ARTIFICIAL INTELLIGENCE

19CS507

UNIT II- Knowledge Representation

What is knowledge representation?

Humans are best at understanding, reasoning, and interpreting knowledge. Human knows things, which is knowledge and as per their knowledge they perform various actions in the real world. But how machines do all these things comes under knowledge representation and reasoning.

Hence we can describe Knowledge representation as following:

- Knowledge representation and reasoning (KR, KRR) is the part of Artificial intelligence which concerned with AI agents thinking and how thinking contributes to intelligent behavior of agents.
- It is responsible for representing information about the real world so that a computer can understand and can utilize this knowledge to solve the complex real world problems such as diagnosis a medical condition or communicating with humans in natural language.
- It is also a way which describes how we can represent knowledge in artificial intelligence. Knowledge representation is not just storing data into some database, but it also enables an intelligent machine to learn from that knowledge and experiences so that it can behave intelligently like a human.

Procedural Vs Declarative Knowledge

What is Procedural Knowledge?

- Procedural or imperative knowledge clarifies how to perform a certain task.
- It lays down the steps to perform.
- Thus, the procedural knowledge provides the essential control information required to implement the knowledge.

What is Declarative Knowledge?

- Declarative or functional knowledge clarifies what to do to perform a certain task.
- It lays down the function to perform.
- Thus, in the declarative knowledge, only the knowledge is provided but not the control information to implement the knowledge.
- Thus, in order to use the declarative knowledge, we have to add the declarative knowledge with a program which provides the control information.

Difference the Procedural and Declarative Knowledge

S.NO	Procedural Knowledge	Declarative Knowledge
1.	It is also known as Interpretive knowledge.	It is also known as Descriptive knowledge.
2.	Procedural Knowledge means how a particular thing can be accomplished.	While Declarative Knowledge means basic knowledge about something.
3.	Procedural Knowledge is generally not used means it is not more popular.	Declarative Knowledge is more popular.
4.	Procedural Knowledge can't be easily communicate.	Declarative Knowledge can be easily communicate.
5.	Procedural Knowledge is generally process oriented in nature.	Declarative Knowledge is data oriented in nature.
6.	In Procedural Knowledge debugging and validation is not easy.	In Declarative Knowledge debugging and validation is easy.
7.	Procedural Knowledge is less effective in competitive programming.	Declarative Knowledge is more effective in competitive programming.

Representations & Approaches to Knowledge Representation

What is knowledge representation?

- Humans are best at understanding, reasoning, and interpreting knowledge.
- Human knows things, which is knowledge and as per their knowledge they perform various actions in the real world.
- But how machines do all these things comes under knowledge representation and reasoning.

Hence we can describe Knowledge representation as following:

- Knowledge representation and reasoning (KR, KRR) is the part of Artificial intelligence which concerned with AI agents thinking and how thinking contributes to intelligent behavior of agents.
- It is responsible for representing information about the real world so that a computer can understand and can utilize this knowledge to solve the complex real world problems such as diagnosis a medical condition or communicating with humans in natural language.
- It is also a way which describes how we can represent knowledge in artificial intelligence. Knowledge representation is not just storing data into some database, but it also enables an intelligent machine to learn from that knowledge and experiences so that it can behave intelligently like a human.

What to Represent:

Following are the kind of knowledge which needs to be represented in AI systems:

- **Object**: All the facts about objects in our world domain. E.g., Guitars contains strings, trumpets are brass instruments.
- **Events:** Events are the actions which occur in our world.
- **Performance:** It describe behavior which involves knowledge about how to do things.

- **Meta-knowledge:** It is knowledge about what we know.
- **Facts:** Facts are the truths about the real world and what we represent.
- **Knowledge-Base:** The central component of the knowledge-based agents is the knowledge base. It is represented as KB. The Knowledgebase is a group of the Sentences (Here, sentences are used as a technical term and not identical with the English language).

Knowledge:

Knowledge is awareness or familiarity gained by experiences of facts, data, and situations.

Following are the types of knowledge in artificial intelligence:

Types of knowledge

1. Declarative Knowledge:

- Declarative knowledge is to know about something.
- It includes concepts, facts, and objects.

- It is also called descriptive knowledge and expressed in declarativesentences.
- It is simpler than procedural language.

2. Procedural Knowledge

- It is also known as imperative knowledge.
- Procedural knowledge is a type of knowledge which is responsible for knowing how to do something.
- It can be directly applied to any task.
- It includes rules, strategies, procedures, agendas, etc.
- Procedural knowledge depends on the task on which it can be applied.

3. Meta-knowledge:

- Knowledge about the other types of knowledge is called Meta-knowledge.

4. Heuristic knowledge:

- Heuristic knowledge is representing knowledge of some experts in a filed or subject.
- Heuristic knowledge is rules of thumb based on previous experiences, awareness of approaches, and which are good to work but not guaranteed.

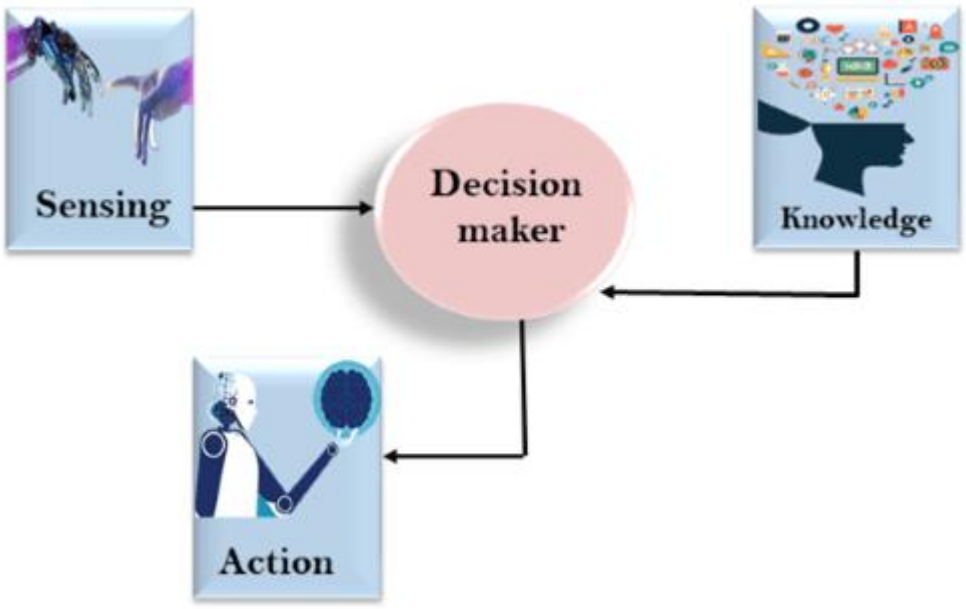
5. Structural knowledge:

- Structural knowledge is basic knowledge to problem-solving.
- It describes relationships between various concepts such as kind of, part of, and grouping of something.
- It describes the relationship that exists between concepts or objects.

