DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Artificial Intelligence
Fourth Year CSE( Sem:I)
2 marks Questions and Answers

UNIT I

1) What is AI?

Systems that think like humans

Systems that think rationally

Systems that act like humans

Systems that act rationally

2) Define an agent.

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.

3) What is an agent function?

An agent’s behavior is described by the agent function that maps any given percept sequence to an action.

4) Differentiate an agent function and an agent program.

<table>
<thead>
<tr>
<th>Agent Function</th>
<th>Agent Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>An abstract mathematical description</td>
<td>A concrete implementation, running on the agent Architecture.</td>
</tr>
</tbody>
</table>
5) **What can AI do today?**
- Autonomous Planning and Scheduling
  - Spacecraft control
  - Goal-directed planning, detection, diagnosis, problem recovery
- Game Planning
  - IBM Deep Blue
  - World chess champion
- Autonomous Control
  - CMU NAVLAB
  - Computer-controlled mini-van
  - Crossed the US without human control over 98% of the time
- Diagnosis
  - Medical diagnosis in several areas of medicine (e.g., pathology)
  - Explanation, justification for decisions
- Logistics Planning
  - DOD's Dynamic Analysis and Replanning Tool
  - Logistics planning of 50,000 vehicles, cargo, people
  - Embarkation, destination, route, conflict resolution
  - Paid back all of DARPA's 30-year investment in AI
- Robotics
  - Robotic surgical assistants
  - Cooperating autonomous robots in reconnaissance
  - Exploration of the Solar System

6) **What is a task environment? How it is specified?**

**Task environments** are essentially the "problems" to which rational agents are the "solutions."

A Task environment is specified using PEAS (Performance, Environment, Actuators, Sensors) description.

7) **Give an example of PEAS description for an automated taxi.**

<table>
<thead>
<tr>
<th>Agent Type</th>
<th>Performance Measure</th>
<th>Environment</th>
<th>Actuators</th>
<th>Sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taxi driver</td>
<td>Safe, fast, legal, comfortable trip, maximize profits</td>
<td>Roads, other traffic, pedestrians, customers</td>
<td>Steering, accelerator, brake, signal, horn, display</td>
<td>Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard</td>
</tr>
</tbody>
</table>

**Figure 2.4** PEAS description of the task environment for an automated taxi.
8) Give PEAS description for different agent types.

<table>
<thead>
<tr>
<th>Agent Type</th>
<th>Performance Measure</th>
<th>Environment</th>
<th>Actuators</th>
<th>Sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medical diagnosis system</td>
<td>Healthy patient, minimize costs,</td>
<td>Patient, hospital, staff</td>
<td>Display questions, tests, diagnoses, treatments, referrals</td>
<td>Keyboard entry of symptoms, findings, patient's answers</td>
</tr>
<tr>
<td></td>
<td>minimize costs, lawsuits</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Satellite image analysis system</td>
<td>Correct image categorization</td>
<td>Downlink from orbiting satellite</td>
<td>Display categorization of scene</td>
<td>Color pixel arrays</td>
</tr>
<tr>
<td>Part-picking robot</td>
<td>Percentage of parts in correct bins</td>
<td>Conveyor belt with parts; bins</td>
<td>Jointed arm and hand</td>
<td>Camera, joint angle sensors</td>
</tr>
<tr>
<td>Refinery controller</td>
<td>Maximize purity, yield, safety</td>
<td>Refinery, operators</td>
<td>Valves, pumps, heaters, displays</td>
<td>Temperature, pressure, chemical sensors</td>
</tr>
<tr>
<td>Interactive English tutor</td>
<td>Maximize student's score on test</td>
<td>Set of students, testing agency</td>
<td>Display exercises, suggestions, corrections</td>
<td>Keyboard entry</td>
</tr>
</tbody>
</table>

**Figure 2.5** Examples of agent types and their PEAS descriptions.

9) List the properties of task environments.
   - Fully observable vs. partially observable.
   - Deterministic vs. stochastic.
   - Episodic vs. sequential
   - Static vs. dynamic.
   - Discrete vs. continuous.
   - Single agent vs. multiagent.

10) Write a function for the table driven agent.
11) What are the four different kinds of agent programs?
   Simple reflex agents;
   Model-based reflex agents;
   Goal-based agents; and
   Utility-based agents.

12) Explain a simple reflex agent with a diagram.

Simple reflex agents
   The simplest kind of agent is the simple reflex agent. These agents select actions on the basis of the current percept, ignoring the rest of the percept history.
13) Explain with a diagram the model based reflex agent.
13a) Explain with a diagram the goal based reflex agent.

Knowing about the current state of the environment is not always enough to decide what to do. For example, at a road junction, the taxi can turn left, turn right, or go straight on. The correct decision depends on where the taxi is trying to get to. In other words, as well as a current state description, the agent needs some sort of goal information that describes
situations that are desirable—for example, being at the passenger’s destination.

