Intelligent Agents: Software and Robotic

“An agent is just something that perceives and acts.” (R&N)

Properties:
- Provides some service to humans
- Has some autonomy
- Can adapt to its environment
- Can interact with other agents
- Has knowledge about its tasks and environment.

Sycara’s Requirements for Agents

- **Situatedness** in either virtual or physical environment
- **Autonomy**, acting without intervention from humans and exerting control over own actions and internal state
- **Adaptivity**, reacting flexibly to changes in environment to pursue goals, learns
- **Sociability**, can act as peer with other agents and humans

Commonalities in Agent Definitions

<table>
<thead>
<tr>
<th>Capability</th>
<th>% in 25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reactive</td>
<td>36</td>
</tr>
<tr>
<td>Autonomous</td>
<td>36</td>
</tr>
<tr>
<td>Software</td>
<td>32</td>
</tr>
<tr>
<td>Deliberative</td>
<td>28</td>
</tr>
<tr>
<td>Intentional</td>
<td>28</td>
</tr>
<tr>
<td>Cooperative or Communicative</td>
<td>24</td>
</tr>
<tr>
<td>Persistence</td>
<td>20</td>
</tr>
<tr>
<td>Adaptable</td>
<td>12</td>
</tr>
</tbody>
</table>

Key Issues

- How do agents decide on goals?
- How do agents decompose tasks?
- How do agents communicate with each other? With humans?
- What capabilities are needed?
- What kinds of architectures support agent design?
### Properties of the Environment

- **Accessible vs. Inaccessible:** complete, accurate, up-to-date information about environment’s state vs. not
- **Deterministic vs. non-deterministic:** actions have single guaranteed effect/ predictable vs. not
- **Episodic vs. non-episodic:** performance depends on discrete episodes with no link between different scenarios
- **Static vs. dynamic:** unchanged except by actions of the agent vs. other forces operating in the environment
- **Discrete vs. continuous:** fixed, finite number of actions and states vs. not

### Abstract Architectures

- Environment is in a set $E$ of discrete states:
  \[ E = \{ e, e', \ldots \} \]
- Agents take actions which transform the state of the environment:
  \[ Ac = \{ \alpha, \alpha', \ldots \} \]
- A run is a sequence of interleaved states and actions:
  \[ r : e_0 \xrightarrow{a_0} e_1 \xrightarrow{a_1} \ldots \xrightarrow{a_{n-1}} e_n \]

Adapted from M. Wooldridge http://www.csc.liv.ac.uk/~mjw/pubs/imas

### Environments (formally)

- A state transformer function represents the environment:
  \[ \tau : R^\alpha \rightarrow \varphi(E) \]
- An environment is a triple
  \[ Env = \langle E, e_0, \tau \rangle \]
  where $E$ is a set of states, initial state is $e_0 \in E$ and $\tau$ is a state transformer function.

Adapted from M. Wooldridge http://www.csc.liv.ac.uk/~mjw/pubs/imas
Agent (formally)

- Agent maps runs to actions:
  \[ \text{Agent} : R^E \rightarrow Ac \]
  by deciding what action to perform based on history of the environment.

Agent Architectures

- Reactive
  - Direct mapping between environment and action
  - Reinforcement learning, reactive planning
- Deliberative
  - Computation or reasoning between sensing and acting

Theses Underlying Reactivity

- The world is its own best model.
- No centralized control
- No internal reasoning/complicated computation
- Complex behavior emerges from interaction of simple behaviors
- Hard-wired or pre-computed responses
- Purely reactive if:
  \[ \text{action}(e) : E \rightarrow Ac \]

Theses Underlying Deliberation

- Internal representation of world
- Sense-Plan-Act cycle
- Probably some off-line, non-real-time processing

Adapted from M. Wooldridge http://www.csc.liv.ac.uk/~mjw/pubs/imas

©CS540 Spring 2014
Expected Utility

- Probability that run \( r \) occurs when agent \( Ag \) is placed in environment \( Env \)

\[
\sum_{r \in R(Ag, Env)} P(r \mid Ag, Env) = 1
\]

- Expected utility of agent \( Ag \) in environment \( Env \) (given \( P, u \)) is:

\[
EU(Ag, Env) = \sum_{r \in R(Ag, Env)} u(r)P(r \mid Ag, Env)
\]

Optimal Agents

- Optimal agent maximizes expected utility, does best on average.
- *Bounded Optimal Agent* is an optimal agent that can actually be implemented on a computer.

