G54DIA:
Designing Intelligent Agents

Lecture 2: Task Environments & Architectures

Natasha Alechina
School of Computer Science
nza@cs.nott.ac.uk

Outline of this lecture

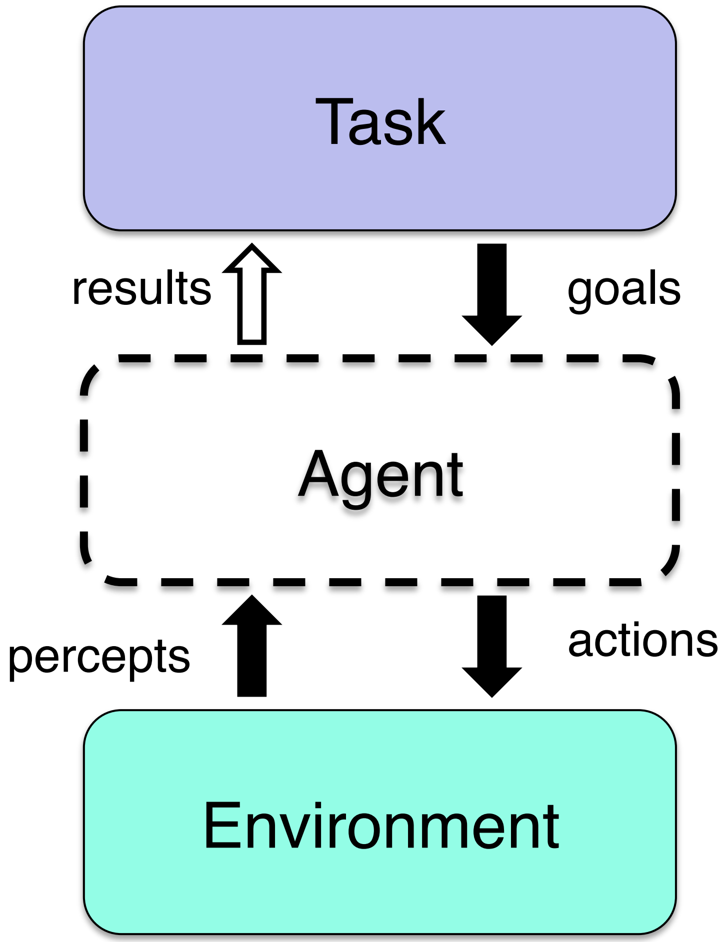
- programming agents & the role of agent architectures
- properties of *task environments*
- examples:
  - MAXIMS: filtering email
  - Fred & Ginger: cooperative robotics
- relationship between task environment and architecture

Designing intelligent agents

- an agent operates in a *task environment*:
  - **task**: the *goal*(*s*) the agent is trying to achieve
  - **environment**: that part of the real world or a computational system 'inhabited' by the agent
- agent obtains information about the environment in the form of *percepts*
- agent changes the environment by performing *actions* to achieve its goals

The task environment defines the problem

- we can sometimes manipulate the task and/or environment to make things easier
- e.g., increasing the contrast of objects to make things easier for a robot's cameras
- however the task environment is usually given



Specifying an agent to solve the problem

- the behaviour of an agent is described by an *agent function* (*action selection function*) which maps a goal and sequence of percepts to an action (and possibly results)

- agent programming is conventionally conceived as the problem of *synthesising an agent function*

- e.g., Russell & Norvig (2003) claim that:

  "the job of AI is to *design the agent program that implements the agent function mapping percepts to actions*"

