Intelligent Agents

Chapter 2

Waitlist and auditing

- The class size will not be increased. I do not have the power to change the order of the waitlist.
- If you can't enroll, you are welcome to audit. I will try to make the web page available. You can come to class and office hours; however, we cannot grade your assignments.
- If you want access to the web page, please email me with the subject "Please add me to the CMPT 310 web page" and include your name and SFU ID.

Outline

- Agents and environments
- Rationality
- Task environment:

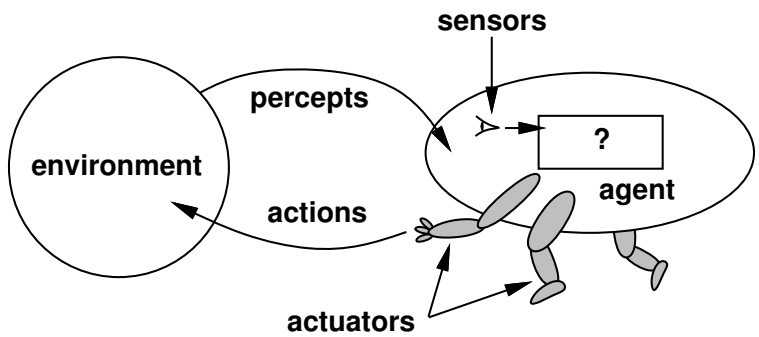
  PEAS:

  - *Performance measure*
  - *Environment*
  - *Actuators*
  - *Sensors*

- Environment types
- Agent types

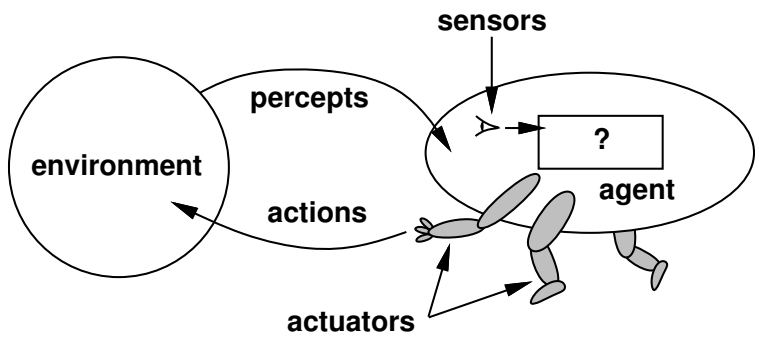
Agents and Environments

- An *agent* is anything that can be viewed as perceiving its *environment* through *sensors* and acting in that environment through *actuators*.



Agents and Environments

- An *agent* is anything that can be viewed as perceiving its *environment* through *sensors* and acting in that environment through *actuators*.

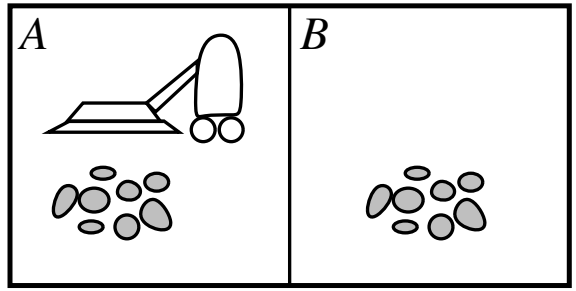


- *Agents* include humans, robots, softbots, thermostats, etc.
- The *agent function* maps from percept histories to actions:

$$f : \mathcal{P}^* \rightarrow \mathcal{A}$$

- The *agent program* runs on a physical *architecture* to give f

Vacuum-cleaner world



Percepts: location and contents, e.g., $[A, \mathit{Dirty}]$
Actions: *Left*, *Right*, *Suck*, *NoOp*

A vacuum-cleaner agent

Agent function:

Percept sequence	Action
$[A, Clean]$	*Right*
$[A, Dirty]$	*Suck*
$[B, Clean]$	*Left*
$[B, Dirty]$	*Suck*
$[A, Clean], [A, Clean]$	*Right*
$[A, Clean], [A, Dirty]$	*Suck*
...	...

Note: This says *how* the agent should function.

- It says nothing about how this should be implemented.

Rationality

Informally a *rational* agent is one that does the "right thing".

- *How well* an agent does is given by a *performance measure*.
- A fixed *performance measure* evaluates the *environment sequence*

  Examples:

  - one point per square cleaned up in time T?
  - one point per clean square per time step, minus one per move?
  - penalize for $> k$ dirty squares?
- A *rational agent* selects an action which maximizes the expected value of the performance measure given the percept sequence to date and its own knowledge.
- The action selection may range from being hardwired (e.g. in an insect or reflexive agent) to involving substantial reasoning.

