DIS, La Sapienza - PhD Course

Autonomous Agents and Multiagent Systems

Lecture 1: Introduction

Yves Lespérance

Dept. of Computer Science & Engineering
York University

Motivation

- Distributed computing, WWW, ubiquity
- Need interoperability
- Open systems (available entities change)
- Need for adaptability, robustness
- Human orientation, task delegation
- Work with huge amount of mostly unstructured information

Agent-Oriented Computing

- View a distributed computing system as a society of agents
- Agents are autonomous, i.e. can make their own decisions about how to act

Key Agent Technologies

- Yellow pages, matchmakers, brokers
- Agent communication languages
- Coordination/negotiation protocols
- Ontologies, semantic markup languages
- Communication infrastructure/middleware
- Agent programming languages, architectures, CASE tools

What is an Agent?

- "An *agent* is a computer system that is *situated* in some *environment* and is capable of *autonomous action* in this environment in order to meet its design objectives" [Wooldridge 02]
- Related to controllers, reactive systems
- Many different notions of agents

Attributes of Intelligent Agents

- Autonomous, i.e. can make their own decisions about how to act
- Reactive, i.e. perceive environment & respond in timely way to changes
- Proactive, i.e. take the initiative & act to achieve their goals
- Social abilities, i.e. can interract, collaborate, negotiate with others

What Could Agents Do?

- Personal digital assistant that can arrange a trip for user by negotiating with service providers
- Air-traffic control system that can cope with emergencies, e.g. craft failures, bad weather
- Autonomous control of a space probe or planetary rover

Applications of Agents

- Industry: Air-traffic control, electricity distribution management
- E-commerce: shopping agents, supply chain integration
- Personal assistants: meeting scheduling, movie/book selection
- Information management: mail/news filtering, information retrieval
- Intelligent interfaces & groupware
- Robotics: Deep Space I, museum guides, soccer
- Believable agents for entertainment & games
- See [Wooldridge 02, Chapter 11] for more info

Agents as Intentional Systems

- Taking the intentional stance [Dennett 87] is attributing beliefs, desires, & rationality to an entity to explain or predict its behavior
- Contrast to physical or design stance
- Useful abstraction with complex systems
- May take this further and assume agents are designed to manipulate explicit representations of their beliefs, desires, etc.

Types of Work on **Individual Agents**

- Agent theories: formal/logical specifications of agents, their mental states, the relation between them and agent's action
- Agent architectures: models for designing systems that behave like agents, integrating planning, decision making, belief update, etc.
- Agent Programming Languages: languages for programming agents

Types of Work on **Multiagent Systems**

- Agent Communication Languages: speech acts languages, content languages, ontologies
- Protocol Design & Negotiation: mechanisms for reaching agreements between competitive agents, often through auctions; based on game and economic theory; analysis of protocol properties

Types of Work on **Multiagent Systems**

- Task Allocation & Coordination: mechanisms for allocating tasks & sharing results, multiagent planning, ensuring that team members stay coordinated
- Multiagent Platforms/Infrastructures: languages and tools for programming MAS, providing basic services such as networking, yellow pages, etc.; e.g. SRI's Open Agent Architecture (OAA), Telecom Italia's Java Agent Development Environment (JADE), which is FIPA compliant

Agent Architecture: IRMA

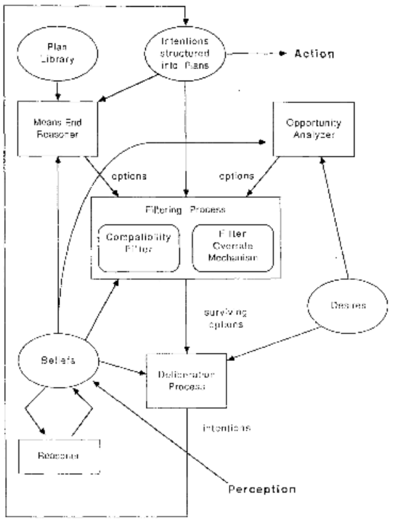


Fig. 1. An architecture for resource-bounded agents.

- •IRMA [Bratman et al 88] architecture for resource-bounded practical reasoning agents
- •Agent has beliefs, desires, & intentions
- •Intentions are desires that agent is committed to
- •Means/ends reasoner generates possible intentions/plans/options
- •Deliberation process decides what options to commit to
- •Current intentions **act as a** filter over new intentions, must be compatible
- •Filter can be overridden when an opportunity or problem is detected

Agent Theory/Logic: [Cohen & Levesque 90]

Start with a linear temporal/dynamic logic:

- (Happens α)
- (Done α)
- [] ϕ - Always ϕ
- ◊ ϕ - Eventually ϕ
- An action exp α can be simple, a test (ϕ?), or a sequence (α1; α2)

C&L Logic: Mental State Operators

- (Bel i ϕ) agent i believes ϕ (KD45 logic)
- (Goal i ϕ) agent i has goal that ϕ (KD)
- (P-Goal i ϕ) agent i has **persistent** goal that ϕ
- (Int i α) agent i has intention to perform α

