Chapter 2: Agent Architectures and Hierarchical Control

1 Agents

Agents interact with the environment with a body. An embodied agent has a physical body. Agents receive information through their **sensors**. Agents act in the world through their **actuators**, also called effectors.

2 Agent Systems

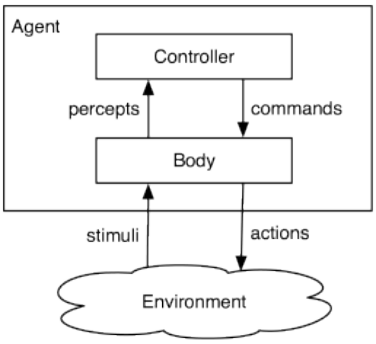

Figure 1: An agent system and its components

An agent system = an agent + its environment
An agent = a body + a controller
A body = sensors (to convert stimuli into percepts) + actuators (to convert commands into actions)

An agent receives stimuli from the environment and carries out actions within it. The controller receives percepts from the body and sends commands to the body. The controller is the brain of the agent.

2.1 The agent function

Let T be the set of time points. Assume T is ordered and has a measure – i.e., it can be mapped to some subset of a real line. T is discrete if there are only a finite number of time points between any two time points. T is dense if there is always another time point between any two time points; this implies there must be infinitely many time points between any two points.

Let P be the set of all possible percepts. A **percept trace**, or **percept stream**, is a function from T into P. It specifies what is observed at each time. Let C be the set of all commands. A **command trace** is a function from T into C. It specifies the command for each time point.

A percept trace is the sequence of all past, present, and future percepts received by the controller. A command trace is the sequence of all past, present, and future commands issued by the controller. The commands can be a function of the history of percepts. This gives rise to the concept of a **transduction**, a function from percept traces into command traces. A transduction is causal if, for all times t, the command at time t depends only on percepts up to and including time t. The causality restriction is needed because agents are situated in time; their command at any time cannot depend on future percepts. A controller is an implementation of a causal transduction. The history of an agent at time t is the percept trace of the agent for all times before or at time t and the command trace of the agent before time t. Thus, a causal transduction maps the agent's history at time t into the command at time t. It can be seen as the most general specification of a controller.

The **memory** or **belief state** of an agent at time t is all the information the agent has remembered from the previous times.

A controller maintains the agent's belief state and determines what command to issue at each time. The information it has available when it must do this are its belief state and its current percepts.

A **belief state transition function** for discrete time is a function:
$remember : S \times P \to S$
where S is the set of belief states and P is the set of possible percepts;
$s_{t+1} = remember(s_t, p_t)$ means that s_{t+1} is the belief state following belief state s_t when p_t is observed.

A **command function** is a function:
$command : S \times P \to C$
where S is the set of belief states, P is the set of possible percepts, and C is the set of possible commands;
$c_t = command(s_t, p_t)$ means that the controller issues command c_t when the belief state is s_t and when p_t is observed.

The belief-state transition function and the command function together specify a causal transduction for the agent. Note that a causal transduction is a function of the agent's history, which the agent does not necessarily have access to, but a command function is a function of the agent's belief state and percepts, which it does have access to.

If there are a finite number of possible belief states, the controller is called a finite state controller or a finite state machine. A **factored representation** is one in which the belief states, percepts, or commands are defined by features. If there are a finite number of features, and each feature can only have a finite number of possible values, the controller is a factored finite state machine. A controller that has an unbounded but countable number of states can compute anything that is computable by a Turing machine.

3 Hierarchical Control

In a hierarchical controller there can be multiple channels – each representing a feature – between layers and between layers at different times.