Rationality

Notes:

- Rational $\neq$ omniscient
  - percepts may not supply all the relevant information
- Rational $\neq$ clairvoyant
  - action outcomes may not be as expected
- Hence, rational $\neq$ successful
- Full, general rationality requires exploration, learning, autonomy

The Task Environment

- To design a rational agent, we must specify the *task environment*
- The task environment has the following components:
  - Performance measure
  - Environment
  - Actuators
  - Sensors
- Acronym: PEAS

PEAS

Consider, e.g., the task of designing an automated taxi:

Performance measure: safety, destination, profits, legality, comfort, ...

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

Actuators: steering, accelerator, brake, horn, speaker/display, ...

Sensors: video, accelerometers, gauges, engine sensors, keyboard, GPS, ...

Internet shopping agent

Performance measure: ??

Environment: ??

Actuators: ??

Sensors: ??

Internet shopping agent

Performance measure: price, quality, appropriateness, efficiency

Environment: ??

Actuators: ??

Sensors: ??

Internet shopping agent

Performance measure: price, quality, appropriateness, efficiency

Environment: current and future WWW sites, vendors, shippers

Actuators: ??

Sensors: ??

Internet shopping agent

Performance measure: price, quality, appropriateness, efficiency

Environment: current and future WWW sites, vendors, shippers

Actuators: display to user, follow URL, fill in form

Sensors: ??

Internet shopping agent

Performance measure: price, quality, appropriateness, efficiency

Environment: current and future WWW sites, vendors, shippers

Actuators: display to user, follow URL, fill in form

Sensors: HTML pages (text, graphics, scripts)

Environment Types

- *Fully observable* vs. *partially observable*
  - If the agent has access to full state of the environment or not

Environment Types

- *Fully observable* vs. *partially observable*
  - If the agent has access to full state of the environment or not
- *Deterministic* vs. *stochastic* vs. *nondeterministic*
  - Deterministic: Next state is completely determined by the agent's actions.

Environment Types

- *Fully observable* vs. *partially observable*
  - If the agent has access to full state of the environment or not
- *Deterministic* vs. *stochastic* vs. *nondeterministic*
  - Deterministic: Next state is completely determined by the agent's actions.

☞ *Uncertain*: not fully observable or not deterministic

Environment Types

- *Fully observable* vs. *partially observable*
  - If the agent has access to full state of the environment or not
- *Deterministic* vs. *stochastic* vs. *nondeterministic*
  - Deterministic: Next state is completely determined by the agent's actions.
- ☞ *Uncertain*: not fully observable or not deterministic
- *Episodic* vs. *sequential*
  - Episodic: Agent's experience is divided into independent episodes (e.g. classification)

Environment Types

- *Fully observable* vs. *partially observable*
  - If the agent has access to full state of the environment or not
- *Deterministic* vs. *stochastic* vs. *nondeterministic*
  - Deterministic: Next state is completely determined by the agent's actions.
- ☞ *Uncertain*: not fully observable or not deterministic
- *Episodic* vs. *sequential*
  - Episodic: Agent's experience is divided into independent episodes (e.g. classification)
- *Static* vs. *dynamic*
  - Dynamic: Environment may change while agent is deliberating.

Environment Types

- *Fully observable* vs. *partially observable*
  - If the agent has access to full state of the environment or not
- *Deterministic* vs. *stochastic* vs. *nondeterministic*
  - Deterministic: Next state is completely determined by the agent's actions.
- ☞ *Uncertain*: not fully observable or not deterministic
- *Episodic* vs. *sequential*
  - Episodic: Agent's experience is divided into independent episodes (e.g. classification)
- *Static* vs. *dynamic*
  - Dynamic: Environment may change while agent is deliberating.
- *Discrete* vs. *continuous*

Environment Types

- *Fully observable* vs. *partially observable*
  - If the agent has access to full state of the environment or not
- *Deterministic* vs. *stochastic* vs. *nondeterministic*
  - Deterministic: Next state is completely determined by the agent's actions.
- ☞ *Uncertain*: not fully observable or not deterministic
- *Episodic* vs. *sequential*
  - Episodic: Agent's experience is divided into independent episodes (e.g. classification)
- *Static* vs. *dynamic*
  - Dynamic: Environment may change while agent is deliberating.
- *Discrete* vs. *continuous*
- *Single-agent* vs. *multiagent*

Environment types

	Crossword	Backgammon	Internet shopping	Taxi
Observable				
Deterministic				
Episodic				
Static				
Discrete				
Single-agent				

Environment types

	Crossword	Backgammon	Internet shopping	Taxi
Observable	Yes	Yes	No	No
Deterministic				
Episodic				
Static				
Discrete				
Single-agent				

