جامعة الجلالة
GALALA UNIVERSITY

Artificial Intelligence Science Program

Dr. Mohamed Abd Elaziz

Environment Properties

- Fully observable vs. partially observable
- Deterministic vs. stochastic/strategic
- Episodic vs. sequential
- Static vs. dynamic
- Discrete vs. continuous
- Single agent vs. multiagent

Environment Properties

- **Fully observable vs. partially observable**
- If an agent's sensors give it access to the complete state of the environment at each point in time, then we say that the task environment **is fully observable**.
- An environment might be **partially observable because** of noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data:
  - For example, a cleaner agent with only a local dirt sensor cannot tell whether there is dirt in other squares, and
  - an **automated taxi** cannot see what other drivers are **thinking.** If the agent has no sensors at all then the environment is **unobservable**.

Environment Properties

- **Deterministic vs. nondeterministic**
- If the next state of the environment is **completely determined** by the current state and the action executed by the agent(s), then we say the environment is **deterministic**; otherwise, it is **nondeterministic**.
- an agent need not worry about uncertainty in a fully observable, so it called deterministic environment. If the environment is partially observable, then it could appear to be nondeterministic.
  - Taxi driving is clearly **nondeterministic** in this sense, because one can never predict the behavior of traffic exactly; moreover, one's tires may blow out unexpectedly and one's engine may seize up without warning.

Environment Properties

- Fully observable vs. partially observable
- Deterministic vs. stochastic / strategic
- Episodic vs. sequential
- Static vs. dynamic
- Discrete vs. continuous
- Single agent vs. multiagent

Environment Properties

- Episodic vs. sequential
- In an **episodic task environment**, the agent's experience is divided into atomic episodes.
  - In each episode the agent receives a percept and then performs a single action. The next episode does not depend on the actions taken in previous episodes.
    - Many classification tasks are episodic.
- In **sequential environments**, on the other hand, the current decision could affect all future decisions.
  - Chess and taxi driving are sequential

Environment Properties

- Fully observable vs. partially observable
- Deterministic vs. stochastic / strategic
- Episodic vs. sequential
- Static vs. dynamic
- Discrete vs. continuous
- Single agent vs. multiagent

Environment Properties

- **Static vs. dynamic**
- If the environment can change while an agent is **deliberating**, then we say the environment is dynamic for that agent; otherwise, it is static
  - Taxi driving is clearly dynamic.
  - Crossword puzzles are static.

Environment Properties

- Fully observable vs. partially observable
- Deterministic vs. stochastic / strategic
- Episodic vs. sequential
- Static vs. dynamic
- Discrete vs. continuous
- Single agent vs. multiagent

Environment Properties

- Discrete vs. continuous
- The discrete/continuous distinction applies to the state of the environment, to the way time is handled, and to the percepts and actions of the agent.
- For example, the chess environment has a finite number of discrete states (excluding the clock).
  - Chess **also has a discrete set of percepts and actions**.
  - Taxi driving is a **continuous-state and continuous-time problem**

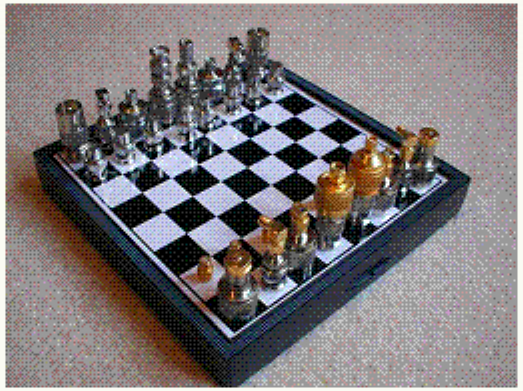
Environment Properties

- Fully observable vs. partially observable
- Deterministic vs. stochastic / strategic
- Episodic vs. sequential
- Static vs. dynamic
- Discrete vs. continuous
- Single agent vs. multiagent

Environment Properties

- Single agent vs. multiagent
- The distinction between **single-agent and multiagent environments** may seem simple enough.
  - For example, an agent solving a crossword puzzle by itself is clearly in a single-agent environment, whereas an agent playing chess is in a two-agent environment.

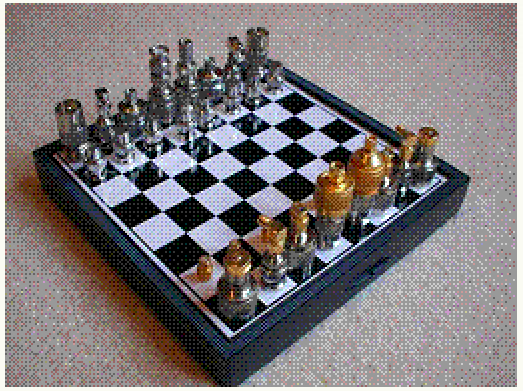
Environment Examples



Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Chess with a clock						
Chess without a clock						

Fully observable vs. partially observable
Deterministic vs. stochastic / strategic
Episodic vs. sequential
Static vs. dynamic
Discrete vs. continuous
Single agent vs. multiagent

Environment Examples



Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Chess with a clock	Fully	Strategic	Sequential	Semi	Discrete	Multi
Chess without a clock	Fully	Strategic	Sequential	Static	Discrete	Multi

Fully observable vs. partially observable
Deterministic vs. stochastic / strategic
Episodic vs. sequential
Static vs. dynamic
Discrete vs. continuous
Single agent vs. multiagent

Environment Examples

Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Chess with a clock	Fully	Strategic	Sequential	Semi	Discrete	Multi
Chess without a clock	Fully	Strategic	Sequential	Static	Discrete	Multi
Poker						

Fully observable vs. partially observable
Deterministic vs. stochastic / strategic
Episodic vs. sequential
Static vs. dynamic
Discrete vs. continuous
Single agent vs. multiagent

