INF5390 – Kunstig intelligens

Intelligent Agents

Roar Fjellheim

Outline

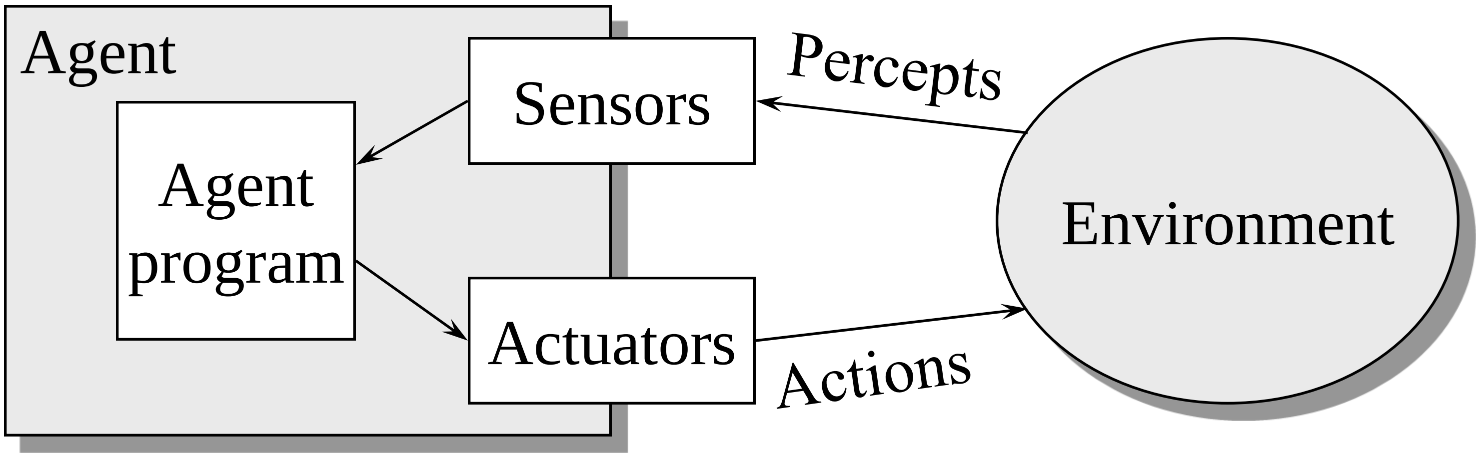
- Intelligent agents
- Environments
- Agent programs
- State representation
- Summary

AIMA Chapter 2: Intelligent Agents

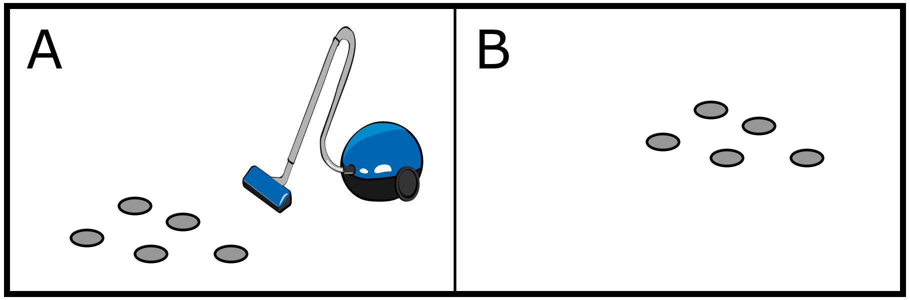
What is an agent?

- An *agent* is anything that can be viewed as *perceiving* its environment through *sensors* and *acting* upon that *environment* through *actuators*
- A *rational agent* is an agent that for each situation selects the *action* that maximizes its *performance* based on its *perception* and built-in *knowledge*
- The task of AI is to build rational agents

Generic agent architecture



Example: Vacuum-cleaner agent



Environment

Percept sequence: [Where am I?, Is it clean here?]	**Action:** Move?/Suck?
[*A*, *Clean*]	*Right*
[*A*, *Dirty*]	*Suck*
[*B*, *Clean*]	*Left*
[*A*, *Clean*], [*A*, *Clean*]	*Right*
[*B*, *Clean*], [*B*, *Clean*], [*B*, *Dirty*]	*Suck*
Etc.	

Definition of rationality

- A *rational* agent:
  - For each possible *percept sequence*, a rational agent selects an *action* that maximizes its expected *performance measure*, given its current *knowledge*
- Definition depends on:
  - Performance measure (success criterion)
  - Agent's percept sequence to date
  - Actions that the agent can perform
  - Agent's knowledge of the environment
- "The best performance under the circumstances .."

Is the vacuum cleaner agent rational?

