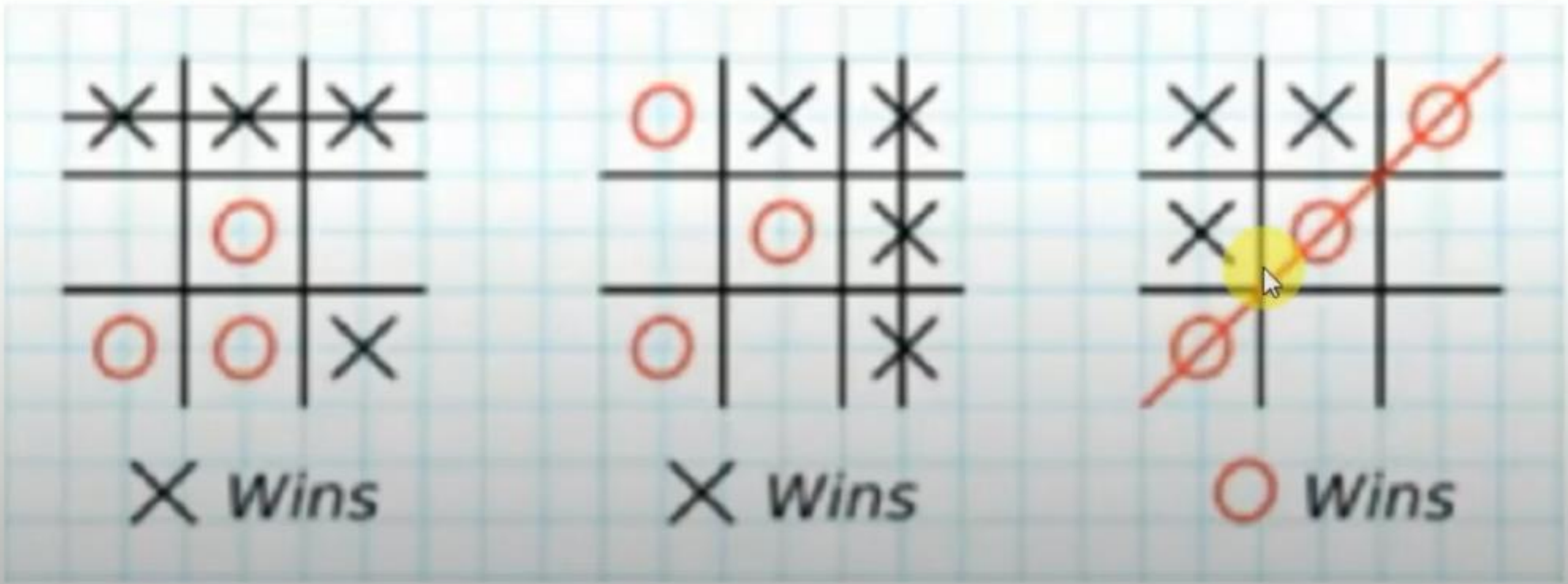
Tic Tac Toe

From table driven to something intelligent

Tic-Tac-Toe Game Playing

- It's a paper and pencil game of two players **X** and **O**, who chooses to mark a spaces on a grid of 3x3.
- The game is won by the player who succeeds in putting three of their marks in a horizontal, vertical or diagonal line



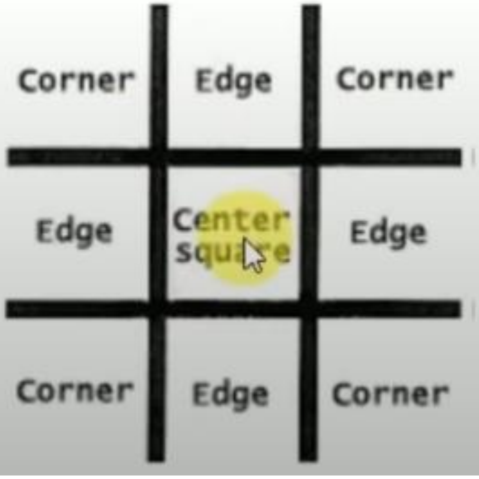
Tic-Tac-Toe Game Playing

- Two players - human, computer.
- The objective is to write a computer program in such a way that computer wins most of the time.
- Three approaches are presented to play this game which increase in
  - Complexity
  - Use of generalization
  - Clarity of their knowledge
  - Extensibility of their approach
  - These approaches will move towards being representations of what we will call AI techniques

