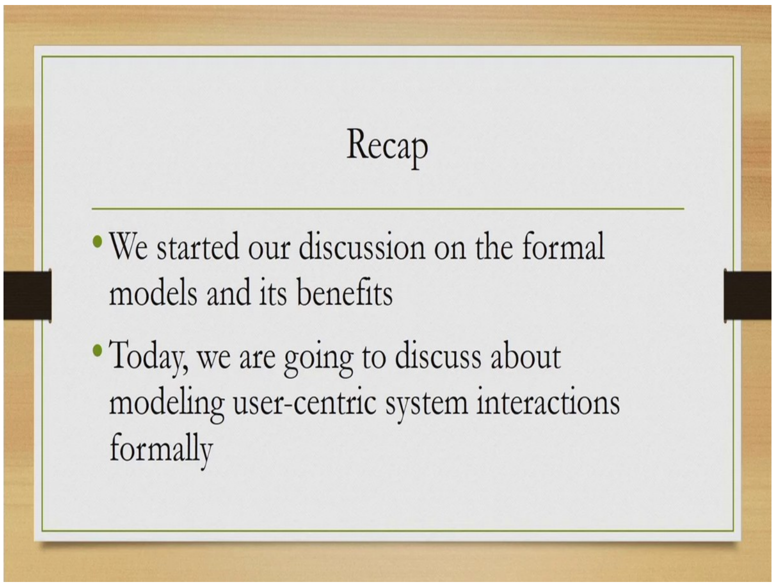
User-Centric Computing for Human-Computer Interaction

Prof. Samit Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

Lecture - 24
Formal modeling of user-computer dialogue

Hello and welcome to lecture number 24, the course User Centric Computing for Human Computer Interaction.

(Refer Slide Time 00:51)



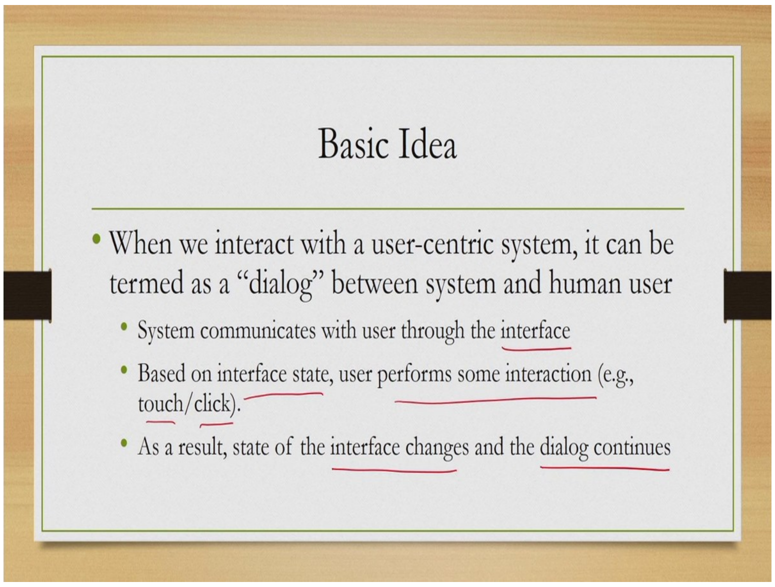
Recap

- We started our discussion on the formal models and its benefits
- Today, we are going to discuss about modeling user-centric system interactions formally

So, before we start today’s a lecture, first let me briefly recollect what we have learned in the previous lecture. In the previous lecture we started our discussion on formal models particularly in the context of user centric systems. So, there we got introduced to some basic ideas that is, what is formalism and what is a formal model and how it is different in the context of user centric system.

Also we have briefly seen the use of the formal models, particularly in the context of verification of properties. Now, these properties are supposed to be in directly related to the usability of the system. So, essentially by verifying these properties by ensuring that the system supports these properties, we can ensure that the proposed system or the system that we proposed to design is usable. Now, in this lecture we are going to understand or learn about these formal models of interactive systems in more details.

(Refer Slide Time 02:08)



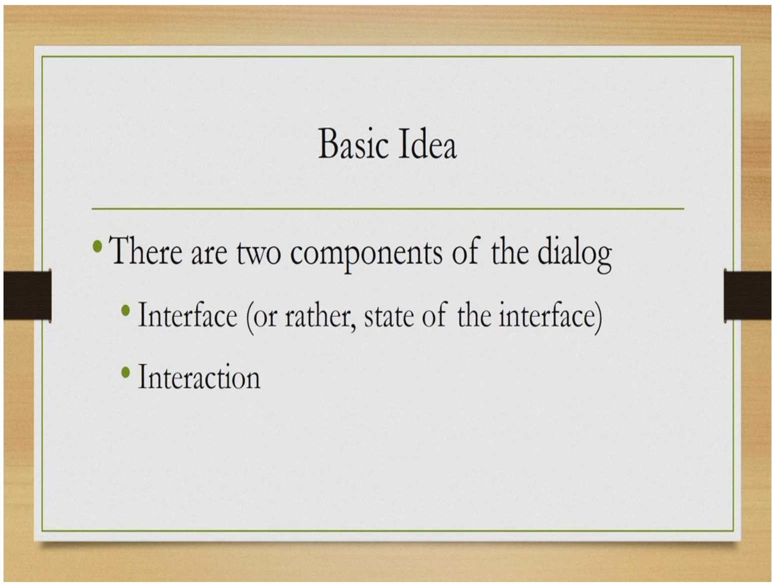
Basic Idea

- When we interact with a user-centric system, it can be termed as a "dialog" between system and human user
  - System communicates with user through the interface
  - Based on interface state, user performs some interaction (e.g., touch/click).
  - As a result, state of the interface changes and the dialog continues

There are many ways in which we can think of a system. One very intuitive way to view user centric system is to think of it as a dialog between a computer and a human user. So, essentially, the system communicates with the user through the interface. So, the interface is the medium of communication for the system. Based on the interface state whatever is the state of the interface the user perform some interactions essentially, touch, click or such interactions. And as a result the state of the interface changes and this process continues or the dialog continues.

So, there is an interface through which the system communicates something to the user perceives that by perceiving the state of the interface. Based on that perception, the user perform some action on the interface which leads to change in the state of the interface and that is the dialog and it continues till the user achieves the desired objective or achieves the goal.

(Refer Slide Time 03:39)

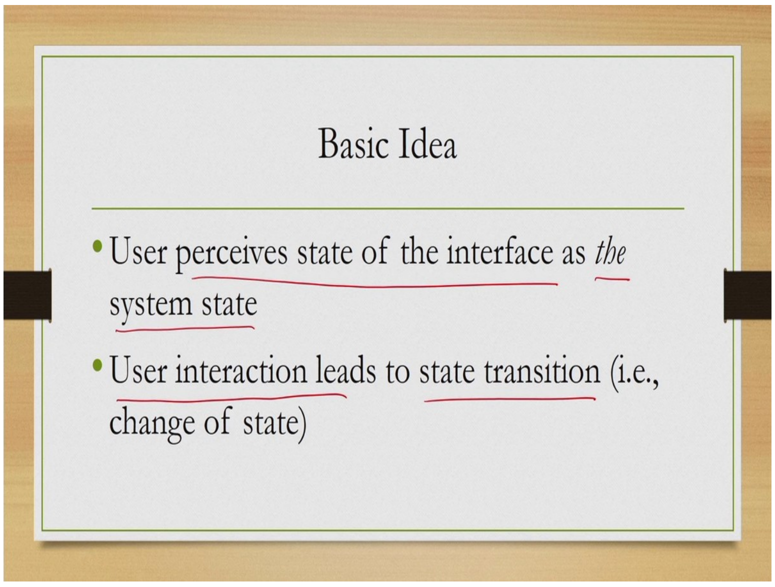


Basic Idea

- There are two components of the dialog
  - Interface (or rather, state of the interface)
  - Interaction

Then, we can see that in this view of the system as a dialog there are two components; the interface and the interaction. By interface we mean the state of the interface and the second component is the interaction.

(Refer Slide Time 03:59)



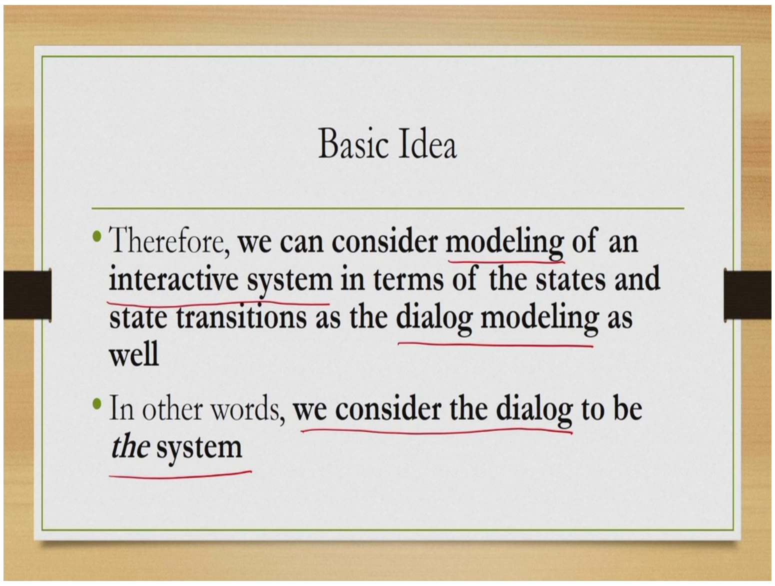
Basic Idea

- User perceives state of the interface as *the* system state
- User interaction leads to state transition (i.e., change of state)

So, in a nutshell, what happens is that user perceives the state of the interface as the system state note the emphasis on the term the. So, whatever is the state of the interface is perceived by the user as the state of the whole system. So, user is not bothered about the internal working, how the memory content looks, how the processor is working, how the battery is consumed or what is the bus utilization bandwidth utilization all these things. The user is only concerned about what he or she can perceive on the interface and that is the state of the system for the user and user interaction leads to state transition.

So, essentially by interacting with the interface, the user can take the system to a new state. These are the basic components of a view of the system where we think of the system as a dialog between the user and the computer.

(Refer Slide Time 05:18)



Basic Idea

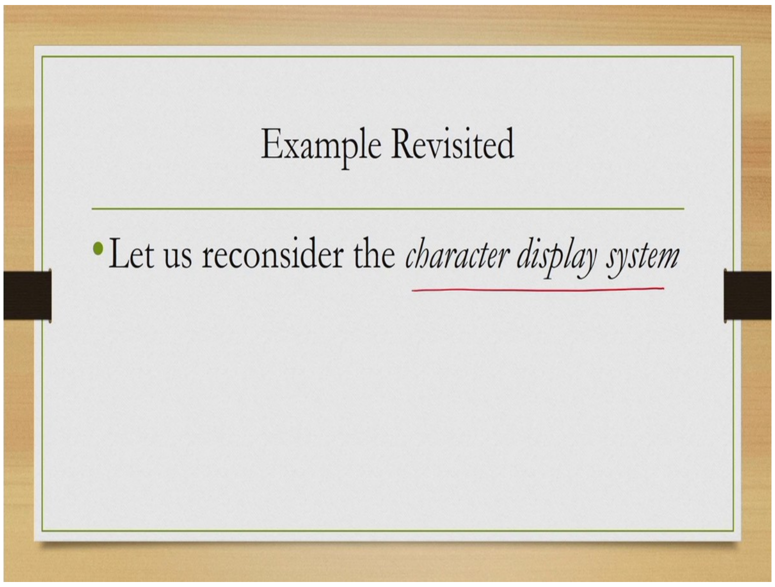
- Therefore, **we can consider modeling of an interactive system in terms of the states and state transitions as the dialog modeling as well**
- In other words, **we consider the dialog to be *the* system**

So, if that is the case then we can consider modeling of the system or modeling of the interactive system as dialog modeling as well. So, when we say that we want to model our interactive system in terms of system states and transition between states, essentially what we are trying to model? We are trying to model the dialog between the system and the user.

So, system modeling is equivalent to modeling the dialog as well. In other words, we considered the dialog to be the system. So, earlier we considered the system to be a composition of system state and transition. Now, we are saying that this is nothing, but the dialog between the user and the computer and our objective is to formally model this dialogue.

Now, earlier in the previous lecture, we have seen that state transition networks can be used to model the system. We can use states and transitions between them use the state transition network notation to model the system of states and transitions; however, all our examples were for simple systems. So, if the system becomes too complex then the basic state transition networks that we have seen in the examples in the previous lecture will not be sufficient to model such complex systems and we required some better modeling techniques.