The relation between knowledge and intelligence:

- Knowledge of real-worlds plays a vital role in intelligence and same for creating artificial intelligence.
- Knowledge plays an important role in demonstrating intelligent behavior in AI agents.

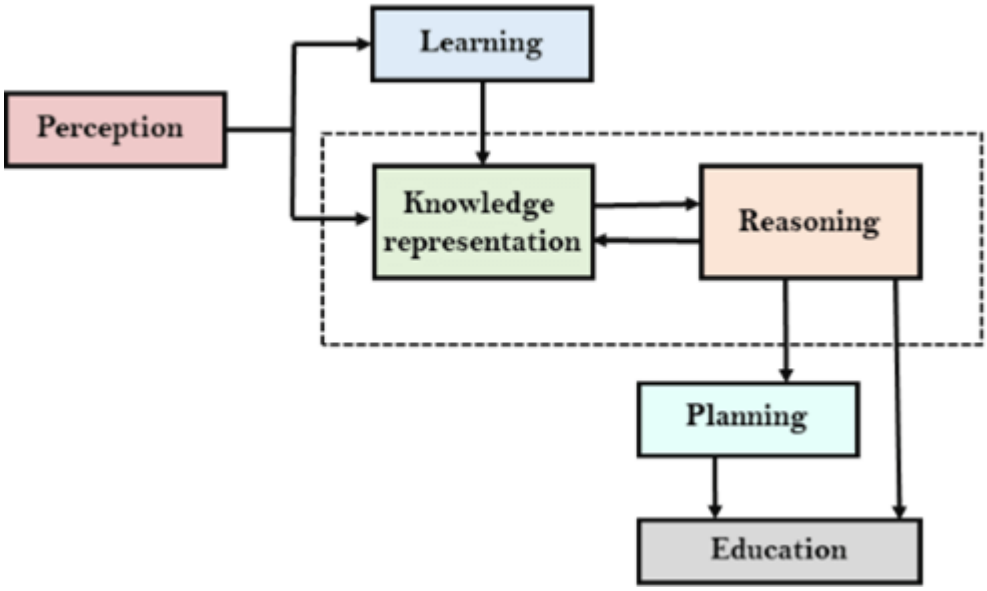
- An agent is only able to accurately act on some input when he has some knowledge or experience about that input.
- Let's suppose if you met some person who is speaking in a language which you don't know, then how you will able to act on that. The same thing applies to the intelligent behavior of the agents.
- As we can see in below diagram, there is one decision maker which act by sensing the environment and using knowledge. But if the knowledge part will not present then, it cannot display intelligent behavior.



AI knowledge cycle:

An Artificial intelligence system has the following components for displaying intelligent behavior:

- Perception
- Learning
- Knowledge Representation and Reasoning
- Planning
- Execution



The above diagram is showing how an AI system can interact with the real world and what components help it to show intelligence. AI system has Perception component by which it retrieves information from its environment. It can be visual, audio or another form of sensory input. The learning component is responsible for learning from data captured by Perception comportment. In the complete cycle, the main components are knowledge representation and Reasoning. These two components are involved in showing the intelligence in machine-like humans. These two components are independent with each other but also coupled together. The planning and execution depend on analysis of Knowledge representation and reasoning.

Approaches to knowledge representation

There are mainly four approaches to knowledge representation, which are given below:

1. Simple relational knowledge:

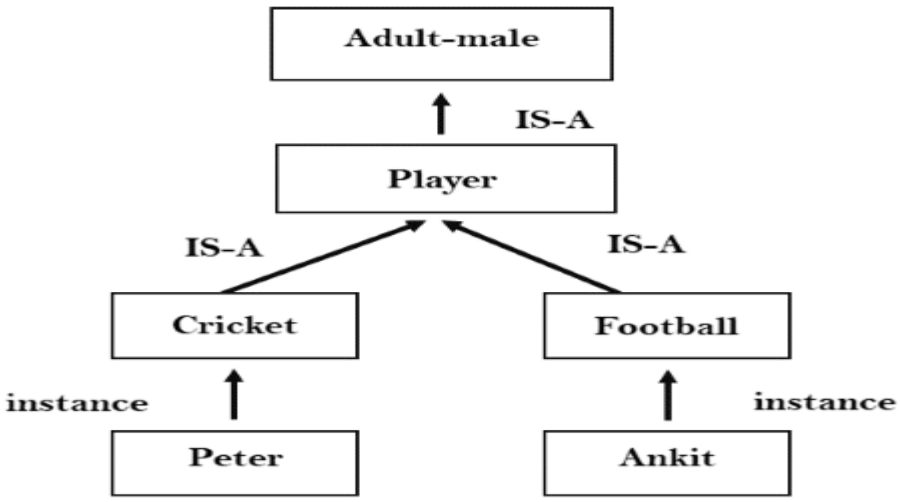
- It is the simplest way of storing facts which uses the relational method, and each fact about a set of the object is set out systematically in columns.
- This approach of knowledge representation is famous in database systems where the relationship between different entities is represented.
- This approach has little opportunity for inference.

Example: The following is the simple relational knowledge representation.

Player	Weight	Age
Player1	65	23
Player2	58	18
Player3	75	24

2. Inheritable knowledge:

- In the inheritable knowledge approach, all data must be stored into a hierarchy of classes.
- All classes should be arranged in a generalized form or a hierarchal manner.
- In this approach, we apply inheritance property.
- Elements inherit values from other members of a class.
- This approach contains inheritable knowledge which shows a relation between instance and class, and it is called instance relation.
- Every individual frame can represent the collection of attributes and its value.
- In this approach, objects and values are represented in Boxed nodes.
- We use Arrows which point from objects to their values.
- **Example:**



3. Inferential knowledge:

- Inferential knowledge approach represents knowledge in the form of formal logics.
- This approach can be used to derive more facts.
- It guaranteed correctness.
- Example: Let's suppose there are two statements:
    a. Marcus is a man
    b. All men are mortal
    Then it can represent as;

    man(Marcus)
    ∀x = man (x) ----------> mortal (x)s

4. Procedural knowledge:

- Procedural knowledge approach uses small programs and codes which describes how to do specific things, and how to proceed.
- In this approach, one important rule is used which is If-Then rule.
- In this knowledge, we can use various coding languages such as LISP language and Prolog language.
- We can easily represent heuristic or domain-specific knowledge using this approach.
- But it is not necessary that we can represent all cases in this approach.