13b) **What are utility based agents?**

Goals alone are not really enough to generate high-quality behavior in most environments. For example, there are many action sequences that will get the taxi to its destination (thereby achieving the goal) but some are quicker, safer, more reliable, or cheaper than others. A **utility function** maps a state (or a sequence of states) onto a real number, which describes the associated degree of happiness.

13c) **What are learning agents?**

A learning agent can be divided into four conceptual components, as shown in Fig. 2.15. The most important distinction is between the learning element, which is responsible for making improvements, and the performance element, which is responsible for selecting external actions. The performance element is what we have previously considered to be the entire agent: it takes in percepts and decides on actions. The learning element uses **critic** feedback from the critic on how the agent is doing and determines how the performance element should be modified to do better in the future.

![Figure 2.15](image-url)  
**Figure 2.15** A general model of learning agents.
Searching Techniques

14) Define the problem solving agent.
A Problem solving agent is a **goal-based** agent. It decide what to do by finding sequence of actions that lead to desirable states. The agent can adopt a goal and aim at satisfying it.
Goal formulation is the first step in problem solving.

15) Define the terms goal formulation and problem formulation.
**Goal formulation**, based on the current situation and the agent’s performance measure, is the first step in problem solving.
The agent’s task is to find out which sequence of actions will get to a goal state.
**Problem formulation** is the process of deciding what actions and states to consider given a goal.

16) List the steps involved in simple problem solving agent.
(i) Goal formulation
(ii) Problem formulation
(iii) Search
(iv) Search Algorithm
(v) Execution phase

17) Define search and search algorithm.
The process of looking for sequences actions from the current state to reach the goal state is called **search**.
The **search algorithm** takes a **problem** as input and returns a **solution** in the form of **action sequence**. Once a solution is found, the **execution phase** consists of carrying out the recommended action.

18) What are the components of well-defined problems?
- The **initial state** that the agent starts in. The initial state for our agent of example problem is described by $In(Arad)$
- A **Successor Function** returns the possible actions available to the agent. Given a state $x$, SUCCESSOR-FN($x$) returns a set of {action,successor} ordered pairs where each action is one of the legal actions in state $x$, and each successor is a state that can be reached from $x$ by applying the action.
  
  For example, from the state $In(Arad)$, the successor function for the Romania problem would return
  
  $\{ [Go(Sibiu), In(Sibiu)], [Go(Timisoara), In(Timisoara)], [Go(Zerind), In(Zerind)] \}$
- A **goal test** determines whether the given state is a goal state.
- A **path cost** function assigns numeric cost to each action. For the Romania problem the cost of path might be its length in kilometers.
19) Differentiate toy problems and real world problems.

<table>
<thead>
<tr>
<th>TOY PROBLEMS</th>
<th>REAL WORLD PROBLEMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A toy problem is intended to illustrate various problem solving methods. It can be easily used by different researchers to compare the performance of algorithms.</td>
<td>A real world problem is one whose solutions people actually care about.</td>
</tr>
</tbody>
</table>

20) Give examples of real world problems.

(i) Touring problems  
(ii) Travelling Salesperson Problem (TSP)  
(iii) VLSI layout  
(iv) Robot navigation  
(v) Automatic assembly sequencing  
(vi) Internet searching

21) List the criteria to measure the performance of different search strategies.

- Completeness: Is the algorithm guaranteed to find a solution when there is one?
- Optimality: Does the strategy find the optimal solution?
- Time complexity: How long does it take to find a solution?
- Space complexity: How much memory is needed to perform the search?

22) Differentiate Uninformed Search (Blind search) and Informed Search (Heuristic Search) strategies.

<table>
<thead>
<tr>
<th>Uninformed or Blind Search</th>
<th>Informed or Heuristic Search</th>
</tr>
</thead>
</table>
| o No additional information beyond that provided in the problem definition  
  o Not effective  
  o No information about number of steps or path cost | o More effective  
  o Uses problem-specific knowledge beyond the definition of the problem itself. |

23) Define Best-first-search.

Best-first search is an instance of the general TREE-SEARCH or GRAPH-SEARCH algorithm in which a node is selected for expansion based on the evaluation function $f(n)$. Traditionally, the node with the lowest evaluation function is selected for expansion.
(1) How Knowledge is represented?
A variety of ways of knowledge (facts) have been exploited in AI programs. Facts: truths in some relevant world. These are things we want to represent.

(2) What is propositional logic?
It is a way of representing knowledge. In logic and mathematics, a propositional calculus or logic is a formal system in which formulae representing propositions can be formed by combining atomic propositions using logical connectives. Sentences considered in propositional logic are not arbitrary sentences but are the ones that are either true or false, but not both. This kind of sentences are called propositions.

