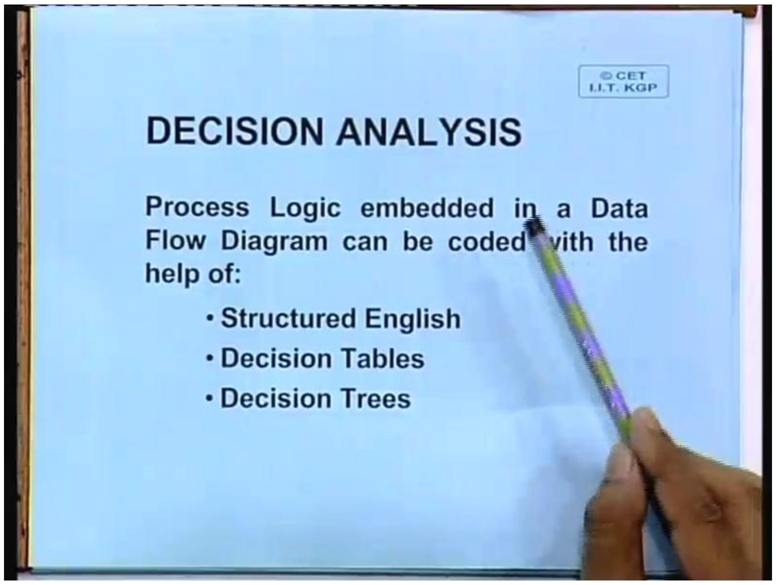
Management Information System
Prof. B. Mahanty
Department of Industrial Engineering & Management
Indian Institute of Technology, Kharagpur

Lecture No. # 13
Decision Analysis – I

Today let us begin a new chapter that is on decision analysis. See, we have seen before that after we have you know done the requirement analysis, we have to go for the data flow diagrammings and the diagrams, the data flow diagrams can hide the decision structures, all right. So, basically those process bubbles will actually include your decision structures, right. So, this process logic can, process logic which is embedded in a data flow diagram can be coded with the help of structured English, decision tables and the decision trees, right.

(Refer Slide Time: 00:02:03 min)



© CET
I.I.T. KGP

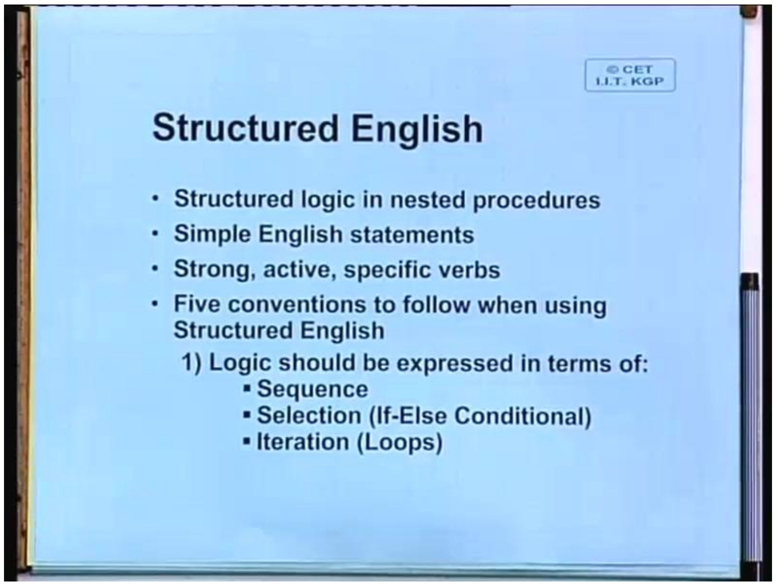
DECISION ANALYSIS

Process Logic embedded in a Data Flow Diagram can be coded with the help of:

- Structured English
- Decision Tables
- Decision Trees

So, we can say that a data flow diagrams have essentially you know the process logic which is embedded in the data flow diagrams, we can code it with the help of structured English, decision tables and decision trees. Is it okay? Now let us see the first one amongst them that is the structured English.

(Refer Slide Time: 00:02:26 min)



© CET
I.I.T. KGP

Structured English

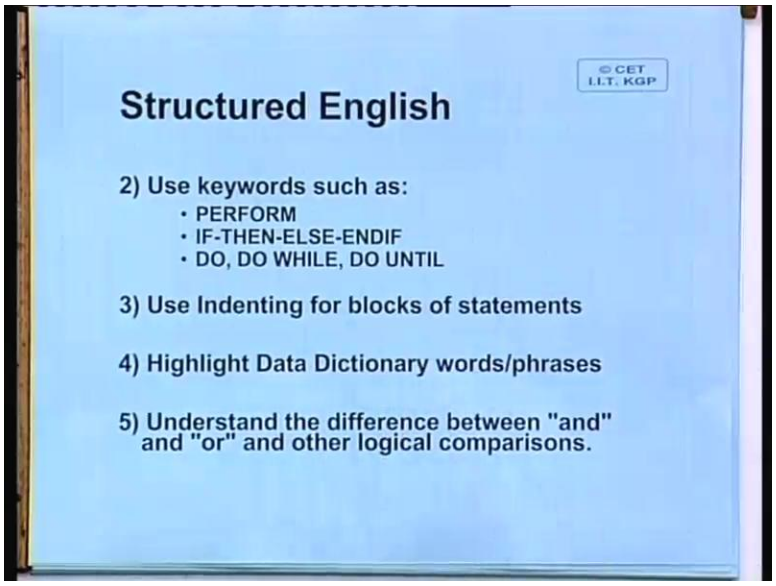
- Structured logic in nested procedures
- Simple English statements
- Strong, active, specific verbs
- Five conventions to follow when using Structured English
  1) Logic should be expressed in terms of:
     - Sequence
     - Selection (If-Else Conditional)
     - Iteration (Loops)

The structured English essentially is the structured logic in nested procedures, right. We can use simple English statements and strong active specific verbs that is basically used for this. What are these verbs? I will give examples. 5 conventions to follow when using structured English. What are those 5 conventions? The first one the logic should be expressed in terms of sequence, selection and iteration. Actually, sequence, selection and iteration are three established programming structures, right. What could be the other programming structures that we should not follow? Basically, the unconditional transfer of control flow that is the goto kind of structures.

