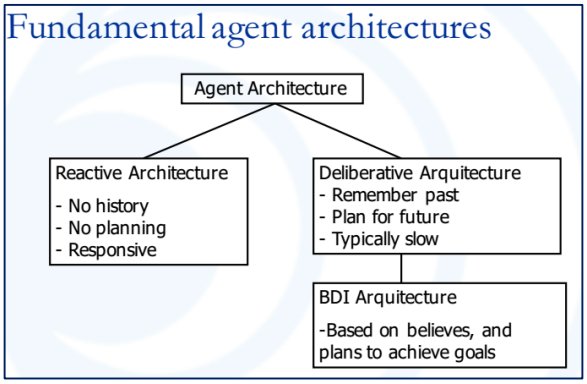
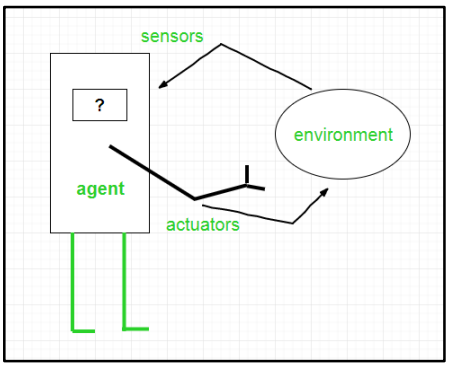
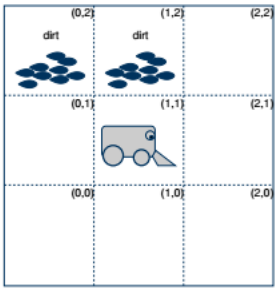
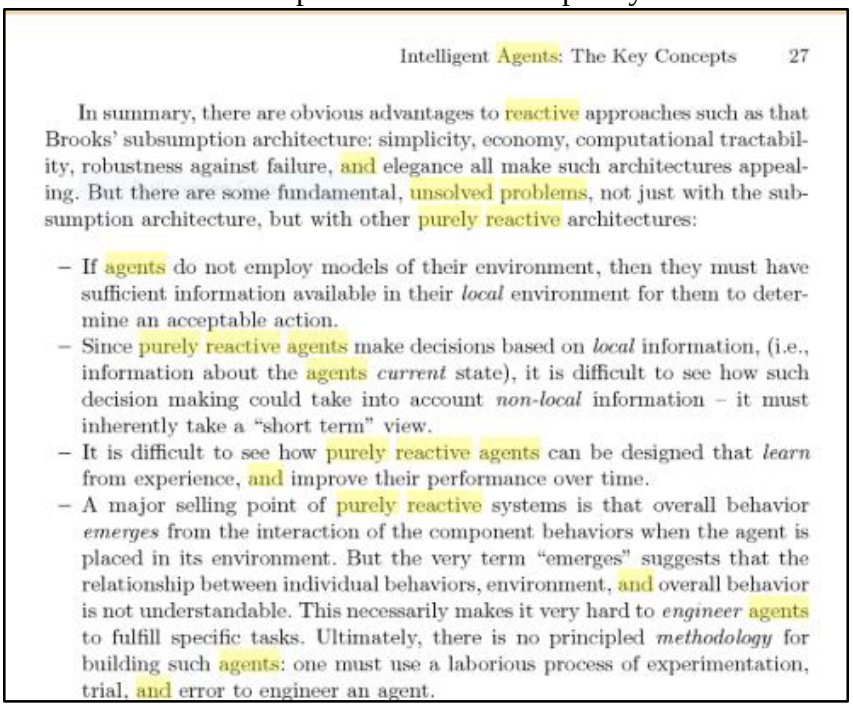
CS8691 Artificial Intelligence – 2017 Regulations	
UNIT IV - SOFTWARE AGENTS	
Architecture for Intelligent Agents – Agent communication – Negotiation and Bargaining – Argumentation among Agents – Trust and Reputation in Multi-agent systems.	
PART - A	
Q.No	**Questions**
1.	Define Purely Reactive Agents.  A *reactive* system is one that maintains an ongoing interaction with its environment, and responds to changes that occur in it (in time for the response to be useful) Purely reactive agents ▪ Some agents decide what to do without reference to their history — they base their decision making entirely on the present, with no reference at all to the past. ▪ We call such agents *purely reactive*: $action : E \rightarrow Ac$ ▪ A thermostat is a purely reactive agent $action(e) = \begin{cases} \text{off} & \text{if } e = \text{temperature OK} \\ \text{on} & \text{otherwise.} \end{cases}$
2.	What are the two types of information source? Data-mining agents This agent uses information technology to find trends and patterns in an abundance of information from many different sources. The user can sort through this information in order to find whatever information they are seeking. A data mining agent operates in a data warehouse discovering information. A 'data warehouse' brings together information from lots of different sources. "Data mining" is the process of looking through the data warehouse to find information that you can use to take action, such as ways to increase sales or keep customers who are considering defecting.

3.	What are characteristics of the subsumption architecture? **Subsumption architecture** is a reactive robotic architecture heavily associated with behavior-based robotics which was very popular in the 1980s and 90s. The term was introduced by Rodney Brooks and colleagues in 1986.[1][2][3] Subsumption has been widely influential in autonomous robotics and elsewhere in real-time AI. • Situatedness – A major idea of situated AI is that a robot should be able to react to its environment within a human-like time-frame. According to Brooks, situated mobile robot should not represent the world via an internal set of symbols and then act on this model. But "the world is its own best model", which means that proper perception-to-action setups can be used to directly interact with the world as opposed to modelling it. • **Embodiment** – Building an embodied agent accomplishes two things. **(1)** The designer to test and create an integrated physical control system, not theoretic models or simulated robots that might not work in the physical world. **(2)** Directly coupling sense-data to meaningful actions. • **Intelligence** – Developing perceptual and mobility skills are a necessary foundation for human-like intelligence. The intelligence is determined by the dynamics of interaction with the world. • **Emergence** – Conventionally, individual modules are not considered intelligent by themselves. It is the interaction of such modules, evaluated by observing the agent and its environment, that is usually deemed intelligent (or not).
4.	State the advantage of vertically layered architecture. **Advantages** Low complexity. If there are n layers there are n-1 interfaces between them. If each layer is capable of suggesting m possible actions then there are at most m2(n-1) interactions • No central control, no bottleneck in the agent's decision making
5.	**Explore some interesting properties of agents and perception.** Artificial intelligence is defined as a study of rational agents. A rational agent could be anything which makes decisions, as a person, firm, machine, or software. It carries out an action with the best outcome after considering past and current percepts(agent's perceptual inputs at a given instance). An AI system is composed of an **agent and its environment**. The agents act in their environment. The environment may contain other agents. An agent is anything that can be viewed as : • perceiving its environment through **sensors** and • acting upon that environment through **actuators**

	
6.	What are four classes of agents? Types of Agents Agents can be grouped into four classes based on their degree of perceived intelligence and capability : • Simple Reflex Agents • Model-Based Reflex Agents • Goal-Based Agents • Utility-Based Agents • Learning Agent
7.	What are logical formulae and logical deduction? **Logical Deduction:** The phenomenon of deriving a conclusion from a single proposition or a set of given propositions, is known as **logical deduction**. The given propositions are also referred to as the **premises**. Logical Deduction is reasoning which constructs or evaluates deductive arguments. Deductive arguments are attempts to show that a conclusion necessarily follows from a set of premises or hypotheses. A deductive argument is valid if the conclusion does follow necessarily from the premises, i.e., the conclusion must be true provided that the premises are true. A deductive argument is sound if it is valid and its premises are true. Deductive arguments are valid or invalid, sound or unsound. Deductive reasoning is a method of gaining knowledge. Deductive reasoning = specific kind of symbolic approach where representations are **logical formulae** and syntactic manipulation used is **logical deduction (theorem proving)** Example: the vacuum world • A small robot to help with housework • Perception: dirt sensor, orientation (north, south, east, west) • Actions: suck up dirt, step forward, turn right by 90 degrees • Starting point (0; 0), robot cannot exit room

<table>
<tr><td></td><td>


• Goal: traverse the room continually, search for and remove dirt
Example: the vacuum world
• Formulate this problem in logical terms:
• Percept is dirt or null, actions forward, suck or turn
• Domain predicates In(x; y), Dirt(x; y), Facing(d)
• next function must update internal (belief) state of agent correctly
• old(Δ) := fP(t1 : : : tn)jP 2 fIn; Dirt; Facingg ^ P(t1 : : : tn) 2 Δg
• Assume new : D × Per ! D adds new predicates to database
(what does this function look like?)
• Then, next(Δ; p) = (Δnold(Δ)) [new(Δ; p)
• Agent behaviour specified by (hardwired) rules, e.g.
In(x; y) ^ Dirt(x; y)) Do(suck)
In(0; 0) ^ Facing(north) ^ :Dirt(0; 0)) Do(forward)
In(0; 1) ^ Facing(north) ^ :Dirt(0; 1)) Do(forward)
In(0; 2) ^ Facing(north) ^ :Dirt(0; 2)) Do(turn)
In(0; 2) ^ Facing(east)) Do(forward)</td></tr>
<tr><td>8.</td><td>What are the unsolved problems with other purely reactive architectures?

