MECH 6317 - DYNAMICS OF COMPLEX NETWORKS AND SYSTEMS
INSTRUCTOR - DR. MARK W. SPONG

Petri Nets in an Automated Assembly Line

Abishek Chandrasekhar

5/4/2016

Abstract: This project explains how Petri Nets can be applied in in an automated manufacturing plant that employs a number of industrial robots (manipulators) to perform a range of tasks such as welding, machining, painting etc. The first half of the project describes the components of Petri Nets, their properties and their analysis methods. Behavioral properties such as Reachability, Boundness and Safety, Conservativeness, Liveness and Reversibility are explained. In the second half, a sample layout of an industrial plant that utilizes eight industrial robots is created and the concept of Petri Nets are applied. The conclusion lists out the other applications of Petri Nets.

1. Introduction

Petri Nets were developed by Carl Adam Petri in 1962 and have been developed further ever since. Petri Nets are modelling tools that can be used to model systems that can be described as distributed, discrete, asynchronous, parallel, nondeterministic or stochastic. Petri Nets have a wide range of applications in various fields such as industrial systems [1], communication systems, workflow management systems[2], dataflow analysis[3], biological pathway modelling [4] etc. This particular project is based on Zurwaski R. and Zhou MengChu's paper [1] wherein they applied the concept of Petri Nets to an assembly system involving robots.

In this project, the concepts behind Petri Nets and their properties are described initially. As an example, a manufacturing layout is created. The model incorporates eight robots that are used to perform six operations. A Petri Net model is made based on the example. The properties of the Petri Net are described with respect to the model system. The analysis methods of Petri Nets are described in the final part of the project.

2. Petri Net

2.1 Objects in a Petri Net and their representations

Petri Nets are bipartite, directed graphs. There are edges and two kinds of nodes (Transitions and Places). Petri Nets are bipartite because no two places or no two transitions can be connected directly.

Edges: The weighted edges are directed arrows connecting the transitions and places. Parallel edges can be drawn and are represented by weights. The default weight of the edge is one.

Transitions: These are nodes that are used to represent an event occurring. For example, in manufacturing, they can be used to describe a machining operation. In computer processing, they can be used to describe the event of processing. They are usually represented by a rectangular solid block or sometimes a box.

Places: These are nodes that are used to describe conditions before or after transitions. For example, they can represent input/output buffers that store data in computers, before and after processing, or input/output palettes in manufacturing. It is an input place if the arcs connect them to a transition and an output place if the arc connects the transition to a place. They are represented by circles.

Tokens: Places can contain tokens, represented by small dots that are used to represent the current state of the system at a particular time and also represent dynamic behavior in a system. The number of tokens in a place can represent the number of resources. In the case of manufacturing, they can be used to represent the quantity of materials or the number of tools in a particular station. An example of a Petri Net can be seen in Figure 1.

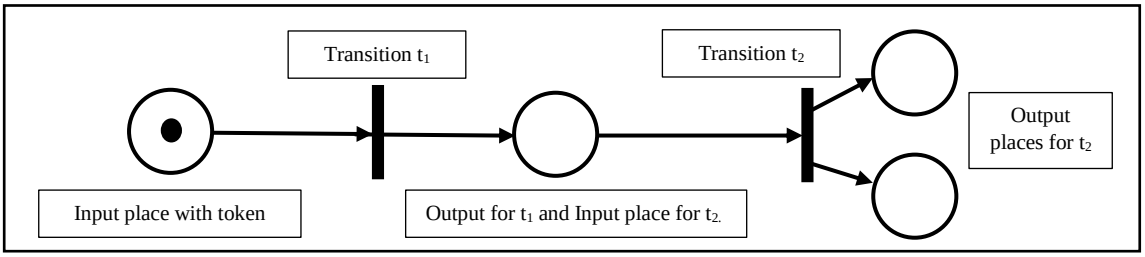

Figure 1. Example of a Petri Net

2.2 Definition

A Petri Net can be defined (according to[1]) as PN = (P, T, I, O, M_0). They can also be defined (according to [4],[5]) as PN = (P, T, F, W, M_0) where

P = {p_1, p_2 ... p_m} is a finite set of places.

T = {t_1, t_2 ... t_n} is a finite set of transitions, $P \cup T \neq \emptyset$, and $P \cap T = \emptyset$.

