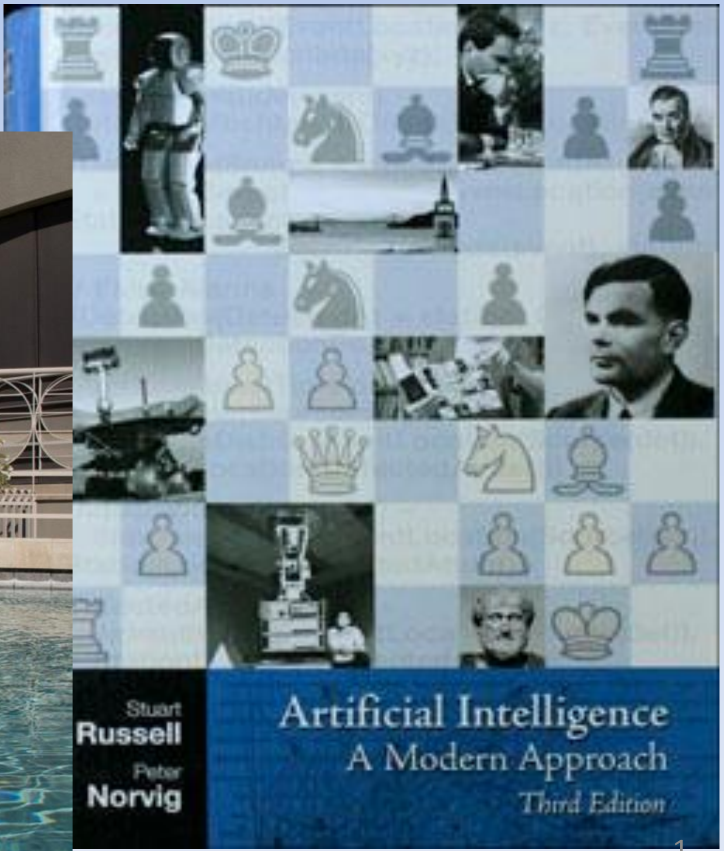
Artificial Intelligence

Chapter 2: Intelligent Agents



Chapter 2a

- Intelligent Agents
- PEAS
- Environment Types

What is AI ??

.. Rational Agents ..

- Take a closer..
  - Agents and Rationality
  - Agent Environments

Intelligence Involves ?

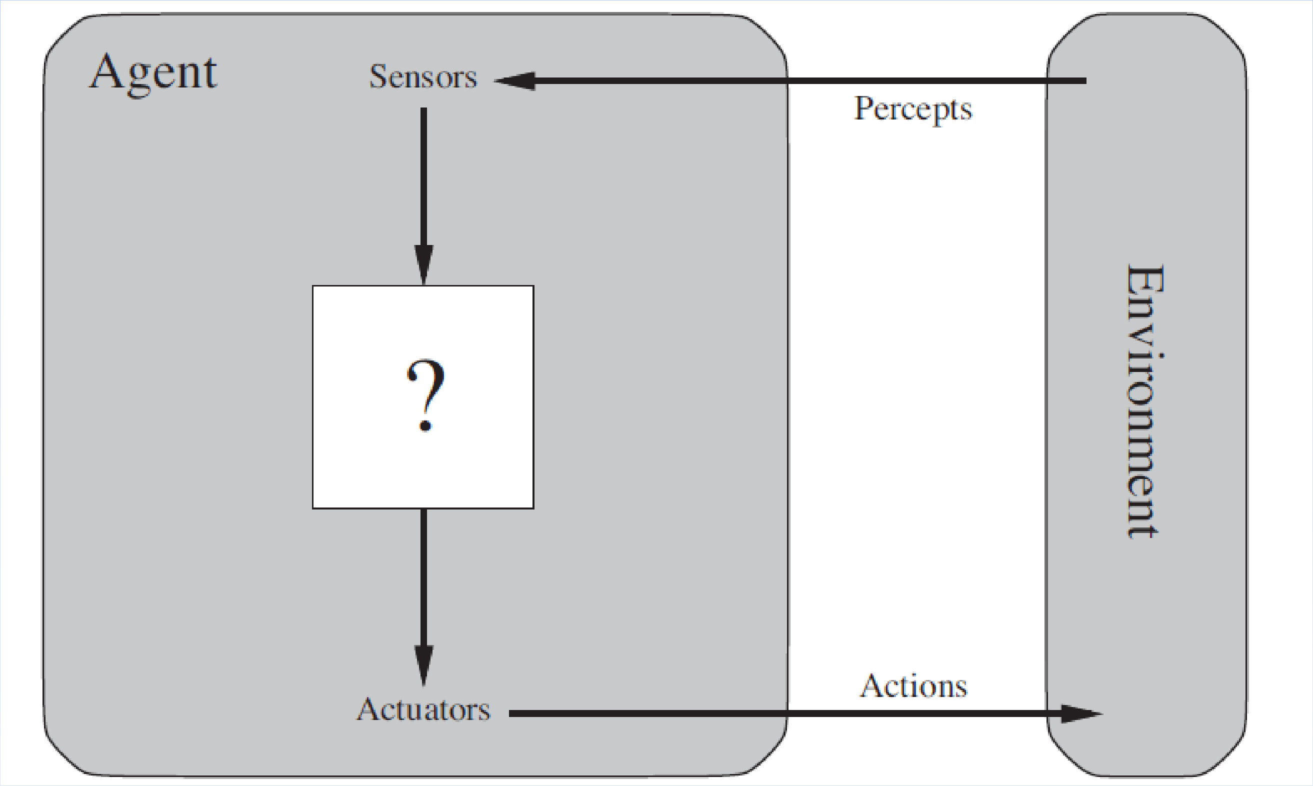
- Interacting with Real World
  - Perceive / Understand / Act
  - Speech Recognition & Understanding & Synthesis
  - Image Understanding
  - Taking Action!! Having an Effect!!!

Intelligence Involves ?

- Knowledge Representation, Reasoning & Planning
  - Percepts drive Modeling World
  - Solving Problems / Planning & Making Decisions
  - Dealing with Unexpected Problems (Uncertainties)
- Learning and Adapting
  - Continuous Improvement
  - Continual Learning & Adapting
  - Constantly Updating/Improving Internal Representations.