Environment types

	Crossword	Backgammon	Internet shopping	Taxi
Observable	Yes	Yes	No	No
Deterministic	Yes	No	Partly	No
Episodic				
Static				
Discrete				
Single-agent				

Environment types

	Crossword	Backgammon	Internet shopping	Taxi
Observable	Yes	Yes	No	No
Deterministic	Yes	No	Partly	No
Episodic	No	No	No	No
Static				
Discrete				
Single-agent				

Environment types

	Crossword	Backgammon	Internet shopping	Taxi
Observable	Yes	Yes	No	No
Deterministic	Yes	No	Partly	No
Episodic	No	No	No	No
Static	Yes	Yes	Semi	No
Discrete				
Single-agent				

Environment types

	Crossword	Backgammon	Internet shopping	Taxi
Observable	Yes	Yes	No	No
Deterministic	Yes	No	Partly	No
Episodic	No	No	No	No
Static	Yes	Yes	Semi	No
Discrete	Yes	Yes	Yes	No
Single-agent				

Environment types

	Crossword	Backgammon	Internet shopping	Taxi
Observable	Yes	Yes	No	No
Deterministic	Yes	No	Partly	No
Episodic	No	No	No	No
Static	Yes	Yes	Semi	No
Discrete	Yes	Yes	Yes	No
Single-agent	Yes	No	Yes (except auctions)	No

Environment types

	Crossword	Backgammon	Internet shopping	Taxi
Observable	Yes	Yes	No	No
Deterministic	Yes	No	Partly	No
Episodic	No	No	No	No
Static	Yes	Yes	Semi	No
Discrete	Yes	Yes	Yes	No
Single-agent	Yes	No	Yes (except auctions)	No

☞ *The environment type largely determines the agent design*

- The real world is:
  - partially observable, stochastic, sequential, dynamic, continuous, and multi-agent

Agent programs

- The *agent function* is a mathematical entity maps from percept histories to actions:

$$f : \mathcal{P}^* \rightarrow \mathcal{A}$$

- *agent program* runs on a physical architecture. It takes percepts $\mathcal{P}$ as input and returns actions A.
- An agent program implements a given agent function f.

Table-driven agent

Table-driven agent:

percepts = A list, initially empty
table = A table of actions, indexed by a list of percepts
Function Table-Driven-Agent(percept) returns an action
    Append percept to percepts
    action ← Lookup(percepts, table)
    return action

Ask:

- What is the size of table?
- What happens to that size as the agent receives more percepts?

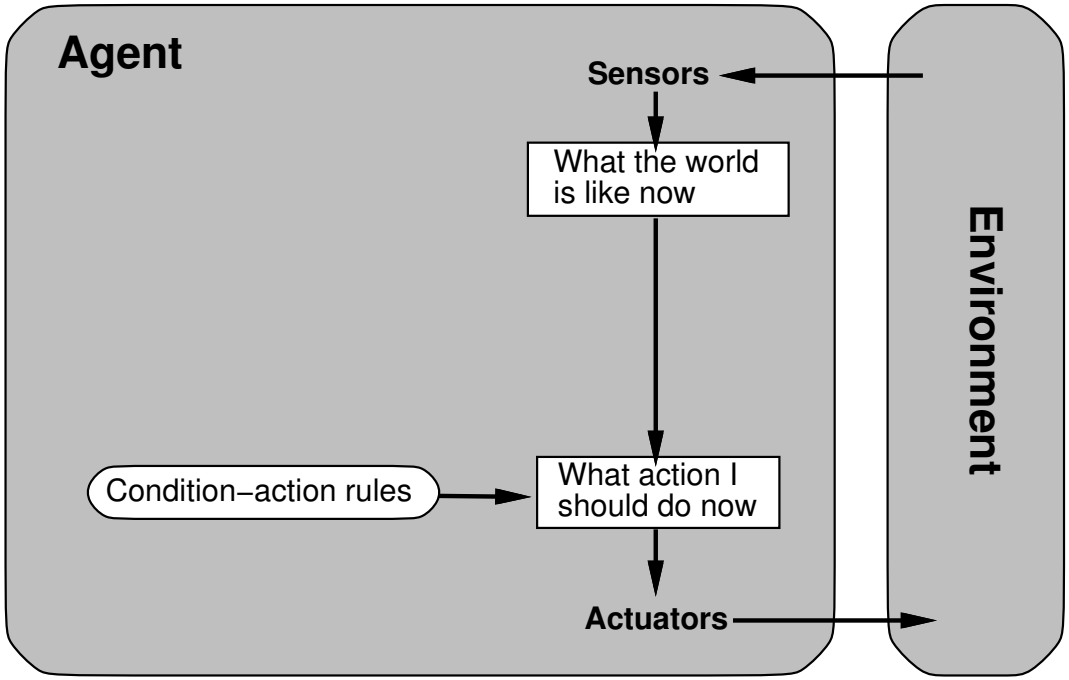
Agent types

There are four basic types in order of increasing generality:

- simple reflex agents
- reflex agents with state
- goal-based agents
- utility-based agents

All these can be turned into learning agents.

