concepts, and methodologies are being developed in order to describe and analyze the movement of information and control in asynchronous and concurrent systems. Petri nets have primarily been used to describe systems of events where some events may occur concurrently, but there are restrictions on these events, concurrency, precedence, or frequency. Because many readers are likely to be unfamiliar with Petri nets, a quick and informal explanation of its foundations and origins is provided. A closer work at a number of Petri net's characteristics is provided. One can start by thinking about how Petri nets can be used to simulate a system of concurrent or parallel operations.

A Petri Net structure is a four-tuple structure. It is represented as $PN = (P, T, I, O)$, where P is a collection of all places; T is a collection of all transitions; I is the matrix that explains the association of input places and the transitions; and O is the matrix that explains the association of output places and the transitions. For a PN consisting of, say, m - places and n - transitions; where $P = \{p_1, p_2, \ldots, p_m\}$ and $T = \{t_1, t_2, \ldots, t_n\}$; matrices I and O can have the values a_{ij} which can take the value either 0 or 1 such that: $a_{ij} = 1, p_i$ is an input (output) place for transition t_j; $a_{ij} = 0, p_i$ is not an input (output) place for transition t_j and also $P \cap T = \emptyset$ i.e. the collection of places and transitions are disjoint.[1]

Sometimes, instead of matrices, functions are used, where the input function can be defined as $I : T \to P^\infty$ and the output function is defined as $O : T \to P^\infty$ where T represents the set of transitions and P^∞ denotes the bags[1]of the places. The core ideas and notations of Petri nets are defined and discussed in more formal terms in the subsections.

Consider a four-tuple Petri net structure in Figure 1.1 and its components are as :

$$\begin{array}{ll} P = \{p_1, p_2, p_3, p_4, p_5\} & \\ T = \{t_1, t_2, t_3, t_4\} & \\ I(t_1) = \{p_1\} & O(t_1) = \{p_2, p_3, p_5\} \\ I(t_2) = \{p_2, p_3, p_5\} & O(t_2) = \{p_5\} \\ I(t_3) = \{p_3\} & O(t_3) = \{p_4\} \\ I(t_4) = \{p_4\} & O(t_4) = \{p_2, p_3\} \end{array} \tag{1.1}$$

The graph features two kinds of nodes: bars and circles. These nodes are called *transitions* and *places* respectively. They are connected by arcs from one place to another. If an arc is from node i to node j, then i is an input to node j, while j is an output to node i. In Figure 1.1, the

[1]A bag is a generalization of sets that allows for numerous occurrences of an element in a bag.

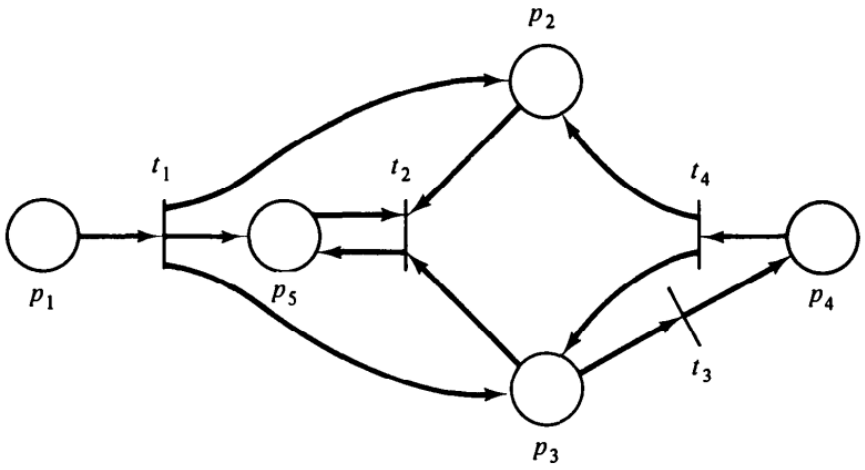

Figure 1.1: A Petri net graph equivalent to the Petri net structure.

place p_2 and p_3 represent outputs of transition t_1, and also p_2, p_3 and p_5 represent inputs of transition t_2.

Dual of a Petri Net:

Since, both the vertex sets V_1, V_2 can be either of the two, places or the transitions, thus the dual of the Petri Net can be accordingly defined, with a resulting interchanged sets of places and transitions. Thus for a given Petri Net, $PN = (P,T,I,O)$, the *dual* of the Petri Net PN is given by $\overline{PN} = (T,P,I,O)$. For example, the dual of the petri net in Figure 1.1 is given by Figure 1.2.

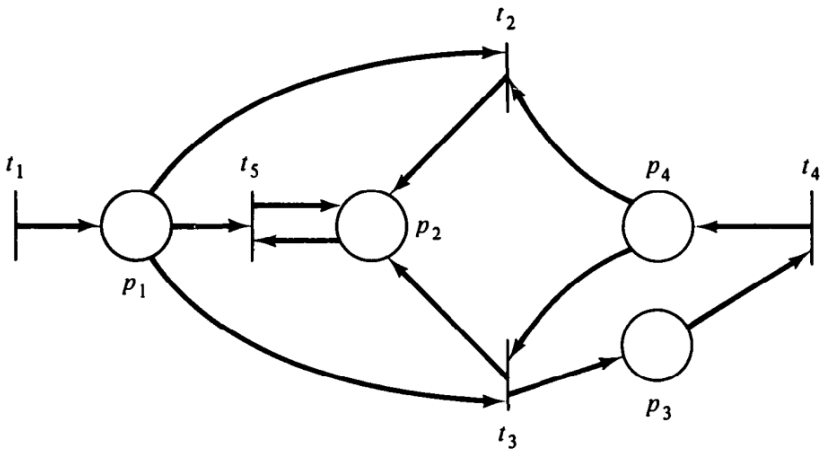

Figure 1.2: The dual of a Petri net in Figure 1.1.

A Petri net contains dynamic properties that emerge from its execution in addition to the static properties represented by the graph. Assume also that the implementation of a coding algorithm is represented by a flowchart, and that the execution is demonstrated by marking the instruction, i.e. being carried out on the flowchart, with a marker, and that the marker travels around the flowchart as the execution proceeds. Similarly, the placement and movement of markers (also known as tokens) within a Petri net govern how it operates. *Tokens* are represented by black dots in places on the net and any PN with the token is referred to as *Marked Petri net.*

The way the tokens are used has certain board game-like characteristics. The guidelines are as follows: The firing of the net's transitions moves the tokens. In order to fire, a transition needs to be enabled(a transition is live when it has a token in each of its input locations). To enable the transition, the enabling tokens must be withdrawn from their input locations and new tokens must be deposited at their output locations.

1.2 Petri Net Marking

A *marking* of a Petri Net PN, at a certain given state t, is the assignment of the tokens to the set of places. It is denoted by $M_t = \{M_1, M_2, \ldots, M_m\}$ where M_i gives the number of tokens that are available at the place p_i at a certain state t. A marking M is a function defined from P, the collection of all places to the positive integers i.e., $M : P \to Z^+$, where clearly, $M(p_i) = M_i$. The initial marking at initial state (at t = 0) is represented by $M_0 : P \to Z^+$. A marked PN w.r.t M_0 is a 5-tuple structure where $PN = (P, T, I, O, M_0)$.

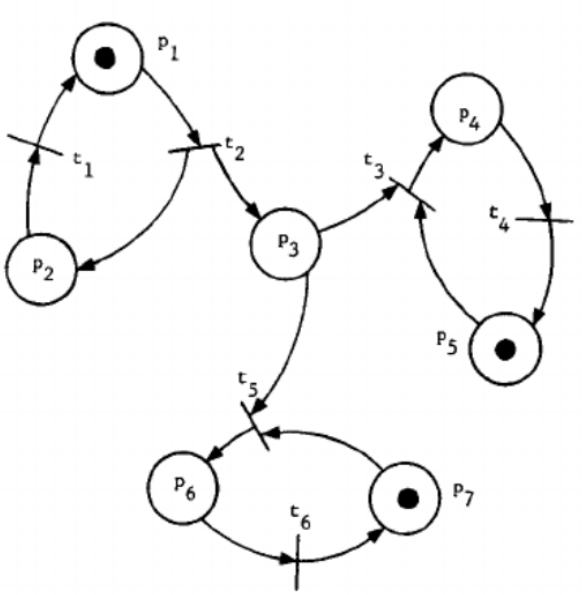

Figure 1.3: A marked Petri Net.

Figure 1.3 is an example of a Petri net graph with a marking. It represents the structure described in the previous section with the marking $M = (1,0,0,0,1,0,1)$. Since the number of tokens in a place is unbounded over the set of all markings, there is an infinite number of markings for a Petri net. The set of all markings for a petri net with m places is simply the set of all m-vectors, N^m. This set, although infinite, is of course denumerable.

1.3 Transition Enabling and Firing

Any transition, say t_j in a system, is *enabled* and can fire with one or multiple input places; if the number of tokens in all the input places is at least equal to the multiplicity of all the input arcs for t_j of those places respectively, i.e., a transition t in a marked Petri net having marking M gets enabled to fire, if for all $p_i \in P(i = 1 \text{ to } m)$,

$$M(p_i) \geq \#\left(p_i, I\left(t_j\right)\right).$$

We also call this the *triggering* of the transition t_j. When a transition t_j of a system occurs or triggers, a token gets removed from all the input places and eventually gets added to the respective output places. One must note here that it is not necessary for the number of input places to be equal to the number of output places w.r.t the triggered transition.

The illustration of the change in marking in a place when a transition fires is shown in Figure 1.4.

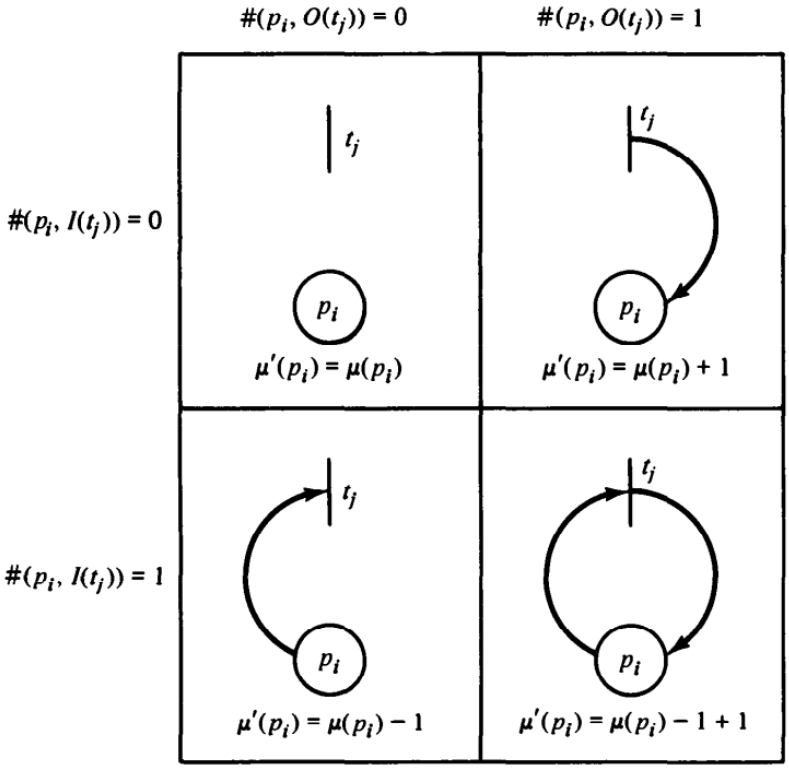

Figure 1.4: Illustration of the change in marking in a place when a transition fires.

Consider a marked PN in Figure 1.5 which shall help us in illustrating the firing rules, where the transitions, t_1, t_3, t_4 are enabled.

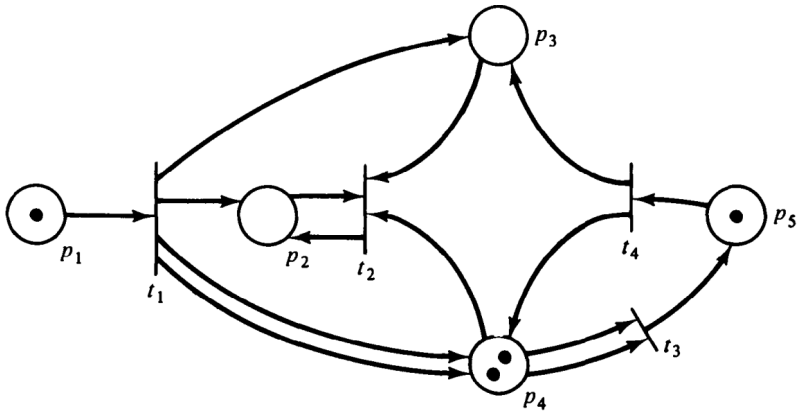

Figure 1.5: A marked PN with marking (1, 0, 0, 2, 1) and transitions t_1, t_3, t_4 are enabled.

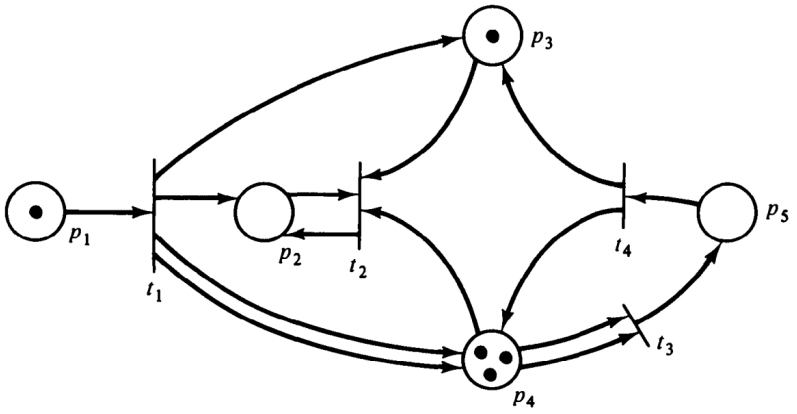

Figure 1.6: Transition t_4 fires, then resulting marking is (1, 0, 1, 3, 0).

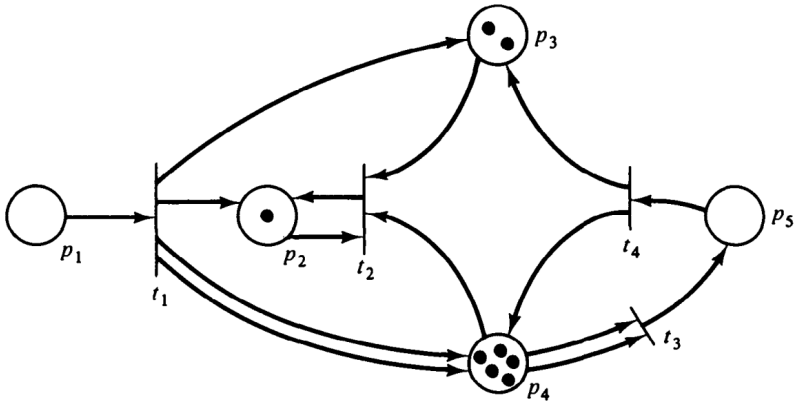

Figure 1.7: Transition t_1 fires, then resulting marking is (0, 1, 2, 5, 0).