Intelligent Agents: The Key Concepts 27

In summary, there are obvious advantages to reactive approaches such as that Brooks' subsumption architecture: simplicity, economy, computational tractability, robustness against failure, and elegance all make such architectures appealing. But there are some fundamental, unsolved problems, not just with the subsumption architecture, but with other purely reactive architectures:

– If agents do not employ models of their environment, then they must have sufficient information available in their local environment for them to determine an acceptable action.
– Since purely reactive agents make decisions based on local information, (i.e., information about the agents current state), it is difficult to see how such decision making could take into account non-local information – it must inherently take a "short term" view.
– It is difficult to see how purely reactive agents can be designed that learn from experience, and improve their performance over time.
– A major selling point of purely reactive systems is that overall behavior emerges from the interaction of the component behaviors when the agent is placed in its environment. But the very term "emerges" suggests that the relationship between individual behaviors, environment, and overall behavior is not understandable. This necessarily makes it very hard to engineer agents to fulfill specific tasks. Ultimately, there is no principled methodology for building such agents: one must use a laborious process of experimentation, trial, and error to engineer an agent.</td></tr>
<tr><td>9.</td><td>Define belief-desire-intention (BDI) architectures

BDI agents

A BDI agent is a particular type of bounded rational software agent, imbued with particular mental attitudes, viz: Beliefs, Desires and Intentions (BDI).</td></tr>
</table>

	Belief-Desire-Intention (BDI) Architecture
	The BDI architecture is based on practical reasoning by Bratman's philosophical emphasis on intentional stance (Bratman, 1987). Practical reasoning is reasoning toward actions - the process of figuring out what to do. This is different from the theoretical reasoning process as it derives knowledge or reaches conclusions by using one's beliefs and knowledge.
	Architecture
	The following defines the idealized architectural components of a BDI system.
	• **Beliefs**: Beliefs represent the informational state of the agent, in other words its beliefs about the world (including itself and other agents). Beliefs can also include inference rules, allowing forward chaining to lead to new beliefs. Using the term *belief* rather than *knowledge* recognizes that what an agent believes may not necessarily be true (and in fact may change in the future).
	  ○ **Beliefset**: Beliefs are stored in database (sometimes called a *belief base* or a *belief set*), although that is an implementation decision.
	• **Desires**: Desires represent the motivational state of the agent. They represent objectives or situations that the agent *would like* to accomplish or bring about. Examples of desires might be: *find the best price*, *go to the party* or *become rich*.
	  ○ **Goals**: A goal is a desire that has been adopted for active pursuit by the agent. Usage of the term *goals* adds the further restriction that the set of active desires must be consistent. For example, one should not have concurrent goals to go to a party and to stay at home – even though they could both be desirable.
	• **Intentions**: Intentions represent the deliberative state of the agent – what the agent *has chosen* to do. Intentions are desires to which the agent has to some extent committed. In implemented systems, this means the agent has begun executing a plan.
	  ○ **Plans**: Plans are sequences of actions (recipes or knowledge areas) that an agent can perform to achieve one or more of its intentions. Plans may include other plans: my plan to go for a drive may include a plan to find my car keys.
	• **Events**: These are triggers for reactive activity by the agent. An event may update beliefs, trigger plans or modify goals.

10.	What are the two types of control flow within layered architectures? There are two types of vertical layered architectures namely one-pass and two-pass control architectures. In one-pass architecture, control flows from the initial layer that gets data from sensors to the final layer that generates action output (see Figure 7). In two-pass architecture, data flows up the sequence of layers and control then flows back down (see Figure 8).

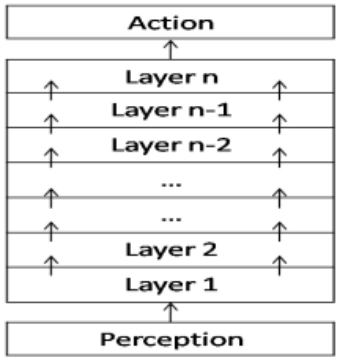
<table>
<tr><td></td><td>


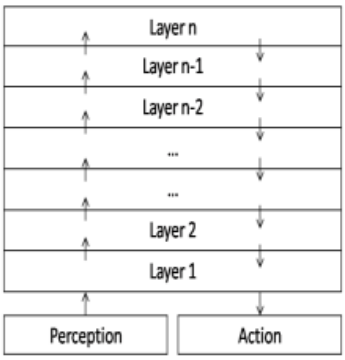
Figure 7: Vertical layer architecture: one-pass


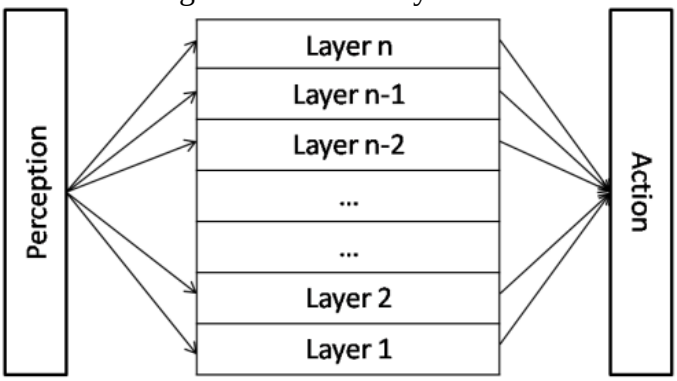
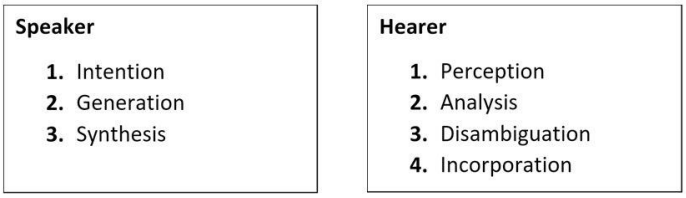
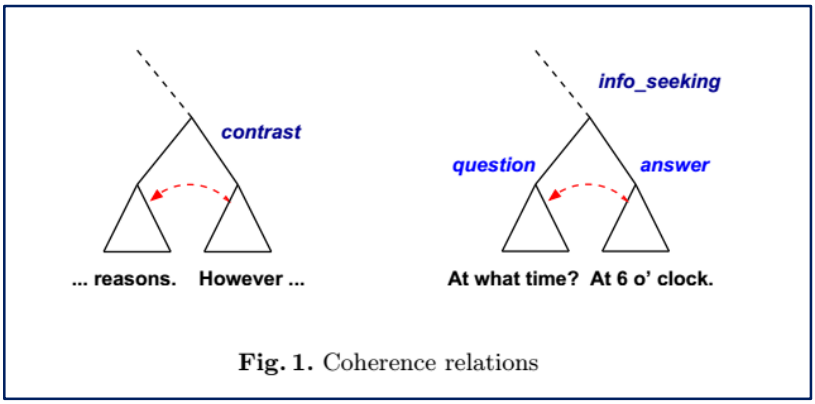
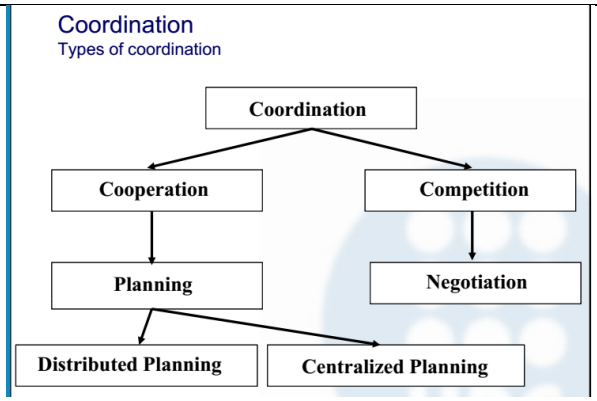
Figure 8: Vertical layer architecture: two-pass
</td></tr>
<tr><td>11.</td><td>
State the advantage of horizontal layered architectures.


Figure 5: Horizontal Layer Architecture
The advantage of horizontal layer architecture is that only n layers are required for mapping to n different types of behaviours.
</td></tr>
<tr><td>12.</td><td>
Define Agent Communication.
Components of communicating agents
communicating consists of the speaker and the hearer. Because for communication to take place, the agent must be able to perform both these tasks. Both these components can be further explained as follows on the basis of their functioning:


</td></tr>
<tr><td>13.</td><td>
Define Coherence.

Coherence
We review the notion of coherence as used in linguistics. Intuitively, a discourse (text or dialogue) can be called coherent when its parts ‘belong together’. Coherence has been studied in natural language semantics and pragmatics under the header of discourse structure. Aspects of coherence that have to do with form are also called cohesion [13]. In natural language, cohesion shows by the use of a consistent vocabulary, a consistent style and parallel syntactic constructions. The use of anaphora and ellipsis to refer back to objects mentioned earlier gives the impression of a coherent discourse. Coherence is strongly related with the topic structure. A discourse of which the topics of each of the
</td></tr>
</table>

	utterances are related, for example becausethey are subtopics, makes a more coherent impression than a text with frequent topic shifts. A common approach to analyze coherence is rhetorical structure theory [19]. The content expressed by difierent utterances is related by rhetorical relations, such as elaboration, explanation or contrast. Rhetorical relations are also called *coherence relations*. They are typically marked by adverbials like 'because' (explanation), or 'however' (contrast). An example is given on the left of flgure 1. If no rhetorical relation can be found to link utterances, the discourse is incoherent. This also relates to the function of utterances. If each utterance contributes to a single purpose, for example to convince the reader or explain something, this increases coherence.  **Fig. 1.** Coherence relations A so called discourse context records the contributions of each of the utterances to the over-all meaning of a discourse. By means of a context, the global notion of coherence can now be reduced to a local notion of coherence with respect to the context. 1. An utterance U is coherent with context C iff a coherence relation R can be found that connects U with some part U' of C. 2. In this case, a new context $C' = C + U$ is created which adds the content of U to C in a way that depends on $R(U, U')$. 3. A sequence of utterances $U_1, ..., U_n$ is called coherent, iff each U_k is coherent with discourse context C_k, which represents the contributions of $U_1, ..., U_{k-1}$.
14.	Define the property of Coordination ➢ **Coordination** is a desired property in a Multiagent System whose agents should perform complex tasks in a shared environment. ➢ The **degree of coordination** in a Multiagent System depends on: -The inability of each individual agent to achieve the whole task(s) -The dependency of one agent on others to achieve the tasks -The need to reduce/optimize resource usage -The need to avoid system halts The need to keep some conditions holding