You see, you must be aware of the goto statement. The goto statement can help you to you know move control from a particular program unconditionally to another part of the program. What happens if you use goto the programming structure or the control flow gets totally broken, gets totally broken. So, it is always advised that you should never use goto like structures. Actually, you should use only three structures, one is sequence, sequence is automatic that is any program whether in c java or any other language it should always follow step by step, one line next line like that then you can have selection if else conditional. So, if condition then do something else do something else or you know switch like structures that is one and third one iterations, loops. What is important is not that these three are there because everybody knows that these are the three basic structures but nothing else should be there that is important, all right. These three should cover all possible logic that you can have.

Then after that number two, point number two you should use keywords such as perform, if then else, endif, do, do while, do until. So, these are the three basic keyword structures, one is perform. Perform essentially means you know perform some kind of a logic. Actually in Cobol kind of language there is this perform structure is there but here the word perform we are using simply as a structured English statement, right. Then you can have if then else, endif blocks, then do, do while or do until kind of blocks. So, these are the keywords that you should be using.

(Refer Slide Time: 00:05:41 min)



© CET
I.I.T. KGP

Structured English

2) Use keywords such as:
- PERFORM
- IF-THEN-ELSE-ENDIF
- DO, DO WHILE, DO UNTIL

3) Use Indenting for blocks of statements

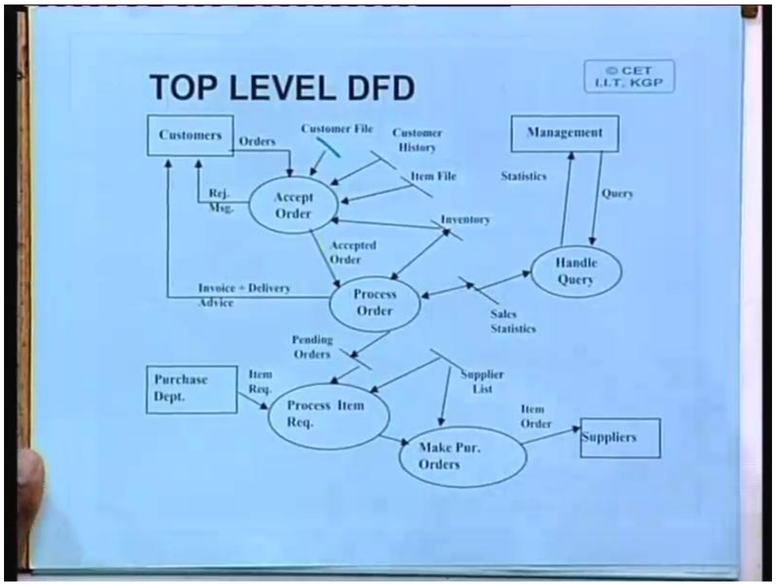
4) Highlight Data Dictionary words/phrases

5) Understand the difference between "and" and "or" and other logical comparisons.

The third one use indenting for blocks of statements this very important because if you do not indent sometimes the control flow may not be very clear. Then highlight data dictionary words or phrases. See what happens at times you may have some sort of words or phrases which you are using in your data dictionary. Basically, let’s say the customer, customer is a data dictionary word in the example that I have given, the supplier then purchase department, the name of the customer. These are all a part of data dictionary. So, we can highlight so that to differentiate it with the other words which you might be using in the basic structure.

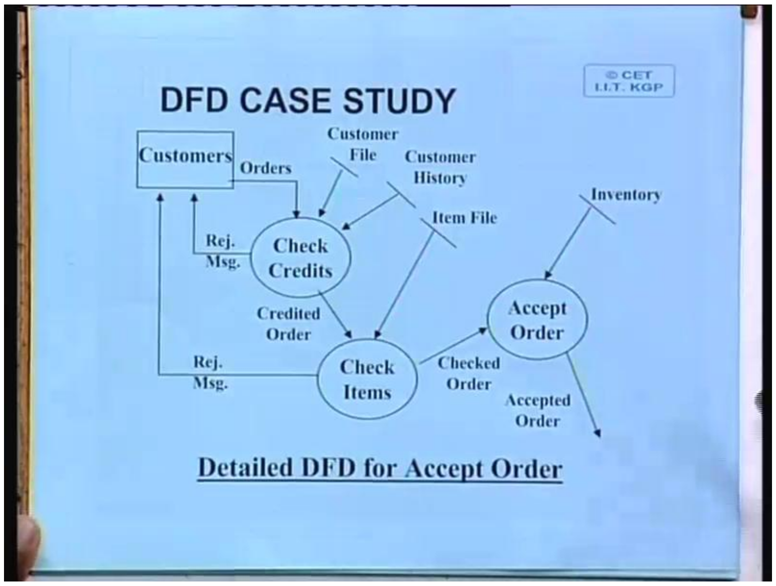
Then understand the difference between “and” and “or” and other logical comparisons for less than equal to or less than, all right. Basically, “and” and “or” they are very different and if you, you know conjunct to blocks by and essentially means both should be satisfied or means anyone may be satisfied, so it can be totally different structure all together.

(Refer Slide Time: 00:07:09 min)



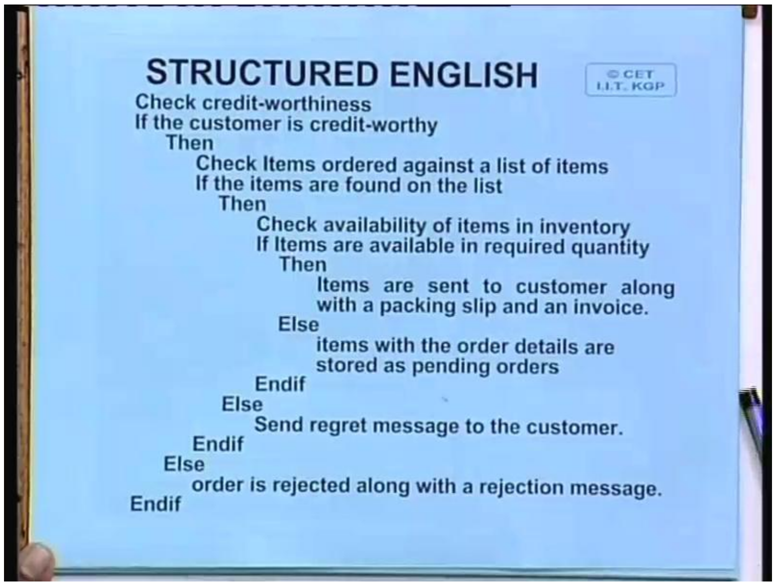
Let’s go back for the time being to our top level data flow diagram that we have discussed in our DFD case study. The customer was placing orders and we have the accept order and we check various things with the customer file, customer history file, item file and inventory file and reject message otherwise all right and this was already known to us.

