

**Fundamentals of Artificial Intelligence**  
**Prof. Shyamanta M Hazarika**  
**Department of Mechanical Engineering**  
**Indian Institute of Technology - Guwahati**

**Module - 3**  
**Lecture - 8**  
**Game Playing**

Welcome to fundamentals of artificial intelligence. Today we will be looking at game playing.

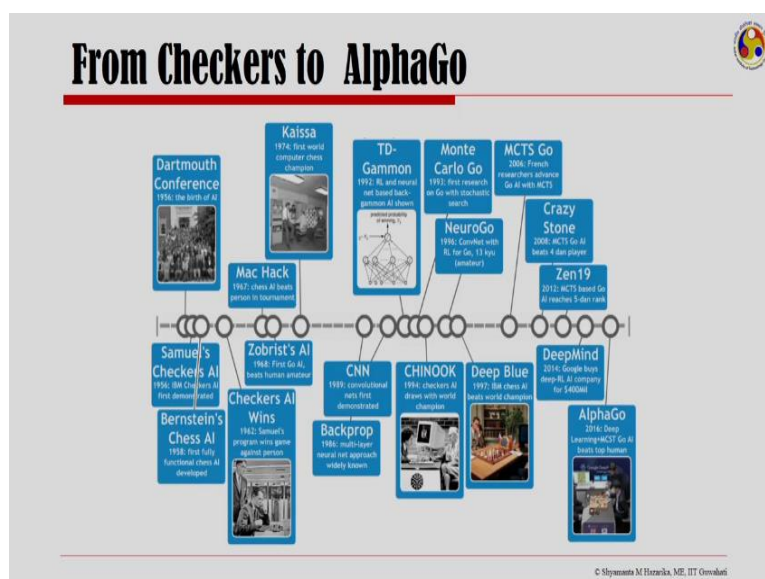
(Refer Slide Time: 00:40)

**Fundamentals of Artificial Intelligence**  
**Game Playing**

  
**Shyamanta M Hazarika**  
Mechanical Engineering  
Indian Institute of Technology Guwahati  
[s.m.hazarika@iitg.ac.in](mailto:s.m.hazarika@iitg.ac.in)  
<http://www.iitg.ac.in/s.m.hazarika/>

Games have been an integral part of the development of artificial intelligence. The quest for writing computer programs that could play games is as old as computers themselves.

(Refer Slide Time: 00:54)



Starting with Samuel's checker AI, we have AI chess playing programs to more recently the AlphaGo. All of these have pushed the frontiers of AI.

**(Refer Slide Time: 01:14)**



## Why Games?

- Games are very good platform for experimentation.
- Games provide a well-defined environment in which states are intrinsically discrete; allowing one to focus entirely on the decision making strategy.
- In games, rules are well defined and success and failure can be measured easily.
- Allows us to reason about multi-agent activity.
  - Problem solving we have studied so far is characterized that only a single agent is involved!
  - Interaction between agents has most commonly been studied by abstracting them as games.

3 © Sreyas M. Hiranika, M.E., IIT Guwahati

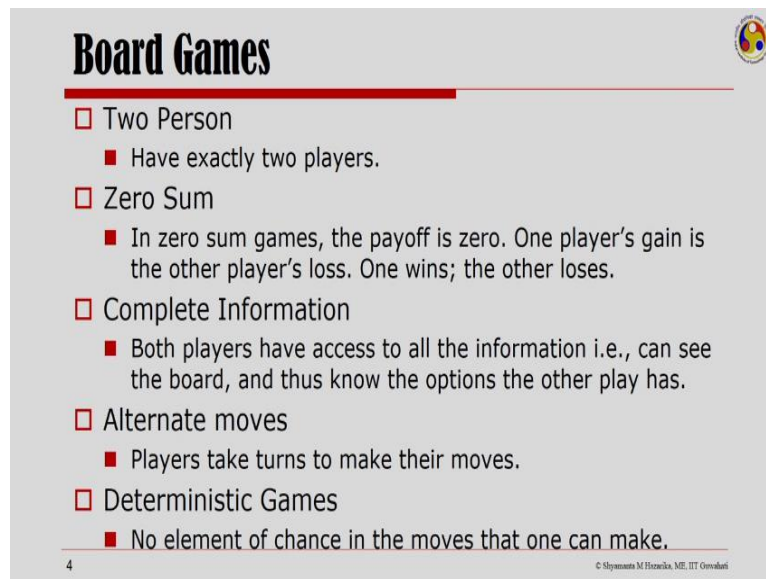
One important realization has been that game allows a very good platform for experimentation. And that is why there is fascination for looking at games. Games provide a well-defined environment in which states are intrinsically discrete. So, that allows one to focus entirely on the decision-making strategy.

If we had not taken this games and would have looked at decision-making strategies to be evaluated by other means, such as a robot playing with a kid, there would be so much of uncertainty involved, so much of probabilities to be taken into account, that we may have lost the very purpose of using it to look at our decision making strategy. In games, rules are well defined.

And success and failure can be measured easily. That is why we look at games as a medium to evaluate the strategies of decision making. Games allow us not only to reason about one algorithm or one process. Games in a way allow multi-agent activity. We had only looked at one algorithm who's aim was to find an optimal path to the goal. Bringing in the idea of games, we could now involve more than one agent.

And the interaction between agents has mostly been studied by extracting them as games. Here we will look at a group of games which are called board games such as chess, checkers or more commonly played the tic-tac-toe.

(Refer Slide Time: 03:22)



**Board Games**

- Two Person
  - Have exactly two players.
- Zero Sum
  - In zero sum games, the payoff is zero. One player's gain is the other player's loss. One wins; the other loses.
- Complete Information
  - Both players have access to all the information i.e., can see the board, and thus know the options the other player has.
- Alternate moves
  - Players take turns to make their moves.
- Deterministic Games
  - No element of chance in the moves that one can make.

4 © Shantanu M. Hiranaka, M.E., IIT Guwahati

Let us now quickly review the characteristics of these board games before we move on to talking about how these games are to be dealt with, with an AI. Board games are those which you play on a board. We are talking of two person games. That is, we have exactly 2 players. These games are zero sum games. Now, zero sum games are those where one player gains, the other player loses.

So, it is like A's gain is B's loss, if A and B are playing a zero sum game. We are talking here, games which are complete information games. Complete information games are those in which both players have access to all the information. That is, they can see the board and thus know the options that the other player has. The other important characteristic of board games that we will deal with is that we are talking of games which are alternate moves.

