

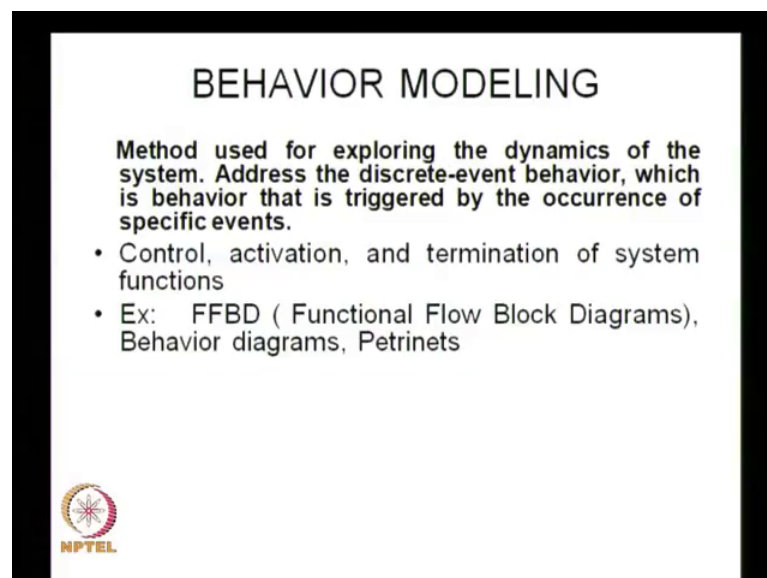
Principles of Engineering System Design
Dr. T.Asokan
Department of Engineering Design
Indian Institute of Technology, Madras

Lecture - 25
Behavior Modelling

Dear friends, welcome back. We have been discussing about the graphical modeling techniques used in system engineering. In the last 2 classes we discussed about the data modeling and process modeling. And we saw few methods of data modeling and process modeling. And we saw few examples also, how these methods can be used for designing of engineering systems, or how this modeling techniques can be used for modeling of the system; especially when the data flow, the relationships, the process relationships of process flow data, the relationship between various functions, and then the interaction between these functions.

Today we will discuss about the behavior modeling. Behavior modeling one of the modeling method we need to do. Because we have a data modeling, process modeling, and the third one is behavior modeling.

(Refer Slide Time: 01:07)



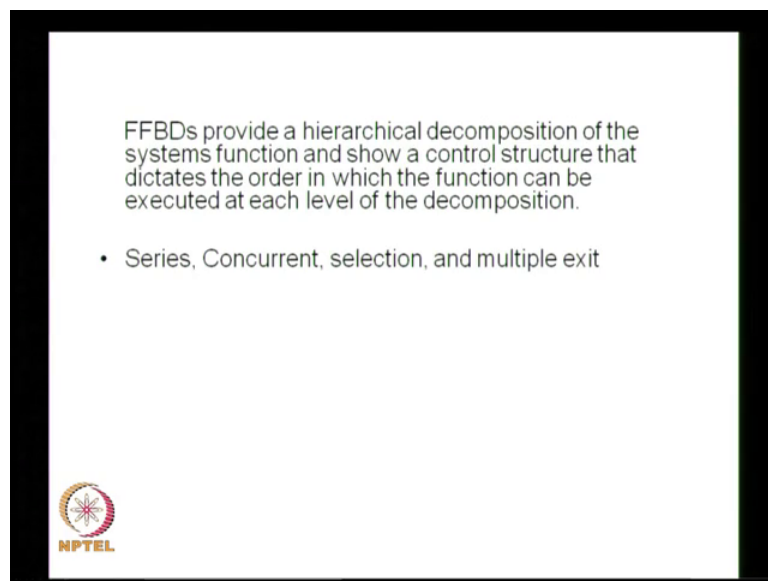
In behavior modeling, we use the method of behavior modeling for modeling the dynamics of the system. So, we want to see how the engineering system works or what is the dynamics of the engineering system, how the control actions are taking place, how

the particular incidences are triggered, or how the a particular incident stops or particular function stops and the next one function starts.

So, all these kinds of dynamics of the system can be a modelled using behavior modeling techniques. Here the control, activation and termination of system functions are basically modelled using the behavior model. So, not only the data and the process we want to model the control, the activation, and termination of system functions. And for this purpose, we have various methods. And one method is known as a function flow block diagram; that is FFBD, and we have other things like a behavior diagrams and petrinets.

So, we will see some of these methods, and then try to understand how this methods can be used for modeling the behavior of systems, or how the dynamics of the system can be modelled using behavior diagrams. So, we will take the FFBD first. That is the function flow block diagram. And then see how to use this function flow block diagram, and then model the behavior.

(Refer Slide Time: 02:30)



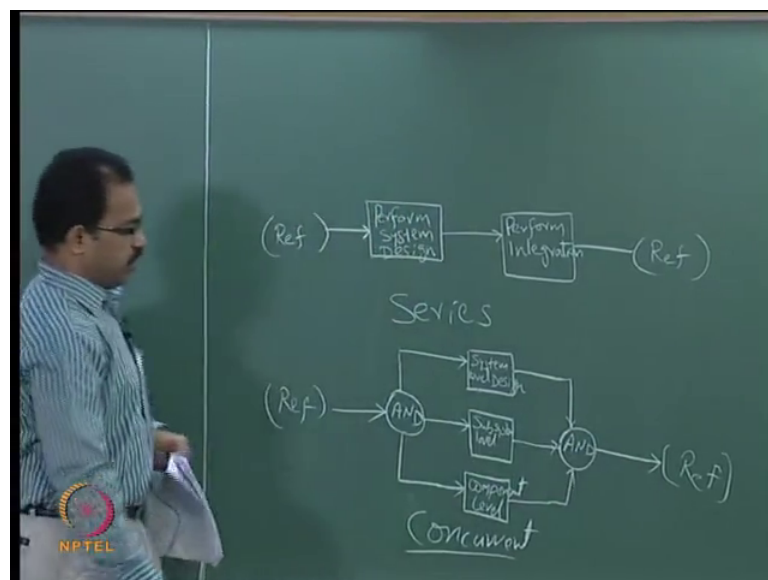
So, the FFBD's they provide a hierarchical decomposition of the system function, and show a control structure that detects the order in which the function can be executed at each level of the decomposition.

So, that is the importance of FFBD. We can actually show the control structure, that they actually dictate the order in which the functions are executed. And the various level of

decomposition. So, there are different ways of executing the function. They are basically classified into 4 categories. One is the series execution. The other one is the concurrent execution. And then the selection, and multiple exit. So, these are the ways in which the functions can be executed in a system.

So, we will see how do you actually model the this kind of behaviors that is a series behavior or concurrent behavior or exit and the selection and exit procedure. So, how do you actually model this using FFBD, we will take a few examples and then show the method by which we can do this. So, in series structure, what will having is as the name suggest the functions will actually execute in a serial way.

(Refer Slide Time: 03:38)



So, as we can see here there maybe different functions. So, we will start with the reference function; which may be an external function or a function of a sub system. So, from the that reference we will have the one function executed, and then it will go to the next function, that is executed and then it will go to the external system or the other subsystem. So, this is the a serial execution as the name suggests the functions will be executed in a serial way.

For example, you can see that if you take the system design activities. Then it is a perform system design activity. So, once we complete such system design activities, then only we go for the integration activity. So, perform integration. Again, it appears to be very simple case, because we are telling something which is very obvious, but this is the

way how we actually show the series execution of functions in the functional flow block diagram.