(Refer Slide Time: 00:07:42 min)



From there what we had done, we had drawn the detailed data flow diagram for accept order. Basically, there are three processes, one was the check order sorry check credits from customer file and customer history otherwise reject and afterwards check items from the item file, reject message otherwise and if it is check then you accept order and put it into the inventory file as an accepted order. So, that was the detailed data flow diagram which we already have.

(Refer Slide Time: 00:08:14 min)



STRUCTURED ENGLISH

© CET I.I.T. KGP

```
Check credit-worthiness
If the customer is credit-worthy
    Then
        Check Items ordered against a list of items
        If the items are found on the list
            Then
                Check availability of items in inventory
                If Items are available in required quantity
                    Then
                        Items are sent to customer along
                        with a packing slip and an invoice.
                    Else
                        items with the order details are
                        stored as pending orders
                Endif
            Else
                Send regret message to the customer.
        Endif
    Else
        order is rejected along with a rejection message.
Endif
```

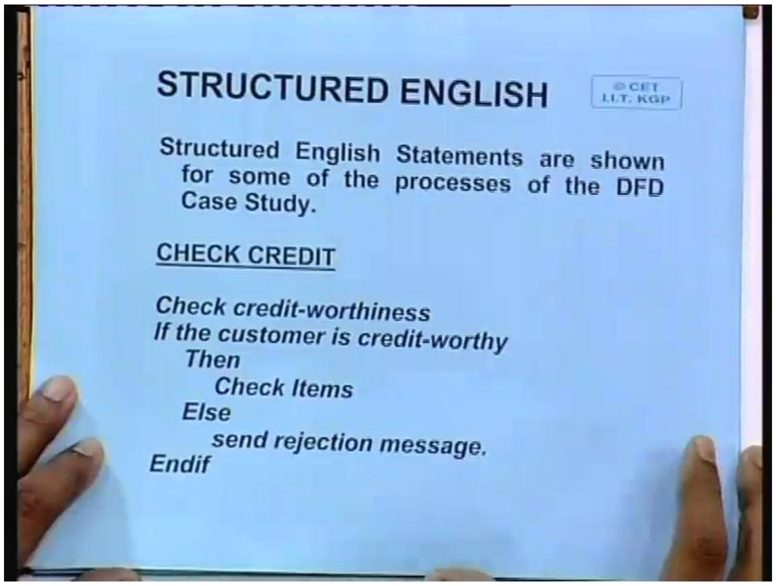
Now, the structured English statement for this particular will look like this. Please, look at these, let me before we going to details let me tell you the structured English statement is not a program,   it’s not a program. It is a pseudo code, it’s a Pseudo code. What is the purpose of the structured English statement? The purpose is this makes the logic clear. You see that’s the idea, the logic should be made clear to the programmer who might be writing the program later. So, basically the program has got a number of parts, number one the data flow diagram, the process logic itself will give first of all what are the inputs, what are their data structures, what are the files that are being handled, what are the outputs you know that is clear from the data flow diagram. What is not clear is what is inside the bubble, all right that inside the bubble if there is a decision logic that decision logic can be expressed in terms of structured English.

So, as you can see here check credit worthiness if the customer is credit worthy, so look at these if then else blocks. If the customer is credit worthy then and go down towards it that is the else, else order is rejected along with a rejection message. So, you have to read it like this, if the customer is credit worthy then check items ordered against a list of items, else order is rejected along with a rejection massage endif. So, these endif belongs to these if and it has an else structure but within the then part we have another if, if the items are found on the least then check availability of the items in inventory, else send regret message to the customer endif, right. Then another if is there in this within this then part if items are available in required quantity then items are sent to customer along with a packing slip and an invoice, else items with the order details are stored as pending orders.

See, these full stops you can ignore, basically this full stops should not be there, it’s a structured English statement as such. Again, let we tell you it is basically a pseudo code the idea is not a programming language, all right. There will be lot of loop holes which the programmer has to feel it up, has to feel it up when you actually translate these into a program. Each and everything cannot be spelled out but basically the process specifications are important in the sense that these process specifications can be utilized to make what is known as program specifications later on, all right. So, you see you can understand the automation process, the advantage that you get out of it if you develop process specifications in this manner you can simply handover the decision logic, the part of the DFD to a programmer even if he is new. He knows probably nothing about the system, he doesn't know the trading companies, buying and selling companies intricate structures but he is a simply developing the program of this.

So, he is handed over the logic, he is been given the inputs, he is been given the outputs and he has to simply develop the program for this component of the software, all right. The integration aspect has already been taken care of within the data flow diagram. Obviously, you must remember that every process need not translate to your programming structure, all right. Some could be manual and similarly on the other side, it is not one to one. The processes are not the only thing that could be translated to program modules. You can have other part of the program modules particularly the input handling and the output handling, all right. They handling the input, validating the input could be one another program module. Similarly, the validating the output, I mean preparing the output in the pre specified format within the print layout could be another part of the program or program module. These we shall discuss later on while we discuss structure design right in the form of structure charts, right.

(Refer Slide Time: 00:13:14 min)



STRUCTURED ENGLISH

© CET I.I.T. KGP

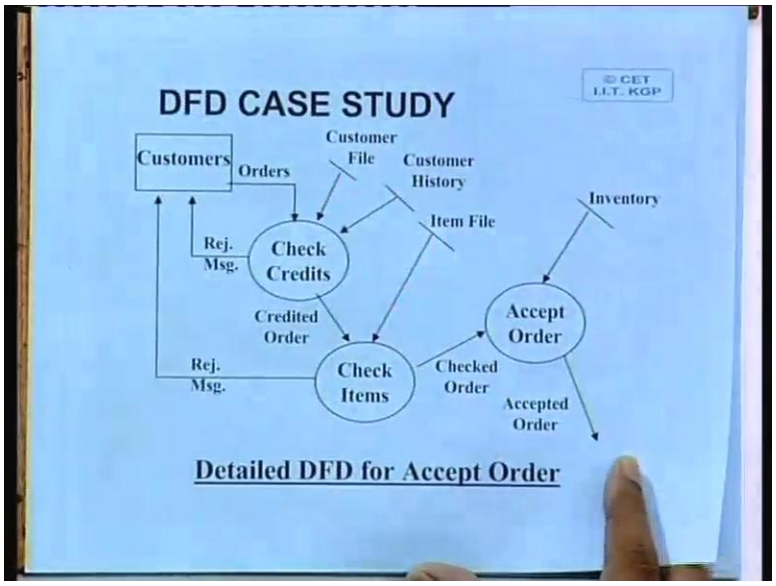
Structured English Statements are shown for some of the processes of the DFD Case Study.