Simple reflex agents



- Action is selected according to the current percept
- So, no knowledge of percept history.

A simple reflex agent algorithm

Function Simple-Reflex-Agent(percept) returns an action

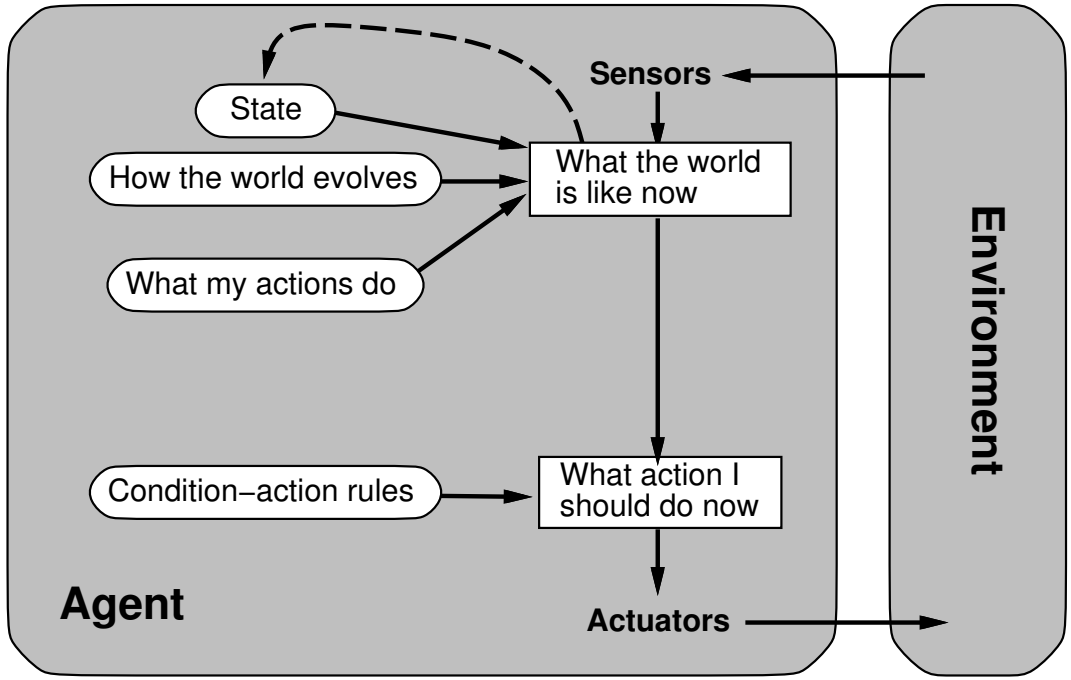
persistent: rules a set of condition-action rules

state $\leftarrow$ Interpret-Input(percept)
rule $\leftarrow$ Rule-Match(state,rules)
action $\leftarrow$ rule.Action
return action

Example

```
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
```

Reflex agents with state



- Also called a "model-based reflex agent"
- Agent keeps track of what it knows about the world.
- Useful for partial observability

A simple reflex agent algorithm

Function Reflex-Agent-With-State(percept) returns an action

persistent: state: the agent's conception of the world state

model: The transition model – how the next state depends on the present state and action

rules: a set of condition-action rules

action: the most recent action (initially none)

