



Lecture 2 of 42

Problem Solving by Search Discussion: Problem Set 1, Term Projects 2 of 3

Friday, 24 August 2007

William H. Hsu

Department of Computing and Information Sciences, KSU

KSOL course page: <http://snipurl.com/v9v3>

Course web site: <http://www.kddresearch.org/Courses/Fall-2007/CIS730>

Instructor home page: <http://www.cis.ksu.edu/~bhsu>

Reading for Next Class:

Sections 2.3 – 2.5, p. 39 – 56, Russell & Norvig 2nd edition

Section 3.1, p. 59 – 62, Russell & Norvig 2nd edition



Term Project Topics, Fall 2007 (review)

- **1. Game-playing Expert System**
 - * “Borg” for Angband computer role-playing game (CRPG)
 - * <http://www.thangorodrim.net/borg.html>
- **2. Trading Agent Competition (TAC)**
 - * Supply Chain Management (TAC-SCM) scenario
 - * <http://www.sics.se/tac/>
- **3. Machine Learning for Bioinformatics**
 - * Evidence ontology for genomics or proteomics
 - * <http://bioinformatics.ai.sri.com/evidence-ontology/>





Agent Programs (Review)

● Software Agents

- * Also known as (*aka*) software robots, softbots
- * Typically exist in very detailed, unlimited domains
- * Examples
 - ⇒ Real-time systems: critiquing, avionics, shipboard damage control
 - ⇒ Indexing (spider), information retrieval (IR; e.g., web crawlers) agents
 - ⇒ Plan recognition systems (computer security, fraud detection monitors)
- * See: Bradshaw (*Software Agents*)

● Focus of This Course: Building IAs

- * Generic skeleton agent: Figure 2.4, R&N
- * function *SkeletonAgent* (*percept*) returns action
 - ⇒ static: *memory*, agent's memory of the world
 - ⇒ *memory* ← *Update-Memory* (*memory*, *percept*)
 - ⇒ *action* ← *Choose-Best-Action* (*memory*)
 - ⇒ *memory* ← *Update-Memory* (*memory*, *action*)
 - ⇒ return *action*



PEAS Framework

● Performance Measure

- * Specified by outside observer or evaluator
- * Applied (consistently) to (one or more) IAs in given environment

● Environment

- * Reachable states
- * "Things that can happen"
- * "Where the agent can go" TAC-SCM
- * To be distinguished (TBD) from: observable states

● Actuators

- * What can be performed
- * Limited by physical factors *and* self-knowledge

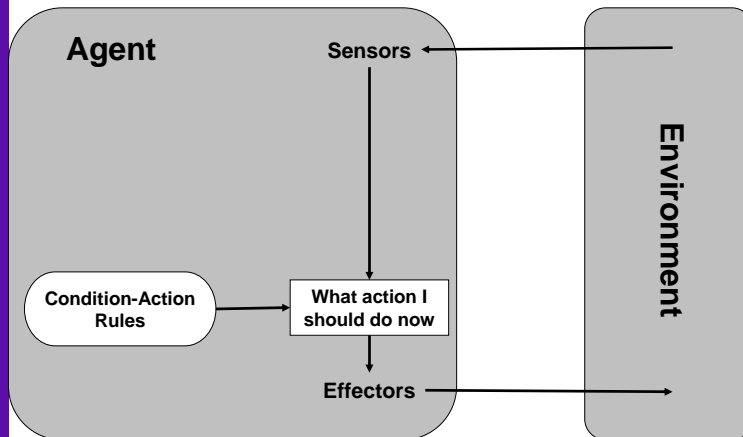
● Sensors

- * What can be observed
- * Subject to error: measurement, sampling, postprocessing





Agent Framework: Simple Reflex Agents [1]



Agent Framework: Simple Reflex Agents [2]

- **Implementation and Properties**

- * **Instantiation** of generic skeleton agent: Figs. 2.9 & 2.10, p. 47 R&N 2^e
- * **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*

- **Advantages**

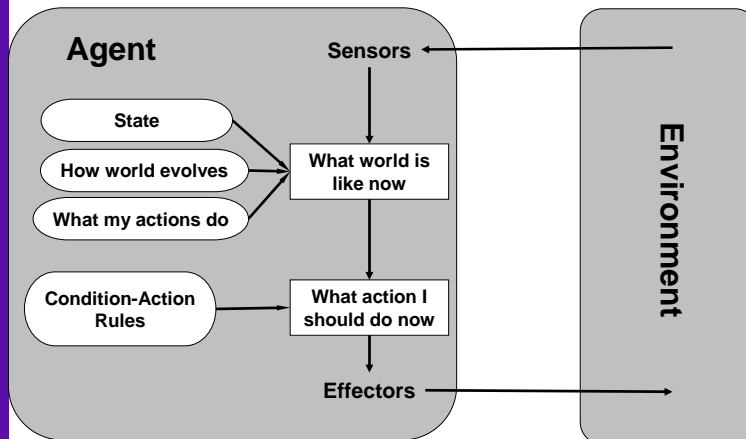
- * Selection of best action based only on rules, current state of world
- * Simple, very efficient
- * *Sometimes* robust

