

# Artificial Intelligence and an Agent Approach to Software Development

COMPX800

Eric McCreath

The Australian National University

Semester 2 2004

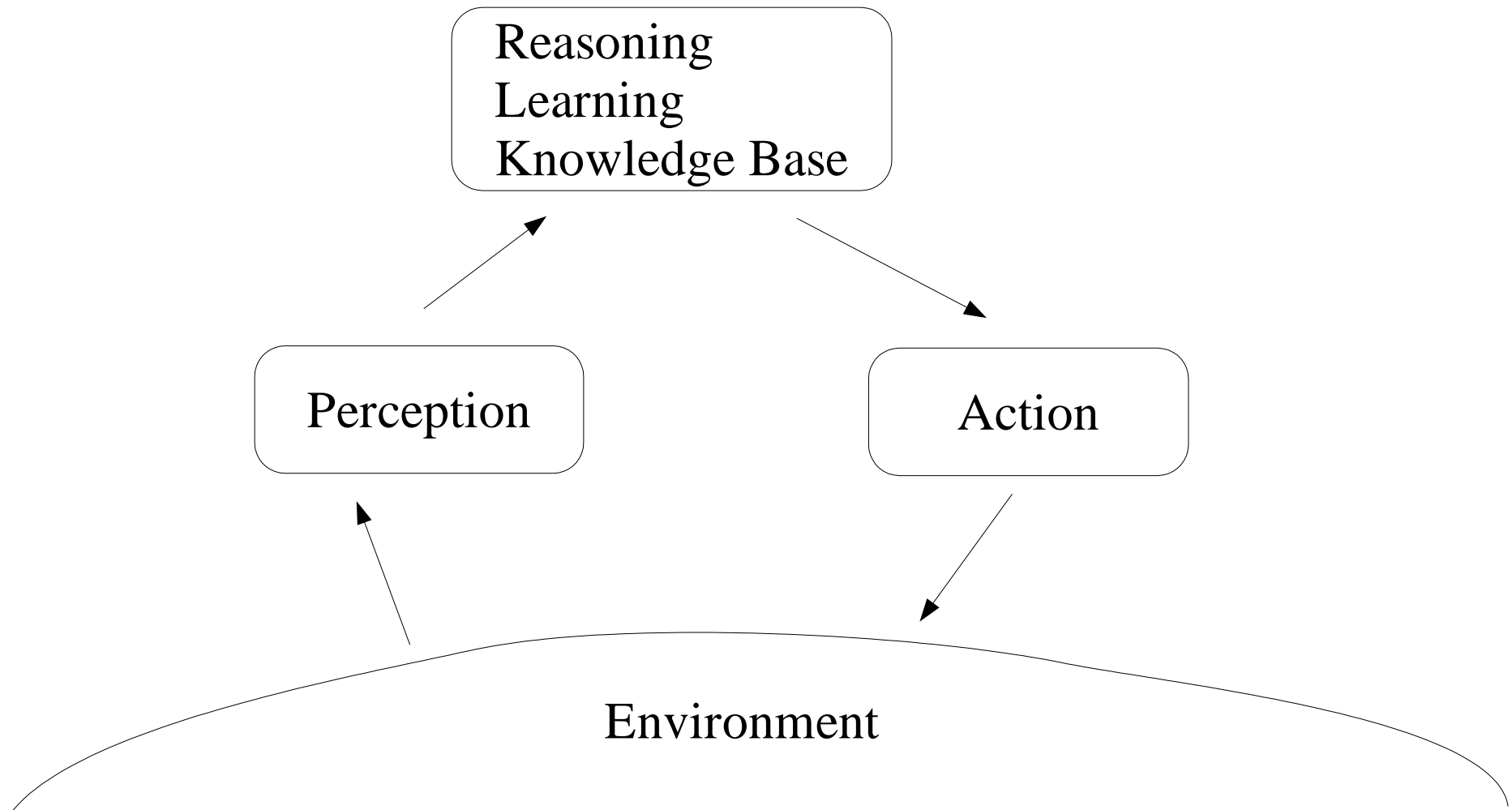
# Outline

2

- A background to Artificial Intelligence
- Agents
- The BDI approach
- JAM - Example
- Utility(if we have time)

# AI Problems

- A framework for AI problems:



# Examples

- Medical diagnosis
- Satellite image analysis
- Mail sorting
- Speech recognition
- Handwriting Recognition
- Game playing
- Planning - DART

# Performance Versus Simulation

- People study AI from a number of different perspectives. These include:
  - Performance : make programs that can perform a particular task.
  - Simulation : help understand human thought and behavior.
- AI is enormous field of research in both depth and breath. Many techniques are maturing and being incorporated in computer systems.
- When AI problems are 'solved' they are often not considered AI problems anymore. AI can be defined as problems that are difficult to address using computers, these are often easy to perform by people.

# Definitions

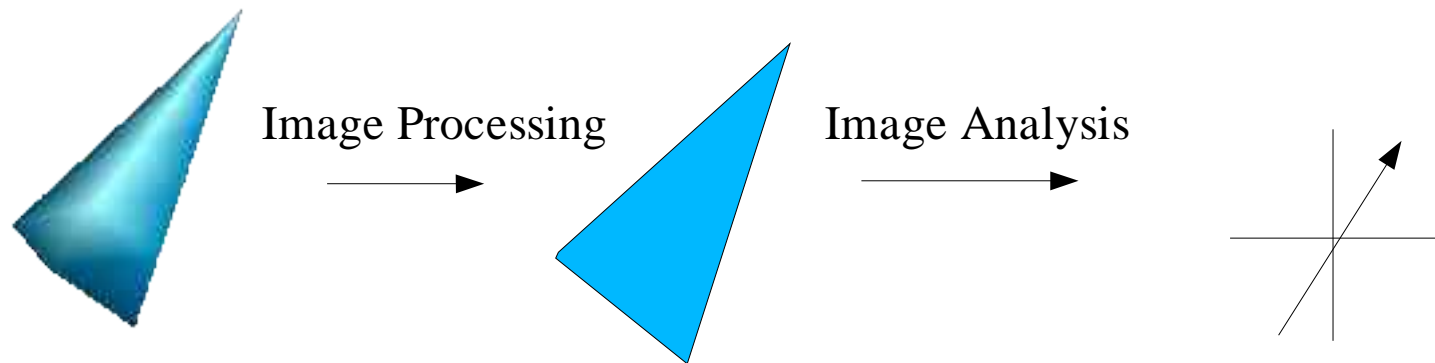
- 'The study of how to make computers do things at which, at the moment, people are better.' Rich and Knight 1991
- '[the automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning ...' Bellman 1978
- 'The study of the computations that make it possible to perceive, reason and act.' Winston 1992