Environment Examples

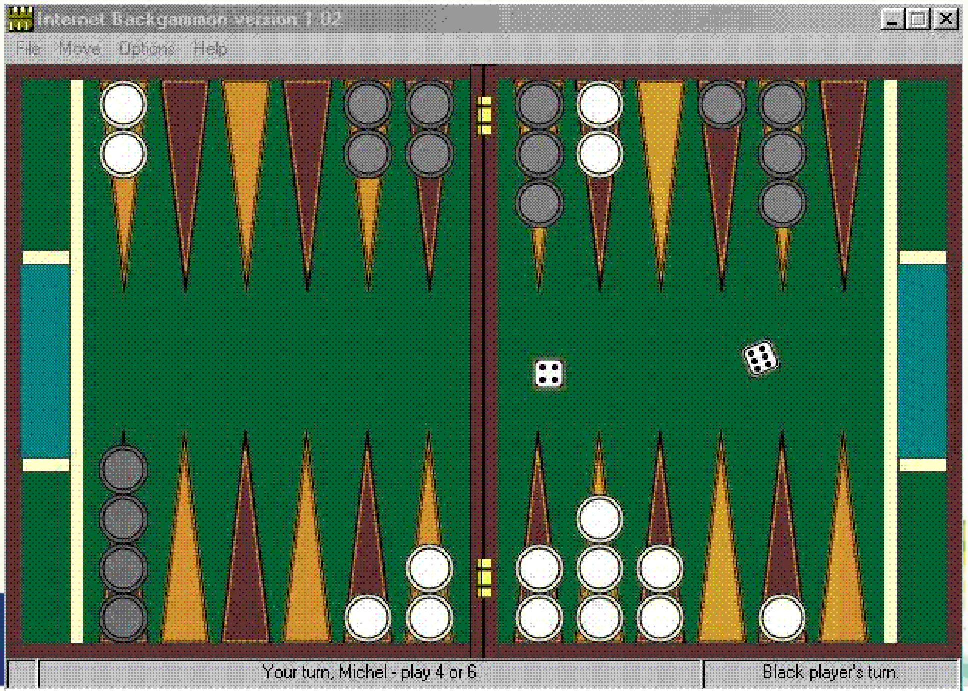
Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Chess with a clock	Fully	Strategic	Sequential	Semi	Discrete	Multi
Chess without a clock	Fully	Strategic	Sequential	Static	Discrete	Multi
Poker	Partial	Strategic	Sequential	Static	Discrete	Multi

Fully observable vs. partially observable
Deterministic vs. stochastic / strategic
Episodic vs. sequential
Static vs. dynamic
Discrete vs. continuous
Single agent vs. multiagent

Environment Examples

Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Chess with a clock	Fully	Strategic	Sequential	Semi	Discrete	Multi
Chess without a clock	Fully	Strategic	Sequential	Static	Discrete	Multi
Poker	Partial	Strategic	Sequential	Static	Discrete	Multi
Backgammon						

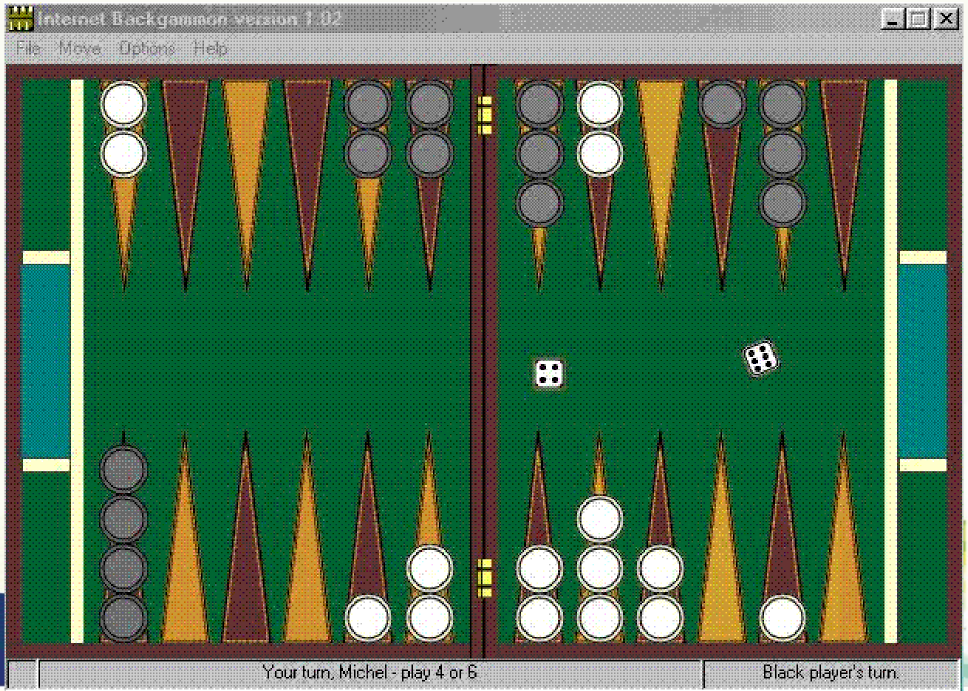
Fully observable vs. partially observable
Deterministic vs. stochastic / strategic
Episodic vs. sequential
Static vs. dynamic
Discrete vs. continuous
Single agent vs. multiagent


Internet Backgammon version 1.02
File Move Options Help
Your turn, Michel - play 4 or 6
Black player's turn.

Environment Examples

Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Chess with a clock	Fully	Strategic	Sequential	Semi	Discrete	Multi
Chess without a clock	Fully	Strategic	Sequential	Static	Discrete	Multi
Poker	Partial	Strategic	Sequential	Static	Discrete	Multi
Backgammon	Fully	Stochastic	Sequential	Static	Discrete	Multi

Fully observable vs. partially observable
Deterministic vs. stochastic / strategic
Episodic vs. sequential
Static vs. dynamic
Discrete vs. continuous
Single agent vs. multiagent



Your turn, Michel - play 4 or 6

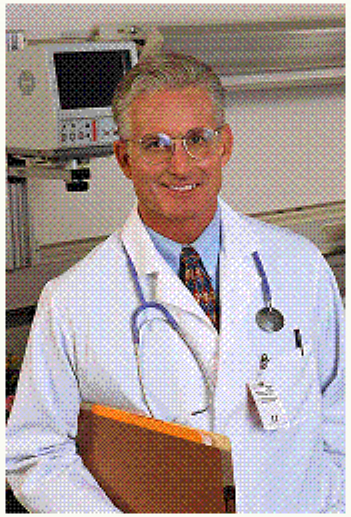
Black player's turn.