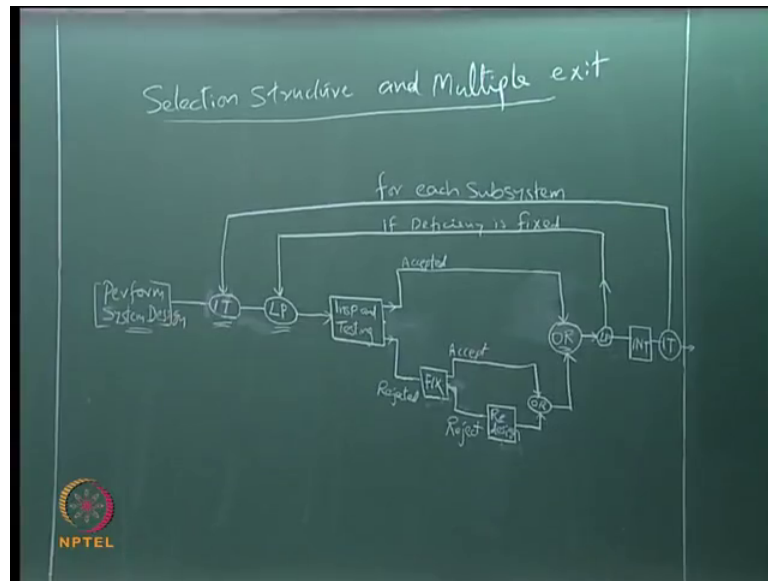
So, we will complete this function, and then go to the next function; that is the serial execution of the functions. And in the concurrents or sometimes it is known as parallel also, in concurrent execution will be having the same reference function, and then we will be having 2 functions executing the same time 2 or more function executing and it is shown by a block and block. And in this and block we will be connecting all the other functions, and here will be having the concurrent functions listed. And again, once it is completed, the next function can start only after the completion of all these functions.

All these processes or the functions. And then it will go to the reference function or the external function. This is the concurrent or parallel execution. So, here again you can have examples like a system level design, suppose if you take the same system engineering, energy system level design. Then you have the subsystem level design, subsystem level and then component level. So, we know that if you want to execute all this functions we need to have a this all need to be done concurrently or parallelly.

So, after that end of this design process only we can go for the testing or integration of the components. So, we need to do all this in a concurrent way or in a parallel way, and once we complete all this design then only go to the next function. That is the concurrent behavior of the functions. So, this need to be done parallelly or concurrently, and once all these are completed then only it will go out. So, this a and shows that all these 3 inputs should be there to for this to go to the next function. That is here the and represents all these needs should be there this and represents all these will go happen at the same time.

So, when this input is coming are the 3 output will go concurrently, and here all these 3 input should be there for the next function to execute. There is the concurrent behavior of the system. The next one is the selection structure and multiple exit.

(Refer Slide Time: 07:36)



In selection structure and multiple exit, it actually will be having series functions, it may be having concurrent functions. And it may be having some selection, that you can actually select one of the functions, and you can have multiple exit or a iterations also. So, here again we will take a case of system design.

So, if I take perform system design has a function, now I will be having many iterations of the system design or the system because there may be some kind of repetitions or represent this iterations by a block IT, and then there may be some loops through which it will be going and that is represented by the loop LP, I will explained how these are actually coming into the behavior, and once we complete this system design we need to have the testing.

So, that is a testing function. So, inspection and testing. And after the inspection testing, there are options like some of the components may pass the inspection, some of the components may fail the inspection. So, we need to select this 2 options, and one of the options need to be chosen from the multiple options available. So, what we do we have this 2 options here. So, one will go for the selected one that is passed one and the other one will go to the which are actually rejected or which is not passing the inspection. So, the cleared system will just go directly to the same main loop and again it will start from here to the next function.

So, will see what actually happened to this one. This is the accepted. So, we are selecting accepted one, and rejected or with some deviation from the required specifications. And here we can actually do a fixing of the problem. So, you fix the problem, and again send it for here. So, once you fix it, there are 2 options one is that it is acceptable. One is it cannot be accepted. So, the small fixing can be done here rejected components can be fixed here itself. And once it is fixed we can actually has a take it accepted or rejected.

Accept or reject one. And if it is rejected we need to actually have a redesign. So, this is redesign. So, the rejected one was to go redesign, and then it will come here. So now, we know that there may be a this rejected component may go through this or through this routes. And many of this routes is acceptable. So, this is an or block. So, we can actually anyone of this component comes here, it has to go to the next level. So, it need not be though there are parallel activities. So, need not wait for the both to happen. So, any one of this happens it can go to the next level. And then so, this is also will come here.

So now we have components coming from both from the inspection the accepted will come directly and the rejected one may go for redesign or may get accepted and it will come here, and it will go to the next level. So, here it is an or loop. So, any one of this will come here. So, we can see from the inspection there are 2 options. So, one of this option go directly. Another option will go through different processes and finally, come here to the this block. And then here will be having a looping, if there is a deficiency fixed, they it has to go to this loop again for the inspection. So, some of the components will come through this. So, there was a deficiency it was fixed, and it is coming over here. So, we need to make sure that if it is coming through this loop, then it has to go back again and then do that inspection and testing and see whether it is actually passing this test or not.

If it is passing this test again it will come through this, and once this is completed it will go to the next level where integration to next subsystem. This is integration this integration to next subsystem, and then it will be having that is one subsystem, then it will be having a nitration this is for the next subsystem design. So, this for next subsystem and then it will go out from here to the next function. So, this is a selection and multiple exit structure. So, this is the loop for if deficiency is fixed. So, if the deficiency is fixed, then a looping will be there. And this is for each subsystem this process will do

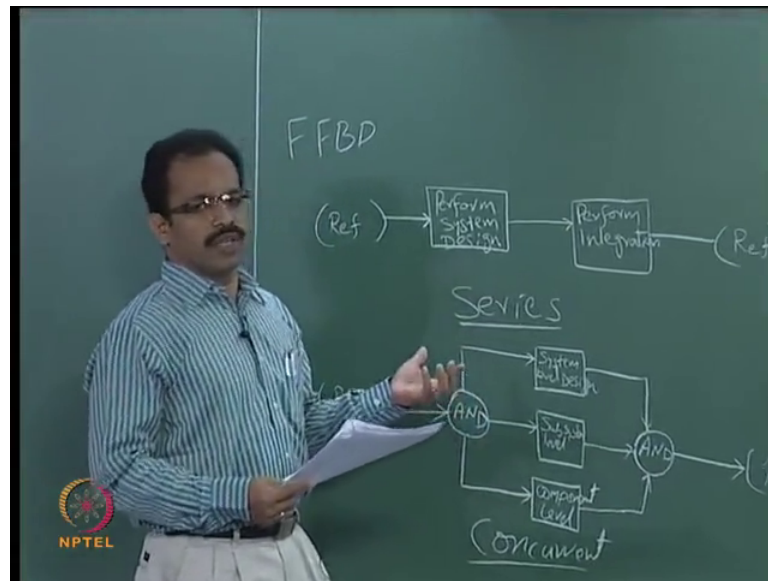
So, for each subsystem we need to complete the same procedure. That is why there is an iteration loop is an iteration and this is a looping because if something is fixed over here. There was a deficiency in one of the components or subsystems, and that will go through a another process where the fixing will be done, and then again it has to go for the testing. So, that is why it is come back. And then do inspection and testing and if it is accepted it will go out. And then come to the integration. And then again, we will be doing the same process for the other subsystems also.

So, we will be having many subsystem and the when system design is complete will be having many subsystems. So, for each subsystem will do the inspection and testing, and if it is accepted it will go out to this block it is not accepted it will go through this loop to fix the deficiency and if it is accepted it will go if it is not accepted it will go for redesign. And whenever there is coming from these loop these output is coming from these loop, then this will go for the inspection again and pass and once everything is passed through this it will go to for the integration. And once the integration of that subsystem is completed then this loop will again repeat because the next subsystem will be taken for inspection, and so on it will continue.

