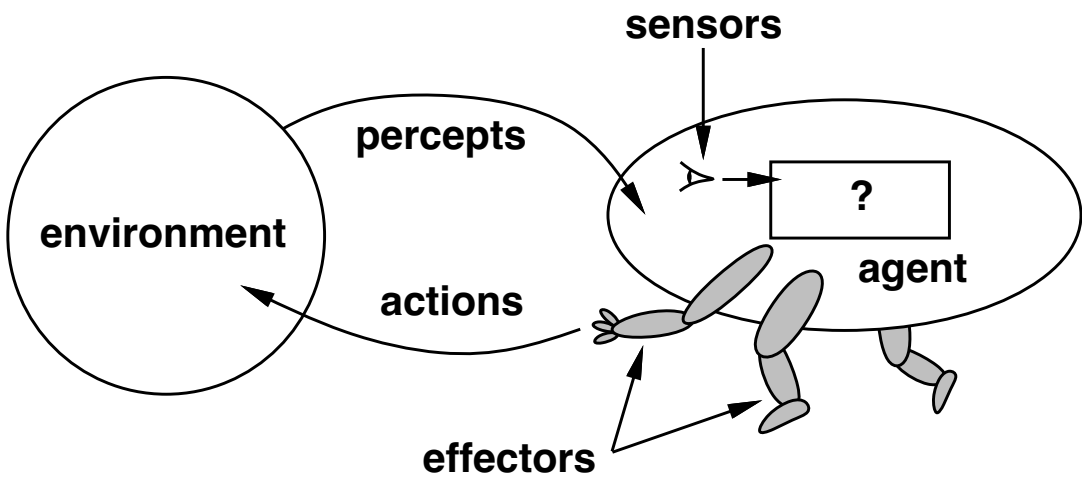
Agent Architectures

Introduction to Artificial Intelligence

G. Lakemeyer

Winter Term 2016/17

Architecture of Rational Agents



- The agent perceives the environment and
- acts according to a *performance criteria*.

Note: Performance criteria are domain dependent.

Example: Autonomous Vacuum Cleaner

Possible performance criteria:

- m^2 per hour
- how clean is the room after vacuuming
- power consumption
- noise emission
- security (e.g. does not run over babys)

Example: Autonomous Vacuum Cleaner

Possible performance criteria:

- m^2 per hour
- how clean is the room after vacuuming
- power consumption
- noise emission
- security (e.g. does not run over babys)

Note: Optimality often impossible

- not all relevant information available
- computational cost may be too high

Ideal Rational Agent 1

Acting rationally depends on

- Performance measure
- Percept sequence
- World knowledge
- possible actions

Ideal rational agent:

> *The agent chooses an action which maximizes its performance for a given percept sequence and knowledge about the world.*

Active sensing necessary to avoid trivialization, e.g.

Crossing a road requires looking to the left and right.

Ideal Rational Agent 2

Ideal mapping

$$\text{Percept Sequences} \times \text{World Knowledge} \longrightarrow \text{Actions}$$

Normally not representable as a table, often not necessary, e.g. Square-Root Agent:

Percept x	Action z
1.0	1.000000000000000
1.1	1.048808848170152
1.2	1.095445115010332
1.3	1.140175425099138
1.4	1.183215956619923
1.5	1.224744871391589
1.6	1.264911064067352
1.7	1.303840481040530
1.8	1.341640786499874
1.9	1.378404875209022
⋮	⋮

function SQRT(x)
  $z \leftarrow 1.0$ /* *initial guess* */
  repeat until $|z^2 - x| < 10^{-15}$
    $z \leftarrow z - (z^2 - x)/(2z)$
  end
  return z

Categorization of Agents

4 kinds: Percepts, Actions, Goals, Environment

Performance Measure

Agent Type	Percepts	Actions	Goals	Environment
Medical diagnosis system	Symptoms, findings, patient's answers	Questions, tests, treatments	Healthy patient, minimize costs	Patient, hospital
Satellite image analysis system	Pixels of varying intensity, color	Print a categorization of scene	Correct categorization	Images from orbiting satellite
Part-picking robot	Pixels of varying intensity	Pick up parts and sort into bins	Place parts in correct bins	Conveyor belt with parts
Refinery controller	Temperature, pressure readings	Open, close valves; adjust temperature	Maximize purity, yield, safety	Refinery
Interactive English tutor	Typed words	Print exercises, suggestions, corrections	Maximize student's score on test	Set of students

