Chapter 1: Artificial Intelligence and Agents

1 What is Artificial Intelligence?

AI is the study of the design of intelligent computational agents. The field studies the synthesis and analysis of computational agents that act intelligently.

An agent acts intelligently when:

- what it does is appropriate for its circumstances and its goals, taking into account the short-term and long-term consequences of its actions
- it is flexible to changing environments and changing goals
- it learns from experience
- it makes appropriate choices given its perceptual and computational limitations

Agent: something that acts in an environment; it does something.
Computational agent: an agent whose decisions about its actions can be explained in terms of computation.

The *central scientific goal* of AI is to understand the principles that make intelligent behavior possible in natural or artificial systems. This is done by

- the analysis of natural and artificial agents
- formulating and testing hypotheses about what it takes to construct intelligent agents, and
- designing, building, and experimenting with computational systems that perform tasks commonly viewed as requiring intelligence.

As part of science, researchers build empirical systems to test hypotheses or to explore the space of possible designs. These are quite distinct from applications that are built to be useful for an application domain. AI, in contrast with other sciences which study intelligence, allows for experimentation using different models of intelligence. Thus, agent models can be setup and the outcome behaviors can be compared and studied.

The *central engineering goal* of AI is the design and synthesis of useful, intelligent artifacts. We actually want to build agents that act intelligently.

Winograd schemas have the property that (a) humans can easily disambiguate them and (b) there is no simple grammatical or statistical test that could disambiguate them.

2 A Brief History of Artificial Intelligence

Church–Turing thesis: Any effectively computable function can be carried out on a Turing machine (and so also in the lambda calculus or any of the other equivalent formalisms).

An agent's actions are a function of its abilities, its history, and its goals or preferences. This provides an argument that computation is more than just a metaphor for intelligence; reasoning is computation and computation can be carried out by a computer.

The early [AI] programs concentrated on learning and search as the foundations of the field. It became apparent early that one of the main tasks was how to represent the knowledge required for intelligent action. Before learning, an agent must have an appropriate target language for the learned knowledge.

2.1 Relationship to Other Disciplines

The science of AI could be described as "synthetic psychology," "experimental philosophy," or "computational epistemology" — epistemology is the study of knowledge.

AI is intimately linked with the discipline of computer science because the study of computation is central to AI. It is essential to understand algorithms, data structures, and combinatorial complexity to build intelligent machines. Much of computer science started as a spinoff from AI, from timesharing to computer algebra systems.

AI can be seen as coming under the umbrella of cognitive science. Cognitive science links various disciplines that study cognition and reasoning, from psychology to linguistics to anthropology to neuroscience. AI distinguishes itself within cognitive science by providing tools to build intelligence rather than just studying the external behavior of intelligent agents or dissecting the inner workings of intelligent systems.

3 Agents Situated in Environments

Definitions:
Agent: a coupling of perception, reasoning, and acting
World: an agent together with its environment

At any time, what an agent does depends on; Fig. 1:

- prior knowledge about the agent and the environment
- history of interaction with the environment, which is composed of
  - stimuli received from the current environment, which can include observations about the environment, as well as actions that the environment imposes on the agent, and
  - past experiences of previous actions and stimuli, or other data, from which it can learn
- goals that it must try to achieve or preferences over states of the world
- abilities, the primitive actions the agent is capable of carrying out.

Purposive agents have preferences or goals. They prefer some states of the world to other states, and they act to try to achieve the states they prefer most. The non-purposive agents are grouped together and called nature.