Environment Examples

Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Chess with a clock	Fully	Strategic	Sequential	Semi	Discrete	Multi
Chess without a clock	Fully	Strategic	Sequential	Static	Discrete	Multi
Poker	Partial	Strategic	Sequential	Static	Discrete	Multi
Backgammon	Fully	Stochastic	Sequential	Static	Discrete	Multi
Taxi driving	Partial	Stochastic	Sequential	Dynamic	Continuous	Multi

Fully observable vs. partially observable
Deterministic vs. stochastic / strategic
Episodic vs. sequential
Static vs. dynamic
Discrete vs. continuous
Single agent vs. multiagent

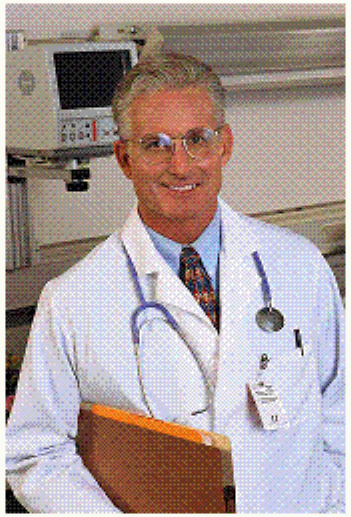
Environment Examples



Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Chess with a clock	Fully	Strategic	Sequential	Semi	Discrete	Multi
Chess without a clock	Fully	Strategic	Sequential	Static	Discrete	Multi
Poker	Partial	Strategic	Sequential	Static	Discrete	Multi
Backgammon	Fully	Stochastic	Sequential	Static	Discrete	Multi
Taxi driving	Partial	Stochastic	Sequential	Dynamic	Continuous	Multi
Medical diagnosis						

Fully observable vs. partially observable
Deterministic vs. stochastic / strategic
Episodic vs. sequential
Static vs. dynamic
Discrete vs. continuous
Single agent vs. multiagent

Environment Examples



Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Chess with a clock	Fully	Strategic	Sequential	Semi	Discrete	Multi
Chess without a clock	Fully	Strategic	Sequential	Static	Discrete	Multi
Poker	Partial	Strategic	Sequential	Static	Discrete	Multi
Backgammon	Fully	Stochastic	Sequential	Static	Discrete	Multi
Taxi driving	Partial	Stochastic	Sequential	Dynamic	Continuous	Multi
Medical diagnosis	Partial	Stochastic	Episodic	Static	Continuous	Single

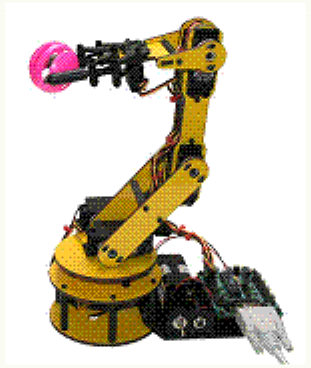
Environment Examples

Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Chess with a clock	Fully	Strategic	Sequential	Semi	Discrete	Multi
Chess without a clock	Fully	Strategic	Sequential	Static	Discrete	Multi
Poker	Partial	Strategic	Sequential	Static	Discrete	Multi
Backgammon	Fully	Stochastic	Sequential	Static	Discrete	Multi
Taxi driving	Partial	Stochastic	Sequential	Dynamic	Continuous	Multi
Medical diagnosis	Partial	Stochastic	Episodic	Static	Continuous	Single
Image analysis						

Environment Examples

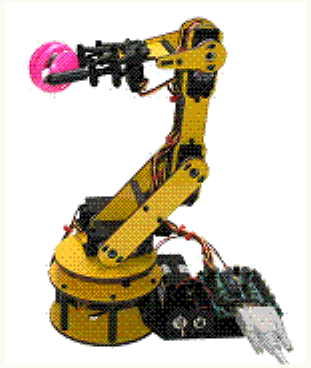
Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Chess with a clock	Fully	Strategic	Sequential	Semi	Discrete	Multi
Chess without a clock	Fully	Strategic	Sequential	Static	Discrete	Multi
Poker	Partial	Strategic	Sequential	Static	Discrete	Multi
Backgammon	Fully	Stochastic	Sequential	Static	Discrete	Multi
Taxi driving	Partial	Stochastic	Sequential	Dynamic	Continuous	Multi
Medical diagnosis	Partial	Stochastic	Episodic	Static	Continuous	Single
Image analysis	Fully	Deterministic	Episodic	Semi	Discrete	Single

Environment Examples



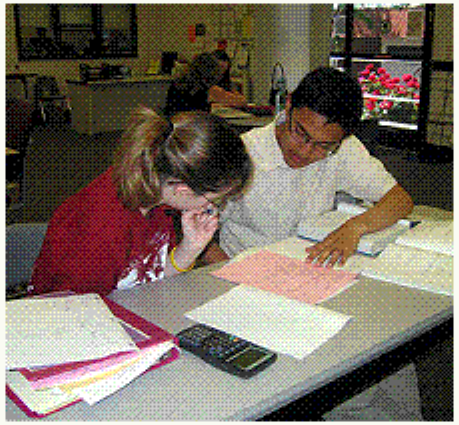
Environment	Obser vable	Determi nistic	Episodic	Static	Discrete	Agents
Chess with a clock	Fully	Strategic	Sequential	Semi	Discrete	Multi
Chess without a clock	Fully	Strategic	Sequential	Static	Discrete	Multi
Poker	Partial	Strategic	Sequential	Static	Discrete	Multi
Backgammon	Fully	Stochast ic	Sequential	Static	Discrete	Multi
Taxi driving	Partial	Stochast ic	Sequential	Dyna mic	Continu ous	Multi
Medical diagnosis	Partial	Stochast ic	Episodic	Static	Continu ous	Single
Image analysis	Fully	Determi nistic	Episodic	Semi	Discrete	Single
Robot part picking						

Environment Examples



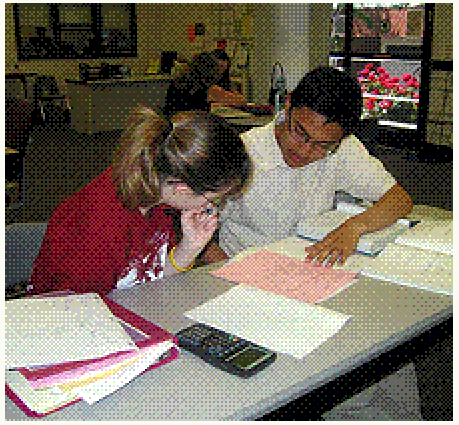
Environment	Obser vable	Determi nistic	Episodic	Static	Discrete	Agents
Chess with a clock	Fully	Strategic	Sequential	Semi	Discrete	Multi
Chess without a clock	Fully	Strategic	Sequential	Static	Discrete	Multi
Poker	Partial	Strategic	Sequential	Static	Discrete	Multi
Backgammon	Fully	Stochast ic	Sequential	Static	Discrete	Multi
Taxi driving	Partial	Stochast ic	Sequential	Dyna mic	Continu ous	Multi
Medical diagnosis	Partial	Stochast ic	Episodic	Static	Continu ous	Single
Image analysis	Fully	Determi nistic	Episodic	Semi	Discrete	Single
Robot part picking	Fully	Determi nistic	Episodic	Semi	Discrete	Single