# Challenges

- Challenges within AI include :
  - Natural language processing/generation
  - Knowledge representation
  - Reasoning
  - Learning
  - Vision
  - Robotics
  - Speech

# Vision

- Understanding images is often a two step process:
  - Image Processing - extracting attributes/characteristics from the image. (edge detection, thresholding, smoothing )
  - Image Analysis - making sense of these attributes/characteristics.



# Reasoning

- Reasoning within an AI system takes on a number of forms. Often it will involve search.
- These searches can be formulated as a search over a directed graph. This is made up of:
  - A collection of states - each state represents a possible 'situation'. There will be a start state and a set of goal states.
  - A collection of productions - these say how you can move from one state to the other states. This forms a directed graph over the states.
  - A function that determines if a state is a goal state or not.

# Sliding Puzzle Example

1	2	3
4	5	6
7	8	

Goal State

1	2	3
4	5	6
7	8	

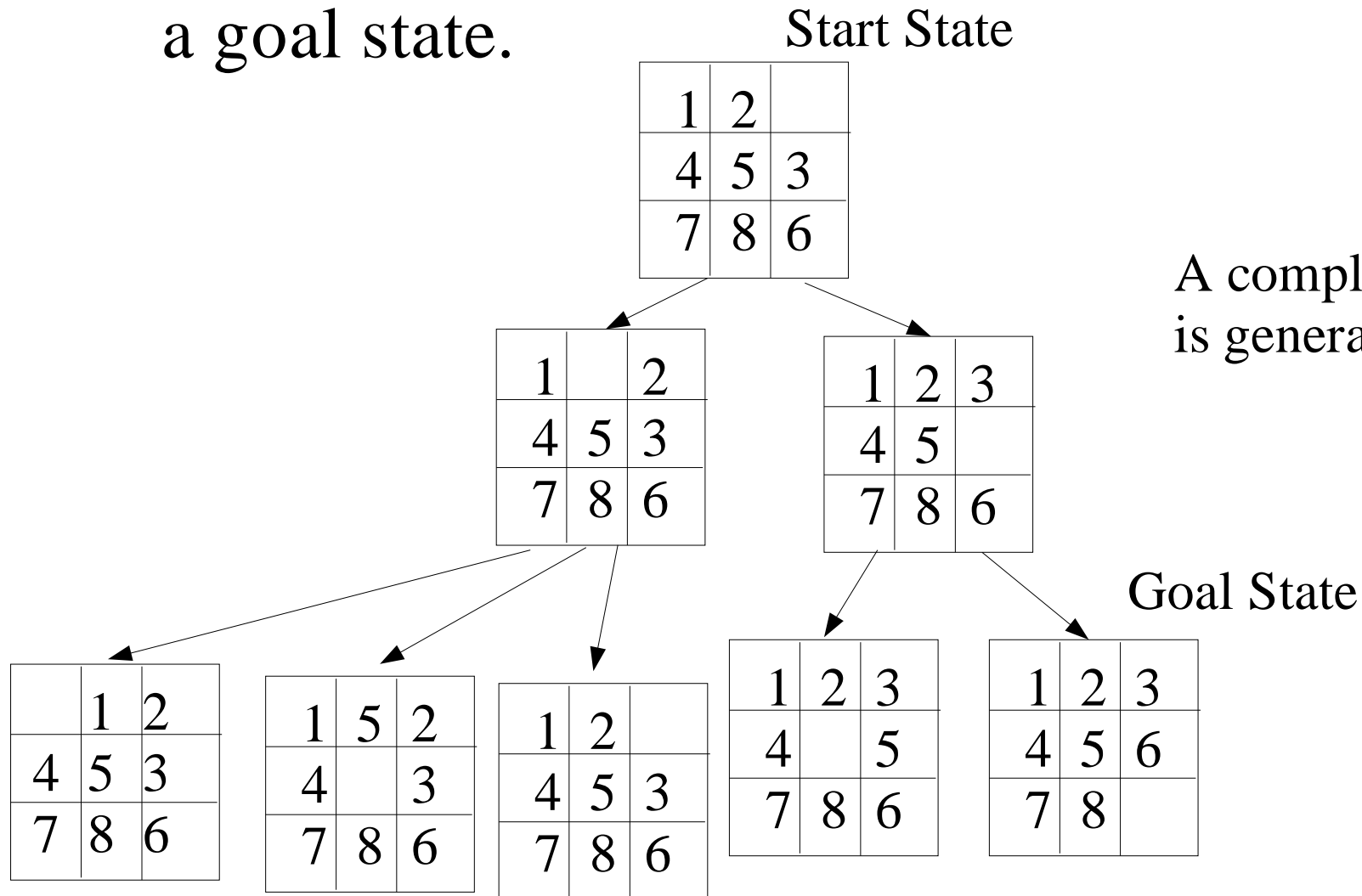
1	2	3
4	5	
7	8	6

1	2	3
4		5
7	8	6

1	2	
4	5	3
7	8	6

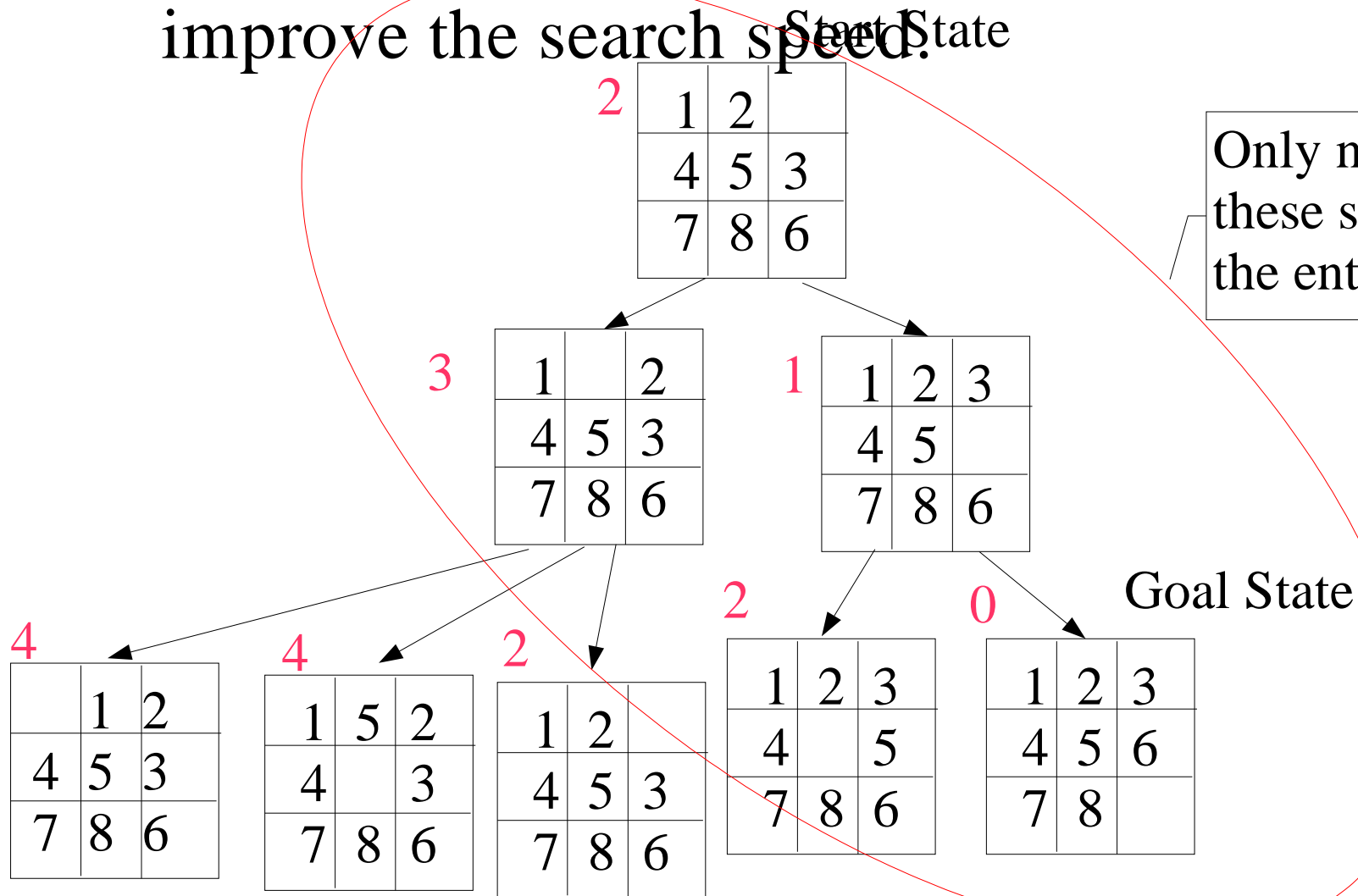
# Search Trees

- A search tree can be used to work out a control system to guide the system from the start state to a goal state.



# Heuristic

- A heuristic may be used to estimate the 'distance' to the closest goal state. This can improve the search speed.



- The idea of the production system also generalizes to reasoning logically.