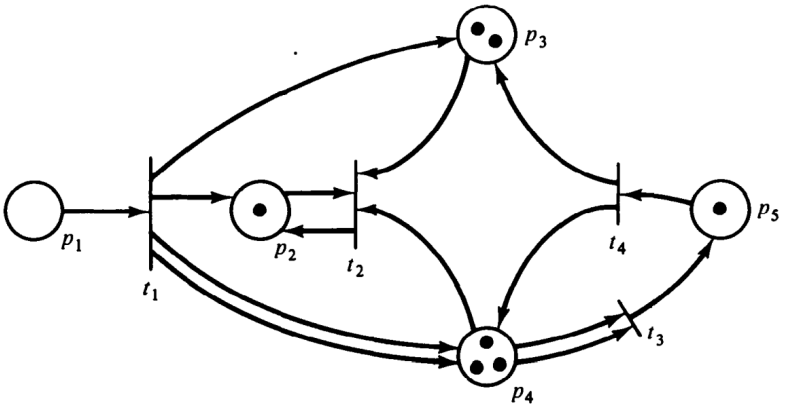

Figure 1.8: Transition t_3 fires, then resulting marking is (0, 1, 2, 3, 1).

If a transition t is enabled then $w(p,t)$ represents the ***weight*** of the arc which shows the availability of tokens to be removed from input place to output place. It is possible for an enabled transition to fire (depending on whether the event occurs).

The well-known chemical reaction $H_2 + O_2 \rightarrow 2H_2O$ is used to demonstrate the aforementioned transition rule in Figure 1.9. Two tokens in each input site in Figure 1.9 (a) indicate the availability of two units of $2H$ and $2O$ as well as the readiness of the transition t. Now, after firing the transition t, a new marking is shown in Figure 1.9 (b).

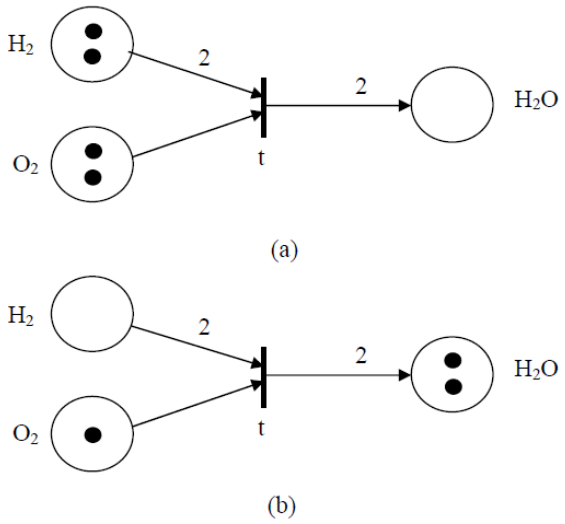

Figure 1.9: (a) Before firing of t, (b) After firing of t.

A transition is said to be a ***source*** and ***sink*** transition; if it has no input place and output place respectively. Unlike a sink transition, which produces tokens, a source transition is unconditionally enabled.

1.4 Petri Net State Space

We define the state of a Petri Net by the corresponding markings at that time. The firing of a transition in a Petri Net represents an alteration in the state of the PN by changing the marking.[1] For a marking $M : P \to Z^+$ of a PN where $M(p_i) = M_i$ and $P = \{p_1, p_2, \cdots, p_m\}$, a Petri Net with m-places has a *state space* which is the set of all markings which shall be equal to N^m.

The change in the state that occurs by firing an enabled transition is defined using a change function ϕ, which is referred to as *next-state function*. [6]
Formally, the next-state function, $\phi : N^m \times T \to N^m$ for a Petri Net $PN = (P,T,I,O)$ with the marking M and a transition $t_j \in T$ is defined if and only if

$$M(p_i) \geq \#\left(p_i, I\left(t_j\right)\right), \forall p_i \in P$$

If $\phi\left(M,t_j\right)$ is defined, then $\phi\left(M,t_j\right) = M'$, where

$$M'(p_i) = M(p_i) - \#\left(p_i, I\left(t_j\right)\right) + \#\left(p_i, O\left(t_j\right)\right), \forall p_i \in P$$

For a given petri net $PN = (P,T,I,O)$ and an initial marking M_0, the PN can be executed by successive transition firings. The two sequences which result from the PN execution are-
1. Sequence of markings: $(M_0, M_1, \cdots)$
2. Sequence of transitions: $\left(t_{j_0}, t_{j_1}, \cdots\right)$
These two above mentioned sequences are related as:

$$\phi\left(M_k, t_{j_k}\right) = M_{k+1}, \quad k = 0,1,2,\cdots$$

The result of the firing of an enabled transition, say t_j is the change in the state from M to M' and we say that M' is *immediately reachable* from M i.e the transition of the state takes place from M to M'.

If a marking M'' is instantly reached from another marking M' i.e immediately reachable from M_0, then it is known as *reachable marking* from M_0. The collection of all markings that can be reached from $M = (P,T,O,I,M_0)$; is defined as the *reachability set* $R(M)$ which represents the "immediately reachable" relationship's reflexive transitive closure, i.e., the *reachability set* $R(PN,M)$ is the smallest set of markings defined as:
1. $M \in R(PN,M)$
2. If $M' \in R(PN,M)$ and $M'' \in \phi\left(M',t_j\right)$ for some $t_j \in T$, then $M'' \in R(PN,M)$, where M is the marking of a given Petri net.

Thus the collection of all states that a marked Petri net can reach through any execution is known as the reachability set. The reachability set of PN attributes is hence a topic covered in many analytical questions.

Chapter 2

MODELING OF PETRI NETS

2.1 Introduction

In many scientific fields, a phenomena is researched by analysing a model rather than the reality itself. A model is a representation, often expressed mathematically, of the characteristics of an item that is being studied that are thought to be the most significant. It is believed that through tinkering with the representation, new insights into the phenomena being represented and the model itself would be attained without the expense, discomfort, or risk of tinkering with the actual reality. Due to the cost and risk involved in handling radioactive materials, for instance, most work on atomic energy has been done via simulation.

Math is a major component in modelling. Many physical phenomena key characteristics may be mathematically represented, and equations or inequalities can be used to explain how these characteristics relate to one another. Mathematical equations may be used to describe a variety of phenomena, particularly in physics and chemistry, including mass, momentum, acceleration, location, and forces. It is necessary to understand both the modelled phenomena and the modelling procedures in order to effectively use the modelling approach. As it can be used to simulate phenomena in other fields, mathematics has become a science in part as a result. For instance, the differential calculus was created as a direct result of the requirement for a method to represent physics continuously changing attributes like location, velocity, and acceleration.

Another modelling tool is the Petri net. They were developed for use in the modelling of a certain category of issues, the category of discrete-event systems with concurrent or simultaneous occurrences. Petri nets serve as models for systems, especially for two characteristics of systems: *events* and *conditions*, as well as the *relationships* between them. According to this perspective, a system will exhibit a set of circumstances at any given moment. The occurrence of particular events may be caused by the existence of particular circumstances. These occurrences may change the system's state, resulting in some prior conditions stopping to hold and other conditions starting to hold.

Consider, the simultaneous holding of the conditions "A card reader is required and the condition "A card reader is available" might result in the event "Allocate the card reader." is a straightforward example. The conditions "No card reader is available" is true as a result of this occurrence, which causes the criteria "A card reader is required" and "A card reader is available" to cease to exist. Figure 2.1 illustrates these conditions, events, and their relationships by using transitions to represent events and locations to represent conditions. Please be aware that even if some circumstances are not depicted, such as "The card reader is assigned," they may still exist in the system.

This approach may be used to simulate more complex systems as well. Consider, for in-

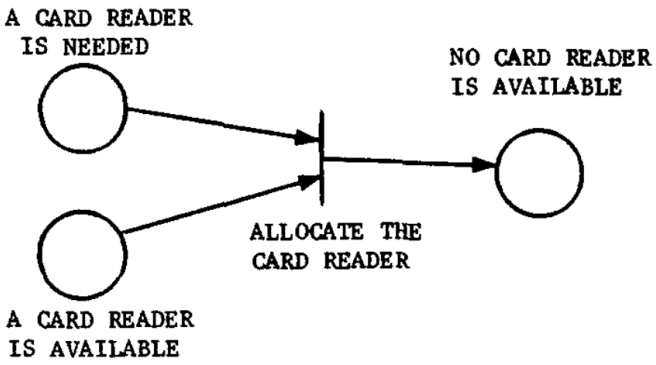

Figure 2.1: A Petri Net model of three conditions and an event.

stance, the computer system that is described as follows:.

- Jobs occur and are added to a list of inputs. If there is a task on the input list and the processor is available, it begins processing the work.
- An output list is created once a job is completed. The processor goes on to the next job if there are still tasks to be completed; otherwise, it waits for another task to be finished.

The processor, input list, output list, and tasks are some of the components that make up this relatively simple system. Many circumstances of interest may be found, including:

- Processor inactive;
- The input list includes a job.;
- Work is being done on a job.;
- The output list includes a task and a number of events.:
- The system gets a new job.;
- Job processing started.;
- Processing of jobs has been completed;
- The system lost a job..

The modeling of this system is shown by the Petri net in Figure 2.2. In this example, the ”job arrives” transition is a source, whereas the ”job departs” transition is a sink.

2.2 Properties of Petri Nets Useful in Modeling

Many aspects of Petri nets and the systems they may mimic are shown in the aforementioned example. Inherent *concurrency* or *parallelism* is one. In the system, the task and the processor are the two primary categories of independent entities. It is unnecessary to synchronise the processor’s and jobs’ activities in the Petri net model for events that are purely related to one or the other. As a result, regardless of the processor’s activity, jobs may enter or exit the system at any moment. It is also easy to depict synchronization circumstances, such as when a job and an idle processor must both be available for processing to begin. Hence, a Petri net would appear

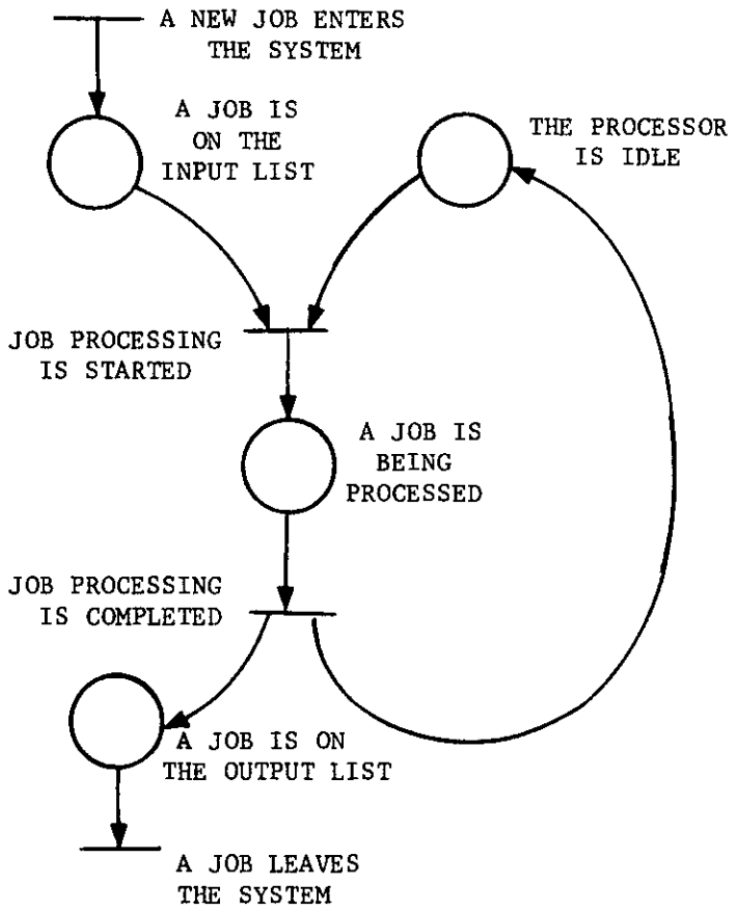

Figure 2.2: Petri Net model of a simple computer system.

to be the best model for distributed control systems that include numerous processes running at once.

Petri net's *asynchronous nature* is one of its main characteristics. A Petri net doesn't have a built-in way to track time or the passage of time. This is consistent with a view of time that holds, logically speaking, identifying a partial ordering of the occurrence of events is the sole significant attribute of time. Real-world events take varying amounts of time, and the Petri net model mimics this variability by not relying on a concept of time to guide the order of occurrences. In order to specify the potential sequences of events in a modelled system, the Petri net structure itself must thus contain all relevant information.

As a result, despite the fact that no information is provided or taken into account about the length of time necessary to perform a work, the event "Job processing is finished" in the net of Figure 2.2 must come after the equivalent event "Job processing has begun" due to the net's structure.The events "A new job enters the system" and "Job processing is completed" might happen before, after, or at the same time as each other while a job is being processed." Conversely, events that do not need to be restricted in terms of their relative sequence of occurrence are unrestricted.

A Petri net is considered as a series of discrete events whose order of occurrence is one of potentially many permitted by the fundamental structure, much like the system it mimics. Due to this, the Petri net's execution exhibits *nondeterminism*. Any of the many enabled transitions may trigger if there is ever a period when more than one transition is activated. Nondeterministic decision-making processes, such as chance or unmodeled forces, are used to determine which

transition fires. This Petri-net feature illustrates the fact that in situations when several things happen at once in real life, the order in which they happen is not fixed and any of a number of possible sequences may take place. The analysis of Petri nets becomes much more complicated due to nondeterminism, despite the fact that it is useful from a modelling perspective.

One restriction in the Petri net modelling of systems is generally accepted as a means of reducing this complexity. It is believed that a transition's firing (the occurrence of an event) occurs *instantaneously*, or without elapsed time. As time is a continuous variable, there is no chance for any two or more events to occur at once, hence two transitions cannot happen at once. The events that are being modelled are referred to be *primitive* events. For instance, the event "Process a job" was modelled in Figure 2.2. This event is broken down into a starting, an ending, which are both instantaneous events, and the noninstantaneous occurrence since it is not a basic event since other events, such other tasks entering and leaving the system, might happen concurrently and it takes a non-zero amount of time. In Figure 2.3, this is shown. The modelling capability of Petri nets is not diminished since this method may be used to any *nonprimitive event*.

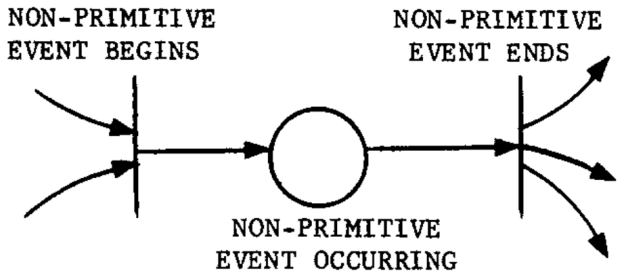

Figure 2.3: Petri Net model of nonprimitive event.

In the modelling of concurrent systems, there are two types of nondeterministic, nonsimultaneous transition firing. Figure 2.4, which depicts *simultaneous* events that may occur in either order, demonstrates one of them. In this case, the two enabled occurrences have no bearing on one another, and there are several conceivable event sequences, some of which include the occurrence of one event before the other and others which do not.