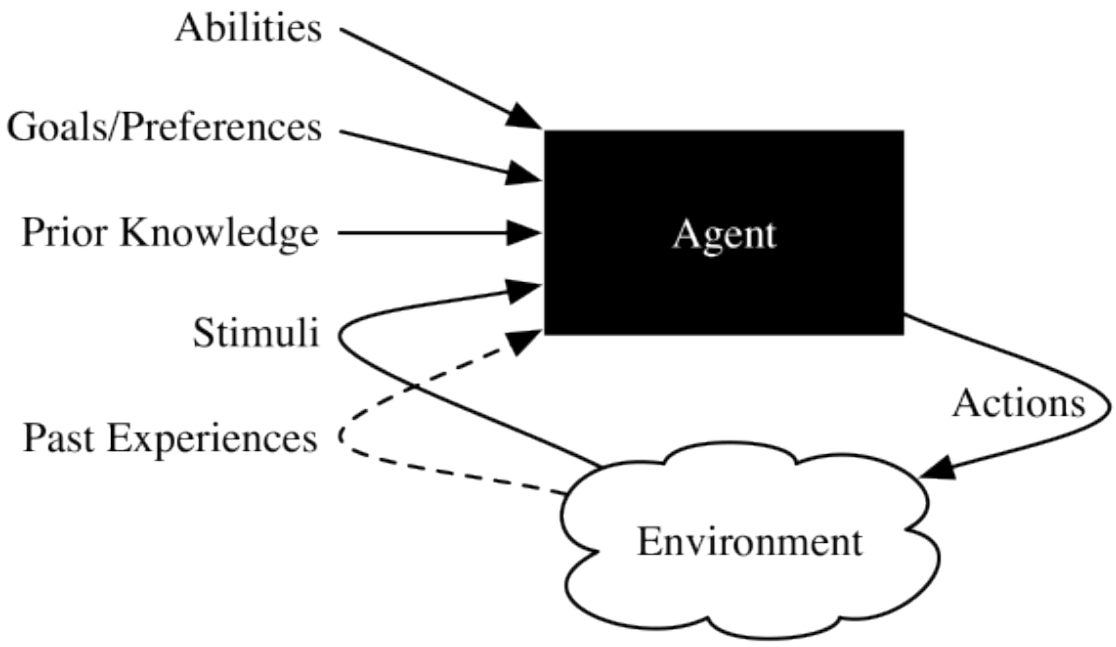

Figure 1: An agent interacting with an environment

4 Designing Agents

4.1 Design Time, Offline and Online Computation

In deciding what an agent will do, there are three aspects of computation that must be distinguished: (1) the computation that goes into the design of the agent, (2) the computation that the agent can do before it observes the world and needs to act, and (3) the computation that is done by the agent as it is acting.

Design time computation is the computation that is carried out to design the agent. It is carried out by the designer of the agent, not the agent itself.

Offline computation is the computation done by the agent before it has to act. It can include compilation and learning. Offline, an agent can take background knowledge and data and compile them into a usable form called a knowledge base. Background knowledge can be given either at design time or offline.

Online computation is the computation done by the agent between observing the environment and acting in the environment. A piece of information obtained online is called an observation. An agent typically must use its knowledge base, its beliefs and its observations to determine what to do next.

One of the advantages of simplifying environments is that it may enable us to prove properties of agents or to optimize agents for particular situations. Proving properties or optimization typically requires a model of the agent and its environment.

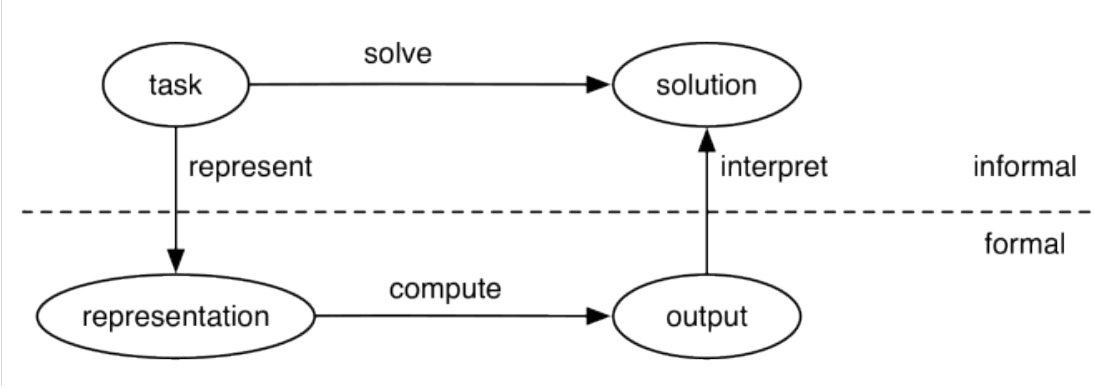

Figure 2: The role of representations in solving tasks

4.2 Tasks

To solve a task, the designer of a system must:

- determine what constitutes a solution
- represent the task in a way a computer can reason about
- use the computer to compute an output, which is answers presented to a user or actions to be carried out in the environment, and
- interpret the output as a solution to the task.

Knowledge is the information about a domain that can be used to solve tasks in that domain. To solve many tasks requires much knowledge, and this knowledge must be represented in the computer. As part of designing a program to solve tasks, we must define how the knowledge will be represented. A **representation language** is used to express the knowledge that is used in an agent. A **representation** of some piece of knowledge is the particular data structures used to encode the knowledge so it can be reasoned with. A **knowledge base** is the representation of all of the knowledge that is stored by an agent.

A good representation language is a compromise among many competing objectives. A representation should be:

- rich enough to express the knowledge needed to solve the task.
- as close to a natural specification of the task as possible; it should be compact, natural, and maintainable. It should be easy to see the relationship between the representation and the domain being represented, so that it is easy to determine whether the knowledge represented is correct. A small change in the task should result in a small change in the representation of the task.
- amenable to efficient computation, or tractable, which means that the agent can act quickly enough. To ensure this, representations exploit features of the task for computational gain and trade off accuracy and computation time.
- able to be acquired from people, data and past experiences.