That is, the 2 players that is involved in the game takes turn to make their moves. And finally, we are talking of deterministic games. Deterministic games are those which does not have any element of chance, in the moves that one make. So, in this portion of our course, we will be looking at how one has a winning strategy for a board game. But then, the board games that we will be talking of would be restricted to two person, zero sum, complete information, alternate moves, deterministic games.

Now, let us take a very simple game to illustrate what we mean by these characteristics of a game. This game is called the Grundy's game. Many of you must have played this while you were young.

(Refer Slide Time: 05:46)

## Grundy's Game

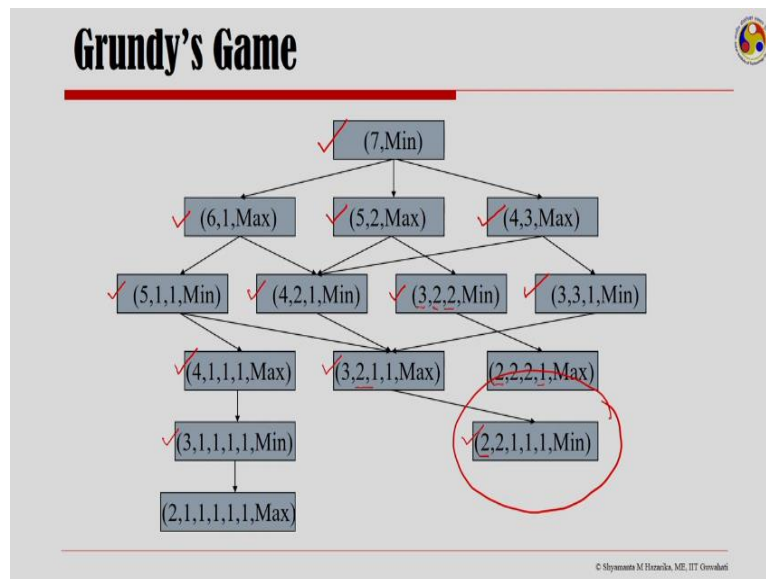
- Two players
- Single pile of objects
  - Say, stack of pennies.
- Divide the stack into two stacks that are unequal
  - First player divides the original stack.
  - Each player alternately does the same to some single stack when it is his turn.
- Game proceeds until every stack has either just one penny or two
  - Continuation becomes impossible!
- Player who first cannot play is the loser!

© Shyamanta M Hazra, IIT Guwahati

The game involves 2 players and we have a stack of pennies; a single pile of objects, either a stack of pennies or people will all even put 7 stones one above the another to make one single pile. The idea of the game is to then divide the stack into 2 stacks that are unequal. So, you start with stacking 7 stones. If it is the opponent's turn, he will then break it up into 5 and 2. So, the first player divides the original stack.

And then, each player alternately does the same to some single stack, when it is his turn. The game proceeds until every stack has either just 1 penny or 2. Because, at that point of time, you cannot divide the stack into 2 stacks that are unequal. And continuation at that point becomes impossible. The player who first cannot play is the loser in this game. So, let us look at Grundy's game.

(Refer Slide Time: 07:02)



Here, I have 7 pennies in 1 single stack and it is Min's turn to play. Now, in such game, we have a notion of 2 players. One is called the Min the other is called the Max. We will see why they are called Min and Max in a moment. But now, let us say the player Min starts to play the Grundy's game with 7 pennies in a single stack. Now, a little thought, if you give, you will realize that 7 can be broken up into 2 unequal stacks in 3 ways.

I could have in 1 stack, 6 pennies. In the other stack, a single one. Or, I could have 5 and 2, or I could have 4 and 3. This is what has been shown as the next level of the game. So, Min starts playing here. And then, breaks it into a stack of 6, 1; 5, 2 and 4,3. It is now Max's turn to play. And he can have the following options. Max could break the 6, 1 into either a 5, 1 stack and the original 1 stack already there. So, there will be 3 stacks now: 5, 1 and 1.

Or Max would break it into 4, 2, with the original 1 remaining there. I would like to draw your attention to the fact that, at this point, we had 6 in a stack and 2 possibilities of breaking it up into unequal stacks is 5, 1 and 4, 2. So, these are the possible moves that Max could take if Min had broken it into 6, 1. On the other hand, if Min had actually divided the original stack into 5, 2, I could have the possibility of generating 4, 2, 1; because, 5 could be broken up into 4 and 1; and I would carry forward the 2 from here.

Or 5 could be broken up into unequal stacks of 3 and 2. So, I would have 3, 2 and 2. Now, if Min had gone to 4, 3; then the possibilities for Max would be 4, 2, 1. Because the third could be broken up into 2 and 1. Or other possibility could be that the 4 could be broken up into 3,

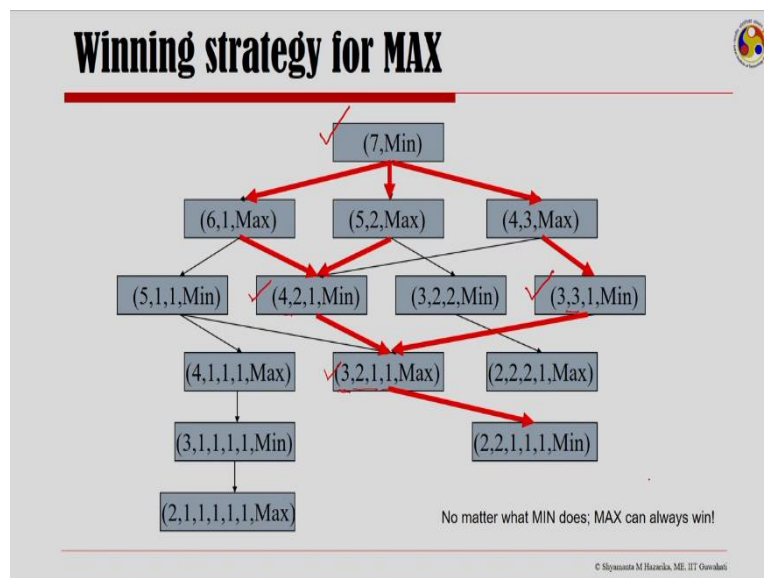
1. So, I would have 3, 3, 1. Next it is Min's turn to play. And from the given possible alternate positions, Min could explore the next level of the game.

That is, 5 could be broken up into 4, 1, 1 unequal stack; or 3, 2. The 4, 2, 1 could be broken up into 3, 1 leading to 3, 2, 1, 1. And the 3 could be broken up into 2, 1; leaving me with 3, 2, 1, 1 stack. The 3, 2, 2 stack, if I start breaking this, I will only create equal number. So, all I can do is break up the 3 stack into uneven stack of 2 and 1. Proceeding further, Max would then be able to break this node into 3, 1.

