

PETRI NETS - PRESENTATION 1

...



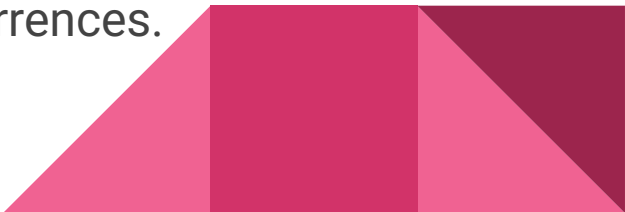
Introduction & Structure of Petri nets

What is Petri net?

A Petri net, also known as a place/transition net, is one of several mathematical modeling languages for the description of distributed systems and represents information flow.

This is used to analyze the flow of information and control in systems, specially systems that can show asynchronous and concurrent behaviour.

The major use of Petri nets has been the modeling of systems of events in which it is possible for some events to occur concurrently but there are constraints on the concurrence, precedence, or frequency of these occurrences.



About Petri net Model

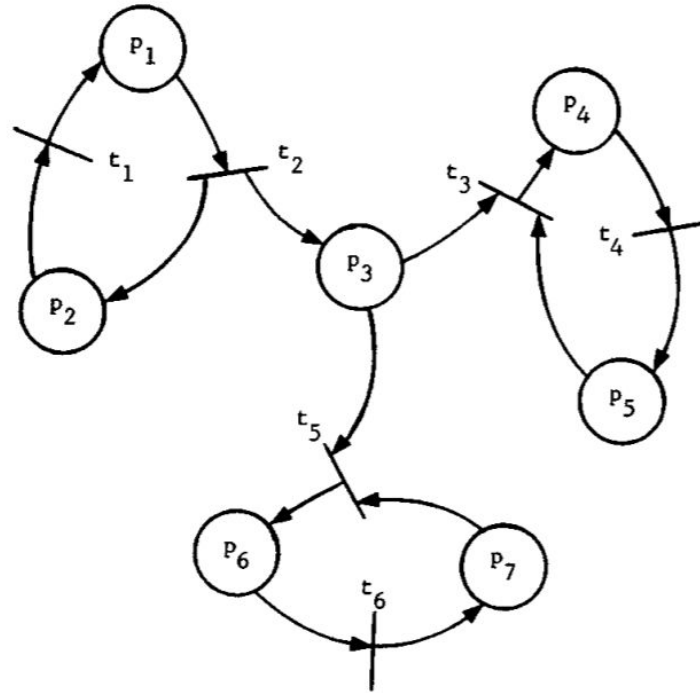
The Petri net graph models the static properties of a system, much as a flowchart represents the static properties of a computer program.

The graph contains two types of nodes:

- 1) **circles (called places)** and
- 2) **bars (called transitions).**

These nodes, places and transitions, are connected by directed arcs from places to transitions and from transitions to places. If an arc is directed from node i to node j (either from a place to a transition or a transition to a place), then i is an input to j , and j is an output of i .





In the above figure, for example, place p_1 is an input to transition t_2 , while places p_2 and p_3 are outputs of transition t_2 .

Dynamic property of Petri net Model

These dynamic properties come into play when the model gets executed.

The execution of a computer program represented by a flowchart is exhibited by placing a marker on the flowchart to mark the instruction being executed, and that as the execution progresses, the marker moves around the flowchart.

Similarly, the execution of a Petri net is controlled by the position and movement of **markers (called tokens)** in the Petri net. Tokens, indicated by black dots, reside in the circles representing the places of the net.



Dynamism continued

Tokens are moved by the firing of the transitions of the net. A transition must be enabled in order to fire.

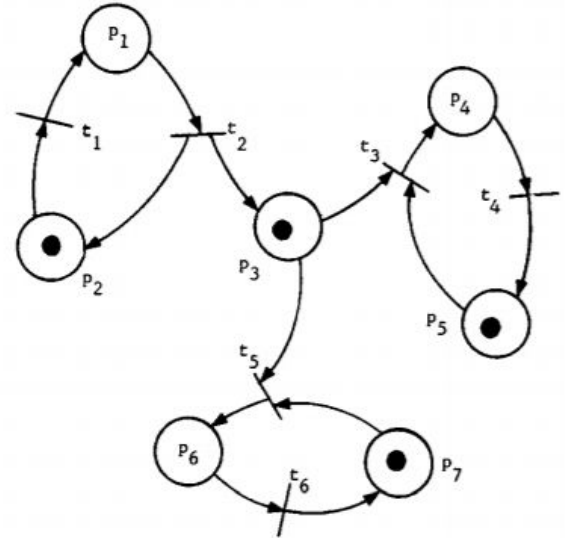
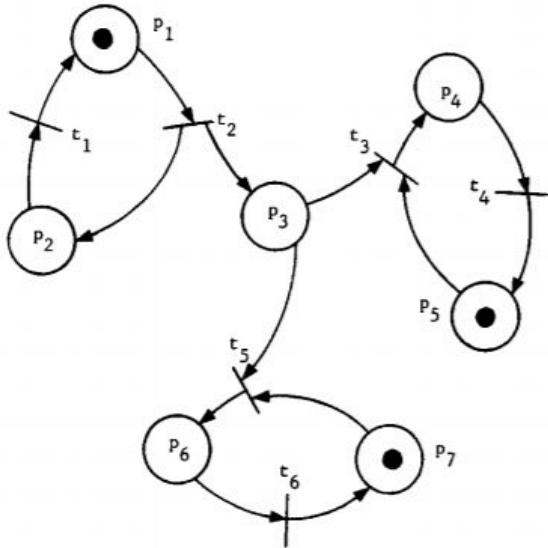
(A transition is enabled when all of its input places have a token in them.)

The transition fires by removing the enabling tokens from their input places and generating new tokens which are deposited in the output places of the transition.



Example

The transition t_2 is enabled since it has a token in its input place (P_1)" Transition t_3 , on the other hand, is not enabled since one of its inputs (P_3) does not have a token. When t_2 fires :




Markings

In one of the previous examples, the token in p_1 was removed and tokens were added to p_2 and p_3 .

The distribution of tokens in a marked Petri net defines the state of the net and is called its **marking**.

The marking may change as a result of the firing of transitions. In different markings, different transitions may be enabled.

Other markings may then be reached since transition firings may continue as long as there is an enabled transition.