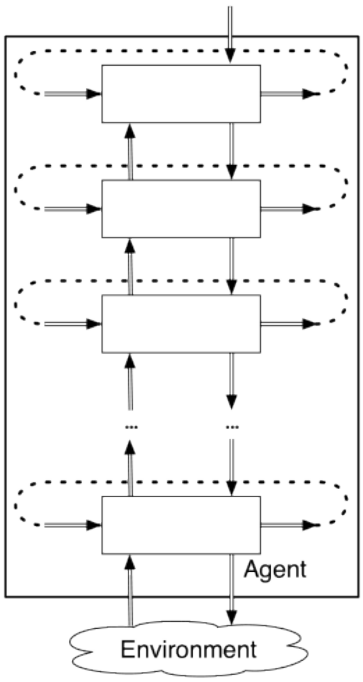

Figure 2: An idealized hierarchical agent system architecture. The unlabeled rectangles represent layers, and the double lines represent information flow. The dashed lines show how the output at one time is the input for the next time.

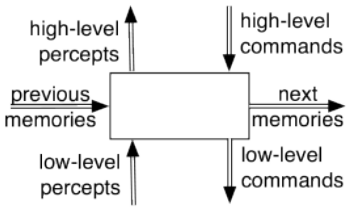

Figure 3: A single layer within an hierarchical agent.

There are three types of inputs to each layer at each time:

- the features that come from the belief state, which are referred to as the remembered or previous values of these features
- the features representing the percepts from the layer below in the hierarchy
- the features representing the commands from the layer above in the hierarchy

There are three types of outputs from each layer at each time:

- the higher-level percepts for the layer above
- the lower-level commands for the layer below

- the next values for the belief-state features

High-level reasoning is often discrete and qualitative, whereas low-level reasoning is often continuous and quantitative. A controller that reasons in terms of both discrete and continuous values is called a hybrid system.

Qualitative reasoning is important because (1) the agent may not know the exact quantitative values (for some feature of the world), (2) the reasoning may be applicable regardless of the actual value, and (3) the agent may do qualitative reasoning to determine which quantitative laws are applicable. Qualitative reasoning uses discrete values: *landmarks* make qualitative distinctions between the modeled individuals, *orders-of-magnitude reasoning* involves approximations, and *qualitative derivatives* which indicate whether some value is increasing, decreasing, or constant.

4 Acting with Reasoning

4.1 Agents Modeling the World

A **model** of a world is a representation of the state of the world at a particular time and/or the dynamics of the world.

At one extreme, a model may be so good that the agent can ignore its percepts. ... Given the state at one time, and the dynamics, the state at the next time can be predicted. This process is known as dead reckoning. At the other extreme is a purely reactive system that bases its actions on the percepts, but does not update its internal belief state.

If both the noise of forward prediction and sensor noise are modeled, the next belief state can be estimated using Bayes' rule. This is known as **filtering**.

A control problem is **separable** if the best action can be obtained by first finding the best model of the world and then using that model to determine the best action.

4.2 Knowledge and Acting

Knowledge is the information about a domain that is used for acting in that domain. A knowledge-based system is a system that uses knowledge about a domain to act or to solve problems.

Some philosophers have defined knowledge as justified true belief. AI researchers tend to use the terms knowledge and belief more interchangeably: Knowledge: general and persistent information taken to be true over a longer period of time, typically not necessarily true, justified only as being useful Belief: transient information revised based on new information

Knowledge base: this is long-term memory, built offline by a learner and is used online to determine the actions Belief state: this2-10- is short-term memory, which maintains the model of current

environment needed between time steps.