So, we can see here we are actually using many blocks here, an iteration, looping and then of course, the over and over also can be used. And in some cases, we may have to use and also. So, in this case there is no and block, but here we have a or block. So, one of these output will be coming out. If both the outputs are need a some situations, then we have to use the and block as shown here in this block. So, we are used or block there, but some cases we may have to use and block also.

So, this is the selection structure and multiple exit. So, in FFBDs that is in the functional flow block diagrams.

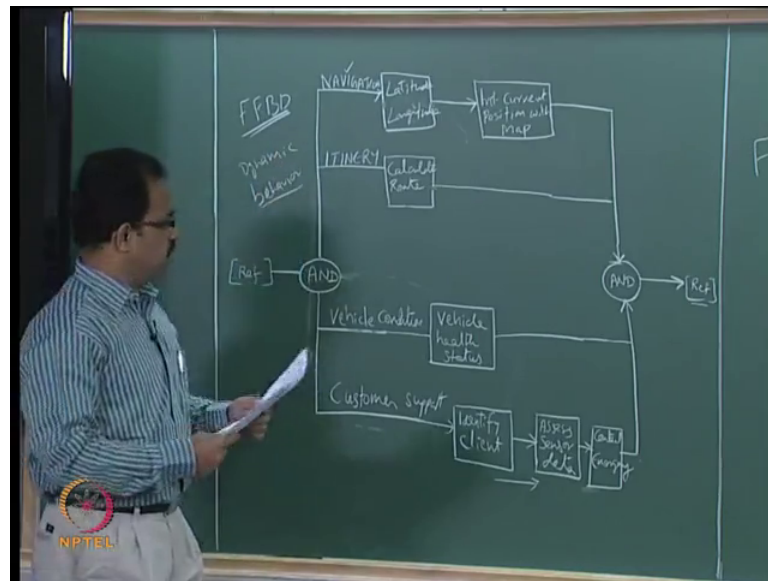
(Refer Slide Time: 15:38)



We have series execution, you have concurrent execution, and we have selection structure and multiplex. It also any of these methods or a combination of this can be used for representing the actual design of a system or actual behavior of a system. So, some subsystem we may have to model the behavior of the subsystem using this method. So, we can actually use the series concurrent or selection and multiple exit or a combination of these to represent the dynamics of the system.

Let me take an example, and then show how it is actually done. I will take an example of the auto link system, what we discussed in the previous class. So, in the auto link system, we will see how the functions can be represented, or how the dynamics can be represented using the FFBD.

(Refer Slide Time: 16:40)



So, will take a reference function or an external system or a previous subsystem, and then see what are the concurrent activities to be carried out. So, there are many concurrent activities to be carried out in the auto link system. So, we will see that there are parallel functions to be executed. So, this is the navigation activities, this is the itinerary of the passenger or the person who is using the system. And this is the vehicle condition function basically looking at the status of the vehicle customer support.

So, these are the 4 functions need to be executed parallelly in order to provide a precede service to the passenger or the person who is using auto link system. If we need to provide the required service like knowing the location or in an emergency providing the support, all this parallel function should be executed. So, what are these functions we will write down those functions here. Basically, we will be looking at the latitude and longitude calculation. Latitude longitudes, and then integrating the current position with the map. So, integrate current position because we get the latitude and longitude once we get this can actually integrate the current position with the map. That will help the passenger for the navigation or the driver for navigation. And that is the required for navigation purpose, and then here we have to calculate the display route. Because based on the input given by the user, he will be giving the itinerary where he wants to go, and once that information is obtained, the route has to be calculate using a database. And then that present latitude and longitude will be calculated and based on that map will be generated to show where he is at present.

And then vehicle condition need to be assessed, because if the vehicle is in good condition or not these to be assessed to help the user. So, vehicle condition or vehicle status or vehicle health status. So, vehicle health status need to be assessed, and then the customer support for. So, need to identify the client, who is actually using it whether he is authorized to use the system or not. So, identify the client assess that sensor data basically whether the are the sensors are working properly and whether GPS data is correct. So, this kind of information to be collected. So, as a sensor data, and then if needed contact emergency. Contact emergency services, that is contact emergency and this all these need to be executed concurrently in order to provide the service. So, that is the outputs.

So, here you can see in this case the auto link system a customer or a driver wants to get the some assistant from the auto link system for his navigation purpose. Or his emergency response whatever there situation is. So, in this case the auto link system should execute all this function in a concurrent way, then only he will be able to the system will be able to provide the necessary output. So, the functions need to be provided are the navigation assistance the itinery especially to calculate the route, the condition of the vehicle whether vehicle is in good condition, and then the customer support in various ways like the client is an authorized person. And whether the all the data systems are functioning properly and whether the emergency service can be provided.

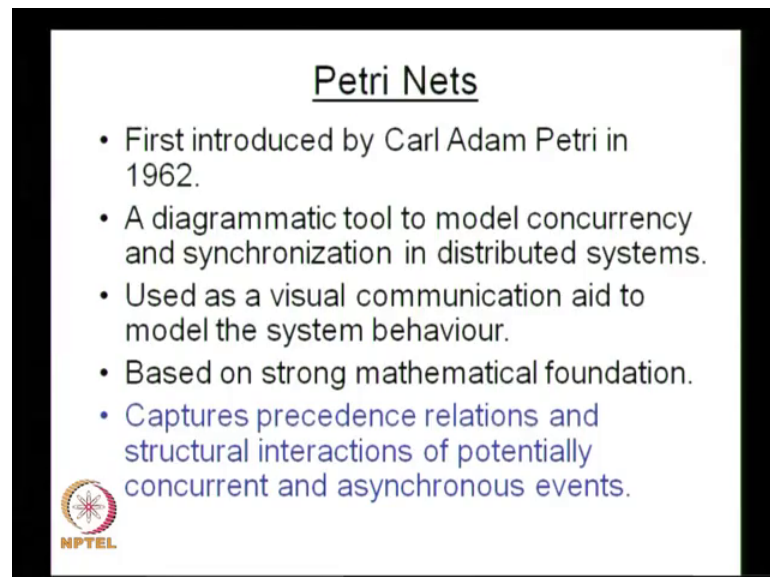
This need to be in a serial way. So, all this functions are executed in a serial way, as you can see here. Here also this has to be done in a serial way. So, a generate the latitude and longitude data, and then integrate that with the current position with the map. So, that will help the user to navigate himself. And then this is to calculate the routes if the user gives the his itinery the route can be calculated, and this can actually be once the route is calculated and this position is not that can actually given to the passenger or the user.

So, that actually we need these 2 for the output. Similarly, we need to know the vehicle condition and the other customer support services. So, all these are provided and once we have all this information then the an output can be given to the customer to use the facility. So, that is the way how we actually represents the behavior of the system using FFBD. So, we are using the FFBD that is a functional flow block diagram to represents

the so, dynamic behavior that is an example for the auto link system. And how do we use the function flow block diagram to represent the dynamics of auto link system.


This was an example for the function flow block diagram. And the next method is we have the method called petri net. So, petri net is also a method used for modelling the dynamics of the system. That is, how the input output can be controlled, how the concurrent activation as well as the status of different functions and how do we actually start a particular process, or what are the condition under which a process can be started. All this can be represented using petri nets.

(Refer Slide Time: 23:26)



Petri Nets

- First introduced by Carl Adam Petri in 1962.
- A diagrammatic tool to model concurrency and synchronization in distributed systems.
- Used as a visual communication aid to model the system behaviour.
- Based on strong mathematical foundation.
- Captures precedence relations and structural interactions of potentially concurrent and asynchronous events.