Requirements for knowledge Representation system

A good knowledge representation system must possess the following properties.

1. **Representational Accuracy:**
   KR system should have the ability to represent all kind of required knowledge.
2. **Inferential Adequacy:**
   KR system should have ability to manipulate the representational structures to produce new knowledge corresponding to existing structure.

3. **Inferential Efficiency:**
   The ability to direct the inferential knowledge mechanism into the most productive directions by storing appropriate guides.
4. **Acquisitional efficiency:**

   The ability to acquire the new knowledge easily using automatic methods.

Forward Vs Backward Reasoning

Difference between backward chaining and forward chaining

- Forward chaining as the name suggests, start from the known facts and move forward by applying inference rules to extract more data, and it continues until it reaches to the goal, whereas backward chaining starts from the goal, move backward by using inference rules to determine the facts that satisfy the goal.
- Forward chaining is called a data-driven inference technique, whereas backward chaining is called a goal-driven inference technique.
- Forward chaining is known as the down-up approach, whereas backward chaining is known as a top-down approach.
- Forward chaining uses breadth-first search strategy, whereas backward chaining uses depth-first search strategy.
- Forward and backward chaining both applies Modus ponens inference rule.
- Forward chaining can be used for tasks such as planning, design process monitoring, diagnosis, and classification, whereas backward chaining can be used for classification and diagnosis tasks.
- Forward chaining can be like an exhaustive search, whereas backward chaining tries to avoid the unnecessary path of reasoning.
- In forward-chaining there can be various ASK questions from the knowledge base, whereas in backward chaining there can be fewer ASK questions.
- Forward chaining is slow as it checks for all the rules, whereas backward chaining is fast as it checks few required rules only.

S. No.	Forward Chaining	Backward Chaining
1.	Forward chaining starts from known facts and applies inference rule to extract more data unit it reaches to the goal.	Backward chaining starts from the goal and works backward through inference rules to find the required facts that support the goal.
2.	It is a bottom-up approach	It is a top-down approach
3.	Forward chaining is known as data-driven inference technique as we reach to the goal using the available data.	Backward chaining is known as goal-driven technique as we start from the goal and divide into sub-goal to extract the facts.
4.	Forward chaining reasoning applies a breadth-first search strategy.	Backward chaining reasoning applies a depth-first search strategy.
5.	Forward chaining tests for all the available rules	Backward chaining only tests for few required rules.
6.	Forward chaining is suitable for the planning, monitoring, control, and interpretation application.	Backward chaining is suitable for diagnostic, prescription, and debugging application.
7.	Forward chaining can generate an infinite number of possible conclusions.	Backward chaining generates a finite number of possible conclusions.
8.	It operates in the forward direction.	It operates in the backward direction.
9.	Forward chaining is aimed for any conclusion.	Backward chaining is only aimed for the required data.

Matching Techniques

Matching Methods:

- Specifying the propensity score or other distance measure (distance)
- Implementing common support restrictions (discard)
- Caliper matching (caliper)
- Mahalanobis distance matching (mahvars)
- Exact matching (exact)
- Anti-exact matching (antiexact)
- Matching with replacement (replace)

- The matching algorithm can be summarized as follows.
- A block of records is read from data source or in a Reference Match from both a data source and reference source.
- All columns are compared and a composite weight is computed for each possible record pair in the block.

- This helps to ensure that any observed differences in outcomes between the treatment and comparison groups can be more confidently attributed to the treatment itself, rather than other factors that may differ between the groups.
- Matching and DiD can use pre-treatment outcomes to correct for selection bias.

What is the score matching technique?

- Propensity score matching (PSM) is a quasi-experimental method in which the researcher uses statistical techniques to construct an artificial control group by matching each treated unit with a non-treated unit of similar characteristics.
- Using these matches, the researcher can estimate the impact of an intervention.

Partial Matching

- A partial matching is a collection A of arcs with the requirement that each vertex is contained in at most one arc.
- This can be represented visually by a graph G = (V,A) called an arc diagram.

- Partial match means that two DNA profiles, while not an exact match, share a sufficient number of characteristics to indicate the possibility of a biological relationship.

Partial Matching

- For many AI applications complete matching between two or more structures is inappropriate
- For example, input representations of speech waveforms or visual scenes may have been corrupted by noise or other unwanted distortions.
- In such cases, we do not want to reject the input out of hand. Our systems should be more tolerant of such problems
- We want our system to be able to find an acceptable or best match between the input and some reference description

Compensating for Distortions

- Finding an object in a photograph given only a general description of the object is a common problems in vision applications
- For example, the task may be to locate a human face or human body in photographs without the necessity of storing hundreds of specific face templates

- A better approach in this case would be to store a single reference description of the object
- Matching between photographs regions and corresponding descriptions then could be approached using either a measure of correlation, by altering the image to obtain a closer fit

- If nothing is known about the noise and distortion characteristics, correlation methods can be ineffective or even misleading. In such cases, methods based on mechanical distortion may be appropriate
- For example, our reference image is on a transparent rubber sheet. This sheet is moved over the input image and at each location is stretched to get the best match alignment between the two images
- The match between the two can then be evaluated by how well they correspond and how much push-and-pull distortion is needed to obtain the best correspondence
- Use the number of rigid pieces connected with springs. This pieces can correspond to low level areas such as pixels or even larger area segments

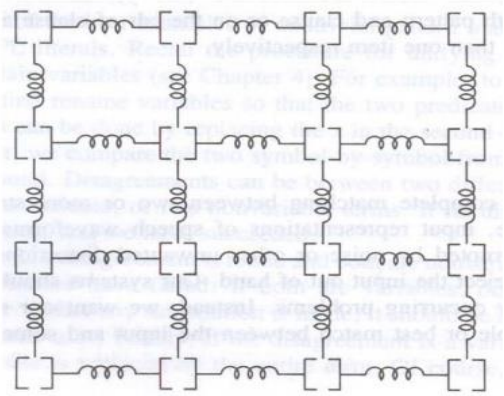


Fig Discrete version of stretchable overlay image

Fuzzy Matching Algorithms

What Is Fuzzy Matching?