In Conflict

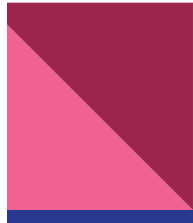
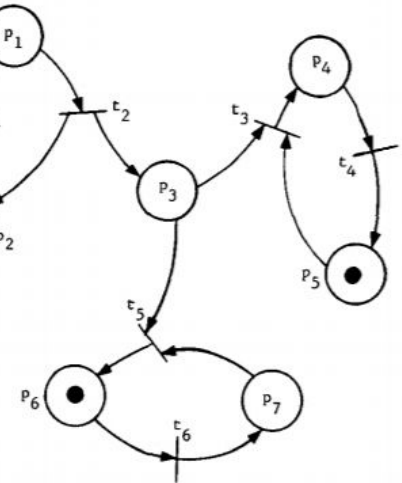
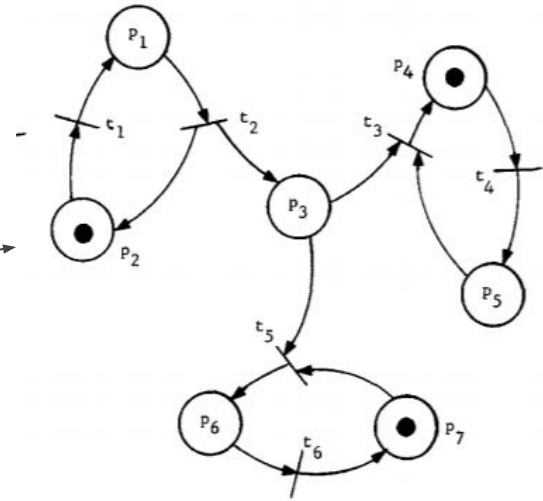
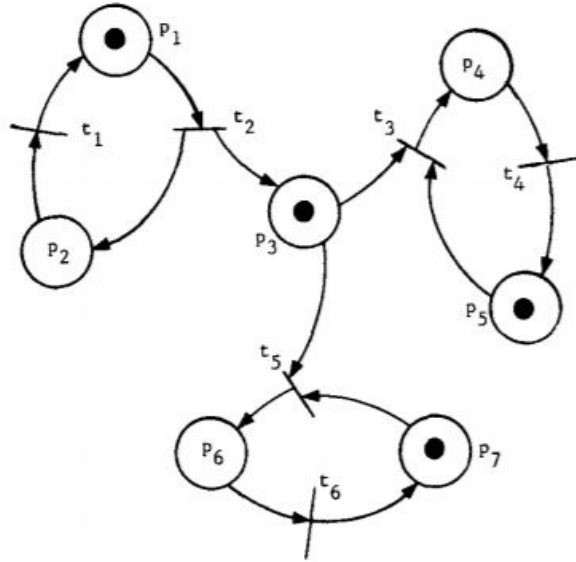
If there are two possible options, and firing either of transitions, disables the other; they are said to be in conflict.

In the following slide, t_3 and t_5 are in conflict.



Here t_3 and t_5 are enabled.

Firing either one of them disables the other.



History of Petri nets

The theory of Petri nets has developed from the work of **Carl Adam Petri, A. W. Holt, Jack Dennis**, and many others. Petri nets originated in the early work of Petri, in Germany, who in his thesis, developed a new model of information flow in systems.

This model was based on the concepts of asynchronous and concurrent operation by the parts of a system and the realization that relationships between the parts could be represented by a graph, or net.



Structure of Petri nets

Relationship between places and transitions:

Two functions connecting transitions to places: **I, the input function**, and **O, the output function**. The input function I defines, for each transition t , the set of input places for the transition $I(t)$. The output function O defines, for each transition t_j , the set of output places for the transition $O(t_j)$.

A Petri net C is defined as the four-tuple $C = (P, T, I, O)$.




Petri net graph

An arc is directed from a place p , to a transition t_j if the place is an input of the transition.

Similarly, an arc is directed from a transition t_j to a place p_t if the place is an output of the transition.

A Petri net graph is a directed graph since the arcs are directed.

Its nodes can be partitioned into two sets (places and transitions) such that each arc is directed from an element of one set (place or transition) to an element of the other set (transition or place), it is a **bipartite directed graph**.



Example of a Petri net

$$C = (P, T, I, O)$$

$$P = \{p_1, p_2, p_3, p_4, p_5\}$$

$$T = \{t_1, t_2, t_3, t_4\}$$

$$I(t_1) = \{p_1\}$$

$$O(t_1) = \{p_2, p_3, p_5\}$$

$$I(t_2) = \{p_2, p_3, p_5\}$$

$$O(t_2) = \{p_5\}$$

$$I(t_3) = \{p_3\}$$

$$O(t_3) = \{p_4\}$$

$$I(t_4) = \{p_4\}$$

$$O(t_4) = \{p_2, p_3\}$$

Markings

A marking (μ) of a Petri net is an assignment of tokens to the places in that net.

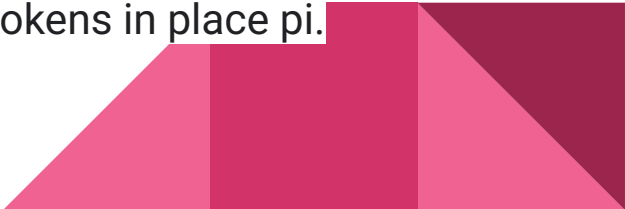
The number and position of tokens in a net may change during its execution. The vector

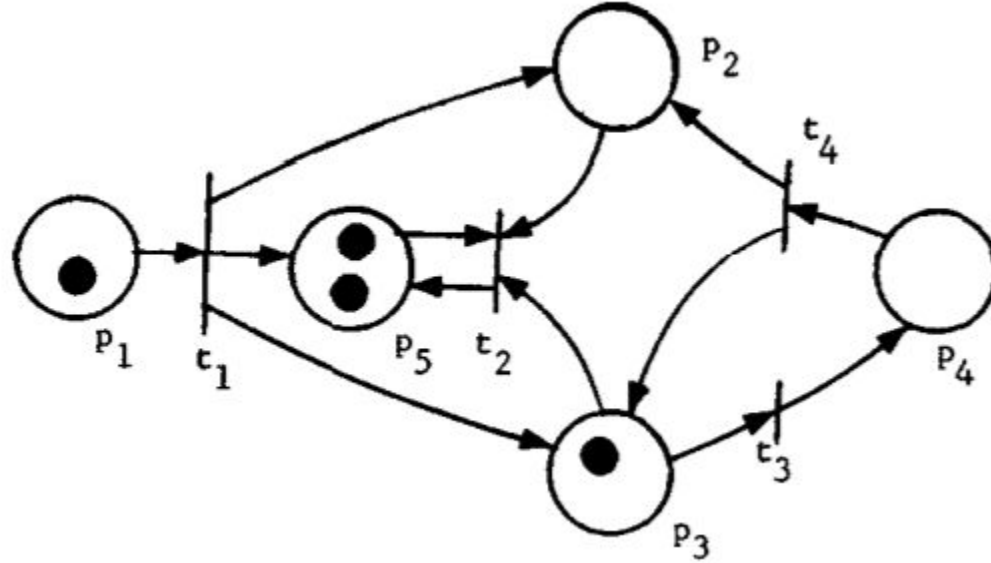
$\mu = (\mu_1, \mu_2, \dots, \mu_n)$ gives, for each place in the Petri net, the number of tokens in that place.

