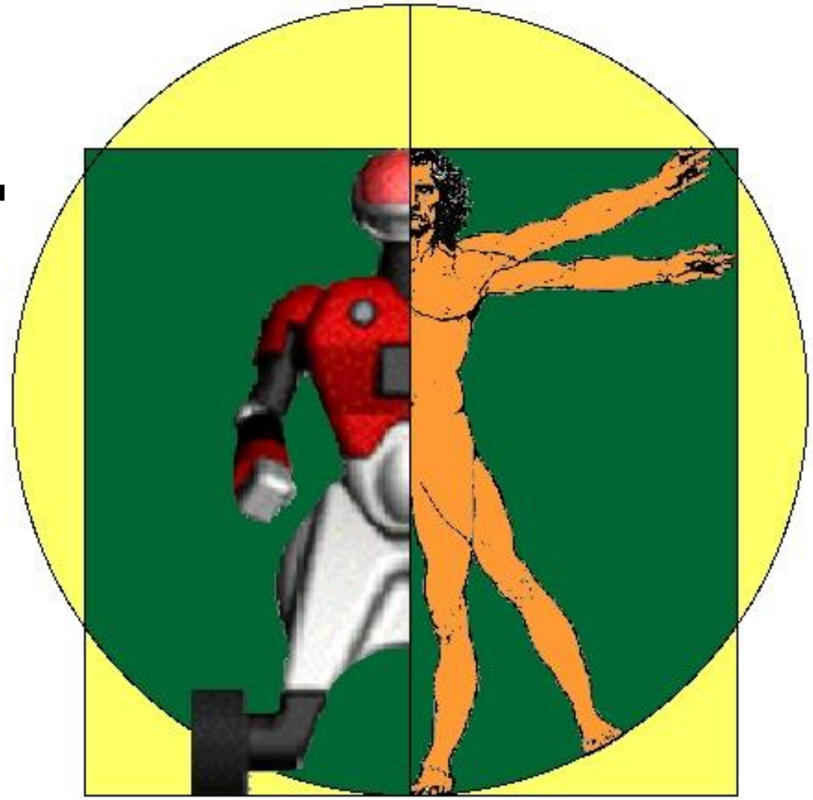


# INTELLIGENT SYSTEMS
# (Artificial Intelligence)

# Artificial Intelligence Job Profiles

- 1. Machine Learning Engineer
- 2. Data Scientist
- 3. Artificial Intelligence Engineer
- 4. Business Intelligence Developer
- 5. Research Scientist
- 6. Big Data Engineer/Architect

# Course Objectives

- At the end of this course, students will have a basic understanding of the modern concepts of Artificial Intelligence.
- Students will be able to write reasonably substantial programs that address typical Artificial Intelligence problems.

# Course Organization

- Unit I: Foundational Issues in Intelligent Systems
- Unit II: Knowledge Representation
- Unit III: Reasoning
- Unit IV: Planning
- Unit V: Applications of Intelligent Systems

# Course Organization

- **Books**

  **a) Artificial Intelligence**

  **Author Name: Elaine Rich and Kevin Knight**

  **b) Introduction to Artificial Intelligence & Expert Systems**

  **Author Name: Dan W. Patterson**

  **c) Artificial Intelligence – A Modern Approach**

  **Author Name: Stuart Russell and Peter Norvig**

- **Lab Practical:**

  **AI Language to be learned: PROLOG (Programming Logic)**

  **Books:**

  **a) Introduction to Turbo Prolog**

  **Author Name: Carl Townsend**

# UNIT I

# FOUNDATIONAL ISSUES IN INTELLIGENT SYSTEMS

# Intelligence

- **Intelligence:**

  The capability of learning, understanding, reasoning and learning from mistakes.

- **Intelligent:**

  having or showing intelligence

Examples: speech recognition, face detection, smell identification, decision making, learning new things.