Example
Some facts in propositional logic:
It is raining. - RAINING
It is sunny - SUNNY
It is windy - WINDY

If it is raining, then it is not sunny - RAINING -> ¬ SUNNY

(3) What are the elements of propositional logic?
Simple sentences which are true or false are basic propositions. Larger and more complex sentences are constructed from basic propositions by combining them with connectives. Thus propositions and connectives are the basic elements of propositional logic. Though there are many connectives, we are going to use the following five basic connectives here:

NOT, AND, OR, IF_THEN (or IMPLY), IF_AND_ONLY_IF.
They are also denoted by the symbols:
¬, ∧, ∨, →, ↔, respectively.

(4) What is inference?
Inference is deriving new sentences from old.

(5) What are modus ponens?
There are standard patterns of inference that can be applied to derive chains of conclusions that lead to the desired goal. These patterns of inference are called inference rules. The best-known rule is called Modus Ponens and is written as follows:

\[ \alpha \Rightarrow \beta, \quad \alpha \]

\[ \beta \]

The notation means that, whenever any sentences of the form \( \alpha \Rightarrow \beta \) and \( \alpha \) are given, then the sentence \( \beta \) can be inferred. For example, if (WumpusAhead A WumpusAlive) \( \Rightarrow \) Shoot and (WumpusAhead A WumpusAlive) are given, then Shoot can be inferred.

Another useful inference rule is And-Elimination, which says that, from a conjunction, any of the conjuncts can be inferred:

\[ \alpha \land \beta \]

\[ \alpha \]

For example, from (WumpusAhead A WumpusAlive), WumpusAlive can be inferred.
(6) What is entailment?
Propositions tell about the notion of truth and it can be applied to logical reasoning. We can have logical entailment between sentences. This is known as entailment where a sentence follows logically from another sentence. In mathematical notation we write:

\[ \alpha \vdash \beta \]

(7) What are knowledge based agents?
The central component of a knowledge-based agent is its knowledge base, or KB. Informally, a knowledge base is a set of sentences. Each sentence is expressed in a language called a knowledge representation language and represents some assertion about the world.

```
function KB-AGENT( percept ) returns an action
    static: KB, a knowledge base
    t, a counter, initially 0, indicating time
    TELL( KB, MAKE-PERCEPT-SENTENCE( percept, t ) )
    action ← ASK( KB, MAKE-ACTION-QUERY( ) )
    TELL( KB, MAKE-ACTION-SENTENCE( action, t ) )
    t ← t + 1
    return action
```

Figure 7.1 A generic knowledge-based agent.

Figure 7.1 shows the outline of a knowledge-based agent program. Like all our agents, it takes a percept as input and returns an action. The agent maintains a knowledge base, KB, which may initially contain some background knowledge. Each time the agent program is called, it does three things. First, it TELLS the knowledge base what it perceives. Second, it ASKS the knowledge base what action it should perform. In the process of answering this query, extensive reasoning may be done about the current state of the world, about the outcomes of possible action sequences, and so on.

(8) Explain in detail the connectives used in propositional logic.
The syntax of propositional logic defines the allowable sentences. The atomic sentences—the indivisible syntactic elements—consist of a single proposition symbol. Each such symbol stands for a proposition that can be true or false. We will use uppercase names for symbols: P, Q, R, and so on.

Complex sentences are constructed from simpler sentences using logical connectives. There are five connectives in common use:

¬ (not). A sentence such as ¬W₁,₃ is called the negation of W₁,₃. A literal is either an atomic sentence (a positive literal) or a negated atomic sentence (a negative literal).
A (and). A sentence whose main connective is A, such as $W_{1,3} \land P_{3,1}$, is called a conjunction; its parts are the conjuncts. (The A looks like an "A" for "And."

\( \lor \) (or). A sentence using \( \lor \), such as $(W_{1,3} \land P_{3,1}) \lor W_{2,2}$, is a disjunction of the disjuncts $(W_{1,3} \land P_{3,1})$ and $W_{2,2}$. (Historically, the \( \lor \) comes from the Latin “vel,” which means "or." For most people, it is easier to remember as an upside-down \( \land \).

\( \rightarrow \) (implies). A sentence such as $(W_{1,3} \land P_{3,1}) \rightarrow W_{2,2}$ is called an implication (or conditional). Its premise or antecedent is $(W_{1,3} \land P_{3,1})$, and its conclusion or consequent is $W_{2,2}$. Implications are also known as rules or if–then statements. The implication symbol is sometimes written in other books as \( \supset \) or \( \rightarrow \).

\( \leftrightarrow \) (if and only if). The sentence $W_{1,3} \leftrightarrow W_{2,2}$ is a biconditional.