All people are mortal.  
All mortal die.  
Socrates is a person.



All people are mortal.  
All mortal die.  
Socrates is a person.  
Socrates is mortal.



All people are mortal.  
All mortal die.  
Socrates is a person.  
Socrates is mortal.  
Socrates will die.

- Deduction is a form of reasoning where the conclusion are certainly/justifiably true give the set of premises.
- Induction is a form of reasoning that takes a set of observations and forms generalization that explains these observations. This form of reasoning is neither certain or justifiable.
  - Could you write a program that worked out the next number : 2,4,6,8,?
- Learning uses inductive reasoning. Information that is learnt is represented by a hypothesis. Hypothesis spaces include things like : neural networks, decision trees, logic, ..

# Will you play Tennis?

15

Given the follow examples:

temp=hot, rain=no, wind=no => yes lets play

temp=cold, rain=no, wind=no => no game

temp=hot, rain=light, wind=breeze => yes play

temp=warm, rain=storm, wind=breeze => no game

A learner may induce a rule like:

If temp = hot then play tennis else don' play tennis.

This could then be used for prediction. A different learner may induce a different rule.

If temp = cold or rain=storm then don' play else play tennis.

Note that both these explain the data exactly, however, they make a different prediction on some unseen examples. i.e. temp=hot, rain=storm, wind= no => ????