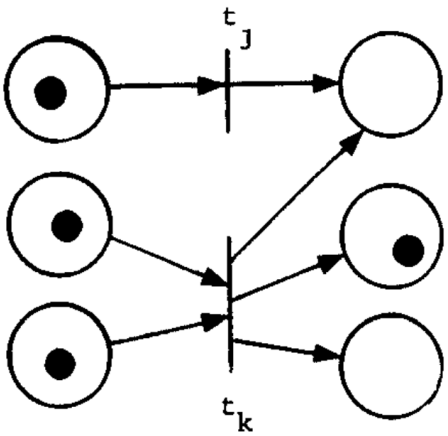

Figure 2.4: Petri Net model of simultaneous events.

Defining events to occur non-simultaneously solves the problem of simultaneous occurrences complicating modelling in the other kind of circumstance. In Figure 2.5, this is shown.

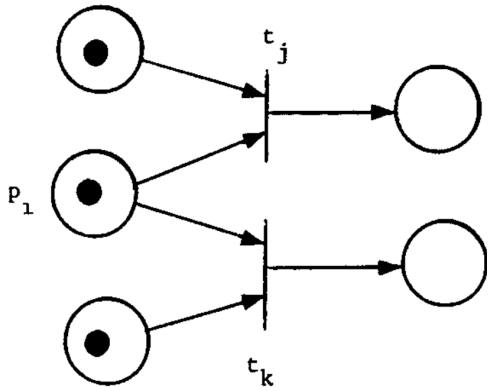

Figure 2.5: Example of conflicting transitions.

The two enabled transitions t_s and t_k are in *conflict*. The token is removed from p_2 and the other transition is disabled, so only one transition may activate. Care must be taken to ensure that, like in above, the Petri net accurately describes a system using Petri nets by reflecting all and only those event sequences that are realistically possible.

Concurrency and *conflict* are two fundamental ideas in comprehending Petri nets, and they are both shown in Figures 2.4 and 2.5. Petri nets value as information flow models also stems from the straightforward manner in which concurrency and conflict can be expressed and analyzed using them.

The fact that Petri nets are *uninterpreted models* is a key feature. The net in Figure 2.2 has been labeled with statements to make the model's goal clear to human observers, but these labels have no effect on how the net really operates. In as much as its construction is concerned, the net in Figure 2.6 is the same as that in Figure 2.2. The locations and transitions in this uninterpreted net, however, are not interpreted; instead, we only deal with the immaterial properties of the net's structure.

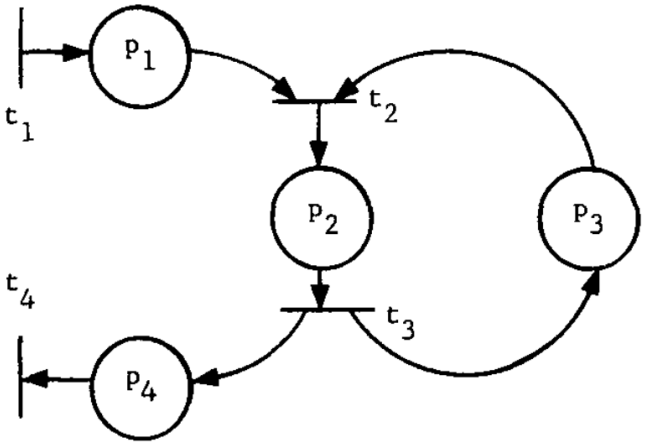

Figure 2.6: An uninterpreted Petri Net.

Petri nets may represent a system *hierarchically*, which is another useful characteristic. For more abstract modelling (abstraction), a single place or transition may be used in lieu of a full net, or subnets can be used in place of locations and transitions to offer more thorough modelling (refinement). Its hierarchical modelling feature is seen in Figure 2.7.

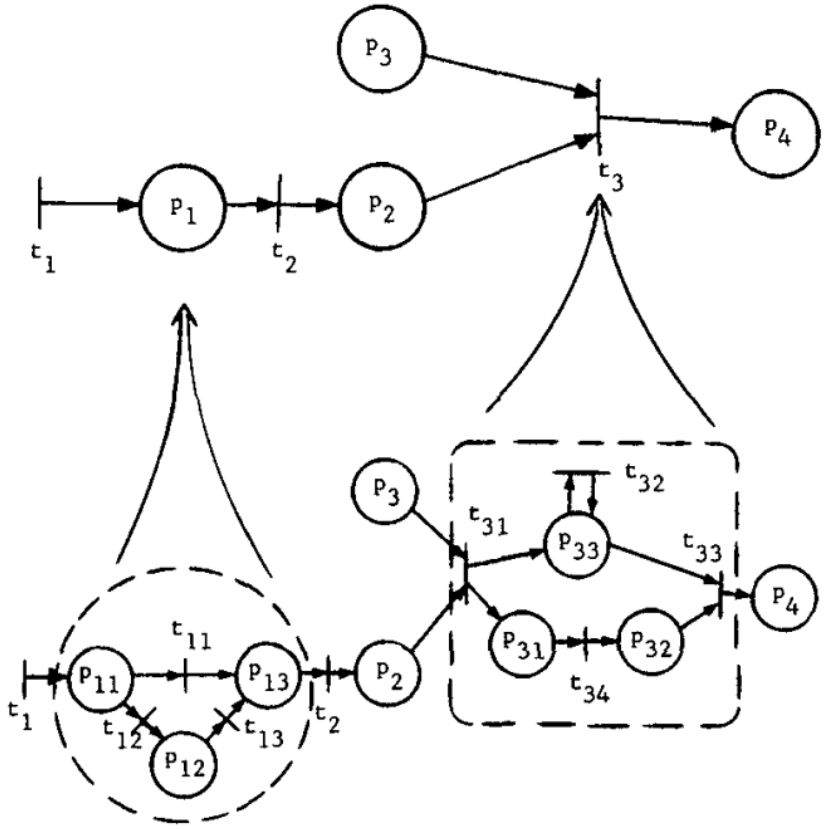

Figure 2.7: Hierarchical modeling in Petri nets by replacing places or transitions by subnets.

The majority of the research on Petri nets has focused on examining the characteristics of a particular net or class of nets. It has not received much explicit focus to build modelling methods particularly for Petri nets. Petri nets, however, seem to be the ideal tool for modelling in certain situations, namely those in which events take place separately and asynchronously.

2.3 Modeling of Hardware

In an attempt to attain maximal parallelism and hence boost effective processing speed, large, powerful computer systems often employ asynchronous parallel activities. For example, numerous functional units are available in computers like the CDC 6600 and the IBM 360/91 to conduct calculations on different registers. The machine's control unit makes an effort to maintain many of these units running concurrently.

To ensure that the outcomes of running the programme with and without parallelism are the same, the introduction of *parallelism* must be regulated in this way. Prior to the execution of certain programme activities, the results of earlier operations must have been correctly calculated. *Determinate* systems provide parallelism to sequential programmes while maintaining accurate outputs. Bernstein has thought about the requirements for retaining determinancy. They are as follows: If b does not need the output of a as an input and the output of b does not change the inputs or outputs of a, then given two operations a and b, where a comes before b in the program's linear precedence, b may start before a is done.

Utilising a reservation table is one way to apply these restrictions to the design of the computer control unit, which is responsible for issuing commands. An instruction for functional unit u utilizing registers i,j, and k may only be given if none of the four of these components

are reserved; these four elements all become reserved if the directive is given. The control unit waits until the command can be issued if it cannot be given at this moment before moving on to the next instruction.

A Petri net may be used to represent this kind of architecture. We assign a place to each functional unit and each register. A token will be present if the unit or register is free; if not, there won't be one. Figure 2.8 depicts a part of a Petri net that might be used to simulate how an instruction would be executed using unit u and registers i, j, and k. Of fact, a far bigger Petri net would be needed to model the full control unit.

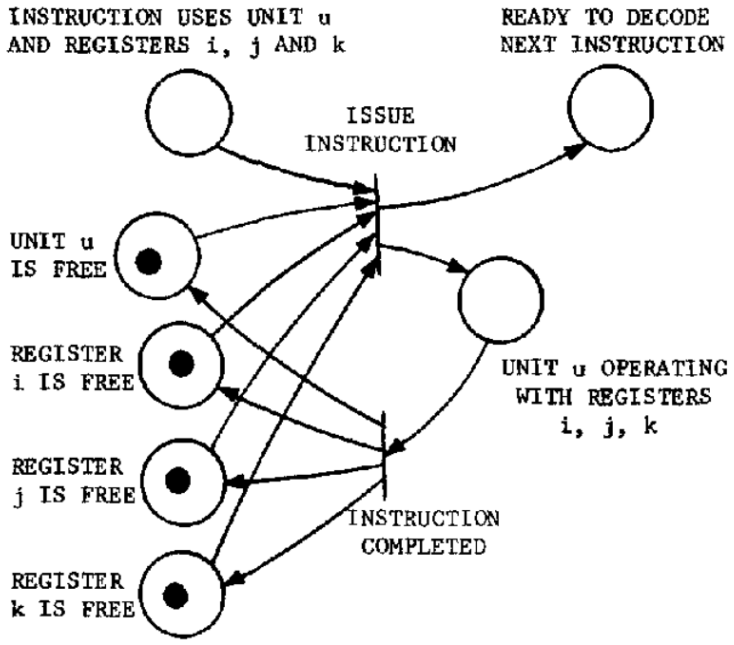

Figure 2.8: Control unit with multiple registers and functional units.

The above-mentioned strategy is a fairly basic way to add parallelism; it does not, for instance, take into account the possibility of many functional units using the same register as an input at once. So, this plan could not result in the highest level of parallelism in the schedules. Other plans, however, are capable of doing this. Petri nets may also be used to simulate these (more complex) methods. One Petri net model, for instance, simulates the CPU of a CDC 6600. The optimal level of parallelism between the various functional units was determined using this model to decide how object code should be created.

Pipelines are yet another method for building high-performance computers. This method works well and is comparable to an assembly line's workflow, especially when processing vector and array data. A lot of steps make up the pipeline, and they could all be running at once. Stage k completes its job, sends the findings along to Stage $(k+1)$, and then searches for fresh work in Stage $(k-1)$. A single operand's whole operation requires nt time units if there are n stages and each takes t time units. The pipe may, however, produce results at a rate of one per t time units if it is constantly fed with fresh operands.

Think about adding two floating-point values as an example. The fundamental steps in activity are:

* Extrapolate the two numbers' exponents;

* If required, compare the exponents and swap them to put the larger and smaller exponents in the correct order.;

* Move the smaller fraction to make the exponent equal;

* Faction addition;

* Post-normalize;

* Pack the exponent and the result's fraction while taking exponent overflow or underflow into account.

A distinct computing unit may carry out each of these stages, passing a specific operand from unit to unit throughout the whole addition process.

Multiple strategies may be used to manage the coordination of the various components. The time permitted for each step of the pipeline is often synchronised with the pipeline control, for constant time t.The output of one unit is transferred down the pipe for every t time unit to serve as the input for the following unit. Due to the fact that processing times might vary from step to stage and depending on the input, this can, nevertheless, needlessly slow down the process.The time required for the post-normalization step in the preceding floating-point addition will depend on the size of the normalization shift, for example, whether it should be to the left or to the right. A pipeline that sends results from step k to stage $(k+1)$ as soon as step k is done and stage $(k+1)$ is free may speed up processing. A Petri net is a simple tool for modelling this method.

Think about a random pipeline step. At this point, operations demand specific inputs and provide specific outputs. Of course, the position of the inputs and outputs is necessary. Typically, this includes registers: utilizing information from its input register, the device creates values for its output register. Then, it must wait till

1. Its input register may accommodate a new input, and

2. since its output register was duplicated into the input register of the next stage, it has been empty.

Thus, the pipeline's control system must be aware of the following situations:

1. input register full

2. input register empty;

3. full output register;

4. output register vacant;

5. A busy unit;

6. unit unoccupied;

7. copying taking place

The Petri net used to simulate how an asynchronous pipeline of this kind operates is shown in Figure 2.9. Petri nets may also be used to define several kinds of pipeline control units.

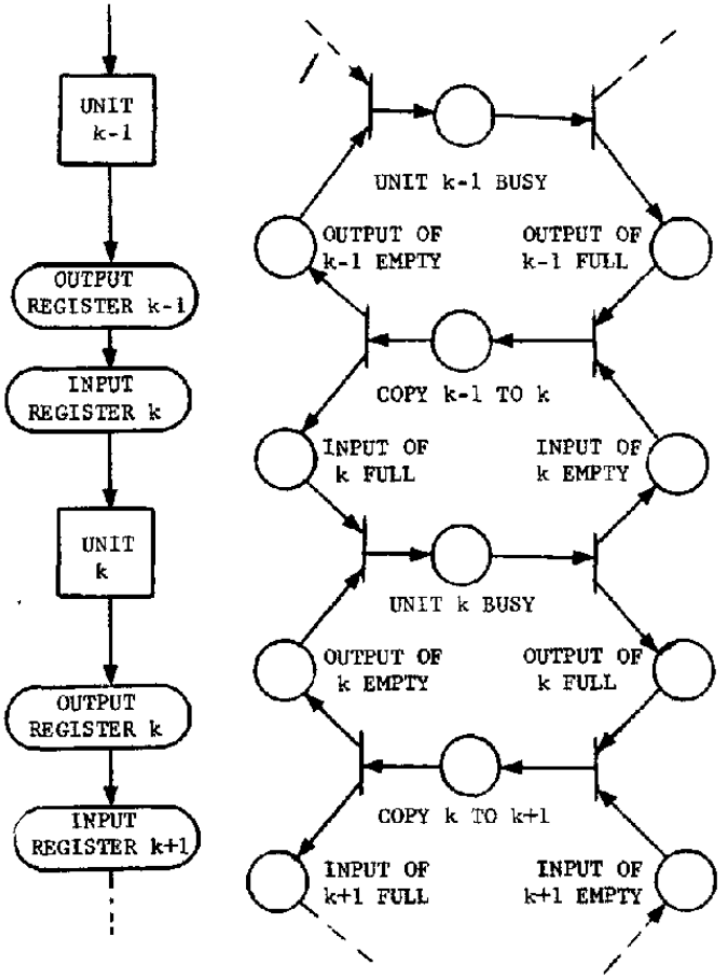

Figure 2.9: Asynchronous pipelined control unit.

2.4 Modeling of Software

Petri nets may also represent software ideas at a more abstract level. An operating system's resource distribution, deadloack, and process coordination may all be modelled. A Petri net may describe a *process* in the same manner as a flowchart does, and the interactions between processes can then be represented by extra places, arcs, and transitions.

Think about the *mutual exclusion problem*, as an example. This is a problem with making sure that two essential parts of code, one in each process, are mutually excluded in time. In other words, Process 2 may not begin its crucial part until Process 1 has finished its own critical section if Process 1 is performing its critical component. The scenario is represented by Figure 2.10 in its most abstract level:

The challenge is defining suitable entrance and exit codes to ensure mutual exclusion. By employing the *P and V operations* as specified in for process synchronization and coordination, the mutual exclusion problem may be readily handled. Only P and V operations may be run, on a semaphore as only P and V operations work on semaphores[1]. These operations can be defined as follows:

$$P(S): \text{ as soon as } S > 0, \text{ set } S := S - 1;$$

$$V(S): S := S + 1.$$

[1] A semaphore S is a variable with integer values.