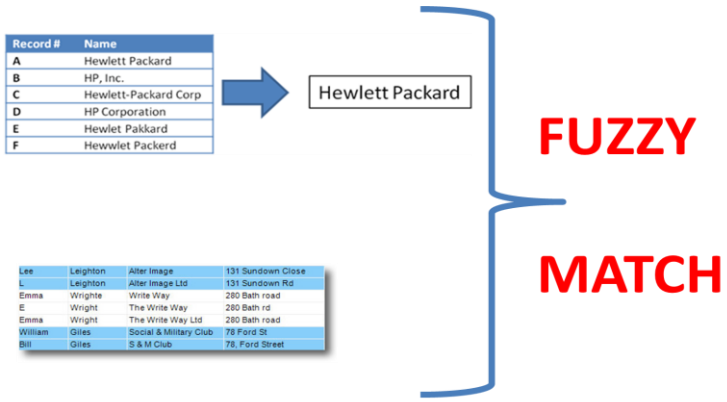
- Fuzzy Matching or Approximate String Matching is among the most discussed issues in computer science.
- In addition, it is a method that offers an improved ability to identify two elements of text, strings, or entries that are approximately similar but are not precisely the same.
- In other words, a fuzzy method may be applied when an exact match is not found for phrases or sentences on a database.
- Indeed, this method attempts to find a match greater than the application-defined match percentage threshold.

Applications of Fuzzy Matching

- We can find fuzzy searches in different applications.
- This technique search resolves the complexities of spelling in all languages, rushed-for-time typers, and clumsy fingers.
- Fuzzy searches are also used to gather user-generated data.
- This is usually unreliable because users misspell the same word or use localized spelling.
- Also, this includes sound and phonetics-based matching.
- Moreover, common applications of approximate matching include matching nucleotide sequences in front of the availability of large amounts of DNA data.
- In addition, we can use approximate matching in spam filtering and record linkage here records from two disparate databases are matched.

Algorithms Used for Fuzzy Matching:

- Naive Algorithm. Among the several pattern search algorithms, naive pattern searching is the most basic. ...
- Hamming Distance Algorithm. ...
- Levenstein Distance Algorithm. ...
- N-gram Algorithm. ...
- BK Tree Algorithm. ...
- Bitap Algorithm.



Fuzzy Match Graphic by Megter.com

- Fuzzy searching matches the meaning, not necessarily the precise wording or specified phrases.
- It performs something the same as full-text search against data to see likely misspellings and approximate string matching.
- it's a very powerful tool that takes into consideration the context of the phrase you wish to look.
- Fuzzy Search is additionally called as approximate string matching.
- It's powerful because often text data is messy. For instance, shorthand and abbreviated text are common in various sorts of data.
- Additionally, outputs from OCR or voice to text conversions tend to be messy or imperfect. Thus, we want to make the foremost of our data by extrapolating the maximum amount of information as possible.
- Fuzzy search is more powerful than exact searching when used for research and investigation. Fuzzy search is very useful when researching unfamiliar, foreign-language, or sophisticated terms, the correct spellings of which don't seem to be widely known.
- Fuzzy search can also be used to locate individuals supported incomplete or partially inaccurate identifying information.

RETE Matching Algorithms

- The Rete algorithm is an algorithm that organizes its data in a tree-like structure.
- Each node in the tree represents a test which will get performed against data being added or removed from memory.
- Each node will have one or two inputs and possibly many outputs.

What is Rete?

- The Rete algorithm is a pattern matching algorithm designed by Dr Charles L. Forgy of Carnegie Mellon University.
- Rete is a Latin word which means net. It is a very efficient algorithm for matching facts against the patterns in rules.
- Understanding of the Rete algorithm will make one easier to understand why writing rules one way is more efficient than writing them another way.

Need of Rete Algorithm

- A complete rule-set should be given to the rule engine for further processing.
- The rule engine matches each rule (i.e. condition) in the ruleset with given facts to decide whether to fire (or execute) the rule or not.
- This is called pattern matching process and this process takes place repeatedly.
- In each cycle the list of facts may be modified: new facts may be added to the list or old facts may be removed from the list.
- These changes may cause previously unsatisfied patterns to be satisfied. Moreover during each cycle the set of rules satisfied must be maintained and updated.
- In most of the cases, actions of the rules change only a few facts in the list. This is called temporal redundancy.
- If a rule engine checks each rule to direct the search for all the facts even if most of them are not modified then it will slow down the process.
- This unnecessary computation can be avoided by remembering what has already matched from cycle to cycle and then computing only the changes necessary for the newly added or removed facts. Rete algorithm does this work perfectly.

Rete Algorithm: Implementation

- The Rete algorithm is implemented by building a network of nodes.
- It is designed in such a way that it saves the state of the matching process from cycle to cycle and re-computes changes only for the modified facts.
- The state of the matching process is updated only as facts are added and removed.

- If the facts added or removed are less in number then the matching process will be faster.

The Inference Cycle

- Each rule has an inference cycle consisting of three phases: match, select and execute. In matching phase, the conditions of the rules are matched against the facts to determine which rules are to be executed.
- The rules whose conditions are met are stored in a list called agenda for firing. From the list of rules, one of the rules is selected to execute or fire.
- The selections strategy may depend on priority, recency of usage, specificity of the rule, or on other criteria.
- The rule selected from the list is executed by carrying out the actions in the right hand side of the rule.
- The action may be an assertion, executing a user defined or built-in function or executing a decision table, or otherwise.
- Note that the decision table engine is used to execute decision tables.