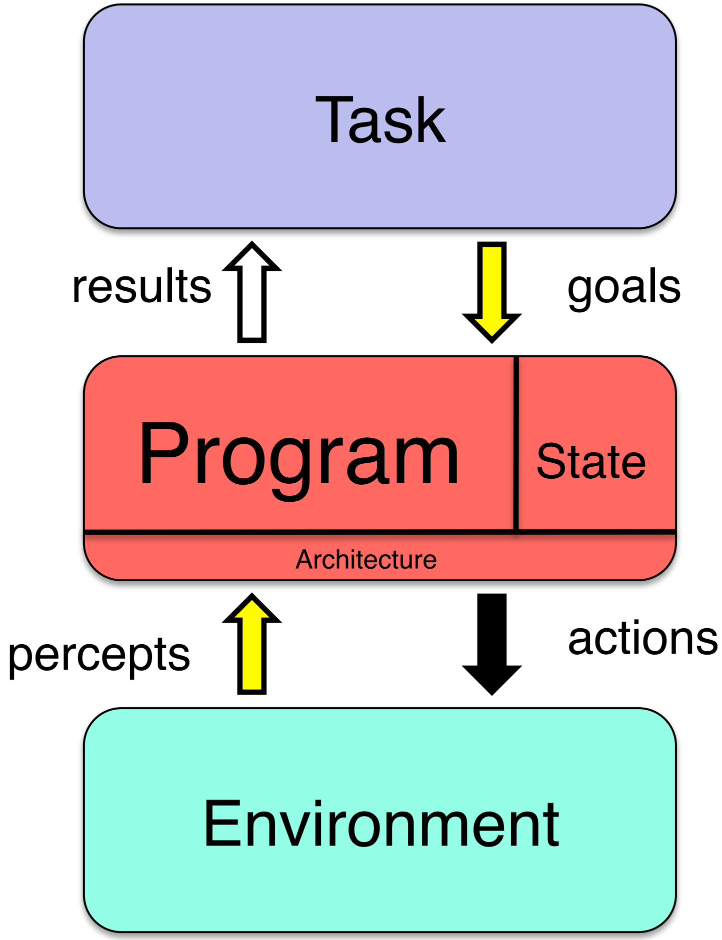
- this is a *very* difficult problem

Russell & Norvig view of an agent

- **program:** implements the agent function mapping from goals & percepts to actions (& results)

- **state:** includes all the internal representations on which the agent program operates

- **architecture:** computing device with sensors and actuators that runs the agent program



Agent architectures

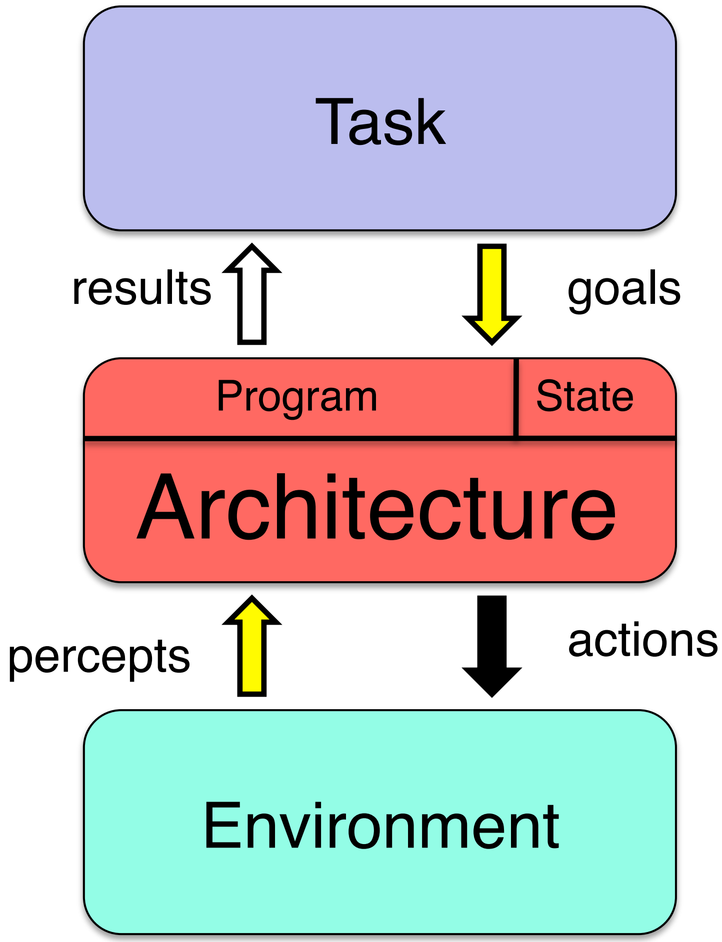
- one way of making the agent programming problem more tractable is make use of the notion of an *agent architecture*
- the notion of an "agent architecture" is ubiquitous in the agent literature but is not well analysed
- often discussed in the context of an agent programming language or platform
- largely ignored by agent text books, e.g. Russell & Norvig (2003) view the architecture as simply "*some sort of computing device with physical sensors and actuators*"

