Lecture 8
Intelligent Agents as a Framework for AI: Part II

Rob Gaizauskas

Lecture Outline

- Review of Key Terminology
- Structures of Agents
  - Agent Programs + Table-driven agents
  - Simple Reflex Agents
  - Model-based Reflex Agents
  - Goal-based Agents
  - Utility-based Agents
  - Learning Agents

- Reading: Russell & Norvig, Chapter 2: Intelligent Agents”

Structure of Agents

- Recall distinction between agent function and agent program
- Job of AI is to design an agent program that implements an agent function
- The implemented program will run on some computing device with (physical sensors) and actuators = the architecture

Agent = architecture + program

Agent Programs

- Agent programs all take the form: given an input from the sensors, here is an action (set of actions) for the actuators
- Note: while the agent function may require access to the entire percept sequence, the agent program only takes current percept as input as that is all the environment supplies
  - If previous parts of the percept sequence are required, agent needs to remember them

Table-Driven Agents

```
function TABLE-DRIVEN-AGENT(percept) returns an action
    persistent: percepts, a sequence, initially empty
                table, a table of actions, indexed by percept sequences, initially fully specified

    append percept to the end of percepts
    action ← LOOKUP(percepts, table)
    return action
```

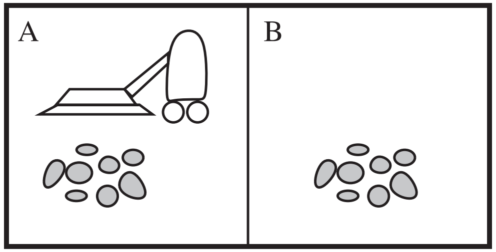
- This simple agent program stores the percept sequence and uses it to index into the percept-action table to determine what to do
  - Vacuum cleaner agent is an example of this agent type
- Note: designing such an agent requires the designer to specify the appropriate action for every possible percept sequence

Table-Driven Agents (cont)

- Table-driven agent approach is doomed to failure
- Suppose *P* is the set of possible percepts and *T* is the lifetime of the agent (= total of percepts it will receive)
- The lookup table for this agent will contain $\sum_{t=1}^{T} |P|^{t}$ entries
- For an agent with a single camera, visual input comes in at ~27 Mb/sec (30 fps, 640x480 pixels with 24 bits of colour)
  – Would require a lookup table with $10^{250,000,000,000}$ entries for one hour's driving

Table-Driven Agents (cont)

- The size of such tables means
  - No physical agent would have space to store such a table
  - No designer would have the time to create the table
  - No agent could ever learn the entries from experience
  - Not clear how a designer would know how to fill table entries
- Nonetheless TABLE-DRIVEN-AGENT does implement the desired agent function.
  - Challenge for AI: how to produce rational behaviour from a reasonable-sized program rather than a huge table
  - There are examples in other areas: vast tables of square roots once used by engineers have been replaced by calculators running a 5 line program for Newton's method
  - R&N: "Can AI do for general intelligent behaviour what Newton did for square roots?"