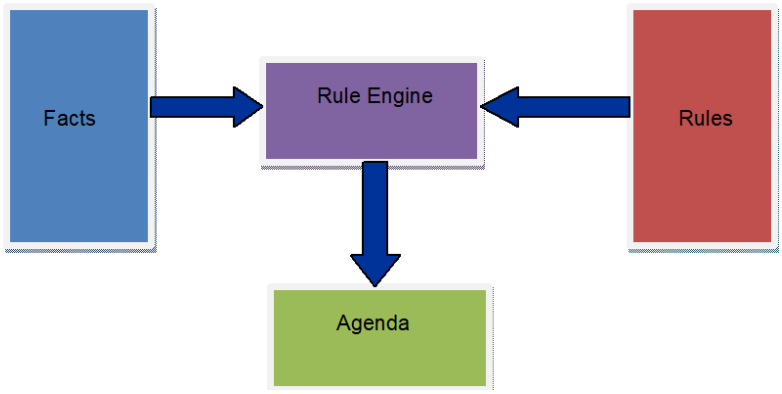


Figure1. Pattern matching: Rules and Facts

Building of the Rete Network

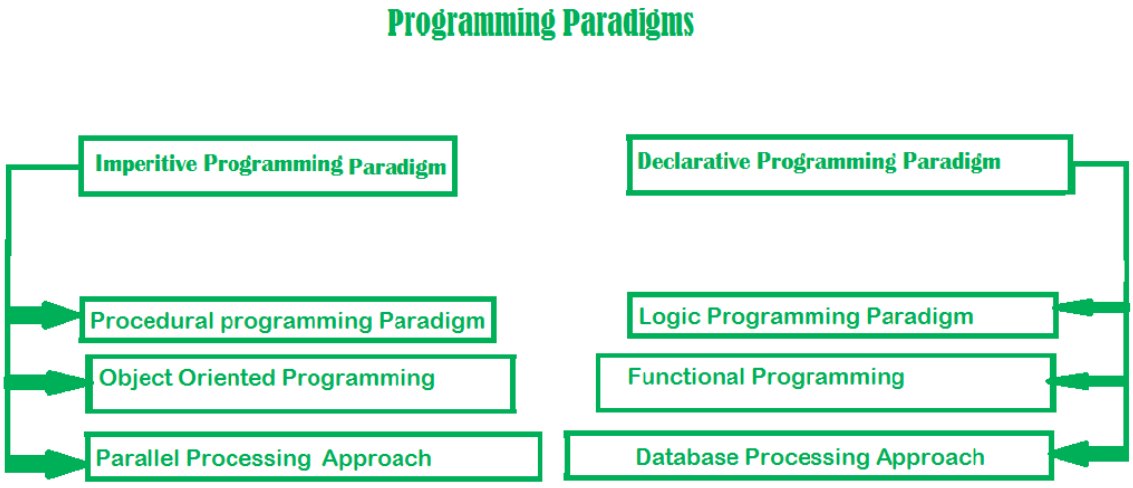
- The Rete network is a direct acyclic graph consists of nodes representing patterns in the conditions of the rules.
- The nodes behave like filters, testing the incoming tokens, and sending only those that pass the test.

- The rete network consists of two parts: alpha network and beta network.
- Alpha network consists of nodes known as alpha nodes.
- Each alpha node has one input that defines intra-elements. Beta network consists of beta nodes where each node takes two inputs to define inter-element conditions.
- The Rete network starts with the root node called Rete Node.
- Kind nodes follow the root node. There should be a kind node for each fact type.
- Alpha nodes are then created for each pattern and connected to the corresponding kind node.
- A condition, for example, a>b or b>c has two patterns, so two alpha nodes will be created: one for a>b and one for b>c. Each alpha node is associated with a memory known as alpha memory. It is used to remember the facts matched.
- Alpha nodes are then joined by beta nodes. Beta node accepts only two inputs. So if there are three alpha nodes then first two nodes will be joined by one beta node.
- Thereafter the output of this beta node and the third alpha node will be joined by another beta node. This way beta nodes support partial matching.
- Each beta node has a memory to store joined patterns.
- Assertion of each fact creates a token. Initially the tokens enter the root node.
- The network then splits a branch for each token type.
- Each kind node gets a copy of the token and performs SELECT operation to select tokens of its kind.
- The kind node delivers a copy of the token node it receives to the alpha nodes. On receiving the token, the alpha nodes do a PROJECT operation and extract components from the token that match with the variables of the pattern.
- That is alpha node, basically, evaluates the conditions. Beta node then determines the possible cross product for a rule.
- Finally actions in the rule will be executed.

Logic Based Programming

- Logical Programming is a type of programming paradigm that uses logic circuits to control how facts and rules about the problems within the system are represented or expressed.
- In it, logic is used to represent knowledge, and inference is used to manipulate it.
- Programming paradigm is an approach to solve problems using some programming language or also we can say it is a method to solve a problem using tools and techniques that are available to us following some approach.
- There are lots of programming languages that are known but all of them need to follow some strategy when they are implemented and this methodology/strategy is paradigms.
- Apart from varieties of programming languages, there are lots of paradigms to fulfill each and every demand.

They are discussed below as follows:



- Functional Programming is a type of programming paradigm in which everything is done with the help of functions and using functions as its basic building blocks.
- In it, we simply try to bind each and everything in a purely mathematical functions' style.
- Programs are generally written at a higher level and are therefore much easier to comprehend.

- Logical Programming is a type of programming paradigm that uses logic circuits to control how facts and rules about the problems within the system are represented or expressed.
- In it, logic is used to represent knowledge, and inference is used to manipulate it.
- It tells the model about how to accomplish a goal rather than what goal to accomplish.

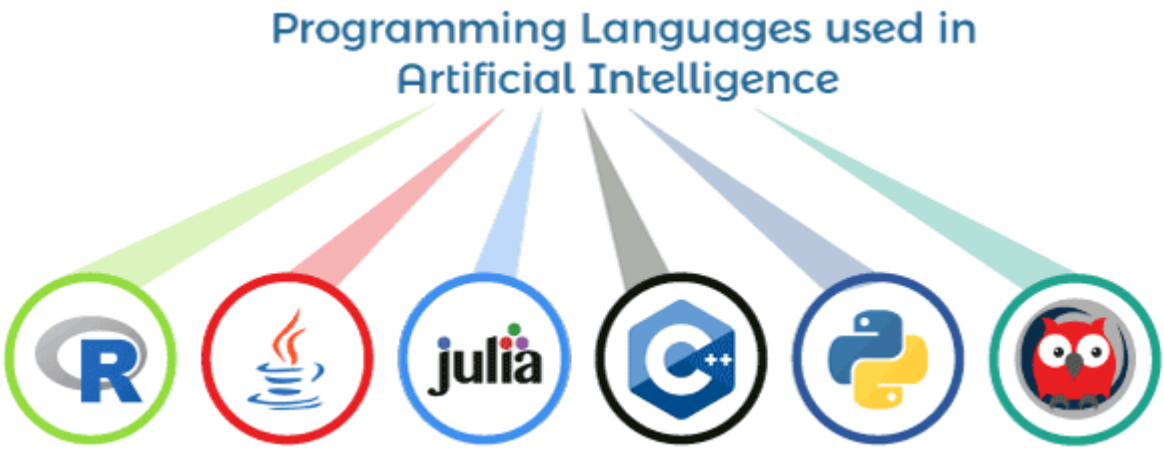
Logical Programming

- It is totally based on formal logic.
- In this programming paradigm, program statements usually express or represent facts and rules related to problems within a system of formal logic.
- These are specially designed for fault diagnosis, natural language processing, planning, and machine learning.
- Its main aim is to allow machines to reason because it is very useful for representing knowledge.
- Some languages used for logic programming include Absys, Cycl, Alice, ALF (Algebraic logic functional programming language), etc.
- It is data-driven, array-oriented, used to express knowledge, etc.
- It usually supports the logic programming paradigm.
- Testing is comparatively more difficult as compared to functional programming.
- It simply uses predicates. Here, the predicate is not a function i.e., it does not have a return value.

