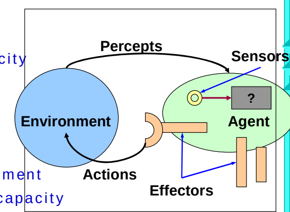
Topic 3: Intelligent Agents

(Lecture notes by W.Hsu are used)

- Today's Reading: Chapter 2, Russell and Norvig or/and Luger 1.1.4
- Intelligent Agent (IA) Design
  - Shared requirements, characteristics of IAs
  - Methodologies
    - Software agents
    - Reactivity vs. state
    - Knowledge, inference, and uncertainty
- Intelligent Agent Frameworks
  - Reactive
  - With state
  - Goal-based
  - Utility-based

Intelligent Agents:Overview

- Agent: Definition
  - Any entity that perceives its environment through sensors and acts upon that environment through effectors
  - Examples (class discussion): human, robotic, software agents
- Perception
  - Signal from environment
  - May exceed sensory capacity
- Sensors
  - Acquires percepts
  - Possible limitations
- Action
  - Attempts to affect environment
  - Usually exceeds effector capacity
- Effectors
  - Transmits actions
  - Possible limitations



How Agents Should Act

- Rational Agent: Definition
  - Informal: "does the right thing, given what it believes from what it perceives"
  - What is "the right thing"?
    - First approximation: *action that maximizes success of agent*
    - Limitations to this definition?
  - Issues to be addressed now
    - How to evaluate *success*
    - When to evaluate *success*
  - Issues to be addressed later in this course
    - Uncertainty (in environment, in actions)
    - How to express beliefs, knowledge

How Agents Should Act

- Why Study Rationality?
  - Recall: aspects of intelligent behavior (last lecture)
    - Engineering objectives: optimization, problem solving, decision support
    - Scientific objectives: modeling correct inference, learning, planning
  - Rational cognition: formulating *plausible* beliefs, conclusions
  - Rational action: "doing the right thing" given beliefs

Rational Agents

- "Doing the Right Thing"
  - Committing actions
    - Limited to set of effectors
    - In context of what agent knows
  - Specification (cf. software specification)
    - Preconditions, postconditions of operators
    - Caveat: not always perfectly known (for given environment)
    - Agent may also have limited knowledge of specification

Rational Agents

- Agent Capabilities: Requirements
  - Choice: select actions (and carry them out)
  - Knowledge: represent knowledge about environment
  - Perception: capability to sense environment
  - Criterion: *performance measure to define degree of success*
- Possible Additional Capabilities
  - Memory (internal model of state of the world)
  - Knowledge about effectors, reasoning process (reflexive reasoning)

Measuring Performance

- Performance Measure: *How* to Determine Degree of Sucesss
  - Definition: *criteria that determine how successful agent is*
  - Clearly, varies over
    - Agents
    - Environments
  - Possible measures?
    - Subjective (agent may not have capability to give accurate answer!)
    - Objective: *outside observation*

Measuring Performance

- Example: web crawling agent
  - Number of hits
  - Number of *relevant* hits
  - *Ratio* of relevant hits to pages explored, resources expended
  - Caveat: "you get what you ask for" (issues: redundancy, etc.)
- When to Evaluate Success
  - Depends on objectives (short-term efficiency, consistency, etc.)
  - Is task episodic? Are there milestones? Reinforcements? (e.g., games)

Knowledge in Agents

- Rationality versus Omniscience
  - Nota Bene (*NB*): not the same
  - Distinction
    - Omniscience: knowing *actual* outcome of all actions
    - Rationality: knowing *plausible* outcome of all actions
    - Example: is crossing the street to greet a friend too risky?
  - Key question in AI
    - *What is a plausible outcome?*
    - Especially important in knowledge-based expert systems
    - Of practical important in planning, machine learning

Knowledge in Agents

- Related questions
  - *How can an agent make rational decisions given beliefs about outcomes of actions?*
  - *Specifically, what does it mean (algorithmically) to "choose the best"?*
- Limitations of Rationality
  - Based only on what agent *can* perceive and do
  - Based on what is "likely" to be right, not what "turns out" to be right

What Is Rational?

- Criteria
  - Determines what is rational *at any given time*
  - Varies with agent, environment, *situation*
- Performance Measure
  - Specified by outside observer or evaluator
  - Applied (consistently) to (one or more) IAs in given environment
- Percept Sequence
  - Definition: *entire history* of percepts gathered by agent
  - NB: may or may not be retained fully by agent (issue: state and memory)

What Is Rational?

