Chapter: 2

Artificial Intelligence An overview

- Introduction
- Developments in AI
- Scope of AI
- Expert Systems
- Natural Language Processing
- References
Chapter 2 Artificial Intelligence an overview

2.1 Introduction

Artificial Intelligence (AI) is concerned with developing computer systems which reason and behave in ways a neutral observer would consider human. The definition of the term Artificial Intelligence is difficult because it is difficult to define intelligence itself. There is a general agreement to the meaning of artificial as something man made or which is not natural, but there is no consensus as to the meaning of the term intelligence. Intelligence may be defined as any of the following:

- ability to reason
- ability to acquire knowledge (learning)
- ability to perceive
- ability to create (innovate)

All the above mentioned abilities are different facets of intelligence. Rich (1990) (1) defines AI as the study of how to make computer do things at which at the moment people are better. Winston (1984) (2) defines AI as the study of ideas that enable computer to be intelligent. A more comprehensive definition is presented by Patterson (1992) (3) who defines AI as a branch of computer science concerned with the study and creation of computer systems that exhibit some form of intelligence. Systems that can learn new concepts and tasks, systems that reason and draw useful conclusions about the world around us, systems that can understand a natural language or perceive or comprehend a visual scene, and systems that perform other types of feats that require human types of intelligence.

The advantages of a system that has human intelligence are many. Such systems can be used wherever human intelligence is required like in business engineering, manufacturing, mining, etc. In addition, such study
has intrinsic advantages as it demands understanding of the human thinking process so as to develop computational models. The following reasons given by Winston (2) justify the study of AI as means of understanding human intelligence:

1. The use of computer demands a clear statement of problems and clear strategy for solutions and this requires a clear thinking about human thinking process.

2. Computer models force precision. Implementing a theory uncovers conceptual mistakes and oversights that ordinarily escape even the most meticulous researchers.

3. Computer implementations quantify task requirements. Once a program performs a task, upper bound statements can be made about how much information processing the task requires.

4. It is usually simple to deprive a computer program of some piece of knowledge in order to test how important that information really is.

### 2.2 Developments in AI

Although AI is relatively a new subject, much of the work which later laid the foundation of AI can be traced back to the last century. Foremost among them is the work of George Boole (1815–1864) on boolean algebra in which he introduced the logical definitions of and, or, etc. Another important contribution was made by Alan Turing (1912–1954) (4) considered as the father of AI. He worked with precursors of modern computers. Turing envisaged in his paper Computing machinery and intelligence that computers could be programmed to exhibit intelligent...
behavior. His most important contribution undoubtedly is the Turing Test. In that test, an operator sits at a keyboard and queries an unknown source which can be either a human being or machine. If the machine provides answers that do not give any clue to the operator whether he is getting answers from the human being or machine, then that machine qualifies as having human-like abilities and reasoning.

The emergence of a new field, cybernetics (coined by Norbert Wiener), brought together many parallels between human beings and machines. Cybernetics is the study of communication in human and machine and combines the concepts from information theory, feedback control systems (both biological and machine), and electronic computers.

The development in the field of linguistics, especially the contribution of Noam Chomsky in formal grammar, helped to a large extent the developments in Natural Language Processing.

However, many consider 1956 as a landmark in the history of AI. In this year, John McCarthy and Marvin Minsky organized the Dartmouth Conference. It is in this conference that McCarthy coined the word 'Artificial Intelligence.'

In 1963, A.L. Samuel wrote a checkers playing program which not only played the game with opponents but also used its experience to improve its performance. In the same year, A Newel developed 'Logic Theorist,' which attempted to prove mathematical theorems. It solved some of the problems in the chapter of Russell and Whitehead's *Principia.* Newel along with J.C. Shaw and H.A. Simon developed yet another program, 'General Problem Solver' (GPS), which they applied to several tasks including symbolic manipulation of logical expressions.
Chapter 2  Artificial Intelligence an overview

2.3  Scope of AI

Although AI is considered largely as a branch of computer science, it is a highly interdisciplinary area making use of ideas from various fields of knowledge like logic, cybernetics, linguistics, information theory, psychology, epistemology, etc. In fact, Schank (1984) believes that it should be considered as a branch of psychology. AI includes:

- Getting computers to communicate with users in human languages like English either by keying in or by speaking to the computer (Natural Language Processing, Voice/speech recognition, Optical Character Recognition).
- Getting computers to deduce conclusions from generalizations (deduction, inference).
- Getting computers to plan sequences of action to accomplish goals (planning).
- Getting computers to recognize patterns in nature (Pattern recognition, Vision).
- Getting computers to offer advice on complicated rules for various situations (Expert systems).
- Getting computers to move objects around in the real world.

