ME 620: Fundamentals of Artificial Intelligence

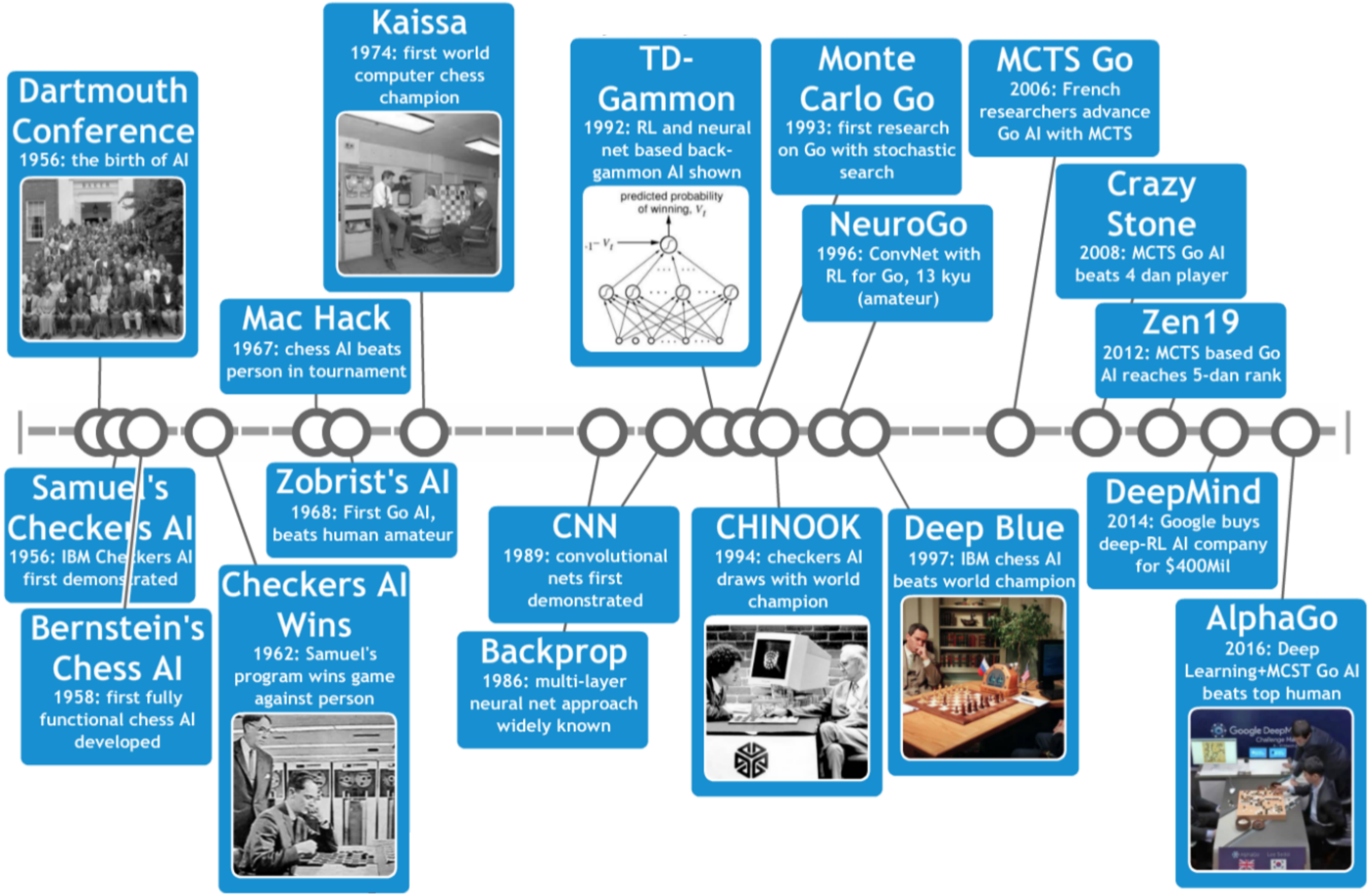
Lecture 11: Game Playing and Minimax Procedure

Shyamanta M Hazarika
Biomimetic Robotics and Artificial Intelligence Lab
Mechanical Engineering and M F School of Data Sc. & AI
IIT Guwahati

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 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.

From Checkers to AlphaGo



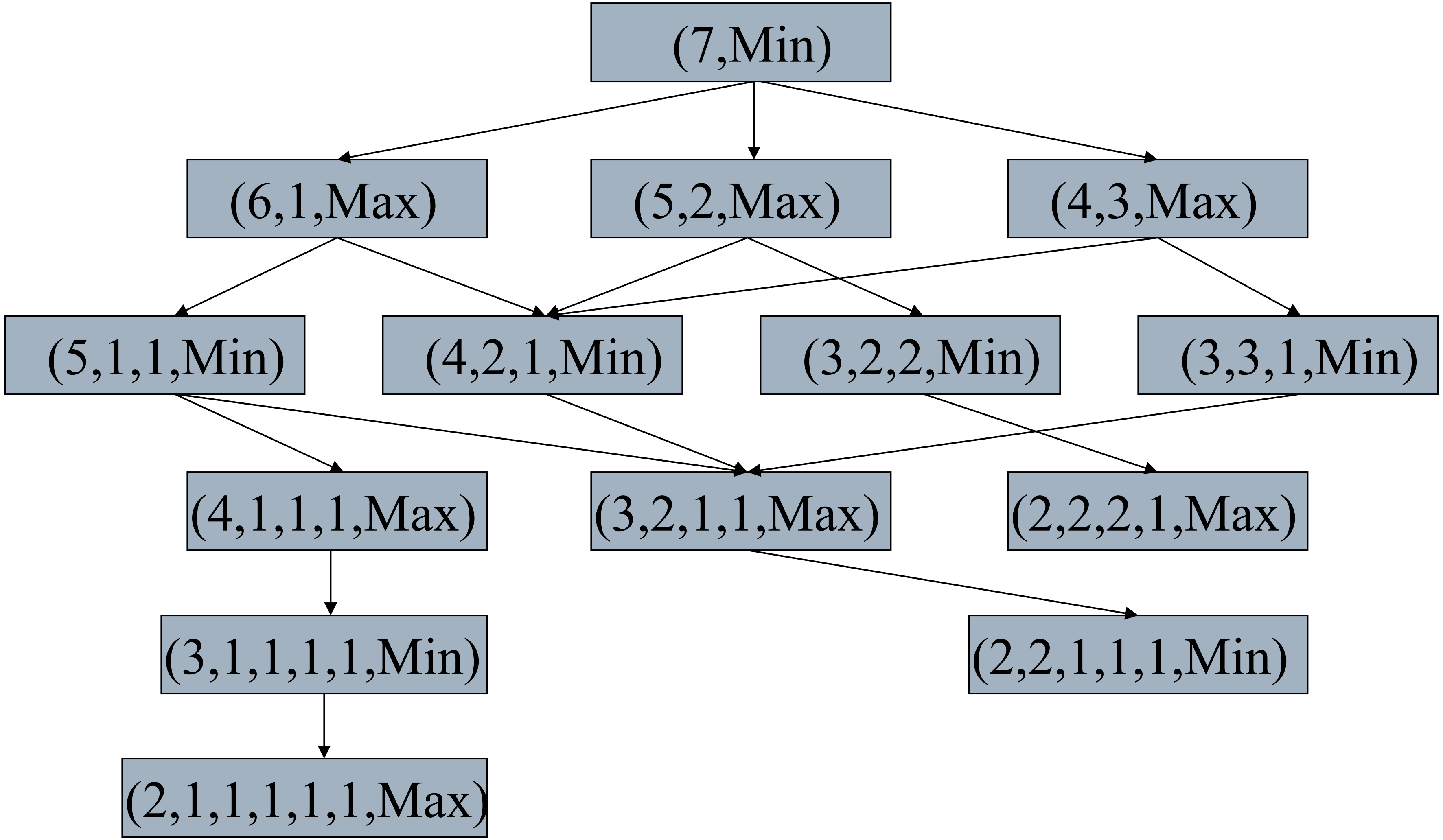
Board Games

- 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 play has.
- Alternate moves
  - Players take turns to make their moves.
- Deterministic Games
  - No element of chance in the moves that one can make.