The second one could be possible to be broken up into 2, 2, 1. And the only possibility now left with Min is that he can work on this node. Here it is not possible for Min to break it up into unequal stack. Because, if I have 2, 2; even if I think of breaking this stack, all I will have is 1, 1 penny in each stack. So, it will not be a legal move according to the rules of Grundy's game.

At this point, Min can play and break it up into 2, 1, 1, 1; at which point, Max will not have a possibility of moving any further. So, this is the total tree on the possibilities of Grundy's game with 7 pennies in the initial step.

**(Refer Slide Time: 12:21)**

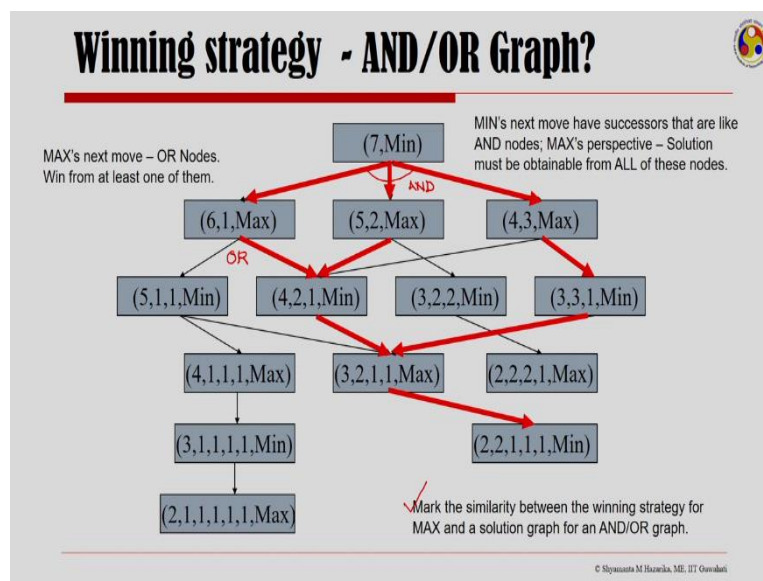


Now, if you look at the winning strategy of Max; what should Max do to ensure that he will win this game. We can note that, no matter what Min does in this game, that means, no matter what Min tries to follow, Max can always win this game. This is precisely because at the first instance, here at the top, Min can follow 3 paths which is either to 6, 1 or to 5, 2 and

to 4, 3. Once he follows these 3, one of the 3, Max can ensure and come to 4, 2, 1 or to 3, 3, 1.

Once Max ensures that he has come to either 4, 2, 1 or 3, 3, 1, Min is forced to play and divide the step, either it is 3 into 2 and 1; or it is 4 into the 3 and 1. That is, at that stage of the game, Min will have to come to this node. Once Min comes to this node, Max ensures that he breaks the 3 stack into 2, 1. And at that point, Min does not have a legal move and Max can win. So, as we can see, no matter what Min does in this game, Max can always ensure that he wins this game.

(Refer Slide Time: 14:09)



Now, let us take a moment and see if this winning strategy for Max has a similarity to the AND-OR graph that we have discussed previously. The AND-OR graph is about having AND nodes and OR nodes at alternate levels. So, if you look at the winning strategy of Max, at one point, you would see that there is a similarity in the winning strategy for Max and a solution graph for an AND arc.

Because, if you look at Min's next move at any point, then the Min's next move have successors that are like AND nodes. From Max's perspective, solution must be obtainable from all of these nodes. Whereas, if you look at Max's next move, you will see that Max's next move is something like the OR node. Because, all Max needs to win this game is to follow at least one of them. So, this winning strategy of Max is very similar to the AND-OR graph that we have discussed previously.

(Refer Slide Time: 15:42)

## Why not AO\* always?

- Many simple games can be handled by search techniques that are analogous to those used for finding AND/OR solution graph
- For more complex games such as chess or checker the AND/OR search to termination is out of question.
  - Complete game tree for chess has approximately  $10^{40}$  nodes.
  - It would take  $10^{22}$  centuries to generate the complete checker tree, if a successor could be generated every 1/3 of a nanosecond!

© Srinivasa M. Hazarek, ME, IIT Guwahati

So, question is, can we use the search algorithm for AND-OR graph, that is the AO \* for searching such game trees? Many simple games actually can be handled by such search techniques that are analogous to those used for finding AND-OR solution graphs. For more complex games such as chess or checkers, the AND-OR search determination is out of question.

It is not possible to generate the whole game tree. Complete game tree for chess has approximately  $10^{40}$  nodes. And, you can see from the volume that it is not feasible to have the complete tree generated. It would take some  $10^{22}$  centuries to generate the complete checker tree, if a successor could be generated every one third of a nanosecond. So, what do we do when we have to search for solutions in such complex games?



(Refer Slide Time: 16:57)

## Why not AO\* always?

- For complex games such as chess or checkers, search to termination is impossible.
  - Our goal in searching such a game tree might be, instead, merely to find a good first move.
  - We could make the move, wait the opponent's reply and search again to find a good first move from this new position.
  - Extract from the search graph an estimate of the 'best' first move.
    - This estimation can be made by applying a static evaluation function to leaf nodes of the search graph.

© Shyamanta M Hazra, M.E., IIT Guwahati

For complex games, search to termination is impossible. So, our goal in searching such a game might be just to find a good first move, rather than the best move possible, that would take me to a win. So, we could make the move, the best possible right now and wait the opponent's reply. And then, we could search again to find a good first move from this new position. So, extract from the first graph best-first move.

That is what we want to do. This estimation, we can make by applying some evaluation function to the leaf nodes of the search graph. In order to do that, let us introduce a couple of concepts before we move on to talk about an algorithm to do whatever we have highlighted about searching complex games.

(Refer Slide Time: 18:01)

## Game Trees

- A game is represented by a game tree.
  - Game tree is a layered tree in which at each alternating level, one or the other player makes the choice.
  - Layers - MAX layers and the MIN layers.

A game starts at the root with MAX playing first and ends at the leaf node.

Leaves of a game tree are labelled with outcome of the game and the game ends there.