The scope of Artificial Intelligence comprises the following areas of which only those concerned with Library and Information Science application are explained in detail:

- Expert systems
- Natural Language Processing
- Pattern Recognition
- Robotics
2.4 Expert Systems

This is one of the major areas of Artificial Intelligence with a wide application to many disciplines. Expert systems are also referred as Knowledge based Systems' or 'Knowledge based Computer Systems'. Expert systems perform in a manner similar to human expert in a specific domain of expertise. The initial efforts to build general problem solving all round expert systems yielded disappointing results. However, DENDRAL (6) brought a shift in Expert System paradigm and changed the course of Artificial Intelligence research emphasising the creation of expert system in a very narrow specific area. DENDRAL was developed at Stanford University in mid sixties for determining the topological structure of organic compounds synthesising organic molecules and planning experiments in molecular biology.

Some of the Expert systems that have been built in different subject areas are MYCIN in Medicine, PROSPECTOR in Geology, TAXADVISER in Law, XCON in computer systems.

In addition, many commercial vendors are offering expert system shells. These shells could be used to build expert systems in many subjects without bothering about the programming skills. This is one of the areas of Artificial Intelligence that offers lot of promise to Library and Information science Professionals. One of the major projects in Library and information Science is PLEXUS' under the leadership of B C Vickery.
2.4.1 KNOWLEDGE BASED SYSTEMS

Some branches of AI have moved beyond pure research. They seek to apply what is known as AI science; they produce marketable AI technologies. One such branch is robotics, which seeks to give computers human mobility and sensory capabilities. Another is Knowledge Based Systems.

Knowledge Based Systems are computer systems designed to simulate the behaviour of human experts in a narrow domain of knowledge. KBS are relatively new and growing branch of AI. Knowledge Based Systems are based on the idea that humans commonly use many types of knowledge when solving problems. Knowledge is an integrated collection of information which when used produces a competent level of performance. Two of the most important forms of knowledge are factual knowledge and heuristic knowledge.

The factual knowledge is the type of learning a person usually gains from reading books and attending school. It is material commonly agreed with and shared. It is public knowledge. Factual knowledge provides a set of general theories. It can suggest highly structured solution processes called algorithms for solving problems in a finite number of steps.

Heuristic knowledge is the type of learning a person gains from experience. Heuristics are rules of thumb or situation-specific theories. Since each person's experiences are unique, heuristics knowledge is personal and private. People use heuristic knowledge every day to solve problems.

In designing Knowledge Based Systems, AI researchers have tried to imitate the working of human mind as well as the behaviour of human
experts Consequently most Knowledge Based System applications incorporate number of components

![Diagram of Knowledge Based System components](image)

**Fig 2.1** Components of Knowledge Based System

The Knowledge Base basically consists of two forms of knowledge as mentioned above i.e., facts, rules, rules of thumb (i.e., heuristics) and opinions or beliefs using one or more schemes for representing the knowledge.

Perhaps the simplest and certainly the most common knowledge representation scheme is 'if then' rules. The IF part contains a set of one or more conditions which must be met before the rule can be used. The THEN part is a set of one or more actions or consequences which occur when a rule does fire.

Rules are excellent for representing some types of knowledge like procedural knowledge. However, other types require different representation structures such as Frames for hierarchies etc.
Knowledge Base (KB) is analogous to long term memory in humans. KB is the place where the computer stores a description of the problem and tracks its solution effort. It is also the place where the Knowledge Base System keeps a record of data gained from external sources like measuring instruments or questions posed to the users.

The inference engine makes use of the knowledge contained in the Knowledge Base and facts supplied by the user to solve the problem at hand. Various control methods guide the reasoning process. Basically it uses either backward chaining or forward chaining reasoning. Backward chaining starts with a goal and tries to verify if all the conditions that lead to the goal are true. In forward chaining procedure, the inference engine tries to establish the conditions that are true and in that process decides which goal satisfies the conditions shown to be true.

The other component of Knowledge Base system is explanation facility. This is important to maximise its effectiveness. This is similar to human experts who defend their judgements to inspire confidence. The user can ask the Knowledge Base System to explain how a certain conclusion is reached and the system can display the rules that led to the conclusion.

There may be other modules such as user interface and knowledge acquisition modules. The user interface module enables the user to communicate with the knowledge Base System in terms of supplying data to the system and asking for explanations. The knowledge acquisition module allows the user or the Knowledge Engineer to add or modify the knowledge contained in Knowledge Base.
The basic components of an Expert system are a knowledge base and an inference engine. The knowledge base may further be logically divided into the following:

- The Meta Knowledge base (MKB)
- The Specification Knowledge Base (SKB)
- The Ground Knowledge Base (GKB)

The meta knowledge base contains syntactic and semantic knowledge about the integrated model of data and processes to be used and it contains mainly of rules both for deduction (Knowledge Flow) and checking (constraints). These rules designate structures.

