Artificial Intelligence

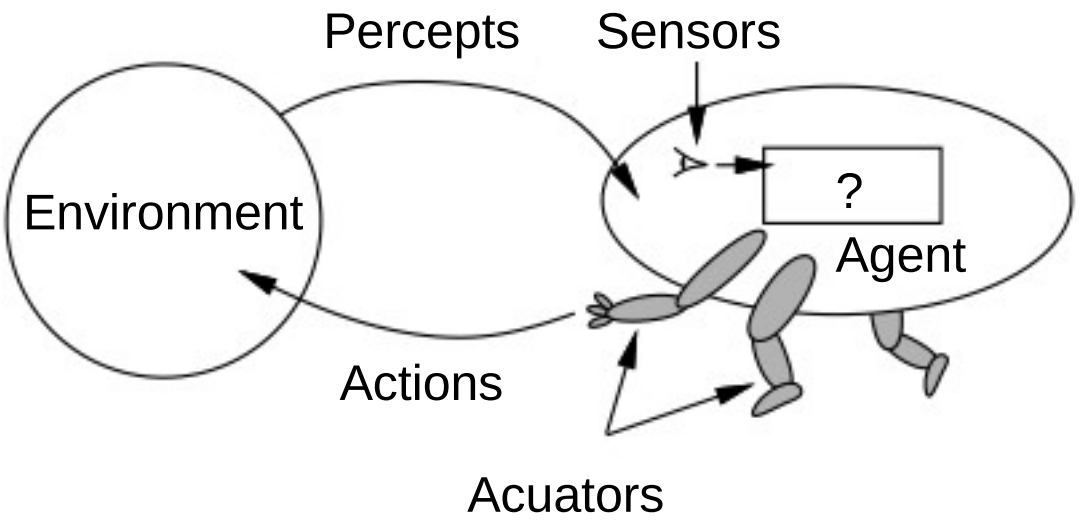
Software Agents

Motivation

- Do tasks that normally human user needs to do automatically.
- Agents are supposed to carry out (probably complex) tasks autonomously.
- Idea: “intelligent agents” react to changes and act in order to reach specific goals.

Motivation

- The agent perceives the environment via sensors and influences it through actuators.
- Agents can carry out tasks autonomously and react to changes in its environment.



Motivating example

Assuming the tasks a taxi driver has to complete:

- Perception: Camera, speedometer, GPS
- Actions: steer, change gear, break, talk to guest
- Goals: Safe, fast, legal, comfortable ride, maximize profit
- Environment: streets, other actors, guests

Definition

- There are many different definitions from various areas, such as software engineering, classical logics, logic programming, robotic:
  - Genesereth/Ketchpel: A program is a software agent if it communicates correctly in an agent language, such as ACL (Agent Communication Language) or KQML (Knowledge Query and Manipulation Language).
  - BDI (Belief, Desire and Intentions) agents are described by their believes, desires, and intentions. This complies with three modalities of a complex modal logic, which can be found in the data structures of the system.

Definition

- Kowalski follows a traditional approach of logic based agents and uses logic programming for the implementation of agents.
- Shoham‘s definition is more focused: A hard- or software is an agent if one analyses it with the help of mental terms.
- Wooldridge/Jennings consider hard- or software an agent if it is:
  - autonomous (independently follows its goals)
  - social (is cooperating with a human or other agents)
  - pro-active (takes initiative) und
  - reactive (perceives its environment and reacts to changes).
- An agent is a computer system capable of autonomous action in some environment in order to meet its design objectives.

Definition

- Autonomous entities that perceive their environment and act upon it.
- Autonomy is the ability to control their own behavior and act without human intervention.
- Agents pursue goals in a such a way as to optimize some given performance measure
- They operate flexibly and rationally in a variety of circumstances
- Does NOT include omniscience, omnipotence, or perfection

Properties

1. Interaction
2. Reactivity
3. Proactiveness
4. Balancing Reactive and Goal-Oriented Behavior
5. Social Ability
6. Mobility
7. Veracity
8. Benevolence
9. Rationality
10. Learning/adaption

Interaction

- Agents may be affected by other agents (including humans) in pursuing their goals
- May take place directly via a *communication language*
- May take place indirectly via the environment
  - Agents sense the actions of other agents and react accordingly

Reactivity

- If a program's environment is guaranteed to be fixed, the program need never worry about its own success or failure – program just executes blindly
  - Example of fixed environment: compiler
- The real world is not like that: things change, information is incomplete. Many (most?) interesting environments are *dynamic*
- Software is hard to build for dynamic domains: program must take into account possibility of failure – ask itself whether it is worth executing!
- 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)