The architecture as a virtual machine

- the *architecture* defines a (real or virtual) machine which runs the agent program
- defines the *atomic operations* of the agent program and implicitly determines the *components* of the agent
- determines which operations happen *automatically*, without the agent program having to do anything
- e.g., the interaction between memory, learning and reasoning
- an architecture constrains kinds of agent programs we can write (easily)

Architectural view of an agent

- **program:** a function mapping from goals & percepts to actions (& results) expressed in terms of virtual machine operations
- **state:** the virtual machine representations on which the agent program operates
- **architecture:** a virtual machine that runs the agent program and updates the agent state



Hierarchies of virtual machines

- in many agents we have a whole hierarchy of virtual machines
  - the agent architecture is usually implemented in terms of a programming language, which in turn is implemented using the instruction set of a particular CPU (or a JVM)
  - likewise some 'agent programs' together with their architecture can implement a new, higher-level architecture (virtual machine)
- used without qualification, 'agent architecture' means the *most abstract* architecture or the *highest level* virtual machine

Software architectures

- this notion of an *agent architecture* is related to the more general notion of *software architecture*
- *software architectures* is the "principled study of the large-scale structures of software systems" (Shaw & Clements 2006)
- "architecture is concerned with the selection of architectural elements, their interactions, and the constraints on those elements and their interactions necessary to provide a framework in which to satisfy the requirements and serve as a basis for the design" (Perry & Wolf 1992)
- standard architectures for many domains and applications, e.g., *n*-tier client-server architectures, service oriented architectures, etc.

Cognitive architectures

- agent architecture is also related to the notion of a cognitive architecture as used in artificial intelligence and cognitive science
- a *cognitive architecture* is an integrated system capable of supporting intelligence
- often used to denote models of human reasoning, e.g., ACT-R, SOAR
- in other cases no claims about psychological plausibility are made
- in this latter sense, cognitive architecture is more or less synonymous with agent architecture as used here

Properties of architectures

- an agent architecture can be seen as defining a *class* of agent programs
- just as individual agent programs have properties that make them more or less successful in a given task environment
- architectures (classes of programs) have higher-level properties that determine their suitability for a task environment
- choosing an *appropriate architecture* can make it much easier to develop an agent program for a particular task environment

Importance of architecture

- focus of this module will mostly be on *agent architectures*:
  - what sorts of architectures there are; and
  - which architectures are appropriate for different tasks and environments
- *to program an agent which is successful in a given task environment, we must choose an architecture which is appropriate for that task environment*