The specification knowledge base contains information about particular applications not necessarily disjoint especially if these applications are related to different aspects of the same management environment. This information is covered by clusters of connected frames and captures the presence or absence of contradictions in the ground knowledge. Each application models information about a system or part of it (an organization or its functional departments for instance). The specifications designate to entities and expressions.

The ground knowledge base consists of information about instances of the specification sort and depending on the application, it may include more or less large amount of extensive data. The naming of those instances is essential for most management purposes (e.g., identification of employees of an enterprise) but irrelevant for other purposes (e.g., simulation of a production line).
Chapter 2 Artificial Intelligence: An overview

There are three major models of Knowledge Representation viz. Rule based, Semantic nets, and Frame based. The present work has used the Frame based knowledge representation model to represent semantics of given subject.

2.4.1.1 Frame Based Knowledge Representation

AI pioneer Marvin Minsky invented the frame concept. In his words (7),

A frame is a data structure for representing a stereotyped situation like being in a certain kind of living room or going to a child's birthday party. Attached to each frame are several kinds of information. Some of the information is about how to use the frame. Some is about what one can expect to happen next. Some is about what to do if these expectations are not confirmed.

We can think of a frame as a network of nodes and relations. The top levels of a frame are fixed and represent things that are always true about the supposed situation. The lower levels have many terminal slots that must be filled by specific instances of data.

Collections of related frames are linked together into frame systems. The effects of important actions are mirrored by transformation between the frames of a system.

Frames are in a way abstractions on groups of facts. Frames play an important role in dealings with potentially large numbers of facts because they help organize facts. Frames are something called classes or prototype or structured object descriptions. The objective of a frame system is to partition data just as modular programming partitions programs. The usual
way is to group together facts about the same thing or object or grouping facts with the same argument values

Frames like semantic networks are intended for two argument predicates. This includes relationship predicates in which both arguments are facts and property predicates in which the first argument is an object and the second argument is a property of that object. In general, it is advised to convert predicates with more than two arguments into two argument predicates expressing relationship or property of objects. Relationship predicates are generally stored only with the first argument frame, then for the second argument frame a different reverse predicate name is used. For instance:

For a kind of \((x \ y)\) the reverse predicate is a generalization of \((y \ x)\).

For part of \((x \ y)\) the reverse predicate in contains \((y \ x)\).

The frames are not necessarily disjoint i.e., a certain object may be represented by more than one frame depending on the context.

Frames can represent things with components. Often just listing component names is insufficiently descriptive. Instead, we should have a separate frame for each component and contains slots in the frame for the whole whose values are pointers to the component frames. This has several advantages (7):

- We can distinguish properties of components not shared by the whole inheritance can always be used otherwise.

- We can describe relationship of the components to one another like the relative location of the parts of a physical object or the relative
time of sub events of an action. Frames for which components can be put in a sequence are called scripts.

- We can distinguish multiple or optional occurrences of a component as instances of some component type frame whose variation as occurrence is described by qualifying slots of the contains slots. For instance, a wheel is a part of a car, and cars have four of them, each with their own properties, and a sunroof is an optional part of a car occurring either zero or one times in every car.

- We only need to identify the most important top level components in our main frame. The frames for these components can describe sub components and so on.

A frame is more than a collection of facts. It is an abstraction in its own right. Often we feel that certain facts are the only essential arcs about some objects, the ones giving descriptive completeness.

However, we may want to describe a set of objects with a frame, not just a single object. That is, the frame should generalize about certain groups or classes of objects, identifying properties each group shares.

Frames in general contain slots that can be unified or filled. Filled slots represent facts. Some slots are so important to fill that their frames do not make sense otherwise. Such slots are definitional part of the definition of the frame.
 Slots with non numeric information can have a format (a formal description of what the values look like) Slots representing real world data can have an associated location, time, and observer. Slots can have default or usual values. Slots may have associated sets of permissible values given by a list if the number of such values is finite or by a range of the values. Slots may also have sets of 'unusual' values that should generate a warning if seen.

Slot anticipation information can be considered as slot itself and are called qualifying slots. They can inherit like regular slots. The values in qualifying slots generally inherit from a higher generalization frame than do the values in the slots they describe when inherited.