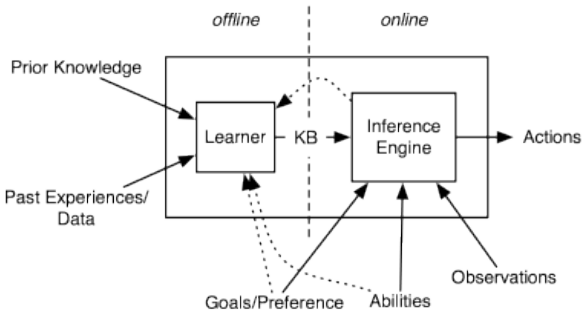

Figure 4: Offline and online decomposition of an agent

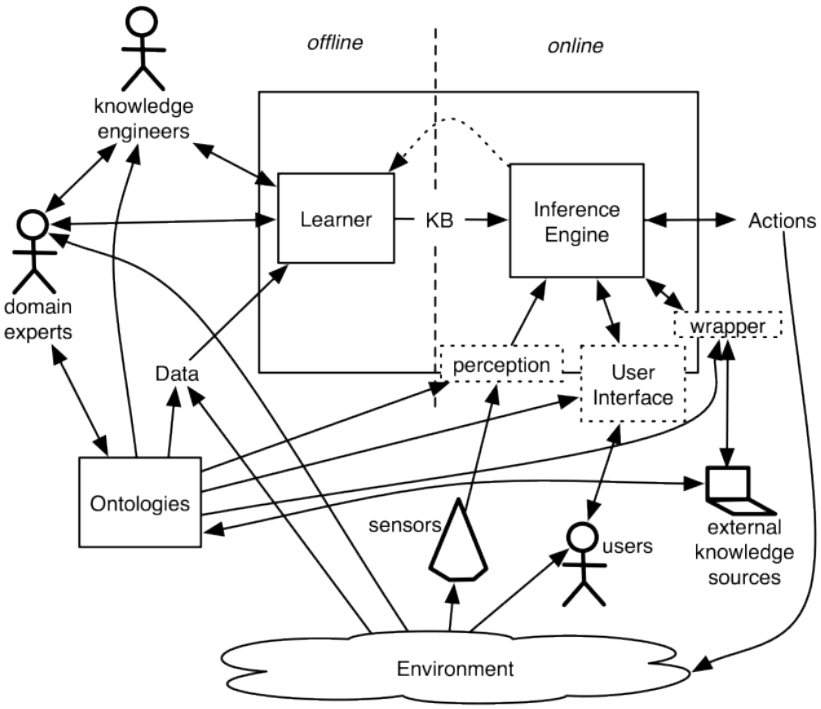

Figure 5: Internals of an agent, showing roles

4.3 Design Time and Offline Computation

In philosophy, ontology is the study of existence. An ontology is a theory about what exists, or what may exist, in a particular domain. In AI, an ontology is a specification of the meaning of the symbols used in an information system, where symbols refer to things that exist. An ontology specifies what exists and the vocabulary used to describe what exists.

The knowledge base is typically built offline from a combination of expert knowledge and data. A **knowledge engineer** is a person who interacts with a **domain expert** to build a knowledge base.

The knowledge engineer knows about intelligent systems, but not necessarily about the domain, and the domain expert knows about the domain, but not necessarily about how to specify knowledge.

4.4 Online Computation

The following roles are involved during online computation:

- A **user** is a person has a need for expertise or has information about individual situations. They are not domain experts, do not know what information is needed, and need a natural interface because they do not understand the internal structure of a system. They need to make informed decisions and need explanations of the recommendations.
- **Sensors** provide information about the environment. A **passive sensor** continuously feeds information to the agent, whereas an **active sensor** is controlled or queried for information.
- An **external knowledge source** can be asked questions and provide the answer for a limited domain. The interface between an agent and an external knowledge source is called a **wrapper**. Having the same symbols mean the same thing is called **semantic interoperability**.

5 Review

The main points you should have learned from this chapter are as follows:

- An agent system is composed of an agent and an environment.
- Agents have sensors and actuators to interact with the environment.
- An agent is composed of a body and interacting controllers.
- Agents are situated in time and must make decisions of what to do based on their history of interaction with the environment.
- An agent has direct access to what it has remembered (its belief state) and what it has just observed. At each point in time, an agent decides what to do and what to remember based on its belief state and its current observations.
- Complex agents are built modularly in terms of interacting hierarchical layers.
- An intelligent agent requires knowledge that is acquired at design time, offline or online.