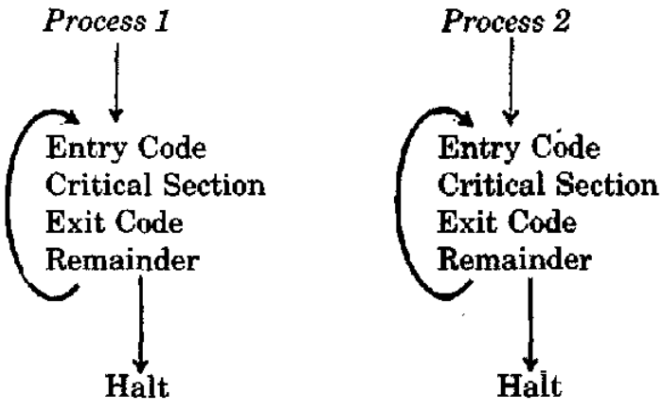

Figure 2.10: Illustration of situation at highest level of abstraction.

These operations cannot be divided. The semaphore must be positive before a process doing a *P* action may decrease it and go on. Simply adding one to the semaphore is what a *V* action does, maybe enabling another process to carry out a *P* activity. Two processes cannot execute *P* or *V* actions on the same semaphore at the same time.
For instance,

Process 1	Process 2
P(mutex);	P(mutex)
”Critical Section”;	”Critical Section”;
V(mutex);	V(mutex);

is a solution of the mutual exclusion problem diagrammed above using P and V operations, which are primitive, and the semaphore ”mutex” which is global to the two processes and has an initial value of one.

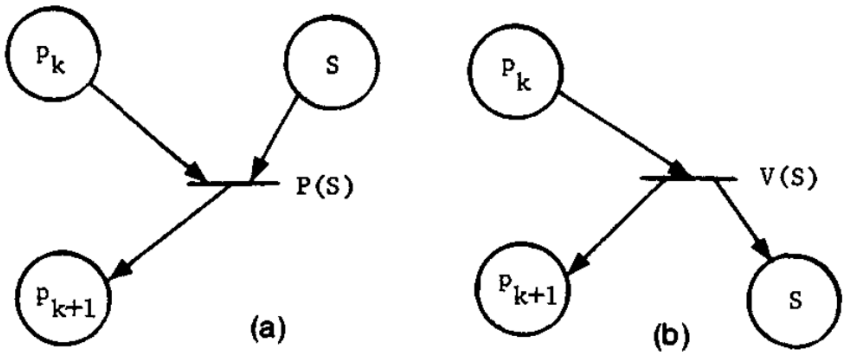

Figure 2.11: Modeling with Semaphores (a) Modeling of a P operation; (b) Modeling of a V operation.

Many people have utilised *P* and *V* operations. Petri nets can simulate systems of processes that use these procedures. A place simulates a semaphore, with the value of the semaphore being modelled by the quantity of tokens in the place. A token is added to the semaphore by a *V* action, and it is removed using a *P* operation (after, if required, waiting for a token to be added). Figure 2.11 shows how to do this.

As shown in Figure 2.12, where the spot *S* represents the semaphore, the mutual exclusion issue may then be modelled. The positions p_2 and p_4 are mutually exclusive, so take note of

that. Petri nets limits have been found and proven via the modelling of *P* and *V* operations.

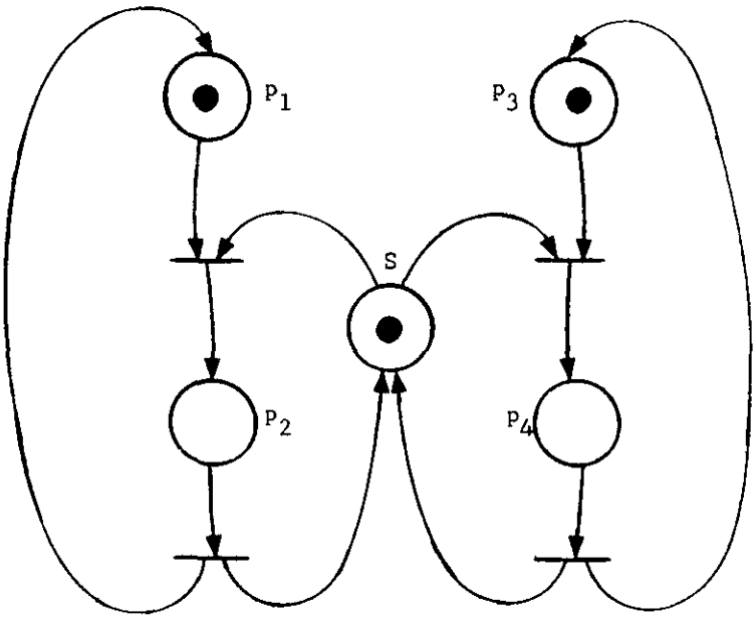

Figure 2.12: Mutual exclusion problem in P/V solution.

Chapter 3

ANALYSIS OF PETRI NETS

3.1 Safeness

Definition: A place $p_i \in P$ of a Petri Net $PN = (P,T,I,O)$ with an initial marking M_0 is *safe* if for all $M' \in R(PN,M_0), M'(p_i) \leq 1$. A Petri Net is said to be *safe* if all the places in that Petri Net are safe.[2]

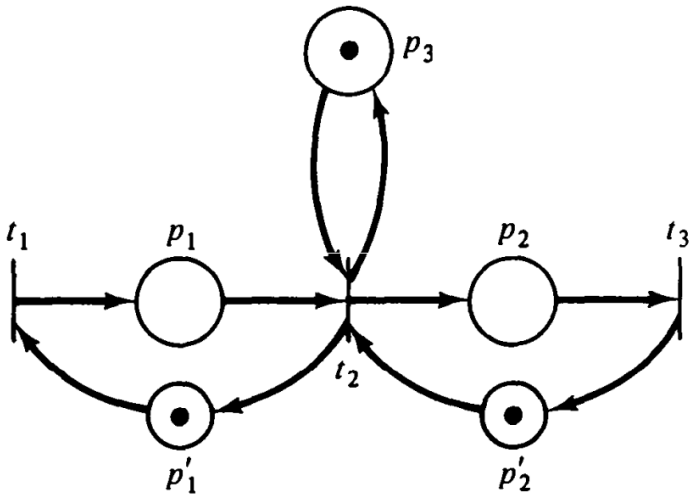

Figure 3.1: Example of a Safe Petri net.

Remark: When modeling a Petri Net as a real hardware device, the safeness property of a Petri Net can be useful for its analysis.

In Figure 3.1, when the transition t_1 is fired it removes a token from the place p'_1 and adds a token in the place p_1, this enables the transition t_2 and when it is fired a token is added to the place p'_1 and a token is deleted from the place p_1. Thus firing of any transition results in one token in either p_1 or p'_1 at a time. Moreover, firing of t_2 removes and adds a token in p_3 and hence there is only one token in p_3. Also a token is added in p_2 and removed from p'_2. Hence a similar dynamics takes place in p_2 and p'_2 as of p_1 and p'_1. The total number of tokens at any time in a place after the firing of any transition is either 1 or 0, hence the Petri Net is Safe.

3.2 Boundedness

Definition: Any place $p_i \in P$ is said to be *n-safe* or *n-bounded*, if number of tokens in p_i cannot exceed n,

$$\text{i.e. } \forall M' \in R(PN, M_0), M'(p_i) \leq n$$

where, M_0 is initial marking and n $\in \mathbf{Z}$.
Therefore, it implies that if a place is *bounded*, then it is n-safe/n-bounded. Moreover, the entire Petri net is *bounded*, if all of its places are bounded.[2]
Remark: If the number of tokens keeps on increasing in any place, the Petri Net would become unbounded. The system which is modeled by such kind of Petri Net would become unstable, hence boundedness is a relevant property on which analysis techniques are performed.

3.3 Conservation

Strictly-Conservative Petri-Net: Any Petri net is said to be *strictly-conservative* if $\forall\ M' \in R(PN, M_0)$, we have

$$\sum_{p_i \in P} M'(p_i) = \sum_{p_i \in P} M_0(p_i)$$

where M_0 is initial marking.[1]
Consider the petri-net in the Figure 3.2,

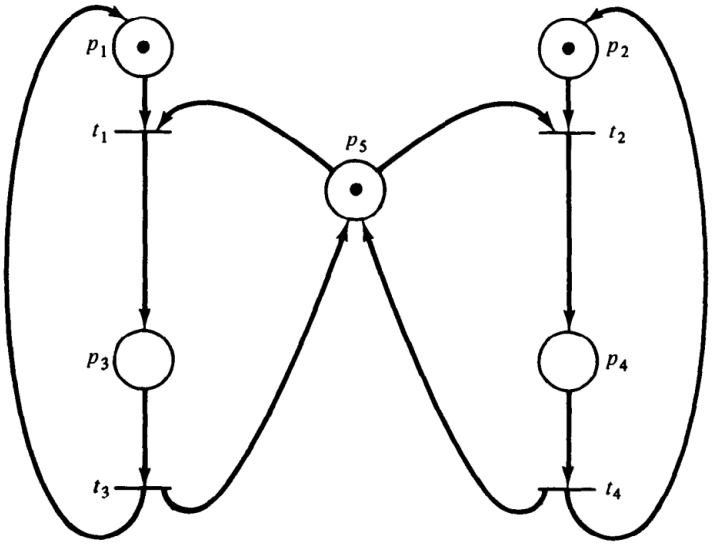

Figure 3.2: Petri net which is not strictly conservative.

Here the enabled transitions are t_1 and t_2.

$$\begin{aligned}
M_0 &= (1,1,0,0,1) \quad \text{(initial marking)} \\
M_1 &= (0,1,1,0,0) \quad (\text{ On firing } t_1) \\
M_2 &= (1,1,0,0,1) \quad (\text{ On firing } t_3) \\
M_3 &= (1,0,0,1,0) \quad (\text{ On firing } t_2) \\
M_4 &= (1,1,0,0,1) \quad (\text{ On firing } t_4)
\end{aligned}$$

The Petri Net in Figure 3.2 is not strictly conservative since the number of tokens in each of the markings is either increased or decreased by one, on the consecutive firing of transitions, and hence the token count is not constant.

Remark: When Petri Net is modeled in such a way that the tokens are represented as resources that are neither created nor destroyed, conservation becomes an important property to monitor.

3.3.1 Conservative with respect to weighing vector

Definition: A Petri Net $PN = (P, T, I, O)$ with an initial marking M_0 is conservative with respect to weighing vector u, where $u = (u_1, u_2, u_3, u_4 \ldots\ldots, u_m)$ and $|P| = m$, $u > 0$ (positive non-zero vector), if for all $M' \in R(PN, M_0)$,

$$\sum_i u_i \cdot M'(p_i) = \sum_i \cdot u_i \cdot M_0(p_i)$$

Remarks:

- A Strictly Conservative Petri Net is conservative with respect to the weighing vector $u = (1, 1, 1, 1, \ldots, 1)$.
- The weighing vector is important concept since the tokens in the places need not be identical in nature, that is some tokens might be of larger relevance to us and thus would be assigned a larger weight, whereas some tokens might be of no importance and thus can be assigned a lesser or 0 weight. Hence in modelling of Petri Net, conservation is an important property which can be investigated with respect to the importance of tokens in the model.
- For the example of conservation in Figure 3.2 , the Petri Net is conservative with respect to the weighing vector $u = (1, 1, 2, 2, 1)$. If we consider the resultant markings after considering the weights of the token, we get

  $$(1,1,0,0,1) \cdot (1,1,2,2,1) = (1,1,0,0,1)$$
  $$(0,1,1,0,0) \cdot (1,1,2,2,1) = (0,1,2,0,0)$$
  $$(1,0,0,1,0) \cdot (1,1,2,2,1) = (1,0,0,2,0)$$

  Hence the total token number count which is 3 , is constant for all the markings after considering the weights of the tokens. Hence the model is conservative with respect to the weighing vector $u = (1, 1, 2, 2, 1)$.
- Conservation and Safeness are special cases of boundedness.

3.4 Liveness

Resource allocation was the motivation to study conservation as a property in Petri Net. Another problem which may arise in resource allocation is deadlock.[2]

Deadlock:
A deadlock occurs when a transition or a collection of transitions can't fire in a given Petri Net i.e. if $\exists$ a transition, say $t \in T$ such that t can never be fired, and therefore, t is said to be *dead*.

Consider the example of resource allocation for two processes and two resources below.
In this model illustrated in Figure 3.3 there are two processes, *process a* and *process b*, also

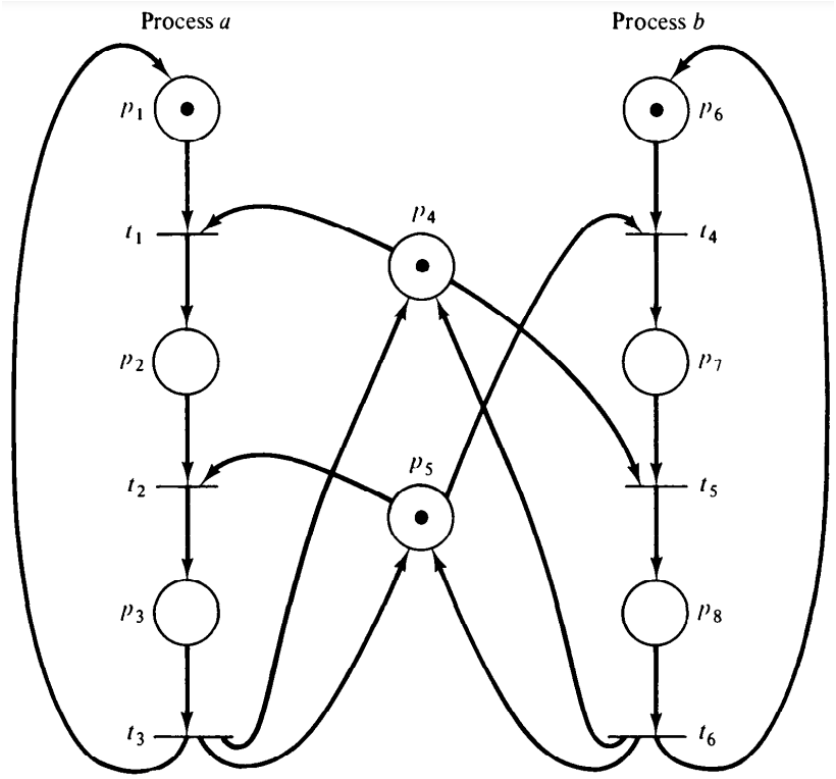

Figure 3.3: Demonstration of Deadlock.

there are two resources, r_1 in place p_4 and r_2 in place p_5. The transition firing sequence t_1,t_2,t_3,t_4,t_5,t_6 and t_4,t_3,t_6,t_1,t_2,t_3 does not produce deadlock.

If both the processes need both resources, then they would have to share the resources in such a way that each of the processes asks for a resource and then later releases it so the other process can use it.

If we consider the transition firing sequence which starts from t_1,t_4, then *process a* would have the resources from p_4 and would want resources from p_5 and similarly *process b* would have resources from p_5 and would be needing resources from p_4.
Thus a deadlock condition would be reached and neither of the two processes would be able to proceed further.

Live:
A Petri Net model is said to be *live* w.r.t an initial marking if it is possible to fire all the transitions at least once using some firing sequence for all the markings in the reachability set.
A transition is *live* if it is not deadlocked. This does not mean that the transition is enabled, but the fact that it can be enabled in future.