# Types of Intelligence

## Types of Intelligence:

- **Natural Intelligence:**

  Capacity of the mind to understand principles, truths, facts or meanings, acquire knowledge, and apply it to practice using the ability to learn and comprehend.

- **Artificial Intelligence or Machine Intelligence:**

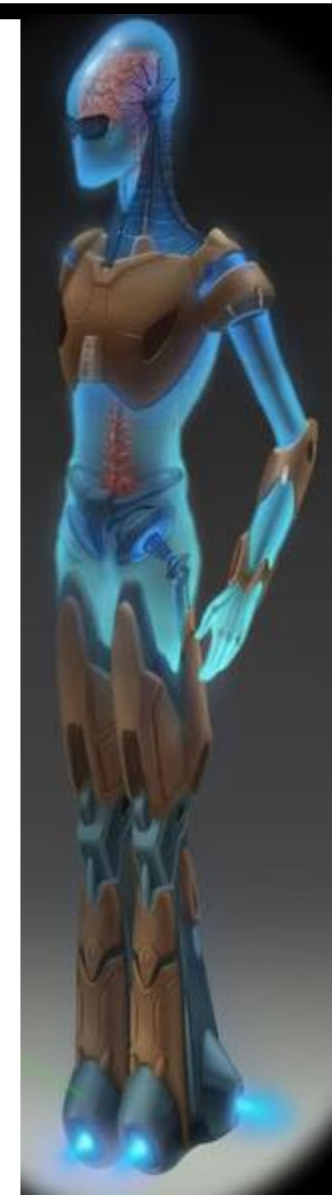
  Study of how to make computers do things which at the moment people do better.

# Intelligent Systems

- **System:** a group of things or parts working together as a whole in an environment.
- **An Intelligent System is a system**
  - ❑ which learns how to act so it reaches its goals/objectives.
  - ❑ which learns from experience and makes decisions about appropriateness of actions.
  - ❑ which continually acts, mentally and physically, and by acting reaches its objectives more often than pure chance would indicate.

# Objectives of AI & IS

- **To understand and model the workings of natural intelligence and human beings**
- **Implement them to create intelligent systems with machine intelligence**
- **Examples:**
  - ❑ **Robots – completely artificial**
  - ❑ **Cyborgs – organisms that combine natural and artificial intelligence as well as physical components, e.g. a cyborg may have half a human brain combined with an artificial brain**



# What is Artificial Intelligence ?

- making computer programs that appear to think?
- the automation of activities we associate with human thinking, like decision making, learning ... ?
- the art of creating machines that perform functions that require intelligence when performed by people ?
- the study of mental faculties through the use of computational models ?

# What is Artificial Intelligence ?

- the study of computations that make it possible to perceive, reason and act ?
- a branch of computer science that is concerned with the automation of intelligent behaviour?
- anything in Computing Science that we don't yet know how to do ? (!)

# What is Artificial Intelligence ?

| | HUMAN | RATIONAL (able to think and make decisions based on reasons) |
|---|---|---|
| THOUGHT | Systems that think like humans | Systems that think rationally |
| BEHAVIOR | Systems that act like humans | Systems that act rationally |

# Thinking Humanly: Cognitive Modelling

- **Cognitive Science**: Interdisciplinary field (AI, psychology, linguistics, philosophy, anthropology) that tries to form computational theories of human cognition.
- **Cognitive Science approach**
  - Try to get “inside” our minds
- **Cognitive science brings together theories and experimental evidence to model internal activities of the brain.**