Simple Reflex Agents

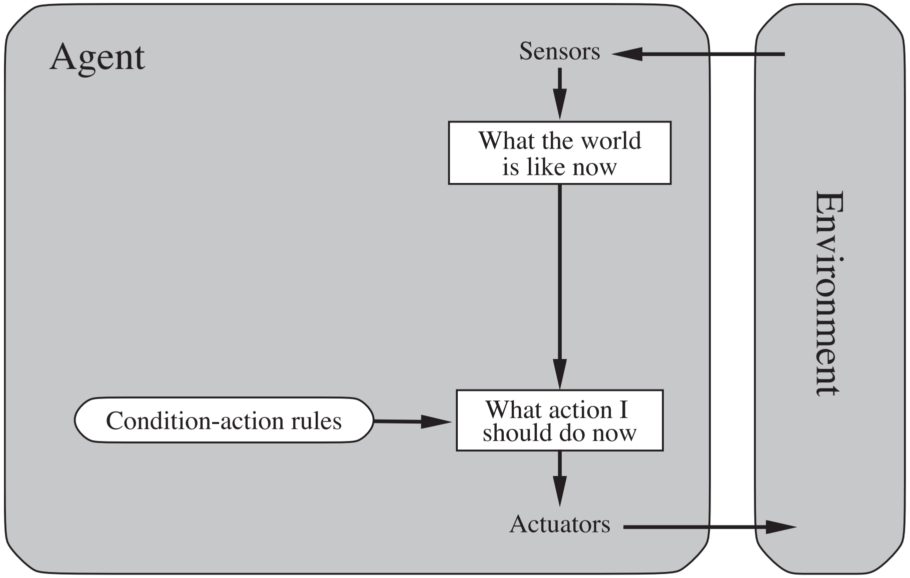
- Simple reflex agents choose actions based on the current percept only – ignore rest of percept history
- Example: vacuum cleaner agent introduced above – acts based only on current square and whether it is dirty
- Can be implemented with a simple program (below)
- By ignoring percept history number of environment states to consider drops from 4^T to 4

function REFLEX-VACUUM-AGENT([*location*,*status*]) **returns** an action

if *status* = *Dirty* **then return** *Suck*
else if *location* = *A* **then return** *Right*
else if *location* = *B* **then return** *Left*

Simple Reflex Agents

- A more general approach to simple reflex agents
  - first build a general a general purpose interpreter for condition-action rules
  - Create rule sets for specific task environments



Rectangles denote current internal state of agent decision process; ovals denote background information used in process

function SIMPLE-REFLEX-AGENT($percept$) **returns** an action
  persistent: $rules$, a set of condition–action rules

  $state \leftarrow$ INTERPRET-INPUT($percept$)
  $rule \leftarrow$ RULE-MATCH($state, rules$)
  $action \leftarrow rule$.ACTION
  return $action$

Simple Reflex Agents

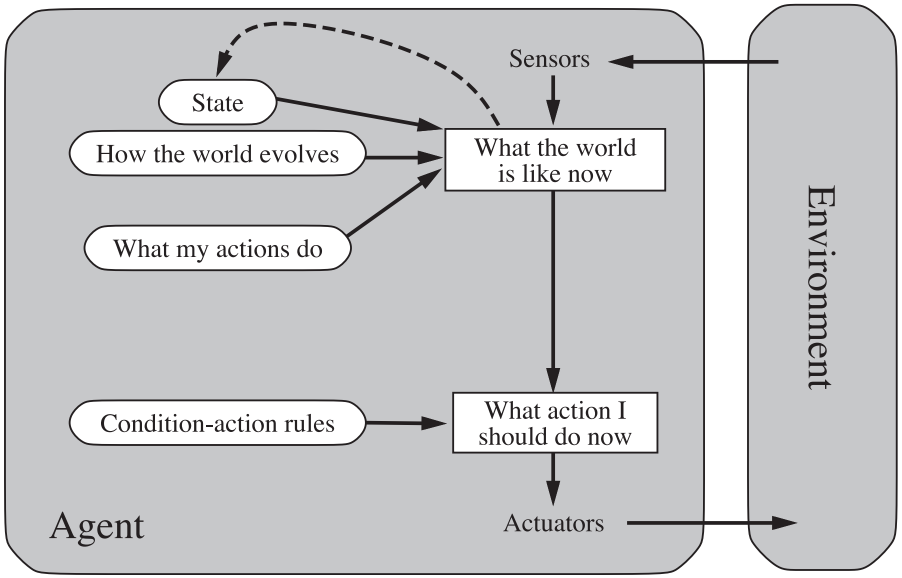
- Simple reflex agents have strength of simplicity; but have limited intelligence
- Only work if the correct decision can be made based on current percept only – i.e. only if environment is fully observable
- Even small amounts of unobservability can cause problems. E.g.:
  - Reflex braking agent relying on single video frame to decide whether to brake could be confused by signal vs brake light
  - Vacuum agent deprived of location sensor could end up in infinite loop (moving right or left)