Environment Examples



Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Chess with a clock	Fully	Strategic	Sequential	Semi	Discrete	Multi
Chess without a clock	Fully	Strategic	Sequential	Static	Discrete	Multi
Poker	Partial	Strategic	Sequential	Static	Discrete	Multi
Backgammon	Fully	Stochastic	Sequential	Static	Discrete	Multi
Taxi driving	Partial	Stochastic	Sequential	Dynamic	Continuous	Multi
Medical diagnosis	Partial	Stochastic	Episodic	Static	Continuous	Single
Image analysis	Fully	Deterministic	Episodic	Semi	Discrete	Single
Robot part picking	Fully	Deterministic	Episodic	Semi	Discrete	Single
Interactive English tutor						

Environment Examples



Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Chess with a clock	Fully	Strategic	Sequential	Semi	Discrete	Multi
Chess without a clock	Fully	Strategic	Sequential	Static	Discrete	Multi
Poker	Partial	Strategic	Sequential	Static	Discrete	Multi
Backgammon	Fully	Stochastic	Sequential	Static	Discrete	Multi
Taxi driving	Partial	Stochastic	Sequential	Dynamic	Continuous	Multi
Medical diagnosis	Partial	Stochastic	Episodic	Static	Continuous	Single
Image analysis	Fully	Deterministic	Episodic	Semi	Discrete	Single
Robot part picking	Fully	Deterministic	Episodic	Semi	Discrete	Single
Interactive English Assistance	Partial	Stochastic	Sequential	Dynamic	Discrete	Multi

Agent Types

- The job of AI is to design an **agent program** that implements the agent function—the mapping from percepts to actions.
- We assume this program will run on some sort of computing device with physical sensors and actuators—we call this the **agent architecture**

agent = architecture + program

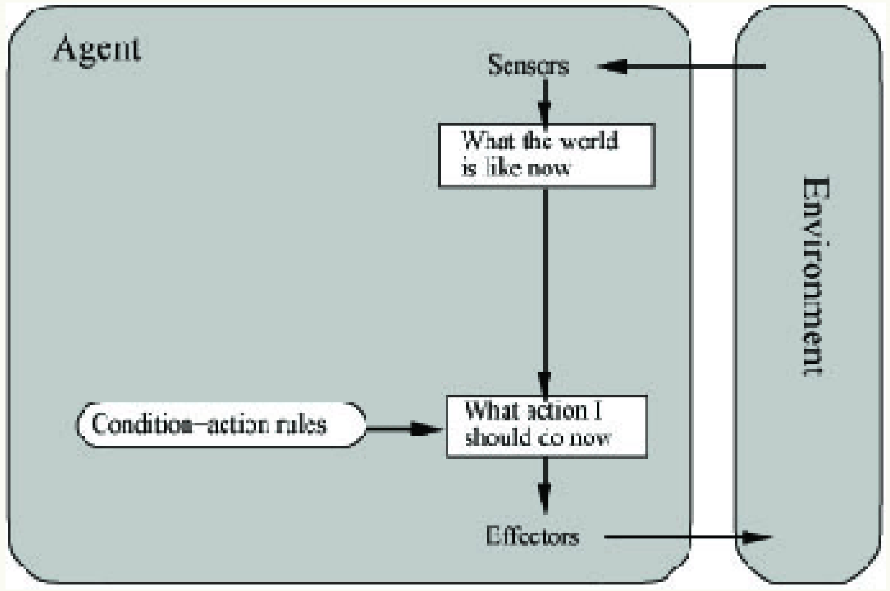
- If the program is going to recommend actions like Walk,
  - The architecture had better have legs.
  - The architecture might be just an ordinary PC, or it might be a robotic car with several onboard computers, cameras, and other sensors.

Agent Types

- Types of agents:
  - Simple reflex agents
  - Model-based reflex agents;
  - Goal-based agents
  - Utility-based agents
  - Learning agent

Simple Reflex Agent

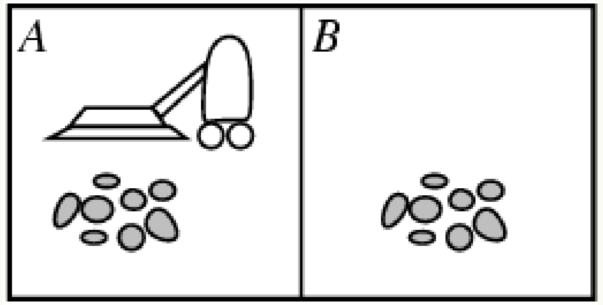
- These agents select actions on the basis of the current percept, ignoring the rest of the percept history.
- Use simple "if then" rules
- Can be short sighted



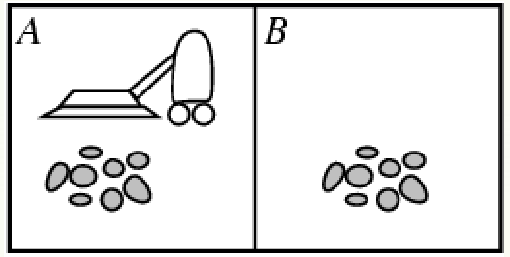
```
SimpleReflexAgent(percept)
   state   = InterpretInput(percept)
   rule     = RuleMatch(state, rules)
   action = RuleAction(rule)
   Return action
```

Example: Vacuum Agent

- Environment: vacuum, dirt, multiple areas defined by square regions
- Actions: left, right, suck, idle
- Sensors: location and contents
  - [A, dirty]



Reflex Vacuum Agent



- **a condition–action rule**
- **if** *car-in-front-is-braking* **then** *initiate-braking*
- the condition–action rules allow the agent to make the connection from percept to action.