	
15.	What are the three aspects to the formal study of communication? **Communications Process** Communications is a continuous process which mainly involves three elements viz. sender, message, and receiver. The elements involved in the communication process are explained below in detail: 1. **Sender** The sender or the communicator generates the message and conveys it to the receiver. He is the source and the one who starts the communication 2. **Message** It is the idea, information, view, fact, feeling, etc. that is generated by the sender and is then intended to be communicated further. **Receiver** He is the person who is last in the chain and for whom the message was sent by the sender.
16.	What are the fields Used in protocol? Protocols play a central role in agent communication. A protocol specifies the rules of interaction between two or more communicating agents by restricting the range of allowed follow-up utterances for each agent at any stage during a communicative interaction (dialogue). Such a protocol may be imposed by the designer of a particular system or it may have been agreed upon by the agents taking part in a particular communicative interaction before that interaction takes place. **EXAMPLE : Auction protocol** **Auction:** An *auction* is defined as an interaction between any number of buyers and a single seller that lasts for a predetermined time, mediated by a broker. Technically, the auction is regarded as a single-item, first-price, open-cry, ascending auction (Parsons et al. 2011; Harris and Raviv 1981). An auction is started as soon as the seller accepts the proposal from the broker to host it, and during its lifecycle

	the broker receives bids from any buyer agent. The broker does not interact with the seller during this time, and therefore can accept or reject an offer based on whether or not the offering agent has violated any norms. Once a buyer has its offer accepted by the broker, the following norms are created amongst the buyer, seller, and broker. a(buyer,broker,true,bid) (3) c(buyer,seller,highest_bid,payment) (4) c(seller,buyer,payment,delivery) (5) Norm 3 states that all buyers are authorized to make bids on the auctioned item. Norm 4 states that the buyer with the highest bid is committed to sending the payment to the seller. This commitment ensures that there is no way for a buyer to retract a bid (that has not been outbid) without violating their commitment. Norm 5 is the same commitment from the direct sales protocol (Norm 2) that handles delivery of the item once it is paid for.
17.	Define Ontology. ➢ A conceptualization is a map from the problem domain into the representation. ➢ A conceptualization specifies: o I What sorts of individuals are being modeled o I The vocabulary for specifying individuals, relations and properties o I The meaning or intention of the vocabulary If more than one person is building a knowledge base, they must be able to share the conceptualization. ➢ An ontology is a specification of a conceptualization. ➢ An ontology specifies the meanings of the symbols in an information system.
18.	Define bargaining. A bargaining problem deals with a situation where some players negotiate over sharing a fixed sum of resources. There are two approaches to analyzing a bargaining problem, namely the cooperative approach and the non-cooperative approach. One well-known and widely adopted cooperative bargaining solution is the Nash (1950) bargaining solution. An equally popular and important non-cooperative bargaining solution is the subgame perfect equilibrium in Rubinstein's (1982) bilateral bargaining model. The Model A finite number of players, called players 1, 2,...,n, negotiate how to split a pie of size 1 via (n − 1) bilateral bargaining sessions. In each bilateral bargaining session, two players negotiate a partial and bilateral agreement that specifies the share of the pie for one of the players who then leaves the game. After a partial agreement, the other player continues to negotiate with the rest of the players over the remainder of the pie. The (n − 1) bilateral bargaining sessions determine (n−1) players' shares of the pie and hence all n players' shares of the pie.

19.	Give the Diagrammatic Representation of Trust and Reputation Models for Multiagent Systems. ➢ "Trust begins where knowledge [certainty] ends: trust provides a basis dealing with uncertain, complex, and threatening images of the future." (Luhmann,1979) ➢ "Trust is the outcome of observations leading to the belief that the actions of another may be relied upon, without explicit guarantee, to achieve a goal in a risky situation." (Elofson, 2001) ➢ Reputation is one of the elements that allows us to build trust. ➢ Reputation has also a social dimension. It is not only useful for the individual but also for the society as a mechanism for social order.
20.	Define agent architecture. Agent architectures ▪ An architecture proposes a particular methodology for building an autonomous agent ▫ How the construction of the agent can be decomposed into the construction of a set of component modules ▫ How these modules should be made to interact ▫ These two aspects define how the sensor data and the current internal state of the agent determine the actions (effector outputs) and future internal state of the agent Main kinds of agent architectures ▪ Reactive architectures ▫ Focused on fast reactions/responses to changes detected in the environment ▪ Deliberative architectures (symbolic) ▫ Focused on long-term planning of actions, centred on a set of basic goals ▪ Hybrid architectures ▫ Combining a reactive side and a deliberative side
PART - B	
1.	What are Abstract Architectures for Intelligent Agents.(13) **Logic-Based Architecture** Logic-based architecture also known as the symbolic-based or deliberative architecture is one the earliest agent architecture that rests on the physical-symbol systems hypothesis (Newell & Simon, 1976). This classical architecture is based on the traditional artificial symbolic approach by representing and modeling the environment and the agent behavior with symbolic representation. Thus, the agent behavior is based on the

manipulation of the symbolic representation.
Building agent in logic-based approach is viewed as a deduction process. An agent is encoded as a logical theory by using specification and the process of selecting the action is through deduction process that reduces the problem to a solution such as in theorem proving.

Reactive Architecture
Reactive agent architecture is based on the direct mapping of situation to action. It is different from the logic-based architecture where no central symbolic world model and complex symbolic reasoning are used. Agent responses to changes in the environment in a stimulus-response based. The reactive architecture is realized through a set of sensors and effectors, where perceptual input is mapped to the effectors to changes in the environment. Brook's subsumption architecture is known as the best pure reactive architecture (Brooks, 1986). This architecture was developed by Brook who has critiqued on many of the drawbacks in logic-based architecture.

One of the advantages of reactive architecture is that it is less complicated to design and implement than logic-based architecture. An agent's behaviour is computationally tractable. The robustness of reactive architecture against failure is another advantage. Complex behaviours can be achieved from the interaction of simple ones. The disadvantages of reactive architecture include (1) insufficient information about agent's current state to determine an activation action due to modelling of environment available, (2) the processing of the local information limits the planning capabilities in long term or bigger picture and hence, learning is difficult to be achieved, (3) emergent behaviour which is not yet fully understood making it even more intricate to engineer. Therefore, it is difficult to build task-specific agents and one of the solutions is to evolve the agents to perform certain tasks
(Togelius, 2003). The work in this domain is referred to as artificial life.

Belief-Desire-Intention (BDI) Architecture
The BDI architecture is based on practical reasoning by Bratman's philosophical emphasis on intentional stance (Bratman, 1987). Practical reasoning is reasoning toward actions - the process of figuring out what to do. This is different from the theoretical reasoning process as it derives knowledge or reaches conclusions by using one's beliefs and knowledge. Human practical reasoning involves two activities namely deliberation and means-end reasoning. Deliberation decides what state of affairs needs to be achieved while means-end reasoning decides how to achieve these states of affairs. In BDI architecture, agent consists of three logic components referred as mental states/mental attitudes namely beliefs, desires and intentions. Beliefs are the set of information an agent has about the world. Desires are the agent's motivation or possible options to carry out the actions. Intentions

are the agent's commitments towards its desires and beliefs. Intentions are key component in practical reasoning. They describe states of affairs that the agent has

	committed to bringing about and as a result they are action-inducing. Forming the intentions is critical to an agent's success. **Layered (Hybrid) Architecture** Layered (hybrid) architecture is an agent architecture which allows both reactive and deliberate agent behavior. Layered architecture combines both the advantages of reactive and logic-based architecture and at the same time alleviates the problems in both architectures. Subsystems are decomposed into a layer of hierarchical structure to deal with different behaviours. There are two types of interaction that flow between the layer namely horizontal and vertical. In the horizontal layer architecture, each layer is directly connected to the sensory input and action output (see Figure 5). Each layer is like an agent mapping the input to the action to be performed. The advantage of horizontal layer architecture is that only *n* layers are required for mapping to *n* different types of behaviours. However, a mediator function is used to control the inconsistent actions between layer interactions. Another complexity is the large number of possible interactions between horizontal layers–*mn* (where m is the number of actions per layer). Vertical layer architecture eliminates some of these issues as the sensory input and action output are each dealt with by at most one layer each (creating no inconsistent action suggestions)
2.	Write briefly on Concrete Architectures for Intelligent Agents.(13) *Concrete architecture.* Concrete architecture is defined by starting from an abstract architecture: each component is assigned a type chosen among the ones the language provides; each macro-instruction of the engine is implemented. *Agent class.* A class is defined by instantiating the components in the concrete architecture which contain the program of the agent.

3.3 Concrete Architecture Definition

In the definition of the concrete architecture, all the components are assigned a type, the global variables are initialized, and the definitions of partially specified procedures are completed. To illustrate two different implementation choices, we consider two concrete BDI architectures, bdi_1 and bdi_2, obtained from the previously defined abstract BDI architecture *bdi*.

In concrete architecture bdi_1, *plans_component* is assigned a type `stack`, *beliefs_component* has type `set`, *goals_component* has type `set` and *intentions_component* has a type `queue`. External events are either events generated by the environment or messages sent by other agents in the system. An agent which is implemented using this architecture will have both reactivity and social ability.

In the architecture bdi_2, *plans_component*, *beliefs_component* and *goals_component* are `sets`, while *intentions_component* is a `stack`[2]. The only perceived events are those generated by the environment. This architecture gives origin to strongly reactive agents without the ability to receive messages. In both concrete architectures, perceived events are collected in the global variable *event* which has type queue.

The implementation of the BDI concrete architectures is depicted in Figure 4 and 5. In bdi_1, an event is perceived from the environment by means of the **perceive**(e_1) procedure. The global variable *event* is updated by inserting the perceived event into it. A message from *sender* is received in parallel with the perception of the environment, and the received message is also put into the event queue. In bdi_2, only events taking place in the environment are perceived and inserted in the event queue.