AI Programming languages

Languages used in Artificial Intelligence

- Artificial Intelligence has become an important part of human life as we are now highly dependent on machines.
- Artificial Intelligence is a very important technology to develop and build new computer programs and systems, which can be used to simulate various intelligence processes like learning, reasoning, etc.



- Python
- R
- Lisp
- Java
- C++
- Julia
- Prolog

1. Python

Python is one of the most powerful and easy programming languages that anyone can start to learn. Python is initially developed in the early stage of 1991. Most of the developers and programmers choose Python as their favourite programming language for developing Artificial Intelligence solutions. Python is worldwide popular among all developers and experts because it has more career opportunities than any other programming language.

Python also comes with some default sets of standards libraries and also provides better community support to its users. Further, Python is a platform-independent language and also

provides an extensive framework for Deep Learning, Machine Learning, and Artificial Intelligence.

Python is also a portable language as it is used on various platforms such as Linux, Windows, Mac OS, and UNIX.

Features of Python

- It is easy to learn than any other programming language.
- It is also a dynamically-typed language.
- Python is an Object-oriented language.
- It provides extensive community support and a framework for ML and DL.
- Open-source.
- Large standard sets of libraries.
- Interpreted language.

Python is an ideal programming language used for Machine Language, Natural Processing Language (NLP), and Neural networks, etc. Due to the flexible nature of Python, it can be used for AI development. It contains various pre-existing libraries such as Pandas, SciPy and nltk, etc. Further, Python also contains simple syntax and easy coding, which makes Python the first choice of AI developers and programmers.

There are some standard Libraries in Python used for Artificial Intelligence as follows:

1. Tensor Flow Python
2. Keras Python
3. Theano Python
4. Scikit-Learn Python
5. PyTorch Python
6. NumPy Python
7. Python Pandas
8. Seaborn Python

2. Java

Java is also the most widely used programming language by all developers and programmers to develop machine learning solutions and enterprise development. Similar to Python, Java is also a platform-independent language as it can also be easily implemented on various platforms. Further, Java is an object-oriented and scalable programming language. Java allows virtual machine technology that helps to create a single version of the app and provides support to your business. The best thing about Java is once it is written and compiled on one platform, then you do not need to compile it again and again. This is known as WORA (Once Written Read/Run Anywhere) principle.

Features of Java

Java has so many features which make Java best in industry and to develop artificial intelligence applications:

- Portability
- Cross-platform.
- Easy to learn and use.
- Easy-to-code Algorithms.
- Built-in garbage collector.
- Swing and Standard Widget Toolkit.
- Simplified work with large-scale projects.

- Better user interaction.
- Easy to debug.

3. Prolog

Prolog is one of the oldest programming languages used for Artificial Intelligence solutions. Prolog stands for "Programming in Logic", which was developed by French scientist Alain Colmerauer in 1970.

For AI programming in Prolog, developers need to define the rules, facts, and the end goal. After defining these three, the prolog tries to discover the connection between them. Programming in AI using Prolog is different and has several advantages and disadvantages.

It may seem like a bizarre language to learn for those programmers who are from a C++ background.

Prolog may not be a great programming language to build something big, but it's a great language to study and think about problems in more logical ways rather than procedural.

Features of Prolog

- Supports basic mechanisms such as
- Pattern Matching,
- Tree-based data structuring, and
- Automatic backtracking.
- Prolog is a declarative language rather than imperative.

4. Lisp

Lisp has been around for a very long time and has been widely used for scientific research in the fields of natural languages, theorem proofs, and to solve artificial intelligence problems. Lisp was originally created as a practical mathematical notation for programs but eventually became a top choice of developers in the field of AI.

Although Lisp programming language is the second oldest language after Fortran, it is still being used because of its crucial features. The inventor of LISP programming was John McCarthy, who coined the term Artificial Intelligence.

LISP is one of the most efficient programming languages for solving specific problems. Currently, it is mainly used for machine learning and inductive logic problems. It has also influenced the creation of other programming languages for AI, and some worth examples are R and Julia.

However, though being so flexible, it has various deficiencies, such as lack of well-known libraries, not so-human-friendly syntax, etc. Due to this reason, it is not preferred by the programmers.

Features of LISP

- The program can be easily modified, similar to data.
- Make use of recursion for control structure rather than iteration.
- Garbage Collection is necessary.
- We can easily execute data structures as programs.
- An object can be created dynamically.

5. R

R is one of the great languages for statistical processing in programming. However, R supports free, open-source programming language for data analysis purposes. It may not be the perfect language for AI, but it provides great performance while dealing with large numbers.

Some inbuilt features such as built-in functional programming, object-oriented nature, and vectorial computation make it a worthwhile programming language for AI.

R contains several packages that are specially designed for AI, which are:

- **gmodels** - This package provides different tools for the model fitting task.
- **TM** - It is a great framework that is used for text mining applications.
- **RODBC** - It is an ODBC interface.
- **OneR** - This package is used to implement the One Rule Machine Learning classification algorithm.

Features of R programming

- R is an open-source programming language, which is free of cost, and also you can add packages for other functionalities.
- R provides strong & interactive graphics capability to users.
- It enables you to perform complex statistical calculations.
- It is widely used in machine learning and AI due to its high-performance capabilities.

6. Julia

Julia is one of the newer languages on the list and was created to focus on performance computing in scientific and technical fields. Julia includes several features that directly apply to AI programming.

Julia is a comparatively new language, which is mainly suited for numerical analysis and computational science. It contains several features that can be very helpful in AI programming.

Features of Julia

- Common numeric data types.
- Arbitrary precision values.
- Robust mathematical functions.
- Tuples, dictionaries, and code introspection.
- Built-in package manager.
- Dynamic type system.
- Ability to work for both parallel and distributed computing.
- Macros and metaprogramming capabilities.
- Support for multiple dispatches.
- Support for C functions.

7. C++

C++ language has been present for so long around, but still being a top and popular programming language among developers. It provides better handling for AI models while developing.

Although C++ may not be the first choice of developers for AI programming, various machine learning and deep learning libraries are written in the C++ language.

Features of C++

- C++ is one of the fastest languages, and it can be used in statistical techniques.
- It can be used with ML algorithms for fast execution.
- Most of the libraries and packages available for Machine learning and AI are written in C++.
- It is a user friendly and simple language.

