

AI Programming

CS662-2008F-04

Agent Oriented Programming

David Galles

Department of Computer Science
University of San Francisco

04-0: Agents & Environments

- What is an “Agent”
- What is an “Environment”
- Types of Agents

04-1: Agents

- Agent Definition from textbook:
 - Perceives the environment (using sensors)
 - Acts upon the environment (using activators)

04-2: Qualities of an Agents

- Autonomy
- Adaptation
- Goal-Directed behavior
- “beliefs” and “intentions”
- Proactive
- Situated in an environment
 - Potentially a simulated or virtual environment
 - Web is a fine “environment” for software agents

04-3: Autonomous Agent

- Rely on its percepts and past experience to make decisions
- Does not require direct user intervention
- Agents usually do not have complete autonomy
 - Why would we not want an agent to have complete autonomy?
- Research Area: Designing an agent that can reason about its own autonomy and know when to ask for help

04-4: Agent Oriented Programming

- “Objects” are passive: receive messages, return data
 - Objects have no agenda, do what you ask of them
- “Agents” are active
 - Can perceive the world, or some part of the world
 - Set of goals, things they want to accomplish
 - Act in the world to achieve those goals
- Example simple “agent”: Thermostat

04-5: Agents as Functional Programming

- Could describe agents using functional programming
- Action = $F(\text{current-percept}, \text{percept-history})$
- Maps a percept sequence to an action

04-6: Agent Example: Vacuum-cleaner world

- Robot vacuum cleaner, cleans your carpet
- Roomba
- Reflex agent (for the most part)
 - Maps current percepts directly to actions
 - Doesn't store past history

04-7: Agent Example: Vacuum-cleaner world

- Two rooms, A and B. Each can be clean or dirty
- Agent's *environment*
- Agent has:
 - Sensors
 - Actuators
 - Percepts
 - Actions

04-8: Agent Example: Vacuum-cleaner world

- Could list all possible percept sequences and associated actions
- Table-base, lookup agent
- Great for simple worlds – perfect behavior – but doesn't scale
- Need a more compact representation of table
 - Give up some accuracy for tractable table size

04-9: Rationality

- We want our agents to be *Rational* – that is, we want them to do the “right thing”
- What is the “right thing”? Performance measure
 - Condition or state of the world we want to achieve
 - Vacuum cleaner world: “Both rooms are clean” – could have additional criteria – minimize time or power consumption

04-10: Rationality

- Rationality is a specification of an *outcome*, rather than a set of behaviors
- A rational agent tries to maximize its performance measure, given percepts and actions
- From text: 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

04-11: Rationality

- “expected” vs. actual – We don’t require that our agent be able to predict the future, or predict unlikely events
- Information gathering might also be a rational action
 - Crossing the street without looking is irrational
- Rational agents must be able to *learn*, if they are situated in complicated environments
 - “Learning” == improving the agents performance in the current environment
 - Reducing uncertainty

04-12: Environments

- All agents exist in an environment
 - Software agents can exist in a software environment
- Task Environment
 - Performance measure
 - Environment
 - Actuators available to the agent
 - Sensors available to the agent

04-13: Environments

- Observability
- Deterministic/stochastic
- Episodic vs sequential
- Static vs Dynamic
- Discrete vs continuous
- Single-agent vs multi-agent

04-14: Observability

- Fully Observable: Agents sensors always give complete information
- Don't need to build a model of the world – can directly view it
 - Chess Playing: Fully observable
 - Bridge / Poker: Partially observable
 - Vacuum cleaner world: Partially observable (if it has only a local sensor – can't see the other room)

04-15: Deterministic / Stochastic

- Word goes through state transitions
- $(CurrentState, AgentAction) \rightarrow NewState$
- Transition is unique, world is deterministic
 - Chess: Deterministic
 - Vacuum world (simplist form): Deterministic
 - In the real world, stochastic elements – unreliable sensors, imperfect suction, etc
 - Driving a car (or just about any agent embedded in the real world): Stochastic

04-16: Deterministic / Stochastic

- We care about *the agent's point of view*
 - We will avoid issues like “Is the world deterministic with sufficient information”
 - Some overlap between Deterministic / stochastic and observability
- Strategic: Deterministic, except for the actions of other agents

04-17: Episodic / Sequential

- Episodic: Each action is independent
 - Perceive, decide, act. Repeat
 - Next decision does not depend on previous states
 - Don't need to think ahead
 - Many diagnosis problems: Spam filter, image recognition, etc
- Sequential: Current decision effects future
 - Make a series of actions to accomplish a goal
 - Involves planning
 - Chess, driving a car

04-18: **Static / Dynamic**

- **Static:**
 - World does not change while agent is thinking
 - No time pressure
 - Chess (mostly), classification
- **Dynamic:**
 - World changes while the agent is thinking
 - Agent must act within time pressure
 - Driving a car, most interactions in the physical world

04-19: Static / Dynamic

- Semidynamic
 - Environment doesn't change, but time pressure
 - Chess with a clock
 - Timed test
 - Classification, when we need the answer *right now*