Model-based Reflex Agents

- To address partial observability agent can keep track of parts of the world it cannot see now
  - i.e. agent should maintain some sort of internal state that depends on percept history – captures some aspects of what is currently unobservable
- To update this internal state over time requires
  - Information about how the world evolves independently of the agent
    - E.g. an overtaking car will be closer now than it was a moment ago
  - Information about how the agent's own actions affect the world
    - E.g. turning the steering rule right causes the car to go right
- Such knowledge is called a model of the world and an agent using such a model is called a model-based agent

Model-based Reflex Agents

- In a model-based reflex agent the current percept is combined with the old internal state to generate an updated state based on the agent's model of how the world works




```
function MODEL-BASED-REFLEX-AGENT(percept) returns an action
  persistent: state, the agent's current conception of the world state
              model, a description of how the next state depends on current state and action
              rules, a set of condition–action rules
              action, the most recent action, initially none

  state ← UPDATE-STATE(state, action, percept, model)
  rule ← RULE-MATCH(state, rules)
  action ← rule.ACTION
  return action
```

Model-based Reflex Agents

- Models and states may be represented in different ways – much of AI explores these differences
- Regardless of what representation is used, agent can rarely determine the current state of a partially observable environment exactly
- Box labelled "what the world is like now" is really just the agent's best guess or guesses
  - Uncertainty about current state may be unavoidable
  - E.g. automated taxi cannot see around large vehicle stopped in front of it

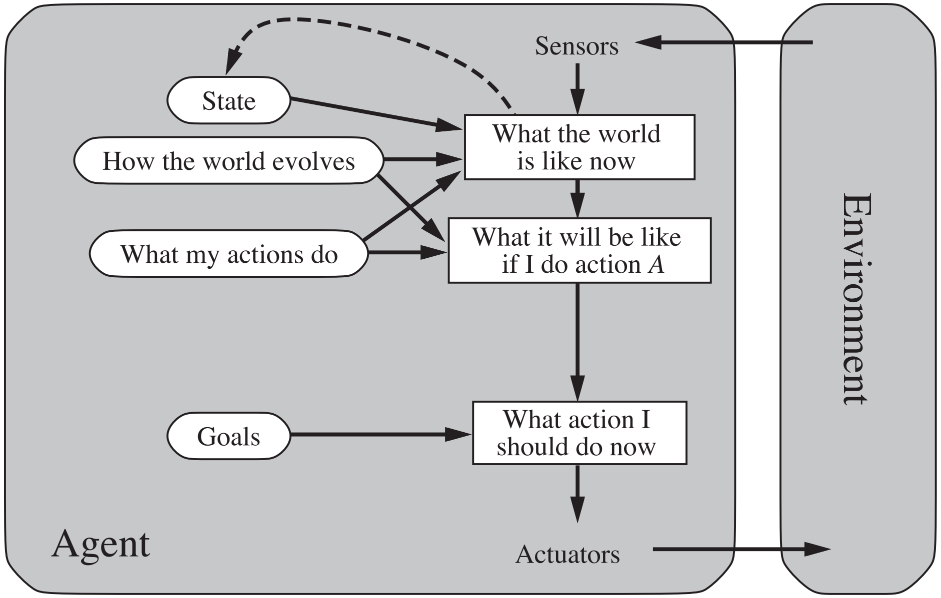
Goal-based Agents

- Knowing about the current state of the environment is generally not sufficient to determine what to do
  - There may be multiple possible actions
- Correct decision depends on goal of the agent
- Given a goal, knowledge of the current state of the environment and a model, an agent can choose actions towards achieving the goal