2.4.1.1 Value and Slot Inheritance

Slot inheritance is distinct in the sense it is the inheritance of the mere concept of a slot and not the value in a slot. In other words, slot inheritance is distinct from value inheritance. In one sense, slot inheritance is more limited than value inheritance in that it usually only works with the a kind of relationship. That is, situations when some frame F has a slot S saying then anything that is a kind of F also has slot S. But in another sense, slot inheritance is more general than value inheritance because it applies higher up in frame hierarchies to slots that do not yet have values filled in.

Usually slot inheritance tells us most of the slots that must be in a sub frame given the slots in a frame. But some slots may be unique to the sub frame. But slot in a frame cannot disappear in a sub frame.
24112 Part-Kind Inheritance

Besides value and slot inheritance there is a third fundamental kind of inheritance viz. part-kind inheritance. It occurs in the interaction between a kind of and part of inheritance. Part-kind inheritance means that there is no need to define every frame in advance for which we want to assert slot values. Some frames must exist because other exist. Part-kind inheritance is often signalled in English by possessives e.g. cars' in the phrase 'cars' wheel.

24113 Procedural Attachment

Inference rules (or inference procedures) can be values in slots. Usually they appear in qualifying slot and represent a way to fill the value in the main slot from other accessible values. This is procedural attachment and the rules are sometimes called if-needed rules. Unlike the rules in Prolog databases these are local; they apply initially just to one slot of one frame. However, rules can inherit just like other slot values so we can often specify a rule useful in many slots in many frames with a single value entry. Extensive and exclusive use of procedural attachment in artificial intelligence has led to a whole new style of programming—object-oriented programming. In Prolog procedures (rules) call on data with object-oriented programming data (in frames) calls on procedures.

An important application of frames is in modelling and construction of stereotypical situations in the world from incomplete knowledge. Empty slots in a frame have expectations about what should fill them from inheritance from qualifying slot information (possible values or permissible values) and from extension statistics. Such expectations can support type checking of user-oriented slot values. They can also support
weaker inference Consider purchasing of equipment for a bureaucratic organization which usually involves many steps and many details. If we know some of the details (slot values) of the purchase, then other details (slot values) are often obvious. For instance, an arriving order was probably ordered six to three weeks ago. Orders from accounting supplies companies are for the accounting department. Orders that come by express mail are probably for the management and should be delivered immediately.

2.4.1.2 Semantic Nets

Semantic network is a method of knowledge representation based on a network structure. A semantic net contains points called nodes connected by links called arcs. The nodes represent objects, concepts, or events. The arcs are used to represent the relations between the nodes. Arcs may be employed for varied purposes. The most common arcs represent the knowledge in hierarchies. These types are usually labelled as is_a or has_part. The arcs may also be used to represent agents or recipient. A typical semantic network comprising nodes and arcs is as shown in the figure below.

![Figure 2.2 Structure of a Semantic Net](image)

Fig 2.2 Structure of a Semantic Net
Semantic nets are a useful way to represent knowledge in domains that use well established taxonomies to simplify problem solving. Semantic nets have been successfully applied in natural language processing to represent complex sentences in English language. The semantic net representation is useful as it provides a standard way of analysing the meaning of different sentences. Also, it points out the similarities in the meaning of closely related sentences but have different structures (8).

For example, consider a sentence *John gave a book to Richard*.

![Agent Recipient and Object Model](image)

**Fig 2.3** Agent Recipient and Object Model

2.4.1.3 **Rule Based Knowledge Representation**

Rule based representation is a popular approach used in expert systems. Rules are employed to state the way in which the inferencing has to be done. Rules provide a formal way of representing recommendations, directives, or strategies (9). Rules are appropriate when the domain knowledge results from empirical associations developed through years of experience in solving problems in a given area. Rules are expressed in the form of IF THEN statements.
For example

1. IF it is cloudy THEN it will rain
2. IF it is an automobile and has four wheels and it is small THEN it is a car

In a rule based expert system, the domain knowledge is represented as a set of rules that are checked against a collection of facts or knowledge about the current situation. When the IF portion of the rule is satisfied by the facts, the action specified by the THEN portion is performed. When the condition is satisfied, the rule is said to fire or execute. A rule interpreter is used to compare the IF portions of rules with the facts and execute the rule whose IF portion matches the facts. The rule's action may modify the set of facts in the knowledge base. For instance, new facts may be added to the knowledge base.

For example:

**Rule**

IF it is an automobile and has four wheels and carries people and small THEN it is a car

**Facts**

- It is an automobile
- It has four wheels
- It carries people
- It is small

**Inference**

It is a car is added to the knowledge in category of automobiles

The new facts added to the knowledge base can themselves be used to form matches with the IF portion of the rules. Thus it contributes to the intuitive learning of expert systems.
The matching of rules IF portions to the facts can produce what are called inference chains. The inference chain is an indicator of how the system reached the ultimate conclusion.