- Agent Knowledge
  - Of environment – "required"
  - Of self (reflexive reasoning)
- Feasible Action
  - What can be performed
  - What agent believes it can attempt?

Ideal Rationality

- Ideal Rational Agent
  - Given: any possible percept sequence
  - Do: ideal rational behavior
    - Whatever action is expected to maximize performance measure
    - NB: expectation – informal sense (for now); mathematical foundation soon
  - Basis for action
    - Evidence provided by percept sequence
    - Built-in knowledge possessed by the agent

Ideal Rationality

- Ideal Mapping from Percepts to Actions
  - Figure 2.2, R&N
  - Mapping *p*: *percept sequence* → *action*
    - Representing *p* as list of pairs: infinite (unless explicitly bounded)
    - Using *p*: specifies ideal mapping from percepts to actions (i.e., ideal agent)
    - Finding explicit *p*: in principle, could use trial and error
    - *Other (implicit) representations may be easier to acquire!*

Autonomy

- Built-In Knowledge
  - *What if agent ignores percepts?*
  - Possibility
    - All actions based on agent's own knowledge
    - Agent said to *lack* autonomy
  - Examples
    - "Preprogrammed" or "hardwired" industrial robots
    - Clocks
    - Other sensorless automata
    - NB: to be distinguished from closed versus open loop systems

Autonomy[2]

- Justificiation for Autonomous Agents
  - Sound engineering practice: "Intelligence demands robustness, adaptivity"
  - This course: mathematical and CS basis of autonomy in IAs

Structure of Intelligent Agents

- Agent Behavior
  - Given: sequence of percepts
  - Return: IA's actions
    - Simulator: description of results of actions
    - Real-world system: committed action
- Agent Programs
  - Functions that implement *program*
  - Assumed to run in computing environment (architecture)
    - Hardware architecture: computer organization
    - Software architecture: programming languages, operating systems
  - *Agent = architecture + program*

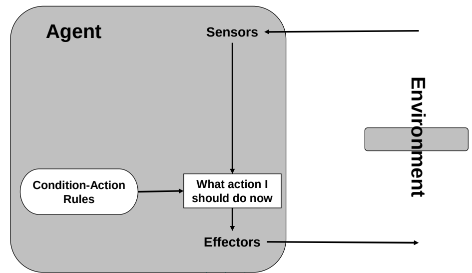
Example:Automated Taxi Driver

- Agent Type: Taxi Driver
- Percepts
  - Visual: cameras
  - Profilometer: speedometer, tachometer, odometer
  - Other: GPS, sonar, interactive (microphone)
- Actions
  - Steer, accelerate, brake
  - Talk to passenger

Example:Automated Taxi Driver[2]

- Goals
  - Safe, legal, fast, comfortable
  - Maximize profits
- Environment
  - Roads
  - Other traffic, pedestrians
  - Customers
- Discussion: Performance Requirements for Open Ended Task

Agent Framework: Simple Reflex Agents [1]



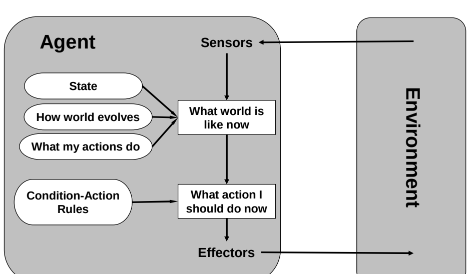
Agent Framework: Simple Reflex Agents [2]

- Implementation and Properties
  - Instantiation of generic skeleton agent: Figure 2.8
  - function *SimpleReflexAgent* (*percept*) returns action
    - static: *rules*, set of condition-action rules
    - *state* ← *Interpret-Input* (*percept*)
    - *rule* ← *Rule-Match* (*state, rules*)
    - *action* ← *Rule-Action* {*rule*}
    - return *action*

Agent Framework: Simple Reflex Agents [3]

- Advantages
  - Selection of best action based only on current state of world and rules
  - Simple, very efficient
  - *Sometimes* robust
- Limitations and Disadvantages
  - No memory (doesn't keep track of world)
  - Limits range of applicability

Agent Frameworks: (Reflex) Agents with State [1]



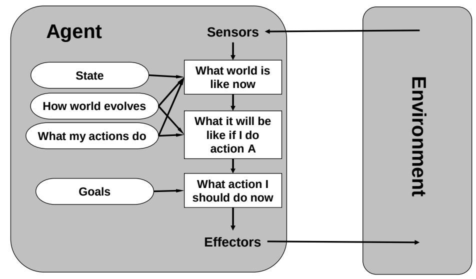
Agent Frameworks: (Reflex) Agents with State [2]