- **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 skeleton agent:** Figures 2.11 & 2.12, p. 49 R&N 2^e
- * **function *ReflexAgentWithState* (*percept*) returns action**
 - ⇒ **static:** *state* description; *rules*, set of condition-action rules
 - ⇒ *state* ← *Update-State* (*state*, *percept*)
 - ⇒ *rule* ← *Rule-Match* (*state*, *rules*)
 - ⇒ *action* ← *Rule-Action* {*rule*}
 - ⇒ **return** *action*

- **Advantages**

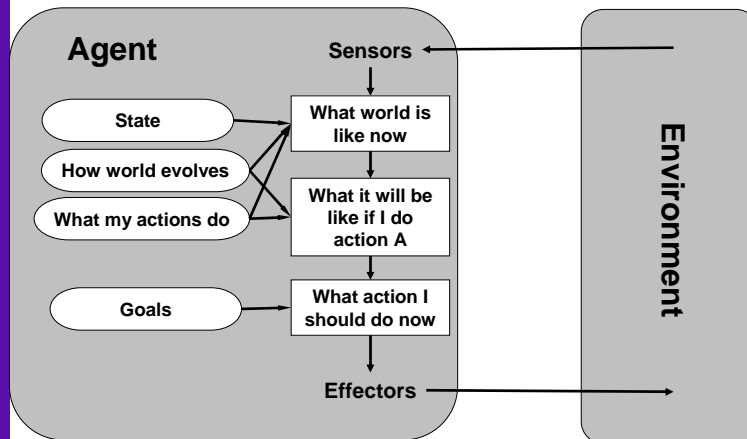
- * Selection of best action based only on rules, current state of world
- * Able to reason over past states of world
- * Still efficient, *somewhat* more robust

- **Limitations and Disadvantages**

- * No way to express **goals** and **preferences** relative to goals
- * Still limited range of applicability



Agent Frameworks: Goal-Based Agents [1]

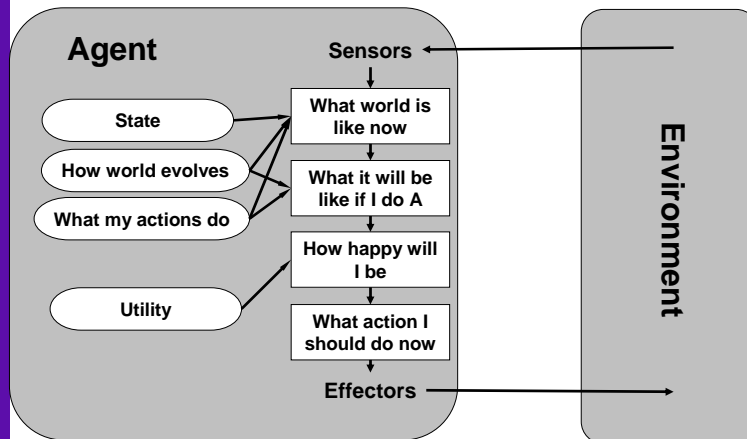


Agent Frameworks: Goal-Based Agents [2]

- **Implementation and Properties**
 - * **Instantiation** of skeleton agent: Figure 2.13, p. 50 R&N 2^e
 - * **Functional description**
 - ⇒ Chapter 11-12 R&N 2^e: classical planning
 - ⇒ Requires more formal specification
- **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**
 - * **May be expensive: can't feasibly 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 skeleton agent: Figure 2.14, p. 53 R&N 2^e
- * **Functional description**
 - ⇒ Chapter 16-17 R&N 2e: making decisions
 - ⇒ Requires representation of decision space

● Advantages

- * **Able to account for uncertainty and agent preferences**
- * **Models value of goals: costs vs. benefits**
- * **Essential in economics, business; useful in many domains**

● Limitations and Disadvantages

- * **How to get utilities?**
- * **How to reason under uncertainty? (Examples?)**



Looking Ahead: Search

- Next Monday - Wednesday: Sections 3.1-3.4, Russell and Norvig
- Thinking Exercises (Discussion in Next Class): 3.3 (a, b, e), 3.9
- Solving Problems by Searching
 - * Problem solving agents: design, specification, implementation
 - * Specification: problem, solution, constraints
 - * Measuring performance
- Formulating Problems as (State Space) Search
- Example Search Problems
 - * Toy problems: 8-puzzle, N-queens, cryptarithmic, toy robot worlds
 - * Real-world problems: layout, scheduling
- Data Structures Used in Search
- Next Monday: Uninformed Search Strategies
 - * State space search handout (Winston)
 - * Search handouts (Ginsberg, Rich and Knight)



Problem-Solving Agents [1]: Goals

- **Justification**
 - * Rational IA: 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**
 - * Translating agent specification to formal design
 - * Chapter 2, R&N: decision loop simplifies task
 - * First step in problem solving: formulation of goal(s)
 - * Chapters 3-4, R&N: state space search
 - ⇒ Goal \equiv {world states | goal test is satisfied}
 - ⇒ Graph planning
 - * Chapter 5: constraints – domain, rules, moves
 - * Chapter 6: games – evaluation function