Potentially Firable:
Any transition t_i is said to be *potentially firable* w.r.t. a marking M_0, if $\exists$ a marking $M' \in R(PN,M_0)$ such that t_i is enabled in M'.

In the case of a non-deadlocked transition, the firing sequence indicating the execution of the transition must be live. In fact, being live ensures the absence of deadlocks and unlimited net transitions.

The levels of liveness for a Petri net with M_0 can be defined as:

- level 0 signifies a dead transition i.e. it will never fire.
- level 1 occurs when at least one transition is enabled by a marking $M \in R(PN, M_0)$.
- level 2 occur when there's firing sequence that contains t atleast n times where $n \in \mathbf{Z}$.
- level 3 occur when there's firing sequence with an indefinite length in which t can occur but also be blocked;
- level 4 occur when there is no way to stop it from firing an infinite number of times.

Thus, a transition that is live at level 0 is referred to as *dead*, while a transition that is live at level 4 is referred to as *live*.

Also, if every transition is live at i^{th} level, then Petri net is live at i^{th} level.

Consider the Figure 3.4 as an example of these levels of liveness.

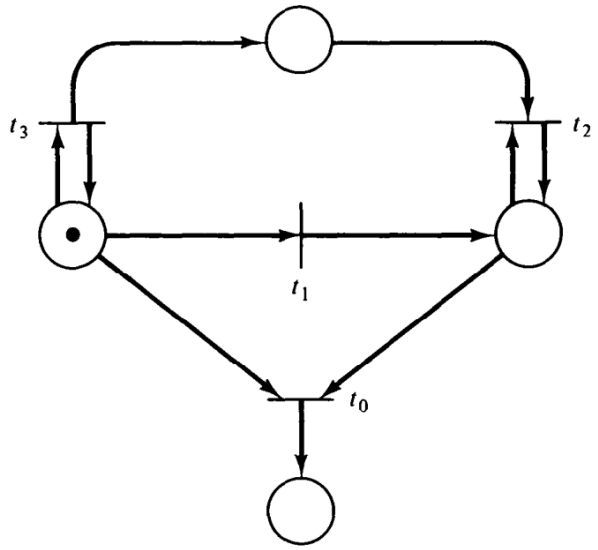

Figure 3.4: Example of Levels of liveness.

Transition t_1 can only ever fire once since it is live at level 1, but transition t_0 can never fire because it is dead. Any number of times that transition t_2 can fire depends on how often transition t_3 fires, although the exact amount is not known. To fire t_2 five times, first fire t_3 five times, then t_1 and t_2 five times. The amount of times that t_2 will fire, however, is fixed after t_1 has fired. Accordingly, t_2 is live at level 2. Contrarily, transition t_3 can fire an unlimited number of times and is therefore active at level 3, but not at level 4, as t_3 is no longer able to fire once t_1 fires.

3.5 Reachability and Coverability

Reachability Problem:

The reachablity problem considers a Marked Petri Net PN with initial marking M_0 and a marking M'. Then it aims at answering if M' is reachable from M_0, that is if $M' \in R(PN, M_0)$? This is an important property to analyze, consider the previous example in Figure 3.3. In this

example, we can see that for the marking $M' = (0,1,0,0,0,0,1,0)$ a deadlock will appear, so we would want to know whether from the initial marking is M' reachable.

Coverability Problem:
The coverablity Problem considers a Petri Net PN with an initial marking M_0 and a marking M', then is there a reachable marking that is, $M'' \in R(PN, M_0)$ such that $M'' \geq M'$?[1],[2],[3]

Remark: This property is important if we want to consider the scenario where we want to ignore the contents of the same places and would want to focus on covering the contents or items of only a few relevant places.

Chapter 4

ANALYSIS OF PETRI NETS USING REACHABLITY TREE

4.1 Introduction

The reachability tree/graph is an analytic technique for representing the reachability set of the Petri Net.

If we want to construct the reachability tree for a given Petri Net, then we need to consider the marking of that Petri Net at that time as a node and an arc would represent firing of transition from the initial marking to the subsequent new set of markings. Consider the Petri Net in the Figure 4.1 given below,

If we want to draw the reachability graph of the Petri Net, then consider the initial marking

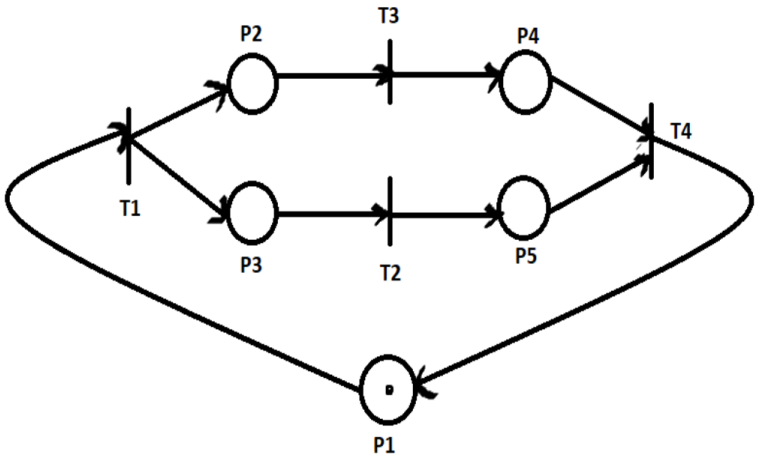

Figure 4.1: A Petri Net Model.

$M_0 = (1,0,0,0,0)$, fire the enabled transition T_1 from here, the new markings are obtained. Figure 4.2 shows how the reachability graph gets constructed following this procedure for the Petri Net model of Figure 4.1.

Many times we want to analyze the Petri Net in such a way that the direct relations in the markings are visible and the digraph has no cycles, thus we draw it as a reachability tree, where the nodes might get repeated, thus deleting the existing cycles. The reachability tree for the previous example of Figure 4.1 is shown in Figure 4.3.

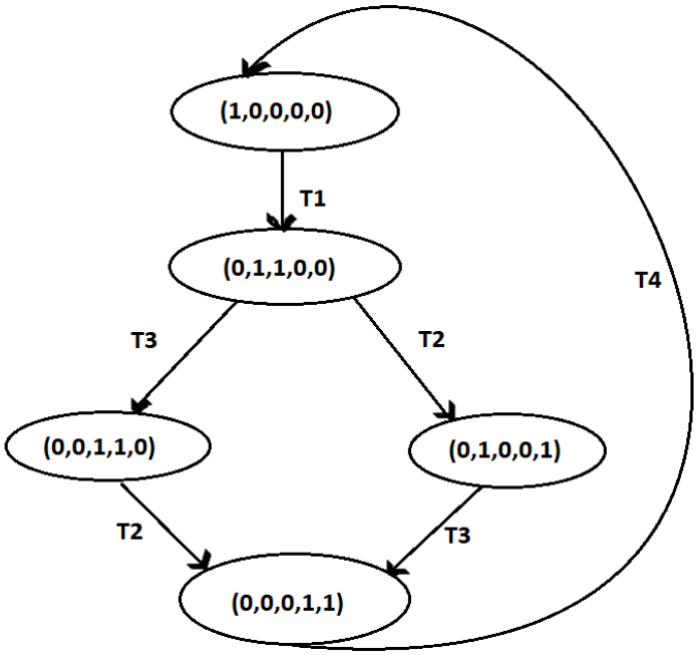

Figure 4.2: Reachability Graph of Petri Model in Figure 4.1.

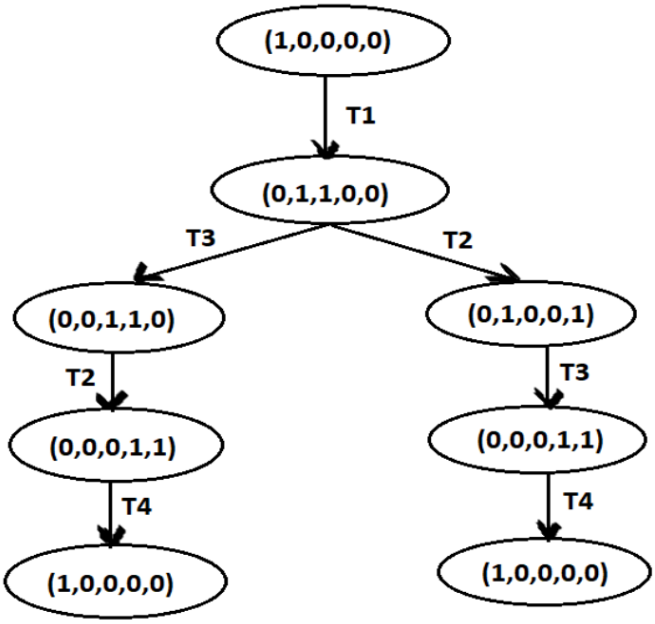

Figure 4.3: Reachability Tree with terminating nodes of Petri Model in Figure 4.1.

4.2 Condition 1

Here on wards we shall be discussing two possible conditions which are possible for constructing the reachability set and the reachability tree of a given Petri Net.

Consider a Petri Net in Figure 4.4 and its reachability tree in shown in Figure 4.5:
On repeating this process again and again in order to proceed, at every stage new marking will be produced, and we will get an Infinite Reachability set and thus Infinite Reachability tree. In order to perform analysis for the Petri Net model we need to limit the size of the tree to a finite one.

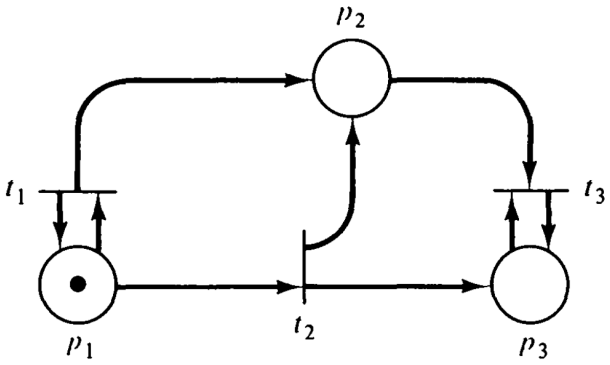

Figure 4.4: A Petri net model.

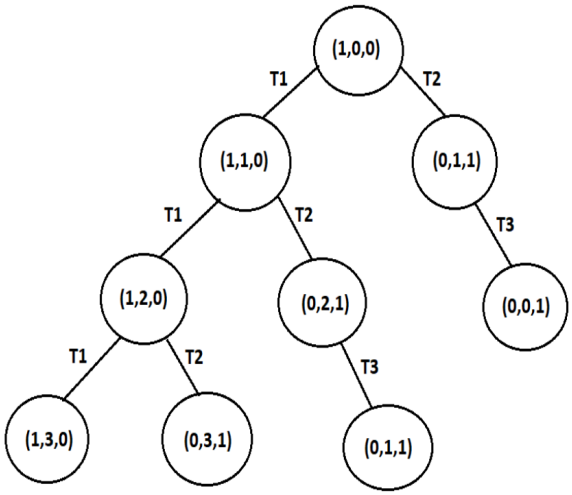

Figure 4.5: Reachability Tree of petri net of Figure 4.4.

4.3 Condition 2

Another possibility that can exist is when the finite reachability set can also produce an infinite reachability tree. Consider the following example in Figure 4.6, in this particular example the reachability set is $\{(1,0),(0,1)\}$ which is finite , but the reachability tree here is alternative in nature and is infinite.

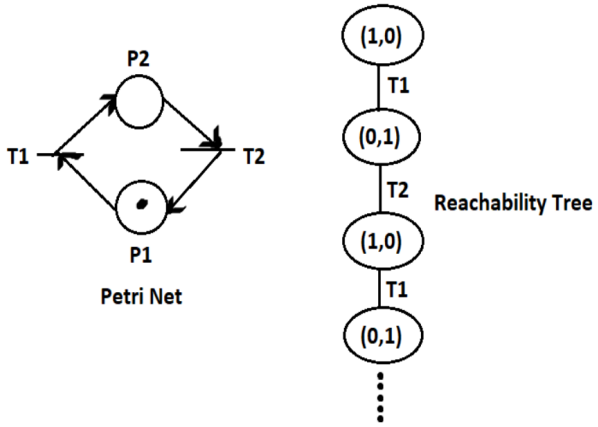

Figure 4.6: Example of a finite reachability set and infinite reachability tree.

4.4 Finite Representation of Infinite Reachability Tree

In order to represent the infinite reachability tree in a finite way, the limitation of new marking is done at each step. The nodes in the infinite tree are categorized in such a way that they limit themselves in a finite manner.

Terminal Node:-

They are the nodes that are represented by dead markings.

Dead Markings: Markings where no transitions are enabled. In this way extension of a marking is not pursued once we reach the terminal node since no transitions are enabled and thus no transition can be fired.

Duplicate Node:-

They are represented by duplicate markings.

Duplicate Marking: Set of markings that have previously appeared in the tree. The markings are not extended further once duplicate nodes are detected since successors of these markings have already been produced in the first occurrence.

4.5 ω Representation

Consider the sequence of transition firing, α which starts from M_0 and ends at the marking M' such that $M' > M_0$, i.e. $M_0 \rightarrow \alpha \rightarrow M'$. The marking M' is same as M_0 except that it has some extra tokens in some of the places, i.e. $M' = M_0 + (M' - M_0) \rightarrow$ extra tokens (> 0)

If α is fired again, this time from M', then again $M' - M_0$ tokens will be added to the marking M'.

$$\text{i.e. } M' \rightarrow \alpha \rightarrow M''$$

$$M'' = M' + \left(M' - M_0\right) = M_0 + 2\left(M' - M_0\right)$$

In general if we fire α sequence of transitions n times then we obtain the marking $\mathbf{M}_0 + \mathbf{n}(\mathbf{M}' - \mathbf{M}_0)$.

Thus for the places which have gained tokens from the sequence α, an arbitrarily large number of tokens can be accumulated just by reiterating the sequence of transitions again and again. This arbitrarily large number of tokens are represented by ω.

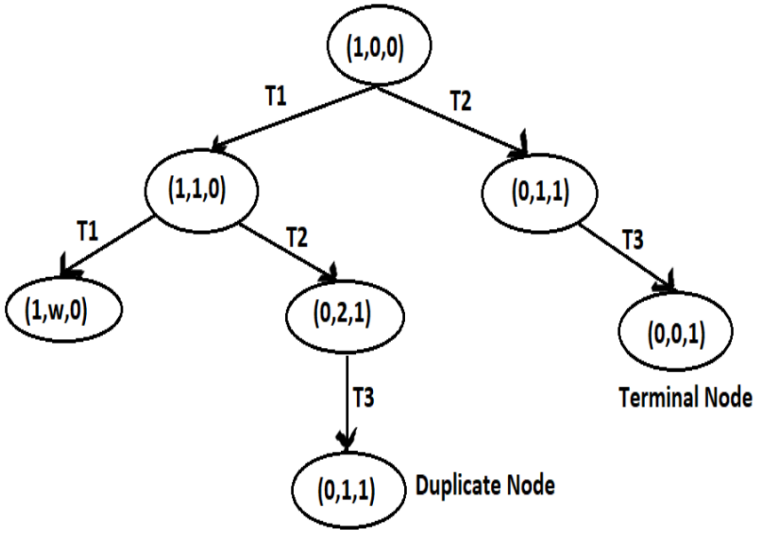

Figure 4.7: Finite Reachability Tree of petri net in Figure 4.5.