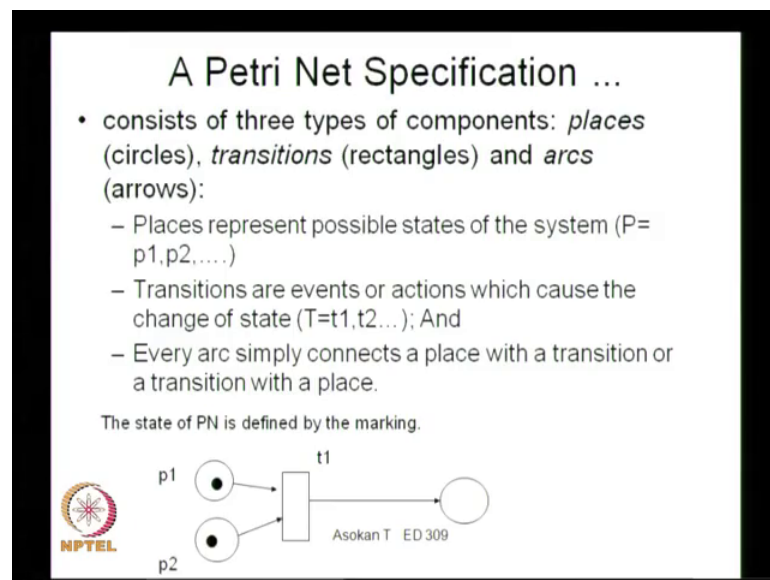
So, as we can see here the petri net was introduced by carl adam petri in 1962 which is a diagrammatic tool to model concurrency and synchronization in distributed system. So, we will show the concurrency of systems, but then in some cases we need concurrency as well as synchronization. For example, if you take the traffic light on the roads, we need to have some kind of a synchronization between the other lights; when you have a junction will be having multiple traffic light. So, we need to have some kind of a synchronization between this traffic light. So, how do we actually represent that kind of synchronization and concurrency, or that can actually be done using petri nets. And then used as a visual communication aid to model the system behaviour.

So, as we know this a graphical tool. So, it actually helps us a visual aid to show the synchronization. And of course, there are strong mathematical foundation for these

methods. And it actually captures the precedence relations, and structural interactions of potentially concurrent and asynchronous events. So, that actually captures the precedence relations. So, what is the precedence which function should precede or succeed the a particular process. So, that actually if this petri net captures that precedence relationship. And structural interactions of potentially concurrent and asynchronous events.

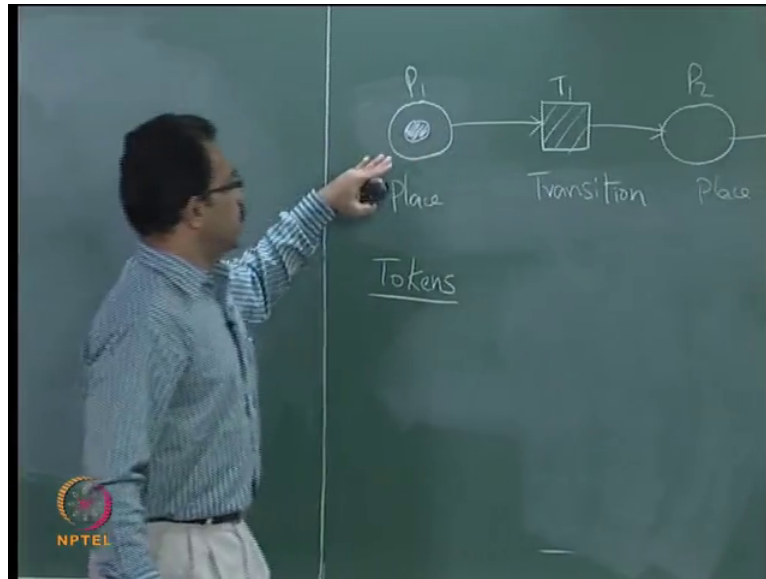
So, there maybe asynchronous events. So, we will actually able to capture the structural interaction between these events using petri nets. We will see a how to develop the petri nets. So, petri net has got a few basic specifications.

(Refer Slide Time: 25:02)



So, it actually consists of 3 types of components. Places transition and arcs. So, these are the 3 components which actually forms the a petri net. So, in petri net we use the places.

(Refer Slide Time: 25:18)

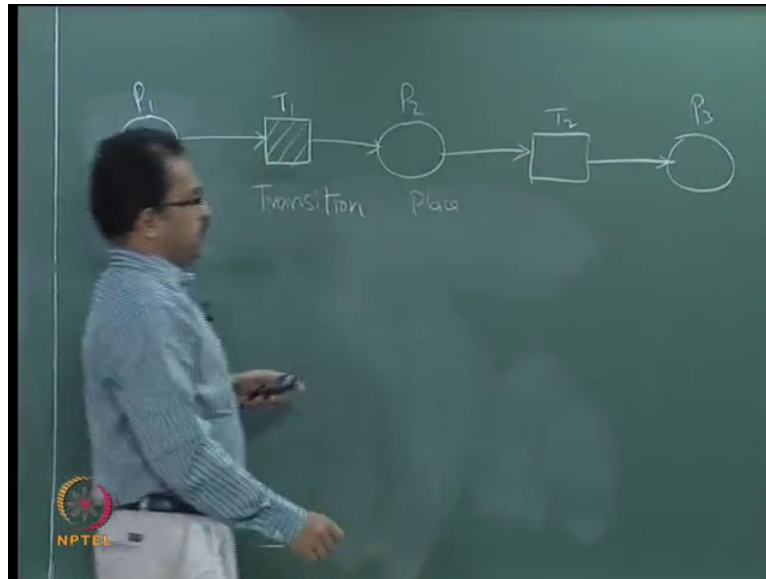


So, the places are represented by circles, and then you have the transitions. The transitions are represented by rectangles. This is a place, transition, and then we have the arcs which actually represents the movement of the or the dynamics of the system.

So, we can actually have the arcs which actually connecting the places and transitions. So, this actually is a place. So, it actually shows one stage, and this transition represents the change of state. So, if there is a change in the state of a system, that actually can be activated by the transition block. So, the translation block when you activate the place value will change, and that is actually shows that this has changed from this state to this state. Or this it has changed it is place value from once to the other one. So, this transition actually represents the transition from 1 place to another place that is the place transition, and the arc which actually represent these are the basic building blocks for the petri net. And then they mentioned the places represent possible states of the system.

So, this are the possible states. So, we can represent this as p_1 p_2 etcetera.

(Refer Slide Time: 26:41)



The possible states. So, they will be having we can have multiple transitions, and multiple places. So, $p_1 p_2 p_3$. So, this will be T_1 transition 1 transition 2 like that. So, we can have many number of places and transition in a petri net. Then the transitions are events or actions which cause the change of stage. So, as I told you. So, it is actually changing from this state to this state. So, the transitions are the which actually makes the place value to change or the status to change from one to the other one.

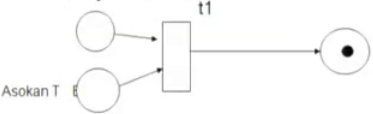
So, every arc connects a place with a transition or a transition with a place. So, that is the arc. So, we have the state transition and arc, place transition and arc, and place actually represent the states $p_1 p_2 p_3$, and transition represents the change from one state to other state. And the state of petri net is defined by the marking. In again this marking; so, the state of a petri net or a state of a particular position is represented by a the marking is a black circle a filled circle will be represented to show the status whether it is in active mode or not will be shown by this black circle or filled circle. And this is known as tokens.

So, the tokens represent the basically the status. So, the state will be represented by this tokens.