I: (P x T) → N is an input function that defines directed arcs from places to transitions, where N is a set of nonnegative integers. Similarly, O: (P x T**)** → N is an output function which defines from transitions to places.

M_0: P → N is the initial marking. A (mx1) vector M, where the elements are denoted as M_p, Number of tokens in corresponding places 'p'.

$F \subseteq (P \times T) \cup (T \times P)$ is the set of arcs.

W is the weight of the arcs. The default value of the arcs is 1.

2.3 Enabling and Firing rule

There are rules that govern the flow of tokens:

Enabling rule: A transition 't' is said to be enabled if each input place 'p' of 't' contains at least the number of tokens equal to the weight of the arc connecting 'p' to 't'.

Firing rule: An enabled transition may or may not fire depending on additional interpretation.

A firing of an enabled transition, 't' removes from each input place 'p' the number of tokens equal to the weight of the arc connecting 'p' to 't'. It also deposits in each output place 'p', the number of tokens equal to the weight of the arc connecting 't' to 'p'.

2.4 Inhibitors

Inhibitors are a special kind of arc. They can be used to check whether a token is present in a particular place. If an inhibitor is present, it changes the enabling conditions of a transition. In other words, the absence of a token enables the transition. It is represented by an arc with a circle at its end.

2.5 Properties

There are two kinds of properties, behavioral and structural. Behavioral properties depend on the marking of the Petri Net, whereas structural properties depend on the topology. The behavioral properties are described below according to [1], [5] and [6]:

2.5.1 Reachability

Reachability is a property which is used to test whether the system can reach a specific state. The firing sequences are given by $\sigma = M_0\, t_1\, M_1\, t_2 \ldots t_n\, M_n$ or $\sigma = t_1\, t_2 \ldots t_n$. The set of all possible markings is given by R(M_0), and all firing sequences is L(M_0).

2.5.2 Boundness and Safety

In systems, a place can only store a restricted amount of data/parts/tools without corrupting/breaking the resource. *A Petri Net is k-bounded if the number of tokens in any place p, where p∈P, is always less than or equal to k. Petri Net is safe if it is 1-bounded.*

2.5.3 Liveness and Deadlock

A Petri Net is said to be live, if there is a possibility to fire any transition of the Petri Net via some firing sequence. There are different levels of Liveness. A Transition 't' is said to be

-L1 live - (potentially firable) if it can be fired at least once in some firing sequence in $L(M_0)$.
-L2 live - if it can be fired at least k-times in $L(M_0)$, where k is any positive integer.
-L3 live - if it can be fired infinitely in some firing sequence in $L(M_0)$.
-L4 live - if it is L1 live in every marking in $R(M_0)$.

A deadlock (L0 live) is a marking wherein no transition can be fired. This happens in cases where an input place is connected to two different transitions in parallel. The weight of the acts connecting the place 'p_1' to t_1 and t_2 is equal to k. In case there is only k or less than 2k number of tokens, this leads to a deadlock. For deadlock to occur, four conditions must occur [1]:

Mutual Exclusion: A resource should be exclusively available to a process.

Hold and wait: The process is allowed to hold the resource while it waits for more resources.

No preemption: The resource that is already allocated to process cannot be removed from the process until the process releases the resource.

Circular wait: Two or more processes in a chain which are waiting for resources held by the process in the next chain.

2.5.4 Conservativeness

A Petri Net is conservative if the number or tokens is unchanged for each marking M that is reachable from the initial marking M_0.

2.5.5 Reversibility and Home state

A Petri Net is reversible if it's possible to reach the initial marking M_0 from any other marking M. In cases where it is not necessary to return to reach the initial state, a home state M_i is sufficient. A Petri Net is home state if it's possible to reach M_i from any other marking.

2.5.6 Coverability

A marking M in a Petri Net is coverable if there exists a marking M' such that $M'(p) \geq M(p)$ for each place 'p' in the Petri Net.

2.5.6 Persistence

If there are two enabled transitions, and if the firing of one does not disable the other, then the Petri Net is persistent.

In the example model that follows, only the behavioral properties are used to describe it, so the structural properties are not explained. The structural properties of Petri Nets are described briefly in [4] and in detail in [5].

2.6 Matrix Representation

In order to describe and analyze Petri Nets, they are given in matrix form.

