Intelligent Agents

Dr. Steven Bethard

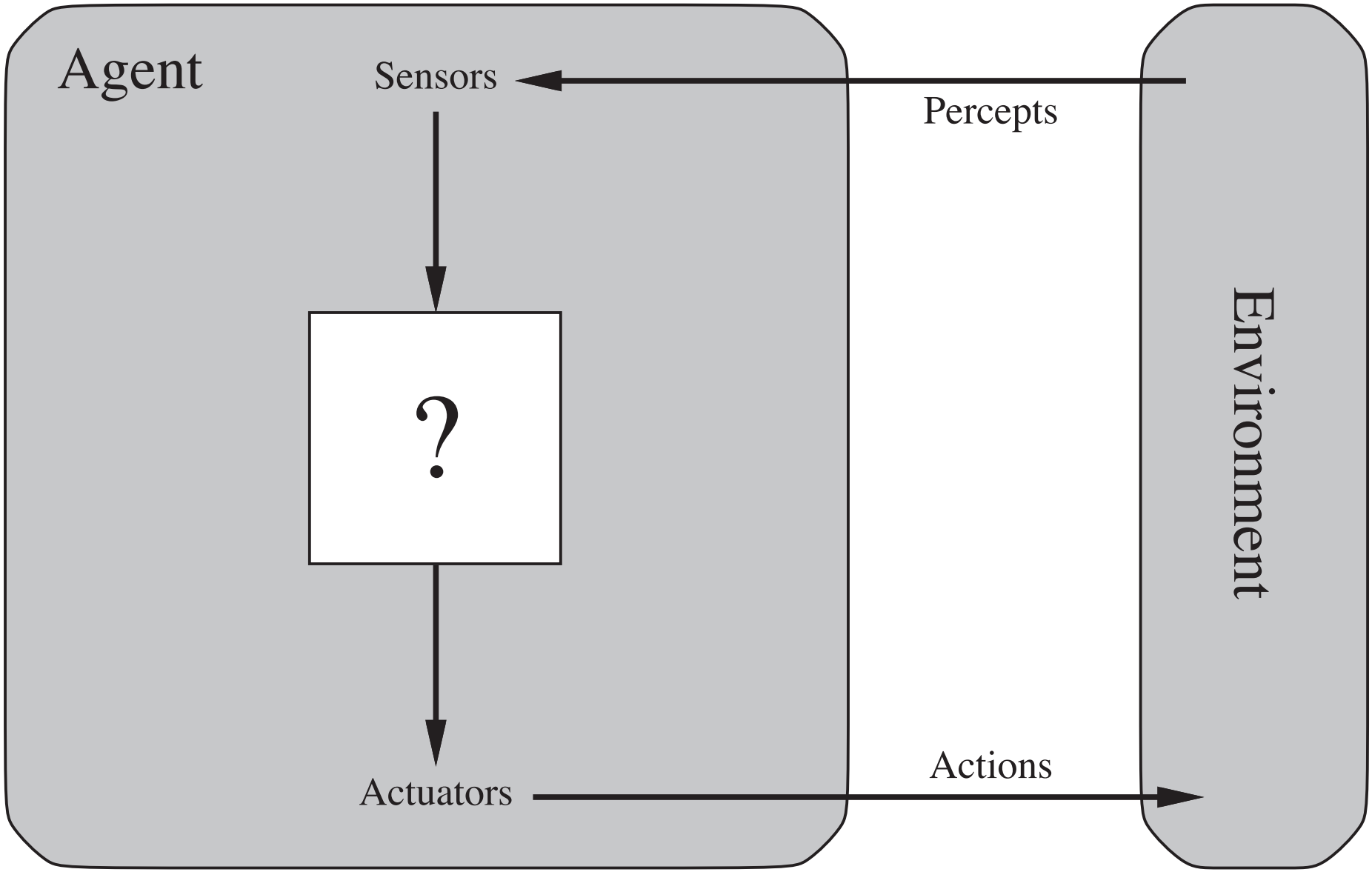
Computer and Information Sciences
University of Alabama at Birmingham