Goal-based Agents

- Choosing an action is straightforward if the goal is satisfied from a single action
  - Generally a sequence of actions is necessary – searching and planning (core areas of AI) may be necessary to determine such sequences
  - Example: Suppose I want to travel to London
    - First need to decide: train vs bus vs drive
    - Suppose choose train then need to decide
      - Which train
      - How/when purchase tickets (find credit card; log in; book tickets)
      - How/when to get to station …

Goal-based Agents



- Such decision making is very different from the condition-action rules of reflex agents
- Requires consideration of the future
  - "What will happen if I do such-and-such?"
  - "What will make me happy?"
- In reflex agents such information is not explicitly represented

Goal-based Agents

- Goal-based agents are less efficient than reflex agents
- However they are more flexible since knowledge that support decisions is explicit and can be modified
- Can update knowledge of what actions will do
  - E.g. automated taxi can update knowledge of how its brakes will work if it starts to rain
  - For reflex agent many condition-actions rules would need to be rewritten
- Can update goals
  - E.g. automated taxi can change where it goes by specifying a new goal as a destination
  - Reflex agents rules about when to turn etc. will only work for a single destination and must all be replaced for a new destination

Utility-based Agents

- Goals are not sufficient to generate high-quality behaviour in many circumstances
  - Typically a goal will be achievable via many action sequences, but some are better than others
  - E.g. taxi may take many routes to destination, but some are faster, safer, more reliable, cheaper, etc.
- Goals are either achieved or not achieved – need a performance measure that lets agent more adequately assess how "happy" different ways of achieving a goal will make it – notion of utility provides that

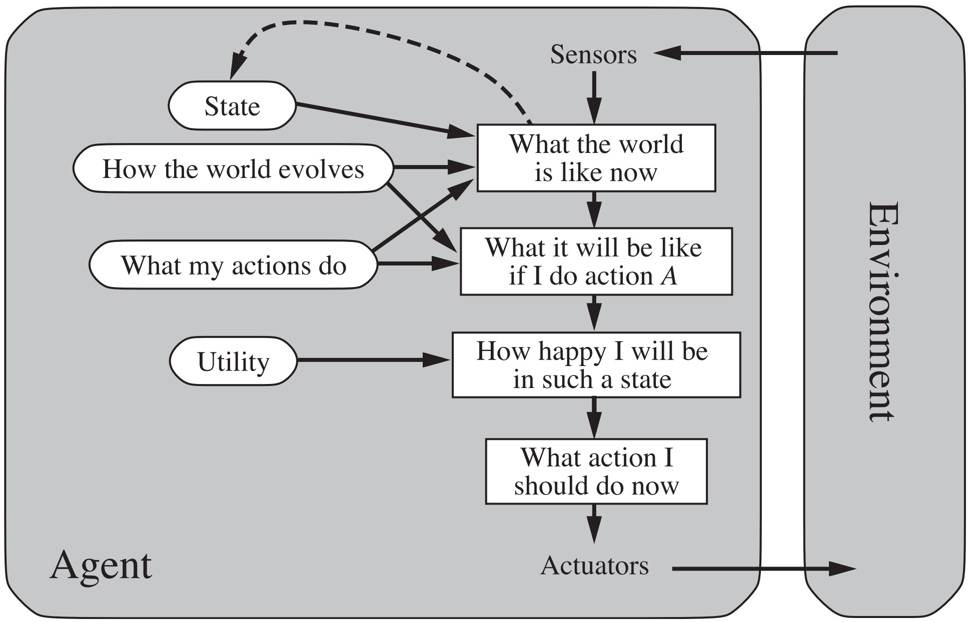
Utility-based Agents (cont)

- Performance measure assigns a score to any sequence of environment states – so it can distinguish better/ worse action sequences
- Agent's utility function is an internalization of the performance measure
- If internal utility function and external performance measure agree then an agent that chooses to maximize its utility will be rational according to the performance measure

Utility-based Agents (cont)