- An incidence matrix A is obtained, where the elements are given by $a_{ij} = a_{ij}^{+} - a_{ij}^{-}$. a_{ij}^{+} is the weight of the arc from transition i to output place j; a_{ij}^{-} is the weight of the arc from input place j to transition i.
- We already know that the initial marking vector M_0 is an (m x 1) vector which gives tokens in places.
- The control vector μ_k is an (n x 1) vector where the ith position is filled in with 1 if the ith transition fires in the k^{th} marking.

Using these, the next marking matrix for the Petri Net can be calculated using the formula:

$$M_k = M_{k-1} + A^T \mu_k, \text{ where } k=1, 2, 3\ldots\text{(from [5])}$$

2.7 Conflict

Static Conflict: This happens when two transitions have the same input place. The place has only one resource to spare, or, in the case of k>1 weighted arcs, have only k tokens to spare.

Dynamic Conflict: This occurs when there are two transitions that are connected to one output place. The firing of one transition will disable the other.

These concepts can be applied to a number of systems. An example is shown below.

3. Automated Manufacturing Plant

In order to showcase an application of Petri Nets, a layout of a manufacturing plant is created. The plant incorporates eight Industrial Robots (manipulators) that perform six operations. There are eight stations (two input stations, four transfer stations and two output stations). The layout is shown below in Figure 2.