12 Jan 2016

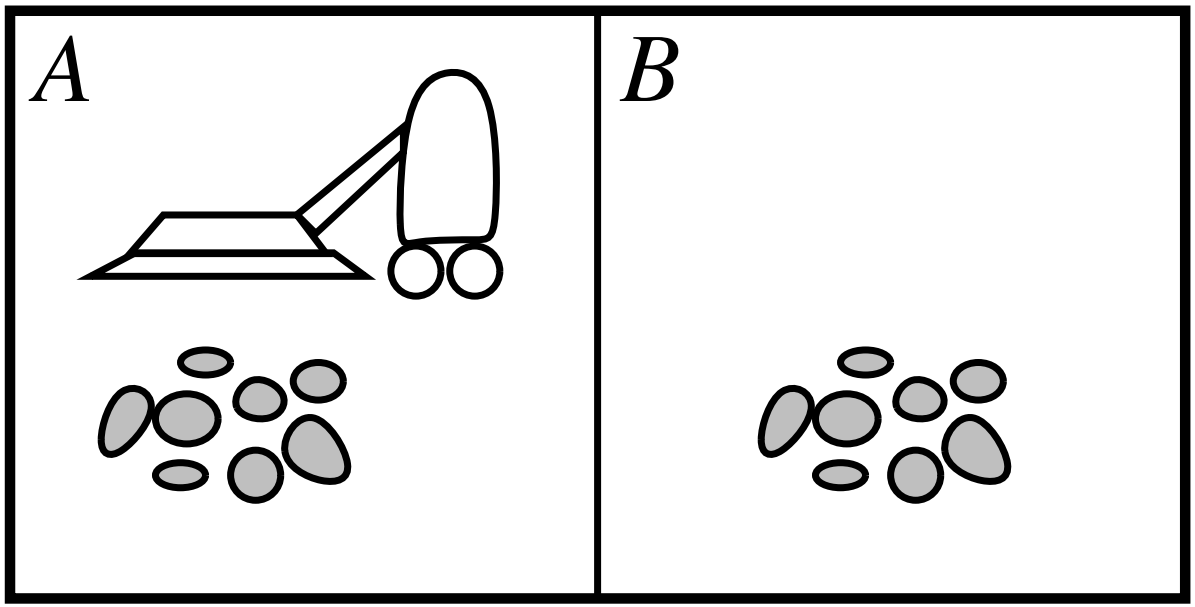
Outline

Outline

Agents and Environments



Vacuum Cleaner World



Percepts: Location and status, e.g. [*A*, *Dirty*]

Actions: *Left*, *Right*, *Suck*, *NoOp*

A Vacuum Cleaner Agent

```
class ReflexVacuumAgent(object):
    def act(self, percept):
        location, status = percept
        if status == "Dirty":
            return "Suck"
        elif location == "A":
            return "Right"
        elif location == "B":
            return "Left"
```

Rational Agents

Performance Measure

How successful was the agent?

E.g. the vacuum cleaner agent:

- *Maximized clean squares*
- *Minimized electricity consumed*

Rational Agent

Selects the action that is expected to maximize the performance measure

Rational Agents

Performance Measure

How successful was the agent?

E.g. the vacuum cleaner agent:

- *Maximized clean squares*
- *Minimized electricity consumed*

Rational Agent

Selects the action that is expected to maximize the performance measure

Rational Agents

Performance Measure

How successful was the agent?

E.g. the vacuum cleaner agent:

- *Maximized clean squares*
- *Minimized electricity consumed*

Rational Agent

Selects the action that is expected to maximize the performance measure

Rational vs. Omniscient

Rational?

- Left turn arrow was red. Didn't check for oncoming traffic. Turned left. Hit by a bus.
- Left turn arrow was green. Didn't check for oncoming traffic. Turned left. Hit by a bus.
- Left turn arrow was green. Checked for oncoming traffic, saw none. Turned left. Hit by bus.

Rational vs. Omniscient

Rational?

- Left turn arrow was red. Didn't check for oncoming traffic. Turned left. Hit by a bus.
- Left turn arrow was green. Didn't check for oncoming traffic. Turned left. Hit by a bus.
- Left turn arrow was green. Checked for oncoming traffic, saw none. Turned left. Hit by bus.

Rational vs. Omniscient

Rational?

- Left turn arrow was red. Didn't check for oncoming traffic. Turned left. Hit by a bus.
- Left turn arrow was green. Didn't check for oncoming traffic. Turned left. Hit by a bus.
- Left turn arrow was green. Checked for oncoming traffic, saw none. Turned left. Hit by bus.