A Basic Agent Program

function SKELETON-AGENT(*percept*) **returns** action
  static: *memory*, the agent's memory of the world

  memory ← UPDATE-MEMORY(*memory, percept*)
  action ← CHOOSE-BEST-ACTION(*memory*)
  memory ← UPDATE-MEMORY(*memory, action*)
  return *action*

A Table-Lookup Agent

function TABLE-DRIVEN-AGENT(*percept*) **returns** *action*
  static: *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*

A Table-Lookup Agent

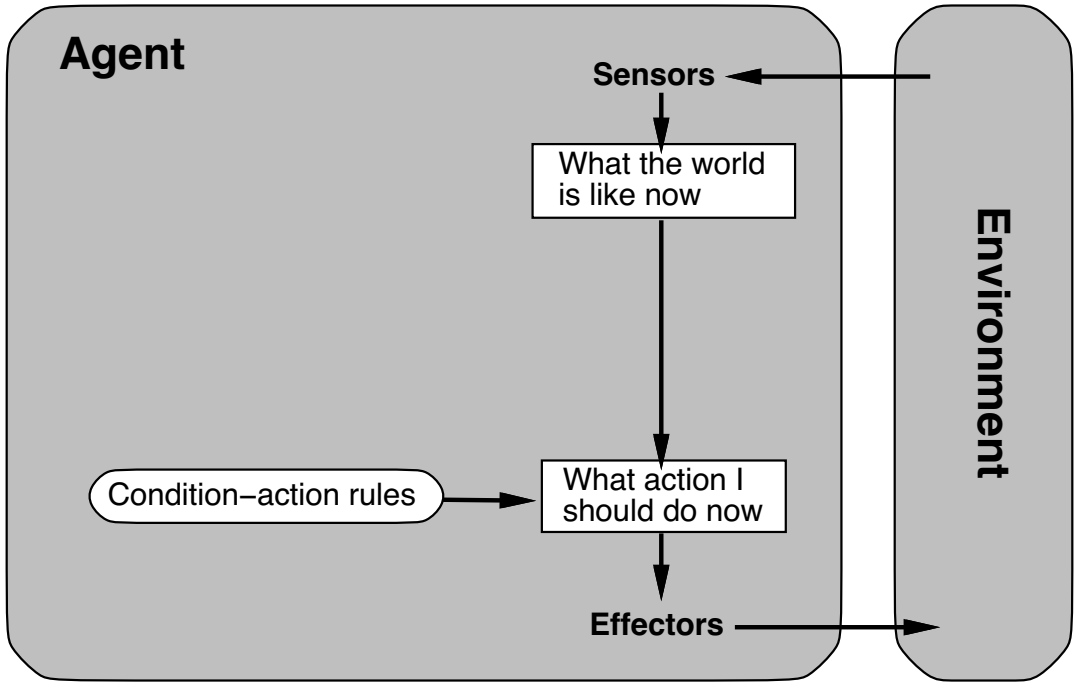
```
function TABLE-DRIVEN-AGENT( percept) returns action
  static: percepts, a sequence, initially empty
          table, a table, indexed by percept sequences, initially fully specified

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

Why is this a bad idea in general?