- Under assumptions:
  - Performance measure: 1 point for each clean square
  - A priori known environment (squares A and B)
  - Only possible actions are *Left, Right, Suck*
  - Perceptions are always correct (location, dirt or not)
- Then agent's performance is rational
- But not rational with other assumptions, e.g.:
  - Penalty for oscillating between clean squares
  - If clean squares can become dirty, it needs to check
  - If unknown environment, it needs to explore
  - Etc.

Tasks and environments

- We build agents to solve *tasks*, i.e. to carry out specific *functions* in specific *task environments*
- PEAS *task environment* characterization:
    - P - Performance measure
    - E - Environment
    - A - Actuators
    - S - Sensors
- The agent *program* implements the function and must take the task environment into account

Examples of agent PEAS descriptions

Agent Type	**Performance measure**	**Environment**	**Actuators**	**Sensors**
Medical diagnosis system	Healthy patient, minimize cost	Patient, hospital, staff	Questions, tests, treatments	Symptoms, findings, patient answers
Satellite image analysis system	Correct image categorization	Images from orbiting satellite	Display categorization of a scene	Color pixel arrays
Part picking robot	Percentage parts in correct bins	Conveyor belt with parts, bins	Jointed arm and hand	Camera, joint angle sensors
Refinery controller	Maximize purity, yield, safety	Refinery, operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical
Interactive English tutor	Maximize student's score on test	Set of students	Exercises, suggestions, corrections	Typed words

Properties of environments

- Fully observable vs. partially observable
  - Sensors detect all aspects relevant for selecting action?
- Single agent vs. multi agent
  - Does the environment include other agents?
- Deterministic vs. stochastic
  - Next state determined by current state and action?
- Episodic vs. sequential
  - Agent’s experience divided into independent episodes?
- Static vs. dynamic
  - Can the environment change while agent deliberates?
- Discrete vs. continuous
  - Limited number of distinct percepts and action?
- Known vs. unknown
  - Outcomes of all actions (or probabilities) known to agent?

Example environments

Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete	Known
Crossworld puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete	Fully
Chess with clock	Fully	Multi	Stochastic	Sequential	Semi	Discrete	Fully
Poker	Partially	Multi	Stochastic	Sequential	Static	Discrete	Fully
Taxi driving	Partially	Multi	Stochastic	Sequential	Dynamic	Continuous	Partially
Medical diagnosis	Partially	Single	Stochastic	Sequential	Dynamic	Continuous	Partially
Image analysis	Fully	Single	Deterministic	Episodic	Semi	Continuous	Partially
Refinery controller	Partially	Single	Stochastic	Sequential	Dynamic	Continuous	Partially
Interactive English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete	Partially

BigDog: A real-world autonomous agent

- BigDog is a rough-terrain robot that walks, runs, climbs and carries heavy loads
- The size of a large dog: about 1 m long, 0.8 m tall and weighs approx. 100 kg.
- Powered by an engine that drives a hydraulic actuation system, with four legs like an animal
- BigDog set a world's record for legged vehicles by traveling 20 km without stopping or refueling
- The ultimate goal is to develop a robot that can go anywhere people and animals can go
- 2013: BigDog company bought by Google

Agent program types

- Table driven agent
- Simple reflex agent
- Model-based reflex agent
- Goal-based agent
- Utility-based agent
- Learning agents

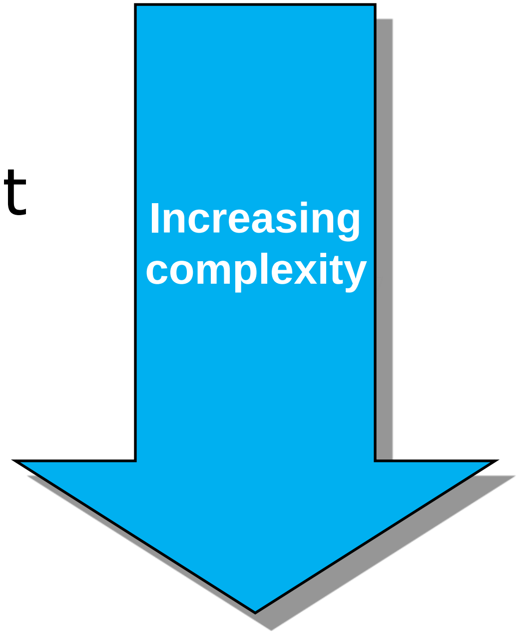


Table driven agent program

function TABLE-DRIVEN-AGENT(*percept*) **returns** *action*
persistent: *percepts*, a sequence, initially empty
table, a table, indexed by percept sequences, initially fully specified
append *percept* to the end of *percepts*
action <= LOOKUP(*percepts*, *table*)
return *action*