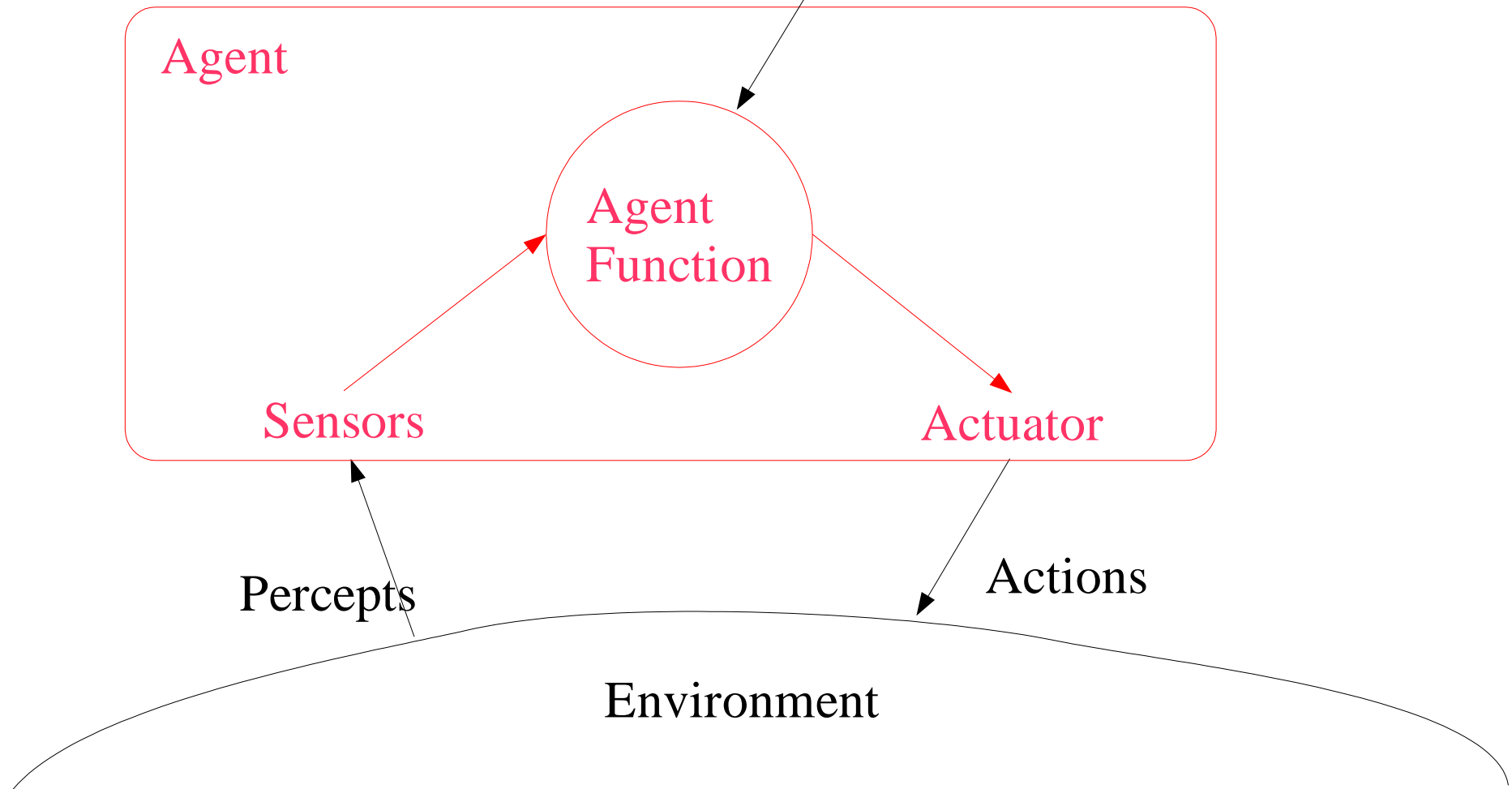
# Considering the Consequences

- AI has the potential to benefit society, however, as with any new technology there are a number of perils.
- There will be winners (and losers?).
- The direction of AI is to be able to '*Do more by doing less*'.
- The questions we will face are more the 'ought computers ...' rather than 'can computers ...'.  
(See Weizenbaum computer Power and Human Reason)

# An Agent Framework

17

The agent function maps a percept sequence to an action.



# Rational Agents

- A *rational agent* will 'do the right think'?
- The right think can be measured against a *performance measured* on the agent within an environment.
- Rationality depends on: the performance measure, the agents prior knowledge of the environment, the percept sequence, and the action that can be performed.
- A definition of a rational agent[RN - p36]:  
*For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.*

# Agent Definition

- Definition from Wooldridge's MultiAgent Systems book (page 16), adapted from Wooldridge and Jennings (1995).

*“An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives.”*

# A rational agent

- *omniscience*  $\neq$  *rationality*
- *Information gathering* is an important part of a rational agent. You would expect a balance between *exploration* and *exploitation*.
- Both learning and autonomy are important for many agents.