(Refer Slide Time 07:06)

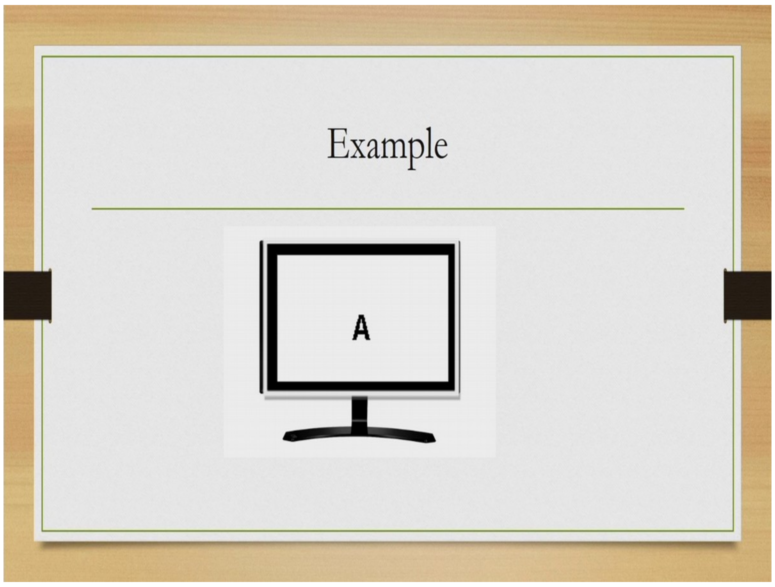


Example Revisited

- Let us reconsider the *character display system*

In this lecture we are going to discuss about few of some advanced modeling languages for modeling complex interactive systems. Let us reconsider the example system that we have introduced in our last lecture the character display system.

(Refer Slide Time 07:20)

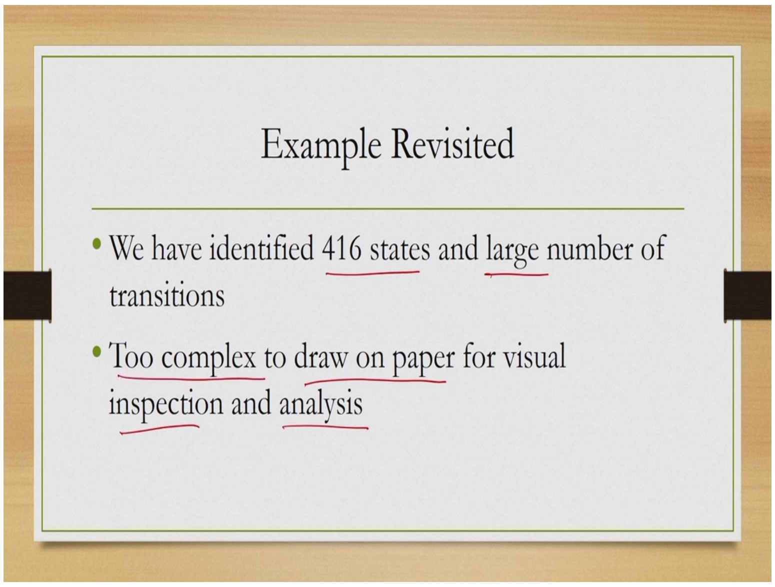


Example

Recollect that, it is only a simple system it has only a very simple interface which is capable of showing only one letter of the English alphabet, one letter in capital. And we can change the front size of this letter and we can change the color of this letter.

So, there are 26 capital letters in the English alphabet, we can change its font size by making it one of the 8 possible values that we have defined as mentioned in the earlier lecture. And also it can change its color between red and black. So, total we have identified 416 states for this simple interface system and between these states there will be transitions. So, the total number of transitions will be large.

(Refer Slide Time 08:05)

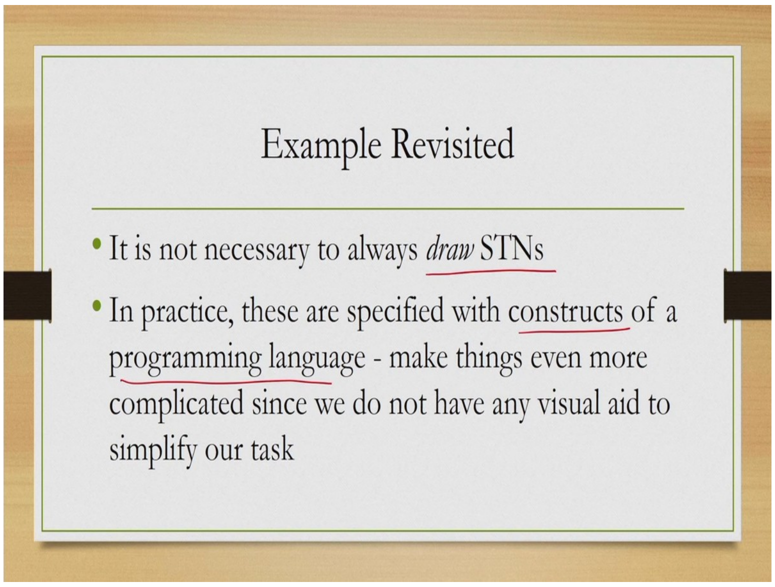


Example Revisited

- We have identified 416 states and large number of transitions
- Too complex to draw on paper for visual inspection and analysis

Now, we want to basically represent this system in the form of STNs; so, for that we need to probably draw it on paper. But as we have already mentioned drawing such a large number of states and transitions on a paper for visual inspection and analysis is too complex a task.

(Refer Slide Time 08:50)

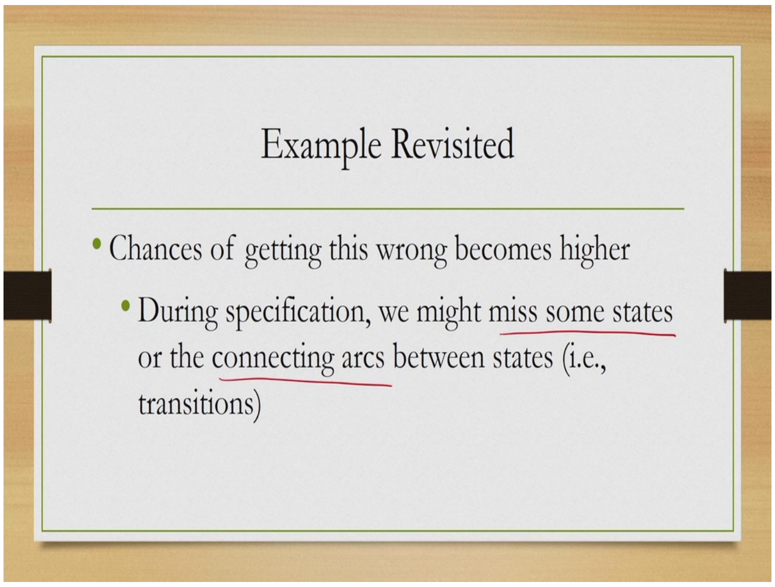


Example Revisited

- It is not necessary to always *draw* STNs
- In practice, these are specified with constructs of a programming language - make things even more complicated since we do not have any visual aid to simplify our task

You may argue that it is not necessary to always draw STNs instead what we can do is that, we can use some programming language construct to specify the STNs and then perform some analysis on this specification. But that itself is not easy unless we are able to visualize the whole system, we are likely to make some mistakes during specification as well.

(Refer Slide Time 09:23)

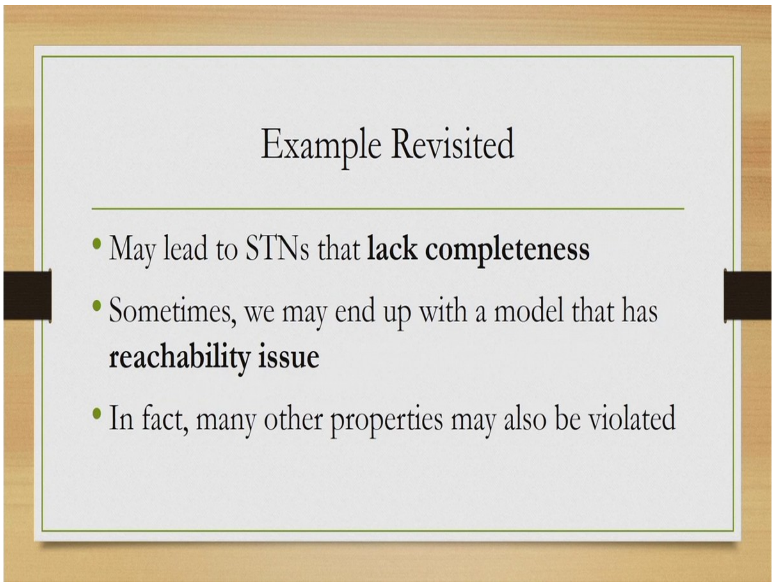


Example Revisited

- Chances of getting this wrong becomes higher
  - During specification, we might miss some states or the connecting arcs between states (i.e., transitions)

For example, we may miss some states or some connecting arcs between states during specification because we are unable to visualize it. Now, what is the consequence of such mistakes? There can be many consequences actually by making those mistakes we can violate various desirable system properties which otherwise would have been there.

(Refer Slide Time 09:39)



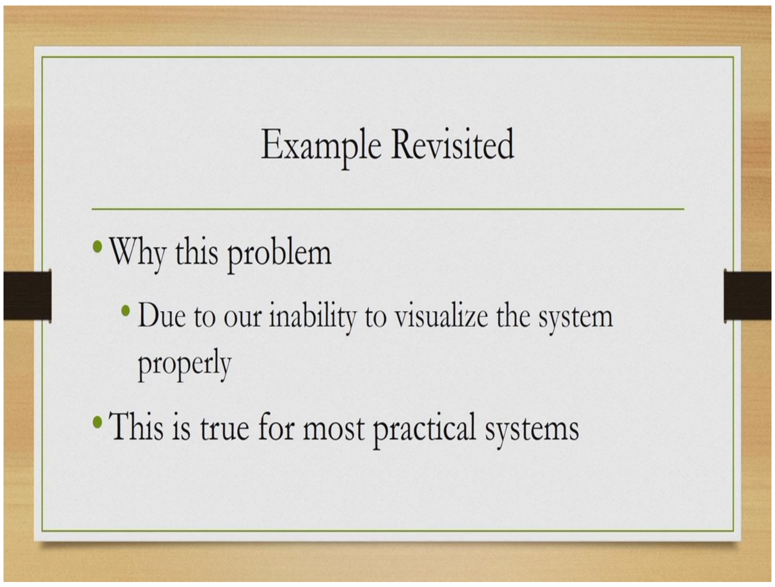
Example Revisited

- May lead to STNs that **lack completeness**
- Sometimes, we may end up with a model that has **reachability issue**
- In fact, many other properties may also be violated

For example, by making the mistake we can actually develop a STN which may lack completeness. Or sometimes we may end up with a model that has reachability issue other properties may also get violated then that defeats the whole purpose. So, we have actually designed something very nice which supports certain properties which ensures usability indirectly.

However, during specification for later analysis or during coding of that model for software development, which involves specification; we may end up specifying the model wrongly which may lead to violation of certain properties. So, our design is ok, but our specification is the problem.

(Refer Slide Time 10:54)

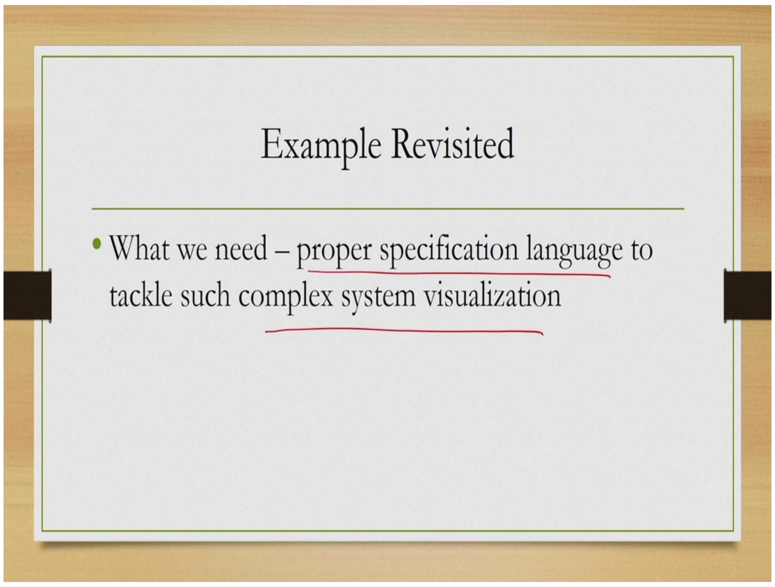


Example Revisited

- Why this problem
  - Due to our inability to visualize the system properly
- This is true for most practical systems

Now, why this problem happened? Primarily, because we are unable to visualize large and complex systems. And how we can resolve this issue?