E.g. The vacuum-cleaner agent

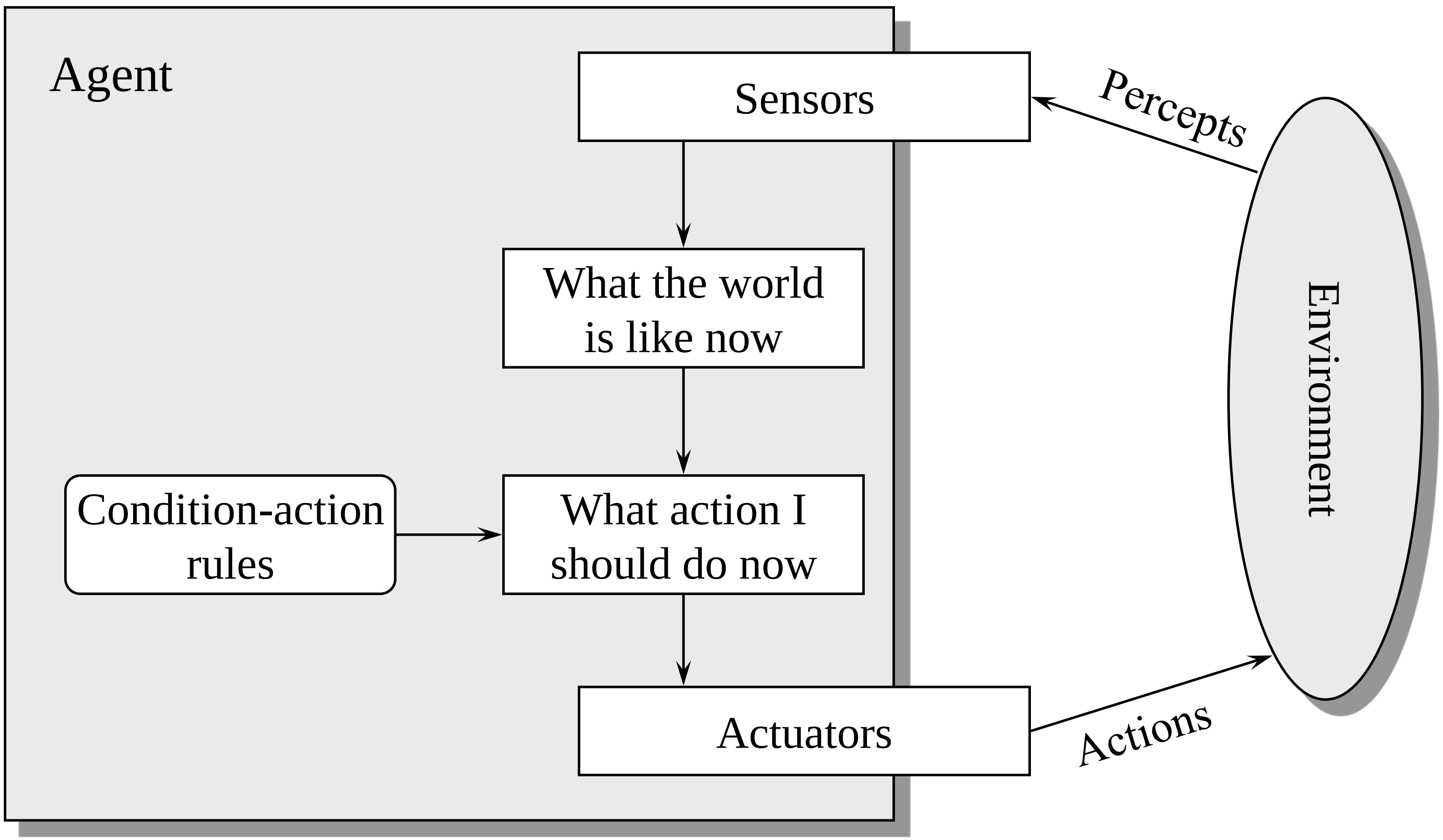
The table driven agent is not viable

- Very large tables needed
  - Table of 10^{150} entries required to play chess
  - Very time consuming table construction
- But portions of the table can be summarized in common input/output associations (rules)
  - E.g. preprocessing of sensory input for taxi driver agent
  - **if** *car-in-front-is-braking* **then** *initiate-braking*