Tic-Tac-Toe Game Playing – Program 1

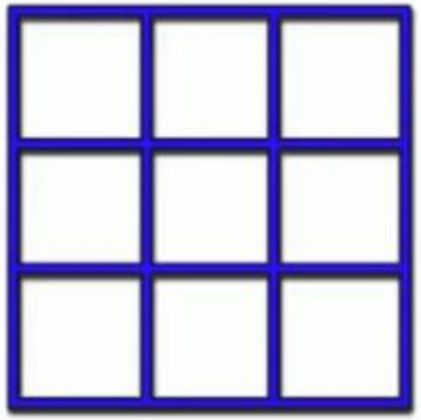
- Assume ,
  - Player 1 - X
  - Player 2 - O
- So, a player who gets 3 consecutive marks first, they will win the game.
- Let's have a discussion about how a board's data structure looks and how the Tic Tac Toe algorithm works.
- **Board Data Structure**

The cells could be represented as Center square, Corner, Edge as like below

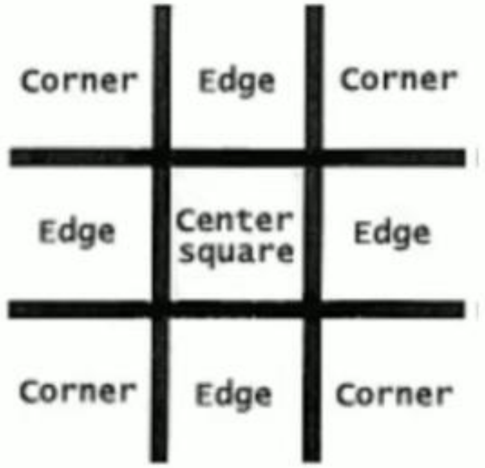


Tic-Tac-Toe Game Playing – Program 1

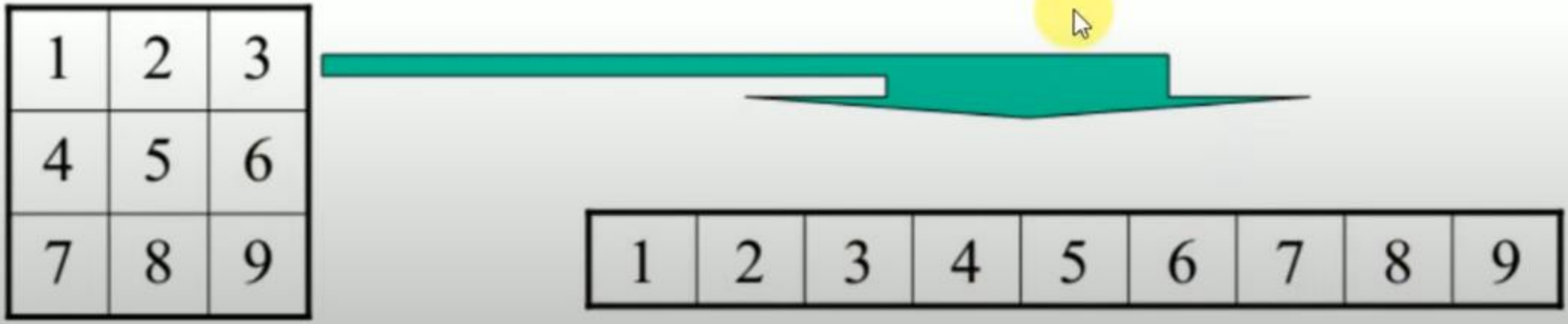
- **Board Data Structure**



The cells could be represented as Center square, Corner, Edge as like below



- **Each square is numbered, which starts from 1 to 9 like following image**



2-D game-board

1-D Vector

Tic-Tac-Toe Game Playing – Program 1

- **Data Structures: Board and Move Table**
- **Consider a Board having nine elements vector. Each element will contain**
  - 0 for blank
  - 1 indicating X player move
  - 2 indicating O player move
- Computer may play as X or O player.
- First player is always X.
- **Move Table:**
- It is a vector of 3^9 elements, each element of which is a nine-element vector representing board position.
- Total of 3^9(19683) elements in move table