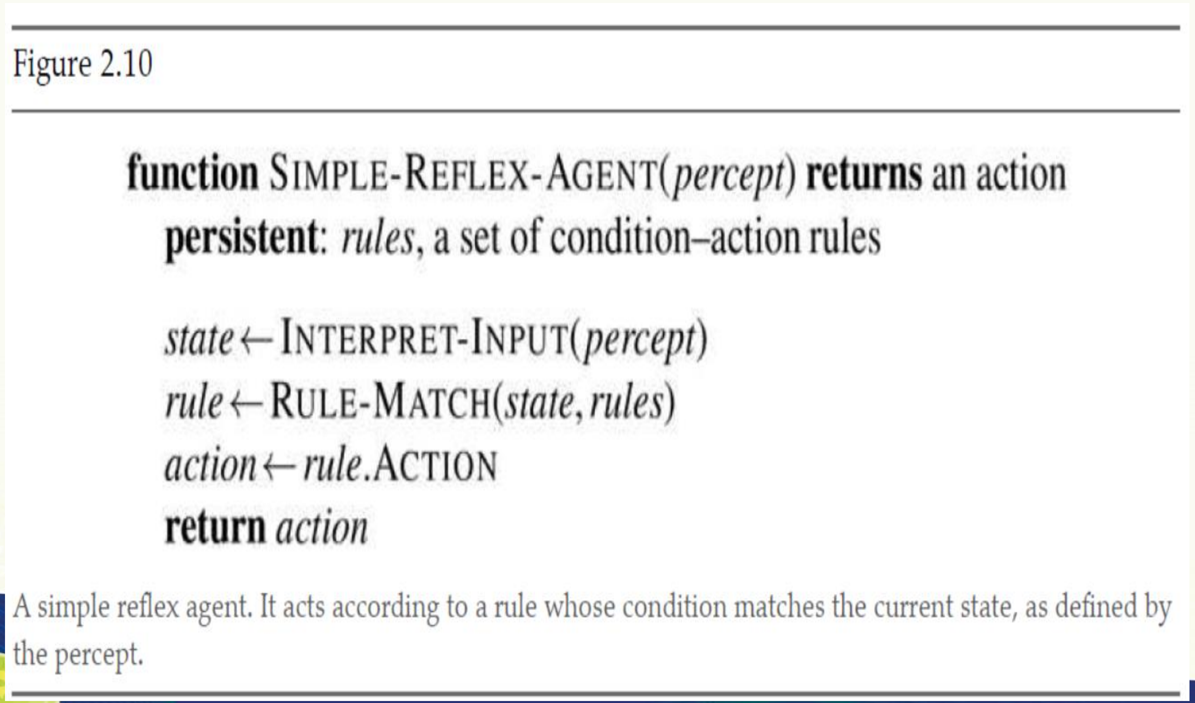
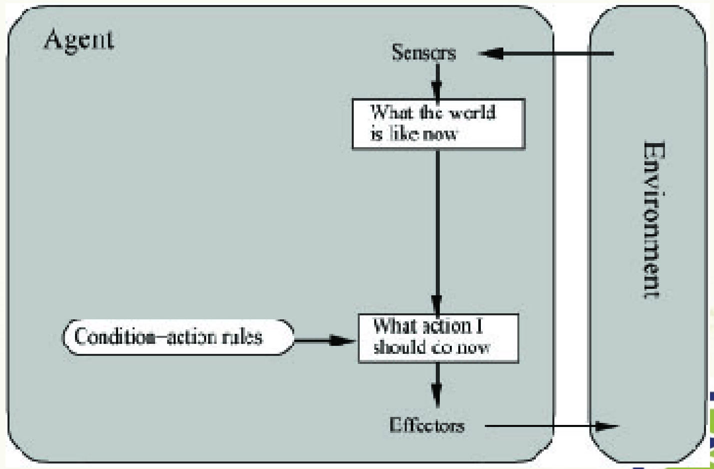


Figure 2.10

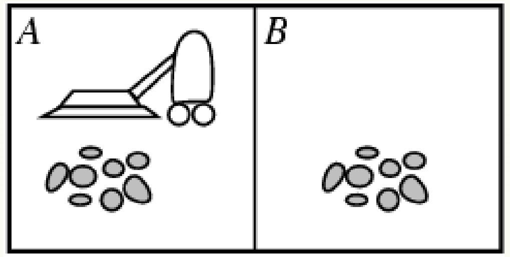
```
function SIMPLE-REFLEX-AGENT(percept) returns an action
    persistent: rules, a set of condition–action rules

    state ← INTERPRET-INPUT(percept)
    rule ← RULE-MATCH(state, rules)
    action ← rule.ACTION
    return action
```

A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.



Reflex Vacuum Agent



- If status=Dirty then return Suck
  else if location=A then return Right
  else if location=B then right Left

Reflex Vacuum Agent

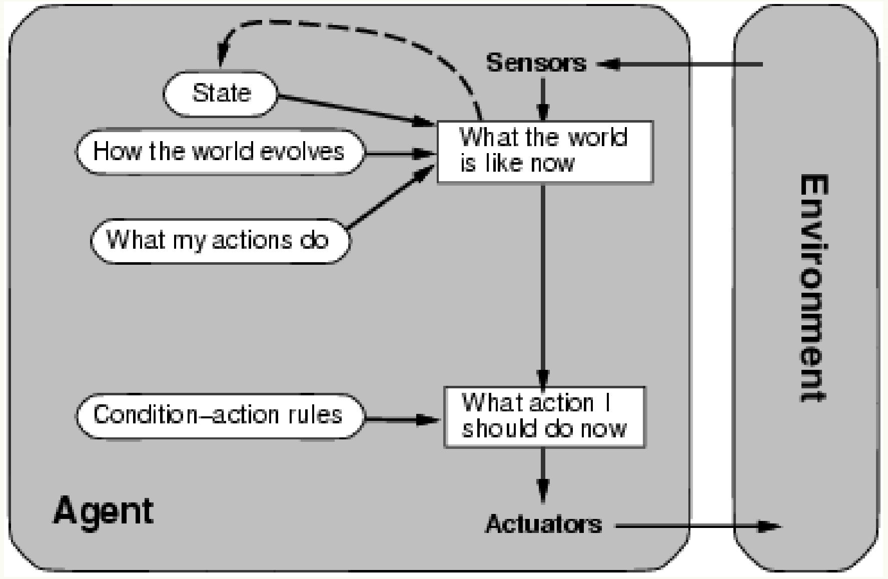
- The **INTERPRET-INPUT** function generates an abstracted description of the current state from the percept, and
- The RULEMATCH function returns the first rule in the set of rules that matches the given state description.
- Actual implementations can be as simple as a collection of logic gates implementing a Boolean circuit.
- **Limitations**:
  1. will work *only if the correct decision can be made on the basis of just the current percept—that is, only if the environment is fully observable.*
     - This works if the car in front has a centrally mounted brake light
  2. infinite loops

Reflex Vacuum Agent

- Suppose that a simple reflex
  - vacuum agent is lacked to its location sensor and has only a dirt sensor
  - Such an agent has just two possible percepts: [Dirty] and [Clean]. It can Suck in response to [Dirty] ; what should it do in response to [Clean] ?
  - Moving Left fails & Moving Right fails
  - Solution: Escape from infinite loops is possible if the agent can randomize its actions
  - it might flip a coin to choose between Left and Right.