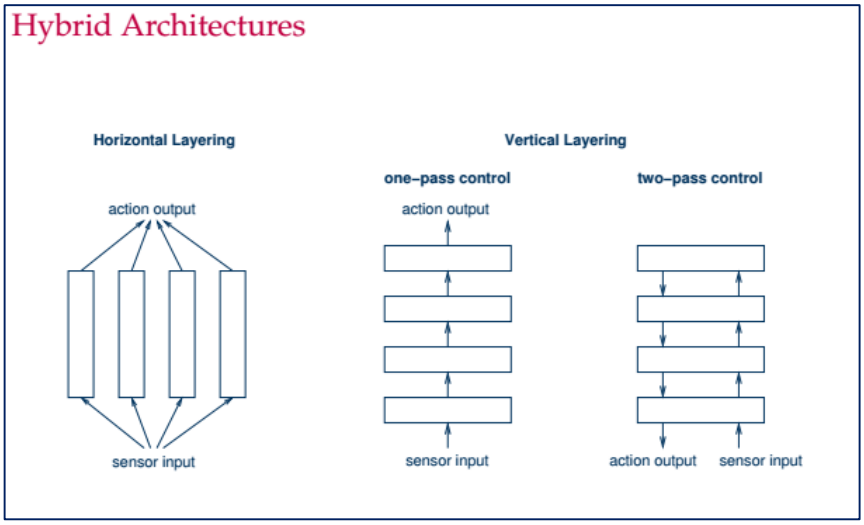
```
architecture {bdi1} is a {bdi} {
  components {
      plans_component: stack;
      belief_component, goals_component: set;
      intention_component: queue
  };
  init_global_vars { ... };
  procedures {
      perceive_event() {
          decl sender, e_1, e_2;
          ( perceive(e_1); event := insqueue(event, e_1)) ||
          ( get_belief_component("sender", !sender); rec(sender, e_2);
          event := insqueue(event, e_2) )
      };
      ...
  }
}
```

Fig. 4. Definition of concrete architecture bdi_1.

3. Write a short note on Layered architectures. (13)

Hybrid Architectures

- *Meta-level control of interactions between these components becomes a key issue in hybrid architectures*
- *Commonly used:* **layered** *approaches*
- *Horizontal layering:*
- *All layers are connected to sensory input/action output*
- *Each layer produces an action, different suggestions have to be reconciled*

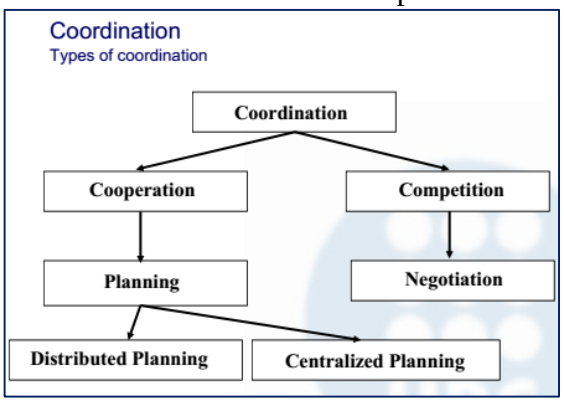
	• *Vertical layering:* • *Only one layer connected to sensors/effectors* • *Filtering approach (one-pass control): propagate intermediate decisions from one layer to another* • *Abstraction layer approach (two-pass control): different layers make decisions at different levels of abstraction*  Touring Machines • Horizontal layering architecture • Three sub-systems: Perception sub-system, control sub-system and action sub-system • Control sub-system consists of • Reactive layer: situation-action rules • Planning layer: construction of plans and action selection • Modelling layer: contains symbolic representations of mental states of other agents • The three layers communicate via explicit **control rules**
4.	Define Agent Communication. Write a short note on coordination, Dimensions of meaning and Message types.(13) A multi-agent system (MAS) may be seen as a collection of collaborative agents „ They can communicate and cooperate with other agents, while keeping their autonomy „ They usually *negotiate* with their peers to reach mutually acceptable agreements during *cooperative problem solving* They normally have limited learning capabilities „ Collaborative agents are usually *deliberative* agents (e.g. BDI model), with some reasoning capabilities ‰ Reactive agents can hardly communicate and collaborate (only through actions that modify the common environment) „ They are usually static, complex agents

Coordination

Wooldridge and Jennings define an *Agent* as a computer program capable of taking its own decisions with no external control (*autonomy*), based on its perceptions of the environment and the objectives it aims to satisfy. An agent may take actions in response to changes in the environment (*reactivity*) and also it may take initiatives (*proactivity*).

A further attribute of agents is their ability to communicate with other agents (*social ability*), not only to share information but, more important, to **coordinate actions** in order to achieve goals for which agents do not have plans they can fulfil on their own, solving even more complex problems.

- **Coordination** is a desired property in a Multiagent System whose agents should perform complex tasks in a shared environment
- The **degree of coordination** in a Multiagent System depends on:
  - The inability of each individual agent to achieve the whole task(s)
  - The dependency of one agent on others to achieve the tasks
  - The need to reduce/optimize resource usage
  - The need to avoid system halts
  - The need to keep some conditions holding



Types of Coordination

Competition and Negotiation

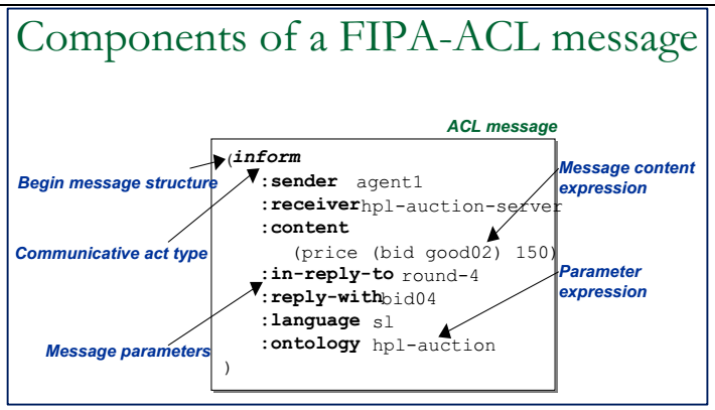
Competition is kind of coordination between antagonist agents which compete with each other or that are selfish

.We will be more interested in **Negotiation**, as it is a kind of competition that involves some higher level of intelligence.

The degree of success in negotiation (for a given agent can be measure by)

The capability of this agent to maximize its own benefit

The capability of not taking into account the other agents' benefit or even trying to minimize other agents' benefit.



Parameters in a FIPA ACL message

- :sender - who sends the message
- :receiver - who is the recipient of the message
- :content - content of the message
- :reply-with - identifier of the message
- :reply-by - deadline for replying the message
- :in-reply-to - identifier of the message being replied
- :language – language in which the content is written
- :ontology - ontology used to represent the domain concepts
- :protocol - communication protocol to be followed
- :conversation-id - identifier of conversation

5.

Explain Negotiation in detail. (13)

Need of negotiation in MAS

- Agents may have incompatible goals, and resources to achieve these goals may be limited; in such cases competition and conflicts may arise
- The effects of the agents' antagonistic behaviour needs to be limited
- Agents must be able to reach compromises, resolve conflicts, allocate goods and resources by way of an agreement
- Agents' interactions are governed by a set of rules: an interaction protocol

Negotiation protocol elements (I)

Public elements

-Negotiation set which represents the space of possible offers/proposals that the agents can make

-The protocol rules which govern the agents' interactions

Negotiation protocol elements (II)

Private elements

- The set of strategies that the agents can use to participate in the negotiation process:
- They are not dictated by the protocol itself
- May take into account the other agents' strategies

- Learning mechanisms

Protocol rules (I)

Admission rules

When an agent can participate in a negotiation
(e.g. eligibility criteria)

Interaction rules

Sequence of admissible/valid actions (e.g. moments in which bids are allowed)

Validity rules

What constitutes a legal offer/proposal (e.g. a new bid must be higher than the last bid)

Outcome determination rules

When an agreement has been reached

Protocol rules (II)

Withdrawal rules

When an agent can withdraw from the negotiation

Termination rules

When a negotiation ends unsuccesfully

Commitment rules

How the commitments that agents make
during the negotiation are managed

Negotiation factors

Number of attributes: one, many

[Multi-attribute auctions]

Number of agents:

One-to-one

One-to-many

Many-to-many

Number of units: one, many

[Multi-unit auction]

Interrelated goods: one good or a number of goods that are substitutable or interdependent

[Combinatorial auctions]

Protocol evaluation criteria (I)

Social welfare – the sum of all agent's payoffs or utilities in a given solution

Pareto efficiency – a solution x is Pareto optimal if there's no other solution x' such that at least one agent is better off in x' than in x, and no agent is worst off in x' than in x

Individual rationality – an agent should not lose out by participating in a negotiation

Protocol evaluation criteria (II)

Stability – mechanism should be designed to be non-manipulable: motivate each agent to behave in the desired manner

Computational efficiency – mechanisms
should be designed so that when agents use them, as little computation is needed as possible

Distribution and communication efficiency –
distributed protocol vs. minimum communication (time, money,…)

6.	Explain Bargaining theories in detail. (13) **Bargaining** or **haggling** is a type of negotiation in which the buyer and seller of a good or service debate the price and exact nature of a transaction. If the bargaining produces agreement on terms, the transaction takes place. Bargaining is an alternative pricing strategy to fixed prices. Optimally, if it costs the retailer nothing to engage and allow bargaining, they can deduce the buyer's willingness to spend. It allows for capturing more consumer surplus as it allows price discrimination, a process whereby a seller can charge a higher price to one buyer who is more eager (by being richer or more desperate). # Bargaining Theories ## Behavioral theory The personality theory in bargaining emphasizes that the type of personalities determine the bargaining process and its outcome. A popular behavioral theory deals with a distinction between hard-liners and soft-liners. Various research papers refer to hard-liners as warriors, while soft-liners are shopkeepers. It varies from region to region. ## Game theory Bargaining games refer to situations where two or more players must reach an agreement regarding how to distribute an object or monetary amount. Each player prefers to reach an agreement in these games, rather than abstain from doing so. However, each prefers that the agreement favor their interests. Examples of such situations include the bargaining involved in a labor union and the directors of a company negotiating wage increases, the dispute between two communities about the distribution of a common territory, or the conditions under which two countries agree on nuclear disarmament. Analyzing these kinds of problems looks for a solution that specifies which component in dispute corresponds to each party involved. Players in a bargaining problem can bargain for the objective as a whole at a precise moment in time. The problem can also be divided so that parts of the whole objective become subject to bargaining during different stages. ## Bargaining and posted prices in retail markets Retailers can choose to sell at posted prices or allow bargaining: selling at a public posted price commits the retailer not to exploit buyers once they enter the retail store, making the store more attractive to potential customers, while a bargaining strategy has the advantage that it allows the retailer to price discriminate between different types of customer. ## Processual theory This theory isolates distinctive elements of the bargaining chronology in order to better understand the complexity of the negotiating process. Several key features of the processual theory include: • Bargaining range • Critical risk • Security point ## Integrative theory Integrative bargaining (also called "interest-based bargaining," "win-win bargaining") is a negotiation strategy in which parties collaborate to find a "win-win" solution to their dispute. This strategy focuses on developing mutually beneficial agreements based on the interests of the disputants.