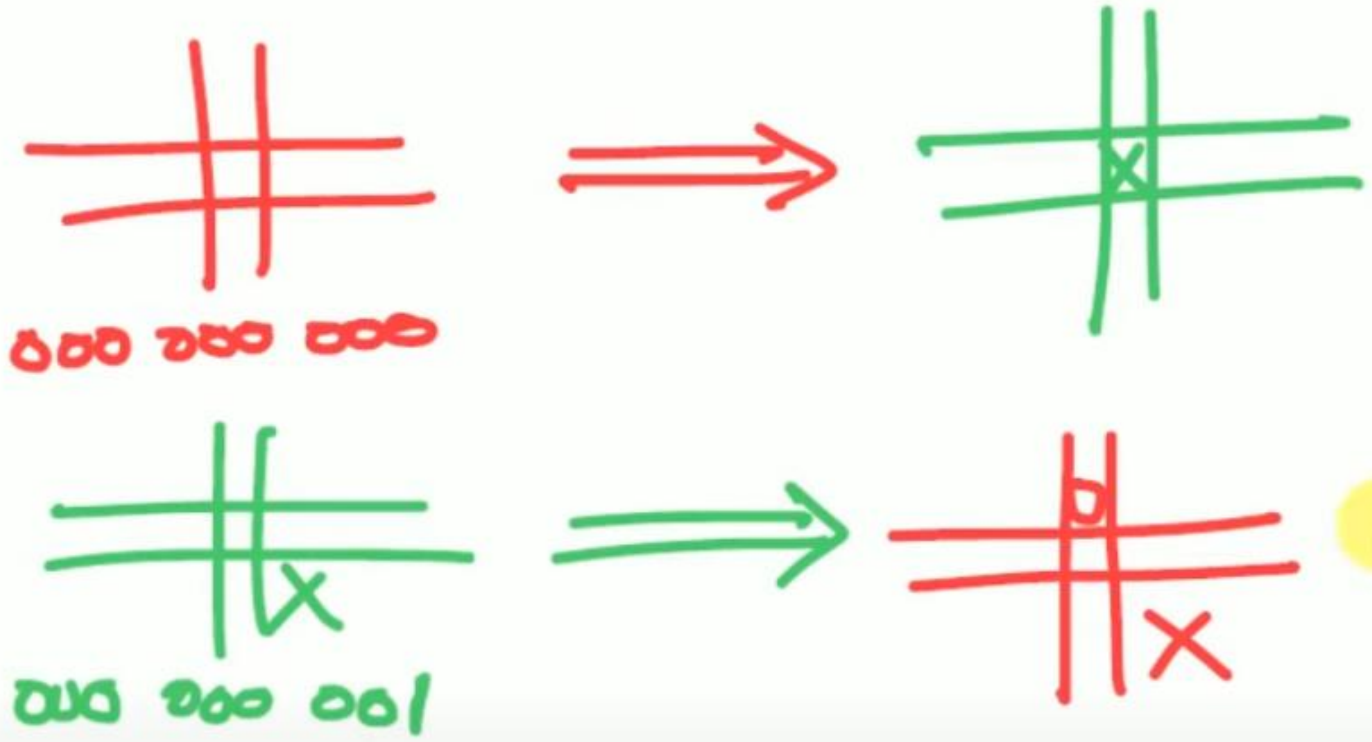
Tic-Tac-Toe Game Playing – Program 1

- **Data Structures: Board and Move Table**
- **Consider a Board having nine elements vector. Each element will contain**
  - 0 for blank
  - 1 indicating X player move
  - 2 indicating O player move
- Computer may play as X or O player.
- First player is always X.
- **Move Table:**
- It is a vector of 3^9 elements, each element of which is a nine-element vector representing board position.
- Total of 3^9(19683) elements in move table

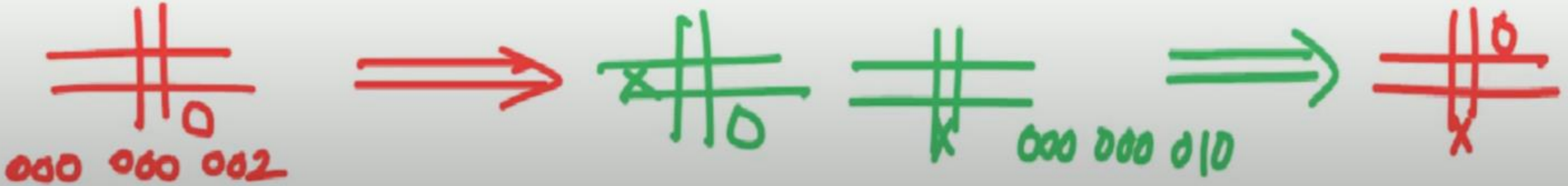
Index	Current Board position	New Board position
0	000000000	000010000
1	000000001	020000001
2	000000002	000100002
3	000000010	002000010
...		

1	2	3	4	5	6	7	8	9

Tic-Tac-Toe Game Playing – Program 1



Index	Current Board position	New Board position
0	000000000	000010000
1	000000001	020000001
2	000000002	000100002
3	000000010	002000010
...		