<u>CHECK CREDIT</u>

```
*Check credit-worthiness*
*If the customer is credit-worthy*
    *Then*
        *Check Items*
    *Else*
        *send rejection message.*
*Endif*
```

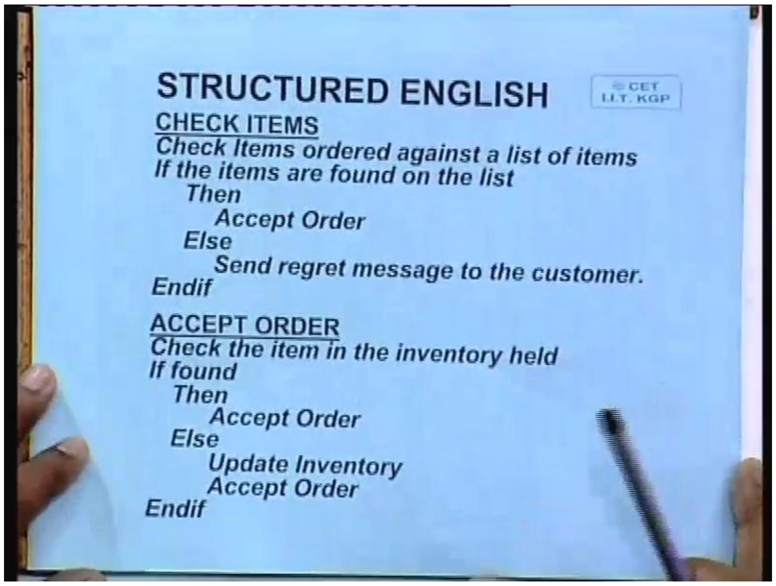
But this is the structure chart that you have seen basically that of the complete process but if we had divided the data flow diagram into the detailed data flow diagram, let me just show it once more.

(Refer Slide Time: 00:13:26 min)



This one suppose when we put the detailed data flow diagram of accept order then we have the customer check credit, check items and accept order, all right. So, when we have divided it like this, our structured English statements would also be smaller. So, something like this for some of the processes that is check credit, check credit worthiness if he customer credit worthy then check items else sent rejection message endif. So, you see the check items is actually the next, next process, all right. So, here what we have done we have made a hierarchy of the structure charts. A hierarchy of the structure charts it is not that every portion is put into one big structure charts, we have put it as stage by stage. The first one is the check credit where if it is if condition then you go to check items.

(Refer Slide Time: 00:14:33 min)



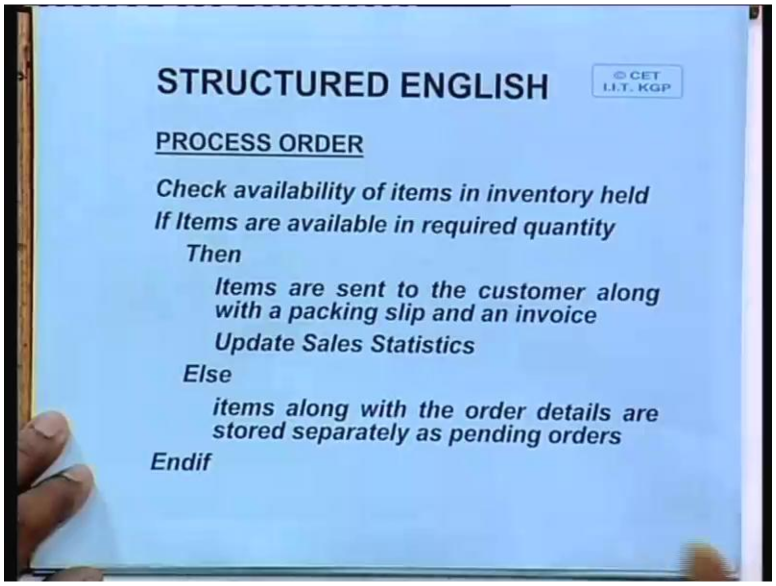
STRUCTURED ENGLISH

```
CHECK ITEMS
Check Items ordered against a list of items
If the items are found on the list
   Then
      Accept Order
   Else
      Send regret message to the customer.
Endif

ACCEPT ORDER
Check the item in the inventory held
If found
   Then
      Accept Order
   Else
      Update Inventory
      Accept Order
Endif
```

Similarly, the check items ordered against a list of items, if the items are found on the list then accept order all right, else sent regret message to the customer endif. See, accept order is further is the next process. What is accept order? Check the item in the inventory held, if found then accept order, else update inventory accept order, all right endif. So, you see what happens sometimes what may happen that the item is actually there but unfortunately it is not entered there. So, what you have to do there? You may have to update inventory and accept order. So, this part was not there in the original module, so this is something new that I have put here. We will discuss about it later.

(Refer Slide Time: 00:15:36 min)



STRUCTURED ENGLISH

© CET I.I.T. KGP

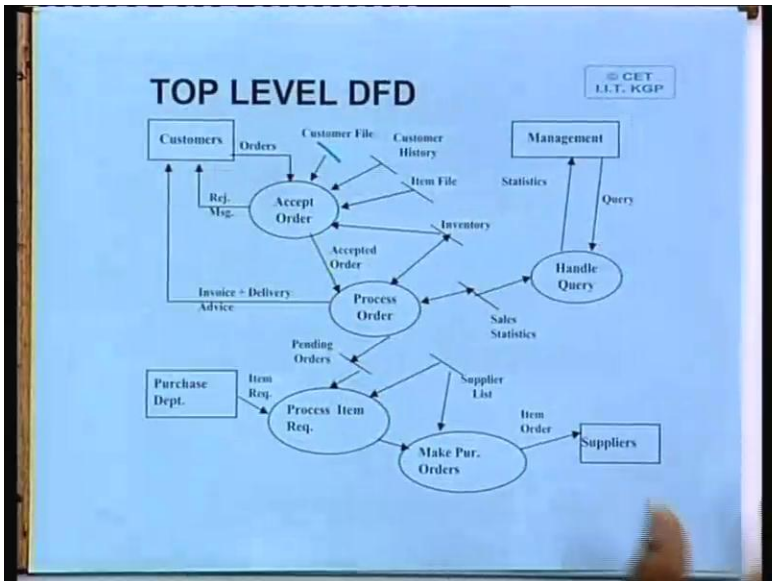
PROCESS ORDER

```
Check availability of items in inventory held
If Items are available in required quantity
  Then
     Items are sent to the customer along
     with a packing slip and an invoice
     Update Sales Statistics
  Else
     items along with the order details are
     stored separately as pending orders
Endif
```