Hence for each marking, the number of tokens in a place is either a nonnegative integer or ω. Terminal nodes, duplicate nodes along with ω representation restrict the infinite tree to a finite one. In the previous example of Figure 4.5, where we were obtaining an infinite reachability tree, then after applying the finite representation techniques, the new tree that we get is shown in Figure 4.7. Note that, firing t_1 as many times, an arbitrary number of tokens can be built in P_2.

4.6 Analyzing properties of Petri Nets through Reachability Tree:

1. A Petri Net is **bounded** iff the symbol ω never appears in the reachability tree. Hence if the Petri Net is bounded, it represents the finite state system. Thus to determine the bound for a particular place, we need to draw the reachability tree and examine the tree for the largest value of the tokens in the markings corresponding to that place.

   Thus the reachability tree helps in determining the boundedness or safeness property for individual places in the Petri Net, or the entire Net.

2. Since the reachability tree is finite , **conservation** can be easily tested by computing the weighted sum of tokens in places for each marking, if the weighted sum is constant and same for all subsequent markings, then the Petri Net is strictly conservative.

3. If the symbol ω appears for any place in a marking and the corresponding element of the weighing vector for that place is 0 , then there can be a scope for the Petri Net to be conservative, but if the weight is positive of the element of the weighing vector for any place then the Net will not be conservative under any circumstances.
   Since now the symbol ω tells us that the number of tokens for some place can be arbitrarily increased, thus clearly the Petri Net won't be conservative.

4. **Coverability Problem** can also be solved through reachability tree by inspecting and scanning the reachability tree since all we wish is to determine for a given marking M', is that if the marking $M'' \geq M'$ is reachable or not.

5. A Petri Net is **deadlock-free** iff the reachability graph of the Petri Net has no node(represented by a marking) without an outgoing arc.

   Consider the example in Figure 4.8, here the reachability graph of the Petri net has no node which does not have an outgoing arc hence it is deadlock-free.

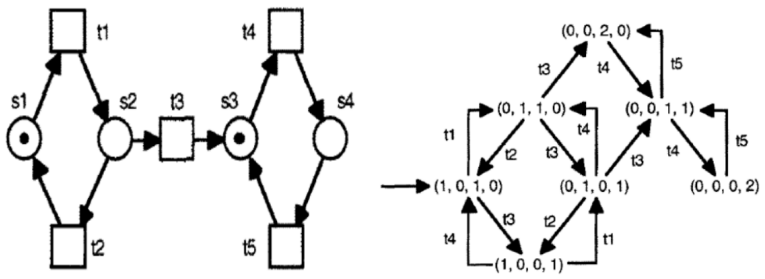

Figure 4.8: Petri Net model and its reachability graph.

6. A Petri Net is **live** iff for each node of the reachability graph of the corresponding Petri Net, there exists a path i.e.

$$M_0 \rightarrow t_1 \rightarrow M_1 \rightarrow t_2 \rightarrow M_2 \cdots M_{i-1} \rightarrow t_i \rightarrow M_i$$

Here the sequence of path $t_1, t_2, \cdots t_i$ is such that it contains all the transitions of the Petri Net.

Chapter 5

ANALYSIS OF PETRI NETS USING MATRIX EQUATIONS

5.1 Introduction

Another technique for analysis of Petri Nets is Linear Algebra methods known as Matrix Equations. A Petri Net $PN = (P,T,I,O)$ can be defined as the tuple $PN = (P,T,A^-,A^+)$ where $A^-$is the matrix for input function and $A^+$is the matrix for output function of the Petri Net.
The matrix $A^-$and A^+have transitions as the rows $(t_1,t_2,\ldots\ldots,t_m)$ and the places of the Petri Net as columns of the matrix $(p_1,p_2,p_3,\ldots\ldots,p_n)$. Thus both input and output matrices are matrices of order $m \times n$.

$$\mathbf{A^-[j,i]} = \#\left(\mathbf{p_i}, \mathbf{I}\left(\mathbf{t_j}\right)\right)$$

Here the quantity represents inputs to transition t_j from place p_i, where $j = (1,2,\ldots,m)$ and $i = (1,2,\ldots,n)$.

$$\mathbf{A^+[j,i]} = \#\left(\mathbf{p_i}, \mathbf{O}\left(\mathbf{t_j}\right)\right)$$

here the quantity represents outputs from transition t_j to place p_j, where $j = (1,2,\ldots,m)$ and $i = (1,2,\ldots,n)$.

Let $e[j]$ be the m unit vector which is zero everywhere except the jth element in the tuple. The transition t_j is expressed through this unit m vector $e[j]$, where

$$e[j] = (0,0,0,\ldots 1,0,0,0\ldots.0)_{1\times m}.$$

Now a transition t_j is enabled in a marking M if,

$$\mathbf{M}_{(1\times\mathbf{n})} \geq \mathbf{e[j]}_{(1\times\mathbf{m})} \cdot \mathbf{A}^-_{(\mathbf{m}\times\mathbf{n})}$$

If t_j is fired in the marking M then the next state function is given by,

$$\phi\left(\mathbf{M}, \mathbf{t_j}\right) = \mathbf{M} - \mathbf{e[j]} \cdot \mathbf{A^-} + \mathbf{e[j]} \cdot \mathbf{A^+}$$

Since earlier we saw that $M'(p_i) = M(p_i) - \#\left(p_i, I\left(t_j\right)\right) + \#\left(p_i, O\left(t_j\right)\right)$.
Hence

$$\phi\left(M, t_j\right) = M + e[j].\left(A^+ - A^-\right) = M + e[j].A$$

where $A = A^+ - A^-$is the **Composite Change Matrix** [6].

Now consider a sequence of transition firings $\alpha = t_{j_1} t_{j_2} \ldots . t_{j_k}$.
Then

$$\begin{aligned} \phi(M,\alpha) &= \phi\left(M, t_{j_1} t_{j_2} \ldots . t_{j_k}\right) \\ &= M + e[j_1] \cdot A + e[j_2] \cdot A + \ldots .. + e[j_k] \cdot A \\ &= M + (e[j_1] + e[j_2] + \ldots .. + e[j_k]) \cdot A \end{aligned} \tag{5.1}$$

Let $g(\alpha) = e[j_1] + e[j_2] + \ldots + e[j_k]$ be the *firing vector* of sequence $t_{j_1} t_{j_2} \ldots . t_{j_k}$. The i^{th} element of $g(\alpha)$ would give the number of times the transition t_j has been fired in sequence $t_{j_1} t_{j_2} \ldots .. t_{j_k}$.
Thus $g(\alpha)$ is the vector of non negative integers.

5.2 Matrix Approach for Reachability Problem

If M' is reachable from a marking M then there exist a sequence of transition firings which will lead M to M', i.e.

$$M' = M + x.A$$

Here x is a solution vector of non-negative integers; if the above equation has no solution then M' is not reachable from M.
Consider the Petri Net in Figure 5.1,

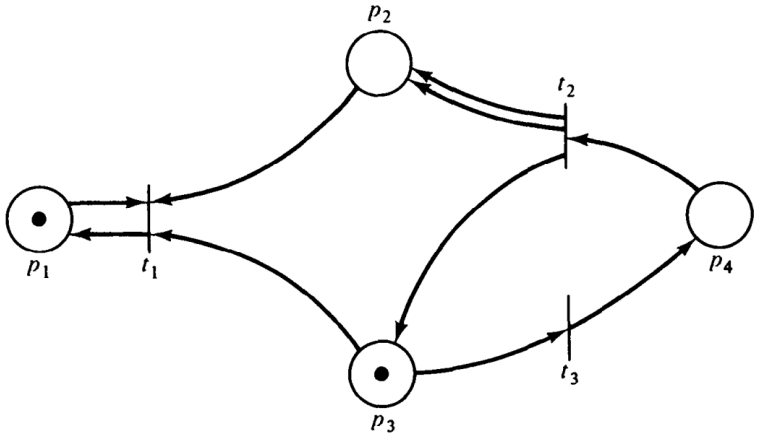

Figure 5.1: A Petri net model for Matrix Analysis.

The initial marking of the Petri Net in Figure 5.1 is $M = (1,0,1,0)$. The input and output matrices are given by ,

$$A^- = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \text{and} \quad A^+ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The composite matrix is given by

$$A = A^+ - A^- = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & -1 & 0 \\ 0 & 2 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

From the figure of the Petri Net model we can see that if t_3 is fired then the new marking would be $M' = (1,0,0,1)$. i.e. $M \rightarrow t_3 \rightarrow M'$.

Important Remarks:

1. Now we obtain this result from matrix analysis.

$$\begin{aligned} M' &= (1,0,1,0) + (0,0,1) \cdot \begin{bmatrix} 0 & -1 & -1 & 0 \\ 0 & 2 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \\ &= (1,0,1,0) + (0,0,-1,1) \\ &= (1,0,0,1) \end{aligned}$$

   Thus we have validated the above result through matrix analysis.

2. If the sequence of firing transitions is given i.e. $M \to \alpha \to M'$ and $\alpha = t_3 t_2 t_3 t_2 t_1$

$$\begin{aligned} g(\alpha) &= e[t_3] + e[t_2] + e[t_3] + e[t_2] + e[t_1] \\ &= e[t_1] + 2e[t_2] + 2e[t_3] \\ &= (1,2,2) \end{aligned}$$

   This $g(\alpha) = (1,2,2)$ is the **Firing Vector**. Now to obtain the new marking after applying this sequence of transition firing,

$$\begin{aligned} M' &= (1,0,1,0) + (1,2,2). \begin{bmatrix} 0 & -1 & -1 & 0 \\ 0 & 2 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \\ &= (1,0,1,0) + (0,3,-1,0) \\ &= (1,3,0,0) \end{aligned}$$

   Hence after firing the sequence of transitions α to the initial marking M the resultant marking obtained is $M' = (1,3,0,0)$.

3. It can be shown that the marking $(1,8,0,1)$ is reachable from $(1,0,1,0)$.
   Consider,

$$(1,8,0,1) = (1,0,1,0) + x \begin{bmatrix} 0 & -1 & -1 & 0 \\ 0 & 2 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

$$(0,8,-1,1) = (0,0,0,0) + x. \begin{bmatrix} 0 & -1 & -1 & 0 \\ 0 & 2 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

   which has a solution $x = (0,4,5)$. Thus marking $(1,8,0,1)$ is reachable from $(1,0,1,0)$.

4. It can be shown that the marking $(1,7,0,1)$ is not reachable from $(1,0,1,0)$

$$(1,7,0,1) = (1,0,1,0) + x \begin{bmatrix} 0 & -1 & -1 & 0 \\ 0 & 2 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

$$(0,7,-1,1) = (0,0,0,0) + x \begin{bmatrix} 0 & -1 & -1 & 0 \\ 0 & 2 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

   For the above equation x has no solution, and since x was the firing vector. Thus we know that no such sequence of transitions exist such that $(1,7,0,1)$ is reachable from $(1,0,1,0)$.

5.3 Matrix Approach for Conservation Problem

For the given Marked Petri Net, if we want to know whether the Petri Net is conservative or not, then the matrix approach can be applied.
For conservation, we must find a *non-zero weighing vector* u for which the weighted sum of tokens overall reachable markings is constant.

If u is a $n \times 1$ vector and M_0 is the initial marking and M' is the arbitrary reachable marking from M_0 then for conservation,

$$M_0 \cdot u = M' \cdot u$$

Since M' is reachable from M_0, then after a sequence of transitions firing α we get,

$$\begin{aligned} M' &= \phi\,(M_0, \alpha) \\ &= M_0 + g(\alpha).A \end{aligned}$$

therefore, we have

$$\begin{aligned} M_0 \cdot u &= M' \cdot u \\ M_0 \cdot u &= (M_0 + g(\alpha) \cdot A) \cdot u \\ M_0 \cdot u &= M_0 \cdot u + g(\alpha) \cdot A \cdot u \\ \Rightarrow g(\alpha) \cdot A \cdot u &= 0 \quad (\text{ true for all } g(\alpha)) \\ A \cdot u &= 0 \end{aligned}$$

Hence we can formulate a test for the conservation of Petri Net.

Definition: A Petri Net is said to be **conservative** iff there exists a vector u of positive integers such that $A.u = 0$, where A is the composite change matrix (or the incidence matrix) of the given Petri Net.
Thus through the matrix approach, we can directly obtain tests for the conservation of Petri Net.

5.4 Issues in Matrix Approach to the Analysis of Petri nets:

5.4.1 Lack of sequencing information in Firing Vector

Consider the following Petri Net in Figure 5.2,
Here the initial marking is $M_0 = (1,0,0,0,0)$. If we want to know that if $(0,0,0,0,1)$ is reachable from $(1,0,0,0,0)$ then,

$$(0,0,0,0,1) = (1,0,0,0,0) + x. \begin{bmatrix} -1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 & -1 \\ 0 & 0 & -1 & 1 & 1 \end{bmatrix}$$

This equation does not have a unique solution.
The solution for x is $(1, x_2, x_6 - 1, 2x_6, x_6 - 1, x_6)$.

If we let $x_6 = 1, x_2 = 1$ and now solving for x we will get the firing vector $g(\alpha) = (1,1,0,2,0,1)$. Thus this implies that both the sequences $t_1t_2t_4t_4t_6$ and $t_1t_4t_2t_4t_6$ correspond to the same vector. Hence we just know about the number of transition firings and nothing about the order of transitions firings.

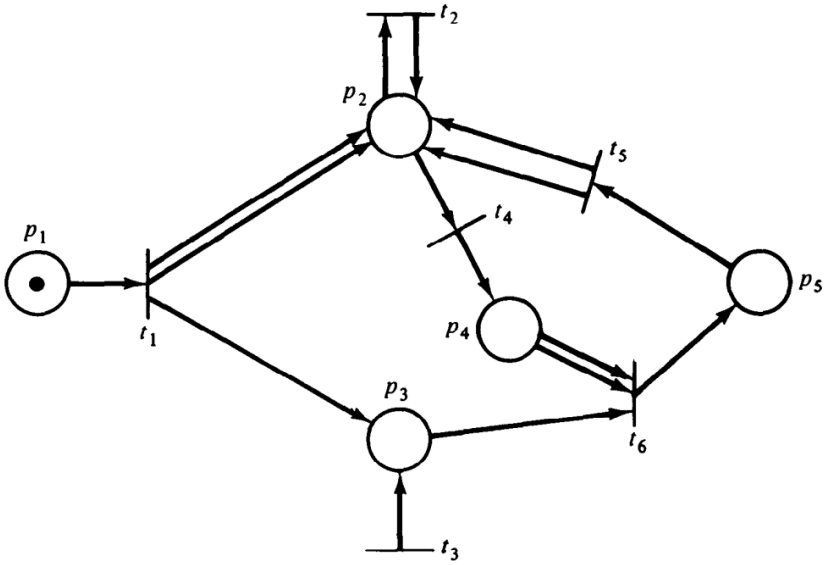

Figure 5.2: A Petri Net model.