Problem-Solving Agents [2]: Definitions

- **Problem Formulation**

- * **Given**

- ⇒ Initial state
- ⇒ Desired goal
- ⇒ Specification of actions

- * **Find**

- ⇒ *Achievable* sequence of states (actions)
- ⇒ Represents 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 [3]: Requirements and Specification

- **Input**

- * Informal objectives
- * Initial, intermediate, goal states
- * Actions
- * Leads to design requirements for state space search problem

- **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 [4]: Objectives

- **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
 - * May have full state and action descriptors
 - * *Sometimes not!*



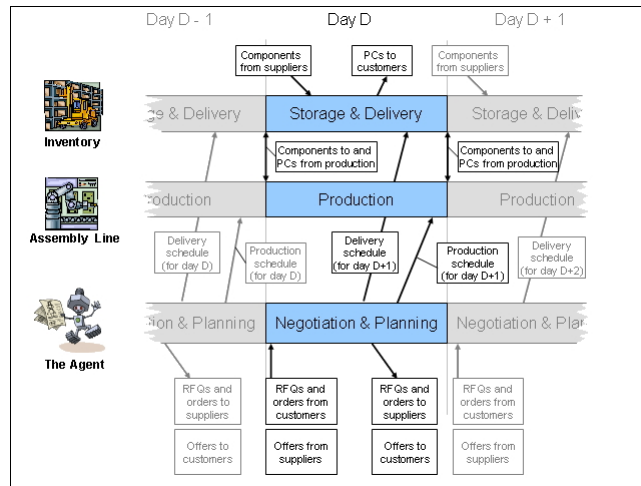
Problem-Solving Agents [5]: Implementation

- **function** *Simple-Problem-Solving-Agent* (*p*: percept) **returns** *a*: action
 - * **inputs:** *p*, percept
 - * **static:** *s*, action sequence (initially empty)
state, description of current world state
g, goal (initially null)
problem, problem formulation
 - * *state* ← *Update-State* (*state*, *p*)
 - * **if** *s.Is-Empty*() **then**
 - ⇒ *g* ← *Formulate-Goal* (*state*) // focus of today's class
 - ⇒ *problem* ← *Formulate-Problem* (*state*, *g*) // today
 - ⇒ *s* ← *Search* (*problem*) // next week
 - * *action* ← *Recommendation* (*s*, *state*)
 - * *s* ← *Remainder* (*s*, *state*) // discussion: meaning?
 - * **return** (*action*)
- **Ch. 3-4: Implementation of *Simple-Problem-Solving-Agent***





Example: TAC-SCM Agent [1] Project Topic 2 of 3



Trading Agent Competition Supply Chain Management Scenario
© 2002 Swedish Institute of Computer Science



Example: TAC-SCM Agent [2] Problem Specification

- **Trading Agent Competition**
 - * Swedish Institute of Computer Science (SICS) Page
<http://www.sics.se/tac/>
 - * Supply chain management (SCM) scenario
<http://www.sics.se/tac/page.php?id=13>
- **Problem Specification**
 - * Study existing TAC-SCM agents
 - * Develop a scheduling and utility-based reasoning system
 - * Use SICS interface to develop a new TAC agent
 - * Play it against other agents using competition server



Formulating Problems [2]: Single-State

- **Single-State Problems**
 - * Goal state is reachable in one action (one move)
 - * World is fully accessible
 - * Example: vacuum world (Figure 3.2, R&N) – simple robot world
- **Significance**
 - * Initial step analysis
 - * “Base case” for problem solving by regression
 - ⇒ General Problem Solver
 - ⇒ Means-ends analysis



Formulating Problems [2]: Multi-State

- **Multi-State Problems**
 - * Goal state may not be reachable in one action
 - * Assume limited access
 - ⇒ effects of actions known
 - ⇒ may or may not have sensors
- **Significance**
 - * Need to reason over states that agent can get to
 - * May be able to guarantee reachability of goal state anyway
- **Determining A State Space Formulation**
 - * State space – single-state problem
 - * State set space – multi-state problems





Terminology

- **Agent Types**
 - * Reflex aka “reactive”
 - * Reflex with state (memory-based)
 - * Goal-based aka “deliberative”
 - * Preference-based aka “utility-based”
- **Decision Cycle**
- **Problem Solving Frameworks**
 - * Regression, Means-ends analysis (MEA)
 - * State space search, PEAS
 - * Representations (later)
 - ⇒ Plans
 - ⇒ Constraint satisfaction problems
 - ⇒ Policies and decision processes
 - ⇒ Situation calculus



Summary Points

- **The Basic Decision Cycle for Intelligent Agents**
- **Agent Types**
 - * Reflex aka “reactive”
 - * Reflex with state (memory-based)
 - * Goal-based aka “deliberative”
 - * Preference-based aka “utility-based”
- **Problem Solving Frameworks**
 - * Regression-based problem solving
 - * Means-ends analysis (MEA)
 - * PEAS framework
 - ⇒ Performance
 - ⇒ Environment
 - ⇒ Actuators
 - ⇒ Sensors
 - * State space formulation