Beliefs, Desires and Intentions (BDI)

- A BDI agent is:
  - Deliberative
  -Describable by states of knowledge (beliefs), goals (desires) and commitments to act (intentions)
- Agent Programming Languages: Agent0
- Agents Communication Languages

Roles for Intentions (Bratman)

- Intentions direct future processing – select task for attention
- Once commit to intention, then other intentions must be consistent with it.
- Monitor progress by success of achieving, maintaining or abandoning intentions
Plans and Intentions

- Plans provide means for satisfying intentions.
- Agents coordinate activities via plans.
- Plans may need to be least commitment to support retracting intentions.

Cohen & Levesque: Formal Theory of Rational Agency

- Modal Operators
  - \((\text{BEL} \; x \; p)\)  \(p\) follows from \(x\)'s beliefs
  - \((\text{GOAL} \; x \; p)\)  \(p\) follows from \(x\)'s goals
  - \((\text{HAPPENS} \; a)\)  \(a\) will happen next
  - \((\text{DONE} \; a)\)  \(a\) just happened
  - \((\text{AGT} \; x \; a)\)  \(x\) is the agent for actions \(a\)

- Action Notation
  - \(a.b\)  action \(b\) follows \(a\)
  - \(ab\)  non-deterministic choice
  - \(a|b\)  \(a\) and \(b\) occur concurrently
  - \(p?\)  \(p\) is true?  \(a.p?\)  after \(a\) occurs then \(p\) holds

C&L Example of Persistent Goal

\[(P-\text{GOAL} \; x \; p \; q) = (\text{BEL} \; x \; \neg p) \& (\text{GOAL} \; x \; (\text{LATER} \; p)) \& (\text{KNOW} \; x \; (\text{PRIOR} \; [(\text{BEL} \; x \; p) \vee (\text{BEL} \; x \; \neg p) \vee (\text{BEL} \; x \; \neg q)] \neg [\text{GOAL} \; x \; (\text{LATER} \; p)])]]\]

Means agent \(x\) believes that \(p\) is currently false, makes its becoming true later into a goal, and \(x\) knows that before abandoning this goal, \(x\) must either believe it is true, believe it can never become true or believe some other condition holds that makes abandoning the goal worthwhile.

Procedural Reasoning System (PRS): A BDI Architecture

From SRI: http://www.ai.sri.com/~prs/prs-arch.html
PRS Application: Space Shuttle Reaction Control

Agent Programming Languages

- Agent Oriented Programming, specialization of OOP
  - Fixes form of agent’s state
  - Fixes form of messages (e.g., KQML)
  - Constrains methods (e.g., cannot commit to incompatible actions)

Agent0 (Shoham)

Agent mental state has three components:
- Beliefs
- Commitments
- Capabilities (what actions can be committed, action descriptions)

Speech act messages
- Inform about belief
- Request commitment
- Unrequest previous commitment
- Agent trusts everything it is told.

Agent0 (cont.)