Figure 7.7 gives a formal grammar of propositional logic:

<table>
<thead>
<tr>
<th>Sentence</th>
<th>→</th>
<th>AtomicSentence</th>
<th>ComplexSentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>AtomicSentence</td>
<td>→</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>Symbol</td>
<td>→</td>
<td>P</td>
<td>Q</td>
</tr>
<tr>
<td>ComplexSentence</td>
<td>→</td>
<td>¬ Sentence</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( Sentence ∧ Sentence )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( Sentence V Sentence )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( Sentence ⇒ Sentence )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( Sentence ⇔ Sentence )</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 7.7** A BNF (Backus–Naur Form) grammar of sentences in propositional logic.

(9) **Define First order Logic?**

Whereas propositional logic assumes the world contains facts, first-order logic (like natural language) assumes the world contains

- Objects: people, houses, numbers, colors, baseball games, wars, …
- Relations: red, round, prime, brother of, bigger than, part of, comes between, …
- Functions: father of, best friend, one more than, plus, …

(10) **Specify the syntax of First-order logic in BNF form.**
(11) Compare different knowledge representation languages.

<table>
<thead>
<tr>
<th>Language</th>
<th>Ontological Commitment (What exists in the world)</th>
<th>Epistemological Commitment (What an agent believes about facts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propositional logic</td>
<td>facts</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>First-order logic</td>
<td>facts, objects, relations</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Temporal logic</td>
<td>facts, objects, relations, times</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Probability theory</td>
<td>facts</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Fuzzy logic</td>
<td>facts with degree of truth ( \in [0, 1] )</td>
<td>known interval value</td>
</tr>
</tbody>
</table>

(12) What are the syntactic elements of First Order Logic?

The basic syntactic elements of first-order logic are the symbols that stand for objects,
relations, and functions. The symbols come in three kinds:

a) constant symbols, which stand for objects;
b) predicate symbols, which stand for relations;
c) and function symbols, which stand for functions.

We adopt the convention that these symbols will begin with uppercase letters.

Example:

Constant symbols:
Richard and John;

Predicate symbols:
Brother, OnHead, Person, King, and Crown;

Function symbol:
LeftLeg.

(13) What are quantifiers?
There is need to express properties of entire collections of objects, instead of enumerating the objects by name. Quantifiers let us do this.

FOL contains two standard quantifiers called
a) Universal (\( \forall \)) and
b) Existential (\( \exists \))

Universal quantification
\( (\forall x) \ P(x) \) : means that \( P \) holds for all values of \( x \) in the domain associated with that variable
E.g., \( (\forall x) \ \text{dolphin}(x) \Rightarrow \text{mammal}(x) \)

Existential quantification
\( (\exists x) P(x) \) means that \( P \) holds for some value of \( x \) in the domain associated with that variable
E.g., \( (\exists x) \ \text{mammal}(x) \land \text{lays-eggs}(x) \)

Permits one to make a statement about some object without naming it.

(14) Explain Universal Quantifiers with an example.

Rules such as "All kings are persons," is written in first-order logic as
\( \forall x \ \text{King}(x) \Rightarrow \text{Person}(x) \)
where \( \forall \) is pronounced as “For all ..”

Thus, the sentence says, "For all \( x \), if \( x \) is a king, then \( x \) is a person."
The symbol \( x \) is called a variable(lower case letters)
The sentence \( \forall x \ P \), where \( P \) is a logical expression says that \( P \) is true for every object \( x \).

(15) Explain Existential quantifiers with an example.

Universal quantification makes statements about every object.
It is possible to make a statement about some object in the universe without naming it, by using an existential quantifier.
Example
"King John has a crown on his head"
\( \exists x \text{ Crown}(x) \land \text{OnHead}(x, \text{John}) \)

\( \exists x \) is pronounced “There exists an x such that ..” or “For some x ..”

(16) What are nested quantifiers?

**Nested quantifiers**

We will often want to express more complex sentences using multiple quantifiers. The simplest case is where the quantifiers are of the same type. For example, "Brothers are siblings" can be written as

\[ \forall x \forall y \quad \text{Brother}(x, y) \Rightarrow \text{Sibling}(x, y). \]

Example-2

“Everybody loves somebody” means that for every person, there is someone that person loves

\[ \forall x \exists y \quad \text{Loves}(x, y) \]

(17) **Explain the connection between \( \forall \) and \( \exists \)**

“Everyone likes ice cream “ is equivalent “there is no one who does not like ice cream”

This can be expressed as:

\[ \forall x \quad \text{Likes}(x, \text{IceCream}) \text{ is equivalent to } \neg \exists \neg \text{Likes}(x, \text{IceCream}) \]

(18) What are the steps associated with the knowledge Engineering process?

Discuss them by applying the steps to any real world application of your choice.

**Knowledge Engineering**

The general process of knowledge base constructiona process is called knowledge engineering.