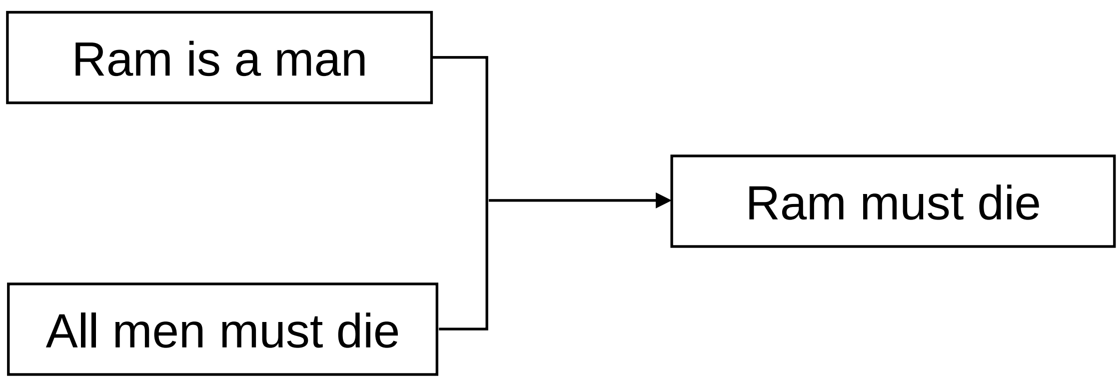
# Thinking Rationally: Laws of Thought

- **Rational - defined in terms of logic**
- **Represent facts about the world via logic**
- **Logic can’t express everything (e.g. uncertainty)**

**Right thinking:- unquestionable reasoning processes**

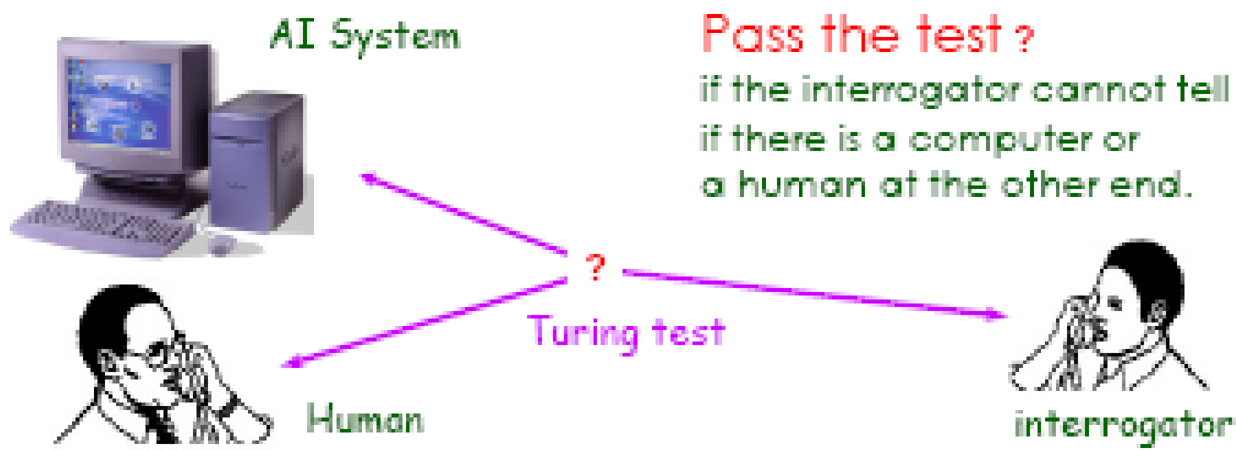
**Syllogism:- form of reasoning in which a conclusion is drawn from two statements.**

**Example:**



# Acting humanly: Turing Test

- Designed by Alan Turing (1950)
- It’s a behavioral test (to check if a system acts like a human)
- **The Turing Test**: a computer is programmed well enough to have a conversation with an interrogator (for example through a computer terminal) and passes the test if the interrogator cannot distinguish, if there is a computer or a human at the other end.