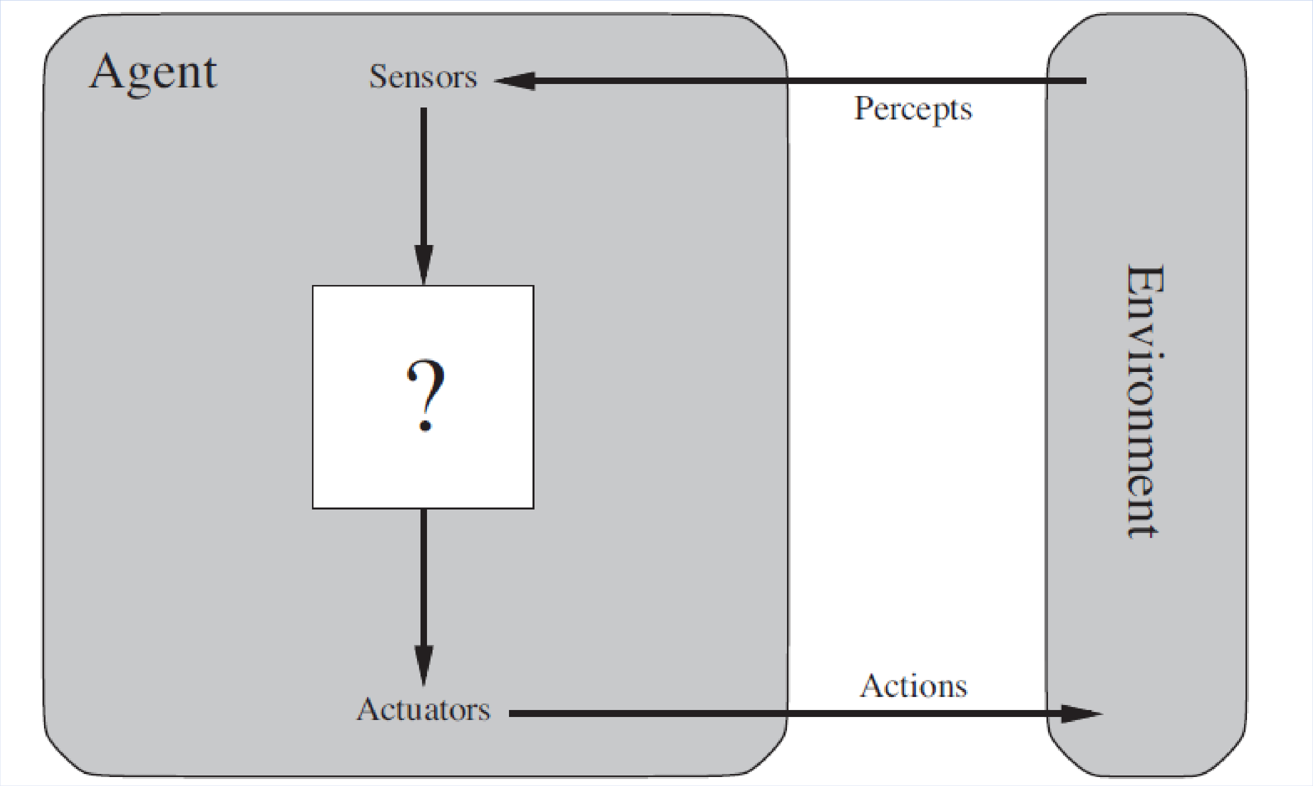
What's an Agent

- **Perceives** environment through *Sensors*
- **Acts** out into environment through Actuators



What's an Agent

- **Percept** refers to the agents sensor data
- **Percept Sequence** is complete percept history



Pool Skimmer

- **Sensors:** When Stuck, Battery Low, etc…
- **Actuators:** Skim debris, Change Direction, Shutdown system, Restart System.

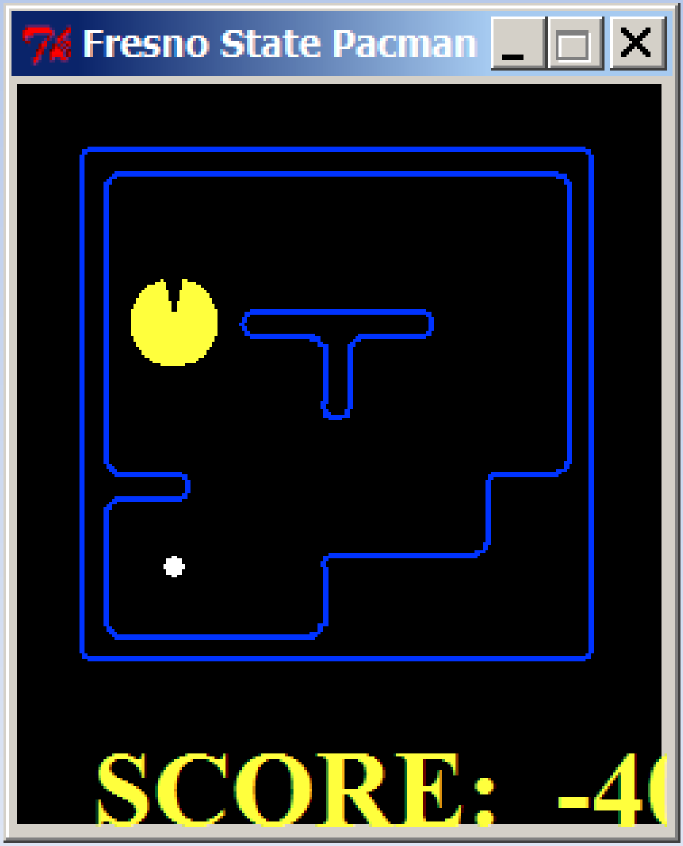
Android – Datta

- **Sensors:** Eyes, Ears, Nose, Etc….
- **Actuators:** Arms, Legs, etc…

Pacman Agent

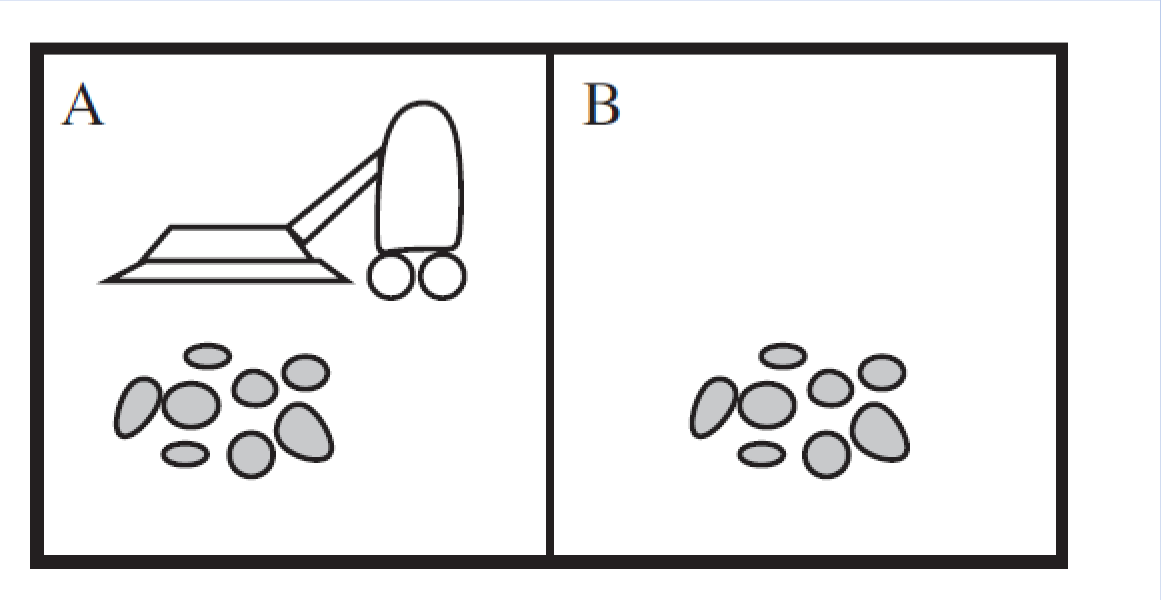
- **Sensors:** Current Location, Capsule Locations, Ghost Locations, etc…

- **Actuators:** Direction



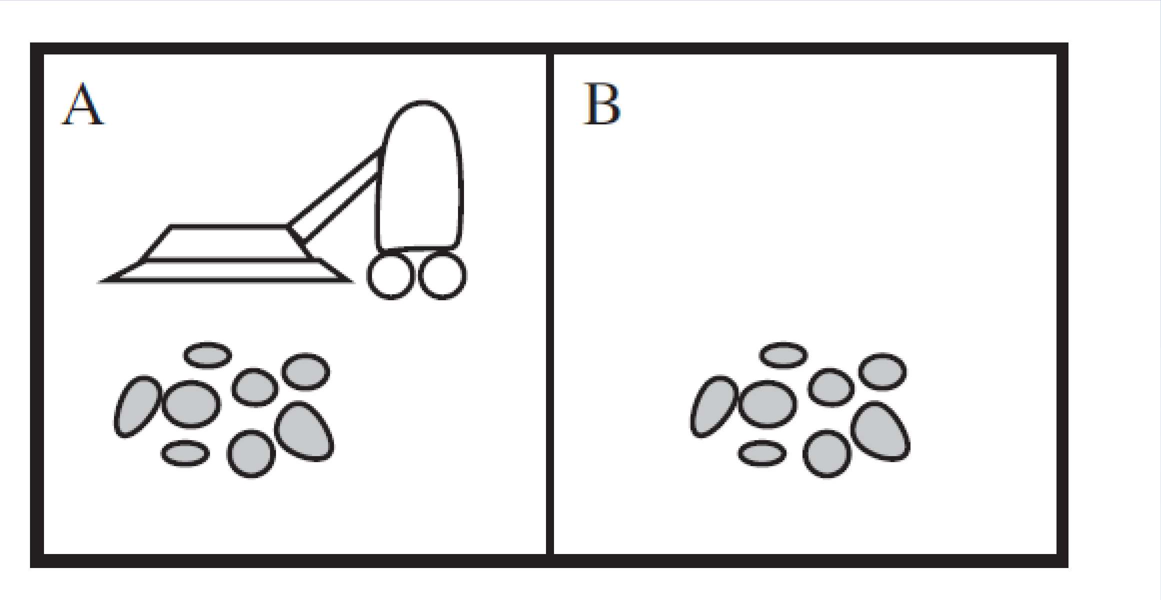
Simple Example World: Vacuum Cleaner World

- World consists of two locations:
  - Square A
  - Square B



Simple Example World: Vacuum Cleaner World

- **Sensors:** Current Location, Dirt
- **Actuators:** Right, Left, Suck



Rational Agent

- Just do the right thing!