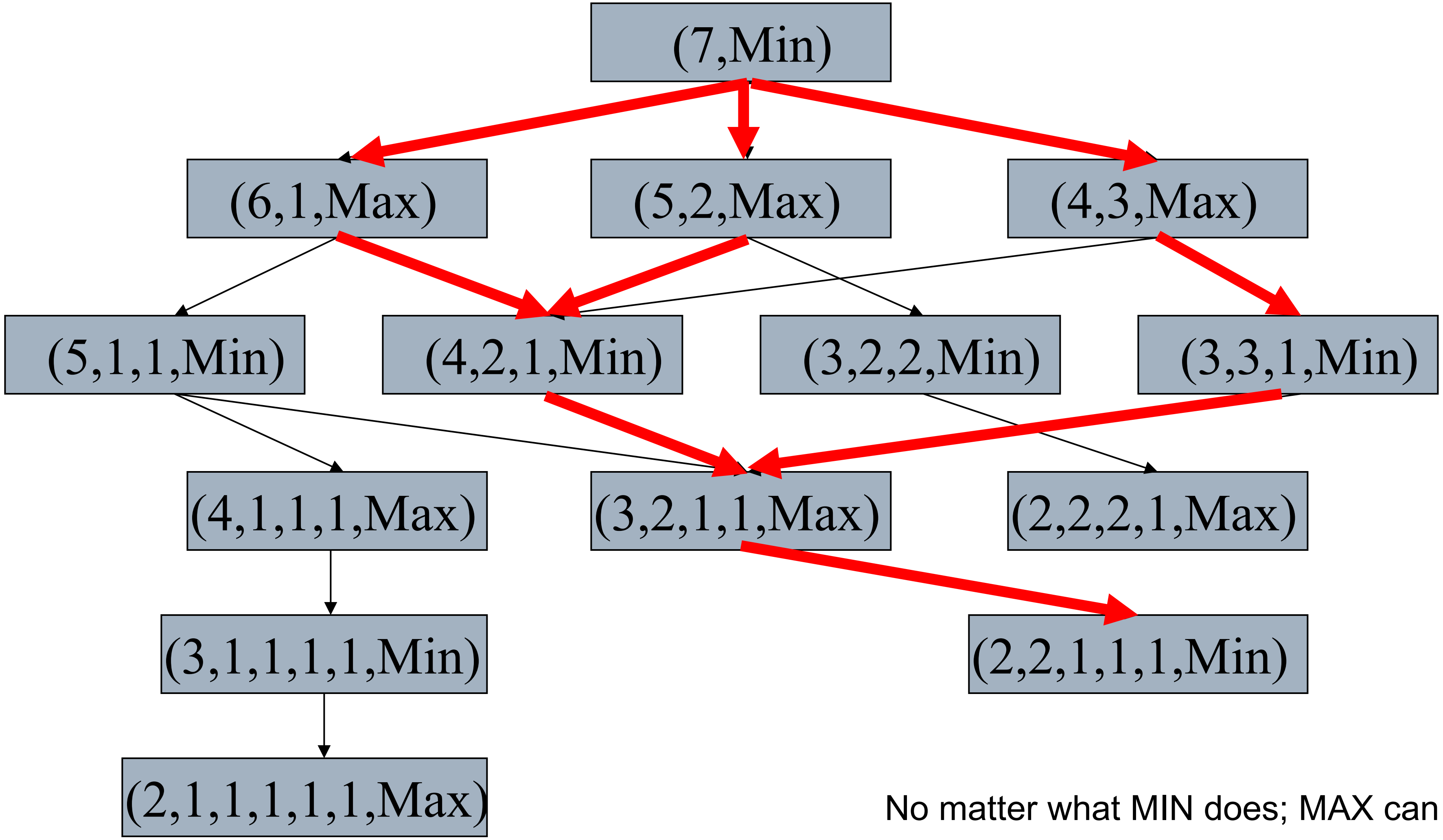
Grundy's Game

- 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!

Grundy's Game



Winning strategy for MAX

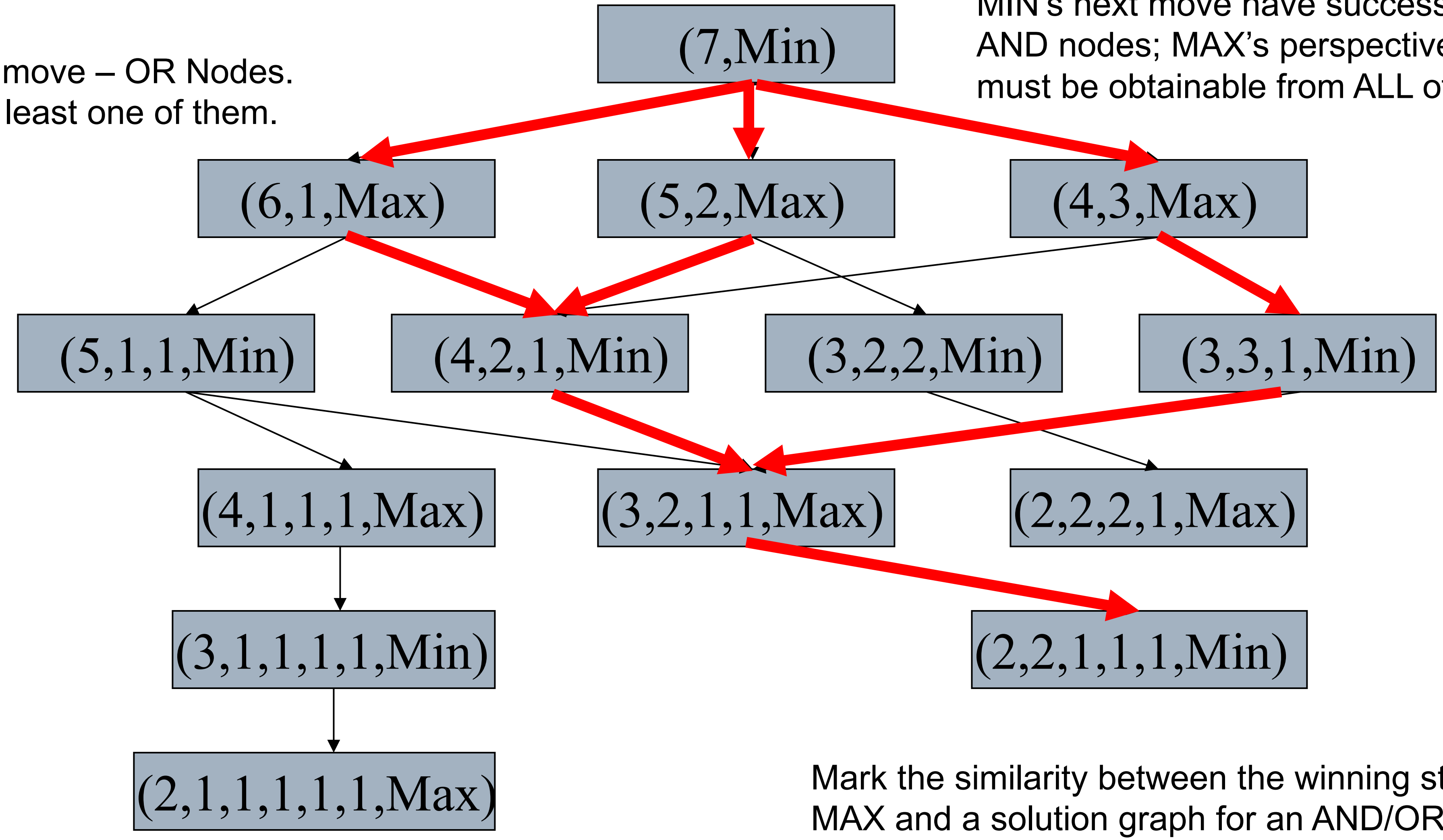


No matter what MIN does; MAX can always win!