A knowledge engineer is someone who investigates a particular domain, learns what concepts are important in that domain, and creates a formal representation of the objects and relations in the domain. We will illustrate the knowledge engineering process in an electronic circuit domain that should already be fairly familiar.

The steps associated with the knowledge engineering process are:

1. **Identify the task.**
   . The task will determine what knowledge must be represented in order to connect problem instances to answers. This step is analogous to the PEAS process for designing agents.

2. **Assemble the relevant knowledge.**
   . The knowledge engineer might already be an expert in the domain, or might need to work with real experts to extract what they know—a process called knowledge acquisition.

3. **Decide on a vocabulary of predicates, functions, and constants.**
   . That is, translate the important domain-level concepts into logic-level names.

Once the choices have been made, the result is a vocabulary that is known as the **ontology** of the domain. The word **ontology** means a particular theory of the nature of being or existence.
4. **Encode general knowledge about the domain.** The knowledge engineer writes down the axioms for all the vocabulary terms. This pins down (to the extent possible) the meaning of the terms, enabling the expert to check the content. Often, this step reveals misconceptions or gaps in the vocabulary that must be fixed by returning to step 3 and iterating through the process.

5. **Encode a description of the specific problem instance.**
   For a logical agent, problem instances are supplied by the sensors, whereas a "disembodied" knowledge base is supplied with additional sentences in the same way that traditional programs are supplied with input data.

6. **Pose queries to the inference procedure and get answers.** This is where the reward is: we can let the inference procedure operate on the axioms and problem-specific facts to derive the facts we are interested in knowing.

7. **Debug the knowledge base.**
   \[ \forall x \text{ NumOfLegs}(x, 4) \implies \text{Mammal}(x) \]
   *Is false for reptiles, amphibians.*

---

**What is universal instantiation?**

**Inference rules for quantifiers**

Let us begin with universal quantifiers. Suppose our knowledge base contains the standard folkloric axiom stating that all greedy kings are evil:

\[ \forall x \text{ King}(x) \land \text{Greedy}(x) \implies \text{Evil}(x) \]

Then it seems quite permissible to infer any of the following sentences:

\[ \text{King}(\text{John}) \land \text{Greedy}(\text{John}) \implies \text{Evil}(\text{John}). \]
\[ \text{King}(\text{Richard}) \land \text{Greedy}(\text{Richard}) \implies \text{Evil}(\text{Richard}). \]
\[ \text{King}(\text{Father}(\text{John})) \land \text{Greedy}(\text{Father}(\text{John})) \implies \text{Evil}(\text{Father}(\text{John})). \]

The rule of **Universal Instantiation** (UI for short) says that we can infer any sentence obtained by substituting a **ground term** (a term without variables) for the variable.\(^1\) To write out the inference rule formally, we use the notion of **substitutions** introduced in Section 8.3. Let \(\text{SubST} (\theta, a)\) denote the result of applying the substitution \(\theta\) to the sentence \(a\). Then the rule is written

\[ \forall v \ a \quad \text{SubST}(\{v/g\}, a) \]

for any variable \(v\) and ground term \(g\). For example, the three sentences given earlier are obtained with the substitutions \(\{x/\text{John}\}\), \(\{x/\text{Richard}\}\), and \(\{x/\text{Father(John)}\}\).

The corresponding **Existential Instantiation** rule for the existential quantifier is slightly more complicated. For any sentence \(a\), variable \(v\), and constant symbol \(k\) that does not appear elsewhere in the knowledge base,

\[ \exists v \ a \quad \text{SubST}(\{v/k\}, a) \]

Universal instantiation (UI)
• Every instantiation of a universally quantified sentence is entailed by it:
\[ \forall v \alpha \]
\[ \text{Subst}(\{v/g\}, \alpha) \]
for any variable \( v \) and ground term \( g \)

• E.g., \( \forall x \ King(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x) \) yields:
  
  \( \text{King}(\text{John}) \land \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John}) \)
  
  \( \text{King}(\text{Richard}) \land \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard}) \)
  
  \( \text{King}(\text{Father}(\text{John})) \land \text{Greedy}(\text{Father}(\text{John})) \Rightarrow \text{Evil}(\text{Father}(\text{John})) \)

Existential instantiation (EI)

• For any sentence \( \alpha \), variable \( v \), and constant symbol \( k \) that does not appear elsewhere in the knowledge base:
\[ \exists v \alpha \]
\[ \text{Subst}(\{v/k\}, \alpha) \]

• E.g., \( \exists x \ Crown(x) \land \text{OnHead}(x, \text{John}) \) yields:
\[ \text{Crown}(\text{C}_1) \land \text{OnHead}(\text{C}_1, \text{John}) \]
provided \( \text{C}_1 \) is a new constant symbol, called a Skolem constant

UNIT V

(20) Define communication.