- How do we recognize the right thing??

Performance Metric

Performance Metric

- Vary from Environment to Environment

- Need care in construction!

Rationality via Performance Measure

- Rationality depends on:
  - Performance Measure
  - Agent’s prior knowledge
  - Agent’s available actions
  - Agent’s percepts

Rational Agent is

- Rational Agents act to maximize performance measure given:
  - Agent's Percepts
  - Agent's prior knowledge

Rationality: Performance Measure & Vacuum World

- Fixed Performance measure evaluates the environment sequence
  - 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
- Rational Agent chooses action maximizing the expected value of Performance Measure given Percept Sequence to Date.
- Rational ≠ Omniscient
- Rational ≠ Clairvoyant
- Rational ≠ Successful
- Rational → Exploration, Learning Autonomy

Task Evironment

- **PEAS**:
  - **P**erformance measure
  - **E**nvironment
  - **A**ctuators
  - **S**ensors

Taxi Driver

Agent Type	Performance Measure	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

Figure 2.4 PEAS description of the task environment for an automated taxi.

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Display of questions, tests, diagnoses, treatments, referrals	Keyboard entry of symptoms, findings, patient's answers
Satellite image analysis system	Correct image categorization	Downlink from orbiting satellite	Display of scene categorization	Color pixel arrays
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, joint angle sensors
Satellite image analysis system	Correct image categorization	Downlink from orbiting satellite	Display of scene categorization	Color pixel arrays
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, joint angle sensors
Refinery controller	Purity, yield, safety	Refinery, operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical sensors
Interactive English tutor	Student's score on test	Set of students, testing agency	Display of exercises, suggestions, corrections	Keyboard entry

Figure 2.5 Examples of agent types and their PEAS descriptions.

Environment’s Nature

Task Environment
Crossword puzzle
Chess with a clock
Poker
Backgammon
Taxi driving
Medical diagnosis
Image analysis
Part-picking robot
Refinery controller
Interactive English tutor

Figure 2.6 Examples of task environments and their characteristics.

Fully Observable vs. Partially Observable

- If 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.
  - Effectively fully observable if sensors detect all relevant aspects to choice of action.
- Noisy or Inaccurate Sensors can create PARTIALLY OBSERVABLE environment.
- Sensor inability to detect relevant aspects of environment.
  - Vacuum Agent with only local dirt sensor
- When there are no sensor data, the environment is UNOBSERVABLE.

Observable

Task Environment	Observable
Crossword puzzle	Fully
Chess with a clock	Fully
Poker	Partially
Backgammon	Fully
Taxi driving	Partially
Medical diagnosis	Partially
Image analysis	Fully
Part-picking robot	Partially
Refinery controller	Partially
Interactive English tutor	Partially

Figure 2.6 Examples of task environments and their characteristics.

Single Agent vs. Multiagent

- Question is really when an entity should be viewed as agent.
- Key distinction is whether the entity’s behavior is best described as maximizing a performance measure.
- Also: Competitive vs. Cooperative

Agents

Task Environment	Observable	Agents
Crossword puzzle	Fully	Single
Chess with a clock	Fully	Multi
Poker	Partially	Multi
Backgammon	Fully	Multi
Taxi driving	Partially	Multi
Medical diagnosis	Partially	Single
Image analysis	Fully	Single
Part-picking robot	Partially	Single
Refinery controller	Partially	Single
Interactive English tutor	Partially	Multi

Figure 2.6 Examples of task environments and their characteristics.

Deterministic vs. Stochastic

- Environment is DETERMINISTIC if the next state is completely determined by the current state and the action execute by the agent.
- Uncertain environment if not fully observable or not deterministic.

Deterministic

Task Environment	Observable	Agents	Deterministic
Crossword puzzle	Fully	Single	Deterministic
Chess with a clock	Fully	Multi	Deterministic
Poker	Partially	Multi	Stochastic
Backgammon	Fully	Multi	Stochastic
Taxi driving	Partially	Multi	Stochastic
Medical diagnosis	Partially	Single	Stochastic
Image analysis	Fully	Single	Deterministic
Part-picking robot	Partially	Single	Stochastic
Refinery controller	Partially	Single	Stochastic
Interactive English tutor	Partially	Multi	Stochastic

Figure 2.6 Examples of task environments and their characteristics.

Episodic vs. Sequential

- Episodic environments are broken down into separate episodes.
- Each Episode is an independent task.
- Each Episode does not depend on previous episodes.
- Classification Tasks are often episodic.
- Sequential environments decisions can have long lasting effects.

Episodic

Task Environment	Observable	Agents	Deterministic	Episodic
Crossword puzzle	Fully	Single	Deterministic	Sequential
Chess with a clock	Fully	Multi	Deterministic	Sequential
Poker	Partially	Multi	Stochastic	Sequential
Backgammon	Fully	Multi	Stochastic	Sequential
Taxi driving	Partially	Multi	Stochastic	Sequential
Medical diagnosis	Partially	Single	Stochastic	Sequential
Image analysis	Fully	Single	Deterministic	Episodic
Part-picking robot	Partially	Single	Stochastic	Episodic
Refinery controller	Partially	Single	Stochastic	Sequential
Interactive English tutor	Partially	Multi	Stochastic	Sequential

Figure 2.6 Examples of task environments and their characteristics.

Static vs. Dynamic

Static

Task Environment	Observable	Agents	Deterministic	Episodic	Static
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static
Chess with a clock	Fully	Multi	Deterministic	Sequential	Semi
Poker	Partially	Multi	Stochastic	Sequential	Static
Backgammon	Fully	Multi	Stochastic	Sequential	Static
Taxi driving	Partially	Multi	Stochastic	Sequential	Dynamic
Medical diagnosis	Partially	Single	Stochastic	Sequential	Dynamic
Image analysis	Fully	Single	Deterministic	Episodic	Semi
Part-picking robot	Partially	Single	Stochastic	Episodic	Dynamic
Refinery controller	Partially	Single	Stochastic	Sequential	Dynamic
Interactive English tutor	Partially	Multi	Stochastic	Sequential	Dynamic

Figure 2.6 Examples of task environments and their characteristics.

Discrete vs. Continuous

Discrete

Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with a clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Poker	Partially	Multi	Stochastic	Sequential	Static	Discrete
Backgammon	Fully	Multi	Stochastic	Sequential	Static	Discrete
Taxi driving	Partially	Multi	Stochastic	Sequential	Dynamic	Continuous
Medical diagnosis	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Image analysis	Fully	Single	Deterministic	Episodic	Semi	Continuous
Part-picking robot	Partially	Single	Stochastic	Episodic	Dynamic	Continuous
Refinery controller	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Interactive English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

Figure 2.6 Examples of task environments and their characteristics.

Known vs. Unknown

Environment’s Nature

- Fully Observable versus Partially Observable
- Deterministic versus Stochastic
- Episodic versus Sequential
- Static versus Dynamic
- Discrete versus Continuous
- Single Agent versus Multi-Agent

- Connect Four????
- Tic-Tac-Toe????

Peter Norvig's Notebooks

- Norvig.Com

Peter@Norvig.com

This site contains technical papers, essays, reports, software, and other materials by Peter Norvig. RSS

NEW

NEW List of Jupyter/Ipython notebooks
Lego Institute for Lego Investigation
English Letter Frequency Counts: Mayzner Revisited
Google's Hybrid Approach to Research, article published at ACM.