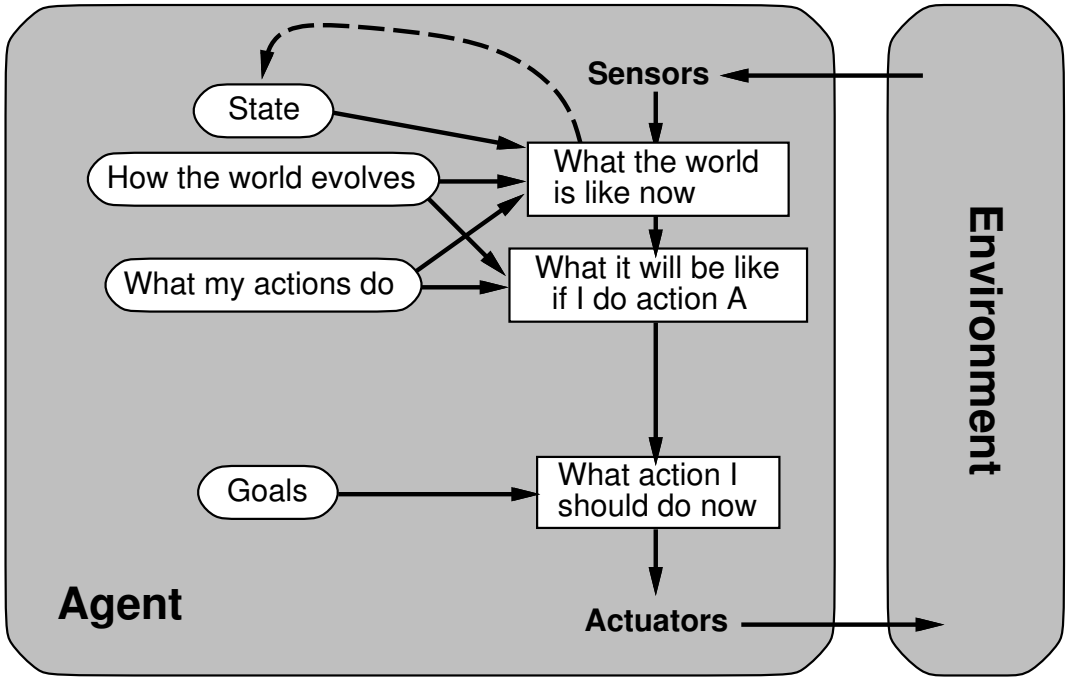
state $\leftarrow$ Update-State(state,action,percept,model)

rule $\leftarrow$ Rule-Match(state,rules)