So, that is the accept order part then after the accept order is there. Suppose, this is the next step that is our process order okay, you are writing something, accept order basically check the item in the inventory held if found then accept order, else update inventory accept order endif, all right. So, this is the basic structure about the accept order. Now, let's go back to our top level data flow diagram once more.

(Refer Slide Time: 00:16:10 min)

TOP LEVEL DFD

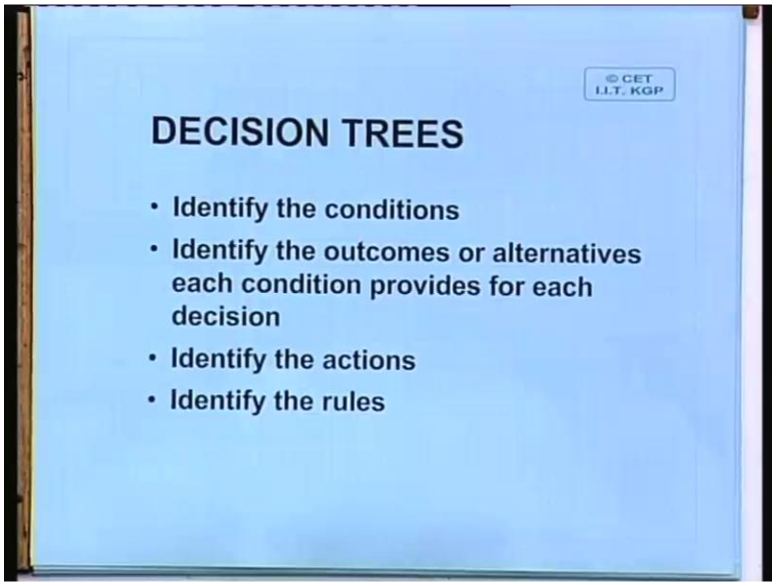


What happens here, if you look at it after your accept order we have the process order. The process order, the accepted order comes again you check against the inventory and give invoice or delivery advice if the item is available, if not put it in the pending orders all right and develop sales statistics and update it. Now, naturally you may think that process order may not be a single process, may not be a single process, there might be a number of processes involved but assuming it's a single process, this is what we can think of. Check availability of items in the inventory held, if items are available in required quantity then items are sent to the customer along with a packing slip and invoice, update sales statistics, else items along with the order details are stored separately as pending orders endif, all right.

So, you see I have simply put the Basic English statement but you can make it a little bit more cryptic, you can make it a little bit more cryptic and put it in the Standard English form, all right. So, this is basically like a pseudo code and this pseudo code can actually be used for our specific purpose. Unfortunately what happens, the logic sometimes can become even more difficult, all right. Today, I will not get into more difficult once today what I would like to concentrate is a basic concept of the decision tree and the decision tables. In tomorrow, next week's class we shall delve into more complex logic.

What happens when the logic is even more complex and how you can make use of decision tree and decision tables to codify them, what is the basic purpose, all right. So, this is the structured English part then after the structured English, the next one is known as the decision tree. See, basically what the decision tree does? The decision tree actually it is you know as the name suggests it's a tree. Now by the way what is a tree, what is, what is a tree, how is a tree different from a network? Sir, it is nodes and branches. Yes, the nodes and branches are there in a network also. That's not the, there are no cycles that's the thing. The basic idea of a network is there are cycles, there are some closed areas. A tree by definition should not have a closed area, should not have a closed area, is it okay should not have a closed area that's the idea. So, there should be nodes but the nodes should not, you should not be once you leave a node you should not be coming away node, all right.

(Refer Slide Time: 00:19:33 min)

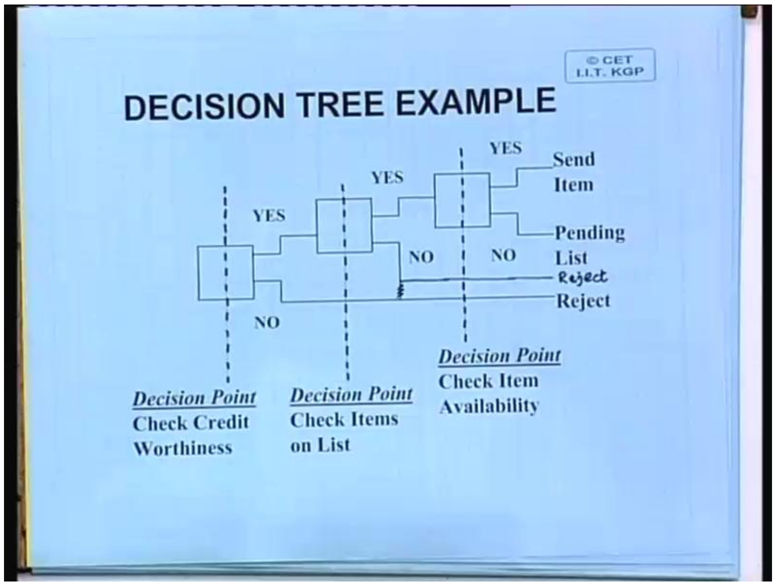

© CET I.I.T. KGP

DECISION TREES

- Identify the conditions
- Identify the outcomes or alternatives each condition provides for each decision
- Identify the actions
- Identify the rules

So, anyhow that's the basic idea of a tree. So, identify the condition and decision tree essentially is a tree where the three things are usually shown, the decision points, the events, decision points, events and probabilities. Out of these the events and probabilities let us forget for the time being just assume that we are showing some decision points. So, identify the conditions, identify the outcomes or alternatives each condition provides for each decision and identify the actions, identify the rules.