Proactiveness

- Reacting to an environment is easy (e.g., stimulus → response rules)
- But we generally want agents to *do things for us*
- Hence *goal directed behavior*
- Pro-activeness = generating and attempting to achieve goals; not driven solely by events; taking the initiative
- Recognizing opportunities

Balancing Reactive and Goal-Oriented Behavior

- We want our agents to be reactive, responding to changing conditions in an appropriate (timely) fashion
- We want our agents to systematically work towards long-term goals
- These two considerations can be at odds with one another
- Designing an agent that can balance the two remains an open research problem

Social Ability

- The real world is a *multi*-agent environment: we cannot go around attempting to achieve goals without taking others into account
- Some goals can only be achieved with the cooperation of others
- Similarly for many computer environments: witness the Internet
- *Social ability* in agents is the ability to interact with other agents (and possibly humans) via some kind of *agent-communication language*, and perhaps cooperate with others

Other Properties

Other properties, sometimes discussed in the context of agents:

- *mobility*: the ability of an agent to move around an electronic network
- *veracity*: an agent will not knowingly communicate false information
- *benevolence*: agents do not have conflicting goals, and that every agent will therefore always try to do what is asked of it
- *rationality*: agent will act in order to achieve its goals, and will not act in such a way as to prevent its goals being achieved — at least insofar as its beliefs permit
- *learning/adaption*: agents improve performance over time

Abstract Architecture for Agents

- Assume the environment may be in any of a finite set E of discrete, instantaneous states:

$$E = \{e, e', \ldots\}.$$

- Agents are assumed to have a repertoire of possible actions available to them, which transform the state of the environment:

$$Ac = \{\alpha, \alpha', \ldots\}$$

- A *run*, r, of an agent in an environment is a sequence of interleaved environment states and actions:

$$r : e_0 \xrightarrow{\alpha_0} e_1 \xrightarrow{\alpha_1} e_2 \xrightarrow{\alpha_2} e_3 \xrightarrow{\alpha_3} \cdots \xrightarrow{\alpha_{u-1}} e_u$$

Abstract Architecture for Agents

- Let:
  - R be the set of all such possible finite sequences (over E and Ac)
  - R^{Ac} be the subset of these that end with an action
  - R^{E} be the subset of these that end with an environment state

State Transformer Functions

- A *state transformer* function represents behavior of the environment: $\tau : \mathcal{R}^{Ac} \rightarrow \wp(E)$
- Note that environments are…
  - *history dependent*
  - *non-deterministic*
- If $\tau(r)=\varnothing$, then there are no possible successor states to r. In this case, we say that the system has *ended* its run
- Formally, we say an environment Env is a triple $Env = \langle E, e_0, \tau \rangle$ where: E is a set of environment states, $e_0 \in E$ is the initial state, and τ is a state transformer function

Agents

- Agent is a function which maps runs to actions:

$$Ag : \mathcal{R}^E \rightarrow Ac$$

An agent makes a decision about what action to perform based on the history of the system that it has witnessed to date. Let AG be the set of all agents

Systems

- A *system* is a pair containing an agent and an environment
- Any system will have associated with it a set of possible runs; we denote the set of runs of agent Ag in environment Env by $R(Ag, Env)$
- (We assume $R(Ag, Env)$ contains only *terminated* runs)

- Formally, a sequence

$$(e_0, \alpha_0, e_1, \alpha_1, e_2, \ldots)$$

represents a run of an agent Ag in environment $Env = \langle E, e_0, \tau \rangle$ if:

- e_0 is the initial state of Env
- $\alpha_0 = Ag(e_0)$; and
- For $u > 0$,

$$e_u \in \tau((e_0, \alpha_0, \ldots, \alpha_{u-1})) \quad \text{where}$$
$$\alpha_u = Ag((e_0, \alpha_0, \ldots, e_u))$$

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*:

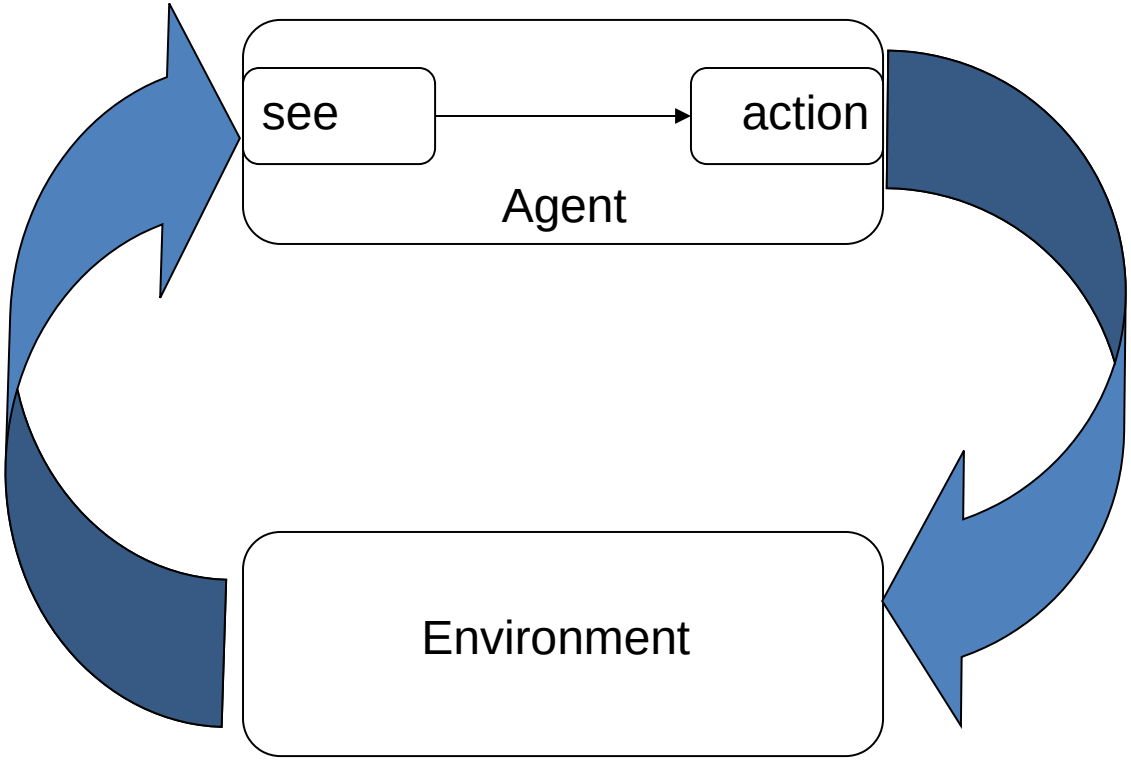
$$action : E \rightarrow Ac$$

- A thermostat is a purely reactive agent

$$action(e) = \begin{cases} \text{off} & \text{if } e = \text{temperature OK} \\ \text{on} & \text{otherwise.} \end{cases}$$

Perception

- Now introduce *perception* system:



Perception

- The *see* function is the agent's ability to observe its environment, whereas the *action* function represents the agent's decision making process
- *Output* of the *see* function is a *percept*:

$$see : E \rightarrow Per$$

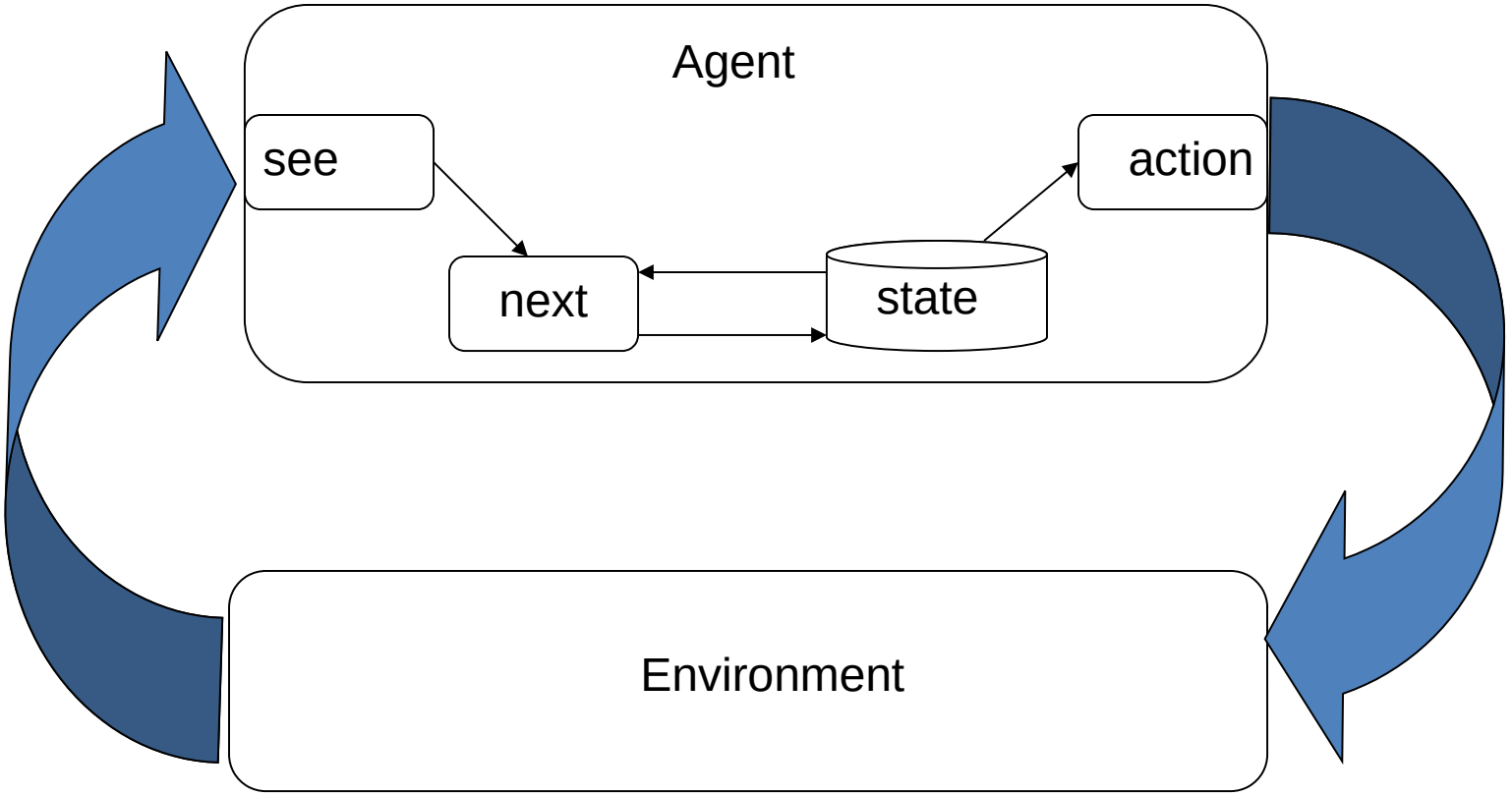
which maps environment states to percepts, and *action* is now a function

$$action : Per^* \rightarrow A$$

which maps sequences of percepts to actions

Agents with State

- We now consider agents that *maintain state*:



Agents with State

- These agents have some internal data structure, which is typically used to record information about the environment state and history.
  Let I be the set of all internal states of the agent.
- The perception function see for a state-based agent is unchanged:

$$see : E \rightarrow Per$$

The action-selection function $action$ is now defined as a mapping

$$action : I \rightarrow Ac$$

from internal states to actions. An additional function $next$ is introduced, which maps an internal state and percept to an internal state:

$$next : I \times Per \rightarrow I$$

Agent Control Loop

1. Agent starts in some initial internal state i_0
2. Observes its environment state e, and generates a percept $see(e)$
3. Internal state of the agent is then updated via $next$ function, becoming $next(i_0, see(e))$
4. The action selected by the agent is $action(next(i_0, see(e)))$
5. Goto 2

Tasks for Agents

- We build agents in order to carry out *tasks* for us
- The task must be *specified* by us…
- But we want to tell agents what to do *without* telling them how to do it

Multi Agent Systems

- Traditional Multiagent Systems
  - Several agents coordinate their knowledge and activities by reasoning about the problem solving process
- Distributed Problem Solving
  - A particular problem is solved by dividing tasks among a number of generally equivalent nodes who divide and share knowledge about the problem
- Modern *multiagent systems* actually cover both

Characteristics of Multiagent Systems

- Each agent has incomplete information
- Control is decentralized
- Data is decentralized
- Computation is asynchronous

Diversity of Multiagent Systems

Agents	Number
	Uniformity
	Goals
	Architecture
	Abilities (sensor & effectors)
Interaction	Frequency
	Persistence
	Level
	Pattern (flow of control)
	Variability
	Purpose
Environment	Predictability
	Accessibility
	Dynamics
	Diversity
	Availability of resources

Common Application Characteristics

- Inherent Distribution
  - Geographically
  - Temporally
  - Semantics – requires different ontologies and languages
  - Functional – requires different cognitive capabilities
- Inherent Complexity
  - Too large to be solved by single, centralized system

Properties of Multiagent Systems

- Speed & Efficiency
- Robustness & Reliability
- Scalability & Flexibility
- Cost
- Distributed Development
- Reusability

Challenging Issues in Multi-Agent Systems

- When and how should agents interact – cooperate and compete – to successfully meet their design objectives?