(Refer Slide Time 11:06)



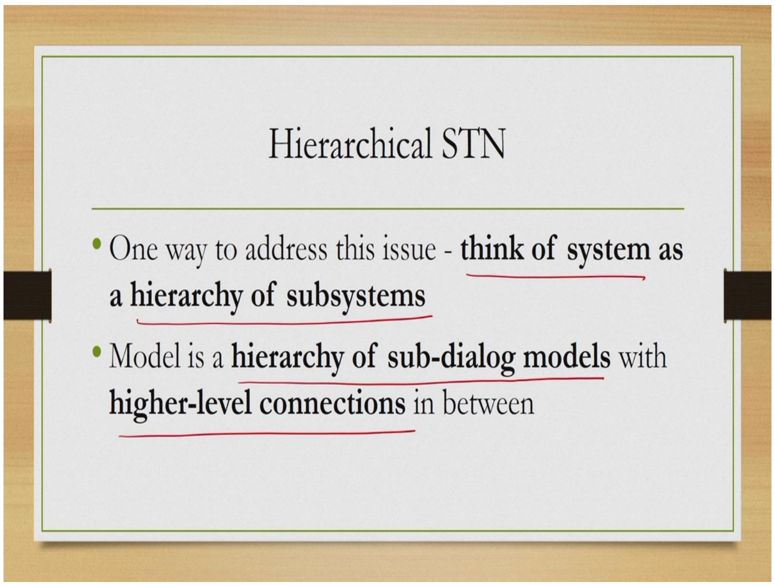
Example Revisited

- What we need – proper specification language to tackle such complex system visualization

We can resolve it by using a proper specification language to tackle complex system visualizations. In fact, most of the practical systems are large and complex they are not simple systems that we have encountered in our examples and unless we are able to properly visualize those systems, it is difficult to encode or specify those systems in computer.

So what do we need is a proper specification language. And in this lecture we are going to learn about a couple of such languages. So, we will start with one language which is called hierarchical STN or hierarchical state transition networks.

(Refer Slide Time 11:50)

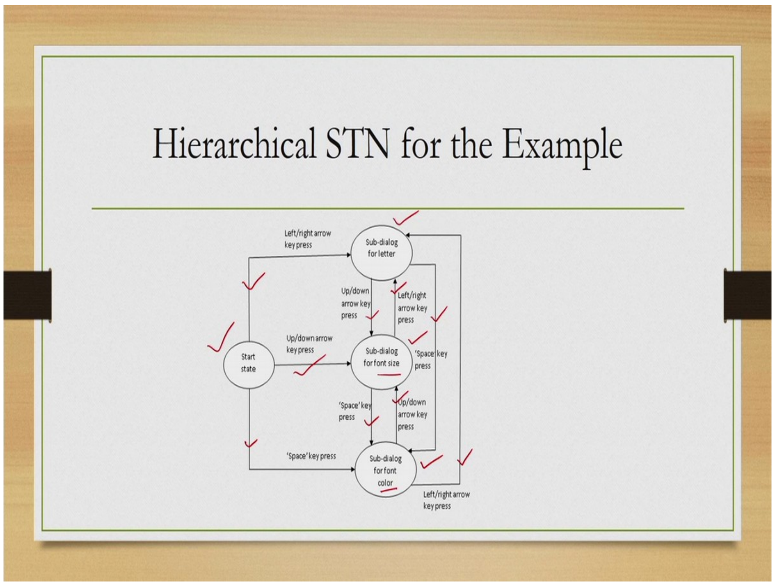


Hierarchical STN

- One way to address this issue - **think of system as a hierarchy of subsystems**
- Model is a **hierarchy of sub-dialog models** with **higher-level connections** in between

The basic idea is that we can think of the whole system as a hierarchy of subsystems. And the overall model is a hierarchy of sub dialog models with higher level connections in between. So, essentially we are thinking of the whole system not as a single system, but as a hierarchy of less complex subsystems and we provide higher level connections to those subsystems to model the whole system.

(Refer Slide Time 12:27)



Hierarchical STN for the Example

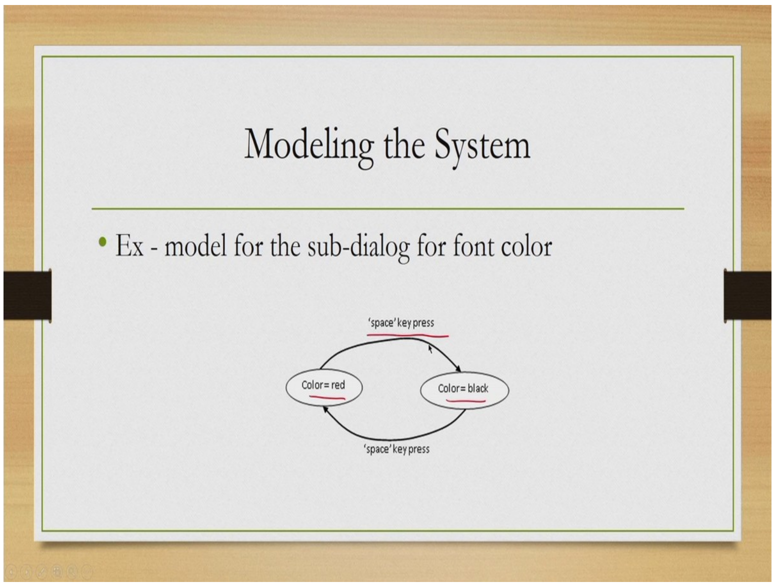
Now, if we want to use this hierarchical STN to model our example system that is the character display system, it will look something like this figure where we have sub dialog models. Say one sub dialog model maybe for letters, one sub dialog model maybe for font size and one sub dialog model for a font color.

Now, between these models we have connections. So, let us have a start state. So, from start state if we press the up or down arrow key then we go to the sub dialog for font size, if we press the left or right arrow key, then we go to sub dialog for letter. From sub dialog for letter if we press up down arrow key then we can come to the sub dialog for font size and at this sub dialogues or at this state of the system.

If we press left right arrow key then we can go to the letter sub dialog again. Similarly, if we press space key when we are in the sub dialog for font size state. Then we can come to sub dialog for font color and by pressing up and down arrow key we can go to sub dialog for font size again. We can come to sub dialog for font color from sub dialog for letter by pressing the space key and by pressing the left right arrow key we can go to the letter sub dialog from the color sub dialog. And from the start state if we press the space key we can come to the font color sub dialogue.

Now, this is a much nicer way to represent the whole system. As you can see we managed to depict the entire system within a small area by assuming that there are sub dialogues and there is higher level connection between those dialogues. So, each sub dialog can be a state transition network in itself or can be another hierarchy upstate transition networks.

(Refer Slide Time 14:46)

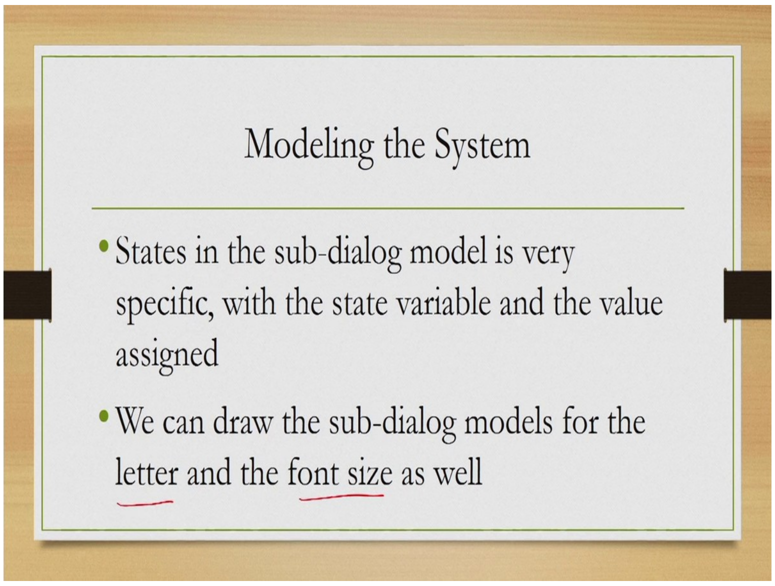


Modeling the System

- Ex - model for the sub-dialog for font color

For example, the sub dialog for font color can be represented as a state transition network as shown in this figure where there are two states color equal to red and color equal to black. And with interaction by pressing the space key we can move between these two states.

(Refer Slide Time 15:21)

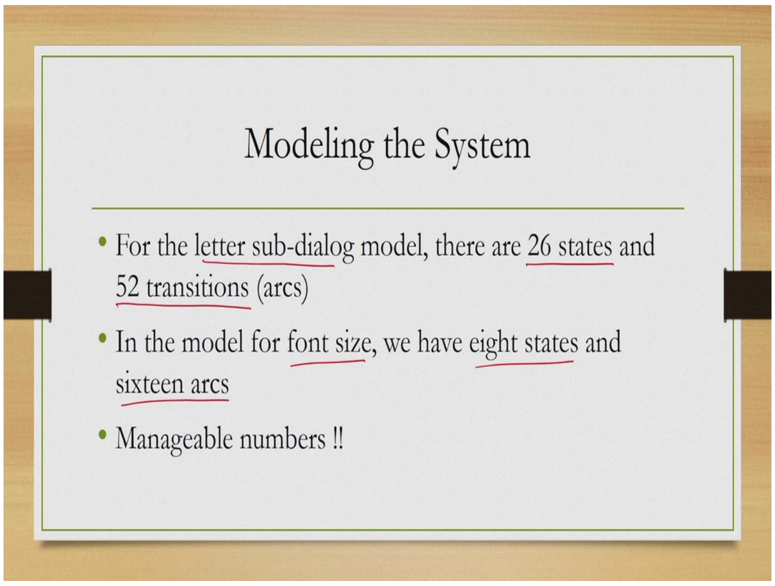


Modeling the System

- States in the sub-dialog model is very specific, with the state variable and the value assigned
- We can draw the sub-dialog models for the letter and the font size as well

Similarly, we can draw sub dialog models for the letter and the font size as well.

(Refer Slide Time 15:28)

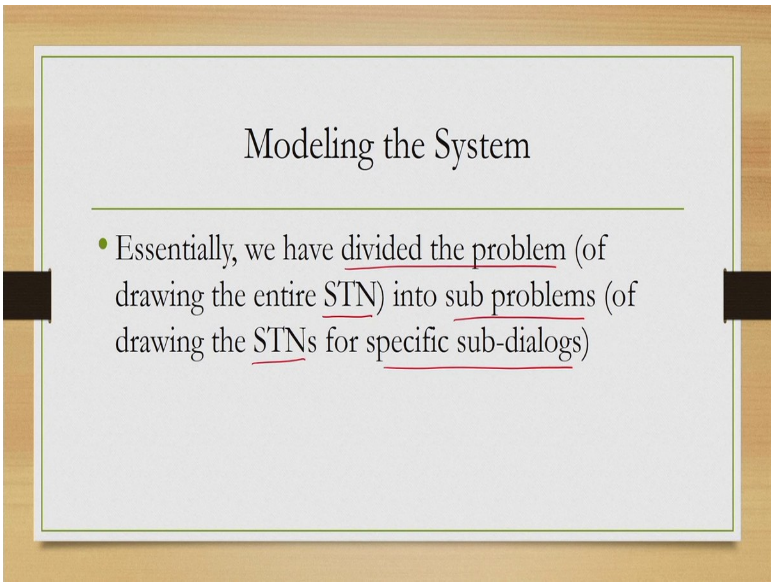


Modeling the System

- For the letter sub-dialog model, there are 26 states and 52 transitions (arcs)
- In the model for font size, we have eight states and sixteen arcs
- Manageable numbers !!

Now, if we are trying to do that, it may be noted that the for letter sub dialog there are 26 states each state corresponds to one of the alphabetic letter. And in between them, there will be 52 transitions or arcs, for font size model since there are 8 font sizes to choose from we can have eight states and sixteen arcs. These are still manageable compared to 416 states and arcs in between.

(Refer Slide Time 16:07)

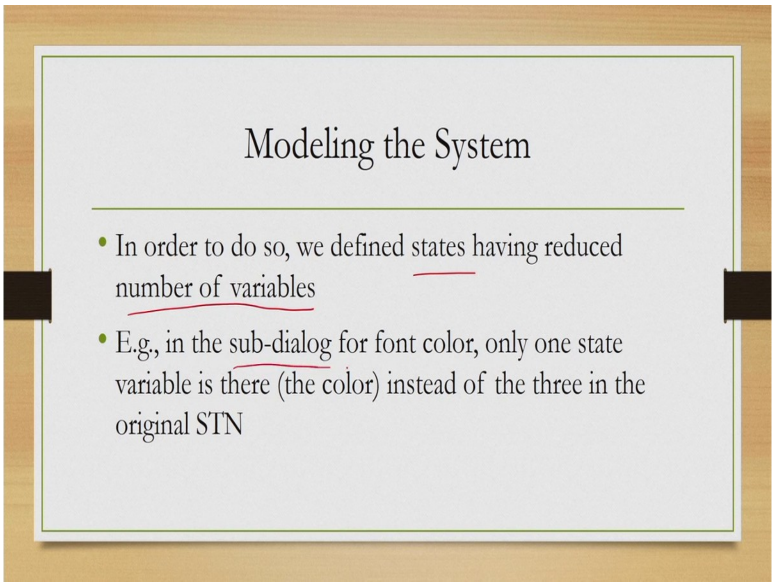


Modeling the System

- Essentially, we have divided the problem (of drawing the entire STN) into sub problems (of drawing the STNs for specific sub-dialogs)

So, what we have essentially done? We have divided the problem into sub problems. So, our entire problem was to draw the STN for the entire system, Now, we have divided it into sub problems that is to draw STNs for specific sub dialogues. And then we have connected the sub dialogues with fewer arcs those are higher level arcs or transitions.