# Turing Test

**For passing Turing Test, the computer would need**

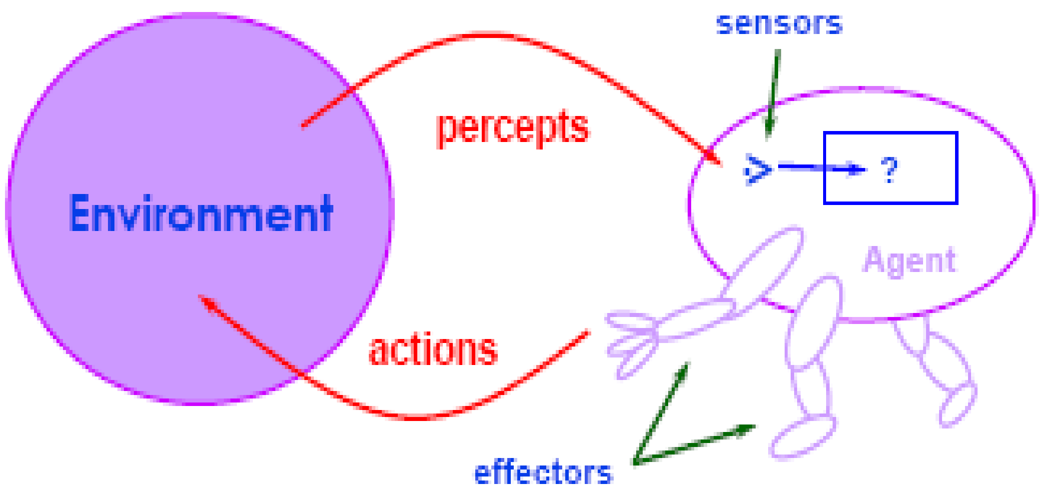
- ❑ **Natural Language**
- ❑ **Knowledge Representation**
- ❑ **Complex Reasoning**
- ❑ **Machine Learning**

**Total Turing Test: include a video signal so that interrogator can test the subject’s perceptual abilities. This would require**

- ❑ **Computer Vision**
- ❑ **Robotics (actuation)**

# Acting Rationally: The rational agent approach

- **Agent:** An entity that perceives (observes) its environment and is able to execute actions to change it.
- Agents have inherent goals that they want to achieve (e.g. survive, reproduce).
- A rational agent acts in a way to maximize the achievement of its goals.




Agents interact with environments through sensors and effectors

# Rational Agents

- **Human Agent:**
  - **Sensors**: 5 senses, i.e., eyes, ears, nose, tongue, skin
  - **Effectors**: hands, legs and other body parts
- **Robotic Agent:**
  - **Sensors**: camera or infrared range finders, microphones, touch screens.
  - **Effectors**: various motors.

# Examples of Intelligent Systems

- **Medical advice system**
  - **Symptoms, findings, patient's answers**
  - **Questions, tests, therapy recommendations**
- **credit card fraud detection**
- **information retrieval, Google**
- **aircraft, pipeline inspection**
- **speech understanding, generation, translation**
- **pattern recognition**
- **Knowledge representation**
- **Many more**

# A Cyborg


MIND CONTROL
OVER MACHINE
EMP
SOURCE
PROPULSION
SYSTEM
INTERFACE
PROBES

# History of Artificial Intelligent Systems

| | |
|---|---|
| 1943 | McCulloch & Pitts: Boolean circuit model of brain |
| 1950 | Turing's "Computing Machinery and Intelligence" |
| 1952–69 | Look, Ma, no hands! |
| 1950s | Early AI programs, including Samuel's checkers program, Newell & Simon's Logic Theorist, Gelernter's Geometry Engine |
| 1956 | Dartmouth meeting: "Artificial Intelligence" adopted |
| 1965 | Robinson's complete algorithm for logical reasoning |
| 1966–74 | AI discovers computational complexity<br>Neural network research almost disappears |
| 1969–79 | Early development of knowledge-based systems |
| 1980–88 | Expert systems industry booms |
| 1988–93 | Expert systems industry busts: "AI Winter" |
| 1985–95 | Neural networks return to popularity |
| 1988– | Resurgence of probabilistic and decision-theoretic methods<br>Rapid increase in technical depth of mainstream AI<br>"Nouvelle AI": ALife, GAs, soft computing |