A very simple reflex agent

```
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 agent (generalized)



Simple reflex agent program

function SIMPLE-REFLEX-AGENT(*percept*) **returns** an action

persistent: *rules*, a set of condition-action rules

state <= INTERPRET-INPUT(*percept*)

rule <= RULE-MATCH(*state*, *rules*)

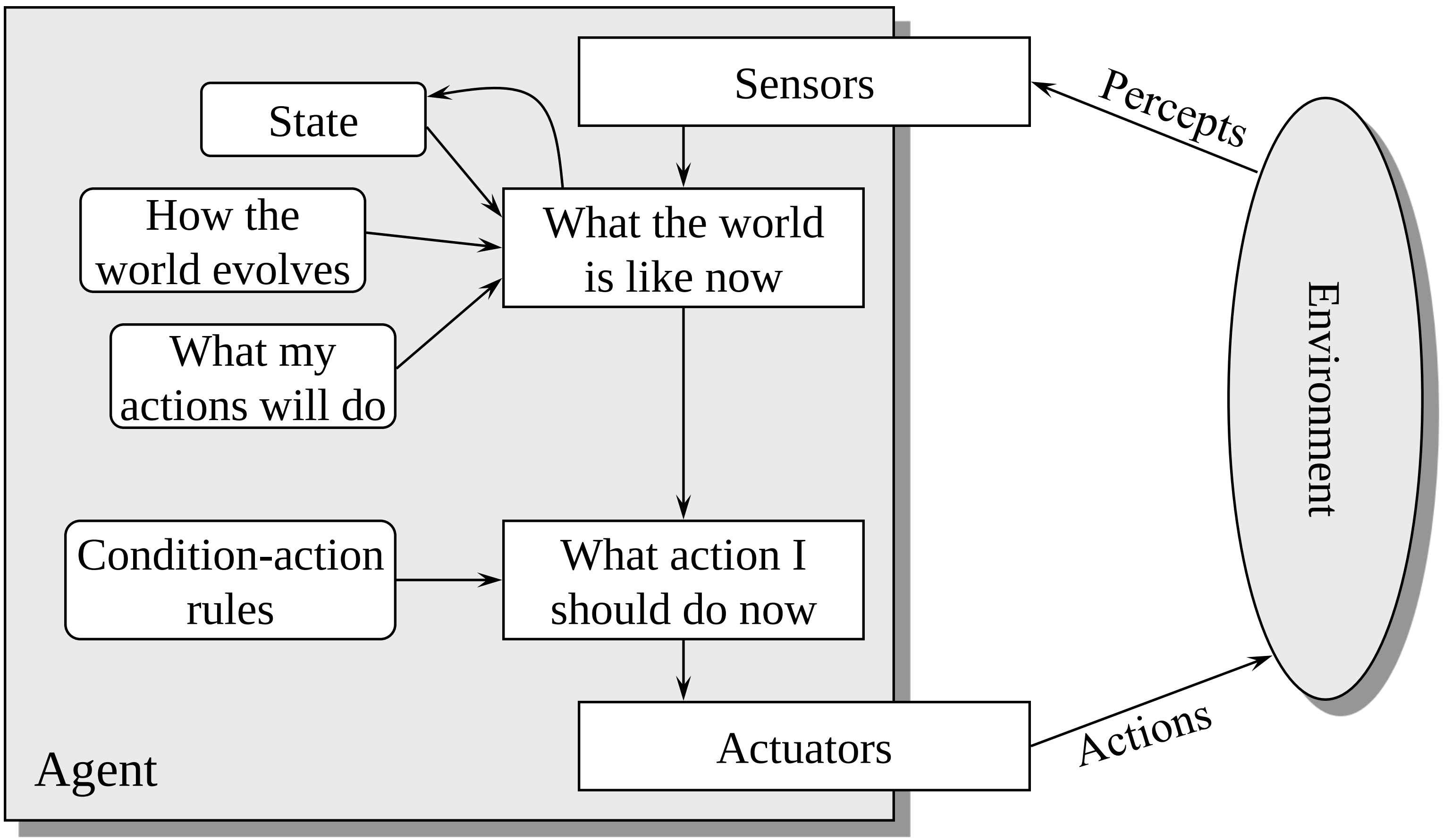
action <= *rule*.ACTION

return *action*

Acting on perception only is not enough

- Sensors do not always give all information required to act
- The agent may need to remember *state* information to distinguish otherwise seemingly identical situations
- The agent needs to know how the state evolves over time
- The agent also needs to know how its own actions affect the state

Model-based reflex agent



Model-based reflex agent program