Overview of LISP

Introduction to LISP

- Lisp is a programming language that has an overall style that is organized around expressions and functions.
- Every Lisp procedure is a function, and when called, it returns a data object as its value. It is also commonly referred to as “functions” even though they may have side effects.

- Lisp is the second-oldest high-level programming language in the world which is invented by John McCarthy in the year 1958 at the Massachusetts Institute of Technology.

Features of Common LISP

- It is machine-independent
- It uses iterative design methodology, and easy extensibility.
- It allows updating the programs dynamically.
- It provides high level debugging.
- It provides advanced object-oriented programming.
- It provides a convenient macro system.
- It provides wide-ranging data types like, objects, structures, lists, vectors, adjustable arrays, hash-tables, and symbols.
- It is expression-based.
- It provides an object-oriented condition system.
- It provides a complete I/O library.
- It provides extensive control structures.

Applications Built in LISP

Large successful applications built in Lisp.

- Emacs
- G2
- AutoCad
- Igor Engraver
- Yahoo Store

Pattern Matching in Lisp

Pattern matching is done by using "Rules". A "rule" consists of three elements:

1. A "pattern" to match against.
2. Some optional extra conditions.
3. A "skeleton" which tells a rewrite-system what to do with the matches.

- The pattern specifies the class of expressions it matches. Patterns contain constants and pattern variable which indicate arbitrary elements or segments (sequences of elements).
- A pattern variable matching a single element is prefixed by a single question mark, and segment variables matching multiple elements are prefixed by two question marks: `(? pattern)` and `(?? segment)`.
- We can further refine our pattern matching using predicates. For example, if we want to match a single element which is a pattern, we would write `(? n numberp)` (or, in Scheme, something like `(? n number?)`).
- We say a rule is "Applicable" to an expression if the rule's pattern matches the provided expression.
- Once a rule is applicable, we instantiate the skeleton to create an expression with the matched subexpressions.

Matching Patterns Against an Expression

There are several cases we need to consider:

1. If we have failed, we should set the dictionary to `:fail`, and bail out.
2. If the pattern is a pattern variable, then we should check the expression matches the pattern variable (relative to whether we've already bound the variable in our dictionary). This is handled by a `match-element` helper function.

3. If the pattern is just some constant, then…well, we're nearly done! So check the expression is the same constant. If so, return the dictionary; otherwise, we've failed, and return :fail instead of a dictionary.
4. If the pattern is a segment variable, we delegate the messy work to a match-segment helper function.
5. If the pattern is a list — some complicated S-expression — which is our default case, we form new-bindings trying to match the head of the pattern with the head of the expression. Unless new-bindings is a :fail-ure, we recursively match on the rest of the pattern against the rest of the expression.

An Expert system Shell in LISP

- This expert system shell is called lisp-shell.
- The logic programming interpreter returned a stream of the substitution sets under which a goal logically followed from a database of logical assertions.
- Expert systems (ES) are one of the prominent research domains of AI.
- It is introduced by the researchers at Stanford University, Computer Science Department.

What are Expert Systems?

- The expert systems are the computer applications developed to solve complex problems in a particular domain, at the level of extra-ordinary human intelligence and expertise.

Characteristics of Expert Systems

- High performance
- Understandable
- Reliable
- Highly responsive

Capabilities of Expert Systems

The expert systems are capable of:

- Advising
- Instructing and assisting human in decision making
- Demonstrating
- Deriving a solution
- Diagnosing
- Explaining
- Interpreting input
- Predicting results
- Justifying the conclusion
- Suggesting alternative options to a problem

They are incapable of:

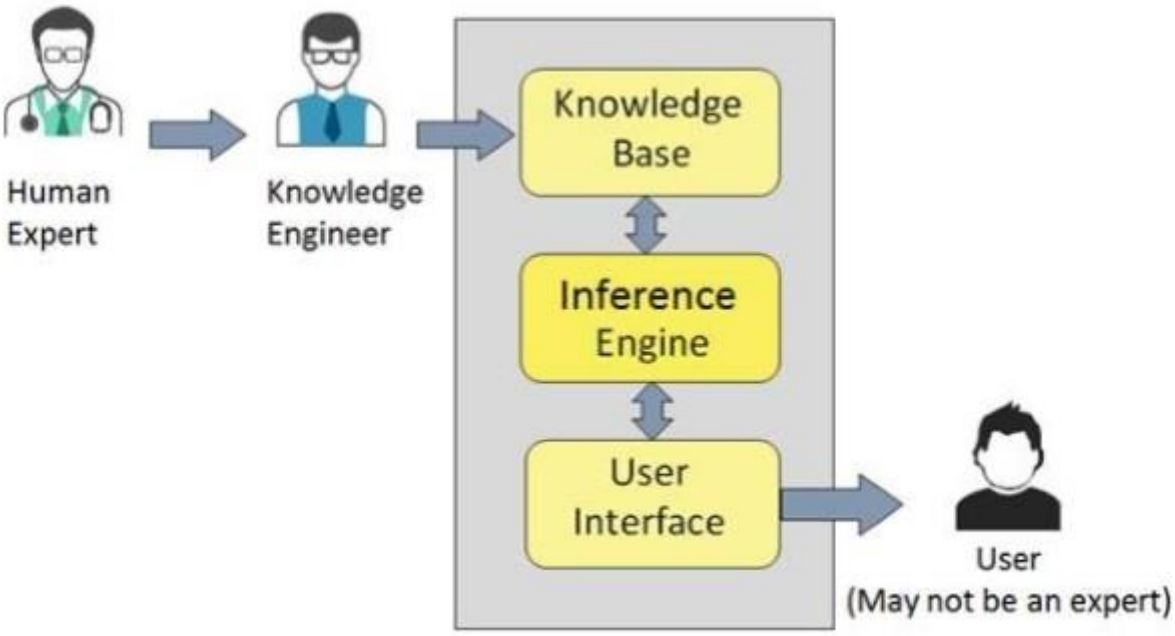
- Substituting human decision makers
- Possessing human capabilities
- Producing accurate output for inadequate knowledge base
- Refining their own knowledge

Components of Expert Systems

The components of ES include:

- Knowledge Base
- Inference Engine
- User Interface

Let us see them one by one briefly:



Expert System Technology

There are several levels of ES technologies available. Expert systems technologies include –

- **Expert System Development Environment :** The ES development environment includes hardware and tools.
- **They are:**
  - Workstations, minicomputers, mainframes.
  - High level Symbolic Programming Languages such as LISt Programming (LISP) and PROgrammation en LOGique (PROLOG).
  - Large databases.
- **Tools:** They reduce the effort and cost involved in developing an expert system to large extent.
  - Powerful editors and debugging tools with multi-windows.
  - They provide rapid prototyping
  - Have Inbuilt definitions of model, knowledge representation, and inference design.