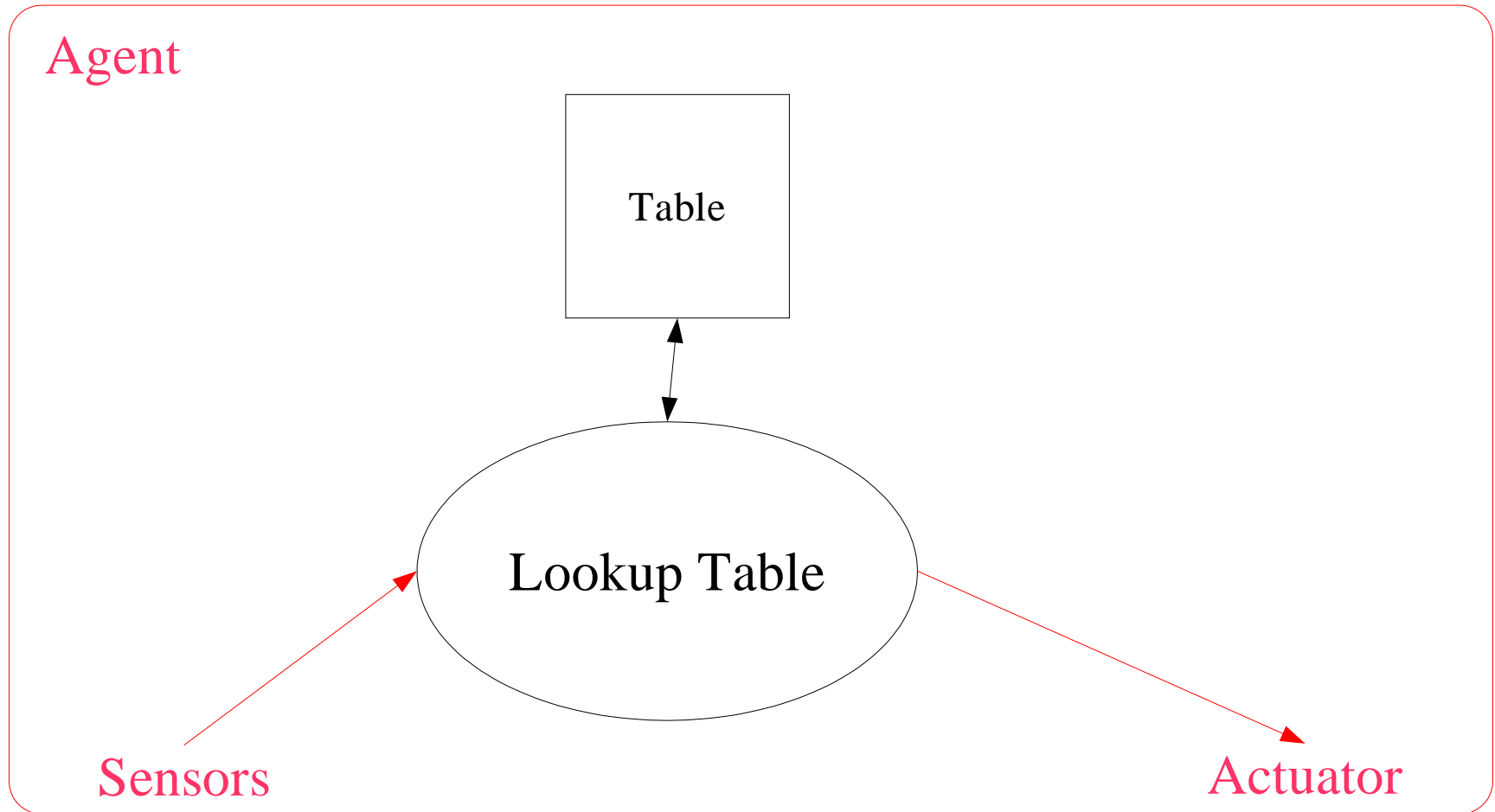
# Task Environment

- PEAS(Performance, Environment, Actuators, Sensors) is a useful way of characterizing the task environment.
- Example[RN p39]

Agent Type	Performance	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits.	Roads, other traffic, pedestrians, customers.	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard.

# Table Driven Agent

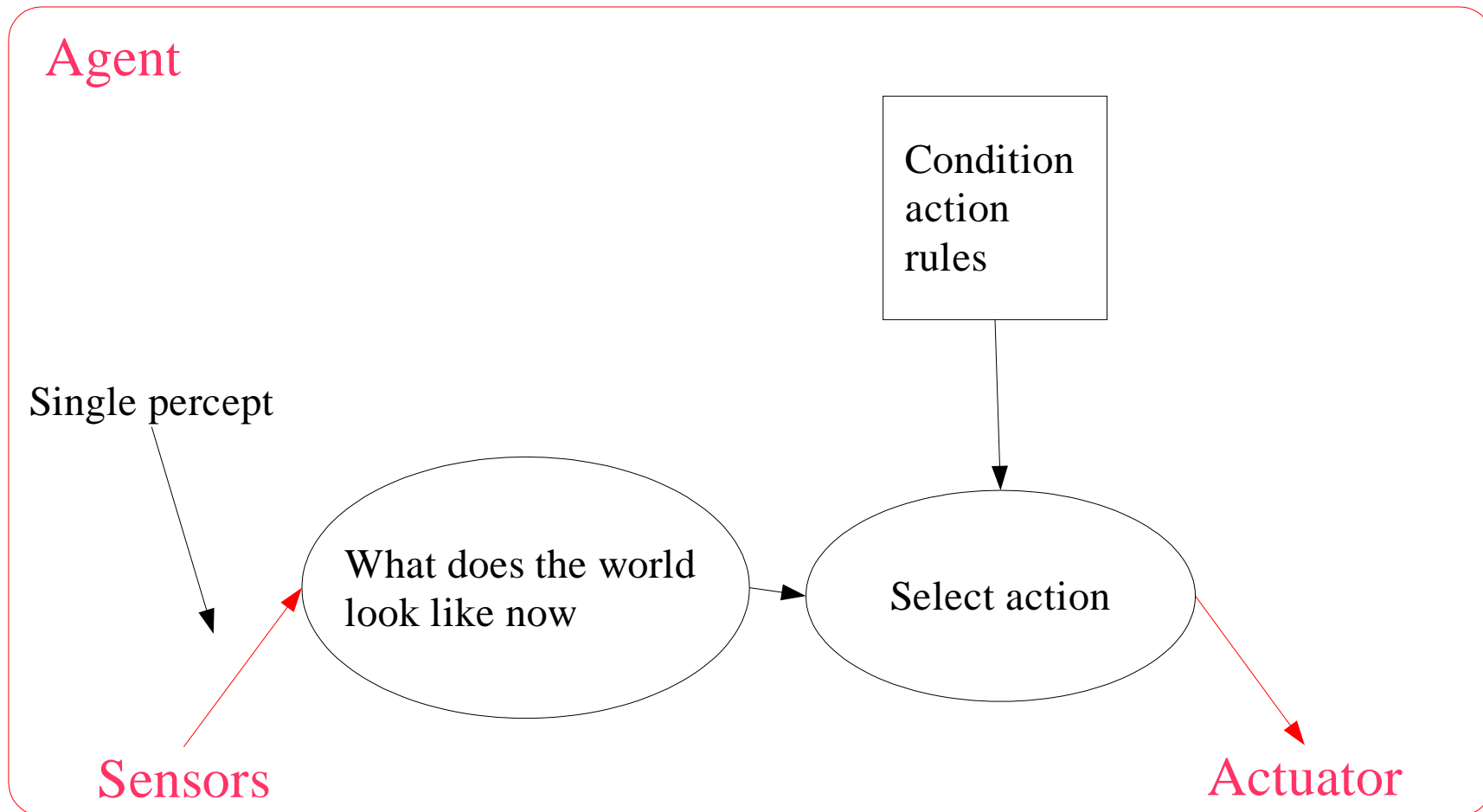
22



Problem - Tables get very very very big!