(Refer Slide Time 16:36)



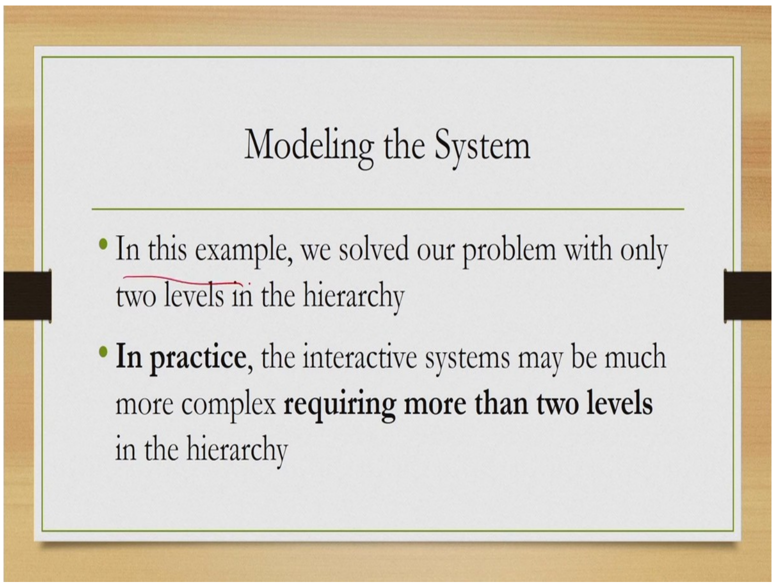
Modeling the System

- In order to do so, we defined states having reduced number of variables
- E.g., in the sub-dialog for font color, only one state variable is there (the color) instead of the three in the original STN

Now, when you say that we reduced the complexity of the problem by dividing it into sub problems, what it means? It means that, we have defined states that are having reduced number of variables hence they have become manageable. For example, the sub dialog for font color has only one state variable which is color.

Unlike in the previous case where we were trying to model the whole system with the single STN where we have defined three state variables for each state which led to 416 combinations. Now, if we have only one variable which can take only two values then there are only two states. So, we will be able to manage this sub dialog model in a much less space.

(Refer Slide Time 17:31)

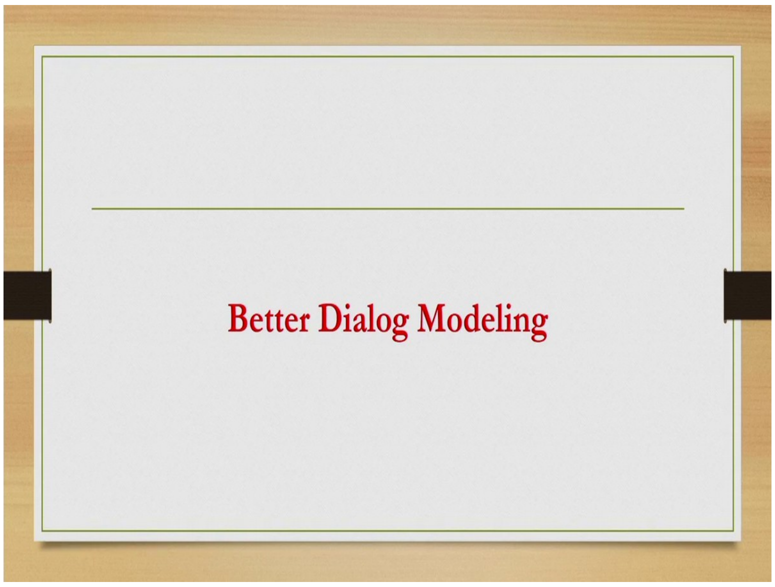


Modeling the System

- In this example, we solved our problem with only two levels in the hierarchy
- **In practice**, the interactive systems may be much more complex **requiring more than two levels** in the hierarchy

Now, in this example we have solved our problem with only two levels in the hierarchy; however, in practice the systems are much more complex than the example system. As I am repeatedly saying and there are two levels may not be sufficient and you probably have to go for many more levels and many more sub dialogues.

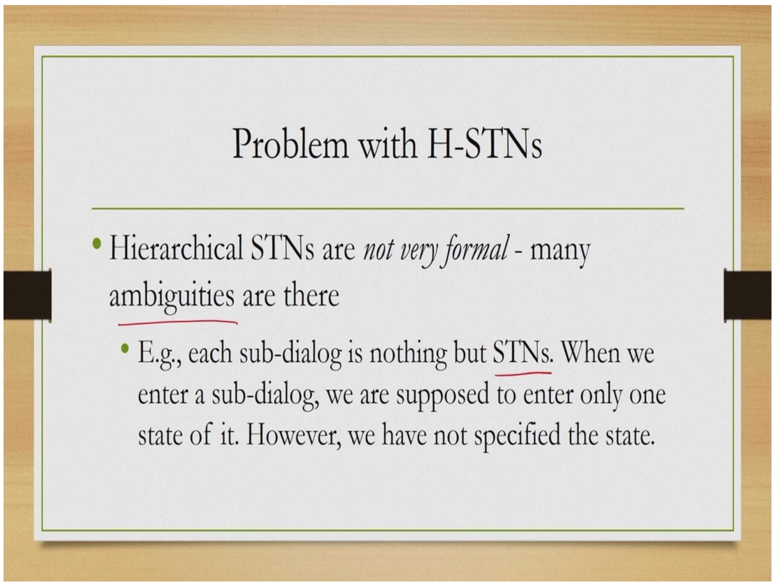
(Refer Slide Time 17:57)



Better Dialog Modeling

Can we do any better? So, whatever we have seen with the hierarchical STN, it is sufficient for giving us a nice and clean model of our example system, but is there any issue with this model?

(Refer Slide Time 18:14)



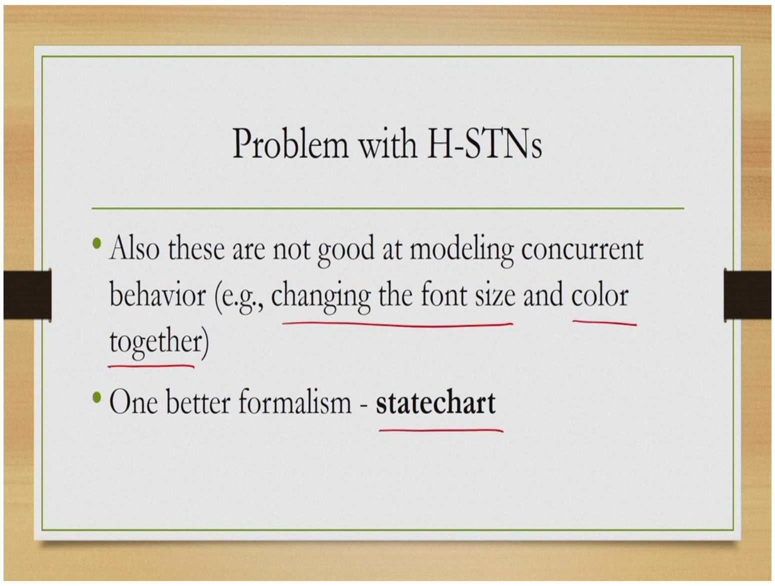
Problem with H-STNs

- Hierarchical STNs are *not very formal* - many ambiguities are there
  - E.g., each sub-dialog is nothing but STNs. When we enter a sub-dialog, we are supposed to enter only one state of it. However, we have not specified the state.

Actually there are few issues, hierarchical STN all though formal, but still there is some ambiguity. For example, when we say that we have sub dialogues and there are higher level connections each sub dialog is essentially STNs in itself.

Now, when we enter a state transition network we are supposed to enter at a specific state in the hierarchical STN notation, we have no way to actually inform which state should be entered first. When the whole sub dialog is entered for the first time or at any instant of interaction when a sub dialog is entered which state to be entered which state of that particular sub dialog to be entered? The hierarchical STN language does not support that level of specification.

(Refer Slide Time 19:16)

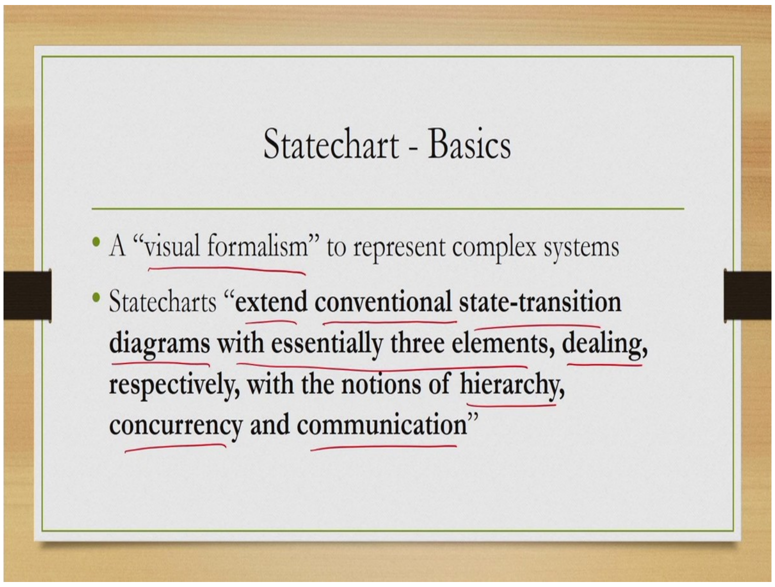


Problem with H-STNs

- Also these are not good at modeling concurrent behavior (e.g., changing the font size and color together)
- One better formalism - **statechart**

The language also is not good at modeling concurrent behavior. For example, if we want to change the font size and color together it is not clear how to represent that behavior with the notion of a hierarchical STN. There are a few more issues; these two are among the major issues namely ambiguity and lack of support for concurrence. Now, we can go for better formalism, than the hierarchical state transition network which is called state chart. So, we will have a brief look at the idea of state chart and how it can be used to model dialogues for interactive systems.

(Refer Slide Time 20:03)

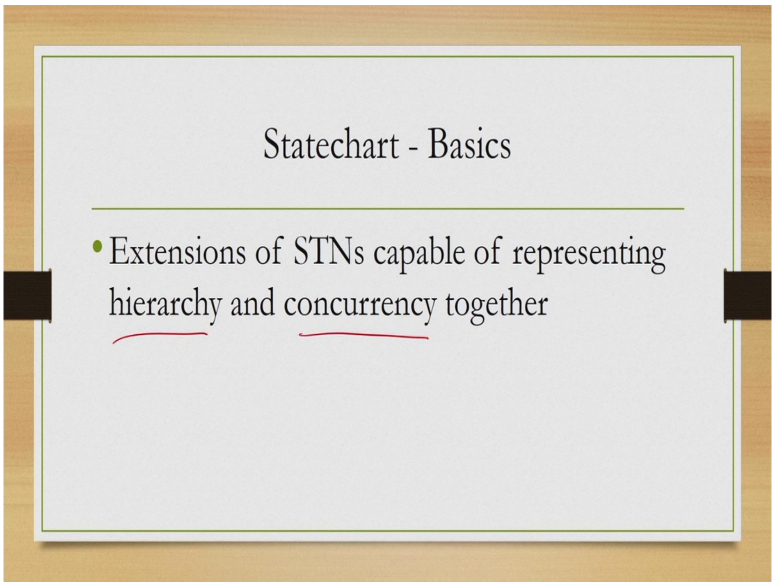


Statechart - Basics

- A "visual formalism" to represent complex systems
- Statecharts "**extend conventional state-transition diagrams with essentially three elements, dealing, respectively, with the notions of hierarchy, concurrency and communication**"

So, what is a state chart? It is kind of visual formalism to represent complex systems. And it extends conventional state transition diagrams or STNs with essentially three elements dealing with hierarchy, concurrency and communication. However, we will not cover all the things in the state chart, we will only cover how it deals with hierarchy and concurrency mostly. So, state charts are nothing, but state transition networks in a modified form.