function MODEL-BASED-REFLEX-AGENT(*percept*)
    returns an action

persistent: *state*, description of the current world state
        model, how next state depends on current state and action
        rules, a set of condition-action rules
        action, most recent action, initially none

state <= UPDATE-STATE(*state*, *action*, *percept*, *model*)

rule <= RULE-MATCH(*state*, *rules*)

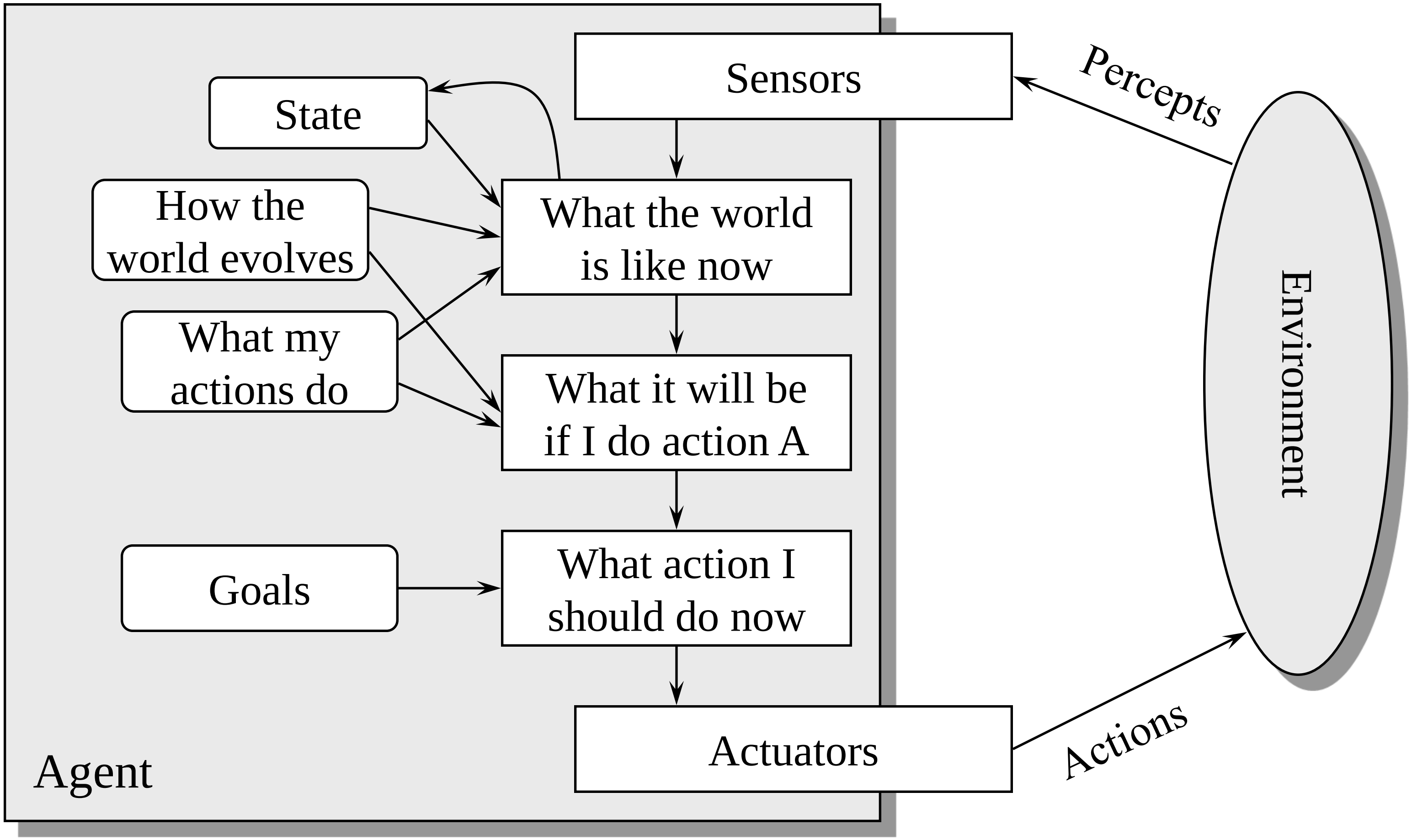
action <= *rule*.ACTION

return *action*

Why does an agent choose an action

- A rational agent chooses an action because it contributes to a desirable *goal*
- In reflex agents the goal is implicit in the condition-action rule (goals are “designed in”)
- More flexible agents have an explicit representation of goal information
- Such agents may reason to select the best action to reach a goal