Outline

Specifying a Driving Task

Performance measure

Environment

Actuators

Sensors

Specifying a Driving Task

Performance measure

safety, destination, speed, legality, comfort. . .

Environment

Actuators

Sensors

Specifying a Driving Task

Performance measure

safety, destination, speed, legality, comfort. . .

Environment

streets/freeways, traffic, pedestrians, weather. . .

Actuators

Sensors

Specifying a Driving Task

Performance measure

safety, destination, speed, legality, comfort. . .

Environment

streets/freeways, traffic, pedestrians, weather. . .

Actuators

steering, accelerator, brake, speaker/display. . .

Sensors

Specifying a Driving Task

Performance measure

safety, destination, speed, legality, comfort. . .

Environment

streets/freeways, traffic, pedestrians, weather. . .

Actuators

steering, accelerator, brake, speaker/display. . .

Sensors

video, accelerometer, microphone, GPS. . .

Describing Environments

Fully vs. Partially Observable

Fully	All relevant to action is visible, e.g. *chess*
Partially	Part of environment unavailable, e.g. *poker*

Deterministic vs. Strategic vs. Stochastic

Determin.	State + action determines next state, e.g. *crossword*
Strategic	State + action + other agent actions determines next state, e.g. *chess*
Stochastic	Next state not fully determined, e.g. *poker*

Describing Environments

Episodic vs. Sequential

Episodic	Old actions irrelevant, e.g. *face detection*
Sequential	Old actions affect current state, e.g. *chess*

Static vs. Semidynamic vs. Dynamic

Static	Environment does not change while deciding, e.g. *chess, poker*
Semi	Performance score changes while deciding, e.g. *face detection*
Dynamic	Environment changes while deciding, e.g. *driving*

Describing Environments

Discrete vs. Continuous

Discrete States, percepts and actions are countable, e.g. *chess, poker*

Continuous States, percepts or actions are real-valued, e.g. *driving*

Single vs. Multiple Agents

Single Single agent, e.g. *crossword, face detection*

Multiple More than one agent, e.g. *poker, driving*

Example Environments

	Solitaire	Chess	Internet Shopping	Taxi
Observable				
Deterministic				
Episodic				
Static				
Discrete				
Single-agent				

Example Environments

	Solitaire	Chess	Internet Shopping	Taxi
Observable	No	Yes	No	No
Deterministic				
Episodic				
Static				
Discrete				
Single-agent				

Example Environments

	Solitaire	Chess	Internet Shopping	Taxi
Observable	No	Yes	No	No
Deterministic	No	Strategic	Partly	No
Episodic				
Static				
Discrete				
Single-agent				

Example Environments

	Solitaire	Chess	Internet Shopping	Taxi
Observable	No	Yes	No	No
Deterministic	No	Strategic	Partly	No
Episodic	No	No	No	No
Static				
Discrete				
Single-agent				

Example Environments

	Solitaire	Chess	Internet Shopping	Taxi
Observable	No	Yes	No	No
Deterministic	No	Strategic	Partly	No
Episodic	No	No	No	No
Static	Yes	Yes	Semi	No
Discrete				
Single-agent				