Winning strategy - AND/OR Graph?

MAX's next move – OR Nodes.
Win from at least one of them.

MIN's next move have successors that are like AND nodes; MAX's perspective – Solution must be obtainable from ALL of these nodes.



Mark the similarity between the winning strategy for MAX and a solution graph for an AND/OR graph.

Why not AO* always?

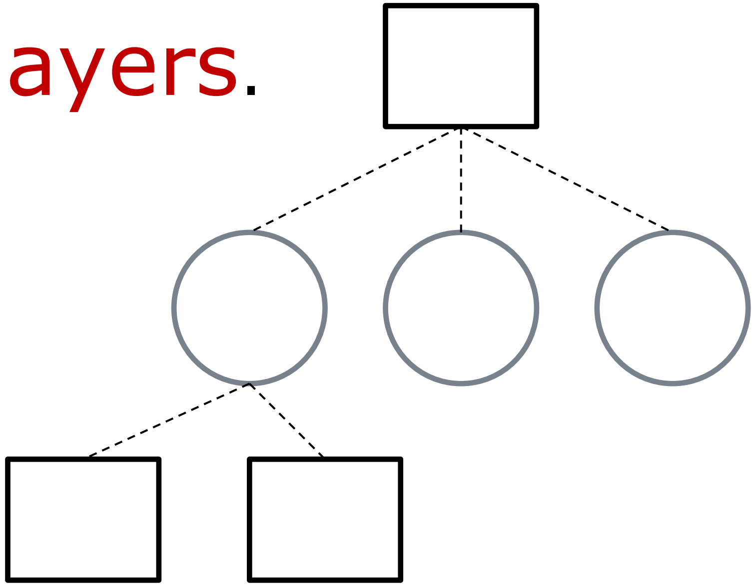
- Many simple games can be handled by search techniques that are analogous to those used for finding AND/OR solution graph
- For more complex games such as chess or checker the AND/OR search to termination is out of question.
  - Complete game tree for chess has approximately 10^{40} nodes.
  - It would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

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.

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.

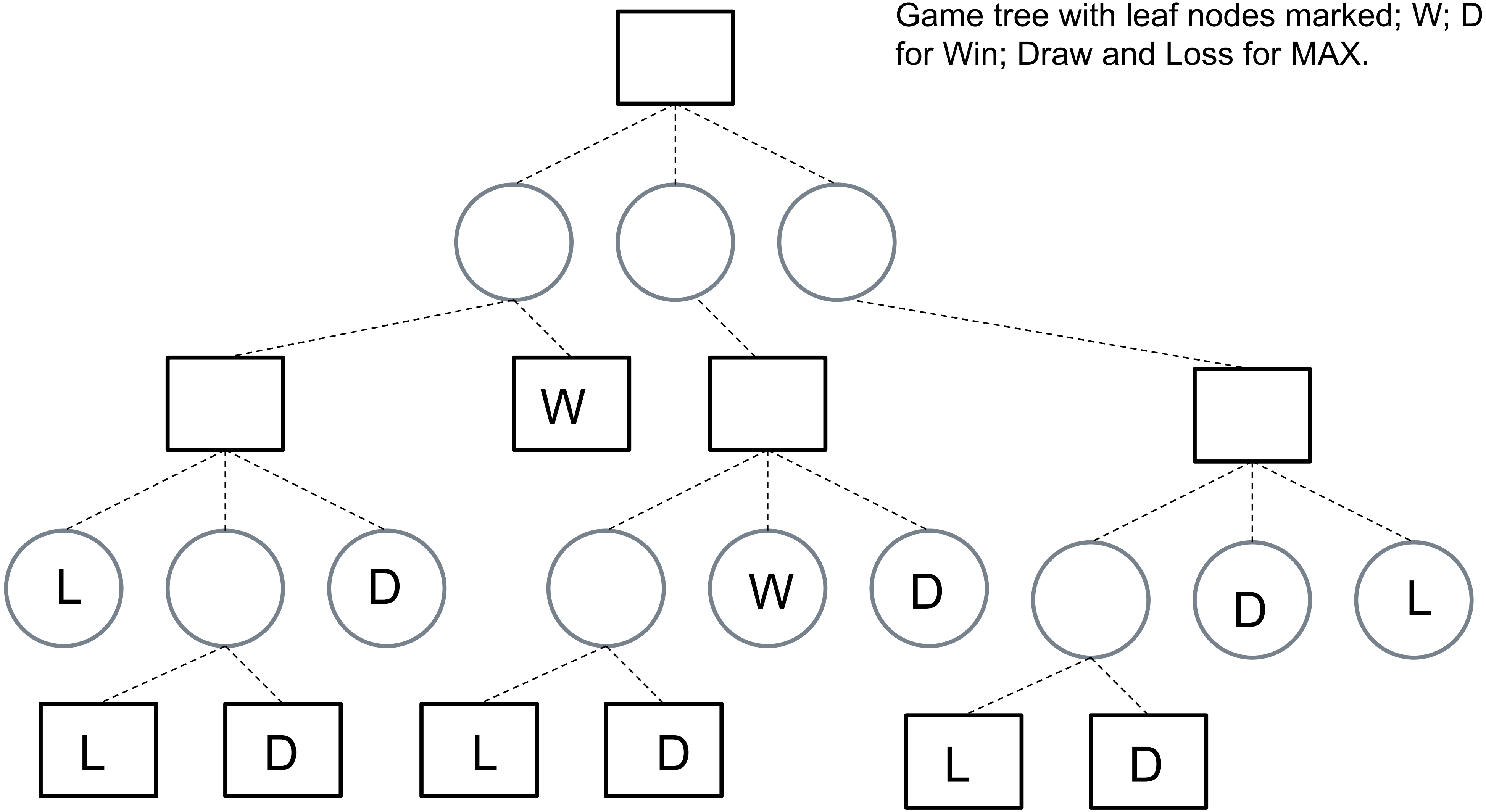


A game starts at the root with MAX playing first and ends at the leaf node.

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

Game Trees

Game tree with leaf nodes marked; W; D; L for Win; Draw and Loss for MAX.



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.