	Interests include the needs, desires, concerns, and fears important to each side. They are the underlying reasons why people become involved in a conflict. **Narrative theory** A very different approach to conceptualizing bargaining is as co-construction of a social narrative, where narrative, rather than economic logic drives the outcome.
7.	Narrate Argumentation among Agents in detail.(13) What is Argumentation? What philosophers call it! Arguing with Others "A verbal and social activity of reason aimed at increasing (or decreasing) the acceptability of a controversial standpoint for the listener or reader, by putting forward a constellation of propositions (i.e. arguments) intended to justify (or refute) the standpoint before a rational judge" [van Eemeren et al] "the giving of reasons to support or criticize a claim that is questionable, or open to doubt" [Walton] What is an Argument? Arguments as Chained Inference Rules Arguments as Instances of Schemes Arguments as Graphs
8.	Briefly explain (i). Communication Levels (4) (a) Sender & Receiver (b) Medium (c) Messge (d) Feedback (ii). Speech Acts (3) **Speech acts** are defined in terms of the effects of the cognitive state of the hearer that are intended by the speaker. They are seen as parts of plans that the participants find and execute Most other **Artificial Intelligence** (**AI**) work on **speech acts** is in the area of Distributed **AI**. In linguistics, a **speech act** is an utterance defined in terms of a speaker's intention and the effect it has on a listener. Essentially, it is the action that the speaker hopes to provoke in his or her audience. **Speech acts** might be requests, warnings, promises, apologies, greetings, or any number of **declarations**. (iii). Knowledge Query and Manipulation Language (KQML)(3) The **Knowledge Query and Manipulation Language**, or **KQML**, is a language and protocol for communication among software agents and knowledge-based systems.[1] It was developed in the early 1990s as part of the DARPA knowledge Sharing Effort, which was aimed at developing techniques for building large-scale knowledge bases which are

	shareable and reusable. While originally conceived of as an interface to knowledge based systems, it was soon repurposed as an Agent communication language. Work on KQML was led by Tim Finin of the University of Maryland, Baltimore County and Jay Weber of EITech and involved contributions from many researchers. The KQML message format and protocol can be used to interact with an intelligent system, either by an application program, or by another intelligent system. KQML's "performatives" are operations that agents perform on each other's knowledge and goal stores. Higher-level interactions such as contract nets and negotiation are built using these. KQML's "communication facilitators" coordinate the interactions of other agents to support knowledge sharing. (iv). Knowledge Interchange Format (KIF)(3) **Knowledge Interchange Format** (**KIF**) is a computer language designed to enable systems to share and re-use information from knowledge-based systems. KIF is similar to frame languages such as KL-One and LOOM but unlike such language its primary role is not intended as a framework for the expression or use of knowledge but rather for the interchange of knowledge between systems. The designers of KIF likened it to PostScript. PostScript was not designed primarily as a language to store and manipulate documents but rather as an interchange format for systems and devices to share documents. In the same way KIF is meant to facilitate sharing of knowledge across different systems that use different languages, formalisms, platforms, etc. KIF has a declarative semantics. It is meant to describe facts about the world rather than processes or procedures. Knowledge can be described as objects, functions, relations, and rules. It is a formal language, i.e., it can express arbitrary statements in first order logic and can support reasoners that can prove the consistency of a set of KIF statements. KIF also supports non-monotonic reasoning. KIF was created by Michael Genesereth, Richard Fikes and others participating in the DARPA knowledge Sharing Effort.[1]
9.	With diagrammatic representation, explain Trust and Reputation in Multi-agent systems in detail.(13) Trust is a multi-dimension entity which concerns various attributes such as reliability, dependability, security and honesty, among others. Trust can be basically defined as "a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action" What is Trust? It depends on the level we apply it: **User confidence** • Can we trust the user behind the agent? – Is he she a trustworthy source of some kind of knowledge? (e.g. an expert in a field) – Does he/she acts in the agent system (through his agents in a trustworthy way? **Trust of users in agents** • Issues of autonomy: the more autonomy, less trust • How to create trust? – Reliability testing for agents – Formal methods for open MAS– Security and verifiability **Trust of agents in agents** • Reputation mechanisms

	• Contracts • Norms and Social Structures Why Trust? (I) In closed environments, cooperation among agents is included as part of the designing process: - the multi-agent system is usually built by a single developer or a single team of developers and the chosen, option to reduce complexity is to ensure cooperation among the agents they build including it as an important system requirement. - *Benevolence assumption*: an agent *ai* requesting information or a certain service from agent *aj* can be sure that such agent will answer him if *aj* has the capabilities and the resources needed, otherwise *aj* will inform *ai* that it cannot perform the action requested. It can be said that in closed environments **trust is implicit.** Why Trust? (II) However, in an open environment *trust* is not easy to achieve, as - Agents introduced by the system designer can be expected to be nice and trustworthy but this cannot be ensured for alien agents out of the designer control These *alien* agents may give incomplete or false information to other agents or betray them if such actions allow them to fulfill their individual goals. In such scenarios developers use to create *competitive systems* where each agent seeks to maximize its own expected utility at the expense of other agents But, what if solutions can only be constructed by means of *cooperative problem solving*? Agents should try to cooperate, even if there is some uncertainty about the other agent's behaviour. That is, to have some explicit represent How to compute trust • Trust values can be externally defined ▪ by the system designer: the trust values are pre-defined ▪ By the human user: he can introduce his trust values about the humans behind the other agents • Trust values can be inferred from some existing representation about the interrelations between the agents ▪ Communication patterns, cooperation history logs, e-mails, webpage connectivity mapping… • Trust values can be learnt from current and past experiences ▪ Increase trust value for agent a_i if behaves properly with us ▪ Decrease trust value for agent a_i if it fails/defects us • Trust values can be propagated or shared through a MAS ▪ Recommender systems, Reputation mechanisms.
10.	Compare and contrast about the negotiation and bargaining.(13) **<u>An Argument Framework for Negotiation</u>**

Bargaining is negotiation of price alone. But negotiation may apply to much more than price, and may not include price at all. However, all negotiation involves an exchange of value, and agreements and promises of performance. Bargaining is often done verbally. Negotiation often involves written records.

Negotiation is a central process in an agent society where autonomous agents have to cooperate in order to resolve conflicting interests and yet compete to divide limited resources. A direct dialogical exchange of information between agents usually leads to competitive
forms of negotiation where the most powerful agents win. Alternatively, an intelligent mediated interaction may better achieve the goal of reaching a common agreement and supporting cooperative negotiation.
In both cases argumentation is the reference framework to rationally manage conflicting knowledge or objectives, a framework which provides the fundamental abstraction "argument" to exchange pieces of information.

Need for argumentation

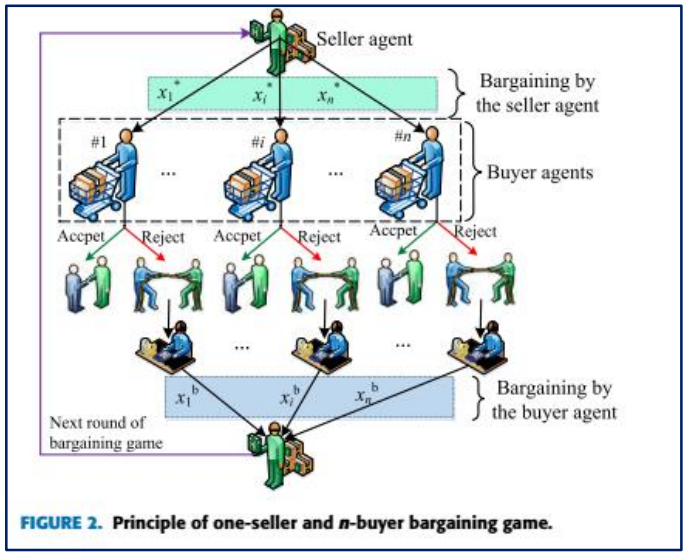
A society mainly evolves through interaction and communication among participating entities. Within a society, people argue and negotiate in order to solve problems, to resolve or reduce conflicts, to exchange information, and to inform each other of pertinent facts. In particular, argumentation is a useful feature of human intelligence that enables us to deal with incomplete and inconsistent information. People usually have only partial knowledge about
the world (they are not omniscient) and often they have to manage conflicting information.

Negotiation

The main form of communication to resolve conflict in human and artificial society is negotiation. Concretely, negotiation is an argumentative process where the participants compete for limited resources or collaborate to find common agreement over their division or allocation.
In the context of multi-agent systems there exist several approaches to realise automated forms of negotiation, through heuristics, game theory and argumentation. Because argumentation involves the requesting, provision and consideration of reasons for claims, it is the most sophisticated of these different forms of interaction for negotiation. However, providing agents with appropriate conceptual models and related software architectures to fully automate argumentation and negotiation in generic (as distinct from particular
well-defined) domains is still an unsolved research challenge.