(Refer Slide Time: 28:09)

A Change of State ...

- is denoted by a movement of *token(s)* (black dots) from place(s) to place(s); and is caused by the *firing* of a transition.
- The firing represents an occurrence of the event or an action taken.
- The firing is subject to the input conditions, denoted by token availability.

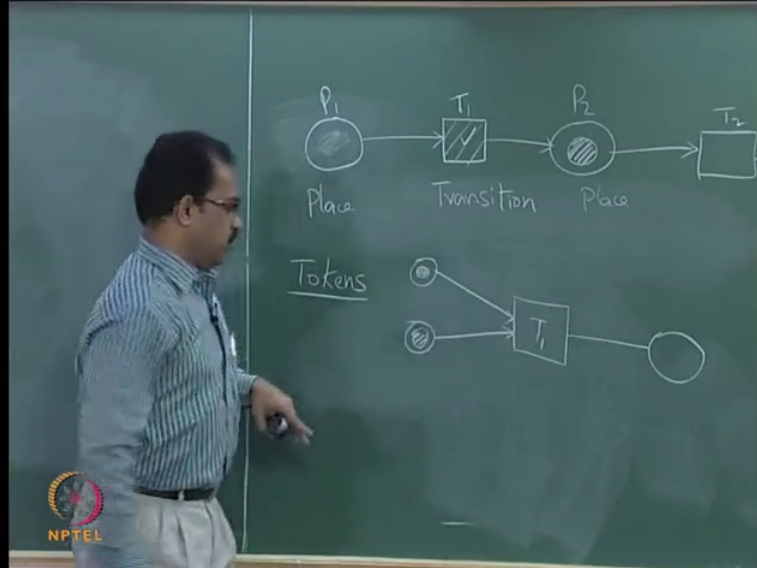


Asokan T




And the change of state that is a the state of a particular place is shown by the token. And change of state by is denoted by the transfer of tokens. So, when this state is changed to this one is actually show the transfer of this token. So, we have a token here. So, when I activate this T 1.

(Refer Slide Time: 28:27)



The chalkboard contains two diagrams. The top diagram shows a Petri net with three places: P_1 (a circle with a token), P_2 (a circle with a token), and an unlabeled place. Between P_1 and P_2 is a transition T_1 (a square with diagonal lines). An arrow points from P_1 to T_1 , and another from T_1 to P_2 . Below this, the word "Tokens" is written, followed by a diagram showing two tokens (black dots) in circles pointing to a transition T_1 , which then points to an empty circle.



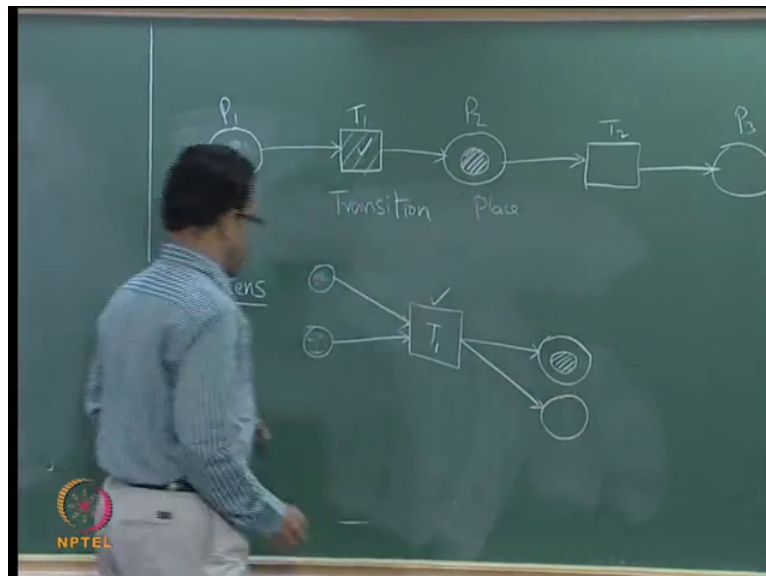
And this is activated this token will be shifted to this one. So now, this is in there active modes. So, that is the change of position or the change of transfer of token.

So, every transition basically causes the change of the token or a transfer of the token. So, this change of state is denoted by movement of token from place to place and is caused by the firing of a transition. So, the firing of the transition, causes the token to move from this place to this place. So, that actually represents the change of state of a particular function or the process. In the firing represents an occurrence of the event or an action taken. The firing is subject to the input conditions denoted by token. So, that is another important point the firing is subjected to the initial conditions.

Now, if this is connected this transition suppose you have a transition like this T_1 . So, we can actually have multiple places connected to this. So, if it is given like this. So, the firing of this token is subjected to the status of this position. So, if there is a token in both these places, then only this transition can take place. So, the firing is subjected to this condition. If this firing has to happen then both this should be having the token.

If there is only one token, then this transition cannot take place; that means, these 2 condition need to be satisfied for this transition to take place. So, in this case when I do the firing of this transition, both the tokens will be going to here and this will show a status over here.

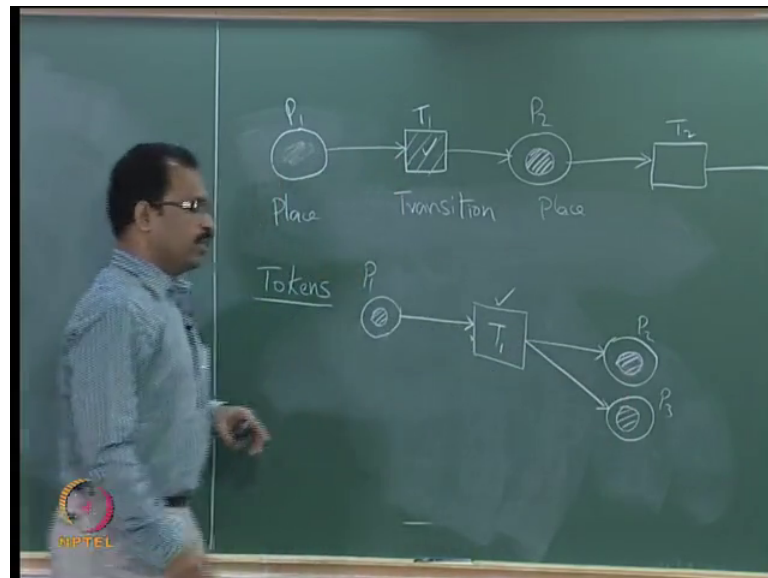
(Refer Slide Time: 30:05)



So, this will be removed, and this will be there will be a token over here for when this is fired.

If there is only one token then this cannot be fired there will not be any transition there will not be any change in the status of the that particular place. This is one situation, and another one is if you have like this, and if you have this kind of a situation. So, you have only one token here. So, these are simply petri net. So, we do not have any place values as such at present.

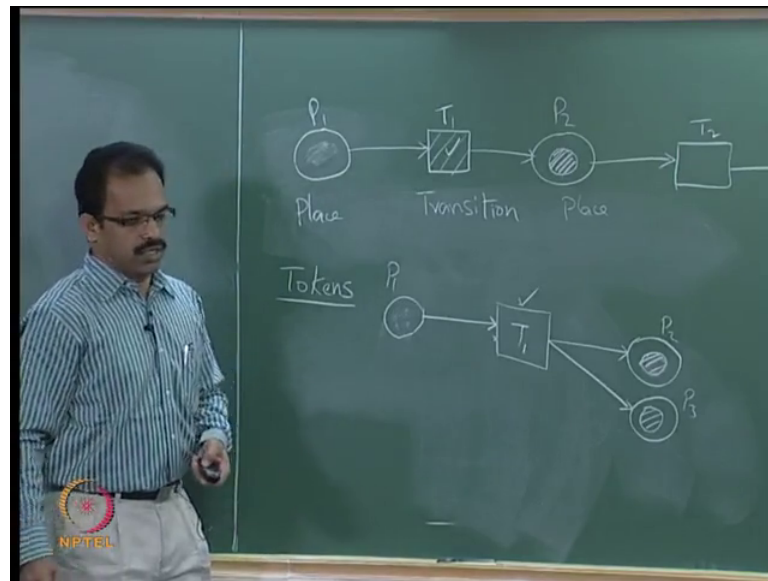
(Refer Slide Time: 30:40)



So, you have a place connected to p 1 connected to this T 1 and p 2 and p 3 and if there is a token here available.