Example Environments

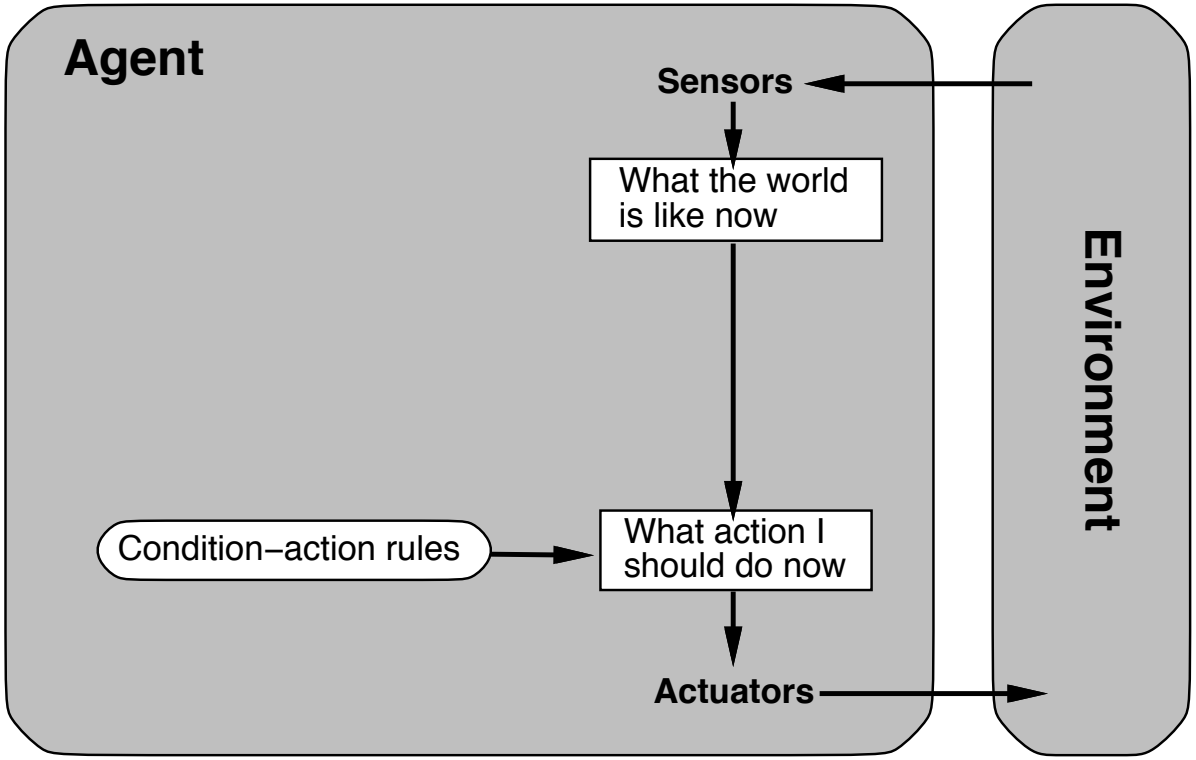
	Solitaire	Chess	Internet Shopping	Taxi
Observable	No	Yes	No	No
Deterministic	No	Strategic	Partly	No
Episodic	No	No	No	No
Static	Yes	Yes	Semi	No
Discrete	Yes	Yes	Yes	No
Single-agent				

Example Environments

	Solitaire	Chess	Internet Shopping	Taxi
Observable	No	Yes	No	No
Deterministic	No	Strategic	Partly	No
Episodic	No	No	No	No
Static	Yes	Yes	Semi	No
Discrete	Yes	Yes	Yes	No
Single-agent	Yes	No	Maybe	No