Task environments & architectures

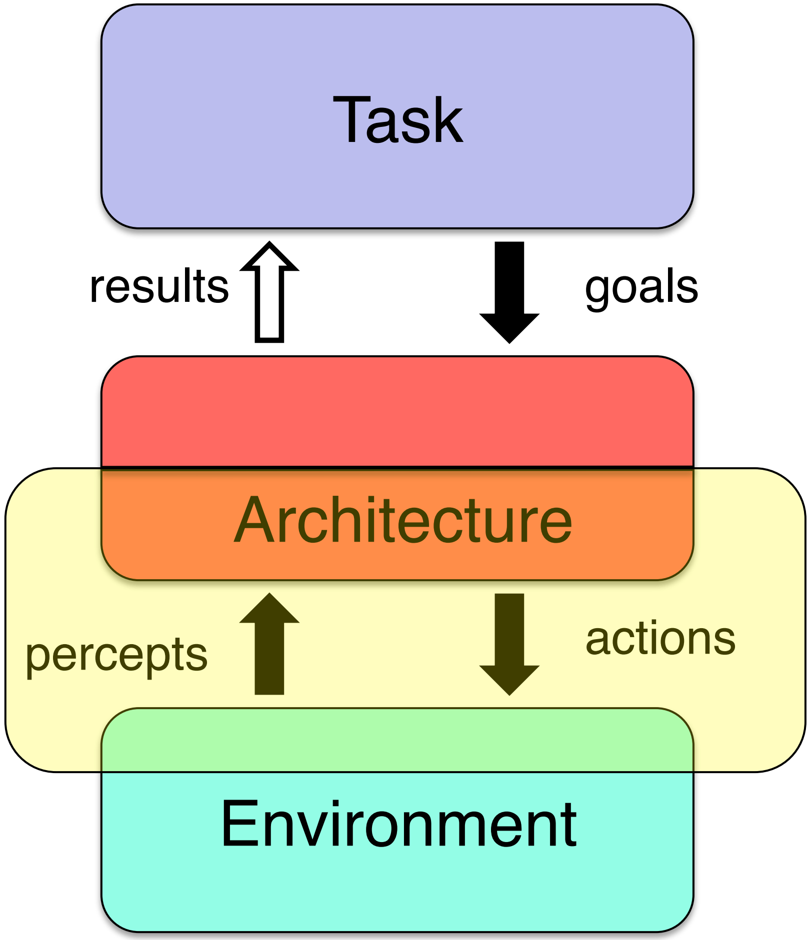
- to choose an architecture which is appropriate for a given task environment we must be able to *characterise* both task environments and architectures

- in this lecture we'll look at some properties of task environments that have an impact on the choice of architecture