- Making an internal utility function congruent with the external performance measure is not the only way for an agent to be rational
  - E.g. Simple reflex vacuum cleaning agent is rational, though has no idea what its utility function is
- Utility-based agent has many advantages in terms of learning/flexibility
- There are cases where goals are inadequate but where utility-based agents can still make rational decisions:
  - When there are conflicting goals, utility function specifies tradeoff
  - When there are multiple goals none of which can be achieved with certainty, utility allows weighting of likelihood of success vs importance of goal

Utility-based Agents



- Most real world settings require decision making under uncertainty (partial observability; stochasticity)
- So rational agents choose actions that maximize expected utility of action outcomes
  - i.e. the utility the agent expects to derive, on average given probabilities and utilities of each outcome

Utility-based Agents (cont)

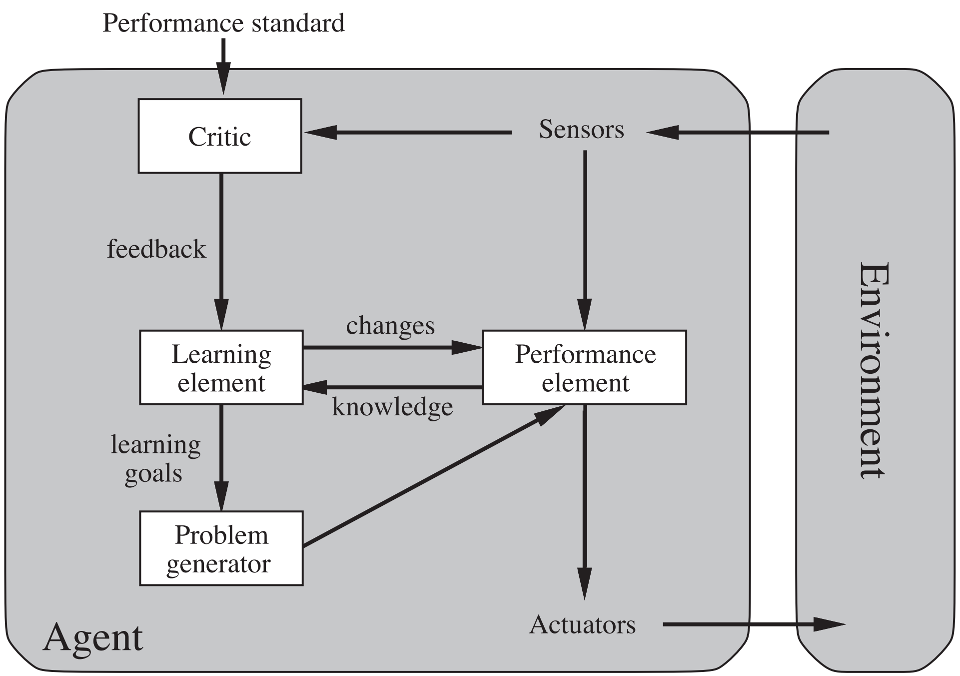
- Is AI "just" a matter of building agents that maximize expected utility?
- Such agents would be intelligent, but it's not so simple ...
  - Such an agent needs to model and keep track of its environment – requires solutions to problems in perception, representation, reasoning and learning
  - Choosing a utility-maximising course of action is also hard
  - Even given this, perfect rationality is usually unachievable in practice due to computational complexity

Learning Agents

- Above we have considered a series of increasing more sophisticated agent program structures that allow agents to choose actions based on modelling
  - The environment
  - The effects of agent actions
  - Agent goals
  - The utility of different action sequences
- Have not considered how such programs come into being
- While they could be coded entirely by hand a "more expeditious method" (Turing's phrase) would to be build learning machines and teach them
- Learning is now the preferred approach in many areas of AI