- Implementation and Properties
  - Instantiation of generic skeleton agent: Figure 2.10
  - function *ReflexAgentWithState* (*percept*) returns action
    - static: *state*, description of current world state; *rules*, set of condition-action rules
    - *state* ← *Update-State* (*state, percept*)
    - *rule* ← *Rule-Match* (*state, rules*)
    - *action* ← *Rule-Action* {*rule*}
    - return *action*

Agent Frameworks: (Reflex) Agents with State [3]

- Advantages
  - Selection of best action based only on current state of world and rules
  - Able to reason over past states of world
  - Still efficient, *somewhat* more robust
- Limitations and Disadvantages
  - No way to express goals and preferences relative to goals
  - Still limited range of applicability

Agent Frameworks: Goal-Based Agents [1]



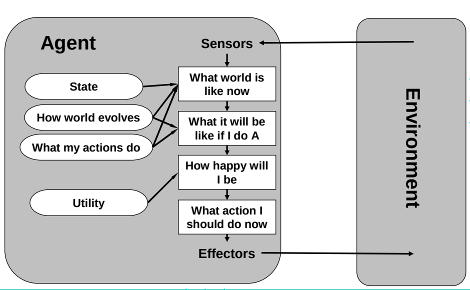
Agent Frameworks: Goal-Based Agents [2]

- Implementation and Properties
  - Instantiation of generic skeleton agent: Figure 2.11
  - Functional description
    - Chapter 13: classical planning
    - Requires more formal specification

Agent Frameworks: Goal-Based Agents [3]

- Advantages
  - Able to reason over goal, intermediate, and initial states
  - Basis: automated reasoning
    - One implementation: theorem proving (first-order logic)
    - Powerful representation language and inference mechanism
- Limitations and Disadvantages
  - Efficiency limitations: can't feasible solve many general problems
  - No way to express preferences

Agent Frameworks: Utility-Based Agents [1]



Agent Frameworks: Utility-Based Agents [2]

- Implementation and Properties
  - Instantiation of generic skeleton agent: Figure 2.8
  - function *SimpleReflexAgent* (*percept*) returns action
    - static: *rules*, set of condition-action rules
    - *state* ← *Interpret-Input* (*percept*)
    - *rule* ← *Rule-Match* (*state, rules*)
    - *action* ← *Rule-Action* (*rule*)
    - return *action*

Agent Frameworks: Utility-Based Agents [3]

- Advantages
  - Selection of best action based only on current state of world and rules
  - Simple, very efficient
  - *Sometimes* robust
- Limitations and Disadvantages
  - No memory (doesn't keep track of world)
  - Limits range of applicability

Looking Ahead: Search

Solving Problems by Searching
- Problem solving agents: design, specification, implementation
- Specification components
  - Problems – formulating *well-defined* ones
  - Solutions – requirements, constraints
- Measuring performance

- Formulating Problems as (State Space) Search
- Data Structures Used in Search

Problem-Solving Agents [1]: Preliminary Design

- Justification
  - Rational IAs: act to *reach* environment that maximizes performance measure
  - Need to formalize, operationalize this definition
- Practical Issues
  - Hard to find appropriate *sequence of states*
  - Difficult to translate into IA design
- Goals
  - Chapter 2, R&N: simplifies task of translating agent specification to formal design
  - First step in problem solving: formulation of goal(s) – "accept no substitutes"
  - Chapters 3-4, R&N: goal ≡ {world states | goal test is satisfied}

Problem-Solving Agents [2]: Preliminary Design

- Problem Formulation
  - Given: initial state, desired goal, specification of actions
  - Find: *achievable* sequence of states (actions) mapping from initial to goal state
- Search
  - Actions: cause transitions between world states (e.g., applying effectors)
  - Typically specified in terms of finding sequence of states (operators)

Problem-Solving Agents [1]:Specification

- Input: Informal Objectives; Initial, Intermediate, Goal States; Actions
- Output
  - Path from initial to goal state
  - Leads to design requirements for state space search problem
- Logical Requirements
  - States: representation of state of world (example: starting city, graph representation of Romanian map)
  - Operators: descriptors of possible actions (example: moving to adjacent city)
  - Goal test: state → boolean (example: at destination city?)
  - Path cost: *based on search, action costs* (example: number of edges traversed)

Problem-Solving Agents [2]: Specification

- Operational Requirements
  - Search algorithm to find path
  - Objective criterion: minimum cost (this and next 3 lectures)
- Environment
  - Agent can search in environment according to specifications
  - Sometimes has full state and action descriptors; *sometimes not*!