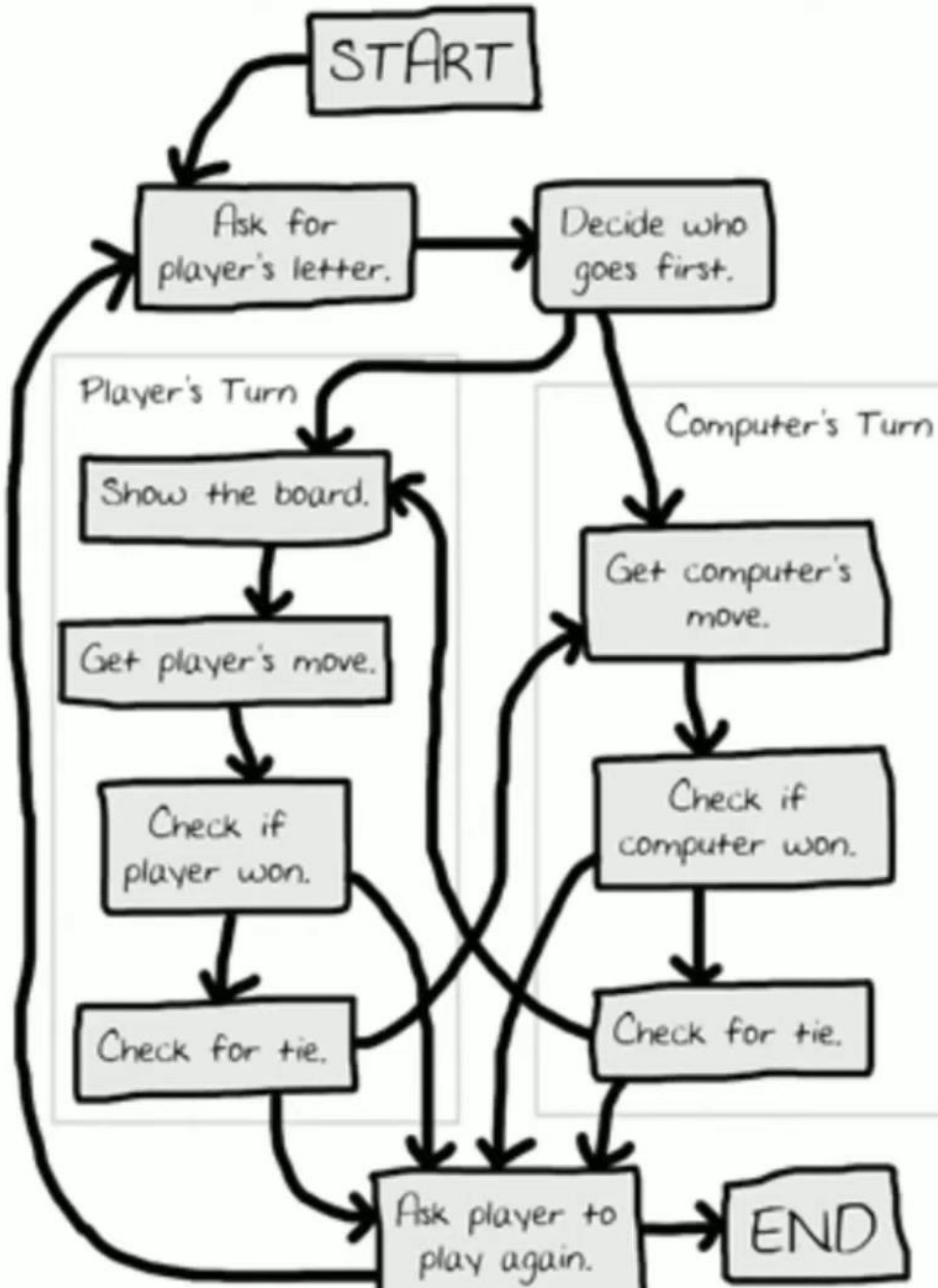


Tic-Tac-Toe

Game Playing

Program 1

Flowchart



Tic-Tac-Toe Game Playing – Program 1

Advantages:

- This program is very efficient in time.

Disadvantages:

- A lot of space to store the Move-Table.
- A lot of work to specify all the entries in the Move-Table.
- Difficult to extend
- Highly error prone as the data is voluminous.
- Poor extensibility
- Not intelligent at all.

Program 2

- Board : A nine-element vector representing the board. We store 2 (indicating Blank), 3 (indicating X), or 5 (indicating O). Turn : An integer indicating which move of the game about to be played. The 1st move will be indicated by 1, last by 9.
- The Algorithm
- The main algorithm uses three functions.
- Make2: returns 5 if the center square of the board is blank i.e. if `board[5]=2`. Otherwise, this function returns any non-corner square `(2, 4, 6 or 8)`.
- `Posswin(p)`: Returns 0 if player `p` can't win on his next move; otherwise, it returns the number of the square that constitutes a winning move. This function will enable the program both to win and to block opponents win. This function operates by checking each of the rows, columns, and diagonals. By multi plying the values of each square together for an entire row (or column or diagonal), the possibility of a win can be checked. If the product is `18` (`3 x 3 x 2`), then `X` can win. If the product is `50` (`5 x 5 x 2`), then O can win. If a winning row (column or diagonal) is found, the blank square in it can be determined and the number of that square is returned by this function.
- `Go (n)`: makes a move in square n. this procedure sets board `[n]` to 3 if Turn is odd, or 5 if Turn is even. It also increments turn by one.
- The algorithm has a built-in strategy for each move. It makes the odd numbered move if it plays `X`, the even-numbered move if it plays O.