There are two important ways in which rules can be used in rule-based expert systems:

Forward chaining
Backward chaining

**Forward Chaining**

In the forward chaining technique, all the rules are executed in order furnished until the given condition is satisfied. If the goal is to infer one particular fact from among many that are given, forward chaining results in waste of time and effort.

\[
\begin{align*}
  \text{IF} & \quad \text{A and B are true} \\
  \text{THEN} & \quad \text{Conclude C is true}
\end{align*}
\]

- IF A THEN B (Rule 1)
- IF B THEN C (Rule 2)
- A (Exists)(Data)
- Conclusion Therefore C

**Backward Chaining**

In this inferencing method, the system starts with what it wants to prove and only executes the rules that are relevant to establish that fact as truth.
For example

Find Out about C (goal)
If B THEN C (Rule 1)
If A THEN B (Rule 2)
Therefore if A then C (Implicit rule)
Question IS A true? (data)

2.5 Natural Language Processing

One of the long standing goals of computer science is to teach computers to understand the language we speak. Acquiring natural language is so easy for children however it becomes increasingly complex to teach it to computers. We are a long way in achieving the goal. Artificial Intelligence scientists have succeeded in building Natural Language interfaces to a large extent using limited vocabulary and syntax. HAL allows Lotus 1-2-3 spreadsheet users to carry out complex program functions using commands expressed in limited English sentence form. Micro's Clout performs a similar role as an interface to the popular RBASE database program Intellect. Artificial Intelligence Inc is a powerful natural language front end system that has been adopted to several main frame database systems.

Presently research in the applications of NLP techniques in Library and Information Science is mostly centred on developing Natural Language interfaces to bibliographic database systems. As a part of it NLP techniques have also been used in substituting the library professional intermediary in translating the user query into a system query. In addition NLP techniques are also being used to identify the descriptors representing the text which so far is done with statistical techniques. IR NLI (10)
provides technical user with a natural language interface to the information services offered by online databases. The system acts as a front end to several databases and decides which will be most appropriate for answering the user's requests. IR NLI combines the expertise of a professional intermediary for on-line searching with the capability for understanding natural language and carrying out a dialog with the user.

Natural language understanding requires a knowledge of how the words are formed. How the words in turn form clauses and sentences. In addition to successfully understand a set of sentences in a given context, it should have a higher level of knowledge. In general, the knowledge that is to be used in natural language understanding is divided into the following (11):

- **MORPHOLOGICAL**. This deals with the morphological structure of words like the word root, prefix, suffix, and infixes. The basic unit in a written word is a morpheme. Thus, this level gives knowledge of word formation.

- **LEXICAL**. This level deals with thesaurus look up, spell corrections, acronyms, and abbreviations, etc.

- **SYNTAXIC**. Syntax deals with the structure and validity of input sentences. How a right combination of words in a particular sequence constitute a valid sentence.

- **SEMANTICS**. Semantics deal with the meaning of words and that of sentences.
• **PRAGMATICS** Pragmatic level deals with sentences in a particular context. This requires a higher level knowledge which relates to the uses of sentences in different contexts.

• **WORLD KNOWLEDGE** In order to carry out effective communication both the communicator and the communicatee should have a background knowledge either to send or to receive a message without any noise. This background knowledge is considered as the world knowledge of a particular domain.

(Phonological level is excluded as it falls in the area of voice/speech recognition system)

Broadly Chomsky's (12) classification of the types of grammar formalism may be correlated to the above levels. An NLP system having pragmatic and world knowledge constitute his type 0 grammars systems with semantic knowledge or context sensitive grammars constitute his type 1 grammars systems with syntactic knowledge or context free grammars constitute type 2' grammars' and systems with most restrictive or regular grammars constitute 'type 3' grammars.

### 2.5.1 COMPUTATIONAL MODELS

Of the above mentioned grammars Type 2 grammars are well understood from the computational point of view especially the Phrase Structure Grammars (PSGs) have been made use of extensively in developing parsers. Another variant of it called Definite Clause Grammars (DCGs) is the basis for a programming language PROLOG (Programming in Logic) and is one of the most popular language in Artificial Intelligence.
A few of the computational models of parsers are discussed below.