# Task Domains of Artificial Intelligence

- **Mundane (ordinary) Tasks:**
  - ❑ **Perception**
    - **Vision**
    - **Speech**
  - ❑ **Natural Languages**
    - **Understanding**
    - **Generation**
    - **Translation**
  - ❑ **Common Sense Reasoning**
  - ❑ **Robot Control**

# Task Domains of Artificial Intelligence

- **Formal Tasks**
  - **Games: Chess, Checkers**
  - **Mathematics: Geometry, Logic, Proving properties of programs**
- **Expert Tasks:**
  - **Engineering: Design, Fault finding, Manufacturing, Planning**
  - **Scientific Analysis**
  - **Medical Diagnosis**
  - **Financial Analysis**

# Foundations of AI

- **What are our underlying assumptions about intelligence?**
- **What kinds of techniques will be useful for solving AI problems?**
- **At what level of detail, if at all, are we trying to model human intelligence?**
- **How will we know when we have succeeded in building an intelligent program?**

# 1.Underlying Assumptions

**Physical symbol system hypothesis**

**A physical symbol system consists of**

- **Symbols - physical patterns, e.g., alphabets**
- **Symbol structure- or expression instances of symbols, called tokens related together,e.g.,words**
- **Processes - act on expressions to produce more expressions, e.g., creation, modification, deletion.**

**Example of physical symbol system is the human mind.**

# 2. AI Techniques

- **Intelligence requires knowledge**
- **Characteristics of knowledge which are undesirable:**
  - ❑ **Voluminous**
  - ❑ **Hard to characterize accurately**
  - ❑ **Constantly changing**
  - ❑ **Differs from data by being organized in a way that corresponds to the situations in which it will be used.**
- **The knowledge should be represented in the AI Technique so that the technique can exploit it.**

# 2. AI Techniques

**Definition: An AI Technique is a method that exploits knowledge that should be represented in such a way that:**

- **The knowledge captures generalization**
  - **A class in C++ is a generalization of objects**
- **The representation can be understood by people who provide the knowledge**
  - **A variable name representing a value should be named in such a manner that all other programmers can understand what it means**
- **The knowledge should be easily modifiable or updatable to reflect changes in the world**
  - **The variable should be easily modifiable by assigning it a new value**

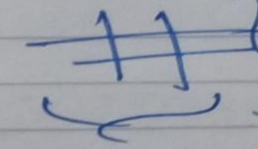
# 2. AI Techniques

- **It can be used in a great many situations even if it is not totally accurate or complete**
  - **Even if a machine does not have enough data to take a decision, it should be able to take a decision and continue its work**
- **It can be used to overcome its own sheer bulk by helping to narrow down the range of possibilities in a situation**
  - **To identify suitable candidates for an interview, the knowledge of their qualifications helps to shortlist the candidates**
  - **Searching for information using Google – a general string vs a specific string**

## Application of AI Techniques to Tic-Tac-Toe: Program 1

* TIC-TAC-TOE PROBLEM:- (Zero-katta)

Example:



Initial State

9 square

Two-Player Game where one player called MAX (X-letter marks) and the opponent called MIN (O-letter marks)

AI term

↳ '3x3' Grid where two players put their letters.

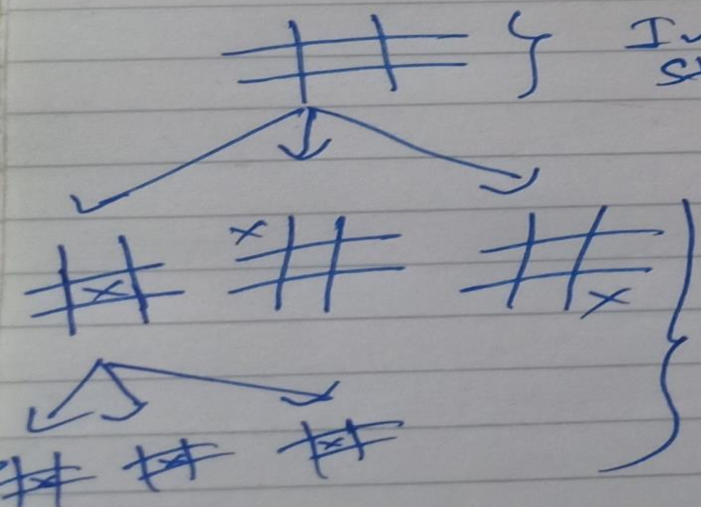
↳ Players with same marks (three) in complete Row/Column/Diagonal wins the game → Winning condition