- Two approaches
  - Bottom up – search for specific agent-level capabilities that result in sufficient group capabilities
  - Top down – search for group-level conventions that appropriately constrain interaction at the agent level

- Leads to several interesting issues …

Challenging Issues in Multi-Agent Systems

1. How to enable agents to decompose their tasks and goals (and allocate sub-goals and sub-tasks to other agents) and synthesize partial results

2. How to enable agents to communicate, what languages and protocols to use

3. How to enable agents to represent and reason about the actions, plans, and knowledge of other agents in order to interact with them

Challenging Issues in Multi-Agent Systems

4. How to enable agents to represent and reason about the state of their interactions

5. How to enable agents to recognize and handle conflicts between agents

6. How to engineer practical multiagent systems

Challenging Issues in Multi-Agent Systems

7. How to effectively balance local computational versus communication
8. How to avoid or mitigate harmful (chaotic or oscillatory) system wide behavior
9. How to enable agents to negotiate and contract with each other
10. How to form and dissolve organizational structures to meet specific goals and objectives

Challenging Issues in Multi-Agent Systems

11. How to formally describe multiagent systems and the interaction between agents and how to ensure multiagent systems are correctly specified
12. How to realize *intelligent processes* such as problem solving, planning, decision making, and learning in a multiagent systems context

Applications of Multiagent Systems

- Electronic commerce
- Real-time monitoring and control of networks
- Modeling and control of transportation systems
- Information handling
- Automatic meeting scheduling

Applications of Multiagent Systems (cont.)

- Industrial manufacturing and production
- Electronic entertainment
- Re-engineering of information flow in large organizations
- Investigation of complex social phenomena such as evolution of roles, norms, and organizational structures

Agents and Objects

- Are agents just objects by another name?
- Object:
  - encapsulates some state
  - communicates via message passing
  - has methods, corresponding to operations that may be performed on this state

Agents and Objects

- Main differences:
  - *agents are autonomous:*
    agents embody stronger notion of autonomy than objects, and in particular, they decide for themselves whether or not to perform an action on request from another agent
  - *agents are smart:*
    capable of flexible (reactive, pro-active, social) behavior, and the standard object model has nothing to say about such types of behavior
  - *agents are active:*
    a multi-agent system is inherently multi-threaded, in that each agent is assumed to have at least one thread of active control

Agents and Expert Systems

- Aren’t agents just expert systems by another name?
- Expert systems typically disembodied ‘expertise’ about some (abstract) domain of discourse (e.g., blood diseases)
- Example: MYCIN knows about blood diseases in humans
  - It has a wealth of knowledge about blood diseases, in the form of rules
  - A doctor can obtain expert advice about blood diseases by giving MYCIN facts, answering questions, and posing queries

Agents and Expert Systems

- Main differences:
  - agents *situated in an environment:*
    MYCIN is not aware of the world — only information obtained is by asking the user questions
  - agents *act:*
    MYCIN does not operate on patients
- Some *real-time* (typically process control) expert systems *are* agents

Intelligent Agents and AI

- When building an agent, we simply want a system that can choose the right action to perform, typically in a limited domain
- We *do not* have to solve *all* the problems of AI to build a useful agent:

  a little intelligence goes a long way!

- Oren Etzioni, speaking about the commercial experience of NETBOT, Inc:
  "We made our agents dumber and dumber and dumber…until finally they made money."

Summary

- Agent preceives the environment and acts.
- Agent = architecture + program
- Ideal agent takes action that maximizes performance at a given perception.
- Actions of an autonomous agent depend on experience.
- Mapping of perceptiuon to actions.
- Reactive agents act on perceptions, goal-oriented agents act to reach a goal, utility-based agents maximize their profit.
- Representation of knowledge!
- Various environments. Most difficult: not accessible, episodic, dynamic, and continuous.

Summary

- Multiagent systems are systems in which multiple interacting agents interact to solve problems
- Key concepts of multiagent systems are agents and agent coordination
- There are many important issues for multiagent systems that attempt to answer when and how to interact with whom

Summary

- Common characteristics of multiagent systems are their inherent distribution and complexity
- Distributed and flexible nature of multiagent systems leads to increased speed, robustness, scalability and reusability