4.3 Defining a Solution

Once a designer has an informal description of a task, the first thing to consider is what constitutes a solution. Tasks are typically not very well specified. Much of the work in AI is motivated by commonsense reasoning; we want the computer to be able to reach commonsense conclusions about the unstated assumptions.

Four classes of solutions:

Optimal The one which is the best according to some measure of solution quality, for e.g., utility, which is a general measure of desirability used in decision theory.

Satisficing The one which is good enough according to some description of which solutions are adequate.

Approximately optimal The one whose measure of quality is close to the best that could theoretically be obtained.

Probable The one that, even though it may not actually be a solution to the task, is likely to be a solution.

4.4 Representations

A **symbol** is a meaningful pattern that can be manipulated. Examples of symbols are written words, sentences, gestures, marks on paper, or sequences of bits. A **symbol system** creates, copies, modifies, and destroys symbols. Essentially, a symbol is one of the patterns manipulated as a unit by a symbol system.

Physical symbol system hypothesis: A physical symbol system has the necessary and sufficient means for general intelligent action. (Newell and Simon, 1976)

An agent can use a physical symbol system to model the world. A model of a world is a representation of an agent's beliefs about what is true in the world or how the world changes. The world does not have to be modeled at the most detailed level to be useful. All models are abstractions; they represent only part of the world and leave out many of the details. An agent can have a very simplistic model of the world, or it can have a very detailed model of the world. The level of abstraction provides a partial ordering of abstraction. A lower-level abstraction includes more details than a higher-level abstraction. An agent can have multiple, even contradictory, models of the world. Models are judged not by whether they are correct, but by whether they are useful.

Choosing an appropriate level of abstraction is difficult for the following reasons:
• A high-level description is easier for a human to specify and understand.
• A low-level description can be more accurate and more predictive. Often high-level descriptions abstract away details that may be important for actually solving the task.
• The lower the level, the more difficult it is to reason with. This is because a solution at a lower level of detail involves more steps and many more possible courses of action exist from which to choose.

• An agent may not know the information needed for a low-level description. For example, the delivery robot may not know what obstacles it will encounter or how slippery the floor will be at the time that it must decide what to do.

The following are two levels that seem to be common to both biological and computational entities:

The knowledge level is the level of abstraction that considers what an agent knows and believes and what its goals are. The knowledge level considers what an agent knows, but not how it reasons. Both human and robotic agents are describable at the knowledge level. At this level, you do not specify how the solution will be computed or even which of the many possible strategies available to the agent will be used.

The symbol level is a level of description of an agent in terms of the reasoning it does. To implement the knowledge level, an agent manipulates symbols to produce answers. Many cognitive science experiments are designed to determine what symbol manipulation occurs during reasoning. Whereas the knowledge level is about what the agent believes about the external world and what its goals are in terms of the outside world, the symbol level is about what goes on inside an agent to reason about the external world.

5 Agent Design Space

The following ten dimensions of complexity in the design of intelligent agents define the design space for AI:

1. Modularity
2. Planning Horizon
3. Representation
4. Computational Limits
5. Learning
6. Uncertainty
7. Preference
8. Number of Agents
9. Interaction
10. Interaction of Dimensions

Modularity is the extent to which a system can be decomposed into interacting modules that can be understood separately. In this dimension, the agent is either flat, modular, or hierarchical. Modularity is typically expressed in terms of a hierarchical decomposition. In the modularity dimension, an agent's structure is one of the following:
• flat – there is no organizational structure
• modular – the system is decomposed into interacting modules that can be understood on their own
• hierarchical – the system is modular, and the modules themselves are decomposed into simpler modules, each of which are hierarchical systems or simple components.

The **planning horizon** dimension is how far ahead in time the agent plans. The time points considered by an agent when planning are called stages. In this dimension, the agent is either a non-planning agent (who does not consider the future when it decides what to do), finite horizon planner (who looks for a fixed finite number of stages; the degenerate case of only one time step is greedy or myopic), indefinite horizon planner (who looks ahead some finite, but not predetermined, number of stages), or an infinite horizon planner (who has plans going on forever, often called process).

The **representation dimension** concerns how the world is described. The different ways the world could be are called states. A state of the world specifies the agent's internal state (its belief state) and the environment state. A state may be described in terms of features, where a feature has a value in each state. A proposition is a Boolean feature, which means that its value is either true or false. When describing a complex world, the features can depend on relations and individuals. What we call an individual could also be called a thing, an object or an entity. A relation on a single individual is a property. There is a feature for each possible relationship among the individuals. In the representation dimension, the agent reasons in terms of states, features, or individuals and relations (often called relational representations).