(Refer Slide Time: 00:20:22 min)



See, what happens this rules part I will discuss, look at this particular small decision table. See, we have three decision points right three decision points. You can, I don't know whether you can see some very faint lines is here right, this faint line basically show that this is the decision point, okay. So, we can just make it little better. See, this rectangle actually show a decision point, so every rectangle is a decision point. So, the first one although see I have put these two no together because both are reject but strictly because this is a tree, we should not do that actually, we could just take it like this, straight like this and we can say this is also reject and this we may remove okay. So, we can say that that at the first decision point, you are checking the credit worthiness and it could be yes or no, if it is no then you reject if it is yes then go to the second decision point check items on list, if it is yes then go to the third decision point, if it is no reject. and the third decision point check item availability, if it is yes send item, if it is no put it on pending list.

So, this is basically decision tree but what actually happens, sometimes you may have to evaluate the options. See, that is where the events and that is where the probabilities can come, right. Say, suppose down the line after you have decided to send item then actually depend on certain probabilities say what may happen, the probabilities could be the payment will be received, payment will be delayed, payment will not be received, all right.

Similar thing you might think of for other options, sometimes. So, some sort of payoffs could be attached to each of the options that you have selected, all right. So, when you have got all these payoffs, what actually can happen depending on what kind of decision rule you are following, the decision rule could be expected value or maximin, minimax, etc and you can evaluate those decision rules and you can go down the decision tree to find out which one is the best possible option. So, you can actually use the decision tree for decision making, all right. So, decision trees are actually used for decision making, with regard to software development I think that role is not very much, right.

Basically, with regard to software development this much is enough just to encode the decisions and put it in a nice format but for the sake of completeness because we have brought this topic will discuss it some other time right towards the end of the decision analysis lectures. Then after your decision tree this thing, what is most important is the decision table. See, what happens in your decision tree, the decision tree is nice but it doesn’t show all possible combinations which may actually arise. See, what happens, if all are yes okay fine but if you take a branch of yes no but can you have something like yes no yes or can you have no yes yes, please evaluate this other possibilities also. See we have taken the way the branches are shown, it is yes yes yes all right or yes yes no or yes no and then don’t care, all right.

Yes, I mean not yes no, no don’t care, yes no don’t care, yes yes yes, yes yes no, these are the 5 branches that we have actually considered. What are the things? 1 2 then 3 4 actually we have considered, is it not. 1 2 3 4 but because there are three conditions and each condition can lead to two possibilities yes or no, technically there are 8 possibilities. Please think about it, you can have no yes yes, you can have no yes no or you can have yes no yes and yes no no. So, you see you may say that why should you bother that they may not arise, they may not arise. For example one something has gone ahead in the first decision point that you have check credit worthiness and you have found it no, why should it be checked in the other two? It shouldn’t be, it should be reject then and there all right but suppose there is a mistake, you see after all you are handling manual processes, basically you might right a program for it but end of the day person has to do it, all right, you are handling human being, it may so happen that even after credit worthiness as failed, somebody may actually has check those yes, all right. So, it is always necessary to explicitly identify the decisions all right for each combination of yes and no figures. This is possible with the help of a decision table.

So, decision table what does it do? Actually, let us also think in a slightly different manner. Let’s take for the example for joint entrance examination to the IITs, right. At some point of time obviously it has been discontinued now, there was a provision for taking abroad you know students abroad in a direct manner, the dasa students right, so called dasa students. The rule that time that was stipulated that these are the people that they should be living abroad.

Now, who is living abroad, it’s a difficult question to answer. Suppose, you have studied in India and you have already got passed in your class 12 examination and decided to go abroad for few months. Will you qualify as a student from abroad and then can you bypass the joint entrance examination and gets through the other channel like sat all right, will you be allowed. So, people thought that you should not be allowed, so they have made a decision rule that person should have stayed for the last 5 years abroad all right and should have appeared in 10 plus 2 from a school abroad. You say these essentially cuts off all these people who might be thinking that okay let me be here up to class 10 and then go abroad and study next two years and apply in this category, right.

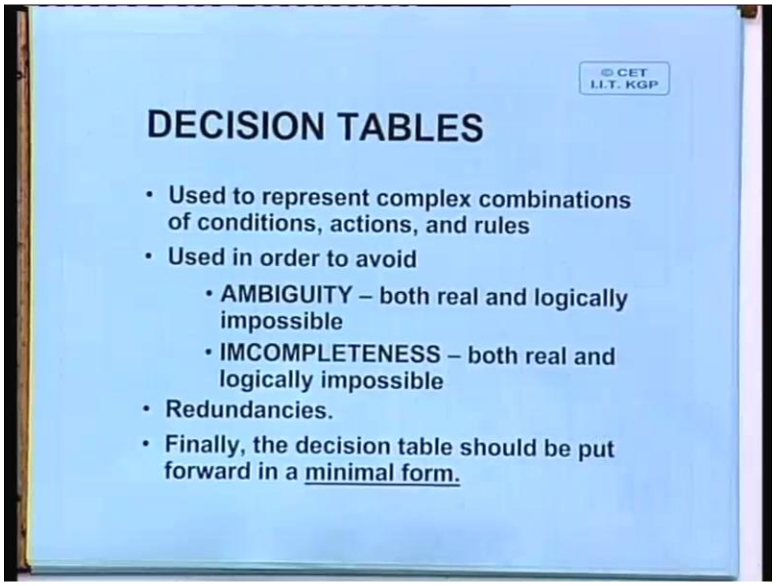
So, you have to be having last five years of your education abroad all right, then some difficulty arose. See, at the time of gulf war, the first gulf war 1991 lot of Indians were actually evacuated all right. So, there was a temporary break in staying abroad of a number of students who had to come to India, stay for maybe one or two years and when everything settled again went back to the middle east. Now, tell me these people are not staying continuously five years abroad, so will they be allowed or we shouldn't be, shouldn't they not be allowed? See, the logic says that since it was a disturbance they should be allowed, all right.