# Simple Reflex Agent

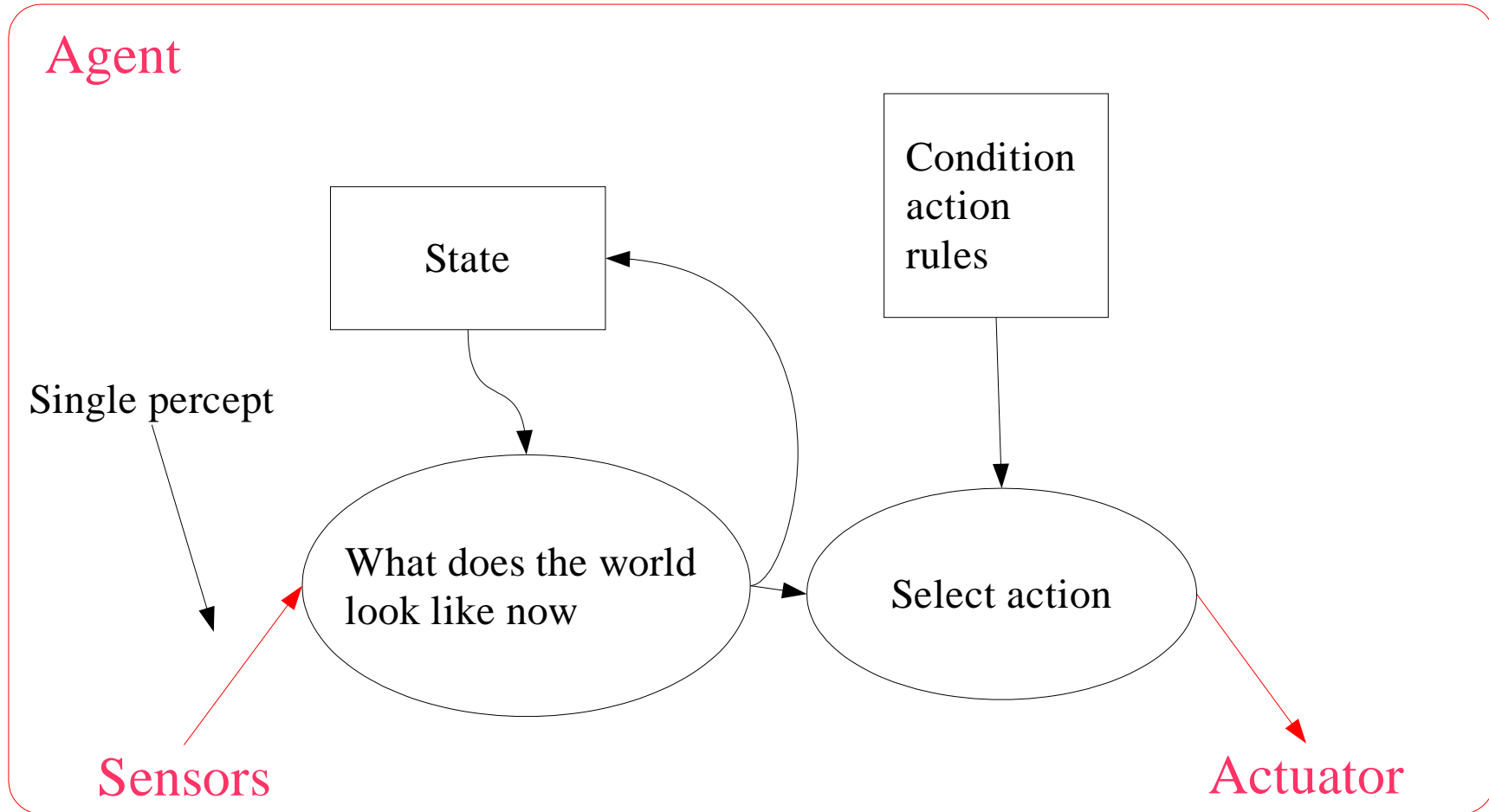
23



- Also a reactive agent. Has trouble remembering where the car is parked!