- **Shells :** A shell is nothing but an expert system without knowledge base. A shell provides the developers with knowledge acquisition, inference engine, user interface, and explanation facility. For example, few shells are given below −
    - Java Expert System Shell (JESS) that provides fully developed Java API for creating an expert system.
    - Vidwan, a shell developed at the National Centre for Software Technology, Mumbai in 1993. It enables knowledge encoding in the form of IF-THEN rules.

Benefits of Expert Systems:

- **Availability**: They are easily available due to mass production of software.
- **Production cost** is reasonable. This makes them affordable.
- **Speed :**They offer great speed. They reduce the amount of work an individual puts in.
- **Less Error Rate :**Error rate is low as compared to human errors.
- **Reducing Risk :** They can work in the environment dangerous to humans.
- **Steady response :**They work steadily without getting motional, tensed or fatigued.

Over view of Prolog

- Prolog or PROgramming in LOGics is a logical and declarative programming language.
- It is one major example of the fourth generation language that supports the declarative programming paradigm.
- This is particularly suitable for programs that involve symbolic or non-numeric computation.

What is Prolog:

- Prolog stands for programming in logic. In the logic programming paradigm, prolog language is most widely available. Prolog is a declarative language, which means that a program consists of data based on the facts and rules (Logical relationship) rather than computing how to find a solution. A logical relationship describes the relationships which hold for the given application.

- To obtain the solution, the user asks a question rather than running a program. When a user asks a question, then to determine the answer, the run time system searches through the database of facts and rules.
- The first Prolog was 'Marseille Prolog', which is based on work by Colmerauer. The major example of fourth-generation programming language was prolog. It supports the declarative programming paradigm.
- In 1981, a Japanese computer Project of 5^{th} generation was announced. After that, it was adopted Prolog as a development language. In this tutorial, the program was written in the 'Standard' Edinburgh Prolog. Prologs of PrologII family are the other kind of prologs which are descendants of Marseille Prolog.
- Prolog features are 'Logical variable', which means that they behave like uniform data structure, a backtracking strategy to search for proofs, a pattern-matching facility, mathematical variable, and input and out are interchangeable.
- To deduce the answer, there will be more than one way. In such case, the run time system will be asked to find another solution. To generate another solution, use the backtracking strategy. Prolog is a weakly typed language with static scope rules and dynamic type checking.
- Prolog is a declarative language that means we can specify what problem we want to solve rather than how to solve it.
- Prolog is used in some areas like database, natural language processing, artificial intelligence, but it is pretty useless in some areas like a numerical algorithm or instance graphics.
- In artificial intelligence applications, prolog is used. The artificial intelligence applications can be automated reasoning systems, natural language interfaces, and expert systems. The expert system consists of an interface engine and a database of facts. The prolog's run time system provides the service of an interface engine.
- A basic logic programming environment has no literal values. An identifier with upper case letters and other identifiers denote variables. Identifiers that start with lower-case letters denote data values. The basic Prolog elements are typeless. The most implementations of prolog have been enhanced to include integer value, characters, and operations. The Mechanism of prolog describes the tuples and lists.

- Functional programming language and prolog have some similarities like Hugs. A logic program is used to consist of relation definition. A functional programming language is used to consist of a sequence of function definitions. Both the logical programming and functional programming rely heavily on recursive definitions.

Applications of Prolog

The applications of prolog are as follows:

- Specification Language
- Robot Planning
- Natural language understanding
- Machine Learning
- Problem Solving
- Intelligent Database retrieval
- Expert System
- Automated Reasoning

Production System using Prolog

It consists of two components: rule and action. Rules recognize the condition, and the actions part has the knowledge of how to deal with the condition. In simpler words, the production system in AI contains a set of rules which are defined by the left side and right side of the system.

Production System in AI

A production system (popularly known as a production rule system) is a kind of cognitive architecture that is used to implement search algorithms and replicate human problem-solving skills. This problem-solving knowledge is encoded in the system in the form of little quanta popularly known as productions. It consists of two components: rule and action.

Rules recognize the condition, and the actions part has the knowledge of how to deal with the condition. In simpler words, the production system in AI contains a set of rules which

are defined by the left side and right side of the system. The left side contains a set of things to watch for (condition), and the right side contains the things to do (action).

What are the Elements of a Production System?

An AI production system has three main elements which are as follows:

- **Global Database:** The primary database which contains all the information necessary to successfully complete a task. It is further broken down into two parts: temporary and permanent. The temporary part contains information relevant to the current situation only whereas the permanent part contains information about the fixed actions.
- **A set of Production Rules:** A set of rules that operates on the global database. Each rule consists of a precondition and postcondition that the global database either meets or not. For example, if a condition is met by the global database, then the production rule is applied successfully.
- **Control System:** A control system that acts as the decision-maker, decides which production rule should be applied. The Control system stops computation or processing when a termination condition is met on the database.

What are the Features of a Production System?

A production system has the following features:

1. **Simplicity**: Due to the use of the IF-THEN structure, each sentence is unique in the production system. This uniqueness makes the knowledge representation simple to enhance the readability of the production rules.
2. **Modularity:** The knowledge available is coded in discrete pieces by the production system, which makes it easy to add, modify, or delete the information without any side effects.
3. **Modifiability:** This feature allows for the modification of the production rules. The rules are first defined in the skeletal form and then modified to suit an application.
4. **Knowledge-intensive:** As the name suggests, the system only stores knowledge. All the rules are written in the English language. This type of representation solves the semantics problem.

What are the Classes of a Production System?

A production system is classified into four main classes which are:

- **Monotonic Production System**: In a monotonic production system, the use of one rule never prevents the involvement of another rule when both the rules are selected at the same time. Hence, it enables the system to apply rules simultaneously.
- **Partially Commutative Production System:** In this production system if a set of rules is used to change state A to state B then any allowable combination of these rules will also produce the same results (convert state A to state B).
- **Non-Monotonic Production System:** This production system increases the problem-solving efficiency of the machine by not keeping a record of the changes made in the previous search process. These types of production systems are useful from an implementation point of view as they do not backtrack to the previous state when it is found that an incorrect path was followed.
- **Commutative Production System:** These type of production systems is used when the order of operation is not important, and the changes are reversible.