Communication is the intentional exchange of information brought about by the production and perception of signs drawn from a shared system of conventional signs. Most animals use signs to represent important messages: food here, predator nearby etc. In a partially observable world, communication can help agents be successful because they can learn information that is observed or inferred by others.

(21) What is speech act?
What sets humans apart from other animals is the complex system of structured messages known as language that enables us to communicate most of what we know about the world. This is known as speech act.

Speaker, hearer, and utterance are generic terms referring to any mode of communication.

The term word is used to refer to any kind of conventional communicative sign.

(22) What are the capabilities gained by an agent from speech act?.

a. **Query** other agents about particular aspects of the world. This is typically done by asking questions: *Have you smelled the wumpus anywhere?*

b. **Inform** each other about the world. This is done by making representative statements:
*There’s a breeze here in 3 4.* Answering a question is another form of informing.

c. **Request** other agents to perform actions: *Please help me carry the gold.*

Sometimes
indirect speech act (a request in the form of a statement or question) is considered more polite: I could use some help carrying this. An agent with authority can give commands (Alpha go right; Bravo and Charlie go left), and an agent with power can make a threat (Give me the gold, or else). Together, these kinds of speech acts are called directives.

d. Acknowledge requests: OK.

e. Promise or commit to a plan: I'll shoot the wumpus; you grab the gold.

Define formal language.
A formal language is defined as a (possibly infinite) set of strings. Each string is a concatenation of terminal symbols, sometimes called words. For example, in the language of first-order logic, the terminal symbols include A and P, and a typical string is "P A Q.". Formal languages such as first-order logic and Java have strict mathematical definitions. This is in contrast to natural languages, such as Chinese, Danish, and English, that have no strict definition but are used by a community.

Define a grammar.
A grammar is a finite set of rules that specifies a language. Formal languages always have an official grammar, specified in manuals or books. Natural languages have no official grammar, but linguists strive to discover properties of the language by a process of scientific inquiry and then to codify their discoveries in a grammar.

What are the component steps of communication? Explain with an example.
The component steps of communication
A typical communication episode, in which speaker S wants to inform hearer H about proposition P using words W, is composed of seven processes:

1) Intention. Somehow, speaker S decides that there is some proposition P that is worth saying to hearer H. For our example, the speaker has the intention of having the hearer know that the wumpus is no longer alive.

2) Generation. The speaker plans how to turn the proposition P into an utterance that makes it likely that the hearer, upon perceiving the utterance in the current situation, can infer the meaning P (or something close to it). Assume that the speaker is able to come up with
the words "The wumpus is dead," and call this $W$.

3) **Synthesis.** The speaker produces the physical realization $W'$ of the words $W$. This can be via ink on paper, vibrations in air, or some other medium. In Figure 22.1, we show the agent synthesizing a string of sounds $W'$ written in the phonetic alphabet defined on page 569: "[thaxwahmpaxsihzdehd]." The words are run together; this is typical of quickly spoken speech.

4) **Perception.** $H$ perceives the physical realization $W'$ as $W_i$ and decodes it as the words $W_2$. When the medium is speech, the perception step is called **speech recognition**; when it is printing, it is called **optical character recognition**.

5) **Analysis.** $H$ infers that $W_2$ has possible meanings $P_1, \ldots, P_n$. We divide analysis into three main parts:
   a) **syntactic interpretation** (or parsing),
   b) **Semantic interpretation**, and
   c) **Pragmatic interpretation**.

   Parsing is the process of building a **parse tree** for an input string, as shown in Figure 22.1. The interior nodes of the parse tree represent phrases and the leaf nodes represent words.

   Semantic interpretation is the process of extracting the meaning of an utterance as an expression in some representation language. Figure 22.1 shows two possible semantic interpretations: that the wumpus is not alive and that it is tired (a colloquial meaning of dead). Utterances with several possible interpretations are said to be **ambiguous**.

   Pragmatic interpretation takes into account the fact that the same words can have different meanings in different situations.

6) **Disambiguation.** $H$ infers that $S$ intended to convey $P$, (where ideally $P, = P$). Most speakers are not intentionally ambiguous, but most utterances have several feasible interpretations. Analysis generates possible interpretations; if more than one interpretation is found, then disambiguation chooses the one that is best.

7) **Incorporation.** $H$ decides to believe $P$, (or not). A totally naive agent might believe everything it hears, but a sophisticated agent treats the speech act as evidence for $P$, not confirmation of it.

Putting it all together, we get the agent program shown in Figure 22.2. Here the agent acts as a robot slave that can be commanded by a master. On each turn, the slave will answer
Define a Lexicon and grammar for language consisting of a small fragment of English.