The number of tokens in place p_i is μ_i , $i=1, 2, \dots, n$.

We may also define a marking function $\mu: P \rightarrow \mathbb{N}$ from the set of places to the natural numbers, $\mathbb{N} = \{0, 1, 2, \dots\}$.

This allows us to use the notation $\mu(p_i)$ to specify the number of tokens in place p_i .
For a marking μ , $\mu(p_i) = \mu_i$.





$\mu = (1,0,1,0,2)$, Thus with a marking becomes the marked Petri net, $M = (P, T, I, O, \mu)$.

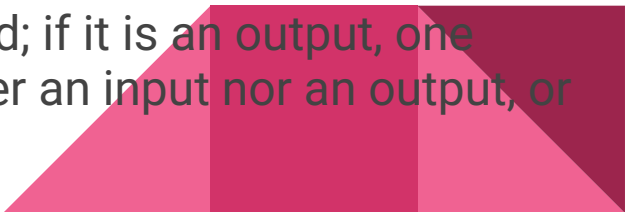
Execution Rules for Marked Petri Nets

A Petri net executes by firing transitions. A transition may fire if it is enabled. A transition is enabled if **each of its input places has at least one token in it**. We saw this in previous slides.

Extra tokens in a place are not affected by firing transition although they may enable additional firings of transition later.

Since only enabled transitions may fire, the number of tokens in each place always remains nonnegative when a transition is fired. Firing a transition can never remove tokens that are not there: if any one of the input places of a transition contains no tokens, then the transition cannot fire.

If a place is an input to the transition, one token is removed; if it is an output, one token is added. No net change occurs if the place is neither an input nor an output, or is both an input and an output.



The State Space of a Petri Net

The state of a Petri net is defined by its marking.

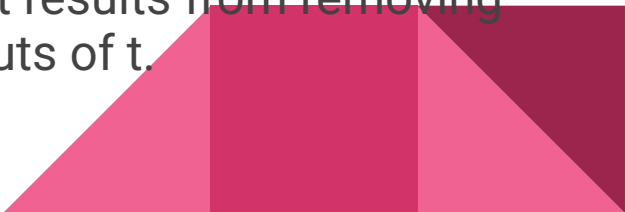
The state space of a Petri net with n places is the set of all markings, i.e., \mathbb{N}^n .

The change in state caused by firing a transition is defined by a partial function ∂ , called the next-state function.

Let's say there is a transition t .

Since t can fire only if it is enabled, $\partial(\mu, t)$ is undefined if t is not enabled in marking μ .

If t is enabled, then $\partial(\mu, t) = \mu'$, where μ' is the marking that results from removing tokens from the inputs of t and adding tokens to the outputs of t .



Two sequences result from the execution of the Petri net:

- 1) a sequence of markings ($\mu_0, \mu_1, \mu_2, \dots$)
- 2) sequence of transitions ($t_{j0}, t_{j1}, t_{j2}, \dots$)

Such that $\partial(\mu_k, t_{jk}) = \mu_{k+1} \quad \dots \quad \text{for } k = 0, 1, 2, \dots$

Given the transition sequence and μ_0 , we can easily derive the marking sequence for the execution of the Petri net and, except for a few degenerate nets, **given the marking sequence we can derive the transition sequence**. Both of these sequences thus provide a record of the execution of the net.



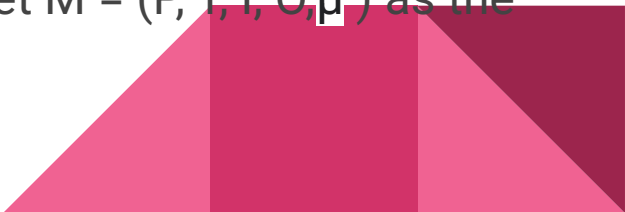
The Reachability Set of a Petri Net

From a marking μ , a set of transition firings is possible. The result of firing a transition in a marking μ is a new marking μ' .

We say that μ' is **immediately reachable** from μ if we can fire some enabled transition in the marking μ resulting in the marking μ' .

A marking μ' is **reachable** from μ if it is immediately reachable from μ or is reachable from any marking which is immediately reachable from μ .

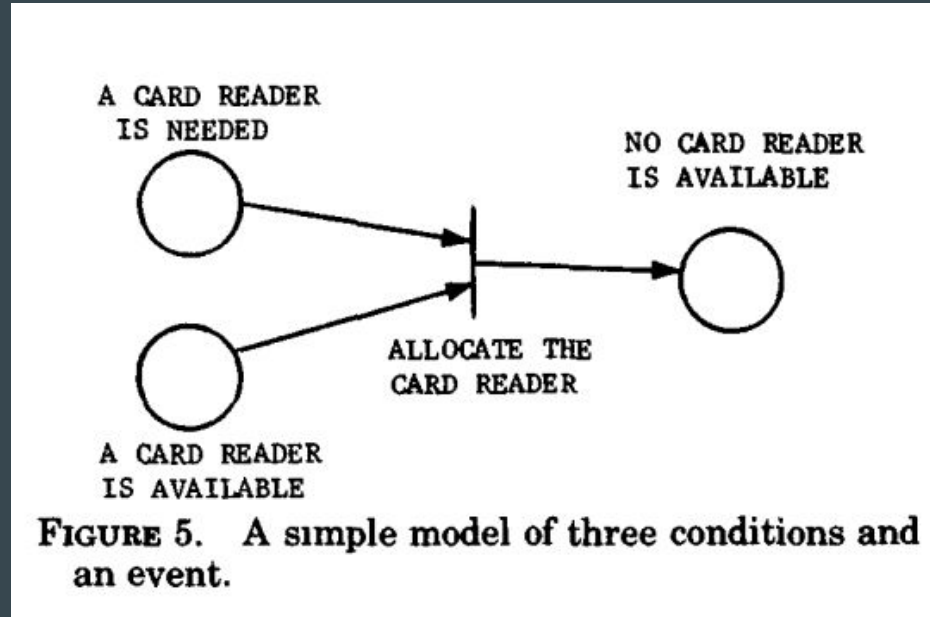
Definition - The reachability set $R(M)$ for a marked Petri net $M = (P, T, I, O, \mu)$ as the set of all markings which can be reached from μ .