So, a new rule has been formed anybody who has stayed 5 years, out of the last 7 years. Say, instead of last 5 years, the rule has been change to 5 years, out of the last 7 years should be considered as a candidate. See, actually these example essentially I gave to you know focus on the point that the decision rules do not remains static, in any business depending on as new conditions come newer decision rules keep coming. Think of a particular organization where you are going to give some facility, a special allowance all right. Now, who should be given some special allowance? Initially it was thought anybody whose salary is less than a certain value should be given these particular allowance all right but later on it has been found that these unique point of you know just a salary is not enough. There are some people who may be getting a little higher salary towards the end of their carrier but they should be basically allowed, so the rule is that if the salary is like this or you are in a certain category, all right.

Similarly, you can keep on adding newer conditions to it. So, either your salary is less than this and you are a B group employee or you are a C group employee or you are nearing retirement and like this. So, what happens the decision rule becomes complex. Why it becomes complex because we are not using the elementary decision rules, we are using a complex of a number of elementary decision rules, all right. Now when the decision rule is so difficult, basically not in terms of elementary rules but in terms of complex of elementary rules.

Suppose, you say a equal to yes or a equal to no, b equal to yes or b equal to no. You can combine a and b together that's a separate issue but if your criteria just a and b where a and b are mutually exhaustive then you are using elementary decision rules but if your decision rule is if a and not b if these are the kind of decision rules you are using then you are using complex decision rules. Invariably in organizations the decision rules tend to be complex then a time come then where there are lot of redundancies, contradictions, anomalies keep coming, all right. So, this is why it is necessary particularly when you are developing programs that you specify the rules in terms of elementary decision rules which are mutually exhaustive to one another and see the completeness of the rules that means all aspects are basically covered.

(Refer Slide Time: 00:33:19 min)


© CET
I.I.T. KGP

DECISION TABLES

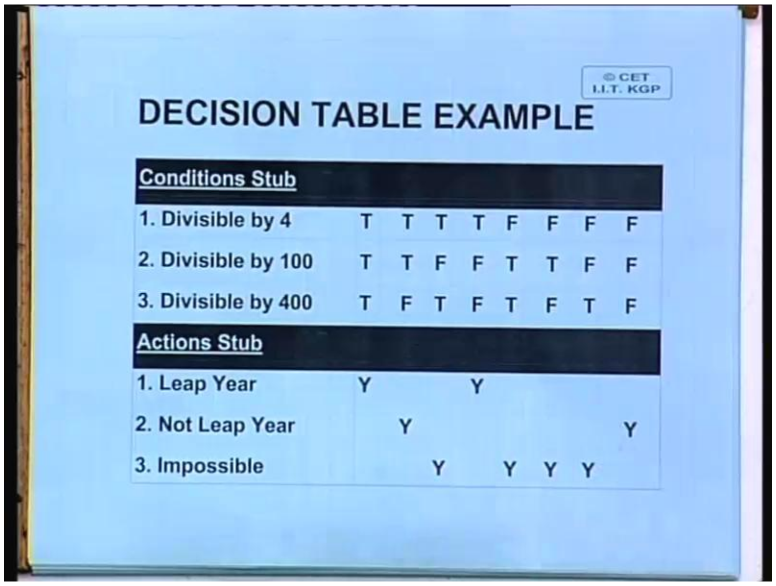
- Used to represent complex combinations of conditions, actions, and rules
- Used in order to avoid
  - AMBIGUITY – both real and logically impossible
  - IMCOMPLETENESS – both real and logically impossible
- Redundancies.
- Finally, the decision table should be put forward in a minimal form.

So, that's the thing about the decision tables. Decision tables are used to represent complex combinations of conditions, actions and rules all right used in order to avoid ambiguity and imcompleteness and finally redundancies. See, the ambiguity could again be real or logically impossible, you may have an ambiguity which is really there or you may have an ambiguity which is really not there that means although the ambiguity is there but it can never happen.

Usually that sort of ambiguity takes place where you have conditions which are not mutually exhaustive. You may have certain situations where you are using conditions which are not mutually exhaustive. So, you may have logically impossible ambiguity, it's an ambiguity but it can never happen, I will give examples.

The imcompleteness that means the decision table is not completely specified, all right. You have not shown details about all aspects of this particular decision table. So, you can have again both real and logically impossible imcompleteness and obviously redundancy is another thing. Finally, the decision table should be put forward in a minimal form. So, essentially what you are beginning with a minimal form, basically the decision rules are given to you in the form of complex logic, expand them, try to formulate them in terms of elementary rules, see all combinations, avoid ambiguity, avoid imcompleteness, remove redundancies and again bring back the minimal form for practical use. So, that is the purpose of the decision table analysis.

(Refer Slide Time: 00:35:22 min)



© CET I.I.T. KGP

DECISION TABLE EXAMPLE

Conditions Stub								
1. Divisible by 4	T	T	T	T	F	F	F	F
2. Divisible by 100	T	T	F	F	T	T	F	F
3. Divisible by 400	T	F	T	F	T	F	T	F
Actions Stub								
1. Leap Year	Y			Y				
2. Not Leap Year		Y						Y
3. Impossible			Y		Y	Y	Y	

So, I'll give a very simple example that is about the leap year, calculation of leap year. So, there are two stubs usually, the condition stub and the action stub. See, there are three conditions divisible by 4, divisible by 100, divisible by 400. Unfortunately what has happened here, they are not mutually exhaustive. Can you see this? A number which is divisible by 4 is also divisible by

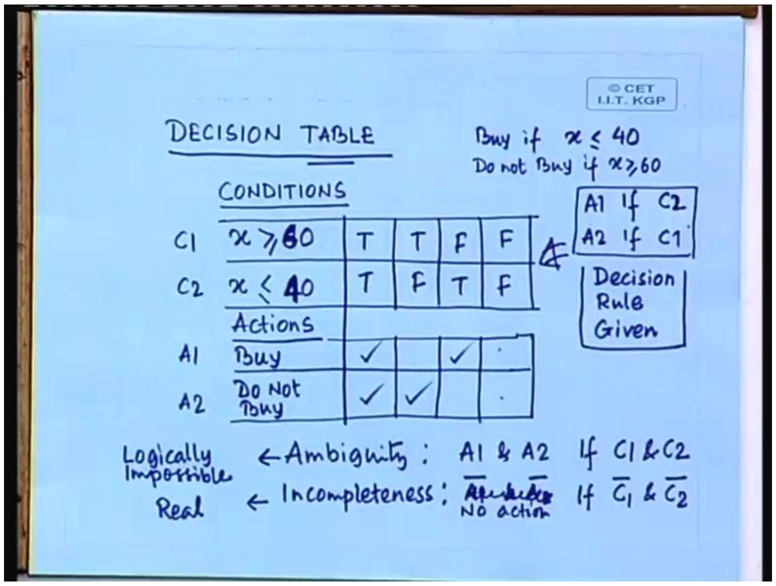
I mean sorry divisible by 100 is also divisible by 4. Is it okay? A number which is divisible by 400 is also divisible by 4. So, they are not mutually exhaustive, these might lead to some kind of inconsistencies later on but then there is no other way but to go for this type of structure because the problem of leap year is like this. So, what we do since each condition can be either true or false, so it's a binary decision table all right. So, we have to illustrate them in terms of all possible combinations amongst these three conditions.