Outline

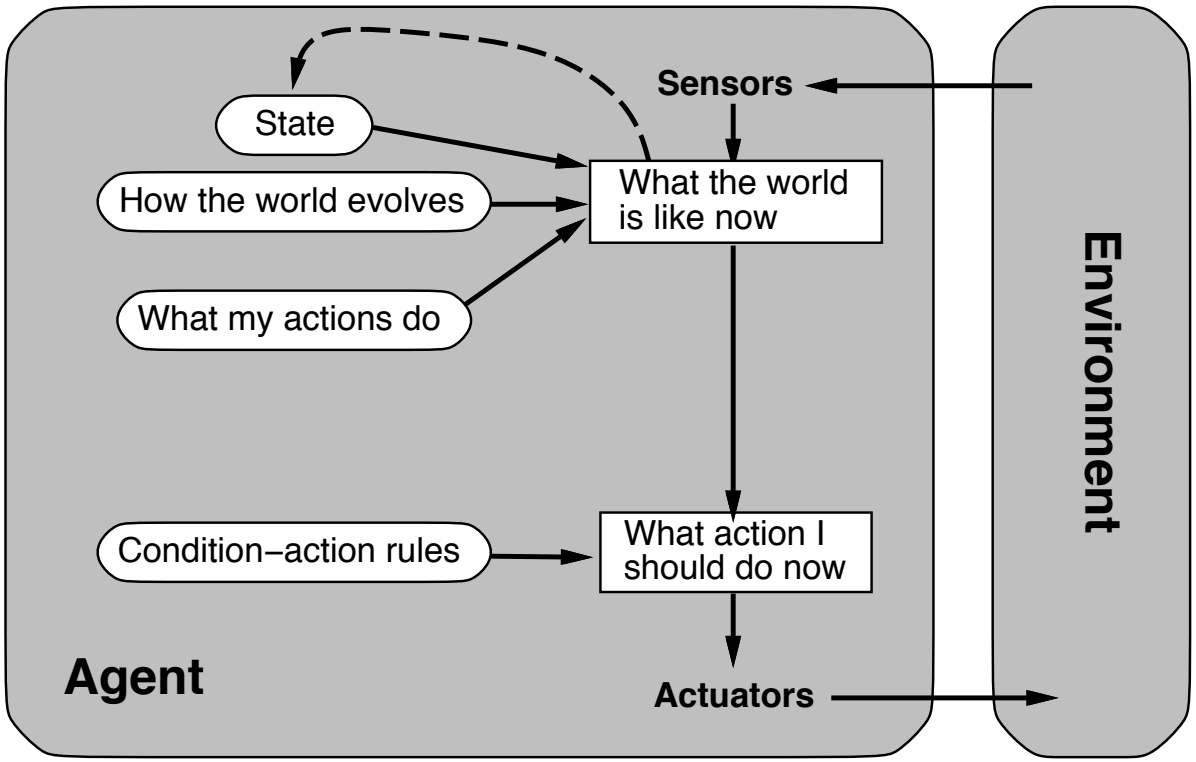
Simple Reflex Agents



Simple Reflex Agent Example

```
class ReflexVacuumAgent(object):
    def act(self, percept):
        location, status = percept
        if status == "Dirty":
            return "Suck"
        elif location == "A":
            return "Right"
        elif location == "B":
            return "Left"
```