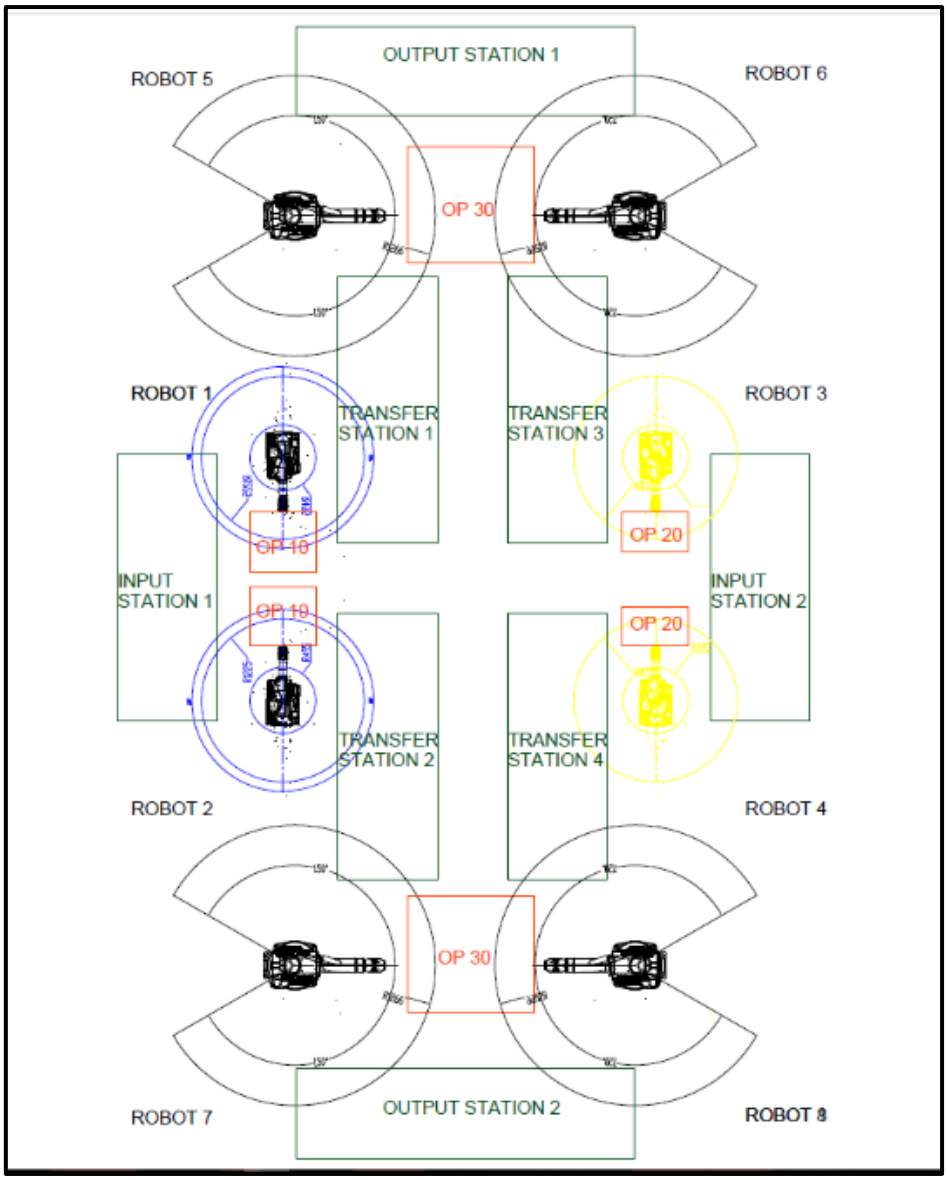

Figure 2. Automated Factory Layout
Draftsight drawing by: Abishek Chandrasekhar
Robot 2-D sketch source: ww.abb.com

- Input station 1 holds Part Type A and contains two parts at a time. Input station 2 holds Part Type B and also contains two parts.
- There are two robots 1 and 2, near I/P station 1 that each pick up one part each from the station and perform the operation OP 10 on their respective parts. Similarly, there are two robots 3 and 4, near I/P station 2 that perform operation OP 20 on their respective parts.
- After all four operations are complete (two OP 10 operations and two OP 20 operations), the robots transfer the parts to Transfer station 1, 2, 3 and 4 respectively.
- Once the part type A and type B are placed in transfer station 1 and 3, the two assembly robots 5 and 6 pick up the parts from the stations 1 and 3 and proceed to assemble the parts together. Similarly, robots 7 and 8 pick up the parts from station 2 and 4 and assemble the parts.
- The two assembled parts are placed in O/P station 1 and 2 respectively.

Based on the above description, the Petri Net for the model is given below in Figure 3.1

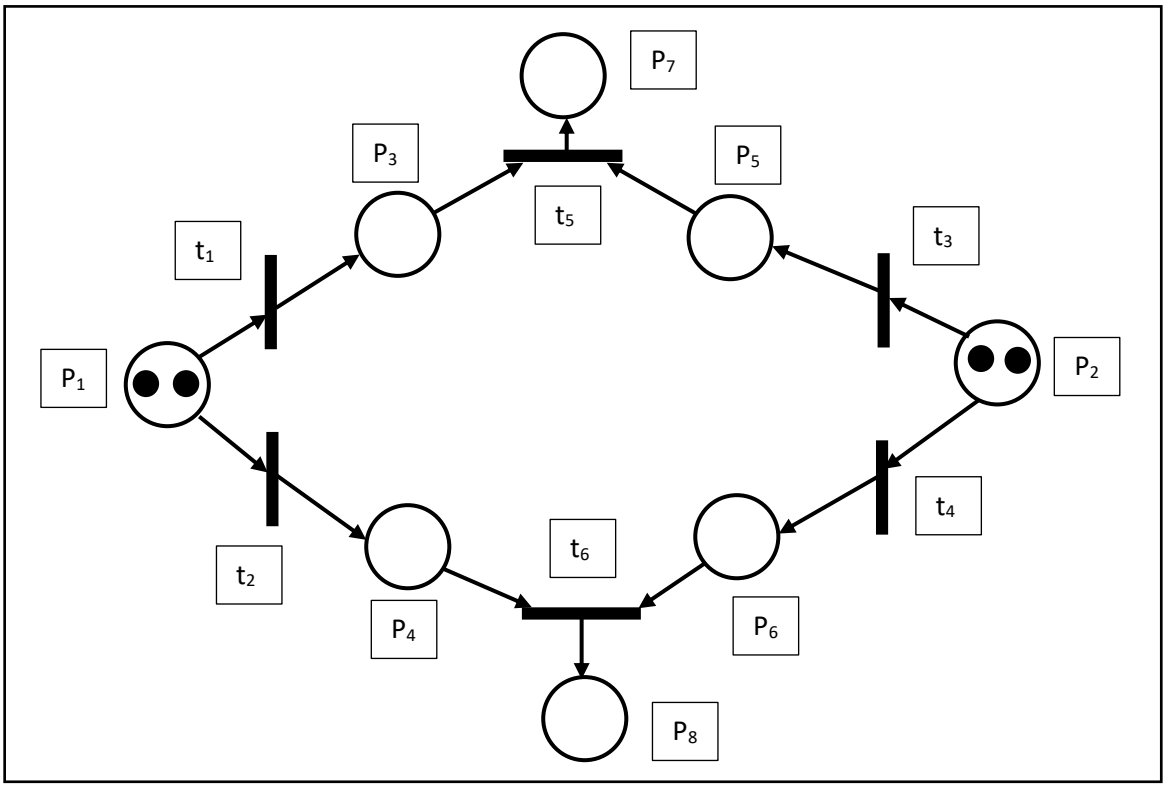

Figure 3. Petri Net model for the Layout

In the Petri Net described in Figure 1, the I/P stations, O/P stations and the transfer stations are places ($p_1 \ldots p_8$), the six operations are transitions ($t_1..t_6$) and the directed arcs describe the flow of the parts (tokens) through the model.