# Model Based Reflex Agent

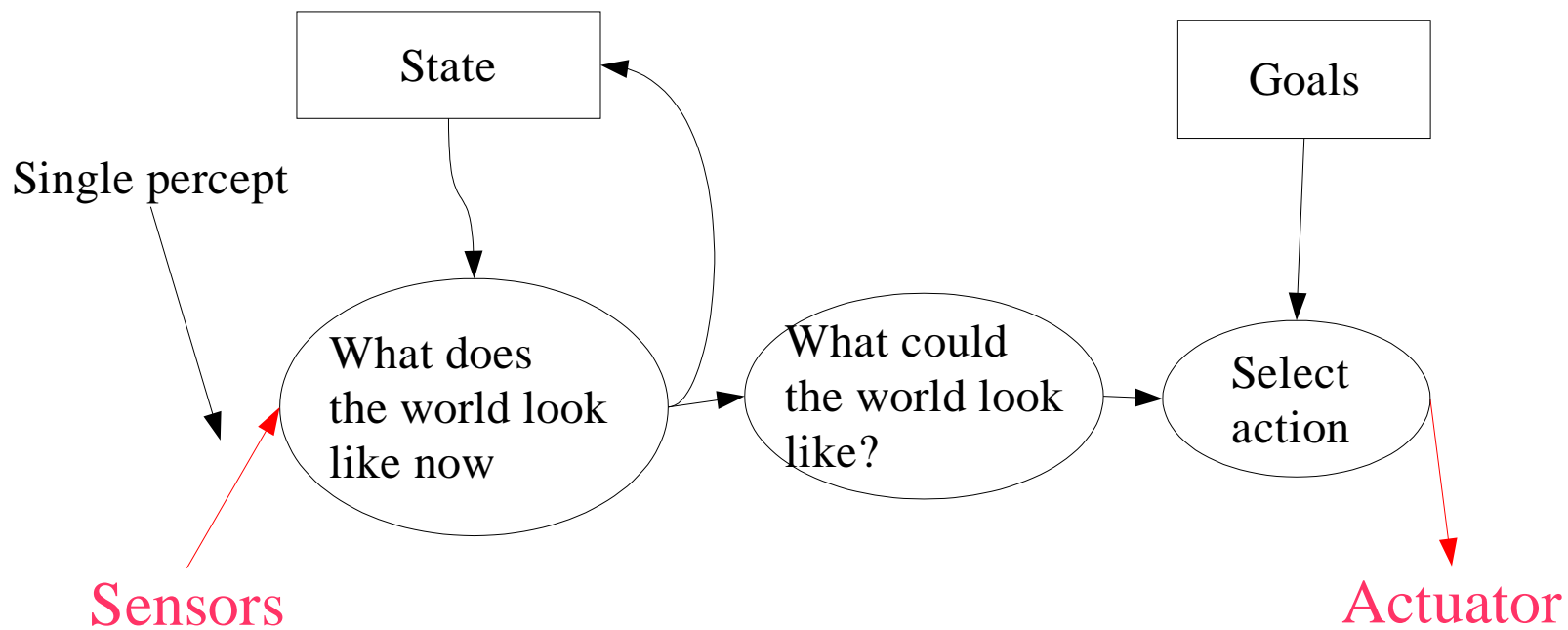
24



# Goal Based Agent

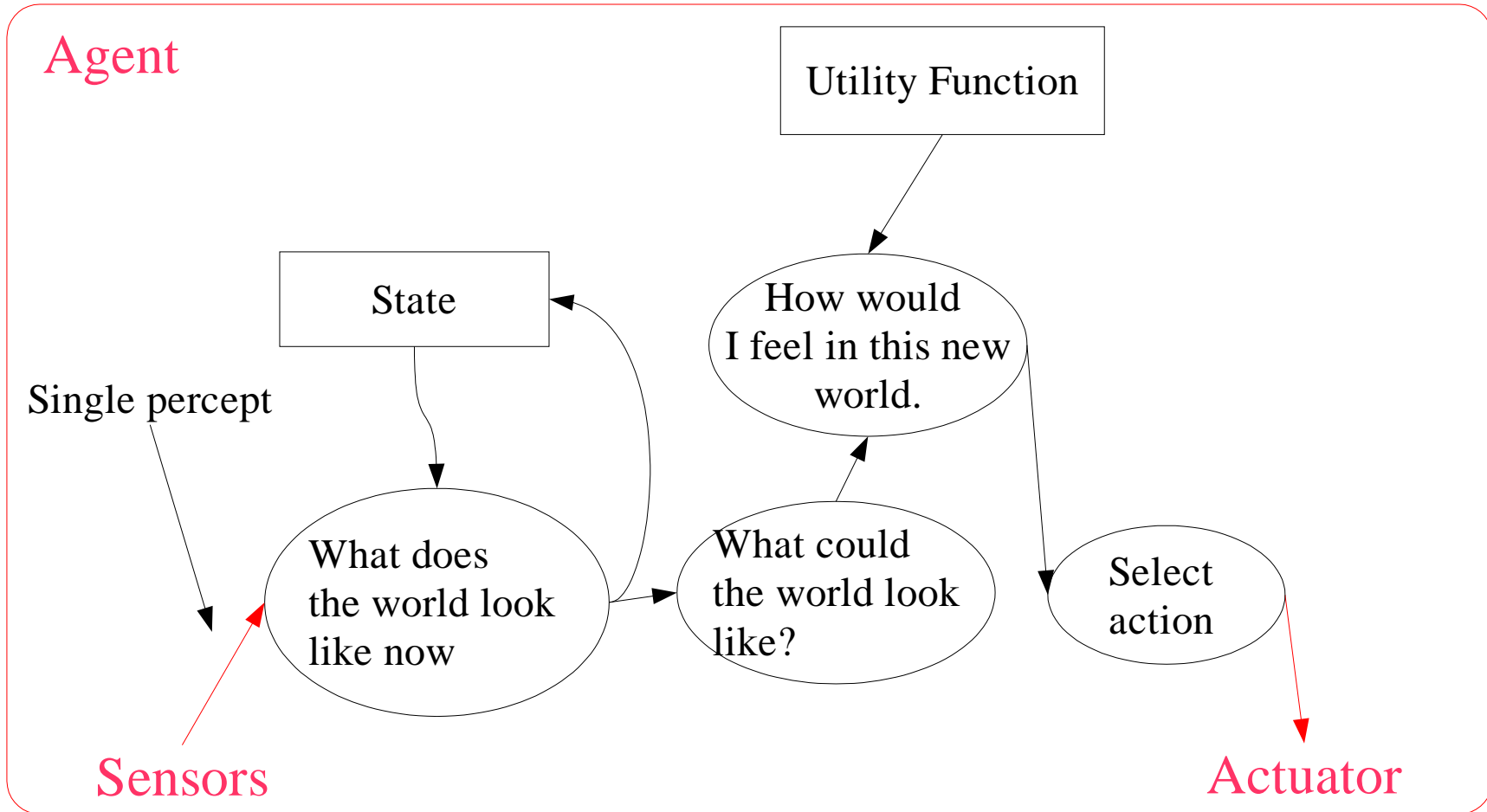
25

Agent

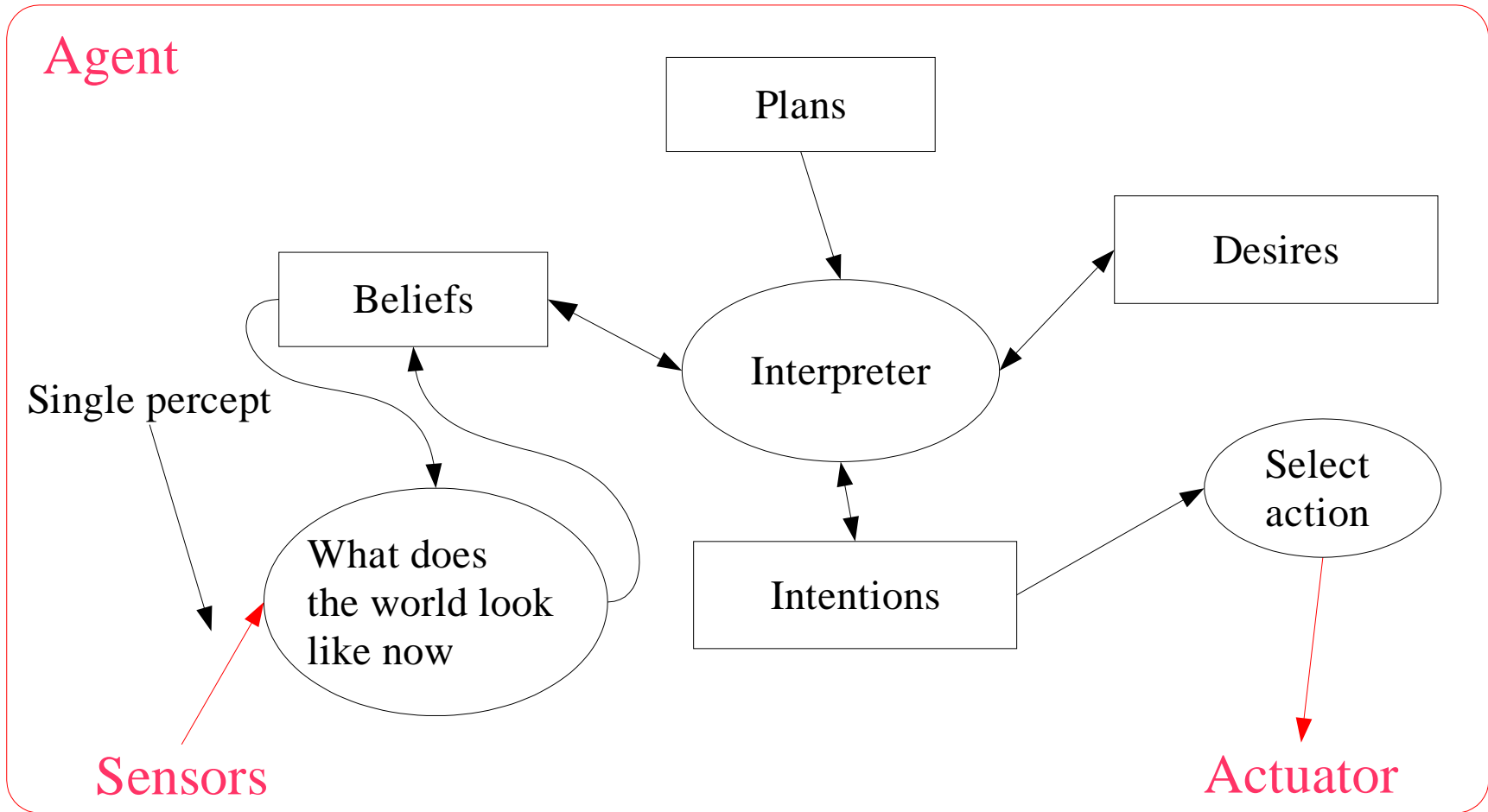


# Utility Based Agent

26



- The agent-oriented programming paradigm was based on a societal view of computation and was proposed by Yoav Shoham (1993).
- The main idea is to base the way programs are described on mentalistic notions like as belief, desire, and intention. This provides an other level of abstraction help us deal with complex systems.
- We often describe human behavior in terms of mentalistic notions. So it may also be useful in describing how programs are to act.



- JAM is a BDI system developed by Marcus Huber it is an extension of the original PRS system.

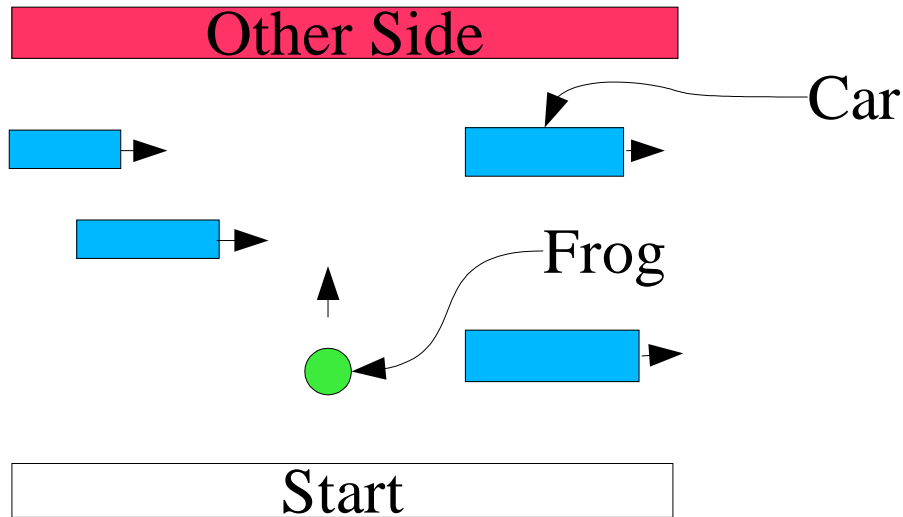
Plans look something like:

Plan:

```
{  
  GOAL: [goal specification]  
  NAME: [name of the plan]  
  CONTEXT: [expression]  
  BODY: [procedure]  
}
```

# Getting the Frog to the other side!

30



Frogs sensors:  
other\_side()  
car(ahead)  
car(here)  
car(behind)

Frogs actuators:  
jump(forward)  
jump(stay)  
jump(back)

# Computational Frog

```
While (! other_side()) {  
    if (!car(ahead)) {  
        jump(forward);  
    } else if (!car(here)) {  
        jump(stay);  
    } else if (!car(behind)) {  
        jump(back);  
    } else {  
        become road kill!!  
    }  
}
```

# BDI Frog

32