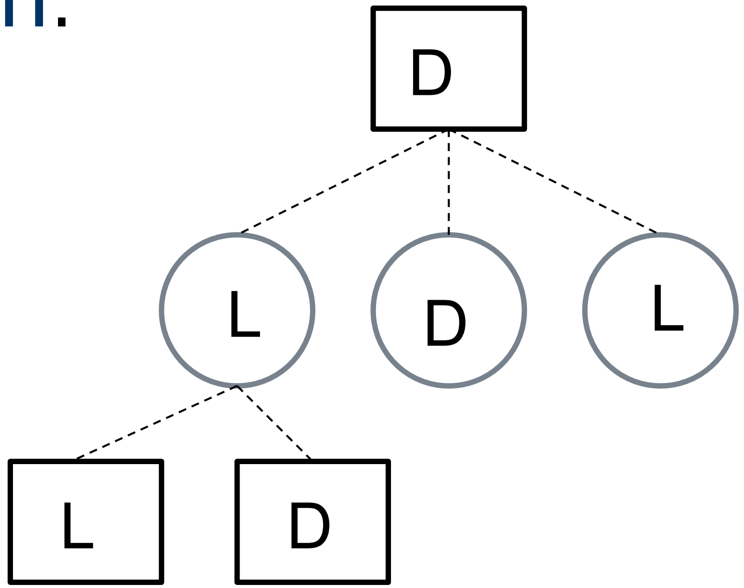
Minimax Rule

- 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

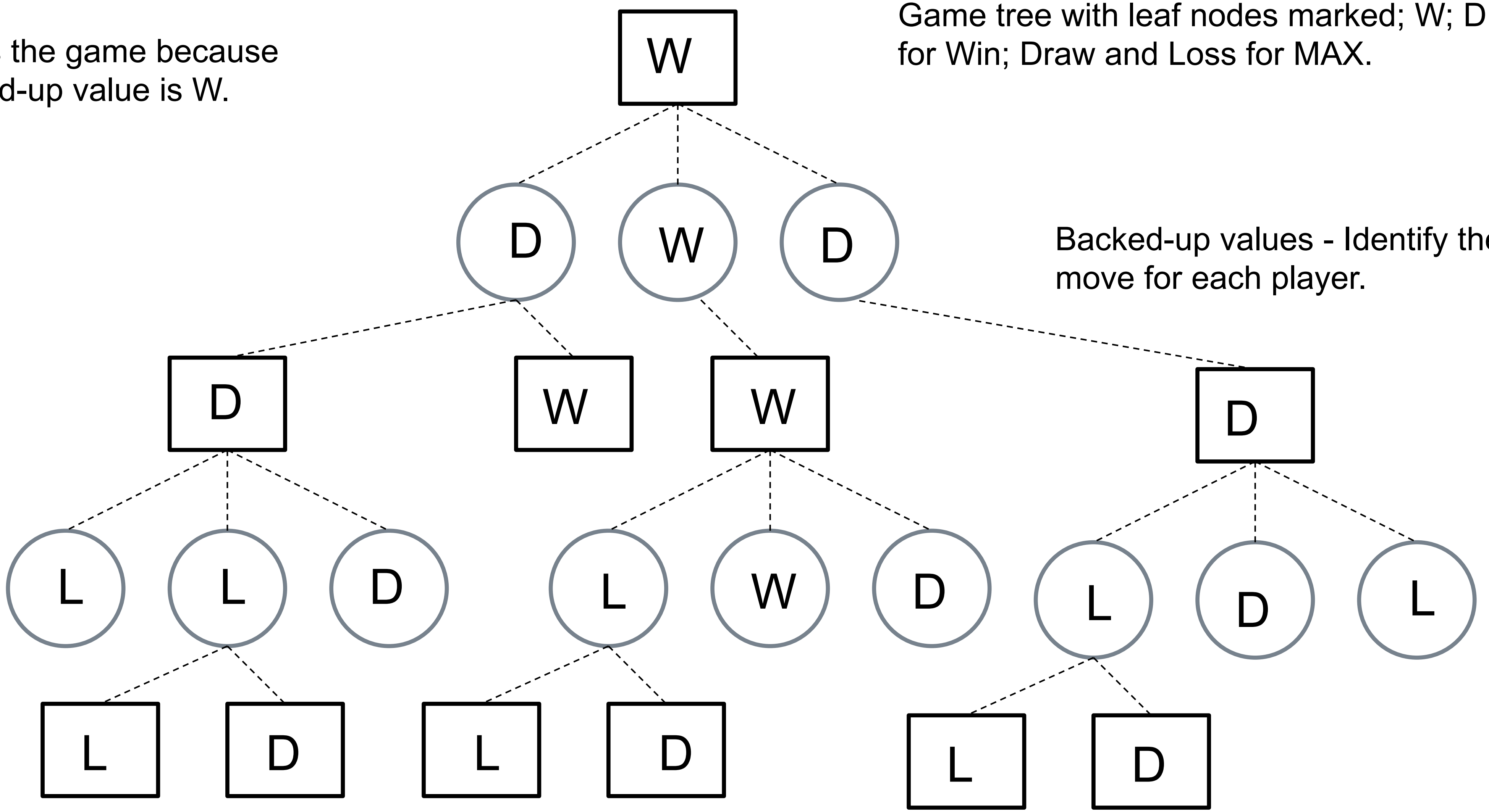
Is from the perspective of MAX; The leaves can be labelled equivalently with numbers

-1 – Loss; 0 – Draw and 1 – Win



Game Trees

MAX wins the game because the backed-up value is W.



Game tree with leaf nodes marked; W; D; L for Win; Draw and Loss for MAX.

Backed-up values - Identify the best move for each player.