e.g Chess: about 35^{100} table entries

- too complex (memory)
- too inflexible

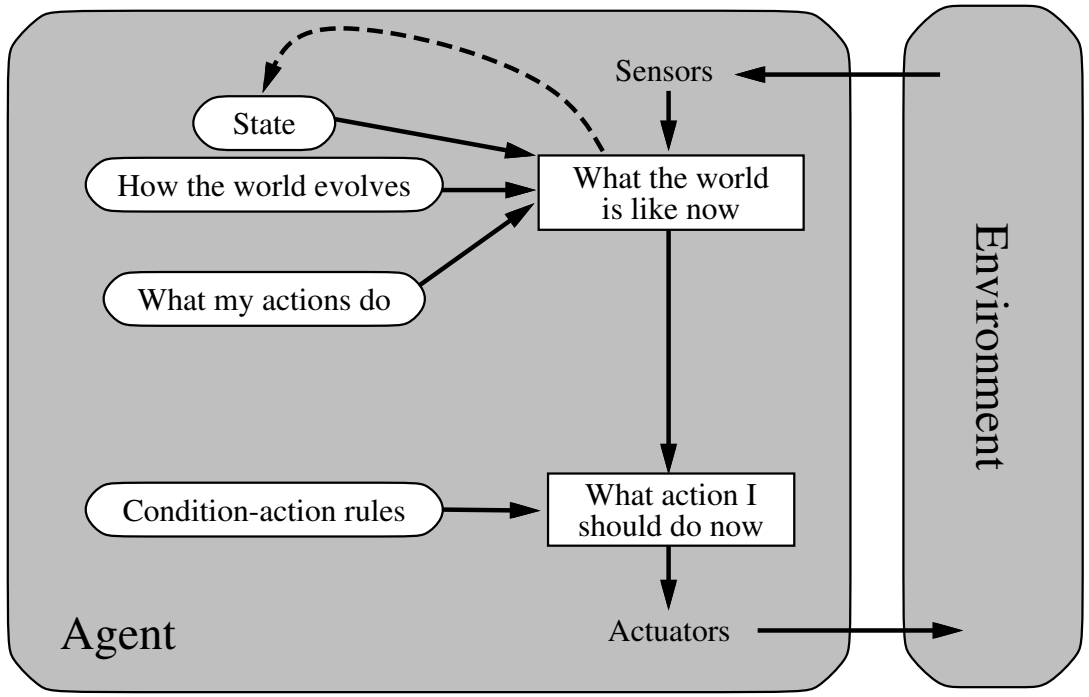
Reflexive Agents



function SIMPLE-REFLEX-AGENT(*percept*) **returns** *action*
  static: *rules*, a set of condition-action rules

  state ← INTERPRET-INPUT(*percept*)
  rule ← RULE-MATCH(*state, rules*)
  action ← RULE-ACTION[*rule*]
  return *action*

Agents with an Internal World Model

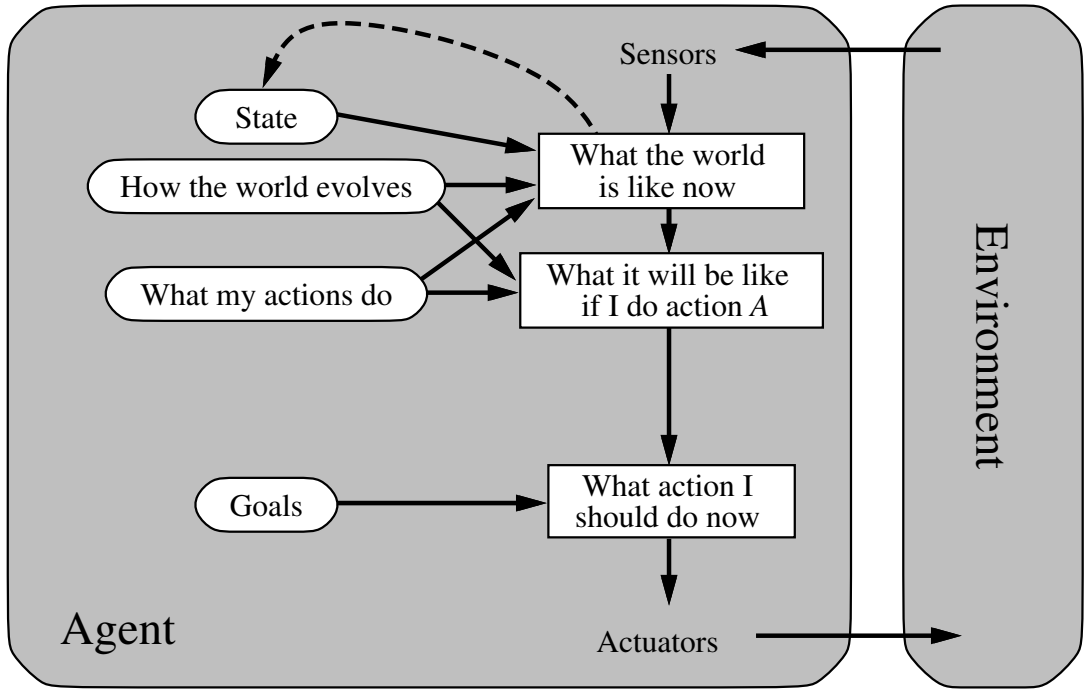


function REFLEX-AGENT-WITH-STATE(*percept*) **returns** *action*
  static: *state*, a description of the current world state
    rules, a set of condition-action rules

  state ← UPDATE-STATE(*state, percept*)
  rule ← RULE-MATCH(*state, rules*)
  action ← RULE-ACTION[*rule*]
  state ← UPDATE-STATE(*state, action*)
  return *action*