Modeling with Petri Nets

...

Modeling a simple resource problem with a Petri Net



Mathematical properties of Petri nets useful for modeling

- Petri nets were designed to model a class of problems, where there are discrete and concurrent events.
- Transitions are instantaneous and atomic (technical term: *primitive events*)
- Two kinds of objects and the relationship between them:
 - Conditions (Places/Circles)
 - Events (Transitions/Bars)
- Many kinds of problems can be modeled via Petri Nets:
 - Software problems (Critical section problem, concurrency)
 - Resource allocation
 - Computational Biology
 - Hardware modeling
 - Workflow management

Modeling a processor with a task queue

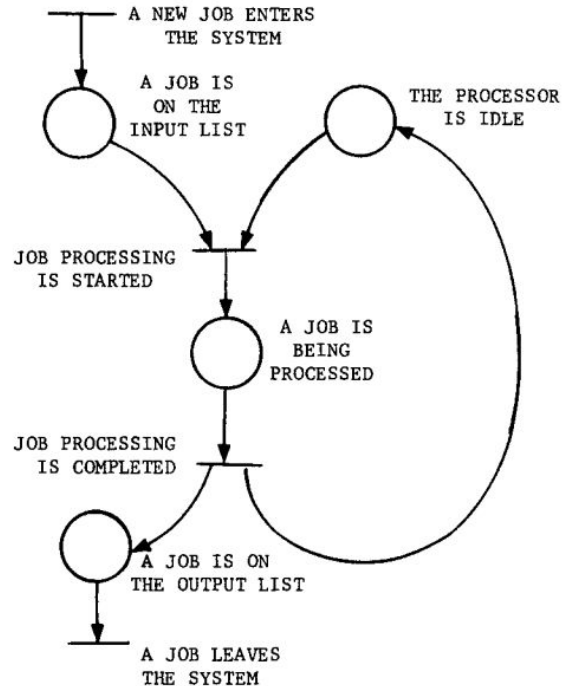


FIGURE 6 Modeling of a simple computer system.

- We start by satisfying initial conditions:
 - Processor is idle
- Trigger event *a new job enters the system*
- Synchronously, processor will process jobs one by one.

Time and concurrency in Petri Nets

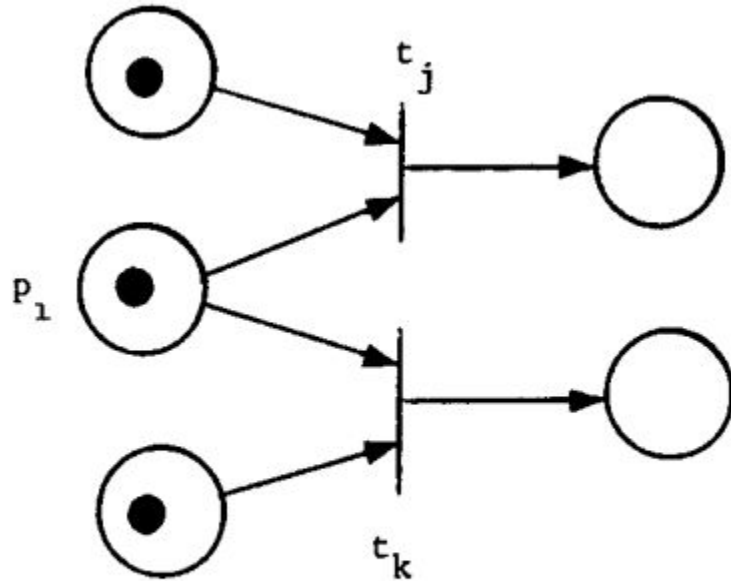


FIGURE 9. Illustration of conflicting transitions
Transitions t_j and t_k conflict since the firing of one
will disable the other.

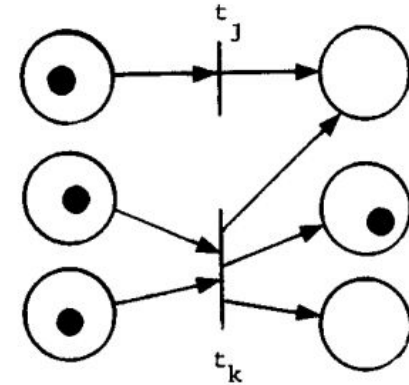


FIGURE 8. Modeling of "simultaneous" events
which may occur in either order.

Modularity in Petri Nets

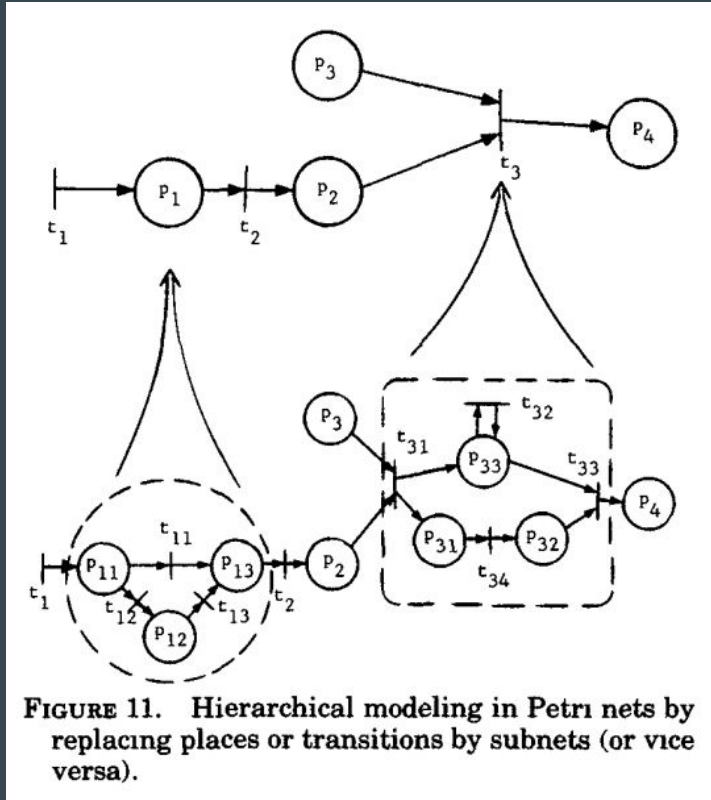
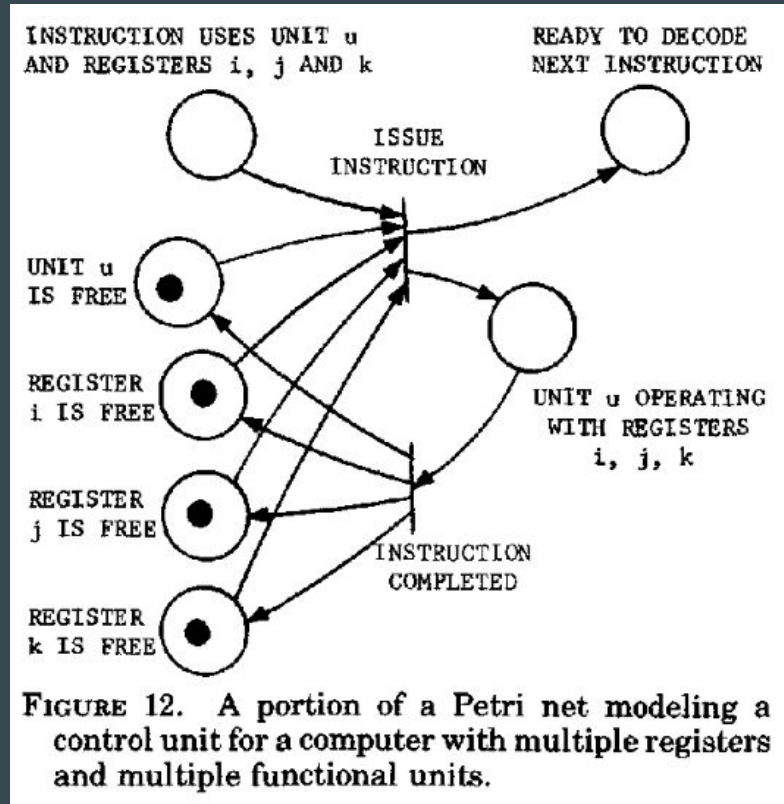