- The reasoning may involve *searching* and *planning*

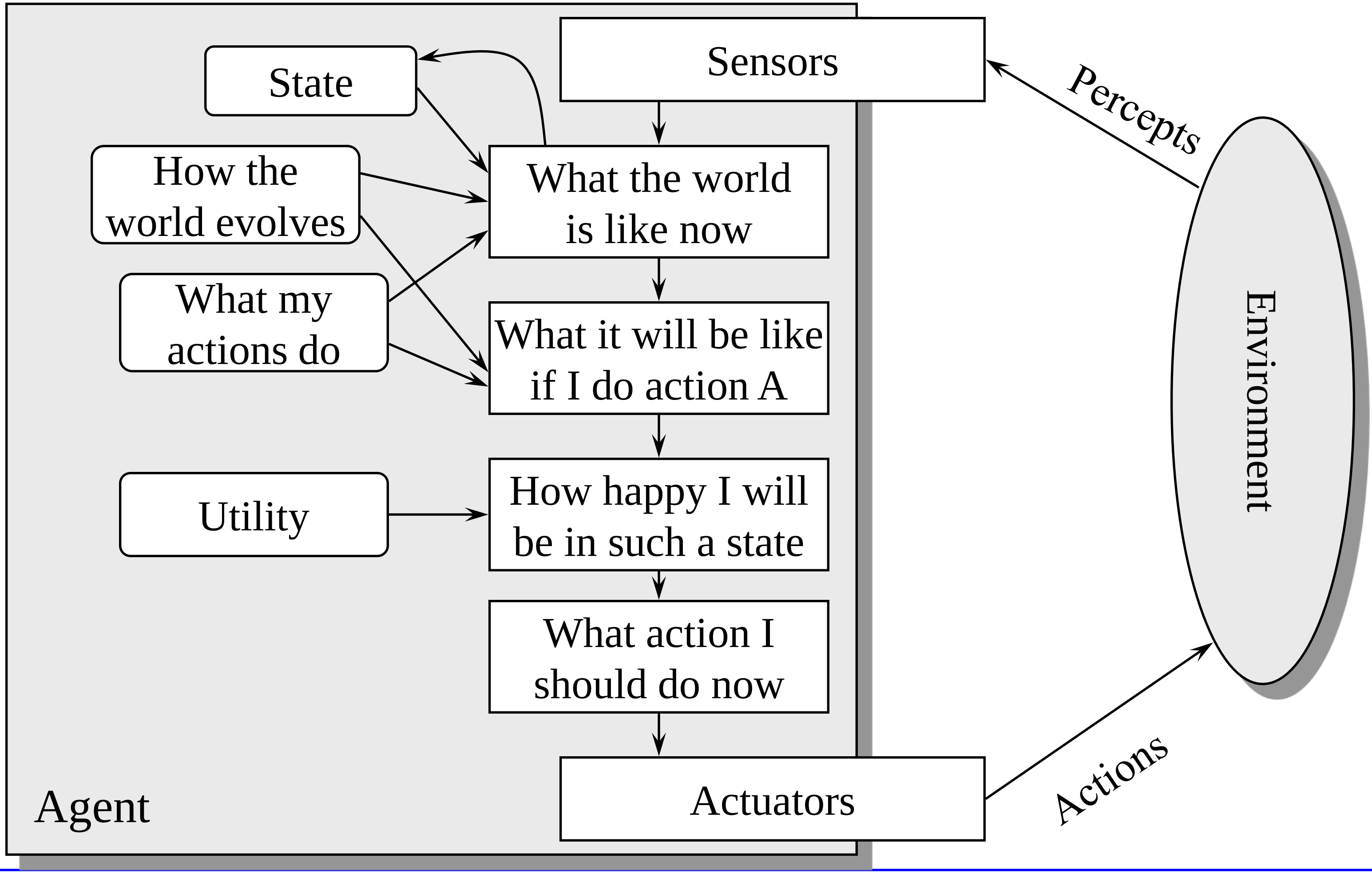
Goal-based agent



All goals and ways to goals are not equal

- Alternative goals may have different values to the agent
- Alternative ways to reach a goal may have widely different costs
- A rational agent ascribes a *utility* to a world state, enabling deliberate choices
- Utility can be used to rank goals, or to select the best path to a goal

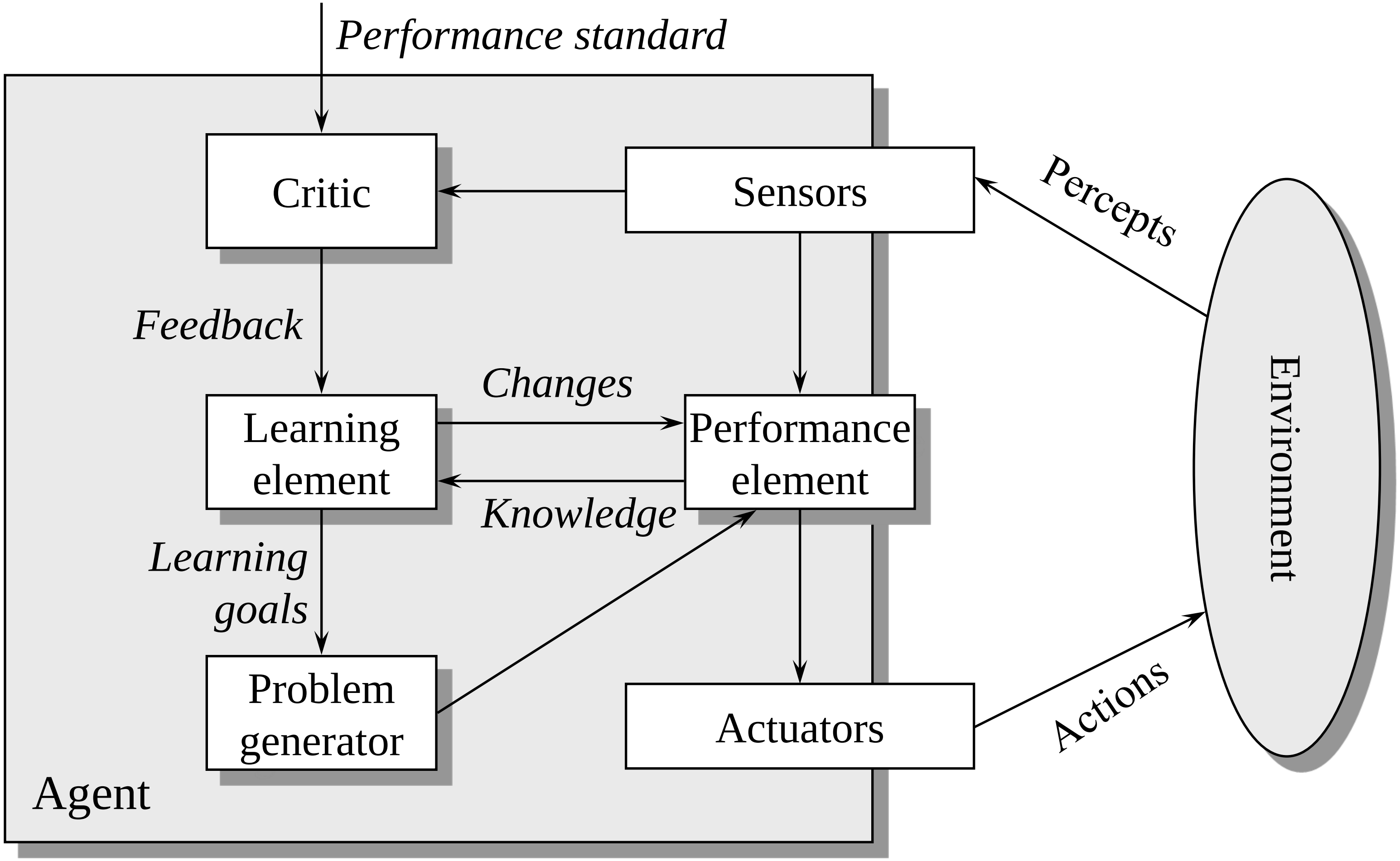
Utility-based agent



How to improve over time

- Previous agent types have assumed "built-in" knowledge, provided by designers
- In order to handle incomplete knowledge and changing requirements, agents must *learn*
- Learning is a way of achieving agent *autonomy* and the ability to *improve performance* over time