- Turn = 1   Go(1)   (upper left corner).
- Turn = 2   If Board[5] is blank, Go(5), else Go(1).
- Turn = 3   If Board[9] is blank, Go(9), else Go(3).
- Turn = 4   If Posswin(X) is not 0, then Go(Posswin(X)) i.e. [block opponent's win], else Go(Make2).
- Turn = 5   if Posswin(X) is not 0 then Go(Posswin(X)) [i.e. win], else if Posswin(O) is not 0, then Go(Posswin(O)) [i.e. block win], else if Board[7] is blank, then Go(7), else Go(3). [to explore other possibility if there be any].
- Turn = 6   If Posswin(O) is not 0 then Go(Posswin(O)), else if Posswin(X) is not 0, then Go(Posswin(X)), else Go(Make2).
- Turn = 7   If Posswin(X) is not 0 then Go(Posswin(X)), else if Posswin(X) is not 0, then Go(Posswin(O)) else go anywhere that is blank.
- Turn = 8   if Posswin(O) is not 0 then Go(Posswin(O)), else if Posswin(X) is not 0, then Go(Posswin(X)), else go anywhere that is blank.
- Turn = 9   Same as Turn=7.

Tic-Tac-Toe Game Playing – Program 2 Magic Square

- Here, we assign board positions to vector elements.
- Sum of all rows, columns and diagonals must me 15.

8	3	4
1	5	9
6	7	2

Algorithm:

- First machine will check, chance to win,
  - difference between 15 and the sum of the two squares
  - If this difference is **not positive** or **if it is greater than 9**, then the original two squares were not collinear and so can be ignored.
- or it will check the opponent of winning and block the chances of winning.

Tic-Tac-Toe Game Playing – Program 2 Magic Square

- Turn – Computer (C)

	C	

- Turn – Human (H)

H ✓		
	C	

Turn – Computer (C)

H		C
	C	

Turn – Human (H)

H		C
	C	
H		

Magic Square

8	3	4
1	5	9
6	7	2

Tic-Tac-Toe Game Playing – Program 2 Magic Square

- **Previous State**

H		C
C	C	
H		

Diff = 15 - (5+4) = 6

6 is not empty, hence Computer can't win the game.

Magic Square

8	3	4
1	5	9
6	7	2

- **Turn – Computer (C)**

H		C
C	C	
H		

Diff = 15 – (8+6) = 1

1 is empty, hence Human can win the game.

Hence Computer Blocks it.

Computer - go to 1

- **Turn – Human (H)**

H		C
C	C	
H	H	

- **Previous State**

H		C
C	C	
H		

- **Turn – Human (H)**

H		C
C	C	
H	H	

- **Turn – Computer (C)**

H		C
C	C	C
H	H	

Magic Square

8	3	4
1	5	9
6	7	2

Diff = 15 - (5+4) = 6

6 is not empty, hence Computer can't win the game.

Diff = 15 – (1+4) = 10

10 is greater than 9, hence Computer can't win the game.