Top Dozen Links on Norvig.com

#1 Gettysburg Powerpoint Presentation and its making (slides)
#2 AI: A Modern Approach (book) and AI on the Web (links)
#3 World's Longest Palindrome (for 20:02 02/20 2002)
#4 Teach Yourself Programming in 10 Years (essay)
#5 Paradigms of AI Programming (book) with code
#6 Java IAQ and Python IAQ (FAQs)
#7 Design Patterns in Dynamic Languages (slides)
#8 Lisp compared to Python, Java, and itself in 1991
#9 Code for Intro AI programming in Python and Lisp
#10 Einstein '05 Performance Review
#11 JScheme: Scheme in Java (software)
#12 Doing the Martin Shuffle (with your iPod)

Artificial Intelligence Books

#2 **AI: A Modern Approach**, *Outstanding ... will deservedly dominate the field for some time* - Nils Nilsson Amazon
#5 Paradigms of AI Programming *Possibly the best hardcore programming book ever.* - Gareth McCaughan Amazon
Verbmobil: Translation for Face-to-Face Dialog - Amazon
Intelligent Help Systems for Unix - Amazon

Free Open Source Software

#5 Lisp for *Paradigms of AI Programming*

Contact Information

Peter Norvig
Director of Research Google
Email: pnorvig@google.com

Vita / resume including online papers; short bio with photos
Me elsewhere on the web; photos I've taken

Java, Lisp and Python Essays

#5 Paradigms of AI Programming with Lisp code
#6 Java IAQ (Infrequently Answered Questions)
#6 Python IAQ (Infrequently Answered Questions)
#8 Python for Lisp Programmers (essay)
#11 JScheme: Scheme implemented in Java (free software)
(How to Write a (Lisp) Interpreter (in Python))
Lisp: Where Do We Come From? What Are We? Where Are We Going?
Silk: A Playful Blend of Scheme and Java (ps)
Lisp as an Alternative to Java (comparison)
Lisp Retrospective (essay)
Tutorial on Good Lisp Programming Style (ps)
Python Accumulation Displays (proposal)
Solving Every Sudoku Puzzle (essay; python)
How to Write a Spelling Corrector (essay; python)

Other Programming Papers and Presentations

On Chomsky and the Two Cultures of Statistical Learning
#4 Teach Yourself Programming in 10 Years (essay)
#7 Design Patterns in Dynamic Languages (slides)

List of IPython (Jupyter) Notebooks by Peter Norvig

Here are some notebooks I have made. You can click on a notebook title to view it in the browser, or on '(download)' to get a copy that you can run and modify on your computer (assuming you have Jupyter installed).

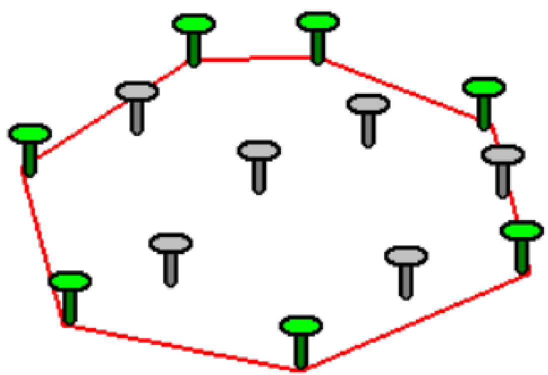
Logic and Number Puzzles

- **Advent of Code 2016** (download)
  Puzzle site with a coding puzze each day for Advent 2016
- **Translating English Sentences into Propositional Logic Statements** (download)
  Automatically converting informal English sentences into formal Propositional Logic.
- **The Puzzle of the Misanthropic Neighbors** (download)
  How crowded will this neighborhood be, if nobody wants to live next door to anyone else?
- **Countdown to 2016** (download)
  Solving the equation 10 _ 9 _ 8 _ 7 _ 6 _ 5 _ 4 _ 3 _ 2 _ 1 = 2016. From an Alex Bellos puzzle.
- **Sicherman Dice** (download)
  Find a pair of dice that is like a regular pair of dice, only different.
- **Beal's Conjecture Revisited^2** (download)
  A search for counterexamples to Beal's Conjecture
- **When is Cheryl's Birthday?** (download)
  Solving the "Cheryl's Birthday" logic puzzle.
- **When Cheryl Met Eve: A Birthday Story** (download)
  Inventing new puzzles in the Style of "Cheryl's Birthday."
- **Sol Golomb's Rectangle Puzzle** (download)
  A Puzzle involving placing rectangles of different sizes inside a square. Bonus: cryptarithmetic.
- **WWW: Will Warriors Win?** (download)
  Golden State Warriors probability of winning the 2016 NBA title.

jupyter
nbviewer

The Convex Hull Problem

Pound a bunch of nails into a board, then stretch a rubber band around them and let the rubber band snap taut, like this:

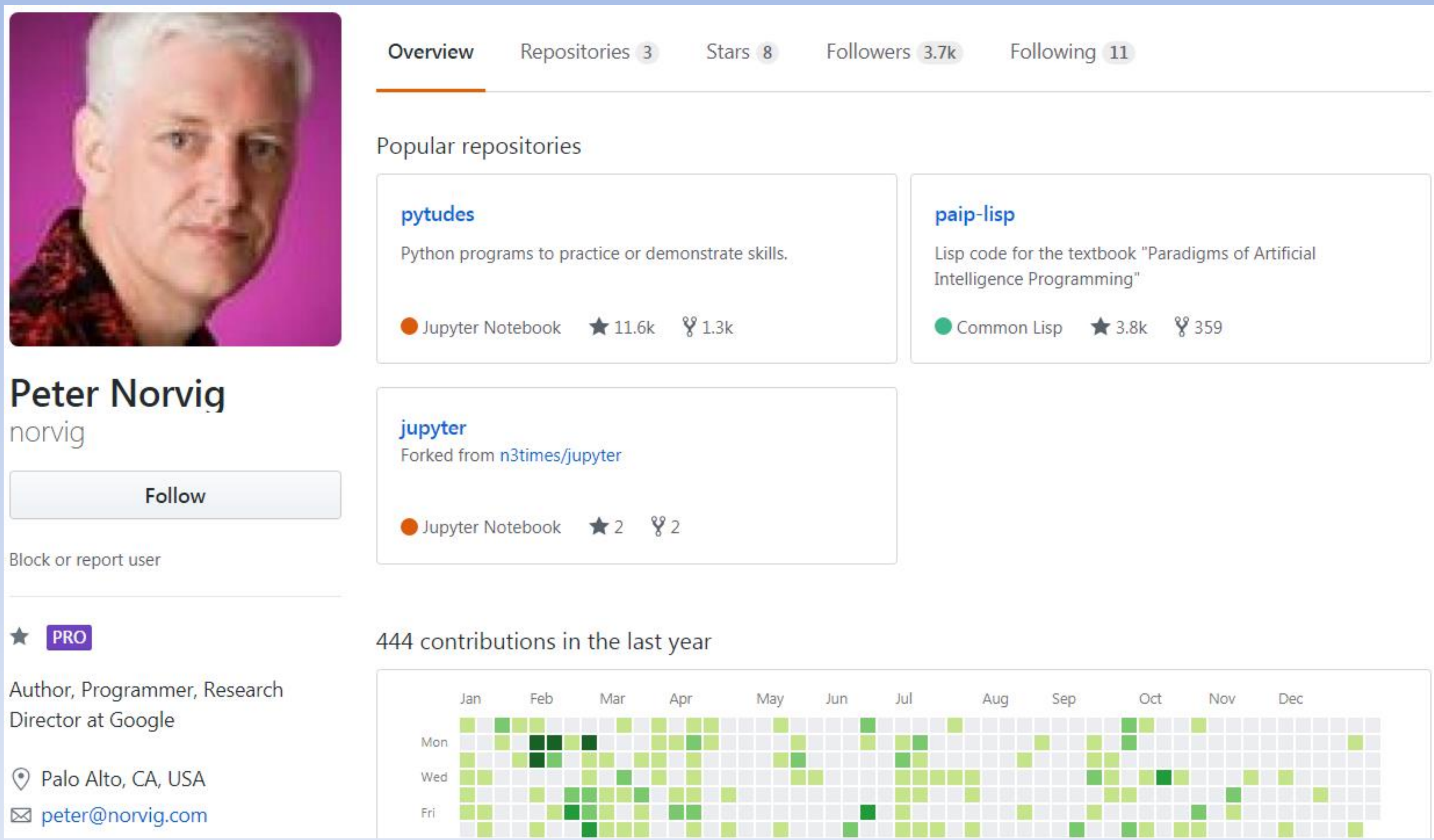


The rubber band has traced out the *convex hull* of the set of nails. It turns out this is an important problem with applications in computer graphics, robot motion planning, geographical information systems, ethology, and other areas. More formally, we say that:

*Given a finite set, **P**, of points in a plane, the convex hull of **P** is a polygon, **H**, such that:*

- *Every point in **P** lies either on or inside of **H**.*
- *Every vertex of **H** is a point in **P**.*
- ***H** is convex: a line segment joining any two vertexes of **H** either is an edge of **H** or lies inside **H**.*

https://github.com/norvig

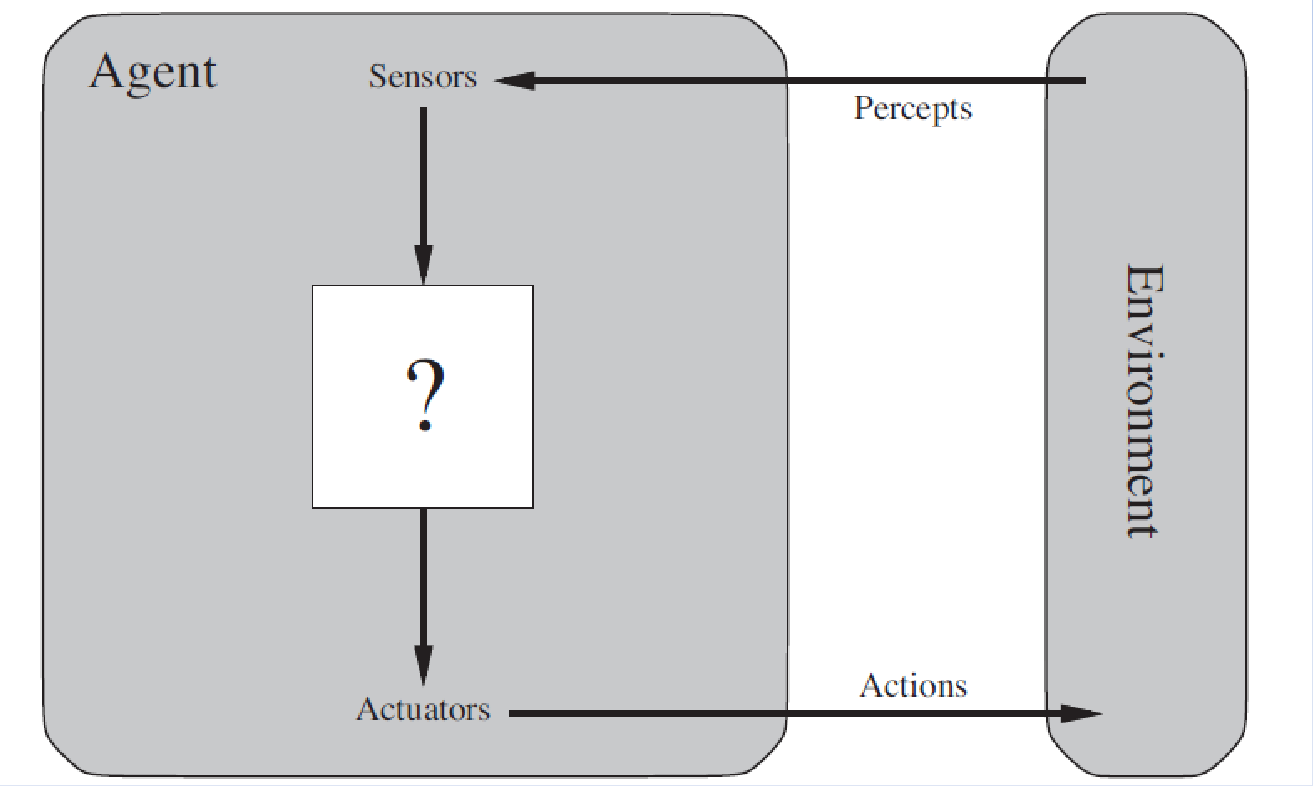


Lecture 2: Chapter 2b

- Agent Types
- Reflex Agent
- Model-Based Reflex Agent
- Goal-Based Agent
- Utility Based Agent
- Learning Agent
- Representing World

What's an Agent

- **Perceives** environment through *Sensors*
- **Acts** out into environment through Actuators



How to Build an Agent
Agent Structure

- Agent consists of:
  - Architecture: Computing device with physical sensors and actuators
  - Program: Implements the Agent Function that maps percepts to actions.

- Agent = Architecture + Program

How to Build an Agent: Agent Program

- INPUT: percept
  - Agent program takes as input the current percept

- OUTPUT: action
  - Based on the current percept as well as the history of previous percepts, the Agent Program returns an action to the actuators.

How to Build an Agent: Agent Function

- INPUT: percept sequence
  - Agent Function takes as input a percept sequence

- OUTPUT: action
  - Agent Function return action based on the percept sequence.

Table Driven Agent
Simplest

- Actions are indexed based on current percept sequence.

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

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

Figure 2.7 The TABLE-DRIVEN-AGENT program is invoked for each new percept and returns an action each time. It retains the complete percept sequence in memory.

Vacuum Cleaner World:
Table Driven Agent (agents.py)

- **Sensors:** Current Location, Dirt
- **Actuators:** Right, Left, Suck

```
TABLE = {((loc_A, 'Clean'),): 'Right',
         ((loc_A, 'Dirty'),): 'Suck',
         ((loc_B, 'Clean'),): 'Left',
         ((loc_B, 'Dirty'),): 'Suck',
         ((loc_A, 'Clean'), (loc_A, 'Clean')): 'Right',
         ((loc_A, 'Clean'), (loc_A, 'Dirty')): 'Suck',
         # ...
         ((loc_A, 'Clean'), (loc_A, 'Clean'), (loc_A, 'Clean')): 'Right',
         ((loc_A, 'Clean'), (loc_A, 'Clean'), (loc_A, 'Dirty')): 'Suck',
         # ...
         }
```

```
def Table_Driven_Agent(percept):
''' returns an action '''
    global PERCEPTS #Initially empty
    global TABLE    #Actions indexed by
                    #  percept sequences
    PERCEPTS.append(percept)
    action = Lookup(PERCEPTS, TABLE)
    return action
```