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

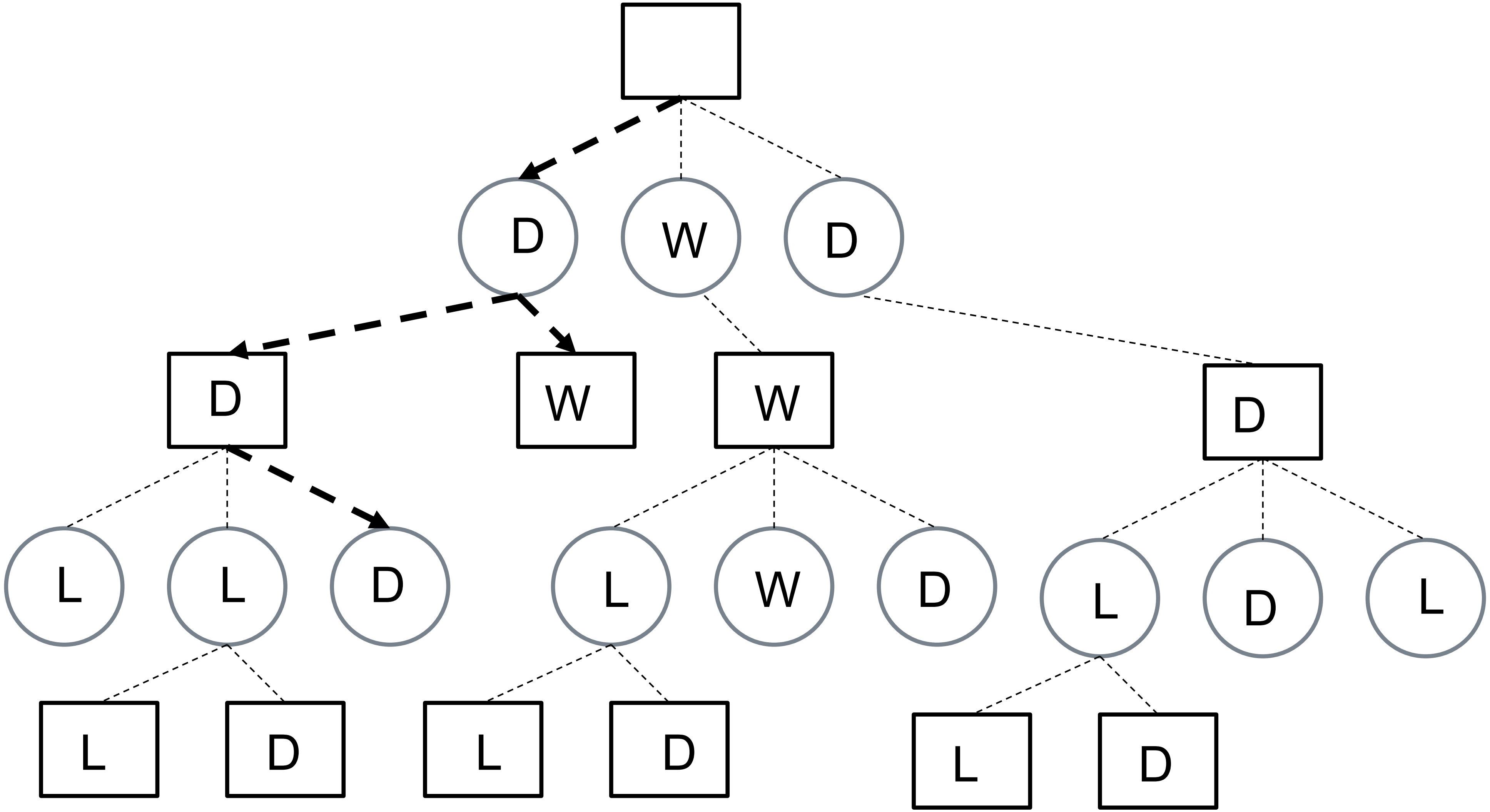
then Choose ONE branch below it

If at the MIN Level

then Choose ALL branches below it

Return the SUB-TREE so constructed.

Strategy



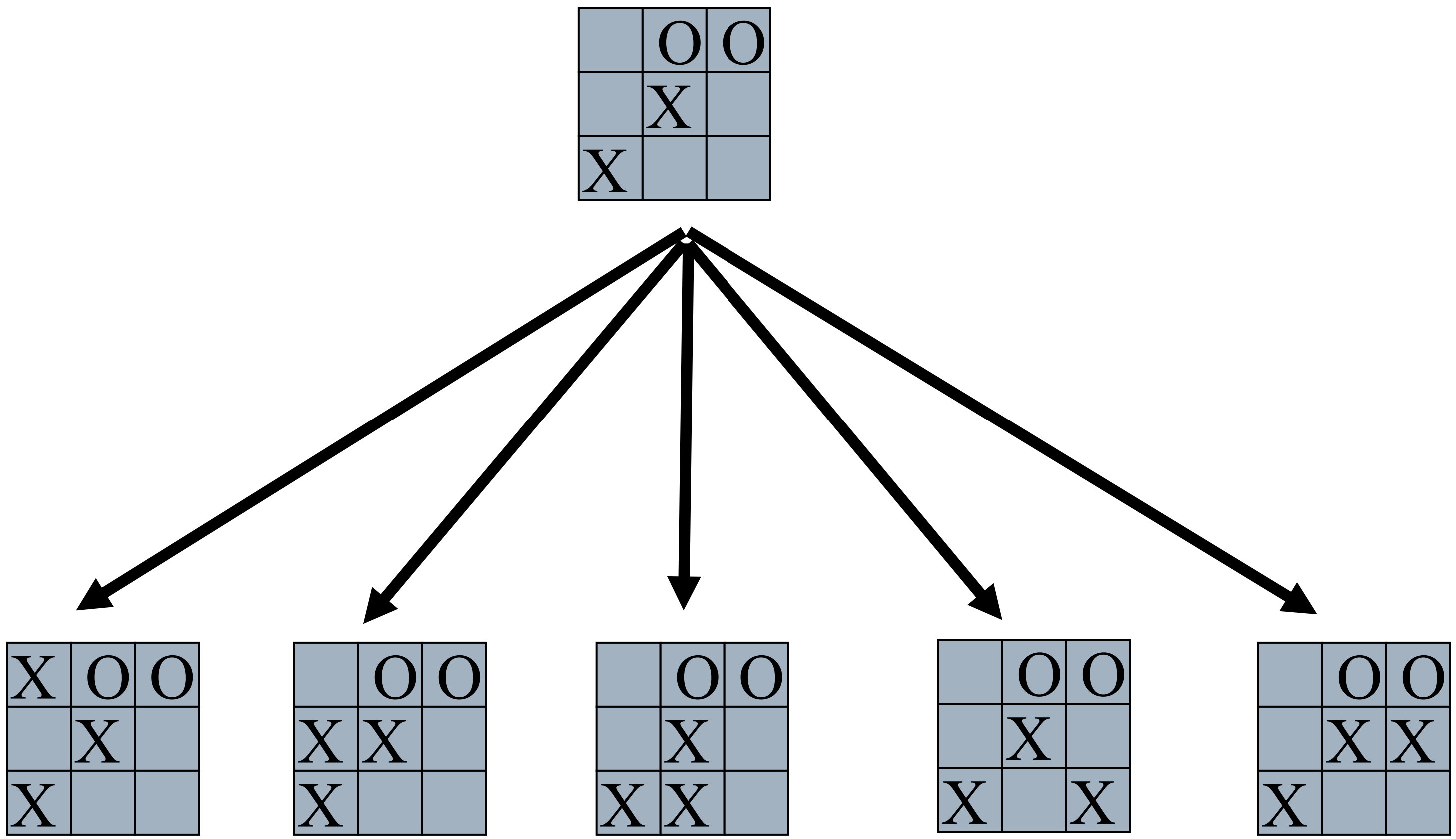
Optimal Strategy

- 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!

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.