action $\leftarrow$ rule.Action

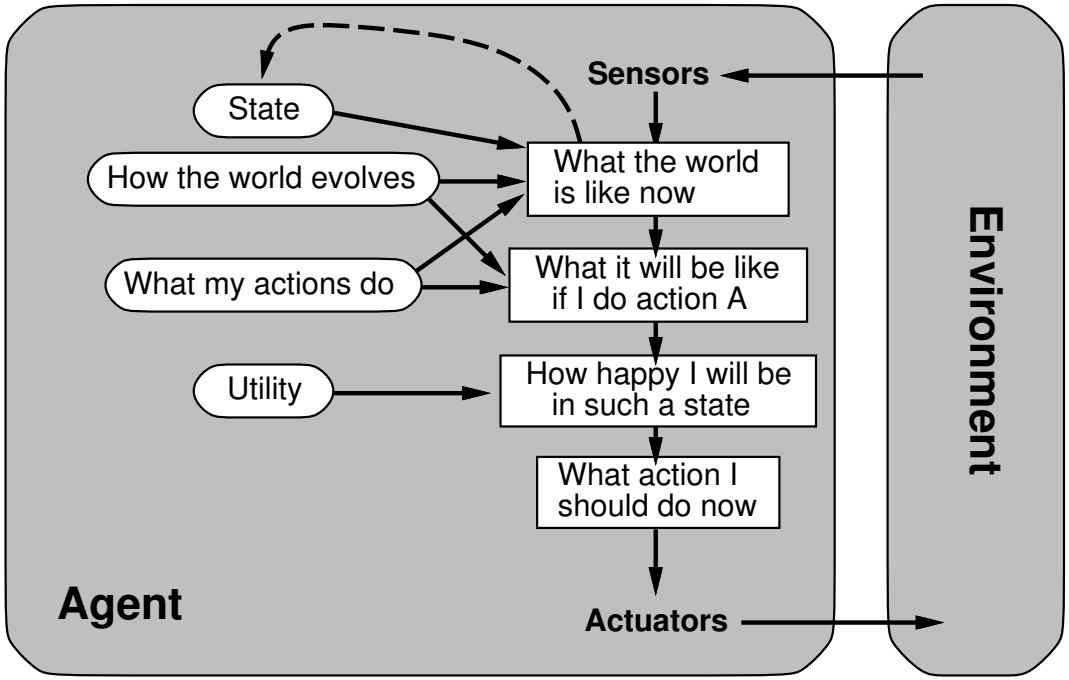
return action

Goal-based agents



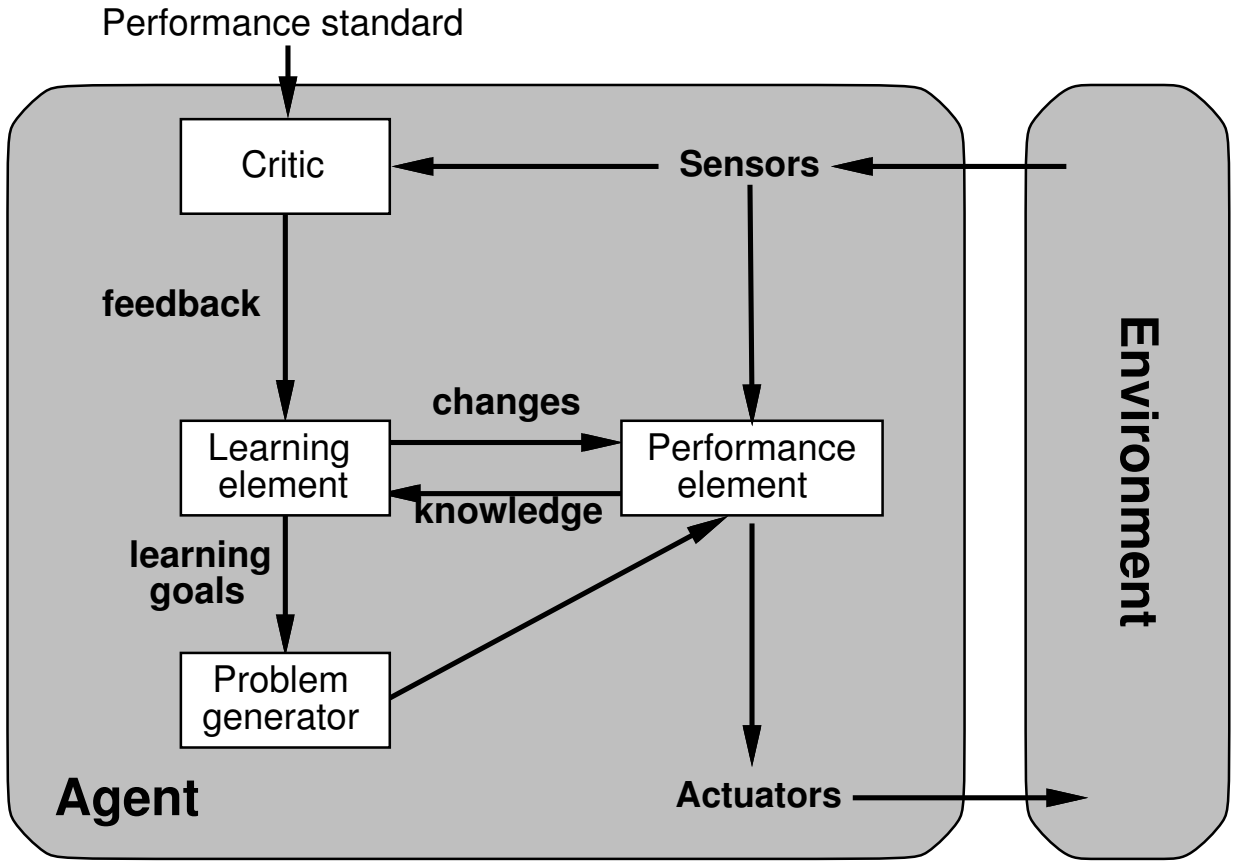
- Agent's actions are determined in part by its *goals*.
- Example: Classical planning.

Utility-based agents



- In addition to goals, use a notion of how "good" an action sequence is.
  - E.g.: Taxi to airport should be safe, efficient, etc.