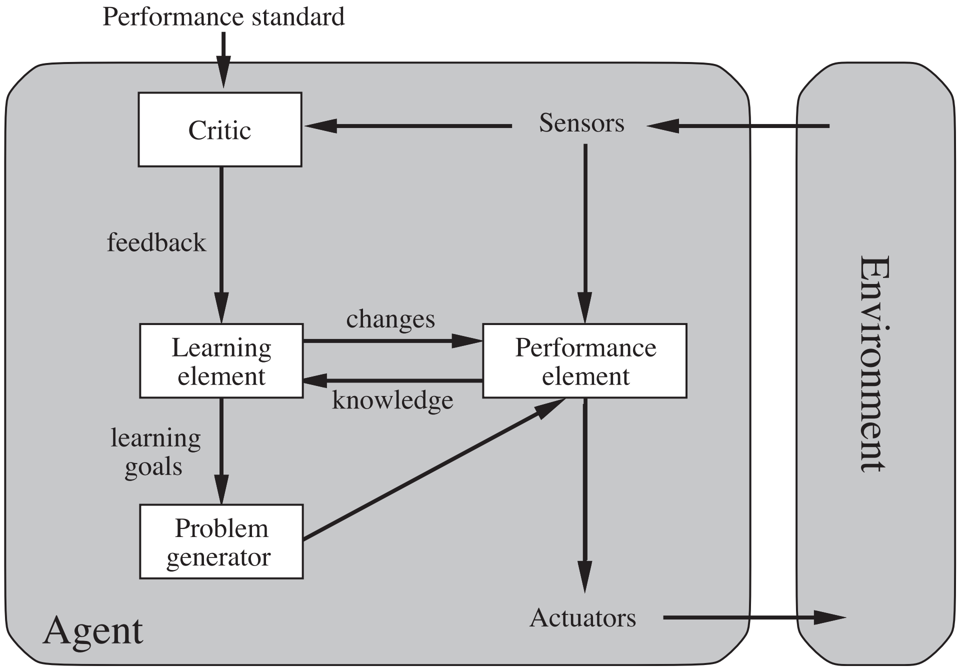
Learning Agents (cont)

- Learning agents may be divided into four components
- Performance element choses actions to be carried out based on percepts (previously we considered this to be the whole agent)



- Learning element is responsible for making improvements
  – Does this in response to feedback received from the critic about how well the agent is doing

Learning Agents (cont)



- Design of learning element depends on design of performance element
  – Given an agent design learning mechanisms can improve every part of the agent
- Critic tells the learning element how well the agent is doing wrt a fixed, external performance standard
- Problem generator suggest actions that will lead to new, informative experiences
  – These may lead to sub-optimal performance in the short term, but may allow agent to discover better actions in the longer term

Learning Agents: Example

- Consider the automated taxi example
- Performance element:
  - Knowledge and procedures taxi has for driving
  - Taxi uses this element to do its driving
- Critic:
  - Observes driving and passes information to learning element
  - E.g. swerving across traffic provokes negative response from other drivers
- Learning element:
  - Learns from critic, e.g., formulates a rule that swerving is bad
- Problem Generator:
  - May suggest experiments, e.g., try brakes different road surfaces under different conditions

Learning Agents: Example

- Learning element may modify any of the knowledge components in the agent design, e.g.
  - "How the world evolves" – may learn from observation of recurrent sequences of environment states
  - "What my actions do" – may learn from observation of results of actions
- For this sort of learning, do not need access to external performance standard
  - Can simply make predictions and observe outcomes of experiments
- To learn utility information, external performance standard must inform agent of negative outcome and learning element must determine what aspect of agent behaviour is responsible for that outcome
  - E.g. violent driving might explain loss of tips

Summary

- The agent program implements the agent function
- There are a variety of agent program designs differing in the kind of information made explicit and used in decision-making
  - Simple reflex agents respond directly to percepts
  - Model-based reflex agents maintain internal state to track evolution of world not obvious from current percept
  - Goal-based agents act to achieve their goals
  - Utility-based agents try to maximise their expected “happiness”
- All agents can improve their performance through learning