- in later lectures we'll look at properties of agent architectures …

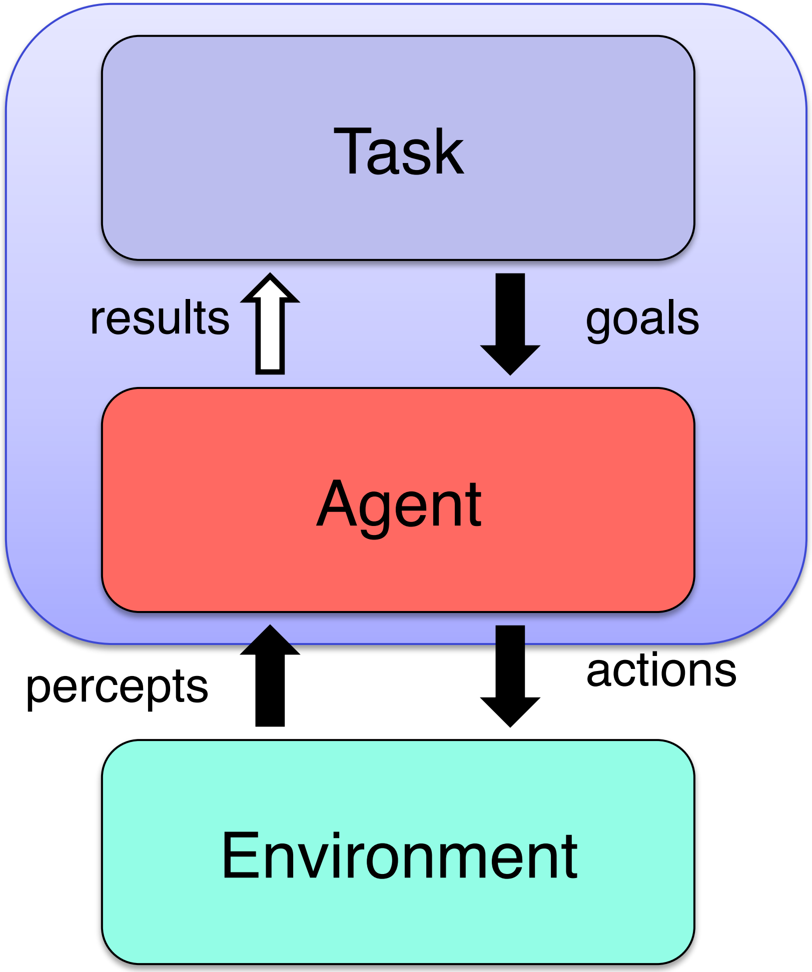
Specifying task & environment

- the task specifies the goals the agent must achieve (and any results required)
- the agent (architecture) and environment jointly determine:
  - the information the agent can obtain (percepts)
  - the actions the agent can perform
- decomposition into task and environment is not always obvious



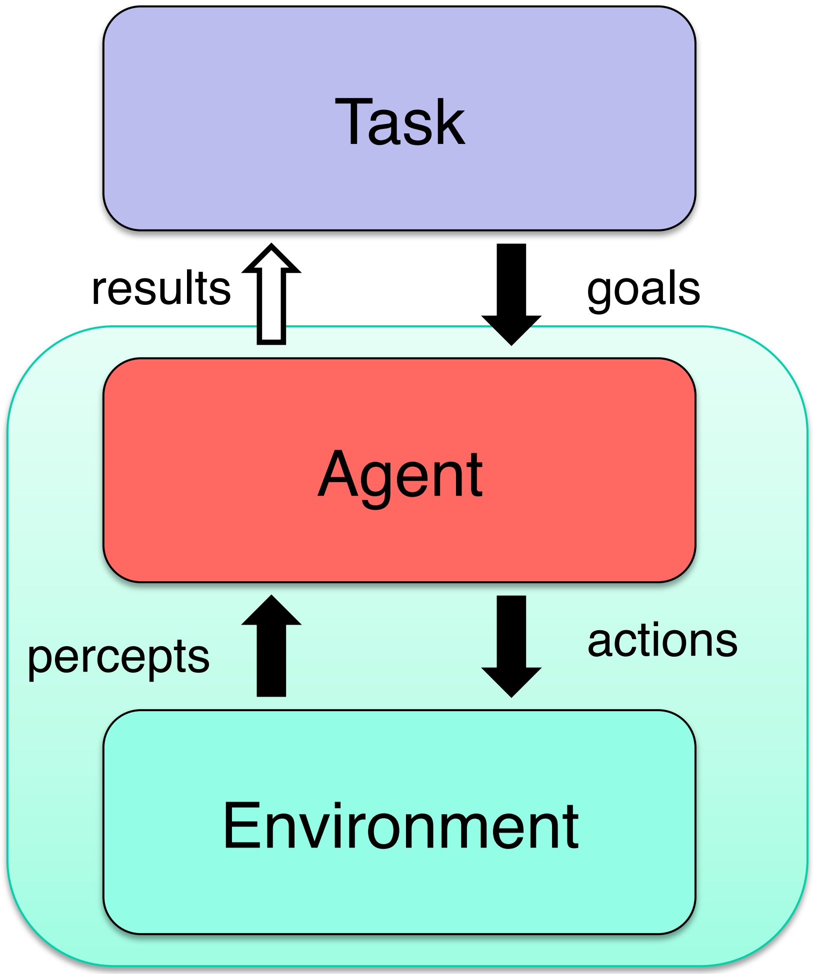
Specifying the task

- some tasks may come from the agent itself (autonomy)
- an agent may generate its *own (top-level) goals*
- e.g., may generate a goal to keep its battery charged
- or it may make use of its own *results*, e.g., in learning