2.5.1.1 Finite State Transition Networks (FSTNs)

The basic assumption of FSTNs is that humans produce sentence one word at a time. Of all the models of representing formal and natural language structures, Finite State Transition Networks (FSTNs) are the simplest and based on the application of directed graphs. An FSTN consists of a number of nodes and labeled arcs. When we traverse a sentence, each node represents a state, and arcs represent rules or test conditions required to make transition from one state to the next state. Thus the total path reflects a valid sequence of words to accept a grammatically correct sentence. The parser proceeds from left to right and imposes constraint rules that limit the choice of words that can be accepted for use at any particular state in the production of a sentence. For example, if the first word is a determiner, the next word should be either an adjective or a noun. In other words, a sentence analysis is viewed as a transition through a series of states.

FSTNs require the incorporation of all valid transition and lexical categories of all words.

![Finite State Transition Network](image)
But it is quite common that in natural language many words belong to more than one part of speech (lexical category). For example, the word information is a noun and also it is an adjective in the compound term information system. The enormous size and complicity of natural languages would require large grammar knowledge base and information about the words and the categories they belong to. For practical application, this poses both the problems of storage and retrieval. The FSTN parsers are useful in dealing with very limited subset of a natural language with limited vocabulary.

Chomsky has pointed out another serious lacuna in the Finite State grammars (FSGs). He has shown that in natural language a sentence can be extended by embedding another structure or sentence. For example, the sentence

\begin{verbatim}
The mouse ran into the hole
\end{verbatim}

\emph{can be extended further as}

\begin{verbatim}
the cat knows the mouse ran into the hole
\end{verbatim}

\emph{can further extended as}

\begin{verbatim}
the cat the dog chased knows the mouse ran into the hole
\end{verbatim}

This limitation of Finite State Grammars is due to the fact that Finite State Grammars (FSGs) are not recursive.

A Prolog program for the implementation of FSTN is presented below. A translation of a very limited set of English grammar is represented in Figure 1. The initial and final states are presented as Prolog facts in the grammar part where each state is represented by a numeral. Each sentence has to start at state numbered \(-1\) and should terminate at state \(-6\). The valid transition from one state to another state as the FSTN recognizer
picks up words in a given sentence is also represented as Prolog facts. For example the facts arc(1 2 det) and arc(1 3 pn) state that at State 1 the recognizer expects either a determiner or a pronoun and if it is a determiner the next state will be 2 and in the case of a pronouns the next state will be 3. The Lexicon part of the program presents English worlds and which part of speech they belong to. The sample sentences that are recognized with the simple grammatical knowledge of this recognizer is presented in list structure containing each word as a member of the list which is an argument to the predicate sentence. The actual FSTN recognizer is presented in the form of Prolog rules having the predicate recognize with two arguments. The first argument presents a sentence or part of it above the recognized argument presents current state.

The rule recognize([], state) serves as the boundary condition i.e. when the recognizer searches the end of a given sentence and when it has no more words to recognize and this is represented by the empty list []). The body of the rule states that the given state should be a final state otherwise it accepts a failure condition.

- **FSTN RECOGNIZER**
- **GRAMMAR**

```prolog
initial(1)
final(6)

arc(1 2 det)
arc(2 3 n)
arc(1 3 pn)
arc(3 4 tv)
arc(4 5 det)
arc(5 6 n)
arc(4 6 pn)
```
Chapter 2 Artificial Intelligence an overview

- LEXICON

  word (the  det)
  word (rama  pn)
  word (sugriva  pn)
  word (helped  tv)
  word (broke  tv)
  word (bow  n)

- SAMPLE SENTENCES

  sentence ([rama helped sugriva])
  sentence ([rama broke the bow])

- RECOGNIZER

  go -
  sentence(Sentence)
  write(Sentence)  nl
  initial(State)
  recognize(Sentence State)

  recognize([] State) -
  final(State)
  write('The sentence is a correct one')
  nl

  recognize([Word|Rest_sentence] State1) -
  word(Word Category)
  arc(State1 State2 Category)
  recognize(Rest_sentence State2)

  recognize(-- -) -
  write('The sentence is a wrong one')
  nl

2 5 1 2 Recursive Transition Networks (RTNs)

Basically RTNs are just like FSTNs except that they have subnetworks. It is quite practical for an arc to point to a subnetwork to be traversed instead of a specific word. That is if we have commonly used bunch of arcs we can express the abstraction by making it into a self-contained subnetwork with specific name. The subnetworks can be referred each time in the network where ever they appear. This naturally allows building
large networks in a modular way. RTNs also allow us to represent efficiently some of the recursive structures in natural language.