According to **2.2 Definition**, the Petri Net above can defined as:

PN = (P, T, I, O, M_0) (by definition from [1])

- P = {p_1, p_2, p_3 , p_4 , p_5, p_6, p_7, p_8}
- T = {t_1, t_2, t_3, t_4, t_5, t_6}
- Initial Marking matrix $M_0 = (2, 2, 0, 0, 0, 0, 0, 0)^T$.

- Control vector $\mu_k = (1, 1, 1, 1, 0, 0)^T$.
- $I = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}$; $O = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$
- $A = \begin{pmatrix} -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & 0 & 1 \end{pmatrix}$

According to formula defined in **2.6 Matrix Representation**, the next marking is: $M_1 = M_0 + A^T \mu_k$

$$= (2, 2, 0, 0, 0, 0, 0, 0)^T + \begin{pmatrix} -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} (1, 1, 1, 1, 0, 0)^T$$

$$= (2, 2, 0, 0, 0, 0, 0, 0)^T + (-2, -2, 1, 1, 1, 1, 0, 0)^T = (0, 0, 1, 1, 1, 1, 0, 0)^T.$$

- Therefore, the next marking matrix M_1 is $(0, 0, 1, 1, 1, 1, 0, 0)^T$.
- The same process is repeated to find the final marking matrix $M_2 = (0, 0, 0, 0, 0, 0, 1, 1)^T$.
- This indicates that at the end of the process, there are two tokens left. This implies that four resources are utilized to produce two final components in this process.

Next, according to **2.5 Properties**, the Petri Net can be characterized as follows.

- Reachability: It is possible for any marking to reach the final marking in the Petri Net. $M_0 \rightarrow M_1 \rightarrow M_2$.
- Boundness and Safety: The Petri Net for this model is 2 – bounded. (There are two tokens at p_1 and p_2). The Petri Net cannot be classified as "safe" as it is not 1- bounded.
- Liveness: The Petri Net is L4 live as it is live in every marking. This holds true as long as there are two tokens present in each of the places p_1 and p_2. There is no deadlock situation.
- Conservativeness: The Petri Net is not conservative as the number of tokens changes. The process starts with four tokens and ends with two tokens.
- Reversibility and Home state: In this particular model, it is not necessary to reach the initial marking from M, but it is possible to reach the home state M_2.
- Persistence: The Petri Net is persistent as the firing of any transition does not disable any other transition.

Conflict: In the above model, there is one possibility of static conflict. This happens in the case where there is only one token present in either p_1 or p_2. In other words, there is only one part present in the input palette. Only one of the corresponding transitions can be fired in such a case. (Either t_1 or t_2/ either t_3 or t_4). There is no possibility of dynamic conflict in this particular system. (Even though the output places p_7 and p_8 have two transitions each that are connected, in this particular case, the two parts are assembled together to form one single part, so there is no dynamic conflict).

4. Timed Petri Nets

For dynamic systems such as the one described above, further performance analysis can be done by introducing the parameter of time. In more complex systems, this can prove to be useful for scheduling, synchronization and calculating cycle times of processes.

For example, in the automation layout, each event can be assigned a time:

- The OP 10 operations can each have an operational time of 15 seconds.
- Similarly, OP 20 and OP 30 have operational times of 7 seconds and 30 seconds respectively.
- Suppose there is a delay while the components are in the places (I/P, O/P and transfer stations) the time delay can also be introduced into the system. (In cases where the transfer stations are conveyor belts, a certain amount of time is required for the next transition to be enabled).

When all the time periods of transitions and time delays are consolidated, the cycle time and the takt times for the manufacturing process can be calculated. This can be an effective means of performance analysis in such dynamical systems.