FIGURE 11. Hierarchical modeling in Petri nets by replacing places or transitions by subnets (or vice versa).

Hardware resource management modeled via a Petri Net



Mutual Exclusion in Petri Nets

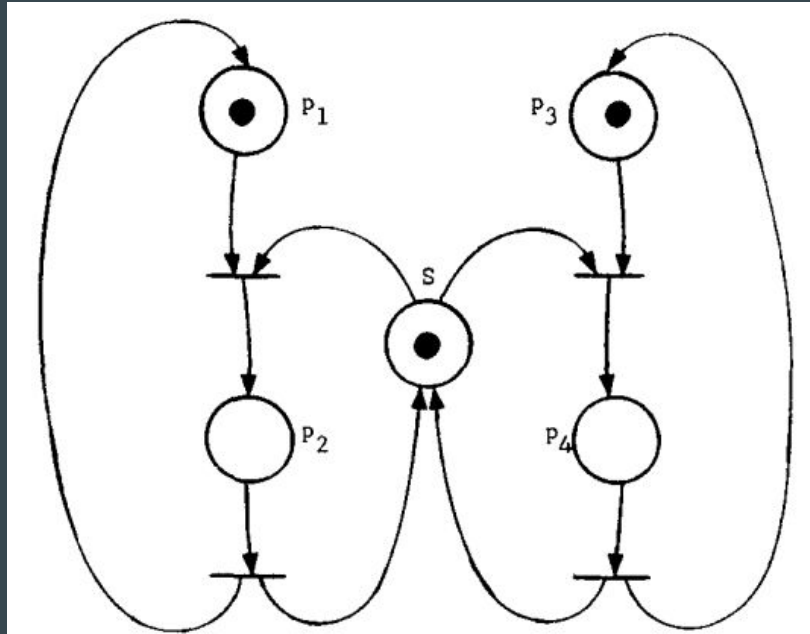
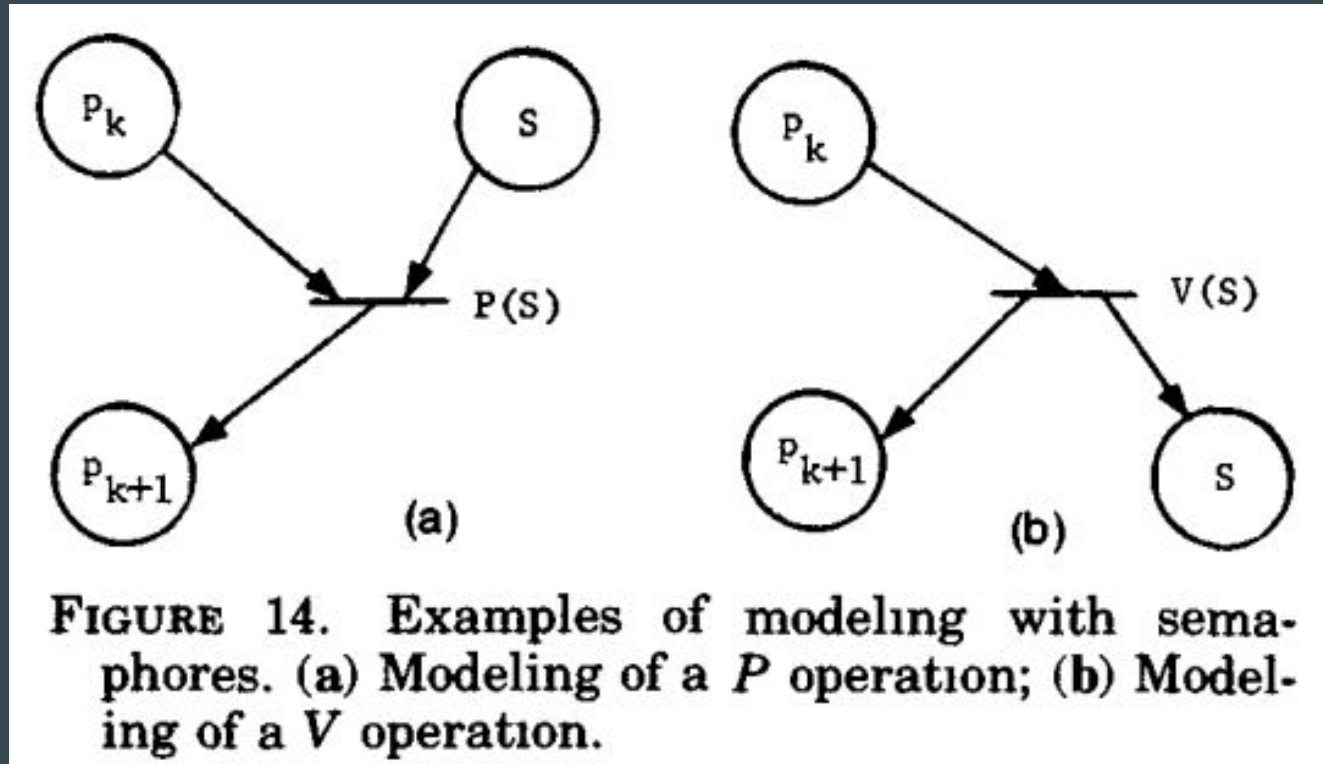


FIGURE 15. A Petri net model of a *P/V* solution to the mutual exclusion problem

To model a mutex situation, we make sure that the state where both P_2 and P_4 carry tokens are not reachable.