Fig 2.5 RTN Main Network

Fig 2.6 RTN Noun Phrase Subnetwork

- RTN RECOGNIZER
- GRAMMAR
- SENTENCE NETWORK
  \begin{align*}
  \text{initial}(s & 1) \\
  \text{final}(s & 4) \\
  \text{arc}(s & 1 2 \ np) \\
  \text{arc}(s & 2 3 \ tv) \\
  \text{arc}(s & 3 4 \ np)
  \end{align*}

- NOUN PHRASE SUBNETWORK
  \begin{align*}
  \text{initial}(np & 1) \\
  \text{final}(np & 3) \\
  \text{arc}(np & 1 2 \ det) \\
  \text{arc}(np & 2 3 \ n) \\
  \text{arc}(np & 1 3 \ pn)
  \end{align*}
Chapter 2 Artificial Intelligence an overview

- LEXICON
  word (the det)
  word (rama pn)
  word (sugriva pn)
  word (helped tv)
  word (broke tv)
  word (bow n)

- SAMPLE SENTENCES
  sentence ([rama helped sugriva])
  sentence ([rama broke the bow])

- RECOGNIZER
  go -
  sentence(Sentence)
  write(Sentence) nl
  initial(s State)
  recognize(s Sentence State)
  go - write( The sent is wrong one ) nl

- MAIN NETWORK RECOGNIZER
  recognize(Net [] State) -
  final(Net State)
  write( The sentence is a correct one ) nl
  recognize(Net [Word|Rest_sent] State1)
  word(Word Category)
  arc(Net Statel State2 Category)
  recognize(Net Rest_sent State2)
  recognize(Net Sent Statel)
  arc(Net Statel State2 Category)
  initial(Category State)
  subnet(Category Sent State Rest_sent)
  recognize(Net Rest_sent State2)

SUBNET RECOGNIZER
  subnet(Subnet Rest_sent State Rest_sent) -
  final(Subnet State)
  subnet(Subnet [Word|Rest_phrase] Statel Rest_sent) -
  word(Word Category)
  arc(Subnet Statel State2 Category)
  subnet(Subnet Rest_phrase State2 Rest-sent)
25131 Augmented Transition Networks

The FSTNs and RTNs are capable of either rejecting or accepting a sentence depending on the syntax of a language. However, a language understanding system should be able to build structures that may be used for further analysis like subject-verb agreement and for analysis of mood, tense, etc. Any RTN which allows additional tests and stores information on the labels can be called as an Augmented Transition Network (ATN).

The main augmentation feature of ATN is its capability to store information in registers so that further tests can be carried out. ATNs provide registers for each of the subjects like NPs, VPs, and PPs. At the end of parsing, the contents of registers are grouped to form a valid sentence structure until then ATNs keep on trying alternative sentence structures.

In addition to the syntactic models discussed so far, there have been attempts to develop semantic models. In this approach, input sentences are transformed through the use of domain-dependent semantic rules which create the target knowledge structure. There are many computational models of semantic grammars like Contextual Dependency Grammar, Modular Logic Grammars, etc.

2514 Definite Clause Grammars

Definite Clause Grammars were developed from Colmerauer's Metamorphosis Grammars (13). DCGs are basically phrase structure grammars (PSGs) annotated with Prolog variables which map directly into Prolog code. Many Prolog interpreters/compilers are distributed with built-in DCG notation. Ability to express DCGs into direct Prolog code is one of the major attractions of Prolog as DCGs can be used to build parsers. DCG formalism is powerful enough to describe both artificial and natural
languages. Here it is necessary to distinguish recognizers from parsers. Recognizers are generally used to find whether a given expression is in conformity with the syntax of a language i.e., the predefined rules for the construction of valid sentences of a language (artificial or natural).

Fig 2.7 Tree structure of DCG

Parsers, on the other hand, not only have the capability of recognizers but in addition, they invoke certain procedures built in syntactic structures so that semantic analysis can be carried out using the output of parsers. A simple English grammar in DCG is presented in the figure above.

- DCG RULES
  - sentence  
  - noun_phrase  
  - verb_phrase  
  - optrel

The right hand side (RHS) comprises of a non terminal symbol whereas the left hand side (LHS) can contain one or more non terminals or
terminals. The general form of the Definite Clauses associated with context free grammar rule is

\[ N \rightarrow V V V \]

In Prolog notation

\[ n0(P0 P) \rightarrow V1(P0 P1) V2(P1 P2) VN(PN-1 P) \]

The notation that is used in Prolog to DCG rules abide the following rules:

- Predicate and function symbols, variables and constants obey normal Prolog syntax.
- Adjacent symbols in the right hand side of a DCG rules are separated by '' operator just like literal in a clause.
- The arrow in DCG rules is ->
- Terminal symbols are written inside Prolog list brackets [ and ]
- The empty string is represented by the empty list constant []