(Refer Slide Time 20:42)

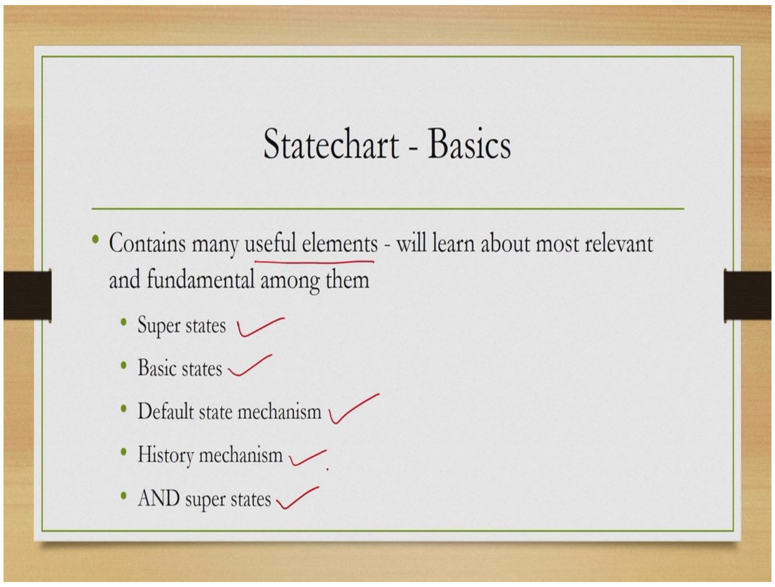


Statechart - Basics

- Extensions of STNs capable of representing hierarchy and concurrency together

And they are capable of representing hierarchy and concurrency together that is the main advantage of having a state chart notation over hierarchical state transition networks.

(Refer Slide Time 20:57)

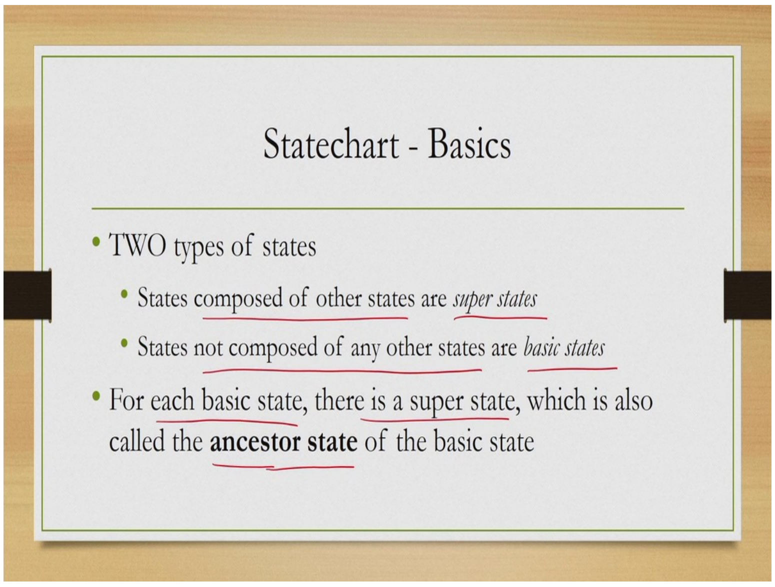


Statechart - Basics

- Contains many useful elements - will learn about most relevant and fundamental among them
  - Super states ✓
  - Basic states ✓
  - Default state mechanism ✓
  - History mechanism ✓
  - AND super states ✓

Now, state chart notation or state chart formalism actually contains many useful elements, but we will focus on few of them which are very fundamental; namely super state, basic state, default state mechanism, history mechanism and super state. So, let us see one by one what these concepts mean. So, state chart as we mentioned is extension of state transition networks.

(Refer Slide Time 21:35)



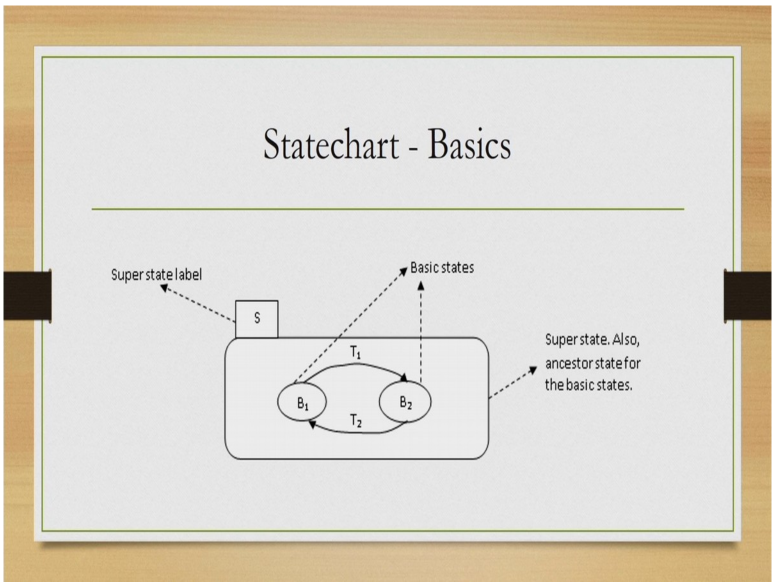
Statechart - Basics

- TWO types of states
  - States composed of other states are *super states*
  - States not composed of any other states are *basic states*
- For each basic state, there is a super state, which is also called the **ancestor state** of the basic state

So, it consists of states and transitions. What these states represent? There are two types of states there are some states that are composed of other states and these are called super states. There are some states which are not composed of any other state these are called basic states. For each basic state there is a super state and this super state is also known as ancestor state.

So, to recap there are two types of states some are composed of other states those are called the super states, some are not composed of other states those are called basic states and for each basic state there is always a super state this super state is sometimes called the ancestors state of that particular basic state.

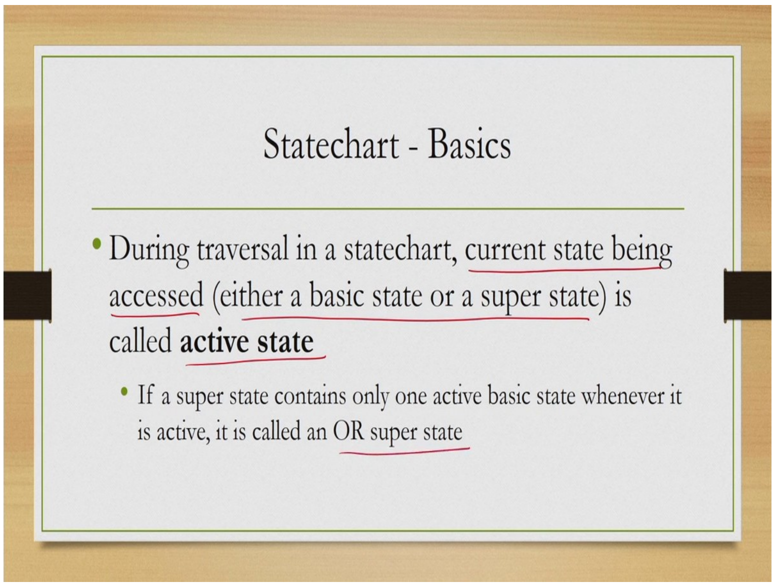
(Refer Slide Time 22:31)



Statechart - Basics

The idea is illustrated in this diagram, here as you can see this is the typical convention used to represent super state basic state and transitions. So, B 1 and B 2 are the basic states and we can define transitions between them as shown here T 1 and T 2 the state that contains these two states B 1 and B 2 is the super state which we can label with the name here the label S is given to this super state. Now, this super state S is the ancestors state for both B 1 and B 2. So, S is the super state which is also ancestor state for B 1 and B 2 B 1 and B 2 are the basic states.

(Refer Slide Time 23:34)

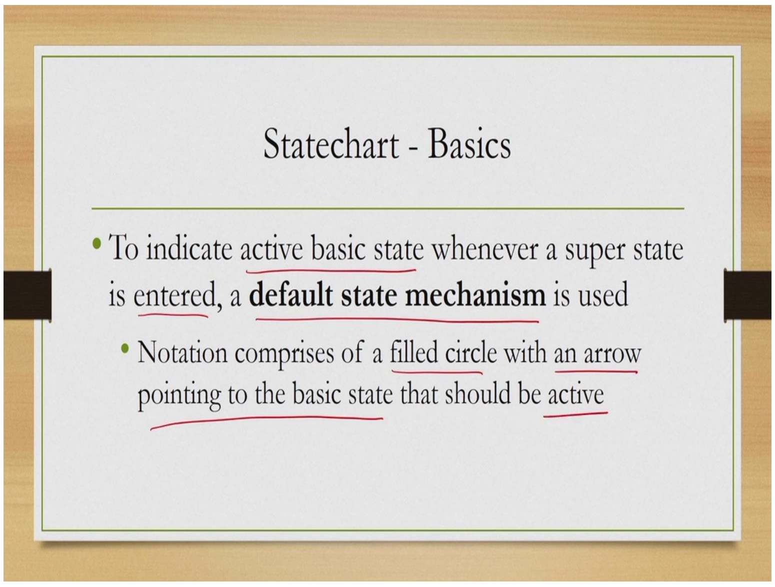


Statechart - Basics

- During traversal in a statechart, current state being accessed (either a basic state or a super state) is called **active state**
  - If a super state contains only one active basic state whenever it is active, it is called an OR super state

Now, state transition networks are meant for transitioning from one state to another which is called state traversal. During the state traversal in a state chart formalism, the current state being accessed either a basic state or a super state is called active state. Now, if a super state contains only one active basic state at a time whenever it is active it is called OR super state.

(Refer Slide Time 24:13)

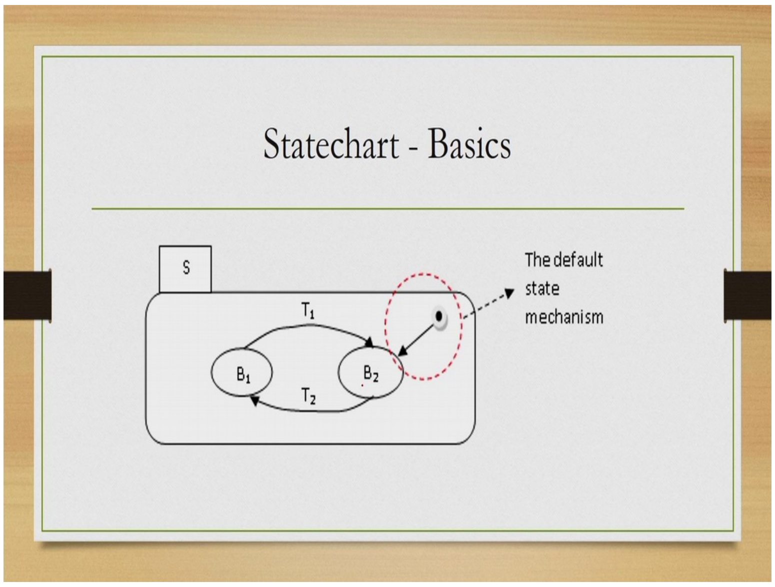


Statechart - Basics

- To indicate active basic state whenever a super state is entered, a **default state mechanism** is used
  - Notation comprises of a filled circle with an arrow pointing to the basic state that should be active

To indicate the active basic state whenever a super state is entered, there is a notation which is called default state mechanism which is used to indicate the behavior. Now, this notation comprises of a field circle with an arrow pointing to the basic state that should be active let us try to understand it with an illustration.

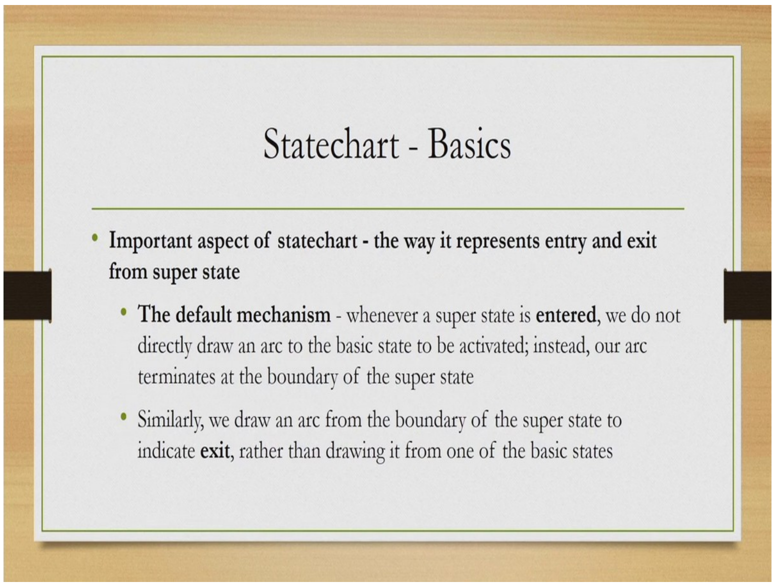
(Refer Slide Time 24:49)



Statechart - Basics

So, in this figure the concepts are highlighted. Now, this field circle with arrow which is shown within this red dotted circle is the default state mechanism that we have just mentioned. So, whenever the superstate S is active, in the overall state chart then the default state mechanism indicates that B 2 should become active.