Argumentation

Argumentation is a verbal and social activity of reason aimed at increasing (or decreasing) the acceptability of a controversial standpoint for the listener or reader, by

	putting forward a constellation of propositions intended to justify (or refute) the standpoint before a rational judge. In summary, argumentation can be seen as the principled interaction of different, potentially conflicting arguments, for the sake of arriving at a consistent conclusion. Perhaps the most crucial aspect of argumentation is the interaction between arguments. Bargaining is negotiation of price alone. But negotiation may apply to much more than price, and may not include price at all. However, all negotiation involves an exchange of value, and agreements and promises of performance. Bargaining is often done verbally. Negotiation often involves written records. **MULTI-AGENT BARGAINING LEARNING** *A. BARGAINING GAME* In a basic two-player bargaining game, the seller agent will firstly make an offer to the buyer agent, if the offer is accepted by the buyer agent, then an bargaining equilibrium (i.e., the strategy of the offer) can be determined, otherwise, the bargaining role will be shifted to be on the buyer agent in the next period until they reach an agreement on the offer. Enlighted by this game, a novel cooperative one-seller and *n*buyer bargaining game is proposed for achieving an efficient coordination between different players.  **FIGURE 2. Principle of one-seller and *n*-buyer bargaining game.**
11.	Examine the Argumentation among Agents.(13) What is Argumentation? “A verbal and social activity of reason aimed at increasing (or decreasing) the acceptability of a controversial standpoint for the listener or reader, by putting forward a constellation of propositions (i.e. arguments) intended to justify (or refute) the standpoint before a rational judge” [van Eemeren et al] “the giving of reasons to support or criticize a claim that is questionable, or open to doubt” [Walton]

Argumentation versus Reasoning

If you are the judge, argumentation becomes (nonmonotonic) reasoning

Process of Argumentation

Constructing arguments (in favor of / against a "statement") from available information.

A: "Tweety is a bird, so it flies"

B: "Tweety is just a cartoon!"

Determining the different conflicts among the arguments.

"Since Tweety is a cartoon, it cannot fly!" (B attacks A)

Evaluating the acceptability of the different arguments.

"Since we have no reason to believe otherwise, we'll assume Tweety is a cartoon." (accept B). "But then, this means despite being a bird he cannot fly." (reject A).

4 Concluding, or defining the justified conclusions.

"We conlcude that Tweety cannot fly!"

Argumentation System

Example model by Prakken

A tuple $(\mathcal{L}, -, \mathcal{R}_s, \mathcal{R}_d, \leq)$

- a logical language $\mathcal{L}$,
- strict rules $\mathcal{R}_s$
- defeasible rules $\mathcal{R}_d$
- partial order $\leq$ over $\mathcal{R}_d$
- *Contrariness function* $- : \mathcal{L} \to 2^{\mathcal{L}}$ captures conflict between formulas

  Classical negation $\neg$ *captured by* $\neg\varphi \in \overline{\varphi}$ *and* $\varphi \in \overline{\neg\varphi}$.

A particular knowledge base $(\mathcal{K}, \leq')$ with:

- $\mathcal{K} \subseteq \mathcal{L}$ divided into:
  - $\mathcal{K}_n$ are necessary *axioms* (cannot be attacked);
  - $\mathcal{K}_p$ are *ordinary premises*;
  - $\mathcal{K}_a$ are *assumptions*;
  - $\mathcal{K}_i$ are *issues*.
- $\leq'$ is a partial order on $\mathcal{K} \backslash \mathcal{K}_n$.

Inference rules:

- *defeasible* rule $\varphi_1, \ldots, \varphi_n \Rightarrow \varphi$ means conclusion φ follows *presumably* from the premises $\varphi_1, \ldots, \varphi_n$
- *strict* rule $\varphi_1, \ldots, \varphi_n \to \varphi$ stands for classical implication

Functions *Perm*(A), *Conc*(A) and *Sub*(A) returns premises, conclusion, and *sub-arguments* of argument A respectively.

Argument

An argument is any of the following:

- $\varphi \in \mathcal{K}$, where $Prem(A) = \{\varphi\}$, $Conc(A) = \varphi$, and $Sub(A) = \{\varphi\}$.
- $A_1, \ldots, A_n \to \psi$, where $A_1, \ldots, A_n$ are arguments, and there exists in $\mathcal{R}_s$ a strict rule $Conc(A_1), \ldots, Conc(A_n) \to \psi$.
- $A_1, \ldots, A_n \Rightarrow \psi$, where $A_1, \ldots, A_n$ are arguments, and there exists in $\mathcal{R}_d$ a defeasible rule $Conc(A_1), \ldots, Conc(A_n) \to \psi$.

where

- $Prem(A) = Prem(A_1) \cup \cdots \cup Prem(A_n)$
- $Sub(A) = Sub(A_1) \cup \cdots \cup Sub(A_n) \cup \{A\}$

Argumentation Scheme

Argumentation schemes are forms (or categories) of argument, representing stereotypical ways of drawing inferences from particular patterns of premises to conclusions in a particular domain (e.g. reasoning about action).

For each scheme, we list:

- Premises
- Conclusion
- A set of critical questions that can be used to scrutinize the argument by questioning explicit or implicit premises.

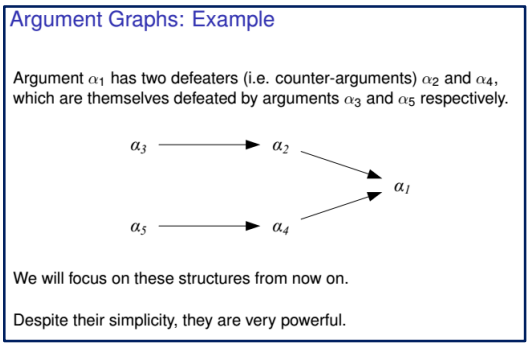
Argumentation Scheme Example

Walton's "sufficient condition scheme for practical reasoning":

In the current circumstances R
We should perform action A
Which will result in new circumstances S
Which will realise goal G
Which will promote some value V.

Associated critical questions include:

CQ1: *Are the believed circumstances true?*
CQ2: *Does the action have the stated consequences?*
CQ3: *Assuming the circumstances and that the action has the stated consequences, will the action bring about the desired goal?*
CQ4: *Does the goal realise the value stated?*
CQ5: *Are there alternative ways of realising the same consequences?*



12.	Describe the trust and reputation in multi-agent systems.(13)

What is Trust?

- It depends on the level we apply it:
  - User confidence
    - Can we trust the user behind the agent?
      - Is he/she a trustworthy source of some kind of knowledge? (e.g. an expert in a field)
      - Does he/she acts in the agent system (through his agents in a trustworthy way?
  - Trust of users in agents
    - Issues of autonomy: the more autonomy, less trust
    - How to create trust?
      - Reliability testing for agents
      - Formal methods for open MAS
      - Security and verifiability
  - Trust of agents in agents
    - Reputation mechanisms
    - Contracts
    - Norms and Social Structures
- Def: Gambetta defines *trust* as ***a particular level of subjective probability with which an agent a_j will perform a particular action both before [we] can monitor such action … and in a context in which it affects [our] own action.***
- Trust is subjective and contingent on the uncertainty of future outcome (as a result of trusting).

How to compute trust?

- Trust value can be assigned to an agent or to a group of agents
- Trust value is an asymmetrical function between agent `a1` and `a2`
  - `trust_val(a1,a2)` does not need to be equal to `trust_val(a2,a1)`
- Trust can be computed as
  - A binary value (1='I do trust this agent', 0='I don't trust this agent')
  - A set of qualitative values or a discrete set of numerical values (e.g. 'trust always', 'trust conditional to X', 'no trust') (e.g. '2', '1', '0', '-1', '-2')
  - A continuous numerical value (e.g. [-300..300])
  - A probability distribution
  - Degrees over underlying beliefs and intentions (cognitive approach)

Trust and Reputation

- Most authors in literature make a mix between trust and reputation
- Some authors make a distinction between them
  - **Trust** is an individual measure of confidence that a given agent has over other agent(s)
  - **Reputation** is a social measure of confidence that a group of agents or a society has over agents or groups
  - (social) Reputation is one mechanism to compute (individual) Trust
    - I will trust more an agent that has good reputation
    - My reputation clearly affects the amount of trust that others have towards me.
    - Reputation can have a ***sanctioning*** role in social groups: a bad reputation can be very costly to one's future transactions.

13. Explain about Planning and acting in the real world .(13)

Planning

The task of coming up with a sequence of actions that will achieve a goal is called **planning.**

- ***Planning*** *is a search problem that requires to find an efficient* ***sequence of actions*** *that transform a system from a given starting state to the* ***goal state***

Classical Planning Environments

we consider only environments that are fully observable, deterministic, finite, static (change happens only when the agent acts), and discrete (in time, action, objects, and effects). These are called **classical planning** environments. In contrast, nonclassical planning is for partially observable or stochastic environments and involves a different set of algorithms and agent designs.

Action Schema

An **action schema** represents a number of different actions that can be derived by instantiating the variables p, from, and to to different constants.
In general, an action schema consists of three parts:

(1) The **action name** and **parameter list**-
for example, Fly(p, from, to) - serves to identify the action.

(2) The **precondition** is a conjunction of function-free positive literals stating wlliat must be true in a state before the action can be executed. Any variables in the precondition must also appear in the action's parameter list.

(3) The **effect** is a conjunction of function-free literals describing how the state changes when the action is executed. A positive literal P in the effect is asserted to be true in the state resulting from the action, whereas a negative literal *P* is asserted to be false. Variables in the effect must also appear in the action's parameter list.