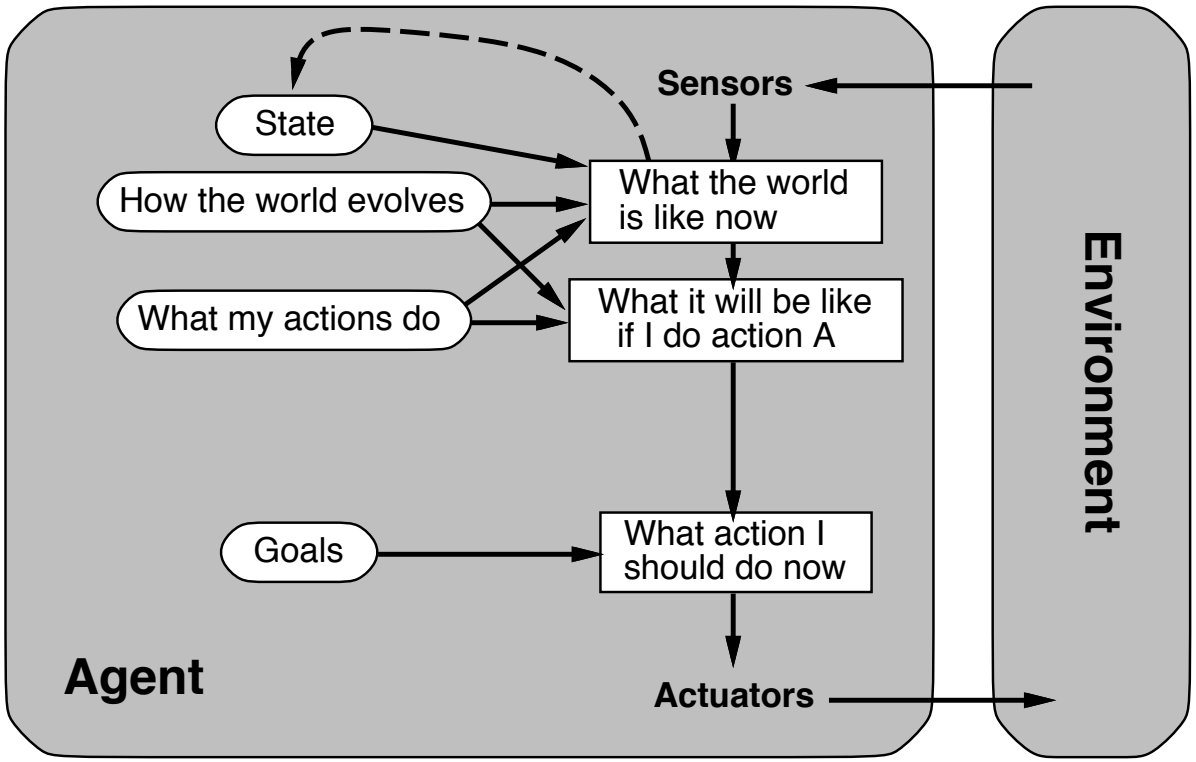
Stateful Reflex Agents



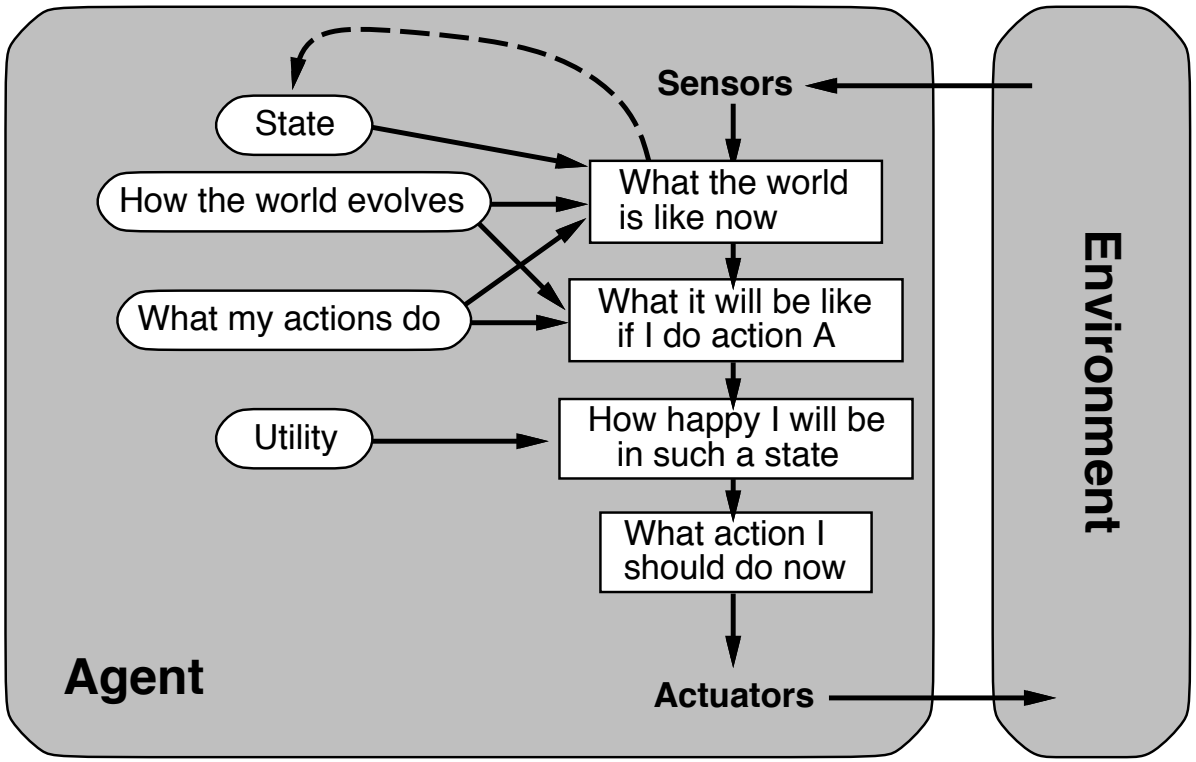
Stateful Reflex Agent Example

```
class StatefulReflexVacuumAgent(object):
    def __init__(self):
        self.time_at_location = 3
        self.directions = {"A": "Right", "B": "Left"}
    def act(self, percept):
        self.time_at_location += 1
        location, status = percept
        if status == "Dirty":
            return "Suck"
        elif self.time_at_location > 3:
            self.time_at_location = 0
            return self.directions[location]
        else:
            return "NoOp"
```