(Refer Slide Time 25:27)

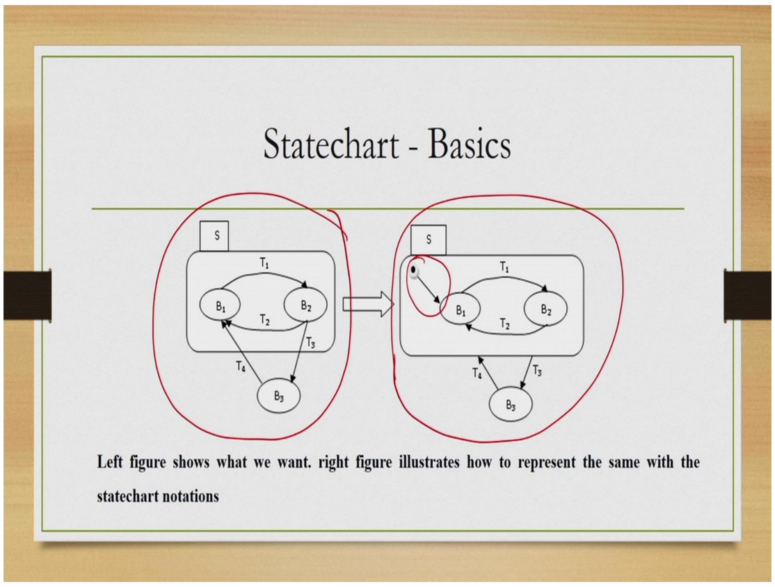


Statechart - Basics

- Important aspect of statechart - the way it represents entry and exit from super state
  - **The default mechanism** - whenever a super state is **entered**, we do not directly draw an arc to the basic state to be activated; instead, our arc terminates at the boundary of the super state
  - Similarly, we draw an arc from the boundary of the super state to indicate **exit**, rather than drawing it from one of the basic states

One important aspect of a state chart is that the way it represents entry and exit from super state. So, default mechanism we have already talked about. There is another convention use that is we typically draw an arc from the boundary of the super state to indicate exit rather than drawing it from one of the basic states. So, when we want to indicate exit from the super state we do not directly draw an arrow or an arc from a particular basic state to outside of the super state. Instead from the boundary of the super state we rather arc.

(Refer Slide Time 26:10)



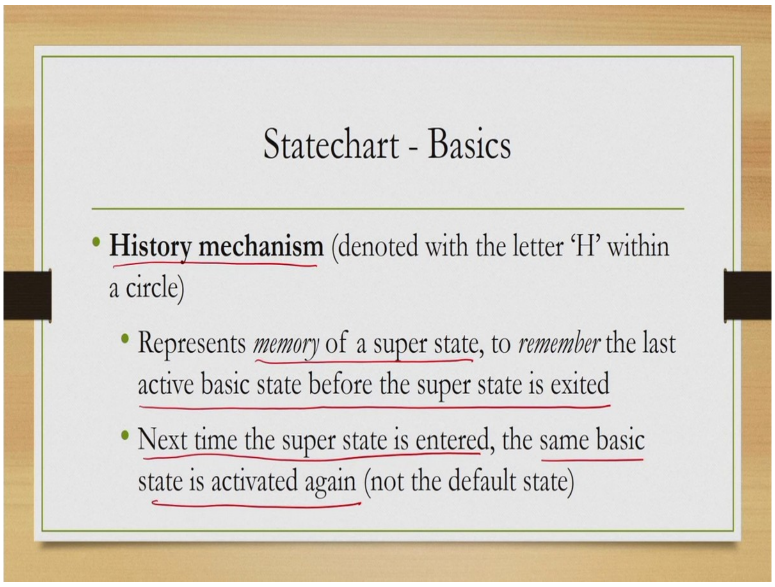
Statechart - Basics

Left figure shows what we want. right figure illustrates how to represent the same with the statechart notations

The idea is illustrated in this figures. So, in the left side what we have is what exactly we want that is we have a super state S 2 basic states. Now, from B 2 we want to go to B 3 and from B 3 we want to go to B 1. So, how to denote this using state chart notation that is shown in the right hand figure.

Here as you can see the arcs are not directly drawn from B 3 to B 1 and B 2 to B 3 unlike in the left figure. Here from B 3 the arcs are extended up to the boundary of the super state the arcs T 3 and T 4 inside the super state we have mentioned this default mechanism that is whenever the super state is entered B 1 should get active. So, it is essentially same as telling that from B 3 if we entered the super state we will go to B 1. So, in this way we can indicate how to specify the entry and exit to a super state.

(Refer Slide Time 27:31)

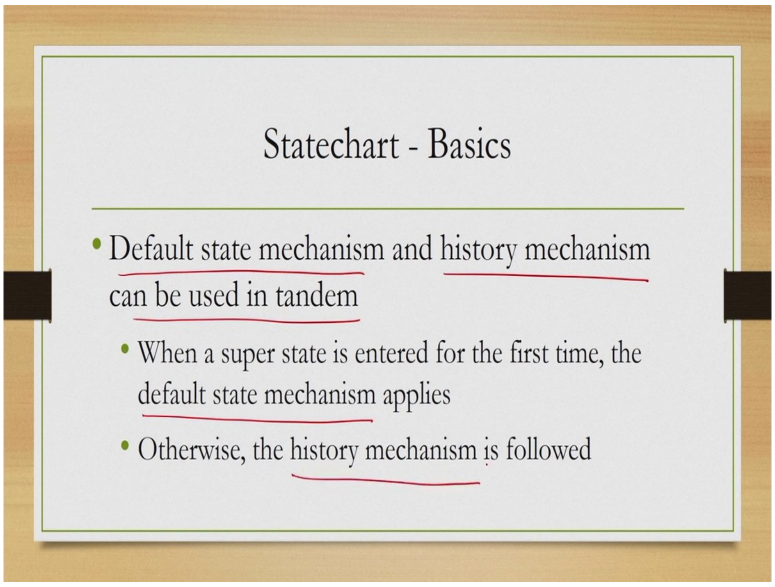


Statechart - Basics

- **History mechanism** (denoted with the letter 'H' within a circle)
  - Represents *memory* of a super state, to *remember* the last active basic state before the super state is exited
  - Next time the super state is entered, the same basic state is activated again (not the default state)

There is another useful notation that is called history mechanism, which essentially indicates a memory of the super state that is the basic state or the active basic state before the super state is exited. So, last time what was the active basic state before we exited the super state. Now, why these memory is important? It ensures that next time the super state is entered the same basic state is activated again. So, we do not need to start from the beginning or the default state.

(Refer Slide Time 28:11)

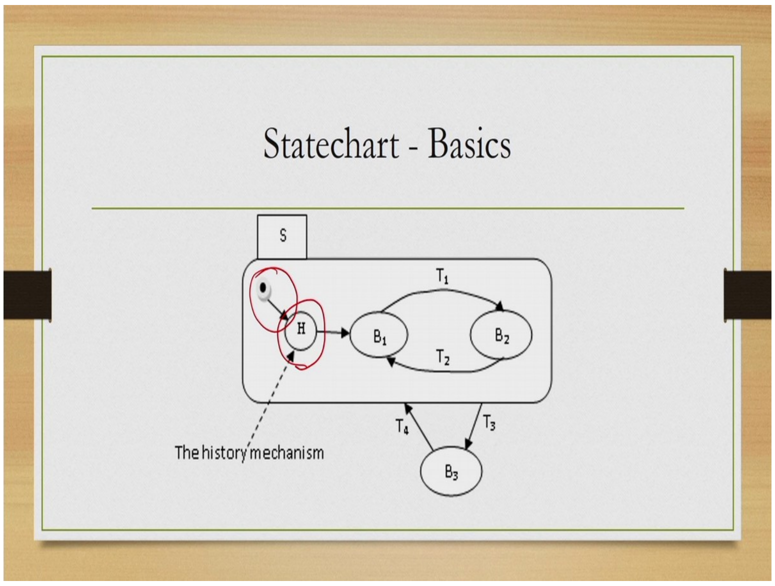


Statechart - Basics

- Default state mechanism and history mechanism can be used in tandem
  - When a super state is entered for the first time, the default state mechanism applies
  - Otherwise, the history mechanism is followed

Default state mechanism and the history mechanism can be used in tandem. So, we can use them together to indicate this behavior at the start of the state traversal and in the middle of state traversal. So, at the beginning we can apply default state mechanism and at any instant during the state traversal we can rely on the history mechanism.

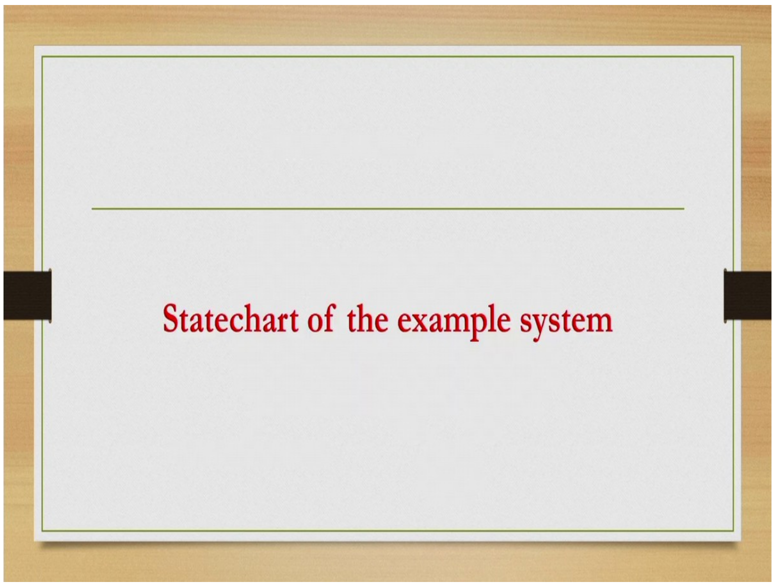
(Refer Slide Time 28:46)



Statechart - Basics

The idea is illustrated in this figure, where we are combining the default mechanism with the history mechanism. So, H within this circle indicates history and the default mechanism we have already seen. So, in this super state both the mechanisms are specified; that means, when the super state S is entered for the first time it starts with the default mechanism. So, it starts with B 1; if it exits then it remembers from where it exited and it starts from that state next time it is entered.

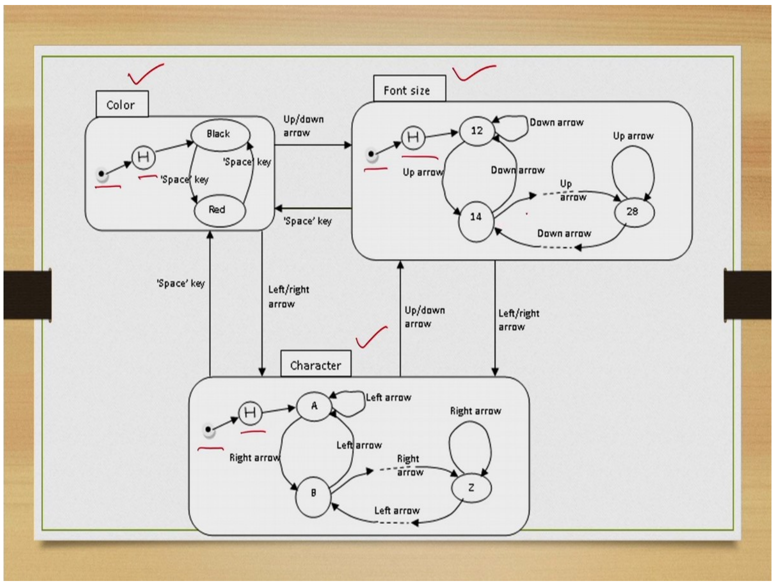
(Refer Slide Time 29:30)



Statechart of the example system

So, with this notation let us now try to see how we can build a state chart for our example system. So, remember that in our example system, we have 416 states. Now, earlier we have seen how to use the hierarchical STN to divide the whole system into subsystems and come up with a hierarchical specification of the whole system. Now, we are going to use the state chart notation to represent the same system.