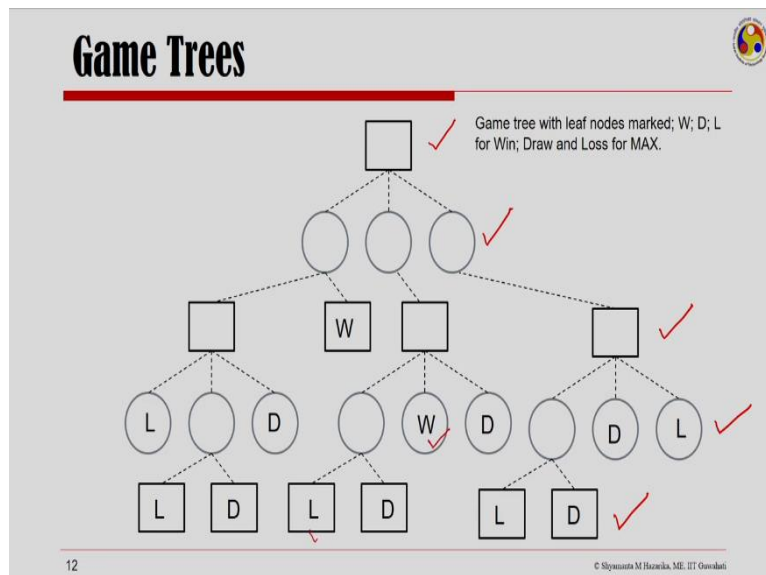
```
graph TD; Root[ ] --- C1(( )); Root --- C2(( )); Root --- C3(( )); C1 --- L1[ ]; C1 --- L2[ ]
```

© Shyamanta M Hazra, M.E., IIT Guwahati

We introduce the concept of a game tree. A game is represented by a game tree. A game tree is actually a layered tree. That is, at each alternate layers, we have the 2 players playing. So, at this layer, Max would be playing. The next layer, it is Min playing. Thereafter, it is Max again. So, a game tree is a layered tree in which at alternate level, one or the other player makes the choice.

And we have what are called the Max layer and the Min layer. Now, a game starts at the root, here, with the Max player starting the game and ends somewhere at the leaf nodes. The leaves of the game tree are labelled with outcomes of the game and the game ends there. Let us look at a game tree more closely.

**(Refer Slide Time: 19:19)**



So, here is a game tree with the first being the Max level, then the Min level. Here it is Max; then Min; Max again; Min; and Max again. Recall that we are talking of board games which have 2 players game. So, 2 players are involved. And the game is alternate moves. That means, one player makes a move. And then, the next player makes a move. So, in this game tree, the leaf nodes actually chose the possibility of win, loss or draw formats.

So, now you should recall that we are talking of a zero sum game. That is a game where one can win; then the other must lose; or it could be a draw. So here, L refers to the loss for Max; D refers to the draw; and W refers to a win for Max. So, each of that leaf nodes could be given values; either one of the 3, either W, D or L. And they will tell weather following that rule, Max wins, loses or there is a draw.

(Refer Slide Time: 21:00)

## Minimax Value

- The minimax value of the game is the backed-up value of the root from all the leaves, and represents the outcome when both the players play perfectly.
- MAX choose a move that yields that value of 1 if available; else 0 if available and will have to choose a -1 only if all the children are labelled -1.
- Backup rule for MIN is exactly the opposite.

13 © Sreyas M Hazare, MIT, IIT Guwahati

Now, let us also introduce a very important concept called the minimax value before we proceed further. So, the minimax value of a game is the backup value of the root from all the leaves; and represent the outcome when both the players play perfectly. That is, let us say we have a game tree and Max moves. He will try to choose the best possible value for himself. So, Max chooses a move that gives the value of 1, if available; else, 0 if available.

And we will choose a  $-1$ , only if all the children are labeled  $-1$ . Now, it is important for us to recall that such zero sum games, either we mark the leaves as W for win, D for draw and 1 for loss. Equivalent to that, we can either mark them as 1 for win, 0 for a draw and  $-1$  for a loss. Now, when Max wants to make a move, every time his idea would be to choose 1, to choose the maximum of the next labelable successors to him.

So, you try to get 1, if available; else 0, if available. And Max will choose a  $-1$  only if all the children are labeled  $-1$ . So, backup rule for Min is exactly the opposite of this. And the minimax value of the game is the backup value of the root from all the leaves.

(Refer Slide Time: 23:07)

## Minimax Rule

- The minimax rule backs up values from the children of a node.
  - For a MAX node, it backs up the maximum of the values of the children.
  - For a MIN node, the minimum.

L – Loss; D – Draw and W – Win

Is from the perspective of MAX; The leaves can be labelled equivalently with numbers

-1 – Loss; 0 – Draw and 1 – Win

14

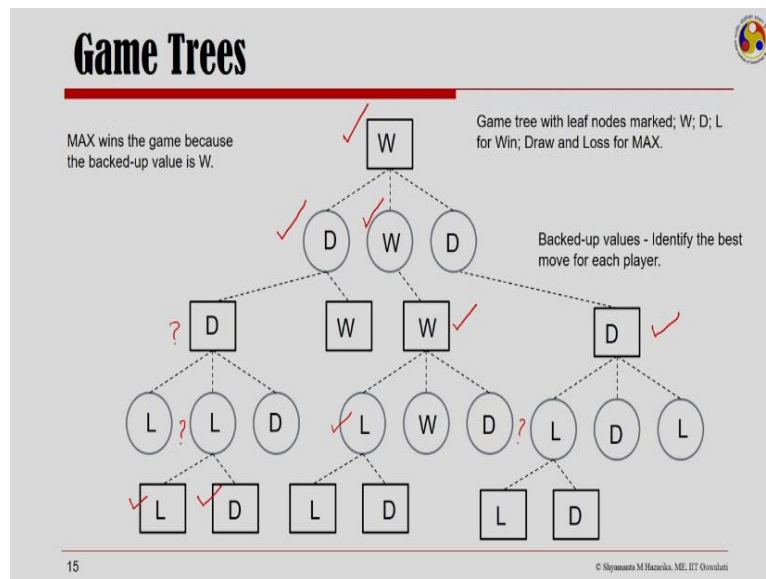
© Sreyananta M Hazarika, M.E., IIT Guwahati

So, here is example for understanding what the minmax rule does. The minmax rule backs up values from the children of a node. So, for a Max node, it backs up the maximum values for the children. For a Min node, it backs up the minimum. Like, here is the root node and Max is playing at the game. Now, we have a leaf node marked L and we have another leaf node marked D. There are leaf nodes marked D and L respectively here.

Now, remember that this level, it is Min who is playing. At his level it is Max who is playing the game. So, when given their children L and D, Min has to backup this value. So, it will take the minimum of the values and back it up as L. Now, this one here, is Max's start. And Max wants to maximize the value of the children. And therefore, it will try to get a backup value of D.