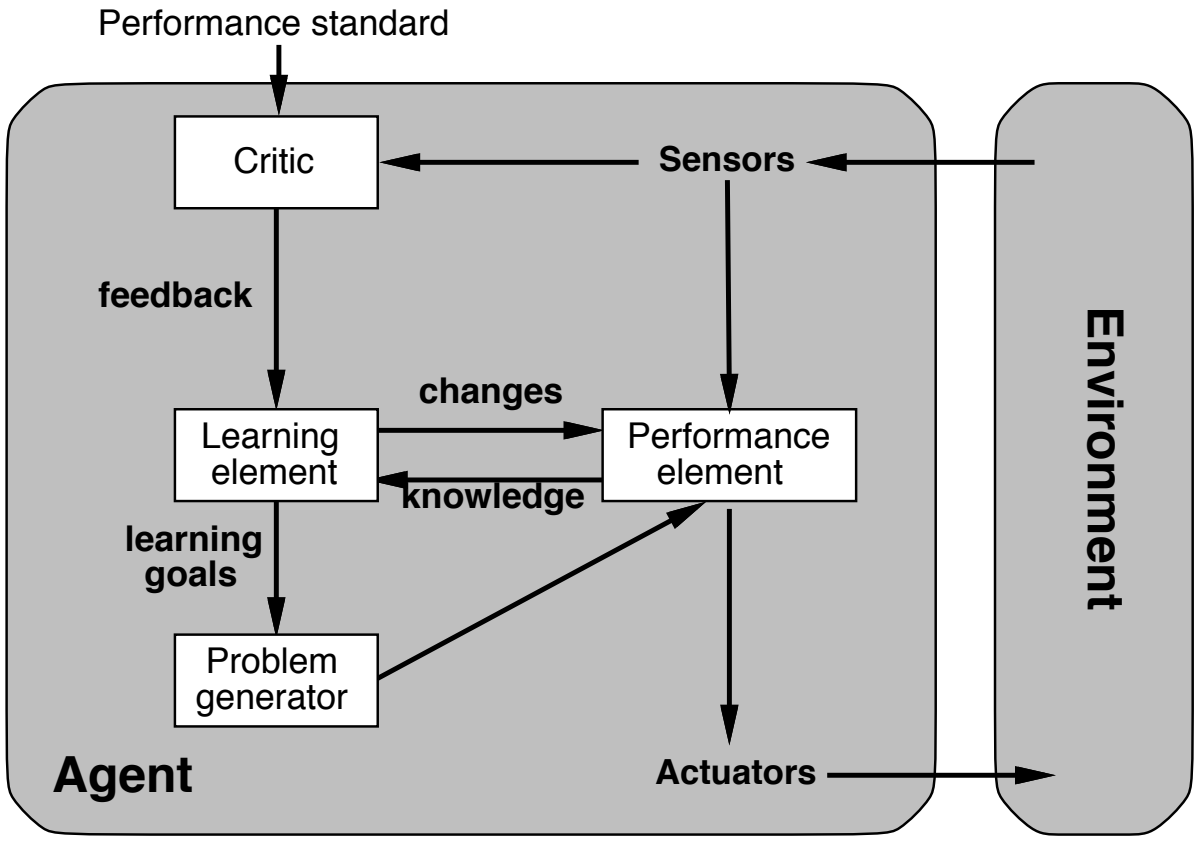
Goal-based Agents



Utility-based Agents



Learning Agents



Key Points

- Agents take actions based on percepts
- Rational agents maximize a performance measure
- Features of task environments:
  - Observable? Deterministic? Episodic?
    Static? Discrete? Single-agent?
- Agent architectures:
  - Reflex, Stateful reflex, Goal-based, Utility-based

Key Points

- Agents take actions based on percepts
- Rational agents maximize a performance measure
- Features of task environments:
  - Observable? Deterministic? Episodic?
    Static? Discrete? Single-agent?
- Agent architectures:
  - Reflex, Stateful reflex, Goal-based, Utility-based

Key Points

- Agents take actions based on percepts
- Rational agents maximize a performance measure
- Features of task environments:
  - Observable? Deterministic? Episodic?
    Static? Discrete? Single-agent?
- Agent architectures:
  - Reflex, Stateful reflex, Goal-based, Utility-based

Key Points

- Agents take actions based on percepts
- Rational agents maximize a performance measure
- Features of task environments:
  - Observable? Deterministic? Episodic?
    Static? Discrete? Single-agent?
- Agent architectures:
  - Reflex, Stateful reflex, Goal-based, Utility-based

Key Points

- Agents take actions based on percepts
- Rational agents maximize a performance measure
- Features of task environments:
  - Observable? Deterministic? Episodic?
    Static? Discrete? Single-agent?
- Agent architectures:
  - Reflex, Stateful reflex, Goal-based, Utility-based