This can be proven in the given Petri net.

Semaphores in Petri Nets



ANALYSIS OF PETRI NETS

Petri nets so far and what next?

- So far , the only use of petri nets we have seen is in modelling of asynchronous and concurrent systems
- Now, we will shift our focus on how to analyse valuable properties of petri nets and thereby, the properties of the system modelled by the petri net
- Petri nets also find their use in design of concurrent systems using proper methods for design and implementation of petri nets

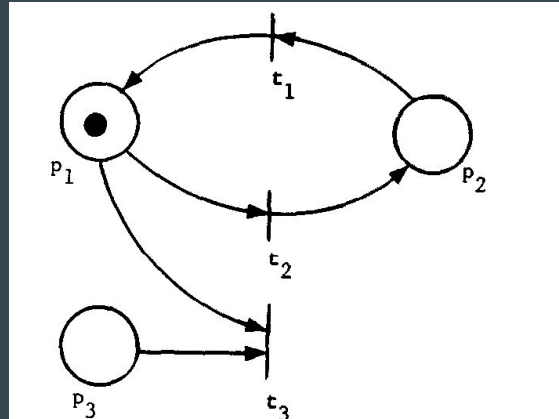
Important questions/properties for analysis

- Safe net - Atmost one token is present in each place of the petri -net. (Avoid using multiple tokens unnecessarily)
- Boundedness - Depicts the maximum number of tokens that can be present in each place of petri net. (K-bounded petri nets).Bound on the number of tokens is important in cases like implementation on hardware(capacity)
- Conservation of tokens - Total number of tokens in the petri net is constant for conservative petri nets.

In other words, each fireable transition has equal number of input and output places.

Important questions/properties for analysis

- Dead transition - Cannot be enabled by any sequence of firing of transitions starting from the present marking.
- Potentially fireable transition - Can be enabled eventually by a sequence of transition firings starting from the present marking.
- Live transition - For all the markings reachable from the present marking, the transition is potentially fireable.



Important questions/properties for analysis

- Liveness - This property is often analysed when we are aiming to model operating systems. eg:- Dead Transition might indicate potential deadlock

There are four forms of liveness definitions for a petri net with given initial marking μ and transition t :-

1. $\delta(\mu', t)$ is defined for some μ' in $R(M)$
2. For every $n > 0$, there exists a sequence of transitions σ for which t occurs at least n times in σ and $\delta(\mu, \sigma)$ is defined
3. There exists an infinite sequence of transitions σ for which t occurs infinite times in σ and $\delta(\mu, \sigma)$ is defined
4. t is a live transition

Important questions/properties for analysis

- Reachability problem - Given a petri net M with an initial marking μ , determine whether the given marking μ' is part of $R(M)$. (The special case of set reachability problem)

Reachability problem is particularly important as a lot of important questions regarding correctness and analysis of systems modelled by petri nets can be reduced to/are equivalent to this problem. Eg:-

Liveness problem - Determine if each transition is live or not

Mutual exclusion problem - Only one process allowed to access a given resource at a given instance of time.

Techniques/Approaches for Analysis

- The most important and basic technique used in analysis of petri nets is generating a finite depiction of its' reachability set $R(M)$ because many properties of petri nets are closely related to its reachability set.
- Reachability tree - A tree structure with the edges/arcs denoting the transitions that changes the petri net state on firing . The nodes represent the current marking of petri net.

Techniques/Approaches for Analysis

Let's see how this tree is created : -

- Label the first node/root with the initial marking
- For all the fireable transitions in the current marking μ , create a new arc directed from the current node to a new node
- This new node will have the marking - $\delta(\mu, t_i)$ as its label and the new arc will have t_i as its label
- Repeat the above process for all new nodes

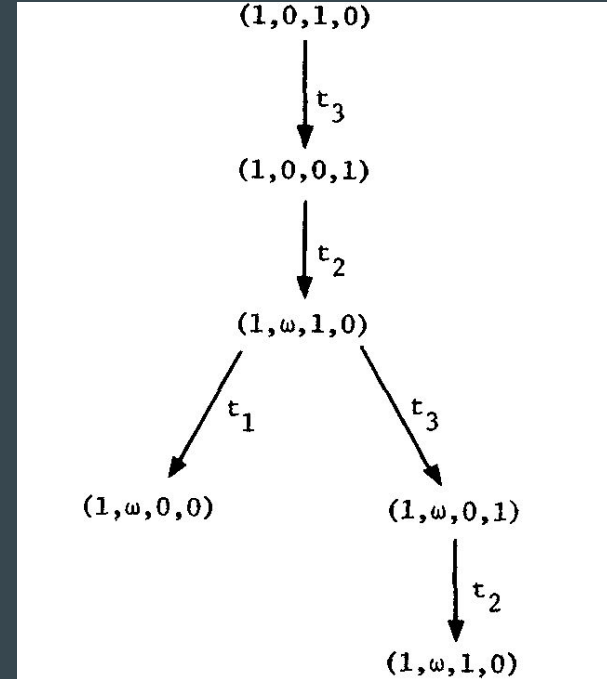
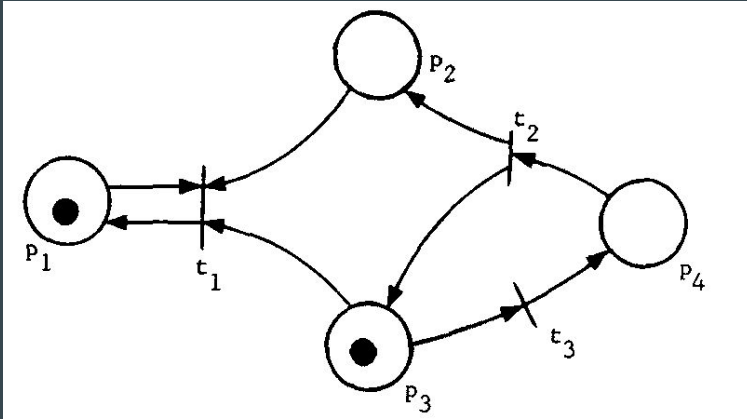
Techniques/Approaches for Analysis

Two important steps to ensure that the reachability tree is finite :-

- If the newly created node is labelled with a marking which matches a previously existing marking on the path from the root to this new node , then the new node is a terminal node .
- If the newly created node is labelled with a marking which is greater than a previously existing marking on the path from the root to this new node, then replace the strictly greater numbers/components in the new node marking by ω .
- ω represents an arbitrarily large value with properties like $\omega \pm a = \omega$, $a < \omega$ for all $a > 0$