So, in terms of its equivalent numbers like - 1 for loss, 0 for a draw, the Min player playing here would back up the minimum value, which is - 1. Given - 1, - 1 and 0, the Max player would love to back up the maximum value which is 0.

(Refer Slide Time: 25:26)



So, here is a game tree and we have marked all the leaf nodes as shown. Now, we would love to back up the values. That is, identify the best move for each player. Given L and D here, the backup value at this point will be an L. So, the backup value at this point is an L again. The backup value here is an L, because this is the Min's level. Now, at this point, it is Max's play. So, the backup value would be D.

Here, the backup value would be W. And at this point, the backup value is D again. This point here, it is Min's choice. So, given a choice between D and W, Min would love to backup D. At this node, it is interesting to note that the only available backup value for Min is W and it is forced to backup to W. It would have loved to backup to L first, then D. But, having no other children except W, it has to backup to W.

So, it backs up to D. And then, finally we have Max with W. So, what this game is showing here is that Max wins this game, because the backup value under the current scenario of leaf nodes is W.

(Refer Slide Time: 27:24)

**Strategy**

- A strategy is the subtree of a game tree
  - ✓ That have one choice for MAX.
  - ✓ All choices for MIN.
- Strategy freezes the choice for the player.
  - Outcome of the game depends on the opponent.

Game playing program produce moves for the player; traditionally MAX.  
Minimax value determines the best MAX can do!

Traverse the tree starting at the root

- ✓ If at the MAX Level  
then Choose ONE branch below it
- ✓ If at the MIN Level  
then Choose ALL branches below it

Return the SUB-TREE so constructed.

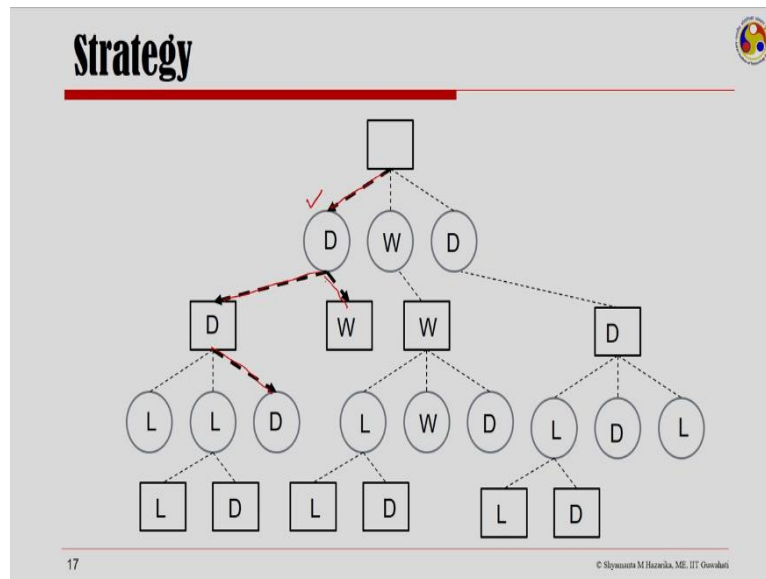
16 © Sreyasanth M Hazareika, M.E., IIT Guwahati

So now, let us understand what do we mean by strategy of playing a game. Given the game tree, the strategy is a subtree. But then, which subtree are we talking about? We are talking about the subtree where Max has only one choice. But for Min, I must explore all choices. So, a strategy is a subtree of a game tree that has one choice for Max and all choices for Min. Once we have a strategy, it actually means that the choice for the player is frozen.

And, the outcome of the game therefore would depend on the opponent. So, most game playing programs produce moves for the player, traditionally Max. Our minimum values determine the best that we can have for the Min player. Maximum values of the children determine what we can have for the Max player. So, the idea of strategy is about doing 2 things. You traverse the tree starting at the root.

If you are at the Max level, then you choose one branch below it. And if you are at the Min level, then you choose all the branches below it. And you return the subtree so constructed. And the subtree gives you a strategy.

(Refer Slide Time: 29:12)



So, here is the game tree that we have been looking at. And the dark arcs that I have marked here with one choice for Max, all possible choices for Min and again one choice for Max is a strategy for Max. So now, having defined what strategy is, how do one ensures to find out an optimal strategy.

(Refer Slide Time: 29:40)

### Optimal Strategy

- Selection of the optimal strategy requires solving the game tree.
  - Search techniques analogous to those used for finding AND/OR solution graph.
  - Games with small trees completely solved.
    - Tic-Tac-Toe - Always Draw; when BOTH play perfectly.
  - Not always feasible!
- For more complex games such as chess or checker the AND/OR search to termination is out of question.
  - Complete game tree for chess has approximately  $10^{40}$  nodes.
  - It would take  $10^{22}$  centuries to generate the complete checker tree, if a successor could be generated every 1/3 of a nanosecond!

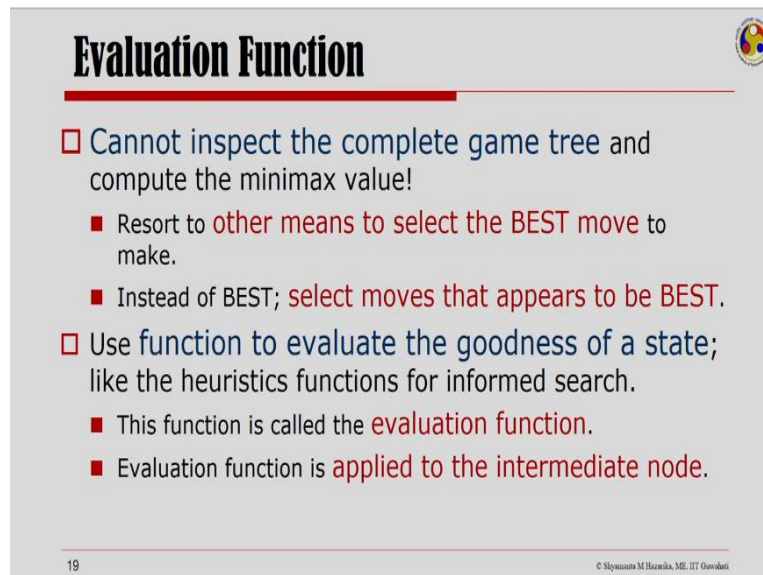
Selection of the optimal strategy requires solving the game tree. And as we have seen for a very simple game like Grundy's game, search techniques analogous to those used for finding AND-OR solution graph is good enough. But, this is true only for games with very small trees. So, games with small trees can be solved completely and we can apply AND-OR solution graph, searching techniques to find out the best strategy.