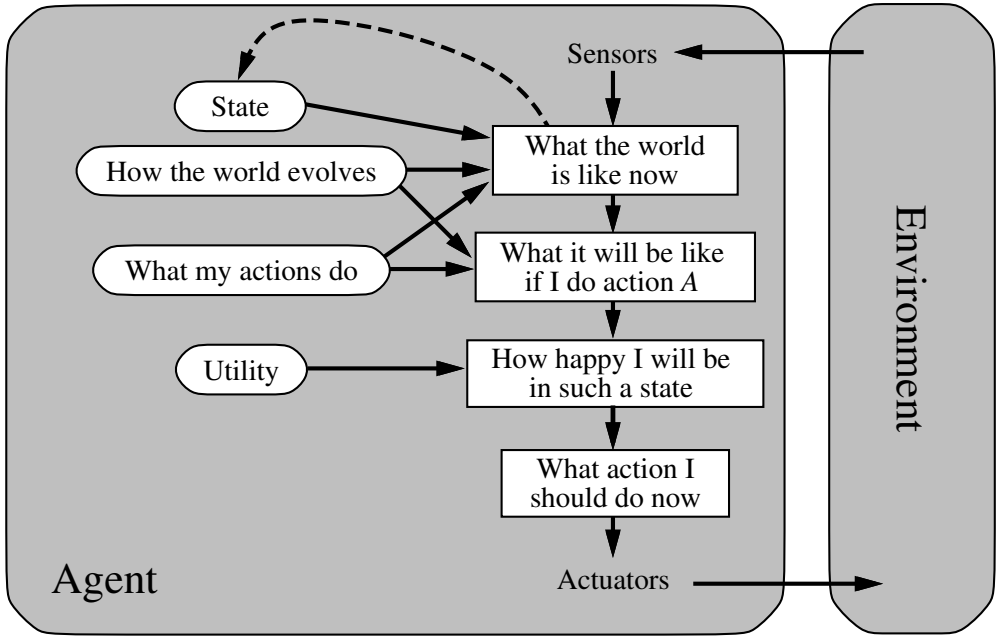
Agents with explicit goals

Need Planning



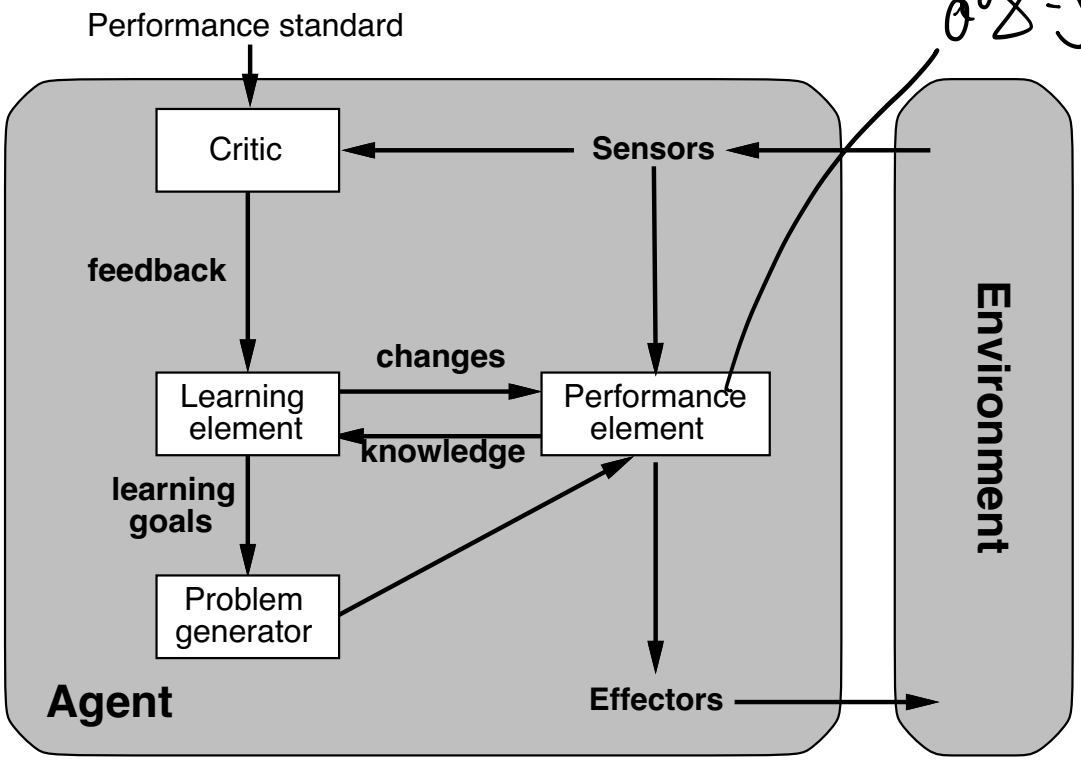
Utility-based Agents

choosing among competing plans



Learning Agents

The goal of learning: Optimize future behavior on the basis of the history of percepts, actions, and knowledge about the world.

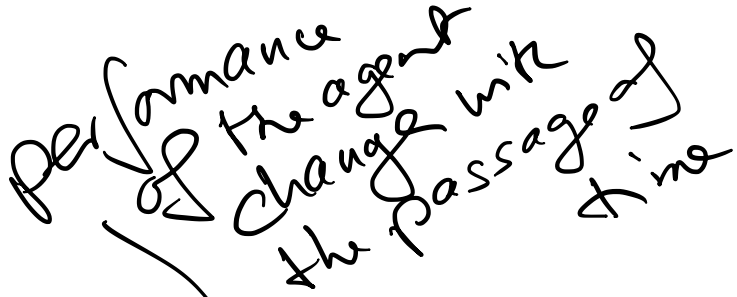


Performance Element: Agent in the old sense.
Critic: Tells the system how good or bad it is performing.
Learning Element: Improves the system.
Problem Generator: Suggests actions to test performance.

Properties of Environments

- accessible vs. nonaccessible
  Are all relevant aspects of the world accessible to the sensors?
- deterministic vs. nondeterministic/stochastic
  Does the next state depend completely on the current state and the action chosen.
- episodic vs. nonepisodic
  Does the choice of an action depend only on the current state or also on the past?
- static vs. dynamic
  Can the world change while deciding on the next action?
- discrete vs. continuous
  Is the world discrete (as in chess) or not (mobile robots)?

Example Environments



Environment	Accessible	Deterministic	Episodic	Static	Discrete
Chess with a clock	Yes	Yes	No	Semi	Yes
Chess without a clock	Yes	Yes	No	Yes	Yes
Poker	No	No	No	Yes	Yes
Backgammon	Yes	No	No	Yes	Yes
Taxi driving	No	No	No	No	No
Medical diagnosis system	No	No	No	No	No
Image-analysis system	Yes	Yes	Yes	Semi	No
Part-picking robot	No	No	Yes	No	No
Refinery controller	No	No	No	No	No
Interactive English tutor	No	No	No	No	Yes