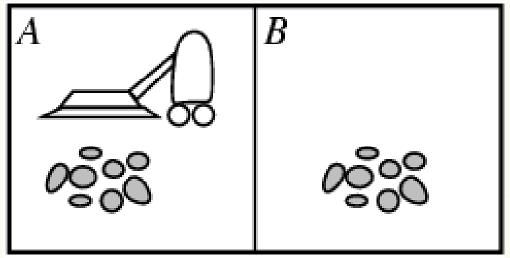
Reflex Agent With State (Model-based reflex agents)

- The most effective way to handle partial observability is for the agent to keep track of the part of the world it can't see now.
- Store previously-observed information
- Internal state



```
ReflexAgentWithState(percept)
  state   = UpdateDate(state,action,percept)
  rule    = RuleMatch(state, rules)
  action = RuleAction(rule)
  Return action
```

Reflex Agent With State

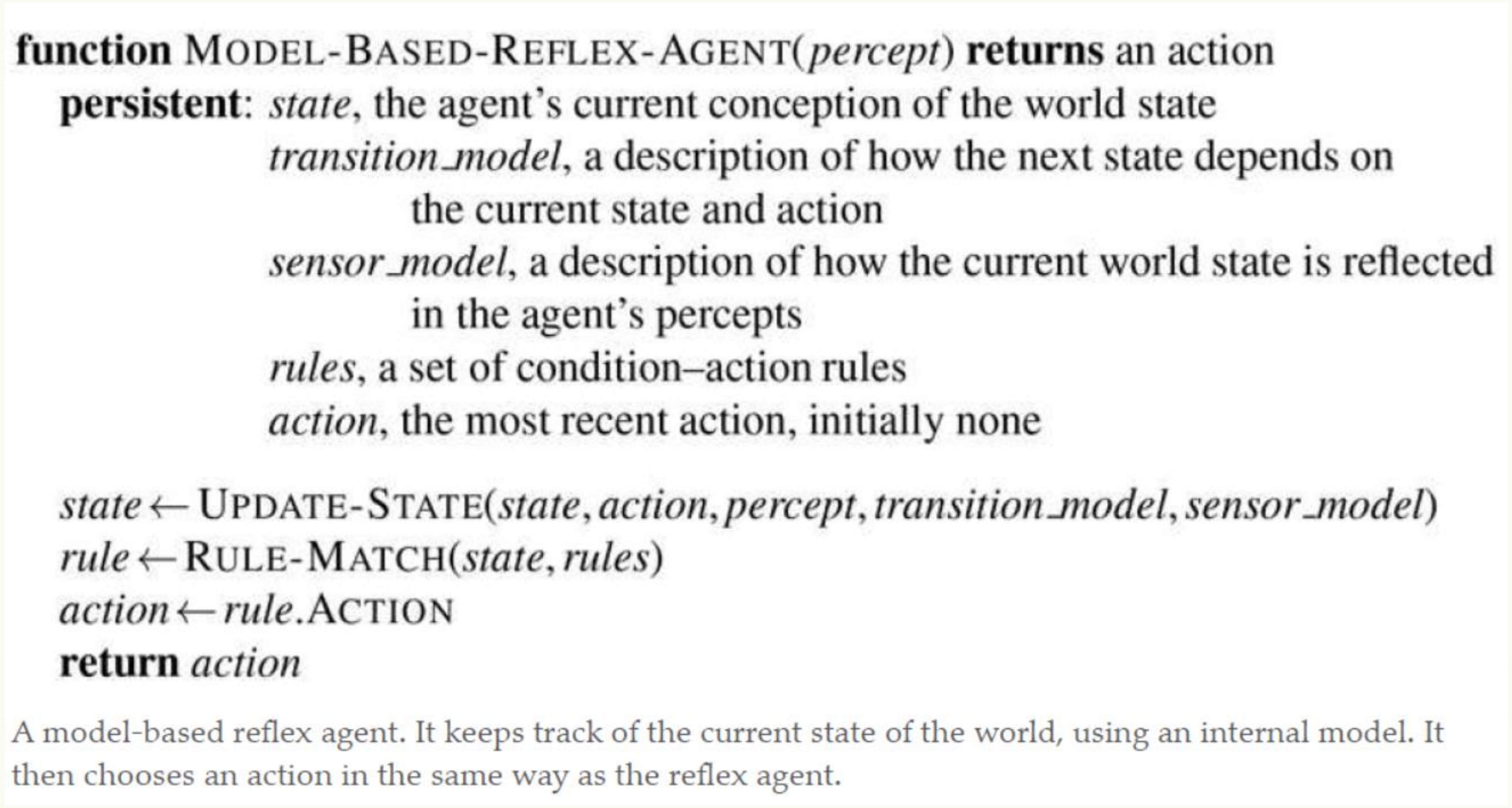


- For the braking problem, the internal state is not too extensive—just the previous frame from the camera, allowing the agent to detect when two red lights at the edge of the vehicle go on or off simultaneously.
- Updating this internal state information requires two kinds of knowledge to be encoded in the agent program.
- First, we need some information about how the world changes over time, which can be divided roughly into two parts:
  - The effects of the agent's actions and how the world evolves independently of the agent.
  - This knowledge about "how the world works"—whether implemented in simple Boolean circuits or in complete scientific theories—is called a **transition model** of the world.

Model-based reflex agents

- Second, we need some information about how the state of the world is reflected in the agent's percepts.
- This kind of knowledge is called a **sensor model**.
- Together, the transition model and sensor model allow an agent to keep track of the state of the world.
- An agent that uses such models is called a **model-based agent**.

Model-based reflex agents



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

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

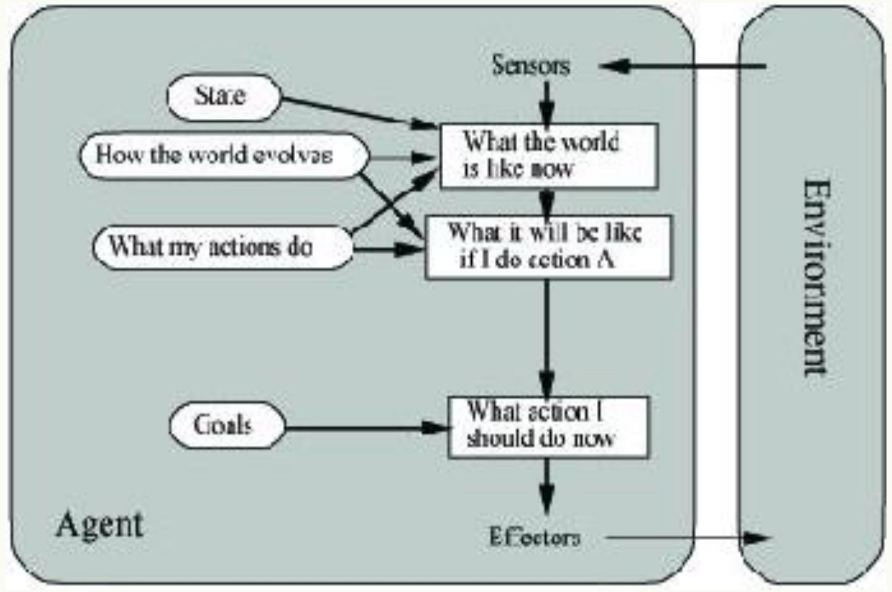
A model-based reflex agent. It keeps track of the current state of the world, using an internal model. It then chooses an action in the same way as the reflex agent.