This is true for even a game like tic-tac-toe. So, a tic-tac-toe game, we can always ensure a draw when both the players play perfectly. So, let me remind you what the tic-tac-toe game is, if you have forgotten from your school days. So, you are supposed to create a matrix and you give an X. Your opponent gives an O. Thereafter, you give an X and your opponent gives an O. You give an X. And at that point, you win the game.

So, such small games like the Grundy's game and games like tic-tac-toe; we can always ensure draw when both players play perfectly for tic-tac-toe. But for complex games like chess, checkers, it is not feasible to have the complete game tree. For more complex games AND-OR search to termination is therefore out of question. Complete game trees, as we have seen for games like chess, involves  $10$  to the power  $40$  nodes.

Therefore, there must be some other trick that I need to apply to get to an optimal strategy for finding out a winning move. This is where we take help of evaluation functions. As we cannot inspect the complete game tree and compute the minimax value, we resort to other means to select the best move.

**(Refer Slide Time: 32:05)**



**Evaluation Function**

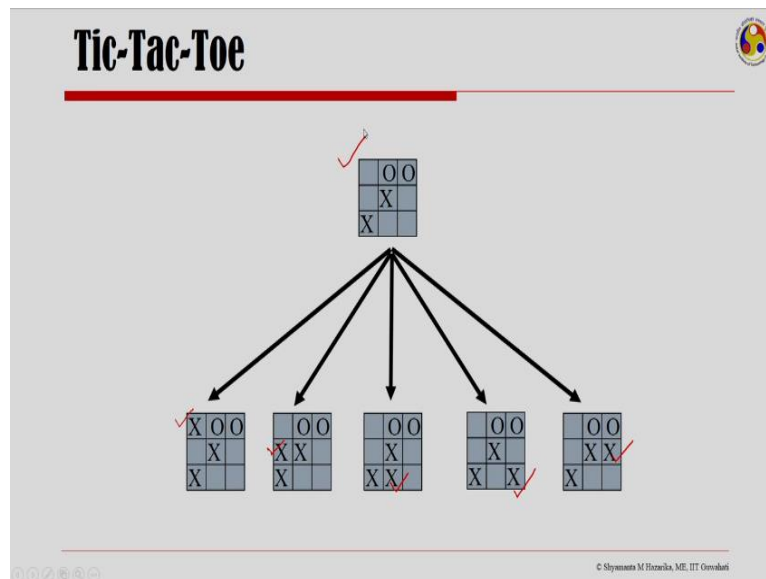
- Cannot inspect the complete game tree and compute the minimax value!
  - Resort to other means to select the BEST move to make.
  - Instead of BEST; select moves that appears to be BEST.
- Use function to evaluate the goodness of a state; like the heuristics functions for informed search.
  - This function is called the evaluation function.
  - Evaluation function is applied to the intermediate node.

19 © Sreyananta M Hazarika, M.E., IIT Guwahati

In fact, we do not select the best move. We select the move that appears to be best for the current move. And then, as we get more information, we revise and take the next best move. So, we use a function to evaluate the goodness of a state. This function is something like the heuristic function for informed search. And this function is called the evaluation function. Evaluation function is applied to the intermediate nodes and they allow us to pick up nodes that appears best.



(Refer Slide Time: 32:50)



So, let us go back to our game of tic-tac-toe and try to understand the use of an evaluation function. Let us say this is the current position of a game of tic-tac-toe. So, I now have to give an X. So, the possibilities of Max is that he can give an X at this corner or an X at the middle. There is a possibility of Max putting his cross here; and at this location or at this location. So, we have 5 possibilities for Max in this current scenario. Question is, which is the best move for Max to go from here?

(Refer Slide Time: 33:59)

The slide is titled "Tic-Tac-Toe" and features a logo in the top right corner. It displays the text "Evaluation Function" in blue. Below this, the following definitions are provided:

- $e(p)$  = number of directions open for Max - number of directions open for Min
- $e(p)$  = + inf if win for Max ✓
- $e(p)$  = - inf if win for Min ✓

Below the definitions, the calculation is shown:

$$e(p) = 6 - 4 = 2$$

The number 2 is circled in red. To the right of the equation is a 3x3 grid representing the current game state:

	O	
	X	

© Sreyananta M Hazarika, ME, IIT Guwahati

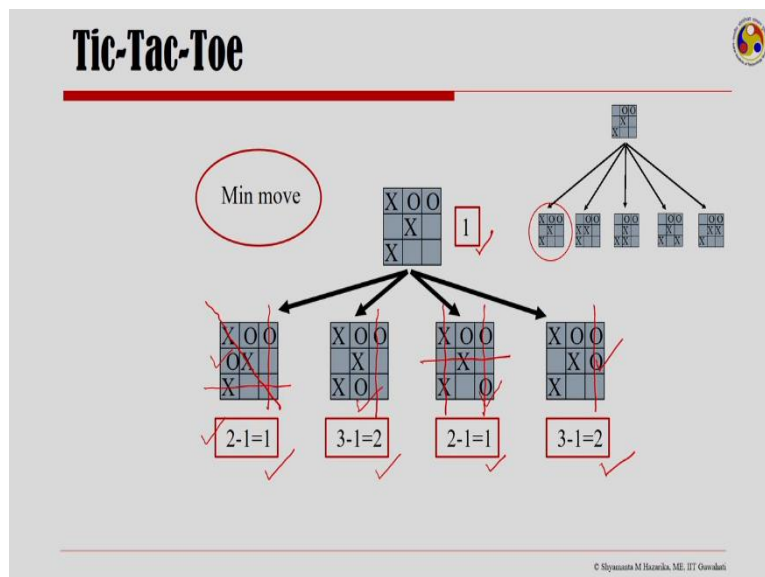
In order to answer that question, we will use an evaluation function for tic-tac-toe. So, let us look back at this tic-tac-toe node. We defined an evaluation function which is the number of directions open for Max minus the directions open for Min. Now, in this position shown on

your screen, you have directions open for Max as this one: 1, 2, 3, 4, 5 and 6. So, the directions open for Max are 6.

Now, let us take a moment and see what are the directions possible for Min. When I say directions possible for Max or Min, I mean here directions that could ensure a win for the player. So, we have one direction in which Min could be in this game; 2, 3 and 4. So, the number of directions which are open for Min is 4. And therefore, we would evaluate this current state configuration as 2 for Max.

Now, there are 2 interesting values that one needs to understand. I say the value of the evaluation function is positive infinity if it ensures that I have a win for Max. And if there is a win for Min, I say that evaluation function evaluates to negative infinity.