Tic-Tac-Toe



Tic-Tac-Toe

Evaluation Function

e(p) = number of directions open for Max –
number of directions open for Min

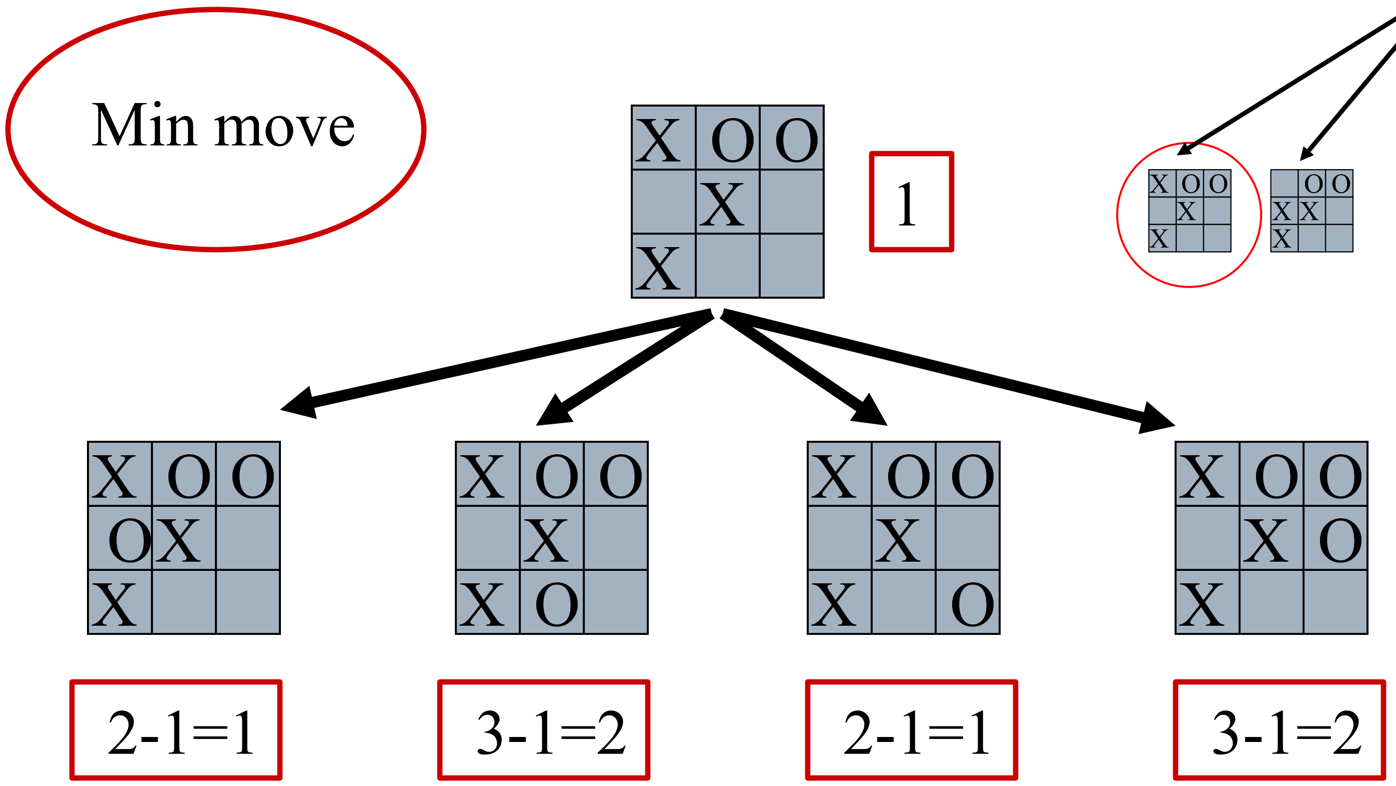
e(p) = + inf if win for Max

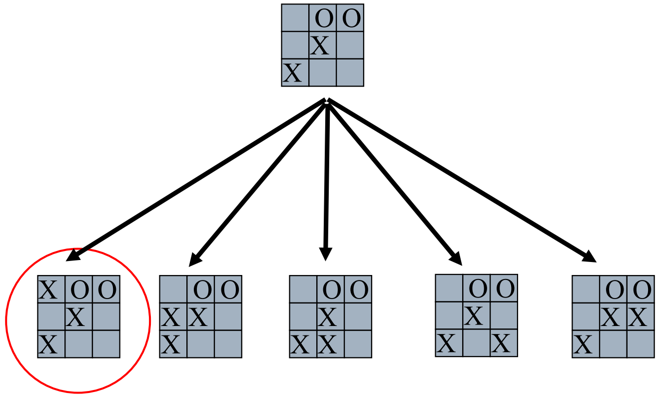