```
Plan {
  Goal :
    ACHIEVE get_to_other side
  Context :
    TEST !car(ahead)
  BODY :
    while (!other_side()) {
      jump(forward)
    }
}
```

```
Plan {
  Goal :
    ACHIEVE get_to_other side
  Context :
    TEST car(ahead)
  BODY :
    MAINTAIN life
}
```

```
Plan {
  Goal :
    MAINTAIN life
  BODY :
    if (!car(ahead)) jump(forward)
    else if (!car(here)) jump(stay)
    else if (!car(behind)) jump(back)
    else FAIL
}
```

- In 1662 Arnauld said[RNp584]:

*To judge what one must do to obtain a good or avoid an evil, it is necessary to consider not only the good and the evil in itself, but also the probability that it happens or does not happen; and to view geometrically the proportion that all these things have together.*

- The utility function captures the notion of an agents preference over particular situation. It effectively give an partial ordering over different situation.
- $U(S)$  is the utility of state  $S$ .

$$EU(A|E) = \sum_i P(\text{Result}_i(A)|Do(A), E) U(\text{Result}_i(A))$$

- You would generally expect a rational agent to select a actions with maximum expected utility (MEU).
- Note that, difference arise between one-shot and sequential decisions.

# Utilities

- Note that, utilities may not be linear. (Money is a good example of this.)
- Also scaling of a utility function does not change utilities assessment.

$$U'(S) = k_1 + k_2 U(S) \text{ where } k_2 > 0$$