So, if this is the situation, when we can fire this transition. Because now it is all condition satisfied that there is a token over here. So, we can actually fire this transition, and when you fire this transition both will be activated. So, both the places will be activated. So, this the numbers does not really matter how many places are connected. The transition simply shows that from this status it has moved to this status any number of places can be connected to this transition or this will be concurrently activated. So, that is the condition for transition, and then after transition what is happening.

(Refer Slide Time: 31:32)




So, basically the petri net if they has got this place transition and arcs and using place transition an arc we can actually show many dynamic situations. So, as shown here. So, if this the condition for transition is basically having set has to satisfy the input conditions. So, if there is a token then only it can be fired otherwise it cannot be fired; that means, if this condition is not satisfied there is no possibility for changing this status to another status. So, that is the condition for transfer of tokens.

(Refer Slide Time: 32:03)

A Change of State

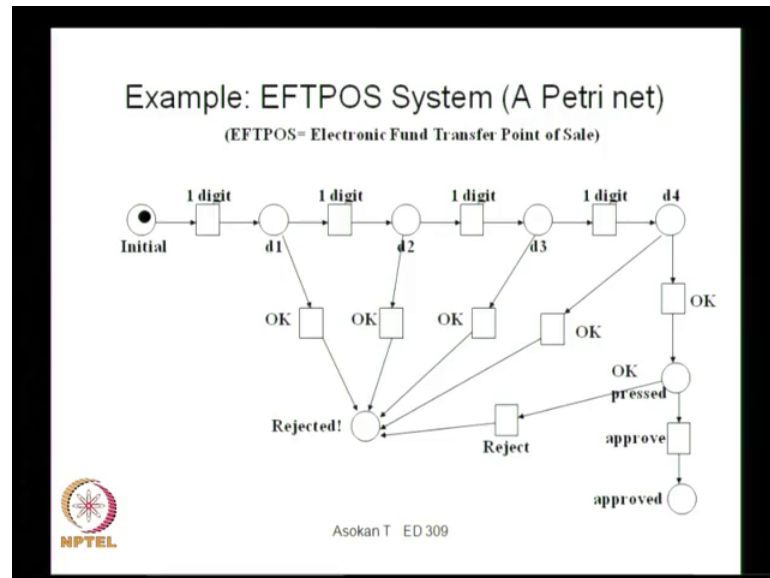
- A transition is *firable* or *enabled* when there are sufficient tokens in its input places.
- After firing, tokens will be transferred from the input places (old state) to the output places, denoting the new state.



Asokan T ED 309

So, this is what I already explain. A transition is firable or enabled when there are sufficient tokens in its input places. So, there should be sufficient tokens in its input places to make a firing. And after firing, tokens will be transferred from the input places to the output places denoting the new state. So, once you do the firing, the tokens will be transferred from the input to the output; it actually represents the change of state.

(Refer Slide Time: 32:30)



We will take an example, and show you how to actually represent a scenario using the Petri nets.

So, here we take again a simple example in electronic fund transfer point of sale. So, we know that we normally go to an ATM machine or an electronic point of sale; we have to insert the card and give the password, and then only it will accept the card and then the transaction will take place. So, how do we actually show this using a Petri net? It is a very simple example for use of Petri net. Petri net can be used for very more complex situations, but you know to understand the use of Petri net, we will take this example.

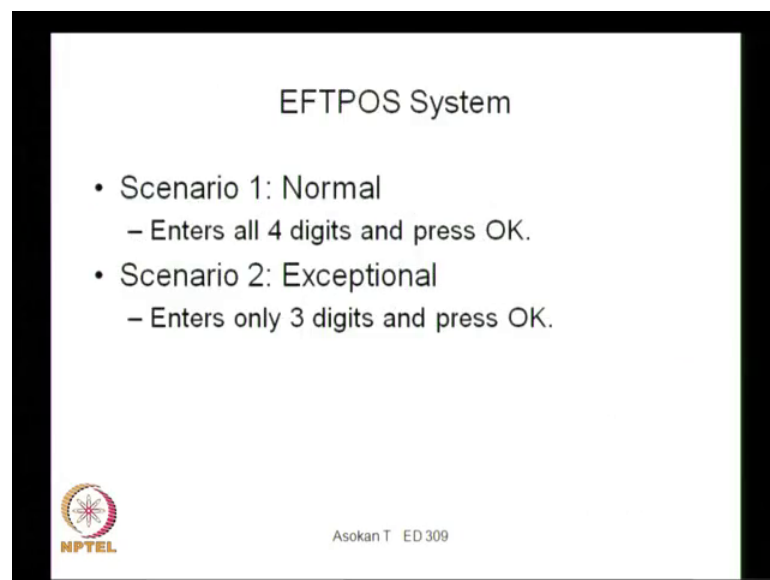
So, look at this diagram. So, this is the initial status. So, we have this initial token available, and the card is inserted; the next is basically you need to give the password to the system. So, here actually you can see there are 4 digits. So, you can have 4 digits inserted. This is the digit 1, then digit 2, this is a second digit, the third digit, and fourth digit. And once all the 4 digits, then you can have a transition, and then approve the transaction and then get the approved transaction out.

So, here you can see 4 digits need to be inserted and (Refer Time: 33:44) you can see that is a transaction. So, one digit is considered as a transition. So, you can actually give first digit, once there is an input this initial token is available you can provide the first digit as a transition. And once you do this then this status will come that is there is a token over here, then you can insert the next digit. And once you have the next digit, then you have your next transition. This is a third digit. Then the place will come here, and then keep on going to this loop, but in between suppose you give one digit then there is something called an button.

So, this button can actually be activated, once you have a token over here. So, this can be activated if you have a token at d 1, but then it will get rejected because you have just entered only one digit. So, it will do not work similarly you can see the other buttons also. After the d 2 also you can press the, but then it will not work because it will go to the rejected situation. So, you need to have are the 4 digits. And then the press the button. And then only it will go to the approve stage and get the transaction done.

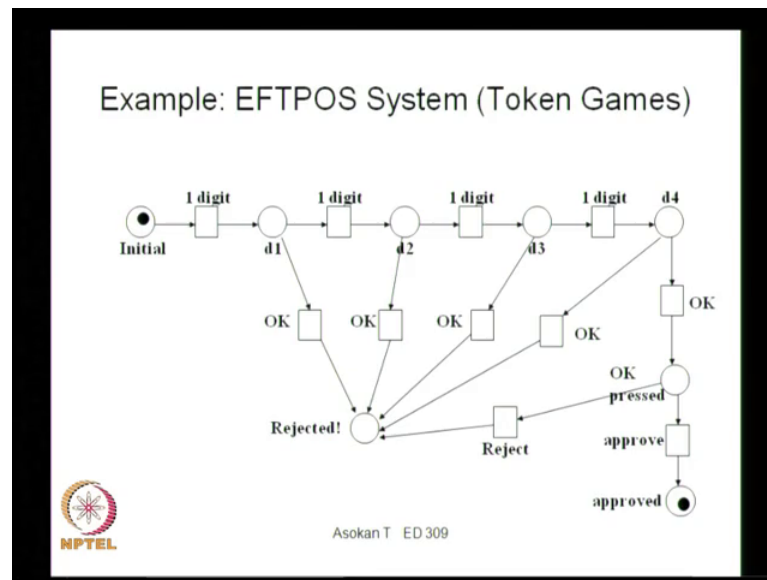
So, we can represent this in the 2 scenarios.