C&L Logic: Properties

- $\models$ (Bel i ϕ) $\rightarrow$ (Goal i ϕ) - realism
- Assumption: eventually, all goals get dropped (no indefinite procrastination)
- Intention is dropped only if agent believes it has been achieved or is unachievable
- Success theorem: if one intends α and one never comes to believe that α is impossible, then eventually α happens

Agent Programming Language: IndiGolog

- High-level programming language for agents [De Giacomo & Levesque 99, Lesperance & Ng 00]
- Based on situation calculus, logic for reasoning about dynamic worlds
- Supports planning and plan execution in dynamic and incompletely known environments
- Supports complex behavior specifications
- Supports ordinary, sensing, exogenous actions
- Implemented on top of Prolog

IndiGolog Agent Structure (1)

- Declarative Part – Application domain dynamics specification in situation calculus
- Includes axioms describing:
  - Initial situation
  - Preconditions of primitive actions, i.e. when they are possible
  - How the world changes when a primitive action is performed (including sensing)
  - Etc.

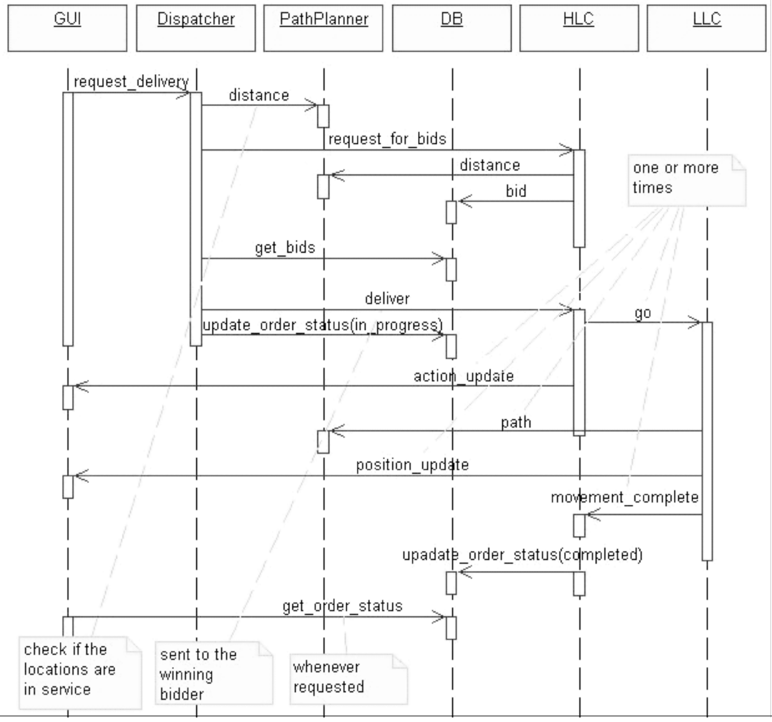
IndiGolog Agent Structure (2)

- Procedural Part – Rich set of constructs for agent behaviour specification
  - Recursive Procedures
  - If-then-else
  - While loops
  - Non-deterministic branch / choice of arguments / iteration
  - Concurrency with or without priorities
  - Interrupts
  - Search (planning) block

E.g. Multirobot Mail Delivery in IndiGolog + OAA [Lapouchnian & Lesperance 02]

- Varying number of robots
- Dispatcher agent assigns incoming orders to mail robots
- Dispatcher, robots implement a variation of contract net protocol
- Robots – two agent architectures
  - High-Level Control (HLC) in IndiGolog – bidding, optimal route planning
  - Low-Level Control (LLC) – motion subsystem
- Also: GUI, PathPlanner, DB

Interactions

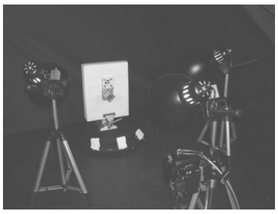


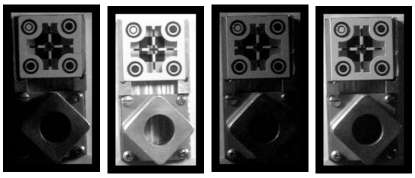
HLC – Behaviour Specification

```
proc(control, [
   prioritized_interrupts([
        %high priority interrupt: handles bid requests
        interrupt([f,t,o],
                bid_requested(f,t,o)=true,
                pi([l,d], [?(l=next_location),
                ?(d=dist(l,f)), bid(o,d)])),
        %medium priority interrupt: handles newly assigned orders
        interrupt([f,t,o], and(canmove,
                delivery(f,t,o)=ordered),
                search(pconc(minimize_distance(0), envSimulator))),
        %low priority interrupt: when nothing to do, wait
        interrupt(true, no_op)
   ]) ]).
```

E.g. Lights and Camera Project [Borzenko et al 06]

- Intelligent control of image acquisition, lights and camera settings
- Applications in space, mining, surgery





Lights and Camera Architecture