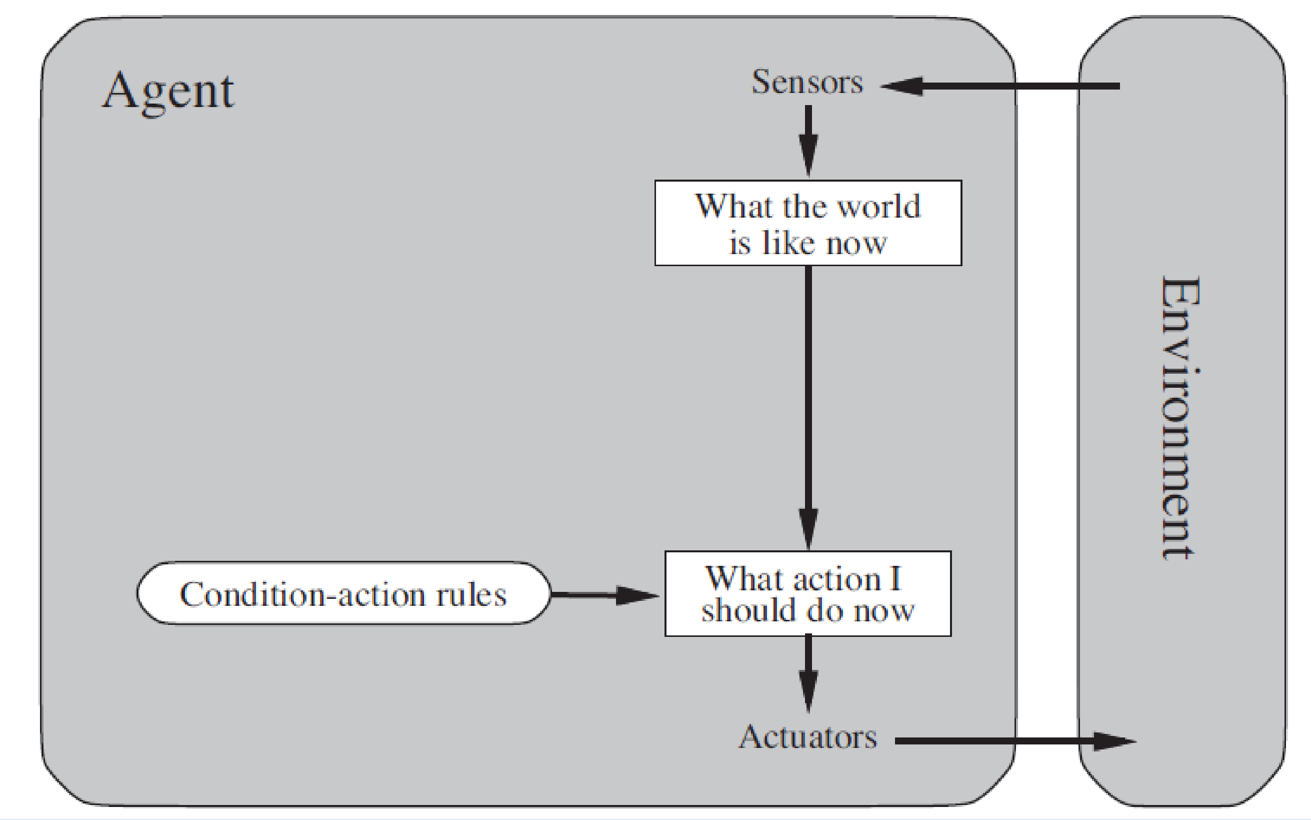
Table Driven Agent

- Actions are indexed based on current percept sequence.
- Table Driven Agent is Dooomed…?

```
def Table_Driven_Agent(percept):
''' returns an action '''
    global PERCEPTS #Initially empty
    global TABLE    #Actions indexed by
                    #  percept sequences
    PERCEPTS.append(percept)
    action = Lookup(PERCEPTS, TABLE)
    return action
```

Simple Reflex Agent

- Acts according to a rule whose condition matches the current state, as defined by the percept.



Simple Reflex Agent

- Acts according to a rule whose condition matches the current state, as defined by the percept.

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*

Figure 2.8 The agent program for a simple reflex agent in the two-state vacuum environment. This program implements the agent function tabulated in Figure 2.3.

Simple Reflex Agent

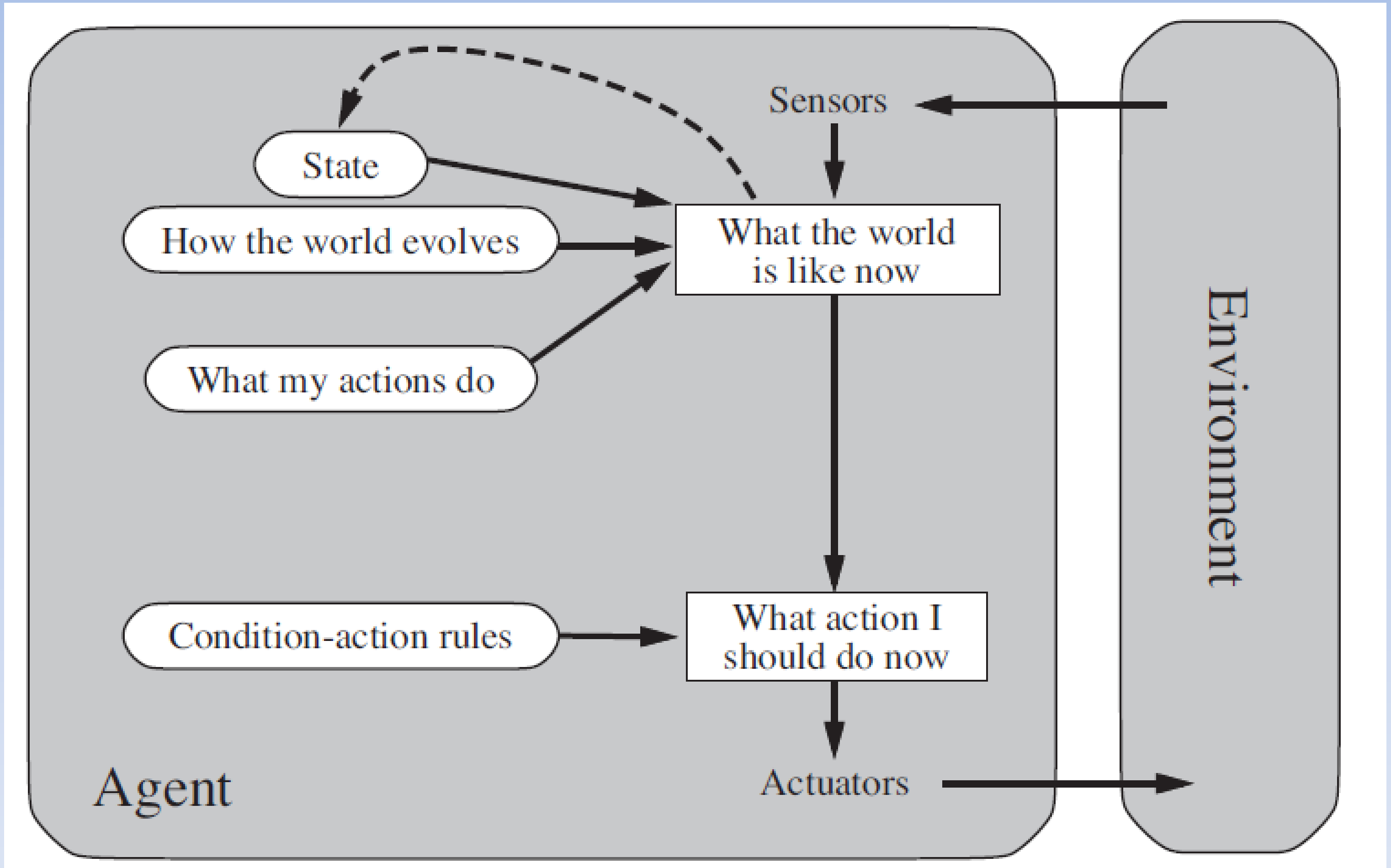
- Acts according to a rule whose condition matches the current state, as defined by the percept.
- Vacuum Cleaner World Example: Agents.py
- Percept = (location, status)
  - Locations = {loc_A, loc_B}
    - loc_A = (0,0)
    - loc_B = (1,0)
  - Status = {‘Clean’, ‘Dirty’}

```
"A reflex agent for the two-state vacuum environment. [Fig. 2.8]"
def program((location, status)):
    if status == 'Dirty': return 'Suck'
    elif location == loc_A: return 'Right'
    elif location == loc_B: return 'Left'
```

Model-based Agent

- Partial Observability Issues
- Agent Needs to keep track of the World it can't see.
- Agent maintains an Internal State
- Internal State is updated due to Agent Percepts
- Internal State Updates need 'Model' of how world works.

Model-based Agent



Model-based Agent

```
def ModelBasedReflexAgentProgram(rules, update_state):
    "This agent takes action based on the percept and state.
        [Fig. 2.12]"
    def program(percept):
        program.state = update_state(
            program.state, program.action, percept)
        rule = rule_match(program.state, rules)
        action = rule.action
        return action
    program.state = program.action = None
    return program
```

