Motivation

- Distributed computing, WWW
- Need interoperability
- Open systems
- Need for adaptability, robustness
- Work with huge amount of mostly unstructured information
Agent-Oriented Computing

- View a distributed computing system as a society of agents
- Agents are autonomous

Key Agent Technologies

- Yellow pages, matchmakers, brokers
- Agent communication languages
- Coordination protocols
- Ontologies, semantic markup languages
- Communication infrastructure
- Agent programming languages & architectures
Attributes of Agents

- Autonomous
- Reactive
- Proactive
- Have social abilities

Typical Applications

- 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
Need for Intelligence in Agents

- Hard to predict all tasks and behaviors in advance
- To get adaptability, need to use AI techniques
- Agents must be able to make new plans to achieve their goals, cope with failures, reason about other agents

E.g. IndiGolog

- High-level programming language for robots and intelligent agents (U of T, York, Rome, etc.)
- Based on situation calculus, logic for reasoning about dynamic worlds
- Supports online/offline 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
  - Action precondition axioms
  - Successor state axioms
  - Sensed fluent axioms
  - Unique names axioms for actions
  - Foundational, domain independent axioms

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 block
E.g. Multirobot Mail Delivery

- 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
HLC – Behaviour Specification

proc(control, [  
  prioritized_interrupts([  
      % high priority interrupt: handles bid requests  
      interrupt([[f,t,o]],  
        bid_requested(f,t,o)=true,  
        pl([[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(o), envSimulator))]),  
      % low priority interrupt: when nothing to do, wait  
      interrupt(true, no_op)  
    ]) ]).

E.g. Lights and Camera Project

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

Intelligent Controller

next light settings, vision parameters

evaluation metrics, e.g. model matching error

Vision Server

Edge Detection \rightarrow Edge Linking \rightarrow Pose Estimation

lights and camera parameters

Image Server

Acquisition Simulation

parameters

images

parameters

images

My Group’s Current Research

- Agent-programming languages & tools
- Planning in dynamic incompletely known domains
- Cognitive vision/robotics
- Semantic web, web services
- AO software engineering & formal methods
References