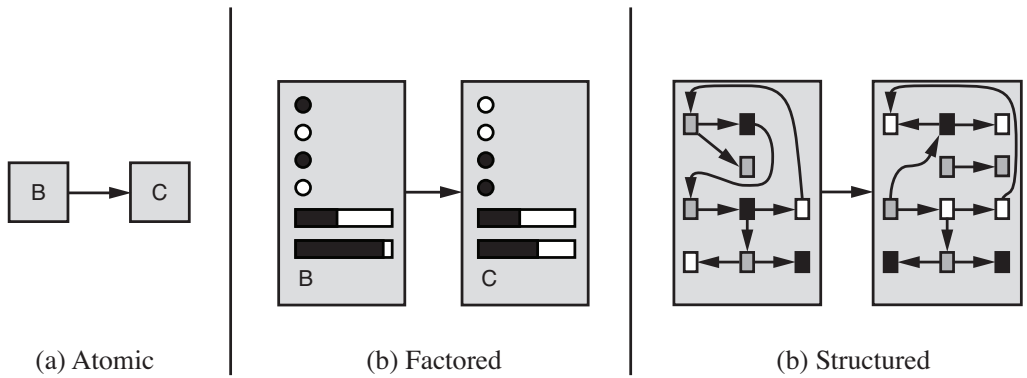
Learning agents



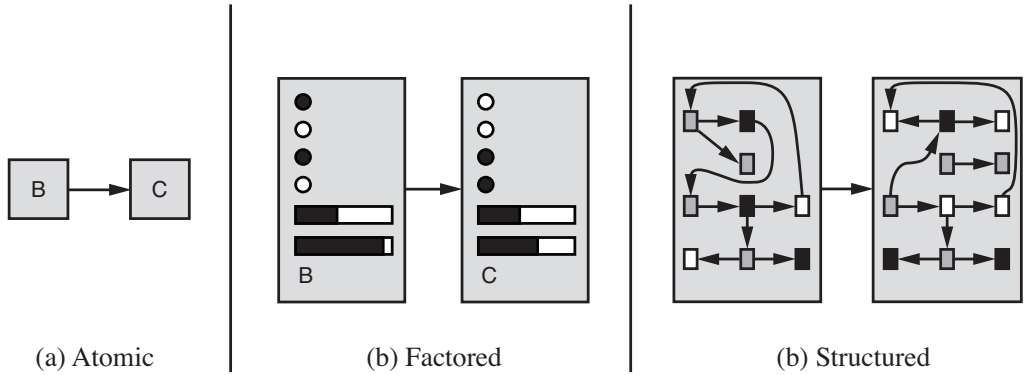
Problem state representation

Three types of representations:

- Atomic: Each state is *indivisible*.
- Factored: A state consists of a *collection* of attributes.
- Structured: States may have arbitrary relationships.



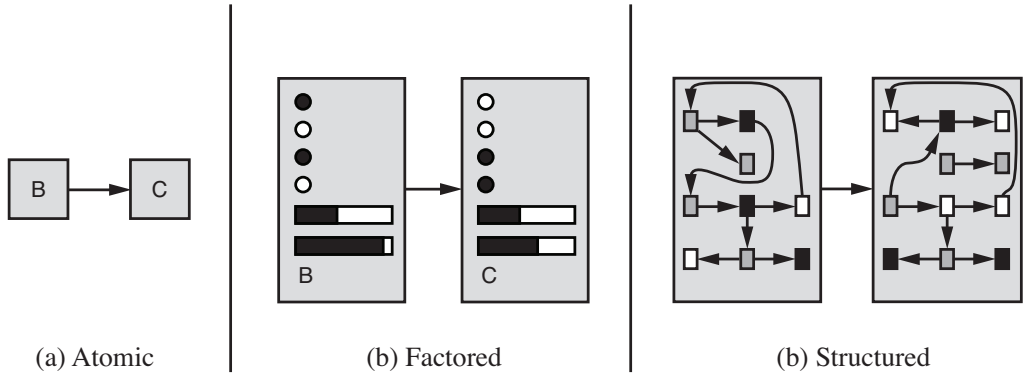
Problem state representation



How large is the representation of the rules of chess using a representation that is:

- Atomic:

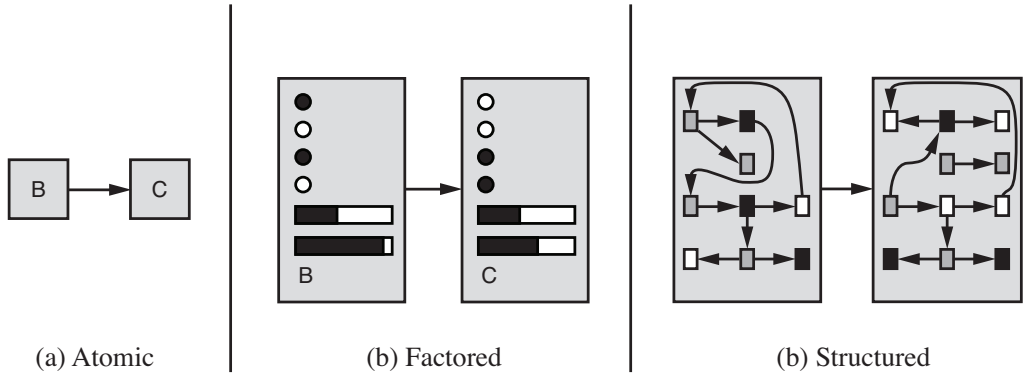
Problem state representation



How large is the representation of the rules of chess using a representation that is:

- Atomic: $(64^6)^2$.