- The field in AI that deals with learning is called *machine learning*, and is very active

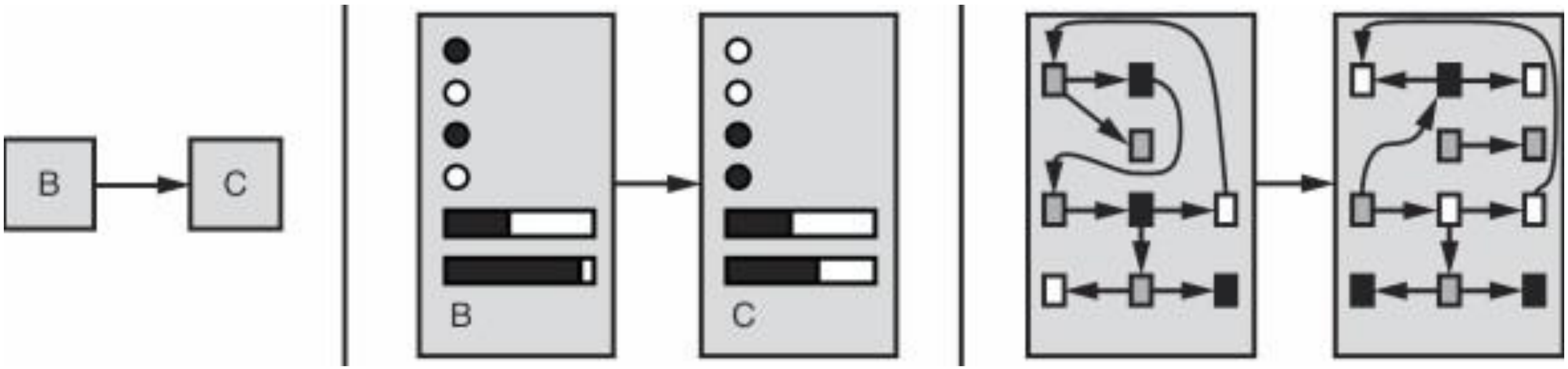
Learning agent



Representing state of the environment

- All agent types need some representation of the state of the environment in order to
  - Evaluate current state
  - Decide next action
  - Predict action result
- Increasingly complex representations
  - *Atomic*: Each state is an indivisible "black box"
  - *Factored*: Each state has attributes with values
  - *Structured*: State has entities related to each other

Examples of state representations



Atomic

- Search and games
- Markov models

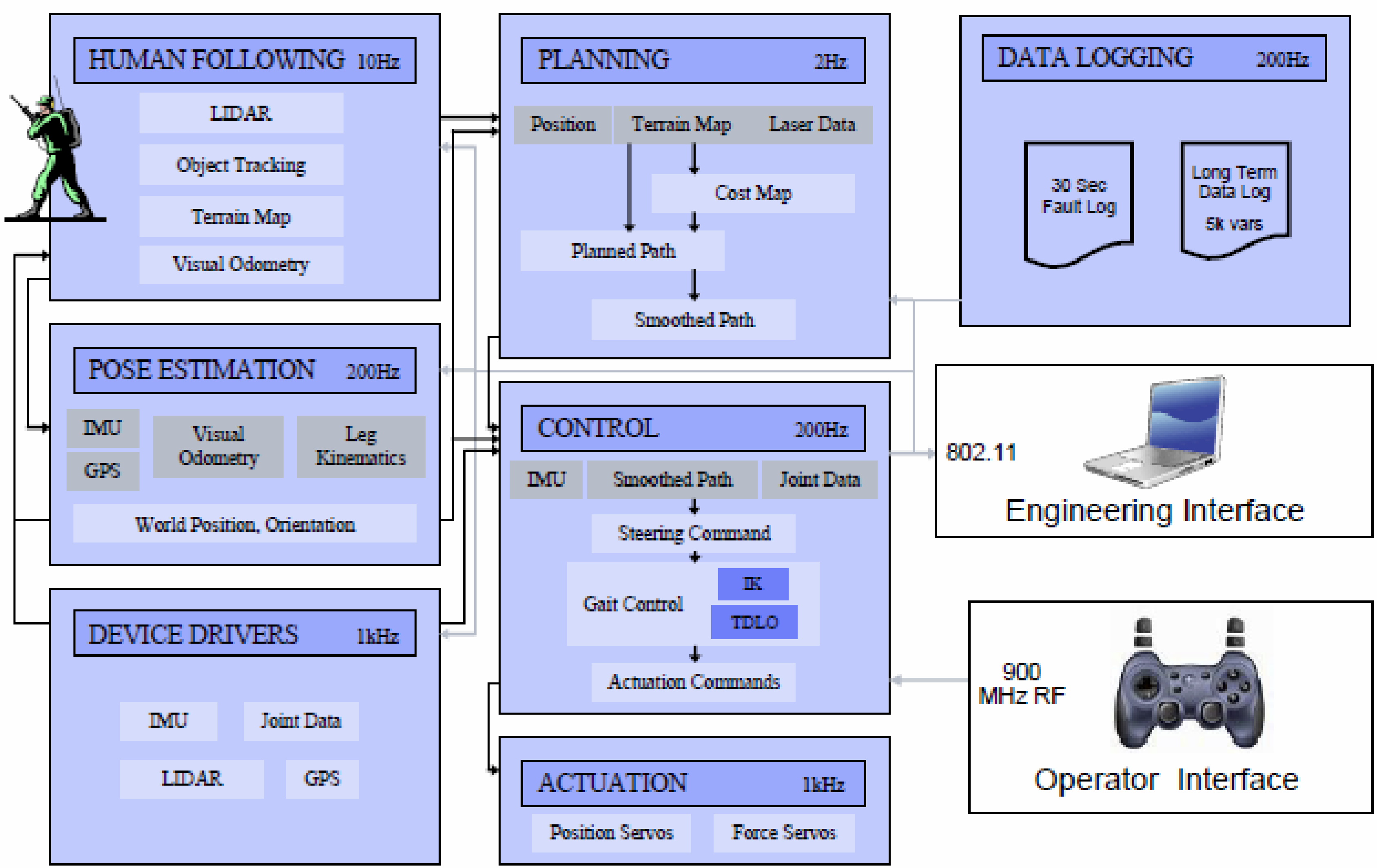
Factored

- Propositional logic
- Planning
- Bayesian networks

Structured

- First-order logic
- Knowledge bases
- Natural language

BigDog’s software architecture



Summary

- An *agent* is something that *perceives* and *acts* in an *environment*
- A *rational agent* is one that always selects the action that maximizes its performance
- *Environments* have varying degrees of complexity and pose different challenges for agent design
- The appropriate design of the agent *program* depends on percepts, actions, goals, and environment

Summary (cont.)

- *Table lookup agents* only useful for tiny problems
- *Simple reflex agents* respond immediately to percepts
- *Model-based reflex* agents remember the state
- *Goal-based agents* act to achieve goals
- *Utility-based agents* maximize utility
- *Learning agents* improve their performance over time
- Agents need environment models of increasing complexity