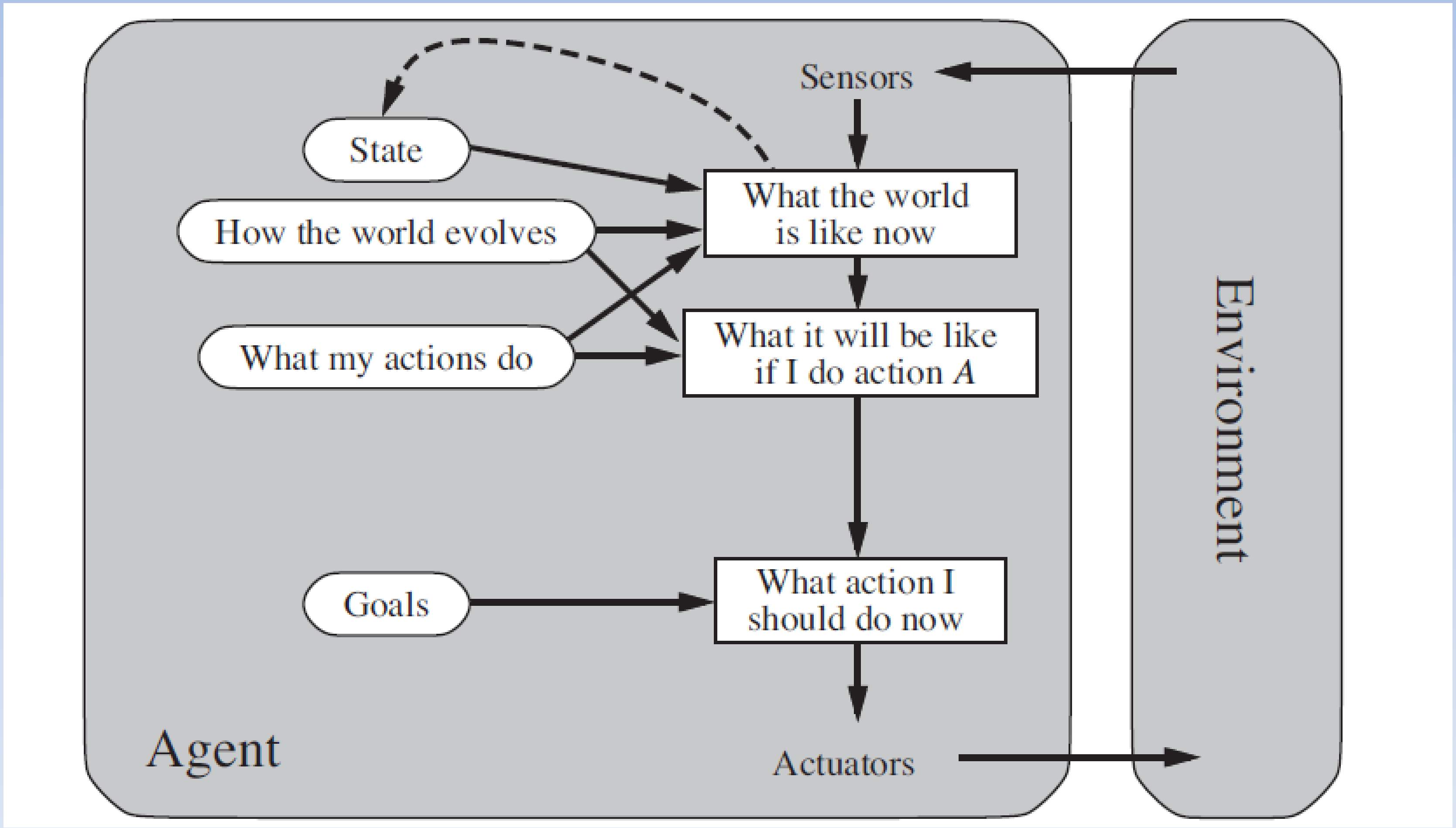
Model-based Agent: Vacuum World

```
def ModelBasedVacuumAgent():
    "An agent that keeps track of what locations are clean or dirty."
    model = {loc_A: None, loc_B: None}
    def program((location, status)):
        "Same as ReflexVacuumAgent,
              except if everything is clean, do NoOp."
        model[location] = status ## Update the model here
        if model[loc_A] == model[loc_B] == 'Clean': return 'NoOp'
        elif status == 'Dirty': return 'Suck'
        elif location == loc_A: return 'Right'
        elif location == loc_B: return 'Left'
    return Agent(program)
```

Goal-based Agent

- Keeps track of state
- ADDS: Keeps track of GOALS
- Chooses actions that will lead to goals.