e(p) = - inf if win for Min

e(p) = 6 – 4 = 2

	O	
	X	

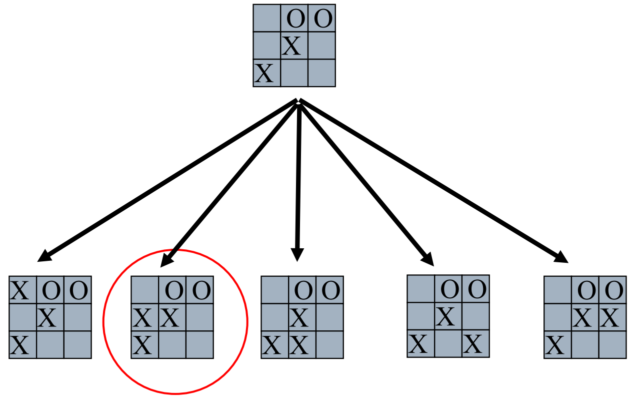
Tic-Tac-Toe

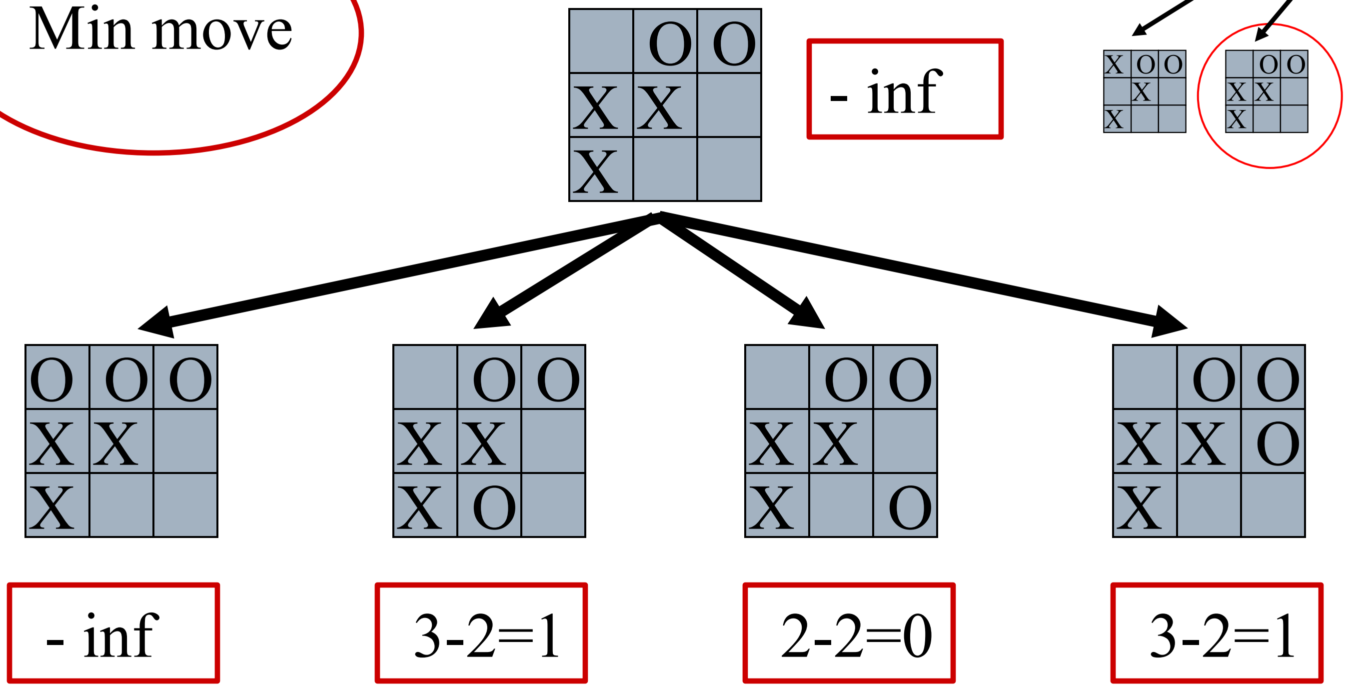




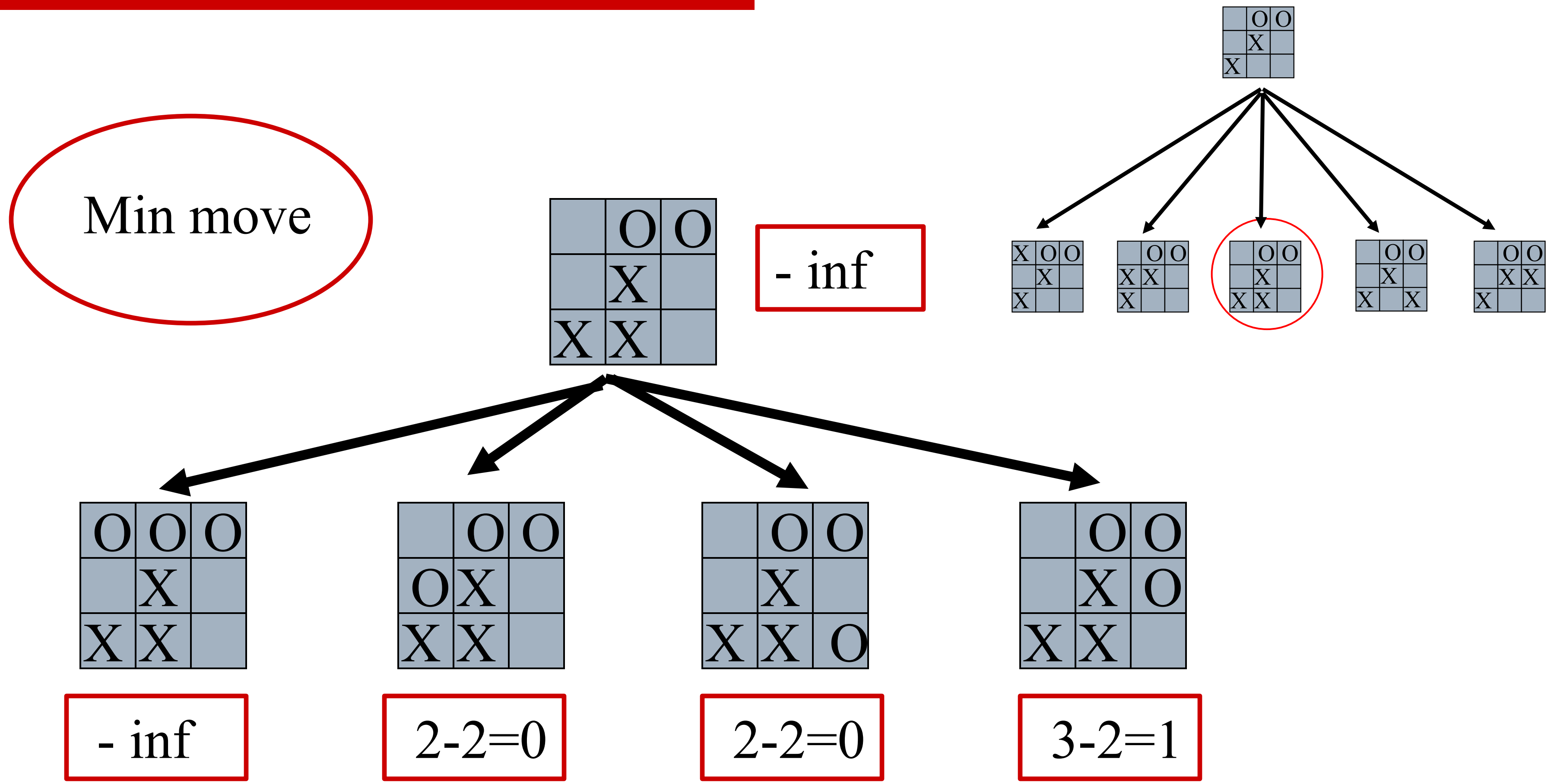
Tic-Tac-Toe

Min move