The Lexicon of $E_0$

First we define the lexicon, or list of allowable words. The words are grouped into the categories or parts of speech familiar to dictionary users: nouns, pronouns, and names to denote things, verbs to denote events, adjectives to modify nouns, and adverbs to modify verbs. Categories that are perhaps less familiar to some readers are articles (such as the), prepositions (in), and conjunctions (and). Figure 22.3 shows a small lexicon.
The Grammar of $E_D$

The next step is to combine the words into phrases. We will use five nonterminal symbols to define the different kinds of phrases: sentence ($S$), noun phrase (NP), verb phrase (VP), prepositional phrase (PP), and relative clause (cl). Figure 22.4 shows a grammar for $E_D$, with an example for each rewrite rule. $E_D$ generates good English sentences such as the following:

John is in the pit
The wumpus that stinks is in 2 2
What is parsing? Explain the top down parsing method.

Parsing is defined as the process of finding a parse tree for a given input string. That is, a call to the parsing function PARSE, such as

```
PARSE("the wumpus is dead", E_0, S)
```

should return a parse tree with root S whose leaves are "the wumpus is dead" and whose internal nodes are nonterminal symbols from the grammar E_0.

*Parsing can be seen as a process of searching for a parse tree.*

There are two extreme ways of specifying the search space (and many variants in between).

First, we can start with the S symbol and search for a tree that has the words as its leaves. This is called **top-down parsing**.

Second, we could start with the words and search for a tree with root S. This is called **bottom-up parsing**.

Top-down parsing can be precisely defined as a search problem as follows:

- The initial state is a parse tree consisting of the root S and unknown children: [S: ?I].

In general, each state in the search space is a parse tree.

The successor function selects the leftmost node in the tree with unknown children. It then looks in the grammar for rules that have the root label of the node on the left-hand side. For each such rule, it creates a successor state where the ? is replaced by a list corresponding to the right-hand side of the rule.
Formulate the bottom-up parsing as a search problem.

The formulation of bottom-up parsing as a search is as follows:

The initial state is a list of the words in the input string, each viewed as a parse tree that is just a single leaf node—for example; [the, wumpus, is, dead]. In general, each state in the search space is a list of parse trees.

The successor function looks at every position $i$ in the list of trees and at every right-hand side of a rule in the grammar. If the subsequence of the list of trees starting at $i$ matches the right-hand side, then the subsequence is replaced by a new tree whose category is the left-hand side of the rule and whose children are the subsequence. By "matches," we mean that the category of the node is the same as the element in the right-hand side. For example, the rule Article + the matches the subsequence consisting of the first node in the list [the, wumpus, is, dead], so a successor state would be [[Article: the], wumpus, is, dead].

The goal test checks for a state consisting of a single tree with root $S$.

See Figure 22.5 for an example of bottom-up parsing.

<table>
<thead>
<tr>
<th>step</th>
<th>list of nodes</th>
<th>subsequence</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>the wumpus is dead</td>
<td>the</td>
<td>Article $\rightarrow$ the</td>
</tr>
<tr>
<td>2</td>
<td>Article wumpus is dead</td>
<td>wumpus</td>
<td>Noun $\rightarrow$ wumpus</td>
</tr>
<tr>
<td>3</td>
<td>Article Noun is dead</td>
<td>Article Noun</td>
<td>NP $\rightarrow$ Article Noun</td>
</tr>
<tr>
<td>4</td>
<td>NP is dead</td>
<td>is</td>
<td>Verb $\rightarrow$ is</td>
</tr>
<tr>
<td>5</td>
<td>NP Verb dead</td>
<td>dead</td>
<td>Adjective $\rightarrow$ dead</td>
</tr>
<tr>
<td>6</td>
<td>NP Verb Adjective</td>
<td>Verb</td>
<td>VP $\rightarrow$ Verb</td>
</tr>
<tr>
<td>7</td>
<td>NP VP Adjective</td>
<td>VP Adjective</td>
<td>VP $\rightarrow$ VP Adjective</td>
</tr>
<tr>
<td>8</td>
<td>NP VP</td>
<td>NP VP</td>
<td>S $\rightarrow$ NP VP</td>
</tr>
<tr>
<td>GOAL</td>
<td>S</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 22.5 Trace of a bottom-up parse on the string "The wumpus is dead." We start with a list of nodes consisting of words. Then we replace subsequences that match the right-hand side of a rule with a new node whose root is the left-hand side. For example, in the third line the Article and Noun nodes are replaced by an NP node that has those two nodes as children. The top-down parse would produce a similar trace, but in the opposite direction.

What is dynamic programming?

Forward Chaining on graph search problem is an example of dynamic programming.

Solutions to the sub problems are constructed incrementally from those of smaller sub problems and are cached to avoid recomputation.
(30) Construct a parse tree for “You give me the gold” showing the subcategories of the verb and verb phrase.

![Parse tree for "You give me the gold" showing subcategorization of the verb and verb phrase.]