**(Refer Slide Time: 36:01)**



Given this definition of an evaluation function and given the fact that I am looking for the best-first move for my Max player, given a state configuration as shown; and want to evaluate the 5 configurations one after another, let us try to play for Min. So, once I have come to this location marked with a circle on the screen, let us see what are the different moves possible for Min.

So, as shown here, Min could have 0 at this extreme left; or a 0 in the middle bottom row, 0 at the bottom of the right; or a zero at the middle of the right. Using the evaluation functions that we have defined, we would compute the values. So, number of positions open for Max at

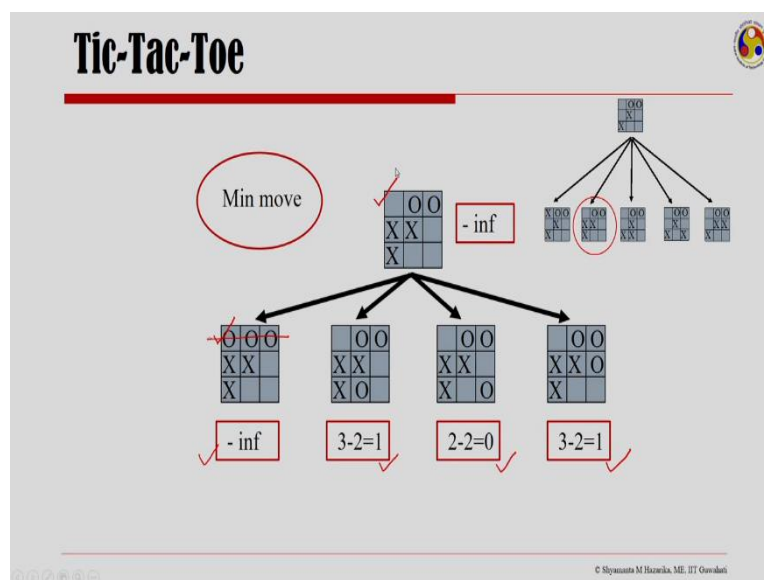
this location is 1 and 2. So, we have a 2 there. Number of directions which is open for Min at this location is this.

So, we have  $2 - 1, 1$  as the value of the evaluation function. The number of directions open for Max is 1, 2 and 3. The number of directions open for Min is only this, which is 1. And therefore, the evaluation function is 2. So, continuing in similar way, I have 2 directions open for Max and only a single direction open for Min. So, that evaluates to 1. Here I have 3 directions open for Max: 1, 2 and 3.

Whereas, I have only a single direction open for Min. So, it evaluates to 2. So, these are the evaluation functions. Now remember, after I have given this move here, that means after Max moves here, it is Min's move. And Min would love to minimize the value of the successor node. So, of these evaluated values, Min would pick up 1 that gives it a value of 1. So, Min would either move here or it would move here.

So, for our evaluation of which is the best move for me, or which is the best move for Max; when I say me here, I mean the player Max. So, which is the best move for Max at this point. I would try to figure out which is the best possible allowable value once I have moved there by Min. So, we see that the value that I get for that particular position that I move to is 1.

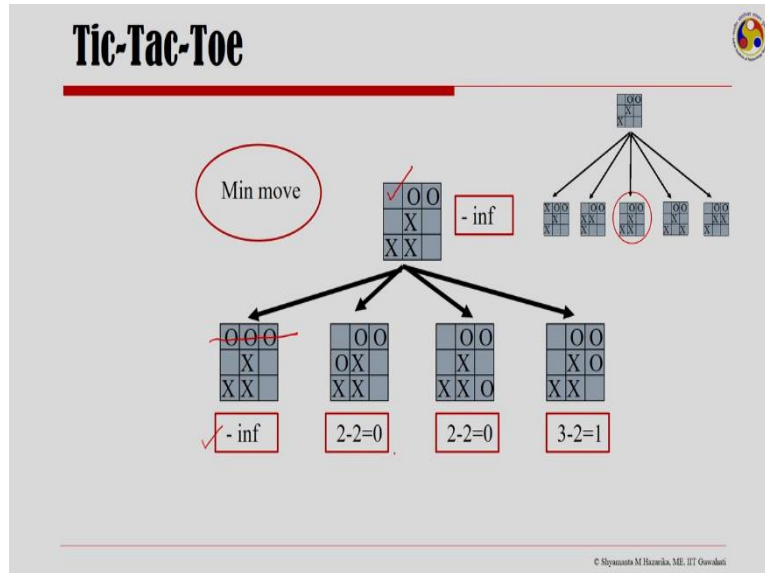
**(Refer Slide Time: 39:38)**



I could then move to the second node. And it is interesting to see that at this node, the moment I allow Min to play, Min could mark this position and it would be in the game. So, the evaluated value is  $-\infty$ . I could evaluate the different values for the different moves

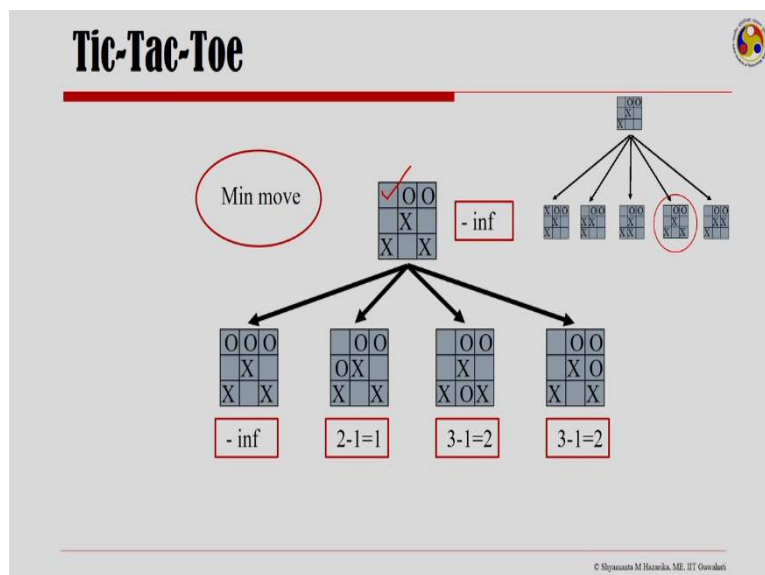
of the Min node to be 1, 0 and 1; as I have explained in the previous move. But then, Min would love to pick up the value that has the minimum. And therefore, this node here would be  $-\infty$ .