(Refer Slide Time 30:03)

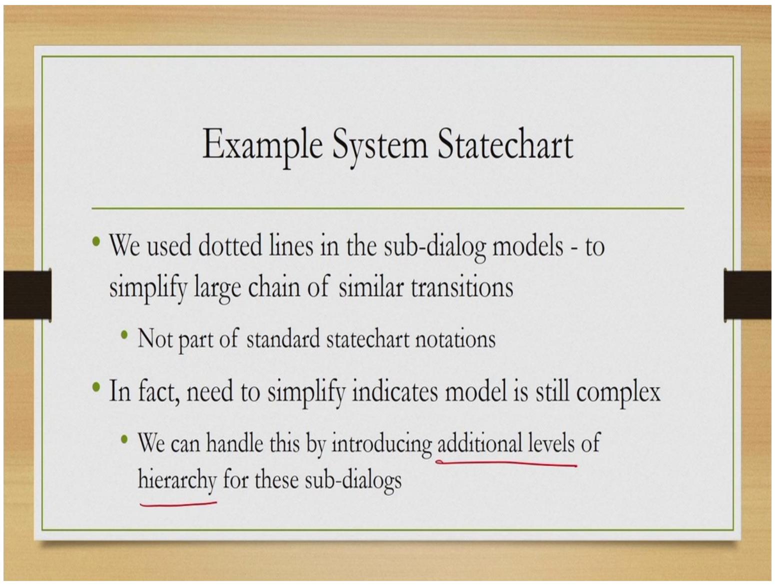


This figure shows a state chart representation of the character display system which we have described earlier. Here as you can see there are three super states color font size and character. So, color super state or a font size super state or character super state contains many basic states. Here we have mentioned that all the super states combined both the default state mechanism and the history mechanism. So, these two mechanisms are present in all the super states.

Now, one thing maybe noted here that in font size super state or character super state we have used some state transition networks with dotted lines; this is not part of the convention. So, typically this is not used in state chart notation, here we have used it to simplify the depiction of a large chain of transitions for font size as well as the character. Since we have used the dotted line it also indirectly indicates that the things are still not manageable we can still go for improvement. So, possibly we may go for some further hierarchy in this case.

So, how the hierarchy is taken care of by defining super states and defining connections between the super states and within each super state like before we have the sub dialogues modeled using STNs; with the added advantage that this time we have specified the history mechanism and default state mechanisms. So, now, we know if one super state is entered say for example, character then where it should start? So, as per the default mechanism so the state A should start whenever the character super state is entered.

(Refer Slide Time 32:23)



Example System Statechart

- We used dotted lines in the sub-dialog models - to simplify large chain of similar transitions
  - Not part of standard statechart notations
- In fact, need to simplify indicates model is still complex
  - We can handle this by introducing additional levels of hierarchy for these sub-dialogs

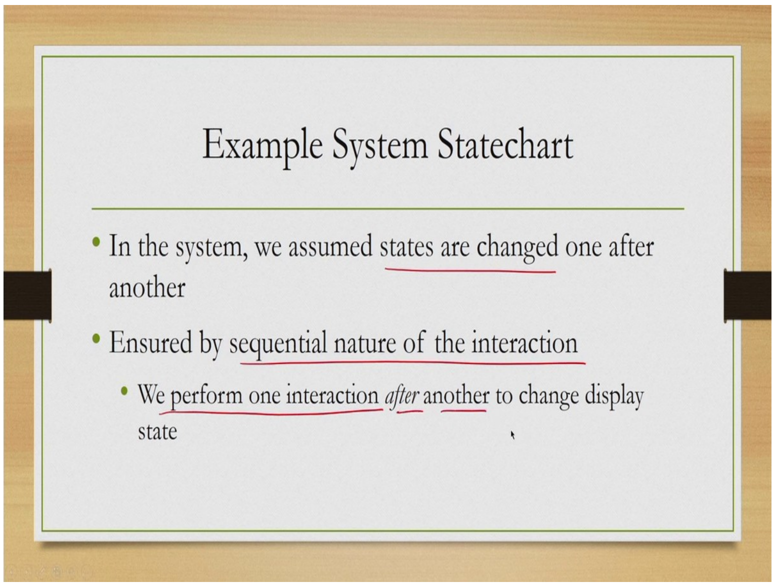
As I said since we have used dotted lines; that means, we are still unable to show the entire system within a single page. So, there is possibly scope for more simplification by adding additional levels of hierarchy or additional levels of super states that is one advantage of using state chart that is it allows us to represent hierarchy. But at the same time it allows us to specify the entry exit points in the hierarchy which is not possible with hierarchical STNs there is another advantage.

(Refer Slide Time 33:09)

Statechart & Concurrency

Many interactive systems are actually concurrent in nature, the way they behave can be termed as concurrent behavior.

(Refer Slide Time 33:17)

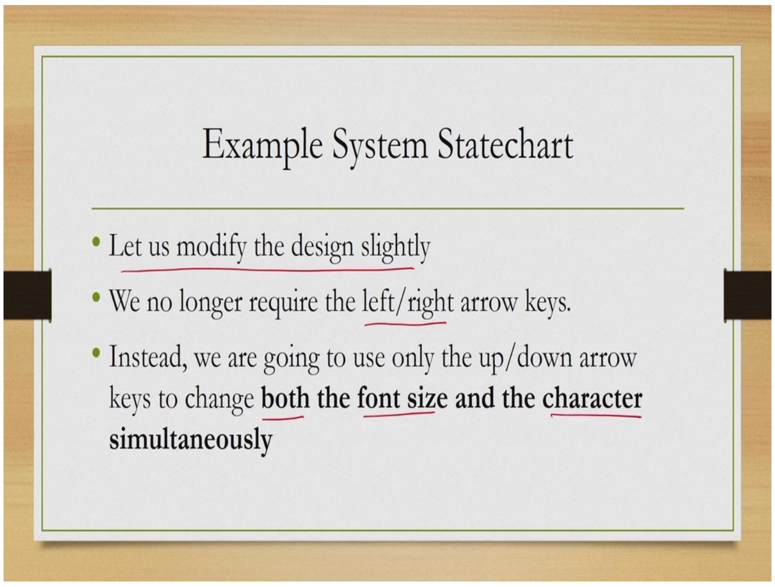


Example System Statechart

- In the system, we assumed states are changed one after another
- Ensured by sequential nature of the interaction
  - We perform one interaction *after* another to change display state

What do you mean by that? In the example system, we assume that the states are changed one after another which was ensured by sequential nature of the interaction. So, we performed the interaction one after another to change the display state.

(Refer Slide Time 33:38)

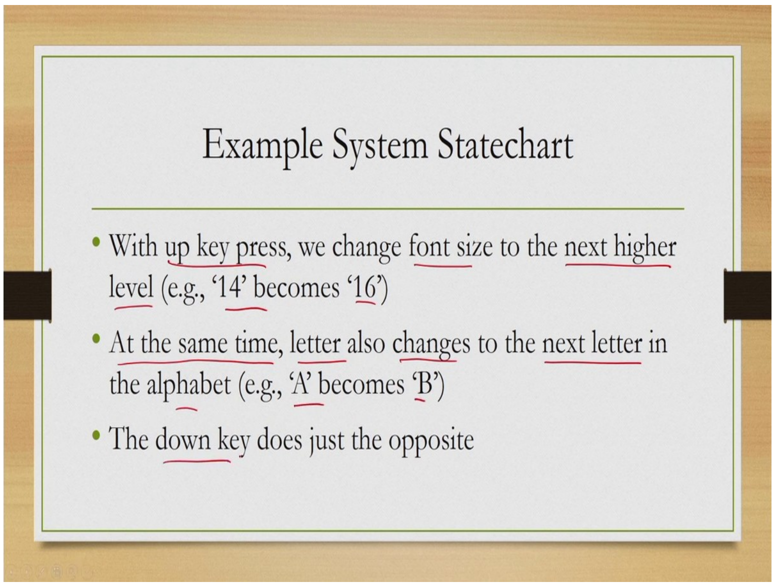


Example System Statechart

- Let us modify the design slightly
- We no longer require the left/right arrow keys.
- Instead, we are going to use only the up/down arrow keys to change **both the font size and the character simultaneously**

Let us now, consider it different design let us modify the design slightly, suppose we are dispensing with the left right arrow key. So, what we are left with is only the up down arrow key. Now, this arrow key we are using to change both font size and character how that is possible? Let us see.

(Refer Slide Time 34:05)

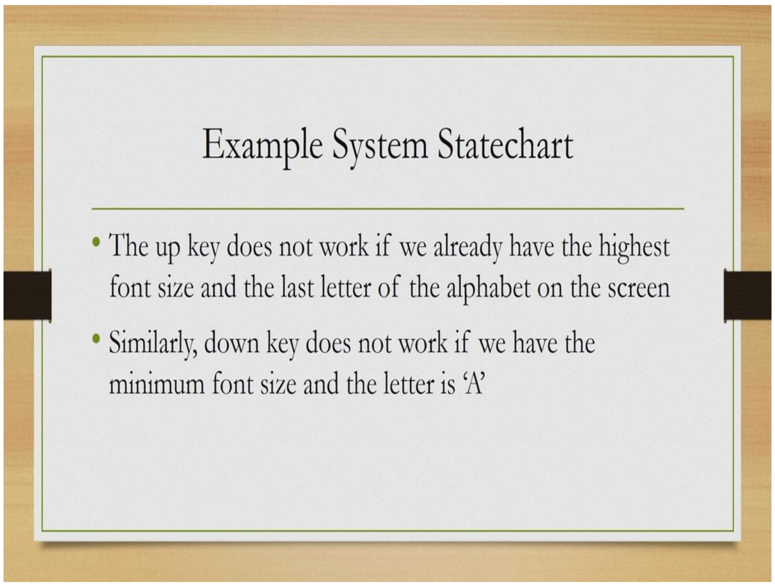


Example System Statechart

- With up key press, we change font size to the next higher level (e.g., '14' becomes '16')
- At the same time, letter also changes to the next letter in the alphabet (e.g., 'A' becomes 'B')
- The down key does just the opposite

If we press the up key then what happens is that, the font size change to the next higher level that is 14 becomes 16 and at the same time simultaneously the letter also changes to the next letter in the alphabet that is A becomes B and the down key does exactly the opposite of this.

(Refer Slide Time 34:30)

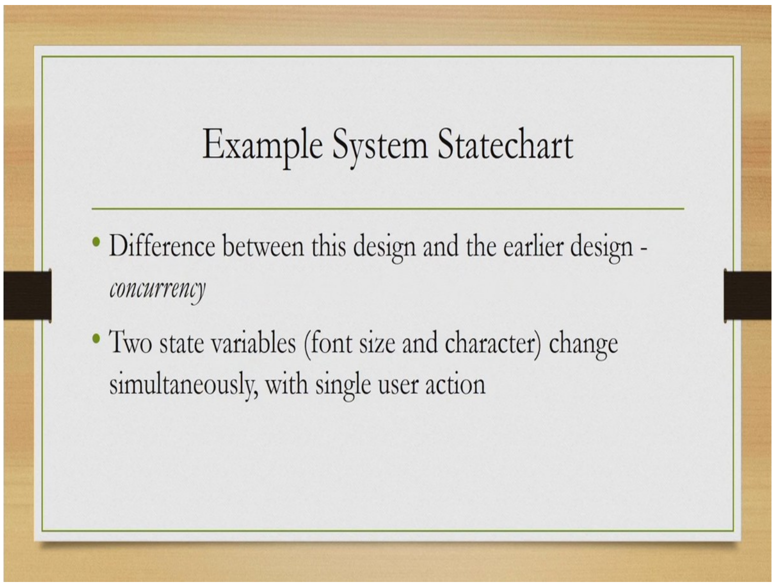


Example System Statechart

- The up key does not work if we already have the highest font size and the last letter of the alphabet on the screen
- Similarly, down key does not work if we have the minimum font size and the letter is 'A'

So we can of course, add certain constraints like the up key or the down key does not work at the boundaries. So, when we have reached the maximum font size or the last letter of the alphabet the up key will not work similarly the down key should not work when we have reached the minimum font size or the first letter of the alphabet.

(Refer Slide Time 34:53)

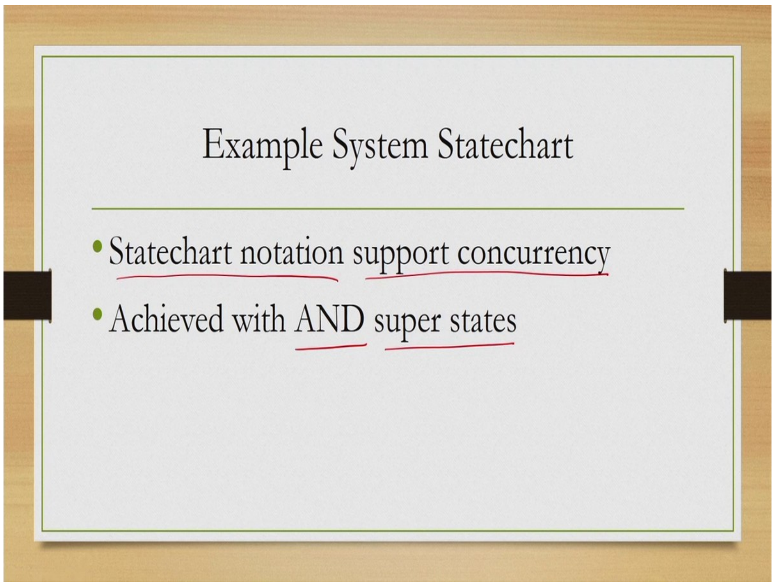


Example System Statechart

- Difference between this design and the earlier design - *concurrency*
- Two state variables (font size and character) change simultaneously, with single user action

So, what is the difference between this design and the previous design? Remember that in the previous design we can actually change font size and letter one after another by using different keys, but here that is not the case here both are changing together. Now, the earlier hierarchical STN model of our system or the state transition networks or the state chart model none of these models of our system actually capture this concurrent behavior. And we need to modify the language actually to specify this concurrent behavior.