Goal-Based Agents

- Knowing something about the current state of the environment is not always enough to decide what to do.
- For example, at a road junction, the taxi can turn left, turn right, or go straight on. The correct decision depends on where the taxi is trying to get to.
- the agent needs some sort of **goal** information that describes situations that are desirable.

Goal-Based Agents

- Goal reflects desires of agents
- May project actions to see if consistent with goals
- Takes time, world may change during reasoning


Sensors
State
How the world evolves
What my actions do
What the world is like now
What it will be like if I do action A
Goals
What action I should do now
Agent
Effectors
Environment

Utility-Based Agents

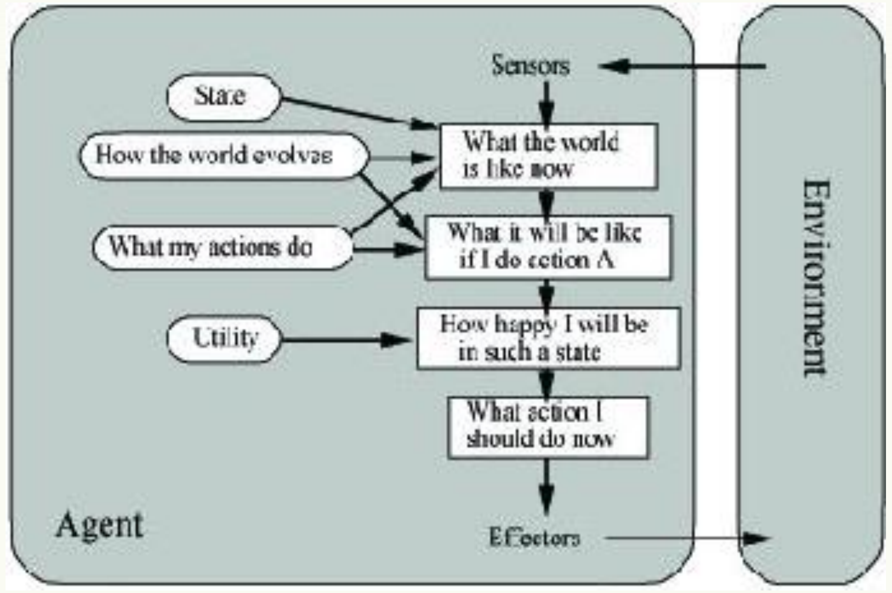
- Goals alone are not enough to generate high-quality behavior in most environments.
- For example, many action sequences will get the taxi to its destination (thereby achieving the goal), but some are quicker, more reliable, or cheaper than others. Goals just provide a simple binary distinction between “happy” and “unhappy” states.
- A more general performance measure should allow a comparison of different world states according to exactly how happy they would make the agent.
- Economists and computer scientists use the term utility instead.

Utility-Based Agents

- Furthermore, in two kinds of cases, goals are unfit, but a utility-based agent can still make rational decisions.
  - First, when there are conflicting goals, only some of which can be achieved (for example, speed and safety), the utility function specifies the appropriate tradeoff.
  - Second, when there are several goals that the agent can aim for utility provides a way in which the likelihood of success can be weighed against the importance of the goals.

Utility-Based Agents

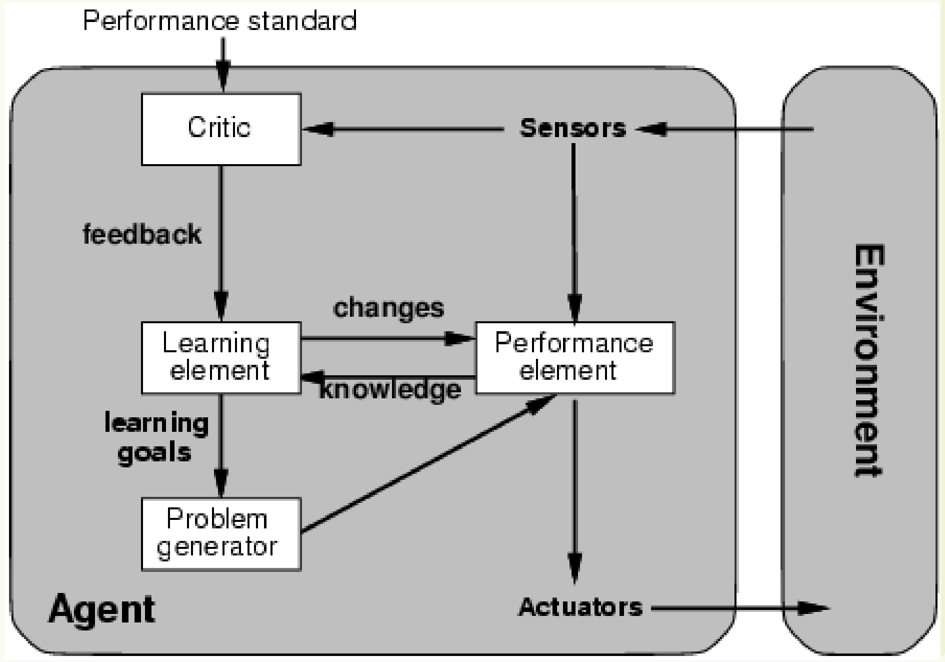
- Evaluation function to measure utility f(state) -> value
- Useful for evaluating competing goals


Sensors
State
How the world evolves
What my actions do
Utility
What the world is like now
What it will be like if I do action A
How happy I will be in such a state
What action I should do now
Agent
Effectors
Environment

Learning Agents

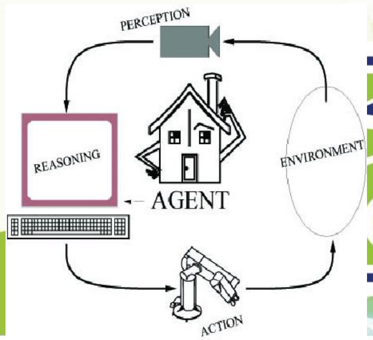
A learning agent can be divided into four conceptual components:

- **learning element**, which is responsible for making improvements,
- **Performance element,** which is responsible for selecting external actions. The performance element is what we have previously considered to be the entire agent: it takes in percepts and decides on actions.
- The **learning element** uses feedback from the **critic** on how the agent **is doing and determines** how the performance element should be **modified to do better** in the future.
- **Problem generator** is responsible for suggesting actions that will lead to new and informative experiences (exploratory).