The DCG (Prolog Code) recognizer for the rules of syntax expressed in the following program:

```
\% DCG Recognizer
s \rightarrow np vp
np \rightarrow det n optrel
np \rightarrow pn
vp \rightarrow tv np
vp \rightarrow iv
optrel \rightarrow []
optrel \rightarrow [that] vp

pn \rightarrow [rama]
pn \rightarrow [ravana]
v \rightarrow [ran]
det \rightarrow [a]
n \rightarrow [program]
tv \rightarrow [killed]
tv \rightarrow [helped]
go -
s([rama killed ravana] []))
```
Parser

However the above code serves only as recognizer To transform it into a parser it should have features to build syntactic structure for valid sentences Prolog code for the DCG parser is given below

DCG PARSER

this program not only recognizes correct sentences but also builds tree structures

\[
s(s(NP VP)) \rightarrow np(NP) \ vp(VP) \\
np(np(Det N Rel)) \rightarrow det(Det) n(N) \ Optrel(Rel) \\
np(np(PN)) \rightarrow pn(PN) \\
v(\text{TV} NP) \rightarrow tv(TV) \ np(NP) \\
v(\text{IV}) \rightarrow iv(IV) \\
optrel(rel(\text{epsilon})) \rightarrow [] \\
optrel(rel(\text{that} VP)) \rightarrow [\text{that}] \ vp(VP) \\
\]

\[
\begin{align*}
\text{pn(np(rama))} & \rightarrow [\text{rama}] \\
\text{pn(np(sugriva))} & \rightarrow [\text{sugriva}] \\
\text{iv(iv(\text{ran}))} & \rightarrow [\text{ran}] \\
\text{det(det(a))} & \rightarrow [\text{a}] \\
\text{n(n(program))} & \rightarrow [\text{arrow}] \\
\text{tv(tv(helped))} & \rightarrow [\text{helped}] \\
\end{align*}
\]

The syntactic structure produced by the above program for the sample sentence looks like,

\[
s(nc(np(np(rama)) \ vp(tv(helped))) \\
\]

In addition to the syntactic models discussed so far there have been attempts in developing semantic models In this approach input sentences are transformed through the use of domain dependent semantic rewrite rules which create the target knowledge structure There are many
computational models is semantic grammars like Contextual Dependency Grammar, Modular Logic Grammars, Lexical Functional Grammars, Case Grammars, etc.

NLP offers a potential if not immediately viable alternative to pure statistical techniques in Information Retrieval. Even if the current usable language processing techniques appear inadequate for full utilization under operational retrieval conditions, there is always hope that new developments may render the linguistic techniques more attractive in the future (10).

In the area of natural language front end systems to data bases and application software, there have been successful and commercially available systems. Lotus Corporation's HAL allows users to interact with Lotus 1-2-3 using restricted number of sentence structures.

AI is one area with very promising applications in the area of Library and Information Science. Much of the work done is in the area of Information processing and retrieval. In the process of acquisition and collection development processes, there are several stages which involve decision making. The decision-making process is well supported by various models of work flow that make the process methodical. In the present work, an item-by-item evaluation and approval system has been adopted in the selection process. Again, there are several parameters within this system. Since such work is involved, it is amenable to the application of Artificial Intelligence techniques, especially in the area of Knowledge Based systems. The objective of the present work is to build the knowledge necessary for the item-by-item selection and the various other parameters involved and also the inferencing into the system so that it behaves as the expert selector.

\[ t = \frac{2}{P} \]
2.6 References


2. WINSTON (Patrick Henry) Artificial Intelligence 2nd ed Addison Wesley London 1984

3. PATTERSON (Dan W) Introduction to artificial intelligence and expert systems Prentice Hall New Delhi 1992


5. SCHANK (R) The cognitive computer on language learning and artificial intelligence Addison Wesley Massachusetts 1984


8. WATERMAN (D A) A guide to Expert Systems Addison Wesley Massachusetts 1986
9 WATERMAN (D A) and HAYES ROTH (F) Patten directed inference systems Academic Press New York 1978

10 BRAJNIK (G) GUIDA (G) and TASSO (C) An expert interface for effective man machine interaction (In Cooperative Interfaces to Information Systems L Bolc M Jarke [eds]) Springer Verlag Berlin FRG 1986 pp 259 308

11 GAZDAR (G) and MELLISH (C) Natural Language Processing in Prolog An introduction to Computational Linguistics Addison Wesley New York 1989

12 CHOMSKY (N) Aspects of the theory of syntax MIT Press Massachusetts 1965

13 PEREIRA (F C N) and WARREN (D H D) Definite Clause Grammars for Natural Language Analysis Artificial Intelligence 13 1980 pp 231 278