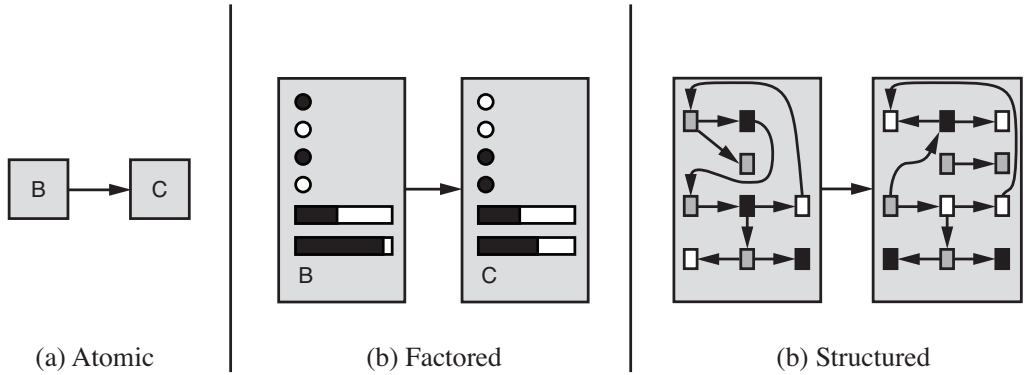
Problem state representation



How large is the representation of the rules of chess using a representation that is:

- Atomic: $(64^6)^2$.
- Factored:

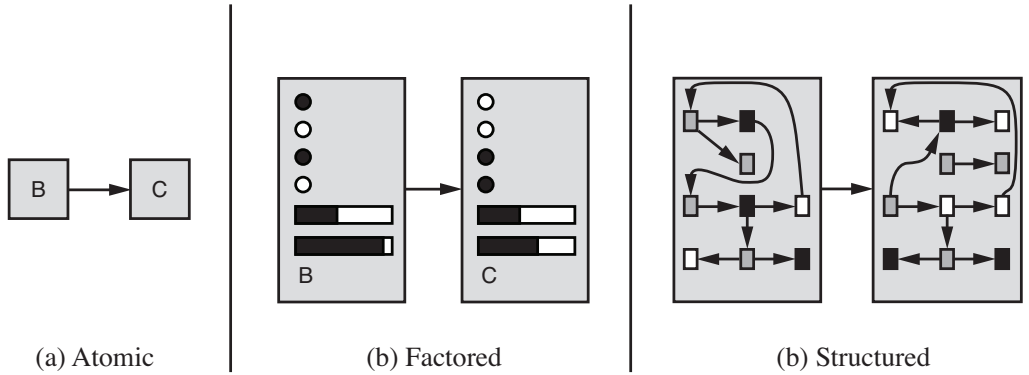
Problem state representation



How large is the representation of the rules of chess using a representation that is:

- Atomic: $(64^6)^2$.
- Factored: $64^2 \cdot 6^2$

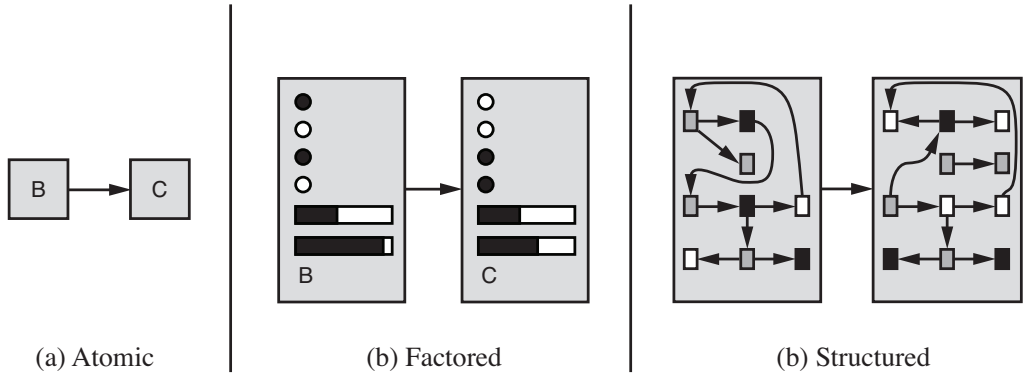
Problem state representation



How large is the representation of the rules of chess using a representation that is:

- Atomic: $(64^6)^2$.
- Factored: $64^2 \cdot 6^2$
- Structured:

Problem state representation



How large is the representation of the rules of chess using a representation that is:

- Atomic: $(64^6)^2$.
- Factored: $64^2 \cdot 6^2$
- Structured: 6

Summary

- *Agents* interact with *environments* through *actuators* and *sensors*
- The *agent function* describes what the agent does in all circumstances
- The *performance measure* evaluates the environment sequence
- A *rational* agent maximizes expected performance
- *Agent programs* implement agent functions
- *PEAS* descriptions define task environments
- Environments are categorized along several dimensions:
  observable? *deterministic*? *episodic*? *static*? *discrete*? *single-agent*?
- Several basic agent architectures exist:
  reflex, *reflex with state*, *goal-based*, *utility-based*