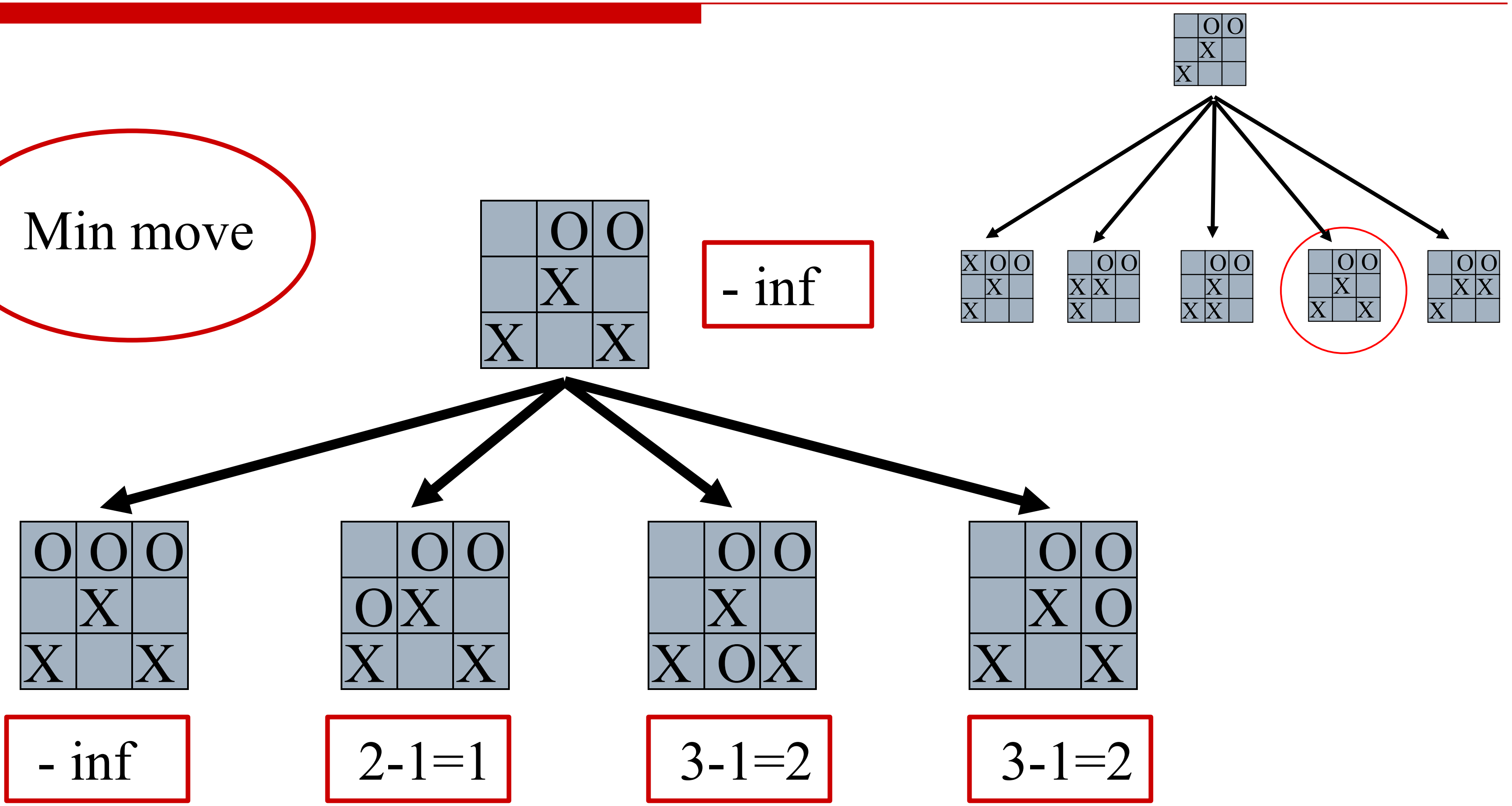


Tic-Tac-Toe

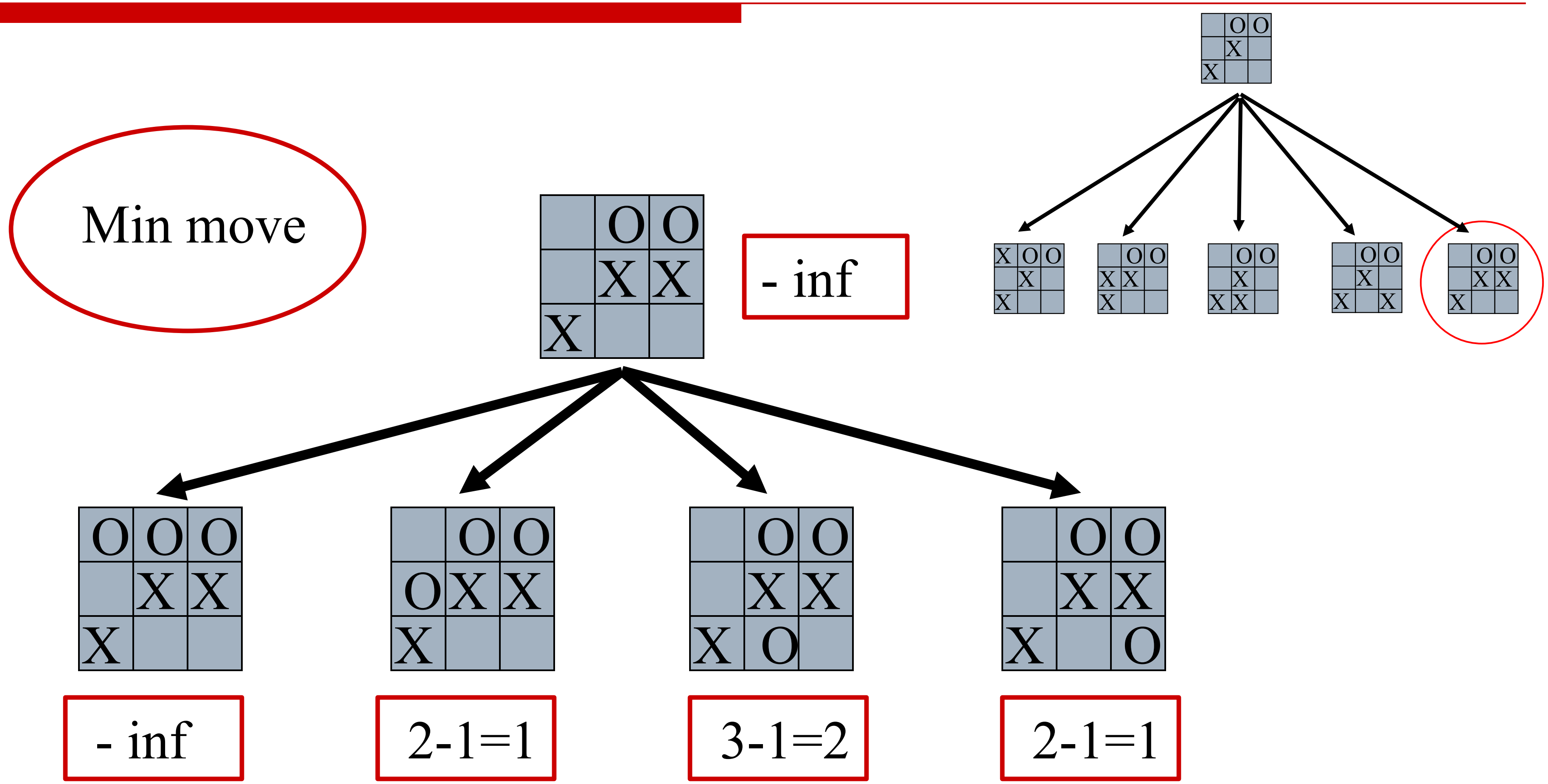


Tic-Tac-Toe

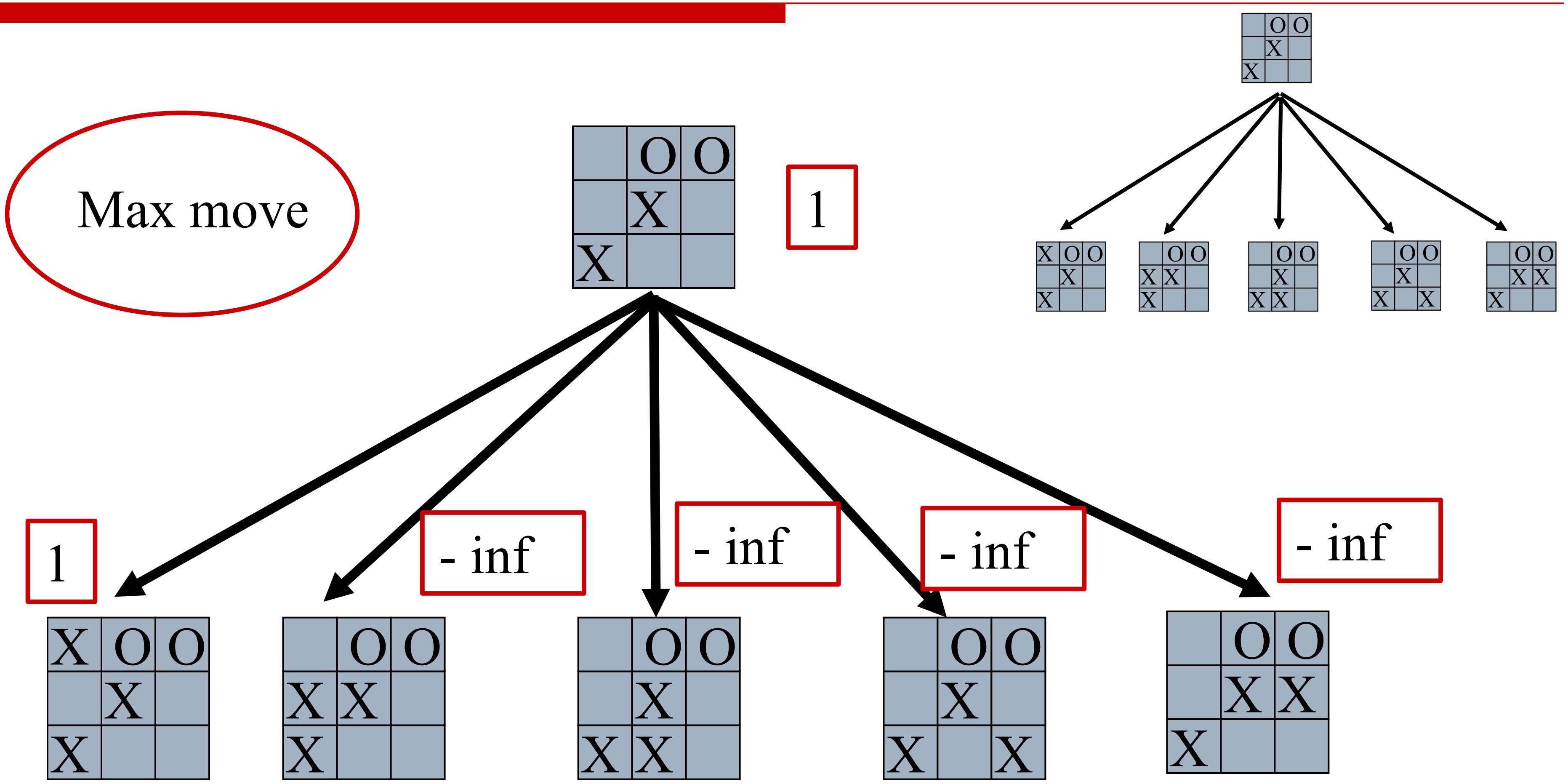
Min move



Tic-Tac-Toe



Tic-Tac-Toe



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.