Actions:
- DO(time, privateaction) execute one of the agent’s capabilities
- REFRAIN(action) do not commit to action
- IF mentalcond THEN action
- Speech act messages

Commitment Rules
- COMMIT(messagepattern, mentalcond, agent, action)
  if the agent receives a message of messagepattern that comes from or mentions agent AND mentalcond is true AND the receiving agent is capable of action AND is not committed to REFRAINing OR if action is REFRAIN then the agent is not already committed.
Software Agents

Operate in virtual worlds, e.g., information systems, WWW, networks…

Examples:
- Search engines (Google)
- Comparison shopping agents (MySimon)
- Recommenders (Amazon)
- “Bob” the Microsoft helper

Mobile Agents

- Roam wide area networks to perform tasks on behalf of owners
- Example: Sony’s Magic Link PDA which assists in managing a user’s email, fax, phone and pager
- Key Technology: TeleScript is an OO programming language for developing distributed applications.
- Sherpa & Google Now look at email, calendar and location to present useful information

Information/Internet Agents

Web search
- Google, citeulike, mendeley
Opinions/reviews Monitoring twitter feeds
Travel: farecast
IBM’s Watson Watson game
Reactive Software Agents

- Few examples of such systems:
  - A-life ant societies and eco-systems
  - Simulated robots

Robotic Agents

- Operate in *physical* environments
  - Vacuuming (IRobot’s Roomba)
  - Delivery in hospitals (Matsushita’s HOSPI)
  - Unmanned aerial vehicle (Predator)
  - Mars rover
  - Search and rescue

Robotics – Why So Hard?

<table>
<thead>
<tr>
<th>Robotics</th>
<th>Chess</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inaccessible</td>
<td>Accessible</td>
</tr>
<tr>
<td>Nondeterministic</td>
<td>Deterministic</td>
</tr>
<tr>
<td>Non episodic</td>
<td>Episodic</td>
</tr>
<tr>
<td>Dynamic</td>
<td>Static</td>
</tr>
<tr>
<td>Continuous</td>
<td>Discrete</td>
</tr>
</tbody>
</table>

What Are Robots Good For?

- Manufacturing and materials handling
- Gofer robots (Fetch Robby, Fetch!)
- Hazardous Environments
- Telepresence and Teleoperation
**What Are Robots Made of?**

- **Structure**: rigid parts of metal or plastic
- **Joints**: pivot, ball joint, bearing & axle,…
- **Actuation**: motors, pistons,…
- **Extending actuation**: wheels, cables, hydraulics,…
- **Power**: batteries, fuel tanks,…
- **Self sensing**: encoders, accelerometers, gyros, strain gauges, volt meters, thermometers, inclinometers,…
- **External sensing**: command links, sonar, microphones, photo-receptors, cameras,…

**Effectors – Locomotion**

- **Walking vs. Rolling**
  - Wheels are always stable, but stairs are difficult!
  - Four legs needed to guarantee stability, which is easy to control but inefficient.
  - Dynamic stability requires few legs, but complicated control.
- **Control**
  - Holonomic: controllable degrees of freedom match robot’s DoF
  - Nonholonomic: controllable DoF less than robot’s DoF

**Effectors – Manipulation**

- Commonly simplified arms/hands
- **Kinematics**: methodology for computing inputs needed to reach physical states
  - Inputs $I$: joint angles, slider positions…
  - Outputs $O$: desired hand position…

$$M : O \rightarrow I$$

**Sensing**

- **Proprioception**: use encoders to measure joint angles, etc.
- **Force & Tactile sensing**
  - Critical in accomplishing compliant motions.
  - Material identification, change detection.
- **Sonar**
  - Like finding your way in a house of mirrors with a flashlight
- **Cameras**
  - CCD, infrared, ladar,…
Paradigms

- **Hierarchical**: strict cycle of sense, plan, act. Global world model.
- **Reactive**: sense-act mapping. No planning or global world model.
- **Hybrid Deliberative/Reactive**: Planning decomposes task into subtasks which can be reactive.

Architectures - Hierarchical

Horizontal task decomposition

SRI: from Shakey, the first AI mobile robot to now

Simple model of the world: what action to take in what state (FSM, rules, policy)
Characteristics of Reactive Architectures

- **Situated agency**: goals arise from robot's ecological niche
- **Emergent Behaviors**: overall performance arises from integration of behaviors
- **No Representation**: only behavior specific sensing, only representation as needed to process sensory data

Brooks' Behavior Based Robotics

- Behavior is a direct mapping from sensory input to a pattern of motor actions designed to achieve some task.
- Organized hierarchically into layers of competence.
- Robust, coherent and real-time behavior arises from hard coded ranking of competence layers.