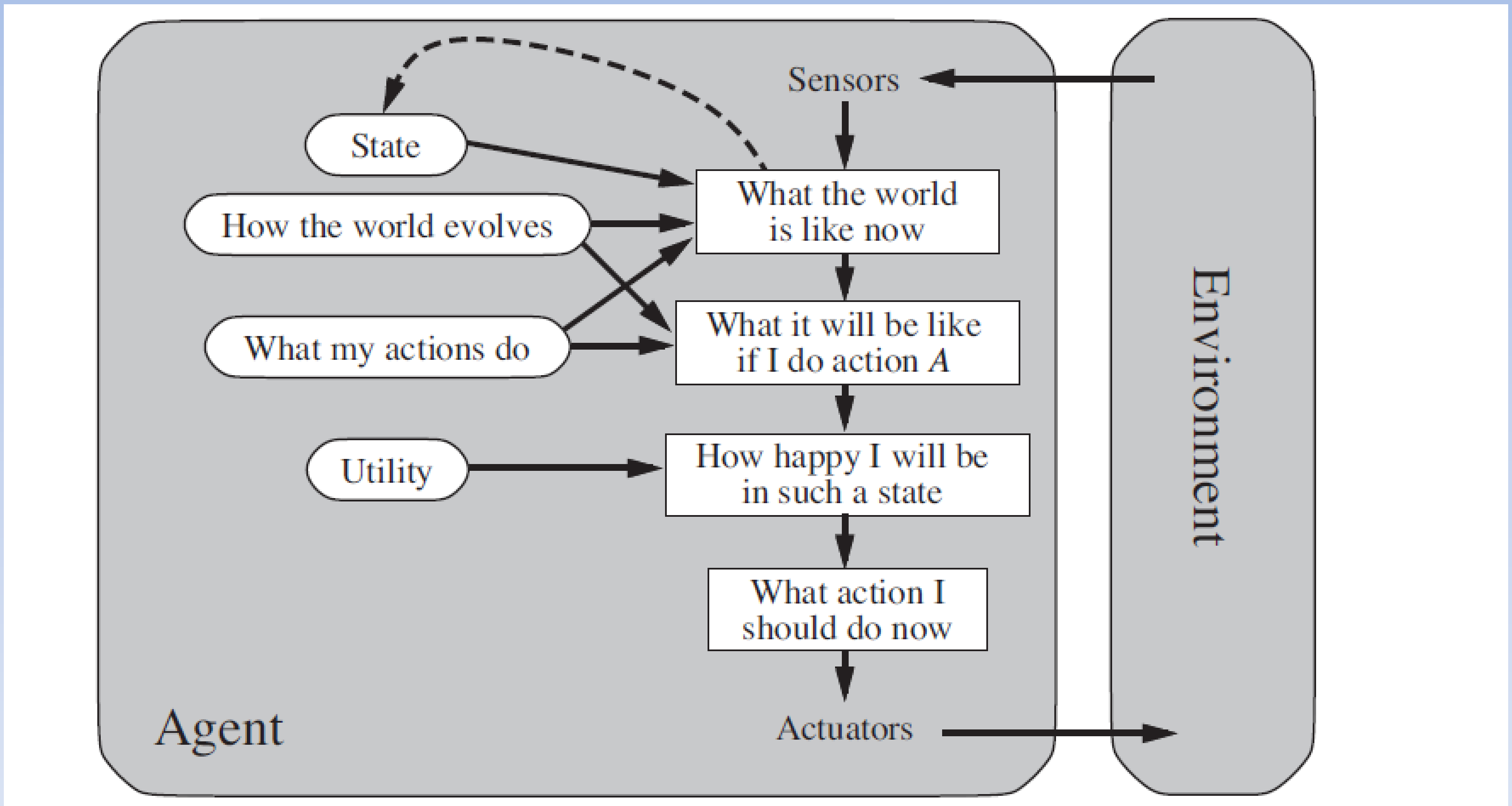
Goal-based Agent



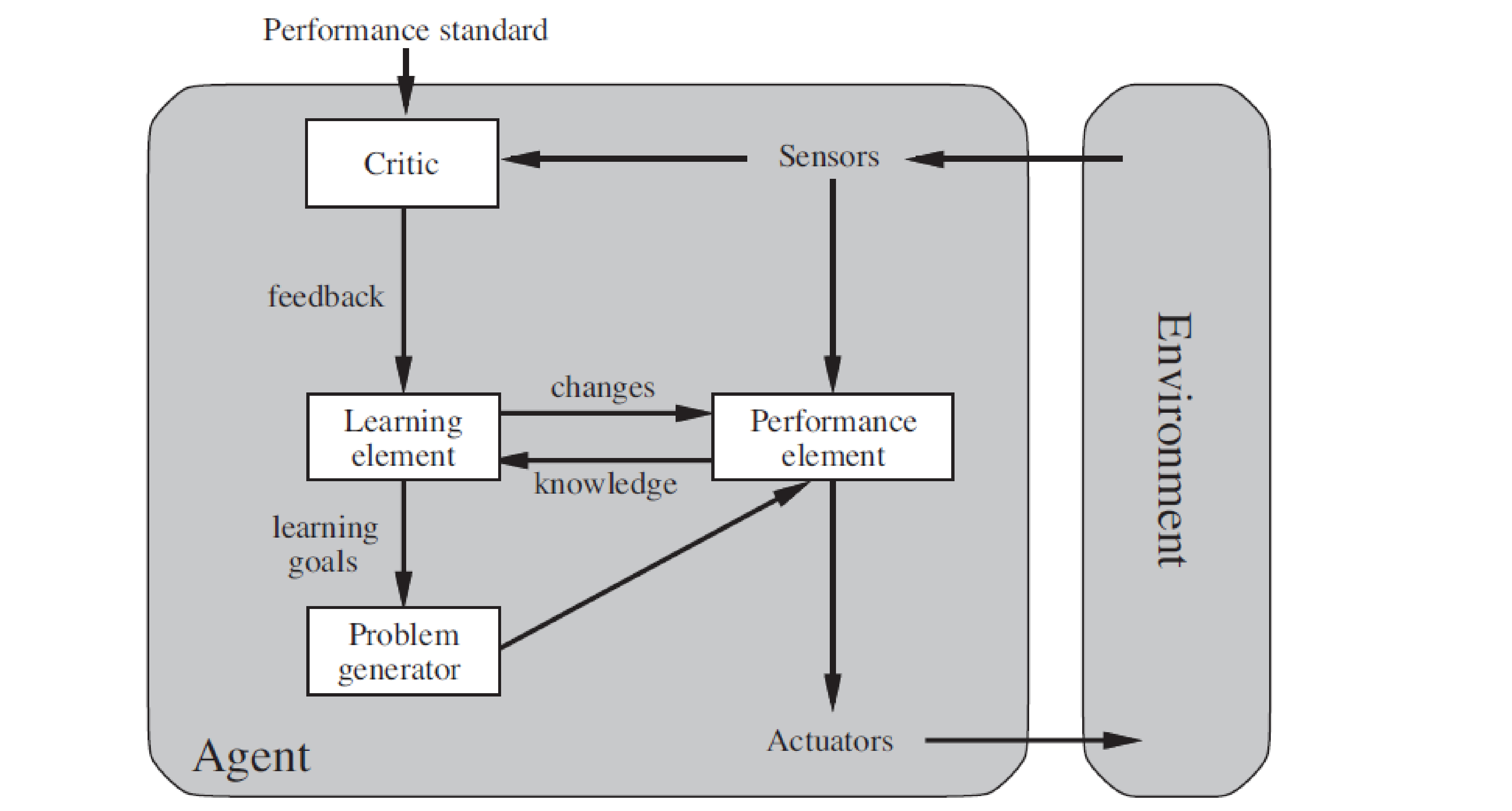
Utility-based Agent

- Uses World States
- Uses a UTILITY Function:
  - Evaluates the preference for states

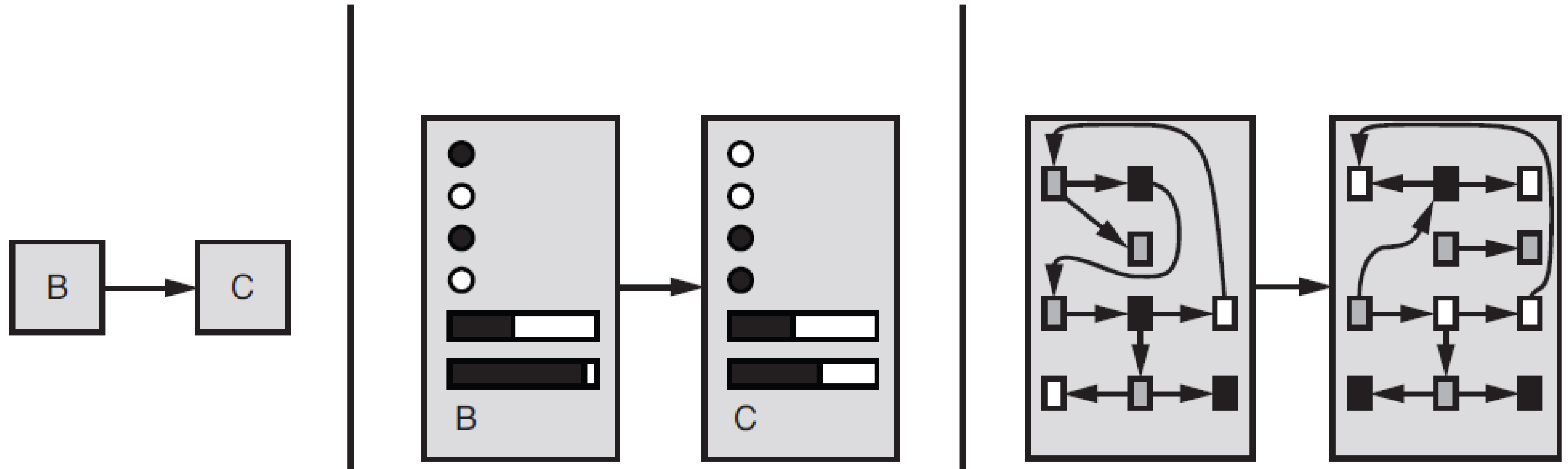
Utility-based Agent



Learning Agent

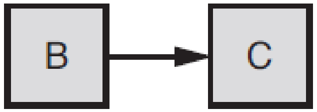


Environment Representations



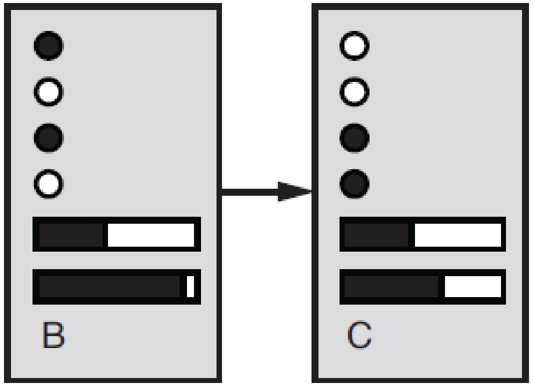
(a) Atomic (b) Factored (b) Structured

Atomic



(a) Atomic

- The world is indivisible
  - No internal structure
- Chapter 3-5 will utilize this type of representation.
- Hidden Markov Models (HMM) utilize this representation (Chapter 15)

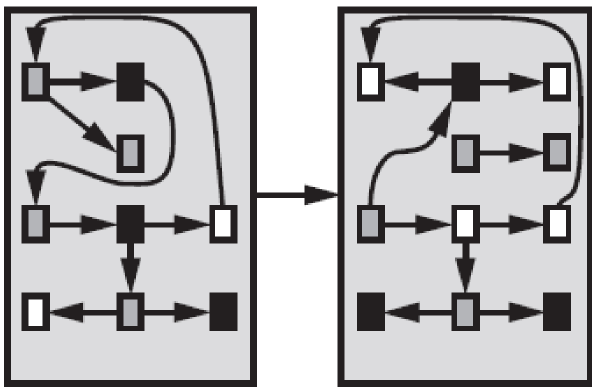


(b) Factored

Factored

- State is split up into fixed set of variables or features.
  - Sudoku represented as a set of variables each with an assigned values.
  - States can now share structures.
- Constraint Satisfaction Algorithms (Chapter 6)
- Propositional Logic/Planning (Chapter 7/10/11)
- Bayesian Networks (13-16)
- Machine Learning (18, 20, 21)

Structured



(b) Structured

- Structured representations are most expressive.
- Allow objects together with relationships between objects.
- Underlie database representations.
- First-Order Logic (Chapter 8, 9, 12)
- Natural Language Understanding (22, 23)