(Refer Slide Time: 34:56)



We will show that one normal scenario, where we enter all 4 digits and press, and then scenario 2; where we exceptional cases where we enter only 3 digits and press what will happen. So, how the dynamics of the system can be represented under these 2 situations.

(Refer Slide Time: 35:12)

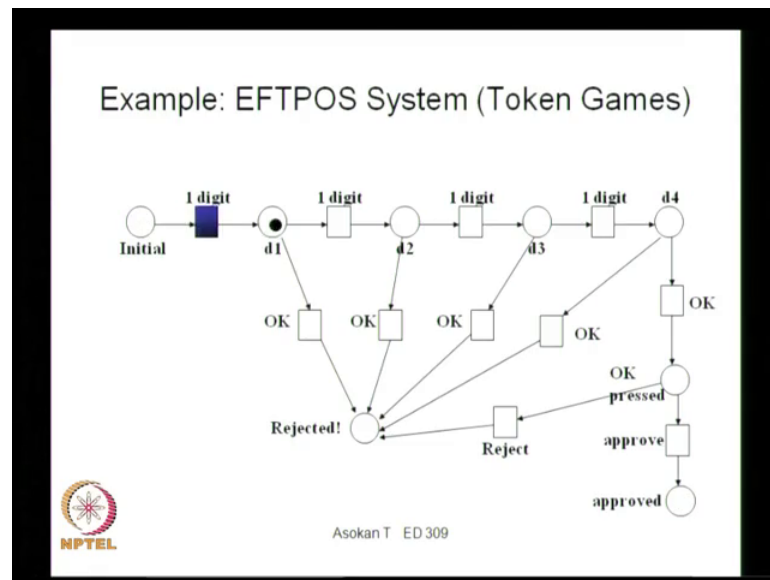


So, let us look at this. So, this is the first situation. So, we have an initial token available at the position. So, we already entered the card and then it is ready to expect the digit. So, if you just give that transition. This transition, then you can see that the token has changed to d 1. From initial the token has moved to d 1. And then if you do the next transition that is next digit. So, you can see that token has moved to d 2, and then one more digit it has moved to d 3, one more digit has moved to d 4. And then you have this button. So, you have already given 4 digits and then press it comes to that stage of token.

So, here you can have an option to reject the transaction, or you can approve the transaction. After entering all the 4 digit if you want to reject, then you can actually give this reject transition what is shown here. Otherwise you can give the approve, once it is approved then the transaction will be approved. And the token has come over here and once the token reaches this point approved again there will be one token available at the initial; that means, it is ready for the next transaction. So, that is the one loop of normal transaction. So, you will be given 4 digits in succession, and then press the approve and it will go to the approved situation. And again it will the token will be made available to the initial status so that you can actually start the operation again.

So, that is a one condition. Now the second scenario, where you press only 3 digit and press then what will happen we can actually show the same thing again using petri net.

(Refer Slide Time: 36:46)




So, the first digit, second digits, then you have the third digits. And now if you press it will go to the rejected and transaction will be completed, or it can be actually a terminated the token will be again made available to the initial one. So, it will be rejected and it will go back the initial one, and he will be getting you can actually start the process again.

So, this is the way how we can actually represent a simple transition or simple a dynamics of a system using petri nets. Again, show you few more examples of using petri nets.

(Refer Slide Time: 37:21)

Example: Vending Machine

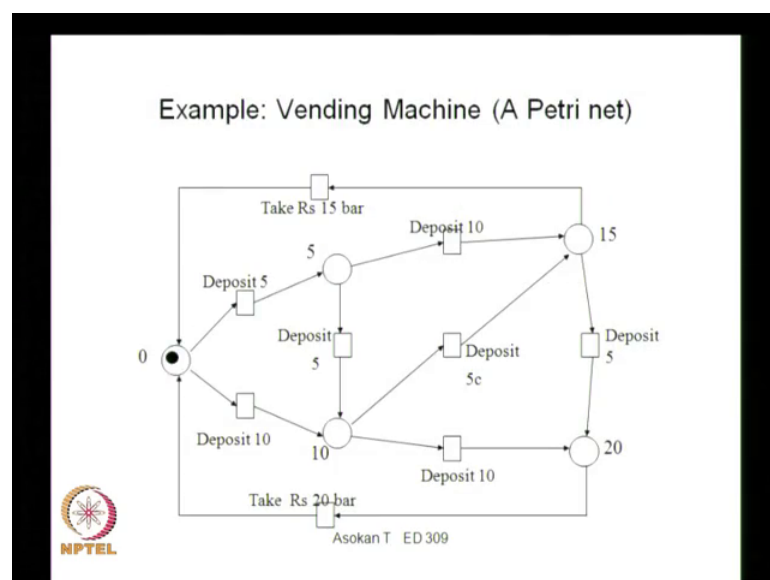
- The machine dispenses two kinds of snack bars – Rs 20 and Rs 15.
- Only two types of coins can be used – Rs 10 coins and Rs 5 coins.
- The machine does not return any change.



Asokan T ED 309

So, this is an example for a vending machine. The machine dispenses 2 kinds of snack bars 20 and 15. And only 2 types of coins can be used. That is rupees 10 and rupees 5, and the machine does not return any change. There are multiple options for you to actually insert the coins and then get the output. So, we need to show all those options and the dynamics using the petri net.

(Refer Slide Time: 37:45)



Here you can see that this is the vending machine petri net. Here you can see the situations that you can actually deposit 5 coin or a 10 coins. So, these are the options


then there are different loops you can actually go through this loop. So, you can go through any one of these loop, that you can go to the deposit 5 deposit 10 and deposit 5 loop. Or you can go to the bottom one deposit 10 deposit 10 and go to the 20 loop.

So, that where we can have different options. So, we will just show a few cases, how do you actually represent this kind of scenario using a petri net?

(Refer Slide Time: 38:18)

Example: Vending Machine

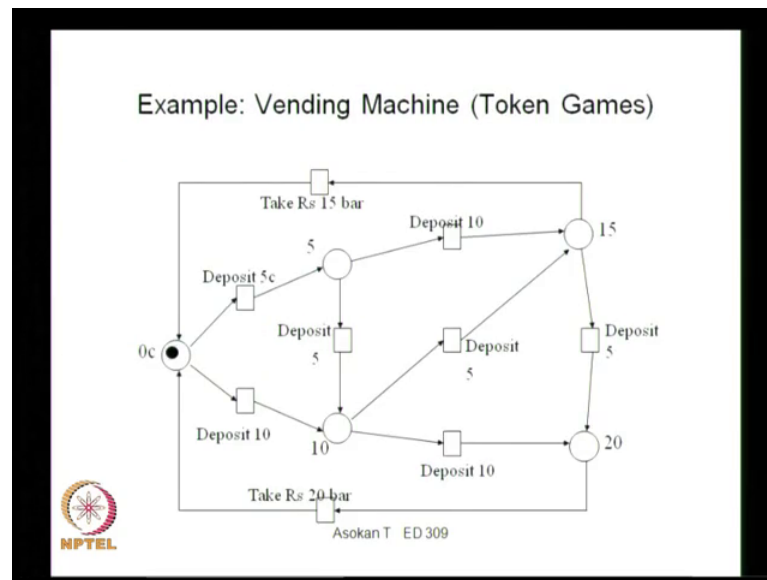
- The machine dispenses two kinds of snack bars – Rs 20 and Rs 15.
- Only two types of coins can be used – Rs 10 coins and Rs 5 coins.
- The machine does not return any change.