Subsumption Architecture

- Each layer
  - includes a set of behaviors operating asynchronously.
  - Is a network of Augmented FSMs (with timers)
  - Derives input from other layers of sensors.
  - May be suppressed (overriding input) or inhibited (overriding output) by high levels.

Hybrid Deliberative/Reactive

- Planning (e.g., path planning, performance monitoring) operates at a higher level than the sense-act cycle of reactivity.
- Planning turns on/off particular behaviors to produce a sequence.
Components of Hybrids

- **Sequencer**: generates sequence of behaviors, including necessary preconditions
- **Resource Manager**: allocates resources to behaviors
- **Cartographer**: creates and maintains world model
- **Mission planner**: translates directives into a robot plan
- **Performance monitor**: tracks progress

Example: Task Control Architecture

- Task Scheduling (Prodigy)
- Path Planning
- Navigation (POMDP)
- Obstacle Avoidance

Sensors  →  Effectors

“Xavier”
Reid Simmons
At CMU

http://www.ri.cmu.edu/projects/project_91.html

Multi-Agent Systems

- Several interacting intelligent agents coordinate or cooperate in their efforts to achieve a common set of goals.
- **Distinguishing features:**
  - Need for inter-agent communication: coordinate and negotiate
  - Synthetic characters/personality
  - Orchestrating cooperation

Multi-Agent System Design

- **Agents**: support autonomous, independent action
- **Society**: support interaction between agents, even when agents cannot be assumed to all share the same goals to carry out the tasks we delegate to them
Multi-Agent Systems: Robotic

Swarms: homogeneous (identical) robots working together on single task
Heterogeneous: soccer, search and rescue with different views
Control: distributed vs. centralized
Cooperation: active vs. non-active

Multi-Agent Systems: Software

Requirements
- Inter-agent communication protocol
- Set of known services/capabilities
- Knowledge of resources
- Policy for carrying out tasks
Platform Services
- Registration of agents’ services
- Matchmaker or yellow pages
- Dictionary or ontology of language
- Security maintenance (trust and clearance)

Agent Communication Languages

Requirements:
- Language with standardized syntax and semantics
- Content language for domain of discourse
- Shared ontology
Problem Examples:
- What was the price of FIEUX yesterday at close?
- Find out how to install a print driver for HP Deskjet 1600CM.
Solutions:
- Knowledge Query and Manipulation Language (KQML)
- Foundation for Intelligent Agent’s (FIPA) ACL

KQML: Knowledge Query and Manipulation Language

Standardized syntactic wrappers
Example: “ask” performative
Example Dialogue: A

A to B: (ask-if (> (size chip1) (size chip2)))
B to A: (reply true)
B to A: (tell (= (size chip1) 20))
B to A: (tell (= (size chip2) 18))

Example dialogue (B)

(stream-about
 :sender A
 :receiver B
 :language KIF
 :ontology motors
 :reply-with q1
 :content m1)

(tell
 :sender B
 :receiver A
 :in-reply-to q1
 :content (= (torque m1) (scalar 12 kgf)))

KQML Layers

Content layer: application’s representation language
Communication layer: bookkeeping information between agents
Message Layer: supplied message type, language and ontology information

Ontology

Central issue in KR: how to organize knowledge to impart meaning
- How to divide up domain
- How to determine important concepts
- What properties to include
- How to support inference
### Task Coordination/Allocation

- **Contract Net protocol**
  1. Recognition
  2. Announcement
  3. Bidding
  4. Awarding
  5. Expediting

---

### Where are we going?

**Turning Test Extra Credit:**

Convince the examiner that he's a computer.

You know you make some really good points.

I'm... not even sure who I am anymore.