These are called Deterministic Timed Petri Nets. There are also Stochastic Timed Petri Nets where the delays are modelled as random variables as explained in [4].

5. Priority Petri Net

In the Petri Net in Figure 3. Petri Net model for the LayoutFigure 3, conflict may arise as stated before. In the case where there is only one token is available to transitions t_1 and t_2, only one transition maybe enabled and fired. In other complex Petri Nets, such a case may lead to deadlock, where in no transition is enabled and no marking sequence is able to fire. The solution for such problems is the use of Priority Petri Nets.

In a Priority Petri Nets, transitions are given individual priority. In the case where there is a shortage of resources, the transition with higher priority is allowed to be enabled and subsequently fired. The other transition is temporarily disabled till sufficient tokens become available in the input place that is present before the transitions.

To showcase this in the given model, consider the concurrent transitions.

- The transition t_2 is given a higher priority than t_1 and also t_4 is given a higher priority than t_3.
- In case there is only one token in either place p_1 or p_2, only transition t_2, t_3 and t_4 are enabled (or in the latter case, t_1, t_2, and t_4). Transitions t_1 and t_3 remain disabled.
- Therefore, the firing sequences and the marking matrices change.
- At the end of the process, the token is present only in place p_8 (output station 2).
- Initial Marking matrix $M_0 = (1, 2, 0, 0, 0, 0, 0, 0)^T$ (or in the latter case $(2, 1, 0, 0, 0, 0, 0, 0)^T$).
- Control Vector $\mu_k = (0, 1, 1, 1, 0, 0)^T$ (or in the latter case $(1, 1, 0, 1, 0, 0)^T$).
- The subsequent marking matrices are as follows
    - $M_1 = (0, 0, 0, 1, 1, 1, 0, 0)^T$ (or in the latter case $(0, 0, 1, 1, 0, 1, 0, 0)^T$).
    - Final marking $M_2 = (0, 0, 0, 0, 1, 0, 0, 1)^T$ (or in the latter case $(0, 0, 1, 0, 0, 0, 0, 1)^T$).
- This shows that even if there is an insufficiency of tokens in the input place, there is always a token that is present in the output station 2 (p_8).
- This will be the case even in the case where there is only one token present in both input places p_1 and p_2.
- This process can insure that production is maintained in a parallel system even if there is a shortage or raw material in the input station of an assembly line.

Prioritizing transitions are modelled using **2.4 Inhibitors** . They can ensure firing occurs in some sequence in the Petri Net.

6. Colored Net

In colored Petri Nets, tokens are assigned specific colors. The Colored Petri Net is defined as: PN = (P, T, I, O, C, M_0) [4], where C: {C_1, C_2, …} a set of colors.

Colored Petri Nets can be used to create models of systems before the systems are actually built. They can be used to check the feasibility of the system or analyze the performance of the system by simulating the firing of the system. In systems where there are a number of different types of resources travelling along the process (concurrent firing), colored Petri Nets can be used to track a particular type of resource. They prove to be an effective modelling and simulation tool.

7. Conclusion

The fundamental concepts behind Petri Nets and their properties were discussed. An automated industrial layout was considered and a Petri Net model was created for the system. The properties of the subsequent Petri Net were defined and analyzed in this project. The matrix representations were used to predict the future markings of the model. Timed Petri Nets and Priority Petri Nets proved to be essential in describing and analyzing the system in more detail. Petri Nets also provide other analysis techniques by drawing the convertibility tree [5]. Petri Nets can also be used for failure analysis by utilizing their concepts to representing a fault tree [7]. Petri Nets prove to be a useful tool to model a range of discrete systems.

8. References

[1] R. Zurawski and M. Zhou, "Petri nets and industrial applications: A tutorial," *IEEE Trans. Ind. Electron.*, vol. 41, no. 6, pp. 567–583, Dec. 1994.
[2] W.M.P. van der Aalst, "The Application of Petri Nets to Workflow Management."
[3] Azadeh Farzan, P.Madhusudan ,"Casual Dataflow Analysis for Concurrent Programs."
[4] Buket Yilmaz, "Application of Petri Nets.".
[5] T.Murata,"Petri nets: Properties, analysis and applications - Proceedings of the IEEE."
[6] "Properties of Petri Nets -Discrete, Continuous, and Hybrid Petri Nets.".
[7] T. S.Liu & S.B.Chiou, "The application of Petri Nets to Failure Analysis."