5.4.2 Solution to Matrix Equation is necessary for reachability but not sufficient

Solving for x in $M' = M + x.A$ in reachability problem is necessary but not a sufficient condition. Consider the Petri Net in Figure 5.3, Here the initial marking is $(1,0,0,0)$. If we want to know

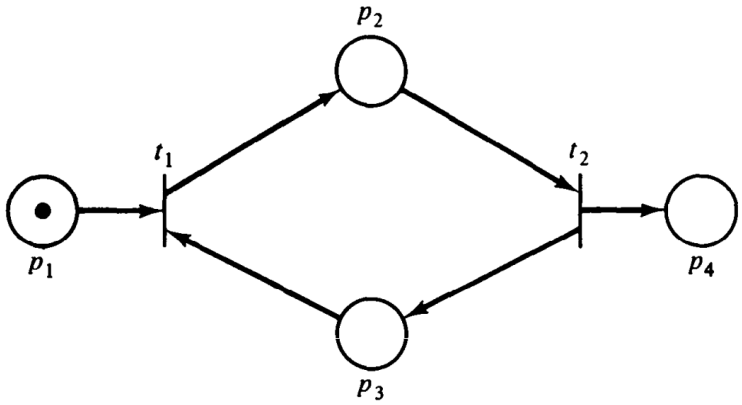

Figure 5.3: A Petri net model for Matrix Methods.

that $(0,0,0,1)$ is reachable from $(1,0,0,0)$,

$$(0,0,0,1) = (1,0,0,0) + g(\alpha) \cdot \begin{bmatrix} -1 & 1 & -1 & 0 \\ 0 & -1 & 1 & 1 \end{bmatrix}$$

$\Rightarrow g(\alpha) = (1,1)$ corresponds to the sequence $t_1 t_2$ or $t_2 t_1$. But if we observe the Petri Net we see that neither of these two transitions are possible since t_1 and t_2 are not enabled in the initial marking $(1,0,0,0)$.

Thus a solution to the matrix equation , $M' = M_0 + x \cdot A$ is not sufficient to prove reachability of the marking.

Important Remarks:

The incidence matrix as used in linear algebra techniques helps in understanding the dynamic behavior of a system.

1. It provides test for conservative property.
2. It can tell if a marking is reachable from an initial marking or not.
3. Nothing much about the behavioral properties (except reachability) can be derived through incidence matrix.

Chapter 6

MODELING OF CURRICULUM USING PETRI NETS

6.1 Introduction

Rates of student retention are being examined in various nations as part of government funding policies for institutions of higher learning. Retention rates are a growing concern for these institutions as a result. It's been discovered that structural characteristics of higher training establishments, consisting of enrolment length, selectivity, and management, have massive institutions with student drop-out. In this respect, student retention and drop-out rates may be greatly influenced by the regularity of a professional curriculum.[4]

A syllabus can be thought of as a set of rigid publications; their collection, and the institutional policies governing a student's continued enrollment for an academic program (credits of course, range of repetition of a course by a student, student development agenda, etc.). Many higher education institutions, in particular, put up student progress schedules that specify the minimal prerequisites for continuing in an academic program. [5]

Models such as Petri nets can be used as a powerful tool for helping decision-makers at all levels and types of organizations make better decisions. Throughout this article, we recommend the use of a Petri net approach to syllabus management as a way to help faculty, students, and university administrators make decisions about how to manage syllabuses. This model may also be used by management to assess the internal coherence of a current and forthcoming development syllabus, as well as to determine a student's career advancement and the likelihood of a student dropping out.[6]

Petri net characteristics and their suitability for representing conditions to stay a student in a course are discussed in section 6.2 and the Petri net Model of Curriculum is shown in section 6.3; its reachability tree is shown in subsection 6.3.2 with some related works based on the model are also discussed further in subsection 6.3.3 and then a final overview of semesters is shown in 6.4.

For a better point of understanding, the curriculum of M.Sc Applied Mathematics, Delhi Technological University is taken; in which there are four semester's and each semester has a certain number of courses that are listed in Table 6.1.

6.2 Guidelines for Modeling the Curriculum

In this paper, we have taken the following assumptions/rules to model the curriculum of M.Sc Applied Mathematics.

a) Requirements for different courses is considered.

b) The maximum number of repetitions allowed for a course is 2.

Table 6.1: M.Sc Applied Mathematics Curriculum Courses.

Semester- I	**Abbreviation**	**Semester- II**	**Abbreviation**
Abstract Algebra	AA	Complex Analysis	CA
Real Analysis	RA	Partial Differential Equations	PDE
Ordinary Differential Equations	ODE	Topology	TOPO
Discrete Mathematics	DM	Linear Algebra	LA
Mathematics Statistics	MS	Numerical Analysis	NA
C++ Programming Language	C++	Python Programming Language	PYTHON
Communicative English	ENG	Fundamentals of Computers	FOC
Semester- III	**Abbreviation**	**Semester- IV**	**Abbreviation**
Functional Analysis	FA	Measure and Integration	MI
Operation Research	OR	Dissertation-2	DT II
Dissertation-1	DT	Discipline Specific Elective-3	DSE III
Discipline Specific Elective-1	DSE I	Discipline Specific Elective-4	DSE IV
Discipline Specific Elective-2	DSE II	General Elective Course-2	GEC II
General Elective Course-1	GEC I		

c) The maximum time a student may remain in a program of study is considered.

A Petri net enables the formal mode description of systems whose dynamics are characterized by *Concurrency*, *Conflicts*, *Synchronization*, and *Mutual exclusion*. Except for mutual exclusion, prerequisites exhibit all of these traits. For instance, in Figure 6.2 we can see that the courses Abstract Algebra, Real analysis, Ordinary differential equations, Discrete mathematics, Mathematical statistics, C++ language, and Communicative English are all concurrent since they are enabled and do not interact with each other i.e. they can occur independently.

Synchronization exists when passing more than one course is required to enroll in one, as it occurs with Functional Analysis when passing Linear Algebra and Complex Analysis are prerequisites.

A Conflict does not express itself overtly, yet it still exists because a course can be passed or not.

6.3 Petri Net Model of the Curriculum

Technically, each node in the acyclic-directed graph of prerequisites corresponds to a course, and each arc denotes the dependencies (prerequisites) between courses. The seven courses shown in Table 6.1, for instance, are abstract algebra, real analysis, ordinary differential equations, discrete mathematics, mathematical statistics, C++ language, and communicative english. These courses must all be taken in the first semester of the academic program and are required for every student.

The arc connecting the courses makes it obvious that enrolling first and passing every subject are prerequisites for completing the semester successfully. In a Petri net, a course is represented by a place, and a place with a token next to it indicates that the course is presently being taken. Once enrolled in a course, there are only two possible outcomes: pass or fail. As a course can only be taken twice, Figure 6.1 can be used to illustrate this. A token in a place represents a student enrolled in a course; indicates first-time enrollment by Course1. Failure in a course necessitates retaking it, as is the case with the term Fail1(shown through a token in Course2).

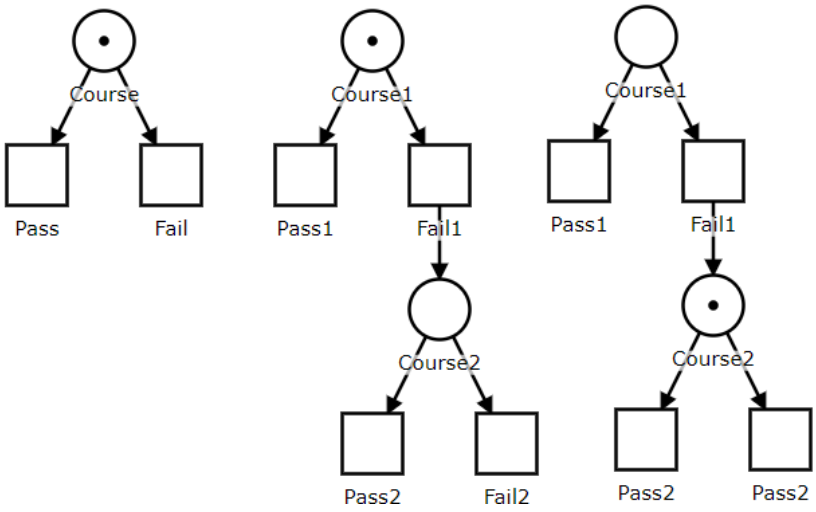

Figure 6.1: A course execution.

The firing of the transition when a student receives a Fail2 grade, it means that they failed the course a second time, which requires removing them from the program in Figure 6.1. The firing of transitions Pass1 or Pass2, on the opposite side, demonstrates that the course was successfully finished on either the first or second try, respectively. [7],[8],[9],[10],[11]

6.3.1 Semester I

First of all, we take abbreviations for the courses of *Semester I* given in Table 1 for convenience in modeling Petri Net and its further analysis.

The place labeled Enrolled in Figure 6.2 is a designation of a student's enrollment in the academic program. Places AA1, AA2, RA1, RA2, etc., represent different possibilities for passing or failing the courses. A student who has completed all of the program's courses is represented by the place Psemend. Transition t_1 marks the start of a semester, and firing it places tokens in places AA1, RA1, ODE1, DM1, MS1, C++1, and ENG1. This universal condition means that a student, once registered, must take courses AA, RA, ODE, DM, MS, C++, and ENG. Firing the transitions $t_2, t_4, t_6, t_8, t_{10}, t_{12}, t_{14}$ respectively represent passing courses AA, RA, ODE, DM, MS, C++, and ENG in the first attempt. However, triggering the transitions $t_{16}, t_{18}, t_{20}, t_{22}, t_{24}, t_{26}, t_{28}$ is equivalent to passing the same courses on a second try. Failures occur for the first time when transitions $t_3, t_5, t_7, t_9, t_{11}, t_{13}, t_{15}$ are triggered. Failures that occur a second time lead to dismissal from the curriculum i.e. this happens when transitions $t_{17}, t_{19}, t_{21}, t_{23}, t_{25}, t_{27}, t_{29}$ are fired.

It is necessary to rule out the potential of a student taking courses because failing a course twice means a student is pulled out of the curriculum. To do this, we use place Ppass, which facilitates transitions $t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}, t_{14}, t_{15}$, $t_{16}, t_{17}, t_{18}, t_{19}, t_{20}, t_{21}, t_{22}, t_{23}, t_{24}$, $t_{25}, t_{26}, t_{27}, t_{28}, t_{29}$. Transitions $t_2, t_3, t_4, t_5, t_6\, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}, t_{14}, t_{15}, t_{16}, t_{18}, t_{20}, t_{22}, t_{24}, t_{26}, t_{28}$ return a token to Ppass, but firing transitions t_{17}, t_{19}, t_{21}, t_{23}, t_{25}, t_{27}, t_{29} the prohibition of taking any more courses after a second failure in a course. The state of Pfail, from which it is impossible to recover, is reached as a result of these failed transitions.

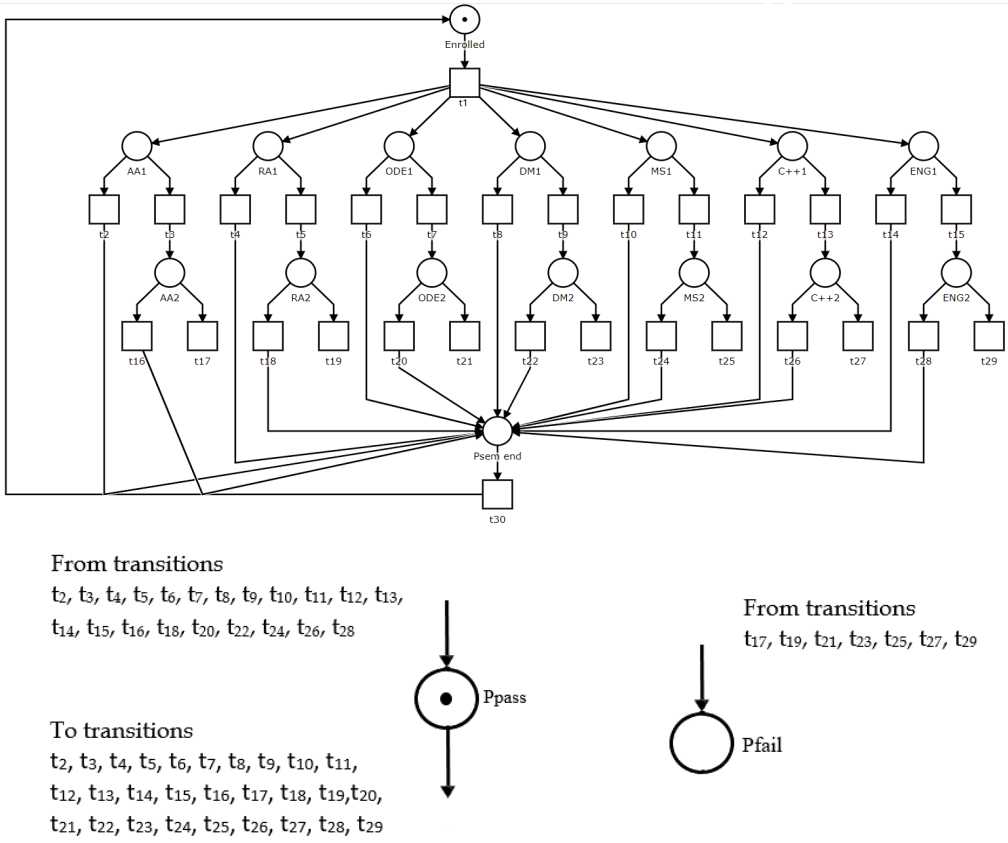

Figure 6.2: The Petri net model of Semester-I.

6.3.2 Reachability Tree of the Petri net model of Semester-I

The reachability tree shows a reachability set of Petri net or we can say all possible sequences of transition firings. The reachability tree of the Petri net model of Semester-I is shown in Figure 6.3.

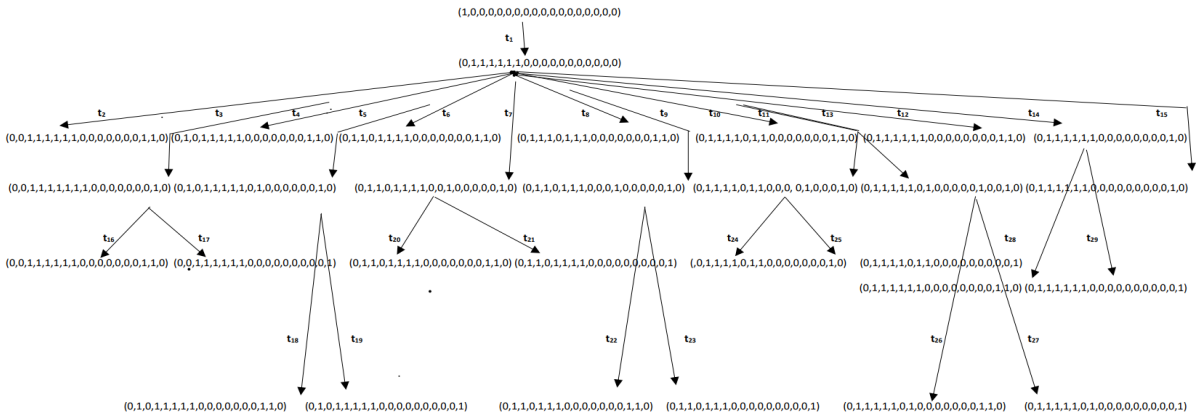

Figure 6.3: The Reachability Tree of the Petri net model of Semester-I.

6.3.3 Analysis of the Petri net model of Semester-I

- **Safeness and Boundedness:**

  Since, it is clear from the above Petri net model that each place can have at most one token in it, except Psemend, as it can have many tokens, and hence, the Petri Net Model of Semester I is not Safe.