Specifying the environment

- similarly, an agent may be part of its *own environment*
- e.g., if it has sensors monitoring its battery level
- agent may also form part of the environment of *other agents*
- e.g., in a multi-agent system



Task environment classification

- we will classify the agent's task environment based on:
  - **task**: properties of the *goals* that must be achieved (for simplicity, we assume the task does not require the return of results)
  - **environment**: properties of the *percepts* and *actions* possible in the environment
- the following properties are *illustrative* only—for particular types of task environments other properties may be more useful
- however the *approach* can be applied to any task environment

Goals and intentional systems

- we focus on the goals that must be achieved to perform the task (top-level goals), not where the goals come from
- we make no assumptions about the agent's program state or architecture (or any subgoals generated by the agent)
- as we'll see in later lectures, not all agents represent goals explicitly, even though they act in a goal-directed manner
- we assume that it is possible to view the agent as an *intentional system*, i.e., that we can ascribe a set of goals to it which characterise its behaviour

The interaction of agent & environment

- we focus on the *means* available to achieve the task goals–the *percepts* and *actions* available to the agent
  - different agents in the same environment may have different percepts and actions available to them
  - e.g., if one agent has fewer sensors than another or if it can perform only a subset of the actions that the other agent can perform
- we make no assumptions about how the agent represents information from percepts or chooses actions

Properties of the task

- **type of goal:** a goal to achieve a particular state in the environment is termed an *achievement goal*; a goal to maintain or preserve a state in the environment is termed a *maintenance goal*
- **number of goals:** if the agent must achieve multiple goals in parallel, we say the agent has multiple goals, otherwise it has a single goal
- **commitment to goals:** if a goal is only abandoned when it is achieved we say the agent is *strongly committed* to its goal(s); otherwise it is only *weakly committed*
- **utilities of goals:** if the reward for achieving each goal is the same, we say the agent's goals have *equal utility*; otherwise its goals have differing utilities
- **constraints on how goals are achieved:** e.g., deadlines, resource bounds

Properties of the environment (percepts)

- **discrete / continuous:** if there are a limited number of distinct, clearly defined, states of the environment, the environment is discrete; otherwise it is continuous

- **observable / partially observable:** if it is possible to determine the complete state of the environment at each time point from the percepts it is observable; otherwise it is only partially observable

- **static / dynamic:** if the environment only changes as a result of the agent's actions, it is static; otherwise it is dynamic

- **deterministic / nondeterministic:** if the future state of the environment can be predicted *in principle* given the current state and the set of actions which can be performed it is deterministic; otherwise it is nondeterministic

- **single agent / multiple agents:** the environment may contain other agents which may be of the same kind as the agent, or of different kinds

Properties of the environment (actions)

- **fallibility of actions:** an action is *infallible* if it is guaranteed to produce its intended effects when executed in an environment which satisfies the preconditions of the action; otherwise it is *fallible*

- **utility of actions:** the utility of an action is the utility of the state which results from the action–the action with maximum utility is *correct*

- **costs of actions:** the resource cost of performing the action– an action is *optimal* if it is correct and there is no other correct action with lower cost