→ Start State = Empty Grid

→ Goal state = win for either of player

MAX (3 consecutive X)

MIN (3 consecutive O)



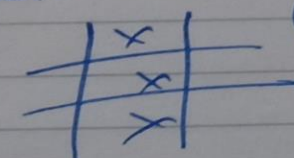
Initial State (9 possible combination but we are taking 3 for demonstration purpose)

9

↓

8 possible outcome for zero

Intermediate state



Goal State
Winning State for MAX

# Program 1 for Tic-Tac-Toe:Data Structures

- **Data structures:**
  - **Board :**

    **A 9-element vector**

    **Each element can contain the value**

    **0 if corresponding square is blank**

    **1 if filled with a X**

    **2 if filled with a O**

  - **Movetable :**

    **Large vector of 19,683 ($3^9$) elements**

    **Each element of which is a 9-element vector whose contents are chosen in such a way so as to allow the algorithm to work.**

| | | |
|---|---|---|
| **1** | **2** | **3** |
| **4** | **5** | **6** |
| **7** | **8** | **9** |

Board

| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
|---|---|---|---|---|---|---|---|---|---|
| **1** | | | | | | | | | |
| **2** | | | | | | | | | |
| **…** | | | | | | | | | |
| **19683** | | | | | | | | | |

Movetable

# Program 1 for Tic-Tac-Toe:Algorithm

**To make a move, do the following:**

**Step 1: Take value of the vector Board as a base 3 number. Convert it to decimal.**

**Step 2: Use the number computed in Step 1 as an index into Movetable and access the vector(X) stored there.**

**Step3: Set Board equal to vector X in Movetable**

# Application of AI Techniques to Tic-Tac-Toe:

## Program 2

# Program 2 for Tic-Tac-Toe: Data Structures

- **Data structures:**
  - **Board :**

    **A 9-element vector**

    **Each element can contain the value**

    **2 if corresponding square is blank (B)**

    **3 if filled with a X**

    **5 if filled with a O**
  - **Turn :**

    **Integer indicating next move**

# Program 2 for Tic-Tac-Toe:Subprocedures

- **Subprocedures used are:**
  - **Make2** : **if Board[5]=2**

    **returns 5**

    **otherwise**

    **returns any blank non-corner square (2,4,6,8)**
  - **Posswin(p)** : ***Poss*ibility of player p *win*ning**

    **if player p can not win on his next move**

    **returns 0**

    **else**

    **returns the number of the square which will be the winning move**

    **(How is the move found?->next slide)**

# Program 2 for Tic-Tac-Toe:Subprocedures

## Posswin(p) : How to find the next move?

**Multiply the 3 values (on the data structure Board) of each of a row or a column or a diagonal and check for specific values:**

**e.g.**

**1. If product = 18 = 3 * 3 * 2 then X can win**

**X X B**

**2. If product = 50= 5 * 5 * 2 then O can win**

**O O B**

**If a winning row is found, the blank element is determined and the number of that square is returned.**

# Program 2 for Tic-Tac-Toe:Subprocedures

- **Go(n) :**
  - Go to square n
  - If Turn is odd, makes Board[n] = 3 (X)
    If Turn is even, makes Board[n] = 5 (O)
  - Increments Turn by 1