(31) What is semantic interpretation? Give an example.
Semantic interpretation is the process of associating an FOL expression with a phrase.

\[
\begin{align*}
\text{Exp}(x) &\rightarrow \text{Exp}(x_1) \text{ Operator}(op) \text{ Exp}(x_2) \{ x = \text{Apply}(op, x_1, x_2) \} \\
\text{Exp}(x) &\rightarrow (\text{Exp}(x)) \\
\text{Exp}(x) &\rightarrow \text{Number}(x) \\
\text{Number}(x) &\rightarrow \text{Digit}(x) \\
\text{Number}(x) &\rightarrow \text{Number}(x_1) \text{ Digit}(x_2) \{ x = 10 \times x_1 + x_2 \} \\
\text{Digit}(x) &\rightarrow x \{ 0 \leq x \leq 9 \} \\
\text{Operator}(x) &\rightarrow x \{ x \in \{+, -, \div, \times \} \}
\end{align*}
\]

Figure 22.14 A grammar for arithmetic expressions, augmented with semantics. Each variable \(x_i\) represents the semantics of a constituent. Note the use of the \(\{\text{test}\}\) notation to define logical predicates that must be satisfied, but that are not constituents.

(32) Construct a grammar and sentence for “John loves Mary”
Construct a parse tree for the sentence “Every agent smells a Wumpus”
Define lexical, syntactic, and semantic ambiguity.

**Lexical ambiguity**, in which a word has more than one meaning. Lexical ambiguity is quite common; "back" can be an adverb (go back), an adjective (back door), a noun (the back of the room) or a verb (back up your files). "Jack" can be a name, a noun (a playing card, a six-pointed metal game piece, a nautical flag, a fish, a male donkey, a socket, or a device for raising heavy objects), or a verb (to jack up a car, to hunt with a light, or to hit a baseball hard).

**Syntactic ambiguity** (also known as structural ambiguity) can occur with or without lexical ambiguity. For example, the string "I smelled a wumpus in 2,2" has two parses: one where the prepositional phrase "in 2,2" modifies the noun and one where it modifies the verb. The syntactic ambiguity leads to a semantic ambiguity, because one parse means that the wumpus is in 2,2 and the other means that a stench is in 2,2. In this case, getting the wrong interpretation could be a deadly mistake.

**Semantic ambiguity** can occur even in phrases with no lexical or syntactic ambiguity.

For example, the noun phrase "cat person" can be someone who likes felines or the lead of the movie Attack of the Cat People. A "coast road" can be a road that follows the coast or one that leads to it.

What is disambiguation?

**Disambiguation**

Disambiguation is a question of diagnosis. The speaker's intent to communicate
is an unobserved cause of the words in the utterance, and the hearer's job is to work backwards from the words and from knowledge of the situation to recover the most likely intent of the speaker. Some sort of preference is needed because syntactic and semantic interpretation rules alone cannot identify a unique correct interpretation of a phrase or sentence. So we divide the work: syntactic and semantic interpretation is responsible for enumerating a set of candidate interpretations, and the disambiguation process chooses the best one.

(36) **What is discourse?**

A discourse is any string of language—usually one that is more than one sentence long. Textbooks, novels, weather reports and conversations are all discourses. So far we have largely ignored the problems of discourse, preferring to dissect language into individual sentences that can be studied *in vitro*. We will look at two particular subproblems: reference resolution and coherence.

**Reference resolution**

Reference resolution is the interpretation of a pronoun or a definite noun phrase that refers to an object in the world. The resolution is based on knowledge of the world and of the previous parts of the discourse. Consider the passage "John flagged down the waiter. He ordered a hamburger."

To understand that "he" in the second sentence refers to John, we need to have understood that the first sentence mentions two people and that John is playing the role of a customer and hence is likely to order, whereas the waiter is not.

**The structure of coherent discourse**

If you open up this book to 10 random pages, and copy down the first sentence from each page. The result is bound to be incoherent. Similarly, if you take a coherent 10-sentence passage and permute the sentences, the result is incoherent. This demonstrates that sentences in natural language discourse are quite different from sentences in logic. In logic, if we TELL sentences A, B and C to a knowledge base, in any order, we end up with the conjunction A ∧ B ∧ C. In natural language, sentence order matters; consider the difference between "Go two blocks. Turn right." and "Turn right. Go two blocks."

(37) **What is grammar induction?**

Grammar induction is the task of learning a grammar from data. It is an obvious task to attempt, given that it has proven to be so difficult to construct a grammar by hand and that billions of example utterances are available for free on the Internet. It is a difficult task because the space of possible grammars is infinite and because verifying that a given grammar generates a set of sentences is computationally expensive.
Grammar induction can learn a grammar from examples, although there are limitations on how well the grammar will generalize.