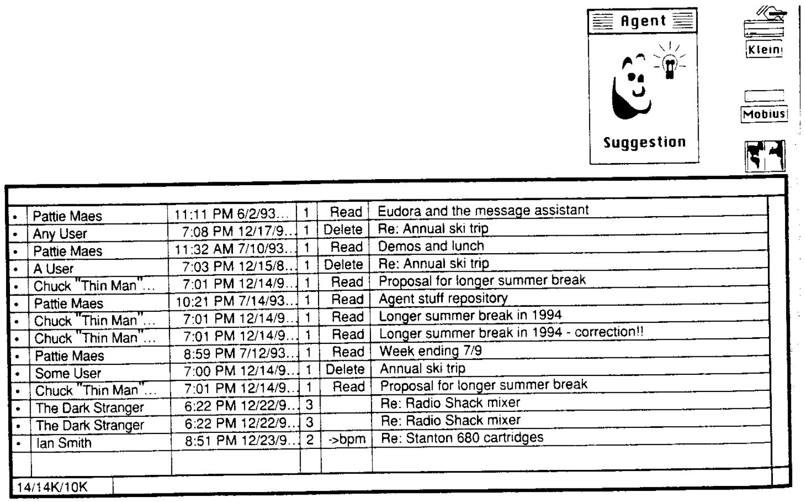
- **communicating actions:** an agent can be said to communicate with other agents in a meaningful way if it interacts with them via some kind of agent communication language

Example: MAXIMS

- agent which learns to prioritise, delete, forward, sort and archive mail messages on behalf of the user

- cooperates with agents of other users

– (Maes 1994)



Example: MAXIMS

- **task:** sequence of achievement goals, strongly committed to its goals, all goals have equal utility, may be constraints on how goals are achieved
- **environment (percepts):** observable, dynamic, deterministic, continuous, may contain other agents
- **environment (actions):** typically infallible, may have differing utilities and costs, may communicate with other agents

Example: Fred & Ginger

- a multi-agent *transportation system*

- two physical robots cooperate to move objects around a 'warehouse'

  – (Coddington & Aylett 1997)

Example: Fred & Ginger

- **task:** agents have both achievement and maintenance goals, and are strongly committed to their goals, goals have differing utility, no constraints on how goals are achieved
- **environment (percepts):** partially observable, static, nondeterministic, continuous, and contains a single (multi-agent) system
- **environment (actions):** fallible, have differing utilities and costs, agents communicate *indirectly* through actions in the environment

Classification can be difficult

- it may not be obvious which features of the agent's task and environment are relevant

- there is often not enough information in published descriptions of systems to perform this kind of classification

  – it is often difficult to tell if two task environments differ

  – it may be difficult to tell if the claimed improvements in performance are due to the architecture or whether they are due to differences in the problem being solved

Task and architecture

- if the agent has at least one *maintenance goal*, then the agent's 'lifetime' is potentially unbounded
- if the agent must pursue *multiple goals* in parallel, it needs some way of choosing which goal it should be working on at the moment
- if the agent is only *weakly committed* to its goals, it needs to be able to be able to decide when it should give up trying to pursue a goal
- if the *utilities* of the agent's goals differ, then its commitment to a goal may change and/or the agent needs to be able to determine which goal it should be pursuing at any given time

Percepts and architecture

- *discrete environments* are usually easier to represent than continuous ones
- if the environment is only *partially observable*, internal state is required to keep track of the current state of the world
- if the environment is *dynamic*, then the time it takes the agent to choose which action to perform becomes important
- if the environment is *nondeterministic*, then any representation or reasoning capabilities will probably require a notion of uncertainty

Actions and architecture

- if actions are *infallible*, the agent does not need to monitor the environment to tell whether an action has succeeded
- if actions have varying *costs* and/or *utilities* and the agent wants to minimise cost or maximise utility, it needs to be able to choose between alternative courses of action
- if the agent can *communicate* with other agents, it must decide what to communicate and when, and what to do with information it receives from other agents

Summary

- an architecture-neutral classification of agents and task environments allows us to:
  - map out what sorts of agents there *are* or *might be*
  - compare different architectures for the *same* task, and the same architecture for *different* tasks
  - state empirical *generalisations* of what works and what doesn't, for example, whether the same architecture can be used in different environments

The next lecture

Reactive Architectures I

Suggested reading:

- Braitenberg (1984), *Vehicles: Experiments in Synthetic Psychology*, MIT Press