Techniques/Approaches for Analysis

Example:-



Analysis using reachability tree

- Both K -bounded petri nets and conservative petri nets have a finite reachability set and ofcourse, a finite reachability tree
- Presence of ω in any node of the tree implies that the reachability set is infinite, the petri net is unbounded and the petri net is non conservative
- Absence of ω in any node of the tree implies that the reachability set is finite and the petri net is bounded . The conservative property is checked via inspection.
- Coverability problem - Check if there exists a marking μ' in $R(M)$ which is greater than or equal to the initial marking μ of a petri net M
- Most analysis problems are solved via inspection of the tree structure including the coverability problem

Reachability Problem

- This problem is equivalent to many problems like:
 - zero reachability problem - Check if an all zero marking is reachable
 - subset reachability problem - Check if there exists a reachable marking with a given subset of places matching that of a given initial marking μ
 - liveness problem - Check if all transitions are live
- There exists an algorithm, which is quite complex in implementation and cost, to solve the reachability problem and hence, the other equivalent problems.

Unsolvable Problems and Complexity

- Few problems related to petri nets are not solvable or are undecidable like:
Subset problem : Check if reachability set of one petri net is subset of another
Equality problem : Check if reachability set of one petri net is equal to another
- Complexity of reachability problem is said to have an exponential time lower bound and exponential space lower bound similar to the coverability problem [Lipton , Rackoff]
- Thus, even though Petri nets are instrumental in modelling and analysis of systems, solving even the basic problems for analysis might be quite expensive

Extensions

We need modeling power and decision power for a successful model.

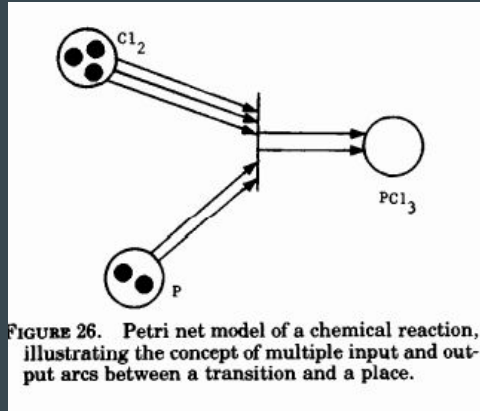
Petri nets have more modelling power than FSMs but it is still difficult or impossible to model some events or conditions by Petri nets

Therefore, we need to extend petri nets to increase their modeling power

Petri nets seem to be just below turing machines in modeling power

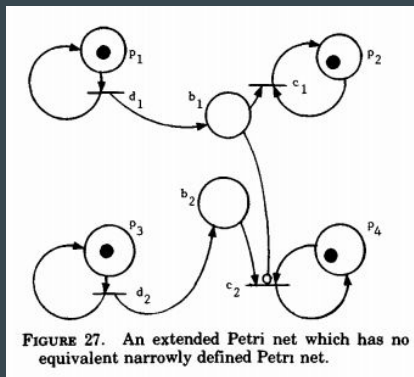
Generalized Petri nets

- A place may contribute or receive more than one token from the firing of a transition.
- Modeled by allowing multiple arcs between transitions and places, signifying the number of tokens needed.
- These nets are equivalent to ordinary Petri nets.



Zero-testing

- Arcs from a place to a transition which allow the transition to fire only if the place has zero tokens in it.
- These arcs are called inhibitor arcs.
- In response to difficulties in the modeling of priority systems with Petri nets.



- It has been shown that such Petri nets have the modeling power of a Turing machine
- Therefore we can show that many of the decidability problems are unsolvable for such petri nets.

Other Extensions

Many other extensions of Petri nets are equivalent to Petri nets with inhibitor arcs:

- Introduction of priorities between transitions:
 - Here, transitions with lower priorities are not allowed to fire if a transition of higher priority is fireable
- Time bounds on transition firings
- Constraint sets that prohibit tokens residing simultaneously in two places

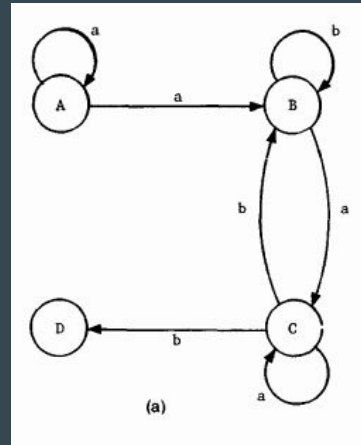
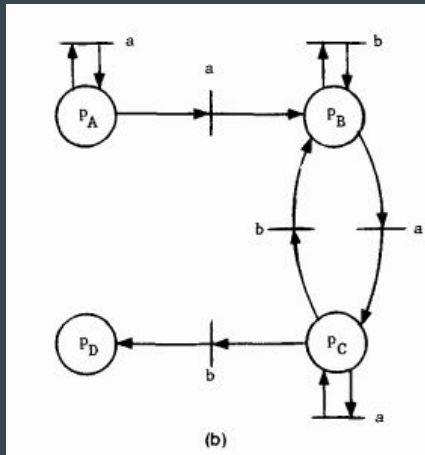
In terms of modeling power Petri nets seem to be just below Turing machines, so any significant extension results in Turing-machine equivalence.

Subclasses

- The limitations on the modeling power of Petri nets relative to Turing machines are balanced out by a compensating increase in decision power.
- For Petri nets many decision problems are equivalent to the reachability problem, which is decidable
- However, the reachability problem is very difficult to solve.
- Thus, from a practical point of view, Petri nets may be too powerful to be analyzed.

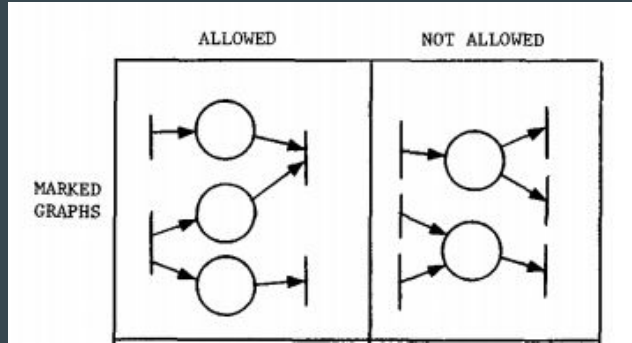
State Machines

- Each transition has exactly one input and one output.
- These nets are obviously conservative, which means they have a finite reachability set and hence are finite-state.
- In fact, they are exactly the class of finite-state machines.