Example: The blocks world
One of the most famous planning domains is known as the **blocks world.** This domain consists of a set of cube-shaped blocks sitting on a table. The blocks can be stacked, but only one block can fit directly on top of another. A robot arm can pick up a block and move it to another position, either on the table or on top of another block. The arm can pick up only one block at a time, so it cannot pick up a block that has another one on it. The goal will always be to build one or more stacks of blocks, specified in terms of what blocks are on top of what other blocks. For example, a goal might be to get block A on B and block *C* on D.
We will use *On(b, x)* to indicate that block *b* is on *x,* where *x* is either another block or the table. The action for moving block *b* from the top of *x* to the top of y will be *Move(b, x,* y).
Now, one of the preconditions on moving *b* is that no other block be on it. In first-order logic, this would be 13 *x On (x , b)* or, alternatively, 'd *x 1 On(x, b).* These could be stated as

	preconditions in ADL. We can stay within the STRIPS language, however, by introducing a new predicate, *Clear(x),* that is true when nothing is on *x.* The action *Move* moves a block *b* from *x* to *y* if both *b* and *y* are clear. After the move is made, *x* is clear but *y* is not. *A* formal description of *Move* in STRIPS is *Action (Move (b,* z, *y)* , PRECOND: *On(b, x)* A *Clear(b) A Clear(y),* EFFECT: *On(b, y)* A *Clear(x) A lOn(b, x) A lClear(y))* . Unfortunately, this action does not maintain *Clear* properly when *x* or y is the table. When x = *Table,* this action has the effect *Clear(Table),* but the table should not become clear, and when *y = Table,* it has the precondition *Clear(Table),* but the table does not have to be clear to move a block onto it. To fix this, we do two things. First, we introduce another action to move a block *b* from x to the table: *Action(MoveTo Table(b, x)* , PRECOND: *On(b, x)* A *Clear(b),* EFFECT: *On(b, Table)* A *Clear(x) A lOn(b, x))* . Second, we take the interpretation of *Clear(b)* to be "there is a clear space on *b* to hold a block." Under this interpretation, *Clear(Table)* will always be true. The only problem is that nothing prevents the planner from using *Move (b, x, Table)* instead of *Move To Table (b, x)* . We could live with this problem-it will lead to a larger-than-necessary search space, but will not lead to incorrect answers-or we could introduce the predicate *Block* and add *Bloctk(b)* A *Block(y)* to the precondition of *Move.* Finally, there is the problem of spurious actions such as *Move(B, C,* C), which should be a no-op, but which has contradictory effects. It is common to ignore such problems, because they seldom cause incorrect plans to be produced. The correct approach is add inequality preconditions as shown in Figure 11.4. $Init(On(A, Table) \wedge On(B, Table) \wedge On(C, Table)$ $\wedge Block(A) \wedge Block(B) \wedge Block(C)$ $\wedge Clear(A) \wedge Clear(B) \wedge Clear(C))$ $Goal(On(A, B) \wedge On(B, C))$ $Action(Move(b, x, y),$ PRECOND: $On(b, x) \wedge Clear(b) \wedge Clear(y) \wedge Block(b) \wedge$ $(b \neq x) \wedge (b \neq y) \wedge (x \neq y),$ EFFECT: $On(b, y) \wedge Clear(x) \wedge \neg On(b, x) \wedge \neg Clear(y))$ $Action(MoveToTable(b, x),$ PRECOND: $On(b, x) \wedge Clear(b) \wedge Block(b) \wedge (b \neq x),$ EFFECT: $On(b, Table) \wedge Clear(x) \wedge \neg On(b, x))$ **Figure 11.4** A planning problem in the blocks world: building a three-block tower. One solution is the sequence $[Move(B, Table, C), Move(A, Table, B)]$.
14.	How do you execute the planning in solving problems? (13)
PART-C	

1	Create and design the architecture of intelligence agent with an example. (15) ## Pedagogical Agents Pedagogical agents have come a long way in 20 years. During this time, they have evolved from largely expressionless, robotic characters to empathetic and caring supporters of learning. The trend to make machines more human-like seems to be having a positive influence on learning outcomes, at least in particular ways, and in certain situations. They have progressed from simple demonstrations, to the focus of large-scale studies that address cognitive and noncognitive outcomes. Despite advances in animation and video-game technologies, pedagogical agents still remain far behind in terms of their use of nonverbal behaviors when compared with expert human teachers. The use of gestures during teaching can reinforce concepts and support comprehension, and when used appropriately, can have a direct impact on learning outcomes (Alibali et al., 2013). Thus, it is likely that the full potential of pedagogical agents has not yet been explored, specifically to explore how they might promote learning via nonverbal signals and appropriate use of gestures. **Strengthen Links Between Agent Behaviors and Learner Emotions** The pedagogical agents provide unique opportunities (over non-agent enabled learning environments) to increase engagement and connect with learners emotionally. The argument rests on the emotional potential of simulating meaningful social interactions and has roots in social agency theory. The pedagogical agents should strive to carefully manage the emotional states of learners, helping those on the brink of frustration and disengagement and challenging those who may be approaching boredom. Using conversational and nonverbal strategies, pedagogical agents have a wide range of communicative strategies available to achieve such a balance. Build Real Relationships How successful teachers connect with students, whether it be through effective nonverbal communication or proper interpretation of student emotions, can act as a blueprint for future pedagogical agent research. ## Teaching Knowledge Pedagogical agents originate from research efforts into affective computing (personal systems able to sense, recognize, and respond to human emotions), artificial intelligence (simulating human intelligence, speech recognition, deduction, inference, and creative response), and gesture and narrative language (how artifacts, agents, and toys can be designed with psychosocial competencies). solving context.

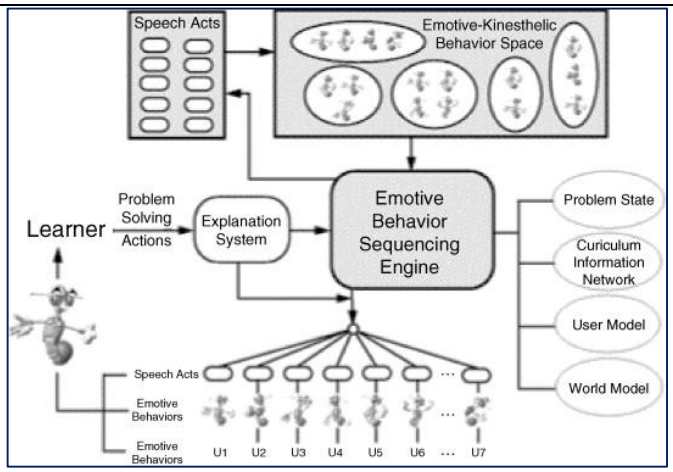


Figure 4.13. An emotive-kinesthetic behavior sequencing architecture used with Cosmos (Lester et al., 1999b).

Pedagogical agents have many liabilities. They are complex to create, text-to-speech with a robotic voice can be annoying to learners, speech recognition technology is not strong enough for widespread use, and text input through natural language understanding (NLU) technology is in its infancy (see Sections 5.5 and 5.6). Animated pedagogical agents are better suited to teach objective information with clear right and wrong answers rather than material based in theory or discussion (Slater, 2000).

4.4.2.1 Emotive Agents

Pedagogical agents often appear to have emotion along with an understanding of the student's problems, providing contextualized advice and feedback similar to a personal tutor (Lester et al., 1997a, 1999a). Human-like attributes can enhance agents' communication skills (i.e., agents rationally respond to the student's emotions or affect). Agents assume a lifelike real-time quality while interacting through a mixed-initiative graphical dialogue. Reacting in real time means the processing time for a tutor to respond to a student appears negligible or the response is immediate as it would be in conversation with another human.

4.4.2.2 Life Quality

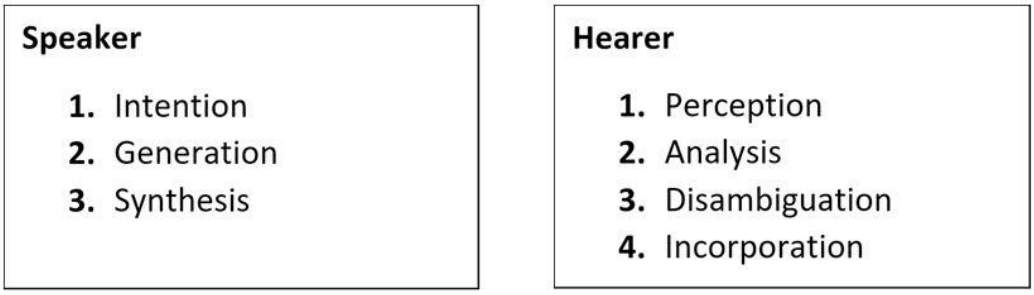
Building *life quality* into agents means that the characters' movements, if humanoid, follow a strict adherence to the laws of biology and physics. This implies that the character's musculature and kinesthetics are defined by the physical principles that govern the structure and movement of human and animal bodies (Towns et al., 1988). Facial expressions may be modeled from a human subject. For example, when a character becomes excited, it raises its eyebrows and its eyes widen. In the stylized traditional animation mode, an excited character might bulge out its eyes and leap off the ground.

Promoting metacognition

The use of adaptive scaffolding and pedagogical agents represent cutting-edge metacognitive interventions in open learning environments. Azevedo et al.