Asokan T ED 309

So, scenario 1 is deposit 5 5 5 5, and then take a 20 rupees snack bar. Or scenario 2 is deposit 10 5, and then take a 15 snack bar. And the scenario 3 is deposit 5 deposit 10 deposit 5 and then take a 20 snack bar. So, we have multiple options. So, all these options can be represented using the petri net.

(Refer Slide Time: 38:40)



So, here is the petri net for this particular scenario. And we have the token here. Initially it is 0. So, it is ready for firing. So, you can actually give a 5 or 10 you can see here that this 5 and 10 are actually can be fired both can be fired because this 5 is connected to the 0, and this 10 is also connected then initial condition for transition is that there should be a token available that the circle mark the 0 or the place marked 0. So, the here actually we can one token is available therefore, you can fire any one of this.

So, if you put a 5-rupee coin then it will go to that 5-rupee loop otherwise it will go to the 10 rupee. So, here see here actually putting 5, then it will goes to that 5 that 5 token is available. Now you can actually do go for this loop 10 or you can actually go for this also. Because both are connected to that particular token that place. So, you can actually use any one of these. So, here if you do the 5 again then it will come to the 10. And now you have an option of depositing 10 or depositing 5. So, if you deposit 5 you can get a 15 rupees snack, or if you go for a 10, then you can take a 20's rupees snack.

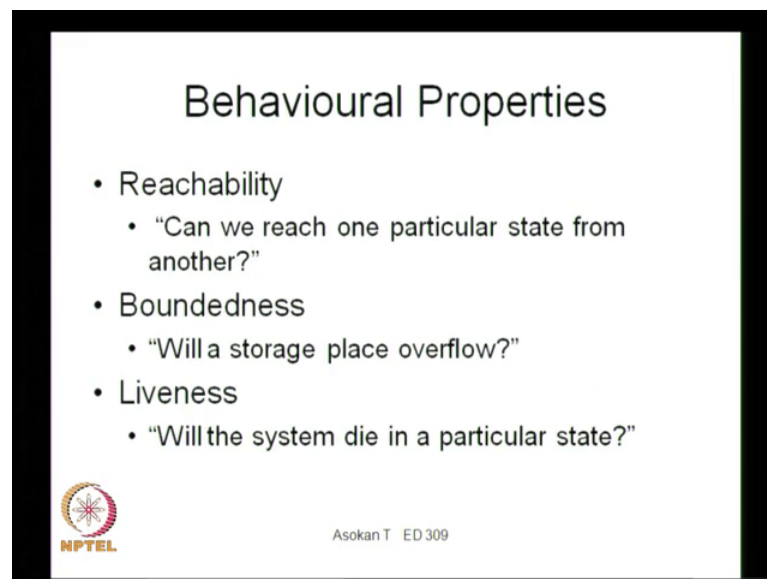
So, here another 5 is given. So, it is the 15 place is now value or that status is reach. So, there is in a option of taking the 15 rupees bar. So, you can actually activate this transition of 15 rupees. So, this kind of activated or you can actually deposit another 5 and take the 20. So, here another deposit another 5 now 20 is activated. So, 20 place is active. So, this is connected to the 20 transition of take 20.

So, we can actually activate this and take the 20 or rupees snack, and then transition will go back to the place 0 will be activated that place will be activated. The token will be transferred from 20 to 0. So, once you take this 20-bar snack it will be transferred to 0. So, that is how that one transaction take place. Now you have another option of 10 then 5, and then taking the 15 rupees snack as you can see here. So, the 15 rupees bar is taken and the token has come back to 0.

Now, the third option is you have 5 rupees deposit. Then another 10 rupees deposit and then 15. And then deposit another 5 and then take a 20 rupees snack and go to the next one. So, these are the ways how we can actually represent the various transaction using petri net. So, again this is not a complex scenario, but of course, you can represent complex scenarios also using petri net.

I have a few examples for the complex scenarios also.

(Refer Slide Time: 41:16)



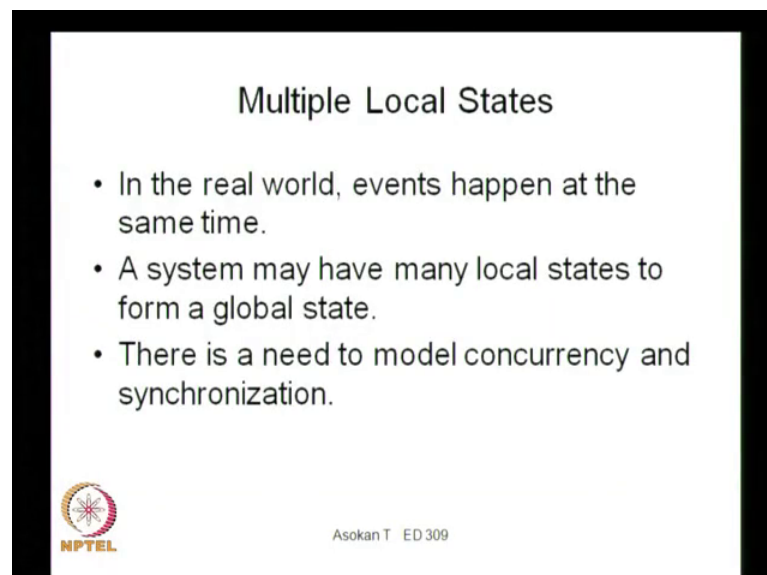
So, here before going to those complex scenarios, we will see those scenarios also, but the behaviour properties that we can actually analyze using petri net are the reachability basically can we reached one particular state from another. So, we have a particular state of a one situation. And we want to reach another situation. So, we need to find out what are the possibilities to reach that situation, whether we can really reach that situation from the present situation.

So, we can model that situations and then see by placing the tokens and the transition, you have to analyze whether we can really reach that situation or not. So, that can be done using petri net. The another one is boundedness will a storage place overflow. So, one some cases like if you have a what you call a storage, or a location where you have multiple storage facilities. And you are getting supply from different resources. So, whether there is a possibility of an overflow of that particular storage. And whether what is the situation under which we need to place an order. So, that we make sure that the optima usage of the storage space is done.

So, that scenario can actually be a modelled using petri net. The another one is the liveness of the situation, whether will the system die in a particular state. So, whether it will go to a situation where we cannot get an output. So, there maybe some situations, where we may not be intended, but still if you analyze some situation we will see that some situation we cannot get ready and output. Or somewhere actually dies an they now output can be done. Because the places you require transitions are not taking place because the conditions are not satisfied.


Such situations also can be unless using petri nets.

(Refer Slide Time: 42:52)



Multiple Local States

- In the real world, events happen at the same time.
- A system may have many local states to form a global state.
- There is a need to model concurrency and synchronization.

 NPTEL

Asokan T ED 309

Sometimes the things will be happen at same time. And then system may have many local state to form a global state. So, some systems we will be having multiple input states, and then this multiple input states will lead to a global state. So, that also can be

modelled using petri net. And there is a need for concurrency and synchronization that also can be modelled and analyzed using petri nets. So, these are the advantages of using petri nets. And will try to see how to model this kind of scenarios of overflow or system getting into a situation where there is no output. Or where how the concurrency can be modelled.

So, these cases we will see in the next class, I will take a few more examples to show the concurrency as well as the overflow situations and then ordering situations. So, how do we actually model such dynamics using petri net. This we will discuss in the next class. So, till we meet goodbye to all of you.