Marked Graphs

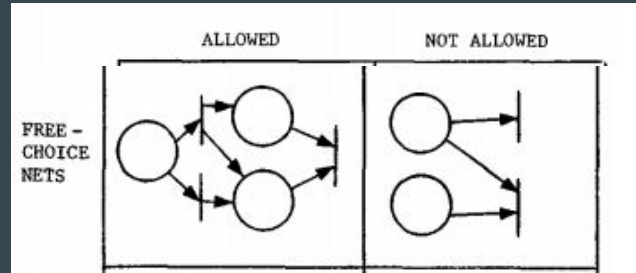
- A marked graph is a Petri net in which each place has exactly one input transition and one output transition.



- Marked graphs have high decision power.
- But, they have limited modelling power as they are only able to model systems without branching
- This solves the problem of conflicts which are difficult to analyse.

Free-choice Petri nets

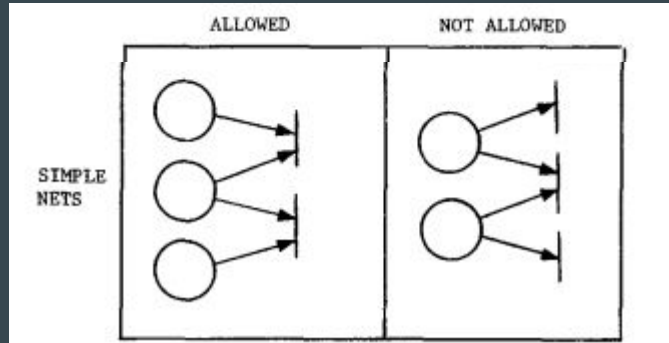
- Each arc from a place is either the unique output of the place, or the unique input to a transition.



- If there is a token in a place, then
 - The token will remain in that place until its unique output transition fires
 - If there are multiple outputs for the place, then there is a free choice as to which of the transitions is fired.
- Liveness and safeness are decidable and we have necessary and sufficient conditions for these properties.

Other Subclasses:

- Simple petri nets: Every transition has at most one shared input place

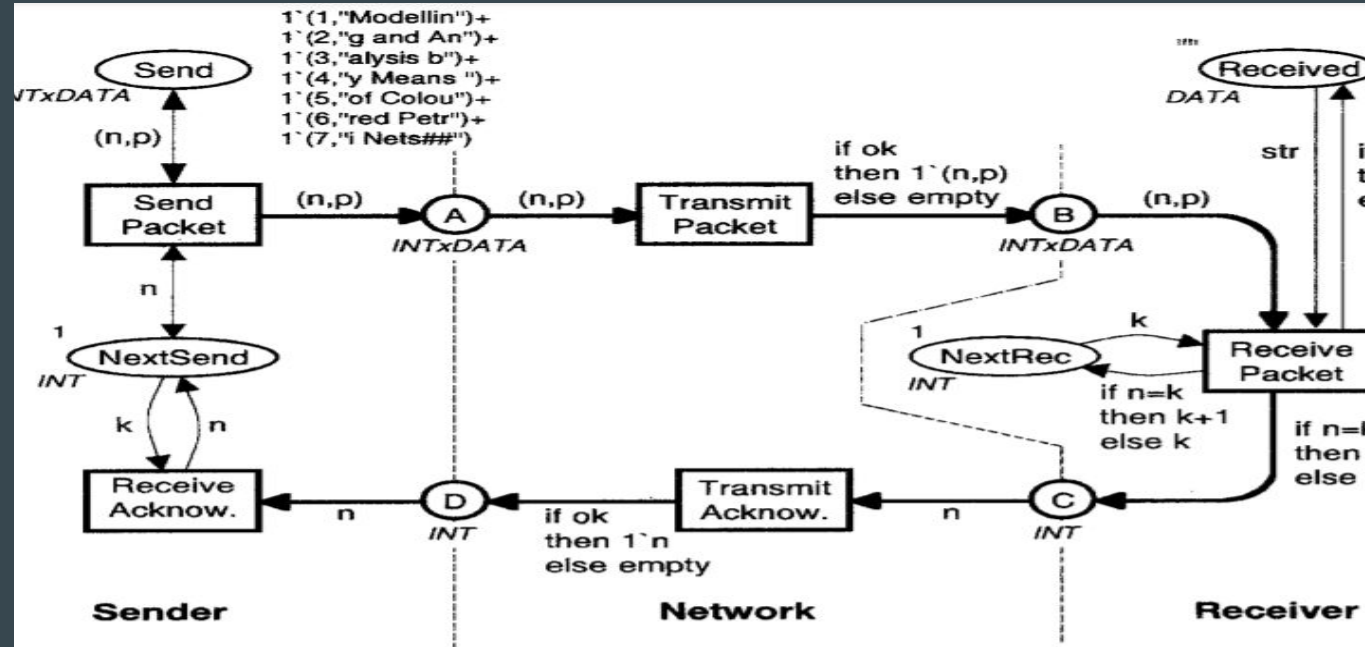


- Persistent Petri nets: A Petri net is persistent if for all transitions t_i and t_j and any reachable marking M , if t_i and t_j are enabled at reachable marking, firing of one cannot disable the other.
- Conflict free Petri nets: If every place that is an input of more than one transition is on a self loop with each transition.
- Conflict free Petri net \Leftrightarrow It is persistent for all initial markings

Colored Petri Nets

Definition 4.2. A **non-hierarchical Coloured Petri Net** is a nine-tuple $CPN = (P, T, A, \Sigma, V, C, G, E, I)$, where:

1. P is a finite set of **places**.
2. T is a finite set of **transitions** T such that $P \cap T = \emptyset$.
3. $A \subseteq P \times T \cup T \times P$ is a set of directed **arcs**.
4. Σ is a finite set of non-empty **colour sets**.
5. V is a finite set of **typed variables** such that $Type[v] \in \Sigma$ for all variables $v \in V$.
6. $C : P \rightarrow \Sigma$ is a **colour set function** that assigns a colour set to each place.
7. $G : T \rightarrow EXPR_V$ is a **guard function** that assigns a guard to each transition t such that $Type[G(t)] = Bool$.
8. $E : A \rightarrow EXPR_V$ is an **arc expression function** that assigns an arc expression to each arc a such that $Type[E(a)] = C(p)_{MS}$, where p is the place connected to the arc a .
9. $I : P \rightarrow EXPR_\emptyset$ is an **initialisation function** that assigns an initialisation expression to each place p such that $Type[I(p)] = C(p)_{MS}$.



Declarations:

```

type INT = integer;
type DATA = string;
type BOOL = boolean;
type INTxDATA = product INT * DATA;
var n, k : INT;
var p, str : DATA;
var ok : BOOL;

```