The **computational limits** dimension determines whether an agent has perfect rationality, where an agent reasons about the best action without taking into account its limited computational resources, or bounded rationality, where an agent decides on the best action that it can find given its computational limitations. Computational resource limits include computation time, memory, and numerical accuracy caused by computers not representing real numbers exactly. An anytime algorithm is an algorithm where the solution quality improves with time.

The **learning dimension** determines whether knowledge is given, or knowledge is learned (from data or past experience).

Uncertainty is divided into two dimensions: one for uncertainty from sensing and one for uncertainty about the effects of actions. The sensing uncertainty dimension concerns whether the agent can determine the state from the stimuli:
• Fully observable means the agent knows the state of the world from the stimuli.
• Partially observable means the agent does not directly observe the state of the world. This occurs when many possible states can result in the same stimuli or when stimuli are misleading.
The dynamics in the effect uncertainty dimension can be
• deterministic when the state resulting from an action is determined by an action and the prior state, or
• stochastic when there is only a probability distribution over the resulting states.
This dimension only makes sense when the world is fully observable. If the world is partially observable, a stochastic system can be modeled as a deterministic system where the effect of an action depends on some unobserved feature. It is a separate dimension because many of the frameworks developed are for the fully observable, stochastic action case.

The **preference dimension** considers whether the agent has goals or richer preferences:
• A goal is either an achievement goal, which is a proposition to be true in some final state, or a maintenance goal, a proposition that must be true in all visited states. For example, the goals for a robot may be to deliver a cup of coffee and a banana to Sam, and not to make a mess or hurt anyone.

• Complex preferences involve trade-offs among the desirability of various outcomes, perhaps at different times. An ordinal preference is where only the ordering of the preferences is important. A cardinal preference is where the magnitude of the values matters. For example, an ordinal preference may be that Sam prefers cappuccino over black coffee and prefers black coffee over tea. A cardinal preference may give a trade-off between the wait time and the type of beverage, and a mess versus taste trade-off, where Sam is prepared to put up with more mess in the preparation of the coffee if the taste of the coffee is exceptionally good.

Taking the point of view of a single agent, the **number of agents dimension** considers whether the agent explicitly considers other agents:
• Single agent reasoning means the agent assumes that there are no other agents in the environment or that all other agents are part of nature, and so are non-purposive. This is a reasonable assumption if there are no other agents or if the other agents are not going to change what they do based on the agent's action.
• Multiple agent reasoning (or multiagent reasoning) means the agent takes the reasoning of other agents into account. This occurs when there are other intelligent agents whose goals or preferences depend, in part, on what the agent does or if the agent must communicate with other agents. Reasoning in the presence of other agents is much more difficult if the agents can act simultaneously or if the environment is only partially observable.

The **interaction** dimension considers whether the agent does
• offline reasoning where the agent determines what to do before interacting with the environment, or
• online reasoning where the agent must determine what action to do while interacting in the environment, and needs to make timely decisions.

Interaction of the Dimensions

Dimension	Values
Modularity	flat, modular, hierarchical
Planning horizon	non-planning, finite stage, indefinite stage, infinite stage
Representation	states, features, relations
Computational limits	perfect rationality, bounded rationality
Learning	knowledge is given, knowledge is learned
Sensing uncertainty	fully observable, partially observable
Effect uncertainty	deterministic, stochastic
Preference	goals, complex preferences
Number of agents	single agent, multiple agents
Interaction	offline, online

Sensing uncertainty probably has the greatest impact on the complexity of reasoning. It is much easier for an agent to reason when it knows the state of the world than when it does not. Although sensing uncertainty with states is well understood, sensing uncertainty with individuals and relations is an active area of current research.

Two of these dimensions, modularity and bounded rationality, promise to make reasoning more efficient. Although they make the formalism more complicated, breaking the system into smaller

components, and making the approximations needed to act in a timely fashion and within memory limitations, should help build more complex systems.

6 Review

The following are the main points you should have learned from this chapter:
• Artificial intelligence is the study of computational agents that act intelligently.
• An agent acts in an environment and only has access to its abilities, its prior knowledge, its history of stimuli, and its goals and preferences.
• A physical symbol system manipulates symbols to determine what to do.
• A designer of an intelligent agent should be concerned about modularity, how to describe the world, how far ahead to plan, uncertainty in both perception and the effects of actions, the structure of goals or preferences, other agents, how to learn from experience, how the agent can reason while interacting with the environment, and the fact that all real agents have limited computational resources.
• To solve a task by computer, the computer must have an effective representation with which to reason.
• To know when it has solved a task, an agent must have a definition of what constitutes an adequate solution, such as whether it has to be optimal, approximately optimal, or almost always optimal, or whether a satisficing solution is adequate.
• In choosing a representation, an agent designer should find a representation that is as close as possible to the task, so that it is easy to determine what is represented and so it can be checked for correctness and be able to be maintained. Often, users want an explanation of why they should believe the answer.