  Although, it is bounded by the number of students enrolled in a class of Semester I. Therefore, the Petri Net Model of Semester I is bounded.

- **Conservation:**

  The Peri net model of Semester-I is not strictly conservative, since

  $$\sum_{p_i \in P} M_0(p_i) = 1$$

  and after firing transition t_1 we have

  $$\sum_{p_i \in P} M'(p_i) = 7.$$

  Hence, The Peri net model of Semester-I is not strictly conservative.

- **Liveness and Deadlock:**

  The Petri net model is live in the given case since there is a means to trigger a transition in any marking, and if a Petri net is live, no deadlock occurs.

6.4 Semester II, III, and IV Overview

For Semesters II, III, and IV, the Petri Net Model can be shown in the Figure 6.4 where an individual student represented as a token; goes to the next semester if he passed the previous semester and we can proceed further for the working of the entire Petri net model in a similar way as done for Semester-I in subsections 6.3.1, 6.3.2 and 6.3.3; its behavioral and structural properties can also be identified parallelly.

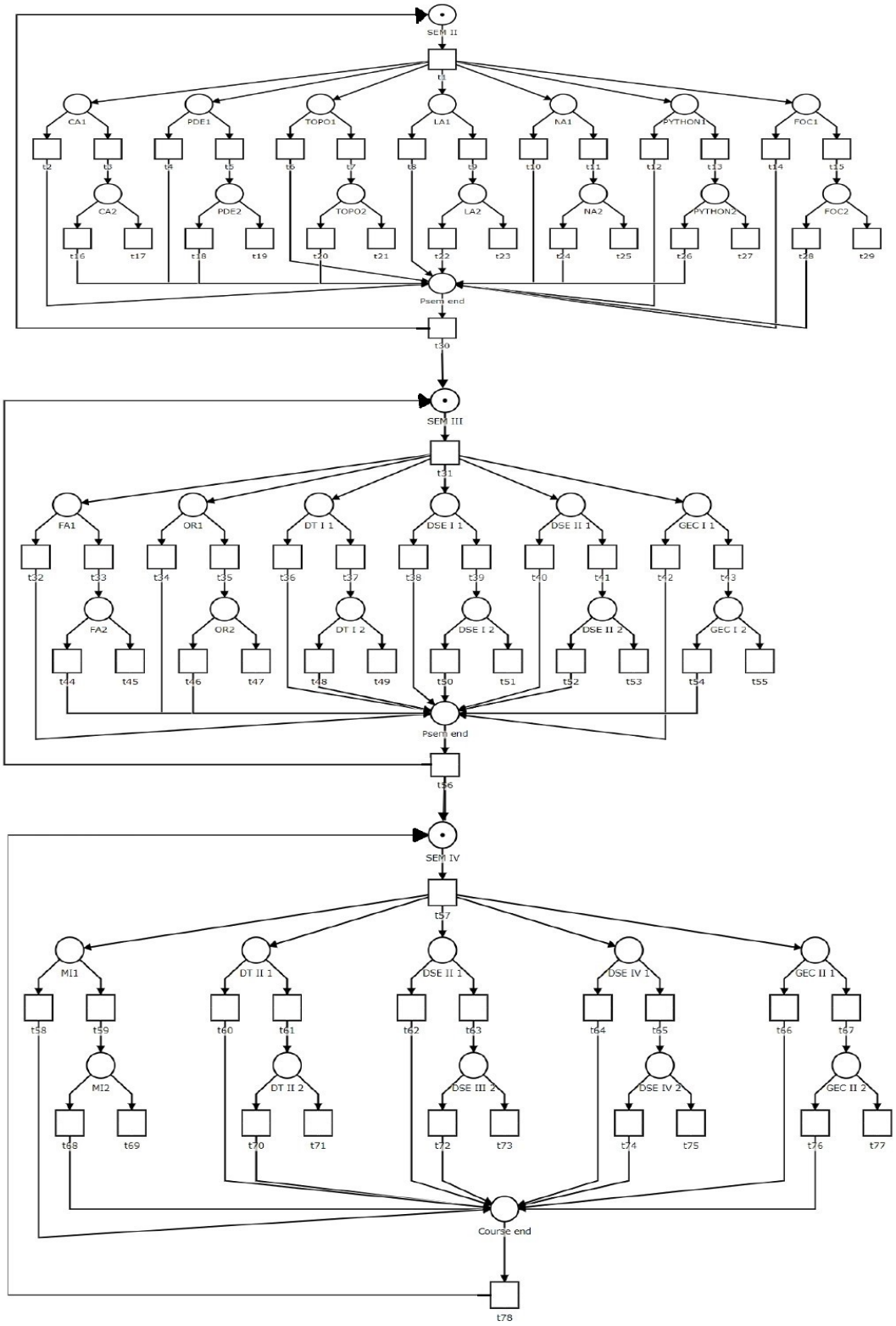

Figure 6.4: The overview of the Petri net model of Semesters II, III, and IV.

Chapter 7

CONCLUSION AND FUTURE WORK

Using a Petri net model of the curriculum, we have presented an approach to analyzing and confirming course prerequisites. The model's consistency and ability to be generated automatically have also been confirmed. Additionally, it is shown how administrators and lecturers could utilize this model to advise students on corrective activities to lower drop-out rates brought on by poor course sequencing.

The model can be improved with a representation of the number of credits given to each course in the following phase of this project. An advance table can be generated that will check to see if the student has completed the minimum number of credits needed since enrollment in the program. We intend to use the model on a group of students in a later phase and create a quantitative analysis utilizing the stochastic extension of Petri nets and this model can also be looked upon on high-level nets and timed nets together with their applications.

Chapter 8

APPENDIX

Table 8.1: Marking of the proposed Petri Net Model of the Curriculum.

Marking	Value of Marking
M_0	$\{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\}$
M_1	$\{0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0\}$
M_2	$\{0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,1,1,0\}$
M_3	$\{0,0,1,1,1,1,1,1,1,0,0,0,0,0,0,0,1,0\}$
M_4	$\{0,1,0,1,1,1,1,1,0,0,0,0,0,0,0,1,1,0\}$
M_5	$\{0,1,0,1,1,1,1,1,0,1,0,0,0,0,0,0,1,0\}$
M_6	$\{0,1,1,0,1,1,1,1,0,0,0,0,0,0,0,1,1,0\}$
M_7	$\{0,1,1,0,1,1,1,1,0,0,1,0,0,0,0,0,1,0\}$
M_8	$\{0,1,1,1,0,1,1,1,0,0,0,0,0,0,0,1,1,0\}$
M_9	$\{0,1,1,0,1,1,1,0,0,0,1,0,0,0,0,0,1,0\}$
M_{10}	$\{0,1,1,1,1,0,1,1,0,0,0,0,0,0,0,1,1,0\}$
M_{11}	$\{0,1,1,1,1,0,1,1,0,0,0,0,1,0,0,0,1,0\}$
M_{12}	$\{0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,1,1,0\}$
M_{13}	$\{0,1,1,1,1,1,0,1,0,0,0,0,0,1,0,0,1,0\}$
M_{14}	$\{0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,1,0\}$
M_{15}	$\{0,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,1,0\}$
M_{16}	$\{0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,1,1,0\}$
M_{17}	$\{0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,1\}$
M_{18}	$\{0,1,0,1,1,1,1,1,0,0,0,0,0,0,0,1,1,0\}$
M_{19}	$\{0,1,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,1\}$
M_{20}	$\{0,1,1,0,1,1,1,1,0,0,0,0,0,0,0,1,1,0\}$
M_{21}	$\{0,1,1,0,1,1,1,1,0,0,0,0,0,0,0,0,0,1\}$
M_{22}	$\{0,1,1,0,1,1,1,0,0,0,0,0,0,0,0,1,1,0\}$
M_{23}	$\{0,1,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,1\}$
M_{24}	$\{0,1,1,1,1,0,1,1,0,0,0,0,0,0,0,0,1,0\}$
M_{25}	$\{0,1,1,1,1,0,1,1,0,0,0,0,0,0,0,0,0,1\}$
M_{26}	$\{0,1,1,1,1,1,0,1,0,0,0,0,0,0,0,1,1,0\}$
M_{27}	$\{0,1,1,1,1,1,0,1,0,0,0,0,0,0,0,0,0,1\}$
M_{28}	$\{0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,1,1,0\}$
M_{29}	$\{0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,1\}$

References

[1] J. L. Peterson, "Petri nets, department of computer sciences university of texas at austin austin, texas 78712."

[2] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[3] J. L. Peterson, "Petri nets," *ACM Computing Surveys (CSUR)*, vol. 9, no. 3, pp. 223–252, 1977.

[4] F. Omrani, A. Harounabadi, V. Rafe *et al.*, "An adaptive method based on high-level petri nets for e-learning," *Journal of Software Engineering and Applications*, vol. 4, no. 10, p. 559, 2011.

[5] R. Carvajal-Schiaffino and L. Firinguetti-Limone, "Petri net based modelling of a career syllabus," *INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL*, vol. 9, no. 4, pp. 397–407, 2014.

[6] M. Kuchárik and Z. Balogh, "Student learning simulation process with petri nets," in *Recent Developments in Intelligent Computing, Communication and Devices: Proceedings of ICCD 2017*. Springer, 2019, pp. 1115–1124.

[7] A. Armstrong and L. Fuhua, "Modeling and personalizing curriculum using petri nets," in *The 18 th International Conference on Computers in Education*, 2010, p. 10.

[8] L. Nyéki, "Modelling some higher education processes using petri nets," *Journal of Applied Multimedia*, vol. 2, pp. 11–16.

[9] H. Indzhov, D. Blagoev, and G. Totkov, "Executable petri nets: Towards modelling and management of e-learning processes," in *Proceedings of the International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing*, 2009, pp. 1–6.

[10] D. C. Borges, H. B. Neto, and J. N. de Souza, "Work in progress—petri nets as applied to the modeling of e-learning cooperative systems," in *2010 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2010, pp. F1D–1.

[11] Z. Balogh and M. Kuchárik, "Predicting student grades based on their usage of lms moodle using petri nets," *Applied Sciences*, vol. 9, no. 20, p. 4211, 2019.

Details of Candidate's Publications

The Paper titled **"Modeling of Curriculum using Petri nets"** is accepted and presented(online) in the conference titled *"Third International Conference on Secure Cyber Computing and Communications(ICSCCC 2023)*, held on 26th-28th May, 2023 at Dr B R Ambedkar National Institute of Technology, Jalandhar, Punjab, India.

Accepted papers will be submitted for inclusion into IEEE Xplore.

PAPER NAME

Final thesis Msc.pdf

AUTHOR

Lubhavika Kishore

WORD COUNT

12275 Words

CHARACTER COUNT

54303 Characters

PAGE COUNT

45 Pages

FILE SIZE

2.4MB

SUBMISSION DATE

May 21, 2023 7:27 PM GMT+5:30

REPORT DATE

May 21, 2023 7:28 PM GMT+5:30

17% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 14% Internet database
- 11% Publications database
- Crossref database
- Crossref Posted Content database
- 6% Submitted Works database

Excluded from Similarity Report

- Bibliographic material
- Quoted material
- Small Matches (Less then 10 words)

Gmail

Lubhavika Parashar <lubhavika1999@gmail.com>

ICSCCC 2023 notification for paper 1842

ICSCCC 2023 <icsccc2023@easychair.org> 19 April 2023 at 18:26
To: Lubhavika Parashar <lubhavika1999@gmail.com>

Dear Authors,
Congratulations!!!
Your paper with paper id 1842 and “Modeling of Curriculum using Petri Nets” is accepted for ORAL presentation in the Third International Conference on Secure Cyber Computing and Communications which is scheduled to be held on 26th - 28th May, 2023 at Dr B R Ambedkar National Institute of Technology, Jalandhar, Punjab, India. Only registered and presented papers would be submitted to IEEE Xplore for possible inclusion.
Following guidelines must be followed while preparing the camera-ready paper to ensure your article is submitted to IEEE for final publication:
• The submitted paper should be in IEEE format. https://www.ieee.org/conferences/publishing/templates.html
• Paper length should be a maximum of 6 pages otherwise the additional payment required for extra pages is Rs.2000/page for Indian authors of all categories and USD 50/page for all categories of foreign authors.
• If the paper contains a plagiarism percentage exceeding 10%, it may be rejected at any stage of the processing.
Detailed instructions regarding the registration process are available at https://www.nitj.ac.in/icsccc2023/
Kindly complete the registration process by filling out the form @ https://forms.gle/giSybTWc9Hz7guMJ7
Please complete the registration process within seven days of receiving the paper acceptance mail. The Demand Draft/ E-receipt of Fund Transfer along with the registration form must be sent to icsccc2023@nitj.ac.in with in ten days of paper acceptance notification. Do not send hard copy of your paper.
After the completion of the registration process, we will initiate communication regarding accommodation.
For any further queries regarding registration, please contact icsccc2023@nitj.ac.in.
We're looking forward to your attendance at the ICSCCC 2023 conference.

With Best Wishes,
ICSCCC – 2023

Gmail **Lubhavika Parashar <lubhavika1999@gmail.com>**

Invitation to Register for ICSCCC-2023 for paper ID 1842

ICSCCC 2023 <icsccc2023@easychair.org> 25 April 2023 at 11:57
To: Lubhavika Parashar <lubhavika1999@gmail.com>

Dear Lubhavika Parashar,

We are delighted to inform you that your paper titled "Modeling of Curriculum using Petri Nets" has been accepted for presentation at ICSCCC-2023, which is scheduled to take place on May, 26-28th 2023 at Dr B R Ambedkar National Institute of Technology, Jalandhar. Congratulations on your accomplishment! Your contribution to the field has been recognized, and we are excited to have you share your work with the attendees.

As an accepted author, we would like to invite you to register for the conference and attend the event. It will be a great opportunity for you to present your work to a diverse audience, network with industry experts and peers, and learn about the latest research in your field.

To register for the conference, please fill out the registration form @ https://forms.gle/v2CYTP9ezyTfQpoU8. Please note that the registration deadline is 5th May, 2023, so we encourage you to sign up as soon as possible.

We request you to kindly ensure that you complete the registration process before the deadline so that we can include your paper in the conference proceedings.
We hope that you will be able to join us at ICSCCC-2023 and make it a success. If you have any questions or concerns, please do not hesitate to contact us at icsccc2023@nitj.ac.in.

Thank you for your contribution, and we look forward to welcoming you to ICSCCC-2023!
Best regards,
ICSCCC-2023 Organizing Committee

Gmail **Lubhavika Parashar <lubhavika1999@gmail.com>**

Presentation Certificate

Dr Harsh Verma <vermah@nitj.ac.in> 28 May 2023 at 12:37
To: Lubhavika Parashar <lubhavika1999@gmail.com>

Dear Lubhavika,

Your paper-id 1842 has been published in the abstract book of ICSCCC-2023. You will get presentation certificate for presenting your paper in the said conference shortly.

with regards,

--
Professor Harsh K Verma
Dean (Academic)
Department of Computer Science and Engineering
Dr B R Ambedkar National Institute of Technology
G T Road Bypass
Jalandhar - 144 011 (Punjab), India
Mobile No. +91 9463001601
Office No. +91-2690301/303 extn 2505(O)