(Refer Slide Time: 40:33)



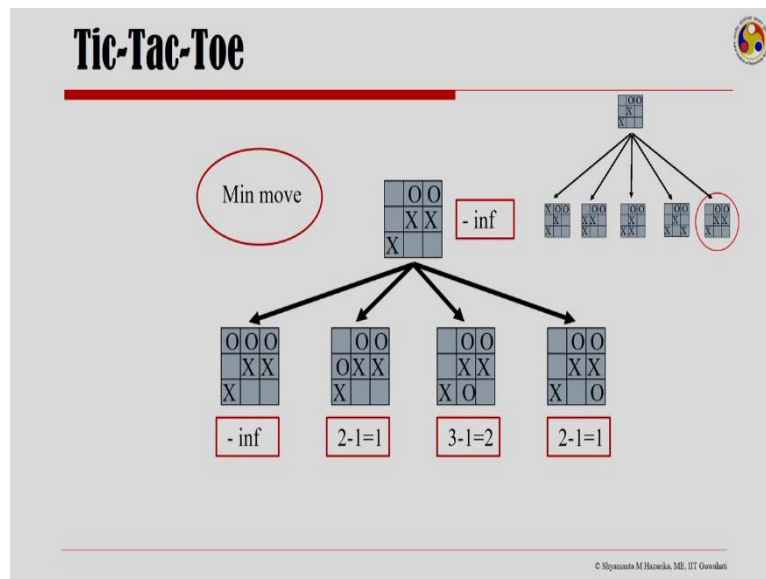
We can move ahead and look at the next node which again gives you the value of  $-\infty$ . As you can see if Max do not fill up this, Min would immediately fill that up and ensure a win. And this will have a  $-\infty$  value for the evaluated function. Whereas, rest of them could be 0 or 1.

(Refer Slide Time: 40:58)



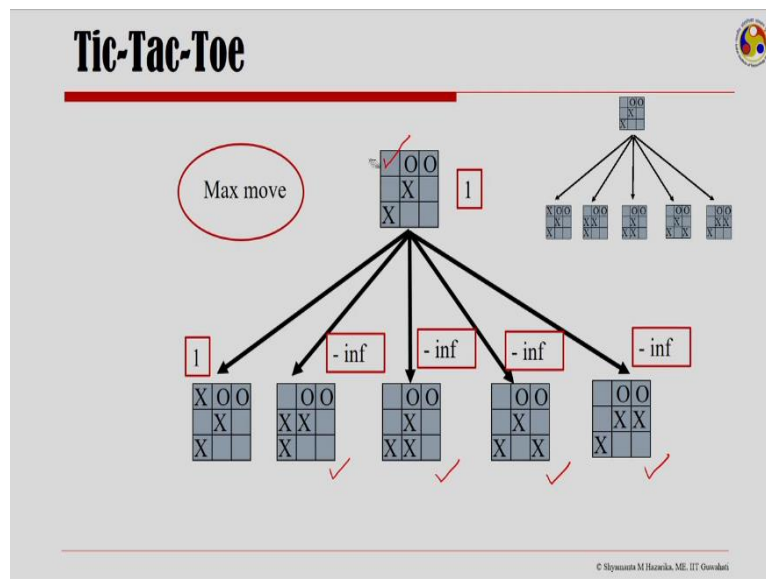
We compute for the fourth option and we see similar values, because this is left out. If this is left out, we will see that there is a win for Min.

(Refer Slide Time: 41:11)



And similarly, for the fifth move. Now, it is interesting to note that, if I start with a move that does not block this winning position for Min, the next move that Min would give is to ensure a win. And the backed-up value for all these nodes will therefore be – infinity.

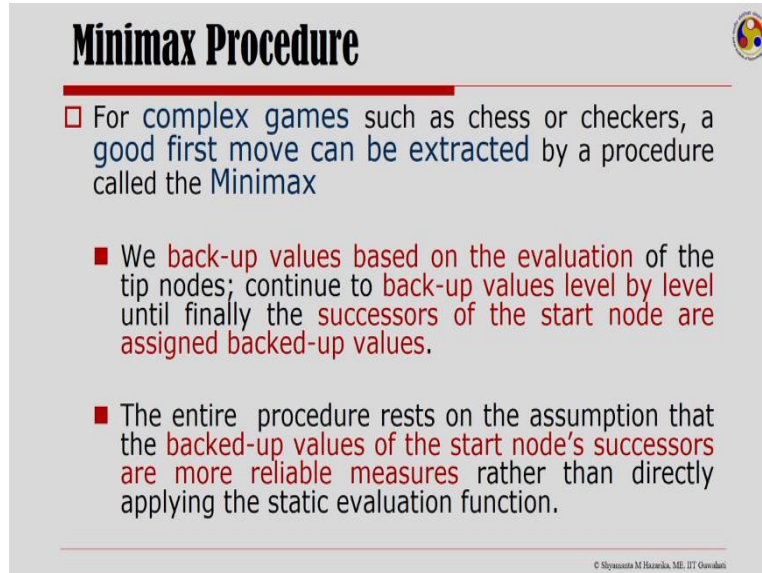
(Refer Slide Time: 41:37)



This is what I have shown in the slide. The backed-up values for each of them based on Min's next move. The only possibility of Min not winning is when I close this position in my next move. And that gives me an evaluated value of 1. So, it is Max's move and I want to Maximize the value of the evaluation function. So, under this scenario I would love to place an X in this particular position.

This is what is done by computing the evaluation function. Now, we have not looked at what is the best path to take to ensure win for Max. But what we have done is computed some value for the next level and somehow backed up those values to see what is the current best possible move.

(Refer Slide Time: 42:59)



## Minimax Procedure

- For complex games such as chess or checkers, a good first move can be extracted by a procedure called the Minimax
  - We back-up values based on the evaluation of the tip nodes; continue to back-up values level by level until finally the successors of the start node are assigned backed-up values.
  - The entire procedure rests on the assumption that the backed-up values of the start node's successors are more reliable measures rather than directly applying the static evaluation function.

© Sreyasanta M Hazarika, M.E., IIT Guwahati

And this is what we will deal with in the next lecture, where we will be talking of a procedure called the minimax procedure. So, for complex games such as chess or checkers, we have been emphasizing that it is not possible to extract the best move. So, a good first move is all that I can extract. And this I do by a procedure call minimax. What we do in this is, we backup values on the evaluation of the tip nodes; and continue to backup values level by level until finally the successor of the start node are assigned backed up values.

The entire procedure actually rests on the assumption that the backup values that I get are more reliable measure, rather than directly applying the static evaluation function. This is what we will see in the next lecture. Thank you very much.