# Program 2 for Tic-Tac-Toe:Algorithm

- **Initially, all the values on Board are 2 (blank)**
- **Turn=1(X)**

  **Go(1) (upper left corner).**
- **Turn=2(O)**

  **If Board[5] is blank, Go(5), else Go(1).**
- **Turn=3(X)**

  **If Board[9] is blank, Go(9), else Go(3).**

| | | |
|---|---|---|
| $X_1$ | | |
| | $O_2$ | |
| | | $X_3$ |

# Program 2 for Tic-Tac-Toe:Algorithm

- **Turn=4(O)**

  **If Posswin(X) is not 0, then Go(Posswin(X)) [i.e., block opponent's win], else Go(Make2).**

**Here Posswin(X)=0 then Go(Make2). Suppose Make2 returns 2, so Go(2).**

- **Turn=5(X)**

  **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, Go(7)**

  **else Go(3).**

**Here Posswin(O) <> 0, Go(8).**

| | | |
|---|---|---|
| $X_1$ | $O_4$ | |
| | $O_2$ | |
| | $X_5$ | $X_3$ |

# Program 2 for Tic-Tac-Toe:Algorithm

- **Turn=6(O)**

  **If Posswin(O) is not 0 then Go(Posswin(O)),**

  **else if Posswin(X) is not 0 then Go(Posswin(X))**

  **else Go(Make2)**

**Here Posswin(X)<>0, Go(7).**

- **Turn=7(X)**

  **If Posswin(X) is not 0 then Go(Posswin(X)),**

  **else if Posswin(O) is not 0 then Go(Posswin(O))**

  **else go anywhere that is blank**

**Here Posswin(O)<>0, Go(3).**

| | | |
|---|---|---|
| $X_1$ | $O_4$ | $X_7$ |
| | $O_2$ | |
| $O_6$ | $X_5$ | $X_3$ |

# Program 2 for Tic-Tac-Toe:Algorithm

- **Turn=8(O)**

  **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**

**Here Posswin(X)<>0, Go(6).**

- **Turn=9(X) (same as Turn 7)**

  **If Posswin(X) is not 0 then Go(Posswin(X)),**

  **else if Posswin(O) is not 0 then Go(Posswin(O))**

  **else go anywhere that is blank**

**Here Posswin(O)<>0, Go(4).**

| | | |
|---|---|---|
| $X_1$ | $O_4$ | $X_7$ |
| $X_9$ | $O_2$ | $O_8$ |
| $O_6$ | $X_5$ | $X_3$ |

GAME OVER

# Application of AI Techniques to Tic-Tac-Toe:

## Program 3

# Program 3 for Tic-Tac-Toe:Data Structures

- **BoardPosition 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.**

# Program 3 for Tic-Tac-Toe:Algorithm

**Steps:**

1. For the next move, look ahead at the board positions that result from each possible move.
2. Decide which position is best
3. Make the move that leads to that position
4. Assign the rating of that best move to the current position

In Step 2, to decide which of a set of board positions is best, do the following for each of them:

2(a) See if it is a win. If so, call it the best by giving it the highest rating.

(continued…)

# Program 3 for Tic-Tac-Toe:Algorithm

**2(b)** Otherwise, consider all the moves the opponent would make next.

See which is worst by recursively calling the first step. Assume the opponent will make that move.

Whatever rating that move has, assign it to the move being considered.

**2(c)** The best move is then the one with the highest rating.

**This algorithm will look ahead at various sequences of moves in order to find a sequence that leads to a win.**

**It attempts to maximize the likelihood of winning, while assuming that the opponent will try to minimize that likelihood.**

**This algorithm is called the MINIMAX Procedure.**

# Comments on the Three Programs

- **Program 1:** Fast but takes up lot of memory

  If it is to be extended to 3D Tic-Tac-Toe, it would require $3^{27}$ entries in MoveTable (not possible)

- **Program 2:** Slower and less efficient

  But requires much less space

  Strategy is simpler but is predetermined

  Cannot be generalized to any sequence of moves.

- **Program 3:** Slower

  Can be extended to games more complex than Tic-Tac-Toe