Pathfinder Medical Diagnosis System

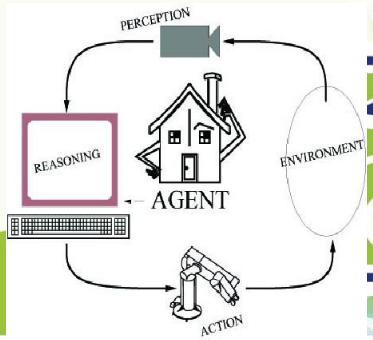
- Performance: Correct Hematopathology diagnosis
- Environment: Automate human diagnosis, partially observable, deterministic, episodic, static, continuous, single agent
- Actuators: Output diagnoses and further test suggestions
- Sensors: Input is test results
- Reasoning: Bayesian networks, Monte-Carlo simulations



TDGammon

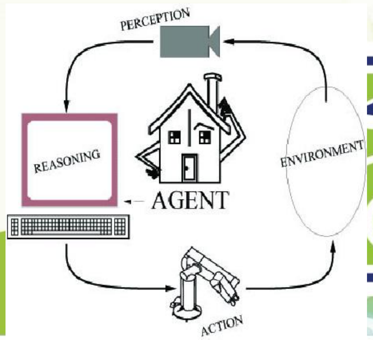
- Performance: Ratio of wins to losses
- Environment: Graphical output showing dice roll and piece movement, fully observable, stochastic, sequential, static, discrete, multiagent

  World Champion Backgammon Player
- Sensors: Keyboard input
- Actuator: Numbers representing moves of pieces
- Reasoning: Reinforcement learning, neural networks



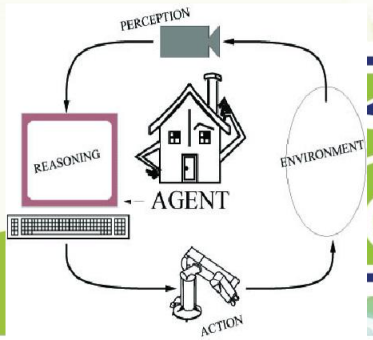
Alvinn

- Performance: Stay in lane, on road, maintain speed
- Environment: Driving Hummer on and off road without manual control (Partially observable, stochastic, episodic, dynamic, continuous, single agent), Autonomous automobile
- Actuators: Speed, Steer
- Sensors: Stereo camera input
- Reasoning: Neural networks



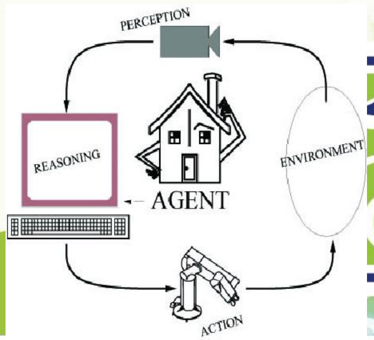
Talespin

- Performance: Entertainment value of generated story
- Environment: Generate text-based stories that are creative and understandable
  - One day Joe Bear was hungry. He asked his friend Irving Bird where some honey was. Irving told him there was a beehive in the oak tree. Joe threatened to hit Irving if he didn't tell him where some honey was.
  - Henry Squirrel was thirsty. He walked over to the river bank where his good friend Bill Bird was sitting. Henry slipped and fell in the river. Gravity drowned. Joe Bear was hungry. He asked Irving Bird where some honey was. Irving refused to tell him, so Joe offered to bring him a worm if he'd tell him where some honey was. Irving agreed. But Joe didn't know where any worms were, so he asked Irving, who refused to say. So Joe offered to bring him a worm if he'd tell him where a worm was. Irving agreed. But Joe didn't know where any worms were, so he asked Irving, who refused to say. So Joe offered to bring him a worm if he'd tell him where a worm was…
- Actuators: Add word/phrase, order parts of story
- Sensors: Dictionary, Facts and relationships stored in database
- Reasoning: Planning



Webcrawler Softbot

- Search web for items of interest
- Perception: Web pages
- Reasoning: Pattern matching
- Action: Select and traverse hyperlinks

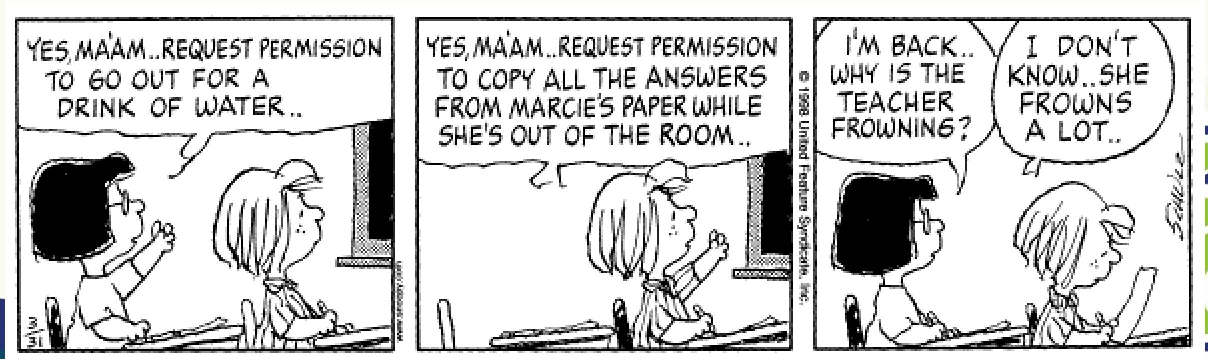


Other Example AI Systems

- Translation of Caterpillar truck manuals into 20 languages
- Shuttle packing
- Military planning (Desert Storm)
- Intelligent vehicle highway negotiation
- Credit card transaction monitoring
- Billiards robot
- Juggling robot
- Credit card fraud detection
- Lymphatic system diagnoses
- Mars rover
- Sky survey galaxy data analysis

Other Example AI Systems

- Knowledge Representation
- Search
- Problem solving
- Planning
- Machine learning
- Natural language processing
- Uncertainty reasoning
- Computer Vision
- Robotics


YES, MA'AM..REQUEST PERMISSION TO GO OUT FOR A DRINK OF WATER..
YES, MA'AM..REQUEST PERMISSION TO COPY ALL THE ANSWERS FROM MARCIE'S PAPER WHILE SHE'S OUT OF THE ROOM..
I'M BACK.. WHY IS THE TEACHER FROWNING?
I DON'T KNOW..SHE FROWNS A LOT..
© 1998 United Feature Syndicate, Inc.