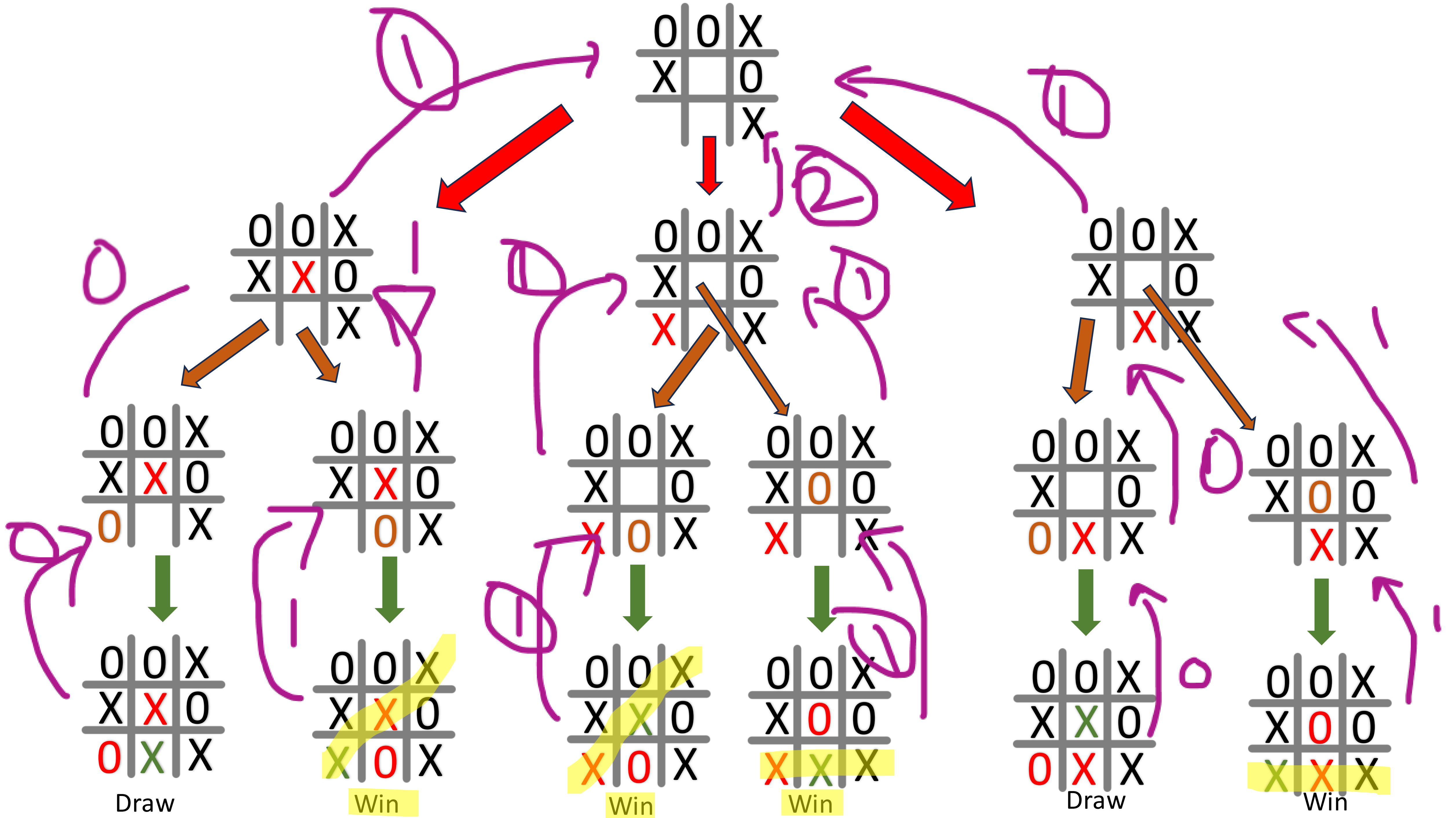
Diff = 15 – (1+5) = 9

9 is empty, hence Computer can win the game. Computer - go to 9

Tic-Tac-Toe Game Playing – Program 3

Data Structures

- Board Position is a structure containing
  - A 9-element array representing the board
  - A list of board positions that could result from the next move
  - A number or rating representing an estimate of how likely the board position is to lead to an ultimate win for the player to move.



Draw
Win
Win
Win
Draw
Win
1
2
0
1
1
0
0
1
1
0
1
0
1

Tic-Tac-Toe Game Playing – Program 3

Comments

- This program will require more time than two others as
  - it has to search a tree representing all possible move sequences before making each move.
  - This approach is extensible to handle
    - 3-dimensional tic-tac-toe and
    - it could be extended to handle games which are more complicated than tic-tac-toe