(Refer Slide Time 25:30)

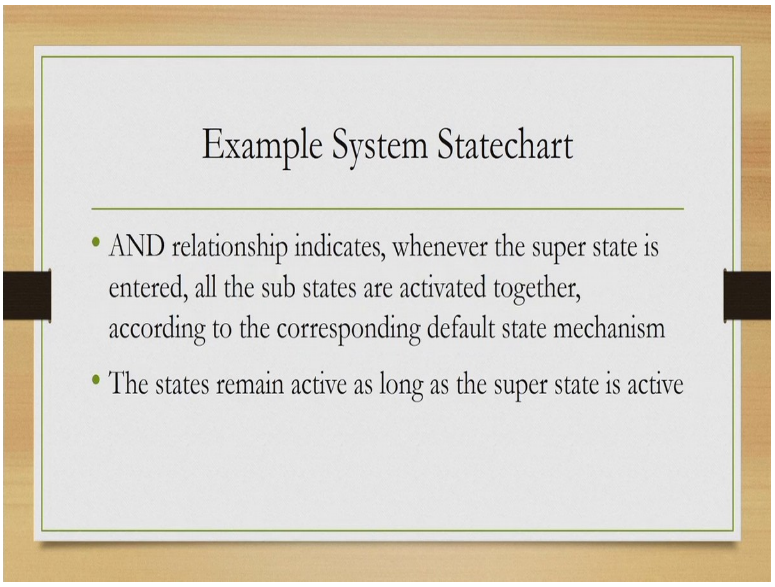


Example System Statechart

- Statechart notation support concurrency
- Achieved with AND super states

Whatever we have learned so, far about state chart is not sufficient to help us resolve this issue. However, it may be noted that state chart notation supports concurrency and that is achieved with the idea of the AND super states. And we need this notation to specify concurrency in our proposed model of the system.

(Refer Slide Time 35:52)

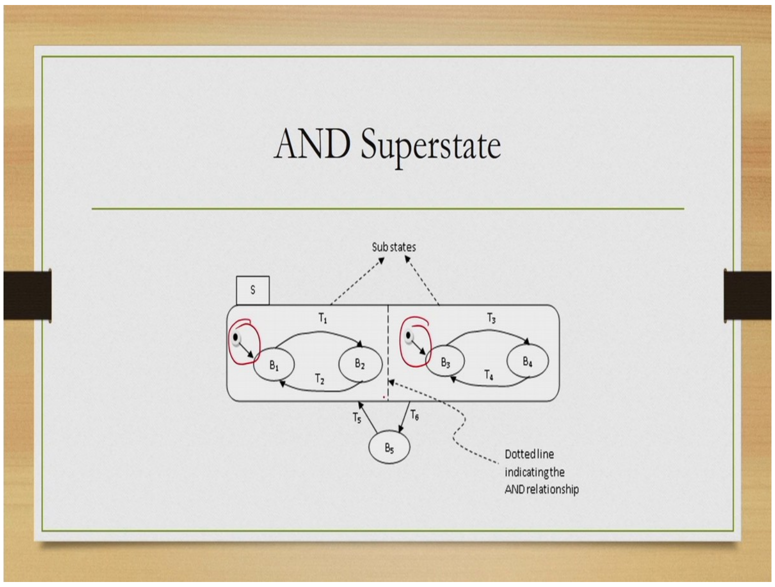


Example System Statechart

- AND relationship indicates, whenever the super state is entered, all the sub states are activated together, according to the corresponding default state mechanism
- The states remain active as long as the super state is active

So, first let us try to understand what is this AND super state. The AND relationship between super states indicates that, whenever the super state is entered, all the sub states are activated together according to the corresponding default state mechanism and the states remain active as long as the super state is active.

(Refer Slide Time 36:25)

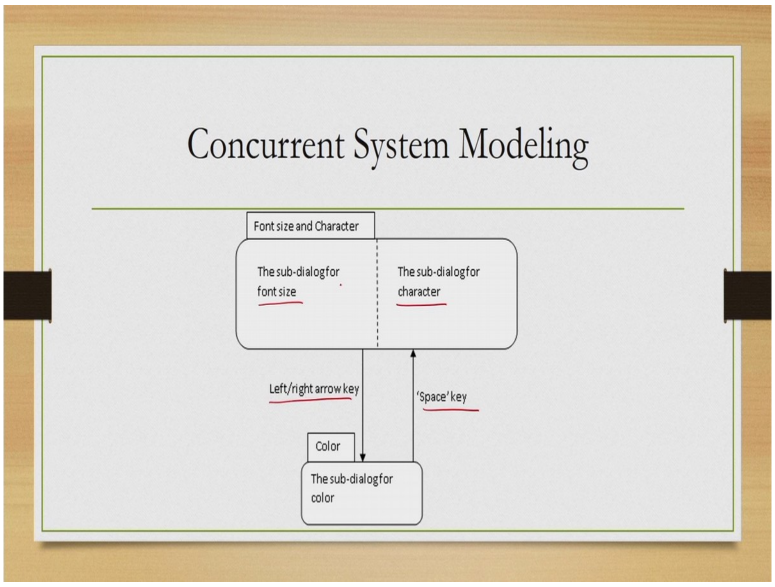


AND Superstate

The figure illustrates the idea. Now, S is a super state it has two sub states T 1 and T 3 each of these can have a sub dialog models as shown here. Now, whenever S is entered both T 1 and T 3 are active simultaneously. Now, what I mean by the term active? That means, whenever they are entered the basic state that is indicated by the default mechanism becomes active together. So, whenever S is entered B 1 and B 3 get active together because the default mechanism indicates.

So, in the case of T 1 the default mechanism says B 1 should be the first state or default state in case of T 3 its states that B 3 should be the default state. So whenever S is entered, both these default states get active together. And the dotted line that separates the two sub states T 1 and T 3 indicates and relationship. So, with this notation we can represent concurrent behavior; here T 1 and T 3 are concurrently active. Now, with this notion of concurrency, we can try to represent the character display system in terms of state chart notation it will look something like this.

(Refer Slide Time 37:58)


Concurrent System Modeling

Here now, we have these two sub states the sub dialog for font size and sub dialog for character they are now, connected by the AND super state notation. So, this font size and character is the super state and it is an AND super state where the sub states are font size and character they are active simultaneously. Then the other super state is color and these AND super state and the colors super state connected with left right arrow key press and space key.

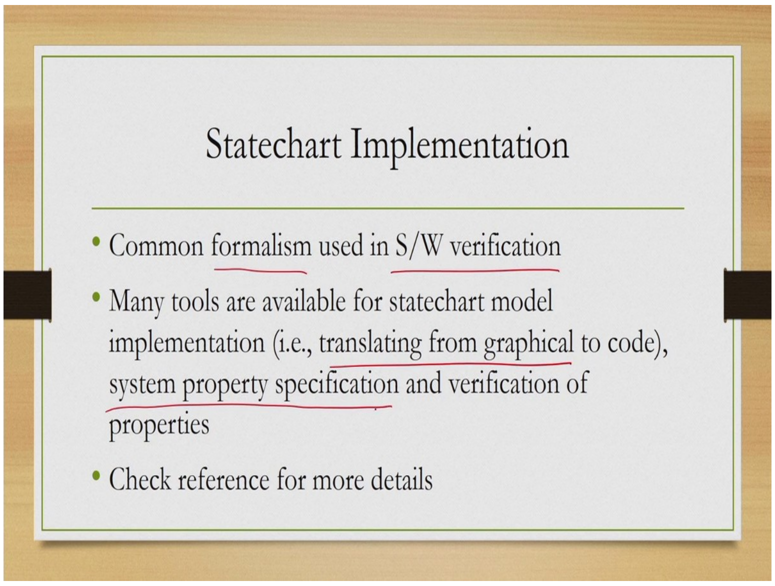
So, it is as you can see it is in a much simpler form then the earlier one where concurrency was not taken into consideration. And as before the sub dialogues can be represented with STNs; however, there is some change. Now, actually to traverse between the sub states in the sub dialogues, we would no longer need the up down arrow key or would no longer need the one set of arrow keys. So, that should be taken into account while modeling the sub dialog for these sub states font size and character. So, that is all about the basic idea how we can use the formal notation to model an interactive system.

So, here we have shown how to use state charts hierarchical STN and state transition networks to model interactive systems. So, first we started with state transition networks and we have found that it is not suitable to model complex large practical systems. So, then we learned about hierarchical state transition networks, where we divide the problem into sub problems and create a hierarchy of sub problems to solve the overall problem. Now, there is some elements of ambiguity in this hierarchical state chart and also it does not support concurrency in order to handle those things.

While retaining the benefit of hierarchical representation, we learned about state chart notations. And we have seen how we can use it to represent complex systems both supporting sequential behavior as well as concurrent behavior. The question is so, that is how we actually represent it. Now, what is next is it only to draw it on a paper that we learn these models these formalisms that is not necessarily that is actually not the purpose.

So, when we say we want to formally represent a system definitely our objective is to be able to encode it in some language. So, that it can be processed by computer in other words we can create a software out of it.

(Refer Slide Time 41:40)



Statechart Implementation

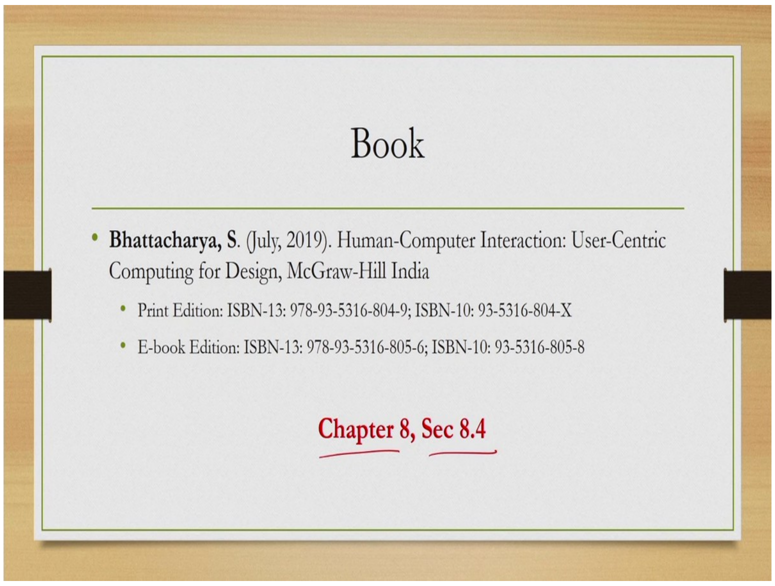
- Common formalism used in S/W verification
- Many tools are available for statechart model implementation (i.e., translating from graphical to code), system property specification and verification of properties
- Check reference for more details

So, there are tools to do that. In fact, state chart notation is a very common formalism used in software property verification. And accordingly there are many tools available for model specification or model implementation; that means, translating the graphical representation to a code form program form. Then there are tools available to specify system properties.

So, state chart does not specify any property state chart is a representation of the system and we can actually verify this model the state chart model to check for presence or absence of system properties. So, what we need is to specify those properties as well and together the specification of the model and the properties can be used for verification.

So, there are tools available do all these things to specify the state chart model to specify properties and to verify those properties. However, we will not go into the details of those are not in the scope of this lecture, but if you are interested you can check the reference that will be mentioned at the end of this lecture.

(Refer Slide Time 43:08)



Book

- **Bhattacharya, S**. (July, 2019). Human-Computer Interaction: User-Centric Computing for Design, McGraw-Hill India
  - Print Edition: ISBN-13: 978-93-5316-804-9; ISBN-10: 93-5316-804-X
  - E-book Edition: ISBN-13: 978-93-5316-805-6; ISBN-10: 93-5316-805-8

Chapter 8, Sec 8.4

So, whatever we have discussed today are taken from this book. So, you can refer to chapter 8, section 8.4; along with that you can refer to other sections if you are interested to learn about. How to use the specification, how to use the models, what are the tools available you can get some introductory idea on these basic concepts. That is all for today.

Thank you and goodbye.