We know that three conditions each is binary, so 2 to the power 3, there are 8 possibilities. How to write them? First you put the first one true then you put the second one true. So, when you put the second one true, third one could be either true or false. Now change the second one to false, again third one could be true or false, the true set is over. Now repeat the entire thing with the false set at the first one, so that's why TTT, TTF, TFT, TFF, FTT, FTF, FFT, FFF right, so it covers all 8 possibilities. Now, action see when all are true, so it's a leap year all right and when else it is a leap year, when the only divisible by 4 is true then also it is a leap year. Actually what happens, 2000 is a leap year right but 1900 was not a leap year that's the idea that we exclude some years, every 400 years we make a leap year but all the years which are divisible by 100 they are not leap year. For example 2100 will not be leap year, 2200 will not be, 2300 also will not be a leap year but 2400 will be a leap year, all right. So, this is a special exception that has been made for years which are divisible by 100.

Basically to compensate for those actually 365 is not exactly 6 hours, little less than 6 hours so to compensate for that. So, you see not leap year will be your TTF right divisible by 4, divisible by 100 but not divisible by 400, it's not a leap year. So, actually a look at the action stub, the action stub is again not mutually exhaustive. We had to do that, we could have taken leap year and otherwise right but unfortunately we have to distinguish between not leap year and impossible conditions. Impossible conditions what about if all are false then definitely it is not a leap year but what about these cases? Suppose, divisible by 4, divisible by 400, it has to be divisible by 100, this has happened because they are not mutually exhaustive. So, we have this impossible conditionalities.

So, actually ambiguities are existing. So, if you look at our previous thing, suppose you do not specify, suppose you just specify the leap year and not leap year conditions or if you just specify the leap year conditions all right then what will happen and say anything else is not leap year. Unfortunately it will be an incomplete decision table because you have not specified the other conditions, all right. So, when you try to fill it up then you find that there are some difficulties, there are some conditions which cannot be filled up. For example divisible by 4, divisible by 400 but not divisible by 100, it cannot happen. So, there is an ambiguity but these ambiguity is not a real ambiguity, it is a logically impossible ambiguity, this is all right. It's a logically impossible ambiguity but sometimes you may have some situations where the ambiguity is logically possible those are the dangerous things, right. That means you are not specified the conditions for which you are trying to have the situation. Say, let's take an example to understand what it is.

(Refer Slide Time: 00:41:16 min)



DECISION TABLE

Buy if $x \leq 40$

Do not Buy if $x \geq 60$

A1 if C2

A2 if C1

Decision Rule Given

CONDITIONS

C1	$x \geq 60$	T	T	F	F
C2	$x \leq 40$	T	F	T	F
	Actions				
A1	Buy	✓		✓	·
A2	Do Not Buy	✓	✓		·

Logically Impossible ← Ambiguity : A1 & A2 if C1 & C2

Real ← Incompleteness : ~~[illegible]~~ No action if $\overline{C_1}$ & $\overline{C_2}$

Suppose, actually there should be other around, so you see we have two conditions, x is greater than equal to 60, x is less than equal to 40, right. So, again they are not mutually exhaustive. Now, assume that both are true. What is actually given to you is buy if you see this is what is given. You have been told, actually this table is not given to you. You have been told buy if x is less than 40, do not buy if x greater than equal to 60, this is what is told to you, right. The decision rule that is specified to you okay you go and buy this item x, do not buy if it's value is above 60 rupees buy if it's value is less than equal to 40 rupees, hypothetical or not assume this is what is given to you. So, in codified term what we say A 1 if C 2, A 2 if C 1, basically if C 2 is true C 2 true can be written as C 2, all right.

C 2 is false can be written as C 2 bar, so A 1 if C 2, A 2 if C 1 this is what is given. So, this is the decision rule, this is the decision rule given. So, actually you see this is our starting point, this is our starting point, using this starting point we have developed this decision rule, I mean decision table. Now what we find where we try to utilize this decision table? Try to use this decision rule. Basically when C 2 is true then it should be A 1 all right that means x less than equal to 40 then you should buy. So, buy and do not buy when x is greater than 60, all right. So, this is the second rule. So, according to second rule, this is the do not buy. Is it clear what we have done? This is the decision rule given to you, we have used the decision rule into our decision table and this is what we have got. What we have got that? When both are true, you both buy and do not buy. Sir, but how can both be true? We will come to that, then if x is greater than 60 do not buy, very good, if x is less than 40 buy that is also very good. But what happens if both are false, no actions specified, all right.

So, this particular table has got ambiguity. What is the ambiguity? A 1 and A 2, if C 1 and C 2 and it has an incompleteness. What is the incompleteness. Yes, A 1 I mean no sorry, no action specified if C 1 is not true and C_2 is also not true. Is it okay? Now, try to understand there these ambiguity you tell me is it a real ambiguity or it's a logically impossible ambiguity. It is a logically impossible ambiguity because it cannot happen, you cannot have both true. So, these decision tables has an ambiguity and these ambiguity is logically impossible. It also has an incompleteness. Now, tell me is this incompleteness real or logically impossible? It's a real incompleteness.

It's a real incompleteness because you can have a situation that is suppose any value between 40 and 60, say a value of 50, so no action is specified. So, therefore you have found that these decision table has got ambiguity as well as incompleteness, so we should do something about it. What should be done first and foremost? The way we have decided the actions, the conditions, conditions stub they are not mutually exhaustive. So, first of all they should be made mutually exhaustive, all right, they should be made mutually exhaustive that could be one suggestion. Second one you must specify action for each of the cases. It is okay? So, stop here today, in next Monday’s class we continue from here.