04-20: Discrete / Continuous

- Consider the state of the world, the percepts and actions of the agents, and how time is handled
- If the possible values are a discrete set, environment is discrete with respect to that characteristic
- If the values are continuously changing, environment is continuous

04-21: Discrete / Continuous

- Discrete
 - Chess, poker, backgammon
- Continuous
 - Image analysis (continuous sensor), car navigation, assembly robot (most all interactions with the physical world)

04-22: Single Agent / Multi-Agent

- Single agent: acting on our own
- Multi-Agent: actions / goals strategies of other agents must be taken into account
- Competitive vs. Cooperative
 - Can have overlap!
- Sometimes easier to view a world with multiple agents as a single-agent, stochastic environment
 - Traffic signals

04-23: Environment Examples

- Chess playing, poker playing, slot-machine playing
- Mars orbiter
- Web-crawling agent
- Interactive English tutor
- Medical diagnosis agent
- Airport face recognition system

Observable, Deterministic/Stochastic, Episodic / Sequential, Static / Dynamic, Discrete / Continuous, Single / Multi

04-24: Types of Agents

- Table-driven agent
- Reflex agent
- Model-based reflex agent
- Goal-based agent
- Utility-based agent
- Learning agent

04-25: Table-Driven Agent

- Keeps a dictionary that maps percept sequences to actions
- Only works for very small domains

```
class TableDrivenAgent(Agent):  
  
    def __init__(self, table):  
        self.table = table  
        self.percepts = []  
  
    def doAction(percept):  
        percepts.append(percept)  
        return table[tuple(percepts)]
```

04-26: Table-Driven Agent

- Exception Handlers
- Square root / log tables
- Doesn't scape at all
 - Chess is a “simple” environment
 - 10^{50} percept sequences

04-27: Reflex Agent

- Select the action based on the current percept
- Ignore history
- Agent is only rational if best action can be chosen based on current percepts
 - Classification agents
 - Thermostat agent
 - Wall following robot

04-28: Reflex Agent

```
class ReflexVacuumAgent(Agent):
    def DoAction(location, status):
        if status == 'dirty':
            return 'suck'
        elif location == loc_A:
            return 'right'
        else:
            retrn 'left'
```

04-29: Model-Based Reflex Agent

- Maintains an internal representation of the world
 - Keep *state* information about the world
- Actions are selected based on the model and current percepts

04-30: Model-Based Reflex Agent

```
class ModelBasedVacuumAgent(Agent):
    def __init__(self):
        self.model = {"Loc_a" : None, "Loc_b" : None}
    def DoAction(location, status):
        self.model[location] = status
        if self.model["Loc_a"] == self.model["Loc_b"] == "clean":
            return 'NoOp'
        elif status == 'dirty':
            return 'suck'
        elif location == loc_A:
            return 'right'
        else:
            return 'left'
```


04-31: Model-Based Reflex Agent

- Examples
 - Vacuum-cleaner agent (with map)
 - Factory Robots
 - Mail delivery robot

04-32: Model-Based Reflex Agent

- Types of models
 - Attributes & values (variables)
 - Probability distribution over variables
 - Data Structures
 - Maps / Graphs / Finite State Machines
 - Facts (propositional logic)

04-33: Goal-Based Agent

- Correct action depends upon what the agent is trying to accomplish
 - Agents knowledge (model)
 - Current state (percepts)
 - What it is trying to achieve (goals)
- Select actions that will accomplish goals
- Often need to do search and planning to determine which action to take in a given situation

04-34: Goal-Based Agent

- Example Goal-based agents:
 - Chess playing robot
 - Taxi-driving robot
- Can blur the lines a little
 - Simple mail delivery robot that follows a set route
 - More robust mail delivery robot that can replan route to handle obstacles

04-35: Utility-Based Agent

- May be many action sequences that achieve a goal
- Utility is used to compare the relative desirability of action sequences
- Cost of actions, time required, relative value of different outcomes
- Useful for partially observable or stochastic environments

04-36: Utility-Based Agent

- Utility function
 - Maps a state of the world onto a real number
 - “Goodness”, or utility of the state
 - Agent tries to maximize utility
 - Useful when there is an easy mapping to utility – like money
 - Online trading, gambling, probability & uncertain environments

04-37: Learning Agent

- An agent may need to update its agent program
- Programmer may not completely understand the environment, or coding by hand may be tedious (learn from the environment, instead of the programmer)
- Environment may change
- A Learning Agent is one that improves its performance with respect to a set of tasks over time
- Essential in complex environments

04-38: Learning Agent

- Learning agents need a performance element and a learning element
 - Performance element: Select current action
 - Learning element: evaluate the correctness of the performance element

04-39: Learning Agent

- Learning can happen *offline* or *online*
- Learning can be *passive* or *active*
- Learning can be *supervised* or *unsupervised*
- Credit assignment is a big problem when learning in sequential environments
 - How do we know which action was correct, and which was bad?

04-40: Summary

- Agent is an autonomous program situated in an environment
- Agent behaves rationally if it acts to optimize its expected performance measure
- Characterizing the environment can help us decide how to build an agent
- More complex environments (usually!) require more sophisticated agent programs