	(2004) demonstrated the effectiveness of adaptive scaffolding in monitoring college students' understanding in hypermedia and providing subsequent support.
2.	Explain about the agent communication. (15) Why do we need Agent Communication? „ *Multi agent systems allow distributed problem solving* „ *This requires the agents to coordinate their actions* „ *Agent communication facilitates this by allowing individual agents to interact* ‰ *allows cooperation* ‰ *allows information sharing* Speech Acts „ *A speech act is an act of communication* „ *Speech does not imply any particular communication media* „ *There are various types of speech act* „ *By using the various types of speech act, agents can interact effectively* Communication protocols „ *There are many situations in which agents engaged in a dialogue with a certain purpose exchange the same sequence of messages* ‰ *When an agent makes a question to another* ‰ *When an agent requests a service from another* ‰ *When an agent looks for help from other agents* „ *To ease the management of this typical message interchanges we can use predefined* **protocols** **Communicating agents** Communicating consists of the speaker and the hearer. Because for communication to take place, the agent must be able to perform both

these tasks. Both these components can be further explained as follows on the basis of their functioning:



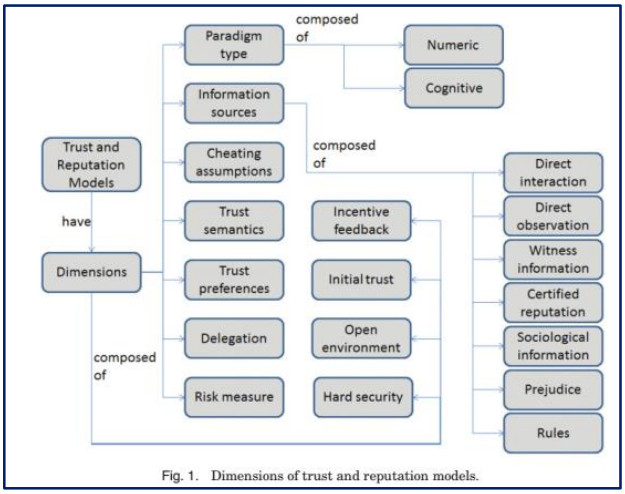
Speaker

1. **Intention:**
   Before speaking anything, we know the intention of what we want to convey to the other person. The same thing is implemented in the communicating systems. This makes communication valid and relevant from the side of the communicating system.
2. **Generation:**
   After knowing the intention of what is to be conveyed, the system must gather words so that the information can be reached to the user in his very own communicating language. So, the generation of relevant words is done by the system after the intention process.
3. **Synthesis:**
   Once the agent has all the relevant words, yet they have to be uttered in a way that they have some meaning. So, after the generation of words, the formation of meaningful sentences takes places and finally, the agent speaks them out to the user.

Hearer

1. **Perception:**
   In the perception phase, the communicating system perceives what the user has spoken to it. This is a sort of an audio input signal which the agent receives from the user and then this signal is sent for the further processing by the system.
2. **Analysis:**
   After getting the audio input from the user which is a sequence of sentences and phrases, the system tries to analyze them by extracting the meaningful terms out of the sentences by removing

	the articles, connectors and other words which are there only for the sake of sentence formation. 3. **Disambiguation:** This is the most important thing that a communicating system carries out. After the analyzing process, the agent must understand the meaning of the sentences that the user have spoken. So, this understanding phase in which the system tries to derive the meaning of the sentences by removing various ambiguities and errors is known as disambiguation. This is done by understanding the Syntax, Semantics, and Pragmatics of the sentences. 4. **Incorporation:** In incorporation, the system figures out whether the understanding that it has derived out of the audio signal is correct or not. Whether it is meaningful, whether the system should consider it or ask the user for further input for resolving any sort of ambiguity.
3.	Develop the trust and reputation in multi-agent systems and make an effective analysis over it. (15) ➢ Trust and reputation concepts are widely used in various fields of computer science, such as evaluation systems, P2P networks, grid computing, game theory, e-commerce, semantic web, software engineering, web services, and recommendation systems. ➢ Another field in which these techniques have been gaining importance is multiagent systems (MASs), which are formed by autonomous agents that interact to achieve their own goals. To achieve their goals, agents must engage in some social activities, such as cooperation, coordination, negotiation, and conflict resolution. ➢ The execution of such activities can bring many problems if agent A establishes a contract with agent B and B does not do it or executes the task dishonestly. ➢ A trusting relationship must exist between these agents when one needs to delegate a task to another. This relationship is addressed by Castelfranchi and Falcone [1998], who state that the confidence an agent has in the other's behavior is a Mental attitude that will influence future decisions. ➢ Another field in which these techniques have been gaining importance is multiagent systems (MASs), which are formed by autonomous agents that interact to achieve their own goals. ➢ To achieve their goals, agents must engage in some social activities, such as cooperation, coordination, negotiation, and conflict resolution [Wooldridge 2009].

	➢ The execution of such activities can bring many problems if agent A establishes a contract with agent B and B does not do it or executes the task dishonestly. A trusting relationship must exist between these agents when one needs to delegate a task to another. This relationship is addressed by Castelfranchi and Falcone [1998], who state that the confidence an agent has in the other's behavior is a mental attitude that will influence future decisions. ➢ In the e-commerce scenario, the human customer can delegate the negotiation authority to his or her personal agent, who will interact and negotiate with other agents or people to reach an agreement. It is necessary to trust that the agent understands the consumer's needs and has the trade competence, ensuring that he or she will not be exploited or cheated by other agents. In electronic auctions, bidders can collude and pay a low price for products, and afterward, they can resell them for a higher price. On the other hand, in a Vickrey auction, the auctioneer can lie to the winner about the price of the second-highest bidder, forcing him or her to pay more than he or she should [Wooldridge 2009]. As we can realize, trust plays an important role in these scenarios, and trust and reputation mechanisms were built to decrease risk in these kinds of interactions. There are several trust definitions in the literature, and one of the most accepted is given by Gambetta [1988], who defines it as a subjective probability that an agent will perform a particular task as expected.  Fig. 1. Dimensions of trust and reputation models.
4.	Analyse about the planning and acting in the real world is happens and explain it. (15) **Planning** The task of coming up with a sequence of actions that will achieve a goal is called **planning.**

- ***Planning*** *is a search problem that requires to find an efficient* ***sequence of actions*** *that transform a system from a given starting state to the* ***goal state***

Classical Planning Environments

we consider only environments that are fully observable, deterministic, finite, static (change happens only when the agent acts), and discrete (in time, action, objects, and effects). These are called **classical planning** environments. In contrast, nonclassical planning is for partially observable or stochastic environments and involves a different set of algorithms and agent designs.

Action Schema

An **action schema** represents a number of different actions that can be derived by instantiating the variables p, from, and to to different constants.
In general, an action schema consists of three parts:

(1) The **action name** and **parameter list**-
for example, Fly(p, from, to) - serves to identify the action.

(2) The **precondition** is a conjunction of function-free positive literals stating wlliat must be true in a state before the action can be executed. Any variables in the precondition must also appear in the action's parameter list.

(3) The **effect** is a conjunction of function-free literals describing how the state changes when the action is executed. A positive literal P in the effect is asserted to be true in the state resulting from the action, whereas a negative literal *P* is asserted to be false. Variables in the effect must also appear in the action's parameter list.

Example: The blocks world
One of the most famous planning domains is known as the **blocks world.** This domain consists of a set of cube-shaped blocks sitting on a table. The blocks can be stacked, but only one block can fit directly on top of another. A robot arm can pick up a block and move it to another position, either on the table or on top of another block. The arm can pick up only one block at a time, so it cannot pick up a block that has another one on it. The goal will always be to build one or more stacks of blocks, specified in terms of what blocks are on top of what other blocks. For example, a goal might be to get block A on B and block *C* on D.
We will use *On(b, x)* to indicate that block *b* is on *x,* where *x* is either another block or the table. The action for moving block *b* from the top of *x* to the top of y will be *Move(b, x,* y).
Now, one of the preconditions on moving *b* is that no other block be on it. In first-order logic, this would be 13 *x On (x , b)* or, alternatively, 'd *x 1 On(x, b).* These could be stated as

preconditions in ADL. We can stay within the STRIPS language, however, by introducing a
new predicate, *Clear(x),* that is true when nothing is on *x.*
The action *Move* moves a block *b* from *x* to *y* if both *b* and *y* are clear. After the move is made, *x* is clear but *y* is not. *A* formal description of *Move* in STRIPS is
Action (Move (b, z, *y)* ,
PRECOND: *On(b, x)* A *Clear(b) A Clear(y),*
EFFECT: *On(b, y)* A *Clear(x) A lOn(b, x) A lClear(y))* .
Unfortunately, this action does not maintain *Clear* properly when *x* or y is the table. When
x = *Table,* this action has the effect *Clear(Table),* but the table should not become clear, and
when *y = Table,* it has the precondition *Clear(Table),* but the table does not have to be clear
to move a block onto it. To fix this, we do two things. First, we introduce another action to
move a block *b* from x to the table:
Action(MoveTo Table(b, x) ,
PRECOND: *On(b, x)* A *Clear(b),*
EFFECT: *On(b, Table)* A *Clear(x) A lOn(b, x))* .
Second, we take the interpretation of *Clear(b)* to be "there is a clear space on *b* to hold a block." Under this interpretation, *Clear(Table)* will always be true. The only problem is that
nothing prevents the planner from using *Move (b, x, Table)* instead of *Move To Table (b, x)* .
We could live with this problem-it will lead to a larger-than-necessary search space, but will not lead to incorrect answers-or we could introduce the predicate *Block* and add *Bloctk(b)* A*Block(y)* to the precondition of *Move.*
Finally, there is the problem of spurious actions such as *Move(B, C,* C), which should be a no-op, but which has contradictory effects. It is common to ignore such problems, because they seldom cause incorrect plans to be produced. The correct approach is add inequality
preconditions as shown in Figure 11.4.

$$
\begin{array}{l}
Init(On(A, Table) \wedge On(B, Table) \wedge On(C, Table) \\
\quad \wedge Block(A) \wedge Block(B) \wedge Block(C) \\
\quad \wedge Clear(A) \wedge Clear(B) \wedge Clear(C)) \\
Goal(On(A, B) \wedge On(B, C)) \\
Action(Move(b, x, y), \\
\quad \text{PRECOND: } On(b, x) \wedge Clear(b) \wedge Clear(y) \wedge Block(b) \wedge \\
\qquad (b \neq x) \wedge (b \neq y) \wedge (x \neq y), \\
\quad \text{EFFECT: } On(b, y) \wedge Clear(x) \wedge \neg On(b, x) \wedge \neg Clear(y)) \\
Action(MoveToTable(b, x), \\
\quad \text{PRECOND: } On(b, x) \wedge Clear(b) \wedge Block(b) \wedge (b \neq x), \\
\quad \text{EFFECT: } On(b, Table) \wedge Clear(x) \wedge \neg On(b, x))
\end{array}
$$

Figure 11.4 A planning problem in the blocks world: building a three-block tower. One solution is the sequence $[Move(B, Table, C), Move(A, Table, B)]$.